The University Of Sheffield.

Disciplined Exploitation of Emergent Properties

Konstantinos Rousis

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Philosophy

The University of Sheffield
Faculty of Engineering
Department of Computer Science

November 2016

Abstract

Digital systems are becoming increasingly complex, requiring significantly more effort and resources in order to be designed, implemented, and maintained. In the last decade, industry and academia alike share the concern that in the near future engineers will have to face unprecedented levels of complexity. Similarly, the belief that traditional engineering approaches will be insufficient for coping with systems of such complexity is gaining increasingly more supporters. An alternative approach suggests the use of implicit engineering techniques which could lead to complex global-level behaviours by focusing solely on the local, or individual, level. The quality of rising macroscopic behaviours which are irreducible, or non-trivial to reduce, to any microscopic properties is more widely known as emergence, especially in the fields of complex and multi-agent systems.

This work aims to investigate the possibility of engineering systems which harness, in intentional and disciplined ways, beneficial emergent properties. An experimental framework is being proposed to assist system designers towards that goal. This framework is based on the results and experience gained by Paunovski during the design of the Emergent Distributed Bio-Organisation (EDBO) case study. EDBO has demonstrated a number of beneficial emergent properties rising out of simple, bio-inspired, local interactions. The original implementation of the EDBO case study is closely coupled with a custom simulation platform; both developed by the same author. This

work provides a basis for separating the EDBO case study from this combined implementation, by documenting it concisely and defining it in a formal manner. This formal model allows for rigorous testing and enables other authors to reuse the EDBO principles in their systems. The model is validated informally through animation and it serves as the basis of an independent implementation which cross-validated many of EDBO's original findings.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr George Eleftherakis and Dr Anthony J. Cowling, for their continuous support during my research and the writing of this thesis. There is no doubt that this work would not have been possible, nor would I be the person I am today, without them.

My heartfelt thanks goes also to the rest of my thesis committee — Dr Ioanna Stamatopoulou, Dr Ilias Sakellariou and Professor Panayotis Ketikidis — for their thorough review, insightful comments and ability to point out new and exciting directions for future work. Special thanks to Dr Ognen Paunovski for inspiring and laying the foundation for a significant part of this work.

This thesis would not have been possible without the support of the South East European Research Centre (SEERC). My thanks to Nikos Zaharis, Athanasia Halatzouka and Eleni Tsimiga.

I would also like to thank the whole Computer Science Department of the International Faculty of the University of Sheffield, City College — Anna, Dimitris, George, Ioanna, Iraklis, Kostas, Panos, Petros, Sofia, Thanos and Thomas — for all their support, encouragement and inspiration during my undergraduate and postgraduate studies.

Finally, the biggest thanks goes to my better half, Isidora, and to my family — Efi, Rena, Emilios, Sofia, Sandro, Anna, Efthychia and Kostas — not only for putting up with me during the more stressful periods of this work, but also for believing in me when I was not.

So long, and thanks for all the fish!

Table of Contents

List of Tables

List of Figures

1 Introduction

The advent of the Internet introduced an exponential increase in technology's penetration to everyday life. While less than three decades ago digital networks could only be found in the military, large universities and research organisations, nowadays the vast majority of the households in developed countries are connected to the Internet. Recent advances in mobile devices, such as laptops, netbooks, tablets, and smartphones, only increased the number of interconnected nodes per household or person. Moving further, the first house appliances which are able to connect to the Internet and perform online services on behalf of their users have long since hit the market.

From a digital-networks perspective, the world can be viewed as a huge overlay network consisting of billions of nodes interacting through a shared channel: the Internet. Yesterday's emerging trends, such as the mobile Internet and the Internet of Things, have become today's norms. This results to a constant increase of complexity: devices have to be managed, maintained, adapted, re-configured, and so on. Classical engineering has successfully tackled these issues for many decades but as the complexity increases so does the effort put into managing this complexity. The main concern is that, eventually, a turning point will be reached in which the complexity will grow beyond the point of being practically manageable with the traditional engineering processes that have been developed so far.

Towards this direction, a number of alternative theories and approaches have been suggested by both industry and academia. IBM acknowledged this threat and proposed *autonomic computing* as a possible solution. Universities and research institutions focused on complexity management by offering graduate-level programmes in the renewed field of *complexity science*. The European Commission has also acknowledged the issue and further supported research and applications towards complexity management with two subsequent calls under the "Complex Systems" initiative in 2003 and 2005; resulting to the funding of numerous STREP, IP, and FET projects.

There is a common and central element in discussions of complexity science and complex systems: *emergence*. A lot of effort has been put by academia on exploring the concept of emergence and how simple interactions at a lower level of organisation can yield complex phenomena at a higher level. Thorough observation of natural systems has demonstrated that the process of emergence is accountable for various biological properties such as survivability and adaptability. Turning to nature for inspiration, researchers have been able to reproduce these processes in their engineered, artificial, systems and yield similar favourable properties such as self-adaptability, self-healing, and other *self-** properties.

The author shares the belief that a point will soon be reached where an explicit design and implementation of complex systems will become infeasible. At the same time, it is apparent that harnessing the power of emergence and incorporating it in a disciplined and intentional manner into system design and engineering could greatly assist with tackling the complexity. This work suggests the investigation of alternative and implicit engineering techniques which could

allow for the rise of complex macroscopic behaviours by focusing solely on the microscopic level. The aim of this work is to identify *disciplined ways*, for example in the form of frameworks, processes, and design patterns, which could facilitate the *intentional engineering* of complex global behaviours by allowing engineers to focus on the local level.

Directing, controlling (even partially), or in the extreme case *engineering emergence*, however, could be extremely hard and even oxymoronic according to most definitions of emergence. It is essential to form a thorough understanding of emergence and its intrinsic issues before exploring any such possibility. Being a topic of great debate among researchers, there is a multitude of proposed definitions of emergence, spanning different disciplines, but a universally accepted one is still missing. In order to proceed it is necessary to examine the proposed definitions from an engineering perspective and conclude with a working definition which can serve as the basis for the rest of this work.

The last decade has seen various scientific attempts at controlling and engineering emergence. While these approaches may differ on many aspects, they invariably use simulation as a means of proving (or disproving) their hypotheses regarding engineering emergence. One has to carefully model the desired system, with the right level of abstraction and refinement, in order to allow for the desired processes and properties to emerge. Consequently, it is deemed essential that an adequate simulation platform is used in order to allow for the experimentation with different case studies and hypotheses which could potentially lead to disciplined ways of exploiting emergent properties.

Another common point in these approaches is the focus on specific application domains and case studies. At this point it seems impossible to make generic claims about engineering emergence and one has to narrow their focus in order to draw any meaningful conclusions. In this direction, Paunovski has focused on the field of *Artificial Distributed Systems* (ADS) and more specifically on the *Emergent Distributed Bio-Organisation* (EDBO) case study[2].

EDBO simulates an overlay peer-to-peer network where each node, referred to as "biobot", participates in a distributed query forwarding mechanism. Biobots act on behalf of their users: when a user initiates a query the biobot tries to satisfy it either directly, if it already offers the requested service, or by forwarding it to its neighbours until it is eventually matched or it expires.

The key differentiation between EDBO and regular peer-to-peer networks is the introduction of energy values at the biobot level. When these values reach certain thresholds, the biobots are allowed to perform different biologically-inspired functions. Biobots receive energy rewards when they contribute to the success of the network; for example by participating in a successful query match. Likewise, they incur energy penalties for communicating with their neighbours or performing bio-inspired actions such as replicating or pairing with other biobots as a means of sexual reproduction. When a biobot loses all of its energy it "dies".

The EDBO case study has already demonstrated its potential at engineering emergent properties [2, 8]. By following an iterative and experimental, yet disciplined, framework for engineering emergence in ADS [7], simulations with EDBO resulted in the emergence of various beneficial behaviours. Introducing biologically-inspired functions

and energy as part of the node's design, allowed for the emergence of complex global properties.

The major flaw of EDBO is that the model is tied to its own, custom developed, simulation platform. As such, there is no formal or even semi-formal description of the EDBO model available. This work devises a formal specification of the EDBO by using the X-machine formalism. This specification is decoupled from any implementation details and it allows for easier reuse of the EDBO model. The model is informally validated by using an independent animator available in the X-machine ecosystem. Moreover, confidence to the model is further increased by a separate implementation of the EDBO core principles in a different application domain, which was based on this same formal model. The next section presents this work's contributions in more detail.

1.1 Contributions

This section presents the main contributions of this work. For those already published in scientific papers, citations are included.

1. *Establish a working definition of emergence that assists towards the quest of engineering emergence.* As Chapters 2 and 3 detail, the scientific community has not reached a consensus on the definition and the intricacies of emergence. There is a large number of different and often conflicting definitions available in the literature. Section 4.1 suggests the adoption of Wolf and Holvoet's [9] definition as a suitable one in the context of computer science as well as towards harnessing beneficial emergent properties in dig-

ital systems. Work presented in Chapters 2, 3, and 4 have been published in [10] and [11].

2. *Provide a disciplined way of exploiting emergent phenomena in artificial distributed systems.* Chapter 4 presents an experimental framework for designing distributed systems that benefit from emergent phenomena. Work presented in that chapter has been published in [12].
3. *Devise a formal model of the EDBO.* EDBO was introduced in Paunovski's PhD thesis [2] with very promising results. However, many of the details were only documented in the implementation of the model and the simulation platform itself. After an extensive study of this combined implementation, a more concise and implementation-independent description was formed. This is presented in parts of Chapters 5 and 6 and has been published in [8]. Based on that, a formal model of the EDBO was constructed by employing the X-machine formalism. This contributed to a better understanding of the original model. More importantly, it enables other researchers and practitioners to understand and reuse the EDBO principles in their own systems. It further paved the road for independent simulation studies on the EDBO case study which allows for cross-validation of the original findings. The formal model is presented in Chapter 6 and has been published in [13].
4. *Improve confidence to EDBO's original results and cross-validate some of the original findings.* The formal model described in the previous contribution has been informally validated through animation techniques, increasing the understanding of the original model as well as confidence on that the devised formal model is

at least partially equivalent to the original. Furthermore, a new implementation of the EDBO in a different application domain was based on this same formal model and it resulted to many of the original findings of the EDBO case study, providing some form of cross-validation. This work has been published in [3] and is presented in Chapter 7.

1.2 Structure

Chapter 2 provides an in-depth literature review on emergence and Chapter 3 presents various attempts at utilising emergent phenomena and incorporating them into engineering processes. In Chapter 4, emergence and the recurring topics surrounding it are discussed from an engineering perspective and a working definition is selected. The author's approach to engineering emergence is also presented along with his contributions on harnessing emergent properties in distributed systems.

Chapter 5 presents the EDBO simulation platform which has been used in previous work and identifies as its main weakness the fact that its implementation is tied together to the EDBO model itself. FLAME is suggested as an alternative, general-purpose simulation platform.

Chapter 6 provides a formal definition of the EDBO model by using the X-machine formalism and Chapter 7 validates this model informally, though animation. Furthermore, it presents a separate implementation of EDBO that was based on the same formal model and has demonstrated similar findings to those of the original EDBO

case study. Finally, Chapter 8 summarizes this work and presents its conclusions along with possible future directions for this work.

2 Understanding Emergence

In the complex systems and multi-agent systems fields emergence typically refers to a *global* (or macroscopic) level behaviour which is either impossible or highly unlikely to be predicted by observing the behaviour of the individual entities at the *local* (or microscopic) level. A typical example is the flocking behaviour of birds, whereby birds fly in a lockstep formation and by analysing the behaviour of each individual bird it would be rather unlikely to predict such a global-level result. Another example is the ability of ants to establish the shortest possible route between their nest and the various sources of food, even when the environment changes, while the behaviour of each ant (i.e. the local or micro level) is rather simplistic and by simply observing it, it would be hard to predict such a global behaviour a priori.

Lexically, *emerge* is derived from the Latin word *emergere* and according to the Oxford English Dictionary is defined as something moving out of something else and becoming visible or apparent. The same source defines *emergence* as the action or process of emerging while *emergent*, in a philosophic context, is defined as a property which arises due to complex causes and can not be analysed solely on its own or by its effects.

This last, philosophic, definition of emergence can be traced back to Aristotle who at the 4^{th} century BC wrote [14]:

> In the case of all things which have several parts and in which the totality is not, as it were, a mere heap, but the whole is something beside the parts, there is a cause [...]

This excerpt is commonly interpreted in the literature as "the whole is greater than the sum of the parts". More than two millennia after, the father of modern economics, Adam Smith, wrote in his seminal work, *The Wealth of Nations* [15]:

> He [the individual] generally, indeed, neither intends to promote the public interest, nor knows how much he is promoting it. By preferring the support of domestic to that of foreign industry, he intends only his own security; and by directing that industry in such a manner as its produce may be of the greatest value, he intends only his own gain, and he is in this, as in many other cases, led by an invisible hand to promote an end which was no part of his intention.

What Smith states is that although individuals may act solely in self-interest, a certain order, which he names *the invisible hand*, is emerging. There is, thus, a possibly positive effect which can be observed at a global level, i.e. the society, and cannot be deduced by observing the micro level, that is each individual.

Emergence is very commonly found in discussions of *systems theory* which take an *anti-reductionism* and *holistic* stance. Reductionism is the belief that a system's behaviour can be completely predicted and understood by analysing and observing its components; anti-reductionism being the exact opposite. Holism, being the opposite of atomism, refers to the theory that some wholes are greater than the sum of their parts or that these parts are explicable only by reference to the whole [16], a definition similar to the one given by Aristotle [14].

Although these definitions are sufficient for providing a basic understanding of emergence as a concept, they are mostly inadequate for approaching emergence as a scientific or engineering subject. Recent years have brought a renewed research interest on emergence, mostly from the communities of complex and multi-agent systems. However, to this day, a universally accepted definition of emergence is still lacking.

The rest of this section presents various attempts to define emergence and tackle the intricate topics inherently related to it such as *novelty, models, levels, surprise*, and *observation*. Following, different emergence classification schemes that have been suggested in the literature are presented.

2.1 Different Perspectives and Definitions of Emergence

Holland famously wrote that *it is unlikely that a topic as complicated as emergence will submit meekly to a concise definition* [17]. While this, to this day, is true, one can gain great insight into emergence by reviewing the various definitions available in the literature, spanning different decades, sciences and disciplines. Figure 1, reprinted from [4], gives an insight into the mathematical and scientific roots of emergence.

Lewes, in 1875, proposed the use of the term *emergent* as an antonym to *resultant*. While a resultant is either the sum or the difference of two or more co-operant forces, an emergent encompasses any result which does not satisfy the aforementioned property [18]. Anderson, nearly a hundred years afterwards, managed to renew

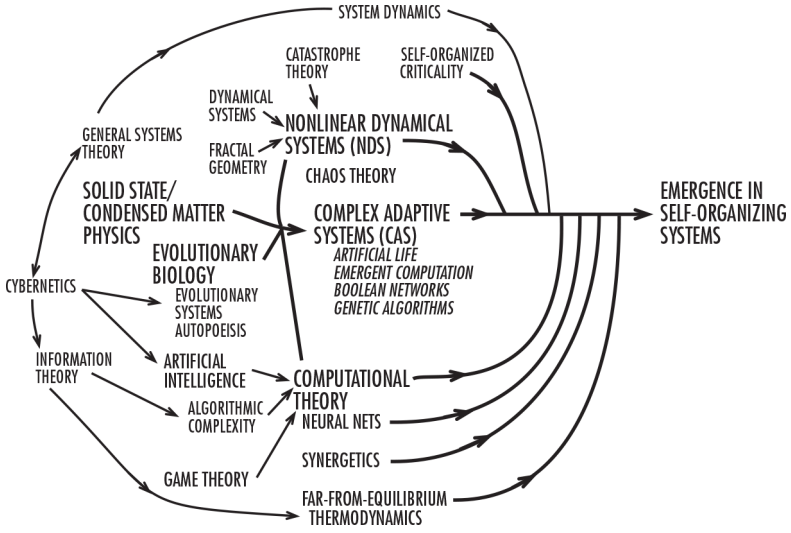


Fig. 1: Mathematical and scientific roots of emergence. Reprinted from [4].

scientific interest in emergence with his claim that the whole is not only more than the sum of its parts (the original notion of emergence according to Aristotle [14]), but it is *different* [19]. Drawing a multitude of examples from molecular biology to psychology and social sciences, Anderson describes how global properties can emerge that are not deducible from the properties and behaviours of a system's local components.

Rosen described emergence in terms of models, defining a system's behaviour as emergent if it cannot be explained anymore by the model which described the system so far [20]. This idea was later named by Cariani as *"emergence relative to a model"* [21], describing how any functional discrepancies between the system's model and the actual behaviour during execution can be classified as emergent.

Heylighen also states that emergence is a process which cannot be captured by a *fixed model* of a system [22]. Following a state-based

approach, Heylighen uses the term model as a construct capable of capturing all possible states of a system as well as the relational constraints which define which states can be reached under different conditions. For example, the movement of a car can be typically modelled with a dynamic state space dimensionality of five: three variables to model its kinematics state, that is its coordinates in the three-dimensional space, one variable for its forward velocity and another one for its yaw rate (accounting for steering). These five variables are sufficient to determine any possible state of the car model given that the dimensions of the state space are bound to be constant. If, on the other hand, the car was somehow to be split into two parts, the state space dimensionality would be immediately doubled. The original model thus would become insufficient for modelling the system after the event of splitting.

