Concepts in complexity engineering

R. Frei*

Intelligent Systems and Networks Group,
Department of Electrical and Electronic Engineering,
Imperial College,
London SW7 2BT, UK
E-mail: work@reginafrei.ch
*Corresponding author

Giovanna Di Marzo Serugendo

CUI – Université Genève,
Battelle – Bâtiment A, Rte de Drize 7,
CH-1227 Carouge, Switzerland
E-mail: giovanna.dimarzo@unige.ch

Abstract: Complexity science has seen increasing interest in the recent years. Many engineers have discovered that traditional methods come to their limits when coping with complex adaptive systems or autonomous agents. To find alternatives, complexity science can be applied to engineering, resulting in a quickly growing field, referred to as *complexity engineering*. Most current efforts come either from scientists who are interested in bio-inspired methods and working in computer science or mobile robots, or they come from the area of systems engineering. This article reviews the definitions of the most important concepts such as emergence and self-organisation from an engineer's perspective, and analyses different types of nature-inspired technology. This is the first part of a set of two-articles on this topic; the second one provides a survey of currently existing approaches to complexity engineering, identifies challenges and gives directions for further research.

Keywords: complex adaptive systems; complexity science; bio-inspired; nature-inspired; engineering; multi-agent systems; self-organisation; emergence; self-* properties; autonomy; robotics.

Reference to this paper should be made as follows: Frei, R. and Di Marzo Serugendo, G. (2011) 'Concepts in complexity engineering', *Int. J. Bio-Inspired Computation*, Vol. 3, No. 2, pp.123–139.

Biographical notes: Regina Frei is currently a Postdoctoral Researcher at the Intelligent Systems and Networks Group, Department of Electrical and Electronic Engineering, Imperial College London, UK. She received his PhD from the Electrical Engineering Department, Faculty of Sciences and Technology, New University of Lisbon, Portugal, and MSc in Micro-Engineering from the Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland. Her research interests are self-organising assembly systems, self-* properties and complexity engineering.

Giovanna Di Marzo Serugendo is a Full Professor at the University of Geneva, Switzerland. From 2005 to 2010, she was a Lecturer at Birkbeck College, London, UK. She received her MSc in Computer Science and in Mathematics from the University of Geneva, Switzerland, and PhD in Software Engineering from the Swiss Federal Institute of Technology in Lausanne (EPFL), Switzerland. Her research interests are related to the engineering of self-* systems. She co-founded the IEEE International Conference on Self-Adaptive and Self-organising Systems and the Editor-in-Chief of the Association for Computing Machinery's Transactions on Autonomous and Adaptive Systems.

1 Introduction

Traditional engineering methods cope well with *reducible systems* (Norman and Kuras, 2004), i.e., those systems which can be decomposed without loss and which are made of parts or sub-systems which are well known, which interact in predefined and well-understood ways and mostly stay the same during the system's life time. For reducible systems, the sum of their parts makes the whole. Systems with emergence, at the contrary, are more (or less?) than the sum of their parts. For instance, swarming birds only follow very simple local rules, but as a whole the swarm exhibits sophisticated dynamic formations. In manufacturing, robotic modules may function perfectly in a certain arrangement, but not in another one. GPS-based mobile services may show good performance at locations where difficulties were expected, and bad performance in areas of good reception because of the interference of other devices.

An increasing number of modern systems do not correspond to the description of a reducible system; their composition as well as the user requirements and the environment dynamically change, and often, their behaviour or some of their characteristics are emergent.

Complexity is omnipresent (Delic and Dum, 2006), and there are two main directions of research:

1 complexity as an emerging phenomenon (in natural or engineered systems) to be understood

2 complexity as an engineering problem to be tackled, mostly by reducing the environmental complexity, or by augmenting the system's capabilities of coping with complexity (Schuh et al., 2006).

Complexity engineering (Buchli and Santini, 2005) can be considered as a third direction, which currently attracts the attention of an increasing number of researchers: using complexity for engineering – not fighting against it, but using it to the engineer's favour. This is the topic of this article. Under the name of *emergent engineering* (Ulieru and Doursat, 2010) argues for the same paradigm change as we suggest with complexity engineering.

It is crucial for the advances in complexity engineering to clarify concepts which, originally used in chemistry, physics, biology or sociology, have been transferred to technological systems and engineering. Researchers often disagree with each other about the meaning and implications of concepts, and this article contributes by discussing and clarifying the most important concepts.

1.1 Scope and organisation

The topic of this article is engineering, not the sole study of complex systems (CS). We therefore do not discuss *natural* CS, but rather consider how to engineer *artificial* CS, and how to use the findings of complexity science. Different complexity disciplines are explained in Section 2.

Researchers such as Lucas (2008), Wolfram (1986), Holland (1975, 1992, 1995, 1998), De Wolf (2007) and Gershenson (2007) have studied the various definitions for CS and their characteristics. We therefore only briefly consider complex systems definitions in Section 3 and lay the focus on various notions which are important for this article, like self-organisation and emergence. The controversies between emergence, surprise, unpredictability, (non-)determinism and others are discussed as well as the differences between distributed and decentralised control.

In Section 4, we reflect on the complexity engineering concepts, draw conclusions and give directions for future work.

2 Engineering types

Complex and unconventional systems require different mind-sets than offered by classical engineering. This is why general and complexity-related engineering comes in various flavours. It is important for the reader to understand the different types of engineering, some of which are only currently emerging, and have not been established as proper disciplines yet. This does, however, not reduce their importance and relevance.

Table 1 gives an overview of the engineering types and the systems which they respectively address; these engineering types are discussed in Sections 2.2 to 2.5.

Table 1 System types and engineering types

	Individual systems	*Systems of systems*
Reducible systems	Classical engineering	Classical systems engineering
Systems with emergence	Complexity engineering	Complex systems engineering

2.1 Preliminary discussion

This section addresses relevant aspects and notions which are required to understand the subsequent classification.

2.1.1 Systems of systems

Systems of systems (SoS) are very large and CS (Bjelkemyr et al., 2007), composed of complex subsystems. The *entwined* nature of the systems' multiple components limits the success of a standard divide-and-conquer approach (Bullock and Cliff, 2004). Classical methodological approaches neglect or are unable to fully capture the sources of emergence and evolvability in distributed networks.

2.1.2 Modularity and its limitations

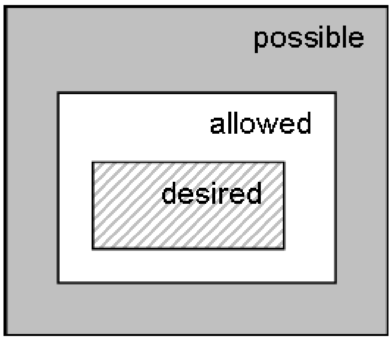
Modularity is a well-known way to divide a large system into parts which can be individually designed and modified. The modules can then be assembled stepwise, and the system's functionality verified accordingly (Kenger, 2006). This works very well for reducible systems. Modularity is closely related to *reductionism*, which reduces the system to the sum of its parts, and goes contrary to the principle of emergence. Reductionism is only valid if the parts are unrelated (which is rarely the case). Nevertheless, analysing the parts can be helpful: they are easier to understand, and their sum gives an idea of the whole, even if incomplete (Auyang, 1998). Also, modules are useful as building blocks to create systems which may or may not exhibit emergent behaviour.

However, in the case of CS, it is wrong to assume that the behaviour of the whole system could be reduced to the sum of its parts (Bar-Yam, 2003). The parts are often strongly dependent on each other and interact in multiple ways. Therefore, one of the challenges of complexity engineering is how to integrate modularity into a framework which can cope with emergent behaviour (if this should be possible). The effects of *composition* arise where many, in themselves often simple, entities interact to form a system; the resulting behaviour is usually not simply the linear sum of the behaviour of the individual components (Gell-Mann, 1995).

2.1.3 Trade-off: creative freedom versus specification

One of the challenges in engineering is the trade-off between the system specification[1] by the designer and the creative freedom of the system (Buchli and Santini, 2005; Choi et al., 2001). More freedom means less control over the system's behaviour.

Figure 1 Desired, allowed and possible areas (*boxes*) of system behaviour



Engineers need to find ways to delimit the system's behaviour while still allowing it sufficient creative freedom to localise solutions in an adaptive way. Indeed, most systems exhibit certain patterns of behaviour, which is enough to make predictions about system behaviour. Consequently, any state belonging to the delimited patterns are acceptable. In other words, to assure that the system will not show undesired behaviour, the system can be bound to a *virtual box*.[2] Inside this box, the system is free, but it may not leave the box [Figure 1, for further discussion see also (Di Marzo Serugendo, 2009)]. While the behaviour is inside the range specified by the *desired* box, no actions need to be taken. If the system leaves this area and remains inside the *allowed* area, no drastic measures need to be taken, but the system should eventually steer itself back towards the desired area. In case a system should diverge take states which are *possible*, but not allowed, immediate actions are necessary to bring the system back on save grounds. The fact that a system even reached this non-allowed area already means that the working parameters or policies need to be adjusted. The difficulties here are:

- Finding the box or pattern which corresponds to the acceptable system behaviour. This means that the designer has to define the limits of what is acceptable, and then somehow relate one borderline to another.
- Describing the box or pattern in a coherent way. Besides a description in natural language, in most cases also a computer-readable/-understandable version is necessary.
- Agreeing a compromise between the normally acceptable system behaviour and the additional freedom we can concede to the system. For instance, normally a mobile robot may not be allowed to enter a certain area of the shopfloor. Nevertheless, allowing it to do so under certain circumstances may enable the other mobile robots to execute their task in a more efficient way, and thus, it may improve the overall system performance. This is why the designer must find a balance between the benefits and potential dangers of crossing the border of acceptable system behaviour.
- Staying inside the box.

