# MACHINE BIOLOGICAL CLOCK
## *The Time Dimension in a Organic-Based Operating System*

Mauro Marcelo Mattos
Computing Systems Department, FURB- University of Blumenau, R.Braz Wanka 238, Brazil
Blumenau, Santa Catarina, Brazil

Keywords: Organic Computing, Machine Biological Clock, Knowledge-based Operating System.

Abstract: A Knowledge-Based Operating System is an embodied, situated, adaptive and autonomic system based on knowledge abstraction which has identity and intelligent behavior when executed. We have identified three dimensions over which such a new operating system paradigm has to be based: (i) physical dimension, (ii) logical dimension, and (iii) temporal dimension. The physical dimension describes the physical components and their structural relationship. The logical dimension describes the functional characteristics of each physical component and the time dimension is provided in order to enable the entire system to perceive the time flow – as a biological clock in human beings. This work presents the Machine Biological Clock concept.

## 1 INTRODUCTION

"*Twenty years ago, on November 12, 1986, Fred Turek, Dave Hudson, Joe Takahashi, and Gene Block founded the Society for Research on Biological Rhythms. Who would have imagined that today we are analyzing the structural biology of bacterial clock proteins? Or that we are performing real-time measurements of rhythmic gene expression within individual cells in a dish? Or that we are delineating the neurobiology of a neurotransmitter that underlies narcolepsy? And that's just for starters. Every other year, we have been gathering at this meeting to announce, debate, and celebrate these advances, in presentations that cut across*." (Schwartz, 2006).

In his effort to describe time as a subjective experience in terms of his philosophy of Phenomenology, Husserl grapples with the succession of perceived moments of conscious existence by using the example of the melody to illustrate his point. In a melody, each successive note exists in time for a moment, and each moment represents a new 'now' point. Each part of the melody is related to its antecedent and to the notes yet to come. Our apprehension of the melody cannot be obtained outside the context of successive notes. "Therefore, the perceiving of a melody is in fact a temporally extended, gradually and continuously unfolding act, which is constantly an act of perceiving." (Husserl, 1893-1917). This temporality is a subjective construct. (Craft, 2000).

According to Stulp and Beetz (2002), agent-based systems are solving more and more complex tasks in increasingly challenging domains, the systems themselves are becoming more complex too, often compromising their adaptivity and robustness.

In robotics projects, system designers cannot foresee each situation, or all the possible outcomes of an action. Therefore, hand-coding the controllers is considered intractable and failure-prone task.

This situation is a reality in general purpose computational systems. According to Hayes-Roth (2006), "the best systems of our times have been hand-crafted by great engineers. These system makers have analyzed the task environments, knowledge requirements, and reasoning skills necessary for successful applications. This approach can work for any well-defined and sufficiently narrow task. But, if the system fails, the engineers would diagnose and debug the errors. They would determine what knowledge to add or modify, how to program it, and how to modify and rebalance the pre-existing programs to accommodate the new performance without harming the parts that already worked well. Automation in adaptation, learning, and knowledge acquisition was very limited – a tiny fraction of the overall knowledge required, which the engineers mostly prepared manually. We have



Marcelo Mattos M. (2008).
MACHINE BIOLOGICAL CLOCK - The Time Dimension in a Organic-Based Operating System.
In *Proceedings of the First International Conference on Biomedical Electronics and Devices*, pages 76-83
DOI: 10.5220/0001046800760083

not yet figured out how to make the systems responsible for their own debugging and improvement".

A promising approach to solve this problem is to provide agents with reflective capabilities. Agents that can reflect on the effects and expected performance of their actions are more aware and knowledgeable of their capabilities and shortcomings. This is called "action awareness" (Stulp and Beetz, 2002).

Another approach is based on an "efficient thought" concept (Hayes-Roth, 2006). This concept lists eight steps that the most complex organizations, in general, perform in parallel. This approach states that the intelligent being (a) observes what's happening in the environment, (b) assesses the situation for significant threats and opportunities, (c) determines what changes would be desirable, (d) generates possible plans to operate those changes, (e) projects the likely outcomes of those plans, (f) selects the best plan, and (f) communicates that plan to key parties before implementing it. Throughout the process, the intelligent being (g) validates and improves its model.

The new research area that intends to solve some of the problems pointed out is called Organic Computing (OC). The main goal of OC is the technical usage of principles observed in natural systems. (Muller-Schloer, 2004).

