

Real-Time Systems – Biologically Inspired Future

Nevena Ackovska
 Institute of Informatics, FNSM
 Sts. Cyril and Methodius University, Macedonia
 Email: nevena@ii.edu.mk

Stevo Bozinovski,
 Department of Mathematics and Computer Science,
 SC State University, USA
 Email: sbozinovski@scsu.edu

Gjorgji Jovancevski
 Faculty of Computer Science and Information Technology
 University “American College Skopje”, Macedonia
 Email: jovancevski@uacs.edu.mk

Abstract - In this paper, we are presenting a bionic approach towards real-time systems research. Observing that a cell is both real-time computing and real-time manufacturing system, the work presented here explores the analogy between the real-time computer systems (and their management) and the molecular biology control systems. This paper is a further elaboration of the idea that the cell is governed by a real-time operating system that resides in the DNA. It also gives insight into how knowledge obtained from the biological systems could lead to further evolution and improvement of the complex human-made real-time systems.

Index Terms – real-time system, operating system, DNA, program compilation, concurrent processing, real-time production

I. INTRODUCTION

Today’s computer systems generally deal with fast (real-time) responses. Their correctness depends not only on the logical result of the computation, but also on the time at which the results are produced [1, 2]. To achieve that, some computer systems have multiple registry sets (since IBM Series/1 real-time computer system), multiple processors, fast memory management systems and so on.

The real-time systems and their features have been implemented by the nature since the origins of life. The scientific community today shows great interest in observing real-time management in biological systems. Much of the attention so far has been given to the neural networks, their parallel processing and real-time capabilities [3]. Although early works on real time properties of the cell biosynthesis process have been reported in 1986 [4, 5], it wasn’t until year 2000 that the first attention was brought to the genetic system as another important biological system that should be considered as a

real-time system [6].

The paper we are presenting here elaborates further in that direction, considering the genetic system and emphasizing its real-time features, in terms of parallel processing, memory management, process management, consistency and so on. It is important to mention that in addition of being a real-time information processing system, it is also a real-time material processing system [7]. This double feature of the real-time system is particularly interesting for this study especially since today’s real-time computing systems are often embedded in complex manufacturing systems. In the following sections, we will focus the attention of the real-time processing community to the real-time information and material processing system of a biological cell.

After this introduction, in the sequel this paper first presents the point that instead in terms of genes it might be better to consider the genetic information processing in terms of files. Consequent real-time information processing concepts such as system disks, file management, process management including inter-process communication, and program compilation in a cell are addressed in Section II. In Section III we deal with the real-time manufacturing in a cell. Pointing out that the cell implements just-in-time (JIT) material processing, we present a case study how the sugar lactose is processed in a cell and how the cell manufactures from relatively small number of building blocks. Chapter IV gives a global picture elaborating the concept of Cell Operating System that orchestrates the processes and resources in a cell. Chapter V discusses some issues related to future of real-time systems considering analogy with the biological systems.

II. REAL-TIME INFORMATION PROCESSING IN A BIOLOGICAL CELL

Information storage and processing in molecular biology is based on the concept of a gene. A very natural question is “What is a gene?” A usual answer is that a gene is a segment of a DNA that encodes for either a protein or a RNA. Also, one could encounter slightly different definitions [8, 9]. In contrast, in studying information storage processing in computer science and engineering a usual reasoning considers the files of that system. Our research in biological information processing starts with a question “what are the files of the genetic system?” Is the concept of a gene corresponding to the concept of a file? Having that as a starting point, in this section we will present our understanding of DNA organization and DNA computing in terms of files and related concepts.

A. Cell files

Looking for a concept of a file in DNA, we found that the transcription units (or scriptons [10]) are analogous to cell files. A transcription unit is a segment of DNA that eventually becomes transcribed to RNA. In prokaryotes (cell without a nucleus), a transcription unit often produces a transcript with several genes (so-called polycistronic RNA). In eukaryotes (cells with a nucleus containing DNA) it produces a precursor RNA, which contains the information about a single gene, but in order to obtain it, additional processing needs to be performed.

