The Impact of Application Non-Functional Requirements on Enterprise Architecture

Darko Galinec
Information and Communication Technology Department
Ministry of Defence of The Republic of Croatia
Bauerova 31/2., Zagreb
darko.galinec@morh.hr

Ljerka Luić
B4b Zagreb
Ulica grada Vukovara 271
ljerka.luic@b4b.hr

Abstract. *Enterprise architecture (EA) is seen as a key enabler for driving information technology (IT) cost down and speed benefit delivery, so that the right solutions are delivered faster and cheaper [6]. However, EA can no longer focus on cost reduction and IT rationalization, but must shift to delivering strategic business value [13]. Many EA teams still use IT architecture as a term synonymous with EA. This misperception limits EA scope and, thus, possible business outcomes and EA value [16]. By producing an EA, architects are providing a tool for identifying opportunities to improve the enterprise, in a manner that more effectively and efficiently pursues its purpose. In this paper we examine means for enterprise architecture improvement, in order to help the enterprise architecture team produce a compelling value proposition. By focusing on non-functional requirements of the software applications within EA description and appliance, we investigate its impact on EA. We find them as one of key artifacts of EA for complex business technology initiatives support in achieving enterprise strategic goals.*

Keywords. application, non-functional requirements, enterprise architecture.

1 Introduction

An enterprise architecture (EA) is a rigorous description of the structure of an enterprise, which comprise enterprise components (business entities), the externally visible properties of those components, and the relationships (e.g. the behavior) between them [5], [4], [21]. Within the definiton of EA mentioned above, description of EA is comprehensive, including enterprise goals, business process, roles, organizational structures, organizational behaviors, business information, software applications and computer systems. Practitioners of EA are "enterprise architects." An enterprise architect is a person responsible for developing the enterprise architecture and is often called upon to draw conclusions from it.

A business application delivers value to the business by providing support for business processes. The business invests in applications because the investment is seen to deliver some real value by delivering the functional requirements e.g. the visible functionality that is necessary for these processes. An application is a complex engineering system that needs to be able to deliver value throughout its life cycle. There are many aspects of the engineering design and construction of an application that do not provide visible support for the business process directly but which are nevertheless critical to the efficient operation of the application: non-functional requirements. Non-functional requirements or attributes have been given different names in different disciplines such as “non-functional properties”, “service level agreement properties”, “quality of service properties” or “extra-functional properties”, which include security, availability, reliability, maintainability, agility, timeliness, location, price, performance and many other aspects of the system that are not immediately obvious to the user but which have a big impact on the applications value.

When new applications are developed the most commonly emphasized non-functional requirements are “cost to go live” and “time to go live”. Failure to consider many other non-functional requirements during the architecture and design phase of the system generally means accepting far higher lifetime costs and lower lifetime satisfaction [12, p. 1].

2 The scope of an enterprise architecture

The term enterprise refers to a complex, socio-technical system that comprises interdependent resources of people, information, and technology that must interact with each other and their environment in support of a common mission [5], [4].
The term "enterprise" is used because it is generally applicable in many circumstances, including

- Public or Private Sector organizations,
- An entire business or corporation,
- A part of a larger enterprise (such as a business unit),
- A conglomerate of several organizations, such as a joint venture or partnership,

- A multiply-outsourced business operation.

Defining the boundary or scope of the enterprise to be described is an important first step in creating the enterprise architecture. It should also be noted that the term "enterprise" as used in enterprise architecture generally means more than the information systems employed by an organization.[14]

2.1 Methods and frameworks

Enterprise architects use various business methods, analytical techniques and conceptual tools to understand and document the structure and dynamics of an enterprise. In doing so, they produce lists, drawings, documents and models, together called "artifacts". These artifacts describe the logical organization of business functions, business capabilities, business processes, people organization, information resources, business systems, software applications, computing capabilities, information exchange and communications infrastructure within the enterprise.