A more realistic example is crystallization where the dissolved molecules move independently of each other causing their state space to be the product of the state spaces of all molecules which could be virtually infinite [22]. In this scenario the initial model of the system is deemed irrelevant after an emergent phenomenon occurs (i.e. the crystallization). Heylighen proposes the use of a meta-model, which also consists of states and transition rules, capable of capturing appropriately systems which exhibit emergence. In this meta-model, each state corresponds to a simple model of the system for a given time instance during its evolution. For example one state could correspond to a model describing the car before the splitting event and another state to a model describing it after that. Transition rules in the meta-model determine the shift from one model to another.

Holland proposes a subtle change in the classical reductionism approach by contradicting the notion "that all phenomena in the universe are reducible to the laws of physics" with "all phenomena are *constrained* by the laws of physics" [17]. The latter alternative allows for the recognition of emergent phenomena even if these are the result of a relatively small set of components, governed by simple rules. Holland further claims that emergence must be the result of self-organisation and not of a centralised coordination and control [17]; an opinion Corning [23] seems to reject emphatically by arguing that self-organisation has become an academic buzzword which is often used uncritically.

Bar-Yam takes an anti-reductionism stance as well but he makes another important realisation: although it is impossible to study emergent phenomena by isolating a system's parts and studying them separately, on their own (i.e. anti-reductionism), it is possible to study them by studying each part under the context of the system as a whole [24]. Supporting his statement he provides an example drawn from the field of neural networks:

> ...there are synapses between each neuron and every other neuron. If we remove a small part of the network and look at its properties, then the number of synapses that a neuron is left with in this small part is only a small fraction of the number of synapses it started with. If there are more than a few patterns stored, then when we cut out the small part of the network it loses the ability to remember any of the patterns, even the part which would be represented by the neurons contained in this part.

The important distinction BarYam makes is that the emergent behaviour is not absent in the behaviour of the individual parts (i.e.

classic anti-reductionism) but rather it is *not readily observable*. If, however, the individual parts are to be studied within the context they are found, the collective behaviour can then be observed as a part of the individual components [24].

Goldstein defines emergence as *"the arising of novel and coherent structures, patterns, and properties during the process of self-organisation in complex systems"* [4] while he emphasises the distinction between the macroscopic level (in which the emergent phenomena are conceptualised) and the microscopic level (from whose components such phenomena arise). Goldstein attributes the following properties to any emergent phenomenon [4]:

- *radical novelty:* the emergent phenomenon should be novel regarding to the system and its parts and not deducible or reducible to the system's components.
- *coherence or correlation:* emergents tend to appear as integrated wholes (at the macro level).
- *global or macro level:* the emergent phenomena should be observable at a global or macroscopic level as opposed to the local or microscopic level in which the components lie.
- *dynamical:* emergents cannot be characterised as static properties of a system but rather they evolve, dynamically, as the system evolves over time.
- *ostensive:* emergents should be perceivable.

The latter point emphasises the need for an observer which plays a central role in emergence and has been the subject of great debates among researchers. Ronald, Sipper, and Capcarrere [25] are consistent with Goldstein's view of the need for an observer and taking

the notion a step further they devised a scheme for testing whether a system is actually exhibiting emergent phenomena or not. The scheme requires a *designer* who designs a system in a language L_1 and an *observer*, fully aware of the design, who describes the observed macro-level system behaviour in a different language L_2. The condition for classifying the system as capable of exhibiting emergence is the element of *surprise*, that is, the observer should be surprised by the global behaviour, documented in L_2, denoting thus the existence of *non-obvious* causal links to the system's design in L_1. Far from formal, the definition of *"design, observation, surprise!"* has been the subject of debates, especially regarding the element of surprise, how this could be formally defined and whether once the observer has gained enough insight into the causal relationships of the system, as to cease the element of surprise, the phenomenon can still be classified as emergent or not [17, 26–30, 6].

The definition of Ronald et al. is in line with Goldstein's requirement for an observer as expressed by his *ostensiveness* requirement [4]. Crutchfield shares this view regarding the necessity for an observer by claiming that the process of detecting emergence is subjective and is restricted to the observer's computational resources [31]. As an example, Crutchfield discusses the case of a self-avoidance random walking algorithm and how similarity patterns can arise from the paths traced by its execution, although the path forming process involves a lot of stochasticity. Crutchfield claims that the emergence in this scenario is in the eye of the observer whose initial predictions of the system might have failed.

Corning, for that matter, disqualifies the need of an observer and he claims that an emergent phenomenon is real and measurable

regardless of whether someone is observing it or not [23]. Fromm shares Corning's opinion and moving a step further, he rejects both the need for the element of surprise and novelty as prerequisites for defining emergence [26]. Kubík opposes the surprise element as well by arguing that it can be misleading and obscure better explanations [27]. Nicely put, *"the moment of surprise can fade away once sufficient information is provided"* and thus he suggests, towards an attempt to provide a more formal definition of emergence, to *"ignore surprise"* altogether. Holland also believes that surprise is not an essential factor for classifying a behaviour as emergent [17] as does Damper [28]:

> What might be surprising on first acquaintance or at a particular stage of scientific knowledge tends to become commonplace, trite or predictable after intensive, lengthy study.

A more formal substitute for the surprise characteristic is the notion of the *gap of complexity* provided by Deguet and Demazeau [29]. The authors agree with Ronald's distinction (and therefore the need) of a designer and an observer but following Holland's paradigm of emergence, "much from little" [17], they devise a complexity metric which is used to prove whether a system exhibits this kind of emergent behaviour. More specifically, they consider "little" as a simple system, S, "much" as a complex phenomenon *phi* that this system exhibits and they classify a feature as emergent if the following equation results in a positive outcome:

$$eFeat = C(\varphi) - C(S) \geq 0$$

In the formula above, C is an abstract complexity measure. For obvious reasons the same complexity metric cannot be applied to

both a phenomenon (*phi*) and a system (S), thus, the authors associate a computational task to each of them [29]. The authors, however, omit to provide any practical example of a system to which they apply their proposed formula. Kubík attempts to provide a formal definition of emergence as well but firstly he suggests an informal definition of *basic emergence* [27]:

> By basic emergence, we mean behavior reducible to agent-to-agent interactions without any evolutionary processes involved (i.e., an agent's behavioral set stays the same during the modeling and the analysis of the system). The environment has no rules of behavior and is changed only by the actions of agents.

Subsequently, he proceeds to formalising this notion by modelling the various concepts involved (such as a MAS, an agent, agents' behaviour) with formal constructs (grammars) and testing the proposed formal framework with the well studied emergence example of gliders[1]. Gordon opposes in principle the need for an external observer by making a more philosophical point that if emergence requires an observer, *"A relation is implied that humans are to artificial life as God is to real life. This precludes a unified treatment of life and artificial life."* [30]. Instead, he suggests that both the designer and the observer should be an integral part of the system under simulation, they should emerge as well, and their emergence should also be simulated. Gordon claims that this can be accomplished by adding the elements of *evolution, embryology, and physics* to Ronald's [25] "Design, Observation, Surprise!" original process as such [30]:

[1] Gliders are spaceship-like patterns which travel across the board of Conway's Game of Life. Gliders can appear even in simulations with a random initial configuration.

1. At a given time, a species is defined by construction rules for its individuals, by which the genotype is translated to a phenotype via genetics and physics.
2. The ability to observe is part of an individual's construction.
3. Surprise! occurs when gene duplication is followed by (viable, heritable, survivable) mutations in subsequent generations that construct new capabilities (morphological, behavioral, or observational).

It is apparent that emergence has become a term used by a diverse set of science and engineering disciplines with a multitude of proposed definitions. Even within the scope of a specific domain, such as complex systems, there is no universally accepted definition. Nonetheless, there is an acknowledged need for classifying different types of emergent processes and phenomena which is the topic of the next section.

2.2 Categorisation and Classification of Emergent Phenomena

Undoubtedly, the most common distinction found in the literature is the one between *weak* and *strong emergence*. The distinguishing attribute is whether a global property characterised as emergent is reducible (even if the reduction is non-trivial) or irreducible (in principle) to the local parts [32]. In the first case the emergent phenomenon is classified as weakly (or reductionistic) emergent while in the latter as strongly (or holistic) emergent. Countless examples

fall under the category of weak emergence, depending on the specific definition used, such as birds' flocking behaviour, various abilities of living organisms (e.g. breathing or reproducing), molecule crystallisation, and many others. Strong emergence, on the other hand, is still treated mostly philosophically and actual examples are lacking; for that matter some authors completely doubt the existence of strong emergent phenomena.

Stephan had originally distinguished among three types of emergence which coincide with the distinction of weak and strong emergence but they were named differently: *weak emergentism*, *diachronic emergentism*, and *synchronic emergentism* [33, 34]. Stephan's weak emergentism coincides with Chalmers [32] notion of weak emergence. More specifically, Stephan defines any systemic property as weakly emergent, that is, any property possessed by the system but not by the individual parts. The various abilities of living organisms such as breathing, walking, and reproducing are a typical example of this kind of properties.

Diachronic and synchronic emergentisms are stronger versions of emergence, based upon the notion of weak emergentism. A diachronic emergent property is one that is novel and cannot be predicted before its first instantiation. Synchronic emergentism includes properties which cannot, in principle, be reduced to the system's parts and their interactions; this view is in accordance with Chalmers' view on strong emergence [32] as well. It can be deduced that synchronic properties are at the same time diachronic as well, but not vice versa.

Nonetheless, as already mentioned, discussions around strong emergence are rather philosophic and the literature is still lacking actual examples. Chalmers [32] takes a cautious stance, avoiding to

make any definite claims regarding whether strong emergence exists at all. He believes, however, that the phenomenon of consciousness is a case of strong emergence as facts about consciousness are not deducible from physical facts alone [32, 35]. To further support his view, Chalmers introduces two arguments [32]:

> First, it seems that a colourblind scientist given complete physical knowledge about brains could nevertheless not deduce what it is like to have a conscious experience of red. Secondly, it seems logically coherent in principle that there could be a world physically identical to this one, but lacking consciousness entirely, or containing conscious experiences different from our own. If these claims are correct, it appears to follow that facts about consciousness are not deducible from physical facts alone.

According to Bar-Yam [36] strong emergence is possible and can occur when there are no constraints in the individual components but only on collectives of those (i.e. global constraints). Laughlin considers *supervenience* a characterising attribute of strong emergence [37]; that is that the system, as a whole (or at the macroscopic level), supervenes in some way to the individual components (or the microscopic level) resulting to new properties and qualities which cannot be accounted for by using a reductionistic approach. On the other hand, Bedau clearly doubts the existence of any strongly emergent phenomena [38]:

> Although strong emergence is logically possible, it is uncomfortably like magic. How does an irreducible but supervenient downward causal power arise, since by definition it cannot be due to the aggregation of the micro-level potentialities? Such causal powers would be quite unlike anything

> within our scientific ken. This not only indicates how they will discomfort reasonable forms of materialism. Their mysteriousness will only heighten the traditional worry that emergence entails illegitimately getting something from nothing.

He does, however, propose the term *nominal emergence* to account for systemic properties which are not possessed by the system's constituents but given sufficient knowledge can be easily predicted [39]. An example of nominal emergence is the circle in which each individual point has no shape on its own and thus being a circle is considered a property of the whole. Given however the knowledge that each of these points have an equal distance from a fixed point it can be easily derived that the figure is a circle.

To account for cases where such knowledge is insufficient for predicting a systemic property, Bedau supports the notion of weak emergence and places it somewhere between nominal and strong emergence [38, 39]. He further provides a more formal definition of weak emergence as: *"Macrostate P of S with microdynamic D, is weakly emergent iff P can be derived from D and S's external conditions but only by simulation."* In this definition, S is an evolving system with various micro- and macro-level states and D governs the time evolution of the microstates in S.

To illustrate the applicability of his definition, Bedau considers a variety of Conway's Game of Life (GoL) scenarios. In GoL, an initial configuration is specified by a two-dimensional grid in which either alive or dead cells are situated. In every iteration, each cell's state is updated according to the state of its eight neighbours (four on its sides and four connected diagonally) in the previous iteration. If

exactly three of them were alive, the cell will become alive as well; in any other case the cell will die[2].

Mapping GoL concepts with Bedau's formal definition of weak emergence is rather straightforward:

System *S*. Any GoL configuration can be perceived as a system. Clearly, GoL scenarios consist of various microstates and macrostates which evolve over time.

Microstate[3]. The individual cells which can be either alive or dead.

Macrostate *P*. GoL structural macrostates such as gliders, glider guns (structures which produce gliders) or eaters (structures which destroy gliders upon collision).

Microdynamic *D*. The microdynamic in GoL is the simple birth-death rule described above, solely responsible for governing the time evolution of any microstates in S.

It thus follows that any structural macrostate P is weakly emergent if deriving its behaviour requires simulation. Two specific examples which fall under this category and thus prove the applicability of his definition are the *R pentomino growth* and the *glider spawning*. To further illustrate the applicability of the definition, an R pentomino refers to a configuration which results in a five cell edge-connected structural pattern which over time leads to new configurations which are different from all of its predecessors. A possible question arising is whether an R pentomino grows indefinitely, i.e. whether the total number of living cells is always increasing (assuming an infinite GoL grid). Extensive simulation proves that it does not as after ex-

[2] The rationale behind these conditions is that cells having two or less alive neighbours die of loneliness and cells with more than 3 alive neighbours die of overcrowding. The exact amount of 3 alive neighbours is perceived as that necessary in order to breed a new cell.

actly 1103 time steps the R pentomino stabilises [40]. This finite bound however could not be predicted or determined without simulation and thus it is considered a weak emergent macrostate of the GoL [39].

Bar-Yam [24] suggests another distinction of emergent phenomena according to the part of the system they affect: *local* and *global* emergence. While local emergence affects only a subset of the system, global emergence pertains throughout the system as a whole. Another interesting observation is that whereas isolating a small part of a system exhibiting local emergence would have little effect on the part and the system overall, removing a small part from a system with global emergent properties would cause this part to yield a completely different behaviour when compared to the one it previously had as part of the system [24].

Heylighen provided a taxonomy of emergence types based on the following attributes [22]:

- *amount of variety* in the system or its subsystems according to their state space.
- *internal* or *external* origin of the variation and selection mechanisms which lead to emergent phenomena.
- *single- or multi-level* character of the creative process and its structure.
- *type of constraints* according to mathematical criteria and their *contingency*.

The first attribute is directly related to the state space of a system: the more possible states and thus state transitions a system (or its subsystems) has the greater the amount of variety would be.

The second attribute characterises whether the mechanisms which may lead to emergent properties (such as attractors, adaptation etc.) are originating from within the system (internality) or from the outside (externality). Heylighen further classifies creative processes, which may not necessarily be characterised as emergent, according to whether they consist of structures with one level (e.g. an animal seeking for food), two levels, or more levels; the human body being an example of a multi-level system [22]. The last attribute is concerned with the classification of the different constraints on a system according to different mathematical properties [41] and the criterion of contingency[4].

Cariani distinguishes between three types of emergence [21]:

- *computational emergence*: a mathematically-based theory concerned with the emergence of *new formal structures*.
- *thermodynamic emergence*: a physically-based theory related to the origins of *new physical structures*.
- *emergence relative to a model*: a biologically-based theory concerned with the origins of *new functions*.

The first type is concerned with the emergence of new formal structures (e.g. symbols, patterns) which can be observed in simulations of computational models such as Turing machines and cellular automata. Typical examples that fall under Cariani's computational emergence include the flocking behaviour of birds and gliders in the game of life. Emergent phenomena regarding new physical structures and lying within the fields of dynamical theory, non-equilibrium ther-

[4] The criterion of contingency [22] states that *"either the constraint is absolute and fixed, precluding and allowing always the same variations, or different variations will be selected in different situations, depending on the circumstances"*.

modynamics, and information theory fit into thermodynamic emergence. Finally, emergence relative to a model is based on Rosen's view of emergence [20], and refers to a functional theory of emergence. The expected behaviour of a device (or an artificial system in general) should be formally modelled a priori and any discrepancies or deviations observed when the device is actually functioning are characterised as emergent phenomena. As a consequence the model originally describing the behaviour of the device has to be revised in order to keep up with the new (emergent) features observed during execution.

A more comprehensive classification of emergent phenomena has been provided by Bar-Yam who separates them into four distinctive types [36]. Type 1 is considered as weak emergence, types 2 and 3 as strong emergence while type 0 as another form of emergence not strong enough to be even classified as weak emergence. The latter type is mostly about the effects that the organisation of individual parts has on the observation of a system as a whole. Type 1 emergence proceeds a step further by describing situations where predicting global properties by merely observing the individual parts is considered impossible or at the best case unlikely [36]. Finally, both types 2 and 3 encompass strong emergent phenomena, with the former focusing mostly on states and ensembles and the latter on environmental information and how these are necessary in order to determine macroscopic properties [36].

Inspired by BarYam's categorisation of emergent types [36] and based on the classification of cellular automata originally proposed by Wolfram [42] and subsequently refined by Eppstein [43], Fromm proposed the following taxonomy of emergent phenomena [1]:

Type I. Simple or nominal emergence without top-down feedback, containing only feed-forward relationships.

Type Ia: Simple Intentional Emergence. Intended emergent properties arising due to the organisation and interactions of the local parts (e.g. the various parts of a clock or components of a software system) which do not even qualify as weakly emergent.

