Computing as Interaction: Agent and Agreement Technologies

Michael Luck
Department of Computer Science
King's College London
London WC2R 2LS, UK
Email: michael.luck@kcl.ac.uk

Peter McBurney
Department of Computer Science
University of Liverpool
Liverpool L69 3BX, UK
Email: mcburney@liverpool.ac.uk

I. Introduction

With the emergence of new paradigms for computing, such as peer-to-peer technologies, grid computing, autonomic computing and other approaches, it is becoming increasingly natural to view large systems in terms of the services they offer, and consequently in terms of the entities or agents providing or consuming services. For example, web services technologies provide a standard means of interoperating between different software applications, running on a variety of platforms. More generally, web services standards now serve as potential convergence point for diverse technology efforts in support of more general service-oriented architectures.

Here, distributed systems are increasingly viewed as collections of service provider and service consumer components interlinked by dynamically defined workflows. Web services must thus be realised by concrete entities or *agents* that send and receive messages, while the services themselves are the resources characterised by the functionality provided.

The important characteristics of these emerging domains and environments are that they are open and dynamic so that new agents may join and existing ones leave. In this view, agents act on behalf of service owners, managing access to services, and ensuring that contracts are fulfilled. They also act on behalf of service consumers, locating services, agreeing contracts, and receiving and presenting results. In these domains, agents are required to engage in interactions, negotiate with one another, make agreements, and make proactive run-time decisions, individually and collectively, while responding to changing circumstances. In particular, agents need to collaborate and to form coalitions of agents with different capabilities in support of new virtual organisations.

II. Agent-Based Computing

Agents can be defined as autonomous, problem-solving computational entities capable of effective operation in dynamic and open environments. Agents are often deployed in environments in which they interact, and sometimes cooperate, with other agents (both people and software) that have possibly conflicting aims. Such environments are multi-agent systems. Agents can be distinguished from objects in that they are autonomous entities capable of exercising choice over their actions and interactions, and may act to achieve individual objectives. Agents cannot, therefore, be directly invoked but can be assigned tasks by their owners. However, they may be constructed using a wide range of technologies. These notions find application in relation to three distinct views.

First, agents provide designers and developers with a way of structuring an application around autonomous, communicative components and lead to the construction of software tools and infrastructure to support design. They provide a new and often more appropriate method for the development of complex systems, especially in open and dynamic environments. To support this view of systems development, particular tools and techniques need to be introduced. For example, agent-oriented methodologies to guide analysis and design are required, agent architectures are needed for the design of individual components and tools and abstractions are required to enable developers to deal with the complexity that is typical of systems with such distribution of control.

Agent technologies are distinct and cover a range of specific techniques for dealing with interactions in dynamic, open environments. They address issues such as balancing reaction and deliberation in individual agent architectures, learning from and about other agents in the environment, eliciting and acting upon user preferences, finding ways to negotiate, agree and cooperate with other agents, and developing appropriate means of forming and managing coalitions.

Finally, multi-agent systems offer strong models for representing real-world environments with an appropriate degree of complexity and dynamism. Simulation of economies, societies and biological environments are typical application areas. The use of agent systems to simulate real-world domains may provide answers to complex physical or social problems that would be otherwise unobtainable, as in the modelling of the impact of climate change on biological populations, or modelling the impact of public policy options on social or economic behaviour. Multi-agent systems have already provided faster and more effective methods of resource allocation in complex environments, such as the management of utility networks, than previous centralised approaches.

III. Trends and Drivers

The development of agent technologies has taken place within a context of wider visions for information technology.

In addition to the specific technologies mentioned above, there are also several key trends and drivers that suggest that agents and agent technologies will be vital. The three considered below are examples; for a large list, see [10].

A. Semantic Web

Since it was first developed in the early 1990s, the World Wide Web has rapidly and dramatically become a critically important and powerful medium for communication, research and commerce. However, the Web was designed for use by humans, and its power is limited by the ability of humans to navigate the data of different information sources.

