AUTONOMOUS SYSTEMS WITH EMERGENT BEHAVIOUR

Giovanna Di Marzo Serugendo
University of Geneva

ABSTRACT

This chapter presents the notion of autonomous engineered systems working without central control, through self-organisation and emergent behaviour. It argues that future large scale applications from domains as diverse as networking systems, manufacturing control, or e-government services will benefit from being based on such systems. The goal of this chapter is to highlight engineering issues related to such systems, and to discuss some potential applications.

INTRODUCTION

Devices from personal computers, to handhelds, to printers, to embedded devices are very widely available. Further, today's wireless network infrastructures make it possible for devices to spontaneously interact. In addition, large-scale communication, information and computation infrastructures, such as networks, or grids are increasingly being built using numerous heterogeneous and distributed elements, which practically cannot be placed under direct centralised control. These elements exhibit certain degrees of autonomy and of self-organisation, such as taking individual decisions and initiatives, interacting which each other locally and giving rise to an emergent global behaviour.

This chapter introduces first the notion of autonomous systems; second it reviews the notions of decentralised control, self-organisation and emergent behaviour, and discusses how they relate to each other. Third, this chapter discusses different issues pertaining to the design and development of autonomous systems with emergent behaviour. Fourth, it reviews techniques currently being established for building those systems. Finally, it provides several examples of applications.

AUTONOMOUS SYSTEMS

We distinguish different classes of autonomous systems. First, autonomous systems as *distributed embedded devices* consist of physical devices having some onboard intelligence and standalone and communication capabilities. Such devices comprise intelligent mobile robots, but also intelligent wearable computing, surveillance, or production devices. Second, from a software point of view, *autonomous agents and multi-agent systems* are a notion first established by the Distributed Artificial Intelligence community. Such systems do not have to cope with the same problems faced with devices situated in a physical environment, e.g. low-battery. However, agents provide a metaphor for software design which incorporates most of the elements present in embedded devices such as autonomous decision-taking processes, communication with other agents, social interactions for collaboration, negotiation, transactions or competition purposes (Wooldridge, 2003). Third, more recently an initial focus has been given from the research community on autonomous software entities interacting with each other in a decentralised self-organised way in order to realise a dedicated high-level functionality

(interactions for collaboration purposes), or giving rise to an emergent global behaviour as a side-effect of their local interactions (interactions for competition purposes). This category of applications or entities is referred to as *self-organising systems* or *systems with emergent behaviour* (Di Marzo Serugendo, 2004). In some sense, this last category combines the first two views where autonomous software populates autonomous devices. Fundamental points of these different views of autonomous systems are: the social interactions arising among the different elements; and the need for adaptation to unforeseen (at design time) situations encountered into dynamic environments.

There is currently a growing interest in autonomous applications able to self-manage, not only from academic research but also from the industry. *Ambient intelligence* envisions seamless delivery of services and applications, based on ubiquitous computing and communication. Invisible intelligent technology will be made available in clothes, walls, or cars; and people can freely use it for virtual shopping, social learning, micro-payment using e-purses, electronic visas, or traffic guidance system (Ducatel, 2001). Ambient intelligence requires low-cost and low-power designs for computation running in embedded devices or chips, as well as self-testing and self-organising software components for robustness and dependability. Based on the human nervous system metaphor, IBM's *Autonomic Computing* initiative considers systems that manage themselves transparently with respect to the applications. Such systems will be able to self-configure, self-optimize, self-repair, and protect themselves against malicious attacks (Kephart, 2003). Recent interest by Microsoft, as part of the *Dynamic Systems Initiative*, indicates as well the importance of self-organisation for managing distributed resources.

Autonomous Computation Entities vs Autonomous Systems
As follows from the discussion above, autonomous systems are composed of one or, more generally, of several autonomous computation entities interacting together. These autonomous computation entities are either embedded into physical, possibly mobile, devices (e.g., in ambient intelligence applications) or part of a given environment that supports their execution and interactions (e.g., multi-agent systems).

DECENTRALISED CONTROL, SELF-ORGANISATION AND EMERGENT BEHAVIOUR

Decentralised control is intimately linked with the notion of emergent phenomena, since some result is expected from a system even if it works with decentralised control. Self-organisation may occur with or without central control; it is related to whether or not the system takes itself the measures to cope with the environmental changes. Even though artificial systems will certainly have at the same time these three characteristics: decentralised control, self-organisation and emergent phenomena, it is important to distinguish each of them. The purpose of this section is to briefly clarify these concepts and to establish the links and differences among these three notions.