Type Ib: Simple Unintentional Emergence. Non-intended emergence concerned with statistical properties that cannot be applied to single local parts but rather to (relations of) collectives. For example pressure cannot be applied to a single particle nor slope can be defined for a single grain of sand [1].

Type II. Weak emergence including self-organisation and top-down (either positive or negative but not both at the same time) feedback.

Type IIa: Weak Emergence (Stable). The most commonly referenced type of emergence, including top-down feedback and bottom-up influences, found in the flocking/schooling behaviour of birds/fishes, the foraging behaviour of ant colonies etc.

Type IIb: Weak Emergence (Unstable). An undesired form of emergence based on positive feedback with typical examples including the creation of bubbles in the stock market and buzzes in the news.

Type III. Multiple emergence with many feedbacks.

Type IIIa: Stripes, Spots, Bubbling. A multiple-feedback emergence such as in the case of short-range positive feedback and long-range negative feedback (i.e. activator-inhibitor sys-

tems) responsible for stripes, spots and other patterns that can be found in animals.

Type IIIb: Tunnelling, Adaptive Emergence. A form of emergence common in adaptive and evolutionary systems related to extinctions, catastrophic events in the environment and sudden jump of barriers (mental or physical).

Type IV. Strong, multiple-level (as per Heylighen [22]) emergence with huge amounts of variety and irreducible in principle. Characteristic examples are the emergence of life out of genes and genetic code and the emergence of culture out of memes and language systems [1].

Table 1 depicts the different feedback types along with local-global boundaries and the jumps/leaps in complexity associated with each emergence subtype. As the figure suggests feedback types include no feedback at all (type Ia), peer-to-peer feedback (type Ib), either positive or negative feedback but not both at the same time (type II), positive and negative feedback simultaneously (type III), and all of these in addition to feedback across different systems (type IV). Furthermore, the more complex the emergent phenomenon the larger the jump or leap in organisation, ranging from a static jump (e.g. type Ia) to a quantum leap in complex adaptive systems (e.g. type IIIb) and gateways to new evolutionary systems (type IV).

Each type of emergence has a different predictability associated with it, as Table 2 suggests. Type I is absolutely predictable and type II is predictable in principle. Type III is much harder to predict if not completely impossible. Lastly, type IV is classified as strong emergence and is not predictable in principle, coinciding with Chalmers' [32] and Stephan's [33] views on strong emergence and

Type		Boundaries	Feedbacks	Jump/Leap
I	**Ia**	agent-system boundary (only in one direction)	no feedback but absolute commands or constraints	intended or static jump to higher level of organization
	Ib	agent-agent boundary	scale-preserving (peer-to-peer) feedback	fluctuations, no jump to significant higher level of organization
II	**II**	agent-group or agent system boundary (in both directions)	scale-crossing (topdown) feedback, positive or negative	dynamic jump to higher level of organization
III	**IIIa**	agent-group or agent system boundary (in both directions)	scale-crossing (topdown) feedback, positive and negative	dynamic jump to higher level of organization
	IIIb	large fitness barriers in complex evolutionary systems	multiple feedbacks in a system	quantum leap in complex adaptive system
IV	**IV**	boundary between different evolutionary systems, barrier of relevance	all of the above, incl. feedback between different systems	gateway or quantum leap in evolution to new (evolutionary) system

Table 1: Boundaries, feedbacks and leaps of complexity for the different emergence types. Adapted from [1].

Type	Name	Roles	Frequency	Predictability	System
I	**Nominal or Intentional**	fixed	abundant	predictable	closed, with passive entities
II	**Weak**	flexible	frequent	predictable in principle	open, with active entities
III	**Multiple**	fluctuating	common - unusual	not predictable (or chaotic)	open, with multiple levels
IV	**Strong**	new world of roles	rare	not predictable in principle	new or many systems

Table 2: The classification seen from the perspective of predictability and system roles. Adapted from [1].

predictability. Furthermore, roles are different for each of the types with fixed ones for type I, flexible for type II, fluctuating (existing roles disappearing and new ones emerging) for type III and lastly, on type IV, completely new worlds of roles appear.

2.3 Summary

While there is a renewed interest in emergence, it is clear that researchers have not yet agreed to a common view or definition of emergence. The debates seem to be endless regarding observers, the elements of surprise and novelty, and different levels, scales, and hierarchies, to name a few.

Although the term engineering emergence might be oxymoronic according to some definitions, there are others that could sufficiently support such a goal. Before concluding to such a working definition, the next chapter presents the most notable approaches to engineering emergence.

3 Different Approaches to Engineering Emergence

Emergence is commonly seen as an unexpected and unpredictable effect. Such approaches to defining emergence, as discussed in detail in the previous section, make the phrase "engineering emergence" an oxymoron. Moreover, emergence in the context of engineering is typically interpreted as the appearance of undesired and unforeseen properties in a system or device, which are commonly described as malfunctions or misbehaviour. In this work, we use the term engineering emergence to refer to disciplined ways of incorporating *desired* and *beneficial* properties into systems by utilising, in some way or another, emergent processes. While many researchers would reject the whole concept when phrased like this, others have been trying to achieve exactly that.

One of the earliest works on the topic was carried out by General Electric's Bush and Kulkarni [5] in the domain of Microelectromechanical Systems (MEMS)[5] which can also be seen from an active networks[6] perspective. The microscopic level in their case study consists of individual MEMS devices while the macroscopic level consists of network-wide qualities and behaviours.

[5] MEMS are very small computational devices with a size ranging from few micrometers to one millimetre.

[6] Active networking refers to an alternative networking paradigm which allows the packets (also known as active packets) to alter the operation of the network by executing arbitrary code within special networking hardware equipment.

MEMS devices have limited resources and computational power making thus impossible to explicitly program intelligent behaviour into them. In their white paper, Bush and Kulkarni attempt to harness emergent phenomena created by the various interactions among the devices as well as the environment in order to achieve global characteristics such as an optimal performance on active networks. A MEMS device can be seen as consisting of an inherent physical behaviour, an explicitly programmed behaviour, and a sense-and-control unit which interacts with the environment via sensing and actuating. Given that the explicitly programmed behaviour requires more computational resources, the authors try to understand and harness the inherent behaviour and the external emergent responses perceived by the sense and control unit.

Bush has further developed a mechanism which allows the injection of network component models into operational networks, aiming to improve the overall performance [44]. In this work Bush attempts to utilise the aggregate emergent behaviour of a set of MEMS devices in order to model a single, more powerful, MEMS device (see Figure 2). The goal is to allow for behaviours and intelligence which would be otherwise impossible to implement on a single MEMS device due to computational limitations.

Stepney, Polack, and Turner believe that "engineering emergence is not an impossible dream" [6]. Following Abbott's observations on emergent properties from an engineering point of view [45], they set as a goal the transition from an abstract design which describes a global emergent, to a lower level implementation of that design [6]. Further, they adopt the views of Ronald et al. [25, 46] on using sep-

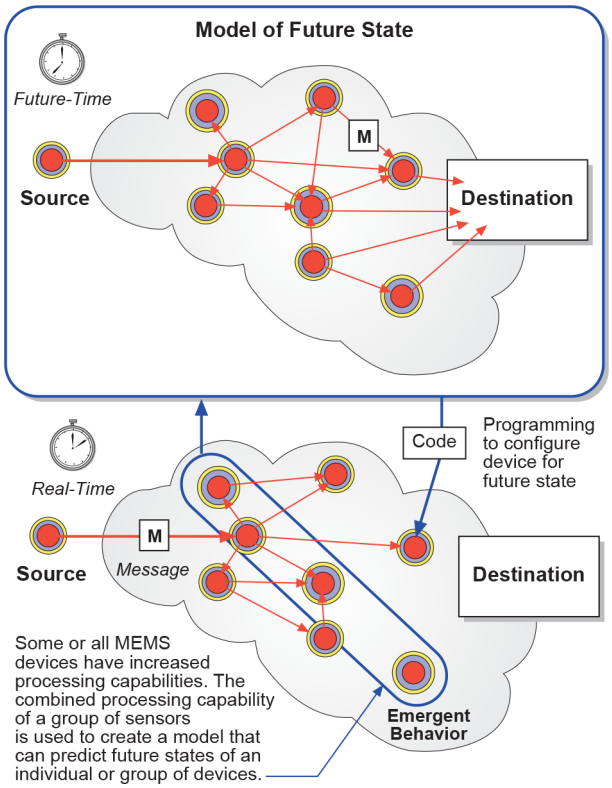


Fig. 2: A network of MEMS devices acting as a single MEMS device with increased processing capabilities. Reprinted from [5].

arate languages for describing the implementation and the abstract, higher level, emergents.

Due to the different concepts employed in these two languages, the authors claim that classic refinement techniques are not sufficient for emergent systems [47] and thus they propose retrenchment [48] as an applicable alternative for such systems [49]. Their main contribution to the engineering of emergence is the architecture depicted in Figure 3.

The architecture comprises three elements: the system of systems (SoS) model, the local system model, and the integrated model.

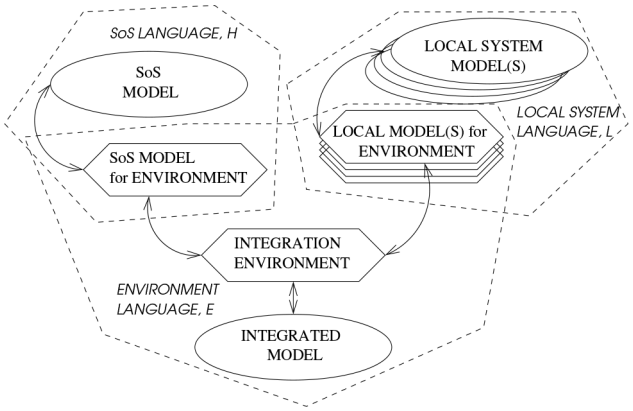


Fig. 3: A three element architecture for engineering emergence. Reprinted from [6].

Macroscopic properties are described in the SOS model while microscopic are described in the local model; a different language, H and L respectively, should be employed for each of them. Finally, a third model is introduced to describe the integration of SOS and local models. This integration model contains any environment characteristic in another language, E, which is shared with both SOS and local environment models. The authors illustrate the use of their architecture by applying it to different scenarios such as Game of Life gliders and nanite platelet systems.

A nanite platelet system, for example, corresponds to the proposed architecture as follows [6]:

- SoS model — a collection of platelets which can move and clot, described in H.
- Local system model — a single nanite platelet which changes according to some local rules which are activated through sensor inputs, described in L.

- Integrated model — the physical environment of a blood vessel, described in E, providing the components' physical locations, movement, and inputs.
- SoS model for the environment — a collection of platelets and clots within a blood vessel.
- Local model for the environment — the individual components (platelets) situated in the environment (blood vessel) and provided with inputs from the latter via their sensors.
- Integration environment — a simulation environment of the real (physical) environment to aid understanding.

Fromm [50] attempts to tackle many intrinsic issues on combining engineering and emergence or self-organisation[7]. Claiming that the problem of engineering emergence equals the problem of science in general, he proceeds by applying a variety of classical scientific methods to this domain and proposes that the solution is an "intelligent design based on the classic scientific method" [50]:

> ...is indeed possible, if we apply the scientific method to the domain of engineering. The solution is therefore an *intelligent design* based on the classic *scientific method*. The problem of "engineering emergence" — to find a simple rule for a complex pattern — equals the problem of science in general: to explain complexity by describing complex natural phenomena with a minimum of primary principles, laws and rules. The central questions of the study of "emergence" and science are similar — how can you find simple local rules to generate specific higher levels of global organisation — and therefore the answers or solutions are similar, too.

[7] The author uses the terms emergence and self-organisation interchangeably.

What Fromm suggests is the use of conventional scientific methods, such as iterative two-way approaches or synthetic microanalysis, applied on artificial worlds (such as Conway's Game of Life) instead of the natural world. Nonetheless the suggestion is rather generic and is limited by the scientist's creativity and experience.

Welch et al. [51] expresses views on engineering emergence that are very close to our perspective on the topic: systems in the near future will be too complex to design and implement *explicitly* and thus we will have to learn to engineer them *implicitly*. They suggest the use of simple, local, behaviour rules on a large number of interacting components (agents) from which they expect that the desired, global, behaviour will emerge. Their simulation platform is based on the CSP and *pi*-calculus formalisms, and implemented in the occam-*pi* language, which allows for massive parallelisation in grid-based computational environments.

In [51], the authors experiment with various formations of Reynold's boids, succeeding in engineering various emergent behaviours such as flocking, feeding frenzy, panic/fear, and directional migrations, among others. They achieve this by focusing solely on the implementation of microscopic-level properties, i.e. attributes of a single bird. For example, by changing the birds' vision angle the collective's flocking behaviour changes as well. In this case the vision angle is an explicitly programmed characteristic which concerns the individual birds, or put differently the microscopic level. The flocking behaviour and its variations (e.g. squabbling), however, relate to a collective of individual birds, or the macroscopic level of the system, and it was neither explicitly programmed nor planned a priori. This

is a fine demonstration of how local properties can control complex, global, behaviours in non-obvious and non-linear ways.

Ulieru and Doursat [52] state that Cyber-Physical Ecosystems (CPEs)[8] is yet another domain in which conventional engineering approaches will be insufficient in tackling with their complexity. They challenge many assumptions of the traditional engineering school such as the need for a system to be well-defined in terms of functionality and performance as well as the top-down approach followed by most designers of distributed systems. They suggest "emergent engineering" as an endogenous (i.e. bottom-up) alternative for designing adaptive systems by focusing only in the microscopic level and allowing for self-organisation and emergence to appear in the global. They further devise three generic principles of emergent engineering [52]:

- *Architecting from the bottom-up without an architect*: where focus is centred on the microscopic state, or the agent level, and the set of variables and rules governing this without any reference to the whole or any explicit development towards the global-level desired functionality (e.g. adaptability).
- *Control without a controller*: referring mostly to the autonomy of the local-level entities where there is no central controller, as in traditionally engineered systems, to coordinate the whole process and instruct each agent what to perform next. Instead, each agent has its own behaviour defined through a set of "micro-rules". Such rules can be further decomposed into two parts: a positive

[8] Cyber-Physical Systems or Cyber-Physical Ecosystems refer to environments combining elements from the physical world with digital artefacts. A typical example is the connection of sensors to the Internet, providing an information channel between the physical and the cyber world.

feedback reinforcing local behaviours which lead to the creation of new macroscopic structures and a negative one correcting agents' behaviours resulting in stabilisation of such structures.

– *Co-evolving the CPE with the environmental dynamics*: where after a CPE (or any other global-level conceptual entity) has attained a certain growth and established some desired functionality, it should be in position to evolve along with the environment it is situated.

During the literature survey of engineering emergence it became apparent that although the field might be still immature there is an ongoing research effort on engineering emergence. Furthermore, there is an increased interest into devising processes, patterns, and tools which could allow for the exploitation of emergent phenomena in more disciplined ways, from both complex systems and multi-agent systems communities. Many different research groups are attempting to achieve what was inconceivable a decade ago.

The next chapter discusses emergence from an engineering perspective and concludes to a working definition of emergence that does not contradict the notion of engineering it. Moreover, it presents previous work on harnessing emergent properties in ADS and introduces a framework that can guide system designers to do the same.

4 Harnessing Emergent Properties in Artificial Distributed Systems

Having presented the most notable attempts at defining emergence and classifying emergent phenomena (Chapter 2), it becomes apparent that the scientific community has not yet reached a consensus on the subject. Section 4.1 discusses emergence from an engineering perspective and concludes to a single definition of emergence that is adopted for the rest of this work. Section 4.2 presents the author's views on the feasibility of engineering emergence and the need of domain-specific approaches and studies. Section 4.3 details the EDBO case study which was devised in order to test, via simulations, various ideas on harnessing emergent properties in ADS. The experience gained during this work [2] has been abstracted into an experimental framework [7], which is presented in Section 4.4.

4.1 Emergence from an Engineering Perspective

A central idea in every definition of emergence appears to be the existence of different levels where actions on a lower level result in emergent properties in a higher one, often in *non obvious* ways. The

debate seems endless regarding the necessity of observers, novelty, surprise, downward causation, and supervenience, and it seems that this will be the case for as long as it takes to explore and understand emergent phenomena in more disciplined and scientific ways.

Nonetheless, it is essential that a single definition is adopted throughout this work. We consider emergence from an engineering perspective and thus some of the solely philosophical debates, like the discussions around *downward causation* and *supervenience*, seem irrelevant to our goal. On the other hand, we believe that *different levels*, such as the microscopic and the macroscopic (and even the mesoscopic according to some authors [52]), are a crucial part of emergence. Like many of the researchers in the field we feel that the element of *surprise* might lead to misconceptions and ambiguity, and hinder the validity of any scientific results. Considering only the observer's surprise for classifying something as emergent would potentially threaten the validity of any engineering approach we may suggest. Instead we prefer to focus on the *novelty* and *added-value* characteristics of these arising properties: if something was not apparent in the microscopic level and was not explicitly engineered, yet it emerges on the macroscopic level and the system benefits from it, it is sufficient to accomplish our aim.

After an extensive literature review, we believe that the definition provided by Wolf and Holvoet [9] fits best the scope of our research and focuses on the computer science aspects of emergence. Moreover, we feel that it sufficiently meets the objectives we set before, from an engineering-emergence point of view:

> A system exhibits emergence when there are coherent emergents at the macro-level that dynamically arise from the interactions between the parts at the micro-level. Such emergents are novel w.r.t. the individual parts of the system.