The eukaryotic files are rather complex and contain segments of a gene, interleaved with segments that do not belong to the gene. Those segments are known as introns (interleaving segments), as opposite to exons (gene expressing segments). To the people involved with genetics, there is a standard question considering this phenomenon: how did it happen that eukaryotic genes became segmented? However, for computer engineers introduced to the concept of a file, the answer is straightforward – busy files are fragmented. Defragmentation (also called “garbage collection”) is sometimes needed in computer file systems. Moreover, it is expected that between two fragments of a file an entire different file could be expected. This leads to the concept of distributed file systems [11, 12]. And indeed this is the case in molecular genetics. After the first evidence that Tetrahymena ribozyme is actually an intron [13], more evidence has been found that genetic files could be found within a complete different file [e.g. 14]. Therefore, our file centered approach offers simple answers to nontrivial problems in genetics [7].

Now, let us consider the exons, the expressing gene segments. Using our approach, it is easy to see that exons could be functional units, such as subroutines (or methods in OOP) of a more complex program file. The subroutines

could be reusable, meaning that the same exon could be used in different RNA’s, example being the building of mRNAs for antibody proteins.

Additional correspondence to the concept of file is that one cannot copy less than a file with its standard operating system operations. Analogously, the cell also cannot transcribe less than a transcription unit by means of standard and normal functioning.

So we believe that the concept of a file is more suitable in describing the DNA transcription process than the concept of a gene. The cell, especially the eukaryotic cell, undergoes extensive file processing: from copying the pre-RNA file until obtaining the RNA message. This process includes operations like: cut (introns), join (exons), right append (trailer string), left append (header string), letter replacement and so on, which are standard file processing operations in a computer operating system [15].

B. Cell disks

Following this line of reasoning, another very important question would be “what is a disk in a genetic information system?” The initial observation could lead to the conclusion that the DNA offers a tape based [16] information processing system. However, a careful examination reveals that DNA tapes are randomly accessible – they are not sequential tapes. So, due to the nature of their accessibility, we may consider them as disks. Indeed the human genome might be considered as a system of 23 pairs of system disks (chromosomes). It is a clearly a distributed file system.

Also the classical concept of disk cylinder can easily be applied to DNA processing (Figure 1).

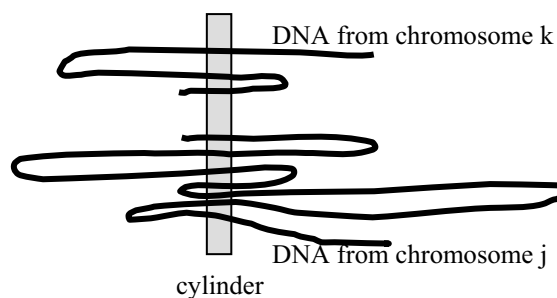


Figure 1. Disk cylinder concept on chromosomes.

The cylinder means that, although it is on different disk plates, the information can be read in the same time by several disk reading heads. The concept of a cylinder enables spatially distant information to be processed in parallel. And indeed, the files on different chromosomes could be processed in parallel (shaded area in Figure 1). For example, in order to produce hemoglobin, protein genes from chromosome 11 and chromosome 16 should be processed in parallel.

C. File and memory management

Classical process management in computer systems considers a resource and several processes competing over that resource. Several mechanisms were developed to address this issue [e.g. 17]. Basically, a mutual exclusion mechanism is used, by which one process is using a resource while other processes are blocked, waiting. It seems that instead of process management, genetic systems rather utilize memory and file management. They simply keep many copies of files that are frequently used. Examples are rRNA files needed for manufacturing ribosomes. Those files are kept in thousands of copies [18]. In order to function in real time, a genetic system needs to produce many copies of the rRNA in the same time. This means that no waiting for this vital resource is allowed. So, in genetic systems, for achieving real-time response, many copies of rRNA files are utilized.

Keeping files in many copies provides also for a mechanism of file consistency. It seems that the voting mechanism is used in a cell to keep some files unchanged in the evolution process. For example the rRNA files are particularly stable. Also the fact that the DNA is a two strand molecule, in which one strand is a complement to the other, enables repair of an error and consistency in obtaining copied information from a DNA strand [18].

