Agents in Decentralised Information Ecosystems: The DIET Approach

P. Marrow[1]*, M. Koubarakis[2†], R.H. van Lengen[3‡], F. Valverde-Albacete[4§], E. Bonsma[1], J. Cid-Suerio[4], A.R. Figueiras-Vidal[4], A. Gallardo-Antolín[4], C. Hoile[1], T. Koutris[2], H. Molina-Bulla[4], A. Navia-Vázquez[4], P. Raftopoulou[2], N. Skarmeas[2], C. Tryfonopoulos[2], F. Wang[1], C. Xiruhaki[2]

[1]Intelligent Systems Laboratory, BTexaCT, Adastral Park, Ipswich IP5 3RE, UK.

[2]Technical University of Crete, Department of Electronic and Computer Engineering, TUC Campus, Chania 73 100, Crete, Greece.

[3]German Research Centre for Artificial Intelligence, Intelligent Visualisation and Simulation Systems Department, Erwin-Schrödinger-Str. 1, D-67608 Kaiserslautern, Germany.

[4]Universidad Carlos III de Madrid, Departmento Tecnologias de las Communicaciones, Avda. de la Universidad S/N, Leganés 28110, Spain.

*E-mail: paul.marrow@bt.com, †E-mail: manolis@ced.tuc.gr, ‡E-mail: lengen@dfki.de, §E-mail: fva@tsc.uc3m.es

Abstract

The complexity of the current global information infrastructure requires novel means of understanding and exploiting the dynamics of information. One means may be through the concept of an information ecosystem. An information ecosystem is analogous to a natural ecosystem in which there are flows of materials and energy analogous to information flow between many interacting individuals. This paper describes a multi-agent platform, DIET (Decentralised Information Ecosystem Technologies) that can be used to implement open, robust, adaptive and scalable ecosystem-inspired systems. We describe the design principles of the DIET software architecture, and present a simple example application based upon it. We go on to consider how the DIET system can be used to develop information brokering agents, and how these can contribute to the implementation of economic interactions between agents, as well as identifying some open questions relating to research in these areas. In this way we show the capacity of the DIET system to support applications using information agents.

1 Introduction: Information Ecosystems

The modern world encompasses a global information infrastructure of staggering complexity. This information infrastructure may be expected to become even more complicated as the capabilities of the Internet and World Wide Web are extended. Means of understanding and effectively exploiting this complexity are urgently needed. The information ecosystem concept, which compares transactions in the information infrastructure to the dynamics of natural ecosystems, may be an important means of doing this.

According to this approach, an *information ecosystem* is a complex web of interactions arising between information producers and consumers where information is interpreted in its widest sense. The term information ecosystem is used by analogy with natural ecosystems. An ecosystem is an entity formed from the combination of communities of living organisms, their interactions with each other, and the physical environment that they inhabit (Waring, 1989). In comparison, in networks of information exchange that make up information ecosystems, we are faced with an emphasis on interactions between entities (information producers and consumers) in an environment of continuous change, caused by commercial, political, social and technological developments.

Information ecosystems ideas can improve our understanding of information infrastructures in two ways. The first is through what has been called the ecology of computation (Huberman, 1988); that is the modelling or simulation of existing computational systems in a manner inspired by natural ecosystems. The second is through the implementation of ecologically inspired interactions in computational architectures in order to complete particular tasks. Previous work in this area includes various aspects of artificial life (Langton, 1989), information economics (IBM, 2000; Kearney & Merlat, 1999) and bottom-up approaches to multi-agent systems (Van Parunak et al., 1997; Moukas, 1996; Menczer & Monge, 1999; Minar et al., 1999).

In this paper we focus on the second direction, considering in particular the use of multi-agent systems to build information ecosystems, and the potential for the development of applications based on these ideas. In the next section we review existing multi-agent systems and their relevance to information ecosystems issues. We show how multi-agent systems together with other technologies can provide a basis for the construction of information ecosystems.

