White Paper

Architecture Modeling Crossovers

WP0218 | December 2015

Peter Harrad

Peter has worked with modeling standards and techniques throughout his 20 years in IT, in a career that has covered software development, solutions architecture and international consulting.

Peter's particular areas of interest are opportunities arising from interdisciplinary touchpoints, how to balance practicality and rigor when modeling, and the importance of viewpoints in addressing different stakeholder perspectives.

Enterprise Architecture is merely one discipline that attempts to govern and use IT effectively. Others include configuration management, strategy mapping, business process analysis... there are several. But the different disciplines don't exist independently of each other – there are overlaps. Given that this is the case, there's a natural impetus to consider sharing information between the repositories of information that each of these disciplines uses.

I've yet to encounter an architecture modeling tool that doesn't offer some kind of capability to import data from, and export data to, other systems. Even open source, community-supported tools such as Archi specifically discuss import/export in their documentation. However, nothing in life comes without some kind of investment, and so the question becomes one of what is the investment involved in setting up such a transfer of data to or from an architecture model, and why do it?

In this paper I'll reference my experiences with such integrations, and examine what factors determine whether doing so is a useful investment, or a pointless exercise. We'll start out by examining what benefits you can gain by architecture modeling and from this we can consider how the value of an architectural model is enhanced by integrating with some other information repository that covers an overlapping area.

Following this, I'll examine the costs that exist in such an integration and the questions that have to be considered in such a situation. This covers not just the obvious costs such as time to create and schedule data extractions, data transforms and data loads, but also the less obvious, indirect costs that result from integrating information from multiple repositories.

Last off, we'll illustrate the discussion by examining five common examples of such integrations, and how the benefits, costs and considerations that we've identified apply to each of these five use cases.

Benefits of integrations

The first step is to consider what benefits we might gain from sharing information between the architectural model and some other repository, and to do this, we need to first remind ourselves the benefits of a model. A model is a decision support tool – it facilitates planning and execution by providing a representation of the operating environment. So, the benefits of integrating with a different repository come from improving the quality of the model or the ability of people to use it. There are several ways that an integration can support this.

- **Reduced data collection costs.** If another team has already collected and is managing the data, the architecture team does not have to expend the time and effort to collect it again.
- **Improved data quality.** Enterprise Architecture necessarily cuts across different domains (this being its primary purpose), but this means that the EA practice is not specialized on a specific area. If the information is collected by the team that specializes in that area, they are going to be more able to detect anomalies and inaccuracies.
- **Improved buy-in.** If the architecture team can point to how they are basing their models on the actual information used by other teams, I've found this lends credibility to their efforts in the minds of executives.

Challenges and costs of integration

The first issue with importing from other tools is the need for data ownership. In other words, can the architecture team make or request changes? And how are the two repositories kept in sync? There are two options:

- Changes can only be made in the source repository and then propagated
- Changes can be made in the architecture repository and then propagated back to the source.

In general, the first approach works best, as it establishes clear ownership and responsibility for the data. Tool considerations also come into play on this question. In the first scenario above, the permissions model of the architecture repository should ideally be able to prevent changes to the imported information. Investigating what options are available in this area should form part of any integration effort.

The next question to address is what to import. It's entirely likely that only certain aspects of the imported information is relevant or useful for the architecture model. It's desirable to only import information that will actually be used, for the simple reason that presenting people with a sea of data fields makes it hard to locate the information that is actually important.

A third aspect to consider is the nature and frequency of updates. If there's an online connection so that changes in one repository are automatically reflected in a different one, then there's no issue. But the majority of import/export interfaces that I've seen are batch-based; in which case decisions need to be taken on how frequently imports take place – which will be driven by how often the information being imported changes.

Last of all, and coupled with the question of frequency is the question of establishing what mechanism you will use to identify errors with the data import that may occur.

Some common integration scenarios

To close this discussion, we'll briefly examine some common scenarios and how the considerations that we've discussed apply to them.

Integrating with a CMDB

Importing assets from a CMDB is one of the classic cases for an architecture tool. A particular bonus here is that many CMDBs use automated discovery, so the cost reduction and data quality aspects are quite noticeable, as well as speaking to the level of trust executives place in the data. So there are clear benefits from doing this.

Now let us consider the challenges. Data ownership questions can come in the play – configuration items such as servers clearly belong in the CMDB, but some product also allow definition of things such as business services. But it's the selection of what to import that poses the biggest challenge here. CMDBs often include things such as ports, VLANs and so on, and many EA tools will outright struggle to accurately model route-port-VLAN mappings (after all, it's not what they or standards like TOGAF or ArchiMate were meant for). So a level of analysis and data mapping will be necessary to ensure that the import is useful for the architectural model. Frequency of updates is likely to be driven more by the tempo of EA than the CMDB; the infrastructure is going to be receiving updates on a daily basis. Last of all, error handling mechanisms don't present any special considerations.

Overall then, the CMDB is a strong candidate for integration with the architecture model.

Process Re-engineering

The touchpoint between architecture modeling and process modeling is an interesting one once you reach the Enterprise Architecture level. In particular, many EA tools incorporate support for process mapping, so the two teams may well be working in the same repository. If this is the case, the tool owner will need to take care to establish appropriate partitioning to enable the two teams to work in a sufficiently decoupled manner. The benefits of data quality and data collection are clear, but the executive buy-in factor is the key benefit for aligning an architecture model with a process initiative.