The Semantic Web is based on the idea that the data on the Web can be defined and linked in such a way that it can be used by machines for the automatic processing and integration of data across different applications [1]. This is motivated by the fundamental recognition that, in order for web-based applications to scale, programs must be able to share and process data, particularly when they have been designed independently. The key to achieving this is by augmenting web pages with descriptions of their content in such a way that it is possible for machines to reason automatically about that content. Among the particular requirements for the realisation of the Semantic Web vision are: rich descriptions of media and content to improve search and management; rich descriptions of web services to enable and improve discovery and composition; common interfaces to simplify integration of disparate systems; and a common language for the exchange of semantically-rich information between software agents.

It should be clear from this that the Semantic Web demands effort and involvement from the field of agent-based computing, and the two fields are intimately connected. Indeed, the Semantic Web offers a rich breeding ground for both further fundamental research and a whole range of agent applications that can (and should) be built on top of it.

B. Web Services and Service Oriented Computing

Web services technologies provide a standard means of interoperating between different software applications, running on a variety of different platforms. Specifications cover a wide range of interoperability issues, from basic messaging, security and architecture, to service discovery and the composition of individual services into structured workflows.

In a more general sense, web services standards serve as a potential convergence point for diverse technology efforts such as eBusiness frameworks (ebXML, RosettaNet, etc), Grid architectures (which are now increasingly based on web services infrastructures) and others, towards a more general notion of service-oriented architectures (SOA). Here, distributed systems are increasingly viewed as collections of service provider and service consumer components, interlinked by dynamically defined workflows. Web services can therefore be realised by agents that send and receive messages, while the services themselves are the resources characterised by the functionality provided. In the same way as agents may perform tasks on behalf of a user, a web service provides this functionality on behalf of its owner, a person or organisation.

Web services thus provide a ready-made infrastructure that is almost ideal for use in supporting agent interactions in a multi-agent system. More importantly, perhaps, this infrastructure is widely accepted, standardised, and likely to be the dominant base technology over the coming years. Conversely, an agent-oriented view of web services is gaining increased traction and exposure, since provider and consumer web services environments are naturally seen as a form of agent-based system [2].

C. Grid Computing

The Grid is the high-performance computing infrastructure for supporting large-scale distributed scientific endeavour that has recently gained heightened and sustained interest from several communities [6]. The Grid provides a means of developing eScience applications such as those demanded by, for example, the Large Hadron Collider facility at CERN, engineering design optimisation, bioinformatics and combinatorial chemistry. Yet it also provides a computing infrastructure for supporting more general applications that involve large-scale knowledge management and service provision.

The Grid thus refers to an infrastructure that enables the integrated, collaborative use of high-end computers, networks, databases, and scientific instruments owned and managed by multiple organisations. Grid applications often involve large amounts of data and computer processing, and often require secure resource sharing across organisational boundaries; they are thus not easily handled by todays infrastructures. The key benefit of Grid computing more generally is flexibility — the distributed system and network can be reconfigured on demand in different ways as business needs change, in principle enabling more flexible IT deployment and more efficient use of computing resources. According to BAE Systems [7], while the technology is already in a state in which it can realise these benefits in a single organisational domain, the real value comes from cross-organisation use, through virtual organisations, which require ownership, management and accounting to be handled within trusted partnerships. In economic terms, such virtual organisations provide an appropriate way to develop new products and services in high value markets; this facilitates the notion of service-centric software, which is only now emerging because of the constraints imposed by traditional organisations. The future of the Grid is not in the provision of computing power, but in the provision of information and knowledge in a service-oriented economy.