Decentralised Control

It is important to distinguish between two kinds of artificial systems working with decentralised control: a) systems, built as a large set of autonomous components,

pertaining to the same system and providing as a whole expected properties, or functions. Otherwise stated, we want to build an application with a well specified functionality, but for complexity reasons, this application is decentralised and made of a large number of autonomous components; b) systems, composed of a large set of autonomous components, spontaneously interacting with each other, for possibly independent or competing reasons. In both cases, autonomous components may be heterogeneous and dynamically joining and leaving the system.

Even though in both cases, most of the issues and discussions are similar, the fundamental difference lies in the engineering process that is behind the building of the system. In the first case, the whole system is designed with emergent functionality in mind. Simply stated, a given collaborative team develops the application ensuring that the expected functionality will emerge. In the second case, the different components are produced by different teams, with different purposes in mind, each being concerned by the fact that their component can interoperate with the others. There is no expected global function or properties emerging, even though emergent phenomena will arise in any case from the different local interactions, and have a causal effect on the whole system, i.e., on the particular behaviour of the individual components.

In the first case, the core idea behind building large-scale systems is to have them composed of autonomous individual components working without central control, but still producing as a whole the desired function. Indeed, decentralised control allows: 1. computation and decisions to be distributed among the different components, thus preventing the need for a central powerful computer; 2. the system is more robust since it does not rely on a single node that may fail and crash the whole system; 3. network and CPU resources are better used in the sense that communication does not occur among a dedicated central node and a large number of components, but locally among the whole set of components; 4. in dynamic systems, where components join and leave the system permanently, decentralised control allows a flexible schema for communication, e.g. with a neighbour instead of with the central entity.

Self-Organisation

There are different definitions of self-organisation as observed in the natural world. We will focus here on three of them, essentially to enhance the fact that there are different kinds of self-organisations, and that designers of such systems must be aware of which kind of self-organisation they are considering when building their system (Di Marzo Serugendo, 2006).

Stigmergy

The theory of stigmergy, defined by Grassé (1959) in the field of swarms, or social insects' behaviour states that: coordination and regulation are realised without central control, by indirect communication of the insects through their environment.

Self-organisation results from the behaviour (of the insects) arising from inside the system. Otherwise stated, swarms' autonomous components are themselves at the origin of the re-organisation of the whole system.

Decrease of Entropy
In the field of thermodynamics, Prigogine and his colleagues have established that open systems decrease their entropy (disorder) when an external pressure is applied (Glansdorff, 1971).

Self-organisation, in this case, is the result of a pressure applied from the outside. It is interesting to compare self-organisation in this case with the swarms' behaviour, where the "initiative" of self-organisation occurs from within the system.

Autopoiesis
Through biological studies, Varela (1979) established the notion of autopoiesis as the self-maintenance of a system through self-generation of the system's components, as for instance cells reproduction.

Self-organisation here is still different from the two other examples above. Indeed, autopoiesis applies to closed systems made of autonomous components whose interactions self-maintain the system through generation of system's components.

Even though differently stated in the few definitions above, and with a different impact on the way and the reasons why self-organisation is produced, we can consider that self-organisation is essentially:

*The capacity to spontaneously produce a **new organisation** in case of environmental changes **without external control**.*

Indeed, in the case of social insects, environmental changes will cause ants or termites to find new paths of food, i.e. change their behaviour in order to still be able to feed the colony. In the case of thermodynamics, external pressure changes will cause gas particles to be more or less excited, change their temperature, etc; thus, reaching a new stable state. Finally, cells or living organisms regenerate the whole system in order to overcome cells' death, and to survive in their given environment.

It is interesting to note that new organisation of a system may occur with or without central control provided it is not external.

Emergent Behaviour

Literature on "emergence" is abundant and varied ranging from philosophical discussions to operational descriptions. One of the most popular definitions of emergence which captures the essence of the emergent phenomena comes from Holland (1998), who states that:

"The whole is more than the sum of the parts".

