Communications of the Association for Information Systems

Volume 15 Article 15

February 2005

The Software Continuum Concept: Towards a Biologically Inspired Model for Robust E-Business Software Automation

Thang N. Nguyen
California State University, Long Beach, tnnguyen@csulb.edu

Communications of the Association for Information Systems

THE SOFTWARE CONTINUUM CONCEPT: TOWARDS A BIOLOGICALLY-INSPIRED MODEL FOR ROBUST E-BUSINESS SOFTWARE AUTOMATION

Thang N. Nguyen
California State University Long Beach
tnnguyen@csulb.edu

ABSTRACT

This paper introduces a new concept, the software continuum concept based on the observation that exists a general parallelism between the software continuum from bits to business/Internet ecosystems and the natural continuum from particles to ecosystems. The general parallelism suggests that homeomorphisms may be identified and therefore some concepts, processes, and/or mechanisms in one continuum can be investigated for application in the other continuum. We argue that the homeomorphisms give rise to a biologically-inspired architectural framework for addressing robust control, robust intelligence, and robust autonomy issues in e-business software and other business-IT integration challenges. As application, we examine the mapping of a major enterprise-level architecture framework to the biologically-inspired framework. Design considerations for robust intelligence and autonomy in large-scale software automation and some major systemic features for flexible business-IT integration are also discussed.

Keywords: business ecosystem, Internet ecosystem, ecology of software, software automation, intelligent systems, business-IT integration

I. INTRODUCTION

Nature and all its levels of organization, from particles to ecosystems have been the source of inspiration of concept development, mechanisms, and processes in many different non-biology disciplines, as detailed in Section II. In nature, one level of organization is the building block of the immediately higher one. Particles (electrons, protons) are building blocks of atoms (Figure 1), atoms of molecules, molecules of macromolecules, macromolecules of genes, genes of DNA/RNA (deoxyribonucleic acid/ribonucleic acid) in chromosomes – from which mRNA (messenger RNA), amino acid and proteins are synthesized – chromosomes of genome in cells, cells of tissues, tissues of organs, organs of organ systems, organ systems of organisms, organisms of population, populations of community and communities of ecosystems.

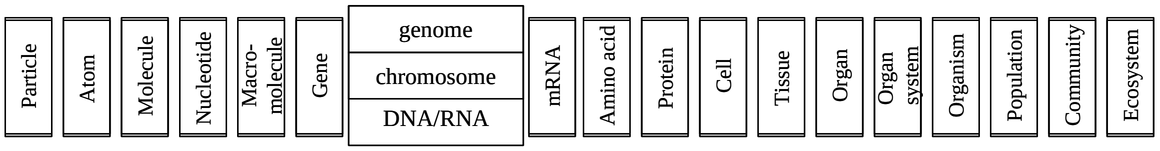

Figure 1. The Natural Continuum and Human Species

At the center of these levels of organization is the organism level, human species in particular, where higher levels of intelligence, autonomy and control exhibit. It is quite common to observe that normal, fully-developed, and healthy adult humans – as organisms - carry out their daily activities, physical or mental, with amazing ease and robustness. The activities can be as simple as raising a hand (hearing or speaking a word) or more involved such as remembering a past experience. They can be quite complex such as making a decision requiring deep thinking processes. The human capabilities include but are not limited to sensation, perception, cognition, memory, emotion, learning and intelligence.

The ease and robustness of an adult human's activities and capabilities, physical or mental, at any given time involve large numbers of chemical reactions and electrical impulses in the human body and are the results of years of a human's growth and development. In this development, a human body begins with the initial formation of a zygote (fertilized egg) developed into an embryo, then to a fetus, and at the end of a normal pregnancy term, to a newborn [Visible Embryo 2003, Carnegie Collection 2003].

The newborn's capabilities are in part gene-based (nature-driven), with the contribution of million of proteins synthesized following the transcription-translation process of the central dogma [Alberts et al., 1998]. The capabilities are nurture-based during infancy. Human capabilities such as robust autonomy and robust intelligence developed from childhood to adulthood are also learned from the acquisition, experience, use and practices of different domain knowledge and skills.

To help establish the basis of a biologically-inspired framework, we start by looking at the software continuum from bits (0 and 1) upward for similarities between software levels of organization and those of the natural continuum from the particle level up (electron and proton). It appears that we can loosely define a general parallelism between the software continuum and the natural continuum (Figure 2).

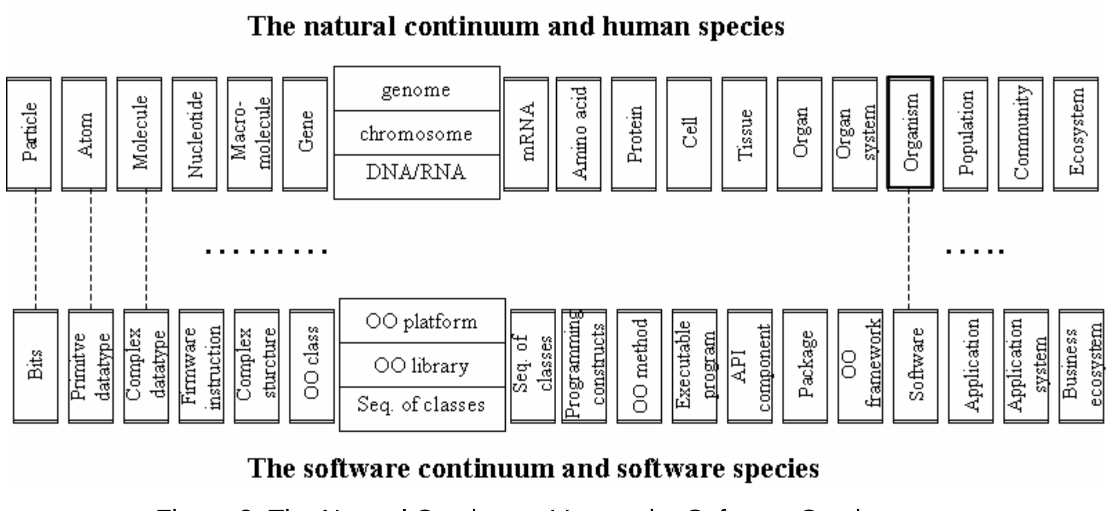

Figure 2: The Natural Continuum Versus the Software Continuum

The general parallelism gives rise to a biological characterization of the software continuum for insights into the conceptualization and realization of robustness in control, intelligence and autonomy and flexibility in business-IT integration.

We organize the remainder of the article as follows.

In Section II, we present a brief overview of previous work leading to many nature-inspired and biologically-inspired accomplishments. We pay particular attention to the work that is considered directly relevant to this research.

In Section III, we detail the claim that a software continuum from bits to Internet ecosystems exists, generally similar to the natural continuum from particles to ecosystem as sketched in Figure 2. Our software continuum concept is inspired partially from nature and partially from many previous genetically, biologically, cognitively and ecologically inspired research publications.

In Section IV, we present a biological characterization of a biologically-inspired nature-nurture-learning framework. We look at the software continuum from various perspectives: anatomical, metabolic, physiological, cognitive and ecological. In the proposed biologically-inspired framework, we present the main considerations for potential applications (Section V). We offer our concluding remarks and directions for future research in Section VI.

II. PREVIOUS WORK

Every level of organization of nature (particle, macromolecule, genome, gene, DNA, organelle, cell, tissue, organ, organ system, organism, population, community and ecosystem) as well as their cross-organizations is a complex system in its own rights. Different groups of researchers pay attention to different levels of organization for hints, metaphorical analogies and inspirations into their own research [Hunter 2004, Johnston 2004, Jog 2002, Lahoz-Beltra et al. 2002, Bozinovska et al. 2001, Shotwell 2001, Levy 1992]. These inspirations ranged from some level-specific features to systemic features, to formulate different theories and models for solutions of many different problems in many different disciplines, from adaptive CMOS hardware [Diorio 2002] and cellular gate technology [Knight 1997] to resource economic models [Geisendorf 1999].

At both the cellular level and organism level, von Neumann fathered the concept of cellular automata. The concept was based on the three abstractions: (1) cross-over and (2) mutation in heredity and genetics (after the Mendelian model) and (3) natural selection (after the Darwinian evolution theory) to arrive at a mathematical model for solving problems such as search and optimization. Von Neumann's cellular automaton or CA was extended into a computational scheme known as genetic algorithm or GA [Holland, 1992] and into genetic programming or GP [Koza et al. 2000]. Together CA, GA and GP define what is called evolutionary computing [Mitchell, 1999].

At the cell and organ level, the artificial neural network (ANN) was modeled mathematically after a single neuron model [McCullough et al. 1943]. It was extended with the notion of cell assembly [Hebb, 1949]. As a result, multi-layered neural nets were realized with many applications. Recently in 2003, Knapp et al. proposed a framework for network security with ideas drawn from biological cell and its membrane functionality [Knapp et al. 2003].

At the tissue, organ and organ-system levels, in the 1940's, von Neumann used biological metaphors of the brain and memory to create the stored program architecture of early computers that is still being used today [von Neumann, 1958]. Anthropomorphic robots were modeled after human limbs: multifingered hands [Iberall and Arbib, 1990, Nguyen and Stephanou, 1992], as well as legs and arms by others. Computer vision, speech recognition and tactile sensing are some examples of the continuing research patterned after the human senses. Autonomic computing was patterned after the autonomic nervous system by IBM, Hewlett-Packard, Microsoft, and Sun Microsystems. Conversely, Kresh and his colleagues looked at the heart as a complex adaptive system [Kresh et al., 2003]. Paton and his colleagues introduced a computational model of the human liver [Paton et al., 2003].

