Biologically Inspired Near Extinct System Reconstruction

Athanasios Bibas, George Spanoudakis, Christos Bellos, Dimitrios I. Fotiadis, *Senior Member*, *IEEE*, and Dimitrios Koutsouris, *Senior Member*, *IEEE*

Abstract— **Recovery software system operations from a state of extensive damage without human intervention is a challenging problem as it may need to be based on a different infrastructure from the one that the system was originally designed for and deployed on (i.e., computational and communication devices) and significant reorganization of system functionalities. In this paper, we introduce a bio-inspired approach for reconstructing nearly extinct complex software systems. Our approach is based on encoding a computational DNA (co-DNA) of a system and computational analogues of biological processes to enable the transmission of co-DNA over computational devices and, through it, the transformation of these devices into system cells that can realise chunks of the system functionality, and spread further its reconstruction process.**

I. Introduction

Complex software systems can suffer from massive failures, due to environmental factors (e.g., massive loss of computational or communication infrastructures, dramatically increased conditions of use due to some social or physical emergency, security attacks) or internal factors (e.g., faults in key system components). Such factors can bring software systems to a *near extinction state*, i.e., a state where a large number of system components become non-operational or physically destroyed, including components with key local or global control responsibilities. In such circumstances, the survived components of the system may also have to operate under an increasingly adverse and continually changing environment (e.g., in cases where damage has been caused by on-going external disaster).

Recovering system operations from a near extinction state is a challenging open problem as it may require excessive system reconstruction using a physical infrastructure (i.e., computational and communication devices) that is different from the one that the system was originally designed for and operated on. Besides this, the system may also need to redirect its focus from a normal and fully functional operating mode to a basic survival operation mode. As a consequence, the software functionality may have to be re-modularised and re-allocated onto computational and communication devices with very different characteristics than the ones that were used originally. A further complication is that a near extinction state might continue to deteriorate, in an unpredictable manner, to full extinction whilst the system itself is trying to recover.

Scenarios of extensive system damage and reconstruction arise often in crisis management, when significant parts of the ICT infrastructure and the software systems running on it (e.g., communication system for emergency responders) may be lost following some natural or other disaster. In such cases, reinstating the damaged system might not be possible through centralized servers and the original communication infrastructure (e.g., WiFi), and require reconstruction from any survived system components (e.g., survived devices of emergency responders) and use of alternative communication means (e.g., ad-hoc heterogeneous networks).

Current state of the art techniques on software system dependability, resilience and recovery address some aspects of this problem (e.g., forecasting dependability levels [1], increasing system resilience through redundancy [2], and development of autonomic self-healing system capabilities [3,4]). They cannot, however, support system reconstruction from a near extinction state.

Software system re-construction from a "near extinct" state is an activity that needs to be undertaken by the survived components of the near extinct system in an *autonomic* manner, i.e., in a self-triggered and self-managing mode without assuming or depending on any form of human intervention. Providing a solution to this problem is more challenging than repairing a system following the detection of faults. This is because it must deal with the extensive loss of key system components and services, the computational infrastructure where the system is deployed, and possibly key system administration actors and functions. And whilst autonomic system behaviour is necessary in such circumstances, it is not sufficient on its own for reconstructing the system.

Since approaches, which are based solely on software and systems engineering methods, have failed to support excessive system reconstruction, to achieve a breakthrough we need to undertake a different and inter-disciplinary approach. Biological organisms have effective DNA-driven reconstruction and recovery mechanisms [5-7]. Inspired by this observation, our approach is to develop a solution for

Manuscript received July 1, 2013.

A. Bibas, is with the 1st Department of Otolaryngology – Head & Neck Surgery, University of Athens, Greece. (e-mail: thanosbibas@hotmail.com).

G. Spanoudakis is with the City University London, Northampton Square, London, EC1V 0HB, UK (corresponding author phone: +442070408413; fax: +442070400244; e-mail: g.e.spanoudakis@city.ac.uk).