In systems composed of individual autonomous computation entities, we will consider that an emergent phenomenon is essentially (Di Marzo Serugendo, 2006):

A structure (pattern, property or function), ***not explicitly represented*** *at the level of the individual components (lower level), and which* ***appears*** *at the level of the system (higher level).*

An additional important point here is that emergent phenomena have a meaning for an *observer* external to the system but not for the system itself. We distinguish two kinds of emergent phenomena:

– Observed patterns or functions which have *no causal effect* on the system itself. If we consider stones ordered by sea, with time a kind of classification of the stones occur. Small, lighter stones are close to the border, while heavy stones are far from it. In this case, this ordering of the stones has no effect at all on the whole system made of the stones and the sea (Castelfranchi, 2001);

– Observed functions which have a *causal effect* on the system. Such functions can be desired or not, but in both cases they have an effect on the system behaviour, and will cause the individual parts to modify their own behaviour.

Artificial systems are composed of a large number of individual components, i.e., of autonomous computation entities. During the course of time a large number of interactions occur, among these components, whose ordering, content and purpose are not necessarily imposed. It becomes then difficult to predict the exact behaviour of the system taken as a whole because of the large number of possible non-deterministic ways the system can behave. However, since we have built the system, the individual behaviour's components and the local rules governing the system are known, it becomes then "in principle" possible to determine the (emergent) system's behaviour. In practice, current techniques or calculations (essentially simulations) are not sufficient and make it almost impossible to determine the result. That is why the result, functions or properties, is said to be "emergent".

When Self-Organisation Meets Emergence

Due to the fact that in most systems, self-organisation and emergent phenomena are observed simultaneously, there is a natural tendency to consider that self-organisation leads to emergent phenomena, or that they are intimately linked. As also pointed out by De Wolf (2005), even though not totally wrong, this assumption needs to be clarified.

Self-organisation without Emergent Phenomenon

Self-organisation happens without observed emergent phenomenon, essentially when the system works under central control. Indeed, self-organisation is the capacity of the system to find a new organisation in order to respond to environmental changes. The new organisation can be identified under internal central control, and thus the possibly observed new organisation is fully deducible from the central entity.

Emergent Phenomenon without Self-Organisation
Emergent patterns, such as zebra stripes, have no causal effect on the whole system. There is no re-organisation of the stripes or of the cells producing the stripes. Stones ordered by sea do not undertake a self-organisation when they are ordered by the sea.

Self-Organisation together with Emergent Phenomenon
We consider that in order to have self-organisation and emergent phenomenon at the same time, the considered system should have the following characteristics:

- *dynamic self-organising system*: individual components are "active"; they may have their own objective and carry out their respective tasks.
- the system works with *decentralised control*;
- *local interactions* occur among the individual components.

Natural or artificial "interesting" systems usually considered by scientists are those of the last category, where we usually have: decentralised control realised under self-organisation, and leading to emergent behaviour.

ISSUES

This section distinguishes five issues related to systems made of autonomous software entities and exhibiting an emergent behaviour.

Interactions among Unknown Autonomous Computation Entities

Autonomous software entities interact with their environment and with other generally unknown software entities. Interaction covers both semantic understanding of the functional and non-functional aspects of a peer entity, and interoperability, which encompasses transactions, service delivery, and exchange of information.

Management of Uncertainty

For autonomous entities situated in a dynamic and insecure environment, uncertainty relates to reliability and trustworthiness of both the environment and interacting peer entities. For instance, an autonomous software entity cannot expect to fully rely on the permanent availability of network accesses, capacity and loads. In addition, a malicious entity can exhibit desirable characteristics, while it has no willingness to realize them; or, even in good faith, an entity can fail to deliver a service because the conditions required for its correct functioning are no longer provided by the environment (software errors, or physical failures).

Adaptability to Changing Environment and Changing User Requirements

Autonomous software considered in this chapter are situated in a physical environment mostly composed of wireless devices, for which: availability of network access is not fully granted; availability of interacting entities is not permanently granted: devices can freely join or leave an interacting zone, or partners; and reduced consumption power conditions may prevent autonomous software residing in wireless devices to perform

their computation at their maximum capacity. In addition to changing environment, autonomous software has to adapt its behaviour to changing user requirement or under the evolution of business practices. For instance, a personal assistant may change the user's agenda, if the user signals some priority activity.

