Technology Diffusion: the Case of Agent Technologies

Jez McKean[*], Hayden Shorter[†], Michael Luck[‡], Peter McBurney[§] and Steven Willmott[¶]

August 7, 2007

Abstract

Despite several examples of deployed agent systems, there remain barriers to the large-scale adoption of agent technologies. In order to understand these barriers, this paper considers aspects of marketing theory which deal with diffusion of innovations and their relevance to the agents domain and the current state of diffusion of agent technologies. In particular, the paper examines the role of standards in the adoption of new technologies, describes the agent standards landscape, and compares the development and diffusion of agent technologies with that of object-oriented programming. The paper also reports on a simulation model developed in order to consider different trajectories for the adoption of agent technologies, with trajectories based on various assumptions regarding industry structure and the existence of competing technology standards. We present details of the simulation model and its assumptions, along with the results of the simulation exercises.

1 Introduction

Agent technologies can be distinguished from other software technologies on the basis of their differing objectives. For agent technologies, the objectives are to create systems situated in dynamic and open environments, able to adapt to these environments and capable of incorporating autonomous and self-interested components. How quickly agent technology is adopted by software developers, therefore, will depend at least partly on how many application domains require systems with these characteristics. At present, these domains include logistics, transportation, utility management and defense [34]. Common to many of these domains are multiple stakeholders or organizations linked in a network, such as a supply-chain, and with mission-critical, real-time processing

[*]jazzle, 84 Ash Grove, Wavertee, Liverpool L15 1ET, UK. Email: jez.mckean@jazzle.co.uk

[†]Aepona Ltd, 20–24 York Street, Belfast BT15 1AQ, UK. Email: hayden.shorter@aepona.com

[‡]Department of Computer Science, King's College London, Strand, London WC2R 2LS, UK. Email: michael.luck@kcl.ac.uk

[§]Department of Computer Science, University of Liverpool, Liverpool, L69 3BX, UK. Email: p.j.mcburney@csc.liv.ac.uk

[¶]Dept Llenguatges i Sistemes Informatics, Universitat Politecnica de Catalunya, Campus Nord, Jordi-Girona 1-3, E-08034 Barcelona, Spain. Email: steve@lsi.upc.edu

requirements. In other words, there are both functional and technical requirements for these applications, a divide that agent technologies are able to bridge.[1]

New software technologies require supporting tools and methodologies before they achieve widespread adoption. A fundamental obstacle to the take-up of agent technology is the current lack of mature software development methodologies for agent-based systems. Clearly, basic principles of software and knowledge engineering need to be applied to the development and deployment of multi-agent systems, as with any software. This applies equally to issues of scalability, security, transaction management, etc, for which there are already available solutions. A key challenge with agent-based computing is to augment these existing solutions to suit the differing demands of the new paradigm, while taking as much as possible from proven methods. For example, agent software development needs to draw on insights gained from the design of economic systems, social systems, and complex engineering control systems. In addition, existing middleware solutions need to be leveraged as much as possible, and this message has been understood: several companies have been working on platforms based on existing and standard middleware that is known and understood in the commercial domain [20].

Already there are clear and visible examples of agent solutions that are providing real business benefit, and a selection of detailed application case studies of agent technologies is presented in [26]. For example, Calico Jack Ltd[2] has been working with the Chief Scientist Office, part of the Scottish Executive Health Department, to develop prototype solutions tackling several key issues in primary health care. The company has delivered an agent-based system that integrates with existing email services and other processes used in doctors' practices, adding new functionality. In particular, and in collaboration with mobile telecommunications company, Orange, new services are being offered to patients by SMS and WAP. By modeling the stakeholders in the primary care system as agents, the system has been easily introduced into an already complex mix of IT processes, interpersonal processes, regulatory processes and the relationships between them. In working with patients, physicians and administrators to tailor the service to their needs, agent-based representation has been key in supporting flexibility in design, implementation and deployment. Among the new services currently offered by the system are the ability to coordinate repeat prescriptions using SMS (reducing load on the practice administrator, and simplifying the process for the patient), and to book appointments and handle reminders through a combination of SMS and email (with the aim of reducing the expensive wasteful missed appointments and smoothing the booking process for patients). The system was trialed in a medical practice in Tayside, UK, with a view to subsequent wider rollout.

Despite examples of deployed agent systems, barriers remain to the large-scale adoption of agent technologies. In order to understand these barriers, it is useful to consider the branch of marketing theory which deals with diffusion of innovations. The next section presents a summary of this theory, as a framework for considering the current state of diffusion of agent technologies. Section 3 considers the role of standards in the adoption of new technologies and describes the agent standards landscape. In Section 4, we explore other aspects of the agent technologies landscape, in particular comparing the development and diffusion of agent technologies with that of object-oriented programming. Our diffusion model is then described in detail in Sec-

[1] Of course, agent software technologies also support important non-functional properties even in systems or domains which do not require the full power of the agent paradigm — for example, the loose coupling of components; interactions as first-class entities; pervasive metadata; etc.

[2] `www.calicojack.co.uk`

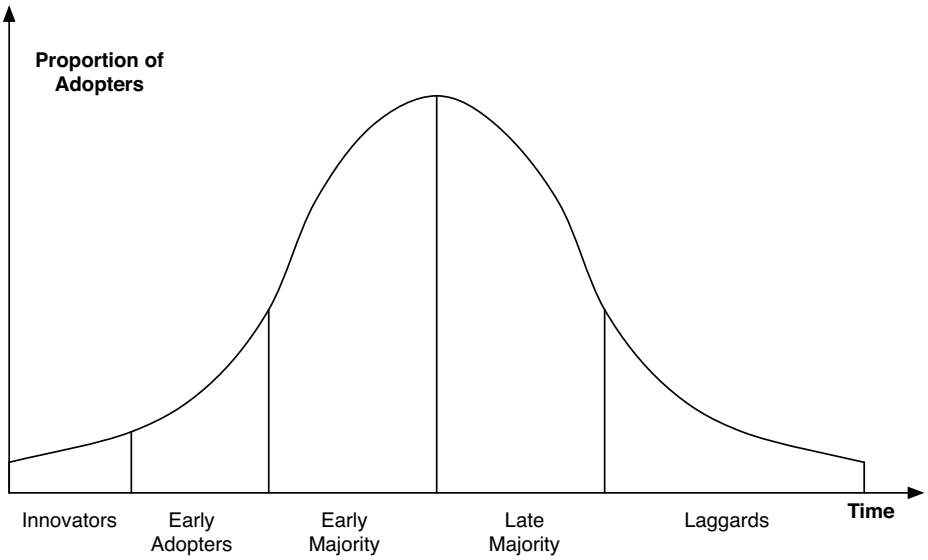

Figure 1: Standard Product Lifecycle

tion 5, along with the assumptions used and results of the simulations undertaken using it. Section 6 concludes the paper.

2 Diffusion of Innovations

In order to understand the current commercial position of agent technologies it is useful to know something about the diffusion of new technologies and innovations. This is a subject long-studied by marketing theorists, such as [29, 23], drawing on mathematical models from epidemiology and hydrodynamics. We begin by considering the concept of the *product life cycle*, illustrated in Figure 1.

Most marketers believe that all products and services are subject to life cycles: sales of a new product or service begin with a small number of customers, grow to a peak at some time, and then decline again, perhaps eventually to zero [17]. Growth occurs because increasing numbers of customers learn about the product and perceive that it may satisfy their needs (which may be diverse). Decline eventually occurs because the market reaches saturation, as potential customers have either decided to adopt the product or have found other means to satisfy their needs, or because the needs of potential customers have changed with time. Most high-technology products are adopted initially only by people or companies with a pressing need, a keen interest in that type of new technology, and the disposable income to indulge their interest. Thus, early adopters are often technologically-sophisticated, well-informed, wealthy, and not averse to any risks potentially associated with use of a new product.

The reasons why such a product life cycle exists or, more specifically, why all the companies or people who will eventually adopt the technology, product or process do not do so immediately at the time the product is launched, are diverse. They are considered below.

- Potential adopters must learn about the new technology before they can consider adopting it. Thus, there needs to be an information diffusion process ahead of the technology diffusion process. In the case of software technologies, potential adopters typically learn about new technologies and applications through industry events, trade publications (including on-line publications), and word-of-mouth communications from business partners. In the 1990s, agent technologies were still relatively unknown to the broader IT community, and it has taken quite some time and effort to change that position. Indeed, a key objective of the three EC-funded AgentLink[3] projects was to inform commercial software developers and their managers about agent technologies, and this was achieved through regular email alerts, a newsletter, and an annual industry conference, the *Agent Technology Conferences*.