C. Bellos and D. Koutsouris are with the Biomedical Engineering Laboratory, Institute of Communications and Computer Systems (ICCS)-National Technical University of Athens (NTUA), Greece. (e-mail: cbellos@biomed.ntua.gr, e-mail: d.koutsouris@biomed.ntua.gr).

D. I. Fotiadis is with the Unit of Medical Technology and Intelligent Information Systems, Dept of Materials Science and Engineering, University of Ioannina, Greece. (e-mail: fotiadis@cc.uoi.gr).

978-1-4799-3163-7/13/$31.00 ©2013 IEEE

extensive system reconstruction based on mechanisms operating using similar principles and processes.

A key element of our approach is the concept of the *computational DNA* of a software system (referred to as "co-DNA" in the rest of this paper). co-DNA models and encapsulates the basic functional units of a complex software system that are required in order to fully reconstruct it, properties of these units that need to be taken into account in system reconstruction and re-organization, and possible ways of re-combining the units into alternative structures. Inspired by the biological analogue that any individual cell in a multi-cellular organism has the same information encoded in its DNA regardless of its committed differentiated path, the co-DNA is physically present in all the functional units of the original system enabling them to function as *system cells*. Then, as in biological organisms, the function of each of these system cells will be determined by the unlocked part of the co-DNA that exists in it.

When a system gets in a near extinction state, its co-DNA can be transmitted across external computational and communication devices that are identified by the survived components of the system, in order to recruit and make them function as system cells as part of the system reconstruction process. Co-DNA transmission and activation is performed by computational processes analogous to biological processes of transmitting DNA and enabling the functioning of cells of biological organisms with proven effectiveness for such organisms. Two types of responses of biological cells are of particular relevance: *tissue repair* and *tissue regeneration* [8-13]. Tissue repair refers to the physiologic adaptation of an organ after injury in an effort to re-establish continuity, and involves recruitment of cell types, different from the original ones, in an effort to establish tissue continuity, without resulting in the exact replacement of lost/damaged tissue. Tissue regeneration refers to the replacement of lost/damaged tissue with an exact copy, such that both morphology and functionality are completely restored. Both these responses are driven and enabled by the DNAs of the relevant organisms. They are also *realised locally* without any form of centralised control. The latter characteristic of biological repair processes is particularly relevant in the case of system reconstruction from a near extinct state, as in such cases both the components that undertake the key control functions of the original system and critical communication lines between system components might have been lost.

As shown in part (a) of Fig. 1, for example, a software system may involve several devices of different computational capabilities (e.g., servers, laptops, tablets, and smart phones). Each of these devices will incorporate the co-DNA of the system with a certain set of genes in it unlocked, as required for the realisation of the functional role of the device in the system. Furthermore, the role of the device (system cell) can be dynamically activated and transformed by altering the configuration of the locked and unlocked genes of the co-DNA of the system that is stored in it dynamically.

The role of co-DNA is also fundamental in the system reconstruction process. System cells that have not been destroyed in a near extinction state can transmit the co-DNA of the system to any computational and communication devices that they can identify, as shown in parts (b1) and (b2) of Fig. 1, respectively. Subsequently, if the co-DNA is accepted by the destination device, it can unlock some of its genes in order to make the device assume specific operations as part of the software system that the co-DNA encodes. The unlocking of genes in the co-DNA that has arrived on a new device can also make it transmit itself to other devices, as shown in part (b3) of Fig. 1.

The genes that will be unlocked in the co-DNA that arrives on a device are determined by a *special regulatory gene* that is always unlocked. This gene encodes the core bootstrapping operations for unlocking other genes. It also incorporates primitive capabilities for detecting the resources and computational capabilities of the device in order to make unlocking decisions. As in biological organisms, this process can enable the reconstruction of a near extinct system through the use of different computational resources, as shown in part (c) of Fig. 1.

