1

Information and Software Systems: from Architecture to Urbanism

Desreumaux, Marc
EDF Production Transport
Information Systems Department
1, Place Pleyel
F93207 Saint-Denis, France
Phone: +33-1-43 69 35 59
marc.desreumaux@edfgdf.fr

Oudrhiri, Radouane
3H Technology
3030 Clarendon Blvd, Suite 320,
Arlington, VA 22201, USA
phone: +1-(703) 90 80 844
fax: +1-(703) 90 80 845
roudrhiri@threeht.com

Abstract

Structured architecture would improve the systems quality by establishing discipline, but does not account of their evolution. We need to go beyond by discovering and using the hidden laws of urbanism that are intimately tied to autonomy, adaptation and evolution notions.

Keywords

Urbanism, architecture, interfaces, adaptative systems, evolving systems

1. INTRODUCTION

The speed of adaptation and the control of evolution of software systems are primary determinants for the organizations' competitiveness. Unfortunately, information and software systems are neither adaptable, nor evolving. The software meets —or does not meet—precise requirements. Hence, we attend to a logic of "everything" or "nothing".

Software is never an isolated system. It is constrained by its users, the other systems that interoperate with and the infrastructure on which it runs. It is not protected from changes. **Software systems are built on the constraints of the past, in order to be used in the present, and generate constraints for the future.**

Will it be possible to e**ase** the **desirable** changes and to **avoid** the **undesirable** and **unexpected** ones?

2. FROM INTUITIVE TO STRUCTURED ARCHITECTURE

Classically, "architecture" designates the systems' composition, their organization and the process of organizing; the art (process), the result (deliverable) and the style.

Building an architecture is still an *ad hoc* process and not yet a science, based on the architect's intuition and experience. The characteristic much sought-after is the **efficiency** with multiple facets: universality of the utilization field, performance, ease of use, cost, etc.

A first improvement consists in introducing a **structured architecture** that defines categories of components and their typical assembling rules. The benefits of this improvement are equivalent to the benefits gained by the shift from intuitive to structured programming: more **discipline** in the process (Dijkstra 1976).

However, **the structured architecture foresees only few on the evolveability of the system**. Currently, the architecture process becomes equivalent to an assembling game based on "constructibles": e.g., Lego® toys. The basic components are real or virtual machines, where the majority of them are equivalent to a von Neumann machine.

There are three fundamental composition laws. (Bass, 1998) provides a set of composition laws called Unit Operations. For us, the three basic ones are:

- The **Nesting** operation: stacking a machine on another one. The stacked machine becomes a sort of language, interpreted by the other one. Concrete examples are Windows® stacked upon DOS®; stacked on the machine, stacked on the CU; etc.
- **Memory sharing**. The same data shared among multiple machines.
- **Communication**. Typically reduced to the message passing.

Each architectural construction may be described by the basic components and the assembling operators (nesting, memory sharing and message passing). The assembling of architectural constructions gives rise to a new architectural construction.

Communication plays an important role for the system flexibility. It is the basis of system's articulations. **The higher the flexibility of the articulations, the more adaptable the system becomes**. Unfortunately, this is not the case of our current systems.

Fundamentally, the structural approach does not change the systems' ability to evolve. The obtained systems are very structural. The majority of existing information systems are **monolithic** and **centralized.** In spite of the system's parallelism, the system behaves sequentially. For example, the current PCs have parallel processors but the software and OS are still sequential; existing information systems run thousands of inter-connected machines, but most of processes (essentially administrative ones) are performed sequentially.

The effective computability has not changed. May be we gained in terms of instantaneous performance, but not in terms of effective computability. This

structural approach assumes a predefined modeling of the real world. Thus it suffers from a **lack of adaptation** and **evolution**.

3. PUTTING MORE RIGOR IN THE INTERFACES

4.1 Autonomous systems

Autonomy is the chief property of architectural components interacting within an **adaptable** information system. Autonomy is a property assigned normally to biological and human systems (Varela 1979) (Lorigny 1992). It is considered here —schematically— as the ability to continue fulfilling its functions and to hold out within a changing environment.

