4

HANDLING THE COMPLEXITY OF IT-ENVIRONMENTS WITH ENTERPRISE ARCHITECTURE

Thomas Birkhölzer,
Fachhochschule Konstanz,
Braunegger Str. 55, 78462 Konstanz; GERMANY
thomas.birkhoelzer@fh-konstanz.de
Jürgen Vaupel
Siemens AG - Medical Solutions
Henkestr. 127; D-91052 Erlangen; GERMANY
Juergen.Vaupel@med.siemens.de

The relation of Enterprise Architecture to the traditional architectural roles of software engineering (platform architect, component architect as defined below) is similar to the relation between "city planning" and "building blue-prints" in the building domain. There is a difference in scale, scope, necessary competences and methodologies. It is the intent of this article to stimulate this understanding summarized in the following two theses:
1. Enterprise Architecture is a necessary and distinct architectural role. Successful large-scale system as well as virtual enterprise development requires appreciation and inclusion of this role in the IT-engineering process.
2. Enterprise Architecture means cross-system coordination with similar stakeholders outside the own business ownership. This distinguishes Enterprise Architecture from traditional architectural roles and implies distinct tasks, methodologies, and required skills.

1. INTRODUCTION

This article is the result of the reflected experiences of the two authors during their work in a central architecture group of Siemens Medical Solutions, i.e. it is rooted in concrete business challenges, but tries to abstract from particular problems in order to extract generic issues.

Siemens Medical Solutions offers its customer the whole bandwidth of IT-solutions for healthcare. In the past, these were mainly standalone products bought, operated and maintained independently. However, these independent island are now rapidly growing together forming complex IT-environments. These complex IT-environments are characterized by deconstruction of traditional packaging and consolidation of common infrastructure and services (see Section 2.2). This process is not healthcare specific – but similar in all application environments.

In an "island" business environment, the success of a business depends basically on self-contained and self-controlled properties of systems and products like functionality, cost, quality, i.e. the emphasis is on internal issues – internal with respect to technology, organization and business.

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35585-6_68

However, the evolution of deconstruction and consolidation described below generates another distinct challenge. In a "consolidated" business environment, the business success depends crucially on a successful *embedding* of own systems and products into the overall environment. This requires more than just some external interfaces, but coordination with and anticipation of this environment.

This task is described in this paper as "Enterprise Architecture". It is the intent of this article to stimulate the understanding of this role and its relation to other architectural roles in software engineering like platform architect and component architect, e.g. see (Barroca, 1999), (Bredemeyer, 1999). This relation is similar to the well-understood and established relation between "city planning" and "building blue-prints" in the building domain.

2. BACKGROUND

2.1 Definition of Architecture

The term Architecture is used to denominate the conceptual tasks similar to the building domain: "Architecture" is the structuring and managing of complex tasks such that overall efficiency increases, while providing four main outputs:

- Principles, e.g. shaping of design methodologies, determination of used technologies
- Partitioning, e.g. the definition of structure and layers, separation into components and modules, reference models
- Interfaces, e.g. the definition of interaction between the pieces.
- Explaining and promoting architectural choices and decisions.

2.2 The Challenge by Deconstruction and Consolidation

In general business terms, horizontal integration brought together businesses that could share key assets like technologies, distribution channels or manufacturing processes. Vertical integration brought together all parts in a value chain in one integrated business to eliminate inefficiencies along the value chain.

These are also key trends in software and system development:

- Vertical integration to provide seamlessly all functionality along a task (value chain).
- Horizontal integration of common infrastructure services to achieve efficiency of resources.

In the past, vertical and horizontal integration was possible only within a system, i.e. "intra-system-integration". This drastically limited the scope of possible integration to the amount of complexity manageable within a single system scope. This changed with today's IT-technologies, since they provide two key advances:

- Advanced networking technologies. These enable sharing resources and services beyond traditional scopes and provide potentially *unlimited reach* of IT-services.
- Advanced interfacing technologies. These enable tight integration even across system boundaries and therefore provide potentially *unlimited richness* of IT-services.

This new possibility to achieve reach and richness, without the need to sacrifice one for the other, drives the process of deconstruction and consolidation. In this context deconstruction means the dissolution and separation of established (business and software) configurations, whereas consolidation is the reallocation and reunion for a shared usage of common services.

3. ARCHITECTURAL SCOPES

3.1 Three Levels of Architecture

For a system with any sizeable complexity, there is not just "one" architecture – but rather several scopes with distinct focus, tasks and responsibilities, but of course with dependencies and overlap.

Although there is in general no common agreement about which levels are most useful, the common sense behind defining and introducing scopes is always the same, see (Abowd, 1996), (Barbacci, 1997), (Kruchten,1995), (Kazman, 1996): Separating different system aspects into scopes helps to manage the system complexity. The following levels are based on our experience and each of them describes a different kind of structure and addresses different architecture and engineering tasks, see Figure 1:

- **Enterprise Architecture**
  The term "Enterprise" should denote the set of all systems, which interact in a specific domain. In general, these systems are designed independently *under different business responsibility.* Moreover, the architecture is typically not "designed" but has evolved over time. Enterprise Architecture therefore deals with *inter*-platform topics. It is also the level, on which Virtual Enterprises are formed.
- **Platform Architecture**
  "Platform" should denote the set of components, which are developed under a *common business responsibility.* In distinction to the enterprise level, there is a central design and decision process available, which could enforce implementations, methods and decisions. The reach of the Platform Architecture corresponds to the reach of this decision process. Platform Architecture deals with *intra-platform* or *inter-component* topics.
- **Component (Application) Architecture**
  Component or application level denotes the set of functionality, which is usually implemented under one implementation responsibility, i.e. within a team. Component Architecture deals with *intra-component* topics.