D. Ambient Intelligence

The notion of ambient intelligence has largely arisen through the efforts of the European Commission in identifying challenges for European research and development in Information Society Technologies. Aimed at seamless delivery of services and applications, it relies on the areas of ubiquitous computing, ubiquitous communication and intelligent user interfaces. The vision describes an environment of potentially

thousands of embedded and mobile devices (or software components) interacting to support user-centred goals and activity, and suggests a component-oriented view of the world in which the components are independent and distributed. The consensus is that autonomy, distribution, adaptation, responsiveness, and so on, are key characteristics of these components, and in this sense they share the same characteristics as agents.

Ambient intelligence requires these agents to be able to interact with numerous other agents in the environment around them in order to achieve their goals. Such interactions take place between pairs of agents (in one-to-one collaboration or competition), between groups (in reaching consensus decisions or acting as a team), and between agents and the infrastructure resources that comprise their environments (such as large-scale information repositories). Interactions like these enable the establishment of virtual organisations, in which groups of agents come together to form coherent groups able to achieve overarching objectives.

IV. Agent Technologies

It should be clear that there are several distinct high-level trends and drivers leading to interest in agent technologies, and low-level computing infrastructures making them practically feasible. In this context, we can consider the key technologies and techniques required to design and implement agent systems that are the focus of current research and development. Because agent technologies are mission-critical for engineering and for managing certain types of information systems, such as Grid systems and systems for ambient intelligence, the technologies and techniques discussed below will be important for many applications, even those not labelled as agent systems. These technologies can be grouped into three categories, according to the scale at which they apply:

- Organisation-level: At the top level are technologies and techniques related to agent societies as a whole. Here, issues of organisational structure, trust, norms and obligations, and self-organisation in open agent societies are paramount. Once again, many of these questions have been studied in other disciplines — for example, in sociology, anthropology and biology. Drawing on this related work, research and development is currently focused on technologies for designing, evolving and managing complex agent societies.
- Interaction-level: These are technologies and techniques that concern the communications between agents — for example, technologies related to communication languages, interaction protocols and resource allocation mechanisms. Many of the problems solved by these technologies have been studied in other disciplines, including economics, political science, philosophy and linguistics. Accordingly, research and development is drawing on this prior work to develop computational theories and technologies for agent interaction, communication and decision-making.
- Agent-level: These are technologies and techniques concerned only with individual agents — for example, procedures for agent reasoning and learning. Problems at this level have been the primary focus of artificial intelligence since its inception, aiming to build machines that can reason and operate autonomously in the world. Agent research and development has drawn extensively on this prior work, and most attention in the field of agent-based computing now focuses at the previous two higher levels. In addition to technologies at these three levels, we must also consider technologies providing infrastructure and supporting tools for agent systems, such as agent programming languages and software engineering methodologies. These supporting technologies provide the basis for both the theoretical understanding and the practical implementation of agent systems.

V. Agreement Technologies

Cross-cutting with agent technologies are *agreement* technologies. In particular, while the trends and drivers above motivate the need for agent solutions, there are several key characteristics that complicate the picture.

- Typically, the applications in such domains span multiple organisations, so that there is no single point of oversight over all components.
- Agents in such applications need to operate in open, public environments in which a variety of third party operators must be able to connect to and use various provided services, so that participants may range from trusted through semi-trusted to untrusted.
- Agents and services must often adhere to complex laws, regulations, rules and agreements during their operation, relating to quality of service, uptime, failure rates, etc, raising the need to monitor not only security breaches but also quality of performance, for example.
- Applications operate in environments in which component subsystems are not all available to all developers, so that many must be treated as black boxes during development and cannot be directly inspected or controlled.

In such applications, two aspects are critical: autonomy and interaction. At the agent-level, agents must establish commitments to bringing about goals in the context of more general autonomous behaviour, typically seeking to maximise their utility. To constrain the potential excesses of autonomous behaviour, *agreements* are made between agents, at the interaction-level, providing some kind of guarantee supporting inter-agent relationships. Such agreements may be informal commitments between individual agents or services, or they may be stronger contractual commitments between individuals or organisations, with enforcement and/or penalties included as part. Specifically, these issues require techniques that enable software components to reach agreements typically on the performance of services. Negotiation, argumentation, decision-making, virtual organisations, contracts, normative reasoning, trust and others are the kinds of technologies that are likely to figure strongly in the next generation of system designed to address these concerns.