An enterprise architecture framework collects together tools, techniques, artifact descriptions, process models, reference models and guidance used by architects in the production of enterprise-specific architectural description. Describing the architecture of an enterprise aims primarily to improve the effectiveness or efficiency of the business itself. This includes innovations in the structure of an organization, the centralization or federation of business processes, the quality and timeliness of business information, or ensuring that money spent on IT can be justified. An Enterprise Architecture Framework is a framework for an enterprise architecture which defines how to organize the structure and views associated with an enterprise architecture.

2.2 View and modeling perspectives

A view of a system is a representation of the system from the perspective of a viewpoint. This viewpoint on a system involves a perspective focusing on specific concerns regarding the system, which suppresses details to provide a simplified model having only those elements related to the concerns of the viewpoint. For example, a applications viewpoint focuses on applications concerns and a applications viewpoint model contains those elements that are related to applications from a more general model of a system [4].

A view allows a user to examine a portion of a particular interest area. For example, an Information View may present all functions, organizations, technology, etc. that use a particular piece of information, while the Organizational View may present all functions, technology, and information of concern to a particular organization. In the Zachman Framework views comprise a group of work products whose development requires a particular analytical and technical expertise because they focus on either the “what,” “how,” “who,” “where,” “when,” or “why” of the enterprise. For example, Functional View work products answer the question “how is the mission carried out?” They are most easily developed by experts in functional decomposition using process and activity modeling. They show the enterprise from the point of view of functions. They also may show organizational and information components, but only as they relate to functions [20].

Modeling perspectives is a set of different ways to represent pre-selected aspects of a system. Each perspective has a different focus, conceptualization, dedication and visualization of what the model is representing. In information systems, the traditional way to distinction between modeling perspectives is structural, functional and behavioral/processual perspectives. This together with rule, object, communication and actor and role perspectives is one way of classifying modeling approaches [10].

2.3 The Zachman framework and The Open Group Architecture framework

The Zachman Framework is often referenced as a standard approach for expressing the basic elements of enterprise architecture. The Zachman Framework has been recognized by the U.S. Federal Government as having "...received worldwide acceptance as an integrated framework for managing change in enterprises and the systems that support them" [1].

Example of enterprise architecture frameworks in military (defense industry frameworks) are:

- DoDAF - the US Department of Defense Architecture Framework,
- MODAF - the UK Ministry of Defence Architecture Framework,
- NAF - the NATO Architecture Framework,
- AGATE - the France DGA Architecture Framework,
- DNDAF - the DND/CF Architecture Framework (Canada).

The Open Group Architecture (TOGAF) framework divides the practice of enterprise architecture into three domains: "Business Architecture", "Information Systems Architecture" and "Technology Architecture" and then subdivides the information systems architecture into "Information Architecture and "Applications Architecture"[18].

3 Information technology architecture and enterprise architecture

EA obviously does not focus only on IT change; EA also addresses business change. When EA derives requirements for change in the different EA areas, as part of EA strategizing to define the business context, it provides a much more business-visible linkage between IT strategy and investments. Changes in technology can be mapped back to information, people or process changes, and through those changes to business strategies. This missing thread of justification, this line of sight, is what a true EA approach really adds, and where IT architecture normally falls short.

3.1 Application systems

Many EA teams and their stakeholders still use the term "IT architecture" to refer to EA. This effectively limits EAs scope and the value delivered, and increases the risk that the EA program will be ignored or cut. IT architecture is not synonymous with EA. IT architecture typically means focusing only on the enterprise technical architecture (ETA) aspects of EA. IT architecture always includes individual solution or project architecture work, which is not EA activity at all. Therefore, move from an IT architecture approach to a full EA approach is to be made. Usage of the term "IT architecture" — which is certainly not a synonym for EA — is to be avoided. Those who still use IT architecture and EA synonymously are to be educated, clarifying the significantly wider scope and value of holistic EA [16].