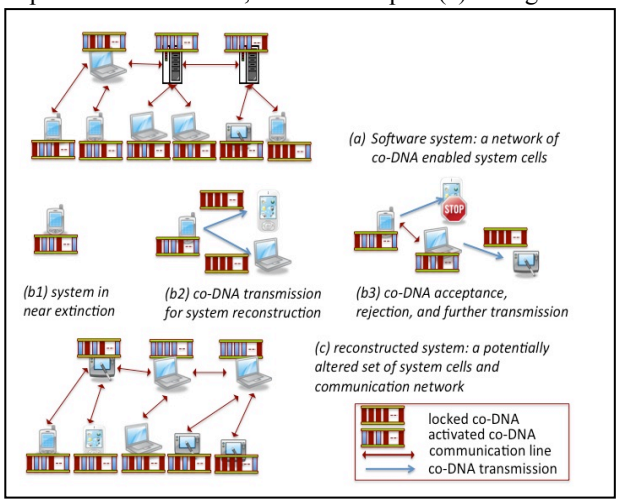

Fig. 1. co-DNA enabled software systems and system reconstruction

II. CO-DNA MODELLING

DNA is a complex molecule that contains the genetic information for the development and function of almost all living organisms, organised in genes. Each gene contains the information required for the production of a protein. Other DNA segments have structural purposes or are involved in the regulation of gene expression. Regardless of their function, however, all cells have the same information in their DNA and what differentiates their type (e.g., muscle cell, heart cell) is the specific set of genes that are unlocked in them (the rest of the genome is present but 'locked').

In an analogous manner, the co-DNA of a software system is a *library of computational genes*, each encoding a functional unit of the system with descriptions of different key characteristics of the unit and the code that implements

it. The co-DNA includes also computational genes that can realise the process of system reconstruction when this becomes necessary. The genes of co-DNA can be locked or unlocked dynamically. Unlocked genes become active functional system components. Locked genes are inactive components.

The existence of the co-DNA of a software system on a computational or communication device can make this device function as part of the system, i.e., to become a *system cell*. This will happen when at least one of the genes of the co-DNA on the device is unlocked. The exact part of system functionality that is assumed by a device depends on the genes of the system co-DNA that are unlocked whilst the co-DNA is within the device.

The software system units that are encoded in the co-DNA correspond to system components at some level in the overall software system architecture. These components may be atomic or composite. Their characterisation as "units" from the perspective of co-DNA modeling reflects the view that, even if they could be decomposed further into more primitive components, the co-DNA model does not encode this possibility and these components will have to be activated and used as composite elements when the gene in the co-DNA, which corresponds to them, is unlocked on a computational device. The description of software system units in co-DNA genes is *multi-faceted* and includes specifications of:

1) the initial architectural model of the system and the role of the particular unit within it
2) alternative patterns of re-assembling the unit with other units in reconstructing the system (depending on constraints arising during the reconstruction process, as shown in parts (a) and (c) of Fig. 1)
3) provided and required interfaces of the unit and the communication protocols through which it may interact with other units (whether they are part of the same co-DNA or other co-DNAs that may be recombined with the gene dynamically (see process P7 below)
4) quality and security properties that the unit requires and can guarantee whilst interacting with other units
5) the information that the unit could reveal about its internal state and the interface through which such information can be obtained by other system units in order to enable them identify a wider "system state" that may be necessary in deciding with which units to connect and how to alter their behaviour if necessary
6) the code implementing the unit, and
7) possible configurations of the code depending on the hosting device where the code should run.

The above facets are necessary in order to support different operations in the system reconstruction process and realize the functionality of the system.

In addition to genes encoding the functional units of the system, the co-DNA incorporates genes with a regulatory role in the system reconstruction process. The latter genes undertake responsibility for functions such as the initial unlocking (and the dynamic locking/unlocking) of other co-DNA genes; transmitting the co-DNA to additional devices; obtaining and analysing information about the operational context of the co-DNA on a device and the state of the system components on the local device where they belong; and transmitting, receiving and acting on signals regarding the overall state of the software system in order to undertake appropriate component adaptation actions on the local device. A conceptual view of the overall co-DNA structure is shown in Fig. 2.