Design and Development

On the one hand, emergent behaviour, as observed in nature or among societies, has fundamental properties such as robustness, behaviour adaptability, learning through experiences, complex global behaviour arising from simple individual behaviour, which software engineers would like to benefit when building complex and large scale systems. On the other hand, because of these properties, such systems are difficult to design correctly and their behaviour, once deployed in a physical environment, difficult or impossible to predict. This is mostly due to the non-linear nature of the interactions occurring among the different autonomous computation entities forming the autonomous systems, i.e. the behaviour of the system as a whole is not a linear function of the behaviour of the individual autonomous computation entities. At the research level; we are currently witnessing the birth of a brand new software engineering field specifically dedicated to emergent behaviour. One of the most delicate points is to ensure that "good" (i.e. expected) properties will actually emerge, while bad (i.e., not expected or not desired) properties will not.

Control of Emergent Behaviour

At run-time, control of emergent behaviour is an important issue related to artificial self-organising systems with emergent behaviour. Indeed, those systems usually demonstrate adaptability capabilities to changing environmental conditions (due to the ability of the system to re-organise) coupled with emergent phenomena, which by definition is difficult to predict. From an engineering point of view, it becomes crucial to have means, at run-time once the system is deployed and executing in its environment, allowing the control of such systems, such as changing the system's global goal, stopping the system if necessary, etc. Solutions for this issue most likely have to be considered at design time already, by for instance incorporating specific features that will be useful for control.

ENGINEERING EMERGENT BEHAVIOUR

This section describes existing design and development techniques (bio-inspired or not), and tools for building autonomous systems with emergent behaviour. Bio-inspired techniques usually rely on stigmergy (Bernon, 2006), but we observe other approaches based on capacity fields, or on trust-based human behaviour (Hassas, 2006). These section reviews: interaction mechanisms among individual autonomous computation entities, middleware computing infrastructure supporting their computations, methodologies and CASE tools for design and development, and formal methods related to self-organisation and emergent behaviour.

Interaction Mechanisms

When building a self-organising system, or a system with decentralised control, at the lowest level, we need to define first the local interactions among the different individual components of the system.

Swarm Intelligence
Swarms, or the stigmergy paradigm, provide a great source of inspiration, especially for fixed and mobile networks systems management such as routing, load balancing or network security. Ants' behaviour has been extensively reproduced in artificial system through artificial pheromones coordinating the work of mobile robots, or mobile agents. More recently, other swarms behaviour is being considered as well, as for instance spiders (Bourjot, 2003) and bees-like (Fabrega, 2005).

Biology – Cells
Besides swarm behaviour, another category of natural mechanisms reproduced in artificial systems concerns mammalian immune systems which have mostly used for network intrusion detection (Hofmeyr, 2000).

Human Behaviour/Trust
Trust-based systems or reputation systems take their inspiration from human behaviour. Indeed, uncertainty and partial knowledge are a key characteristic of the natural world. Despite this uncertainty human beings make choices, take decisions, learn by experience, and adapt their behaviour. As mentioned above, uncertainty is an issue when building decentralised open systems.

Most artificial trust-based management systems combine higher-order logic with a proof brought by a requester that is checked at run-time. Those systems are essentially based on delegation, and serve to authenticate and give access control to a requester (Weeks, 2001). Usually the requester brings the proof that a trusted third entity asserts that it is trustable or it can be granted access. Those techniques have been designed for static systems, where an untrusted client performs some access control request to some trusted server. Similar systems for open distributed and decentralised environment have also been realised: the PolicyMaker system is a decentralised trust management system (Blaze, 1996) based on proof checking of credentials allowing entities to locally decide whether or not to accept credentials (without relying to a centralised certifying authority). Eigentrust (Kamvar, 2003) is a trust calculation algorithm that allows calculating a global emergent reputation from locally maintained trust values. Recently, more dynamic and adaptive schemas have been defined, which allow trust to evolve with time as a result of observation, and allows to adapt the behaviour of entities consequently (Cahill, 2003).

Artificial Mechanisms
In addition to the digital pheromone, which is the artificial counterpart of the natural pheromone used by the ants, new electronic mechanisms directly adapted to software applications are being developed. The notion of tags, a mechanism from simulation models, is one of them. Tags are markings attached to each entity composing the self-organising application (Hales, 2003). These markings comprise certain information on the entity, for example functionality and behaviour, and are observed by the other entities. In this case the interaction occurs on the basis of the observed tag. This is useful

if applied to interacting electronic mobile devices that do not know each other in advance. Whenever they enter the same space, for example a space where they can detect each other and observe the tags, they can decide on whether they can or cannot interact.