To conclude, the trade-off between creative freedom and specification is an important issue, and further investigation is necessary.

2.1.4 Complexity science versus complexity engineering

Complexity science, the study of CS, has seen an increasing interest in the last decades, pioneered by the Santa Fé Institute in New Mexico, USA (Waldrop, 1992). Ever since characteristics of CS in diverse areas have been thoroughly studied. Only few authors, however, take an engineer's perspective towards using the findings of complexity science for designing systems, even though the term *complexity engineering* appeared already in 1986 in the context of pattern recognition in *cellular automata* (Wolfram, 1986).

As we use it today, complexity engineering aims at the concrete use of complexity-inspired methods for engineering. "In complexity science, one looks for underlying and unifying principles among many systems. In complexity engineering, we look into these different systems and their underlying principles from the point of view of application" (Buchli and Santini, 2005).

Complexity engineering has not been established as a proper discipline yet. Literature about methods or frameworks is still scarce. One of the reasons may be basic misunderstandings over common terms such as emergence, i.e., The seeming contradiction between engineering and emergence may arise because engineers freely move from so-called *predictive* definitions, in which emergence is equated to surprise, towards definitions of *strong emergence* where higher-level patterns can be used as design templates (Johnson, 2005). To overcome such problems, the broader dissemination of clear definitions (see Section 3.4) is important.

The techniques and concepts from *complexity* science need to be formalised in order to be usable in engineering (Buchli and Santini, 2005). Most complexity research is still in an early stage of development, in the 'trial and error intuitive engineering phase'. So far, there are almost no methodologies, no common language and no common body of experience. Only a collection of examples, methods and metaphors for modelling complex, self-organising systems exist. Integrated theoretical foundations are still lacking (Heylighen, 2008).

This situation is a strong motivation to our endeavour to establish complexity engineering as a broadly known discipline, to deliver useful definitions, and to give an overview of the existing methods and approaches.

2.2 *Classical engineering*

Classical engineering is what is taught in most common engineering courses at universities, and what most scholarly books transmit. The engineering sciences have centuries old traditions, and in comparison, complexity science is relatively recent. It is still mostly considered as somewhat alternative, and we therefore refer to the 'old' school as *classical* or *traditional*.

Such engineering is essentially applying methods and tools to solve problems using a reductionist approach, be it top-down or bottom-up. This means that 'what you see is what you get', and there is no space to consider concepts like emergence.

Classical engineering, the majority of scientific models as well as our intuitive understanding are based on reductionism or analysis, predictability and objectivity, determinism, correspondence theory of knowledge and rationality (Gershenson, 2007). An example could be the *divide and conquer* strategy of software development strategies and Newtonian mechanics (Heylighen, 2008): a problem is cut into its simplest components, and each of them is treated separately. They are described in a complete, objective and deterministic manner. Once each of them is resolved individually, then they are joined and, voilà, the entire problem is solved. Most engineers would indeed make the (often reasonable) hypothesis that the parts interact in some well-known and predefined ways without further influencing each other. Modularity then works at its perfection.

For application examples of classical engineering see Table 2.

Table 2 Application examples of *classical engineering*

Engineering area	*Application example*
Software engineering	Planning algorithms
Mechanical engineering	Construction of a crane
Electrical engineering	Hierarchical control of a machine
Production engineering	Dedicated assembly station
Biotechnology	Fruit fly breeding

2.3 *Classical systems engineering*

Systems engineering, as described by the *international council on systems engineering* (INCOSE), is an interdisciplinary approach focusing on all aspects of systems. It considers all phases of a product, from its concept to production, operation and disposal, as well as all the involved parties, such as suppliers, manufacturers and customers. Although systems engineering attempts to consider the entire system instead of only parts of them, it is indeed a classical engineering approach because it ignores emergence and related concepts.

Systems engineering emphasises the importance of managing the whole as well as its parts, of seeing the interconnectedness of decisions, of taking a collective view.

For application examples of classical systems engineering see Table 3.

Table 3 Application examples of *classical systems engineering*

Engineering area	*Application example*
Software engineering	Large database systems with several subsystems
Mechanical and electrical engineering	Cars, trains, ships, air planes
Production engineering	Dedicated assembly line
Biotechnology	Production of vaccines

2.4 *Complexity engineering*

Complexity engineering is the creation of systems using tools originating from complexity science. The question is not so much in which ways complexity engineering would be better than classical engineering, but rather, in which situations classical engineering comes to its limits and complexity engineering can help. This is mostly the case with CS (discussed in Section 3.1): systems which are composed of many interacting components, where the interactions are multiple and changing in time; open systems; systems which have to function in a dynamic environment and strongly interact with it. CS use adaptation, anticipation and robustness to cope with their often unpredictable environment (Gershenson, 2007), and complexity engineering therefore requires tools which take these issues into account.

Such systems, said to have *emergent functionality* (Steels, 1991), are useful in cases where there is a lot of dependence on the environment and it is difficult or impossible to foresee all possible circumstances in advance. Traditional systems are therefore unlikely to be able to cope with such conditions. Systems with emergent functionality can be seen as a contrast to reducible systems and usually hierarchical functionality; the latter means that a function is not achieved directly by a component or a hierarchical system of components, but indirectly by the interaction of lower-level components among themselves and with the world. Careful design at micro level leads to behaviours at macro level which are within the desired range.

Typically, no single entity within the system knows how to solve the entire problem. The knowledge for solving local problems is distributed across the system (Gershenson, 2007), and together, the entities achieve an emerging global solution. The right interactions need to be carefully engineered into the system, so that the systems self-organising capabilities serve our purpose, i.e., they do satisfy and support the requirements (Buchli and Santini, 2005).

Complexity engineering will not lead to systems which are unpredictable, non-deterministic or uncontrolled. The output (i.e., certain aspects) may be predicted and controlled – it is how the system arrived to that output that can not be known, complex or not computationally reproducible (Buchli and Santini, 2005). However, it remains an open question if the latter is acceptable for all application domains. The system's development cannot be completely separated from the system's operation in the case of a CS (Norman and Kuras, 2004).

For application examples of complexity engineering see Table 4.

Table 4 Application examples of *complexity engineering*

Engineering area	*Application example*
Software engineering	Peer to peer systems, grid
Mechanical and machines materials engineering	Made of intelligent' materials, which recognise when parts undergo too much strain
Electrical engineering	Traffic control
Production engineering	Individual evolvable assembly systems
Biotechnology	Tissue engineering, growing organs in the test tube

2.5 CS engineering

In contrast to classical systems engineering, which treats reducible systems, CS engineering will apply the methods from complexity engineering to SoS. CS engineering (Kuras, 2006) is appropriate to address problems which are continually changing or which require concepts at multiple scales or levels to be fully understood. The notion of *higher* and *lower* scales of conceptualisation gives rise to the metaphor of a *ladder of scales*, in contrast to the often-used concept of a *hierarchy of scales*.

Table 5 Application examples of CS engineering

Engineering area	*Application example*
Software engineering	Self-organising displays (Puviani et al., 2010)
Electrical engineering	Large-scale traffic management
Production engineering	Evolvable assembly systems including their supply networks and customers
Biotechnology	Man-made biological ecosystems
Robotics	Open mobile robots coalitions

CS engineering is typical for cases where SoS constantly evolve, where different parts integrate or compositions dissolve at any instant. There is both internal competition and collaboration which stimulates evolution. Specific outcomes of complex-system development cannot be specified in advance. But they can be shaped (i.e., strongly and persistently influenced) (Wolfram, 2002), i.e., by guiding policies as used in MetaSelf (Di Marzo Serugendo et al., 2010).

For application examples of CS engineering see Table 5.

2.6 Inspiration from nature

Not only CS, but also nature in general inspires many researchers and engineers. The following classification attempts to structure this broad field.

Bio-inspiration in technology can take various forms. Each of them has particular goals and strategies, and researchers should be aware of them. The items 1, 2a and 2b on the following list correspond to the three-research phases of inspiration by nature described in Frei and Barata (2010). The last three-items are additional. Table 6 gives an overview of inspirations and applications.

1 *Using technology to understand natural systems*: Biologists, chemists and physicists have for a long time been using technological tools to help them investigate natural systems and to verify the established models. The palette of such tools includes oscilloscopes, gyroscopes as well as compound pendulums. More recently, computers allowed researchers to run large-scale simulations with thousands of iterations. Even more sophisticated, nowadays researchers use robots to emulate natural systems, and they even succeed in incorporating robotic 'cockroaches' into real cockroach swarms (Correll and Mondada, 2007; Halloy et al., 2007).

2a *Using ideas from natural systems to make lab experiments and find usable mechanisms*: This refers to the experimental phase of *bionics*. Researchers understood long ago that they can learn from nature and use mechanisms discovered in natural systems to solve engineering problems. However, most mechanisms need to be adapted in order to be usable, and this can only happen through an experimentation phase in the lab. Different versions are often discovered by changing the initial mechanisms, and the researchers can let their creativity play.

2b *Using ideas from natural systems to build industrial technology*: The final goal of most bionic developments is using them in real-world applications. This means that they have to comply with industrial standards. It has been achieved for many technologies, such as ultrasound, radar and sonar systems, dolphin-shaped boats, ultra-hydrophobic and self-cleaning surfaces based on the Lotus effect, and cat-eye reflectors. Researchers now increasingly approach distributed and autonomous adaptive systems, which are more difficult to build than other bionic applications.