In order to be autonomous, a system must have a **boundary**, which delimits and separates its inside from the outside. It is also an area of **exchanges** between the inside and the environment. The autonomous system owns and maintains internal **resources** in **reserve**. These resources are consumed or exchanged by the system with its environment, allowing it to fulfill its functions, and to adapt itself to the environment hazards.

From the informational standpoint, a software component exchanges resources that are **data**. The boundary that isolates it from the environment and which allows it to perform exchanges is the **presentation** layer. The mechanism that transforms, somewhat, the internal data into released information to outside on one hand, and that controls the information coming from outside and transforms them into internal resources, on the other hand, is the **processing** layer.

The structure of an autonomous component presents an analogy with the structure of a living cell.

4.2 Communication between autonomous systems

Communications between the autonomous components correspond to the exchanges of information that are performed through mutual presentations. The **presentation** of information is characterized by:

- the information content (the series of significant digits),
- its syntactic structure,
- its morphologic code (the way each digit is coded, generally depending on the medium),
- the support (medium),
- the rhythm, or more generally the periods of time within the component sends or receives.

The physical transport along the medium within space and time, has effect of transferring informational content when it is “aware” of its morpho-syntactic structure. Of course, this transfer of informational content handled by media has the objective of transferring the **significance**.

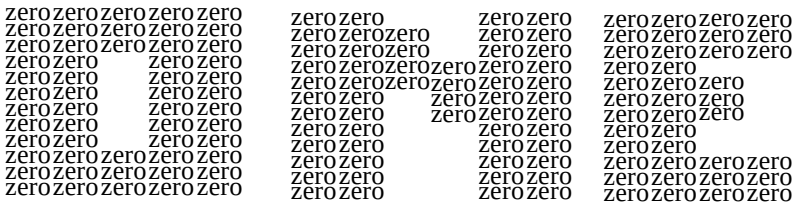

Figure 1: A message does not transport only a single significance, but several ones that are interpreted by the corespondents relatively to their scales of coding-decoding.

Communication between autonomous systems present a strong **nested** and quasi-**fractal** nature (Nottale, 1993). The Figure 2 hereafter, presents an example of dialogues between two corespondents. The context of the first correspondent is a paper manufacturer who wants to order 100 paper pencils, at the unit price of $6, taxes included. The second corespondent is a cosmetic salesman, selling make-up pencils at a unit price of $30, taxes excluded, and receives a purchase order of 100 pencils.

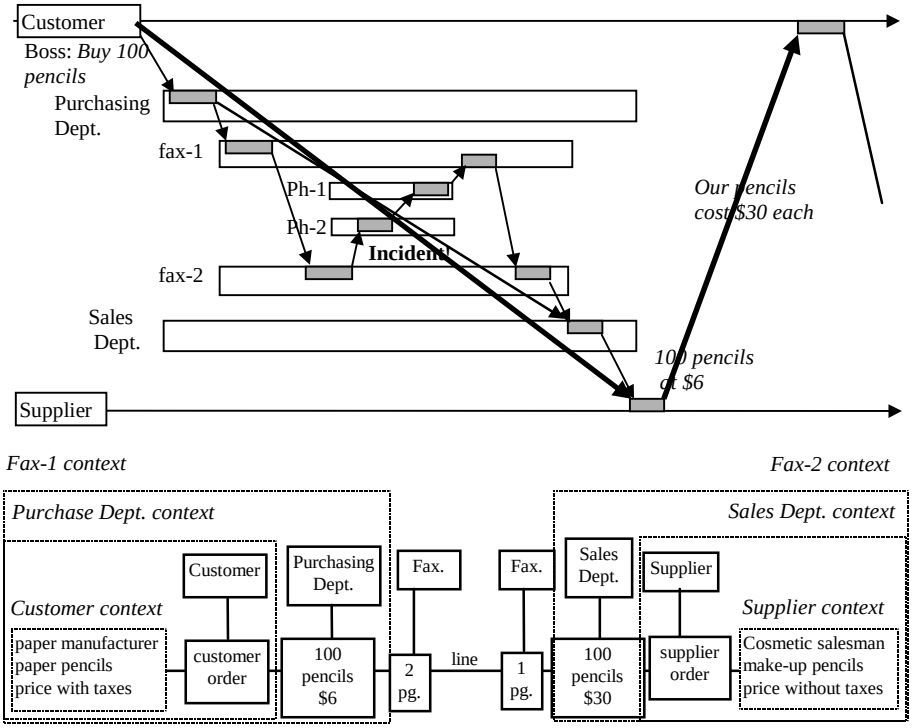

Figure 2: An example of communication between two corespondents using a chain of intermediaries