At the organism level, unconventional bug-like and humanoid robots were created by Brooks and his team [Brooks, 2002]. Abelson introduced the discipline of amorphous computing [Abelson, 2000]. Ray focused on synthetic biology [Ray, 1993]. Maturana and Varela co-authored the autopoiesis theory [Maturana and Varela, 1980].

At the cognitive level, an experiment by Turing in the 50's [Turing, 1953], has evolved into an active research area called artificial intelligence or AI [McCarthy 2003]. AI focuses on human symbolic-logic capability in declarative and procedural knowledge as found in expert systems, knowledge-based systems, and some decision support systems. Many others pursued different directions such as machine learning and common sense reasoning [Minsky, 1994]. Included are mathematics-based reasoning under uncertainty such as Bayesian theory, fuzzy logic [Zadeh, 1988] and the theory of evidence [Shafer, 1976].

At the population and community level of organization, we have Weiser's ubiquitous computing [Weiser, 1993], and von Neumann and Langton's artificial life [Marchal 1998, Langton 1995]. Investigations on artificial life or ALife are being conducted.

At the ecological level, the ecology of strategy was added by Iansiti and Levien [2004] after Moore's concept of business ecosystem [Moore, 1993] in his ecology of competition [Moore, 1996]. Moore's concept was extended to Cisco's Internet ecosystem [Cisco Internet Ecosystems, 2000]. The European community quickly adapted the concept to organize the multi-year billion dollar Digital Business Ecosystems project [DBE, 2001]. This project was initiated by a consortium of 20 European institutions and industries to foster local economy.

Many major biological breakthroughs and advances resulted, including:

1. theoretical formulations and models in psychology such as Hebb's cell-assembly [Hebb,1949].
2. discoveries such as DNA in 1953 [Watson and Crick, 1953] that defined molecular biology, and
3. research projects such as the Human Genome Project coordinated by US Department of Energy in 1990 [HGP, 2003].

Specifically, Hebb's cell assembly formulation not only impacted the field of psychology, but also helped create the theory of connectionism in Artificial Intelligence and a research field known as Parallel Distributed Processing. Molecular biology is the main source towards bioinformatics. The Human Genome Project finished in 2003 with a complete sequencing of all 30,000+ human genes has prompted researchers into new trails.

With the new gene sequencing, Marcus attempted to shorten the space between the genes and the mind [Marcus, 2004]. Also influenced by molecular biology are numerous researches in developmental biology, neurophysiology, neuroscience, neuropsychology and other disciplines by many noted researchers: Lorente De No, Kandel, Lashley [Kandel, 2000]. Other research streams include systemic features of complex systems investigated at all level of organization such as complexity, self-organization, emergence, and the notion of wisdom of the body in the sense of Starling, Cannon and Nuland [Nuland, 1997]. More recent are the new biologically-inspired research directions discussed in Hickman [Hickman et al. 2000] and in research programs such as Biological Information Technology and Systems – BITS of National Science Foundation [NSF 01-102, 2001] and Artificial Intelligence and Cognitive Science [NSF 03-600, 2003].

Several comments may be made on the previous work cited in this section.

- First, some biologically-inspired research efforts were somewhat disconnected and isolated, sometimes conflicting. Computer programs in some disciplines were considered as cells, but in others they were viewed as organisms.
- Second, most AI research largely ignored biology especially in the area of cognition research [Cliff, 2003].
- Third, some models were restricted to a partial functionality of a particular organ system. For example previous ANN models [Sajda, 2002] did not account for the regulatory role of hormones in the endocrine system which is crucial in the metabolic and physiologic regulation at the neural network operation level.
- Fourth, no obvious links were found between the modeling of mechanisms from one level of organization to those of the next level or between different models at the same level of organization.

Whereas DNA in the genome is the common ground for both the creation of life and the maintenance of life, no common ground is yet established for all the biologically-inspired research disciplines listed above. We claim that the new concept of software continuum will serve as the common ground. It will bridge the existing gap and take advantages of previous findings in biologically-inspired research, as argued in Section III.

III. THE SOFTWARE CONTINUUM

Conventional programming languages are a set of powerful tools in many previous and current biologically-inspired computing research and related disciplines. These include LISP, C/C++, Java and specially designed languages such as ACL - Agent Communication Language or Tierra [Ray, 1993]. However, the question of a “genomic” programming language has neither been explicitly asked nor implied. Basically we ask whether an object-oriented programming (OOP) platform such as Sun J2SE/J2EE or Microsoft .NET, can be extended and considered as similar to the role of genome rather than being just a toolset.

The first reason leading to the previous question is that the biological genome and an OOP language evolved somewhat similarly. The genome is “thought of as a summary of many generations of previous experience in dealing with environment” [Grobstein, 2004]. An OOP platform can be thought of as a summary of many generations of evolutionary programming experience.

The main reason, however, results from the observable similarities between corresponding levels of organization in nature and in software as illustrated in Figure 2. We want to find out whether OO libraries can be considered as chromosomes, OO classes as genes, the sequence of OO packages in the library as DNA sequence. Will it make sense to consider basic machine instruction set as set of codons, basic programming constructs as nucleotides, single algorithms as exons, OO methods (functions) at class level or instance level as proteins, OO utility methods as enzymes, and OO executable programs as biological cells? At levels higher than the cell, we ask whether OO API (application programming interface) are similar to tissues, OO components (e.g. COM/DCOM) to organs, OO frameworks (e.g. Oracle’s PeopleSoft, SAP R/3) to organ systems and OO application systems to organisms.

Intuitively, we know that the similarities between corresponding levels of organization, if they exist, will not be complete or perfect. Some similarities may express strong resemblance, others weaker. Differences are expected to exist.

BITS AS PARTICLES, SIMPLE DATA STRUCTURES AS ATOMS, COMPLEX DATA STRUCTURES AS MOLECULES

At the lowest level of the software organization (Table 1, right side), data and machine instructions are expressed as a series of 0's and 1's,. The bits are grouped according to some encoding scheme representing them. The bits that define the set of characters and other symbols are similar to the particles i.e. the electrons and protons (Table 1, left side). The Lewis configurations [Petrucci et al. 1993] of particles uniquely describe the corresponding atoms from hydrogen to ununoctium, the elements of the periodic table.

Table 1: The Natural Continuum Versus the Software Continuum

The natural continuum and biological building blocks	The software continuum and object-oriented building blocks
Particles (electrons, protons)	Bits (0's, 1's)
Atoms	Primitive data types (e.g. char, int, float)
Molecules	Non-primitive data types (e.g. String, struct)
Monomers	Fields
Polymers	Lists, Arrays
Macromolecules	Complex data structures, records, XML structures
Genome	OO foundation classes
Chromosomes	OO libraries
Genes	Object classes
Introns and exons	Data members and Methods
DNA/RNA	Chain of OO classes across libraries
DNA/RNA polymerase	Control program and instruction counter
Nucleotides	Firmware instructions
Amino acids	High-level programming constructs (assignment, if-then-else, while or do-while)
Proteins (hemoglobin,..)	Active algorithms, instant methods (transport, remote procedure calls)
Enzymes	Utility algorithms, utility methods
Cells	Executable programs/Dynamic Link Library (DLL)
Bacteria, Viruses, Prokaryotes	Computer viruses, Worms
Tissues (epithelial, connective, ..)	Components (Interface, API, Beans, EJB, ..)

A sequence of bits may also represent machine code. For example, '10000111' in the Intel 8085 instruction set is the encoded instruction "Add the contents of register A to the accumulator". Similarly, a chain of nucleotides represents an amino acid as a nutrient broken down from food proteins. A sequence of nucleotides also represents a genetic code, e.g. "GGC" is DNA codon for glycine.

The characters, special symbols and number digits define primitive data types such as character, integer, float, double, long or short, much as the electrons and protons define the atoms. The atoms in turn form molecules much as primitive data types form complex data types. More complex molecules such as nucleotides in DNA are formed from the combination of atoms and other molecules. Similarly, more complex data types are formed from simple ones such as those found in COBOL copybook, C struct, or C++ or Java classes. Thus, bits are similar to particles, primitive encoded data (e.g. ASCII coded data) to atoms (e.g. H, O, N or C) and non-primitive data types (e.g. String) to molecules, (e.g. CO_2, H_2O). Then, fields can be considered as analogous to monomers, and lists, arrays or XML structures to polymers and macromolecules.

OOP PLATFORM AS GENOME, LIBRARY AS CHROMOSOME, CLASS AS GENE AND SEQUENCE OF OO PACKAGES IN THE LIBRARY AS DNA

Organisms are specified by genomes. The genome in the zygote contains all biological information and is a blueprint for the development of an organism following fertilization.

OO programs are specified by OOP platforms. Unlike genome to organism development, a vendor-provided OOP platform to program development is incomplete, in the sense that OOP needs additional codes by software developers to specify a program. An OOP platform contains a basic set of available source codes to create an OO program but OOP alone does not generate new programs by itself. The development of the application program needs the intense collaboration of a software developers' knowledge and skills. Software developers must add the definitions of new data fields and new methods for newly defined or derived classes. The additions by software developers together with OOP-provided libraries then constitute a "blueprint" for a software program.

A genome contains genes in DNA packaged in chromosomes. Similarly, an OOP has classes in OO packages stored in the source libraries. In a living cell, there are existing proteins and enzymes that work together with genes to transcribe DNA into mRNA. The mRNA is then translated into new proteins. Similarly in OOP there are binary libraries that contain object code (pre-compiled) members distributed along with the source libraries.

A gene is a specific sequence of DNA in the chromosome. It is a functional unit of inheritance. An OO class is a member of its library which is similar to a chromosome. A class is a functional unit of OO inheritance, from which another OO class can be derived.