In regards to the challenges, data ownership may or may not be an issue. There are some frameworks, such as IBM's Information Framework (IFW) that depend on defining reusable business tasks that are incorporated into end-to-end business processes. Defining what to import is not a complex area, but one where clarity is necessary. However, adopting this governance mechanism does answer the question around frequency of updates and means that error handling is rarely a concern. While defining a process map is often a part of requirements definition at the solution architecture level, enterprise architecture takes a higher-level view; it's the overall processes that the process team identify that will form part of the architecture model, not the task-level entities.

Overall then, process engineering offers noticeable potential benefits, particularly as regards executive buy-in, but also requires the introduction of good governance mechanisms.

Project Planning Integration

Integrating the project catalog with the architecture model is an interesting case, as the value of doing so depends wholly on the scope of the architecture effort. It provides benefits around data quality and data collection costs (and buy-in) – if the projects are worth tracking at all. This is only going to be the case if the architecture model has reached the level of explicitly modeling roadmapping and architecture planning. This is not always the case.

Now let's examine the challenges involved. There are going to be clear issues around data ownership. In a lot of ways, the touchpoint between project planning and the overall architecture model speaks directly to the overall governance of both. As with process modeling, the governance mechanism adopted will also define when updates cascade from project tracking to the architecture model. Data mapping issues will also be an important consideration – mapping the work items and affected items will likely be a manual process that will have to be addressed as part of the governance. Error handling, at least, does not present particular considerations.

In summary then, a tie-in to project planning requires the same introduction of good governance mechanisms as process planning, but the benefits will depend on whether roadmapping and transition planning is seen as a key focus for the architecture team.

Software inventory systems

Software inventory systems are another classic sweet spot for integration with architecture modeling. The work that such an effort does to identify the applications used by an organization and their attributes will clearly benefit architecture decisions…but the mapping from applications to processes, data and so on that the architecture repository contains will help the software inventory effort in evaluating the value and costs of the applications they consider. Both factors speak to the trust that business leadership will have in this catalog. So – a very strong candidate for adoption; in fact, it's not unusual that the application inventory is managed within the architecture repository itself.

Updates will likely face the same tempo; and error handling on import will either have no special considerations or no considerations at all if the two efforts are working off the same dataset. Likewise, there are generally little or no problems with needing to filter what information is recorded. The challenges are going to lie in the human aspect – and who is allowed to update the information, under what mechanisms.

Overall then, software inventory systems are the strongest candidate for integration with the architecture model out of our five cases.

Strategy maps and business model canvases

Integrating the strategy maps and business model canvases that the organization uses to determine its direction would seem to be an excellent candidate at first – after all, the dream of EA is to drive IT architecture from business considerations. However, in practice it requires a significant investment. Let's start by considering the potential benefits. Data quality, data collection costs and executive buy-in all benefit greatly from reusing such artefacts – assuming that they exist.

But the challenges are significant. Data ownership is clear – the executive suite and their consultants own the information contained within – but this leads to significant questions around data updates. Specifically, how to ensure that the architecture team are kept aware of the current version of thinking in the area? They'll need to engage in a certain level of selling the benefits of providing this information to them. The second major challenge comes from data selection – in so far as deciding how the often unstructured data contained in such artefacts are mapped to the metamodel of the architecture repository. Essentially, it requires teasing out a high degree of precision from people and artefacts that don't think precisely. So, while there is decent potential payoff, there is a significant initial and ongoing investment involved in integrating the two efforts.

Overall then, strategy maps and business model canvases seem like a strong candidate at first, but will require significant investment to implement. Consequently it will need a high degree of executive enthusiasm to justify doing so.

Conclusion

The benefits of integrating your architecture model with other information sources come from the following sources:

- **Reduced data collection costs** – reusing information from another source removes the need to collect it
- **Improved data quality** – bringing in information from a more specialized source can often mean that the data itself is of a higher quality
- **Improved buy-in** – using the information from another groups' repository reassures executives as to model accuracy.

But each integration is going to present certain challenges, these being:

- **Establishing data ownership** – including investigating what options to architecture tool has to prevent changes to imported data
- **Selecting the appropriate information to import and appropriate mapping to the architecture metamodel** – to keep a high 'signal to noise' ratio
- **Defining frequency of updates** – driven primarily by the volatility of the data concerned
- **Incorporating mechanisms for error handling** – everything breaks from time to time – how is this catered for?

Developing an architecture model involves the creation of an informational asset for the organization, and just as organizations look to improve efficiency by integrating their overall information silos, it's natural that architecture teams think in terms of leveraging the asset that they are creating in the same way.

By applying the factors outlined earlier in this paper, we've seen in five different examples how an organization can make an evaluation of whether such an integration is necessary, and if it is, the priority it needs to be given as a work item.

Orbus Software UK
London

Orbus Software US
New York

Orbus Software AUS
Sydney

Orbus Software RSA
Johannesburg

enquiries@orbussoftware.com | **www.orbussoftware.com**

Seattle Software Ltd. Victoria House, 50-58 Victoria Road, Farnborough, Hampshire, GU14 7PG. T/A Orbus Software. Registered in England and Wales 5196435