International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:6, 2012

Architecture Exception Governance

Ondruska Marek

***Abstract*—**The article presents the whole model of IS/IT architecture exception governance. As first, the assumptions of presented model are set. As next, there is defined a generic governance model that serves as a basis for the architecture exception governance. The architecture exception definition and its attributes follow. The model respects well known approaches to the area that are described in the text, but it adopts higher granularity in description and expands the process view with all the next necessary governance components as roles, principles and policies, tools to enable the implementation of the model into organizations. The architecture exception process is decomposed into a set of processes related to the architecture exception lifecycle consisting of set of phases and architecture exception states. Finally, there is information about my future research related to this area.

***Keywords*—**Architecture, dispensation, exception, governance, model

I. Introduction

A. Article Assumptions

THERE are assumptions related to the architecture exception governance model that must be articulated before the model is presented.

The list of assumptions follows:

1) Projects are the only vehicles delivering changes in the organization.
2) The model is independent on architecture description [1] language.
3) The definition of architecture is adopted from Framework Togaf [5].
4) The focus of architecture governance is limited only on IS/IT architecture.
5) The incoming architecture exception (AE) is considered as final. It means that the consideration if a deviance from architecture and rules is architecture exception is in responsibility of architecture governance – architecture compliance processes.

B. Importance of AE Approach – Why to Do it

Nowadays, architecture governance is not new activity performed by organizations. Actually, the importance of the activity is growing because of increasing complexity of organizations, especially when IT systems are considered. Many Frameworks focused on architecture governance embed model of transition between AS-IS architecture and target TO-BE architecture [5]. Practice shows that projects that implement different organization's requirements cannot always comply with the defined TO-BE architecture and gaps in architecture transition emerge.

M. Ondruska, is with Faculty of Informatics and Statistics, University of Economic, Prague, 1360 67 Czech Republic. (e-mail: marekon@gmail.com).

Actually, this occurs very often in projects. It is necessary to govern and manage these discrepancies to eliminate them and achieve the target architecture. Later, there is defined special term for these discrepancies named architecture exception. Practice shows, that if these discrepancies are not governed well, these discrepancies change inevitably the target architecture, therefore the target architecture is never achieved. It is necessary to govern the discrepancies in such a way that they are eliminated in reasonable time horizon. The solution time horizon must be definite.

C. Known approaches to architecture exceptions

The area of architecture exceptions / discrepancies is not new. There are approaches that address and give a basic frame how to solve that area. Let us briefly summarize chosen well known approaches.

1) Togaf – Architecture Governance – Dispensation Process

"A Compliance Assessment can be rejected where the subject area (design, operational, service level, or technology) are not compliant.

In this case the subject area can:

i. Be adjusted or realigned in order to meet the compliance requirements
ii. Request a dispensation

Where a Compliance Assessment is rejected, an alternate route to meeting interim conformance is provided through dispensations. These are granted for a given time period and set of identified service and operational criteria that must be enforced during the lifespan of the dispensation. Dispensations are not granted indefinitely, but are used as a mechanism to ensure that service levels and operational levels are met while providing a level of flexibility in their implementation and timing. The time-bound nature of dispensations ensures that they are a major trigger in the compliance cycle." [5]

2) SOA Governance Framework – Dispensation Process

"The dispensation process is the exception and appeals process that allows a project or application team to appeal noncompliance to established processes, standards, policies and guidelines as defined within the governance regimen. Examples include service funding, service ownership, service identification, etc. The result would be a granted exception." [4]

D. Benefits of the Presented Model

The benefits of the article and the presented architecture exception governance model come from definition of an overall model for governing of architecture exceptions. It means that it is not only focused on processes, but defines all the next necessary components of governance model. The model is not limited on dispensation process, but it defines full set of processes related and needed to govern architecture exceptions in a relationship with architecture exception lifecycle. Finally, the granularity adopted for the model is higher than for the above known approaches.

E. Used Methodology

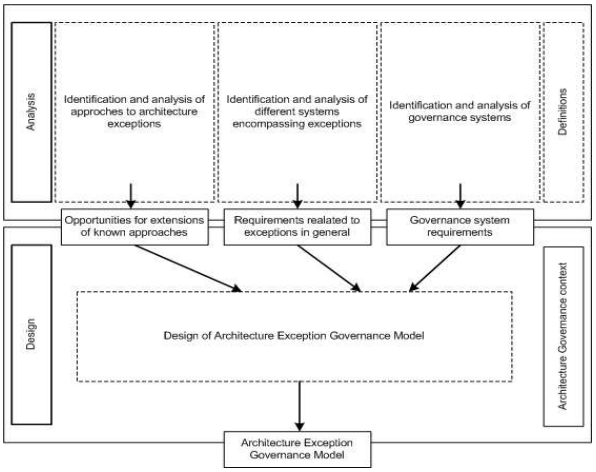

Fig. 1 Design Methodology for the Model

Fig. 1 shows the proposed methodology that has been used to design the model of architecture exception governance. As next, the methodology is described, but it is not possible to present all the aspects in high detail due to limited scope of the article. Let us focus on the main aspects of the methodology.