3 *Using the 'engineering toolbox'*[3] *on natural systems*: Denominated *biotechnology, bio-medical engineering, genetic engineering* or similar, these disciplines use engineering technology on natural substrates such as

living cells, bacteria and sometimes higher animals. Researchers grow virus cells in tanks to produce vaccines, they try reproducing epidermic tissue and inner organs or genetically modified animals. Many different technologies are being used to diverse purposes. As a specific example, when a certain gene is implanted and then inherited to future generations, cancerigenous cells can become fluorescent, which facilitates their identification under the microscope.[4]

4 *Using biotechnology methods for software engineering*: Researchers in computer science now often take inspiration from methods used in biotechnology, in particular in cell engineering. Methods which work for living cells supposedly also work for software agents.

5 *Using ideas from engineering to build new models for understanding natural systems*: Probably the most recently initiated discipline considers architectures and mechanisms used by engineers to create technological systems which have nothing to do with natural systems. Natural scientists then use such ideas to build new models for understanding natural systems (Reeves and Fraser, 2009), in the sense that if engineers have come up with ideas, maybe nature has invented them long ago.

Complexity engineering, as treated in this article, belongs to the class 2a/2b in the sense that it uses inspiration from nature for engineering.

Table 6 Engineering and natural systems

Phase	*Inspiration/assisting tools*	*Application/goal*
(1)	Technological tools	Understanding natural systems
(2a)	Natural systems	Lab experimentation
(2b)	Natural systems	Industrial engineering, technology
(3)	Engineering methods	Biotechnology on living substrates
(4)	Biotechnological methods	Software engineering
(5)	Software engineering methods	Building artificial models to understand natural systems

3 Definitions and terms

Many of the terms discussed in this article are often used with an intuitive understanding in colloquial speech. Also in scientific work, they take varying meanings. The following subsections discuss the definitions of these terms in scientific use.

- *Agents*: By *agent* we refer to an entity which is able to act in a fairly autonomous way, according to norms/rules/policies and in order to achieve a goal. An agent can be something like an ant, a human person, a robot or a software agent. It consists of some kind of brain or computational power and often also has some kind of embodiment.
- *Systems*: As a working definition, a system may be considered as a set of entities (often agents), which interact with each other as well as with the environment, and some infrastructure or passive components/entities.

3.1 Introduction to complexity

Complexity issues have been studied within various contexts, i.e., physical phenomena (Nicolis and Prigogine, 1977), cellular automata (Wolfram, 1986; Langton, 1986), ICT systems (Bullock and Cliff, 2004), supply chain networks (Choi et al., 2001), management (van Eijnatten, 2005), networks (Schuh et al., 2006; Mitchell, 2006), modelling (Oliver et al., 1997), the laws of diversity (Ashby, 1956), natural disasters such as earthquakes (Ball, 2004) and epidemics (Gladwell, 2000), adaptation in Holland (1975, 1992, 1995, 1998), the dynamics of CS (Bar-Yam, 1997), chaos theory (Newman, 1996), and engineering aspects (Rzevski, 2004; Rzevski and Skobelev, 2007; Rouse, 2003; Abbott, 2006; Bar-Yam, 2003, 2005; Woodard, 2006; Zapf and Weise, 2007). The search of the mechanisms behind emergence and self-organisation has also been approached by many complexity researchers, such as Kauffmann (1995), Heylighen (2003), Camazine et al. (2001) and Steels (1991).

In some way, many open questions are related to each other, and common characteristics can be identified when investigating, i.e., how often earthquakes of a certain strength happen, why certain neighbourhoods become dangerous, how and why epidemics spread, through how many degrees of separation we are linked to any other person in the world, etc. The study of non-linear systems, dynamic systems, differential equations, non-determinism is intimately related to the nature of intelligence, the creation of structure and organisation, the creation of life, emergence, self-organisation, the micro- and the macroscale, etc. Table 7 places complexity between deterministic and statistical science, in terms of the scope in time and numbers of entities considered.

CALResCo (Lucas, 2008) is a valuable source for all kinds of question concerning complexity science, which searches the laws that apply at all scales, the inherent constraints on visible order. Typical systems may be described as: "Critically interacting components self-organise to form potentially evolving structures exhibiting a hierarchy of emergent system properties". This is a confirmation that the study of complexity science may prove to be useful for agile manufacturing.

The study of CS requires a conceptual framework which should include three different perspectives (Amaral and Ottino, 2004): non-linear dynamics and chaos theory, statistical physics including discrete modelling, and network theory, which is especially useful for understanding the internet and other communication networks, the structure of natural ecosystems, the spread of diseases and information, the structure of cellular signalling networks, and infrastructure robustness.

Some authors' strategy is to avoid complexity as far as possible, and they use metrics to determine the degree of complexity of a given configuration (Kuzgunkaya and ElMaraghy, 2006). Most researchers, however, aim at gaining a better understanding of complexity before dismissing it as assumingly being disturbing or useless.

The assumption that principles and mechanisms which are successful in nature will also work in technology/engineering is not undoubted. Besides the numerous similarities they share (Frei and Barata, 2010), there are also important differences between nature and engineering (Spilker, 2007). Namely:

- In nature, there is time and space for failures. In engineering, we must get it right the first time (or at least very soon after a test phase), and we must avoid failures.
- The main goal is (supposedly) only survival of the species. In technology, we have very specific goals.

Taking these differences into consideration is certainly sensible.

3.1.1 Complexity definitions

For instance, industrial assembly systems are complex; there is a plentitude of often conflicting interests and objectives being pursued, and the overall behaviour of the system results from the behaviour of many individual components which mutually and multi-laterally[5] influence each other. Complexity science is an area of research which studies exactly this kind of systems, and is therefore potentially a useful tool for assembly engineers. To understand how complexity might help, it is necessary to understand complexity itself – which is not evident, especially as it comes in many different flavours, depending on both the field of research and the researcher.

Complexity can be defined as "the name given to the emerging field of research that explores systems in which a great many independent agents are interacting with each other in many ways" (Waldrop, 1992). Examples of such systems (Auyang, 1998) could be electrons and molecules, which require cohesive and disruptive forces to work the way they do. Instantiations of this principle are ordering and disordering forces, kinetic energy and binding energy, coherence and disruption, transaction cost and administrative cost, etc.

Quite different sounds this definition:

> [Complexity is] "that property of a language expression which makes it difficult to formulate its overall behaviour, even when given almost complete information about its atomic components and their interrelations." (Edmonds, 1999)

Various researchers have tried to classify complexity types:

1 Random complexity, probabilistic complexity, deterministic chaos, emergent complexity and Newtonian dissipative structures (Maguire and McKelvey, 1999).

2 *Effective complexity versus underlying simplicity with a certain amount of logical depth*, which may also seem complex (Gell-Mann, 1995).

3 Complexity can also be classified by the following characteristics (Philipp et al., 2006):

   - Time-related: static or dynamic.
   - Organisational: process-related or structural.
   - Systemic: internal or external.

4 The *external complexity* (Jost, 2004) is the amount of input, information, or energy obtained from the environment which the system is capable of handling. The *internal complexity* is the complexity of the input representation which the system receives. CS often increase their external complexity to reduce their internal complexity.

Complexity is characterised by non-linear relationships between parts, openness, feedback loops, emergence, pattern formation, and self-organisation (Grobbelaar and Ulieru, 2007). In linear systems, effect is directly proportional to cause, whereas in non-linear systems, the effect may be any. Non-linearity comes in many flavours, tending to occur when a system's interactions are multiple, ecologically embedded, non-additive, inseparable, heterogeneous, interactive, asynchronous, lagged, or delayed (Bullock and Cliff, 2004).

3.1.2 Complex systems

CS can be defined in various ways. Most scientists consider CS as being composed of a large number of relatively simple heterogeneous components, which interact multi-laterally and in changing ways; collective behaviour emerges. The interactions sometimes result in non-linear behaviour, and there are multiple feedback loops. CS often evolve, adapt, and exhibit learning behaviours. They typically exhibit emergence and are often self-organised.

The original Latin word *complexus* signifies *entwined* or *twisted together* (Heylighen, 1996). A CS is thus made of more than one part, and the parts are at the same time distinct and connected. It is therefore inherently difficult to model them. Often, there are circular causal relationships: one part influences the other, which in turn influences the first, and so on.

CS refer to 'a set of systems which share some common behavioural and structural properties', where the meaning of structure can be spatial, temporal or functional (Grobbelaar and and Ulieru, 2007).

The micro level interactions between parts of the system may either be independent or coherent, resulting in different collective behaviours (Grobbelaar and and Ulieru, 2007):

- coherent interactions: coordination at microscale only
- independent interactions: random behaviour at microscale, coordination at macroscale
- correlated behaviours: coordination at micro and macroscale.

3.1.3 Complex adaptive systems

Systems which emerge over time into a coherent form, and adapt and organise themselves without any singular entity deliberately managing or controlling it, belong to the class of *complex adaptive systems* (CAS) (Holland, 1995). CAS are many body systems, composed of numerous elements of varying sophistication, which interact in a multi-directional way to give rise to the systems global behaviour. The system is embedded in a changing environment, with which it exchanges energy and information. Variables mostly change at the same time with others and in non-linear manner, which is the reason why it is so difficult to characterise the system's dynamical behaviour.

CAS often generate 'more of their kind' (Gell-Mann, 1995), which means that one CAS may generate another. To characterise them, researchers describe their components, environment, internal interactions and interactions with the environment.

