Implementing Self-Organising Virtual Enterprises Using Social Behaviour Nets

Ping Jiang[1], Quentin Mair[2] and Mingwei Yuan[3]

[1] Department of Computing, The University of Bradford, Bradford BD7 1DP, UK
[2] School of Engineering and Computing, Glasgow Caledonian University, Glasgow G4 0BA, UK
[3] Department of Information and Control, Tongji University, Shanghai 200092, China
qma@gcal.ac.uk

Abstract. Frameworks to support Internet-based virtual enterprises are currently characterised as having fixed process models, fixed meta-data models and a fixed set of users at project inception. This paper proposes the SoBeNet architecture to support a highly dynamic, self-organised, Internet-based approach based on social interaction. The architecture is based on a multi-agent approach, Internet communication being achieved through RSS feeds. RSS conveys state changes by carrying data defined in a common ontology. Each agent assesses state changes using a virtual sensor by calculating the semantic distance between the change and its own interests. Each agent also contains a behaviour network which uses an activation spreading paradigm to react to a combination of its own goals and state changes. The SoBeNet test-bed is based on the Java JADE environment.

Keywords: multi-agent, behaviour network, ontology, virtual organisation

1 Introduction

Social interaction is a dynamic, changing sequence of social actions between individuals (or groups) who try to affect or take account of each others' subjective experiences or intentions. Today the Internet has augmented the traditional methods of social interaction. According to the UK Office of National Statistics there are now around 15 million households in Britain alone with Internet access. As a result the Internet has become an important tool for social interaction supporting activities such as chatting, shopping, dating and information exchange. An Internet-based virtual organisation (VO) will provide a set of tools to support the creation and operation of an organisation of *ad-hoc* partners who have come together for a cooperative project. The Internet supports individuals who may want to work together across organisational boundaries but who do not have much prior knowledge on which to base relations. VOs are highly dynamic and their operation unpredictable. Therefore a key aspect of social interaction in VOs to study is how to organise independent individuals with a limited understanding of each other. It is also necessary to address

how to exchange local information and orchestrate local processes together. These characteristics reflect the *ad hoc opportunistic* nature of a VO [1][2].

Research into the support necessary to implement social interaction in a VO has been widely carried out in different areas such as: virtual enterprises [3], e-business [4], professional virtual communities (PVCs) [5] and grid computing; common requirements have been identified [6][7][8]. These include: the ability to assign job priorities through collaborative shared goals, coordinated and controlled access to shared resources, delegation, discovery services and collaborative processes. Two general approaches can be taken to implement a VO: transaction-oriented layer-based and as a distributed multi-agent system [3]. The transaction-oriented layer-based approach usually applies a top-down design approach to specify goals and subgoals. A protocol for interaction is defined beforehand by a consortium or standards body as a reference model, including meta-data and process models, and is subsequently deployed by customising the cooperation layer of an existing IT infrastructure. This produces a structurally static VO. Examples of this approach can be seen in the ARPA NIIIP (National Industrial Information Infrastructure Protocols) and the EU DIECoM (Distributed Integrated Environment for Configuration Management) projects. However it has been found that this approach is deficient in open and dynamic environments due to its inflexibility and operational inefficiency [3]. The next generation of VOs are faced with a more dynamic environment, requiring user functionality akin to that found in current Web 2.0 applications: RSS feeds, WIKI, Blogs, Social Book Mark, FaceBook, Podcast etc. Such VOs will be required to be able to provide flexible and fast responses to changing information and goals during operation. In comparison to the transaction-oriented layer-based approach we propose that multi-agent based approaches [1][8][9][10] are more suitable to support the next generation of VOs. These may offer a better approach to support the lifecycle of VOs from partnership seeking, partnership configuration, partnership execution to partnership dissolution. An agent can be viewed as perceiving its environment through sensors and acting upon that environment through actuators [11]. Whilst RSS feeds in the Internet provide almost instant awareness of information updates, software agents could help Internet users conduct real-time and automatic interaction with others. Although general frameworks to support distributed agent-based VOs have been developed, e.g. mobile-agent based architectures [1], the required intelligent mechanisms and techniques have not been well studied to support such social interaction.