As first, the methodology is decomposed into two main phases. The Analysis Phase encompasses all the activities related to analysis as identification and analysis of different known approaches to architecture exceptions and the others depicted. The Design Phase is primarily about the design of the model base on the integration of knowledge gained in the Analysis Phase.

Between these two phases there are identified interfaces in the form of requirements that had to be followed during the Design Phase. It means that the model is implementation of these requirements

II. Governance

As for any governance system, let us set the main components of such system. The components are typically considered: processes, roles, organization structure, principles and procedures, tools. The definition respects chosen aspects of approaches described in [2], [3] and [6].

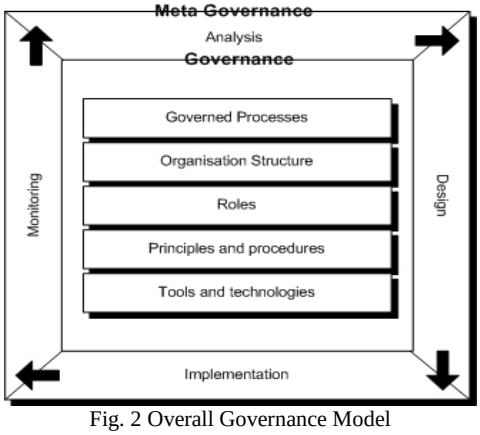

Fig. 2 Overall Governance Model

Meta Governance: The Governance system must be developed in time, it is not a static system, but it is a dynamic one. The Meta governance system ensures the improvement of the governance in the organization in time.

The Governance: is a system of processes, roles, principles and policies and tools that are implemented to ensure that the organization (governed area) is governed in a way that the set organization's goals and strategy are met.

A. Governance Processes

1) Analysis - analyzing of the current governance model and its implementation from the issues and problems point of view.
2) Design - redesigning of the current governance model to better address the governance needs.
3) Implementation (includes training) - implementation of new version of the model.
4) Monitoring - monitoring of the implemented governance model, looking for issues and problems to be solved.

B. Governed Processes

Governed processes are specific processes of governed area (architecture exceptions). They are not generic like the governance processes defined above. They are discussed later as part of the AE Governance Model.

C. Roles, Organization Model, Artifacts and Tools

Roles, Organization model, Artifacts and Tools as the next components of Governance model must be defined.

1) Roles - the responsibility framework for different process tasks must be set.
2) Organization model - Assigning of concrete people, organization units to defined logical roles (the previous component called "Roles"). This component is typically setup in the implementation of governance. It is not covered by the article.
3) Artifacts - principles and policies encapsulated in different forms (patterns, models, guidelines and others).
4) Tools - mainly the application support tools.

III. Architecture Exception

A. Architecture Exception Definition

Architecture Exception is an entity (type), that gives a designed solution of the project into relationship with defined TO-BE architecture. Simultaneously, it must be true, that the relationship is evaluated as discrepancy. The discrepancy emerges when there exists at least one inconsistency between the designed solution and the TO-BE architecture on the defined level of architecture description detail. Instance of architecture exception means a concrete occurrence or emergence of architecture exception. Finally, the exception has the attributes depicted on Fig. 3.