VI. Adoption of agent and agreement technologies

Despite the benefits, agent and agreement technologies have not yet entered the mainstream in the way that object-oriented technologies have. The majority of commercial organisations adopting agent technologies would be classified as early adopters, so considerable potential exists for further applications of the technology.

To date, the range of applications has included: automated trading in online marketplaces; simulation and training applications in defence domains; network management in utilities networks; user-interface and local interaction management in telecommunication networks; schedule planning and optimisation in logistics and supply-chain management; control system management in industrial plants, such as steel works; and simulation modelling to guide decision-makers in public policy domains, such as transport and medicine [11].

For example, Tankers International, which operates one of the largest oil tanker pools in the world, has applied agent technology to dynamically schedule the most profitable deployment of ships-to-cargo for its Very Large Crude Carrier fleet [8]. An agent-based optimiser was developed by Magenta Technology for use in real-time planning of cargo assignment to vessels in the fleet. 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. In similar vein, after implementing recommendations derived from an agent-based simulation model of a corrugated box plant developed by Eurobios, SCA Packaging was able to make a 200% return-on-investment in the first month [4].

Yet agent technologies are still only in the early-adopter phase of diffusion. There are a number of reasons for this. Firstly, research in the area of agents is also still only in its infancy. Here, a reasonable comparison is with object-oriented (OO) approaches, where the initial research commenced in 1962, some 32 years before the public release of the first version of Java and the widespread commercial adoption of OO technologies. So knowledge of agent technologies is still not widespread among commercial software developers. Secondly, since it is still young, 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 it. Thirdly, the applications for which agent technologies are most suited are those involving interactions between autonomous intelligent entities. While some applications of this sort may be implemented as closed systems inside an organisation, most potential applications require the participation of entities from more than one organisation. Automated purchase decisions along a supply-chain, for example, require the participation of the companies active along that chain.

The application domains for which agent technologies are best suited typically require coordination and collaboration between multiple organisations, a factor that complicates adoption decisions by the companies or organisations involved.

VII. Challenges

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 maximise adaptability and robustness of systems in open environments. It is here that we can see how agent technologies may be a disruptive force. By tackling a different set of objectives, agent technologies address different problems and different applications than do, for example, 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 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.

A. Broad Challenges

Each of the compelling visions discussed in the context of trends and drivers above — the Semantic Web, ambient intelligence, the Grid, autonomic systems — will require agent technologies, or something very like them, before being realised: agent technologies are upstream of these visions and mission-critical to them. For agent-based computing to support these visions, considerable challenges remain, both broad, overarching challenges across the entire domain of agent technologies, and challenges specific to particular aspects. The broad challenges are as follows.

- Creating tools, techniques and methodologies to support agent systems developers. Compared to more mature technologies such as object-oriented programming, agent developers lack sophisticated software tools, techniques and methodologies to support the specification, development and management of agent systems.
- Automating the specification, development and management of agent systems. 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 organisations. These challenges are probably several decades from achievement, and will draw on domain-specific expertise (for example, economics, social psychology and artificial intelligence).

- Integrating components and features. Many different theories, technologies and infrastructures are required to specify, design, implement and manage agent systems. Integrating these pieces coherently and cost-effectively is usually a major undertaking in any system development activity, a task made more challenging by the absence of mature integration tools and methodologies.
- Establishing appropriate trade-offs between adaptability and predictability. Creating systems able to adapt themselves to changing environments, and to cope with autonomous components, may well lead to systems exhibiting properties that were not predicted or desired. Striking a balance, appropriate to the specific application do- main, between adaptability and predictability is a major challenge, as yet unresolved either theoretically or practically. Associated with predictability is the requirement for practical methods and tools for verification of system properties, particularly in multi- agent systems that are likely to exhibit emergent behaviour.
- Establishing appropriate linkage with other branches of computer science and with other disciplines, such as economics, sociology and biology. One task here is to draw appropriately on prior research from these other areas and disciplines. Another task is to avoid reinvention of existing techniques and methods, whether by agent researchers or by others. Awareness-building between areas and disciplines, and coordination of research and development activities, are essential if the appropriate linkages are to be established and maintained.

