AGENTCITIES TASK FORCE

ESOA WG – Mission Statement

Agentcities Task Force Technical Note
actf-note-xxxxx, *06 February, 2003*

Authors (alphabetically):
Giovanna Di Marzo Serugendo, University of Geneva, Switzerland
Noria Foukia, University of Geneva, Switzerland
Cecilia Gomes, New University of Lisbon, Portugal
Beatriz Lopez, University of Girona, Spain
Anthony Karageorgos, UMIST, United Kingdom
Soraya Kouadrimostefaoui, University of Fribourg, Switzerland
Philippe Massonnet, CETIC, Belgium
Omer Rana, University of Cardiff, United Kingdom

Copyright © 2002 Agentcities Task Force (ACTF). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the ACTF or other organisations, except as needed for the purpose of developing Agentcities standards in which case the procedures for copyrights defined in the Agentcities Standards process must be followed, or as required to translate it into languages other than English. The limited permissions granted above are perpetual and will not be revoked by the ACTF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE AGENTCITIES TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Status

Draft

This version: http://www.agentcities.org/note/xxxxx/actf-note-xxxxxa.html
Latest version: http://www.agentcities.org/note/xxxxx/

Abstract

By self-organising applications, we mean essentially applications running without central control, and whose different behaviour of individual entities lead to an emergent coherent result. Biologically inspired applications, autonomous agents, applications running on peer-to-peer networks all exhibit self-organising behaviour. These applications belong to several diverse domains, but we will focus on applications related to Grid computing, Games and Simulations, Business Process Infrastructure, Network Management, Robots and the Agentcities Network. By engineering self-organising applications, we mean all the methodologies and techniques necessary to design, deploy, and maintain those applications.

52 This document describes the framework and the goals pursued by the Engineering Self-
53 Organising Applications (ESAO) WG within the Agentcities project.
54

© 2002 Agentcities Task Force

Contents

© 2002 Agentcities Task Force

1 Introduction

By self-organising applications, it is usually meant applications or systems that take inspiration from biology, physical world, chemistry, or social systems. Among others we can mention systems that reproduce socially-based insect behaviour, such as robots or artificial life. Characteristics of these applications are their ability to achieve complex collective tasks with simple individual behaviours, without central control or hierarchy.

If we consider now, two or more software entities that communicate with each other, but that have been designed independently, and that are supposed to interact without any central control, we notice that they present self-organising properties. Indeed, there is some global result emerging from their interaction; entities have only local knowledge but participate to some collective task; they are heterogeneous in their design and behaviour, but nevertheless work together; and they engage spontaneous local interactions.

Then, more generally, we can consider self-organising applications as being applications made of several pieces and acting without any central control, and whose different individual autonomous behaviour lead to an emergent coherent result. They include applications presenting a socially-based collective behaviour, but also multi-agent systems, pervasive or wireless applications, as well as those to be deployed on a Grid or on a peer-to-peer (P2P) network. With the advent of wireless portable device assistants (PDAs), the number of unanticipated entities wishing to communicate will increase drastically. They will form a highly dynamic environment, with running entities leaving and arriving continuously, making it impossible to rely on traditional engineering techniques.

Indeed, self-organising software raises engineering questions that techniques defined for traditional systems cannot answer:

- How can such an application not simply crash when part of the network is down or unreachable? Exception handling may be no longer sufficient for an autonomous component that needs to continue its execution despite failures;
- How can the different components interact and co-ordinate properly without run-time errors? Syntactical type checking needs to be assisted by a semantical description of behaviours and interactions;
- How to be sure that the application as a whole behaves correctly and delivers the desired (or a minimal) functionality in every situation? Design time model checking or testing need to be completed by run-time verification and guidance;
- What is the best way to design and program these applications? How can we perform the separation of concerns? How to engineer the applications such that a verification becomes possible?

Currently, these applications are developed in an ad hoc manner, leading to solutions that do not exploit entirely the decentralised aspect of the application, and consequently do not benefit from their natural robustness, and fault-tolerance.

Primary objectives of the Agentcities initiative include semantic interoperability between systems deployed in the network, and their ability to discover one another and to provide each other with services. Traditional Web services, but also envisioned complex systems that will run on the Agentcities network naturally present self-organising principles: autonomous components interact and produce emergent results; heterogeneously designed entities discover themselves at run-time; entities join and leave permanently teams or groups. Today these systems are engineered by adhering to common standard, and pre-defined communication schemas, which prevent them from being truly self-organised.