In this paper, we introduce a computational model for what we call "machine biological clock" in order to make possible to build a really reflexive environment where the system can perceive the time flow. This concept, that has been ignored in previous works is, in our point of view, the most important concept that has to be considered if we plan to build intelligent systems. Or, as said in Brachman (2002) 0, if "we want to transform them from systems that simply react to inputs into systems that are truly, in a word, cognitive. Most formal and intuitive definitions tell us that cognition is about knowing. Our image of a cognitive system, then, is one that can indeed know things and act on that knowledge. It can take explicit knowledge gleaned in a host of ways and go beyond it to important implicit knowledge, ranging from pure and simple logical deduction to what we might call 'plausible reasoning' ".

The paper is organized as follows: a problem's contextualization in operating system area is presented in section 2; a knowledge-based operating system concept in section 3; an overview of the biological clock concept in section 4; section 5 describes the time dimension in a KBOS system; section 6 presents some related works and in the conclusion section the final comments are presented.

## 2 WHY A NEW CONCEPT?

In our point of view, three concepts contribute to reduce our possibilities in building really intelligent systems: (i) the multitasking concept, (ii) the operator concept and, (iii) the program concept.

Traditional operating systems support the notion of a hardware abstraction level in which each application is supposed to possess its own processor (and other resources). This situation and the fact that, in general, all commercial operating systems are based on a multitasking concept (introduced in 1964), contribute to the permanence of problems identified 30 years ago (Linde, 1975). The problems range from security to usability, including lack of adequate behavior in fluctuating execution conditions and user's privacy (Brachman, 2002).

Today we are also faced by new demands like pervasive computing and organic computing, where self-adaptation and self-reconfiguring are the main goals.

Besides that, there are two other concepts that contribute to make things worse: (i) the operator concept, and (ii) the program concept.

The operator function was necessary during the first years of computing since computers were big and difficult to use. Operators, at that time, were responsible for turning the machine on/off, starting programs, collecting reports, restarting programs and so on. This scenario has changed as the computers become smaller, cheaper and faster as they are today. However, what was a real need in the past is employed today as if there was no other way to interact with computers. In fact, we are nowadays operators – all of us using some kind of computer (desktops, palmtops, and mobile phones). We are trained today to learn how to pull virtual buttons the same way the former operators were trained to pull real buttons in real panels on those old mainframes.

This aspect has consequences and the program concept is the main one. A program could be thought as the programmer's hands virtually extended inside our machines. The programmer has the knowledge about some specific domain and knows how to establish the correct sequence of steps in order to solve the problem. In this scenario, we are users of such routines – in other words: operators. Programs, within this context, are the way through which programmers can implement their procedural knowledge about the problem's domain.

This model does not enable the actual operating system to acquire knowledge about what is happening inside the machine. We believe that this is one of those several sources of problems that we experience today. So, we have proposed a new model.

# 3 A KNOWLEDGE-BASED OS

The novel concept introduced in Mattos (2003) says that a knowledge-based operating system is: "an embodied, situated, adaptive and autonomic system based on knowledge abstraction which has identity and intelligent behavior when executed".

The whole system is built inside a shell which gives the endogenous characteristic. A hyper dimensional world model (Mattos, 2005) enables the entire system to perceive evolving and/or fluctuating execution conditions (fig. 1).

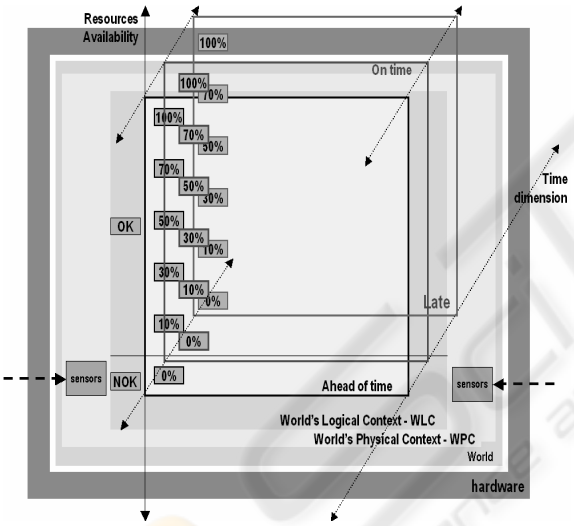


Figure 1: Hyper-dimensional World Model.