It remains open if there are CS which are not adaptive. Some researchers agree, as, depending on its definition, adaptivity may require diversity and natural selection, as shown in ecosystems (Grobbelaar and and Ulieru, 2007).

3.2 Self-organisation

A well-known definition was suggested by Camazine et al. (2001):

> "Self-organisation is a process in which patterns at the global system emerges solely from numerous interactions among the lower-level components. Moreover the rules specifying interactions among the system's components are executed using only local information without reference to the global pattern."

The following definition is a few years more recent:

> "Self-organisation is the dynamical and adaptive mechanism or process enabling a system to acquire, maintain and change its organisation without explicit external command during its execution time; there is no centralised or hierarchical control. It is essentially a spontaneous, dynamical (re-) organisation of the system structure or composition." (Di Marzo Serugendo et al., 2006a, 2006b)

The identification of a boundary of the system is extremely important when deciding if a system is self-organising or not: defining an entity with controlling influence as external disqualifies a system from being self-organised, whereas the situation is different if the entity is considered as being internal.

By some researchers, self-organisation may also be seen as the spontaneous creation of globally coherent pattern out of local interactions (Heylighen, 2003) (although this is usually considered as the definition of emergence, see Section 3.4).

This shows how controversial the research area still is. Preconditions for having self-organisation in engineered systems, based on characteristics discussed in Correia (2006), De Wolf and Holvoet (2005), Di Marzo Serugendo et al. (2006b) and Heylighen (2003), are:

- Autonomous and interacting units.
- No external control; the question of corresponding system boundary definition arises.
- Positive and negative feedback. For instance, monetary rewards/punishments for successful collaboration and achievement of tasks respectively contract breaching or failures.
- Fluctuations/variations which lead to the typical far-from-equilibrium state, which is in manufacturing systems given by disturbances and changing production requirements, such as changing volumes and fluctuating part deliveries or equipment down-times.
- Safety measures in case the system should drift towards undesired or harmful behaviour.
- A flat internal architecture, as opposed to a hierarchical one, with dynamically changeable organisation of the interacting agents.

Adaptation means achieving a fit between system and environment; thus, every self-organising system adapts to its environment (Heylighen, 2003).

Mechanisms which lead to self-organisation in engineered systems include stigmergy (known from social insects, such as ants releasing pheromones in the environment), gossip, trust, collaboration/competition, swarms (as seen in schools of fish or flocks of birds), and chemical reactions. Most of these mechanisms happen according to a set of rules which can be identified. For instance, the entities in swarms respect three-principles, such as:

1 advance

2 stay close to your peers

3 avoid collisions.

Depending on the case and the mechanism, the rules can be more numerous, more complicated, and more complex. For engineering purposes, they may be adapted and extended.

A working definition for self-organisation seen from an engineering perspective is given in Section 3.2.2.

3.2.1 Weak and strong self-organisation

When it comes to concrete applications, it makes often sense to differentiate between *weak* and *strong* self-organisation. Not all cases do fully comply with the rules' of strong self-organisation, but still, there is some form of self-organisation.

In the *strong* case, the self-organisation happens without any centralised control, whereas in the *weak* case, there may be some internal (centralised) control or planning (Di Marzo Serugendo et al., 2005).

3.2.2 Working definition of self-organisation

After studying the existing definitions in literature as well as an engineering perspective on complexity concepts, we suggest the following *working definition*, based on the research done and experience gained in the scope of our work (Frei, 2010):

- *Self-organisation*: Systems which self-organise are typically composed of many, at least partially autonomous components. These components have certain characteristics and skills, and have at least one way of communicating with their peers and the environment. The environment dynamically changes and influences the system. The components engage in interactions with their peers; they may collaborate, compete, negotiate, gossip, and establish varying levels of trust between each other. This depends on the mechanism which leads to self-organisation. The components may have individual goals, but also shared or global goals. The system is not under any type of external or central control, although in engineered systems, the self-organisation process happens according to certain rules which were defined by the system designer. These rules may be dynamically changed, even at run-time, and thus allow the designer to influence the system at any time. Self-organisation is scalable, robust, and fault-tolerant, i.e., insensitive to small perturbations and local errors as well as component failure, thanks to redundancy. Self-organising systems exhibit graceful degradation, meaning that there is no total break-down because of minor local errors. Self-organisation is a *dynamic process* in many-body systems and may occur with or without emergence.

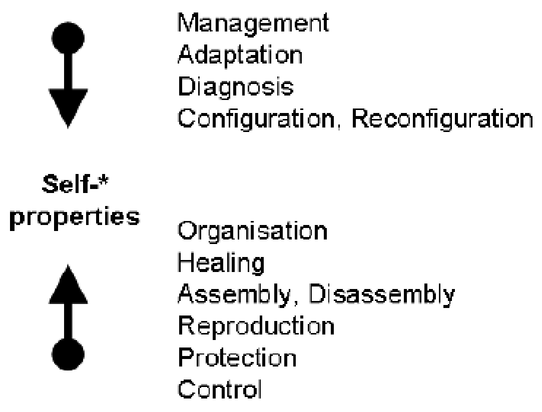
3.3 Self-* properties

In literature, diverse interpretations of self-organisation, self-adaptation, self-management, self-(re)configuration, self-healing and emergence can be found. Many of them focus on one single term; only few mention the links between the concepts. For instance, self-adaptation is included in self-managed systems, and self-management is included in self-organisation, according to the classification in Muehl et al. (2007).

One important differentiation to be made is the direction of the property: self-organisation and self-healing is bottom-up, whereas, self-adaptation, self-management and self-healing are top-down, as illustrated in Figure 2.

Besides the differences in the orientation (bottom-up or top-down), most often the name given to the property is a question of the focus: the behaviours can sometimes not even be clearly classified as 'pure organisation' or 'pure healing' etc., and most often self-* properties have an emergent character. As an example, when a system re-organises its internal structure to recover from a failure, is this self-organisation or self-healing? Is self-organisation used for self-healing? Or is it emergence, because the process is based on local rules and produces a new, global result? It depends on the rules which define the behaviour, but an observer may not know them.

Figure 2 Bottom-up and top-down self-* properties



De Wolf suggested a taxonomy of self-* properties (De Wolf and Holvoet, 2007) which focuses on *decentralised autonomic computing* and discusses characteristics of self-* properties and implications for their engineering. Among other criteria, the taxonomy considers if a self-* property is achieved on macroscopic or microscopic level, if it is on-going or one-shot, if it is time/history dependent or independent, if it evolves in a continuous or abrupt way, and it is adaptation-related or not. The taxonomy gives examples of mechanisms leading to self-* properties and classifies application examples according to the considered characteristics, but it does not indicate to which kind of self-* property a mechanism or application belongs.

Even though the following classification is general, the following non-exhaustive list of working definitions is influenced by the domain of robotics and artificial intelligence. These working definitions (Frei and Barata, 2010) are not conclusive, but they give indications and contribute to a base for further research: they intend to trigger other researchers to reflect about them. The term *self* generally refers to the absence of external control. After a general description, an application to assembly systems follows.

- *Self-adaptation*: a system adjusts itself to changing conditions without major physical modifications. ≫ For instance, in the case of an industrial assembly system (Frei, 2010), when more urgent orders arrive, a robot can increase its working speed.

- *Self-configuration*: a system prepares itself for functioning, including the adjustment of parameters and calibration. ≫ A robot adjusts its movement accuracy to the desired value.
- *Self-reconfiguration*: mostly encompasses self-adaptation and self-configuration, but also some physical change (including software and hardware). ≫ When a conveyor module fails, and there is an alternative conveyor path to reach the affected destination, the modules adapt their behaviour and use the alternative path until the module has recovered from the failure. Alternatively, a new conveyor module is requested from the user and integrated into the existing system.
- *Self-organisation*: a system creates or adapts its own structure to reach a goal. ≫ Modules form coalitions to provide the requested skills.
- *Self-assembly*: sub-systems or modules connect with each other to form the whole. ≫ A robot self-assembles with a gripper which it can autonomously pick up from its toolwarehouse.
- *Self-disassembly*: a system decomposes itself into subsystems or modules. ≫ A coalition which is not necessary any more disassembles. For instance, a robot will place its gripper back in the toolwarehouse.
- *Self-diagnosis*: modules can find out and state what is wrong with themselves. ≫ A feeder which cannot provide parts will check if there are no ready parts inside, or if there is a blockage, or if there is any other problem preventing normal functioning.
- *Self-repair/self-healing*: a system can treat its problems and maintain or re-establish functionality. ≫ A blocked feeder will restart its software, execute calibration movements, and if still blocked, ask the user for help.
- *Self-reproduction/self-replication*: a system can create a copy of itself. ≫ A module coalition incentivises suitable modules to form the same type of coalition.
- *Self-protection*: a system can protect itself from intruders or attacks. ≫ In case an assembly system was open enough for strangers to gain access to it, i.e., over the internet, it would need to protect itself from harm.
- *Self-control*: the system steers itself. ≫ The modules control their own behaviour, i.e., guided by policies.
- *Self-management*: a system can take care of itself. This may include self-protection, self-healing, self-configuration, self-optimisation, self-adaptation etc. ≫ At production time, the modules maintain themselves as well as their neighbours in good conditions. They manage their multi-lateral interactions, provide the requested services, schedule maintenance etc.

