White Paper

Four Tips for Successful Enterprise Architecture Tool Implementation

WP0181 | March 2015

Ross Hocking
Business and Enterprise Architecture Consultant

Ross has experience delivering Process, Enterprise Architecture and Governance, Risk & Compliance projects globally across multiple sectors but with a focus in the public sector and finance.

Ross has a particular interest in the pragmatic implementation of tools and methods, Application portfolio Management (APM), Business Process driven change and the positioning of successful EA departments within organizations today. Ross has certification in TOGAF, ArchiMate and COBIT along with extensive implementation experience.

When implementing an Enterprise Architecture (EA) tool, there are many factors which influence its success including the number of users, expectations of usage within the user community and stakeholder attitude. Although these variables are important, the team implementing the tool can help to mitigate these risk factors by making good decisions throughout the implementation process.

The four tips in this white paper are essentially key decision points within the deployment process, along with some guidance to help navigate through these decision points. The content of this paper is based partly on research and partly on experiences of over fifty EA tool deployments over nearly ten years.

Tip 1 – Select the Most Practical Metamodel

Probably the most important factor when implementing an EA tool is which metamodel to support. Successful implementations often look back upon this decision as having been a critical success factor. When the metamodel chosen is incompatible with the organization, common complaints include:

- I don't understand how to create the views I require with the viewpoints available
- I know I can't create the views I require with the viewpoints available
- Consistently I am unable to create the relationships I need to build my model

The primary driver behind incorrect metamodel selection is the desire to implement a complex solution without the Enterprise Architecture (EA) maturity level required and without a realistic plan to attain such maturity quickly enough. Frameworks can outline the skills, roles and experience levels an organization may wish to include in an architecture team, it may be said in order to maximize the chances of success, such as TOGAF with the Architecture Skills Framework [1]. Most other frameworks lack such rigor.

In order to demonstrate the different levels of complexity offered by two of the most popular enterprise architecture metamodels, TOGAF [2] and ArchiMate [3], take the below example showing the different objects which can be used to model an application from different perspectives:

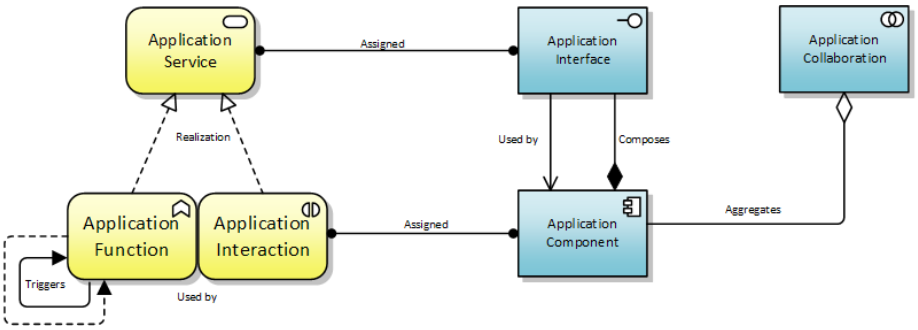

Fig1. Representation of ArchiMate Metamodel

This offers a level of detail which isn't found in many other metamodels, "the concepts of this language [ArchiMate] are sufficiently generic and expressive to model many of the aspects within specific domains" (Langkorst et al, 2013). In contrast to this, the TOGAF metamodel has much less detail with regards to the objects used to model an application, simply specifying a Logical Application Component and a Physical Application Component as shown in the below metamodel extract:

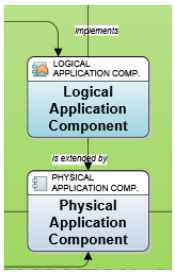

Fig2 Representation of the TOGAF Metamodel

In order to implement ArchiMate it could be said a higher level of knowledge is required to ensure users are able to effectively build and navigate the model vs. the TOGAF alternative. However with the level of detail offered in ArchiMate the user community is able to use a greater level of expression. The issue of complexity is not limited to TOGAF and ArchiMate metamodels, there are many custom metamodels developed by organizations and tool vendors, each with its own complexity level which has a significant impact on tool adoption.

Tip 2 – Select Suitable Modeling Language(s) and Notation(s)

Which modeling language(s) and notation(s) an organization uses plays a key role in the success of a tool implementation. Some languages are understood best by those with specialist knowledge, such as UML which is easier for those with a computer science back ground to understand according to Fowler and Scott (1999). Others such as ArchiMate use meaning as a way to communicate the language, which should enable a broader audience to understand the notation without specific training, “We do not put the notation of the ArchiMate language central, but rather focus on the meaning of the language concepts and their relations” (Langkorst et al, 2013). This is achieved by using color coding to highlight the role of the object type for example the below is a central theme in ArchiMate:

- Green is Passive Structure, defined as “An object on which behavior is performed” [3]
- Yellow is Behavior Element, defined as “a unit of activity performed by one or more active structure elements” [3]
- Blue is Active Structure, defined as “an entity that is capable of performing behavior” [3]

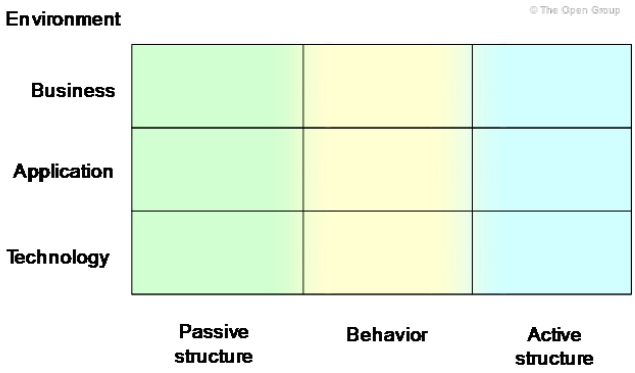

Fig 3 ArchiMate [3] Architectural Framework

There are many other languages, often used in conjunction with one another, such as Business Process Modeling Notation (BPMN) and Unified Modeling Language (UML), which focus on modeling specific areas such as business process flows (BPMN) and Software Engineering (UML). Typically an organization will select a set of languages according to their needs and skills. This process is critical since the most successful organizations select languages which they have had success with before or which they have a prior knowledge of.