The definition incorporates every aspect we consider relevant to emergence such as the need for different levels (or points of view of the system) and the interaction between micro-level components. At the same time it lacks any reference to the controversial issues of observers, the need for surprise, downward causation, and supervenience, which we believe would only introduce unnecessary complexity to engineering attempts, even though they might be of great importance in a philosophical context.

A common criticism of this definition is the recursive use of the word "emergent" as a means of defining "emergence", however, we feel that Wolf and Holvoet sufficiently clarify this by stating that [9]: *"the definition uses the concept of an 'emergent' as a general term to denote the result of the process of emergence: properties, behavior, structure, patterns, etc."* An emergent, thus, can be any relevant attribute apparent at the macro-level of a system, but not at the micro-level.

A single definition however cannot possibly encompass and sufficiently describe all different forms and types of emergence; there is a further need for a classification scheme similar to the ones discussed in Section 2.2. The author believes that Fromm's classification of emergence types and forms [1] is the most comprehensive and complete work on the subject available so far. It succeeds in distinguishing different types and subtypes of emergence while at the same time providing a critical view on various inherent and intri-

cate concepts of emergence such as predictability, roles, boundaries, leaps, and feedback types. For the purpose of this work the author assumes Fromm's classification, unless explicitly stated otherwise.

4.2 Engineering Emergence

The author believes that neither the complex system field nor engineering as a discipline are mature enough as to allow for catholic claims about engineering emergence. It is necessary to research emergent phenomena extensively in order to be able to harness them in our engineered models, in a domain-by-domain case. Generic claims of engineering emergence are impossible to make and it is doubted that a generic "one size fits all" solution can or will be provided soon, if ever. The author proposes the study of each specific domain separately, in an attempt to identify generic patterns and guidelines which could assist with introducing desired emergent properties. The domain of cyber-physical ecosystems (CPEs), for example, has yielded a need for emergent engineering as a characteristic attribute of CPEs is that they cannot be defined a priori but rather emerge [52]. Stepney, Polack and Turner have centred their research on engineering emergence on the field of molecular nanotechnology [6] while Welch et al. focused on herd formations [51]. Similarly, Bush has shown some promising results in the field of MEMS devices and active networks [5].

This work focuses on the domain of ADS. In many domains centralised architectures have been replaced by decentralised approaches which are able to offer significant advantages in utilisation of network resources. Nevertheless, the increased demand and complex-

ity of applications and services operating within distributed environments has demonstrated the need for more efficient, robust and adaptive solutions that will operate without manual configuration and management. Systems are getting increasingly complex and it is inevitable that a point will be reached where humans will not be able to cope with such complexity; both due to resource and capability limitations.

It is for such situations that humans turn to nature for inspiration. A multitude of useful, typically self-*, properties can be observed to emerge from natural complex systems such as ant colonies and bird flocks. The ability to guide and use these emergent processes could prove very valuable since emergence is considered to be the basis for a variety of very useful phenomena including self-organisation, self-optimisation, adaptation as well as other beneficial properties encountered in complex systems. Incorporating such behaviours in ADS could offer significant benefits to the development and performance of the system, making it highly available, scalable and robust. Having even partial control of this process, however, proves to be extremely difficult.

Section 4.3 details the EDBO case study which was devised in order to test, via simulations, various ideas on harnessing emergent properties in ADS. The experience gained during this work [2] has been abstracted into an experimental framework [7], which is presented in Section 4.4.

4.3 The EDBO Case Study

The EDBO [2] case study was devised in order to validate a number of hypotheses and gain insight into various aspects of distributed systems exhibiting emergence. In EDBO each agent, referred to as *biobot*, is considered to be a network peer (node) acting simultaneously as service provider and consumer. In other words, each biobot offers services to its peers while at the same time it issues requests for services it needs.

Beyond the typical service capabilities which are common to most peer-to-peer applications, EDBO employs a distributed query forward mechanism. Instead of relying on a central authority (similar to a UDDI registry or a central fileserver index) for locating desired resources, each peer sends to its neighbours a special type of message informing them of the resource it wishes to utilise. If the recipient can fulfil the request it replies positively to the requester, otherwise it forwards further the request to its own neighbours either randomly or based on keyword similarity criteria. The whole forwarding chain is restricted by a time-to-live (TTL) attribute in order to be finite.

The main differentiator of EDBO compared to other file sharing and distributed service paradigms is the introduction of energy values. Following a biologically-inspired approach to networking, each biobot possess some energies, emulating natural organisms which in principle try to maximise their own energy. Different energies can be affected by various events such as successfully matching another peer's request or being part of a successful forwarding chain (i.e. the node's decision to forward the request to a certain biobot resulted in the resource being found). Higher amounts of energies reward biobots by allowing them to perform an array of bio-functions such

as replication, migration, and (a)sexual reproduction. On the other hand, depletion of energies could result in the death of a biobot. This evolutionary approach emulates the process of natural selection, or survival of the fittest, in which the best biobots live longer and are given the chance to pass their characteristics to offspring while the weaker ones vanish. What makes a biobot better than another varies and depends on the application requirements as well as the hypothesis in hand. It could be the case that a biobot is considered successful when the services it provides are on high demand. In such a scenario giving the chance to successful biobots to reproduce might allow the network to handle specific traffic peaks and lead to increased scalability. In this context, scalability is a macroscopic quality which might emerge due to interactions of microscopic entities (i.e. the biobots).

4.4 An Experimental Framework

Previous work from Paunovski, Eleftherakis, and Cowling [53] concentrated on devising a framework for studying emergence in a disciplined way that provided insight into the causal relations between microscopic and macroscopic levels. In a second phase, the aim is to develop systems that exploit desired emergent properties, focusing on ADS, aiming in the long term to provide designers and engineers with tools, models and patterns. Towards this direction and in an attempt to put forward several ideas on how to approach and revise engineering practices when dealing with systems which exhibit emergence, an abstract process has been devised which was followed throughout the experimentation. Figure 4 depicts the experience gathered on engineering ADS which exploit emergent phenomena, as a successor

to the previous framework for studying emergence. It is an abstract and laborious process of experimentation, attempting to provide a disciplined approach to harnessing emergent properties as a means of providing desired behaviours to complex systems.

The starting point (step 1) is defined as the analysis of the system to be built and the separation of the desired properties of that system into two categories: the first includes functionality that is sought to be developed by following standard (software) engineering processes and methodologies (step 2) and the other consists of desired properties which are considered to be related with emergent phenomena, such as self-adaptability (step 3). Step 4 consists of modelling the so called "normal" behaviour as a multi-agent system (MAS). The abstract model can be afterwards refined to a more formal and less ambiguous model such as an extended finite state machine (FSM). At this stage, the use of validation and verification techniques is strongly encouraged. Proper verification of the standard system model can ensure at later stages that the presence or absence of particular emergent properties is the effect of the "planted", emergence design, and not due to bugs and design flaws in the standard model.

Step 3 is concerned with the design, through a process of experimentation, of the additions and modifications to the MAS model, which may lead to the appearance of the desired emergent phenomena. As already argued, there cannot be strict or specific guidelines for this step; depending on the domain and the type of emergence expected, different patterns can be utilised to form an initial hypothesis. These additions, combined with the standard model, form the *extended system model* (step 5).

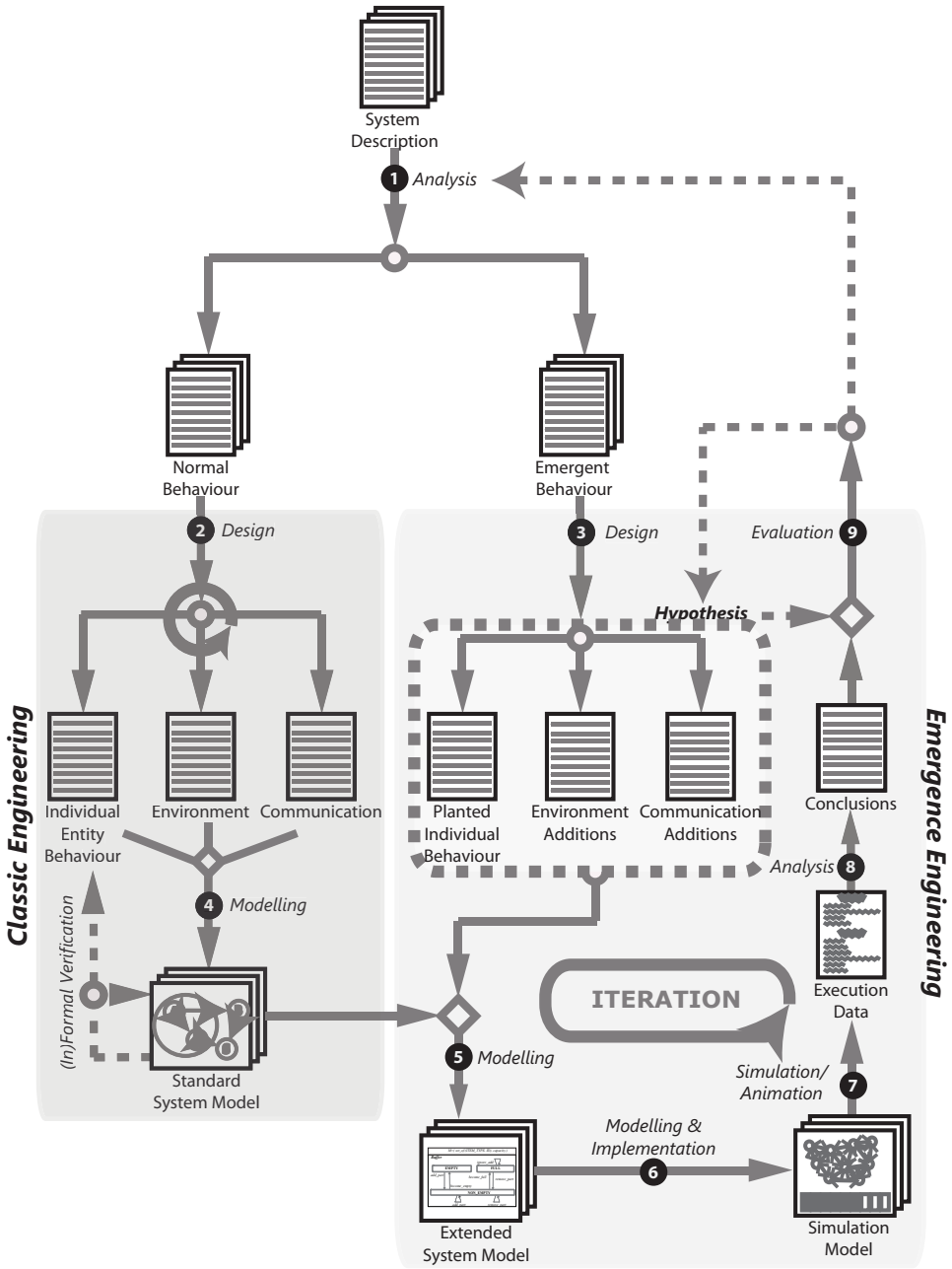


Fig. 4: An abstract process for engineering emergent properties in complex systems. Reprinted from [7].

In step 6, this model is then refined, translated or implemented in order to serve as a simulation model on which experiments can be run. After simulating and analysing the results, in step 7, it should be possible to reach some conclusions regarding the initial hypothesis (step 8). The last step, 9, consists of comparing and evaluating the conclusions against the initial hypothesis. At this stage, there is either a conjecture that the hypothesis is sound so the desired functionality can be implemented or the hypothesis is disproved in which case another iteration starts at step 5.

As already stressed, simulation can sometimes be the only means of testing and supporting such ideas. The next chapter describes the EDBO simulation platform which has been used so far for running the EDBO case study and introduces FLAME; a more general-purpose simulation platform which could overcome various limitations encountered so far.

5 Simulation Platforms

Simulation is of utmost importance for complex systems as it allows for concepts and ideas to be tested before proceeding to the actual implementation. By means of simulation it is possible to better understand the emergent phenomena which occur in natural systems and try to reproduce them in artificial designs. Previous work by Paunovski, Eleftherakis, and Cowling [53] has focused on developing a simulation platform which will aid to the better understanding of emergence in complex natural systems. The platform has been later expanded by the same authors in order to allow for the simulation of the EDBO case study. This platform is considered in the next section while Section 5.2 introduces FLAME: a general-purpose, agent-based simulation platform which could overcome various limitations identified in EDBO.

5.1 The EDBO Simulation Platform

In order to test the various hypotheses of the EDBO case study a platform has been developed which allows for the simulation of various biobots with different parameters and strategies [2]. Biobots are situated in the BioSpace which serves as a model of the environment. BioSpace offers a number of services that enable biobots to communicate with each other, to perform energy management, to move in the logical space of the network, to manage relationships

with neighbours and to perform reproduction. The attributes which can be customised through the platform include:

- *initial state parameters*: such as the number of biobots, the services they provide, initial energy levels and energy costs, etc (see Table 3 for a detailed list of parameters).
- *discovery strategy*: determines how a biobot should decide to which neighbour to forward an incoming query that it cannot satisfy itself. A random strategy forwards the query to a neighbour chosen in a non deterministic manner. A similarity-based strategy requires an additional vector to be associated with each of the neighbours, containing a list of representative keywords for each of them. Upon receiving a query, its keywords are compared with these vectors in order to make a more informed decision. Finally, a complex discover strategy relies on a heuristic algorithm which provides estimates for each neighbour on how likely it is to result in a successful query match. The algorithm utilises a number of metrics such as keyword similarity, proximity, history, and success of the neighbour.
- *relationship types*: with available options being simplex and duplex. In simplex a biobot may establish an one-way relationship with another biobot without the consent and knowledge of the latter. In this case the latter has no reference to the former. In duplex relationships, relationships are mutual and thus consent of both is required.
- *movement strategy*: determines towards which position a biobot should move once it decides to relocate. Options include moving towards the nearest partner, the most successful one (in terms of

energy) or closer to whichever biobot has forwarded the highest number of queries.

– *relationship formation strategy*: determines how new relationships between biobots are formed. It can be random or based on other characteristics such as the partners' success or distance.
– *birth and death strategies*: define when a biobot can replicate or reproduce and when it should die. Different strategies allow for random events or for events based on the energy levels of the biobots.

Global Parameters	
Total Cycles	15,000 cycles
Statistics Period	500 cycles
Cycle Quorum	90% of population
BioSpace Size X	1,000 units
BioSpace Size Y	1,000 units
Neighbour. Radius	10 units
Total Keywords	300 keywords
Query Parameters	
Query Load	200/20/200
Query Load Period	5,000 cycles
Query TTL	30 cycles
Keywords per Query	10 keywords

BioBot Parameters	
Init. Population	500 entries
Services per BioBot	20 services
Init. R. Buffer Size	10 relationships
Max. R.Buffer Size	30 relationships
Min. R.Buffer Size	5 relationships
Service Parameters	
Total Service Types	800 types
Keywords per Service	10 keywords
Query Service Range	- Random - Restricted
Que. Ser. Range Size	30 types
Que. Ser. Range Prob.	0.8 (80%)

Discovery Energy Parameters	
Initial Bot Energy	2,000 units
Query Path Reward	1,250 units
Message Forward Cost	0.085 units x Distance
Movement Cost	3 units per 1 spatial unit
Service Energy Parameters	
Initial Service Energy	2,000 units
Query Match Reward	800 units
Query Similarity Reward	50 units
Query Processing Cost	1.5 units

Table 3: A complete list of the initial parameters. Adapted from [2].

The main disadvantage of EDBO is that the simulation platform is tied to the model, both of which have been implemented as a standalone Java application. Thus, in order to alter the model someone has to change the actual program itself making the process time consuming and error prone. The majority of the initial simulation parameters (as depicted in Table 3) can be easily customised as they are stored in a separate XML file, however, this is not the case for the various strategies described above. Another concern is the ability of the simulator to perform in a timely and responsive manner under heavy conditions where thousands of biobots interact

with each other. Being developed as a standalone Java application indicates that the application cannot be run in a distributed or parallel fashion which limits the available resources to those of a single computer. An alternative simulation platform which overcomes both issues is FLAME, discussed in more detail in the next section.

5.2 FLAME

The FLexible Agent-based Modelling Environment[9] (FLAME) is a general-purpose, agent-based simulation platform developed by the University of Sheffield in collaboration with the Software Engineering Group at the STFC Rutherford Appleton Laboratory. FLAME's main advantage is that it is optimised for high performance computing which renders possible the simulation of hundreds of thousands of agents concurrently. Nonetheless, FLAME has available ports for most personal computers running Windows, Linux, or MacOS, making development easier and without imposing the need for a supercomputer.

Overall, FLAME is a flexible and powerful simulation environment which has been tested in many complex case studies with many thousands of agents executing in parallel. It has received significant attention from multiple scientific disciplines and has already been utilized in large scale projects like the modelling of European economic policy design [54], understanding how the Escherichia coli bacterium responds to oxygen [55], and agent-based modelling of the behaviour of epithelial tissues [56].