3.4 Emergence

Emergence describes how order appears out of chaos (Holland, 1998). Both emergence and self-organisation (Section 3.2) are concepts which first appeared in physics (phase transitions) and chemistry (molecules and material properties), and were then also observed in other domains, including biology (cells, DNA, brain, etc.), game theory, social science, economics and engineering. A general theory of emergence is still missing (Brueckner, 2000).

Most systems which exhibit emergence can be modelled in terms of the interaction of agents. Building blocks are combined to form a higher level system. Emergent phenomena are often hierarchical: complex ones are composed of simpler ones (Holland, 1995).

Definitions found in literature include:

Emergence is a bottom-up effect, which generates order from randomness (Mueller-Schloer, 2004). It results in a self-organised increase of order, in space or time. A global behaviour arises from the interactions of its local parts; cannot be traced back to the individual parts (De Wolf and Holvoet, 2005). Desirable and undesirable emergent behaviour in distributed systems results from the non-linear interaction of completely deterministic processes (Parunak and VanderBok, 1997). None of the entities composing the system knows how to achieve the emergent phenomenon (Di Marzo Serugendo et al., 2006b).

Although controversial, emergence does not only exist in the eye of the observer; it is intrinsic to the system (Holland, 1998). Novelty does not depend on the experience of the observer, neither. It refers to the new class of words used to describe the global phenomenon, new in the sense of different from those used for the local level description. However, novelty is not the same as surprise, as surprise is related to the preparation of the observer, and novelty is not.

According to Holland (1998), for engineered systems, emergence happens according to rules. The designers have to find the level of detail where they can set the rules and therefore control emergence. Notice that also this is a very controversial statement, as for most other researchers, this describes self-organisation, and not emergence.

It is mostly agreed that an emergent property (Auyang, 1998):

1 of a whole is not the sum of the characters of its parts.

2 is of a type which is totally different from the character types of its constituents.

3 is not deducible or predictable from the behaviours of the constituents investigated separately.

A *resultant* is different from an *emergent* (Auyang, 1998): A resultant is closely tied to the material content of the constituents. Linear systems have resultant behaviours and are traceable. The principles of superposition, aggregation and additivity apply. An emergent has a structural aspect, there is novelty and non-additivity. For instance, conductivity is resultant, whereas, superconductivity is emergent. Nevertheless, both properties involve the same ‘ingredients’. Emergent properties can in principle be predicted by analysing the lower levels; in practice, we are not always capable of doing it (Lucas, 2008).

Different forms of emergence (Castelfranchi, 2001) exist: *Diachronic*: develops in time. This may happen when new technologies are introduced and they combine with previously existing modules. *Synchronic*: different ways of looking at a given info from one level to another, e.g., emergence of a significant pattern, structure or form from the point of view of a given observer. *Descriptive*: synchronic, but not related just to the observer's conceptualisation and description; *objective* emergence if causal effect on environment. May occur when new system behaviours cause the user to take previously not necessary actions. *Cognitive*: becoming aware of previously ignored knowledge. A system designer or user may experience this.

Some authors consider that not only system characteristics may emerge, but also goals (Louzoun and Atlan, 2007) and functionalities (Capera et al., 2004; Steels, 1991). In the context of engineering, this may be interpreted as systems which can do things they were not made for.

A working definition for emergence seen from an engineering perspective is given in Section 3.4.2.

3.4.1 Weak and strong emergence

To bring the classical notions of emergence, discussed before, closer to the reality of engineered systems, two-classes of emergence are proposed (De Wolf, 2007; Fromm, 2005):

For *strong* emergence, the global level must show further development. There is non-linear dependence of the global functionality on the components and their interactions between themselves and the environment.

Weak emergence means that the local-to-global dependence may be *quasi-linear* – but still, the appearance of the global phenomenon is not self-evident and needs some kind of *inspiration*.

> "A macrostate is weakly emergent if it can be derived from micro-states and micro-dynamics but only by simulation." (Bedau, 1997)

3.4.2 Working definition of emergence

After studying the existing definitions in literature as well as an engineering perspective on complexity concepts, we suggest the following *working definition*, based on the research done and experience gained in the scope of our work (Frei, 2010):

- *Emergence*: Systems exhibiting emergence most often consist of at least two different levels: the macro level, considering the system as a whole, and the micro level, considering the system from the point of view of the local components. Local components behave according to local rules and based on local knowledge; a representation of the entire system or knowledge about the global system functionality is neither provided by a central authority nor reachable for the components themselves. They communicate, locally interact with each other and exchange information with the environment. From the interaction in this local world emerge global phenomena, which are more than a straight-forward composition of the local components' behaviours and capabilities. Typically, there is a two-way interdependence: not only is the global behaviour dependent on the local parts, but their behaviour is also influenced by the system as a whole. Nobody in the system knows how to achieve the emergent phenomenon, and nobody has complete knowledge of the system or a global observer's perspective. An emergent phenomenon is a *structure or pattern*, *visible at global level.*

3.5 Chaos

A system may be viewed as *deterministic* if the current state(s) of the system determine its future state(s) in the presence of random noise, environmental inputs and unknown initial conditions; a deterministic dynamic system whose behaviour is hard to predict is called a *chaotic* system (Grobbelaar and Ulieru, 2007).

Chaos in common language means confusion or the lack of fixed principles, whereas chaos in mathematics is behaviour according to certain rules (Auyang, 1998). The methods for mathematically describing chaotic behaviour founded by Poincaré and Lorentz bring structure into seemingly random behaviour (Waldrop, 1992).

Chaos is different from randomness: chaotic systems behave according to strange attractors. This means that under a set of conditions (i.e., within the attractor basin), a system will always move towards a certain state or set of states. To leave them, the system requires a certain energy input (disturbance). In mathematical terms, chaotic systems are deterministic, whereas randomness has no structure at all.

In complexity terms, *entropy* is the tendency of systems to create chaos from order, while *extropy* is the tendency of systems to create order from chaos (that is, emergence) (Lucas, 2008).

Chaotic systems have been discovered in domains as diverse as mathematics, physics, biology, chemistry, meteorology, fluid dynamics, astronomy and statistical mechanics and logistics (Ranjan et al., 2003). Also industrial assembly systems exhibit chaotic behaviour:

- cause and effect are not always in a linear relation, as a small perturbation may cause a total system breakdown
- a successful assembly system will tend towards an attractor which stands for the correctly assembled product, although it may assume different states on the way there
- assembly system behaviour is bound to certain limits (robots cannot suddenly start doing crazy things), although within the given boundaries, the behaviours may vary (different robots may dynamically take over the insertion of a bolt, according to their availability and performance).

3.5.1 Sensitivity to initial conditions

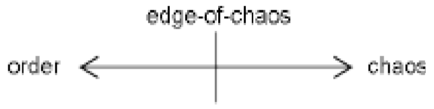
The *butterfly effect* stands for sensitive dependence on initial conditions. Little causes do not necessarily lead to little effects, and big causes to big effects. Future outcomes are arbitrarily sensitive to tiny changes in conditions (Gell-Mann, 1995). Simple mechanisms may cause considerable complexity, as well as complex sources may lead to simple phenomena. In the same sense, CS can give rise to turbulence and coherence at the same time. Brought to a simple formula, we may say: "In the middle of chaos, there is order. In the middle of order, there is chaos" (Gleick, 1987).

Manufacturing systems often exhibit sensitivity to specific conditions and to disturbances. Certain factors, like energy disruptions or an abnormal increase of temperature and humidity may lead to system breakdown, while others have no significant effect (i.e., the occurrence of extreme noise would disturb human operators, but not bother robots). Some disturbances may have consequences in some cases, but lack any effect in others. For example, a robot using optical sensors reacts sensitively to changing light conditions, whereas a robot working with tactile sensors remains unaffected.

3.5.2 Edge-of-chaos

Various terms are being used for the state somewhere between stable order and chaos (see Figure 3), among others: *dynamic order*, *instability in order*, and *self-organised criticality* (Ball, 2004; Gladwell, 2000).

Figure 3 Somewhere between order and chaos



Constantly stable equilibrium states would block evolution. Dynamic systems get again and again into states where a little stimulus can trigger a major reaction. This gives the systems energy to evolve and makes new phenomena emerge. "The edge of chaos is a point between chaos and order when creativity and stability fuse, where living systems are at their most inventive, where there is the highest chance that something distinct and unique will emerge" (Webb and Lettice, 2005).

Analogies may be drawn between sand piles, earth quakes, wars, extinctions of species and revolutions (Buchanan, 2000). Figure 3, somewhere between order and chaos. The world organises itself into a critical state at the edge-of-chaos: small events can stay small or grow to enormous importance and have heavy consequences. The power law describes such phenomena. It expresses that if we double the energy (or any other quantity being studied), the probability of the phenomenon to appear is half, a quarter, etc. For instance, if the probability of an earthquake of strength x to happen is y, the probability of an earthquake of $2x$ to appear is $\frac{y}{2}$. All events can have the same kind of trigger. There is no fundamental difference between small and big events. No-one knows if the next event falls onto a *finger of instability*, which leads to its propagation, or if it will stay small and not propagate. When building models of such events, it is often possible to greatly simplify. Researchers will find the same power laws in their models as in reality, if the fundamentals of the models are correct. It is at the edge-of-chaos that epidemics do or do not spread (Gladwell, 2000).

Failures and perturbations in manufacturing systems often follow power laws as well. This is why the systems must be able to cope with frequent small failures as well as with big rare ones.

3.5.3 Phase space/state space

Phase space or *state space* diagrams are used to represent the behaviour of systems, with all the states which are reachable for a system, and the transitions in-between, as a function of system parameters. The bifurcation diagram shows where the previously uniform behaviour of a system separates into different directions, and possibly diffuses into an unlimited number of different behaviours.

