Information/Material Processing Synergy: Flexible Manufacturing and Operating System Metaphor for a Biological Cell

Nevena Ackovska[1] and Stevo Bozinovski[2]

[1] Institute of Informatics, University St. Cyril and Methodius,
1000 Skopje, Macedonia
nevena@ii.edu.mk
[2] South Carolina State University,
Orangeburg, SC, USA
sbozinovski@scsu.edu

Abstract. This paper presents the information/material processing synergy in both biological and human made systems. It is a further elaboration on the metaphors previously proposed for genetic information processing, such as the robotics/flexible manufacturing metaphor and the cell systems software metaphor. Related issues are also discussed, such as file system, program preparation and its parallel and distributed features, including interthread communication, among others. The paper proposes that, from a manufacturing science viewpoint, the protein biosynthesis process can be viewed as a CAD/CAM system for molecular biology.

Keywords: manufacturing science, information/material processing synergy, cell operating system, cell CAD/CAM system, distributed systems, protein biosynthesis multithreading, nanotechnology.

1 Introduction

The understanding of genetic information processing evolved through several metaphors. A metaphor represents paradigm transformation – we use knowledge from a familiar system in order to understand and develop solutions in another system. The first metaphor explaining genetic processes was the biochemistry metaphor, which basically relied on the fact that "DNA is an acid". The important breakthrough was the linguistic and information processing metaphor which stated that "DNA is a string of letters" [1], [2]. We proposed the robotics and flexible manufacturing metaphor [3], which pointed out that "DNA contains programs for machines and robots, and for example tRNA is a mobile robot". The next metaphor we proposed [4] is the operating systems and systems software metaphor, which states that "DNA is the cell real-time distributed operating system". This paper continues our research in understanding genetic information processing using the metaphors we proposed.

The idea is that a biological cell has some kind of distributed operating system, which resides on several minidisks (chromosomes) [4], [5]. The paper we are presenting here elaborates further in that direction, emphasizing that in addition to being a *parallel distributed information processing* system, a genetic system is also a real-time and *parallel distributed material processing* system [3], [6]. The basis for the material/information processing synergy is presented in Table 1.

Table 1. Information/material synergy

	Information	Material
Processing	Information processors	Material processors
Transport	Information transmission	Material transportation
Storage	Information bases	Material bases

The information/material synergy is interesting for both computer science and manufacturing science. In the following sections, we will focus the attention to the parallel and distributed information/material processing system of a biological cell. In this paper we also propose that a CAD/CAM concept is being used in genetic manufacturing.

2 The Cell and Its Parallel and Distributed File Processing Features

When we, as computer engineers, think of parallel features of computer systems, we always imagine segments of code that could be processed in parallel. Sometimes, we have the opportunity to design the hardware components of the system in order to maximize the parallel features of the software we work with. Sometimes we are able to design both, the hardware and the software for the needed system, and make its parallel features as optimal as the price/performance ratio allows us.

With the living beings this conformism is not applicable. When there is a specific situation, the living system has to act in a certain, fast manner, so it could survive. Therefore, the parallel and distributed features of the living systems are incorporated in the way they are constructed.

Here we will present some analogies between human made systems and biological systems regarding the parallel and distributed nature of information and material processing systems.

2.1 Cell Files

When studying genetics, the crucial concept is the concept of a gene. Thus, a very natural question is “What is a gene?” A usual answer is that a gene is a segment of DNA that encodes for either protein or RNA. Also, one could encounter slightly different definitions [7].

In computer science and engineering, a usual reasoning about an information processing system considers the files of that system. So, for genetic information processing we might ask the question "*what are the files of the genetic information processing system?*" Is the concept of a gene analogous to the concept of a file? Having this 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.

Looking for a concept of a file in DNA, we found that the transcription units (or scriptons [8]) are analogous to cell files. A transcription unit is a segment of DNA that eventually becomes transcribed to RNA. In prokaryotes (cells 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 (pre-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 is sometimes needed in computer file systems. Moreover, it is expected that between two fragments of a file an entire different file could be placed. This fact points to the concept of distributed file systems [9], [10]. And indeed this is the case in molecular genetics: after the first evidence that Tetrahymena ribozyme is actually an intron [11], more evidence has been found that genetic files could be found within a different file [12]. Therefore, our file-centered approach offers simple answers to nontrivial problems in genetics [13].