Smart tagging systems are already being deployed for carrying or disseminating data in the fields of healthcare, environment, and user's entertainment. For instance, in the framework of data dissemination among fixed nodes (Beaufour, 2002) propose a delivery mechanism, based on the local exchange of data through smart tags carried by mobile users.

Mobile users or mobile devices do not directly exchange smart tags; they only disseminate data to fixed nodes when they are physically close to each other. Data information vehicled, by smart tags, is expressed as triples indicating the node being the source of the information, the information value, and a time indication corresponding to the information generation. Smart tags maintain, store, and update these information for all visited nodes. A Bluetooth implementation of these Smart Tags has been realised in the framework of a vending machine (Beaufour, 2002).

In smart tagging systems, data remain structurally simple, and understandable by human beings, and does not actually serve as a basis for autonomous local decisions.

Middleware Computing Infrastructures

Besides local interaction mechanisms favouring communication and cooperation among the individual components, for artificial system we may need computing infrastructures, also called middleware, supporting the chosen mechanisms, and acting as the artificial environment for the system's component. For instance, such middleware supports the evaporation of the artificial pheromone, or allows mobile agents to perform their execution or to move from one host to another one.

These infrastructures are usually *coordination spaces* providing uncoupled interaction mechanisms among autonomous entities, which asynchronously input data into a shared tuple space, and may retrieve data provided by other entities.

The TOTA environment (Tuples On The Air) propagates tuples, according to a propagation rule, expressing the scope of propagation, and possible content change (Mamei, 2003). Such a model allows, among others, to electronically capture the notion of digital pheromone, deposited in the tuple space and retrieved by other agents. The propagation rule removes the pheromone from the data space, once the evaporation time has elapsed.

Alternatively, the Co-Fields (coordination fields) model drives agents' behaviour as would do abstract force fields (Mamei, 2002). The environment is represented by fields, which vehicle coordination information. Agents and their environment create and spread such fields in the environment. A field is a data structure composed of a value (magnitude of field), and a propagation rule. An agent then moves by following the coordination field, which is the combination of all fields perceived by the agent. The environment updates the field according to the moves of the agents. These moves modify the fields

which in turn modify the agent's behaviour. This model allows representing not only complex movements of ants, and birds, but also tasks division and succession.

Anthill is a framework for P2P systems development based on agents, evolutionary programming, and derived from the ant colony metaphor. An Anthill distributed system is composed of several interconnected nests (a peer entity). Communication among nests is assured by ants, i.e., mobile agents travel among nests to satisfy requests. Ants observe their environment, and are able to perform simple computations (Babaoglu, 2002).

Methodologies and CASE Tools

Finally, at the highest level, from the designer point of view, it is crucial to rely on a development methodology and tools supporting the different phases of development of systems with emergent behaviour. Research in this field is at its infancy, and very few results are available.

The Adelfe (Bernon, 2002) methodology supports designers in taking decision when developing a multi-agent system exhibiting emergent phenomena, and in helping developers in the design of the multi-agent system. It is based on the AMAS (Adaptive Multi-Agent Systems) theory where self-organisation is achieved through cooperation among the agents, i.e. agents avoid non-cooperative situations.

Ongoing research in this field seem to favour solutions combining formal or traditional models with simulations in order to be able: on the one hand, to formally define the system, its expected properties, and the behaviour of the individual components; and on the other hand (through simulation of these models) to be able to validate or invalidate the design, and to predict some emergent phenomena.

Models and Formal Specifications

Mathematical equations, cellular automaton and neural networks have since long been used to understand complex systems, emergent patterns and the human neuronal activity. More recently, since autonomous software agents naturally play the role of individual autonomous computation entities, multi-agent systems are also being used to model complex systems, and to derive, through simulation, results about emergent phenomena, adaptability characteristics, starting conditions, parameters, etc. The purposes of multi-agent based models are of two different natures. From the one hand, the related simulations serve as experiments to better understand a complex system, or to (in)validate a given theory; a purpose similar to that pursued with cellular automaton and neural networks models. From the other hand, for artificial systems essentially, agent-based simulations help predict the run-time behaviour of a given system, tune the different parameters, etc. Multi-agent systems are particularly interesting when considering artificial systems with emergent behaviour, since those systems involve mobility, social interactions (negotiation, competition, and collaboration), and a high-number of entities interacting in a networked environment. The combination of all these features can be hardly modelled through mathematical models, cellular automaton or neural networks. For the same reasons, a third purpose of the use of multi-agent systems is currently being investigated, and it consists in actually building artificial self-organising applications with autonomous software agents.