We then go on to outline how these ideas have informed our development of a decentralised light-weight agent platform in which to develop information ecosystem applications, through the DIET project. DIET (the acronym stands for Decentralised Information Ecosystem Technologies), is a European collaborative research project. We give an overview of the design principles behind this software, which can be used as a basis for the development of a range of information processing applications.

We then look to the future and to the type of information processing that the DIET system can support. We consider the different classes of agents that are useful in these scenarios (information producers, consumers, and brokers) and give an overview of their functionality and interactions. Finally we consider some of the major open questions that need to be considered in the development of applications based upon decentralised multi-agent systems such as the DIET system.

2 Agents for Information Ecosystems

Software agents are arguably a key technology with which to tackle information ecosystems issues. This is because software agents can deal efficiently with information in many different ways. Multi-agent systems, in particular, can produce a complex network of informational interactions appropriate for information ecosystems. All software agents can be described as information agents, but we prefer the definition of an information agent (Maes, 1994; Decker & Sycara, 1997) as one that has a function concerned with the manipulation of information. Using information agents of this sort, one can construct an interacting network of information users and consumers, that is an information ecosystem.

For example, Decker and Sycara (1997) describe the construction of software architectures for information agents on the Internet. They combine three types of agents, interface, task and information agents, that together can carry out a portfolio management function. Sycara et al. (1996) describe a related agent system for information filtering. Moukas (1996) reports on *Amalthaea*, another distributed information filtering application. In these agent-based systems the interactions between the agents and the resulting collective properties of the agents combined into a society become more important than centralised control of the software system used.

Van Parunak et al. (1997) exploit the collective properties of such distributed software systems in the industrial context to make what they call industrial synthetic ecosystems. These software systems have been applied to a range of real industrial problems, such as shop-floor scheduling. They differ from many conventional control systems in that the key components, the software agents, are very simple, diversified and generalised, and it is their interaction in the synthetic ecosystem that produces the relevant control of the industrial system. As such, the industrial synthetic ecosystem of Van Parunak et al. could alternatively be considered an information ecosystem.

Agent systems of this sort typically exploit the emergent consequences of market-based interaction. Wellman and colleagues (e.g., Wellman, 1996; Wurman & Wellman, 1997) have used market-based interactions to control distributed systems, in a field they call market-oriented programming. Applications using this technique have been developed for resource allocation (Wellman, 1996), information service provision and other areas.

Kearney and others in the DYNAMO project (Kearney and Merlat, 1999; Kearney et al., 2000) have exploited the potential of economic interactions for the control of supply chains, developing systems of interacting trading agents that use market and evolutionary mechanisms to set prices. These systems are able to utilise economies of scale, and also retain the flexibility to deal with changing market conditions. IBM's Information Economics Project (IBM, 2000) provides another example of research into agent systems engaged in economic interactions.

Other examples of multi-agent systems relevant to information ecosystems research include the InfoSpiders system developed by Menczer and Monge (1999). This system implements a scalable information search algorithm by use of cooperative agents, drawing explicitly on ecological metaphors. By contrast, the Hive system (Minar et al., 1999) uses distributed agents to link networked resources on a local network. Another example is given by Brewington et al. (1999) who use a mobile multi-agent system for distributed information retrieval. The MATS system developed by Ghanea-Hercock et al. (1999) also uses mobile agents, in this case inspired by social insects, to control a distributed processing application over a network.

It can be seen from these examples that there is a substantial track record in the application of multi-agent systems to the type of information manipulation

problems that are pertinent to the study of information ecosystems.

The variety of features of natural ecosystems in addition to those that inspire multi-agent systems suggests that ideas from other areas may also inform the construction of information ecosystems. One feature of natural ecosystems is the existence of resources. Living organisms survive in the context of limited resources. Information ecosystems do not need to have resources explicitly defined. However the inclusion of resources allows the development of control and allocation algorithms based upon biologically-inspired resource accounting, or by drawing upon parallel inspiration from economics (Huberman, 1988; Wellman, 1996; Wellman and Wurman, 1997).