D. Processes and Inter-process Communication

The cell system is a highly distributed one and many processes are running in parallel and some of them are running in a pipeline (example being the polysome, sequence of several ribosomes over a single mRNA in prokaryotes). In such a system it is possible to assume existence of interprocess communication among parallel processes, or threads. In order to study possible mechanisms of such an inter-process communication, we developed a software model of a minimal protein biosynthesis system.

Figure 2 shows a model of multithreading in the protein synthesis process. Three agents, RNA polymerase, aminoacyl-tRNA synthetase, and ribosome are engaged in an inter-process communication (IPC) during the protein biosynthesis. The RNA polymerase waits for a demand of a protein, initiates reading of a gene for that protein, then, while reading a DNA base, appends an RNA base to the evolving RNA; then it releases the mRNA. The aminoacyl-tRNA synthetase attracts an appropriate amino acid, attracts an appropriate tRNA, loads the amino acid to the tRNA, and then releases the loaded tRNA. The ribosome searches for an mRNA, reads the starting codon and then attracts a tRNA anticodon that will bring an appropriate amino-acid. It assembles the protein and releases it.

Our model suggests that the tRNA molecules are used as messengers, a material carrying messages, actually mobile robots. They have a signal site, anticodon, and a material carrying site, the aminoacid binding site. As the

programming model suggests, the material part is manipulated using robot programming commands like load, unload, and assemble.

The basic multithreading model in Figure 2 can be modified in several ways. Instead of using the primitives release and attract, one can use the standard primitives as send and receive, or some classical primitives such as post and wait [19], used in the IBM Series/1 real-time operating system [20]. This simulation implies ways of understanding the inter-process communication mechanisms in biological cells and implementing them in multithreading mechanisms of future computer software systems.

thread RNA polymerase

repeat

wait demand (protein)
initiate reading gene(protein)

repeat

read DNA_base from gene(protein)
append mRNA_base(DNA_base) to mRNA
until stopcondition(gene(protein))

release mRNA

until degradation

thread a_i-tRNA synthetase

repeat

attract a_i
attract tRNAⁱ
load a_i to tRNAⁱ
release a_itRNAⁱ(anticodon)

until degradation

thread Ribosome

begin

attach_to mRNA

repeat

read codon from mRNA
attract a_itRNAⁱ (anticodon(codon))
unload a_i from a_itRNAⁱ
assemble a_i to protein

until stopcondition(mRNA)

release protein

recycle

Figure 2. A model of a multithread process and inter-thread communication in a cell

The software model above is not intended to be a detailed simulation of the protein biosynthesis process. Details involving proteins as initiation, elongation and termination factors are not included in the model. The model studies only possible inter-process communication (IPC) between the main agents, represented as threads.

E. Compilation process – From Source Code to a Robot

Contrary to computer systems, where a compilation process takes a file and produces an executable module which then could end up into an EPROM which in turn could end up into a robot, the cell compilation process starts with a file and ends up with a working robot. The cell robots are some RNAs and some enzyme proteins. They are produced in real-time, by demand. So, once demand is sensed by DNA, it makes an appropriate source file accessible, out of which the requested program is read (linear form of RNA). This program is then compiled and assembled into a robot by obtaining its 3D functional structure. An example of such a robot is the tRNA, which is a shuttle robot that carries amino acids into a ribosome for a protein assembly. Another example is the rRNA, which, being part of a ribosome, performs a protein assembly function. In addition to RNA robots, the cell produces protein robots, enzymes, which are the major working force in a cell. This includes carrier mobile robots like kinesin, which travels along the cell cytoskeleton, or a two-armed robot, lac-repressor in E. Coli, which grabs the DNA and disables its access.

III. REAL-TIME MATERIAL PROCESSING IN A BIOLOGICAL CELL

The cells are just-in-time (JIT) production systems, rather than just-in-case (JIC) systems. That makes them the ultimate real-time systems: from processing information in real time, through producing tools and robots in real time, to producing the final product. All the products, including needed machinery and robots, are produced on demand.

A. Case Study: Real-Time Processing of Lactose

Let us consider a simple example of milk consumption. Once the milk enters our intestines, resident E. Coli bacteria receive signals of the presence of the sugar lactose. That is the material that should be processed into glucose and galactose. Figure 3 shows the processing.