Now, let us consider the exons, the gene expressing 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. An example of such a phenomenon could be the building of mRNAs for antibody proteins.

We believe that while the genes are the proper concept when talking about heredity, we propose that the concept of a file is very useful in describing the DNA transcription process. This makes the first step in the analogy between computer systems and genetic systems. We believe that the whole transcription process could be better explained in file processing terms instead of linguistic terms. The cell, especially the eukaryotic cell, undergoes extensive file processing: from copying the pre-RNA file to 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 every modern computer operating system [4].

2.2 Cell Disks

Following the same 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 [14] 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. It is important to observe that DNA can easily be represented by the classical concept of a cylinder. 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. In a sense, the concept of a cylinder enables spatially distant information to be processed in parallel. And indeed, the cell disk (the complete genome) is distributed over several disk plates, or minidisks (chromosomes). The files on different chromosomes could be processed in parallel. For example, the files needed for producing an important organism substance, the hemoglobin, are distributed over both chromosome 11 and chromosome 16; parallel distributed processing is needed in order to obtain a single molecule of hemoglobin.

2.3 Process Management vs. Memory Management

Classical process management in computer systems considers a resource and several processes competing over that resource. Several mechanisms have already been developed to address this issue [15]. Basically, a mutual exclusion mechanism is used, where one process is using a resource while other processes are blocked, waiting.

It seems that in genetic systems that mechanism is not employed. Instead of process management, genetic systems rather utilize memory management. They simply keep many copies of files that are frequently used. It seems that the redundancy in the genetic system really enables parallelism. It also enables more space where the processes could be processed, since the files themselves are spatially distant. The execution can be run on several processors, distributed spatially.

Examples of such files are rRNA files needed for production of ribosomes. Those files are kept in thousands of copies [7]. In order to function in real time, a genetic system needs to produce many copies of the rRNA in the same time, in order to respond to a change in the environment. This means that no waiting for this vital resource is allowed. So, in genetic systems, in order to achieve a real-time response, many copies of rRNA files are utilized, and they are all processed in parallel, and distributed over a vast region in the cell. They are produced to become part of ribosomes, the cell processors.

During evolution, the files could be produced in as many copies as needed. This is a possible cause for file fragmentations. It also gains to redundancy. But, each process could work on its own copy of the same file. This is a space management mechanism, rather than a time management one. It is interesting that the novel approaches toward operating systems design propose a similar prospective. In the Nooks approach [16] the kernel module is cloned and then the clone is modified outside the kernel. The cloning mechanism is used in resolving the issues with threads, where instead of blocking a thread on access to shared data, a proxy of the data is generated, like in the Promisses approach [17].

There is another, very popular and rapidly developing system, which supports the claim that space management tends to replace process management. The Google File System (GFS) [18] is a scalable distributed file system for large distributed data-intensive applications. It provides fault tolerance while running on inexpensive commodity hardware, and is used by a large number of users. The large cluster used provides hundreds of terabytes of storage across thousands of disks on over a thousand machines, and it is concurrently accessed by hundreds of clients. The design space for this file system emphasizes that the space management, file redundancy and statistically based algorithmic approach are the key concepts in making GFS the supreme contemporary file system.

2.4 Time-Sharing vs. Autonomy

It seems that the time-sharing concept is not applied in the living systems. Their components are built to be extremely autonomous while they have a task to run. If the process is vital, no waiting for a processor is allowed. Instead, the new routine, run by a newly built processor will be started, it will build the needed autonomous processor, distribute it through the cell, and the desired process will be started. And, most probably, many copies of that same process will be started on many newly built processors for solving that particular problem.

2.5 Consistency

Keeping files in many copies requires 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 [7].