A consequence of varying availability of resources and interaction with the environment is the occurrence of adaptation by individuals to changing circumstances. In the natural world a distinction can be made between behavioural adaptation during the lifetime of an individual and evolutionary adaptation over multiple generations. The distinction may be worth maintaining in the information ecosystem context. The field of evolutionary computation provides many possibilities for the implementation of evolutionary adaptation in an information ecosystem (e.g., Bäck et al., 1997); a wide range of other machine learning techniques are available to implement behavioural adaptation.

Another source of inspiration from natural ecosystems are the social insects, where relatively simple and unintelligent creatures can combine to produce very complex artefacts and behaviours (Wilson, 1971). Such organisms have already inspired work in multi-agent systems (e.g., Marrow and Ghanea-Hercock, 2000). Ideas from social insects among others have informed the field of Artificial Life (e.g., Langton, 1989) which covers a wide range of models and systems with characteristics that include many of those already mentioned. It too may be able to contribute to the development of agent-based information ecosystems. In particular the emphasis in Artificial Life research on developing complex systems through the bottom-up combination of simple elements to support the emergence of complexity may prove useful in developing information ecosystems based upon the interaction of many software agents.

3 The DIET Approach

The variety of sources of inspiration from economics to Artificial Life suggests that an initial focus on developing flexible systems of interacting agents is appropriate. For this reason we are interested in developing DIET initially with comparatively lightweight agents. If individual agents can be kept as lightweight as possible, many more agents can be incorporated in the system, and their numbers can be varied much more easily. While the development of lightweight agents precludes the inclusion of some computationally intensive capabilities at the individual agent level, there is potential for the emergence of such properties in the overall system, through interactions between agents. This emphasis on lightweight agents and bottom-up interaction is in consequence the strategy that we follow in the DIET core platform, although it does not preclude the incorporation of more heavyweight agents where required when extending this platform. More heavyweight agents may, for example, be needed to include sophisticated reasoning or communication capabilities. Nor does it preclude the use of more top-down Artificial Intelligence techniques, which complement and enhance the functionality provided by bottom-up interactions.

Based upon these ideas we are able to address the goals of the DIET project through the construction of an agent platform to study and implement information ecosystems.

The goals of the DIET project are the following:

- To design and implement a novel agent[1] framework via a substantially bottom-up and ecosystem-oriented approach leading to an open, robust, adaptive and scalable software platform.
- To validate and demonstrate the usefulness of the platform via four tasks/applications: information retrieval, information alert, information mining, and information trading.
- To research into the effects of alternative forms of interaction among different types of agents under ecologically inspired software models.

The first goal depends, in our view, on keeping the functionality, memory and processing requirements associated with each individual agent to a minimum. However agents should be designed such that they can combine or interact in a variety of ways so as to carry out functions that they are not capable of individually. In doing so, we keep close to the inspiration from natural ecosystems that underpins the information ecosystem concept. Starting our design in this way makes it much more likely that the emergent multi-agent framework will be scalable as higher numbers of agents are needed, or large amounts of information need to be processed.

This lightweight, bottom-up design should also contribute to the aim of supporting adaptive responses in the platform. Lightweight agents could more easily

[1] Agents may be called infohabitants in DIET. The terminology has its roots in the call for proposals from the European Commission.

serve as the subject of population-based adaptive algorithms such as evolutionary algorithms. In addition, the diversity of possible configurations possible in interactions between lightweight agents should assist the search for robust solutions, and allow easy modification if these are not initially found.

The flexibility of the design approach that we advocate here should allow us to consider the application of the DIET software platform to a variety of different information manipulation applications, based upon a set of information processing operations that will be required by some or all applications. At the same time it should provide a platform for the study of some of the outstanding research issues concerning interactions in multi-agent systems.

In the next section we outline the features of the DIET software platform, upon which all applications built in the DIET system will be based.