## 3.1 The Dimensions

We have identified 3 dimensions over which such a new operating system paradigm has to be based: (i) Physical dimension, (ii) Logical dimension, and (iii) Temporal dimension.

The physical dimension describes the physical components and their structural relationship.

The logical dimension describes the functional characteristics of each physical component. It is called: physical context of a device. A state machine describes the dynamic aspects of the component's behavior. Merging the entire physical context of all physical devices described at the physical dimension, we obtain the world's physical context (WPC). The logical dimension is provided by a DEVS run-time environment (Mattos, 2005b).

The Time Dimension is provided in order to enable the entire system to perceive the time flow – as a biological clock in human beings.

## 3.2 Knowledge based OS and Knowledge Acquisition

Some works (Samsonnet et al, 1982; Vilensky, Arens and Chin, 1984; Chikayama, 1988; Genera 2003; Larner, 1990; Yokote, 1992; Li et al, 1995; Patki, Raghunathan and Khurshid, 1997) have been described as aiming to develop a complete knowledge-based operating system. Other approaches consists of applying IA techniques through making kernel implants (Seltzer, Small and Smith, 1995) in order to get better user interfaces in traditional operating systems (Pasquale 1988; Cockcroft,1995; Hernández, Vivancos and Botti 1998; Zomaya, Clements and Olariu, 1998; Kandel, Zhang and Henne, 1998; Siraj, Bridges e Vaughn, 2001). However, all of them have failed into achieving their objectives because the main concept over which they should base their work has not been clearly specified – "what" is the knowledge. This aspect has transformed those supposed new operating systems projects into traditional operating systems architectures with many specialized libraries over some multitasking platform.

Knowledge in this context is conceived as being a set of logical-algebraic operational structures that makes possible to organize the system functioning according to interconnection laws and behavior laws. It is well known that a significant obstacle to the construction of knowledge-based systems is the process of knowledge acquisition.

The key to this process is how we may effectively acquire the knowledge that will be implemented in the knowledge base. In an operating system environment, this is not an easy task. It is usually done by hooking the OS API calls and recording logs for further analysis. This is a time and resources consuming process. The main drawbacks to this approach are: (i) the information gathering process impacts the overall performance, influencing other applications that aren't involved in the application context being considered; (ii) this impact on performance also interferes with the application being considered; and (iii) this scenario is probably different from that of where the application was developed.

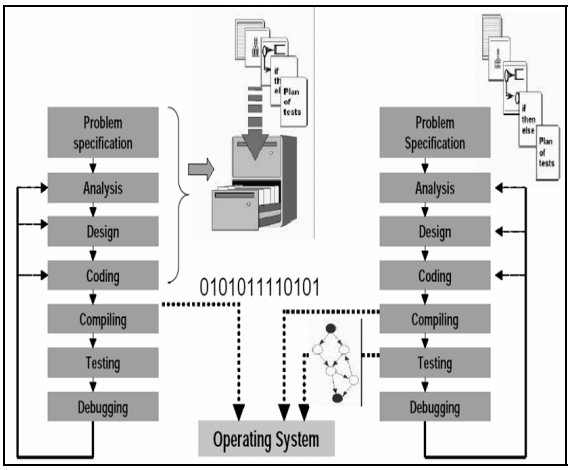


Figure 2: (a) General framework (b) Proposed framework.

### 3.2.1 Learning Phase

Figure 2a shows the traditional application life-cycle characterizing that the documentation produced during the analysis and design, in general, is stored in folders after the implementation phase is concluded. The traditionals operating systems usually receives only binary code to manage.

In our approach, the development process must be oriented by the application dynamics. As a prerequisite, a KBOS environment establishes that each application developed must have (i) a state-machine that describes all dynamic behaviour and (ii) the source-code associated (Figure 2b).

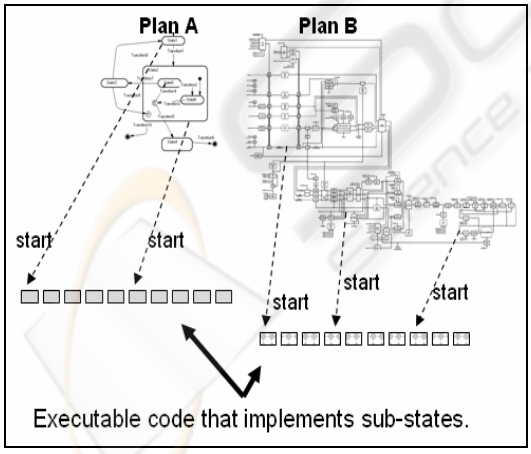


Figure 3: Plans registering relative starting time for each sub-plans or sub-states.

