R&S

Responsibility and Sustainability
Socioeconomic, political and legal issues
ISSN: 2340-5813
Vol 1(2), pp: 61-67 *Research note*

ON A MODEL OF ENTERPRISE ARCHITECTURE FOR PUBLIC INSTITUTIONS IN ROMANIA

Cristian Bologa *(Babes Bolyai University of Cluj Napoca, Romania)* ✉
José Luis Vázquez *(University of León, Spain)*
Gheorghe Faur *(Babes Bolyai University of Cluj Napoca, Romania)*
Paul Bresfelean *(Babes Bolyai University of Cluj Napoca, Romania)*
Nicolae Ghisoiu *(Babes Bolyai University of Cluj Napoca, Romania)*

Abstract:

This paper presents a possibility to conceive the conceptual design of a Romanian public institution using a methodology known as the "Enterprise Architecture". The Enterprise Architecture will be applied to any public institution, regardless of hierarchical level in the overall organizational structure. This Architecture can be divided into four components: Business Architecture, Information Architecture, Software Architecture and Technological Architecture.

As a conclusion, the need for creating a generic, abstract Architecture is highlighted, as well as the need of future researches on exploring and identifying those patterns for application of Enterprise Architecture in the public sector with significant roles in the development of the specific software architectures that serve the goals of such a hierarchical system, as well as aiming to ensure the uniformity of the centralized information, at least at the level of national institutions.

Keywords: *public institutions*; *enterprise architecture*; *information architecture*; *technology architecture*

EN TORNO A UN MODELO DE ARQUITECTURA DE EMPRESA PARA LAS INSTITUCIONES PÚBLICAS EN RUMANÍA

Resumen:

Este artículo plantea la posibilidad de concebir el diseño conceptual de una institución pública rumana usando la metodología conocida como "Arquitectura de Empresa". La Arquitectura de Empresa será aplicada a cualquier institución pública, sin importar su nivel jerárquico en la estructura organizativa global. Esta Arquitectura puede dividirse en cuatro componentes: Arquitectura de Negocio, Arquitectura de Información, Arquitectura de Software y Arquitectura Tecnológica.

Como conclusión, se destaca la necesidad de establecer una Arquitectura genérica y abstracta, así como de futuras investigaciones que exploren e identifiquen los patrones de aplicación de la Arquitectura de Empresa en el sector público con papeles significativos en el desarrollo de las arquitecturas de software específicas al servicio de los objetivos de tal sistema jerárquico, así como tratando de asegurar la uniformidad de la información centralizada, al menos al nivel de las instituciones nacionales.

Palabras clave: *instituciones públicas*; *arquitectura de empresa*; *arquitectura de información*; *arquitectura tecnológica*

✉ Faculty of Economics (Babes Bolyai University of Cluj Napoca), Teodor Mihali 58-60 400591-Cluj Napoca (Romania)
e-mail: *cristian.bologa@econ.ubbcluj.ro*

1. Introduction: on the structure of Romanian public institutions

According to the Romanian legislation the term "public institution" appears as a broad and generic name including the Parliament, the Presidential Administration, the Ministries, other organs of public administration, other public authorities, public institutions, autonomous and subordinated institutions, regardless their financing method.