4 The DIET software platform

The DIET software platform is designed to form the base for information management applications. To be useful in practise, the framework needs to support applications that are:

- *adaptive*: Information gets updated constantly, and new information is generated. Users of the information, and their preferences, as well as the system load and infrastructure, can also change. To operate efficiently, information management applications have to adapt to these changes.
- *scalable*: There is a massive amount of information available in the real world, consider for example the World Wide Web. For an information management system to be useful, it need to be built without any implicit limits on its size.
- *robust*: Failures are inevitable in large-scale, dynamic, distributed systems. So the system needs to be able to cope with them. It needs to handle failing hardware, as well as cope with high system load. Performance should gracefully degrade when parts of the system fail.
- *decentralised*: A lot of information is located in a distributed form, as the World Wide Web demonstrates. Decentralisation also helps to enhance scalability, by avoiding critical bottlenecks, and robustness, as it reduces the reliance on particular parts of the system.

The DIET platform has therefore been designed with these properties in mind.

4.1 Layered architecture

The architecture of DIET software is layered, incorporating modularity that allows for the flexible extension of the framework (see figure 1). The kernel of the DIET software resides in the bottom layer, the *core layer*. It provides the fundamental functionality available to all implementations in the DIET architecture, but also embodies the constraints under which all DIET agents must operate. The *application reusable component layer* (ARC layer) includes optional components that are useful to various applications. It also contains general components that allow for the validation and testing of DIET applications. The *application layer* is the third layer and contains application-specific code. Associated with this layer may be validation components, to enable validation of applications developed using the DIET platform. Finally, the DIET architecture provides for software for visualisation of the components within the platform.

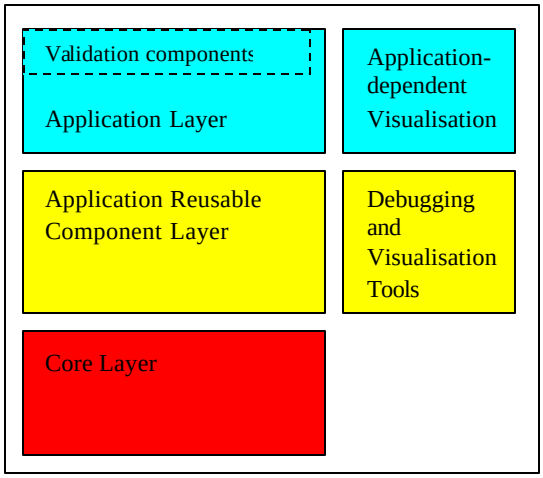

Figure 1: The DIET architecture.

4.2 The kernel

The two fundamental components defined by the DIET kernel are infohabitants and environments. (Recall that an “infohabitant” is a term for agents within DIET, and we will use it in the description of the software platform). The only capability built into every infohabitant is the ability to communicate with each other. Every infohabitant has an identity consisting of a binary name tag and a binary family tag. The name tag allows infohabitants to be uniquely identified, and the family tag can be used to look up infohabitants by functionality. Family tags can be used as a shared identifier for a group of agents, or as an identifier which can be set according to an agreed rule, or as an evolved identifier which externally distinguishes a specific sub-species of adaptive agent.

An environment provides a location for infohabitants to reside in. It also provides infohabitants access to the

basic services provided by the DIET kernel. An environment always resides on a single computer. However, one computer can host multiple environments, and DIET is designed to run on multiple computers.

The DIET kernel has been designed to be as lightweight as possible. It provides only essential services, and always in a very basic way. This helps to make it more scalable, and enables it to deliver its functionality quickly and efficiently. Robustness is explicitly addressed in the DIET core by directly exposing infohabitants to potential failure. This allows them to adapt to it and change their behaviour accordingly.