In this paper we propose a novel synthesis of approaches to deal with a dynamic and partially observable VO for social interaction in the Internet. We propose a **Social Behaviour Network** (SoBeNet) i.e. a distributed behaviour network-based approach organised in a more dynamic, loosely coupled and self-organisational way than that previously attempted in VOs. Our behaviour net concept is based on the goal-driven behaviour nets first proposed by Maes [12]. A SoBeNet is a distributed multi-agent system which uses the Internet. An agent in a SoBeNet senses RSS updates of other agents via virtual sensors and feeds these to a virtual controller. The virtual controller manages a behaviour net fragment, execution of the behaviour only taking place when it has enough activation energy from its goal or from state changes detected through RSS. After the agent executes a state change it will update its RSS documents and cause activation energy to be spread to other behaviour nets. Hence distributed agents

interact with each other in a spontaneous way and can form a VO with diverse local goals/missions.

The rest of this paper is structured as follows. Section 2 presents a high-level overview of our SoBeNet system. Section 3 focuses on our main contribution in this paper – the adaptation of Maes behaviour networks to the SoBeNet system. Section 4 discusses the implementation of a SoBeNet software platform built on the Java Agent Development Environment (JADE). Section 5 discusses a case study validation scenario. Finally Section 6 presents conclusion and future work.

2 SoBeNet Architecture

A SoBeNet is composed of distributed agents supporting the whole lifecycle of social interaction in the Internet but without a predefined process model. Multiple agents work towards diverse local goals but are self-organised into a cooperative social behaviour network. The interaction between agents takes place spontaneously through RSS feeds which allow changes of web content to be monitored. The schematic diagram of a SoBeNet is shown in Fig.1:

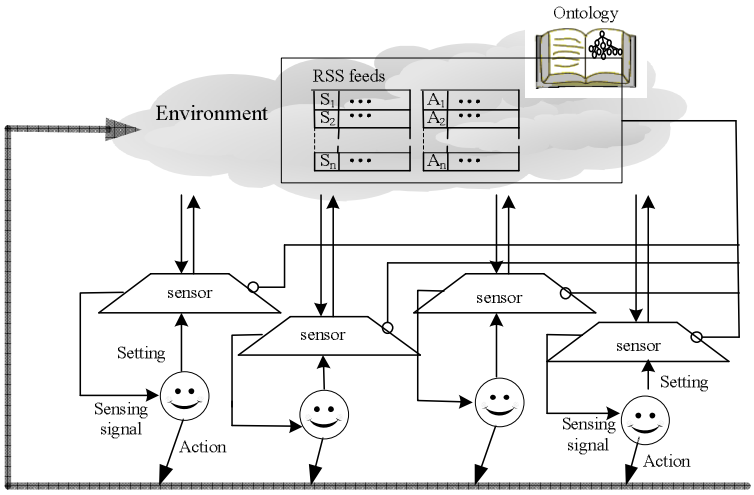

Fig.1. Architecture of a SoBeNet

An ontology allows rich semantics to be carried via RSS as OWL documents. These collectively constitute the virtual state space of the SoBeNet. To avoid an extremely complicated and large state space only the relevant aspects of a set of tasks are modelled in an ontology. The RSS documents in the state space change dynamically. The use of an ontology enables agents to understand the meaning of RSS feeds from other agents. This functionality is implemented by virtual sensors.

The structure of our proposed agents is shown in Fig.2. Each agent in a SoBeNet has several virtual sensors. A virtual sensor can perceive and reason about changes in RSS feeds and acts as a text mining agent for the semantic retrieval of RSS

documents. It serves two purposes. Firstly it allows behaviour net fragments to detect relevant changes in the state space and goals i.e. detect events for behaviour activation. Secondly, for interpretation of RSS documents, it facilitates fuzzy matchmaking between the expected criteria for interaction and the received RSS feeds from candidates. A virtual sensor conducts logic-based binary assessment about the executability of a behaviour and is also able to provide a measure of the semantic closeness between two RSS documents.