- In addition, for non-digital products and services, the supplier needs to physically distribute the product or service to customers. Establishing and filling sales channels may take considerable time and effort, and thus delay uptake of the product or service. For agent technologies and supporting software and tools, because these are software, physical distribution is not a factor inhibiting adoption.

- Even after they learn about a new technology, not all eventual adopters will have the same extent of need for the product. Early adopters are likely to be those customers with the most pressing needs, needs which are not currently satisfied by competing or alternative technologies. The early adopters of supercomputers, for instance, were organizations with massively large-scale processing requirements, such as research physicists, meteorologists, and national census bureaux; later users included companies with smaller, but still large-scale, processing requirements, such as econometric forecasting firms and automotive engineering design studios. With agent technologies, early adopters have included companies in the transport and logistics sectors, defense and aerospace, and utility and supply chain networks [26]. However, the needs being satisfied in these different industry sectors vary. In the case of transport and logistics, there is great need for optimization and dynamic re-scheduling, which multi-agent simulations enable. In defense and aerospace applications, needs include realistic simulations for training and strategy purposes, and effective management of large-scale complex adaptive systems. In utility and supply chain networks, agent systems allow explicit representation of the many diverse entities inter-connected in the network, and thus may support middleware that interacts effectively with different legacy systems, or manages competing privacy requirements, for example.

  Note that every mention we make here of *needs* in this paper is always made from the perspective of the potential adopter, and includes not only technical requirements. Thus, an external observer may conclude that a particular company has no objective need for agent technology, but if the company itself perceives otherwise then it does indeed have a need. A company whose technical programming problems could be solved with object-oriented technologies but which decides to adopt agent-based methods on the basis of a desire to adopt the latest technology has indeed a need for agent technologies. Advanced technologies, like all other goods and services, may be adopted for reasons of fashion, mimicry or imitation, and may, in being so adopted, still satisfy needs. As discuss in greater

[3] www.agentlink.org

detail in [20, Chapter 6], technologies may be subject to unwarranted claims and other forms of *hype*, and market adoption may even follow a pattern of unduly-optimistic and then unduly-pessimistic expectations, the so-called *hype cycle*.

- Of those potential adopters with a need, not all will have the financial resources necessary to adopt the new technology. Most new technologies, products and processes are expensive (relative to alternatives) when first launched. But prices typically fall as the base of installed customers grows, and as new suppliers enter the marketplace, attracted by the growing customer base. Thus, later adopters typically pay less than do early adopters for any new technology. Likewise, the total costs of adoption also typically fall, as complementary tools and products are developed in tandem with a new technology. For example, the rise of Web Services has, for many, provided such tools and products through the development of an infrastructure that supports the deployment of agent systems. In addition, it has also created (or perhaps more precisely, brought to visibility) new problems that agent solutions can address. If a company's needs are not pressing, the company may benefit by waiting for the price and other adoption costs to fall before adopting.

- Similarly, not all potential adopters share the same attitudes to technological risk. The risks associated with adopting a new technology also typically fall, as bugs are eliminated, user-friendly features are added, and complementary tools and products are developed. Each subsequent release of an operating system, such as Windows or Linux, for example, has entailed lower risks to users of unexpected losses of data, obscure hardware incompatibilities, exception conditions, etc. In the case of agent technologies, telecommunications companies have been enthusiasts for research and development of agent technologies (and supporting tools, such as Telecom Italia's sponsorship of JADE[4] or British Telecommunications' support for DIET[5] and ZEUS[6]), but these companies have only rarely deployed agent systems onto their main networks because of the risks involved.

- Finally, for many advanced technologies and products the value to any one adopter depends on how many other adopters there are. These so-called *network goods* require a critical mass of users to be in place for the benefits of the technology to be fully realizable to any one user. For example, a fax machine is not very useful if only one company purchases one; it will only become useful to that company as and when other companies in its business network also have them. Thus, when considering open, dynamic systems as a driver for agent technologies, the benefits can only be realized when there exists an ecosystem of relevant services. For example, Grid computing aims to support virtual organizations, but real applications in this space require the infrastructure and the services for it to provide a valuable proposition for many commercial organizations.

These reasons for the existence of product life cycles mean that companies or people who adopt a new technology or purchase a new product later in its lifecycle may do so for very different reasons than do the early adopters; later adopters may even have different needs being satisfied by the product or technology. For example, in

[4] `http://jade.cselt.it/`
[5] `http://diet-agents.sourceforge.net/`
[6] `http://sourceforge.net/projects/zeusagent`

most countries the first adopters of mobile communications services were mobile business and tradespeople, and wealthy individuals. These were the customer segments with both the greatest needs for communications on the move, and the spare wealth to purchase the products to satisfy these needs. Only as prices fell have residential consumers, non-mobile office workers, and teenagers become users, and their needs are very different from those earlier into the market. This changing profile of adopters creates particular challenges for marketers [25].

How quickly new products and technologies reach saturation is also a difficult question to answer. If one considers an innovation such as written communication, which began several thousand years ago, diffusion has been very slow. Perhaps as many as half the world's population have still to learn to read and write. In contrast, cellular mobile telephones are now used by 1.7 billion people, a position reached in just over two decades from the first public cellular networks [15].

3 Standards and Adoption

3.1 The Role of Standards

The fact that many technology products and processes are network goods means that the presence or otherwise of technology standards may greatly impact adoption. If a standard exists in a particular domain, a potential adopter knows that choosing it will enable access to a network of other users. The greater the extent of adoption of the standard, the larger this network of users will be. Thus, one factor inhibiting adoption of Linux as an operating system (OS) for PCs was the fact that, until recently, most users had adopted the *de facto* standard of Microsoft Windows; while the user of a stand-alone machine could use any operating system they desire, installing an uncommon OS would mean not having access to the professional services, software tools and applications which support or run on the operating system. If adopting a technology is viewed as akin to choosing a move in a multi-party strategic game, where the potential adopter wishes to select that technology option which will be also chosen by the majority of his or her peers, then the existence of a standard may weight the payoffs in favor of a particular option and against others [35].

The development of standards arises from various possible sources. Standards may be imposed upon a user community by national Governments or international organizations, as with the adoption of GSM by all European and many other nations, for second-generation mobile communications networks; the communications regulatory agencies of the United States, in contrast, decided not to impose a particular technology standard in this domain. Alternatively, standards may be strongly recommended to a user community by a voluntary standards organization, as in the case of many Internet standards; two machines connected to the Internet may use any interconnection protocols they themselves agree on; for example, not necessarily the standard protocols, such as TCP and UDP, defined by the Internet Engineering Task Force. Finally, standards may emerge from multiple independent choices of one particular technology over others made by many individual adopters; the common QWERTY typewriter layout is one such bottom-up standard [11].

3.2 Agent-related Standards

If standards are not imposed by some Government or regulatory agency, then scope exists for multiple voluntary organizations to recommend competing standards or for competing standards to emerge from user decisions. To some extent, this may be occurring in the agent technologies domain, with several organizations having developed or aiming to develop standards related to the interoperation and interaction of intelligent software entities: the Foundation for Intelligent Physical Agents (FIPA),[7] the Global Grid Forum,[8] the Object Management Group,[9] and the World Wide Web Consortium.[10]

The growth of the World Wide Web and the rapid rise of eCommerce have led to significant efforts to develop standardised software models and technologies to support and enable the engineering of systems involving distributed computation. These efforts are creating a rich and sophisticated context for the development of agent technologies. For example, so-called service-oriented architectures (SOAs) for distributed applications involve the creation of systems based on components, each of which provides pre-defined computational services, and which can then be aggregated dynamically at runtime to create new applications.

The development of standard technologies and infrastructure for distributed and eCommerce systems has impacted the development of agent systems in two major ways. Firstly, many of these technologies provide implementation methods and middleware, enabling the easy creation of infrastructures for agent-based systems, such as standardised methods for discovery and communication between heterogeneous services. Secondly, applications now enabled by these technologies are becoming increasingly agent-like, and address difficult technical challenges similar to those that have been the focus of multi-agent systems. These include issues such as trust, reputation, obligations, contract management, team formation, and management of large-scale open systems.

In terms of providing potential infrastructures for the development of agent systems, technologies of particular relevance include the following.