Mutations occur in genes. Likewise, changes to codes occur in existing OO classes. Of course mutations in genes are not the same as modifications to current instructions in OO classes, but they are both alterations to the originals. In this sense, we might say modification such as method overriding and method overloading is weakly analogous to mutations.

BASIC INSTRUCTIONS AS NUCLEOTIDES, PROGRAMMING CONSTRUCTS AS AMINO ACIDS, COMPILED CODES AS mRNA, METHODS (FUNCTIONS) AS PROTEINS, UTILITIES AS ENZYMES, AND COMPILATION & LINKAGE-EDITING AS TRANSCRIPTION & TRANSLATION PROCESS

DNA is a long chain of nucleotides with bases (A, T, C and G). The sequence of OO codes is a long chain of basic statements. These statements are translated into basic machine instructions. Thus, basic instructions are like nucleotides. According to the central dogma, proteins are "coded" via a transcription process from DNA to mRNA inside the nucleus and via a translation process from mRNA involving ribosomes to become proteins outside the nucleus. Analogously, the executable codes are the results of a compilation (translation) process to obtain object code or byte code and a linkage-editing process from source instructions.

Although DNA determines the sequence of 20 amino acids in proteins, the information in DNA is not used directly. RNA as a copy of DNA must be transformed to mRNA following a process called transcription that occurs in the cell nucleus. During transcription, the DNA is unwound, and the mRNA is synthesized in the direction of 5'-3' template strand of DNA. The transcription is carried out one nucleotide at a time with the help of an enzyme called RNA polymerase. The primary mRNA obtained is then edited in a process that removes the introns and joins the exons together to define a unique feature or function to mature the mRNA. The matured mRNA is further involved in the translation process outside of the nucleus to prepare the mRNA from a nucleotide sequence to an amino acid sequence. This sequence is linked in the correct order by ribosomes.

Analogously, method (functions) of an OO class are made of four basic programming constructs namely *assignment* (=), *if*-statement, *while*-statement and *do-while* statement, much like amino acids. These statements are high-level, "structured" English. They can't be used directly by the computer. They must be translated into machine language instructions. During compilation, the compiler instruction counter acts much like a RNA polymerase which facilitates the transcription process.

The instruction counter examines one instruction at a time just as the DNA polymerase opens up the DNA chain one nucleotide at a time to allow the DNA template to perform complementary-based pairing with incoming ribonucleotides. The introns (non-coding sequence) of the gene are like the space reserved for data members in a particular OO class. The instruction counter enters and exits a block of codes delimited by the set of "{" and "}"just as the polymerase recognizes the promoter region and the terminator region in DNA genes. The compiled code is followed by a linkage-editing process to prepare it as an executable program. Some similarities discussed in this paragraph were identified by Melanie Mitchell [Mitchell, 2000].

The biological process and the programming process appear to be running in opposite directions. While the nucleotides (lower level) in DNA are translated into amino acid (higher level) sequence in proteins, the programming constructs (high level) in OO method or program, are actually translated into machine instructions (lower level). While introns are stripped from mRNA during translation, OO data members (variables) are reserved spaces in the OO compilation process.

Genes are pulled from DNA and used in a transcription process to synthesize mRNA, and in turn mRNA synthesizes different proteins in a translation process. Similarly, the import statements in Java or #include statements in C++ pull OO source codes from the library during compilation. The compiled codes are link-edited to obtain an executable program much like mRNA synthesizing the intended protein. During the gene expression, enzymes such as DNA polymerase and RNA polymerase initiate, elongate, enhance, terminate or inhibit the transcription and/or translation process from genes to proteins. This is analogous to the program instruction counter and other utility algorithms of the compiler that initiate, facilitate, terminate, or inhibit the creation of new methods for new classes.

OO EXECUTABLE PROGRAMS AS BIOLOGICAL CELLS

A cell is a biological unit bounded by a cell membrane. It has a nucleus. All its organelles contained in the cytoplasm of a eukaryotic cell. Ionic particles (K++, Na++ etc.) move in and out of cell membranes by diffusion to generate action potentials for work. In a similar fashion, an OOP program (console or window application) has the main ( ) method considered as nucleus, where the main program control occurs. The curly brackets that delimit the start and end of a source program are functionally similar to the cell membrane. The coded methods in the program are similar to the cell organelles. Data move into a program via argument variables which are the inputs to be processed by the program. The program outputs the results by the return variables.

A cell can be prokaryotic i.e. it has no nucleus. Similarly, an OO program can be a compiled DLL (dynamic link library), Java applet or servlet. These elements do not have main (), much as the prokaryote does not have a nucleus. Cells such as eukaryotes can be either germ (reproductive) cells or somatic cells. Sperms and eggs are germ cells. A fertilized egg or zygote is a fusion of sperm and egg. In a similar fashion, an initial OO program can be considered as the fusion of a software developer's software concept and OOP libraries at inception. A program can replicate (e.g., [Mitchell, 1999])although it is not the same as a zygote that divides. Other examples of programs which can copy themselves are computer viruses and computer worms.

All cells except germ cells in a human body are somatic. Somatic cells are specialized cells created for a particular task. Examples are skin cells, blood cells and nerve cells. Analogously, OO executable programs are created to address a particular processing task. Examples are GUI programs capable of receiving inputs and display outputs much like sensory (skin) cells, or artificial neural nets much like nerve cells. Somatic cells divide to create two identical daughter cells. Groups of somatic cells differentiate to become constituents of different organs. OOP programs do not divide as cells do, however, they somewhat “differentiate”: by overriding or overloading existing methods from their parent or super class. This process is somewhat like a zygote that differentiates into ectoderm, mesoderm and endoderm.

An OO program contains certain major functionalities similar to cell organelles. Read and write data using get and set methods are much like the transport mechanism (out and in) across cell membranes. Decomposing data into different small pieces (fields) for computation and composing the fields into newly combined format (records) are like the operations conducted at ribosomes and at the Golgi apparatus in cells. Moving data from one method to another among different methods is like the function of the endoplasmic reticulum. Temporarily storing data internally is similar to the function of a vacuole. Transforming or converting data from one format to the next to give different meaning to the information it carries is like mitochondria that produce ATP (adenosine tri-phosphate - to be explained further in the next subsection) for use in various chemical reactions.

Other comparable constituents are shown in Table 1. Not all methods of an OO program are comparable to organelles in a cell, of course.

DATA AS MATTER, INFORMATION AS ENERGY, MESSAGES AS SECRETED CHEMICALS OR ACTION POTENTIALS, MESSAGE FORMATS AS NEUROTRANSMITTERS

A living cell requires energy to maintain its structure and processes by chemical reactions [Fox, 1996]. The reactions are coupled in the sense that one reaction releases energy for use by another reaction. The methods of an executing program are coupled, i.e. the data outputted by one method are fed into another method like coupled chemical reactions in cells. The data that are read by a method (get method) is like oxidation reaction. The one that is written (set method) is like reduction reaction. We claim that data seem similar to matter (nutrients) and information to energy (ATP). The methods in the program are like chemical reactions in a cell.

A chemical reaction is endergonic if it needs energy and exergonic if it releases energy. A method that requires input data is like an endergonic reaction. If the method outputs the data, it’s like being “exergonic”. Exergonic chemical reactions gear toward one endergonic reaction: the creation of ATP.

Table 2: Biological Processes versus Software Processes (cont.)

The natural continuum and biological building blocks	The software continuum and object-oriented building blocks
Cellular energy	Raw data
Glycolysis, Krebs cycle	Encoding scheme
ATP	Encoded data
Energy	Information
Photosynthesis	Information retrieval
Respiration	Information release
Oxidation	Information read
Reduction	Information write
Blood	Data stream
Heat	Transient information
Potential energy	Persistent information

The breakdown of ATP is used to provide energy for other chemical reactions in the cell. The breakdown of input data and build-up of output data by a program are thus similar to the breakdown of nutrients and build-up of ATP. Therefore we might say that data to a program is similar to matter to a cell, and information to a program is like energy to a cell.

Cells sense and respond to their environment which consists of nutrients, other cells and/or foreign cells (bacteria and viruses). OO programs constantly sense and respond to their environment. Like cells, they perform one-on-one communication and/or broadcast information via messages.

A signaling cell produces a particular molecule (protein, peptide, amino acid, nucleotide, etc.) that is received by the target cell by means of a particular receptor protein following a precise binding. A calling program issues a particular message addressing a particular called program that expects an incoming message of a precise format. The molecule arriving at the receiving cell is then transformed in a series of signal transduction steps into in a usable format. The called program has a method to receive message and many other methods to transform the data from the calling program into meaningful information.

Cells can signal over short range (neighboring cells) or long range. Likewise, program calls can be local or remote. Cells use hormones to broadcast signals by secreting them into the bloodstream. There are three types of hormones: autocrine (secreting internally to the cell), paracrine (secreting to neighboring cell) and endocrine (secreting to remote cell). Similarly, programs can pass messages internally, locally, or remotely to other programs.

Beside endocrine, another type of communication is exhibited in neural cell signaling that can go a long distance. An electrical impulse created at the neuron cell body or dendrite travels along the axon to another neuron. At the junction (i.e., a synapse) the impulse is converted to a chemical signal called a neurotransmitter that generates a new impulse to travel along the axon terminal. From neuron to neuron, action potentials and neurotransmitters interchange to reach the target. From program to program, messages and their formats interchange to reach the target program. Classical RPC (remote procedure call), Java RMI (remote method invocation), LPC (local procedure call), and API (application programming interface) calls are similar to a neuronal action potential, a secreted chemical or a contact-dependent binding via cell-surface receptors (i.e. ion-channel-linked, G-protein-linked or enzyme-linked receptors). Their mechanisms may be different but the nature of the calls is similar.