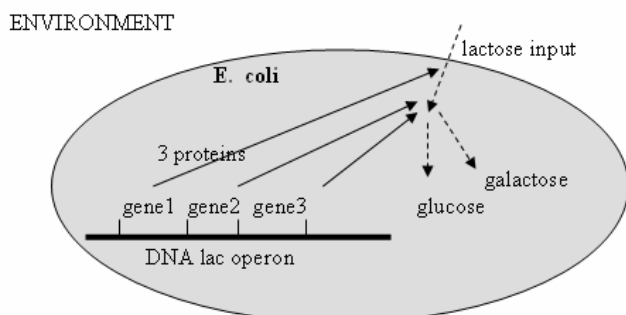


Figure 3. Real time processing of an external event: lactose enters a cell

Once a lactose molecule is sensed by the E. Coli signal

proteins, they start processing a file containing 3 programs (genes). The enzyme RNA polymerase will transcribe three mRNA source codes. They will be compiled into three proteins: β -galactosidase, lactose permease, and β -galactoside transacetylase. They will work as three robots and will disintegrate lactose into galactose and glucose. The process is highly regulated and real-time. The lactose related robots exist only when the lactose is present, when they reach level of up to 5000 molecules in only few minutes. When lactose is not present any longer, the lactose regulating robots are not assembled. Furthermore, all those lactose processing robots will be disintegrated, and their components will be recycled into other robots or other needed cell components.

This example illustrates that the cell not only answers to real-time demand by producing the needed output (glucose in this example), but it also produces in real time all of its needed machinery for such an output. The cell doesn't store its needed tools and machines for a long time. The high responsiveness of the cell is due to the speed of assembly, not through inventories [21].

B. Common building blocks

Additional adaptation towards real-time processing of a cell is that it keeps just a small number of building blocks. For the synthesis of all the major cell molecules, such as DNA, RNAs, proteins, polysaccharides and lipids, needed for the normal functioning of the cell, only 20 amino acids, 5 different types of nucleotides and a small number of saccharides and fatty acids are needed. These major molecules (or parts of them) are constantly recycled and produced solely on demand [21]. It should be noted, however, that some biological systems, like humans, are not able to produce all the amino acids they need, and therefore they must be imported into the system from the surrounding environment.

IV. ORCHESTRATING THE SYSTEM - REAL TIME CONTROL

The cell is an active autonomous agent. Example of a cell is the bacterium Escherichia coli; another example is a specialized cell in a multi-cellular system such as the human skin. In both cases the cell receives signals from the environment, including signals from other cellular agents, and responds to them. The cell has special sensors for various signals, and some of the cells have motors which allow them to move in the environment. As an example, E. coli has about 50-70 sensors for various signals from the environment, and about 6-8 motors that are actuators for its flagella. Using its sensors, a cell would move towards an attracting spot in the environment, following a chemical gradient [9, 22]. The cell actively responds to environment changes. The response can be a behavioral one or a product manufacturing one [6, 23]. Another type of response is either reproduction or termination of its life, as a specific reaction to an

environment change [24].

To get a feeling of the potential information and control complexity of the cell activities, we take into consideration its three main components: resources, processors, and processes. Let us look at the potential complexity of the best-known model organism, *Escherichia coli*. It has a genome consisting of 4800 genes, a proteome of about 2500 proteins, and a metabolome (set of all metabolic reactions) of an unknown number [24]. Not all proteins are processors, but we can estimate that it is a parallel distributed processing system of up to thousand processors executing their programs written in the genome, working over about 5000 resource segments (genes and control segments), and carrying out an unknown number of processes.

Taking in consideration such a complex system that deals with the real-time features of its components, one should naturally presume that the system itself should be governed by a control sub-system.

There are several levels of information and material processing in a cell. As Figure 4 emphasizes, the cell communicates with two environments: one is the behavioral environment, which it faces during its life, and the other is the genetic environment, through which it communicates its genetic material.