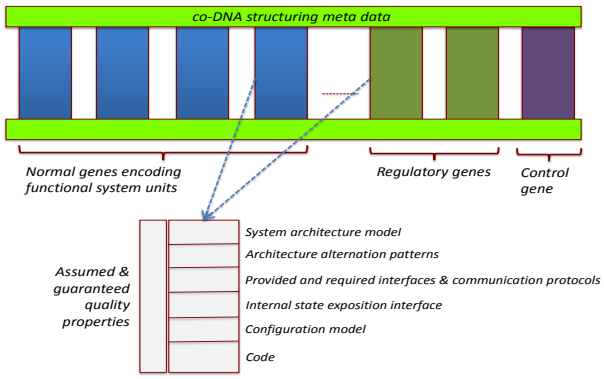

Fig. 2. Conceptual co-DNA structure.

III. BIOLOGICALLY INSPIRED SOFTWARE SYSTEM RECONSTRUCTION PROCESSES

Having described the basic modeling facets of co-DNA, we can now turn our attention to specific biological processes that can be simulated to enable system reconstruction:

Tissue communication and awareness of injury (P1): In biological systems, following injury and loss of tissue, a variety of local events will signal the initiation of tissue response. An increase in the concentration of certain chemical signals (chemotactic agents) released from injured cells will signal the recruitment of certain type of cells to the area of injury that will assist in wound healing. At the same time, a fall in the extracellular concentration of certain chemical signals, continuously produced by the cells themselves, may trigger cell division (mitosis) of the remaining cell population. With tissue regeneration and the resultant increase in the number of cells, the concentration of the chemical signal, will again rise. This would signal the restoration of the original cell population and cell replication will stop.

In software system reconstruction, each system cell is modeled to transmit a tracer signal to neighbouring nodes. The tracer signal is modeled as a random walk, and hence each node will receive an average number of 'visits' over time. The fall of this number below an expected threshold will signify loss of a critical mass of nodes and will trigger a response from the surviving nodes. The appropriate response for this scenario is the recruitment of new unassigned nodes for restoring system functionality. Depending on the architecture, there could also be local and global constraints to satisfy. co-DNA encodes such constraints instructing the

node, which receives them to act in a network building capacity until local constraints are satisfied and then continue monitoring this state. Depending on the schemes for dealing with this issue, there might be simultaneous competing reaction by different nodes or over-reaction. Thus, appropriate schemes for self-regulation are also needed, as part of the co-DNA encoding.

Cell signaling & response (P2): Cell communication is realised through signals that are sent and received by cells or come from the environment. Once a signal reaches its target molecule (usually a protein), it works to change the behaviour of the cell. Each cell receives a complex combination of signals, which can simultaneously trigger many different signaling pathways. Each step in a signaling pathway provides an opportunity for communication between different cells. Through this communication, the cell integrates information from different signaling pathways to initiate an appropriate response.

Similarly, co-DNAs can be transmitted across different computational devices through networks using appropriate protocols. These may be *non-standard network overlay protocols***.** They could also imitate malware techniques in order to be used across different platforms. An overlay network layer protocol should support co-DNA receipt, dispatch, confirmation, encoding, decoding, and storage actions. When a device receives a signal containing a co-DNA transmitted and activates it, the device establishes an application layer communication with other software cells of the same system. At that level an additional communication protocol is required to let the device act in an application compatible manner. The protocol of this communication must be determined by the co-DNA itself.

In response to appropriate signals, a cell may survive, divide (cell replication), differentiate or die in a programmed way (apoptosis) (Fig. 3). Similarly the surviving system elements could 'replicate' themselves, leading to the creation of separate virtual machines running on the same computational device. The benefit of splitting a device into two or more separate virtual machines is that the resulting machines could behave in an independent way, assuming roles of different software system cells. Subsequently, they may also behave in different ways depending on the transformations of their individual co-DNAs after the initial split that led to their creation.

