International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-3 Issue-2, December 2013

Architecture Exception Governance Reference Model - Togaf Framework Extension

Marek Ondruška

***Abstract** – The paper proposes an extension for architecture framework Togaf. In particular, it addresses on architecture exceptions and their governance. The article covers a reference model for architecture exception governance (AEG RM) and the way how to integrate it with Togaf Framework. As part of AEG RM there is defined an entity called architecture exception with its main attributes. AEG RM defines all the processes necessary for architecture exception governance, roles and responsibilities, principles a procedures and supporting tools. There is one chapter dedicated only for integration of Togaf and AEG RM. As summary, the paper has two main focuses. The first one is to present architecture exception governance reference model. The second is to integrate the reference model with Togaf architecture Framework. The article requires at least basic knowledge in architecture governance and architecture framework Togaf.*

***Index Terms**—Architecture, Exception, Governance, Reference Model, Togaf.*

I. INTRODUCTION

This paper introduces an extension of architecture framework Togaf [5]. In particular, it focuses on architecture dispensations (exceptions) and their governance. Togaf itself includes "architecture compliance" and "architecture dispensation" processes in architecture governance section to cover that area. The following text respects this approach and develops it into more detail and extent. Actually this paper introduces an entire **architecture exception governance reference model** that can help to implement Togaf into organizations of different kind. The model is designed to be a reference model for this field of interest, so it is possible to use it beyond the Togaf, but in this article the focus is concentrated on integration with Togaf. AEG RM is not specific for concrete business field and is general enough to cover all of them.

II. ARCHITECTURE EXCEPTION

As first, let us define the core entity around which the governance model is designed. **Architecture exception** (AE) is an entity that brings together designed project solution with existing target (TO-BE) architecture. Simultaneously, there must be true that this relationship is evaluated as inconsistency. The inconsistency emerges when there is at least one deviation between the designed solution and the target architecture at defined level of architecture description [1] detail. An instance of architecture exception is one particular occurrence of architecture exception entity.

As target architecture it is meant the future vision of architecture and all the principles and procedures captured in different architecture artifacts as models, patterns and so on related to the future architecture. Sometimes, architecture exception is called architecture dispensation, but it is the same.

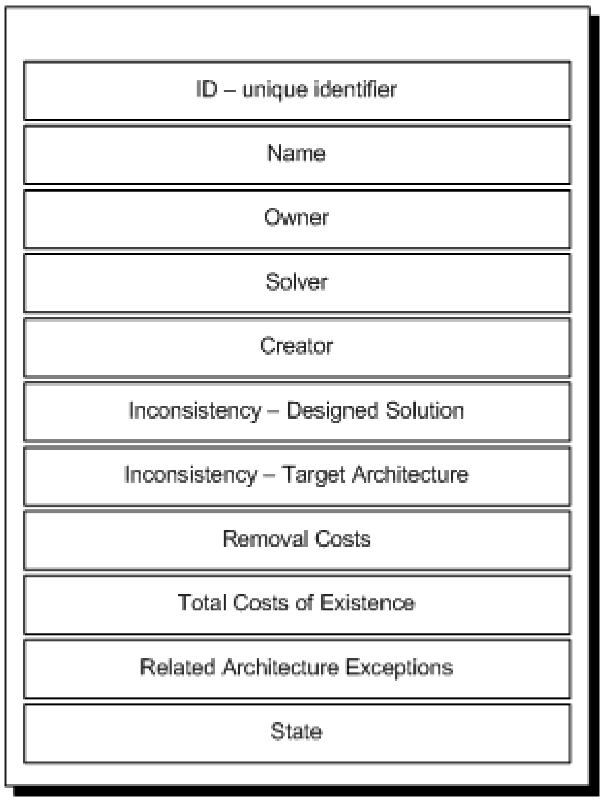
ID – unique identifier
Name
Owner
Solver
Creator
Inconsistency – Designed Solution
Inconsistency – Target Architecture
Removal Costs
Total Costs of Existence
Related Architecture Exceptions
State

Fig 1. Architecture Exception Attributes

A. AE Attributes

As next, a set of the most important attributes of architecture exception is defined. The set represents a minimal set see Fig. 1, but it can be extended if necessary during implementation into and tailoring for specific environment.