The goal of the Engineering Self-Organising Applications (ESOA) WG is to describe problems and challenges related to the design, deployment and maintenance of self-organising applications, and to bring pieces of solutions in the framework of the Agentcities project, but also more generally for next generation information networks.

© 2002 Agentcities Task Force

This document first briefly describes the applications and domains, presenting self-organising aspects, on which the WG will concentrate. It then presents two case studies: an emergency scenario that is likely to be implemented on the Agentcities network by the Social Systems WG; and a Web Service composition based on context-awareness. Both scenarios will serve as case studies throughout the documents produced by the ESOA WG. This document then reviews current techniques, but also emerging methodologies, for designing and implementing self-organising applications. Finally, it lists open issues related to the engineering of self-organising applications.

© 2002 Agentcities Task Force

2 Self-Organising Applications

Self-organisation principles or properties are present in applications ranging from agents, to decentralised systems, and to biologically inspired systems. This section mentions briefly applications and domains presenting self-organising aspects. A more complete survey of self-organising applications can be found in [ESOA02b].

2.1 Biologically Inspired Systems

Biologically Inspired Systems. As mentioned in many researches dealing with the biological diversity, living organisms are a good example of self-organised entities because they exhibit macroscopic order from microscopic disorder within themselves. One of the most studied cases of natural behaviour is the social insects colonies. Indeed, each colony acts/represents a complex system where individual/simple entities organise themselves to ensure the survival of the colony by means of a reactive individual behaviour and a co-operative collective one. Such behaviours can be observed from different activities: foraging, building nests, sorting larvas,...etc. Co-operation in these systems is mediated by an efficient communication mechanism relying on the inscription of task evolution in the environment. This indirect communication between the different members of the colony through the environment is called stigmergy and was introduced for the first time by Grassé in [Grassé59]. This paradigm has inspired many computer scientists across various research domains such as robotics [Kube98], network routing [Dorigo98], network load balancing [Fenet98], and optimisation algorithms [Mallon00].

In all these cases the global and complex collective behaviour, emerging from interactions between simple entities, is dominating.

2.2 Grid

Distributed Computing Initiatives. Peer-to-peer (P2P) computing applications run on sets of peer machines, i.e., with no central server. Examples of these applications include: files sharing, such as the one provided by Gnutella[1]; computing power sharing from SETI@home[2], a scientific experiment that uses Internet-connected computers in the Search for Extraterrestrial Intelligence (SETI); common storage such as provided by OceanStore[3] [Rhea01].

The common point of all these applications is their decentralised nature; the sharing by all, sometimes anonymous, users of the available computing resources (information, CPUs, and storage capacities); and the highly dynamic network topology, since nodes can join and leave continuously the network.

Grid Computing. Grids are persistent environments that enable software applications to integrate instruments, as well as computational and information resources managed by diverse organisations in widespread locations [Foster01].

The self-organising aspect of Grid applications lays in the fact that heterogeneous, independent software, hardware, databases are asked to be combined in order to perform some world-wide computation.

[1] http://www.gnutella.com/system
[2] http://setiathome.ssl.berkeley.edu
[3] http://oceanstore.cs.berkeley.edu

© 2002 Agentcities Task Force

2.3 Business Process Infrastructure

(to be completed)

Manufacturing Systems. *(to be completed)*

Electronic and Mobile Services (rather than web services).
Services are defined as software components, or building blocks that are provided in order to be assembled, and re-used in a distributed Internet-based environment [Pernici00], [Maamar]. Development of customised services by integration of existing ones referred to as service composition, has received a lot of attention in the last few years. Mainly the development of such services goes through three main steps: first services are matched against a user query (discovery) in order to select the relevant service offers, secondly a feasible order in whish these services can be executed is determines (generally using workflow models) and finally the composite service is executed.

The process of discovery, composition and execution of services can be viewed as a self-organising system, where. each service provider participates in the task of forming “a new composite service” (the emergent behaviour is the formation of composite services) without being aware of it.

2.4 Networks

Ad-Hoc networks. Technologies such as PDAs and Bluetooth are permitting a new form of *ad hoc* network, spontaneously built while people carrying PDAs are located in a close area. These networks permit in turn new forms of applications where people need to locate other people or information in their vicinity.