In addition to models and simulations, formal reasoning about adaptability characteristics and emergent properties is a research area under consideration in the field of engineering of systems with emergent behaviour. As is the case for software engineering related to "traditional" software, formal specifications allow deriving formal models, on which reasoning of different kinds can be performed in order to provide design-time results about the system seen as a whole. We can distinguish different works depending on the use and purpose of the formal specifications. From a very abstract perspective, category theory has proved useful for reasoning about emergent properties arising among interacting components, those properties being expressed through an underlying logic (Fiadeiro, 1996). From a more concrete point of view, recent work has shown interest in emergent properties related to multi-agent systems (Zhu, 2005). In addition to the use of formal specifications and reasoning at design-time, we can mention as well the use of formal specifications at run-time and its expected benefits for both designing and controlling (at run-time) emergent behaviour (Di Marzo Serugendo, 2005).

APPLICATIONS

This section presents several application domains, current and undergoing realisations, as well as some visionary applications: networking, manufacturing or cultural applications based on stigmergy and swarm like systems, and self-managing global computers. Additional descriptions of self-organising applications can be found in (Mano, 2006).

Networking Systems

Seminal work by (Bonabeau, 1999) describes different types of swarm behaviour and explains how to apply it to different applications: ant foraging is useful for routing in communication networks, and consequently for optimisation of network-based problems; ant division of labour is useful for task allocation; ants' management of dead bodies is useful for clustering.

T-Man is a generic protocol based on a gossip communication model and serves to solve the topology management problem (Jelasity, 2005). Each node of the network maintains its local (logical) view of neighbours. A ranking function (e.g. a distance function between nodes) serves to reorganise the set of neighbours (e.g. increasing distance). Through local gossip messages, neighbour nodes exchange or combine their respective views. Gradually, in a bottom-up way, through gossiping and ranking, nodes adapt their list of neighbours, and consequently change and re-organise the network topology. The T-Man protocol is particularly suited for building robust overlay networks supporting P2P systems, especially in the presence of a high proportion of nodes joining and leaving the network.

The SLAC (Selfish Link and behaviour Adaptation to produce Cooperation) algorithm (Hales, 2005) favours self-organisation of P2P network's nodes into *tribes* (i.e. into specialised groups of nodes). The SLAC algorithm is a selfish re-wiring protocol, where by updating its links with other nodes in order to increase its utility function, a specific node leaves its current tribe, and joins a new one. In addition to P2P systems, the SLAC

algorithm has many potential applications, for instance to organise collaborative spam / virus filtering in which tribes of trusted peers share meta-information such as virus and spam signatures.

In the field of mobile ad-hoc networks, a self-organised public key management has been defined. The idea is that each node simply carries a subset of the certificates issued by other users. This alleviates the need of centralised certification authorities (Capkun, 2003).

For intrusion detection and response in computer networks (Foukia, 2005) the immune system serves as a metaphor for detecting intruders, and the stigmergy paradigm is used for responding to the attack. Mobile agents permanently roam the network in order to locate abnormal patterns of recognition. Once an attack is detected, a digital pheromone is released so that the source of attack can be located, and a response to the attack can be given. Mobile agents specialised for tracking the source of the attacks are created by the system and act as ants by following the pheromone trail up to the source of the attack.

Manufacturing Control

The stigmergy paradigm serves also for manufacturing control (Karuna, 2003). Agents coordinate their behaviour through a digital pheromone. In order to fulfil manufacturing orders, they use mobile agents that roam the environment, and lay down pheromonal information.

Cultural Heritage

In the field of cultural heritage, a system inspired by bees' behaviour has been designed by (Fabrega, 2005). This system allows independent non-specialised people to enter information on a given subject. The underlying system then creates new concepts as they are entered into the system by users and correlates together existing concepts. The bees' queen maintains the number and type of bees; it creates new bees whenever a new concept appears. Different types of bees look for information (nectar), bring that information into cells (honey comb), validate the information, or look for similarities in honey combs.

Self-managing Systems

In order to help human administrators in managing large systems, such as self-managing distributed operating systems or networks, self-managing systems are being investigated and research efforts are dedicated to these systems.