1) **ID** - is a unique identifier of an architecture exception instance.
2) **Name** - is a short name that refers to the content of architecture exception.
3) **Description** - is a short text that accurately explains the architecture exception.
4) **Owner** - is a role that is responsible for instance/instances of architecture exceptions from assignment to acceptance of its removal (during the AE lifecycle).

Manuscript published on 30 December 2013.
***** Correspondence Author (s)
Marek Ondruška*, System Analysis Department, University of Economics in Prague, Prague, Czech Republic.

5) **Solver** - is a role that is responsible for architecture exception removal (typically project).
6) **Creator** - is a role that caused an architecture exception emergence (typically other project).
7) **Inconsistency - designed solution** - is a reference to a part of solution design where the inconsistency exists.
8) **Inconsistency - target architecture** - is a reference to a part of model, standard, pattern or other architecture artifact that is violated by the inconsistency - designed solution.
9) **Removal Costs** - represent costs for architecture exception removal (can change in time, therefore it must be updated, see support processes for more detail).
10) **Total Costs of Existence** = removal costs + sunken costs
11) **Related Architecture Exceptions** - are references to other instances of architecture exception entity that are in some way related to the particular exception instance.
12) **State** - is the state of architecture exception at the particular point in time.

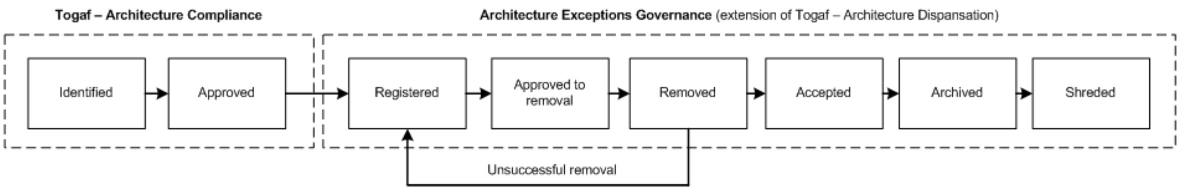
Togaf – Architecture Compliance
Architecture Exceptions Governance (extension of Togaf – Architecture Dispansation)
Identified
Approved
Registered
Approved to removal
Removed
Accepted
Archived
Shreded
Unsuccessful removal

Fig 2. Architecture Exception States

B. AE Lifecycle States

There is a relationship between AEG RM processes and AE states. It means the processes change the state of architecture exception. Relationships among the states are depicted on Fig 2. Now, let us define all the states.

1) **Identified** - architecture exception comes into this state after architects identify existence of this architecture exception. Let us call it (draft or candidate architecture exception).
2) **Approved** - architecture exception is approved when architecture bodies (for example architecture committees) approves candidate architecture exception as architecture exception.
3) **Registered** - is the first state when an architecture exception is governed by architecture exception governance. The previous states where related to compliance process rather than dispensation process.
4) **Approved to removal** - when architecture committee in collaboration with owner decides that there is a particular instance of architecture exception that should be removed. For example there is a suitable project for its removal or the sunken costs are too high and the exception must be removed .
5) **Removed** - this state comes when the solver proclaims the architecture exception is removed.
6) **Accepted** - is the state when owner and architecture bodies accept the removal of architecture exception removal.
7) **Archived** - there exist reasons why to store information about architecture exceptions after it was removed. Audit purposes are good example why to store the information for defined time period. After removal, architecture exception is in archived state.
8) **Shredded** - this is the state when architecture exception is shredded. This does not necessary means the record about architecture exception is shredded too, but all the documentation related to architecture exception is deleted from evidence.

III. GOVERNANCE

There are many definitions addressing the area of governance [2], [3], [5], [6]. To precisely analyze what governance is about is beyond this article, but it is necessary to setup a basic notion. This article defines governance as all the processes, roles and responsibilities, principles and procedures and tools that are necessary to govern the lifecycle of architecture exception. This notion of governance is adopted later in the paper when AEG RM is being designed.