Intercellular and extracellular signaling cascades can distribute a signal to initiate many chemical reactions or neural processing in parallel. They can also converge to a particular location to give a complex response. The distribution (e.g. multiple thread processing) and convergence of messages inside a program or between programs are common in OO programs.

OO COMPONENTS/PATTERNS AS BIOLOGICAL TISSUES, FRAMEWORKS AS ORGANS, SOFTWARE PRODUCTS AS ORGAN SYSTEMS AND SOFTWARE APPLICATIONS AS LIVING ORGANISMS

Multiple cells of similar functions make up tissues. Tissues of more than one type make up organs. The activities of and interactions between tissues determine the physiology of the organs (Table 3).

The OO components such as COM/DCOM (component object model/distributed COM), CORBA-compliant ORB (Common Object Request Broker Architecture-object request broker) or Java RMI as well as beans or EJB (enterprise java beans) are component software architectures that allows applications to be built and put together for interoperability [Iyer et al. 2003, Linthicum 2001]. OO components are like the tissues which make up of different organs.

The OO components and patterns for building a GUI in an OO application are much like epithelial tissue (e.g. skin). Via the GUI, data are captured for processing. Via skin, sensation and pain are captured. Blood tissue carries nutrients, wastes, oxygen and carbon dioxide across the circulation system. Neural tissue carries electrical

Table 3. The Natural Continuum Versus The Software Continuum (cont.)

The natural continuum and biological building blocks	The software continuum and object-oriented building blocks
Tissues (epithelial, connective,..)	Components (interface, API, Beans, EJB,..)
Nervous tissue	Control program, executive programs
Muscle tissue	Action agents, probes
Epithelial tissue	Interface
Connective tissue	RPC, DCE, ORB programming
Glands	Applets, Servlets, ASP, JSP, databases
Organs	Software packages, shrink-wrapped
Brains	Operating systems, executives
Heart	Software engines, databases
Kidney	Editing, Filtering systems
Lungs	GUI interfaces, user interfaces
Muscles	Command and Control systems
Limbs	Adapters, Connectors
Organ systems	Software components systems
Skin	System and/or Network management
Circulatory system	Network, communication
Sensory system	Agent system, robotic sensing (tactile, vision, etc.)
Nervous system	Neural nets
CNS/ANS	AI
Immune system	Error handling system, security
Respiratory system	Regulation system
Digestive system	Data manipulation, garbage collection
Reproduction system	Genetic programming
Organism	Software species

impulses across the neural nets. TCP/IP protocol suite (an implementation of the ISO/OSI – International Standardization Organization/Open System Interconnect - reference model) and message- oriented middleware [Altman et al. 1999] such as BM MQSeries [Blakeley 1995] or Microsoft MSMQ for transport of asynchronous messages across applications are similar to blood tissue and neural tissue.

OO frameworks are found in many vendors' products (e.g. ERP – enterprise resource planning, SCM – supply chain management, Oracle's Peoplesoft, SAP R/3) are much like organs that provide complex functionalities. Relational databases and other types of DBMS (Oracle, DB2, IMS, Informix, Sybase) are analogous to the heart. Databases stored data and pump data records in and out of the database or modify them by SQL statements such as select, insert, delete, and update. The heart stores blood and pumps it to and from atria and ventricles. ERP, SCM frameworks and databases can then be considered as biologically similar to organs. Table 3 lists other OO constructs which are similar to various biological organs.

Like human organs, OO frameworks or databases do not function as-is. They need be augmented with user application codes to become operational OO application systems just as organs that must be part of an organ system (e.g. circulation, respiration, etc.) to be operational. OO *application systems* (e.g. university's administrative systems) therefore are analogous to the biological organ systems.

The above similarities between lower-level building blocks (between cells and programs, tissues and components, organs and frameworks/databases, and organs systems and applications) suggest that software application systems can be considered somewhat similar to organisms.

OO E-BUSINESS APPLICATIONS AS POPULATION/COMMUNITY AND BUSINESS/ INTERNET ECOSYSTEMS AS NATURAL ECOSYSTEMS

Similar species constitute a population. The collection of different species makes up a community. Likewise, OO homogenous software collection forms a population. OO heterogeneous software collection (e.g. the collection of different software installed in a computer as well as across many machines and operating systems) are similar to natural communities (Table 4).

Table 4. The natural continuum versus The software continuum (cont.)

The natural continuum and biological building blocks	The software continuum and object-oriented building blocks
Organism	Software species
Microbe	Computer virus
Population	Platform (mainframe, Unix, Windows,.., STP, EDI, stocks, manufacturing, Scientific, industrial,..)
Community	Heterogeneous platform (SCM, ERP, enterprise application integration,..)
Ecosystem	Business ecosystem (Internet ecosystem)

There are different kinds of software “species” in an operating environment. These include: (1) mainframe operating systems, UNIX of different flavors, Sun Solaris, HP OS, Linux, Microsoft Windows, etc., (2) applications generated by different languages (traditional application systems on the mainframe such as IBM CICS - Customer Information Control System), (3) DBMS - Oracle, DB2, IMS, Informix or Sybase), (4) network operating systems (Novell, CISCO) and (5) OO application systems. They interact, “live” together and make up of the ecology of software.

In fact, this notion of a software community and the relationship among its members can be extended to the notion of Internet ecosystems and business ecosystems. Thus, the software continuum concept from bits to business/Internet ecosystem introduced in the previous sections appears mappable to its biological counterpart, the natural continuum from particles to natural ecosystems.

IV. TOWARDS A BIOLOGICALLY-INSPIRED SOFTWARE AUTOMATION FRAMEWORK FOR E-BUSINESS: A BIOLOGICAL CHARACTERIZATION OF THE SOFTWARE CONTINUUM

The general parallelism elaborated in section III gives rise to a characterization of the software continuum from perspectives similar to those of the natural continuum, namely, the anatomical, metabolic, physiological, cognitive and ecological perspectives. Our intention is to examine whether the biological characterization can offer new insights into homotopic properties towards a biologically-inspired framework for the making of robust e-business software automation as well as business-IT challenges.

ANATOMICAL ASPECT: FULLY-CONNECTEDNESS

In a human body, a certain level of organization, e.g. organ, is the container of the immediately lower level components e.g. tissue (Figure 3). Some elements or components of a container in an organism are however not entirely isolated from those in another container. Examples are nerve cells in the central nervous system and peripheral nervous system (containers: brain and spinal cord) of a human body. The nervous system reaches (connect) every part (every other container: organs and tissues) of the body. The connectivity of the nerve cells demonstrates the *fully-*

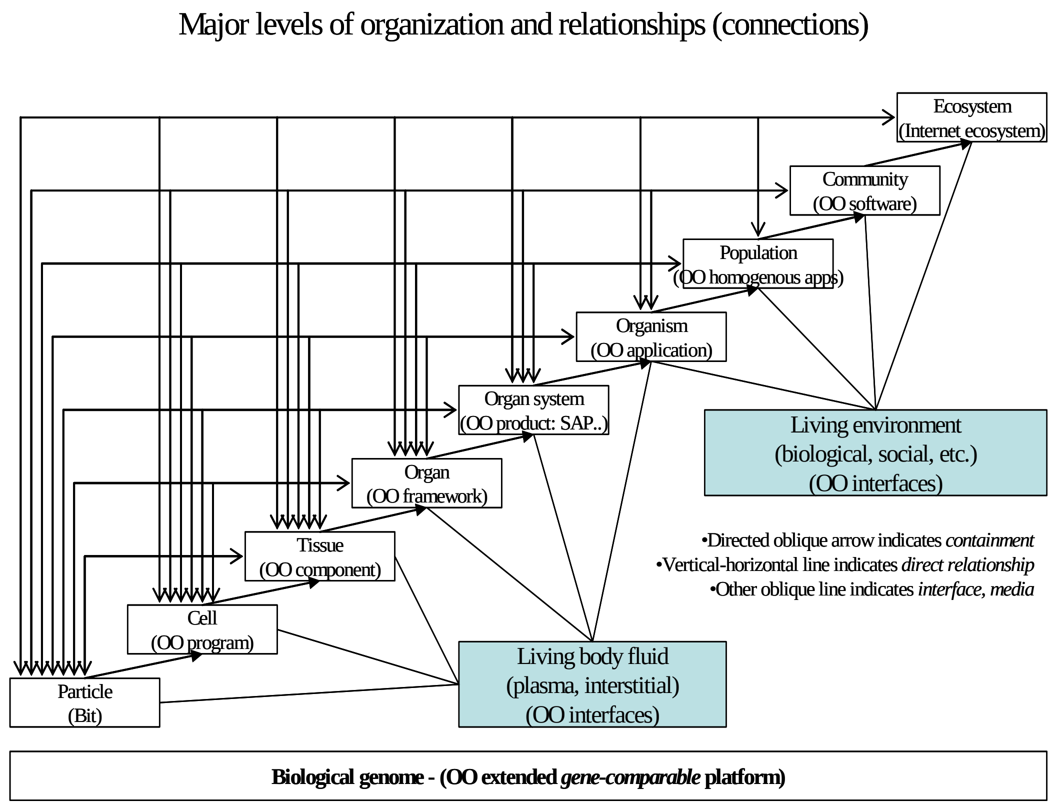
Major levels of organization and relationships (connections)
Ecosystem (Internet ecosystem)
Community (OO software)
Population (OO homogenous apps)
Organism (OO application)
Organ system (OO product: SAP..)
Organ (OO framework)
Tissue (OO component)
Cell (OO program)
Particle (Bit)
Living environment (biological, social, etc.) (OO interfaces)
Living body fluid (plasma, interstitial) (OO interfaces)
•Directed oblique arrow indicates *containment*
•Vertical-horizontal line indicates *direct relationship*
•Other oblique line indicates *interface, media*
Biological genome - (OO extended *gene-comparable* platform)

Figure 3. Biologically-Inspired OO Framework for Software Automation

connectedness (or near-fully) property in the body. The interface between the cells is what Bernard called the “milieu interieur” and Nuland dubbed the “constant sea within”.