The correspondents use different intermediaries to communicate: components, actors, organizations, processes, etc. Each one has its proper context, medium, rhythm, and **scale**. The contexts are nested. The Purchasing Department's context could not be understood only within the customer's context, and so on. At the same time the Boss uses the Purchasing Department as a **medium**. The scaling mechanism plays an important role within the interpretation process.

4.3 Necessity of more rigorous engineering of communications inter-autonomous systems

The technology and engineering of communication inter-autonomous systems are not as formalized as, for example, those of the databases or even the user-interfaces. However, the difficulties of software and information systems are found nowadays more at the interfaces than the databases!

The interfaces represent articulations between the system components. If components are not autonomous, or if articulations are too rigid, the system cannot be flexible. This is a real problem for enterprises, not only a question of technology. The enterprises' merge, acquisition require connecting information systems that were independent before, or separating them. The exchanges between enterprises and their external partners are more performed automatically and **formalized**. The work is changing: individuals are asked to cooperate and to increase synergy. We attend to more jobs' professionalization by structuring the exchanges between them. The exchanges of information are becoming more **formal** and **dematerialized**.

In order to illustrate this, consider a supplies and stocks management activity within an enterprise. When this activity is "autonomized", it forms an interface between the consumers and the suppliers, and hence increases the consumers' autonomy relatively to the suppliers. Stocks are a representation of the **shift** introduced between demand and supply, expressed on three aspects:

- The **time**: a difference between a demand and the order corresponding to the stock level.
- The **space**: the stock represents the space-buffer.
- The **scale**: we may buy wholesale what could be consumed in retail.

The supplies and stocks management activity reconciles different points of view with different scales. This autonomy increases the enterprise reactivity, which takes the opportunity of realizing some non-planned activities, while the Purchasing Department may allow rules of competition to operate. There is then, at the same time, a local and global optimization.

Interfaces between components represent this type of articulation in space, time, scale, and information coding means. Standardized engineering practices, able to build them and to make them evolve are certainly within our range (Desreumaux, 1995).

4. FROM ARCHITECTURE TO URBANISM

5.1 Management practices of large software-intensive systems

Some large companies and organizations in France (banks, insurance, energy, telecommunications, etc) put in place a process called **urbanism of information systems, which** encompasses several activities such as:

- Standardizing architectures.
- Promoting high level communication services, e.g. middleware.
- Separating the purely applicative software systems from the software infrastructure.
- Purchasing and integrating COTS rather than building new ones.
- Structuring systems logically.
- Planning the previous evolutions.
- Maintaining a cartography of software systems, information flows, business processes.
- Organizing information systems processes in order to facilitate and to promote reuse.

This pragmatic know-how allows effectively, with intuition and energy, to increase the control of large software-intensive systems. But, a more rigorous mechanism that may allow understanding the phenomena in order to better react on them, could not exist?

Some first elements of the answer derive from the analogy that we could make with the large sets of habitations. The word "architecture" has already been borrowed from this field, when it comes to erecting a building or a software component. Could we pursue the analogy when it comes to understanding the evolution laws of large sets of software components?

5.2 Cities Urbanism

The origin of the words family "urbanism" is very interesting.

The French word "urbanité", urbanity, appears in 1370, from Latin *urbanitas,* meaning "from city, having the characteristic way of life of city dwellers". Urbanity: **1**. Refinement and elegance of manner; polished courtesy **2**. Urban life.

In 1867, **Cerda**, a Spanish engineer published "*teoria de la urbanización*". *Urbanización* designates **1**. The process of space arrangement, planned or not. **2.** The underlying laws.

The job of the "*urbanizador*" is to discover **hidden** and unconscious laws, in order to understand and to use them, knowingly, for conception and arrangement of constructed spaces.