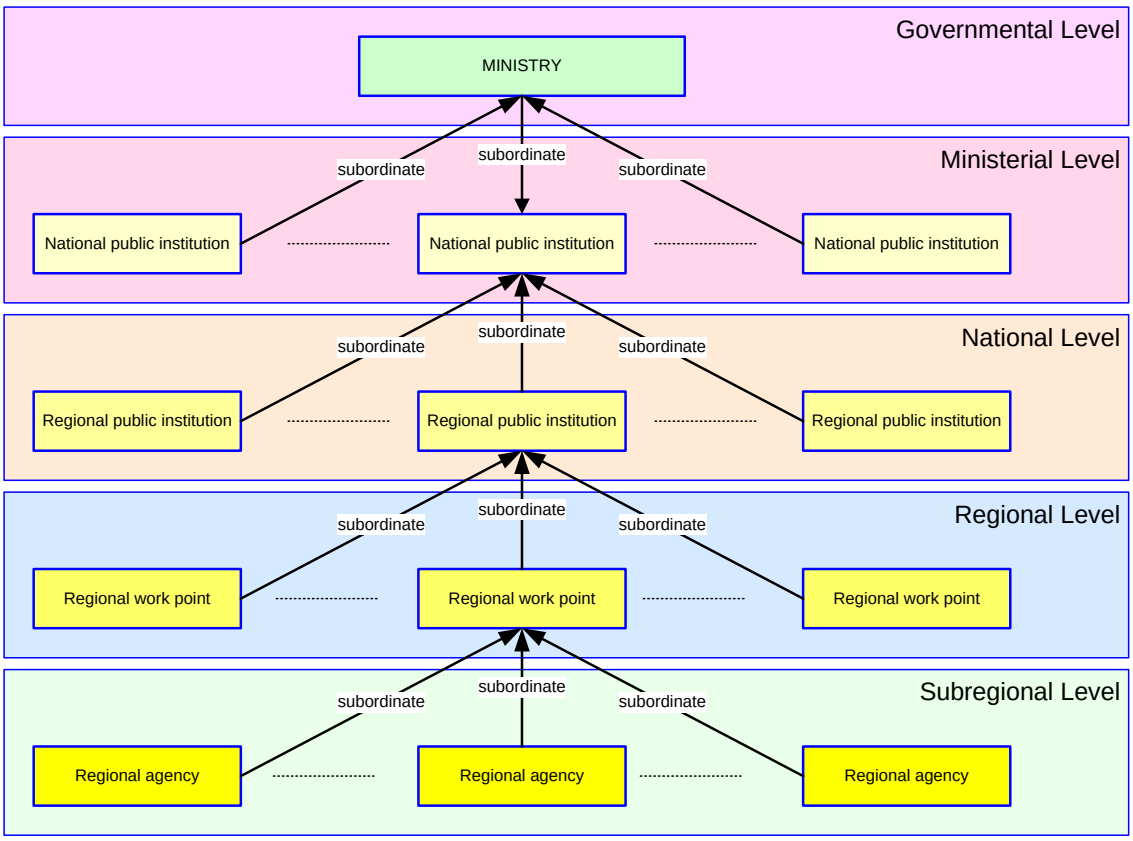
Currently, ministries are organized on a hierarchic-pyramidal scale, which includes the Ministry on top of the hierarchy. Subordinate to ministries public institutions and other institutions can be found, performing co-ordinately at ministerial level. Such scheme continues at national level, with working points at regional level (for example, the county level), and sub-working points at sub-regional level (Figure 1).

The issue that arises in this hierarchical approach is to ensure the satisfaction of information necessities of high-level institutions, providing consistent enough data to allow substantiation of the decisions at each level of the hierarchy, depending on the institutional level and the ability to transfer and to consolidate such data at higher levels as comprehensive, relevant and consistent information. The aim is to provide each institution with the capacity to make its own decisions according to its competence and to formulate strategies to address its activities. At higher levels of the hierarchy this also includes the capacity to formulate and even propose law initiatives in the respective fields, according to current government policy rules.

2. Computerization of public institutions

In the informatics area, each ministry tends to have its own computerization policy or even to have no such policy. Post-revolutionary Romania had not a coherent policy on computer issues at central, government level. This is why, it has not yet been possible to bind a National Information System (NIS), in which public institutions can obtain the data they need from each other in a digital, efficient way, avoiding data redundancies and ensuring to stakeholders (mainly to economic agents) consistent services to meet certain quality standards, which is one of the main engines of economic development.