Cellular differentiation is the process by which a less specialized cell becomes a more specialized cell type. Differentiation changes a cell's size, shape, metabolic activity, responsiveness to signals, and ultimately, functionality. These changes are largely due to highly controlled modifications in gene expression (epigenetics). With a few exceptions, cellular differentiation almost never involves a change in the DNA sequence itself. Thus, different cells can have very different physical characteristics despite having the same genome. Closely related to this process is *metaplasia*, a type of cellular adaptation to chronic injury. Metaplasia occurs when a differentiated cell of a certain type is replaced by another cell type, which may be less differentiated. Differentiation in software systems may be modeled by system cells acquiring new functions through dynamic unlocking of co-DNA functions triggered by signals that specific system cells get from their environment (e.g., low battery, memory resource exhaustion, changes to device configurations).

In contrast to cell death following an external noxious effect, *apoptosis* is a programmed cell death process, designed for the normal elimination of unwanted cell populations. Regulation of apoptosis is mediated by a number of genes in DNA and their products. In system recovery, the recruited software system cells may be also modeled for programmed shutdown. This may be necessary when, in the absence of any centralised form of control in the reconstruction process, the system reaches a state of too many elements of a given type. This function can be triggered by "apoptosis" functions encoded in the co -DNA of a system cell or newly sent external co-DNAs.

Tissue remodeling (P3): Healing is a complex and dynamic process of restoring cellular structures and tissue layers. As healing develops over time, the cellular and non-cellular elements of the healed tissue change configuration to resemble the original tissue before injury. Similarly, as original system cells recover, the software system needs to have the ability to reconfigure itself according to its original architecture.

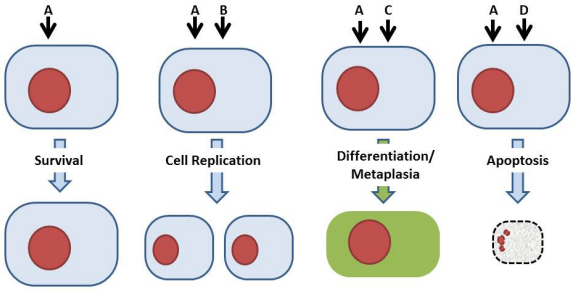

Fig. 3. Cell responses to different signals (represented by A-D)

DNA Replication (P4): DNA has the ability to replicate itself through complex gene regulated mechanisms, involving special proteins. The co-DNA should have the ability to replicate itself faultlessly through a novel protocol leading to the creation of exact copies that can be transmitted to different computational devices. co-DNAs should also encode mechanisms for identifying random faults during their replication and correct them.

DNA Transformation and recombination (P5): Transformation is the naturally occurring process of gene transfer, which involves absorption of the genetic material by a cell through the fusion of a foreign DNA with the native DNA resulting in the genetic expression of the received DNA. If DNA material from the cell of an organism is incorporated into the DNA of a host cell of a different organism, then a recombinant or chimeric DNA is constructed. In another form of recombination, known as transpositional recombination, mobile elements

(transposons) are inserted into a target DNA and can play a critical role in the spread of several factors. Recombinant DNA (rDNA) molecules can bring together genetic material from multiple sources, creating sequences that would not otherwise be found in biological organisms. rDNA is possible because DNA molecules from all organisms share the same chemical structure.

The ability to incorporate the co-DNA of an external software system, through transfection, to a host device with a different original function is also important for system reconstruction (Fig. 4). The transfection of co-DNA may take different forms, e.g., forced transfection (when near extinction is reached in critical systems such as emergency response systems), near shut down transfection and denial of service. Transformation may need to be regulated to address security and operational constraints (e.g., to avoid over-utilisation of the target cell or the transfection of malware).

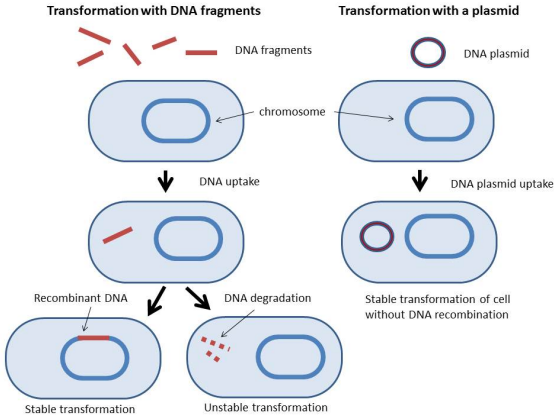

Fig. 4. Mechanisms of transformation. Exogenous DNA can be integrated to the host DNA molecule (recombinant DNA) or in special DNA types (plasmids) it can transform the cell (i.e. expression of plasmid genes) without integration of the DNA material into the host cell's chromosome.