2.6 Pipeline Processing

The cell system is a highly distributed one and many processes are running in parallel and some of them are running in a pipeline. One prominent example is the polysome, sequence of several ribosomes over a single mRNA in prokaryotes. Many ribosomes are actually accessing the same file, but in a pipeline manner!

2.7 Parallel Processing: Inter-thread Communication

In such a system it is possible to assume an existence of communicating threads that compose one process. In order to study possible mechanisms of such an inter-process communication, we developed a software model of a minimal protein biosynthesis system (Fig. 1).

Three agents – RNA polymerase, aminoacil-tRNA synthetase, and ribosome – are engaged in inter-thread communication during the protein biosynthesis. The process is initiated on demand for a protein which activates an RNA polymerase agent. It reads the DNA bases and appends an RNA base to the evolving RNA. Once an mRNA is released, a ribosome reads it, codon by codon. For each codon it looks for corresponding anticodon-tRNA^{aa}, which will bring the needed aminoacid *aa* (Fig. 1). Actually, the physical representations of the genetic code are the tRNAs. The ribosome assembles the protein and releases it. Another agent, aminoacil-tRNA synthetase, works in parallel with the ribosome and RNA polymerase. It assures that the tRNA is loaded with the corresponding aminoacid. It receives an appropriate amino acid, receives an appropriate tRNA, loads the amino acid to the tRNA, and then releases the loaded tRNA.

```
thread RNA polymerase
repeat
  receive 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
      receive $a_i$
      receive $tRNA^i$
      load $a_i$ to $tRNA^i$
      release $a_i tRNA^i$(anticodon)
until degradation
```

```
thread Ribosome
begin
      receive mRNA
      repeat
        read codon from mRNA
        receive $a_i tRNA^i$ (anticodon(codon))
              unload $a_i$ from $a_i tRNA^i$
              assemble $a_i$ to protein
      until stopcondition(mRNA)
      release protein
recycle
```

Fig. 1. Understanding protein biosynthesis using inter-thread communication metaphor

Our model uses the **receive** and **release** primitives for inter-thread communication. The amino-acid handling part is modeled using robotic language commands, such as load, unload and assemble. Instead of using the primitives receive and release, one can use the standard primitives **receive** and **send**, or some classical primitives such as **wait** and **post** [19] which were used in some pioneering real-time multithreading robot control efforts [20], [21].

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 neglected. The model studies only possible inter-thread communication between the main protein synthesis agents. Petri nets have been also used for modeling the process shown above [4], [22].

2.8 Compilation as Manufacturing – From Source Code to a Robot

In building autonomous robots the software is developed separately from the hardware. A source code is compiled into an executable code and then an EPROM is produced that is embedded into a robot. Our research suggests that the robot producing process in cells is an extension of the compilation process: the cell operating system takes a source file (gene) and produces a robot in the form of an RNA (ribozyme) or a protein (enzyme). The cell robots and other cell machines are produced real-time, on demand. So, once demand is sensed by the DNA, it makes an appropriate source file accessible, out of which the requested program is read (the 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 protein assembly. Another example is the rRNA, which, being a part of a ribosome, performs a protein assembly function. In addition to RNA robots, the cell produces protein robots, i.e. 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.

Here we propose that protein manufacturing is actually the CAD/CAM (Computed Aided Design/Manufacturing) system in a cell. The CAD part consists of the genes carrying software for protein or RNA production, while the CAM part is the entire machinery (the genosome) that obtains the product described in a gene.

3 Just-in-Time Manufacturing

The cells are just-in-time (JIT) systems, rather than just-in-case (JIC) systems. When need is sensed, they produce the needed elements very promptly. Most of those processes are actually running in parallel: from parallel information processing, to parallel material processing in distributed sites in the cell. Not only the information, but all the products, including needed machinery and robots, are produced on demand. 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.

In order to do so, the cell keeps many copies of the most needed programs, in order to access them all in parallel when need is sensed. Furthermore, it will produce all the needed machinery for processing these programs in parallel.