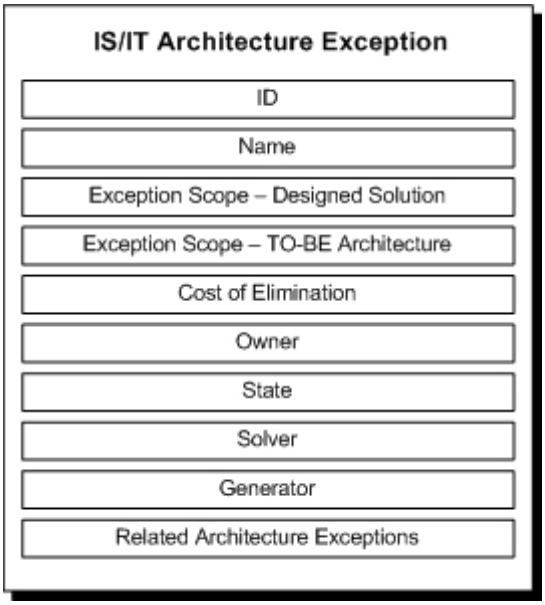

Fig. 3 Architecture exception entity

B. AE Attributes Definition

1) ID – unambiguous identification of architecture exception.
2) Name – assigned name to architecture exception.
3) Exception scope – designed solution – reference or description of the solution that is not in compliance with TO-BE architecture.
4) Exception scope – TO-BE architecture – reference or description of TO-BE architecture that is violated with the solution.
5) Costs of elimination – calculated costs for elimination of architecture exception.
6) Owner – assigned responsible person to architecture exception.
7) State – actual state of the architecture exception.
8) Solver – assigned solver/project to elimination of architecture exception.
9) Generator – identification of project that has generated the exception.

C. AE Lifecycle Phases and States

Now, let us focus on the architecture exception lifecycle Fig.4. The goal is to set a model of phases and states that constraint the lifecycle of architecture exception.

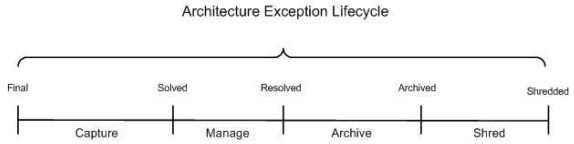

Fig. 4 Architecture exception lifecycle

1) AE States

1) Final – the final state means that the scope of architecture exception is fixed before the exception is being processed with the Architecture Exception Governance.
2) Solved – when architecture exception is being solved
3) Resolved – the state when the elimination of architecture exception by project is accepted by architecture exception owner.
4) Archived – An architecture Exception is in the archived state when has been resolved but it is necessary to keep record about it, because of for example audit purposes.
5) Shredded – This state means that the architecture exception is deleted and documentation is no longer accessible.

2) AE Phases

1) Capture – The capture phase covers all the activities necessary to start solving the architecture exception from register to assignment of the architecture exception owner.
2) Manage – The manage phase is about architecture exception solving. The goal of this phase is to eliminate the architecture exception through suitable project.
3) Archive – The archive phase consists of processes related to archiving.
4) Shred – The shred phase covers shredding of architecture exceptions, when they are no longer needed for any purposes.

IV. Architecture Exception Governance

A. Processes (Governed)

As first, let us define the process model of architecture exception governance. The processes are categorized in compliance with architecture exception lifecycle. It means the processes are grouped or mapped to the phases of the lifecycle:

Capture

1) Managing and registration of incoming architecture exceptions
2) Assignment of owner to architecture exception
3) Costs evaluation of architecture exception elimination
4) Finalization of architecture exception registration

Manage

5) Searching and assignment of suitable architecture exception solver
6) Provisioning of finance and budget to solver to architecture exception elimination
7) Monitoring of architecture exception elimination development
8) Acceptation of the elimination / solution of architecture exception

Archive

9) Architecture exception archiving process
10) Provide archived architecture exception documentation

Shred:

11) Shredding of architecture exceptions with expired archive time period

Others / cross processes

12) Restructuring / redefinition of architecture exceptions in portfolio when target architecture is changed
13) Assure that architecture standards need for architecture exception assessment are defined

14) Consolidation of Architecture portfolio – merging and splitting of architecture exceptions

The governance model block is a set consisting of process, related roles, related principles and policies, related tolls. Formally, the block = {process, roles, principles and policies, tools} for given process that setups the context. Fig. 5 depicts the governance blocks of IS/IT architecture exception governance related to the project and IS/IT architecture governance.

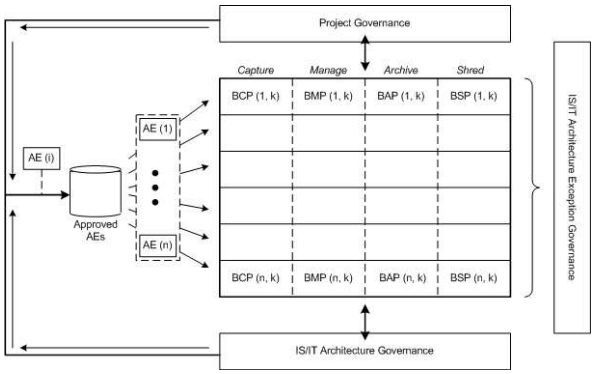

Fig. 5 Architecture, project and exception governance

How to describe the governance model block:

Identification of block – example BCP (i, k), where i – identification of the architecture exception, k – is identification of process in the phase.