- Baseline Technologies:
  - The Extensible Markup Language (XML) is a language for defining markup languages and syntactic structures for data formats. Though lacking in machine-readable semantics, XML has been used to define higher-level knowledge representations that facilitate semantic annotation of structured documents on the Web.[11]
  - The Resource Description Format (RDF) is a representation formalism for describing and interchanging metadata.[12]
  - The Web Ontology Language (OWL) provides an XML, RDF and description-based language for the representation of Ontologies. The language provides a structured way to describe knowledge that might be reused in a wide range of web applications.[13]

[7] Now IEEE FIPA. See: `www.fipa.org`.
[8] `www.ggf.org`
[9] `www.omg.org`
[10] `www.w3.org`
[11] `www.w3.org/XML/`
[12] `www.w3.org/RDF/`
[13] Web Ontology Language Summary Page at W3C: `http://www.w3.org/2004/OWL/`

- eBusiness:
  - Electronic Business using eXtensible Markup Language (ebXML) aims to standardise XML business specifications by providing an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner.[14]
  - RosettaNet is a consortium of major technology companies working to create and implement industry-wide eBusiness process standards. RosettaNet standards offer a robust non-proprietary solution, encompassing data dictionaries, an implementation framework, and XML-based business message schemas and process specifications for eBusiness standardisation.[15]
- Universal Plug & Play:
  - Jini network technology provides simple mechanisms that enable devices to plug together to form an emergent community in which each device provides services that other devices in the community may use.[16]
  - Universal Plug and Play (UPnP) offers pervasive peer-to-peer network connectivity of intelligent appliances and wireless devices through a distributed, open networking architecture to enable seamless proximity networking in addition to control and data transfer among networked devices.[17]
- Web Services:
  - Universal Description, Discovery and Integration (UDDI) is an initiative aimed at creating a platform-independent, open framework for describing services and discovering businesses using the Internet. It is a cross-industry effort driven by platform and software providers, marketplace operators and eBusiness leaders.[18]
  - The Simple Object Access Protocol (SOAP) provides a simple and lightweight mechanism for exchanging structured and typed information between peers in a decentralized, distributed environment using XML.[19]
  - The Web Services Description Language (WSDL) provides an XML grammar for describing network services as collections of communication endpoints capable of exchanging messages, thus enabling the automation of the details involved in applications communication.[20] The Web Services Choreography Description Language (WS-CDL) allows the definition of abstract interfaces of web services, i.e., the business-level conversations or public processes supported by a web service.[21]
  - Emerging standards in the area of Semantic Web Services such as the OWL-based Web Service Ontology (OWL-S)[22] and the Web Services Mod-

[14] www.ebxml.org
[15] www.rosettanet.org
[16] www.jini.org
[17] www.upnp.org
[18] www.uddi.org
[19] www.w3.org/TR/soap/
[20] http://www.w3.org/TR/wsdl
[21] www.w3.org/TR/2004/WD-ws-cdl-10-20040427/
[22] http://www.daml.org/services/owl-s/

eling Ontology (WSMO)[23] aim to combine the powerful knowledge representations available using RDF and OWL with Web Services implementations in order to support advanced service description, discovery, matchmaking and communication.

3.3 Impacts of Standards

The emergence of such standards activities perhaps indicates the importance placed by industry participants in creating technology standards. However, the presence of multiple and possibly-competing standards is likely to confuse potential adopters of the technology, and one could ask who benefits from so many standards initiatives. One response to this question is to note that that no technology development company would wish the widespread adoption of a technology standard which favored a competitor's products over their own. Thus the view has been expressed that large software development companies may actually behave so as to *divide and conquer* the various competing standards bodies, by, for example, participating intensely in one standards organization at one time and then another organization at another time.

Faced with competing recommendations for standards, potential adopters can respond in different ways. One result may be decision paralysis, with a user or company deciding to postpone adoption of a new technology until the standards position is clearer. Thus, in this case, multiple competing standards may inhibit uptake of a new technology and hence inhibit market growth. On the other hand, each of the proponents of competing standards has an interest in promoting their particular solution, and so the presence of multiple standards may lead to faster and more effective dissemination of information about the new technology than would be the case if there was only one standard. In this view, therefore, competing standards may actually encourage uptake of a new technology and its market growth. Which of these countervailing pressures actually dominates in any one situation will depend on the other factors influencing the decision processes of a potential adopter; for example, the extent to which the proposed technology satisfies an unmet need, the criticality of the need, and the extent of network effects in the product concerned.

Related to the issue of standards and network effects in adoption decisions by potential users of new technologies is the issue of business ecologies. Most companies and organizations are enmeshed in a network of business relationships, with customers, suppliers, competitors, and other stakeholders. If a major downstream customer or an upstream supplier of mission-critical inputs insists on adoption of a particular technology or standard as a condition of business, then a company may adopt it much sooner than they would otherwise. Thus, for example, the US company General Electric, has insisted that most of its suppliers, including even law firms providing legal advice to GE, bid for its business through online auctions. Of course, such pressure received by a potential adopter along a supply chain or across a business network may also greatly reduce the risks and costs associated with a new technology. Thus, adoption decisions under such circumstances are not necessarily irrational. Recent research has considered the impact of networks of influence in business ecologies on software adoption decisions, e.g. [36].

[23] `http://www.wsmo.org/`

4 Diffusion of Agent Technologies

4.1 Current Deployment

It is interesting to consider the position of agent-based computer technologies within the framework of the marketing and standards discussions above. Adoption of agent technologies has not yet entered the mainstream of commercial organizations undertaking software development, unlike, say, object-oriented technologies. The majority of commercial organizations adopting agent technologies would, in our opinion, be classified as *innovators* or *early adopters*. We believe this because we know of only a small number of deployed commercial and industrial applications of agent technology [26], and because we believe considerable potential exists for other organizations to apply the technology.

To date, agent technologies have been deployed in only a small number of industrial sectors, and only for particular, focused, applications. These applications have included: automated trading in online marketplaces, such as for financial products and commodities [27]; simulation and training applications in defence domains [2, 13]; network management in utilities networks [8]; user interface and local interaction management in telecommunication networks; schedule planning and optimization in logistics and supply-chain management [7, 14]; control system management in industrial plants, such as steel works [16]; and simulation modeling to guide decision-makers in public policy domains, such as transport and medicine, e.g., [3]. For a recent review of agent technology applications, see [26]. We discuss open versus closed applications later in the paper.

4.2 Objects versus Agents

In assessing the current status of diffusion of agent technologies, it is instructive to consider the history of the development and diffusion of object-oriented software technologies. Initial research in what are now called object-oriented programming (OOP) approaches commenced in 1962, with the invention of Simula, the first object-oriented language, by Ole-Johan Dahl and Kristen Nygaard in Norway [4, 5]. The term *object-orientation* was only coined in 1970 by Alan Kay. Although the 1970s saw several further developments, including the invention of the language Smalltalk at Xerox PARC in 1973 and the introduction of frames by Marvin Minsky in 1975, developments really only gathered pace in the 1980s and 1990s. The year 1983 saw C++ formally established, 1985 the first textbook released [32], and 1986, the establishment of the OOPSLA and OODBS conferences. The *Journal of Object Oriented Programming* began publication in 1988.

These primarily research initiatives were followed by rapid developments of a more practical nature, including the formation of the Object Management Group in 1989, the development of the Java language in 1991 (although not publicly released until 1995), and the establishment of relevant standards that include CORBA (first specification in 1992, CORBA 2.0 in 1994), UML in 1994, and ANSI C++ in 1998. It was 2001 before Dahl and Nygaard received an ACM A. M. Turing Award for their work on Simula in 1962. Thus, between the first object-oriented language and the establishment of the ANSI standard for C++ was 32 years. This is an extended period over which the technologies and techniques involved came to maturity and to widespread adoption.

The history of the development of object-oriented programming may provide a guide to the future development and diffusion of agent technology. In order to com-

pare the diffusion of the two technologies, we need to be aware of the differences of the two technologies, and the wider computing environments in which they have appeared. Both agent and object technologies are essentially disruptive technologies that provide (among other benefits) more effective and flexible techniques for software design and development. However, there are several important differences in the business environments in which the two technologies find themselves.