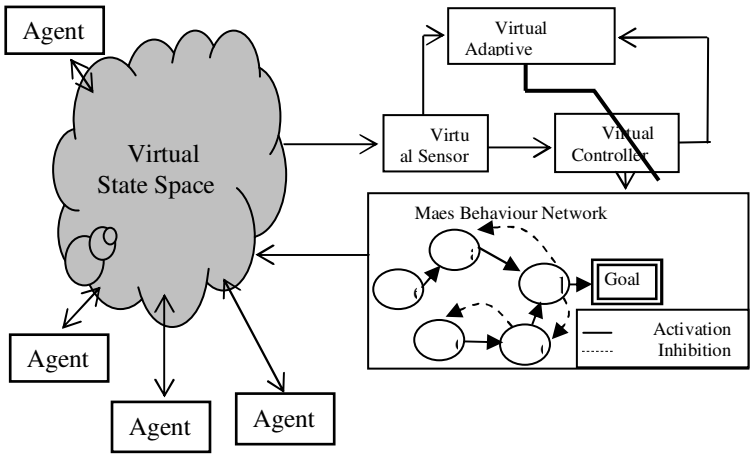

Fig. 2. SoBeNet agent structure diagram

Each agent (Fig. 2) contains a local Maes behaviour network which is governed by a virtual controller. The virtual controller selects reasonable actions according to activation/inhibition energies induced from the virtual sensors and local goal, which will be presented in section 3. The social interaction in the VO is thus driven by goals and the state of artefacts in the virtual state space but without an explicit cooperation model.

In a SoBeNet each virtual sensor perceives the state changes of other agents from RSS feeds. To define the semantic data content of RSS feeds for a specific domain, we define an ontology for the data source. This is defined in OWL. We use a semantic distance measure to evaluate the relevance of RSS feed changes together with a hierarchical clustering analysis approach. In this paper, for brevity, we do discuss this further but the interested reader is referred to our other work in [13].

In addition each agent maintains a virtual adaptive machine to provide the virtual controller with learning and predictive capability. This maintains a belief model about the VO-based on historical experience and predicts other agents' future behaviour patterns. An agent is thus able to reason on the suitability of other agents to take part in collaboration for achieving its goal.

3 An Energy Driven Social Behaviour Network

A behaviour network can be represented as a three-layer network as shown in Fig.3: sensor layer, behaviour layer and goal layer. The virtual sensors are located in the sensor layer. A user can set goals or sub-goals at the goal layer. The behaviour layer includes behaviour modules linked by activation or inhibition channels for energy spreading. A behaviour network can be formalised as a tuple (G,M, Π), where G is a set of goals; M is a set of behaviour modules and Π is a set of parameters that control the energy spreading. An agent executes a behaviour module in M when its activation energy level exceeds an activation threshold defined in Π.

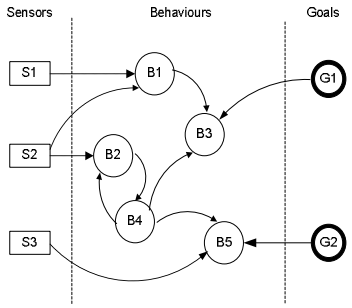

Fig.3. A behaviour network

Energy control parameters defined in Π are normalised into [0,1] following [15], which includes:

$\gamma \in [0,1]$ activation parameter of module
$\delta \in [0,1]$ inhibition parameter of module
$\beta \in [0,1]$ inertia parameter of activation
$\phi \in [0,1]$ activation parameter of state
$a \in [0, \hat{a}]$ activation threshold,with $\hat{a}$ the upper bound
η normalisation constant

The energy control parameters affect the performance of a behaviour network. Their values can be set by users from their own experience or set through a machine learning process.