Figure 1 Organizational structure of a public government institution



Source: own elaboration

Trends and attempts to computerize public institutions have always turned into independent, uncorrelated projects which effects were dissipated in particular initiatives involving the promoter or, at the most, some of those other subordinate institutions to the former one. In this way, many (most) of the computerization projects in public Romanian institutions were based solely on the implementation of "on demand" particular solutions. When developing a certain law, ordinance, or work arrangement, public managers have taken steps to find such particular solutions in the informatics area usually in three different ways:

- Contracting by auction of systems to solve the punctual problem: in this case, the issue of subcontracting of a specific informatics solution, which is not part of a regular pattern, such as Enterprise Resource Planning systems (ERP) or Customer Relationship Management (CRM) leads in most cases to implementation failures or to partial operation of the system and not the optimal parameters it should work. This is because a subcontractor can only do an analysis and design according to the concrete specification on required computer systems, without information about the institution's strategic vision or the architectural requirements it needs. Thus, rarely he may include in the analysis the multitude of elements that are necessary in practice.

- Developing proprietary software at national level: if the human and material resources are limited, systems functioning becomes weak due to the development of self-go method, where each element not taken into account when designing the application must be reconsidered at each change, whether at law (the law is developed incomplete or too ambiguous to meet the necessities for which it was developed) or in functional nature terms (inability to implement a specific solution).

One example is the decision to ask from the economic agents electronically signed forms in digital format, without taking into account the fact that there is no regulation requiring this agent to hold such an electronic signature. Usually, these types of software are monolithic applications to be recompiled at each change. Territorial distribution of reconstructed modules leads to unnecessary disruptions and losses for the lower level of the institutions that prepare the system aiming the implementation of more or less flexible solutions, often with limited resources and giving chance to oversized developments, with troubles of inconsistency and lack of scalability, avoiding the possibility of reuse of modules or components.

- Lack of an informatics strategy at national level: the decision to design systems to meet emerging requirements is left in hands of subordinated institutions, which choose either to follow one of the above first two methods, or simply to ignore the idea of introducing a specific novelty, preferring methods that work in a classic, bureaucratic, non-computerized way. This approach is found very frequently, resulting in a patchwork of applications that in essence perform the same tasks, dispersed in a wide variety of technologies.

3. Considerations on software architectures

The term "software architecture" began to appear in the literature in the 70's (Brooks 1995). Later on, authors as David Garlan, Mary Shaw, Dewayne Perry and Alexander Wolf developed the most important studies and papers in the field of software architectures in the early 90's. Since the beginning of the emergence and consolidation of the concept of software architecture, its definition has been approached in different ways by several authors.

So, Shaw and her colleagues stated that "the architecture of a software system defines that system in terms of components and interactions between these components. In addition to specifying the structure and topology of the system, architecture presents a correlation between system requirements and the components of the built software" (Shaw et al. 1995).

Bass, Clements and Kazman considered software architecture as "the architecture of a program or system is the structure or structures containing the system software components, externally visible properties of those elements and relations between them" (Bass et al. 1998).

Perry and Wolf suggested a model of software architecture that consists of three components: elements, form and rational analysis. In their own words, "by analogy with building architecture, we propose the following software architecture model: software architecture = {elements, form, rational analysis}, that is, a software architecture is a set of architectural (or design) elements of a certain form" (Perry and Wolf 1992).

Finally, Gacek, Abd-Allah, Clark, and Boehm considered that "a software architecture includes: a collection of software and system components, connections and constraints, a collection of stakeholder necessities, a rational analysis to show that the components, connections and constraints define a system which, if implemented, will satisfy stakeholder needs" (Gacek et al. 1995).

Although these definitions seem to refer to the same concepts, there were debates and positions on the definition of terms as "component" or "structure". All these opinions were formulated over time and summarized in 2000 under the standard *IEEE - 1471-2000 Recommended practice for Software Architecture description*, which establishes that it is "the fundamental organization of a system consisting of components, their relationships with each other, relations with the environment in which there are principles that guide the design and development system" (IEEE 2000).