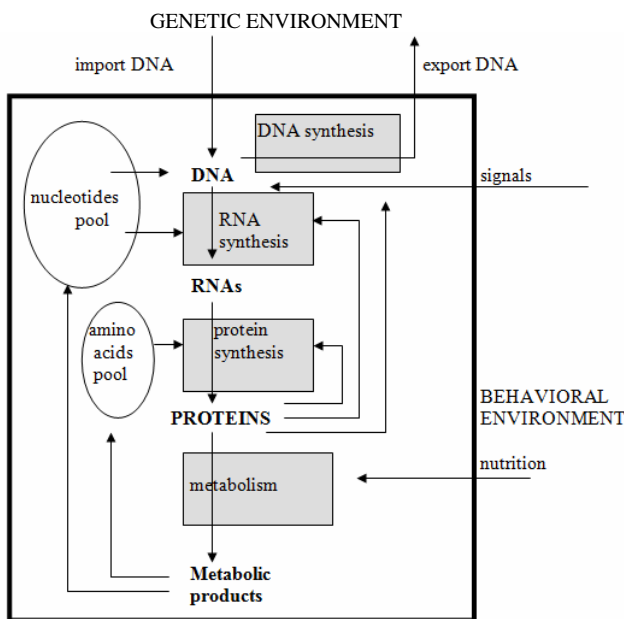


Figure 4. Several levels of processing in a biological cell

Fig. 4 also emphasizes several levels of processing, relevant for the cell control hierarchy, such as DNA level, RNA level, protein level, and metabolism level.

The cell has a hierarchical control structure. At the lowest level of control in prokaryotes are the operons [25], structures that control simultaneous activation of a group of genes. Regulons are at a higher level, controlling

simultaneous activation of various genes and operons. Modulons are at a level higher, controlling regulons, operons and genes. It is not known whether this is the highest level of control in prokaryotes. In eukaryotes the control structure is even more complex.

All these structures are somehow coordinated. For example, when a cell is undergoing replication, all the other processes are oriented towards the support of this extremely complex process. Taking this into account, our observation is that there should be a general control structure that orchestrates the priorities of the cell activities. This observation leads us to consider the hypothesis of the existence of a Cell Operating System (COS). Moreover, the COS should be considered a database operating system, which takes care of a large database, the DNA itself [15]. This concept is shown in Figure 5.

Figure 5 points out that the DNA is a database operating system, which has a rather large gene database to take care of. Database operating systems have been of interest in systems software for some time [26, 27]. It also points out that the lower level of control systems consists of various feed-forward and feedback regulatory loops controlled by operons or some other control structures. The genome in Figure 5 denotes a transcription-translation machinery that, given a gene, produces a protein or RNA. Thus, the Cell Operating System is a real-time database operating system, comprising the hierarchy of control levels of the cell control system and taking care of the gene database.

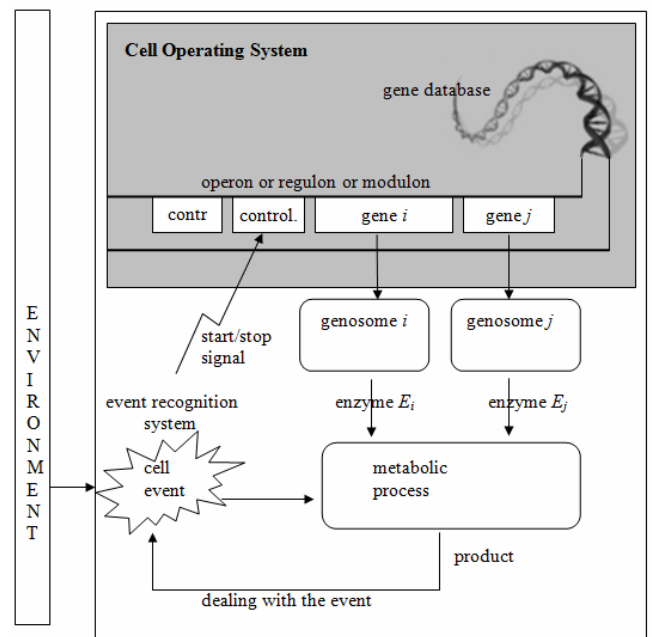


Figure 5. The cell operating system concept

V. BIOLOGICALLY INSPIRED FUTURE OF REAL – TIME SYSTEMS

Real-time computation is about real-time application; real-time computers are actually tools for real-time processes in general. Real-time application in general is due to the need of real-time product. This means that the future of real-time systems needs to integrate real-time material and information processing. And that is what the nature has done long ago. The following table shows that those two types of processing are actually conceptually equivalent.