This is the input for our system to learn about the application's intentions and to generate the executable code. When the KBOS runtime environment receives a new application state-machine, it starts a procedure to convert sub-states and portions of source-code into execution plans. Those execution plans are built in a parallel functional decision trees format and constitutes the executable code that KBOS recognizes.

The knowledge at OS level emerges from a library of execution plans and from the system's experience in running those plans.

Figure 3 shows an example with two plans A and B. Each plan has its own sub-state architecture (according to their particular purpose). Each sub-plan is implemented by a set of machine instructions (executable code). The time needed to execute each sub-state is determined during the software test phases by the development team and delivered with the software (including the state machines – dynamic model of the software).

## 4 BIOLOGICAL CLOCK

According to Siegel (2006), living organisms evolved an internal biological clock, called the circadian rhythm, to help their bodies adapt themselves to the daily cycle of day and night (light and dark) as the Earth rotates every 24 hours. The term 'circadian' comes from the Latin words for about (circa) a day (diem). Circadian rhythms are controlled by "clock genes" that carry the genetic instructions to produce proteins. The levels of these proteins rise and fall in rhythmic patterns. These oscillating biochemical signals control various functions, including when we sleep and rest, and when we are awake and active. Circadian rhythms also control body temperature, heart activity, hormone secretion, blood pressure, oxygen consumption, metabolism and many other functions.

A biological clock has three parts: a way to receive light, temperature or other input from the environment to set the clock; the clock itself, which is a chemical timekeeping mechanism; and genes that help the clock control the activity of other genes. (Siegel, 2006)

In the last few decades, scientists have discovered the genes responsible for running the internal clocks: period (per), clock (clk), cycle (cyc), timeless (tim), frequency (frq), double-time (dbt) and others. Genes that control circadian rhythms have been found in organisms ranging from people to mice, fish, fruit flies, plants, molds and even single-celled, blue-green algae known as cyanobacteria. (Siegel, 2006)

The master circadian clock that regulates 24-hour cycles throughout our bodies is found in a region

called the suprachiasmatic nuclei (SCN) in the hypothalamus of the brain. The SCN is made up of two tiny clusters of several thousand nerve cells that "tell the time" based on external cues, such as light and darkness. The SCN regulates sleep, metabolism, and hormone production (Siegel, 2006).

The SCN is believed to synchronize "local" clocks in organs and tissues throughout the body, either through hormones or changes in body temperature. Gene-operated clocks independent from the brain's master pacemaker have been found in the liver, lung, testis, connective tissue and muscle (Siegel, 2006).

### 4.1 Perception of Time

People and other animals are able to perceive the duration of intervals between events, and the accuracy of their perceptions can be assessed. In situations in which there are many different time intervals, these can be combined for the assessment of the typical interval. Associative learning is dependent upon time perception, and the mechanisms of time perception involve an internal clock (Church, 2006).

According to Craft (2000), “we take it for granted that each experience takes place in a continuum of time. In the course of the daily routine one rarely, if ever, stops to notice as such the passing of physical and psychological events while they happen. But each passing moment, noticed or unnoticed, is a successive instance of the ‘now’ that marches on into a presumably infinite past that we construct in memory. By extension, we anticipate a succession of future ‘now’ moments that have yet to come. In this process we meld our “experience of heterogeneous events into a coherent sense of persistence” (Flaherty, 1999 apud Craft, 2000).

It has been physically hypothesized that biological processes and environmental factors provide cues through which we construct our perception of temporality. It has been suggested that there is some sort of “biological clock” that regulates our perception of time. This is not to be confused with the more commonly known “circadian” rhythms which have been usually observed in most living things. Flowering plants, bees and other animals demonstrate the existence of internal or innate timing mechanisms in their blossoming and movement patterns. Although much attention has been given to these patterns in human subjects, more notably in the isolation studies of Aschoff and Weaver, such circadian clocks are not likely to be involved in human time perception on the fine scale of minutes and seconds (Campbell, 1990 apud Craft 2000).