Software product development is typically carried out in project settings in which complexity, unpredictability, and continuous change is common [9]. Software development projects, as a form of work, often involve complex problem solving as well as potentially changing customer requests, tasks, colleagues, and physical places of work. New software product development is typically carried out in projects. There are several established approaches to new product development, which have focused on exploring processes of new product development projects. Much mainstream new product development research has investigated how such projects are managed and how new product development processes can be improved [3].

3.2 Enterprise architecture as a Service

If the process isn't real life then people will subvert it. Processes are not the be all and end all of the way organizations work because much of what happens is independent of the formal processes. Trying to formalize interactions between different business teams can make life overly complex [6].

Architecture is seen as a key enabler for driving IT cost down and speed benefit delivery, so that the right solutions are delivered faster and cheaper.

For most of stakeholders (project teams, service teams, planners and architects, infrastructure providers, business and IT planners) the "Architecture Process" is obscure. According to their perception "architecture is there to enforce compliance with standards", "architecture stifles innovation", "architecture slows our projects down".

As a solution, delivering the architecture capabilities as a service is proposed where key stakeholders engage with the service, their engagement is event driven, not process driven and architecture processes are disengaged from other project processes. Metrics drive behavior: service metrics illustrate what architecture delivers independently of what projects deliver, positive measures encourage use of the service.

3.3 System attributes

Business can invest in an application system by developing, licensing or subscribing, but the primary aim of the business stakeholders tends to be the functionality: what the application system does. The investment is being made because some business benefit is being sought, and the evaluation of the target application is centered around the way in which it will deliver that business benefit. Alongside these "functional fit" evaluation criteria there are often some "non-functional" system attributes which need evaluation. These might include issues such as "ease of use" or "multi-language support".

Very few organizations have an effective mechanism for defining requirements for a broad range of application attributes in a consistent manner. There are many different system attributes but most enterprises seem to be somewhat haphazard when it comes to defining the minimum required measurement to be applied to each and every attribute. For example the requirements team may fail to explicitly specify the requirements for "availability" in terms of hours per day or week or month and frequency of permissible downtime. If no such attribute requirements definition exists then every design will be evaluated without reference to this attribute. Later on, when the investment decision has been made and money has been spent, there is plenty of opportunity to rue the absence of an "availability" goal [12, p. 3].

When a business team decides to invest in a new business application, ownership of the functional specification resides with the lead business team, whether this is a transformation of business process improvement team or simply the management team of the funding group. The business owners of the functional specification will doubtless be assisted by consultants, business analysts and process design specialists, but the final decision rests with "the business".

4 Non-functional requirements

The ownership of non-functional requirements (attributes) is much less clear. Some may be a visible component of the project brief (e.g. "the application must be available 24/7 to support all the countries where we operate) but inadequately specified or over-specified. But many system attributes are completely missing from the project brief – not mentioned, not specified, not designed for and not funded. This is dangerous – particularly because it means that many different stakeholders will have different expectations of what they think that they are paying for.

The completeness of the design specification of any application must be the responsibility of the Information technology department – no other functional group in the enterprise has the capability to understand the nonfunctional requirements and the cost implications of setting design targets. Within the information technology department it is the enterprise architecture team who should shoulder this responsibility, working alongside the project management office.

4.1 ISO 9126 Software Quality Model

ISO 9126 provides structure for understanding non-functional requirements: ISO 9126-1 (Table 1). The six groupings (Functionality, Reliability, Usability, Efficiency, Maintainability and Portability) are each decomposed into a number of sub-definitions [8].

The challenge for an enterprise architecture team is to extract from this source the key elements that should be articulated for any given project proposal, and to develop a local set of tools that support the understanding of and discussions about the appropriate level that any specific non-functional requirements should achieve.