Firstly, because the software ecosystem in which object technology found itself was less mature than that currently for agent technology, adoption of new technologies was easier then. Object technology began in an era in which computing as a discipline and as an industry was relatively immature, and limited in scope. Although potential for applications certainly existed, there were then fewer programmers, fewer organizations engaged in software development, and no long-term or widespread experience of software engineering techniques, methodologies, tools, standards or paradigms, as is the case now. Consequently, the changes required for the adoption of object technologies were far less substantial and challenging than it is now for agent technologies.

Secondly, because the installed base of software was smaller, the impact of changes could be greater. While there are still many problems to be tackled in computing, the degree of improvement, in terms of productivity or efficiency, to be realized from specific advances decreases as the general level of maturity in computing increases. Thus, while there was no step change arising through object orientation, the gradual improvement in the state of software is likely to be even less marked with agent technologies.

Thirdly, the computing environment is much more heterogeneous, distributed and diverse currently than at any time previously, and it continues to change further in these directions. The consequence of this is a plethora of standards, techniques, methodologies and, importantly, multiple vested interests, legacy systems and corporate activities that must be integrated, managed, overcome or otherwise addressed for broad acceptance of new paradigms. Investment in new technologies at this point of the IT adoption cycle presents a much more challenging problem than previously. For all these reasons, it is likely that no technology in the near future will have anything like the impact of object orientation.

In summary, the growth of software development as a commercial activity over the last four decades may mean a much larger potential target market for new technologies, such as agent-based computing, than was the case in the past. But this growth also means that the impact of new technologies may be less than previously. Moreover, large-scale adoption of a new technology for software engineering may require the development and growth of a *supporting ecosystem* — conferences, journals, specific languages, development tools, middleware, application case studies, etc. A key lesson from the experience of object technology diffusion is that this supporting ecosystem may take several decades to develop, and its absence may inhibit adoption of the technology. How one should best develop such a supporting ecosystem for new software technologies is a very interesting question, which we may pursue in future work.

4.3 Methodologies and tools

Many of today's challenges in software design stem from the distributed, multi-actor nature of new software systems and the resulting change in objectives implied for software engineering. The development of methodologies for the design and management of multi-agent systems seeks to address these problems by extending current software engineering techniques to explicitly address the autonomous nature of their components and the need for system adaptability and robustness. A wide range of agent-

oriented methodologies have so far been developed, often addressing different elements of the modeling problem or taking different inspirations as their basis, yet there is no clear means of combining them to reap the benefits of different approaches. Similarly, agent-oriented methodologies still need to be successfully integrated with prevailing methodologies from mainstream software engineering, while at the same time taking on board new developments in other challenge areas [19].

More specifically, creating tools, techniques and methodologies to support agent systems developers remains a very big obstacle to broader adoption. Compared to more mature technologies such as object-oriented programming, agent developers lack sophisticated software tools, techniques and methodologies to support the specification, development, verification and management of agent systems. At the level of particular technologies or subareas within the field, automating the specification, development and management of agent systems is also an issue, since agent systems and many of their features are still mostly hand-crafted. For example, the design of auction mechanisms awaits automation, as does the creation and management of agent coalitions and virtual organizations [28]. These challenges are probably several decades from achievement, and will draw on domain-specific expertise (for example, economics, social psychology and artificial intelligence).

In summary, as a result of the relative immaturity of research and development in agent technologies, the field lacks proven methodologies, tools, and complementary products and services, the availability of which would act to reduce the costs and risks associated with adoption.

4.4 Closed and Open Applications

The applications for which agent technologies are most suited are those involving interactions between autonomous intelligent entities with diverse, and potentially-conflicting, goals. Some applications of this sort may be implemented as closed systems inside a single company or organization. For example, many companies find themselves under strong competitive pressures to deliver their products and receive material inputs using just-in-time delivery processes. In one of SCA Packaging's corrugated box plants, customer orders often arrive simultaneously for a range of different boxes, each order with its own color scheme and specific printing, and often to be delivered at very short notice. Because of the complexity of factory processes and the difficulty of predicting customer behavior and machine failure, large inventories of finished goods must therefore be managed. SCA Packaging turned to Eurobios to provide an agent-based modelling solution in order to explore different strategies for reducing stock levels without compromising delivery times, as well as evaluating consequences of changes in the customer base [6]. The agent-based simulation developed by Eurobios allowed the company to reduce warehouse levels by over 35% while maintaining delivery commitments.

Similarly, Tankers International, which operates one of the largest global fleets of oil tankers, has applied agent technology to dynamically schedule the most profitable deployment of ships-to-cargo for its Very Large Crude Carrier fleet. An agent-based optimiser, *Ocean i-Scheduler*, was developed by Magenta Technology for use in real-time planning of cargo assignment to vessels in the fleet [14]. The system can dynamically adapt plans in response to unexpected changes, such as transportation cost fluctuations or changes to vessels, ports or cargo. Agent-based optimisation techniques not only provided improved responsiveness, but also reduced the human effort necessary to deal with the vast amounts of information required, thus reducing costly mistakes,

and preserving the knowledge developed in the process of scheduling.

These are examples of closed agent systems, since the stakeholders belong to one organization. Most potential applications of agent technologies, however, require the participation of entities from more than one organization; simulation applications can only contribute in particular well-defined applications. Automation of purchase decisions along a supply-chain, for instance, requires the participation of the companies active along that chain, so that implementing a successful agent-based application will require agreement and coordination from multiple companies. Unless most or all stakeholders agree to participate, the application will have little value to those who do. In other words, the application domains for which agent technologies are best suited typically exhibit strong network good effects, a factor which significantly complicates technology adoption decisions by the companies or organizations involved. We note, moreover, that agent technologies can also better support the development of non-functional features of software, such as human-machine interfaces and links to legacy systems.

For example, Netherlands-based Acklin BV was asked by a group of three insurance companies, from Belgium, the Netherlands and Germany, to help automate their international vehicle claims processing system [1]. At present, European rules require settlement of cross-border insurance claims for international motor accidents within three months of the accident. However, the back-office systems used by insurance companies are diverse, with data stored and used in different ways. Because of this and because of confidentiality concerns, information between insurance companies is usually transferred manually, with contacts between claim handlers only by phone, fax and email. Acklin developed a multi-agent system, the KIR system, with business rules and logic encoded into discrete agents representing the data sources of the different companies involved. This approach means the system can ensure confidentiality, with agent access to data sources mediated through other agents representing the data owners. Access to data sources is only granted to a requesting agent when the relevant permissions are present and for specified data items. Because some data sources are only accessible during business hours, agents can also be programmed to operate only within agreed time windows. Moreover, structuring the system as a collection of intelligent components in this way also enables greater system robustness, so that business processes can survive system shutdowns and failures. The deployment of the KIR system immediately reduced the human workload at one of the participating companies by three people, and reduced the total time of identification of client and claim from six months to two minutes! For reasons of security, the KIR system used email for inter-agent communication, and the two minutes maximum time is mainly comprised of delays in the email servers and mail communication involved.

This example demonstrates one reason why the agents community has expended so much effort on developing standards for agent communication and interaction, as undertaken by IEEE FIPA, so that agent systems may interoperate without the need for prior coordinated technology adoption decisions. However, as noted above, the agent technology standards landscape is currently one in which multiple organizations have developed or are developing standards for the interoperation and interaction of intelligent software entities. In these circumstances, adoption of agent technologies is not necessarily promoted by the presence of competing, and subtly different, standards.

5 Modeling Diffusion of Agent Technologies

As part of our efforts to understand this area, we developed a simple computer simulation model to study the diffusion of agent technologies under different assumptions regarding technology standards. Our model uses assumptions about adoption decision processes and the relationships between different companies, and has not been calibrated against any real market data. It was intended only to provide a means for exploration of relationships between relevant variables and indicative information about these relationships. We fully recognize that the results of a generic model such as this will be highly dependent on the structure and assumptions used to create the model. Moreover, the features of specific markets, such as that for agent technologies, may result in very different outcomes from those described here. Thus these results should not be considered as guidance for specific marketing strategies or industrial policies in the agent technology domain. Nevertheless, the modeling process has value because it provides insight into relationships, and acts to stimulate intuition, as described in the following subsections.

5.1 Model Design: Overview

The domain of application for the simulation model was intended to be all organizations engaged in software development (whether undertaken only internally or for external clients) with a potential need for agent technologies. Such organizations potentially adopting agent technologies were represented as individual nodes in a graph. Directed connections (edges) between nodes were used to represent the influence of one organization over another in a decision to adopt or not adopt agent technologies. Thus, for example, a company making large or frequent purchases may be able to influence technology decisions of its suppliers. Because different industries have different degrees of concentration and different networks of influence, our model incorporated several different network topologies which we believe to be representative of the diversity of real-world industrial and commercial networks. These are presented in Subsection 5.3 below.