The concept of ordered time, the fixed intervals of minutes and seconds, and the synchronicity between 10 minutes on the clock and the same amount of time in lived duration is, at least partially, a socialized phenomenon (Flaherty, 1991 apud Craft, 2000).

## 5 THE TIME DIMENSION

In the current paradigm of computing systems, the time is a variable that has to be explicitly read in order to enable software entities to perceive the time flow. It implies that if the program does not read that variable, it will not be able to perceive the time flow.

In a KBOS system, the time flow is part of the system and it is implemented in two phases: (i) by establishing units of time flow perception and (ii) by time-stamping the state machine that drives the executable code (in KBOS run-time environment executable code is implemented as parallel functional decision trees) with that information.

The first phase makes possible the creation of a kind of “machine biological clock” (MBC) concept – a time unit that the whole system can perceive as flowing and that is different from the time-slice concept as adopted in multitasking based operating systems today. The second phase enables parts of code to perceive time without the need of explicitly reading clock variables.

Figure 4 shows the situation in which one of the sub-states of plan A at the second MBC unit perceives that its sub-state is delayed in comparison with the original situation presented by the figure 3. Figure 4 also shows that hardware events (time ticks from real-time clock and other hardware events from mouse, hard disk and network card) demand specific plans to be executed (fig. 5).

The MBC makes possible to introduce another concept: the work capacity (WC) of a machine. This concept is related to the amount of work a machine can do in some BMC and introduces the capacity of a system to perceive if it is becoming overloaded or not.

Figure 6a shows an example of a program that is time dependent. In general, the program needs to call the getTime API in order to discover the current time and make some calculation to discover if it is delayed, on time or ahead of time. Also, in general, only some portions of the code running in a system have to deal with such constraints so we have a mix of code dealing with time and code that was not

conceived to deal with time running together on the same environment.

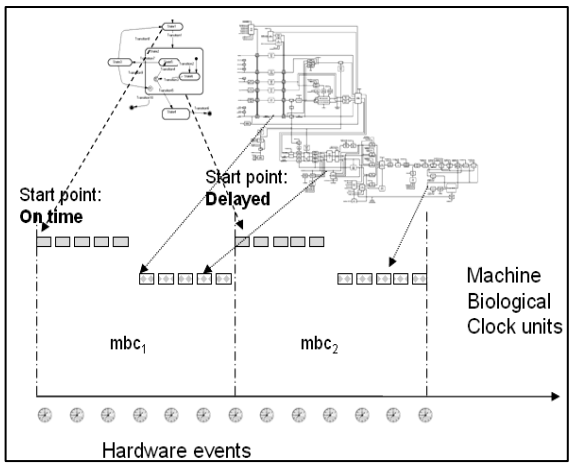


Figure 4: Plans A and B perceiving delays.

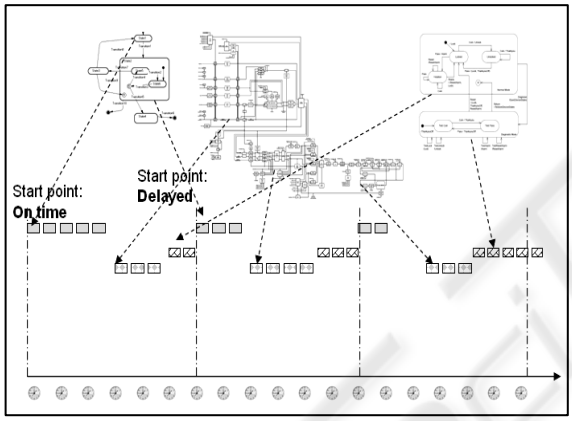


Figure 5: Plans A, B and C perceiving delays.

Figure 6b shows how a plan (in KBOS context) should be developed: each procedure/function has to explicitly declare sections where the time dimension has to be considered as a functional requisite. During the execution time of that plan, the system activates the appropriated section (delayed, onTime or aheadOfTime) according to the situation of that plan. If the developer do not know what to do in some situation, he can explicitly use an IDoNotKnowWhatToDo clause and the KBOS run-time will take appropriate actions.

The time dimension also makes possible to introduce the notion of space concept. In other words, if two different plans perceive that both are delayed it is equivalent to say to each one that there is someone else sharing resources within the same MBC unit. It leads the logical path of some plan to be changed to another path that implements the same functionality but demands less resources. The self-adaptive and self-reconfigurable characteristic of the system is based on this facility.