The conclusion reached on the positioning of software architecture in the case of software development process was well defined by Garlan (2000), based on his famous scheme (Figure 2).

The main idea of software architecture is to bring together various technical problems or potential risks in the early stages of the software process. A reasoning based on architectural descriptions that often takes the form of formal analysis usually contributes to identify problems.

Architecture Tradeoff Analysis Method (ATAM) is one of the most important architectural review processes (Kazman et al. 1998; Kazman et al. 2000; Kazman et al. 2003). This model follows an approach based on scenarios, each one of them is analyzed through the most appropriate analytical techniques (Bengtsson et al. 2000; Lassing et al. 2002). An important advantage of scenario-based architectural analysis approach is that non-functional requirements analysis is replaced by concrete situations.

Like other methods of architectural analysis, ATAM is presented in a sequence of activities or steps, including:

i) *collection of scenarios* from the stakeholders involved, which are generally composed of use cases and modifiable scenarios based in practice;

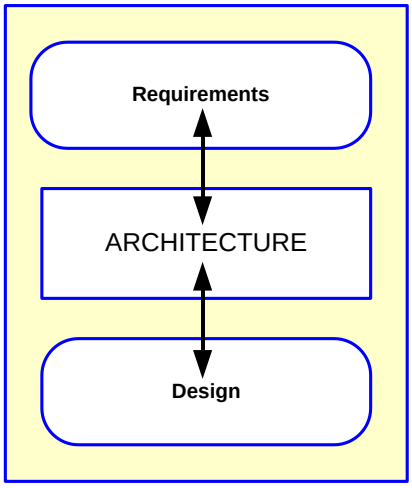
ii) *representation of architecture*, which includes a description of the actual current architecture;

iii) *analysis of specific properties*, at which impacts are analyzed using the available scenarios on the architecture;

iv) *analysis of the most appropriate choices*, step in which the architectural elements are identified by their frequency problem in the analysis of specific property; and

v) *proposed changes*, final step at which the analysis is used as the basis for designing and deciding on architectural improvements.

Figure 2 Place of software architecture in software development process



Source: adapted from Garlan (2000)

Software architecture is a model, a particular form of abstraction of software systems. It should be noted that the abstract models of software have existed since the early computer era. Compared to existing abstract models of software developed over time, software architecture presents some distinguishing features that distinguish it from the first ones: i) it has a high level of abstraction; ii) it refers to complex systems; iii) its designs are made from components and connectors; iv) it describes the structure or topology of the system software; v) it is concerned in particular with the external properties of the relations between these components and software architectures; and vi) it represents a bridge between the requirements analysis phase and the design of the system itself.

Information Technology (IT) software architecture as a subject discipline has developed because of the complexity and difficulty of the use of other techniques to run them, as Interconnection Languages Modules –Module Interconnection Languages, MILs– (Prieto and Neighbors 1986), Hardware Description Languages (IEEE 1988), ADA packages (Cardelli et al, 1989), Interface Definition Languages –IDLs– (Messerschmitt 1999), Structural Analysis (DeMarco 2002), Communicating Sequential Processes (Hoare 2003), or Object-Oriented Modelling Languages (Booch et al. 2005).

These are only a few examples. Many other techniques, concepts and notations preceded conceptual software architecture. The main problem is the difficulty of understanding and assimilation of these methods and techniques. Maybe software architecture as a concept has relatively brought not many news in the modelling software field, but it has contributed to structure concepts in a coherent and understandable way. However, the most important is not the concept of software architecture as a distinct academic discipline, but its fundamental concern on how software systems can be represented as well as understood.