3.1 Common Building/Coding Blocks

Additional adaptation towards parallel processing of a cell is that it keeps just a small number of building blocks: there are only 20 different aminoacids used for protein synthesis. This enables rapid switching between demanded products and flexible manufacturing.

It should also be noted that the cell produces its software (e.g. mRNA) and its hardware (e.g. enzyme or ribozyme) in a similar way, by appending building blocks to a sequence which in turn folds into a 3D structure. It seems that this approach in interleaving the software and hardware should be the next step in human made systems.

4 Nanotechnology and the Future of Parallel Processing Systems

Parallel distributed computation is about speeding up the applications. Real-time application in general is due to the need of real-time product. Our study shows that the future parallel distributed processing systems needs to *integrate parallel distributed material and information processing*. And this is what the nature has done long ago.

Today scientists make efforts to bring information and material processing closer to one another. Nanotechnology science makes it possible to produce nanostructures that do some kind of processing while having the means to produce other nanorobots that do specialized work in cells (and possibly self-replication) [23]. These robots have their software and 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 [24].

One open problem today 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 [25]. However, in 2005 there were some reports on nanorobots that could move in a desired direction, and were symbolically named nanocars [26], [27]. The future of these nanovehicles is to be able to carry controlled load, so they could be used for precise drug delivery. There has been a suggestion [28] that nanomachines based on the RNA are needed. It relates nanotechnology vehicles research with our early suggestion that tRNA is a mobile robot.

Having the aforementioned 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 be interested in faster algorithms, for example for floating point division, but ultimately the effects 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".

5 Conclusion

The integration of information processing and material processing is present in biological systems since the existence of life. Humans are now building flexible manufacturing systems realizing that they were actually invented by biology. Therefore, the information/material processing synergy, the concept we introduce in this paper, is essential for life.

Our research is based on two approaches: a biocybernetics one and a bionics one. The biocybernetics approach observes the biological cell as a flexible manufacturing system, controlled by a cell operating system and systems software. The bionic approach suggests improvement in human made systems by implementing the knowledge we've gained from studying biological systems.

Future systems should be built from modules that inherently contain features of mutual communication and self assembly, similar to proteins. The lesson that we could learn from molecular biology is that the compilation process in cells is actually a CAD/CAM process. We discussed that the cell robots and other OS embedded systems, such as lac-repressor or tRNA, are developed in a single program compilation/production line, not separately, as in today's technologies. To do so, the cell builds both hardware and software using the same material. These building blocks are constantly recycled, which is another important feature of genetic manufacturing and information processing.

The integration of material and information processing is already of great interest in the scientific community. Nanotechnology science makes it possible to produce nanostructures that do specialized work in cells. These robots have their software and hardware interleaved in the same building-coding blocks. Perhaps, just like the evolution of natural life, the evolution of artificial life will also continue with big steps from the nano level. In human-made systems this feature is not implemented, although there is some effort in this direction [29].

We believe that our approach based on flexible manufacturing and robotics offers better understanding of genetic processes. Conversely, from genetic information processing we have learned that the synergy between material and information processing is a crucial approach in the manufacturing and information processing and should be used in building human made systems.