The fully-connectedness provides only connectivity. It does not necessarily permit the receiver to accept the information from the sender. The transfer of information requires a proper interface at the receiving end. For example the hormones secreted by various endocrine cells and glands reach the cells of the target organs or tissues via the blood circulation as transport mechanism from the heart to aorta, arteries, arterioles and finally to capillaries. Proper receptors situated on the surface of the target cells must be available for the hormonal binding to take place.

There are different interface structures among levels of organization. We differentiate two major types of interface in biological systems: *intra*-organism, and *inter*-organism. In the first type which occurs within an organism, the interface is identified as the interstitial fluid between cells or the plasma fluid in the blood. The second type of interface, inter-organism, occurs among organisms via other media such as air and water ranging from viral and bacterial airborne from an organism to another or macroscopically social contacts to the exchange of body fluid or blood transfusion between two human bodies. The fully-connectedness serves as a means for end-to-end transport and signaling, locally or remotely.

The connectedness property is found in the layered software as shown in Figure 3. Similar interface types (intra- and inter-) are observable in software calls. The fully-connectedness property however is not found in most software. Layered software systems and software architectures were designed to hide complexity (e.g. Parnas' information hiding) as indicated by the oblique arrows between two adjacent levels of organization.

In fact, unlike the biological building blocks, in telecommunications and in most software systems and applications' building blocks, a particular software layer does not connect to those which are not immediately above or below it. For example, the ISO/OSI reference model defines 7 layers [Comer et al, 1994]: (1) *physical* at the bottom, (2) *datalink*, (3) *network*, (4) *transport*, (5) *session*, (6) *presentation*, and (7) *application* at the top. A particular layer does not have the ability to directly influence the operations, functions and behavior of other distant layers as seen in a human body.

Thus, we claim that fully-connectedness or *near-fully-connectedness* within a software organism is a property which should be observed if robustness feature is desired in biologically-inspired software.

METABOLIC ASPECT: HORMONES AND HORMONAL CONTROL

Metabolism is the collection of chemical reactions that (1) breakdown compounds to capture the required energy (2) synthesize small molecules and (3) convert the excess of compounds for storage or for waste excretion. The chemical reactions occur at the intracellular and intercellular level. An example at the intracellular level includes the breakdown of simple carbohydrates (e.g. glucose) to release ATP in a process called *catabolism* and in the acquisition of energy to produce other molecules in the process of *anabolism*. Both catabolism and anabolism make up the metabolism in the body [Solomon et al. 1996]. An example at the intercellular level is the gas exchange between air and blood in the lungs and between the blood cells and tissues of the body via the capillaries [Fox 1996].

While it is not obvious to think of software functions which are homotopic to the glycolysis, Krebs cycle and oxidative phosphorylation/energy transport chain occurred in the cytoplasm and/or mitochondria of a cell, the metabolism concept is applicable to software programs. In fact, catabolism and anabolism respectively can be considered similar to the decomposition and composition of information (which is viewed as energy, as argued in section III).

From the metabolic aspect, the occurrence of pathways requires two biological elements: binding and enzymes. Specific bindings exist just as signatures in the software function calls do (a signature consists of the name of the function call, and its arguments and associated data types). Enzymes (and co-enzymes) are present for the chemical reactions to take place just as software utilities (e.g. decimal-to-integer or vice versa) – considered as similar to enzymes – must be available for the preparation of proper data format for processing. The binding and the enzymes allow proper metabolic pathways (e.g. the product of one reaction is the substrate of the next reaction in a series of reactions) to occur.

For a particular metabolic pathway, for example the hormonal pathway, there are three possible actions: (1) *endocrine*, (2) *paracrine* (3) *autocrine* as discussed previously. Note that in hormonal pathways, the binding of hormonal receptors can be of *agonist* type (inducing events) or *antagonist* type (blocking the binding of the agonist).

The caller-callee scheme implemented in software is point-to-point within a program or between two programs, just as the hormonal pathway. Existing software schemes have the same capability (local and remote calls) just as hormonal schemes (i.e. autocrine, paracrine and endocrine) do. There is no agonist-antagonist type of implementation in software, however.

In hormonal pathways, the pituitary organ is the master gland of the human body. It secretes hormones under the control of the hypothalamus. The anterior pituitary and posterior pituitary secrete a collection of hormones that affect virtually all physiological processes. These hormones in turn can be inhibited or released by those secreted by the hypothalamus via negative feedback control mechanism. The two stage-control mechanism (hypothalamus-pituitary) is rarely found in current software program architecture. Current work in artificial neural nets does not consider the endocrine idea.

We claim that the software infrastructure should allow a communication mechanism to be “hormone-like”: in addition to the capability of point-to-point, selected broadcast and full broadcast a biologically-inspired software should have the agonist/antagonist property as well as hormonal dual control.

PHYSIOLOGICAL ASPECT: WIDE COLLABORATION AND HOMEOSTASIS

Organisms have the basic biological levels of organizations: cell, tissue, organ and organ system. At the lower level, their cells are subject to all the same biological mechanisms (e.g. mitosis, meiosis, glycolysis, Krebs cycle, etc.). At the higher level, their bodies are governed by the same life processes (e.g. respiration, excretion, reproduction, growth, etc.). All levels of organizations obey the same physical and chemical laws.

At the cellular level, the physical body involves the same types of cells, germ and somatic, and the same kinds of organelles in each cell. The resulting cells (in order of billions) by division and differentiation bathe in the same types of fluid: interstitial and blood plasma for the exchange of oxygen, carbon dioxide, nutrients digested from food, and wastes to be disposed (re. section III). At the tissue level, the body has the same four major types of tissues: epithelial, nervous, muscular and connective. The tissues make up of many types of organs: brain, heart, lung, liver, etc. and ten organ systems.

The organs and organ systems of the organisms work in collaboration. The respiration system gets the oxygen during inhale and attaches the oxygen to the hemoglobin of red blood cells for transportation via the pulmonary circulatory system. It also collects carbon dioxide to dispose during exhale. The digestive system breaks down the food into the nutrients. The latter are absorbed most commonly through the intestinal wall into blood vessels of the systemic circulatory system. Working together with the respiratory, digestive and urinary system, the systemic circulatory system delivers nutrients and oxygen to every single cell where nutrients are absorbed at the interstitial fluid. Carbon dioxide and waste of different kinds, organic and inorganic, are collected and then disposed.

This collaboration between the organ systems is to keep all cells alive and active. In conjunction with the dual, layered hormonal control discussed in the previous section, the physiology of the organism via wide collaboration is to maintain homeostasis. Homeostasis expresses the internal harmony. The controls for homeostasis involve the interoperability of the entire body. Failure of control will lead to catastrophic result. It is known that blood concentration of glucose throughout the body must be higher than 1mg per ml. If it drops below that threshold, neurons will misbehave, organs and organ systems will fail and coma and death will ensue.

In some fashion, the GUI or presentation layer of the software application is partially similar to the sensory system and partially to the respiratory system. Raw or XML data are captured, processed and outputted back to the presentation layer. The manipulation of data by the business logic layer of the software is like the digestive system. The data are processed by the database logic where the database engine acts more or less like the heart. The data are distributed via a network communication much as the circulatory system does the blood. The heart pumps the blood carrying nutrients and hemoglobin throughout the body just as the database pumps the data in and out via the network communications for processing by the database and other applications.

Trillions of chemical reactions take place to allow the whole body to continue in homeostasis and health. All computing processes are to keep the business, scientific and industrial applications and systems connected, alive, well behaved and in adequate communication.

COGNITIVE ASPECT: NATURE AND NURTURE DEVELOPMENT, AND THE WISDOM OF THE BODY

The cognitive capabilities involve two major processes: autonomic and learned. The first involves nature, the second involves nurture. The following provides a little more details on the cognitive capability development.

It is common knowledge that at the end of the first trimester of human pregnancy, even though an embryo (now called a fetus) is about three inches long and one ounce heavy, it is anatomically *almost* complete. The major organ, the brain, is being completed [Scheibel, 2004]. The heart beats, the blood circulation flows, and the kidneys begin to make urine. During the second trimester, the fetus is able to suck the thumb or perform swallowing motions and react to stimuli such as pressure, as well as hear the mother’s voice while in the uterus. During the third trimester, the fetus becomes active, coordinated suck and swallow reflex develops, their eyes responds to changes between dark and light within the womb. Motions are noticeable, and brain nerves grow rapidly. Prior to birth, the last organ, i.e. the lungs, is maturing. At birth, newborns are considered as *anatomically and physiologically complete*, which allow the partial development of the autonomic cognitive capabilities by nature and nurture.

From birth, newborns are to be nurtured. They are fast learners. They begin to tune in to basic sounds of words and sentence patterns, the building blocks of thought. At the end of one month, they can produce some vowel sounds. They can follow objects with their eyes. They form images as their versions of reality. They develop a sense of perception. Between two and four months, they can roll from back to side, lift their head, stretch the limbs, hold objects indicating the start of their autonomous capability. They recognize their names after several months. They understand simple requests. They also learn how to crawl after six to eight months, and walk after the first year. They have a vocabulary of about 50 words in their first year. Most amazingly, they develop simple concepts of “in” vs. “out”, “ugly” vs. “beautiful”, “bad” vs. “good” and show some coordination of perception and action as well as of perception and decision. Continuing the growth and development by nature and nurture, the newborns’ brain makes more synaptic connections, strengthen their cell signaling and increases the number of synapses. Detailed description of the preceding discussion can be found in many standard textbooks on psychology, developmental psychology and cognitive science [Morris el al. 2002, Craig et al. 2001, Sobel 2001].