Table 1. Equivalence between information and material processing

	information	material
processing	processor	processor
transport	transmission	transportation
storage	storage	storage

Today scientists make an effort in bringing information and material processing closer. Nanotechnology science makes it possible to produce nanostructures that do some kind of processing while having the meanings for production of other nanorobots that do specialized work in cells (and possibly self replication) [28]. These robots have their software and their hardware interleaved in the same building-coding blocks. There is also a concern of using the combination of DNA information-encoding and recognition properties, and the enzymatic machinery capability for DNA manipulation in the field of DNA computation [29].

Today, the first controlled steps for nanorobots are already made. There have been reports that a walking nanorobot has been constructed [30].

One open problem today that represents a grand challenge in the nanotechnology is how to construct carrier nanorobots, for example robots that could transfer an atom, or a whole another molecule, from one place in the medium, to another place [30]. However, in 2005 there were some reports on nanorobots that could move in desired direction, and were symbolically named nanocars [31, 32]. The future of these nanovehicles is to be able to carry controlled load so they could be used for precise drug delivery. Although DNA is still the most prominent example of a molecule of choice for building nanorobots, there have been also reports for building nanomachines based on the RNA properties [33]. These machines have the ability to attain reasonable control of the arrangement of matter at a molecular level.

Having all previously mentioned in mind, and the possibilities that nanotechnology science offers we could presume that perhaps, just like the evolution of natural life, the evolution of artificial life will also continue from the nano level.

Of course, it is important that along the way we would be interested in faster algorithms, let's say for floating point division, but ultimately the effect of that algorithm will be used in some real-time application, for example robot jumping, or for just-in-time production. Therefore probably it is sensible to try to speed up the whole process, just as the nature has done – create the hardware and the software of the machines from the same “material”.

VI. CONCLUSIONS

This paper presents a bionic approach towards the real-time systems. A bionic approach suggests that it is a good practice to learn from nature in order to build human-made systems. That approach proved effective in some areas of systems building, a prominent example being neural networks, today widely used as controllers and pattern recognition devices in various control systems. Here we propose that we could learn how to manage up the real-time processes in the human-made systems, taking in consideration the management of the processes in molecular biology.

It seems interesting that the real-time assembly of products, as well as the machines, is essential for the cell survivor. In human-made systems this is not implemented, although there is some effort in this direction [21].

The biological cell has very synchronized control system. In order to orchestrate such complex processes we presume the existence of some sort of operating system that resides in the DNA, and expands on several levels of hierarchy. This operating system is not strictly software system; it takes care of all the processes that take place in the cell. It governs both information and material flow in the cell.

The idea is that perhaps it's not essential to make information processing faster and extremely real-time. What is important is how to make the result of the whole system that requires real-time result faster and more reliable. The answer that nature has already invented is simple - make everything as real-time as possible: from information processing, tools and robot processing, to material processing. This paper also establishes a possible connection between the real-time computation and the nanotechnology community.

It is our belief that establishing a bionic relation between the molecular biology research and real-time systems research will prove useful for both, generally considered distinct, areas of research. We hope that this new frontier for real-time systems offers an exiting and fruitful field for research towards a new generation of biologically inspired real-time systems.