Expected properties of self-managing systems are to self-configure, self-optimise their parameters, self-repair in case of errors, and to ensure themselves their protection. These characteristics place these systems under the category considered in this chapter. Indeed, such systems work more efficiently without central control; they need to adapt to changes, i.e. to re-organise; this system is dynamic since, for instance, components in

error have to leave the system and being replaced by new ones, updated components have to seamlessly integrate the system (Kephart, 2003).
However, for self-managing systems, considered as autonomous systems with emergent behaviour, the situation may be even more complex. Indeed, such systems have three aspects. First, they need to manage themselves; this can be considered a "regular" case of self-organisation. Second in addition to themselves they need to manage any additional resource pertaining to the system. Third they need to interact with a human administrator, this implies that such system need a mean to receive global orders from the administrators, and these orders have to be split down into low-level goals or tasks, and conversely, the results or information the self-managing system wants to provide to the human administrator have be coherently ¨packed into a single meaningful information. Individual components cannot send directly to the administrator their respective individual results.

CONCLUSION

We already observe that technologically advanced societies heavily rely on autonomous devices full of autonomous software (PDA, mobile phones, portable computers) interacting with each other in a more or less autonomous way. Our vision is that future applications will in fact be composed of autonomous systems organised in a society of devices and software seamlessly interacting together for supporting end-users citizens in their everyday life.

We currently observe that artificial system reproduce natural self-organisation principles. They are borrowed from biology, social behaviour of insects or humans. Different artificial techniques are used for realising these systems: from indirect interactions, to reinforcement, to adaptive agents, to cooperation, to establishment of dedicated middleware. The interest of self-organisation and emergence lies in the natural robustness and adaptation of these systems, and in the relative simplicity of the different components participating to the system. However, it is interesting to notice that, despite any benefit emergence and self-organisation can bring to a system, they are not necessarily a good thing. Indeed, in addition to the expected emergent behaviour, unexpected emergent behaviour will necessarily arise from the different interactions of the system. This behaviour will have a causal effect on the system, and especially in the case of self-interested agents, the optimum order (the stable state reached by the re-organisation) can actually be bad for individuals or even for everybody. Additionally, current engineering techniques have their limits in terms of control of the emergent behaviour, design of the system and prediction of the emergent expected or not behaviour. Research in this field is still beginning, and much work is needed before any commercial application is widely available for the public.

Acknowledgement

This work is supported by Swiss NSF grant 200020-105476/1.

References

Babaoglu, O. & Meling, H & Montresor, A. (2002). Anthill: A framework for the development of agent-based peer-to-peer systems. Proceedings of the 22th International Conference on Distributed Computing Systems (ICDCS '02). 15-22.

Beaufour, A. & Leopold, M. & Bonnet, P. (2002). Smart-tag based data dissemination. ACM International Workshop on Wireless Sensor Networks and Applications (WSNA'02). 68-77.

Bernon, C. & Chevrier, V. & Hilaire, V. & Marrow, P. (2006). Applications of self-organising multi-agent systems: An initial framework for comparison. Informatica, In press.

Bernon, C. & Gleizes, M.-P. & Peyruqueou, S. and Picard, G. (2002). ADELFE, a Methodology for Adaptive Multi-Agent Systems Engineering. Proceedings of the Engineering Societies in the Agent World (ESAW'02). 156-169.

Blaze, M. & Feigenbaum, J. & Lacy, J. (1996). Decentralized trust management. IEEE Symposium on Security and Privacy. IEEE Computer Society, 164-173.

Bonabeau, E. and Dorigo, M. and Théraulaz, G. (1999). Swarm Intelligence: From Natural to Artificial Systems. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press.

Bourjot, C. & Chevrier, V. & Thomas, V. (2003). A new swarm mechanism based on social spider colonies: from Web weaving to region detection. Web Intelligence and Agent Systems. 1(1), 47-64.

Cahill, V. & al. (2003). Using trust for secure collaboration in uncertain environments. IEEE Pervasive Computing Magazine. Special issue dealing with uncertainty. 2(3), 52-61.

Capkun, S. & Buttyan, L. & Hubaux, J.-P. (2003). Self-Organized Public-Key Management for Mobile Ad-Hoc Networks. IEEE Transactions on Mobile Computing. 2(1):52-64.

Castelfranchi, C. (2001). The theory of social functions: challenges for computational social science and multi-agent learning. Journal of Cognitive Systems Research. 2(1), 5-38.

De Wolf, T. & Holvoet, T. (2005). Emergence Versus Self-Organisation: Different Concepts but Promising When Combined. Engineering Self Organising Systems. Volume 3464 of LNAI. Springer-Verlag, 1-15.