4. Architecturing a public institution

Architecturing a public institution can be achieved using a methodology known as Enterprise Architecture (EA). This concept focuses in the methodology of designing large scale systems, such as those of public institutions, especially regarding the concept of a reference architecture that can be used as a model by different public institutions so that they can achieve national uniformity in designing information to government agencies and ultimately could lead to the concept of e-government, interoperability and collaboration among public sector institutions.

This methodology includes the integration of modelling actions of (Spewak and Hill 1993): i) processes of the computerized field; ii) software applications to be implemented; iii) information and data to be obtained; and iv) technological models that can be used.

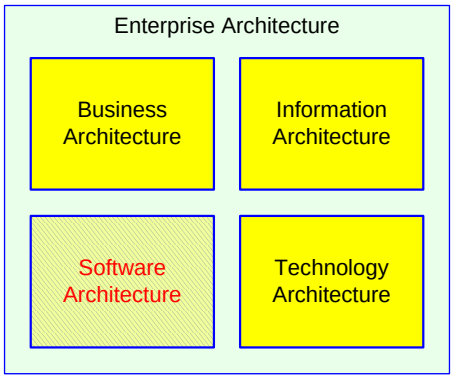
Given the huge quantities of data to be processed, transmitted and analyzed, and due to the need of providing communication channels with other similar institutions, in essence an organization must have a coherent overall view on four key elements: i) architecture of business processes (*business architecture*); ii) *software architecture* to support business processes; iii) architecture of information and data used or obtained from business processes by structuring their coherent and obtain historical operational information or using techniques such as database design and design business intelligence solutions (*information architecture*); and iv) *technological architecture* suitable for achieving the aims of the institution.

Modern institutions, integrated from an informational point of view, should be regarded as entities adaptable to technological advantages, i.e. to the benefits of practical and theoretical results from the field of information and data architecturing on existing knowledge architecturing software components at a larger scale far away from the simple application. There must be a close –often osmotic– connection among the above elements. In this way, *Enterprise Architecture* can be represented very schematically as in Figure 3.

Information systems of a public institution are large systems. In an attempt to cover a short space, a discrepancy between the detailed requirements analysis and the design of the computer system, the concept of *Software Architecture* plays a very important role.

Then, using *Information Architecture* as part of Enterprise Architecture by emphasizing the organization of data within the enterprise and data manipulation operations will lead to obtaining information from data collected Efficiency in business information systems.

Figure 3 Position of Software Architecture in Enterprise Architecture



Source: adapted from Garlan (2000)

Business Architecture includes the vision, mission, objectives and business processes. It appears as the main determinant element of the Information Architecture structure, as data are provided from the corporate environment in which defined architecture is applied.

Finally, the *Technology Architecture* encompasses the entire technology which is used in an organization (business or public institution), both in production (machines, equipment) and information processes (hardware, software).

5. Conclusions

The content of this paper is referred to the application of Information Systems (IS) and Technologies (IT) in Romanian public institutions according to their hierarchical structures and relationships, their defining features and characteristics, and their status of computerization, as well as taking in mind the opportunities for the improvement and standardization of information systems architectures by determining and implementing or adapting viable solutions from the software community.

As a conclusion, we should highlight the need for creating a generic, abstract Architecture, which could be applied to any public institution, and regardless of hierarchical level that the firm occupies in the public organization as a whole. Such Architecture needs of a certain kind of data streams in order to intelligently and easily obtain information, which leads to the need of specifically designing an *Information Architecture*.

In most cases, although the processes that are taking place at the medium levels in the hierarchy of public institutions are identical, there are also incompatibilities and inconsistencies in results which use to complicate or even make impossible to achieve a consistent centralization of data and information, even when using at this purpose specialized and advanced techniques such as Enterprise Application Integration (EAI) or no matter the hesitating attempts of implementing Service Oriented Architecture (SOA).