References

1. Watson, J., Crick, F.: Molecular structure of nucleic acids: a structure of deoxyribose nucleic acid. Nature 171, 737–738 (1953)
2. Crick, F.: On protein synthesis. In: Proc. Symp. Society for Experimental Biology, vol. 12, pp. 138–163 (1958)
3. Bozinovski, S.: Flexible manufacturing systems: A biocybernetics approach. In: Vukobratovic, M., Popov, E. (eds.) Robotics and Flexible Manufacturing, Moscow, pp. 192–197 (1986)
4. Bozinovski, S., Jovancevski, G., Bozinovska, N.: DNA as a real time, database operating system. In: SCI 2001, Orlando, pp. 65–70 (2001)
5. Bolsover, S., Hyams, J., Jones, S., Shephard, E., White, H.: From Genes to Cells. Willey-Liss (1997)
6. Bozinovski, S., Bozinovska, L.: Manufacturing Science and Protein Biosynthesis. In: Callaos, N., Badawy, W., Bozinovski, S. (eds.) Systemics, Cybernetics, and Informatics, Orlando, pp. 59–64 (2001)
7. Brown, T.A.: Genomes, 2nd edn. Willey-Liss (2002)
8. Ratner, V.: Control Systems in Molecular Genetics, Nauka, Novosibirsk (1975)
9. Nutt, G.: Centralized and Distributed Operating Systems. Prentice-Hall, Englewood Cliffs (1992)
10. Tanenbaum, A.: Distributed Operating Systems. Prentice-Hall, Englewood Cliffs (1995)
11. Kruger, K., Grabowski, P.J., Zaug, A.J., Sands, J., Gottschling, D.E., Cech, T.R.: Self-splicing RNA: Autoexcision and autocyclization of the ribosomal RNA intervening sequence of tetrahymena. Cell 31, 147–157 (1982)
12. Been, M.: Versatility of Self-Cleaving Ribozymes. Science 313, 1745–1747 (2006)
13. Ackovska, N., Bozinovski, S., Jovancevski, G.: A New Frontier for Real – Time systems – Lessons from Molecular Biology. In: Proc. IEEE SoutheastCon, pp. 224–228 (2007)
14. Ackovska, N.: System software of minimal biological systems, PhD Thesis, Skopje (2008)
15. Andrews, G., Schneider, F.: Concepts and notations for concurrent programming. ACM Computing Surveys 15, 3–43 (1983)
16. Swift, M., Bershad, B., Levy, H.: Improving the reliability of commodity operating systems ACM Trans. Compuer Systems 23, 77–110 (2005)
17. Lee, E.: The problem with threads. Computer, 33–42 (2006)
18. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google File System. In: Proc. 19th ACM Symp. on Operating Systems Principles, Bolton Landing, NY, pp. 29–43 (2003)

19. Witt, B.: Communication modules: A software design model for concurrent distributed systems. IEEE Computer Magazine (1985)
20. Bozinovski, S., Sestakov, M.: Multitasking operating systems and their application in robot control. In: Proc. Workshop on Macedonian Informatics, Skopje, pp. 195–199 (1983) (In Macedonian)
21. Bozinovski, S.: Parallel programming for mobile robot control: Agent based approach. In: IEEE Conf. on Distributed Operating Systems, pp. 222–228. Computer Society Press (1994)
22. Balasubramanian, N., Yeh, M.-L., Chang, C.-T., Chen, S.-J.: Hierarchical Petri Nets for Modeling Metabolic Phenotype in Prokaryotes. Industrial and Engineering Chemistry Research 44, 2218–2240 (2005)
23. Liao, S., Seeman, N.C.: Translation of DNA Signals into Polymer Assembly Instructions. Science 306, 2072–2074 (2004)
24. Zhang, M., Tao, W., Tarn, T.-J., Xi, N., Li, G.: Interactive DNA Sequebce and Structure Design for DNA Nanoaplication. IEEE Transactions on Nanobioscience 3(4), 286–292 (2004)
25. Hogan, J.: DNA robot takes its first steps. Journal reference: Nano Letters, http://www.newscientist.com/article.ns?id=dn4958, doi:10.1021/n1049527q
26. Morin, J.-F., Shirai, Y., James, M.: En Route to a Motorized Nanocar. Organic Letters 8(8), 1713–1716 (2006)
27. Shirai, Y., Osgood, A.J., Zhao, Y., Kelly, K.F., Tour, J.M.: Directional Control in Thermally Driven Single-Molecule Nanocars. Nano Letters 5(1), 2330–2334 (2005)
28. Chworos, A., Severcan, I., Koyfman, A.Y., Weinkam, P., Oroudjev, E., Hansma, H.G., Jaeger, L.: Building Programmable Jigsaw Puzzles with RNA. Science 306, 2068–2072 (2004)
29. Demeester, L., Eichler, K., Loch, C.H.: Organic Production Systems: What the Biological Cell Can Teach Us About Manufacturing. Manufacturing and Service Operation Management 6(2), 115–132 (2004)