B. Specific Challenges

Specific technical challenges continue to change as the field of agent-based computing advances and matures, and as related areas (like those discussed above) emerge and galvanise efforts that contribute to the general area. Inevitably, standards will continue to be critical, but it is not clear whether these should come from within the agent community or should emerge from more general computing infrastructure progress. Nevertheless, in addition to the broad challenges, there are challenges specific to different aspects and features of agent systems [3], [5].

1) Trust and reputation: Sophisticated distributed systems are likely to involve action in the absence of strong existing trust relationships. While middleware addresses secure authentication, and there exist techniques for verification and validation, these do not consider the harder problems of establishing, monitoring, and managing trust in a dynamic, open system. As discussed earlier, we need new techniques for expressing and reasoning about trust and reputation, on both an individual and a social level to enable interaction in dynamic and open environments.

2) Virtual organisation formation and management: Virtual organisations (VOs) have been identified as one of the key contributions of Grid computing, but principled and well-defined procedures for determining when to form new VOs, how to manage VOs and portfolios of VOs, how to manage competing and complementary VOs, and ultimately how and when to disband them, are still missing. Moreover, the development of procedures and methods for the automation of VO creation, management and dissolution also provide major research and development challenges. In addition, once such procedures have been defined, creating formal representations of them to support their automated deployment by agents themselves at runtime will be a major research challenge.

3) Resource allocation and coordination: The coordinated, autonomic management of distributed resources requires new abstractions, mechanisms and standards in the face of multiple, perhaps competing, objectives from different stakeholders, and different definitions of individual and social welfare. Most R&D effort to date has focused on allocation and coordination mechanisms drawn from human societies (for example, common auction protocols), but the processing power and memory advantages of computational devices mean that completely new mechanisms and protocols may be appropriate for automated interactions, in particular for multi-objective coordination and negotiation. In addition, as with VOs, the automation of the design, implementation and management of mechanisms is a major challenge.

4) Negotiation: To date, work on negotiation has provided point solutions. There is a need for a solid theoretical foundation for negotiation that covers algorithms and negotiation protocols, while determining which bidding or negotiation algorithms are most effective under what circumstances. From the system perspective, behaviour arising through the interplay of different negotiation algorithms must be analysed, and determining what kind of negotiation to consider, and when, must be established. Finally, effective negotiation strategies and protocols that establish the rules of negotiation, as well as languages for expressing service agreements, and mechanisms for negotiating, enforcing, and reasoning about agreements are also needed. Incorporating capabilities for disagreement and justifications (i.e. arguments) in negotiations is also a major research challenge.

5) Contracting and Verification: While impressive progress is being made in new generations of network application technologies such as Web Services and Grid Computing, fundamental questions remain about how such systems can be effectively deployed in a safe, dependable, and secure manner. Current point solutions address particular aspects of the problem, but do not address what is arguably the central issue behind how such large-scale applications are designed, deployed and managed: how to reliably model, track and manage dependencies between the components of networked applications to ensure secure, dependable operation at runtime. To address this, we must draw on a range of techniques developed in the fields of social science, agent technology, protocol engineering and software engineering to make it possible to model, build, verify and monitor systems on the basis of dynamically generated, cross-organisational contracts.