Gene expression regulation (P6): The regulation of gene expression includes a wide range of mechanisms that are used by cells to increase or decrease the production of specific gene products (proteins). This increases the versatility and adaptability of an organism by allowing the cell to express protein when needed. Gene expression regulation is a multi-level process that generally requires suppressors and trigger factors. In our approach, *near-extinction events* can be modeled to trigger or suppress code execution. Moreover, upon reaching a certain level of complexity, a different process permitting the individual to integrate the vast quantity of interactions with the outside world may also emerge. This process (aka *epigenesist*) is characterized by the possession of a basic structure that is entirely defined by the genome (the innate part) but can also be subjected to modification through lifetime interactions of the individual with the environment (the acquired part).

As shown in Fig. *5*, the DNA consists of: (i) dominant genes alleles offering the different ways of developing each cell and generating a phenotype, (ii) functional genes alleles offering the different ways of each organ function, and (iii) control genes alleles that constitute all the control processes and the control processes themselves. All genes will be silent (not active) except defining and controlling genes. This structure inspires our envisaged conceptual structure of co-DNA discussed in Sect. II.

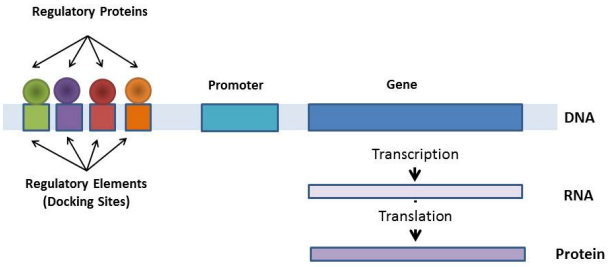

Fig. 5. Gene regulation

IV. Related Work

The development of biologically inspired solutions to computational problems has become a significant trend within the last couple of decades. *Swarm Intelligence* [14] and *Social Insect* principles [15] have been used to address problems of distributed search, optimization and routing in wireless sensor networks [16]. *Firefly Synchronization* [17] has been used to address robust and distributed clock synchronization [18]. *Artificial immune* [19] and *activator-inhibitor systems* have inspired solutions to distributed coordination, network autonomicity and adaptability, and system misbehaviour/anomaly detection [20]. Also problems of content distribution, overlay network formulation, and coordination in massively distributed systems have been solved based on the bio-inspired principles of *epidemic spreading* [21] and *cellular signaling* (networks) [22]. In addition, DNA has been used to build basic computational units (e.g., logic gates), and DNA like structures have been used as a model of parallel computation in transactional systems [23]. Inspired by natural ecosystems, SAPERE has developed a framework for decentralized deployment and execution of self-aware and adaptive services for future pervasive network scenarios [24]. Related research includes also *genetic algorithms* [25], and *genetic programming* (GP) [26]. Genetic algorithms focus on evolving a population of candidate solutions to an optimization problem, towards a better solution. GP is used in genomics focusing on typical genetic analysis and gene network inference.

Inspired by the autonomy of the human nervous system, *autonomic computing* (AC [27,28]) is also concerned with the development of self-managing capabilities (e.g., self-configuration, self-optimization, self-protection, self-healing and self-protection) for software systems. AC typically advocates a reference control model of monitoring, analysis, planning, execution and knowledge management capabilities (MAPE-K [29]) that can be introduced to a normal system to give it autonomic capabilities. MAPE-K has been realized in some frameworks (e.g., ABLE [30], KX [31], AC Toolkit [32]). AC research has also generated autonomic system specification and adaptation policy languages (e.g., ASSL [33,34]), and alternative implementations for MAPE-K

capabilities (e.g., monitoring and context awareness [35], planning [36], knowledge [37] and process adaptation [38]).

