Enterprise Modeling – What We Have Learned, and What We Have Not

Håvard D. Jørgensen

Commitment AS, PO Box 534
N-1327 Lysaker, Norway
havard.jorgensen@commitment.no

Abstract. Over the last 25 years, enterprise modeling has evolved from a loose collection of business oriented modeling practices to comprehensive frameworks for enterprise architectures. What was initially a combination of industrial and software engineering methodologies applied to enterprise design and development, is today dominated by the concerns of IT development, management and governance. This talk summarizes experiences from years of practice, research and development in the field, and points out directions for future development.

Keywords: Enterprise architecture, Model-driven applications, Active knowledge architecture.

1 Background

Like many in the field, my involvement with enterprise modeling started with business process analysis, in this case in the early 90s [4]. Development of flexible process execution systems soon lead to customizable, model-driven application portals in general [3]. Process modeling was extended to cover modeling of other enterprise aspects. The techniques found utilization in interoperability and integration, and model-driven composition of applications from underlying services [5]. After some years of Enterprise Architecture (EA) methodology and tool development, we turned our focus towards industrial product development and design [2], looking to learn more from the area that enterprise modeling came from in the first place [6].

2 Lessons Learned

Many of the lessons presented below are obvious to enterprise modeling practitioners. They are included here because outsiders coming into the field sometimes get them wrong. Other lessons may be more controversial, and they may not be applicable in every situation.

First, let us look at four related lessons that concern the scope of enterprise modeling and enterprise architecture:

1. Don't reduce enterprise architecture to IT architecture.
2. “What can we automate” is not always the right question.

3. Don't frame business problems as IT problems.
4. Discuss purpose, scope and level of ambition throughout the architecture's lifecycle.

Some companies that provide enterprise application integration (EAI) describe their IT level integration platforms as enterprise architectures, and the people who develop them as enterprise architects. There is a clear danger that EA will go the same way as "business process", and be hi-jacked from the business domain to the IT domain.

In the overall practice of enterprise modeling, these lessons should be highlighted:

5. Don't confuse analysis and design.
6. Keep models and languages as simple as possible.
7. Don't let the language be a straight-jacket.

Enterprise modeling is used for analyzing the current situation, and for designing future solutions to identified problems, as in to-be business processes and architectures [7]. Analysis should focus on identifying the most important elements, dependencies and aspects of a domain. Often however, people perform a lot of synthesis during analysis, constructing complex structures of the elements, such as business processes and organizational hierarchies. Too much structuration during analysis is dangerous because the model will contain a lot of noise, and it makes the approach more conservative.

This leads on to the 6th principle, simplicity. A common condition is *analysis paralysis*, where decision making is continuously postponed in order to create a better foundation and justification. Enterprise modeling research also tends to violate the simplicity requirement. Countless papers are produced with extensions of modeling languages in order to represent some additional aspect more precisely. Often the added value is small, as the most important concerns could already be represented in existing languages. The cost in terms of increased complexity, on the other hand, is large. Validation that takes this issue into account is rare.

During modeling we have experienced a few common challenges:

8. Beware of hierarchies, they are often local to a viewpoint and not shared by everyone.
9. Beware of matrices and streamlined frameworks, often you need to break the system.
10. Beware of leaving important things out at the boundaries.
11. Don't confuse views with the underlying architecture elements.

Multiple, aspect oriented classification of model elements is important in order to allow overlapping hierarchies and multiple views for different stakeholders' perspectives.

With respect to the enterprise model content, we often found that

12. Product models are more important and fundamental than process models.
13. Processes are better understood by focusing on the decisions to make, the issues to solve, and the results to produce, than on the administrative ordering of steps.
14. Multi-dimensional analysis, combining e.g. processes with the data it manipulates and the organizational roles responsible, is superior to single-dimensional data modeling and business process modeling.

15. During decomposition, don't expect to "go all the way down" using a single modeling language. Another language is likely to be a better match for detailed models than the one used for high-level overviews.
16. Aspects that can be separated on one level of detail may be inherently woven together on another level. Products, processes and organization models do for instance become thoroughly intertwined in work execution.
17. Relationships are the most important kind of element in most models.

For managing dependencies, relationships should be modeled at the right level of detail. If every element is related to every other element, you're at a too high level of abstraction. If there is a clear one-to-one correspondence between two sets of elements, you have probably included too much detail.