Each behaviour module M_i in M can be defined by another tuple (p_i, b_i, Eff_i, h_i, r_i) in which b_i represents a functional behaviour. p_i represents a precondition list to be satisfied in order to activate b_i, and r_i is the desired resolution of a virtual sensor. Taking p_i as the subscriber interest, the virtual sensor receives a RSS stream s and generates the degree of behaviour executability, denoted as $\tau(p_i,s)$. A higher resolution expects a more precise check of executability but a lower resolution means a fuzzier condition. Eff_i:={eff^+, eff^-} defines a set of effects after execution of the behaviour b_i with eff^+ representing the positive effects and eff^- representing the negative effects, $eff^+ \cap eff^- = \phi$. Exp(eff^+) and Exp(eff^-) denote the expectations of energy injection from eff^+ and eff^- to a linked behaviour respectively. The value h_i denotes the

activation energy level of b_i, accumulated from goals or other modules. When the activation energy level exceeds the activation threshold $a_i \in \Pi$, the behaviour can be activated. Energy flows through links between behaviour modules. Consider a link from behaviour *A* to behaviour *B*. *B* is called a successor of *A* if *A* has a positive effect (eff^+) in the proposition of *B*'s precondition, whilst *A* is called a predecessor of *B*. There is a conflicting link from *A* to *B* if *A* has a negative effect (eff^-) in the proposition of *B*'s precondition. A behaviour module can take energy from current states and final goals directly or indirectly. A direct acquisition means energy obtained from the current sensors and goals; an indirect acquisition means energy obtained from other behaviour modules.

Execution of a behaviour network is carried out by repeatedly scanning activation energy in individual behaviour modules. For each step only the behaviour with the highest energy level will be activated.

A key characteristic of behaviour networks is that the behaviours that currently cannot be executed spread activation energy backwards to predecessor behaviours. This allows sub-goals to be addressed. As this can be applied iteratively it facilitates a VO to be self-organising.

4 SoBeNet Implementation

A SoBeNet test-bed has been developed using JADE (Java Agent Development Framework). JADE was developed by Telecom Italia Lab in compliance with FIPA (Foundation for Intelligent Physical Agents) specifications. As the JADE architecture is based on Java this allows us to easily extend the architecture to incorporate our SoBeNet functionality. JADE is also open source. It is efficient and scalable for large multi-agent systems [14] and flexible enough to be deployed in fixed and mobile environments [15]. JADE provides support for automatic document exchange in XML and RDF [15].

Each running instance of JADE is a container which could include one or more agents. The set of active containers constitutes a platform in which only a single main container need be active for administration and coordination purposes, i.e. the main container contains the JADE AMS (Agent Management System) and DF (Directory Facilitator) components. Distributed agents can communicate with each other through asynchronous messages defined in ACL (Agent Communication Language). The main container includes a blackboard agent in addition to the AMS and DF components. This provides the virtual state space describing the environment and the information sources monitored by the virtual sensors. The blackboard agent receives RSS feeds subscribed to by user agents and provides a common space for information exchange between agents. This information includes RSS data, domain ontology, agent IDs and a message routing table. Each container representing a user includes a sensor agent S_i and a controller agent C_i. The sensor agent implements the proposed virtual sensor. A user is allowed to subscribe to RSS feeds, define interests and configure sensor parameters, e.g. resolution. The controller agent maintains a set of behaviour modules. Each extends the JADE behaviour class to implement in code a local behaviour network. Each behaviour module is activated by energy from virtual

sensors and other behaviours. The user is allowed to configure the controller agent, e.g. energy spreading parameters and activation threshold. Note that each user can also receive RSS feeds from its sensor agent and publish a RSS information update using its controller agent directly rather than via the Blackboard agent. Due to the current lack of websites allowing bi-directional and interactive RSS information exchange, we use the blackboard agent as a common information pool for carrying out interactive RSS information exchange.

5 Case Study Validation Scenario

We have validated the work by creating a virtual organisation for a new product development as a simulation scenario. Given 20 SoBeNet agents which are either job-hunters or development leaders, we attempt to show that a hierarchical virtual organisation can be organised automatically. The information sources come from RSS feeds including websites at Yahoo Hotjobs, UK Academic Jobs, and CareerBuilder. Because no RSS feeds are available today to announce personal abilities, a simulated RSS source is created in the blackboard agent. This is obtained by inserting bindings of user names to job descriptions received from job advertisement websites.

An agent will publish a job description in RSS and wait for applications. Once a competent applicant appears it will notify the user and the development leader will confirm if the job should be offered. The social interaction for organising a development team is completed automatically by generated events; from virtual sensors, timers and user input.