The following services are offered by the DIET kernel:

- New infohabitants can be created. To create an infohabitant, you specify the type of infohabitant to create and the parameters to use. The kernel checks the type of the infohabitant to ensure that it obeys a few basic rules. The kernel also assigns a randomly generated name tag to the infohabitant. This is a simple, efficient mechanism to allocate names that are in practice unique. For instance, when tags are 128 bits long, the probability that two or more infohabitants in a group of one million have the same name is 1.5×10^{-26}. The infohabitant can choose its own family tag, but the kernel ensures that the identity of an infohabitant remains fixed throughout its lifetime.
- Infohabitants can connect to other infohabitants in the same environment. Once connected, infohabitants can communicate by sending messages or passing additional objects to each other. By only allowing local connections, infohabitants can receive immediate feedback when sending a message. It was either delivered successfully, or it was rejected. Connections can be established in only two ways, by specifying either a complete identity or the family tag of the required infohabitant. Both look-up mechanisms can be implemented and used very efficiently. More complicated directory functionality can be built on top of this base functionality and provided by infohabitants if needed.
- Infohabitants can move to another environment. Infohabitant mobility is useful to make the system more adaptable to changing circumstances. It allows an infohabitant to select from various execution environments, depending upon the availability and efficiency of environmental services, and it can help in the formation of neighbourhoods of social interaction between infohabitants. The kernel does not guarantee that every migration attempt succeeds. When an infohabitants wants to move, but the destination environment can not accept it, for example because it is currently off-line or has reached its full capacity, the infohabitant dies.

For both communication and migration the kernel support is minimal. It can therefore be implemented very efficiently. It also allows more sophisticated functionality to be build on top of it, implemented in a way best suited to the conditions in which it is used. The minimalistic implementation is partly achieved because the kernel only fullfills any request when it can easily do so, but immediately fails when it cannot. It has the additional advantage that it implicitly forces infohabitants to take the effect of their actions on their execution environment into account.

Another feature of the kernel is that explicit limits can be imposed on several elements in the system such as the number of threads that are in use and the size of the message buffer of every infohabitant. These limits help to make the DIET platform more robust with respect to system load. For example, when a new message arrives at an infohabitant whose incoming message buffer is full, it is simply rejected. This mechanism ensures that the system will not run out of memory when one or more infohabitants receive more messages than they can handle. It also exposes the sending infohabitant to the congestion so that it can adapt its behaviour accordingly.

The kernel has also been designed to support lightweight infohabitants. The minimal requirements by an infohabitant on system resources such as memory and CPU use is very low. One interesting feature is that infohabitants can temporarily give up their thread when they do not need it. When an external event occurs, e.g. a message arrives, the kernel attempts to give it a thread again so that it can handle the message.

4.3 The ARC layer

Above the core layer the DIET software platform will include a range of components that will draw upon ecological and evolutionary inspiration, as well as machine learning and other mechanisms, in order to provide diverse sources of flexibility to facilitate the adaptation of DIET agents to changing circumstances. This layer will also include the elements that support a range of information management and manipulation applications.

The ARC layer also provides infohabitants that offer services not directly provided by the DIET kernel. Infohabitants can access these services through their family tags. For example, the ARC layer defines a CarrierPigeon infohabitant that can be used to deliver messages to infohabitants in a different environment and thus provides basic remote communication. Other infohabitants defined in the ARC layer can use these CarrierPigeons to offer various more sophisticated ways to communicate remotely.

4.4 A simple example

The DIET platform is geared towards a bottom-up software design. A simple example application implemented in DIET is now described to illustrate how to best exploit the functionality provided by the DIET kernel.

This application is built in layer three of the DIET architecture (recall that, from figure 1, this layer, the application layer, is where application-specific code is located). It draws upon software components in the application reusable component (ARC) layer and in the core layer.

The application has been designed to arrange Linker infohabitants in a sorted sequence. The Linker infohabitants are sorted according to (the binary values of) their identity. All Linkers are passive and only react to incoming messages. Each tries to maintain a connection to two other Linkers: both as close as possible to its own identity but one with a lower identity and one with a higher identity. When a Linker receives a message with the identity of another Linker, it checks if it improves either of its existing links. If it does, it updates the link and sends its own identity to the corresponding Linker. Otherwise it forwards the received identity to the link with an identity closest to it. The sorting process is driven by Trigger infohabitants. They are active. Every once in a while they randomly select two Linkers and tell one about the other's identity. Figure 2 shows a screenshot of a visualisation of this application. Individual Linker and Trigger infohabitants are shown by blocks – lines between them indicate links being formed between infohabitants.