[9] http://www.flame.ac.uk

The main idea behind FLAME is the use of X-machines as a means of modelling individual agents. X-machines, initially introduced by Eilenberg in 1974 [57], is a formal method that enhances the class of Finite State Machines (FSMs) by introducing memory and functions. Holcombe and Ipate [58] renewed interest in X-machines by devising a variant named Stream X-machine (SXM). An SXM is defined by an input stream, an output stream, a set of values that describe its memory structure, a set of states, a state transition set and a set of functions (see Figure 5 for a diagrammatic notation). Labels in the transitions are functions which are triggered through an input symbol and a memory instance to produce an output symbol and a new memory instance. SXMs have been shown to be particularly suited for specification purposes [59] as well as for modelling agent-based systems [60–62].

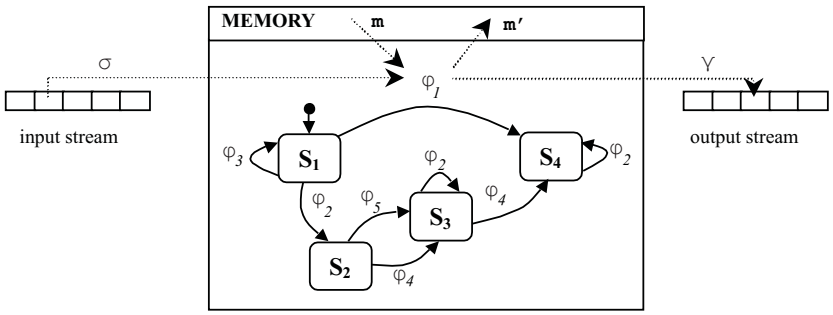


Fig. 5: An abstract X-machine in diagrammatic notation.

Thus each FLAME agent, or *agent machine*, has a particular initial state and initial memory, a set of possible (reachable) states and a set of functions which accept input, produce output and affect state transitions. It is also possible for a function to have certain preconditions. All agent instances are run simultaneously by FLAME

with an iteration starting at the start states of each agent type and terminating when every agent has reached an end state (see Figure 6).

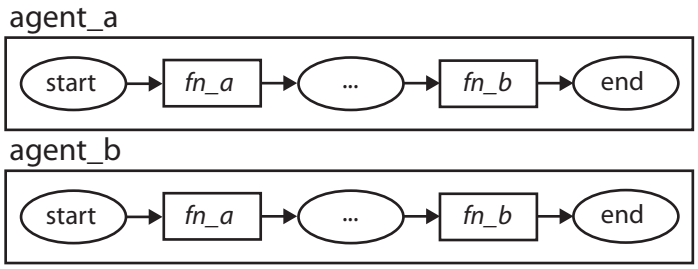


Fig. 6: An iteration with two agents instances of the same type running on parallel.

The communication between agents in FLAME is based on the Communicating X-Machines (CXMs) formalism which in essence consists of SXMs capable of exchanging messages. Different proposals have been made regarding the way messages are exchanged among machines with the most accepted one utilising a communication matrix [63]. However, this approach is not suitable for cases where SXMs are used to model agents [64, 65]. Instead, an input-centric method is used where each message is stored on a global read-only space accessible by all agents. The messages are sorted by message type which is up to the modeller to define. This implies that each agent machine is responsible for fetching on its own any message that it might be expecting and discarding any which is deemed irrelevant. Building on top of this simple approach, it is possible to simulate various communication schemes such as direct messaging (where each message has a recipient id and the agents read only the messages bringing their id) or proximity communication (where the

message carries the sender's position and the agents discard messages from agents which are further away than a certain value).

In practice, FLAME is implemented in C code which was highly optimised for parallel execution. A simulation model is defined by a *model description* and the *agent functions*. The model description is specified in the X-machine Markup Language (XMML) file which builds on XML syntax and is accompanied with an XML Schema (XSD) for syntax validation purposes. An editor[10] is also available, allowing the specification of a model through graphical means instead of directly writing XML. The model can hold the following information:

- *General information* regarding the model such as name, author names, version, description etc.
- *Environment constants* such as π or other model-specific values.
- *Agent types* defining in detail the available agents (not their instances): name, memory variables, functions, conditions and state transitions.
- *Message types* that agents can exchange defined by their name and the different variables they carry.
- *Datatypes* for representing complex structures which cannot be modelled via standard datatypes of C.
- *Time periods* consisting of multiple simulation runs or other time periods already defined.

The agent functions are specified once for each different agent type in separate files, as executable C code. In addition, FLAME offers a set of useful macros for performing common operations such as

[10] http://www.flame.ac.uk/docs/flameeditor/v1/

reading all messages of a specific type (by using the macro *start_messagename_message_loop*) or sending a message (with *add_messagename_message*($var1, ...varN$)). Agent death is made possible at any point of the simulation by returning any non zero value from such a function.

Finally, the initial simulation state is configured in a separate XML file in which the modeller is expected to define the initial agent population along with values for their variables and any environment constants declared in the model specification. Once a simulation is started, FLAME produces a separate XML file for each iteration (unless it was instructed to do otherwise) containing every agent attribute and current values. These files can be then used to draw conclusions either in a manual way (e.g. by having a program parsing the output data) or by using the FLAME visualizer which was recently released. Additionally, a GPU-programming extension has been created under the name FlameGPU[11] which allows for efficient GPU processing of models. The extension provides a mapping between agent specifications and optimised CUDA code, allowing modellers with no knowledge of CUDA programming to harness the computation power of the GPU paradigm while at the same time producing real time visual results.

[11] http://www.flamegpu.com

6 A Formal Model of EDBO

The EDBO case study [2] has demonstrated the potential of engineering emergent properties in ADS. By following an iterative and experimental, yet disciplined, framework for engineering emergence in ADS [7], simulations with EDBO resulted in various beneficial macroscopic behaviours emerging. Introducing biologically-inspired functions and energy as part of the node's design, allowed for the emergence of complex global properties.

As discussed earlier, the major flaw of EDBO is that the model is tied to its own, in-house developed, simulation platform. As such, there is no concise formal or semi-formal description of the EDBO model available. In an attempt to increase the confidence in this case study, this chapter defines EDBO using the X-machine formalism.

Foremost, this allows for the easy reuse of the paradigm in other simulation platforms and environments. Furthermore, by using a formal, mathematical, language to describe the model, a more complete insight can be obtained while at the same time it allows for resolution of ambiguities. The rich ecosystem surrounding X-machines enables us to increase our confidence in EDBO by providing a variety of formal and informal validation and verification techniques such as animation, testing, and model checking. Finally, X-machine models can be relatively easily adjusted in order to be simulated in FLAME [65] which could allow to reproduce (and cross-validate) existing results with a well-tested, general-purpose, simulation platform.

The next section provides a thorough description of the EDBO case study, its basic entities and the initial results gathered during simulations. Much of the content was gathered by extracting critical information from the simulation platform itself. Section 6.2 introduces the X-machine formalism which is subsequently used to model EDBO in a formal way, in Section 6.3.

6.1 Emergent Distributed Bio-Organisation

EDBO[12] is a generic case study inspired by the Bio-Networking Architecture [66]. EDBO reflects an overlay network in which each peer acts both as a resource provider and consumer; a typical property of the nodes in peer-to-peer (P2P) networks. Each network peer, referred to as *biobot*, is situated in the *biospace* which acts as a middleware between the physical and the logical layer. Consequently, physical implementation details, such as network connections among nodes, are completely abstracted from the model. An overview of the main EDBO components is depicted in Figure 7.

Resources, Relationships, and Energy

Each biobot serves a set of abstract resources which, depending on the specific application domain and case study, could represent Web services, documents, songs, or any other kind of digital information. The decentralized nature of the EDBO model avoids utilization of a centralized registry or repository. Consequently, the discovery of

[12] EDBO has been briefly introduced in Section 4.3.

resources is achieved through a *distributed query forwarding* mechanism in which all biobots participate. Resources are typically modelled by keywords that identify them, so that discovery strategies can be expressed in terms of various algorithms for matching sets of keywords.

If a biobot cannot satisfy a query, it will forward it to another one (determined according to the discovery strategy used), which in turn will try to satisfy the query or forward it further until either the resource is located or the query fails (either a resource cannot be found or the time-to-live flag has expired).

Connections at the logical layer are represented as *relationships*. Each biobot has a set of neighbours with whom it can communicate directly and forward messages. The relationship set may vary over time as existing connections are removed and new are formed.

The key characteristic of EDBO is the introduction of *energies*. Each biobot possesses a *discovery* energy as well as a *service* energy allocated to the resources it manages. Discovery energy represents the usefulness of a biobot in the process of locating resources on the network. The more a node contributes towards the overall success of resource discovery, the higher its discovery energy is expected to be. A service energy is allocated for each of the biobot's shared resources in order to represent their usefulness, i.e., how popular these are among other network peers.

Both types of energies are initialised to a fixed value (set as a simulation parameter) and they are subsequently updated due to the occurrence of various system events. For example, when a biobot matches a query, the service energy of the biobot is increased. Similarly, a biobot that took part in a forwarding chain that resulted

in a successful query match is awarded discovery energy. Biobots need energy in order to perform various functions such as forwarding a query or establishing a new relationship. More interestingly, biobots whose energies exceed a certain threshold are allowed to perform additional functions such as migrating (relocating), replicating (cloning) or reproducing. On the other hand, a biobot whose energy is depleted is subject to death (removal from the system). Reproduction and death reflect a common property of distributed systems where components and nodes may be added or removed at any instance. A detailed discussion of energy consumption and rewarding mechanisms is available in [2].

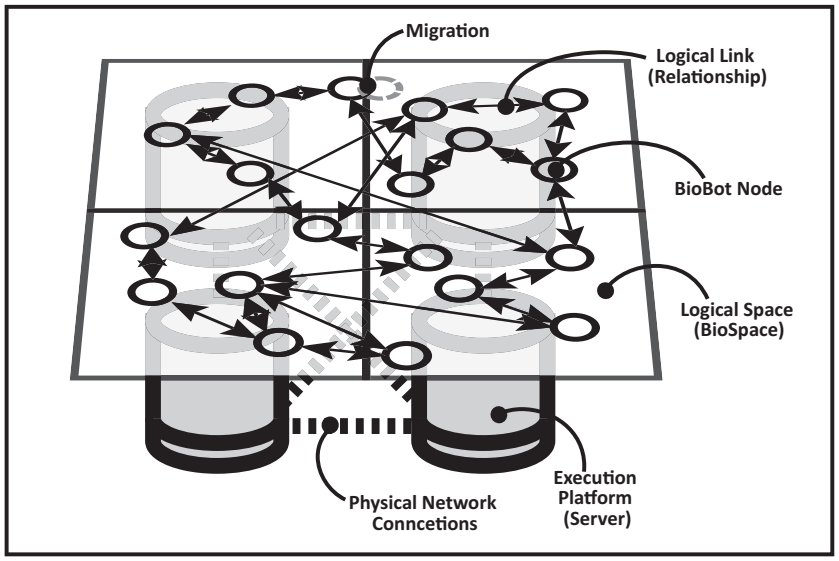


Fig. 7: Overview of the EDBO components.

Emergence in EDBO

The introduction of biologically-inspired behaviour and energy-based quantification as part of the biobots' life cycle allowed for the emergence of various macroscopic qualities and behaviours. While a detailed discussion of these findings is available in [2], a short summary follows:

- **Scalability**. Biobot population was able to scale to different demand peaks by introducing birth and death events. During phases of high query loads there was an increase in the birth events (usually of biobots with total service energy above average) while when the query load was lowered there was a significant increase in death events. Thus, the population of biobots was increasing in order to handle the extra load, and decreasing accordingly when the load was low.
- **Robustness and Availability**. Although extreme dynamic conditions were simulated, with biobots entering and leaving the network constantly, the overall connectivity was always maintained due to the autonomy provided to each biobot in forming new relationships.
- **Super-node formations**. Although initially the biobot network was completely unstructured, biobots invariantly achieved ad-hoc super-node network formation. This was visible during visualization of the simulation results where super-nodes, their connections, and the clustering of the network could be clearly observed.

6.2 The X-Machine Formalism

The stream X-machine[13] formalism has been briefly introduced in Section 5.2. More formally, a deterministic X-machine [58] is an 8-tuple $X = (\Sigma, \Gamma, Q, M, \Phi, F, q_0, m_0)$ where:

- Σ and Γ are the input and output alphabets respectively.
- Q is the finite set of states.
- M is the (possibly) infinite set called memory.
- Φ, the *type* of the machine X, is a set of partial functions φ that map an input and a memory state to an output and a possibly different memory state, $\varphi : \Sigma \times M \rightarrow \Gamma \times M$.
- F is the next state partial function, $F : Q \times \Phi \rightarrow Q$, which given a state and a function from the type Φ determines the next state. F is often described as a state transition diagram.
- q_0 and m_0 are the initial state and initial memory, respectively.

X-machines can be employed in similar modelling situations to those using statecharts and other analogous notations such as SDL. However, X-machines offer several significant advantages over them. First, they provide a mathematical modelling formalism for the system which in turn allows an X-machine specification to be model checked [67], facilitating thus the verification of the desired model properties. Moreover, X-machines offer a formal strategy for testing the implementation against the model which, under certain assumptions, is guaranteed to determine correctness [58].

X-machines have been shown to be particularly well suited for specification purposes [59] as well as for modelling agent-based systems [60–62, 64]. The latter makes the formalism appropriate for

[13] In the following sections the term X-machine is used to refer to the stream X-machine variant.

modelling of the EDBO case study which can be viewed as a multi-agent interaction model. Furthermore, X-machine models can be convienently used as a building block for a FLAME[14] model, allowing for simulations with a more general-purpose, agent-based platform.

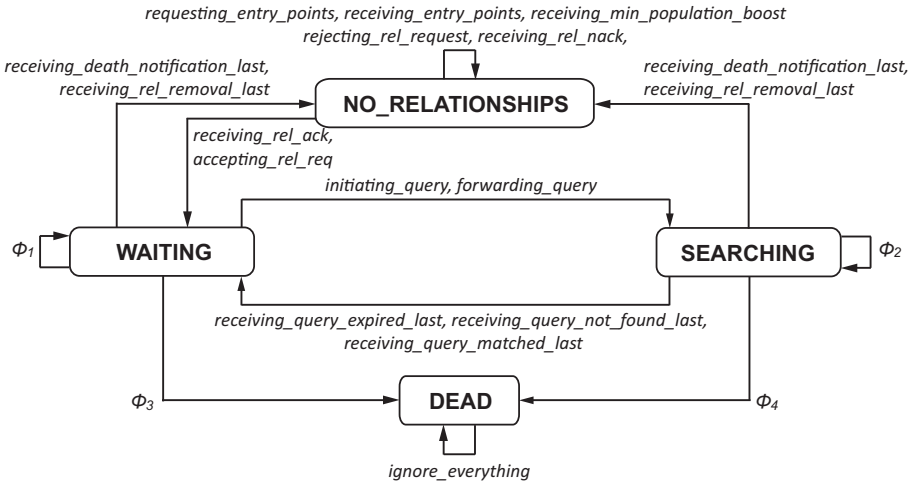
6.3 Modelling EDBO with X-machines

As already discussed in Section 6.1, EDBO represents an abstract distributed system that incorporates biological attributes and functions. During simulation scenarios, a number of different strategies and parameters were used in order to evaluate different properties of the model operation. An example of such a strategy is how a biobot should determine to which peer a relationship request should be sent; it could be the closest node (in terms of spatial distance), the most successful node (in terms of its energy levels) or the most similar node (in terms of the resources being offered).

Biobots' behaviour might change drastically depending on the particular strategies employed. Thus, each combination of strategies requires a distinct formal model. The model presented in this section describes the behaviour of biobots assuming the following selection of strategies:

- *Proximity relationship acquisition strategy*. When a biobot intends to establish a new relationship, it selects the biobot which is closest to it in terms of spatial distance.
- *Random discovery strategy*. A biobot which is unable to satisfy a query forwards it to a biobot randomly selected from its neighbour pool.

[14] FLAME was introduced in Section 5.2.



Φ_1 = { accepting_neighbourship_request, rejecting_neighbourship_request, receiving_neighbourship_ack, receiving_neighbourship_nack, receiving_death_notification, proposing_relationship_candidates, receiving_relationship_candidates_and_requesting, receiving_relationship_candidates_and_ignore, requesting_relationship_candidates, matching_query, receiving_min_population_boost, receiving_min_population_boost_and_relocating, receiving_min_population_boost_and_replicating, receiving_min_population_boost_and_reproducing, matching_query_and_relocating, matching_query_and_replicating, matching_query_and_reproducing }

Φ_2 = { forwarding_query, initiating_query, receiving_query_expired, receiving_query_not_found, receiving_query_matched, receiving_query_matched_and_relocating, receiving_query_matched_and_replicating, receiving_query_matched_and_reproducing } $\cup$ Φ_1

Φ_3 = { forwarding_query_and_dying, initiating_query_and_dying, matching_query_and_and_dying, receiving_relationship_candidates_requesting_and_dying, receiving_min_population_boost_and_dying }

Φ_4 = { receiving_query_matched_and_dying } $\cup$ Φ_3

Fig. 8: The X-machine model of a biobot.

– *Most successful partner movement strategy*. When a biobot relocates (migrates), it moves closer to the most successful neighbour as determined by the sum of its energies.
– *Energy-based birth and death strategies*. The decision to die or give birth (either by replicating or reproducing) is based on energy thresholds.

The modelling process was focused on the individual biobot as it represents the core of EDBO. Figure 8 depicts the state transition diagram of the X-machine model, consisting of the identified biobot states as well as the functions that label the transitions.