Ad-hoc networks are wireless, self-organising systems, formed by co-operating nodes within communication range of each other, that form temporary networks. Their topology is dynamic, decentralised, ever-changing and the nodes may move around, join and leave, arbitrarily.

Pervasive/Ubiquitous Computing. Communicating computer systems are now invading our everyday's life: phones, cars, domestic appliances, clothes, smart-cards, PDAs. Communication occurs within short or long distances, depending on the wireless or networked nature or the systems. Application domains are as diverse as: health monitoring, home networking, automobile network, mobile e-business, spontaneous networks, smart spaces, sensors nets.

The ubiquitous nature of applications running on such computers make them self-organised, since software running on heterogeneous devices engage interactions either spontaneously, or in a more disciplined manner in order to realise some expected service.

2.5 Games and Simulations

Robocup. *(to be completed)*
RoboCup is an international joint project to promote AI, Robotics and related fields. It attempts to foster research in AI and robotics by providing standard problems, i.e. a football competition (RoboCup Soccer) and a rescue task (RoboCup Rescue), where a wide range of technologies can be integrated and examined.

Disaster rescue is one of the most serious social issues which involves very large numbers of heterogeneous agents in the hostile environment. The intention of the RoboCupRescue project is to promote research and development in this socially significant domain at various levels involving multi-agent team work co-ordination, physical robotic agents for search and rescue, information infrastructures, personal digital assistants, a standard simulator and

© 2002 Agentcities Task Force

decision support systems, evaluation benchmarks for rescue strategies and robotic systems that are all integrated into a comprehensive systems in future. Its origin is drawn from the observation of the inefficiency of the current rescue efforts when a significant natural disaster occurs. This was illustrated at the time of the catastrophe of Kobe in Japan. The problems that are faced are very close to multi-agent systems in a strongly dynamic environment.

A generic urban disaster simulation environment is constructed on network computers. Heterogeneous intelligent agents such as fire fighters, commanders, victims, volunteers, etc. conduct search and rescue activities in this virtual disaster world. Real-world interfaces such as helicopter image synchronises the virtuality and the reality by sensing data. Mission-critical human interfaces such as PDA support disaster managers, disaster relief brigades, residents and volunteers to decide their action to minimise the disaster damage.

A RobocupRescue situation exhibits self-organising aspects, since there is no agent at the RobocupRescue scenario that knows what is happening everywhere. Central buildings will know information from the environment if they receive inputs from the corresponding team agents. Team agents can co-operate each other but with a local view. A message sent by an agent can only be perceived by agents 30m around. Each type of agents can perform actions upon a set. Agents can only receive and send four mails every simulation cycle. Central buildings can divert agents to specific zones of the map in order to make a more profitable operation.

Games. *(to be completed)*

2.6 Robots

Robotics is particularly a good domain for studying self-organisation. Robots can be viewed as agents enjoying the following characteristics [Kouadri02]:

- **Embodiment**: robots have bodies and experience the world directly. Their actions are part of a dynamic with the world and have immediate feedback on the robots.

- **Situatedness**: the robots are situated in the world, they do not deal with abstract descriptions, but with the ``*here*'' and ``*now*'' of the environment that directly influences the behaviour of the system.

- **Autonomy**: robots operate without the direct intervention of humans or others, and have control over their actions and internal states.
- **Social ability**: robots can communicate via some kind of robot-communication languages, and typically have the ability to engage in social activities in order to achieve their goals.

- **Mobility**: robots are mobile; this mobility provides a simple abstraction for a complex distributed system.

- **Adaptability**: robots can adapt their individual behaviors, or the collective behaviors to new situations, while in a continuous interaction with the environment.

2.7 AgentCities

Autonomous Agents. An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors, in order to satisfy an individual goal. Agents act autonomously, without the direct intervention of humans, or any central control, and are able to interact with other agents, or with humans, in order to complete their own problem solving [Weiss99,Jennings98].

Agents, and their learning capabilities, inherently vehicle the notion of self-organisation. Indeed, each agent autonomously takes decisions, on the basis of its local (incomplete)

knowledge, that will allow it to solve its goal; and the learning process smoothly influences the self-organising behaviour.