An attractor is a state towards which a system will always tend, as long as it is under a set of initial conditions. The *Lorentz attractor* and other *strange attractors* describe systems which never quite settle into a state, but eternally oscillate within a certain range of states, never taking the same state twice (Hongler, 1994). If we know a system's strange attractor and its dimensionality (the number of dimensions of the corresponding state space), we can make predictions about the systems behaviour, i.e., the performance of an automated production line (Hongler, 1994).

3.5.4 Noise, perturbations and local maxima

Perturbations are a challenge and a chance at the same time. Systems must cope with perturbations and not let themselves drift away from their normal functioning. However, perturbations can also be helpful: systems which require some kind of optimisation may tend to be stuck in local minima and thus not be able to evolve towards better solutions without the system being disturbed or otherwise stimulated.

The *cybernetic law of requisite variety* by Ashby (1956) teaches that the greater the variety of possible perturbations, the greater the variety of controlling actions it needs. This means that a system which is always perturbed in the same way will always require the same corrective measures. However, if there is a plenitude of different influences on the system, it will need a correspondingly varied set of ways of reacting.

Complexity engineers should try to use perturbations to their benefit. For instance, when a robot fails and other robots resolve to collaborate in an unusual way to cope with the failure of their peer, this new collaboration may be discovered as an efficient way of executing the task, and thus be retained for further use even after the peer's reparation.

3.5.5 Further concepts

The concepts described in this subsection have not been explained yet but are important for the general understanding of complexity science and chaos theory.

- *Fractals*: Inspiration for fractal manufacturing systems (Ryu et al., 2006). Fractals have a self-similar structure at arbitrarily small scale, meaning that new similar structures appear when zooming in; self-similarity may also be stochastic or approximate.
- *Attractors basin*: Like a river has a watershed basin that drains to it, every attractor has a basin. Of particular interest are the basin boundaries, which are often fractal.
- *Fitness function/landscape*: Organisms must be fit for survival and thus react to the requirements of the ever-changing environment. These requirements can be described by a fitness function. The closer an organism matches the fitness function, the better adapted it is to the current life condition. The criteria for endurance or elimination of new characteristics are most often multiple and form a *fitness landscape*.
- *Spontaneous order*: CS can spontaneously organise themselves into coherent patterns. Conflicting constraints lead to a *rugged* fitness landscape, which means that the fitness parameters do not evolve linearly/smoothly. The ruggedness is determined by the internal organisation of the organism.
- *Convergence*: Happens when a system tends towards the desired state/solution. If the system cannot reach it, no matter how long it runs (it may oscillate endlessly, or tend towards an undesired state, such as a chaotic attractor), the system does not converge. The *speed of convergence* (Di Marzo Serugendo, 2009) describes how quickly a system reaches the desired state.
- *Downward causation*: Influence of the global/macro system on its components/the micro elements, derived from the constituents' self-organisation. Also an emergent phenomenon can exhibit downward causation, that is, the emergent phenomenon influences the elements which lead to the emergence.
- *Equilibrium*: Self-maintained state of a (partially) isolated system. Equilibria can be stable, meta-stable, unstable quasi-stable, local/relative or global/absolute.

3.6 Dependability, robustness and similar terms

The terms *robustness*, *dependability*, *resilience*, *redundancy*, *degeneracy* and *graceful degradation* are often used in the same context: they all refer to how a system copes with failures and perturbations.

- *Dependability* is the ability of a system to deliver a service that can justifiably be trusted (Avizienis et al., 2004). For instance, a cash machine must always provide the same service, and we must be sure that nothing else happens when we are requesting a certain amount of cash. Central to this definition is the notion that it is possible to provide a justification for placing trust in a system. In practice this justification often takes the form of a dependability case which may include test evidence, development process arguments and mathematical or formal proof.
- The original meaning of *resilience* refers to the maximal elastic deformation of a material. In the context of computer science and robotics (Bongard et al., 2006; Di Marzo Serugendo et al., 2007), resilience means *dependability when facing changes*, or in other words, its ability to maintain dependability while assimilating change without dysfunction. In the case of MetaSelf (Di Marzo Serugendo et al., 2010), a key feature for dynamic resilience is the availability of dependability metadata at runtime. For instance, for dynamically attributing a new server, it is necessary to know the dependability values of the servers in question.
- *Dynamic resilience* is a system's capacity to respond dynamically by adaptation in order to maintain an acceptable level of service in the presence of impairments' (Di Marzo Serugendo et al., 2007), whereas *predictable dynamic resilience* refers to the capacity to deliver dynamic resilience within bounds that can be predicted at design time. Accordingly, for MetaSelf, *resilience metadata* is information about system components, sufficient to govern decision-making about dynamic reconfiguration. *Resilience policies* serve as guidelines for reconfiguration.
- *Stability* means in manufacturing that a process always delivers the same result, as long as the conditions are within a certain specified range. A system must continuously deliver correctly assembled products and cope with perturbations or failures.
- *Robustness* means that a system does not easily get disturbed in its normal functioning. It can cope with failures, changing conditions and is able to remain usable.
- *Redundancy* means that there are more than one elements with the same functionalities in a system. It is the standard solution of engineers to cope with failures, and it involves structurally identical elements. Redundancy is costly, because the redundant resources remain unused, and therefore redundancy is often avoided as far as possible. Self-organising systems have typically a lot of redundancy, which leads to an inherent robustness against many failures.

- *Degeneracy* (Sole et al., 2002) is an alternative which can be observed in natural systems such as the brain. In case of a lesion, structurally different brain regions can adapt to take over the tasks of the damaged area. The same can also be achieved in technological systems: i.e., a robot may request new coalition partners to form composite skills, which allow them to take over the task of a failing original robot.
- With *graceful degradation*, a damaged or perturbed system does not totally break down. It maintains at least part of its functionality, even if with reduced performance.

4 Discussion, conclusions and directions

After reviewing the most important concepts in complexity engineering, we now analyse the general situation. The role of the observer is addressed in Section 4.0.1, and challenges as well as limitations of self* systems in 4.0.2. At the end, we draw conclusions (Section 4.1).

4.1 The role of the observer

The role of the observer in determining whether or not a system exhibits emergence was treated in Section 3.4. The discussion here is more general, not limited to emergence.

What we perceive as an observer (or as many different observers) is often different from what really exists (Gershenson, 2007). The observer mostly has a very limited perspective. Not everything happening in a system is visible; the fact that something cannot be seen does not mean that it does not exist.

From the perspective of the observing designer, there is always a temptation to suppose that the created interactions do indeed take place, even if they are not visible. The designer should therefore try not to jump to conclusions which may not be well-founded. Similarly, an observer who is not the designer is always tempted to make interpretations of the observed and find explanations which may not correspond to reality. Also here, caution is appropriate.

According to discussions at the *4th Technical Forum Group on Self-Organisation* (*TF4*), some researchers think that an emergent phenomenon is meaningful to the observer (only?), and only if the observer is also the designer. Only the observer-designer determines if a phenomenon is indeed emergent because this person knows how the system works. This means that somebody who does not understand how a system works cannot correctly judge what is happening, i.e., cannot say if a phenomenon is a self-* property, or if it is under centralised control. In many situations, a system will, however, perform differently when under centralised control than when acting in a distributed-autonomous way. A careful observer may be able to determine the differences and come to the right conclusions.

As a matter of fact, a system which is made to run independently from a human observer (i.e., literally all systems we are considering here), will function while being observed or not. We therefore argue that observation can only help the observer to understand the system, but it does not change anything at the level of the system.

4.2 Challenges and limitations with self- systems*

Self-* properties (addressed in Sections 3.2 and 3.3) are an important part of complexity engineering. They allow systems to play active and increasingly autonomous roles in accomplishing their tasks, but there are also challenges and limitations to the possibilities of self-* properties:

- *Sensitivity to initial conditions*: Systems may efficiently find a way to accomplish their task under certain initial conditions, but not be able to do so when the conditions are slightly different. *Autonomous guided vehicles* (AGVs) may serve as an example: we suppose that their task is to pick up a variety of finished products from assembly stations and deliver them to boxes according to customer orders. If the AGVs start from distributed locations, they may very quickly settle into an efficient rhythm of picking up and delivering products. But when the AGVs start from a single point, it may take them much longer to coordinate the tasks between them, and thus their performance is affected. Engineers thus have to consider their system's sensitivity to initial conditions, and attempt to find solutions to mitigate the effects.
- *Parameter tuning*: Many applications depend on diverse parameters which have to be tuned in order for the system to run smoothly. Human operators often supervise the tuning, or do it manually by trial-and-error. Suitable strategies need to be developed if the system is to do is autonomously.
- *Latency to find new stable states*: Most self-* systems can eventually find stable states or stable solutions to achieve their tasks, but it takes time. This means that the designer and user of self-* systems must be able to accept delays.
- *No solution found/no convergence*: In certain cases, a self-* system may not be able to solve the task given, or its calculations may never converge. The designer has to preview this and arrange for a way out, such as alerting the user and/or settling for a solution which requires the relaxation of certain constraints.
- *Analysis of self-* properties*: It is inherently difficult to analyse self-* properties. The system may find ways to fulfil tasks which the designer did not plan or preview. The other way round, the designer may intend the system to act in a certain way, and in reality, it is all different. Also, the interplay between various self-* properties is difficult to analyse. Further research efforts are certainly necessary.
- *Dependability/resilience*: it must be assured that the system does what it is supposed to do, independent from the actual situation and circumstances, and this is

challenging, especially for the type of system considered here. Thanks to their redundancy, these systems are often inherently robust to certain types of failures, and this robustness comes for free. They may, however, be fragile when facing other faults. For further discussion see Di Marzo Serugendo (2009).