IV. PROCESSES

Architecture exception governance processes are grouped into five domains related to architecture exception lifecycle - capture, manage, remove and archive see Fig 3. The support domain of processes is not related to exactly one domain, but it often goes cross domains. Identification and acceptance of an architecture exception instance is done before the exception is governed with the architecture exception governance. Actually, this is done in Togaf – process compliance. This process should be extended with principles that define what exception is and what is not. Architecture exception governance is more detailed developed what Togaf calls "dispensation process". Architecture exception governance goes into higher detail and defines all the parts of governance: all the processes, roles and responsibilities, principles and policies and tools.

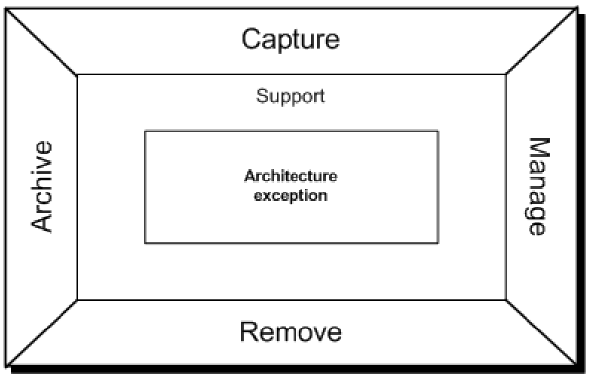

Fig 3. AEG Process Domains

A. Capture

Processes related to Capture phase are mainly about registration of new incoming architecture exceptions into evidence (architecture portfolio), after they were accepted in compliance process.

1) **Registration of emerged and accepted architecture exception** - provides initial AE registration service. New approved AE is registered with the required metadata and documentation into the evidence (Architecture portfolio).
2) **Assignment of architecture exception owner** - finds the suitable owner of AE who is capable of AE removal governance.
3) **Cost assessment of architecture exception removal** - assess the costs of architecture exception removal in future project/s.
4) **Finalization of registration** - provides final check and completion of AE registration.

B. Manage

Processes related to Manage phase are about considering whether an architecture exception instance should be handed over for removal or not.

1) **Decision making about AE removal** - is decision making process focused on starting the removal phase of AE instance/s at right time and with suitable solver.
2) **Search for suitable solver of the AE** - is complement process to the previous one in a way that it brings the suitable solver.

C. Remove

Processes related to Remove phase are the architecture processes that help the solver (project) with the removal and govern it. How project (solver) functions is a subject of standard project methodologies.

1) **Provide documentation of AE** - is about AE documentation provisioning from the owner to the solver.
2) **Design and accept of AE removal way** - adds the AE solution into the concept of solution design. Solution Design must remove the AE.
3) **Removal progress monitoring** - monitors the removal process to support the removal achievement. If something goes wrong, this process can trigger escalation procedures and other methods to resolve the unpleasant situation .
4) **Acceptance of final removal** - final acceptance of AE removal.

D. Archive

Processes related to Archive phase address the area of storing the architecture exceptions in evidence after they were resolved/removed for audit purposes as example.

1) **AE archiving** - archive AE for defined archival period.
2) **Provide information about AE** - provide information about AE during archive phase on demand.
3) **Discard AE from AE evidence** - delete the AE from evidence.

E. Support

Processes related to Support phase are not the core processes that govern architecture exception, but they are those processes that support the core processes or solve some specific areas not directly in relationship with exception and its lifecycle or they cross more domains than one.

1) **Update of architecture exceptions because of target architecture change** - when target architecture is changed, it is necessary to validate all the impacted exceptions and update them
2) **Request for missing/non-existing architecture standards** - if there are missing architecture artifacts for some architecture area, there should exist a process enabling to request for delivery of such artifacts.
3) **Architecture portfolio consolidation** - this process is about merging and splitting of architecture exceptions in portfolio. In some cases, there is reasonable to remove not only one, but more grouped architecture exceptions in the same project.
4) **Risks escalation** - One of the standard processes in organizations is risk management. There must be a way how to generate architecture exception specific risks and let them be managed by risk management.
5) **Architecture exception governance improvement process** - is process of architecture exception governance continual improvement. It means it covers such activities as analysis, design, implement, monitor of architecture exception governance.
6) **Budget and finance management for AE removal** - this process addresses budget and finance management to enable AE removals in projects.
7) **Update of removal costs and sunken costs of architecture exceptions in portfolio** - these two financial parameters must be regularly checked because they are not stable in many cases in time and should be updated to reflect the real situation.