Required attributes of each block are as follows:

1) Process name
2) Process goal
3) Process trigger
4) Incomes
5) Activities
6) Outputs
7) Roles and Responsibilities
8) Principles and policies
9) Tools

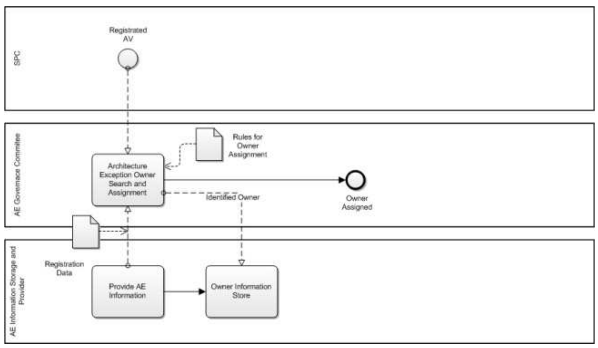

Fig. 6 BCP02 - Assignment of owner to architecture exception – example

Block Identification: BCP02 - Assignment of owner to architecture exception, Fig 6.:

1) Process name: Assignment of owner to the architecture exception
2) Process goal: Find and assign an owner to the architecture exception
3) Process trigger: Registration of new incoming architecture exception
4) Incomes: Documentation of the architecture exception Activities: AE owner search and assignment, provide AE information
5) Outputs: Identified owner of architecture exception
6) Roles: SPC (Single Point of Contact), AE Gov. Committee, AE Information Storage and Provider
7) Principles and policies: Rules for owner assignment, requirements on registration data
8) Tools: Architecture exception portfolio (application)

B. Principles and Policies

1) Architecture exception documentation – templates with requested areas of information about architecture exception.
2) Architecture exception owner assignment – set of principles supporting the assignment of owner to architecture exception.
3) Architecture exception evaluation – evaluation methodology and rules how to evaluate an architecture exception.
4) How to find the right architecture exception solver – defined rules or criteria how to find a project that is capable to solve the specified architecture exception.
5) Budget calculation and assignment to solver rules.
6) Architecture exception elimination acceptance criteria.
7) Architecture exception archiving rules – shredding periods, …

C. Roles

1) Generator of architecture exception
2) Single point of contact
3) Architecture exception owner
4) Architecture exception evaluator
5) Architecture exception solver
6) Architecture exception governance committee
7) Provider of architecture exception information
8) Finance provider to architecture exception elimination

D. Tools

1) Architecture exceptions portfolio
2) Architecture exception evaluator

V. WHY GOVERNANCE NOT ONLY PROCESSES

The discussion why it is necessary to build whole governance model and not only processes is based on definition of governance. The goal is to show, that there exists components that must be included in the model, but they are not processes.

The definition of governance says that there are five components of governance: processes, roles, principles and policies, organization structure and finally tools.

As it was discussed, it is necessary to setup roles and responsibilities like architecture exception owner that is crucial role in the architecture exception governance. As next, there are specific principles and policies that must be adopted like architecture exception cost of elimination evaluation methodology. Finally, processes related to architecture exception governance should be supported with tools that enable efficient performance of the processes.

As a result, it is important to define not only processes as architecture exceptions or dispensation process, but even all the next components of governance model.

VI. Conclusion

The whole architecture exception governance model has been presented. There are areas that were not discussed as detail design of architecture exception governance processes, implementation of the model into organizations, or integration of the model into governance structures adopted by organizations. Let us emphasize project governance and architecture governance that must be integrated with the architecture exception governance model. All the mentioned areas are my future directions of research in the area of architecture exceptions.

References

[1] ISO. (2000). IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. Retrieved from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=875998

[2] Marks, E.A. (2008) Service-Oriented Architecture Governance for the Services Driven Enterprises, John Wiley & Sons.

[3] Oracle. SOA Governance: Framework and Best Practices. Retrieved from http://www.oracle.com/us/technologies/soa/oracle-soa-governance-best-practice-066427.pdf

[4] The Open Group. (2009) SOA Governance Framework. Retrieved from http://www.opengroup.org/projects/soa-governance/uploads/40/19263/SOA_Governance_Architecture_v2.4.pdf

[5] The Open Group. (2011) TOGAF Version 9.1. The Open Group. Retrieved from http://pubs.opengroup.org/architecture/togaf9-doc/arch/

[6] Weill P., Ross J.W. (2004). IT Governance: How Top Performers Manage IT Decision Rights for Superior Results. Harvard Business School Press.