A project brief should include sections for each of the high-level attribute groupings, with option selection of specific lower-level attribute definitions. The non-functional requirements specifications should be [12, p. 6]:

- **Complete:** including "no known requirements"
- T**ransparent:** avoiding using technical jargon – explain the value choices that will need to be made in terms that non-technical managers can understand
- **Ranged:** Not specifying a single value. Rather, the recommended lower and upper levels of specification should be specified and the cost and performance implications should be explained.
- **Measurable:** Showing how the achievement of the specified level can be tested.
- **Comparative:** Comparing the proposed behavior of the system with known references, both within and outside the business. e.g. – "the same level of availability as our email system". This greatly assists the non-technical manager in understanding the proposal.

The project tends to become the dominating way of organizing operations in many industries. An increasing number of organizations are identified as ‘project-based’, i.e. organizations where almost all operations are organized as projects and where permanent structures fill the function of administrative support. The projectified society means that more and more organizational members are being redefined as project workers and project managers [2], which has an effect on their identity. Enterprise logic, that is, initiative, energy, self-reliance, boldness, willingness to take responsibility for one’s actions, might even become a major element in their self-identities [17]. However, because project management focuses on structure, activities, and control, identity issues in project settings have been relatively unexplored.

4.2 ISO 9126 observations

ISO 9126 is an international standard for the evaluation of software. The standard is divided into four parts which addresses, respectively, the following subjects: quality model; external metrics; internal metrics; and quality in use metrics.

For the most part, the overall structure of ISO 9126-1 is similar to past models, although there are a couple of notable differences. Compliance comes under the functionality characteristic, this can be attributed to government initiatives like SOX. In many requirements specifications all characteristics, that are specified, that are not pure functional requirements are specified as non-functional requirements. It is interesting to note, with ISO 9126, that compliance is seen as a functional characteristic.

Using the ISO 9126-1 (or any other quality model) for derivation of system requirements brings clarity of definition of purpose and operating capability. For example a rules engine approach to compliance would enable greater adaptability, should the compliance rules change. The functionality for compliance could be implemented in other ways but these other implementation methods may not produce as strong an adaptability characteristic as rules, or some other component based, architecture.

Also, a designer typically will need to make trade offs between two or more characteristics when designing the system. Consider highly modularized code, this code is usually easy to maintain, i.e. has a good changeability characteristic, but may not perform as well (for central processing unit resource, as unstructured program code). On a similar vein a normalized database may not perform as well as a not normalized database. These trade offs need to be identified, so that informed design decisions can be made.

Although ISO 9126-1 is the proposal for a useful quality model of software characteristics, it is unlikely to be the last. The requirements (including compliance) and operating environment of software will be continually changing and with this change will come the continuing search to find useful characteristics that facilitate measurement and control of the software production process [8].

Table 1: ISO 9126-1 software quality model - structure for understanding non-functional requirements

Characteristics	Subcharacteristics	Definitions
	Suitability	This is the essential Functionality characteristic and refers to the appropriateness (to specification) of the functions of the software.
	Accurateness	This refers to the correctness of the functions, an ATM may provide a cash dispensing function but is the amount correct?
Functionality	**Interoperability**	A given software component or system does not typically function in isolation. This sub characteristic concerns the ability of a software component to interact with other components or systems.
	Compliance	Where appropriate certain industry (or government) laws and guidelines need to be complied with, i.e. SOX. This sub characteristic addresses the compliant capability of software.
	Security	This sub characteristic relates to unauthorized access to the software functions.
	Maturity	This sub characteristic concerns frequency of failure of the software.
Reliability	**Fault tolerance**	The ability of software to withstand (and recover) from component, or environmental, failure.
	Recoverability	Ability to bring back a failed system to full operation, including data and network connections.
	Understandability	Determines the ease of which the systems functions can be understood, relates to user mental models in Human Computer Interaction methods.
Usability	**Learnability**	Learning effort for different users, i.e. novice, expert, casual etc.
	Operability	Ability of the software to be easily operated by a given user in a given environment.
Efficiency	**Time behavior**	Characterizes response times for a given thru put, i.e. transaction rate.
	Resource behavior	Characterizes resources used, i.e. memory, cpu, disk and network usage.
	Analyzability	Characterizes the ability to identify the root cause of a failure within the software.
Maintainability	**Changeability**	Characterizes the amount of effort to change a system.
	Stability	Characterizes the sensitivity to change of a given system that is the negative impact that may be caused by system changes.
	Testability	Characterizes the effort needed to verify (test) a system change.
	Adaptability	Characterizes the ability of the system to change to new specifications or operating environments.
Portability	**Installability**	Characterizes the effort required to install the software.
	Conformance	Similar to compliance for functionality, but this characteristic relates to portability. One example would be Open SQL conformance which relates to portability of database used.
	Replaceability	Characterizes the *plug and play* aspect of software components, that is how easy is it to exchange a given software component within a specified environment.