Formally, the set of inputs is $\Sigma =$ `biobots` $\times$ `service` $\times$ `message_type` where `biobots` is a sequence of `biobot` and `service` is a set of `keyword`, `biobot` and `keyword` being defined as basic types (could be any alphanumeric value), and `message_type` = {`search_query`, `query_expired`, `neighbourship_request`, `neighbourship_ack`,`neighbourship_nack`, `pending_death`, `request_relationship_candidates`, `reply_relationship_candidates` ...}. Practically, `biobots` consists of a set of biobot identifiers (uniquely identifying a biobot) which represent a query's path; with the first biobot id belonging to the query initiator and the last one to the next hop. A `service` input describes a specific service which is desired to be located and for some functions (e.g. relating to reproduction) could be null. Finally, `message_type` provides a semantic description of the input type such as a query expiration or a neighbourship request. For example, a received input of `((b3),nil, pending_death)` would imply that biobot `b3` is dying and thus it should be removed from the neighbourship buffer.

The set of outputs is $\Gamma =$ `MESSAGES` $\times$ `biobots` $\times$ `service`, where `MESSAGES`={`initiated_query`, `matched_query`, `forwarded_query`, `received_query_expired`, `neighbour_died`, `accepted_relationship_request`, `rejected_relationship_request`, `neighbour_accepted_us`, `neighbour_rejected_us`, `proposed_neighbours`, `relationship_request_sent` ...} and `biobots` and `service` as already defined in Σ.

The output parts are very similar to those defined in Σ with the only difference being that instead of a `message_type`, an element of the set `MESSAGES` is outputted. At this level of modelling, where a single biobot instance is being described, this output message has

a solely informative purpose. If, however, separate instances of this model were to be initiated and set up in order to communicate with each other (by using Communicating X-machines or FLAME for example), each output message would serve as a direct input to the stream of another biobot instance [68, 69].

The set of states is $Q =\{$ `no_relationships`, `waiting`, `searching`, `dead` $\}$, with the initial state, q_0, being `no_relationships`. The biobot can be seen either as "alive" or "dead", reflecting a common characteristic of distributed systems where existing components can be removed or new ones be added at any time during execution. In EDBO this is represented as birth and death of biobots. When a biobot first joins the network it has no relationships and thus can perform only a limited set of functions (state `no_relationships`). In order to establish its first connection, which will serve as an entry point to the network, a biobot has to issue a special request to the BioSpace, requesting a list of biobots that have communicated with BioSpace recently. Upon acquiring an entry point, the biobot is able to perform most of EDBO's functionality (state `waiting`). When a biobot initiates (on behalf of its user) or forwards (on behalf of another biobot) its first search query, it moves to the `searching` state and stays there until every active query (stored in the memory variable `active_queries`) has been matched, expired, or failed. Finally, it is possible that from either of the two states a biobot could move to the `dead` state if its latest action caused its energy levels to fall below a predefined death threshold (memory's constant `death_threshold`).

The memory is $M =$ (`biobot`, `active_queries`, `location`, `biobots`, `services`, `total_service_energy`, `total_`

discovery_energy, max_neighbours, relocation_discovery_cost, communication_discovery_cost, query_match_service_reward, similarity_threshold, death_threshold, initial_energy, neighbour_buffer_size), with biobots being a set of biobot, services being a set of service, location defined as a 2-tuple of (x,y), biobot and service being basic types, and the rest of the variables defined as the set of positive integers including 0.

The majority of the memory values represent global simulation parameters such as the initial discovery energy per biobot or the various energy rewards and costs occurring upon specific events. Different simulation scenarios would require different memory values which can be altered directly in the initial memory. An indicative initial memory for a scenario revolving around document services could be m_0 = (b1, 0, (23, 43), nil, ((english, italian, translation), (technical, documents, reports)), 2000, 2000, 50, 1, 0.8, 800, 40, 5, 2000, 5). In this case the biobot offers two unique services, each of them characterised by three different keywords.

The next-state partial function is F, described as a state transition diagram which is depicted in Figure 8. The type of the machine is Φ which is defined as the set of all functions shown in the same figure. The functions are discussed in more detail in the next chapter where they are implemented in XMDL and animated using the X-System animator.

6.4 Summary

This chapter has demonstrated the applicability of formally modelling the EDBO case study. Utilizing X-machines as a formal modelling construct provided a great insight into the inner workings of the EDBO paradigm, which in another case might have gone unnoticed. Having the complete model defined formally allows for the possibility of guaranteeing the presence of desired properties, as well as the absence of undesired ones, by applying directly formal verification techniques such as model checking. In addition, the X-machine testing strategy enables the model to be proven equivalent to any existing or future concrete implementation (for example in a programming language). Finally, this work served as a feasibility study of formally describing the EDBO case study, opening thus the possibility of transitioning to communicating X-machines and FLAME.

Of equal importance is the fact that by describing the model in a concise way, possible reuse of the EDBO model is enabled. Nonetheless, it is important to verify that the formal model is equivalent to the original work that introduced EDBO. This is tackled in the next chapter via informal validation, through the XMDL animator, and a separate research work in the field of Internet of Things and autonomous distributed sensor networks, that was based on the model presented in this chapter.

7 Validating the Formal Model of EDBO

Describing EDBO as an X-machine model allows for a concise and unambiguous understanding of the EDBO model. Moreover, it completely separates the model from its simulation platform. However, further steps need to be taken in order to increase confidence in the formal model and its equivalence to the original one.

Towards this direction, the formal model of the previous chapter along with its functions have been coded into XMDL in order to enable the animation of the model. XMDL serves as an interchange language for describing X-machine models and is supported by appropriate tools (syntax and type checker, visual editor, compiler, animator, etc.) [70]. The definition of the functions along with some of the informal animations are presented in Section 7.1.

Finally, a related but independent work has used the formal model of EDBO as the basis of a new set of tools that realize the concepts of EDBO in the field of Internet of Things and distributed sensor networks. This work has promising results that cross-validate the initial observations of the EDBO case study, further validating the formal model and its equivalence to the original one. Section 7.2 describes this work in more detail.

7.1 Animating EDBO with XMDL

In order to increase confidence in the formal model of EDBO, the complete X-machine model of a biobot has been coded in XMDL. This offers a number of advantages. First, different simulation scenarios can be easily defined and run with the XMDL animator [70], which can increase confidence in the model by providing the means for an informal validation. Additionally, the absence or presence of any desired properties can be verified by specifying conditions in temporal logic formulae and performing model checking [67]. Finally, by utilizing the X-machine testing strategy [58], any concrete implementation of the EDBO paradigm could be formally tested against this specification in order to guarantee equivalence between model and implementation.

XMDL allows the definition of all aspects of an X-machine model. Basic variable types come with built-in support and additionally it is possible to declare custom complex types. Figure 9 lists the code for the basic type definitions needed for modelling a biobot. In this example, a `biobot` can be any of `0`, `b1` up to `b10`. The `biobots` type is a sequence of `biobot` instances.

Similarly, the possible keywords and services offered by the biobots are defined. In a real world scenario this could not possible be defined in advance, but for the needs of animating the X-machine model this is necessary. Finally, the different simulation parameters, such as the total energy, the cost of performing different actions and the various thresholds, are also described as `Natural0`.

The various transitions between states can be easily described with the `transition` keyword, followed by an initial state, a function, and the resulting state. The state names correspond directly

```
#model edbo.

/* BIOBOT AND PATHS/LISTS/NEIGHBOURS */.
#type biobot = {0,b1,b2,b3,b4,b5,b6,b7,b8,b9,b10
   }.
#type biobots = sequence_of biobot.

/* SERVICE WITH KEYWORDS AND LIST (AVAILABLE
   SERVICES) */.
#type keyword = {streaming, radio, video,
   documents, translation, english, greek,
   italian, french, german, web, internet, law,
   computer_science}.
#type no_keyword = {0}.
#type services_with_keywords = set_of keyword.
#type service = services_with_keywords union
   no_keyword.
#type list_of_services = sequence_of service.

/* POSITION */.
#type xloc = Natural0.
#type yloc = Natural0.
#type location = (xloc, yloc).

/* SIMULATION PARAMETERS */.
#type total_discovery_energy = Natural0.
#type total_service_energy = Natural0.
#type active_queries = Natural0.
#type max_neighbours = Natural0.
#type relocation_discovery_cost = Natural0.
#type communication_discovery_cost = Natural0.
#type query_match_service_reward = Natural0.
#type similarity_threshold = Natural0.
#type death_threshold = Natural0.
#type initial_energy = Natural0.
#type neighbour_buffer_size = Natural0.
```

Fig. 9: Basic types for the XMDL model of EDBO.

to those defined in the state diagram machine in Figure 8, page 64. Part of the transition map is depicted in Figure 10.

The core components of the X-machine, as coded in XMDL, are shown in Figure 11. The memory for even a single biobot is a complicated list, holding the identifier of the current biobot, its neighbours, along with a list of active queries and a multitude of simulation parameters. This is necessary as each X-machine model needs to hold everything into its memory; there is no global or shared state between different models such as models of different biobots.

In the same manner, an `init_state` defines the initial state of the machine and `init_memory` its initial memory. This is an arbitrary example used to animate, as shown later, the X-machine model.

Finally, the lengthiest part of Figure 11 is the definition of `input` and `output` which correspond directly to the formal model's definition of Σ (set of inputs) and Γ (set of outputs). Although both of them have been detailed in Section 6.3, it is worth noting that each and every possible message type needs to be explicitly defined. At the end, the different input and output message types are defined separately, so that they can be mixed in the 3-tuples of `input` and `output` along with the sequence of `biobots` and `service`.

The final part of the XMDL model is the definition of the individual functions. Figure 12 presents indicatively the model's `forwarding_query` function coded in XMDL. Following the function's name, the input and the initial memory are defined. A function is applicable only if the input and the current memory matches those listed in its definition. Strings starting with a question mark (`?`) represent variables which can be manipulated later. Following the `if`

```
/* RELATIONSHIP TRANSITIONS */.
#transition (idle,
   accepting_neighbourship_request)=idle.
#transition (searching,
   accepting_neighbourship_request)=searching.
#transition (idle,
   rejecting_neighbourship_request)=idle.
#transition (searching,
   rejecting_neighbourship_request)=searching.
#transition (idle,receiving_neighbourship_ack)=
   idle.
/* ... */

/* DISCOVERY TRANSITIONS */.
#transition (idle,matching_query)=idle.
#transition (searching,matching_query)=searching
   .
#transition (idle,forwarding_query)=searching.
#transition (searching,forwarding_query)=
   searching.
#transition (idle,initiating_query)=searching.
#transition (searching,initiating_query)=
   searching.
#transition (searching,receiving_query_expired)=
   searching.
#transition (searching,
   receiving_query_expired_and_idling)=idle.
/* ... */
```

Fig. 10: State transitions of EDBO in XMDL.

```
#states = {idle,searching,no_relationships,dead
   }.

#memory (biobot, active_queries, location,
   biobots, list_of_services,
   total_service_energy, total_discovery_energy,
    max_neighbours, relocation_discovery_cost,
   communication_discovery_cost,
   query_match_service_reward,
   similarity_threshold, death_threshold,
   initial_energy, neighbour_buffer_size).

#init_state {idle}.
#init_memory (b1,0,(4,12),(b9,b4,b7,b10),{(radio
   ,web),(greek,documents),(streaming,video)
   },2000,2000,10,3,1,800,50,5,2000,6).

#type output_message = {initiated_query,
   matched_query,forwarded_query,
   received_query_expired,
   accepted_relationship_request,
   rejected_relationship_request,
   neighbour_accepted_us,neighbour_rejected_us,
   neighbour_died,proposed_neighbours,
   relationship_request_sent}.

#type message_type ={search_query,query_expired,
   neighbourship_request,neighbourship_ack,
   neighbourship_nack,pending_death,
   request_relationship_candidates,
   reply_relationship_candidates}.

#input (biobots, service, message_type).
#output (output_message, biobots, service).
```

Fig. 11: The basic properties of the X-machine EDBO model in XMDL.

keyword, an optional set of conditions can specify when the function should be applicable. The output messages along with the new memory of the X-machine are defined after the `then` keyword with variable manipulation, when necessary, taking place after the `where` keyword.

```
#fun   forwarding_query ( (?query_path,?service_request,search_query),
       (?own_id, ?active_queries, ?location, ?neighbours, ?available_services,
       ?total_service_energy, ?total_discovery_energy, ?max_neighbours_limit,
       ?current_neighbours_buffer,?relocation_discovery_cost,?death_threshold,
       ?communication_discovery_cost, ?query_match_service_reward, ?init_energy,
       ?similarity_threshold)) =
if     ?query_path belongs biobots and ?service_request belongs service and
       ?service_request not_belongs ?available_services and ?path_length > 1 and
       ?own_id belongs query_path and ?total_discovery_energy >= ?occurred_cost
then   ((forwarded_query, ?new_query_path, ?service_request),
       (?own_id, ?active_queries, ?location, ?neighbours, ?available_services,
       ?total_service_energy, ?new_discovery_energy, ?max_neighbours_limit,
       ?current_neighbours_buffer,?relocation_discovery_cost,?death_threshold,
       ?communication_discovery_cost, ?query_match_service_reward, ?init_energy,
       ?similarity_threshold))
where  ?path_length <- cardinality ?query_path and
       ?next_hop <- random_biobot(?neighbours) and
       ?spatial_diff <- spatial_difference(?location, ?next_hop) and
       ?occurred_cost <- communication_discovery_cost * ?spatial_diff
       ?new_discovery_energy <- ?total_discovery_energy - ?occurred_cost and
       ?new_active_queries <- ?active_queries + 1 and
       ?new_query_path <- ?next_hop addatendof ?query_path.
```

Fig. 12: The model's `forwarding_query` function coded in XMDL.

In the `forwarding_query` example of Figure 12, the code specifies that the function can only be applicable if the input contains a query path which represents a set of biobots, a service request and the string literal `search_query`. Moreover, the function states as a pre-condition that the requested service is not offered by this X-machine instance, otherwise the query would be matched instead of being forwarded. As a result, the X-machine picks randomly a neighbour and adds it at the end of the current query path, indicating the next hop of the query. Moreover, the associated discovery

cost (specified as a simulation parameter in the initial memory m_0) is subtracted from the biobot's total discovery energy.

Having the whole model coded in XMDL enables the animation of it with arbitrary simulation parameters. X-System is a collection of tools which, among others, features the compilation of XMDL models to Prolog code which can be subsequently animated via a Prolog interpreter. The user is allowed to input values directly to the interpreter and observe how the X-machine model behaves in response to that input. Alternatively, it is possible to specify an external file with a sequence of inputs, run the whole simulation and store the results in an output file for further examination.

Figures 13, 14, and 15 show an indicative list of inputs and simulation results, in an interleaved manner. The simulation parameters used for this run are those depicted in the XMDL model described above (see Figure 11, page 74).

Figure 13 depicts a scenario in which the requested service (as denoted by the keywords) can be provided by the biobot, thus the `matching_query` function is applied. In this scenario the discovery energy is increased, as a post-condition of this function, by 800 service energy points. The reward is specified in the initial memory of the biobot and specifically in the `query_match_service_reward` simulation parameter. Additionally, the resulting state continues to be `idle` as the query has been successfully matched and the biobot waits for further requests.

A second scenario, shown in Figure 14, involves querying for a service that the biobot cannot fulfill. The first part of the input, `[0]`, denotes that the query was initiated by the biobot's user, as opposed to another biobot. Consequently, the `initiating_query`

```
[[0,b4,b8,b5],[radio,web],search_query].

State : idle Input ?
Input : [[0, b4, b8, b5], [radio, web],
   search_query]

Applied Function : matching_query
Output: [matched_query, [0, b4, b8, b5], [radio,
    web]]
        Memory : [b1, 0, [4, 12], [b9, b4, b7,
           b10], [[radio, web], [greek,
           documents], [streaming, video]],
           2800, 2000, 10, 3, 1, 800, 50, 5,
           2000, 6]
        At State :idle

------------------------------------------------

[[0,b2,b9,b3],[streaming,video],search_query].

State : idle Input ?
Input : [[0, b2, b9, b3], [streaming, video],
   search_query]

Applied Function : matching_query
Output: [matched_query, [0, b2, b9, b3], [
   streaming, video]] Memory : [b1, 0, [4, 12],
   [b9, b4, b7, b10], [[radio, web], [greek,
   documents], [streaming, video]], 3600, 2000,
   10, 3, 1, 800, 50, 5, 2000, 6] At State :idle
```

Fig. 13: Animating the EDBO X-machine model with the X-System animator: matching 2 different queries.

```
[[0],[web,video],search_query].

State : idle Input ?
Input : [[0], [web, video], search_query]

Applied Function : initiating_query
Output: [initiated_query, [b1, b9], [web, video
   ]] Memory : [b1, 1, [4, 12], [b9, b4, b7, b10
   ], [[radio, web], [greek, documents], [
   streaming, video]], 3600, 1999, 10, 3, 1,
   800, 50, 5, 2000, 6] At State :searching

-------------------------------------------------

[[0],[translation,english],search_query].

State : searching Input ?
Input : [[0], [translation, english],
   search_query]

Applied Function : initiating_query
Output: [initiated_query, [b1, b9], [translation
   , english]] Memory : [b1, 2, [4, 12], [b9, b4
   , b7, b10], [[radio, web], [greek, documents
   ], [streaming, video]], 3600, 1998, 10, 3, 1,
    800, 50, 5, 2000, 6] At State :searching
```

Fig. 14: Animating the initiation of 2 different queries.

function is applied, indicating that the biobot will ask one of its neighbours for the requested service, on behalf of the user. In this case, there is no service energy reward as there was no match. Instead, the biobot suffers a discovery energy penalty of 1, as defined in the `communication_discovery_cost` simulation parameter, to account for the cost of communication to another biobot. Finally, the resulting state is that of `searching`, denoting that the biobot is waiting for an answer to its query. While the biobot is in the `searching` state, it can still process further queries, as indicated by the last input/output pair of Figure 14.