Agentcities Network. Agentcities is an initiative designed to construct a world-wide, open network of platforms hosting diverse agent based services. The aim is to enable the dynamic, and autonomous composition of services, creating compound services to address changing needs. The initiative is founded on technologies, such as agent technology, Semantic Web technologies, UDDI discovery services, eBusiness standards and Grid Computing.

Services and compound services that are expected to run on the Agentcities network have self-organising aspects such as: correct dynamic composition of independently designed services; or spontaneous interactions among services.

3 Case Studies

This section presents two case studies that will serve throughout the different papers produced by the working group. The emergency scenario describes how rescue scenario of the future. The particularity of this case study is the heterogeneity and the relative complexity of its components. The second case study concerns the composition of Web services based on context-awareness. This mid-term case study shows an example of self-organising behaviour of components of the same nature (Web services).

3.1 Emergency Scenario

Let's define a city as a cluster of organised housing, production and transportation facilities established within the atmospheric, aquatic, soil, fauna and flora environment. City related functions include:

1. Provide housing facilities for people and production media
2. Provide a network for water, energy and goods supply
3. Provide mobility infrastructures and services
4. Provide health, education and entertainment services
5. Threatening life
6. ....

An emergency scenario is defined as a situation where one of the facilities defining the city concepts and its functions is influenced. For example:

A. Harbour fire, related to mobility infrastructures
B. Gas explosion, related to network for water, energy and goods supply
C. Earthquakes, related to environmental parameters
D. Forest fire, related to environment, flora and fauna
E. Accident in an industrial plant, related to threatening life
F. Catastrophic scenarios regarding big-uncontrolled shows (soccer, pop-starts, ...), related to threatening life

In a disaster scenario several emergency services need to be deployed in order to restore a normal functioning. For example, fire brigades can stop fire in a gas explosion, and then the gas company restore the damaged pipe. In the meanwhile, police forces divert the traffic to alternative routes.

Agent technology can help in two dimensions in order to mitigate the disaster mainly by following two approaches:

1. Agents provide support to missing damaged functionality of services;
2. Agents provide training and guidelines for real emergencies.

(explain why it is considered a SOA, characteristics)

© 2002 Agentcities Task Force

3.2 Web Services Composition Scenario

(scenario: to be completed)

(explain why it is considered a SOA, characteristics)

4 Taxonomy

Links with definition of complexity theory *(to be completed)*

The characteristics of the systems described above are the following:

Heterogeneity. Independently designed and developed software systems, as well as software having different goals and communication means, need to interact with each other. Traditional techniques rely on the use of a common ontology and communication protocols to enable interoperability.

Decentralised Control. Due to their heterogeneous and dynamic nature, to their quantity, it is impossible to centrally or hierarchically control a set of running entities. Entities have no global knowledge, they have only a local incomplete knowledge, and their large number leads to local interactions among them. Nevertheless, a coherent result or series of results emerges from their interactions, i.e., the interaction of correct components leads to a correct global behaviour at run-time.

Changing Topology. Components join and leave permanently a group, a team, or a network. The choice of the group to join is even not foreseen at design time.

Pervasive. Computer systems are embedded in every object, and the network connectivity is pervasive, both for traditional and ad-hoc networks.

Run-time co-ordination. Entities have to co-ordinate correctly, i.e., to understand each other at run-time, to interoperate, to discover resources and functionality, and to engage collaborations and negotiations on the basis of expected behaviour of peers.

Adaptable. There is a need for adaptation to a changing environment, i.e., to overcome network problems, even in case of unreachable network zones, low bandwidth, or insufficient memory space; and to offer acceptable quality of service, even with low-level peers. Some computing systems need to run anytime, despite the needs for evolution and maintenance.

The ESOA WG will focus on the following application domains: Grid computing (comprising P2P networks), Networks (traditional and ad-hoc networks), Games and Simulations, Business Process Infrastructure, and Agentcities. The emergency scenario falls into both the Games and Simulations domain, and the Agentcities domain.

Table 1 lists the main self-organising characteristics, classified by domains, for the applications mentioned in Section 2.