4.3 Consequences of investment decisions

Each software project commences with requirements specification phase. Unspecified requirements present a constant rise of errors, disillusionment, and costly solutions required to fix them. Frequent changes of processes and technologies direct adaptive and tool supported requirement specification.

The dangers and cost of standardization in organizations have been neglected, including impacts on subjectivities [7]. Complex ways in which individuals respond to dominant discourses of organizations seem to be under-explored [19].

The basic effort for designers and architects is in the conflict between getting quick and cheap solutions and the need to design a system where the lifetime total cost of ownership (TCO) is acceptable. The problem is that for a project manager the objective is to deliver the project and for the business manager under pressure, "benefits soon" is tangible whereas "benefits later" may well accrue to somebody else. All this means that many of the necessary non-functional requirements are ignored or ruled out of scope or out of budget.

There will be times when it is the correct choice for the enterprise to decide to make the application quickly and not to add the complexity and cost of higher levels of performance of non-functional requirements. However, if the business decides that "quick and cheap" is what they want, it is then invidious of the business to complain later that the resultant application system is expensive to operate or slow to adapt to changing business demands.

The challenge for enterprise architects is to ensure that the consequences of investment decisions are clearly presented and understood by all the appropriate stakeholders – not just the managers who are seeking the solution, but the more senior levels of the management team. In many cases, the pressure on a project team is to find a way of delivering the required functionality for less cost in the initial project. This downward pressure on the cost of the project tends to result in less attention being paid to the non-functional requirements which affect the operational costs and the change management costs after the "go live" date [12, p. 11].

4.4 Setting up the non-functional requirements at an appropriate level

In order to set the non-functional requirements for an application at an appropriate level to support the business it is necessary to have some view of how the application will evolve over its life. If an application is likely to be very stable then there is little point in taking extraordinary efforts to make the application highly adaptable. If the application will only ever be used within one country then there is little value in investing in high localizability levels in the design.

One of the fundamental determinants of the lifetime total cost of ownership is the overall volatility of the application. This volatility has two constituent components: functional volatility and scope creep. Functional volatility examines the extent to which the business rules or data are likely to change throughout the life of the application. Scope creep examines the way in which the application is likely to extend its reach though the addition of new functionality or integration with other applications. The Maintainability non-functional requirements for the application should be set at a level that is appropriate to the expected future volatility [12, p. 12].

Maintenance is discipline in information and communications technology which is frequently underfunded and undervalued. Enterprise architects should see maintenance as an essential component of any well-run information technology department and should seek to ensure that all aspects of the maintenance task are properly resourced and valued. In particular, it is important to look at the ways in which the maintenance function can modify the application to change the characteristics of the system through perfective maintenance. This is particularly important where the investment team have taken a decision to downgrade key non-functional requirements in order to save funds and time in the initial implementation, but have accepted that there will be a need to reinstate the original higher levels of performance in order to be able to get maximum value from the application throughout its life.