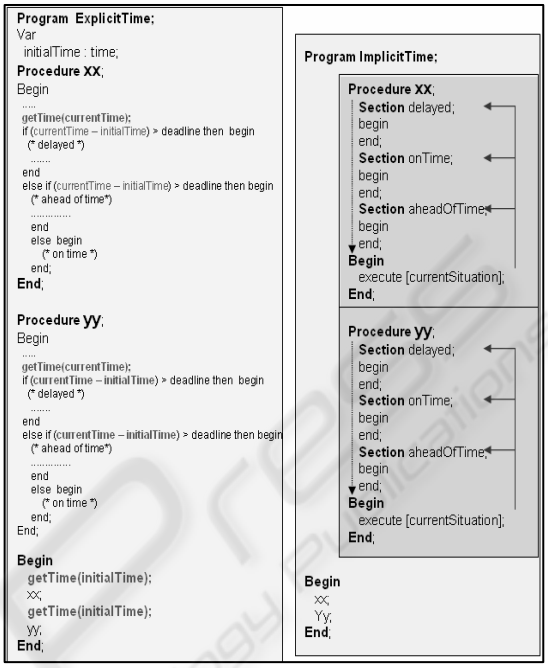


Figure 6: (a) Explicit time (b) implicit time.

## 5.1 Identity and Intelligent Behaviour

A KBOS framework enables to introduce the identity concept, which is resultant from the embodiment, situatedness, adaptiveness and autonomy characteristics. This leads to an emergence concept. According to Muller-Schloer (2004), emergence is defined as a property of a total system which cannot be derived from the simple summation of properties of its constituent subsystems. Emergent phenomena are characterized by (i) the interaction of mostly large numbers of individuals (ii) without central control with the result of (iii) a system behavior, which has not been explicitly "programmed" into the individuals.

In this sense, the set of characteristics enables the system to perceive, in an individualized manner, a set of events occurring in some instant of time. Thus, the intelligent behavior emerges from the previous characteristics plus the relationship between the system and the surrounding environment.

# 6 RELATED WORK

Stulp and Beetz (2006), proposed a novel computational model for the acquisition and application of action awareness, showing that it can be obtained by learning predictive action models from observed experience and also demonstrating how action awareness can be used to optimize, transform and coordinate underspecified plans with highly parametrizable actions in the context of robotic soccer. The system works in two moments: (a) idle time when the agent learns prediction models from the actions in the action library and, (b) during operation time, when action chains are generated.

Tannenbaum (2007) argues that self-awareness means learned behaviors that emerge in organisms whose brains have a sufficiently integrated, complex ability for associative learning and memory. Continual sensory input of information related to the organism causes its brain to learn its (the organism's) physical characteristics, and produce neural pathways, which come to be reinforced,, so that the organism starts recognizing, several features associated to each reinforced pathway. The self-image characteristic provides a mechanistic basis for the rise of the concept of emergency of behavior that, on its turn, is connected to the concepts of self-awareness and self-recognition. On the basis of all that process there is the notion of time perception.

# 7 CONCLUSIONS AND FURTHER WORK

We have briefly given an overview of an endogenous self-adaptive and self-reconfigurable approach to operating system design and introduced the MBC concept. The main aspect to be pointed out is that there is no separation between what is known as operating system and what is known as application programs in today's paradigm.

In a KBOS environment all the executable code takes part into the system and all the code has the ability to perceive the time flow. This perception enables the whole code to execute self-adaptation without explicitly demanding routines.

# REFERENCES


R.J.Brachman. *Systems That Know What They're Doing*. IEEE- Intelligent Systems. Nov/Dec 2002. pp.67-71.

T.Chikayama. *Overview of the Parallel Inference Machine Operating system (PIMOS)*. Proc. Of the Intl. Conference of Fifth Generation Computer Systems. Pp. 230-235. 1988.

R.M.Church. *Time Perception*. Encyclopedia of Cognitive Science. John Wiley & Sons, Ltd. 2006.

A.Cockcroft. *New release of the SE Performance Toolkit*. March 1995. Available in www.sun.com/960301/columns/adrian/ column7.html, March 1995.

B.Craft. *Twitch of the Snooze Button: Time Perception and Cognition in Humans*. Submitted in partial fulfillment of the requirements for HCI 450, DePaul University. January 31, 2000.

Genera Concepts: *The Best Software Environment Available*. March 2003. Available in http://kogs-www.informatik.uni-hamburg.de/~MOELLER/ symbolics-info/GENERA/genera.html.