The role of a user is not fixed; conceivably a user can take the role of a job-hunter for the overall task and then decompose the job into subtasks and then become a development leader for the purposes of subcontracting. Thus a hierarchical VO can be formed. The goal here is to subcontract to others the development of new hardware with an embedded software element. After execution of the publish-job behaviour the job specification is added and the "match-candidates" behaviour is activated. The virtual sensor detects the top 3 candidates as Agents A, B and C. Agent A is identified as the best matched. Agent A then divides the job into subtasks and becomes the development leader for the hardware development and software development subtasks. The candidates with highest similarity, Agents B and C, are selected by the "Match-job" behaviour and invited to join the team. Agent C refused the invitation for "hardware development" and Agent B with the second highest similarity was recruited into the team. This task reallocation process is repeated until no further subcontracting is possible.

The simulation shows that each agent does not know the other agents in advance and executes independently according to sensed information. Social interaction happens naturally by activating distributed local behaviours. This is more suitable for interaction in dynamic and uncertain environments than traditional process model-based approaches.

6 Conclusions

Social interactions in the Internet are often characterised by a set of organisations or individuals who come together to perform an intellectually focussed project over some period of time. Such virtual organisations may exhibit unpredictable events and operate with previously unknown participants. It is often difficult to adapt traditional process model-based approaches especially if a reorganisation or problem arises mid-project. In a SoBeNet each person is aided by a software agent which embeds semantic sensors and a behaviour network, taking RSS feeds as the information source. The behaviour nets manage all agent actions enabling them to achieve individual goals concurrently. This ability allows them to react to changes in the Internet e.g. the alteration of a behaviour mid-project or the replacement of a project partner. Social interaction is thus spontaneous. A SoBeNet test-bed was developed using the Java-based JADE toolkit and validated with a new product development scenario.

References

1. Aerts, A.T.M., N.B. Szirbik, and J.B.M. Goossenaerts, "A flexible, agent-based ICT architecture for virtual enterprises," Computers in Industry, 49, 311-327, 2002.
2. Martinez, M.T., K.H. Park, and J. Favrel, "Virtual enterprise: organisation, evolution and control," Int. J. Production Economics, 74, 225-238, 2001.
3. Camarinha-Matos, L. M., H. Afsarmanesh, "Elements of a base VE infrastructure", Computers in Industry, 51, 139–163, 2003.
4. Burn, J., P. Marshall, M. Barnett, E-business Strategies for Virtual Organizations, Elsevier, Kent, UK, 2002.
5. Santoro, R., Bifulco, A., "PVC Reference Framework", European Society of Concurrent Engineering – Net, http://www.ami-communities.eu/, 2008
6. EC DataGrid, "VOMS vs EDG security requirements," Work Package 6, 2002.
7. EC ECOLEAD, "Challenges in virtual organisations management," D32.1, Work Package 3, 2004.
8. Norman, T. J., A. Preece, et al., "Agent-based formation of virtual organizations," Knowledge-Based Systems, 17, 103–111, 2004.
9. Goldman, C. V. and J. S. Rosenschein, "Evolutionary patterns of agent organizations," IEEE Trans. Systems, Man, And Cybernetics—Part A, 32, 135-148, 2002.
10. Subbu, R. and A. C. Sanderson, "Network-based distributed planning using coevolutionary agents: architecture and evaluation," IEEE Trans. Systems, Man, And Cybernetics—Part A, 34, 257-269, 2004.
11. Russell, S. and P. Norvig, Artificial Intelligence: A Modern Approach. Prentice Hall, Upper Saddle River, New Jersey, second edition, 2003.
12. Maes, P., "Situated agents can have goals," Robotics and Autonomous Systems, 6, 49-70, 1990.
13. Jiang, P., Q. Mair Q., Z. Feng, "Agent alliance formation using ART-networks as agent belief models," Journal of Intelligent Manufacturing, 18 (3): 433-448, 2007.
14. Chmiel, K., M. Gawinecki, P. Kaczmarek, M. Szymczak, M. Paprzycki, "Efficiency of JADE agent platform," Scientific Programming, 13, 1-14, 2005.
15. Bellifemine, F., G. Caire, A. Poggi, G. Rimassa, " JADE: a white paper," EXP, 3(3), 6-19, 2003.