We might say that the newborns, at the time of their birth, possess more or less the same average potentials of learning and the abilities of skill development. The newborn interacts with the environment over time to further grow and develop, and to acquire and use different domain knowledge. Gradually the newborn becomes a toddler, a child, a teenager and then an adult capable of cognitive activities as well as physical and mental capabilities: walking, running, understanding new knowledge, reasoning and acting upon in new situations. The combined (physical and cognitive) growth allows the newborn to integrate sensation, perception, action and decision, etc. to develop speech, vision, taste, smell, touch, balance, motion, memory, etc. and become skillful physically and mentally.

The fascinating creation of organisms and their growth and development involves both nature and nurture, followed by learning from numerous life experiences. Since the discovery of molecular biology, these high-level cognitive capabilities (e.g. thinking, thought, memory, etc.) can be explained by linking them to their low-level biological basis [Marcus, 2004]. We claim that the software continuum fosters a software development similar to the development by nature to be followed by a development by nurture.

ECOLOGICAL ASPECT: SELF-ORGANIZATION, EMERGENCE, EVOLUTION AND COMPLEXITY

The natural continuum concepts also gives rise to other important systemic properties, e.g. *emergence* and *self-organization* as observed in natural and biological systems. These concepts have been extensively studied by numerous researchers [Flake, 2003] in the context of general system theory [von Bertalanffy 1969, Capra 1996], complexity theory [Kauffman 1995, Woffram 2002], chaos theory [Nicolis et al. 1989, Prigogine 1984, Mandelbrot 1983, Lorenz 1993] at all levels of organization. The concepts offer links to Starling's concept of wisdom of the body [Cannon 1938, Nuland 1997, Zajicek 2003].

The systemic features will be addressed in details in a separate paper. In the next section, we attempt to elaborate some design considerations for robustness in e-business software automation servicing the Internet.

V. EXAMPLE MAPPING OF A MAJOR ARCHITECTURE FRAMEWORK TO THE BIOLOGICALLY INSPIRED FRAMEWORK

To apply the software continuum concept to real-life large-scale applications, this section attempts to draw an example mapping between a major architecture framework (AF) such as Homeland Security/Homeland Defense (HS/HD) AF [Dawson 2003] and the proposed biologically-inspired framework. We hope to gain insights into the conceptualization and realization of two important features of software automation driving the applications: robustness in software intelligence and autonomy and flexibility in strategy-operations integration.

HS/HD deals with all sorts of threats against our homeland. The threats range from some large refugee flow potentially offending the nation security to weapons of mass destruction. A brief discussion on the basic understanding of HS/HD is given by Larson and Peters [Larson et al. 2004].

To address these threats, HS/HD requires an integrated cross-enterprise architecture and infrastructure protection. It requires timely and accurate, shared and easily accessed information across all federal, state and local jurisdictions for decision making [GAO Homeland Security, 2003]. Many vendors have proposed different approaches and a wide range of tools for HS/HD enterprise architecture (EA) especially in the area of information sharing.

The discussion on HS/HD mapping may well be presented [Farah-Stapleton 2004] from the ARMY/DARPA FCS (Future Combat Systems) C4ISR context (Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance) since the latter covers a larger range of operations than HS/HD. Note that the information on both HS/HD and FCS/C4ISR presented here is available from plenty of unclassified documents published on the Internet. The information is obtainable using various search keywords such as "Homeland Security architecture", "DODAF", "FCS", "C4ISR", "SoSCOE", "network-centric" or "service-based architecture", to name a few.

The DoD AF V.1 [DoD, 2003] defines a common approach for all DoD other architectures such as C4ISR AF. Although it was born after the C4ISR AF V2 [C4ISR 1997], it was the product of lessons learned from the C4ISR AF V.1 and V.2 effort [Wood et al, 2003, GAO 2004]. It became not only a reference model for military mission but its prospect was intended to encompass potentially most federal and business enterprises. These include TEAF of Treasury Department and FEAF for federal enterprises.

AN EXAMPLE ARCHITECTURE FRAMEWORK MAPPING

FCS is a collection of systems for the Army's future force. The FCS development started in 2002 will cost 22+ billion dollars over a period of 12 years. It involves a few dozens of contractors/suppliers selected among the best of the industry. It is currently at the SDD phase (system development and demonstration - Milestone B). Its supporting C4ISR software has an estimated 34+ million lines of code. The C4ISR is supposed to address a wide spectrum of military operations [2003 Army] ranging from peace keeping, through special operations, to future conventional large-scale warfare of the year 2014 and beyond.

To that end, C4ISR uses and exploits the network-centricity concept for the development and deployment of its system of systems (SoS) integrating a family of 18+ manned and unmanned air/ground combat systems called platforms. The different platforms include vehicles, sensors and munitions. Its design is to ensure that the Army's Objective Force – an agile, flexible, deployable, mobile, lethal, survivable and sustainable future force – and its basic combat unit called Unit of Action (UA) are strategically responsive and dominant. The underlying concepts for the UA operational requirements are: *see first, act first, finish first, and finish decisively*. The UA is equated to the brigade level of organization.

The C4ISR architecture framework is also service-based, e.g. most if not all C4ISR activities and tasks are based on the "service requester- service provider" scheme over a networked collection of capabilities. The central piece of the C4ISR service-based architecture is the SoSCOE (System-of-Systems Common Operating Environment). It consists of many service families ranging from basic common system-level services to distributed application services. The range of services include knowledge services, administrative services, communication services, data store services, information services, interoperability services and many other supporting services [SoSCOE, 2003].

From the application perspective, the UA uses the following major service components: planning and preparation service (PPS), situation understanding service (SU), battle command mission execution service (BCME) and warfighter machine interface (WMI). It is supported by a network communication system encompassing some important existing (JTRS – Join Tactical Radio System – and Win-T – Warfighter Information Network/Tactical) and future communication technologies. Sensor-based SU service can be equated to See first and Understand first, PPS service to Understand first and Act first, and BCME service to See first, Understand first, Act first and Finish decisively. The application service components rely on the SoSCOE knowledge service, among others. The latter is an enterprise-level, complex workflow-like, policy-based, inference-driven task integration network (TIN) to handle different roles and scopes of operations. SoSCOE is the engine driving all important activities.

As the central piece of C4ISR, SoSCOE architecture dictates the scope of extensibility of the UA to cope with a wide range of warfare. It is currently focused on the conventional large-scale warfare. The extensibility to other warfare remains to be explored.

From the biologically-inspired framework perspective, we consider the FCS C4ISR UA as a higher organism and predator. The SoSCOE or an equivalent system is then a combination of some major organ systems coordinating the collection of biologically-inspired services satisfying the properties described in Section IV. As such, the UA network communications is much similar to the circulation and/or endocrine backbones. The PPS and SU services can be partially mapped to those provided by the nervous system. The BCME and WMI define the other physiological capabilities (autonomy and cognitive maps) guiding the behavior of the entire digital organism.

Each of the 18 platforms used by the UA can be considered as different organism or microorganism. These platform organisms can be considered as belonging to or living outside of the UA. Depending on the nature of the platform, each can also be considered as an organ or major organ system of the higher organism.

In the following, we elaborate the FCS C4ISR considerations for: (1) the making of C4ISR software automation to attain the robustness in intelligence and autonomy and (2) the flexibility of high-level strategy to low-level operations integration in the C4ISR for its adaptation to different warfare in the entire spectrum of operations other than just conventional large-scale warfare.

THE MAKING OF C4ISR UA: A GENOMIC, ANATOMICALLY-NECESSARY, PHYSIOLOGICALLY-SUFFICIENT AND COGNITIVELY-ABLE SOFTWARE

The following sketches a general process by which the software automation (e.g. Homeland Security or C4ISR) can be conceptualized and constructed. The process consists of the following major activities:

1. **Anatomically necessary software build:** Model and prototype a generic software program that is anatomically necessary (rather than anatomically complete) similar to embryonic development [Kimball 2003]. The embryo-like development follows four major phases: software conceptualization (similar to fertilization), patterning (similar to the separation of ectoderm, mesoderm and endoderm from the blastocyst), differentiation (partial development of organ-like and organ system-like functionality), and growth (to become anatomically necessary). One such attempt is drawn in Nguyen [Nguyen, 2003].
2. **Physiologically sufficient software build:** Develop the "fetus" software in a fashion similar to physiological development. This activity is to build features of the software that are operational physiologically. The software has a brain-like component based on the McCullough and Pitts's neuron model, Hebb's cell-assembly and connectionist model with appropriate considerations for reverberation effect and feedback mechanisms. It has a systemic communication with ideas borrowed from human systemic circulation. It also has the information/data manipulation capability with ideas borrowed from the digestive system. It has the self-repair mechanism with ideas borrowed from the immune system.
3. **Cognitively able software build:** Nurture the "newborn" software in a fashion similar to growth and development of a newborn child. This activity is to build intelligence (simply defined as the timely collaboration of perception and decision making) and autonomy (the timely collaboration of perception and action) and tune its "software brain/mind" in selected domain knowledge. The software might use methods and techniques in AI (e.g. neural nets, reasoning schemes, smart machines) and those of cognitive science in a fashion similar to child learning and development.
4. **Software copied for intended application services:** Copy the developed software for added domain knowledge to support C4ISR UA*,* i.e. build once, adapt anywhere. This is where the ultimate cost-benefit advantage of the proposed process will be seen.

The above generally constitutes a software organism that is "anatomically necessary", "physiologically sufficient", "cognitively able", and "copiable".