V. Roles

Let us define the main roles of architecture exception governance that are related to the processes defined above.

1) **AE Owner** - is responsible for assigned set of architecture exceptions. In particular, the role should push the exceptions through the lifecycle of architecture exception.
2) **AE Solver** - is responsible for removal of architecture exception
3) **AE Creator** - is responsible for architecture exception existence
4) **AE SPC and Portfolio Manager** - is responsible for gathering all the incoming architecture exceptions, for their proper evidence.
5) **AE Costs Evaluator** - is responsible for evaluation of AE costs of removal and sunken costs
6) **AEG Owner** - is responsible for architecture exception governance improvement
7) **AE Removal Way Designer** - is responsible for design of removal way of architecture exception
8) **AE Decision and Acceptance Committees** - are responsible mainly for acceptance of AE removal and acceptance of chosen solver and way of removal.

VI. Principles and Policies

There are many principles and policies related to architecture exception governance. Let us mention some of the main ones.

There must be principles and policies for calculation of removal costs or there must be principles and policies for architecture exception documentation. As next, principles and policies covering different criteria that are monitored for separate exceptions or there must be principles and policies that help make a decision when and which architecture exceptions should be hand over for removal.

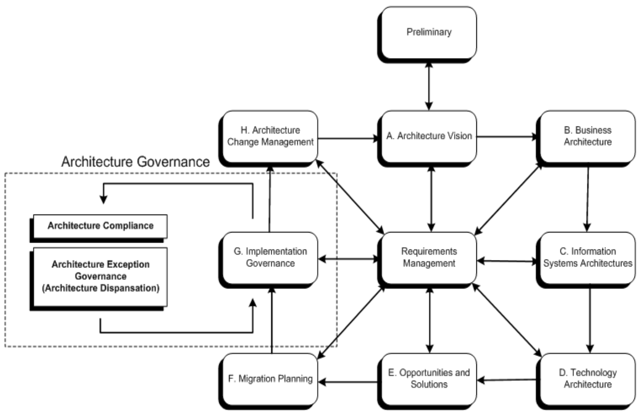
Preliminary
H. Architecture Change Management
A. Architecture Vision
B. Business Architecture
Architecture Governance
Architecture Compliance
Architecture Exception Governance (Architecture Dispansation)
G. Implementation Governance
Requirements Management
C. Information Systems Architectures
F. Migration Planning
E. Opportunities and Solutions
D. Technology Architecture

Fig 4. Relationship of Togaf ADM and AEG RM

VII. TOOLS

Architecture exception portfolio - is a tool that supports most of the above mentioned processes with use cases as “create new architecture exception”, “read/display architecture exception”, “update architecture exception”, “delete architecture exception” and display different reports and views of exceptions. The main purpose of architecture portfolio is evidence of architecture exceptions with their documentation.

VIII. INTEGRATION WITH TOGAF

Togaf architecture framework addresses architecture exceptions area with definition of two processes "compliance" and "dispensation". These processes are part of architecture governance. Architecture governance is applied in the implementation governance phase of ADM cycle. This is the first place of architecture exception governance integration with Togaf. There is no difference if we consider only a subset of architectures addressed by Togaf as business architecture, application architecture, data architecture and technology architecture. This means the integration can be done for example only for IS/IT architecture or for the entire enterprise architecture. For the purpose of this article, the enterprise architecture is considered.