4.3 Conclusions

Although this article is directed at promoting complexity engineering, the authors are aware of the fact that complexity engineering is not always the most adequate solution. They should be chosen when classical engineering comes to its limits (compare Section 2.2), or when alternative ways of solving a problem are desired.

We have positioned complexity engineering within other engineering domains, such as systems engineering and classical engineering. We reviewed the definitions of important notions such as self-organisation and emergence, and explained the controversies between unpredictability, complexity and other related terms.

The second part of this set of two-articles on complexity engineering reviews existing methods. Please refer to Frei and Di Marzo Serugendo (2011).

Acknowledgements

This work was started while Regina Frei received his PhD grant from the Portuguese Foundation for Science and Technology. She currently receives a post-doc grant from the Swiss National Science Foundation.

References

Abbott, R. (2006) 'Complex systems + systems engineering = complex systems engineering', in *Conf. on Systems Engineering Research*, Position paper, Los Angeles, CA, USA.

Amaral, L. and Ottino, J. (2004) 'Complex networks – augmenting the framework for the study of complex systems', *European Physical Journal B*, Vol. 38, No. 2, pp.147–162.

Ashby, W. (1956) *An Introduction to Cybernetics*, Chapman & Hall, London.

Auyang, S. (1998) *Foundations of Complex-System Theories in Economics, Evolutionary Biology, and Statistical Physics*, Cambridge University Press, Cambridge, UK.

Avizienis, A., Laprie, J., Randell, B. and Landwehr, C. (2004) 'Basic concepts and taxonomy of dependable and secure computing', *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1, pp.11–33.

Ball, P. (2004) *Critical Mass: How One Thing Leads to Another*, Arrow Books, London, UK.

Bar-Yam, Y. (1997) *Dynamics of Complex Systems. Studies in Nonlinearity*, Addison-Wesley, Reading, MA, USA.

Bar-Yam, Y. (2003) 'When systems engineering fails – toward complex systems engineering', in *IEEE Int. Conf. on Systems, Man & Cybernetics (SMC)*, Vol. 2, pp.2021–2028, Washington DC, USA.

Bar-Yam, Y. (2005) 'About engineering complex systems: Multiscale analysis and evolutionary engineering', in Brueckner, S., Di Marzo Serugendo, G., Karageorgos, A. and Nagpal, R. (Eds.): *Engineering Self-organising Systems: Methodologies and Applications, ESOA 2004*, LNCS, Vol. 3464, pp.16–31, Springer Berlin.

Bedau, M. (1997) 'Weak emergence', *Philosophical Perspectives: Mind, Causation, and World*, Vol. 11, pp.375–399.

Bjelkemyr, M., Semere, D. and Lindberg, B. (2007) 'An engineering systems perspective on system of systems methodology', in *IEEE System of Systems Engineering*, San Antonio, Texas, USA, pp.1–7.

Bongard, J., Zykov, V. and Lipson, H. (2006) 'Resilient machines through continuous self-modeling', *Science*, November, Vol. 314, No. pp.1118–1121.

Brueckner, S. (2000) 'Return from the ant – synthetic ecosystems for manufacturing control', PhD thesis, Institute of Computer Science, Humboldt-University, Berlin, Germany.

Buchanan, M. (2000) *Ubiquity: The Science of History... or Why the World is Simpler than we Think*, Phoenix, London.

Buchli, J. and Santini, C. (2005) 'Complexity engineering, harnessing emergent phenomena as opportunities for engineering', Tech. rep., Santa Fé Institute Complex Systems Summer School, NM, USA.

Bullock, S. and Cliff, D. (2004) 'Complexity and emergent behaviour in ICT systems', Tech. rep., HP-2004-187, Hewlett-Packard Labs.

Camazine, S., Deneubourg, J-L., Franks, N., Sneyd, J., Theraulaz, G. and Bonabeau, E. (2001) *Self-organization in Biological Systems*, Princeton University Press, Princeton, NJ, USA.

Capera, D., Picard, G. and Gleizes, M-P. (2004) 'Applying ADELFE methodology to a mechanism design problem', in *Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, Vol. 3, pp.1508–1509, New York, USA.

Castelfranchi, C. (2001) 'The theory of social functions: challenges for computational social science and multiagent learning', *Cognitive Systems Research*, Vol. 2, No. 1, pp.5–38.

Choi, T., Dooley, K. and Rungtusanatham, M. (2001) 'Supply networks and complex adaptive systems: control versus emergence', *Operations Management*, Vol. 19, pp.351–366.

Correia, L. (2006) 'Self-organised systems: fundamental properties', *Revista de Ciencias da Computacao*, Vol. 1, No. 1, pp.1–10.

Correll, N. and Mondada, F. (2007) 'Modeling self-organized aggregation in a swarm of miniature robots', in *Int. Conf. on Robotics and Automation (ICRA), Workshop on Collective Behaviors inspired by Biological and Biochemical Systems*, Rome, Italy.

De Wolf, T. (2007) 'Analysing and engineering self-organising emergent applications', PhD thesis, Department of Computer Science, Katholieke Universiteit Leuven, Belgium.

De Wolf, T. and Holvoet, T. (2005) 'Emergence versus selforganisation: different concepts but promising when combined', in Brueckner, S.A., Di Marzo Serugendo, G., Karageorgos, A. and Nagpal, R. (Eds.): *Engineering Self-Organising Systems*, LNAI, Vol. 3464, pp.1–15, Springer, Berlin Heidelberg.

De Wolf, T. and Holvoet, T. (2007) 'A taxonomy for self-* properties in decentralised autonomic computing', in Parashar, M. and Hariri, S. (Eds.): *Autonomic Computing: Concepts, Infrastructure, and Applications*, pp.101–120, CRC Press, Taylor and Francis Group.

Delic, K. and Dum, R. (2006) 'On the emerging future of complexity sciences', *ACM Ubiquity*, Vol. 7, No. 10, p.1.

Di Marzo Serugendo, G. (2009) 'Robustness and dependability of self-organising systems – a safety engineering perspective', in *Int. Symp. on Stabilization, Safety, and Security of Distributed Systems (SSS)*, LNCS, Vol. 5873, pp.254–268, Springer, Berlin Heidelberg, Lyon, France.

Di Marzo Serugendo, G., Fitzgerald, J. and Romanovsky, A. (2010) 'Metaself – an architecture and development method for dependable self-* systems', in *Symp. on Applied Computing (SAC)*, pp.457–461, Sion, Switzerland.

Di Marzo Serugendo, G., Fitzgerald, J., Romanovsky, A. and Guelfi, N. (2006a) 'Dependable self-organising software architectures – an approach for self-managing systems', Tech. rep., BBKCS-05-06, School of Computer Science and Information Systems, Birkbeck College, London, UK.

Di Marzo Serugendo, G., Gleizes, M-P. and Karageorgos, A. (2006b) 'Self-organisation and emergence in MAS: an overview', *Informatica*, Vol. 30, pp.45–54.

Di Marzo Serugendo, G., Fitzgerald, J., Romanovsky, A. and Guelfi, N. (2007) 'A metadata-based architectural model for dynamically resilient systems', in *ACM Symposium on Applied Computing (SAC)*, pp.566–573, ACM, Seoul, Korea.

Di Marzo Serugendo, G., Gleizes, M. and Karageorgos, A. (2005) 'Self-organization in multi-agent systems', *Knowledge Engineering Review*, Vol. 20, No. 2, pp.165–189.

Edmonds, B. (1999) 'What is complexity? – the philosophy of complexity per se with application to some examples in evolution', in Heylighen, F. and Aerts, D. (Eds.): *The Evolution of Complexity*, Kluwer, Dordrecht.

Frei, R. (2010) 'Self-organisation in evolvable assembly systems', PhD thesis, Department of Electrical Engineering, Faculty of Science and Technology, Universidade Nova de Lisboa, Portugal.

Frei, R. and Barata, J. (2010) 'Distributed systems – from natural to engineered: three phases of inspiration by nature', *Int. J. of Bio-inspired Computation*, Vol. 2, Nos. 3/4, pp.258–270.

Frei, R. and Di Marzo Serugendo, G. (2011) 'Advances in complexity engineering', *Int. J. of Bio-inspired Computation*, Vol. 1, No. 1, pp.11–22.

Fromm, J. (2005) 'Ten questions about emergence', http://arxiv.org/abs/nlin/0509049.

Gell-Mann, M. (1995) 'What is complexity?', in *Complexity*, Vol. 1, John Wiley and Sons, Inc., New York, USA.

Gershenson, C. (2007) 'Design and control of self-organizing systems', PhD thesis, Faculty of Science and Center Leo Apostel for Interdisciplinary Studies, Vrije Universiteit, Brussels, Belgium.

Gladwell, M. (2000) 'The tipping point: how little things can make a big difference', *Abacus*, London, UK.

Gleick, J. (1987)*Chaos*, Vintage, London.

Grobbelaar, S. and Ulieru, M. (2007) 'Complex networks as control paradigm for complex systems', in *IEEE Int. Conf. on Systems Man and Cybernetics (SMC)*, pp.4069–4074, Montreal, Canada.