	Grid	BPI	Networks	Games/Sim	Robots	Agentcities
Heterogeneity	x					x
Decentralised	x	x	x	x	x	x
Changing Top.			x			
Pervasive			x			
Run-time coor.	x			x		
Adaptable			x	x		

Table 1: Application Domains and Main Self-Organising Characteristics

© 2002 Agentcities Task Force

We have explained above the characteristics of SOAs, we will now precise what we do *not* consider a SOA. A system made of several entities, but all designed by the same team, such as a traditional client/server system; entities designed by different independent teams, but whose interfaces and communication protocol are known in advance; or a system which is centrally or hierarchically controlled.

5 Engineering of SOAs

A more detailed review of the techniques currently used for engineering self-organising applications is given in [ESOA02a].

5.1 Current Techniques

Current techniques address some of the characteristics mentioned above in isolation, usually with low-level concerns (API descriptions), relying on standards, and on agreed, pre-defined communication elements. However, there is currently no means, or software engineering technique, of developing software entities and to be sure that they will be able to interact with other entities and that the outcome of the interaction is an accepted behaviour.
(Link with document on issues?)

Standards. List of standards *(to be completed)*
For instance, component-oriented programming requires standards to allow independently created components to interoperate, and specifications enabling the composer to decide what can be composed under which conditions.

Agreed communication elements. List … *(to be completed)*
For instance, Web services development requires the programmers to search Web services repositories for retrieving the specification (interface, protocol, conversation), of the services they want to communicate with. Specifications are expressed using a common XML format.
Autonomous agents communicate by adhering to some common communication language, or ontology.

Agreed interfaces and standards offer a good basis for interoperability, since they ensure syntactical compatibility, and communication protocol adherence. However, requests for Web services cannot be expressed on the basis of the caller's needs. They are driven by what is offered by potential furnishers. Similarly, agents requests must conform to standard primitives, thus tuning of particular requirements may be impossible. In addition, once these entities enter in some interaction, there is no means to ensure the run-time correctness of the interaction. A Web service, invoking another Web service, assumes that the partner's behaviour is compatible with the published specification, and that the issue of their common interaction will not leave one of them in a run-time error state. This problem is even more acute for a Web service satisfying a request: it has no idea of the partner's behaviour, it is committed to satisfy the request, which is syntactically correct, but it has no guarantee on the issue of the interaction. By adhering to ontologies, agents also perform some assumptions on the partner's behaviour - relying on the underlying ontology semantics - and no guarantees are given regarding the interaction issues.

A Rescue Scenario is in essence a self-organising application. Several rescue services/agents act in a same scenario with the same goal: rescue people. However, each agent has a set of abilities, namely, ambulances are able to transport victims but are not able to extinguish fires. No co-ordination strategy has been planned in advance, and if so (for example, emergency plans in chemical factories), the critical circumstances make plans planned in advance fail very often. Moreover, services can be up and down, depending on their evolution in the rescue scenario (a fire brigade can be stuck, for example, between two blocked roads). Services should co-operated in order to get a coherent emergent result.
See [SCER02] for further details on the Rescue Scenario.

© 2002 Agentcities Task Force

5.2 Emerging Trends

Technologies, which take their inspiration from biology, or social systems, as well as software engineering solutions for developing those systems, are emerging.

Biologically/Socially based systems. *Pervasive Intelligence* views and designs multi-agent systems as ``ecosystems of physical agents'', organised after biological, physical, or chemical principles [Servat02].

Amorphous Computing envisions programming models of collective behaviour [Abelson00], where local behaviour is obtained using primitives derived from morphogenesis and developmental biology. Amorphous computing was driven by software engineering concerns to obtain coherent behaviour from non trustable computing devices interconnected in local, unknown, irregular and time-varying ways.

New Software Engineering Techniques. The behaviour of large software systems is being recognised as sharing similarities with human or biological organisations, where the global behaviour derives from the local behaviour of individual entities. The need for new software engineering techniques for modelling, testing and maintaining these software under an intentional and biological perspectives are advocated [Zambonelli02].

In order to address engineering requirements for complex systems, architecturally independent methodologies that enable to build robust and scalable systems are necessary. ROADMAP, an extension of the agent-oriented software engineering Gaia methodology, has been proposed [Juan92]. It enriches the Gaia methodology with additional models capturing the environment, the organisational structure of the system, the computational organisation of the interacting roles, and the social goals.