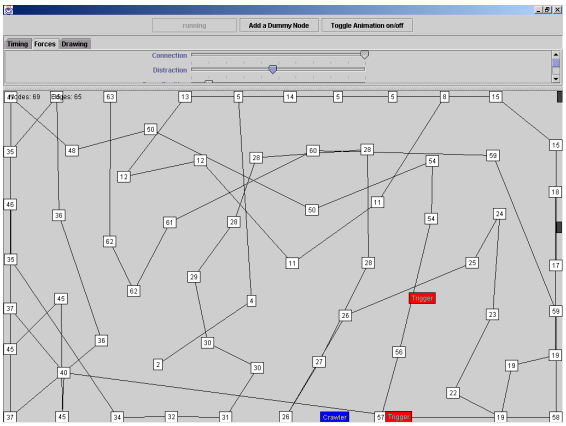

Figure 2. Example of an application running on the DIET software platform.

The application has a few interesting features. Firstly, although admittedly simple, this local and stochastic algorithm always leads to a perfectly sorted list. Secondly, as sequences get longer, a single trigger action typically results in a larger cascade of messages. Therefore, Triggers have a simple mechanism to adapt their activity to the perceived system load. This simple adaptation strategy means that the system operates close to optimal efficiency despite changing system load. Thirdly, the algorithm is robust. Even when messages get lost or Linker infohabitants are killed, the algorithm keeps on functioning. Finally, the ability of the kernel to support light-weight infohabitants means that the application can run on a single computer with more than 100,000 Linker infohabitants.

5 Information Ecosystems at Work: Producers, Consumers and Brokers

We now consider some of the functions that will be required in working information ecosystems. These will be implemented in layers two (the ARC layer) and layer three (the application layer) of the DIET architecture. To do this we need to think about the information infrastructure in which information ecosystem applications will be deployed.

The global information infrastructure of the future will be populated by millions of agents that act as *information producers*, *information consumers* or intermediaries for transactions (*brokers*). DIET is devoted to the study of ecosystem-inspired techniques for the development of systems inhabited by such *information agents.*

In information ecosystems *information brokering* is a very important problem. For example, in the WWW today there is an important need for "middlemen" that will allow information (or service) requesters to find efficiently information (or service) providers that can fulfil their requests. Similarly, information (or service) providers need effective ways to target their advertisements to appropriate consumers. Some of this functionality is today provided in a very rudimentary way by search engines, web directories, vertical portals or various e-commerce firms. To realise the full potential of the WWW, we need to go beyond the simple functionality offered by these technologies and design brokers with the following attributes:

- They must be able to handle a wide range of descriptions of information content as advertised by providers, and requested by consumers.

- They must be efficient with respect to communication, scalable (i.e., their performance should not degrade in the

presence of large numbers of events to be handled), robust (i.e., less vulnerable to failures) and adaptive (i.e., can accommodate varying workloads and varying numbers of information requesters and providers).

The need for brokers with the above functionality is not new and has already been documented in many branches of distributed computing and in open multi-agent systems e.g., (Sycara et al., 1997). Previous work on information brokering can be classified in various ways (Chi Wong and Sycara, 2000) but here we would like to concentrate on the degree of decentralisation offered by existing systems. A review of related research reveals three main approaches:

- The *centralised approach* where a single broker is employed. Typical examples of this approach are the matchmakers of SHADE and COINS (Kuokka and Harada, 1995), and A-Match (Sycara et. al., 1999). Other centralised systems that are *not* multi-agent systems but could be viewed as such are information dissemination systems like SIFT (Yan and Garcia-Mollina, 1999) and information integration systems like TSIMMIS (Garcia-Mollina et al., 1997).
- The *distributed approach* where the brokering task is handled by multiple co-operating brokers. This approach is usually mentioned in many papers but very few implemented systems of this kind exist (we only know of IDIoMS [Soltysiak et al., 2000]). There are also very few papers dealing with theoretical aspects of the problem (see Jha et al., 1998 for example).
- The *"revolutionary"* approach that does away with brokers altogether (i.e., brokering is carried out individually by each participating agent). To the best of our knowledge this approach has only been argued in Shehory (1999). Outside of the area of multi-agent systems the same approach has been implemented in peer-to-peer content sharing systems such as Gnutella (2000) and Freenet (Clarke et al., 2000).