Di Marzo Serugendo, G. & Gleizes, M.-P. & Karageorgos, A. (2006). *Self-organisation and emergence in MAS: An overview*. Informatica, In press.

Di Marzo Serugendo, G. (2005). On the Use of Formal Specifications as Part of Running Programs. Technical Report. Department of Information Systems. University of Geneva.

Di Marzo Serugendo, G. & al. (eds). (2004). Engineering Self-Organising Systems, Volume 2977 of LNAI. Springer-Verlag.

Ducatel, K & al. (2001). Scenarios for Ambient Intelligence in 2010. Technical Report, Institute for Prospective Studies.

Hofmeyr, S. & Forrest, S. (2000). Architecture for an Artificial Immune System. Evolutionary Computation Journal. 8(4), 443-473.

Fabrega, M. & Lòpez, B. & Masana, J. (2005). How Bee-like Agents Support Cultural Heritage. Proceedings of the Engineering Self-Organising Applications Workshop (ESOA'05). 206-220.

Fiadeiro, J. L. (1996). On the Emergence of Properties in Component-Based Systems. Proceedings of the International Conference on Algebraic Methodology and Software Technology (AMAST'96). Volume 1101 of LNCS. Springer-Verlag. 421-443.

Foukia, N. (2005). IDReAM: Intrusion Detection and Response executed with Agent Mobility. The International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'05). Utrecht, The Netherlands. 264-270.

Glansdorff, P. & Prigogine, I. (1971). Thermodynamic study of structure, stability and fluctuations. Wiley.

Grassé, P. P. (1959). La reconstruction du nid et les interactions inter-individuelles chez les bellicositermes natalenis et cubitermes sp. la théorie de la stigmergie: essai d'interprétation des termites constructeurs. Insectes Sociaux. 6, 41-83.

Hales, D. & Edmonds, B. (2003). Evolving Social Rationality for MAS using "Tags". The International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'03). ACM Press.495-503.

Hales, D. (2005). Choose Your Tribe! Evolution at the Next Level in a Peer-to-Peer Network. Proceedings of the Engineering Self-Organising Applications Workshop (ESOA'05). 61-76.

Hassas, S. & Di Marzo Serugendo, G. & Karageorgos, A. & Castelfranchi, C. (2006). Self-organising mechanisms from social and business/economics approaches, Informatica, In press.

Holland, J. H. (1998). Emergence – from Chaos to Order. Oxford University Press.

Jelasity, M. & Babaoglu, O. (2005). T-Man: Gossip-based Overlay Topology Management. Proceedings of the Engineering Self-Organising Applications Workshop (ESOA'05). 1-15.

Kamvar, S. D. & Schlosser, M. T. & Garcia-Molina, H. (2003). The Eigentrust algorithm for reputation management in P2P networks. The Twelfth International World Wide Web Conference (WWW 2003). 640-651.

Karuna, H. & al. (2003). Self-organising in multi-agent coordination and control using stigmergy. Engineering Self Organising Systems. Volume 2977 of LNAI. Springer-Verlag, 105-123.

Kephart, J. O. & Chess, D. M. (2003). The Vision of Autonomic Computing. Computer. 36(1), 41-50.

Mamei, M. & Zambonelli, F. (2003). Self-Organization in MultiAgent Systems: a Middleware Approach. Engineering Self Organising Systems. Volume 2977 of LNAI. Springer-Verlag, 233-248.

Mamei, M. & Zambonelli, F. & Leonardi, L. (2002). Co-fields: Towards a unifying approach to the engineering of swarm intelligent systems. Third International Workshop on Engineering Societies in the Agents World (ESAW'03), Volume 2577 of LNCS. Springer-Verlag, 68-81.

Mano, J.-P. & Bourjot, C. & Lopardo, G. & Glize, P. (2006). Bio-inspired mechanisms for artificial self-organised systems. Informatica, In press.

Varela, F. (1979). Principles of Biological Autonomy. Elsevier.

Weeks, S. (2001). Understanding trust management systems. IEEE Symposium on Security and Privacy. 94-105.

Wooldridge, M. (2003). An Introduction to Multi-Agent Systems. Wiley.

Zhu, H. (2005). Formal reasoning about emergent behaviours of MAS. Proceedings of the Seventeenth International Conference on Software Engineering and Knowledge Engineering (SEKE'05).