Dynamic methodologies, such as the Karma-Teamcore framework, enable rapid and robust team formation and integration of distributed, heterogeneous agents [Tambe00]. The system designer specifies the hierarchy and the high-level goals of the agent organisation. The KARMA assistant derives requirements for roles and searches for agents matching the required criteria. Teamcore wraps domain agents, and automatically enforces co-ordination synchronisation actions. The wrappers actually execute the team-oriented program.

Information Exchange. Ontologies, Entish ….

5.3 Case Studies

Emergency Scenario.
Every service/agent has been designed independently, according to different human and organisational aspects. There are so many kind of organisations that can take part in a rescue scenario that it is difficult to establish a common framework to built a methodology. The methodology should be in line with agent technology. But, what should be the communication protocol? What about the interaction protocol? Which is the ontology? The two first questions can be solved by standards as FIPA. However, standards are designed to be used in a normal functioning. In the rescue scenario, however, communication is a critical issue. The volume of messages can be huge and the communication channels can be damaged.

Regarding ontology: at which level should it be defined? Local? National ? International? Assuming that the disaster has been in a city, it is not enough to assume also that the services provided will come from the city (local). The scale of the overall system is then also a matter of study. How is it possible to define a methodology for so many kind of services/agents?

© 2002 Agentcities Task Force

Web Services Composition. (to be completed)
In the case of the web services composition, some solutions begin to appear …

6 Software Engineering Issues

Both agents and distributed systems communities feel that new, best-adapted, software engineering principles, are necessary for designing, developing and maintaining systems presenting self-organising properties.

From one hand, biologically inspired solutions emerge. On the other hand, there are calls for brand-new software engineering practices. However, the engineering of self-organising applications is at its infancy. No recommendations or best practices have yet been established. There are still open issues that need to be filled in the next years by agents and software engineering community, especially issues related to the key properties expected by self-organising applications: robustness, dependability, autonomy of design, and of behaviour.

Design and Models. It is impossible to describe the behaviour of a self-organising system in terms of its components; impossible to evaluate the effect of local interaction at the global level (chaos), difficult to predict the global dynamic behaviour of the model. Indeed, boundaries of software systems become fuzzy, it becomes hard to understand and control the overall behaviour, especially in the presence of heterogeneous entities interacting in unknown, irregular and time-varying ways.

Verification and Validation. It is impossible to verify, or to test the global behaviour of millions of heterogeneous interacting entities constantly joining and leaving a system .

Evolution. It will become necessary to let systems evolve (e.g., maintenance) without stopping them.

ESOA WG Goals. The Engineering Self-Organising Applications (ESOA) WG intends first to clarify the links that exist between complex open systems, self-organisation, and multi-agent systems. Second, the ESOA WG advocates the need for new methodologies, and tools for designing, implementing, and maintaining large self-organising systems. Third, it aims at bringing pieces of solutions in the framework of the Agentcities network testbed.

7 Conclusion

Our feeling is that a modern system must be considered as a large organisation, with its global behaviour being an emergent property resulting from the local interactions of the entities or agents that form the system.

Traditional techniques for modelling and designing are not suited with the dynamicity, and openness encountered by the local entities forming the system. Indeed, local entities must be given the means to overcome problems encountered during their execution life, which occurs in a global dynamic environment. They must be engineered such that they have the means to survive, i.e, to overcome failures and unexpected behaviour or partners, to adapt to changing topologies and partners, to discover each other, and to co-ordinate properly. In addition, the system as a whole must behave properly, even though the complexity of the system make it impossible to centrally control the whole system. Through local interactions and local co-ordination only, the global system has to exhibit a correct macro-level behaviour.

© 2002 Agentcities Task Force

Therefore, new software engineering practices considering such systems as self-organised, or as social decentralised organisations, and focusing on robustness, scalability, biologically inspired principles, have to be considered.

8 References

[Abelson00] *Amorphous computing*. H. Abelson, D. Allen, D. Coore, C. Hanson, G. Homsy, T. Knight, R. Nagpal, E. Rauch, G. Sussman, and G. Weiss. Communications of the ACM, 43(5):74--82, May, 2000.

[Christensen01] *Web Services Description Language (WSDL) 1.1*. E. Christensen, G. Curbera, F. Meredith, and S. Weerawarana. W3C Note, 2001.

[Dorigo98] *Ants colonies for adaptive routing in packet-switched communication networks.* M. Dorigo and G. Di Caro. Lecture Notes in Computer Science, page 673, 1998.