In France, since 1873, *urbaniser*, urbanizing, means: **get someone to accept the urbanity.** The word *urbanisme*, (urbanism) appears since 1910, in order to designate: **1**. The science of urbanity. **2**. The study of methods allowing to **adapt** the urban habitat to the person's needs. **3**. The set of techniques for applying these methods.

Fundamentally, the architecture refers to a *construction*, whereas the urbanism refers to an *evolution*. The concept “architecture” is closed around the edifice, its structure and its style, whereas “urbanism” is open on environment, person’s needs and necessity of the mannerliness within a society. Each edifice has to be integrated among other edifices, with urbanity.

5.3 The Urbanism of large information and software-intensive systems

For information systems, “urbanity” would mean their ability to fulfill their functions, to hold out and to fit in within a changing environment. The environment is composed from a set of human, software and hardware systems that interoperate with the system, and which are themselves in interaction**. Urbanity and autonomy are thus intimately tied**. In some way, the urbanity of information systems would mean their ability “to live in a society”.

From the beginning, urbanism integrate description of space, social, economic, political, cultural dimensions. Later environmental and ecological dimensions. **The role of communications is up-front of a considerable importance.**

We find again the same characteristics and fundamental problematics within the urbanism of information systems: change from planned and individual construction to evolution of collective space, consideration of legacy systems, multi-disciplinarity, environmental protection, determinant role of communications.

5.4 Large networks of components

There is no simple relationship between the laws underlying two sand seeds and a sand heap; between Brownian interactions of some molecules, and the gas properties. Observation of chemical communications between ants does not allow to predict ant's nest organization. The understanding of a dialogue between two humans explains poorly the crowds’ behavior. The nerve exchanges impulse within the synapses of some neurons do not account for all the mind and thought complexity.

There is, almost always, a quantum leap in quality between the interactions among some individuals and the global behavior of a large collection: changing from order to disorder, or vice versa, or from an order to another order.

This qualitative leap may be found (again) within systems comprising a large number of components, like large information systems. Each software component is **individually** programmed, under control, sensitive to the least change within its environment and therefore not very evolving, it becomes “hard”. In the opposite, the entire system is not programmed, has unexpected behaviors and even unpredictable, and stays “soft” relatively to changes. The entire system has some adaptation abilities, and we can observe its evolution.

Structural assembling components is a matter of architecture. However, the assembling of large number of components provides other results, qualitatively

different. The underlying laws of large system communities are not simply deduced from the assembling laws of architectures. The urbanism: discovery and usage of behavioral, adaptation and evolution laws of large collections cannot be structuraly described.

A solution to approach this laws will be the construction of models such as mesh automata. (Weisbuch, 1989) (Rumelhart, 1989) provide a large class of these models. Many of them demonstrate flexibility, adaptability, fault-tolerance, learning abilities. But they are not directly “programmable”, and their behavioral laws are sometimes counter-intuitive. They are not very predictable.

They allow understanding what could mean “controlling” large system collections. A system cannot be known and programmed in the least details, on one hand like the von Neumann machine, and on the other hand be permanently adaptative to the environment hazards.

5. REFERENCES

Bass L., Clements P., Kazman Rick (1998) *Software Architecture in Practice*, Addison-Wesley.

Desreumaux M., Oudrhiri R. (1995). "Systèmes d'Information et Interfaces". *2ème Congrès Biennal de l'AFCET*, 25-27 octobre, Toulouse.

Dijkstra E.W. (1976). A discipline of programming, Prentice Hall.

Lorigny J. (1992). *Les systèmes autonomes, Relations aléatoire et sciences de l'esprit*, Dunod-AFCET, Paris.

Nottale L. (1993). *FRACTAL SPACE-TIME AND MICROPHYSICS, Toward a Theory of Scale Relativity*, World Scientific.

Rumelhart D.E., McClelland J.L.. (1989) *Explorations in Parallel Distributed Processing*, MIT Press.

Varela F.J. (1979). *Principles of Biological Autonomy* Elseiver/North-Holland, New-York.

Weisbuch G. (1989). *Dynamique des systèmes complexes, Une introduction aux réseaux d'automates*. InterEditions/Editions du CNRS, Paris.