Figure 15 depicts 2 further scenarios. In the first input/output pair, a service is requested which cannot be provided by the current biobot. Unlike the previous example, however, this query was not initiated by the user. As the first part of the input indicates (`[0, b3, b10, b7, b5]`), the query has been forwarded to this biobot by another biobot with identifier "b5". As the biobot cannot fulfill this query, it forwards the query (function `forwarding_query`) to another neighbour not already in the forwarding chain (in this example "b9").

In the next scenario, the biobot receives a `query_expired` message type regarding an earlier query it had initiated on behalf of the user. After the query has been forwarded by this biobot (b1) to its neighbour (b7) and in turn to the neighbour's neighbour (b2), the query expired, indicating that the query could not be fulfilled. The state remains at `searching` as the biobot, at its current state, has another 4 active queries, as denoted by the second value of the memory tuple. Finally, the `[end_of_inputs]` special input terminates the simulation run.

```
[[0,b3,b10,b7,b5],[translation,greek],
   search_query].

State : searching Input ?
Input : [[0, b3, b10, b7, b5], [translation,
   greek], search_query]

Applied Function : forwarding_query
Output: [forwarded_query, [0, b3, b10, b7, b5,
   b9], [translation, greek]] Memory : [b1, 3,
   [4, 12], [b9, b4, b7, b10], [[radio, web], [
   greek, documents], [streaming, video]], 3600,
    1997, 10, 3, 1, 800, 50, 5, 2000, 6] At
   State :searching

-----------------------------------------------

[[0,b1,b7,b2],[web,video],query_expired].

State : searching Input ?
Input : [[0, b1, b7, b2], [web, video],
   query_expired]

Applied Function : receiving_query_expired
Output: [received_query_expired, [0, b7, b2], [
   web, video]] Memory : [b1, 4, [4, 12], [b9,
   b4, b7, b10], [[radio, web], [greek,
   documents], [streaming, video]], 3600, 1995,
   10, 3, 1, 800, 50, 5, 2000, 6] At State :
   searching

-----------------------------------------------

[end_of_inputs].

State : idle Input ?
Input : [end_of_inputs]
```

Fig. 15: Animating a query forwarding and a query expiration scenario.

Although this type of informal validation can be a great source of feedback for the designer of a formal model, it is impossible to observe properties on the macroscopic level, such as those emergent behaviours described in the original EDBO case study. The reason is that the animation deals with a single biobot and not a collection of biobots, communicating with each other.

FLAME is much more suited for this task as it allows the simulation of multiple agents in parallel and provides the necessary tools to automate large-scale simulation experiments. Alternatively, an independent implementation based on this formal model could also provide the means of validating the equivalence of the original model and the formal model devised previously. The next section presents work towards that direction.

7.2 Applying the Model

The formal model described in Chapter 6 has been further applied to a different application domain. In [3], we have proposed an architecture for building a self-organising overlay network of Cyber-Physical Systems (CPS), with a particular focus on sensor networks. This architecture is heavily inspired by nature, and specifically the fact that innovative design of individual nodes (microscopic level) can lead to the emergence of desired global properties (macroscopic level). It aims to offer an Internet of Things (IoT) solution of interacting CPS in the form of a middleware that facilitates interaction and interconnection of things in a distributed manner, providing scalability, self-adaptation, and self-organisation. The applicability of the pro-

posed architecture has been validated with a realistic design and an implementation solution that could support real-world scenarios.

This work was based directly on the formal model devised earlier and it serves as a proof of concept of EDBO. Given its promising results, discussed later in this section, it increases further the confidence to the formal model and its equivalence to the original EDBO case study by Paunovski [2].

The proposed architecture is directly based on EDBO's formal model described in Chapter 6 and it aims to serve as a main infrastructure that will enable any authorized consumer to perceive the required sensor or other types of data as if connected to the nervous system of an organism. It provides a solution that enables utilization of emergent behaviors to achieve availability and scalability of resources and the organisation and optimization of the network. The proposed middleware is a realization of the above-described research prototype network (EDBO) that achieves several self-* properties using the bio-inspired solution described earlier. Thus, it is composed of biobots (logical nodes) that are realized in a BioSpace (middleware). Biobots are capable of communicating with CPS if in range, enabling bridging of the desired CPS and the middleware.

EDBO for CPS facilitates a plug and play solution which allows the addition of such a system to the network at anytime. By adding a new CPS (provider) to the proposed middleware, which is based on the EDBO architecture, a service will be automatically provided and discovery will be enabled in a fully decentralized manner. Consumers realized as compatible clients will be able to discover and then consume all the provided data from the connected CPS without the

need of any central control. The abstract proposed architecture is depicted in Figure 16.

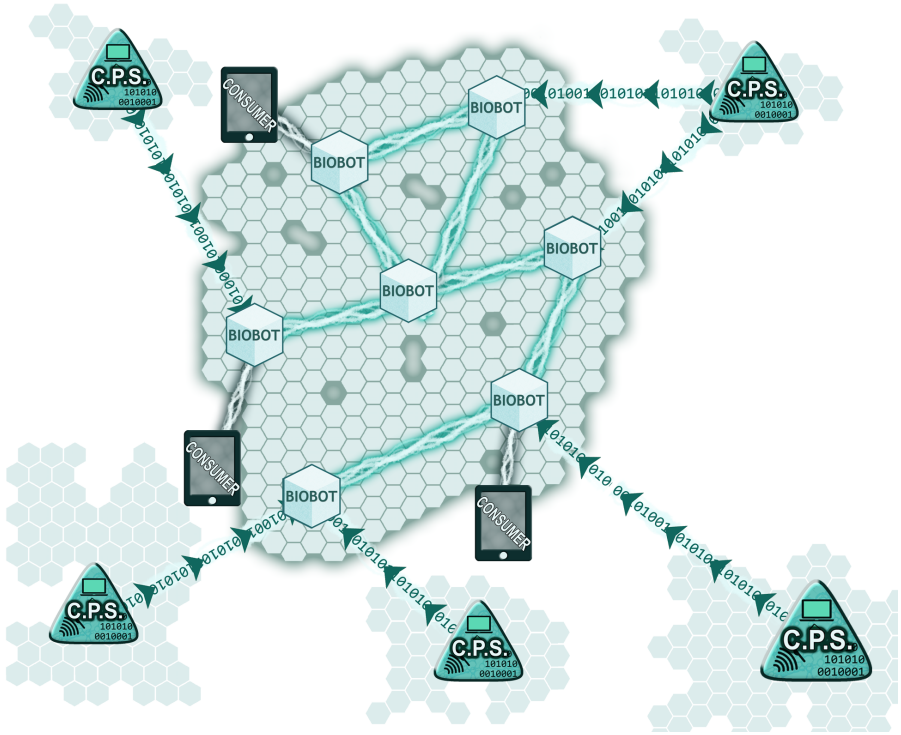


Fig. 16: An abstract representation of the proposed architecture. Reprinted from [3].

This architecture has been implemented using the Jadex Active Components (JAC) framework. This implementation offers the first concrete and independent implementation of the EDBO model, outside of the original simulation platform described in Chapter 5.

More importantly, the implementation has been thoroughly tested in order to evaluate whether EDBO's promising results for self-adaptivity and self-optimization in ADS could be demonstrated in

another scenario. The evaluation of the developed middleware has been focused on investigating the capabilities of the middleware towards realizing a self-adaptive, self-optimizing ADS. The optimal approach to determining the behavior of an ADS developed with the EDBO middleware was designed to be the runtime evaluation in the form of a case study comparing results of a system with no self-organisation capabilities to one that possesses them.

Past results on the evaluation of the model have shown that the most optimal strategy out of the available ones is that of the complex processing of the meta-data. This is the case where biobots tend to form meaningful relationships in the long term (self-optimization) and are capable of handling user behavior fluctuations by reevaluating current relationships and reproducing or replicating (self-adaptive). Consequently, the middleware was configured to use two different strategies for the same case and compare the results: the complex strategy and the random strategy.

Evaluation was carried out through the facilities provided by the JAC framework. Specifically, each AC has an inherent logging mechanism that can be activated and its entries can be viewed at real-time during execution via the component viewer offered by the JCC. Additionally, the design of the system enforces the observer pattern and hence a custom implementation of the abstract observer provided with the middleware offers the means to record all important data from the state changes and interactions of each biobot. A parser has been developed to read and review the resulting logs, producing information on the data sought after to determine the properties of the system. The data gathered regarded response accuracy (how many queries were satisfied on average) and response delay (aver-

age number of hops for each query). These values were measured at the very beginning of runtime and compared to a few moments later (short-term adaptivity) and nearing the end of runtime (long-term optimization), in order to determine if the desired properties emerged in the system. As such, these metrics constitute the Key Performance Indicators (KPIs) for the purposes of this quantitative evaluation: determining efficiency of the system over the passage of time (totally unstructured origins, adaptation, organised system in the end).

In order to determine how the system can adapt to user behavior, the evaluation scenario required the development of a querying function that sends various Keywords to the biobots, emphasizing a select Keyword at the beginning (to establish a baseline), again after a little time during runtime and near the end again. Naturally, this evaluation has been carried out as a demonstration of the system using an elementary case study that could simulate the following scenario: an ADS of weather sensors is deployed, with end-users in the beginning asking for data on specific locations and again later on. As an example, a hailstorm affecting crops in the area shortly after deployment prompts users to request the data from that location mostly, hence the system has to adapt to this sudden need and try to prepare for future needs. For some time, users also check on other areas but come back again shortly after to check, and again much later on again for the specific area, at which point the system should display its self-organisation capabilities.

The results gathered from the evaluation can be found in Table 4. Positive results on response accuracy demonstrate that the system manages to adapt to user behavior in the short-term, which

may become even more significant in the long-term as the system optimizes itself to serve more queries similar to the ones it received at the beginning and the first half of its runtime. Delay results did not provide much information regarding short-term capabilities of the system, which may be attributed to the low initial energy levels of biobots restricting more partners. Nonetheless, there were some changes towards the end of the evaluation hinting at the possibility of self-organisation considering that metric, too.

Time Frame	**Avg. Response Accuracy**		**Avg. Response Delay (hops)**	
	Random	*Meta-data*	*Random*	*Meta-data*
After 5 minutes	60.25%	64.09%	2.921	2.896
After 20 minutes	57.92%	71.23%	3.057	2.842
After 50 minutes	58.51%	89.56%	3.013	2.439

Table 4: Response accuracy and delay regarding the selected service for the runtime evaluation scenario using the random and complex (meta-data) strategies. Reprinted from [3].

As expected, the random strategy produced no positive results whatsoever regarding self-organisation. On the contrary, it appears that random partner selection leads to biobots having "bad" relationships and wasting energy on forwarding queries aimlessly. Unlike the evaluation results of the model, no super-bots have been observed, which is attributed to the limited resources available for evaluating an ADS at runtime as compared to simulation results. Finally, a criticism of the model, or at least for its initial configurations regarding energy levels, stemmed from the fact that several biobots that could not satisfy queries (or forward them) died after some time, which prohibited access to services they offered; in one

case, the single biobot offering a particular service died, effectively losing all access to that service in the network.

Overall, the system demonstrates the capacity to address the issue of load balancing and, moreover, it can do so without any human intervention. This is proving the self-organisation and self-adaptivity capabilities of the system. There is a differentiation here from the traditional approaches to load-balancing that similar solutions employ, where several factors are measured over time during network operation and then settings are adjusted to account for expected load balance depending on network size and connected users. Examples of these approaches that have been studied have used specific hardware settings of nodes such as remaining energy [71], or knowledge of the general structure of the network such as the very efficient SAAS-RWSNs [72]. In such cases the adjustments happen when these values reach the appropriate point that the new settings have to be applied, and this requires the existence of a centralized mechanism that keeps all these data for all network nodes so that they can be accessed by every node. Contrary to this process, the EDBO middleware is capable of adapting to the changes in its operational environment in a continuous rather than in a discrete manner. Further, it does so without any knowledge of the network structure. The advantage in this case would be the example of 990 users connected but experiencing a slow response because the next optimization is to take place when the number goes above 1000. In the case of the EDBO the system should have adapted gradually to better support these users already.

7.3 Summary

This chapter presented the validation of the EDBO formal model detailed in Chapter 6. First, XMDL was used to code the EDBO X-machine model along with its functions and allow for different simulation scenarios. This provided an informal validation, through animation, which increased the confidence to the formal model.

Second, the same formal model served as the basis of an independent work on a separate application domain, this of Internet of Things and distributed sensor networks. This work offered the first concrete and separate implementation of the EDBO model. Results gathered have been very promising and cross-validate the original results claimed by the EDBO study regarding the emergence of desired properties. This further validates the formal model of Chapter 6, as well as its equivalence to the original EDBO model.

8 Summary and Conclusions

We are living in a fast paced world where technological revolutions regularly affect our every day lives. People tend to get increasingly more interconnected to each other and their own multitude of devices. This greatly increases the complexity and the level of sophistication needed in order to manage all these devices and the new ways of interconnecting.

In the past decade there has been a strong interest in alternative approaches of tackling this kind of complexity: from IBM's autonomic computing to various technology leaders' renewed interest in deep learning. Many of these approaches suggest a decentralized connectivity scheme, such as the revolutionary Blockchain technology behind Bitcoin, in order to eliminate single points of failure, attack, or authority. They all have at least one thing in common: they are inspired by nature and biology.

In this direction, researchers and practitioners have been trying to engineer systems that harness emergent phenomena in order to tackle highly-complex tasks in a simple, elegant, and indirect way. The goal is to engineer simple interactions at the microscopic level, for example among the interactions of different agents, which hopefully will result to the desired complex effect at the macroscopic level, for example a self-healing network.

Doing that in a predictable or even disciplined way is very challenging. Foremost, there is a lack of consensus on what is emergence.

As detailed in Chapter 2, there has been a wealth of attempts to define emergence, spanning different disciplines and multiple millennia. Very often they contradict each other and most of them define emergence in a way that inherently opposes the concept of engineering emergence, making the term an oxymoron.

Nonetheless, there are researchers that have attempted to intentionally harness emergent phenomena, as presented in Chapter 3. By studying them it becomes apparent to the author that it is impossible to make generic claims about engineering emergence, as discussed in Chapter 4. This field, like most fields, is not yet mature enough to allow for universal, one-size-fits-all, solutions.

This work focused on the field of ADS. With this field in mind, as well as the points identified as important by other researchers who attempted to engineer emergence, Wolf and Holvoet's [9] definition of emergence was selected as a working definition.

Based on these findings, a framework was proposed in Chapter 4 which is founded on previous experience within the field of ADS and attempts to introduce emergent phenomena in those. The framework encompasses two important characteristics which the author deems necessary to engineer emergence in ADS: constant hypothesis formation and validation through simulation. The framework is rather abstract and the practice is often laborious. Nonetheless it presents a disciplined way of engineering emergent properties in ADS, based in real experience.

Part of the experience which led to this framework is the EDBO case study, devised by Paunovski on his PhD thesis [2] and presented in more detail in Chapters 5 and 6. EDBO is a notable attempt at engineering emergent properties in ADS with very promising results.

By introducing biological properties at the agent level and allowing for biobot to biobot and biobot to biospace (environment) interactions, global phenomena such as super-cluster formation, network redundancy and self-optimization have been observed.

An identified weakness of EDBO is the coupling of the model to the simulation platform itself, both custom developed in a general purpose programming language. This hinders other people from understanding the model in an unambiguous way and prevents possible reuse of the model in the design of other systems and case studies. Moreover, it makes impractical to simulate the same case study in a different simulation platform and thus it impedes the cross-validation of the original findings.

This work tackles these issues by devising a formal, platform- and implementation-independent model of EDBO (Chapter 6). The model is built on the X-machine formalism, allowing the use of a number of tools built around it to be used in the model itself. Chapter 7 presents the results of animating that model with X-System, increasing confidence to the model as well as understanding of the original EDBO itself.

The formal model further enabled reuse of EDBO in a different application domain. In [3], the formal model was used as a basis of implementing a middleware for a sensor network distributed architecture. This work demonstrated very promising results as well, many of which cross-validate the original findings of the EDBO case study. The next section presents possible directions that this work can take in the future.

8.1 Future Work

The formal model of EDBO, presented in Chapter 6, sheds light on the intricacies of the EDBO case study and enables other parties to reuse its design in other systems and application domains. The next step towards that direction is to formally test the implementation of the individual biobots by using the X-machine testing strategy. This requires the careful design of a set of testing simulation scenarios.

A disadvantage of the current mode is its inability to model communication. This is a crucial feature of EDBO as it relies on communication among biobots and between biobots and the environment in order to exhibit emergent properties at the macroscopic level. In order to model these communication channels, another variant of this formalism can be used known as the Communicating X-machine (CXM), briefly introduced in Chapter 5. Using CXM, the communication can be formally defined by allowing biobot functions to receive input from and write output to other biobots, or the environment. This can allow for the formal modelling of biobot-to-biobot and biobot-to-BioSpace communication, which can be later validated and verified informally through an existing animator. The current X-machine model can serve as a building block for a CXM EDBO model.