Halloy, J., Sempo, G., Caprari, G., Rivault, C., Asadpour, M., Tache, F., Said, I., Durier, V., Canonge, S., Ame, J., Detrain, C., Correll, N., Martinoli, A., Mondada, F., Siegwart, R. and Deneubourg, J-L. (2007) 'Social integration of robots into groups of cockroaches to control self-organized choices', *Science*, Vol. 318, No. 5853, pp.1155–1158.

Heylighen, F. (1996) 'What is complexity?', available at http://pespmc1.vub.ac.be/COMPLEXI.html.

Heylighen, F. (2003) 'The science of self-organization and adaptivity', in Kiel, E. (Ed.): *The Encyclopedia of Life Support Systems, Knowledge Management, Organizational Intelligence and Learning, and Complexity*, EOLSS Publishers, Oxford, UK.

Heylighen, F. (2008) 'Complexity and self-organisation', in Bates, M.J. and Maack, M.N. (Eds.): *Encyclopedia of Library and Information Sciences*, Taylor & Francis.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems*, MIT Press, Cambridge, MA, USA.

Holland, J. (1992) *Adaptation in Natural and Artificial Systems: an Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, Cambridge, MA, USA.

Holland, J. (1995) *Hidden Order: How Adaptation Builds Complexity*, Addison-Wesley, Reading, MA, USA.

Holland, J. (1998) *Emergence – From Chaos to Order*, Oxford University Press, Oxford, UK.

Hongler, M-O. (1994) *Chaotic and Stochastic Behaviour in Automatic Production Lines*, Springer Berlin Heidelberg.

Johnson, C. (2005) 'What are emergent properties and how do they affect the engineering of complex systems?', *Reliability Engineering and System Safety*, Vol. 91, No. 12, pp.1475–1481.

Jost, J. (2004) 'External and internal complexity of complex adaptive systems', *Theory in Biosciences*, Vol. 123, No. 1, pp.69–88.

Kauffmann, S. (1995) *At Home in the Universe: The Search for the Laws of Self-organization and Complexity*, Oxford University Press, Oxford, UK.

Kenger, P. (2006) 'Module property verification – a method to plan and perform verifications in modular architectures', PhD thesis, Department of Production Engineering, Royal Institute of Technology (KTH), Stockholm, Sweden.

Kuras, M. (2006) 'Complex-system engineering', http://cs.calstatela.edu/wiki/images/c/c5/Kuras.pdf.

Kuzgunkaya, O. and ElMaraghy, H. (2006) 'Assessing the structural complexity of manufacturing systems configurations', *Int. J. of Flexible Manufacturing Systems*, Vol. 18, No. 2, pp.145–171(27).

Langton, C.G. (1986) 'Studying artificial life with cellular automata', in (Farmer, D., Lapedes, A., Packard, N. and Wendroff, B. (Eds.): *5th Annual Conf. of the Center for Nonlinear Studiesi*, Los Alamos; *Evolution, Games and Learning: Models for Adaptation in Machines and Nature*, pp.120–149, Amsterdam, The Netherlands.

Louzoun, Y. and Atlan, H. (2007) 'The emergence of goals in a self-organizing network: a non-mentalist model of intentional actions', *Neural Networks*, Vol. 20, No. 2, pp.156–171.

Lucas, C. (2008) 'The complexity & artificial life research concept for self-organizing systems', available at http://www.calresco.org.

Maguire, S. and McKelvey, B. (1999) 'Complexity & management: moving from fad to firm foundations', *Emergence: A J. of Complexity Issues in Organizations & Management*, Vol. 1, No. 2, pp.19–61.

Mitchell, M. (2006) 'Complex systems: network thinking', in Working papers, available at http://www.santafe.edu/research/publications/workingpapers/06-10-036.pdf, Sante Fé Institute, NM, USA.

Muehl, G., Werner, M., Jaeger, M., Herrmann, K. and Parzyjegla, H. (2007) 'On the definitions of self-managing and self-organizing systems', in Braun, T., Carle, G. and Stiller, B. (Eds.): *KiVS 2007 Workshop: Selbstorganisierende, Adaptive, Kontextsensitive Verteilte Systeme (SAKS)*, pp.291–301, Bern, Switzerland.

Mueller-Schloer, C. (2004) 'Organic computing: on the feasibility of controlled emergence', in *2nd IEEE/ACM/IFIP Int. Conf. on Hardware/Software Co-Design and System Synthesis CODES+ISSS*, pp.2–5, ACM, New York, USA.

Mueller-Schloer, C. and Sick, B. (2008) 'Controlled emergence and self-organization', in Wuertz, R. (Ed.): *Organic Computing, Understanding Complex Systems*, pp.81–104, Springer Berlin Heidelberg.

Newman, D. (1996) 'Emergence and strange attractors', *Philosophy of Science*, Vol. 63, No. 2, pp.245–261.

Nicolis, G. and Prigogine, I. (1977) *Self-organization in Non-Equilibrium Systems: From Dissipative Structures to Order through Fluctuations*, J. Wiley & Sons, New York.

Norman, D. and Kuras, M. (2004) 'Engineering complex systems', available at http://www.mitre.org.

Oliver, D., Kelliher, T. and Keegan, J. (1997) *Engineering Complex Systems with Models and Objects*, McGraw-Hill, New York.

Parunak, H. and VanderBok, R. (1997) 'Managing emergent behaviour in distributed control systems', in *Instrument Society of America (ISA-Tech)*, p.97, Anaheim, Canada.

Philipp, T., Boese, F. and Windt, K. (2006) 'Autonomously controlled processes – characterisation of complex production systems', in *3rd Int. CIRP Conf. on Digital Enterprise Technology (DET)*, Setubal, Portugal.

Puviani, M., Di Marzo Serugendo, G., Frei, R. and Cabri, G. (2010) 'A method fragments approach to methodologies for engineering self-organising systems', Submitted to *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*.

Ranjan, P., Kumara, S., Surana, A., Manikonda, V., Greaves, M. and Peng, W. (2003) 'Decision making in logistics: a chaos theory based analysis', *CIRP Annals – Manufacturing Technology*, Vol. 52, No. 1, pp.381–384.

Reeves, G. and Fraser, S. (2009) 'Biological systems from an engineer's point of view', *PLoS Biology*, Vol. 7, No. 1, pp.32–35.

Rouse, W. (2003) 'Engineering complex systms: implications for research in systems engineering', *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, Vol. 33, No. 2, pp.154–156.

Ryu, K., Yucësan, E. and Jung, M. (2006) 'Dynamic restructuring process for self-reconfiguration in the fractal manufacturing system', *Int. J. of Production Research*, Vol. 44, No. 15, pp.3105–3129.

Rzevski, G. (2004) 'Designing complex engineering systems', in *Volga Conf. on Complex Adaptive Systems*, Keynote paper, Samara, Russia.

Rzevski, G. and Skobelev, P. (2007) 'Emergent intelligence in multi-agent systems', Tech. rep., Magenta Technology.

Schuh, G., Sauer, A. and Dring, S. (2006) 'Modeling collaborations as complex systems', in *4th Int. Industrial Simulation Conf. (ISC)*, pp.168–174, Palermo, Italy.

Sole, R., Ferrer-Cancho, R., Montoya, J. and Valverde, S. (2002) 'Selection, tinkering and emergence in complex networks', *Complexity*, Vol. 8, No. 1, pp.20–31.

Spilker, H. (2007) 'Werden Roboter zur Gefahr für die Menschen? Interview with Prof. Alois Knoll', *Technology Review*, June, p.106.

Steels, L. (1991) 'Towards a theory of emergent functionality', in Meyer, J-A. and Wilson, S. (Eds.): *From Animals to Animats: 1st Int. Conf. on Simulation of Adaptive Behaviour*, pp.451–461, Paris, France.

Ulieru, M. and Doursat, R. (2010) 'Emergent engineering: a radical paradigm shift', *J. of Autonomous and Adaptive Communications Systems*.

van Eijnatten, F. (2005) 'Methodological aspects of chaos and complexity in organisation and management', Tech. rep., Eindhoven University of Technology, The Netherlands.

Waldrop, M. (1992) *Complexity*, Simon & Schuster Paperbacks, New York, USA.

Webb, C. and Lettice, F. (2005) 'Performance measurement, intangibles, and six complexity science principles', in *3rd Int. Conf. on Manufacturing Research (ICMR)*, Cranfield, UK.

Wolfram, S. (1986) 'Approaches to complexity engineering', *Physica.*, Vol. D, No. 22, pp.385–399.

Wolfram, S. (2002) *A New Kind of Science*, Media, Champaign, IL, USA.

Woodard, C. (2006) 'Architectural strategy and design evolution in complex engineered systems', PhD thesis, Business Studies Department, Harvard Univ., Cambridge, MA, USA.

Zapf, M. and Weise, T. (2007) 'Offine emergence engineering for agent societies', in *Proc. of the Fifth European Workshop on Multi-Agent Systems EUMAS'07*, Hammamet, Tunesia.

Notes

1 In this section, the term *specification* is not used with its meaning in computer science, but rather its meaning in manufacturing engineering; a specification is a description of the identified requirements.

2 Another way of expressing this is: "The behaviour of a complex system will be a combination of pre-set objectives and constraints as defined by the system developer, and adaptive *islands* where the system is allowed to make its own decisions" (Mueller-Schloer and Sick, 2008).

3 Safe synthetic biology, see http://www.synbiosafe.eu

4 Explanations from the *Biotechnology* lectures (academic year 2004–2005) by Prof. Florian Wurm at the Swiss Federal Institute in Lausanne, Switzerland

5 *Multi-lateral* describes a relationship with several peers at the same time.