Once a set of languages have been selected, the user community should ensure some shared level of understanding to avoid working in silos, divided by the knowledge of a specific notation. As Langkorst et al (2013) advises, the use of complex languages, which are difficult for non-experts to understand “frequently leads to misunderstandings that hinder the collaboration of architects and other stakeholders”.

Whichever modeling language and notation is selected, often the first thing EA teams seek to do is to customize the language. Whilst customization is often necessary, in order to deliver results quickly, as a way to lessen the learning curve it introduces a level of confusion for advanced users. These advanced users may fully grasp the intended use of each object in the language and see some overlap between the original objects and the customizations. As Gerben Wierda (2012) warns “Only when you have enough experience are you capable of really estimating the effect of the choices you have when changing the language”

Tip 3 – Set Clear Standards and Guidelines

As an implementation consultant, communicating the importance of standards and guidelines to support a tool is often a simple task, however the definition of them is in reality a difficult and time consuming task. Diligent tool implementation teams will often perform a comprehensive review of current content in order to focus in on the specific challenges the user community will face post implementation. This is done by creating comprehensive documentation delivered in an effective way, such as through a Microsoft SharePoint site, training sessions or documents available through the tool for example.

Although the structure of documentation, to support your tool implementation, isn’t the most important factor in the deployment, the definition of the below documents has proven to be effective and comprehensive in my experience:

Standards: A set of rules ,which must be adhered to, including for example:

- Mandatory metadata i.e. each business process must have a description
- Diagram structure i.e. each diagram must have one and only one tab
- Extensions to modeling language rules which your organization mandates i.e. BPMN sub-processes must never be expanded
- Metamodel dependencies i.e. all business services must have at least one business process linked
- Tool standards i.e. a document must never be checked out to one user continuously for more than one week

Guidelines: A set of best practices, which should be adhered to, including for example:

- Suggested important Metadata i.e. the number of users an application has should be populated

- Diagram structure i.e. where a heatmap is used, a legend should always be added
- Extensions to modeling language rules i.e. a shape should have the connector added to the left edge where possible
- Metamodel dependencies i.e. each application should be linked to a logical application where possible
- Tool standards i.e. each document should have a status

One example of the importance of rigor, in the development of modeling standards and guidelines, was found when a project I was working with was piloting the selected tool and they needed to model processes to define the requirements for a system. Our selected viewpoints supported process modeling from a human perspective, however failed to include the simultaneous modeling of the human process and the system process. This kind of oversight, as obvious as it sounds in hindsight, is easy to make if insufficient stakeholder engagement and content review is performed.

Almost every tool implementation is backed up by an agreed set of standards and guidelines, whether informal or well documented, however the degree to which these standards and guidelines are communicated and followed has a direct impact on the success of the tool implementation project

Tip 4 – Governance

As important as agreed standards are, expect the community to create non-compliant content. In the real world standards need to be enforced not only to maintain the integrity of the tool, but also as a way to provide feedback to the user community, helping to both drive education on your chosen standards and methods and improve the quality of the content produced by an architecture team.

The impact of a deviation from the standard is mostly minimal, however over the course of time such deviations can become significant. Effective governance is often centered around an automated core, implemented not by one person but distributed among a team.

Governance stands out as a common theme among organizations who have achieved successful tool implementation. Some useful techniques can include:

- Health check reporting
- Diagrams reviews (peer or gatekeeper)
- Content completeness reporting
- Metamodel utilization reporting
- Effective use of libraries to ‘fence off’ content prior to review

Conclusion

During an Enterprise Architecture tool implementation, there are many critical success factors, this white paper has attempted to outline some considerations an organization may make when embarking upon an implementation.

One of the main themes throughout this paper has been that there is no 'one size fits all' for EA tool implementations, what works for one organization may not work for another for reasons including maturity, team structure and team goals.

While each implementation is unique, it has been observed that successful organizations select suitable metamodels and notations, then subsequently implement pragmatic standards and guidelines supported by comprehensive governance procedures.

References

1. The Open Group (2011), The Open Group Architectural Framework (TOGAF) Version 9.1. The Open Group, www.opengroup.org/togaf/
2. The Open Group (2012), ArchiMate 2.0 Specification, Technical Standard, The Open Group, www.opengroup.org/archimate/
3. Langkorst et al 2013, Enterprise Architecture at Work, Modeling, Communication and Analysis, 3rd Edition, Springer
4. Fowler M, Scott K (1999), UML Distilled: A Brief Guide to the Standard Object Modeling Language, 2nd edition. Addison-Wesley
5. Gerben Wierda (2012), Mastering ArchiMate, 1st Edition

Orbus Software
3rd Floor
111 Buckingham Palace Road
London
SW1W 0SR
United Kingdom

+44 (0) 870 991 1851
enquiries@orbussoftware.com
www.orbussoftware.com

orbus software