R.Hayes-Roth. P*uppetry vs. Creationism: Why AI Must Cross the Chasm*. IEEE Intelligent Systems. September/October 2006; 21(5):7-9.

L.Hernández,E.Vivancos,V.Botti. *Intelligent Scheduling of Production Systems in a Real-Time Architecture*. IBERAMIA'98,1998,p429-438.

A.Kandel,Y.Zhang, M.Henne. *On use of fuzzy logic technology in operating systems*. Fuzzy Sets and Systems 99, Elsevier Science, pp 241-251, 1998.

Larner,D.L. A *Distributed, Operating System Based, Blackboard Architecture for Real-Time Control*. CACM. 1990.

R.Linde. *Operating Systems Penetration*. AFIPS Conf. Proceedings, Vol 44, 1975.

M.M.Mattos. *Main Concepts to Build Knowledge-Based Operating Systems*. Doctoral thesis. UFSC-Universidade Federal de Santa Catarina, Brasil, Novembro, 2003. (In Portuguese)

M.M.Mattos. *Next Generation of Operating Systems Design Based on Knowledge Abstraction*. In: Proceedings of the IADIS International Conference on Applied Computing. Algarve, Portugal. 2005.

M.M.Mattos. *KBOS Run-Time Environment Based on DEVS Formalism*. In: Proceedings of the IADIS - International Conference on Applied Computing. Algarve, Portugal. 2005.

C.Müller-Schloer. *Organic computing: on the feasibility of controlled emergence*. In Proceedings of the 2nd IEEE/ACM/IFIP international Conference on Hardware/Software Codesign and System Synthesis (Stockholm, Sweden, September 08 - 10, 2004). CODES+ISSS '04. ACM Press, New York, NY, 2-5. DOI= http://doi.acm.org/10.1145/1016720.1016724.

A.B.Patki , G.V.Raghunathan ,A.Khurshid. *FUZOS—Fuzzy Operating System support for Information Technology*. Proceedings of Second On-line World Conference On Soft Computing In Engineering, Design And Manufacturing. Cranfield University, UK, June 1997.

J.Pasquale. *Using Expert Systems to Manage Distributed Computer Systems*. IEEE Network. Set.1988.

J.P.Sansonnet,M.Castan,C.Percebois,D.Botella,J.Perez. *Direct Execution of LISP on a List Directed Architecture.* Proceedings of ASPLOS. Palo Alto, Califórnia, March 1982, pp. 132-139.

M.Seltzer,C.Small,K.Smith. *The Case for Extensible Operating Systems.* Harward Computer Center for Research in computing Technology -Technical Report TR-16-95 Depto. Of Computer Science, Harward University. 1995.

L.J. Siegel. *The Time of Our Lives*. Learn.Genetics – Genetic Science Learning Center. The University of Utah. In: learn.genetics.utah.edu/features/clockgenes. 2006.

A.Siraj.,S.Bridges,R.Vaughn. *Fuzzy Cognitive Maps for Decision Support in Intrusion Detection Systems*.2001. Available in http://www.sc.msstate.edu/~security/iids/publications/nafips_ifsa_2001.htm.

F. Stulp and M. Beetz. *Action awareness – enabling agents to optimize, transform, and coordinate plans*. In Proceedings of the Fifth International Joint Conference Autonomous Agents and Multiagent Systems (AAMAS), 2006.

W. J. Schwartz. *President's Welcome*. In Proceedings of the Tenth Meeting of Society for Research on Biological Rhythms, Sandestin, FL. May 21–25, 2006.

E.Tannenbaum. *Speculations on the emergence of self-awareness in big-brained organisms*. Jun 2007. Available in http://eprintweb.org/S/article/q-bio/0701017.

R.Vilensky,Y.Arens,D.Chin. *Talking to Unix in English: An Overview of UC*. CACM 17,6,pp.574-593, Junho 1984.

Y.Yokote. *The Apertos Reflective Operating System: The Concept and Its Implementation*. Proceedings of OOPSLA'92, ACM Sigplan Notices, v. 27, pages 414–434, 1992.

A.Zomaya,M.Clements, S.Olariu. *A Framework for Reinforcement-Based Scheduling in Parallel Processor Systems.* IEEE Transactions on Parallel and Distributed Systems. V9, N3,p249-260,Mar 1998.