In DIET we are mainly interested in the two decentralised solutions to information brokering mentioned above. We believe that decentralised configurations of agents can provide the robustness, scalability and adaptivity needed in the global information ecosystems of the future.

We are currently developing a distributed textual information retrieval and alert application on top of the basic DIET API. Using this application as our testbed we would like to explore the following research questions, among others:

- What are appropriate communication languages, protocols and algorithms for systems of multiple co-operating brokers?
- What kind of properties can *emerge* in societies of co-operating brokers?
- What implementation techniques are appropriate for developing societies of multiple co-operating brokers?

The above questions largely concentrate on the aspects of information brokering that have to do with management of information, and would also interest more traditional computer science audiences (e.g., distributed systems researchers). DIET aims to go beyond this view and additionally consider the economic interactions arising by various combinations of information producers, consumers and brokers. Economic interactions have been used previously to drive decentralised agent systems in contexts such as market-based computation (Wellman, 1996). The flexible response to changing circumstances generated by the flow of economic resources has strong parallels with resource dynamics in natural ecosystems (Waring, 1989) and is thus worth considering in an information ecosystem together with more directly nature-inspired ideas.

In implemented systems like the ones mentioned above information producers, consumers and brokers should be rewarded for contributing useful information, and should be penalised for being idle or for consuming system resources. The main question of our investigation then becomes to analyse and implement configurations of *economically-motivated* information agents that exhibit sophisticated and/or useful behaviour. For example:

- What configurations of information agents can achieve states where the overall utility of the system (and not just the utility of each individual) has been maximized?
- How can we design and implement information sharing systems that discourage the free-riding behaviour now seen in Napster and Gnutella (Adar & Huberman, 2000; Mojo Nation, 2000)?

The above kinds of questions are particularly interesting given the potential for intelligent information services to provide solutions for future e-commerce systems and other commercial applications.

6 Conclusions

We have introduced a new approach to the development of multi-agent systems as decentralised information ecosystems through the DIET project. The DIET software platform provides a basis for groups of information producers, consumers and brokers to

interact and so contributes to the emergence of more adaptable and robust multi-agent systems in the future.

Acknowledgements

Research described in this paper was supported in part by the Future and Emerging Technologies arm of the IST Programme of the European Union, under the FET Proactive Initiative – Universal Information Ecosystems (FET, 1999), through project DIET (IST-1999-10088). We also acknowledge the support of BT's Group Technology Programme.

We thank the members of the BTexaCT Intelligent Systems Laboratory for stimulating discussion and comments during the preparation of this paper.

References

E. Adar, and B.A. Huberman. Free Riding in Gnutella. Available from http://www.parc.xerox.com/istl/groups/iea/papers/gnutella/

T. Bäck, D. Fogel, and Z. Michalewicz (eds.) *Handbook of Evolutionary Computation*. Institute of Physics, Bristol, 1997.

B. Brewington, R. Gray, K. Moizumi, D. Kotz, G. Cybenko, and D. Rus. Mobile agents in distributed information retrieval. In: *Intelligent Information Agents*, M. Klusch (ed.), Springer, Berlin, 1999.

H. Chi Wong, and K. Sycara. A Taxonomy of Middle-Agents for the Internet. In: *Proceedings of 4th International Conference on Multi Agent Systems* (ICMAS-2000), Boston, Massachusetts, 2000.