The problem with these decisions is that all too often the intention gets lost as maintenance budgets are trimmed and the demands of corrective and adaptive maintenance overwhelm the team.

4.5 Post-implementation analysis

Most well-run organizations now understand the value of post-implementation benefits analysis. This important activity needs to be matched by continuous evaluation of the non-functional performance of the application. All the non-functional requirements should have appropriate measures and regular reporting mechanisms that assess current performance against the designed objectives and against current needs. It may be that some non-functional requirements have been set at too high a level, and these could be allowed to be downgraded. Others will be behaving as designed, but the needs of the business have changed and now higher levels of performance may be required. All this data should be used to refine the way in which non-functional requirements are set and managed for future projects.

Modifying the design attributes of an application is generally an expensive and difficult task. If it is to be undertaken then it is best done generally during the first third of the expected life cycle of the application. Eventually, as an application is being prepared for decommissioning, there will be a period when no changes should be contemplated. The investment profile for an application should recognize that the initial project to create the application is just the beginning of the investment, not the end.

Since enterprise architecture is a comprehensive framework used to manage and align an organizations IT assets, people, operations, and projects with its operational characteristics, it defines how information and technology will support the business operations and provide benefit for the business. Well-documented and well understood enterprise architecture enables the organization to respond quickly to changes in the environment in which the organization operates. It serves as a ready reference that enables the organization to assess the impact of the changes on each of the enterprise architecture components. It also ensures the components continue to operate smoothly through the changes [14]. Enterprise architects should work with project sponsors to find ways of financing the essential post-project activities that will be necessary to make the application a valuable asset for as long as possible.

5 Conclusion

This paper briefly reviews some typical requirements and highlights potential contributions that application non-functional properties can make to enterprise architecture. In this connection pertinence of non-functional requirements for enterprise architecture is especially emphasized.

Non-functional attributes describe the nature, mechanism, or context of the application execution (how and under which conditions is the application doing). Hence, we examine application non-functional attributes impact on enterprise architecture and find them substantial for enterprises, suggesting their long-term implications for enterprise architectures. Modeling non-functional requirements is the critical step for achieving successful realization of complex application development projects.

According to this we consider application non-functional requirements as an architectural element affecting IT as well as organizational issues and thus enterprise architecture appliance. Therefore application non-functional requirements raise the question for interdependencies between IT and organization. Application non-functional requirements are major component in complex information and technology systems which have a significant influence on business processes. They are of great importance through all steps of application development, starting with requirement specification, over non-functional requirements modeling, to their implementation.

Successful application development depends tightly on many other requirements beside functional ones. Very late and missing specification of security attributes in application development cause that developers in the later development phases must manage different security requirements and configurations ad-hoc and manually, which negatively affects development costs. We find that there is a considerable gap between the importance of application non-functional requirements in EA and the way they are used in enterprises today. Consequently, application development must support specification of non-functional requirements in the design phase and traceability i.e. mapping of non-functional requirements to software solutions within enterprise architecture.

Cost reduction alone does not provide a compelling justification for an enterprise architecture program. With the business focus shifting to agility and competitive advantage, enterprise architects need to focus on improving the availability of management information, supporting business growth and enabling strategic business development.

An application strategy represents a plan to achieve a business outcome via the use of technology, where the result is recognized to be the optimum balance between the conflicting requirements of stakeholders. It is a long-term view that enables companies to make short-term technology decisions. An effective application strategy enables enterprises to balance stakeholders' (internal and external) needs to decide on a path for technology investments. The more often this plan is in place and followed, the better prepared a enterprise will be for the future.

The enterprise architecture value proposition should be focused on the major strategic drivers articulated in the business strategy and the way EA program will support the business in achieving its strategic goals by transparently linking the architectural direction and requirements to those strategic goals. This results in an EA program that not only serves the business better by focusing on what is important to the business, but also elicits much stronger support and engagement from the business, which significantly increases the program chances of success.