So, future research guidelines should explore and identify those patterns for application of Enterprise Architecture in the public sector with significant roles in the development of the specific software architectures that serve the goals of such a hierarchical system, as well as aiming to ensure the uniformity of the centralized information, at least at the level of national institutions. In this sense, architectural templates will be used to implement centralized viable solutions.

Acknowledgements

The research results presented in this paper were obtained as part of the CNCSIS IDEI_1598 Grant "Invatamantul superior si piata muncii. Cercetari bazate pe tehnologii informatice privind corelatia dintre calificarile cerute de piata muncii si cunostintele reale ale studentilor" ("Higher education and the labour market. Information technology-based research on the correlation between the qualifications required by the labour market and students' knowledge"), managed by Prof. Dr. N. Ghisoiu.

References

Bass L, Clements P, Kazman R (1998) Software architecture in practice. 2nd ed. Addison Wesley, Boston

Bengtsson P, Lassing N, Bosch J, van Vliet H (2000) Analyzing software architectures for modifiability. Research Report 11/2000. University of Karlskrona/Ronneby, Ronneby (Sweden)

Booch G, Rumbaugh J, Jacobson I (2005) The unified modeling language user guide. 2nd ed. Addison Wesley, Boston

Brooks F (1995) The mythical man-month: essays on software engineering. Anniversary ed. Addison Wesley, Boston

Cardelli L, Donahue J, Jordan M, Kalsow B, Nelson G (1989) Modula-3 report (revised). System Research Center, Palo Alto CA

DeMarco, T. (2002) Structured analysis. Beginnings of a new discipline. In: Broy M (ed), Denert E (ed) Software pioneers. Contributions to software engineering. Springer, Berlin, pp 520-527

Gacek C, Abd-Allah A, Clark B, Boehm B (1995) On the definition of software system architecture. Paper presented at the ICSE 17 Software Architecture Workshop. Los Angeles, 24th-25th April

Garlan D (2000) Software architecture: a roadmap. In: Finkelstein A (ed) The future of software engineering. ACM Press, Pittsburgh PA, pp 91-101

Hoare, CAR (2003) Communicating sequential processes. Prentice Hall, Upper Saddle River NJ

IEEE (1998) IEEE standard VHDL language reference manual. Institute of Electrical and Electronics Engineers, New York

IEEE (2000) IEEE 1471-2000 recommended practice for architectural description for software-intensive systems. Institute of Electrical and Electronics Engineers, New York

Kazman R, Klein M, Barbacci M, Longstaff T, Lipson H, Carriere J (1998) The architecture tradeoff analysis method. Paper presented at the 4th International Conference on Engineering of Complex Computer Systems (ICECCS98). Monterey, 10th-14th August

Kazman R, Klein M, Clements P (2000) ATAM: a method for architectural evaluation. Software Engineering Institute Technical Report CMU/SEI-2000-TR-004. Software Engineering Institute Carnegie Mellon, Pittsburgh PA

Kazman R, Nord R, Klein M (2003) A life-cycle view of architecture analysis and design methods. Technical note CMU/SEI-2003-TN-026. Carnegie Mellon University, Pittsburgh PA

Lassing N, Bengtsson P, van Vliet H, Bosch J (2002) Experiences with ALMA: Architecture-Level Modifiability Analysis. Journal of Systems and Software 61(1):47-57

Messerschmitt D (1999) Interface definition languages. University of California, Oakland CA

Perry D, Wolf A (1992) Foundation for study of software architecture. ACM SIGSOFT Software Engineering Notes 17(4):40

Prieto-Díaz R, Neighbors JM (1986) Module interconnection languages. Journal of Systems and Software 6(4):307-334

Shaw M, DeLine R, Klein DV, Ross T, Young D, Zelesnik G (1995) Abstractions for software architecture and tools to support them. IEEE Transactions on Software Engineering 21(4):314-335

Spewak SH. Hill SC (1993)., Enterprise Architecture Planning; Developing a Blueprint for Data, Application and Technology, John Wiley & Sons, New York