The commonality of our co-DNA based system reconstruction approach with the above strands of research is that it also aims to draw upon biological mechanisms with proven properties in order to develop a novel solution to a challenging computational problem that requires forms of autonomic behaviour. However, our focus and the challenges that our approach aims to address are entirely different from the above work, as we are targeting to address the re-construction of large-scale software systems starting from a state of excessive damage, operating with no capabilities of central control within a continually changing and possibly increasingly adverse environment (e.g., on-going disaster/emergency).

V. Conclusion

In this paper, we have introduced a bio-inspired approach for reconstructing nearly extinct complex software systems. This approach is based on encoding the co-DNA of a system and computational analogues of biological processes enabling its transmission over computational devices and, through it, the transformation of the latter into system cells that can realise chunks of the system functionality, and spread further its reconstruction process.

Having outlined the fundamental structure of co-DNA and the key biological processes that are plausible to utilize in the system reconstruction process, we are currently developing the computational framework that will realize our approach. This development is informed by two case studies. The first case study is a complex *Crisis Management Software Ecosystem (CMSE)* that reaches a near extinction state following a massive scale failure caused by a natural disaster. The second case study is a telecommunication network, involving different layers, including radio access, the core network and/or the backhaul/backbone network.

VI. References

[1] F. Salfner, M. Lenk, & M. Malek, "A survey of online failure prediction methods", *ACM Comp. Surveys*, 42(3), 2010.

[2] B. Littlewood, L. Strigini, "Redundancy and diversity in security". *Computer Security–ESORICS* 2004. Springer, 2004. 423-438

[3] H. Huebscher, J. McCann, 'A survey of autonomic computing – degrees, models and applications", *ACM Computing Surveys*, 40(3), 2008.

[4] H. Psaier, H. Dustdar, "A survey on self-healing systems: approaches and systems", *Computing* 91(1), 2011.

[5] A. R. Joyce, B. Palsson, "The model organism as a system: integrating 'omics' data sets", *Nature Publishing Group*, March 2006 | Volume 7.

[6] Y. Derbal, "on modeling of living organisms using hierarchical coarse-graining abstractions of knowledge", *J. Biol. Syst.*, 21, 1350008 (2013).

[7] B. Alberts, D. Bray, K. Hopkin, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, Essential Cell Biology, March 27, 2009

[8] S.V.Perryman, K.G. Sylvester, "Repair and regeneration: opportunities for carcinogenesis from tissue stem cells", *J Cell Mol Med,* 2006 Apr-Jun;10(2):292-308.

[9] K.P. Krafts, "Tissue repair: The hidden drama", *Organogenesis*. 2010 Oct-Dec; 6(4):225-33.

[10] T.S. Stappenbeck, H. Miyoshi, "The role of stromal stem cells in tissue regeneration and wound repair". *Science*. 2009 Jun 26;324(5935):1666-9.

[11] E. Tanaka, B. Galliot, "Triggering the regeneration and tissue repair programs". *Development*, 136(3):349-53, 2009

[12] A.C. Heinrich, S.A. Patel, B.Y. Reddy, R. Milton, P. Rameshwar, "Multi- and inter-disciplinary science in personalized delivery of stem cells for tissue repair", *Curr Stem Cell Res Ther*. 2009 Jan;4(1):16-22.

[13] B. Galliot, E. Tanaka, A. Simon, "Regeneration and tissue repair: themes and variations", *Cell Mol Life Sci*. 2008 Jan;65(1):3-7.

[14] M. Farooq, G.A. Di Caro, "Routing protocols for next-generation networks inspired by collective behaviors of insect societies: An overview", *Swarm Intelligence, Natural Computing, Springer*, 2008, pp. 101–160.

[15] G. Theraulaz, E. Bonbeau, "A brief history of stigmergy", *Artificial Life* 5 (2) (1999) 97–116.

[16] H.F. Wedde, M. Farooq, Y. Zhang, "Beehive: an efficient fault-tolerant routing algorithm inspired by honey bee behavior", *Ant Colony, Optimization, and Swarm Intelligence, LNCS*, vol. 3172, Springer, 2004, pp. 83–94.