In the simulation model, nodes were then modeled as independent and autonomous decision-makers, each node making decisions to progress (or not) through a technology adoption life-cycle. The five stages in this life-cycle were:

1. Agent technology not yet considered;
2. Agent technology under consideration;
3. Agent technology being trialed;
4. Agent technology partially adopted; and
5. Agent technology fully adopted.

Time in the model was assumed to be discrete and linear, with nodes making decisions between successive timepoints, based on the status of various causal factors at the most recent timepoint. Each timepoint may be considered as a generation in the adoption lifecycle.

At each stage in the life-cycle, a node may decide to proceed to the next stage, to remain at the current stage, or to return to the previous stage. The mechanism used by each node at each stage to make these decisions depends on a number of relevant

factors, which are presented in Subsection 5.2 below. For each node and for each decision, these factors were then combined through a factor-weighting mechanism, also described below; the outcome of this combination was a decision: to progress forward to the next state in the technology adoption lifecycle; to remain in the current state; or to revert to the earlier state.

The simulation model was designed so that the factor weights and the weighting mechanism could be user-defined. The particular parameter values and formulas used in the model were developed on what are believed to be reasonable assumptions regarding real-world decision processes in this domain, informed by the marketing literature cited earlier and by the experience gained in collecting the AgentLink III agent technology application case studies [26]. Subsections 5.2 and 5.3 present the model design and structure in detail.[24]

5.2 Model Design: Agent Decision Process

As mentioned above, each node in the model decides at each stage in the life cycle whether to proceed to the next stage, or remain at the current stage, or return to a previous stage. We now list the causal factors which influence this decision. These factors were drawn from a study of the marketing literature [18, 21, 33] and the economics literature relating to network goods and standards [35, 36].

- The current need of the organization for the technology, denoted by variable B. The value of this variable for each node was assigned randomly to the node at commencement of the simulation, from a uniform probability distribution on the domain [0,1]. A value of 0 indicates no need, while a value of 1 indicates full (or very pressing) need. As noted earlier, we make no judgment on the objective nature of the need for the technology.

$$B \sim U[0, 1].$$

- The costs of adoption, denoted C. These costs were assumed to fall as the number of nodes progressing through the adoption lifecycle increased, according to a monotonically-decreasing S-shaped curve. In formal terms, where x is the proportion of nodes which have adopted the technology at the current time, and for a, b definable parameters:

$$C = \frac{1}{1 + \frac{x}{e^{a-bx}}}$$

  In order to achieve a value of $C \in [0, 1]$, values of $a = 2$ and $b = 6$ were selected by trial and error; no meanings should be attached to these particular values. Nodes were assumed randomly to be able to afford the technology at the cost-level pertaining at each timepoint. This was implemented by comparing the value of C to the value of a variable drawn from the uniform distribution on [0,1]; precisely when C was less than this variable, the node was assumed to be able to afford the technology. Simple algebraic manipulation shows that the probability that a node can afford to adopt the technology when the cost is at level C is $1 - C$.

[24] Further information on the design and use of the model can be obtained from [22]; this report, a user guide and the model itself may be downloaded freely from `www.csc.liv.ac.uk/research/techreports/tr2005/tr05008abs.html`.

- The availability of complementary software tools, denoted D. These tools were assumed to be increasingly available as increasing numbers of nodes moved through the technology adoption cycle, according to a monotonically-increasing S-shaped curve. With greater availability of tools, nodes were increasingly likely to move forward along the technology adoption lifecycle. The variable x is again the proportion of nodes which have adopted the technology at the current time. Likewise, the parameters a and b are user-definable in the model as before; these were again assigned values of $a = 2$ and $b = 6$.

$$D = \frac{1}{1 + \frac{e^{a-bx}}{x}}$$

- The presence of one or more technology standards. The existence of a single standard was assumed to encourage technology adoption by nodes, while the presence of more than one standard encouraged adoption in some nodes and discouraged it in others. This is modeled with two variables, E and F. Variable E takes either the value 0 (denoting no standards) or 1 (denoting a single standard). In the case of two competing standards, Variable F is generated randomly from a uniform probability distribution on the domain [0,1], to denote the probability of non-adoption in the presence of competing standards.

$$E \in \{0, 1\}$$
$$F \sim U[0, 1].$$

- The success of a technology trial, denoted G. While not all trials are successful, an unsuccessful trial does not necessarily lead to non-adoption of the technology; an organization may have pressing needs for the technology which lead it to adopt the technology despite the failure of a trial. Success or failure was assigned randomly to those nodes undergoing a trial at each timepoint. This was modeled by a variable G, generated randomly from a uniform probability distribution on the domain [0,1].

$$G \sim U[0, 1].$$

- The extent of influence of other connected nodes over each node, denoted H. For example, large downstream customers may strongly influence upstream suppliers in their choice of technologies. It is through this factor that the network topology impacts upon the adoption decisions of individual nodes, and so demonstrates the network-good feature of the technology. The mechanisms encoding modeling of influence in the model are presented, along with the specific topologies used, in Subsection 5.3 below.

In the absence of any real-world data on the relative influence of different factors on adoption decisions, it was decided to assume each factor was represented in the decision function as a linear variable. This was achieved by calculating a value $A \in [-1, 1]$ for each node at each time point, comprising the weighted sum of the values of each factor:

$$A = w_1 B + w_2 C + w_3 D + w_4 E + w_5 F + w_6 G + w_7 H$$

	Status			
Decision Factor	Not Yet Considered	Under Consideration	Trial Underway	Partially Adopted
Need (B)	5	4	4	4
Cost (C)	-1	-3	-3	-5
Tool Availability (D)	1	4	4	3
One Standard (E)	1	4	4	3
Competing Standards (F)	-1	-4	-2	-3
Trial Outcome (G)	0	0	5	4
Network Influence (H)	5	5	5	5

Table 1: Relative Decision-Factor Weights for each Adoption State

for weights, $w_1, w_2, \ldots, w_7$. Because the factors relating to costs (variable C) and multiple, competing standards (F) are believed to inhibit adoption, the corresponding weights w_2 and w_5 were negative. The particular weights used differed according to the current state of adoption of the node at the time, to reflect the dynamic nature of the influence of these causal factors on decision-making stage-by-stage. The relative weights for each variable were assigned on the basis of reasonable assessments of the relative importance of each factor at each stage of adoption, informed by the qualitative understanding gained by several of the authors in undertaking the data collection for the AgentLink III agent technology application case studies [26]. The relative weights of each decision factor, expressed as integers, are shown in Table 1.[25] No weights are shown for the final state, *Fully Adopted*, because nodes were assumed to remain in that state once it was reached.

The weighted index variable A was then interpreted as a probability. If $A \in (0, 1]$, A was understood as the probability that the node would move to the next state in its adoption life-cycle. If $A = 0$, the node would remain in its current state. If $A \in [-1, 0)$, A was understood as the probability that the node would move back to the previous state in its adoption life-cycle. The decision process to achieve these transitions was undertaken by comparing the absolute value of variable A to a variable drawn from a uniform probability distribution on $[0, 1]$.

5.3 Network Topologies and Influence

The simulation model makes use of a network topology linking nodes with edges of variable strengths. If it was desired to model a particular industry or industry sector, the model would permit the user to create a domain-specific network structure, and to run simulations of adoption diffusion over this network. In contrast to such an approach, our simulation aimed to explore the impact of standards on technology adoption for a range of different industry structures. Accordingly, we created several different network topologies, which were collectively desired to represent a range of typical industry structures. In doing so, we did not consider truly random networks (as used, for example, in the simulations of [36] and [35]), because we do not believe that actual industry networks are truly random. The generic topologies selected were:

Unconnected: 50 nodes representing companies, each without influence on one another. This topology models an industry which is highly disaggregated, with

[25] Note that in calculating variable A, the weights shown here were normalized to ensure $A \in [-1, 1]$.