At this time of our research, the key notion is *partially* genomic, i.e. only enough genomic OO classes are developed to mimic certain selected human capabilities. Obviously this task is not easy and is bound to be expensive and time consuming.

FLEXIBLE STRATEGY-OPERATION INTEGRATION

High-level strategy to low-level operations linkage in a business enterprise is a complex reality. Most research on this linkage therefore is directed mainly towards the alignment between strategy and operations to simplify the issues under consideration. The alignment, as its name implies, hinders, however, the exploitation of either high-level abilities or low-level capabilities, and the flexibility of changing them.

It is no different for the HS/HD or C4ISR strategy-operation integration. First, the inertia embedded in the organizational hierarchy or pyramid can't be ignored [Schenk 2000] in these organizations. Second, the current emphasis on SoSCOE service-based architecture for large-scale conventional warfare makes it difficult to address the strategic-operational development of warfare of different types such as peace making, urban operations, guerrilla, counterinsurgency, and/or counterterrorism.

We claim that the software continuum framework offers a direct linkage between the proposed biologically-inspired HS/HD or C4ISR and the different types of warfare strategic development. It is because the biologically-inspired framework offers, in addition to the robust capabilities, a rich collection of self-healing, self-protection processes much as the living organisms do against microorganisms and different types of diseases. Guerilla and terrorism warfare can be equated to biological attacks by microorganisms. Many known biological schemes can be considered for applicability to the software automation if a biologically-inspired approach is adopted. This initial idea of the general mapping not only preserves the current properties of SoSCOE service concept of the C4ISR such as service independence, common interface, and invocation, but also offers higher flexibility to address different types of warfare. The flexibility in strategy-operation integration is inherent because the living organism is flexible and robust for a wide range of higher functions and behavior. Thus, the newly-mapped, biologically-inspired, resulting UA interacts with the other supporting organisms in an ecology of warfare to fight against the future enemy forces to gain competitive edge, to become responsive and dominant in the battlefield.

VI. CONCLUDING REMARKS

Because biology is complex, so are biologically-inspired systems. The software continuum concept mimicking the natural continuum, proposed in this research is therefore complex. The work presented in this paper constitutes a general biologically-inspired foundation during the general conceptualization phase in the quest for robust, intelligent and autonomous software development. Our work is incomplete and still evolving. We still need to sharpen the details in many areas. The effort is long-term but potentially viable. The proposed biologically-inspired framework will be cost-effective since "we build once, adapt anywhere". It is similar to the Java concept of *write once, run anywhere* [Curtin, 1998]. It has the potential to include new computational technologies if available and applicable, just as humans are able to learn new knowledge and develop new skills.

Previous research tackled many areas of biological levels of organization. The biologically-inspired mechanisms continue to become candidates for our adaptation. Current OOP packages for biology-computer interface work are available for extension, and other packages that are gene-comparable can be built. So are existing OO components and frameworks. Some current mathematical models are available for use, for example, von Neumann's cellular automata, McCullough and Pitts' neural model, Hebb's cell assembly model, and Barabasi's work on cell network [Barabasi 2004]. Some capabilities that emerge from the bottom up may be investigated since our framework follows the same biological levels of organization.

The software continuum concept presented in this paper may be viewed as a demonstration of Hewitt's [2003] comment on continuity in his discussion on Read's mind and brain

> "*For an evolutionary theory to be convincing we must, as I say, be able to demonstrate continuity, that is, we need to be able to point to a continuous range of function right up from inanimate molecules through levels of increasing complexity to the human. So that what exists nascently at a lower level can be seen to have fulfilled its potential further up the ladder; in particular, we would like to be able to show how mind emerges from matter…*"

In general, the links between the natural continuum and software continuum appear natural. Biological systems and man-made systems start from the same point of departure: the elements (atoms) described in the periodic table, where hydrogen is the element with the lowest atomic number 1 and ununoctium is the currently highest, with atomic number 118. In fact, a biological cell, the unit of life, is made of organic compounds (macromolecules) which in turn are made of molecules of various inorganic elements (C, H, N, O and others). A software program, considered here as the unit of computer execution, is made of software instructions which are translated into machine codes running on computer hardware. The hardware itself is built from materials made of elements drawn from the same set of available atoms described in the periodic table.

When applying the concept of the software continuum to real world problems we notice that it has the potential to address infrastructure issues as implied in its levels of organization comparable to the levels of organization of nature. This observation opens up areas of investigation in support of business ecosystem and Internet ecosystem concepts and their business models as sketched by James Moore and Cisco or the ICT European consortium for the DBE project. It also leads us to another question, with some philosophical implication: whether the man-made OOP "platforms" (C++ or Java) have been unconsciously patterned after biological systems due to the link between them: the human mind.

ACKNOWLEDGEMENTS

Thanks to Professor Jenny Gilsdorf of CSU Long Beach for extensively reviewing and editing the final version of the article and to Ms. Stephenie Caravan for the initial proofreading.

Editor's Note: This article was received on May 17, 2004. It was with the author for 6 months for two revisions. It was published on February 24, 2005

VII. REFERENCES

EDITOR'S NOTE: The following reference list contains the address of World Wide Web pages. Readers who have the ability to access the Web directly from their computer or are reading the paper on the Web, can gain direct access to these references. Readers are warned, however, that

1. these links existed as of the date of publication but are not guaranteed to be working thereafter.

2. the contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.

3. the authors of the Web pages, not CAIS, are responsible for the accuracy of their content.

4. the author of this article, not CAIS, is responsible for the accuracy of the URL and version information.

2003 Army Transformation Roadmap. (2003). http://www.army.mil/2003TransformationRoadmap/

Abelson, H. et al. (2000). "Amorphous Computing", *Communications of the ACM* (43)5

Alberts, B. et al. (1998), *Essential Cell Biology*, NewYork: Garland Publishing.

Alexander, Christopher. (1964). *Notes on the Synthesis of Form*, Boston: Harvard University Press

Altman R. et al. (1999). "Middleware: The Glue for Modern Applications", *AIMS Strategic Analysis Report*, R-08-2601, Gartner Group, July 26, 1999

Barabasi, Albert-Laszlo and Z. N. Oltvai. (2004). "Network Biology: Understanding the Cell's Functional Organization". *Nature Review – Genetics*. Volume 5, February 2004, pp. 101-

114, http://www.nd.edu/ ~networks/PDF/NatureGen_Vol5-Feb04_Barabasi.pdf

Blakeley, B., H. Harris and R. Lewis (1995). *Messaging and Queuing Using the MQI*, New York: McGrawHill.

Bozinovska, N., G..Jovancevski, and S. Bozinovsli. (2001). "Biological Cell and Its System Software", http://ii.pmf.ukim.edu.mk/ciit/2001/papers/2Ciit-19.pdf

Brooks, R.A., (2002). "Humanoid Robotics", *Communications of the ACM*, (45), 3, March, pp. 33 –38

C4ISR Architecture Framework V2. (1997). http://www.afcea.org/education/ courses/archfwk2.pdf

Cannon, W. (1939), *The Wisdom of the Body*, New York: W. W. Norton & Company.

Capra, Fritjof. (1996). *The Web of Life*, New York: Anchor Books

Carnegie Collection. (2003). http://nmhm.washingtondc.museum/ collections/ hdac/anatomy.htm

Cisco Internet Ecosystems. (1999). "The Internet Ecosystem: A Business Model for the Internet Economy". Cisco Systems. http://www.cisco.com/warp/ public/750/indicator/index.html

Curtin, Matt. (1998). "Write Once, Run Anywhere: Why It Matters?" The Source for Developers, http://java.sun.com/features/1998/01/wora.html

Cliff, Dave. (2003). "Biologically-inspired Computing Approaches to Cognitive Systems: a Partial Tour of the Literature". *HP Technical Reports, HPL-2003-11,* http://www.hpl.hp.com/techreports/2003/HPL-2003-11.html

Comer, D. E. and D. L. Stevens, (1994). *Internetworking with TCP/IP*, Vol. III, Englewood Cliffs, NJ: Prentice-Hall.

Craig, G. J. and D. Baucum. (2001). *Human Development, 9th edition*, Upper Saddle River, NJ: Prentice Hall.

DoD (2003). "DoD Architecture Framework, Volume 1". http://www.defenselink. mil/nii/doc/DoDAF_v1_Volume_I.pdf

Dawson, William F. (2003). "Homeland Security Information Sharing Architecture", http://www.dtic.mil/ndia/2003interop/Lunch.pdf

DBE. (2001). "The Digital Business Ecosystem".http://www.ee.ic.ac.uk/philippe/ dbe_summary_cc.pdf

Diorio, C., D. Hsu and M. Figueroa. (2002) "Adaptive CMOS: From biological Inspiration to Systems-on-a-Chip", *Proceedings of the IEEE*, 90(3).

Farah-Stapleton et al. (2004). "Proposing a C4ISR Architecture Methodology for Homeland Security".http://www.dodccrp.org/events/2004/CCRTS_San_Diego/CD/papers/220%20.pdf.

Flake, G. William, et al. (2004) "Self-Organization and Identification of Web Communities", http://webselforganization.com/

Fox, S. I. (1996). *Human Physiology*. Boston: Wm.C. Brown Publishers.

GAO (2004). "Defense Acquisitions The Army's Future Combat Systems' Features, Risks and Alternatives", GAO-04-635T.

GAO Homeland Security (2003). "Information Sharing Responsibilities, Challenges and Key Management Issues", GAO-03-715T.