I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. Available from http://freenet.sourceforge.net/

K. Decker, and K. Sycara. Intelligent adaptive information agents. *Journal of Intelligent Information Systems* 9(3):239-260, 1997.

FET. Future and Emerging Technologies web site: http://www.cordis.lu/ise/fetuie.htm, 1999.

Frictionless web site: http://www.frictionless.com

H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. D. Ullman, V. Vassalos, and J. Wisdom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems* 8(2):117-132, 1997.

R. Ghanea-Hercock, J.C. Collis, and D.T. Ndumu. Co-operating mobile agents for distributed parallel processing. In: *Proceedings of Autonomous Agents 1999*, Seattle, 1999.

Gnutella web site: http://www.wego.gnutella.com/

B. Huberman, (ed.) *The Ecology of Computation*. Elsevier, Amsterdam, 1988.

IBM Information Economies Project web pages: http://www.research.ibm.com/infoecon/

S. Jha, P. Chalasani, O. Shehory and K. Sycara. A Formal Treatment of Distributed Matchmaking. *Proceedings of Autonomous Agents' 98*, pp. 457-458, 1998.

P. Kearney, and W. Merlat. Modelling market-based decentralised management systems. *BT Technology Journal* 17(4):145-156, 1999.

P. Kearney, R.E. Smith, C. Bonacino, and T. Eymann. Integration of computational models inspired by economics and genetics. *BT Technology Journal* 18(4):150-161, 2000.

D. Kuokka, and L. Harada. Matchmaking for Information Agents. *Proceedings of IJCAI '95*, pages 672-678, Montreal, Canada, 1995.

C.G. Langton. *Artificial Life*. Addison-Wesley, Redwood City, 1989.

P. Maes. Agents that reduce work and information overload. *Communications of the ACM* 37(7):31-40, 1994.

P. Marrow and R. Ghanea-Hercock. Mobile software agents – insect-inspired computing. *BT Technology Journal* 18(4):129-139, 2000.

F. Menczer, and A. E. Monge. Scalable web search by adaptive online agents: an InfoSpiders case study. In: *Intelligent Information Agents*, M. Klusch, (ed.), Springer, Berlin, 1999.

N. Minar, M. Gray, O. Roup, R. Krikorian, and P. Maes. Hive: distributed agents for networking things. In: *Proceedings of ASA/MA '99*, 1999.

Mojo Nation web site: http://www.mojonation.com/

A. Moukas. Amalthaea: information discovery and filtering using a multiagent evolving ecosystem.

In: *Proceedings of PAAM96*, 1996.

S. Soltysiak, T. Ohtani, M. Thint and Y. Takada. An Agent-Based Intelligent Distributed Information Management System for Internet Resources. Available at http://www.isoc.org/inet2000/cdproceedings/2f/2f_1.htm

O. Shehory. A Scalable Agent Location Mechanism. ATAL 1999:162-172, 1999.

K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert* 11(6):36-46, 1996.

K. Sycara, K. Decker, and M. Williamson. Middle-Agents for the Internet. *Proceedings of IJCAI-97*, 1997.

K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *SIGMOD Record* 28(1):47-53, 1999.

H. Van Parunak, J. Santer, and S. Clark. Toward the specification and design of industrial synthetic ecosystems. In: *Proceedings of ATAL'97*, 1997.

R. H. Waring. Ecosystems: fluxes of matter and energy. In: *Ecological Concepts*. J. M. Cherrett (ed.), Blackwell Scientific, 1989.

M. P. Wellman. Market-oriented programming: some early lessons. In: S. Clearwater (ed.), Market-Based Control: A Paradigm for Distributed Resource Allocation. World Scientific, Singapore, 1996.

M. P. Wellman, and P. R. Wurman. Market-aware agents for a multiagent world. In*: Proceedings of MAAMAW-97*, 1997.

E. O. Wilson. *The Insect Societies*. The Belknap Press, Cambridge, 1971.

T. W. Yan, and H. Garcia-Molina. The SIFT Information Dissemination System. *ACM Transactions on Database Systems* 24(4):529-565, 1999.