A HyperDimensional Knowledge-based Operating System Design

Mauro Marcelo Mattos,Dr. (mattos@furb.br**)**

1. The problem

Traditional operating systems support the notion of 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 multitasking concept (introduced in 1964), contribute to enforce problems identified 30 years ago [1]. This problems range from security to usability, including lack of adequate behavior in fluctuating execution conditions and user's privacy. However, today we are faced with new demands like pervasive computing, teleworking and e-commerce 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 today operators – all of us using some kind of computer (desktops, palmtops, 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 in this context are the way that 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. This is one of those many sources of problems that we experience today.

2. Our solution – A new kind of knowledge-based OS

This work presents a short overview of an endogenous self-adaptive and self-reconfigurable approach to operating system design that is being built to improve the system's security and privacy based on (i) a hyper dimensional world model, (ii) on DEVS formalism as a runtime environment and, (iii) uses the concept of plans instead of programs.

The first step in our work was to define what is a knowledge-based operating system[2]: "an embodied, situated, adaptive and autonomic system based on knowledge abstraction which have identity and intelligent behavior when executed". 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.

The identity aspect is resulting from the embodiment, situatedness, adaptiveness and autonomy characteristics. Those characteristics enable the system to perceive in an individualized manner a set of events occurring in some instant of time. The intelligent behavior is resulting from the previous characteristics plus the relationship the system with the surrounding environment. The last important aspect to be considered is that all this "intelligent behavior" is possible just when the system is running. This means that if (or when) the system is turned off and further turned on again, we are faced with the symbol-grounding problem (update the surrounding characteristics in order to present intelligent behavior in this "new possible world").

The whole system is built inside a shell (endogenous characteristic) - outside is the real world. A hyperdimensional world model enables the entire system to perceive evolving and/or fluctuating execution conditions. We have identified 3 dimensions over which such a new operating system paradigm has to be based on: (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. Joining the entire physical context of all physical devices described at the physical dimension, we obtain the world's physical context (WPC). One important aspect is that changes in availability in one logical device affects the whole system and this situation is immediately sensed by all logical devices of the world model in such a way that any fluctuation in resources availability will be noticed by all applications running in the system (including the OS). The logical dimension is provided by a DEVS run-time environment and the Temporal Dimension is provided in order to enable the entire system to perceive the time flow – as a human being.

3. Conclusions and further work

We have briefly described an overview of an endogenous self-adaptive and self-reconfigurable approach to operating system design. Considering that a knowledge-based operating system has some kind of identity property, it is possible to promote the entire system to an agent category. Being an agent, the theory of Organizations in Multiagent Systems (MAS) can be adopted in order to make possible agents interact with other agents living in the surroundings. In this sense, any classical model used to describe or project an organization (agent centered, or organization centered) can be adopted to enable the agents cooperate to improve the efficiency of some particular solution that (i) one doesn't have enough resources (or doesn't enough knowledge) to solve, or (ii) besides having enough knowledge and/or resources, it knows that its neighbors have the possibility to solve the problem in a faster way.

4. References

[1] R.Linde. Operating Systems Penetration. AFIPS Conf. Proceedings, Vol 44,1975.

[2] M.M.Mattos. Fundamentos conceituais para a construção de sistemas operacionais baseados em conhecimento. Tese de Doutorado. UFSC- Universidade Federal de Santa Catarina, Brasil, Novembro, 2003.