Geisendorf, Sylvie. (1999). "Genetic Algorithms in Resource Economic Models", http://www.santafe.edu/research/publications/workingpapers/99-08-058.pdf

Grobstein, Paul. (2004). Genes, Environments, and Individual Choice' http://serendip.brynmawr.edu/gen_beh/Lett-NYT-12-94.html

Hebb, D. (1949) *The Organization of Behavior: A Neuropsychological Theory*. New York: Wiley.

Hewitt, Peter, (2003). “Mind and Brain”, http://www.wordless.com/CGI/ article.asp?ArticleId=29

HGP (2003). “The Human Genome Project”. http://www.ornl.gov/sci/ techresources/ Human_Genome/home.shtml

Hickman J. J., F. Darema, and W. R.Adrion, (2000). “Biological Computation: How does biology do information technology?” *Biological Computation NSF Workshop*.September 21, 2000, Washington D.C.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems*, *2nd edition*, Cambridge, MA: MIT Press.

Hunter, L. (2004). *Molecular Biology for Computer Scientists in Artificial Intelligence and Molecular Biology*. http://www.biosino.org/mirror/www.aaai.org/Press/Books/Hunter/ hunter-contents.html

Inasiti M. and R. Levien (2004). Strategy as Ecology, Harvard Business Review, Vol. 82, Issue 3, March 2004

Iberall, T., and M.A. Arbib (1990), “Schemas for the Control of Hand Movements: An Essay on Cortical Localization”, in M.A. Goodale,(ed.) *Vision and Action: The Control of Grasping*), NJ: Ablex Publishing Corporation.

Iyer, Bala et al. (2003). “Web Services: Enabling Dynamic Business Network”, *Communications of the AIS*, Vol. 11, pp.525-554.

Johnston, J. (2004). “Vibrant Cells: Cellular Automate, Artificial Life, Autopoiesis”. www.stanford.edu/dept/HPST/WritingScience/ML_chap3.pdf.

Jog, N. (2002). “The Epistemology of Digital Organisms”. www.mindspring.com/ ~ninad/alife/Epistemology.htm

Kandel, E. (2000). “The Biological Basis for Learning and Memory”, *HHMI Bulletin*. (3)3.

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

Kimball, J. (2003). “Embryonic Development”. http://users.rcn.com/jkimball.ma.ultranet/ BiologyPages/E/EmbryonicDevelopment.html.

Knapp, Kenneth et al. (2003). “Defense Mechanism of Biological Cells: A Framework for Network Security Thinking”, *Communications of the AIS*, Volume, Volume 12, Article 47, pp. 701-719

Knight, Tom. (1997). “Cellular Gate Technology”. http://willware.net:8080/ knight.html

Koza, J. R., et al. (2000), “Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming”, *Genetic Programming and Evolvable Machines*, (1) 121-164, Netherlands: Kluwer Academic Publishers

Kresh, J. Y., I. Izrailtyan, and S. Wechsler. (2003). “The Heart is a Complex Adaptive System”. www.interjournal.org

Langton, C. G. Editor (1995). *Artificial Life: An Overview*, Cambridge, MA:MIT Press.

Lahoz-Beltra, R and V. Di Paola (2002). “Towards a Computational View of Cell: Do Cells Bear a Resemblance with Computers?” http://www.mdpi.net/ papers/fis2002/121/LahozDOC.htm

Larson, Eric V. and John V. Peters. (2001). Preparing the US Army for Homeland Security. http://www.rand.org/publications/MR/MR1251/

Levy, S. (1992). “Artificial Life: A Report From The Frontier Where Computers Meet Biology”. New York: Vintage.

Linthicum, D. (2001). *B2B Application Integration: e-Business-enable Your Enterprise*, Boston: Addison-Wesley.

Lorentz, E. (1993). *The Essence of Chaos*, Seattle, WA: University of Washington Press.

Mandelbrot, B. (1983). *The Fractal Geometry of Nature*. New York: Freeman

Marchal, P. (1998). “John von Neumann: The Founding Father of Artificial Life, Artificial Life”, (4)3, Cambridge, MA: *MIT Press Journal*.

Marcus, G. (2004).*The Birth of the Mind: How a Tiny Number of Genes Creates The Complexities of Human Thought*. New York: Basic Books.

Maturana H. and F. Varela. (1980). “Autopoiesis: The Organization of the Living”**.** In *Autopoiesis and Cognition: The Realization of the Living.* Volume 42. The Netherlands: D. Reidel Publishing Company, pages 59-138

McCarthy, J. (2003) “What Is Artificial Intelligence, http://www-ormal.stanford.edu/jmc

McCullough, W. S. and W. H. Pitts (1943) “A Logical Calculus of the Ideas Immanent in Nervous Activity”. *Bulletin of Mathematical Biophysics*, (5)115-133.

Minsky, Marvin. (1994), “Will Robots Inherit the Earth?” *Scientific American*, (274)4 October

Mitchell, M. (1999). “Evolutionary Computation: An Overview”, *Annual Review of Ecology and Systematics*, (20) pp. 593-616

Mitchell, M. (2000). *Life and Evolution in Computers*, Santa Fe, NM: Santa Fe Institute.

Moore, J. (1993), “Predators and Prey: A New Ecology of Competition”, *Harvard Business Review.* May-June 1993, pp.75-86.

Moore, J. (1996), *The Death of Competition*, New York: Harper Collins.

Morris, C. G. and A. A. Maisto. (2002). *Understanding Psychology, 6th*. Upper Saddle River, NJ: Pearson Education.

NSF 03-600 (2003). National Science Foundation, *AI & Cognitive Science Program Announcement 03-600*.

NSF 01-102 (2001). National Science Foundation, *BITS 01-102 Program Announcement 01-102.*

Nguyen, T. N. (2003). “An Embryonic Approach to Object-Oriented Software Development”, *Proceedings of 34th Decision Sciences Institute Conference*, Washington D.C. November

Nguyen, T. N. and H. E. Stephanou. (1992). “A Topological Model for Robot Prehension”. *IBM Journal Research and Development*. Volume 36, no. 3.

Nicolis, G. and I. Prigogine (1989). *Exploring Complexity*, New York: W. H. Freeman.

Nuland, Sherwin. (1997). *The Wisdom of The Body*. NewYork: Knopf.

Paton, R. C., H. S. Nwana, M. J. R. Shave & T. J. M. Bench-Capon. (2003) “Computing at the Tissue/Organ Level”, www.csc.liv.ac.uk/~biocomp/reports/tissue_comp/tissue.html

Petrucci, R. and W. Harwood (1993) *General Chemistry: Principles and Modern Applications, 6th edition*, New York: MacMillan Publishing Co.

Prigogine, I. S. (1984). *Order out of Chaos*. New York: Bantam

Ray, T. S. (1993). “An Evolutionary Approach to Synthetic Biology, Zen and The Art of Creating Life”, http://darplace.tomrad.ro/file.PDF

Sajda, P. (2002). “Neural Network”, *Encyclopedia of the Human Brain*, (1) San Diego, CA:

Academic Press.

Shafer, Glenn. (1976). *A Mathematical Theory of Evidence*, Princeton, NJ: Princeton University Press.

Scheibel, A. B. (2004). “Embryonic Development of the Human Brain. New Horizons for Learning”. http://www.newhorizons.org/neuro/shceibel.htm.

Schenk, Donald (2000). Equiping the Brigade Combat Team, http://www.dtic.mil/ndia/cannon/schenk.pdf

Shotwell, A. (2001). “Biology in Computer System”, online, http://ivytech7.cc.in.us/mathsci/alife/computers.PDF.

SoSCOE (2003). About SOSCOE. http://www.boeing.com/ids/soscoe/about.htm,

http://www.army.mil/fcs/factfiles/network-soscoe.html

Sobel, C. P. (2001). “The Cognitive Sciences: An Interdisciplinary Approach”, Mountain View, CA*: MayField Publication*.

Solomon, E., et al.(1996). *Biology*, fort Worth, TX: Saunders College Publishing.

Turing, A. (1953.). “Computing Machinery and Intelligence”,www.ecs.soton.ac.uk/~harnad/Papers/Py104/turing.html

Visible embryo. (2003) www.visembryo.com/baby/index.html

Watson, J.D. and F. H. C. Crick (1953). Molecular Structure of Nucleic Acids. *Nature*, vol. 171, pages 737-738

Weiser, M. (1993). “Hot Topics: Ubiquitous Computing”. *IEEE Computer*. Volume 26, No. 19, October.

Wood, William G. and Sholom Cohen. (2003). DoD Experience with the C4ISR Architecture Framework, *Technical Note* CMU/SEI-2003-TN-027, http://www.sei.cmu.edu/publications/documents/03.reports/03tn027.html?si

Von Burtalanffy, L. (1969), *General System Theory*, New York: George Braziller

Von Neumann, J. (1958) *The Computer and The Brain*, New Haven, CT: Yale University Press.

Woffram, S. (2002). *A New Kind of Science*. Wolfram Media, http://www.wolframscience.com/

Zadeh, Lofti. (1988). Fuzzy Logic. *Computer*. (21) 4, pp. 83-93

Zajicek, G. (2003). “WOB as a Non-Linear Massively Parallel Biological Computer”, www.what-is-cancer.com/papers/straming/wobcomputer.htm.

ABOUT THE AUTHOR

Thang N. Nguyen teaches undergraduate and graduate Information Systems Courses at California State University Long Beach. He earned a BS in Electrical Engineering from Laval University, Quebec, P.Q. Canada in 1966, an MS in Information and Computer Science from Georgia Institute of Technology in 1973, and a Ph.D. in Information Technology and Engineering from George Mason University in 1990. He worked for Candle Corporation, IBM Corporation, Litton Computer Services and other firms in various capacities in software development, IT management and education/training. His research interests include intelligent robotics, intelligent and autonomous software, symbolic-numeric integration, biological computing, e-business application integration and business-IT integration.