[Fenet98] *Ant Based System for Dynamic Multiple Criteria Balancing.* S. Fenet and S. Hassas. Ants'98, Brussels, 1998.

[Foster01] *The Anatomy of the Grid - Enabling Scalable Virtual Organizations.* I. Foster, C. Kesselman, and S. International Journal of Supercomputer Applications, 15(3), 2001.

[Grassé59] *La reconstruction du nid et les interactions inter-individuelles chez les bellicoitermes natalenis et cubitermes, la théorie de la stigmergie - essai d'interprétation des termites constructeurs*. P.P. Grassé. Insectes Sociaux, no. 6, pages 41-81, 1959.

[Jennings98] *A Roadmap of Agent Research and Development*. N. Jennings, K. Sycara, and M. Wooldridge. Autonomous Agents and Multi-Agent Systems, 1(1):7--38, 1998.

[Juan92] *ROADMAP: Extending the Gaia Methodology for Complex Open Systems.* T. Juan, A. Pearce, L. Sterling. In C. Castelfranchi and W. Lewis Johnson, editors, Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems, AAMAS'02, pages 3-10. ACM Press, 2002.

[ESOA02a] *Issues and Challenges in current Technology for Engineering SOAs*. ACTF-Technical report, AgentCities ESOA WG, 2002.

[ESOA02b] *A survey of self-organising applications*. ACTF-Technical report, AgentCities ESOA WG, 2002.

[Kouadri02] *Collective Adaptation of a Heterogeneous Communicating Multi-Robot System*. Soraya Kouadri Mostéfaoui and Michèle Courant. In the Proceedings of the International Arab Conference on Information Technology, ACIT'2002, P. 1038-1044, University of Qatar, Doha-Qatar December 16th - 19th, 2002.

[Kouadri03] *CB-SeC a Context-Based Service Discovery and Composition Framework for Pervasive Environments*. Soraya Kouadri Mostéfaoui. Internal working Paper, University of Fribourg, Switzerland.

[Kube98] *Cooperative transport by ants and robots.* C. Ronald Kube and E. Bonabeau. Robotics and Autonomous Systems, 1998.

© 2002 Agentcities Task Force

[Maamar] *Towards a Composition Framework for E-M Services*. Z. Maamar, B. Benatallah, and Q. Z. Sheng. In the Proceedings of the First Workshop on Ubiquitous and Embedded wearable and mobile devices.

[Mallon00] *Ants estimate area using buffon's needle.* E.B. Mallon and N.R. Franks. Proceedings of the Royal Society, London, April, 2000.

[Pernici00] *Designing components for e-services*. B. Pernici and M. Mecella. In proceedings of the VLDB Workshop on Technologies for E-Services, Cairo, Egypt (2000).

[Rhea01] *Maintenance-Free Global Data Storage*. S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz. IEEE Internet Computing, 5 (5) 40-49, September-October, 2001.

[SCER02] *Position paper: Main challenges in Agent Technologies to Service Coordination for Emergency Response Applications*. Agentcities Rescue WG, 2002

[Servat02] *Combining amorphous computing and reactive agent-based systems: a paradigm for pervasive intelligence?* D. Servat, A. Drogoul. In C. Castelfranchi and W. Lewis Johnson, editors, Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems, AAMAS'02, pages 441-447. ACM Press, 2002.

[Tambe00] *Building Dynamic Agent Organizations in Cyberspace.* M. Tambe, D. V. Pynadath, N. Chauvat. IEEE Internet Computing, 4 (2) 65-73, March-April, 2000.

[Weiss99] *Multiagent Systems - A Modern Approach to Distributed Artificial Intelligence*. G. Weiss, editor. The MIT Press, Cambridge, Massachussetts, 1999.

[Zambonelli02] *From Design to Intention: Signs of a Revolution*. F. Zambonelli and H. Van Dyke Parunak. In C. Castelfranchi and W. Lewis Johnson, editors, Proceedings of the First International Joint Conference on Autonomous Agents and MultiAgent Systems, AAMAS'02, pages 455--456. ACM Press, 2002.

© 2002 Agentcities Task Force

Change Log

8.1 Version a: 06.02.03

Page 1: Initial version

© 2002 Agentcities Task Force