Nonetheless, animating complex scenarios in a practical and efficient manner is infeasible with the current CXM tool set. An alternative solution targeted to such use cases is the FLAME simulator, introduced in Chapter 5. By using FLAME to simulate the EDBO case study it would be possible to observe emergent properties at the macroscopic level, allowing for easier experimentation with different simulation scenarios, starting agents, simulation values, and

strategies. Furthermore, it would make possible the cross-validation of the original findings by using an independent, general-purpose, simulation platform.

As described earlier, FLAME is founded on the theory of CXM and as such the transition from an X-machine or a CXM model to a FLAME model is possible and even convenient when compared to the transition from another formal method. Nonetheless, it does not come without its challenges. A particular difficulty lies in the fact that FLAME does not allow for environment variables and functions. As already mentioned, EDBO heavily relies on the environment to carry out energy logistics and facilitate relationship acquisition, relocation, reproduction, and the like. A possible solution would be to model the environment as a separate agent of a distinct agent type, which holds on its memory system-wide values which are expected to change (e.g. total amount of energy or total number of relationships).

Another concern is the inability of FLAME to model agent birth. While it provide the means for simulating the death of a previously instantiated agent, there is no way to instantiate a new agent during a simulation. This hinders the process of modelling agent self-replication and sexual reproduction, both of which are integral parts of the EDBO case study. A workaround on this issue would be to instantiate a number of inactive agents at the start of the simulation which can later become active, emulating agent births as needed. These agents will form an "unborn agent pool" from which other agents can draw when they decide to replicate or reproduce. Alternatively, FlameGPU could also be a viable option as it claims to support agent birth and reproduction.

Finally, it would be interesting to apply the framework proposed in Chapter 4 as well as the core principles of EDBO in different application domains. ADS is a large field which encompasses any distributed network consisting of any type of nodes, which has been artificially designed; i.e. it is not natural. As such, potential case studies include peer-to-peer sharing networks, wireless sensor networks (WSNs), decentralised approaches to Web services, the World Wide Web, and many others.

References

1. Fromm, J.: Types and forms of emergence. Technical Report AO/0506028, Distributed Systems Group, Kassel University (2005)
2. Paunovski, O.: Exploring Emergent Phenomena: Towards Analysis and Synthesis of Emergent Formations in Complex Systems. PhD thesis, University of Sheffield (2012)
3. Eleftherakis, G., Pappas, D., Lagkas, T., Rousis, K., Paunovski, O.: Architecting the IoT paradigm: A middleware for autonomous distributed sensor networks. International Journal of Distributed Sensor Networks **11**(12) (2015)
4. Goldstein, J.: Emergence as a construct: History and issues. Emergence **1**(1) (1999) 49–72
5. Bush, S.F., Kulkarni, A.B.: Engineering emergent protocols. White paper, General Electric Global Research (2001) Available online at http://www.crd.ge.com/˜bushsf/EmergenceWhitePaper.pdf.
6. Stepney, S., Polack, F., Turner, H.: Engineering emergence. In: ICECCS '06 Proceedings of the 11th IEEE International Conference on Engineering of Complex Computer Systems. (2006) 89–97
7. Eleftherakis, G., Paunovski, O., Rousis, K., Cowling, A.J.: Harnessing emergent properties in artificial distributed networks: an experimental framework. In Ritson, C., Andrews, P., Stepney, S., eds.: 4th Workshop on Complex Systems Modelling and Simulation, Paris, France, Luniver Press (2011) 141–144
8. Eleftherakis, G., Paunovski, O., Rousis, K., Cowling, A.: Emergent distributed bio-organization: A framework for achieving emergent properties in unstructured distributed systems. In Fortino, G., Badica, C., Malgeri, M., Unland, R., eds.: Intelligent Distributed Computing VI. Volume 446 of Studies in Computational Intelligence. Springer Berlin Heidelberg (2013) 23–28
9. Wolf, T.D., Holvoet, T.: Emergence versus self-organisation: Different concepts but promising when combined. In: Engineering Self-Organising Systems. (2004) 1–15
10. Rousis, K., Eleftherakis, G., Cowling, A.J.: An engineering perspective on emergence. In Bratanis, K., Dranidis, D., Ketikidis, P., Lazouras, L., Nikolaidou, E., eds.: 7th Annual South East European Doctoral Student Conference, Thessaloniki, Greece, SEERC (2012) 453–463
11. Rousis, K., Eleftherakis, G., Cowling, A.J.: A literature survey on engineering emergence. In Gonidis, F., Gkasis, P., Lazouras, L., Stamatopoulou, I., eds.: 8th

Annual South East European Doctoral Student Conference, Thessaloniki, Greece, SEERC (2013)
12. Eleftherakis, G., Paunovski, O., Rousis, K., Cowling, A.J.: Harnessing Emergent Properties in Artificial Distributed Networks: An Experimental Framework. In Stepney, S., Welch, P., Andrews, P.S., Ritson, C.G., eds.: Proceedings of the 2011 Workshop on Complex Systems Modelling and Simulation, Paris, France, August 2011, Luniver Press (2011) 141–144
13. Rousis, K., Eleftherakis, G., Paunovski, O., Cowling, A.J.: Formal modelling of a bio-inspired paradigm capable of exhibiting emergence. In Ivanovic, M., Budimac, Z., Radovanovic, M., eds.: Balkan Conference in Informatics, 2012, BCI '12, Novi Sad, Serbia, September 16-20, 2012, ACM (2012) 223–228
14. Aristotle: Metaphysics. Edited and translated by W. D. Ross. NuVision Publications (2009)
15. Smith, A.: An Inquiry into the Nature and Causes of the Wealth of Nations. General Books LLC (2010)
16. Soanes, C., Stevenson, A.: Concise Oxford English Dictionary. 11 edn. Oxford University Press (2004)
17. Holland, J.H.: Emergence: From Chaos to Order. Addison-Wesley (1998)
18. Lewes, G.H.: The Problems of Life and Mind. Truebner (1879)
19. Anderson, P.W.: More is different. Science **177**(4047) (1972) 393–396
20. Rosen, R.: Anticipatory Systems: Philosophical, Mathematical, and Methodological Foundations. Oxford Press (1985)
21. Cariani, P.: On the Design of Devices with Emergent Semantic Functions. PhD thesis, State University of New York, Binghamton (1989)
22. Heylighen, F.: Modelling emergence. World Futures: the Journal of General Evolution **31**(Special Issue on Emergence) (1991) 89–104
23. Corning, P.A.: The re-emergence of "emergence": A venerable concept in search of a theory. Complexity **7**(6) (2002) 18–30
24. Bar-Yam, Y.: Dynamics of Complex Systems. Addison-Wesley (1997)
25. Ronald, E., Sipper, M., Capcarrere, M.: Design, observation, surprise! A test of emergence. Artificial Life **5**(3) (1999) 225–239
26. Fromm, J.: Ten questions about emergence. Technical Report AO/0509049, Distributed Systems Group, Kassel University (2005)
27. Kubík, A.: Toward a formalization of emergence. Artificial Life **9**(1) (2002) 41–65
28. Damper, R.I.: Editorial for the Special Issue on "Emergent Properties of Complex Systems": Emergence and levels of abstraction. International Journal of Systems Science **31**(7) (2000) 811–818

29. Deguet, J., Demazeau, Y.: A complexity based feature to support emergence in MAS. In: Proceedings of the International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS), Springer (2005) 616–619
30. Gordon, R.: The emergence of emergence: a critique of "design, observation, surprise!". Rivista di biologia **93**(2) (2000) 349–356
31. Crutchfield, J.P.: The calculi of emergence: Computation, dynamics, and induction. Physica D: Nonlinear Phenomena **75**(1–3) (1994) 11–54
32. Chalmers, D.J.: Strong and weak emergence. In Davies, P., Clayton, P., eds.: The Re-Emergence of Emergence. Oxford University Press (2006)
33. Stephan, A.: Varieties of emergence in artificial and natural systems. Zeitschrift für Naturforschung C-A Journal of Biosciences **53**(7–8) (1998) 639–656
34. Stephan, A.: Varieties of emergentism. Evolution and Cognition **5**(1) (1999)
35. Chalmers, D.J.: The Conscious Mind: In Search of a Fundamental Theory. Oxford University Press, Oxford (1996)
36. Bar-Yam, Y.: A mathematical theory of strong emergence using multiscale variety. Complexity **9**(6) (2004) 15–24
37. Laughlin, R.B.: A Different Universe: Reinventing Physics from the Bottom Down. Basic Books (2005)
38. Bedau, M.A.: Weak emergence. Noûs **31**(11) (1997) 375–399
39. Bedau, M.A.: Downward causation and the autonomy of weak emergence. Principia **6**(1) (2002) 5–50
40. Poundstone, W.: The Recursive Universe. Contemporary Books, Chicago (1985)
41. Heylighen, F.: Relational closure: a mathematical concept for distinction-making and complexity analysis. In: Cybernetics and Systems '90, World Science Publishers (1990) 335–342
42. Wolfram, S.: Universality and complexity in cellular automata. Physica D: Nonlinear Phenomena **10**(1–2) (1984) 1–35
43. Eppstein, D.: Wolfram's classification of cellular automata (2000) Available online at http://www.ics.uci.edu/~eppstein/ca/wolfram.html.
44. Bush, S.F.: Active virtual network management prediction. In: Parallel and Discrete Event Simulation Conference. (1999)
45. Abbott, R.: Emergence explained: Abstractions: Getting epiphenomena to do real work: Essays and commentaries. Complexity **12**(1) (2006) 13–26
46. Ronald, E.M.A., Sipper, M., Capcarrère, M.S.: Testing for emergence in artificial life. In: Proceedings of the 5th European Conference on Advances in Artificial Life. ECAL '99, London, UK, Springer-Verlag (1999) 13–20
47. Polack, F., Stepney, S.: Emergent properties do not refine. Electronic Notes in Theoretical Computer Science **137**(2) (2005) 163–181

48. Banach, R., Poppleton, M.: Retrenchment: An engineering variation on refinement. In Bert, D., ed.: B98 The 2nd International B Conference, Springer-Verlag (1998) 129–147
49. Banach, R., Jeske, C., Fraser, S., Cross, R., Poppleton, M., Stepney, S., King, S.: Approaching the formal design and development of complex systems: The retrenchment position. In: Workshop on Software and Complex Systems, 9th IEEE International Conference on Engineering of Complex Computer Systems. (2004)
50. Fromm, J.: On engineering and emergence. Technical Report AO/0601002, Distributed Systems Group, Kassel University (2006)
51. Welch, P.H., Wallnau, K.C., Klein, M.: Engineering emergence: an occam-pi adventure. In: CPA 2009 Proceedings of the 32nd Communicating Process Architectures Conference. (2009)
52. Ulieru, M., Doursat, R.: Emergent engineering: a radical paradigm shift. International Journal of Autonomous and Adaptive Communications Systems **4**(1) (2011) 39–60
53. Paunovski, O., Eleftherakis, G., Cowling, T.: Disciplined exploration of emergence using multi-agent simulation framework. Computing and Informatics **28**(3) (2009) 369–391
54. Deissenberg, C., van der Hoog, S., Dawid, H.: Eurace: A massively parallel agent-based model of the european economy. Applied Mathematics and Computation **204**(2) (2008) 541 – 552 Special Issue on New Approaches in Dynamic Optimization to Assessment of Economic and Environmental Systems.
55. Steinsiek, S., Frixel, S., Stagge, S., Bettenbrock, K.: Characterization of E. coli MG1655 and frdA and sdhC mutants at various aerobiosis levels. Biotechnology (2011) [Epub ahead of print].
56. Pogson, M., Holcombe, M., Smallwood, R., Qwarnstrom, E.: Introducing spatial information into predictive NF-kB modelling — an agent-based approach. PLoS ONE **3**(6) (2008) e2367
57. Eilenberg, S.: Automata, Languages, and Machines. Volume A. Academic Press (1974)
58. Holcombe, M., Ipate, F.: Correct systems: building a business process solution. Applied Computing Series. Springer-Verlag, Berlin, Germany (1998)
59. Holcombe, M.: X-machines as a basis for system specification. Software Engineering Journal **3**(2) (1988) 69–76
60. Kefalas, P.: Formal modelling of reactive agents as an aggregation of simple behaviours. In: Methods and Applications of Artificial Intelligence. Volume 2308 of Lecture Notes in Computer Science LNCS., Springer-Verlag (2002) 461–472

61. Eleftherakis, G., Kefalas, P., Sotiriadou, A.: Formal verification of agent models. In: Proceedings of the 2nd Hellenic Conference on AI (SETN02). (2002) 425–435
62. Kefalas, P., Holcombe, M., Eleftherakis, G., Gheorghe, M.: A Formal Method for the Development of Agent-Based Systems. In: Intelligent Agent Software Engineering. Idea Group Publishing (2003) 68–98
63. Balanescu, T., Cowling, A.J., Georgescu, M., Holcombe, M., Vertan, C.: Communicating stream X-machines are no more than X-machines. Journal of Universal Computer Science **5**(9) (1999) 494–507
64. Coakley, S., Smallwood, R., Holcombe, M.: Using X-machines as a formal basis for describing agents in agent-based modelling. In: Proceedings of the 2006 Agent-Directed Simulation Conference. (2006)
65. Holcombe, M., Coakley, S., Smallwood, R.: A general framework for agent-based modelling of complex systems. In: Proceedings of the 2006 European Conference on Complex Systems. (2006)
66. Suda, T., Nakano, T., Moore, M., Enomoto, A., Fujii, K.: Biologically inspired approaches to networks: The bio-networking architecture and the molecular communication. In Li, P., Yoneki, E., Crowcroft, J., Verma, D., eds.: Bio-Inspired Computing and Communication. Volume 5151 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2008) 241–254
67. Eleftherakis, G., Kefalas, P.: Formal verification of generalised state machines. In: PCI '08: Proceedings of the 2008 Panhellenic Conference on Informatics, IEEE Computer Society (2008) 227–231
68. Kefalas, P., Eleftherakis, G., Kehris, E.: Modular Modeling of Large-Scale Systems using Communicating X-machines. In: 8th Panhellenic Conference on Informatics. Volume I., Cyprus (2001) 20–29
69. Stamatopoulou, I., Kefalas, P., Gheorghe, M.: Modelling the dynamic structure of biological state-based systems. BioSystems **87**(2-3) (2007) 142–149
70. Kefalas, P., Eleftherakis, G., Sotiriadou, A.: Developing tools for formal methods. In: PCI '03: Proceedings of the 2003 Panhellenic Conference on Informatics, Washington, DC, USA, IEEE Computer Society (2003) 625–639
71. Zhang, Z., Wang, Y., Song, F., Zhang, W.: An energy-balanced mechanism for hierarchical routing in wireless sensor networks. International Journal of Distributed Sensor Networks **2015** (2015)
72. Kim, K., Ha, C., Ok, C.: Network structure-aware ant-based routing in large-scale wireless sensor networks. International Journal of Distributed Sensor Networks **501** (2015) 521784

73. Eleftherakis, G., Kostic, M., Rousis, K., Vasilescu, A.: Stigmergy inspired approach to enable agent communication in emergency scenarios. In: Proceedings of the 7th Balkan Conference on Informatics Conference. BCI '15, New York, NY, USA, ACM (2015) 22:1–22:8
74. Eleftherakis, G., Rousis, K., Ketikidis, P.: Innovation resilience by engineering emergence. In: 8th HSSS National and International Conference, Thessaloniki, Greece (2012)
75. Eleftherakis, G., Rousis, K., Cislaghi, M., Somaschini, S.: Security requirements in judicial information systems: Experience from a European project for judicial cross-border collaboration. Special Issue on Information Assurance and Data Security, Journal of Information Assurance and Security (JIAS) **4**(6) (2009) 519–529
76. Eleftherakis, G., Rousis, K., Cislaghi, M., Somaschini, S.: A view on the role of information security on ICT-enabled judicial systems. In van Engers, T., Eleftherakis, G., eds.: 1st International Conference on ICT Solutions for Justice (ICT4Justice '08), Thessaloniki, Greece, CEUR-WS.org (2009) 35–46
77. Cislaghi, M., Somaschini, S., Eleftherakis, G., Rousis, K.: A new approach to international judicial cooperation through secure ICT platforms. In van Engers, T., Eleftherakis, G., eds.: 1st International Conference on ICT Solutions for Justice (ICT4Justice '08), Thessaloniki, Greece, CEUR-WS.org (2009) 24–34
78. Rousis, K., Eleftherakis, G., Cowling, A.J.: A formal approach to service composition using stream X-machines. In Psychogios, G., Proedrou, F., Kalyva, F., Eleftherakis, G., eds.: 5th Annual South East European Doctoral Student Conference, Thessaloniki, Greece, SEERC (2010) 389–400

A List of Author's Publications

Table 5 contains a list of papers that have been already published in international journals or conference proceedings and have been either authored or co-authored by this author.

Field	**Publications**
Emergence and Complex Systems	[13], [10], [11], [3], [8], [12], [73], [74]
Information Security	[75], [76], [77]
Formal Methods	[78]

Table 5: List of the author's publications in international conference proceedings and journals so far.