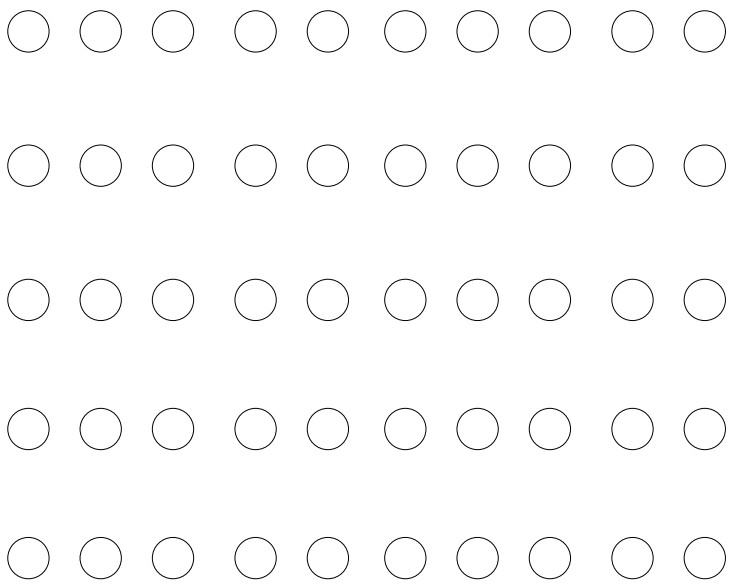

Figure 2: Unconnected Topology

independent technology decision-making by each company or organization in the industry. This is illustrated in Figure 2.

Dense connections: 50 nodes with many interconnections, indicative of many and diverse influences from one company to another. This topology models an industry which disaggregated, but where peer relationships are important in technology decisions, as illustrated in Figure 3.

Shallow supply chains: 5 major nodes (parents), each connected to and influencing 9 subsidiary nodes (children), in a cluster formation. This topology models an industry where supply chains are not deep, as shown in Figure 4.

Deep supply chains: 5 parent nodes, each connected to and influencing 9 subsidiary nodes linked together as in a linear supply chain. This topology models an industry where supply chains are deep, and upstream suppliers of companies themselves have their own distinct supply chains further upstream. This topology is shown in Figure 5.

Overlapping supply chains: 5 parent nodes, each connected to and influencing 9 subsidiary nodes linked together as in a linear supply chain, with at least one child node also influenced by a second parent. This topology models an industry where supply chains are deep, and suppliers companies sell to multiple companies downstream. This topology is shown in Figure 6.

In addition to allowing users to define their own topologies, the implemented model also permits users to define the strength of each edge, or set of edges, within a network, as a variable in the range $[0, 1]$. A larger value of this variable is understood as a stronger connection, allowing parent nodes to have greater influence over child nodes.

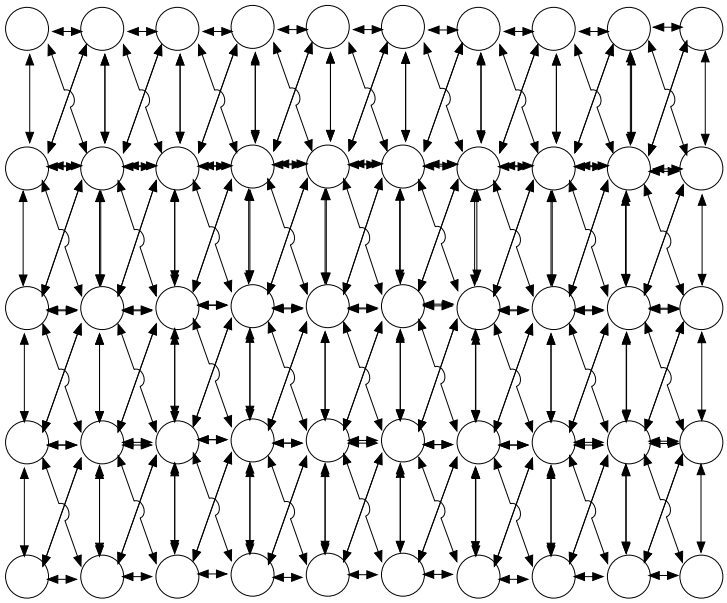

Figure 3: Densely Connected Topology

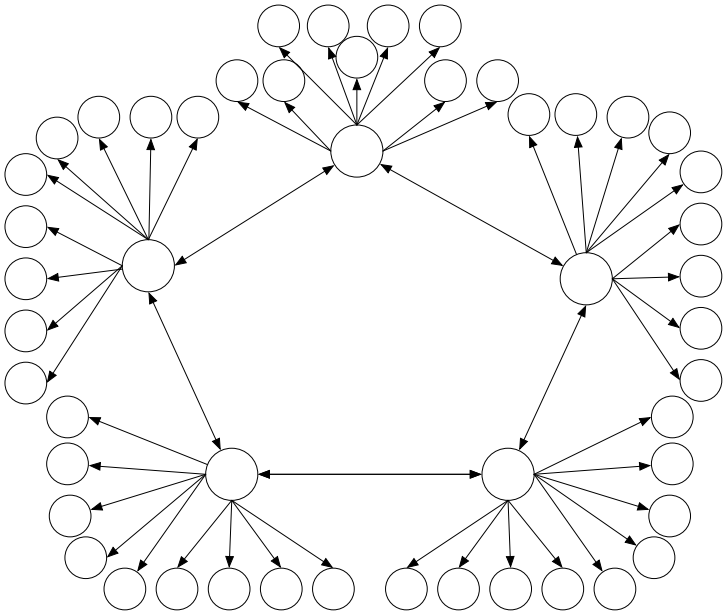

Figure 4: Shallow Supply Chains Topology

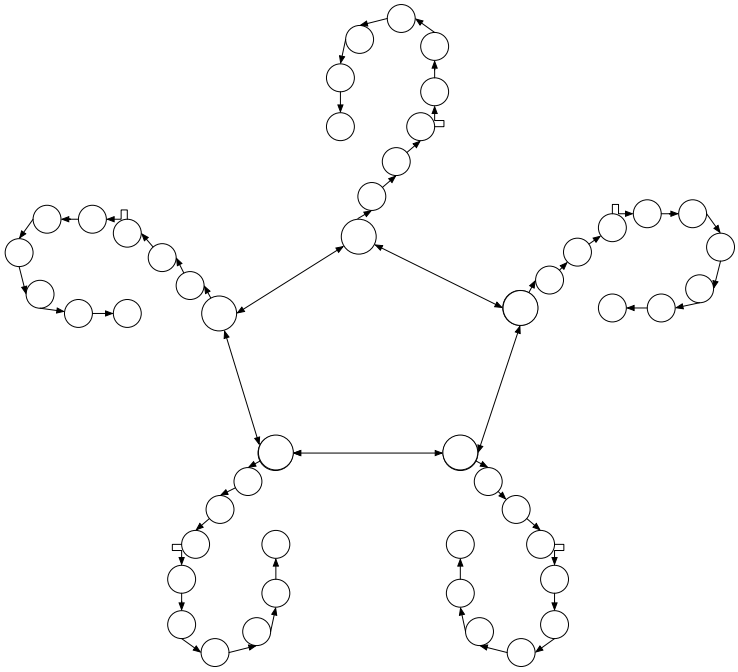

Figure 5: Deep Supply Chains Topology

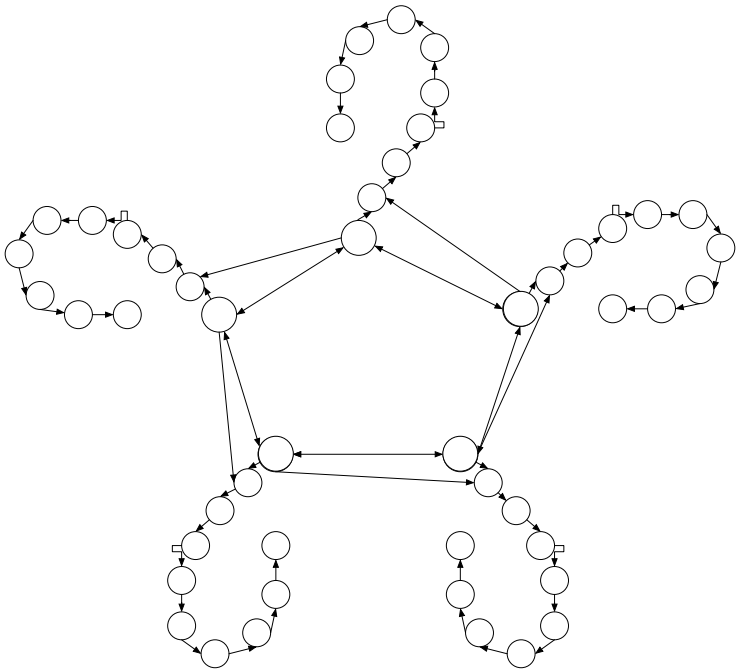

Figure 6: Overlapping Supply Chains Topology

Decision Algorithm	*Description*
Basic	Ignores parents, and returns $H = 0.5$.
Standard	Calculates H as the average state of all parent nodes, each weighted by its respective edge strengths.
Parental Pressure	Calculates H from the current state of that parent node connected by an edge with the greatest strength.
Viral	Calculates H as the current state of the parent which is furthest along in the adoption cycle.
Sceptical	Calculates H by the Standard algorithm and then multiples this by a user-definable modifer. The default value for the modifier is 0.5.
Bouncing	As for Sceptical, but modifier is either +1 or -1, each with equal probability.