REFERENCES

- [1] J. Stankovic, K. Ramamritham, ed. *Hard Real-Time Systems*. IEEE Computer Society Press, 1988.
- [2] J. Stankovic, "Recent Advances in Operating Systems", *Lecture Notes*, Computer Science Department, University of Massachusetts, Amherst, 1995
- [3] D. Rumelhart, J. McClelland, PDP Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, 1986
- [4] S. Bozinovski "Flexible manufacturing systems: A biocybernetic approach" In E. Popov and M. Vukobratovic (eds.) *Proc Symp Robotics and Flexible manufacturing*, Moskow, Russia, 1986. p. 192-197
- [5] S.Bozinovski, L. Bozinovska "Flexible Production Lines in Genetics: A Model of Protein Biosynthesis Process", *Pro. International Conference on Robotics, Dubrovnik*, Yugoslavia, 1987, pp. 1-4
- [6] S. Bozinovski, B. Mueller, F. diPrimio, "Biomimetic autonomous factories: Autonomous manufacturing systems and systems software" *GMD Report 115*, German National Research Center for Information Technology, Bonn, 2000
- [7] N. Ackovska, S. Bozinovski, G. Jovancevski, "A New Frontier for Real – Time systems – Lessons from Molecular Biology", *Proc. IEEE SoutheastCon 2007*, pp. 224-228
- [8] T. A. Brown, *Genetics, A molecular Approach*, Chapman & Hall, 1992
- [9] S. Bolsover, J. Hyams, S. Jones, E. Shephard, H. White, *From Genes to Cells*. Willey-Liss, 1997
- [10] V. Ratner, *Control Systems in Molecular Genetics*, (In Russian) Nauka, Novosibirsk, 1975.
- [11] G. Nutt, *Centralized and Distributed Operating Systems*. Prentice Hall, 1992
- [12] A.Tanenbaum, *Distributed Operating Systems*. Prentice Hall, 1995.
- [13] K. Kruger, P. J. Grabowski, A. J. Zaug, J. Sands, D. E. Gottschling, T. R. Cech, "Self-splicing RNA: Autoexcision and autocyclization of the ribosomal RNA intervening sequence of tetrahymena", *Cell*, Vol 31, 1982, pp.147-157
- [14] M. D. Been, "Versatility of Self-Cleaving Ribozymes", *Science*, Vol. 313, 2006, pp. 1745-1747
- [15] S. Bozinovski, G. Jovancevski, N. Bozinovska "DNA as a real time, database operating system". *Proc SCI 2001*, Orlando, 2001, pp. 65-70
- [16] S. Bozinovski, G. Jovancevski, N. Ackovska, "Closed Loop DNA Operating System Migration", *Proc. SCI 2003*, Vol 8, 2003, pp. 3-7
- [17] G. Andrews, F. Schneider, "Concepts and notations for concurrent programming" *ACM Computing Surveys*, Vol 15, 1983, pp. 3-43
- [18] T.A Brown, *Genomes*, 2-nd Ed., Willey-Liss, 2002
- [19] B. Witt, "Communication modules: A software design model for concurrent distributed systems" *IEEE Computer Magazine*, 1985
- [20] S. Bozinovski, "Parallel programming for mobile robot control: Agent based approach" *Proc. IEEE Conf on Distributed Computing Systems*, Poznan, 1994, pp. 222-228
- [21] L. Demeester, K. Eichler, C. H. Loch "Organic Production Systems: What the Biological Cell Can Teach Us About Manufacturing", *Manufacturing and Service Operation Management*, Vol. 6, No. 2, INFORMS, 2004, pp. 115-132
- [22] Berg J., Tymoczko J., Stryer L., *Biochemistry*, Freeman and Company, 2002
- [23] Kilian A., Müller B. "Life-like learning in technical artifacts: Biochemical vs. neuronal mechanisms", *Proc. 9th. International Conference on Neural Information Processing*, Singapore, 2002, pp 296-300.
- [24] Lengeler J., Mueller B., di Primio F. "Cognitive abilities of unicellular mechanisms" (In German), *GMD Report 57*, German National Research Center for Information Technology, Sankt Augustin, 1999
- [25] Monod J., Pardee A., Jacob F. "The genetic control of cytoplasmic expression of 'inducibility' in the synthesis of b-galactosidase by Escherichia coli", *Journal of Molecular Biology* 1, pp. 165-178, 1959
- [26] Gray J. "Notes on database operating systems", *Operating Systems: An Advanced Course*, Springer Verlag, 1978, pp. 393-481
- [27] Singhal M., Shivaratri N. *Advanced Concepts in Operating Systems*, McGraw-Hill, 1994
- [28] S. Liao, N. C. Seeman "Translation of DNA Signals into Polymer Assembly Instructions", *Science*, Vol. 306, 2004, pp. 2072-2074
- [29] M. Zhang, W. Tao, T.-J. Tarn, N. Xi, G. Li, "Interactive DNA Sequencing and Structure Design for DNA Nanoapplication", *IEEE Transactions on Nanobioscience*, Vol. 3, no. 4, 2004, pp. 286-292
- [30] J. Hogan, DNA robot takes its first steps, Journal reference: *Nano Letters* (DOI: 10.1021/nl049527q), <http://www.newscientist.com/article.ns?id=dn4958>
- [31] J.-F. Morin, Y. Shirai, James M. Tour, "En Route to a Motorized Nanocar", *Organic Letters*, Vol. 8, No. 8, 2006, pp. 1713-1716
- [32] Y. Shirai, A. J. Osgood, Y. Zhao, K. F. Kelly, J. M. Tour, "Directional Control in Thermally Driven Single-Molecule Nanocars", *Nano Letters*, Vol. 5, No. 1, 2005, pp. 2330-2334
- [33] Chworos, I. Severcan, A. Y. Koyfman, P. Weinkam, E. Oroudjev, H. G. Hansma, L. Jaeger, "Building Programmable Jigsaw Puzzles with RNA", *Science*, Vol. 306, 2004, pp. 2068-2072