[17] G. Werner-Allen, G. Tewari, A. Patel, M. Welsh, R. Nagpal, "Firefly inspired sensor network synchronicity with realistic radio effects", in *Proc. SenSys*, 2005, pp. 142–153.

[18] D. Lucarelli, I.-J. Wang, "Decentralized synchronization protocols with nearest neighbor communication", in *Proc. 2nd International Conference on embedded Networked Sensor Systems*, ACM, 2004, pp. 62–68. doi:10.1145/1031495.1031503.

[19] S. Sarafijanovic, J.-Y. Le Boudec, "Artificial immune system for collaborative spam filtering", in *Proc. NISCO* 2007, 2008, pp. 39–51.

[20] S. Sarafijanovic´, J.-Y. Le Boudec, "An artificial immune system for misbehavior detection in mobile ad-hoc networks with virtual thymus, clustering, danger signal, and memory detectors", *Artif. Immune Syst*. (2004) 342–356.

[21] J.W. Mickens, B.D. Noble, "Modeling epidemic spreading in mobile environments", in *Proc. 4th ACM Workshop on Wireless Security*, 2005, pp. 77–86.

[22] F. Dressler, I. Dietrich, R. German, B. Krüger, "Efficient operation in sensor and actor networks inspired by cellular signaling cascades", in *Proc. Autonomics*, 2007, pp. 1–10.

[23] M. Meisel, V. Pappas, L. Zhang, "A taxonomy of biologically inspired research in computer networking", *Computer Networks* 54(6): 901-916 (2010).

[24] Project Website: http://www.sapere-project.eu/

[25] W. Banzhaf, P. Nordin, R.E. Keller, F.D. Francone, "Genetic Programming - An Introduction", Morgan Kaufmann and Heidelberg:dpunkt, San Francisco, CA, 1998.

[26] M.W.Khan, M. Alam, "A survey of application: Genomics and genetic programming, a new frontier", *Genomics* 100 (2012) 65–71.

[27] H. Huebscher and J. McCann, "A survey of autonomic computing – degrees, models and applications", *ACM Computing Surveys*, 40(3), 2008.

[28] H. Psaier, and H. Dustdar. "A survey on self-healing systems: approaches and systems", *Computing* 91(1), 2011.

[29] Computing, Autonomic. "An architectural blueprint for autonomic computing," in IBM White Paper 2006

[30] J. P. Bigus, D.A. Schlosnagle, J.R. Pilgrim, W. N. Mills, Y. Diao, "ABLE: A toolkit for building multiagent autonomic systems". *IBM Systems Journal*, 41(3), 350-371, 2002.

[31] G. Kaiser, J. Parekh, P. Gross, G. Valetto, "Kinesthetics extreme: An external infrastructure for monitoring distributed legacy systems", in *Proc. Autonomic Computing Workshop*, 22-30, 2003.

[32] B. Jacob, R. Lanyon-Hogg, D.K. Nadgir, A.F. Yassin, "A practical guide to the IBM autonomic computing toolkit", IBM Redbooks, 2004.

[33] E. Vassev, M. Hinchey, "ASSL: A Software Engineering Approach to Autonomic Computing", *IEEE Computer*, 42(6):106-109, June 2009.

[34] Broto, Laurent, et al. "Autonomic management policy specification in Tune", in *Proc. 2008 ACM Symp. on Applied computing*. 2008.

[35] S. Agarwala, et al. "QMON: QoS-and utility-aware monitoring in enterprise systems", IEEE International Conference on Autonomic Computing, 2006. ICAC'06.

[36] A. Ranganathan, R. Campbell, "Autonomic pervasive computing based on planning", in Proc. *International Conference on Autonomic Computing*, IEEE, 2004.

[37] G. Tesauro, "Reinforcement learning in autonomic computing: A manifesto and case studies", *Internet Computing*, IEEE 11.1 (2007): 22-30.

[38] Lee, Kevin, et al. "Workflow adaptation as an autonomic computing problem", in *Proc. 2nd workshop on Workflows in support of large-scale science*. ACM, 2007.