In this regard, the paper suggests importance of non-functional requirements established within enterprise architecture and identifies is as component that prevents loss, leverages organizations functioning and enhances business to continue to maintain a level of sustainable competitive advantage, as well as to exploit business opportunities.

References

[1] Chief Information Officers (CIO) Council: **Federal Enterprise Architecture Framework**, USA, 1999.

[2] Cicmil S., Hodgson D.E.: **New possibilities for project management theory: a critical engagement**, Project Management Journal, 37(3): 111-122., Project Management Institute (PMI), USA, 2006.

[3] Cooper R.G.: **Stage-gate systems: a new tool for managing new products**, Business Horizons, 33(3): 44-54 SciVerse ScienceDirect, NL, 1990.

[4] Enterprise Architecture Research Forum (EARF): **EARF definition for EA**, available at `http://earf.meraka.org.za/earfhome/defining-ea`, Accessed: 17th January 2011.

[5] Giachetti R.E.: **Design of Enterprise Systems, Theory, Architecture, and Methods**, CRC Press, Boca Raton, FL, USA, 2010.

[6] Haine P.: **Managing Processes as Services**, Business Process Management Conference Europe 2005, BPM Group, London, UK, 2005.

[7] Hodgson D., Cicmil, S.: **The politics of standards in modern management: making 'the project' a reality**, Journal of Management Studies, 44(3): 431-450., Wiley, USA, 2007.

[8] International Standardization Organization (ISO): **ISO 9126 Software Quality Characteristics**, available at http://www.sqa.net/iso9126.html, Accessed January 11th 2011.

[9] Kolehmainen S.: **The dynamics of control and commitment in IT firms**, Information Society and the Workplace: Spaces, Boundaries and Agency, Routledge, UK, 2004.

[10] Krogstie J., Sølvberg A.: **Information systems engineering - Conceptual modeling in a quality perspective,** Kompendiumforlaget, Trondheim, Norway, 2003.

[11] Kyte A.: **A Framework for the Lifetime Total Cost of Ownership of an Application**, Research, Gartner, Inc., 2010.

[12] Kyte A.: **Applications — Functional Requirements are Only Half the Story**, Gartner Briefing, Gartner, Inc., Zagreb, 2010.

[13] Lapkin A., Papegaaij B.: **Business Strategy Defines Enterprise Architecture Value**, Research, Gartner, Inc., USA, 2010.

[14] National Institutes of Health (NIH) Enterprise Architecture: **What is enterprise architecture?**, available at `http://enterprisearchitecture.nih.gov/About/What/`, Accessed: 18th January 2011.

[15] Nelson D.S., Sribar V.T., Kyte A.: **Signs Indicate a Train Wreck is Coming, Unless You Modernize IT**, Research, Gartner, Inc., 2008.

[16] Robertson B.: **IT Architecture Is Not Enterprise Architecture**, Research, Gartner, Inc., USA, 2010.

[17] Storey J., Salaman G., Platman K.: **Living with enterprise in an enterprise economy: freelance and contract workers in the media**, Human Relations, 58(8): 1033-1054., SAGE, UK, 2005.

[18] The Open Group: **The Open Group Architectural Framework (TOGAF) 8.1.1**, available at `http://www.opengroup.org/architecture/togaf8-doc/arch/toc.html`, Accessed: 14th January 2011.

[19] Thomas R., Davies A.: **Theorizing the micro-politics of resistance: new public management and managerial identities in the UK public service**, Organization Studies, 26(5): 683-706., SAGE, UK, 2005.

[20] US Department of the Treasury Chief Information Officer Council: **Treasury Enterprise Architecture Framework**, Version 1, USA, 2000.

[21] Weill P., MIT Center for Information Systems Research: **Enterprise Architecture**, The Sixth e-Business Conference, Barcelona, Spain, 2007.