The Togaf – compliance process is about check and validation of designed solution against defined standards and different architecture artifacts. As a result, the solution can be compliant with the standards. Otherwise, the project must change the design to be compliant or can request a dispensation. This is the reason, why Togaf defines dispensation process. The compliance process must be enhanced with criteria that help identify when a deviation from standards is an exception and when it is not. It means that organization can define certain criteria that identify exception, but do not have to. That is the case all the deviations that are not resolved by project are automatically considered as exceptions. This is an implementation detail that must be resolved in accordance with specific implementation environment (organization). As a recommendation, it is reasonable to setup these criteria, because not all the deviations make sense to track and govern as exceptions. Even the number of them would be too high. It would lead to high costs for operation of AEG.

Architecture exception governance reference model promotes the dispensation process on governance level and extends it in detail and extent. For example, the reference model suggests that there is not only one process, but rather there is a process model as was described before. The architecture governance and therefore dispensation process is integrated with ADM - implementation governance phase. As it was described, AEG RM is an extension of architecture dispensation process that is why this integration is reusable even for integration AEG RM with Togaf. In this case, the integration of AEG RM is particularly with architecture development method (ADM) see Fig 4.

Second place of Togaf and architecture exception governance integration is through part III – reference models. As described before, architecture exception governance is designed as reference model and can be used as more detailed developed dispensation process. So, it is possible to add the reference model among the others that Togaf references see Fig 5.

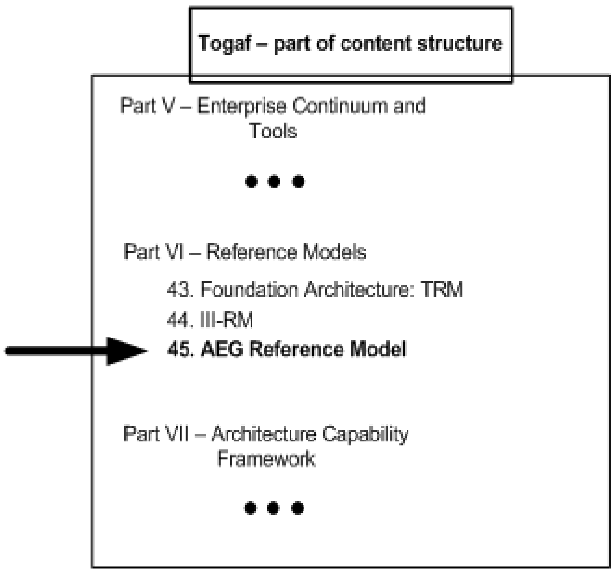
Togaf – part of content structure
Part V – Enterprise Continuum and Tools
• • •
Part VI – Reference Models
43. Foundation Architecture: TRM
44. III-RM
45. AEG Reference Model
Part VII – Architecture Capability Framework
• • •

Fig 5. Togaf Reference Models - AEG RM

IX. CONCLUSION

AEG RM as was described in the paper is a result of my long term research in the area of architecture governance. Nowadays, the reference model is being tailored and implemented in banking sector organization. The goal is to get feedback and then to fine-tune the reference model to be useful for all the organizations that need to implement system for architecture exception governance. Organizations that use Togaf Framework for architecture governance can benefit from section called integration with Togaf during implementation of the reference model for architecture exception governance.

REFERENCES

[1] ISO. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. 2000. Retrieved from http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=875998
[2] Marks, E.A. Service-Oriented Architecture Governance for the Services Driven Enterprises, John Wiley & Sons. 2008.
[3] Oracle. SOA Governance: Framework and Best Practices. Retrieved fromhttp://www.oracle.com/us/technologies/soa/oracle-soa-governance-best-practice-066427.pdf
[4] The Open Group. SOA Governance Framework. 2009. Retrieved from http://www.opengroup.org/projects/soa-governance/uploads/40/19263/SOA_Governance_Architecture_v2.4.pdf
[5] The Open Group. TOGAF Version 9.1. The Open Group. Retrieved from http://pubs.opengroup.org/architecture/togaf9-doc/arch/
[6] Weill P., Ross J.W. IT Governance: How Top Performers Manage IT Decision Rights for Superior Results. Harvard Business School Press. 2004.

Published By:
Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)
© Copyright: All rights reserved.