Table 2: Influence Algorithms Implemented in the Simulation Model

For simplicity in the simulation exercises reported in this paper, edge strengths were assumed to be all equal, and so they played no part in the results reported in Subsection 5.5.

Similarly, the implemented model permits users to select one from a collection of algorithms which calculate the degree of influence of parent nodes on child nodes at each timepoint, namely the decision factor variable H mentioned in Subsection 5.2 above. These algorithms are described in Table 2. For flexibility of simulation applications, the implemented model permits users to apply any of these algorithms, either to the entire network or to sets of nodes; the algorithm(s) selected may also be changed mid-simulation.

5.4 Model Calibration and Comparison

It is important to recognize that the factor-weights and the decision mechanism have not been calibrated directly against any real-world agent technology adoption decisions in companies or organizations. To undertake calibration of these elements of the model would require collection of market research data on the decision-making processes of organizations considering adoption of agent technologies. Similarly, the five generic network topologies studied in our simulation are abstractions of real-world networks, and may not apply in any particular industry sector. Complete model calibration would require data from randomly-selected organizations in sufficient numbers to give statistical validity to inferences from the model, an expensive and perhaps impossible task. Some calibration may still be possible even without statistically-valid sample sizes, by, for example, basing factor weights on data from a small number of in-depth case studies of the type presented in [12], a study of decision-making in a financial derivatives trading firm.

Given that we have not calibrated the model against any real-world decision data, one could ask what value there would be in undertaking simulation exercises using the model. In [30], economist Ariel Rubinstein presents four purposes for economic modelling in general: to predict behavior; to guide decision-making by economic agents or policy-makers; to sharpen the intuition of economists when studying complex phenomena; and to establish linkages between theoretical economic concepts and everyday thinking. The second, third and fourth of these purposes may be realized even with-

out model calibration, because even an uncalibrated model may provide understanding or guidance which was not available beforehand. Indeed, for multi-agent phenomena, where macro-level system properties may emerge from very different micro-level agent interactions, a multi-agent simulation model can be of great value in aiding human understanding of a domain, particularly in the absence of any other information. As Gilbert and colleagues note [10], uncalibrated simulation models can be used to refine theory, by exploring the consequences of various assumptions and initial conditions. Even for prediction, however, an uncalibrated model may be able to reveal relationships, or their absence, which were previously unrecognized. The simulation results presented in Subsection 5.5, for example, hold across all five industry topologies studied, thus indicating that industry structure was not decisive to the results presented. For further discussion of issues relating to model assurance and verification of multi-agent simulation models, see [24].

It may be useful to compare our approach to related research. Our model differs from traditional models of diffusion of innovations in the marketing literature, e.g., those of [18] or [21], in that we assume adoption decisions by individual firms are influenced by those other firms to which the company is directly connected, and not just by the proportion of adopters amongst the population as a whole. As a consequence, the topology of the network linking companies together is potentially important in the rate and pattern of diffusion, and indeed for other macro-level properties of the model. Participating firms in such networks may well differ in their position in the topology, as may their degree and nature of connectivity. Such heterogeneous-agent models have become of interest to many researchers in economics and social sciences in recent years. For instance, Gilbert and colleagues [10] develop a generic simulation model of an innovation network in which companies are endowed with specific capabilities. Over time, companies may add to their capabilities, either through undertaking research or by exchanging skills with other companies. In this way, companies form partnerships with one another, and networks emerge from these. The authors apply the model to biotechnology and mobile communications domains, in order to study the properties of networks of innovations. There are two main differences between the work of Gilbert *et al.* and our work here. Firstly, [10] focuses on innovation as a means of adding to a company's stock of capabilities, which is not something we consider at all; we ignore any relationship between the technology being adopted and the adopting company's skills or longer-term business performance. Secondly, [10] treats partnerships and networks as emergent properties of the simulation model, whereas we treat the inter-company network as a input to the model which remains static over time.

Other research which presents heterogeneous-agent simulation models of innovation diffusion includes the work of Fagiolo and Dosi [9] and Silverberg and Verspagen [31]. The former work explores the choice by individual companies of deeper exploitation of existing innovations versus a search for new innovations, and studies by means of simulation the macro-level consequences, such as levels of economic growth, of different micro-level choices. The latter paper aims to model the initiation or creation of innovations in a network, where new technologies depend for their creation on the sufficient adoption of earlier technologies. Both these papers have a concern for the public policy implications of innovation and innovation-diffusion, which is not a concern of ours. On the other hand, neither paper considers standards in any depth nor engages with the marketing literature on the diffusion of innovations.

The research closest to our model are the monographs of Tim Weitzel [35] and Falk von Westarp [36]. Weitzel is interested in the impact of standards on the adoption of network-good technologies across networks of companies, and develops a game-

Network Topology	No Standards	Single Standard	Two Standards
Disaggregated industry (non-connected nodes)	66.9	26.5	48.4
Disaggregated industry with dense relationships	66.7	26.8	48.7
Industry with shallow supply chains	25.0	17.6	22.1
Industry with deep, independent supply chains	76.5	26.6	49.1
Industry with deep, overlapping supply chains	67.6	19.8	48.7

Table 3: Average Numbers of Generations to 100% Adoption

theoretic model for the adoption decision of each agent. For any one company, the utilities of adopting different technologies depend upon the numbers of other companies using the technology which are connected to the first company. The model is partly calibrated with data relating to the adoption of X.500 Directory Services in a network of German banks. The work of von Westarp seeks to model the corporate market for computer software with a particular focus on identifying the causal determinants of adoption decisions by companies. The model is calibrated with data from the market for Enterprise Resource Planning (ERP) software. However, von Westarp's model allows *de facto* technology standards to emerge by means of multiple individual adoption decisions, rather than standards-compliance being an attribute of the potential technology under consideration for adoption, as in our model. Both Weitzel and von Westarp study the impact of different network topologies on the diffusion of technologies, just as we do. However, both authors consider a range of topologies chosen without regard to the likely structures of inter-company business networks; we consider that a weakness in applying their findings to real-world marketplaces.

5.5 Simulation Results

One thousand simulation runs were undertaken for each of the five business topologies and with different assumed numbers of technology standards (zero, one and two). For simplicity, the initial conditions adopted were that all edges were of equal strength, and that the influence decision algorithm used to calculate variable H was the *Standard Decision Algorithm*. As described in Table 2, this algorithm calculates H as the average state of all parent nodes, each weighted by its respective edge strength. Because the model assumes several decision factors, namely B, F and G, to be represented by random variables, the outcomes of successive simulation runs could well be different; this was indeed the case. In addition, three other decision factors, C, D and H, depend on the current status of some or all other nodes in the network, and so their values may differ according to the specific diffusion path taken by the network as a whole; this was also found to be the case. In each simulation run, the diffusion model ran until all nodes had adopted the technology, and the number of generations required to reach this end-state was then recorded. These measurements were than averaged across the 1000 simulation runs for each topology and each standards regime, and the results are shown in Table 3.

As would be expected, the network topology can have a major difference in the numbers of generations needed to reach full adoption. Likewise, for any given topology, the presence of a single standard may reduce the time steps needed for full adoption by more than half. Interestingly, having two competing standards inhibits full adoption, but not as greatly as having no standard at all. Thus, the model provides indicative support for the positive impact of standards on technology adoption decisions.

It is also noteworthy that this impact is seen regardless of the business topology or, in other words, regardless of the industry structure, at least for those topologies included in the model simulations.