Nevena Ackovska is born in Skopje, Macedonia in 1975. She holds B.Sc. in Computer Engineering, Informatics and Automation at Electrical Engineering Faculty (2000) and M. Sc. in Bioinformatics at Faculty of Natural Sciences and Mathematics (2003) at "Sts. Cyril and Methodius University" in Skopje, Macedonia. In 1995 she moved in USA and continued her education at the University of Massachusetts, Amherst, MA (Dean's List). She is currently finishing her Ph.D. research in the field of Bioinformatics at "Sts. Cyril and Methodius" University in Skopje, Macedonia.

She works as a Teaching and Research Assistant at the Institute of Informatics, Faculty of Natural Sciences and Mathematics at "Sts. Cyril and Methodius" University in Skopje, Macedonia. She is author of three books (in Macedonian) and more than 20 articles. Her research interest is the field Bioinformatics and Beings, both living and artificial.

Ms. Ackovska is IEEE and EMBS student member. She is a member of the Computer Society of Macedonia, and was a member of Save the Children Association of Macedonia (Secretary International from 1996-1997). She holds many awards at national and regional competitions in mathematics and biology.

Stevo M. Bozinovski was born in Bitola, Macedonia. He earned BSc (1973) in Computer Science, MSc. (1975) in Electronics, and PhD (1982) in Computer Science all from university of Zagreb, Croatia. All the corresponding theses were in area of neural networks.

He is an Associate Professor of Computer Science at South Carolina State University, Orangeburg, SC, USA. Previously working experience includes IBM in Bayreuth, Germany; Adaptive Networks group, University of Massachusetts, Amherst, USA; RoboCup team, Behavior Engineering group, GMD, Bonn, Germany; and Laboratory of Intelligent Machines, Sts Cyril and Methodius University in Skopje, Macedonia. He is author of four books: Artificial Intelligence (1994), Consequence Driven Systems (1995), Robotics (1997), and Operating Systems (1998). His pioneering achievements in science include solving the delayed reinforcement learning problem in neural networks (1981), controlling a robot using EEG signals (1988), and introducing both Robotics/Flexible Manufacturing (1987) and Operating Systems (2000) metaphors for genetic systems research. His current interest includes understanding and building natural and artificial creatures, including biological agents and robots. Currently he is the PI of a million dollar NSF grant in the area of brain-machine interface.

Dr. Bozinovski is a member of IEEE, New York Academy of Sciences, and Macedonian Biocybernetics Society.

Gjorgji Jovancevski was born in Macedonia. He earned his BSc (1978) in Computer Science from University of Belgrade, Serbia; MSc. (1986) in Computer Science from University of Zagreb, Croatia and PhD (1994) in Computer Science from University of Skopje, Macedonia. The PhD thesis was in the area of neural networks.

He is a Professor of Computer Science at University American College Skopje in Skopje, Macedonia. Previously working experience includes professor of Computer Science at the University "Sts Cyril and Methodius University" in Skopje, Macedonia.

He is author of two university books: Basic Programming in C++ (2007) and Algorithms and Programs (2005), and more than 30 books for secondary schools in last 15 years. His current scientific interest includes artificial intelligence and biological agents.