In modeling tool development, and in selecting the right modeling tools for your needs, it is important to

18. Decide whether you need a multi-purpose modeling platform, or a modeling application specialized for one usage area.

Tool vendors struggle to find their place in the large number of applications domains for business oriented modeling. Establishing a general purpose architecture that can support the analyses needed by a wide range of stakeholders, requires a flexible and extensible modeling platform.

Finally, here is some advice for how an architect should behave:

19. Be open, humble, and willing to expose your mistakes.
20. Take charge, set directions.
21. Don't listen to management.
22. An architecture that is not actively used will die. Motivating stakeholders to participate, is an ongoing challenge.

Some architects treat the architecture as their own property, and see organizational stakeholders primarily as sources of information. Other inexperienced architects see themselves too much as enablers, and become followers of the whims of strong forces in the organization. Managers are of course important stakeholders, and generally also important sources of information for an architecture. We should however be cautious, as relying to one-sided on management perspectives can make it difficult to capture the concerns of the core, value-adding work performance. A typical result is business process models that include several activities for administration, planning, monitoring, reporting, reviewing, quality control, setting up, and closing etc., but very few details about how the actual work is to be performed. Management views should be complemented by operational views.

3 Future Directions

In the short term, the future development of enterprise modeling practice is likely to remain dominated by enterprise architectures for IT management. In this area, standardization of modeling languages is beginning, e.g. BPMN, and it will probably be a major trend in the next five years. Compared to e.g. UML, enterprise modeling deals with a much larger scope of content, and aims to support a much larger set of practices.

Standardization will be more difficult in this domain. Whether any single dominant vendor will emerge with a strong enough position to establish a de facto industry standard, is difficult to tell. If we end up relying on a poorly suited metamodeling framework from another domain, such as UML/MOF, the complexity of even a limited general purpose enterprise modeling language may prevent its use in practice [5].

Over the past decade, more and more EA frameworks have been proposed, often targeting different industries and application areas. Most proposals position themselves with reference to general purpose frameworks such as Zachman and TOGAF, but further alignment is hampered by the great complexity. A simplified TOGAF core should probably be established as a first step towards a consolidated, extensible standard EA framework, capable of supporting domain specific extensions. The core framework should include the definition of standard modeling concepts, but it is difficult to predict whether the framework or the language standard will arrive first.

Industrial design and development is in need of a language for capturing and sharing knowledge throughout the entire product lifecycle. While interoperability is becoming a reality between the tools used in the later stages of product development, conceptual and functional systems design is still not integrated. In these early stages, documents, rather than structured data/models, are still dominating. To develop enterprise modeling languages for conceptual and functional design is thus a major challenge for future research.

Finally, we see *enterprise model driven applications* as a promising approach to construct more customizable, agile, modular and service-oriented IT solutions. Pilot solutions applying what we call "active knowledge architectures" (AKA) show that applications can indeed be put together from simple building blocks, using high level enterprise models. AKA execution environments combine

- Model-driven data definition and management,
- Model-driven process, task, and rule execution,
- Model-driven views and user interfaces,
- Model-driven role-based filtering and access control.

From this perspective, standard enterprise modeling languages should be aligned with industry standards for data exchange. More information about our ideas for future modeling approaches and model-driven applications is found in the *Active Knowledge Modeling* blog [1].

Acknowledgements

This presentation draws on the experience of several colleagues and collaborators, in particular Frank Lillehagen, who founded Metis in 1986 and has been developing enterprise modeling ever since.

References

1. Active Knowledge Modeling blog, `http://activeknowledgemodeling.com`
2. Johnsen, S., et al.: Model-based Adaptive Product and Process Engineering. In: Rabe (ed.) Ambient Intelligence Technologies for the Product Lifecycle (2007)

3. Jørgensen, H.D., Krogstie, J., Ohren, O.P., Johnsen, S.G.: Interactive Models for Tailorable and Evolving Information Systems. Journal of Applied Systems Studies 6(1) (2005)
4. Jørgensen, H.D.: Interactive Process Models. PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway (2004), `http://ntnu.diva-portal.org/smash/record.jsf?pid=diva2:125137`
5. Jørgensen, H.D., Karlsen, D., Lillehagen, F.: Collaborative Modeling and Metamodeling with the Enterprise Knowledge Architecture. An International Journal on Enterprise Modeling and Information Systems Architectures 1(1) (2005)
6. Lillehagen, F., Krogstie, J.: Active Knowledge Modeling for Enterprises. Springer, Heidelberg (2008)
7. Stirna, J., Persson, A., Sandkuhl, K.: Participative Enterprise Modeling: Experiences and Recommendations. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 546–560. Springer, Heidelberg (2007)