6 Conclusions

Hardware and software have improved significantly in performance and availability over the six decades of modern computing. As these changes have occurred, the objectives of programmers have also changed. Initially, most programmers sought to minimize memory usage and to maximize throughput or processing speeds in their applications. With increasing availability and lower costs of memory, and increasing microprocessor speeds, these objectives became far less important. Instead, by the 1970s and 1980s, the object-oriented paradigm sought to maximize the modularity and re-usability of code, and to minimize post-deployment system maintenance. However, these objectives too have become dated. Partly, this is because the development of proven OOP methods and support tools have enabled the objectives to be readily achieved, and indeed, taken for granted, over the last two decades. More importantly, however, the rise to prominence of the Internet has led to a new understanding of the nature of computation, an understanding which puts interaction at its centre. In this context, the agent-oriented paradigm has sought to maximize adaptability and robustness of systems in open environments.

It is here that one can particularly see how a new technology may be a disruptive force. By tackling a different set of objectives, agent technologies address different problems and different applications than do object technologies. It is not simply that the rules of the game have changed, but rather that a different game is being played. In a world of millions of independent processors interconnected via the Internet and, through it, engaged in distributed cognition, a software design team can no longer assume that software components or their designers will share the same goals or motivations, or that the system objectives will remain static over time. Systems therefore need to be able to adapt to dynamic environments, to be able to configure, manage and maintain themselves, and to cope with malicious, whimsical or just plain buggy components. The power of the agent paradigm is that it provides the means, at the appropriate level of abstraction, to conceive, design and manage such systems.

Acknowledgments

We gratefully acknowledge the financial support of the European Communities, received through project AgentLink III (IST-FP6-002006 CA), under the Sixth Framework of the Information Society Technologies programme. Of course, the views expressed in this paper are those of the authors only, and are not necessarily those of the European Commission, its agencies or its staff. The EU is not responsible or liable for any use or non-use that may be made of the information or models contained herein. We are also grateful to Elizabeth Sklar for discussions on multi-agent network models.

References

[1] C. J. van Aart, K. van Marcke, R. F. Pels, and J. L. F. C. Smulders. International insurance traffic with software agents. In F. van Harmelen, editor, *Proceedings of*

the Fifteenth European Conference on Artificial Intelligence. IOS Press, 2002.

[2] J. Baxter and R. Hepplewhite. Agents in tank battle simulations. *Communications of the ACM*, 42(3):74–75, 1999.

[3] A. L. C. Bazzan. A distributed approach for coordination of traffic signal agents. *Journal of Autonomous Agents and Multiagent Systems*, 10(2):131–164, 2005.

[4] O.-J. Dahl. The roots of object orientation: the simula language. In M. Broy and E. Denert, editors, *Software Pioneers: Contributions to Software Engineering*, Programming, Software Engineering and Operating Systems Series. Springer, Heidelberg, 2002.

[5] O.-J. Dahl and K. Nygaard. Simula: A language for programming and description of discrete event systems. introduction and user's manual. Technical Report 11, Norwegian Computing Centre, Oslo, Norway, 1965.

[6] V. Darley and D. Sanders. An agent-based model of a corrugated-box factory: the trade-off between finished goods stock and on-time-in-full delivery. In H. Coelho and B. Espinasse, editors, *Proceedings of the Fifth Workshop on Agent-Based Simulation*, 2004.

[7] K. Dorer and M. Calisti. An adaptive solution to dynamic transport optimiation. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track*, pages 45–51. ACM Press, 2005.

[8] T. C. Du, E. Y. Li, and A. P. Chang. Mobile agents in distributed network management. *Communications of the ACM*, 46(7):127–132, 2003.

[9] G. Fagiolo and G. Dosi. Exploitation, exploration and innovation in a model of endogenous growth with locally interacting agents. *Structural Change and Economic Dynamics*, 14:237–273, 2003.

[10] N. Gilbert, A. Pyka, and P. Ahrweiler. Innovation networks - a simulation approach. *Journal of Artificial Societies and Social Simulation*, 4(3), 2001.

[11] L. Gomes. Ventures column. *Wall Street Journal*, 25 February 1998.

[12] I. Hardie and D. MacKenzie. Assembling an economic actor: the *agencement* of a hedge fund. *The Sociological Review*, 55(1):57–80, 2007.

[13] R. Hill, J. Chen, J. Gratch, P. Rosenbloom, and M. Tambe. Intelligent agents for the synthetic battlefield. In *Joint proceedings of the Fourteenth National Conference on Artificial Intelligence and the Ninth Conference on Innovative Applications of Artificial Intelligence*, pages 1006–1012, Providence, RI, USA, 1997.

[14] J. Himoff, P. Skobelev, and M. Wooldridge. MAGENTA technology: Multi-agent systems for industrial logistics. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track*, pages 60–66. ACM Press, 2005.

[15] IDC. *Worldwide Mobile Phone 2005-2009 Forecast & Analysis Report (Report 33290)*. International Data Corporation, May 2005.

[16] S. Jacobi, C. Madrigal-Mora, E. Leon-Soto, and K. Fischer. Agentsteel: an agent-based online system for the planning and observation of steel production. In *AAMAS '05: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 114–119, New York, NY, USA, 2005. ACM Press.

[17] T. Levitt. Exploit the product life cycle. *Harvard Business Review*, 43(6):81–94, 1965.

[18] G. L. Lilien, P. Kotler, and K. S. Moorthy. *Marketing Models*. Prentice-Hall, Englewood Cliffs, NJ, USA, 1992.

[19] M. Luck, P. McBurney, and J. Gonzalez-Palacios. Agent-based computing and programming of agent systems. In R. Bordini, M. Dastani, J. Dix, and A. El Fallah Segrouchni, editors, *Programming Multi-Agent Systems*, volume 3862 of *Lecture Notes in Artificial Intelligence*, pages 23–37. Springer, 2006.

[20] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction. A roadmap for agent based computing*. AgentLink III, 2005.

[21] V. Mahajan, E. Muller, and F. M. Bass. New-product diffusion models. In J. Eliashberg and G. L. Lilien, editors, *Handbooks in Operations Research and Management Science, Volume 5: Marketing*, pages 349–408. North-Holland, Amsterdam, 1993.

[22] J. McKean, H. Shorter, P. McBurney, and M. Luck. The AgentLink III technology diffusion model. Technical Report ULCS-05-008, Department of Computer Science, University of Liverpool, Liverpool, UK, 2005.

[23] D. F. Midgley. *Innovation and New Product Marketing*. Croom Helm, London, 1977.

[24] D. F. Midgley, R. E. Marks, and D. Kunchamwar. The building and assurance of agent-based models: an example and challenge to the field. *Journal of Business Research*, 2006. *In press*.

[25] G. A. Moore. *Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Consumers*. HarperCollins, New York City, NY, USA, 1991.

[26] S. Munroe, T. Miller, R. Belecheanu, M. Pechoucek, P. McBurney, and M. Luck. Crossing the agent technology chasm: Lessons, experiences and challenges in commercial applications of agents. *Knowledge Engineering Review*, 21(4):345–392, 2006.

[27] J. Nicolaisen, V. Petrov, and L. Tesfatsion. Market power and efficiency in a computational electricity market with discriminatory double-auction pricing. *IEEE Transactions on Evolutionary Computation*, 5(5):504–523, October 2001.

[28] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Agent-based formation of virtual organisations. *Knowledge-Based Systems*, 17:103–111, 2004.

[29] E. M. Rogers. *Diffusion of Innovations*. The Free Press, New York City, NY, USA, 1962.

[30] A. Rubinstein. *Modeling Bounded Rationality*. Zeuthen Lecture Book Series. MIT Press, Cambridge, MA, USA, 1998.

[31] G. Silverberg and B. Verspagen. A percolation model of innovation in complex technology spaces. *Journal of Economic Dynamics and Control*, 29(1–2):225–244, 2005.

[32] B. Stroustrup. *The C++ Programming Language*. Addison Wesley, 1985.

[33] G. L. Urban and J. R. Hauser. *Design and Marketing of New Products*. Prentice-Hall, Englewood Cliffs, NJ, USA:, 1993.

[34] T. Wagner, L. Gasser, and M. Luck. Impact for agents. In M. Pechoucek, D. Steiner, and S. Thompson, editors, *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems: Industry Track*, pages 93–99. ACM Press, 2005.

[35] T. Weitzel. *Economics of Standards in Information Networks*. Information Age Economy Series. Physica, Heidelberg, 2004.

[36] F. von Westarp. *Modeling Software Markets: Empirical Analysis, Network Simulations, and Marketing Implications*. Information Age Economy Series. Physica, Heidelberg, 2003.