In addition to verification of individual system components (which are often not available or may be extremely complex) we can also use formal specifications of publicly declared

commitments between systems. This separates the responsibilities in distributed application design (between component service providers and the overall designer of the system) and allows verification procedures to operate only over higher level public specifications which are potentially less detailed that specifications of internal functioning of components. We need:

- theoretical frameworks for contract-based computing: providing standard models with formal semantics for large-scale open distributed application environments and dependencies between components in such environments;
- contract-based Web Services application frameworks, providing a suite of practical tools combining standard Web Service based application design capabilities with additional contracting features to constrain and formalise component interactions, and providing entry points for higher level verification and monitoring tools; and
- verification, monitoring and analysis tools for dependable systems — implemented tools that apply model checking techniques over distributed applications in order to verify and monitor their behaviour, working towards better specified and understood networked applications.

6) Methodologies: Many of todays 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 methodologies have so far been developed, often addressing different elements of the modelling 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.

7) Service architecture and composition: There is a need for integrated service architectures providing robust foundations for autonomous behaviour, in order to support dynamic services, and important negotiation, monitoring, and management patterns. This will aid application and deployment of agent technologies to the Grid and other domains. While web service technologies define conventions for describing service interfaces and workflows, we need more powerful techniques for dynamically describing, discovering, composing, monitoring, managing, and adapting multiple services in support of virtual organisations, for example. This is likely to take the form of agent-oriented architectures based on peer-to-peer or other novel structures.

VIII. Future prospects

The vision of agent-based computing itself is enough to constitute a grand challenge because of the need to bring together multiple technical and scientific disciplines as well as stakeholders across different sectors. The specific technical challenges continue to change as the field of agent-based computing advances and matures and as related areas emerge and galvanise efforts that contribute to the general area.

Inevitably, standards will continue to be critical, but it is not clear whether these should come from within the agent community or should emerge from more general computing infrastructure progress. Nevertheless, some key challenges have already been articulated in relevant areas.

In seeking to identify these challenges and to identify the emerging opportunities, AgentLink (a European project aimed at promoting industrial and commercial deployment of agent technologies) has developed a roadmap for agent-based computing. The roadmap provides a more complete assessment of the current state of the art, considers the research issues, reviews existing deployments for business benefit and outlines the likely future development of both the research field and the commercial environment.

Acknowledgements

This paper draws on the AgentLink Roadmaps [9], [10] and the CONTRACT project. The CONTRACT project is co-funded by the European Commission under the 6th Framework Programme for RTD with project number FP6-034418. Notwithstanding this fact, this paper and its content reflects only the authors' views. The European Commission is not responsible for its contents, nor liable for the possible effects of any use of the information contained therein.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 35–43, May 2001.

[2] D. Booth, H. Haas, F. McCabe, E. Newcomer, M. Champion, C. Ferris, and D. Orchard. Web services architecture. Note 11, W3C Working Group, 2004.

[3] S. Bullock and D. Cliff. Complexity and emergent behaviour in ict systems. Technical report, Foresight Report, DTI, UK, 2004.

[4] 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.

[5] I. Foster, N. R. Jennings, and C. Kesselman. Brain meets brawn: Why grid and agents need each other. In *Proceedings of the Third International Conference on Autonomous Agents and Multi-Agent Systems*, pages 8–15. ACM Press, 2004.

[6] I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 2004.

[7] A. Gould, S. Barker, E. Carver, D. Golby, and M. Turner. Baegrid: From e-science to e-engineering. In *Proceedings of the UK e-Science All Hands Meeting*, 2003.

[8] J. Himoff, P. Skobelev, and M. Wooldridge. Magenta technology: Multi-agent systems for industrial logistics. In *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2005.

[9] M. Luck, P. McBurney, and C. Preist. A manifesto for agent technology: Towards next generation computing. *Autonomous Agents and Multi-Agent Systems*, 9(3), 2004.

[10] M. Luck, P. McBurney, O. Shehory, and S. Willmott. *Agent Technology: Computing as Interaction (A Roadmap for Agent-Based Computing)*. AgentLink, 2005. `http://www.agentlink.org/roadmap`.

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