Obviously, the need of the different architecture levels evolves according to the underlying size and complexity. Historically, small projects have even been conducted by single programmers without any explicit architecture at all. When systems grew more complex, application architecture was needed to synchronize designs among many developers, see (Bredemeyer, 1999). Platform architecture is driven by the strive for synergy and reuse across single projects, products or systems: sharing common infrastructure, functionality and services, applications become components of a broader context. Enterprise Architecture is just the next step in this evolution to establish a division of labor, competence and responsibility.

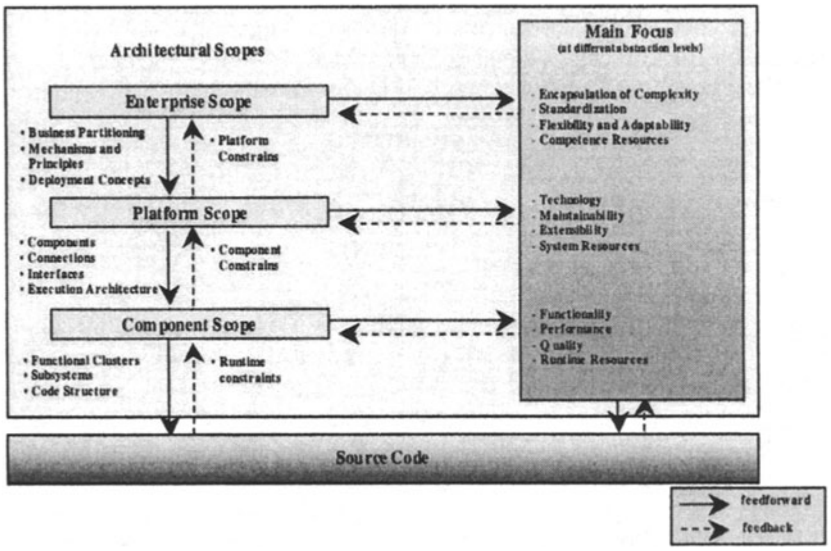

Figure 1 - Foci of Architectural Scopes

3.2 Enterprise Scope

The enterprise view describes the domain in – oterms of major business elements and the relationship among them, which includes the development and deployment of system mechanisms and principles. The enterprise scope is tied most closely to the business domain. In this view, the system functionality is partitioned in different business aspects and includes the identification and management of business opportunities.

Enterprise Architecture is mostly technology independent with a scope across different systems and technology generations. This work includes the anticipation of future business requirements and their consideration in an enterprise architecture. The primary engineering tasks of this view are:

- How can the domain complexity be encapsulated to come to a system of separated and interacting pieces with distinct business scopes and competences?
- Which standardization can and should be leveraged?
- How can long-term flexibility and adaptability be achieved?
- How can new and on-the-horizon concepts are deployed?
- How can cross-system compatibility and conformance rationalization be supported?

3.3 Platform Scope

In this view, the system functionality is mapped to architecture elements (components) with coordination and data exchange (connectors). On one hand problems and solutions in the platform scope are primarily viewed in system terms and address concrete technologies. On the other hand they should be relatively independent of concrete implementations and particular software and hardware techniques. The engineering concerns addressed by the platform view include the following:

- Which structure and components are necessary to fulfill the *scope* of requirements of the platform with optimal usage of system resources?
- Which technologies should be used and how are commercial off-the-shelf components to be integrated with the rest of the system?
- How does the system incorporate portions of the prior generations of the product and how will it support future generations (requirements)?
- How can validation be supported?

3.4 Component Scope

In the component view, the components and connectors from the platform scope are mapped to subsystems and components. Here an architect addresses how the conceptual solution can be realized by application code with today's software technologies. The primary engineering tasks are the following:

- Which modules are necessary to fulfill the actual requirements (as put forth in the current product definition) for implementation, how can dependencies between modules be minimized, how can reuse of modules and subsystems be maximized?
- What system support/services does it use?
- What techniques can be used to isolate the product from changes in off-the-shelf components, the underlying infrastructure, or changes to standards?
- How can testing be supported?

3.5 The influence of the three different views on architectural tasks

Though IT development consists of several iterations of segmentation and refinement, depending on the specific tasks in the software development cycle the involvement of the three levels of architecture varies. Whereas Enterprise Architecture is mostly involved in the more generic software development tasks, like the business partitioning and the cross platform interface definition, Platform Architecture takes over to define component partitioning and intra-platform interfaces based on the cross platform interfaces. Based on these intra-platform interfaces Component Architecture designs and develops specific components.

The involvement and the influence of an enterprise architect in business partitioning and cross platform interface definition is very high. His involvement in the definition of intra-platform interfaces and platform partitioning is low, like his involvement in component design. To identify, define and manage business partitions means for an enterprise architect, that he needs to be a generalist.

Platform and component architects on the other hand need to be specialists with a detailed understanding of their systems. They plan, design and control the current development of modules for the system, e.g. during the construction phase the platform architect decides on the used technology like EJB, .NET or C# versus Java. His main focus is the realization of requirements like performance or scalability.

Of course, there is no single and sharp borderline between these different process steps, yet they are more sliding transitions and may vary depending on the specific organization.

As a consequence of the described differences, each above-mentioned architecture scope needs an unique architectural role with a distinct set of competencies to fulfill the described tasks. Nevertheless, each of the architectures needs to be synchronized across the different abstraction layers. This means negotiation and tuning between the different architects and their respective architectures.

On the other hand every architect in his scope has to be something like an "evangelist" and needs to promote and vote for his architecture in his domain to achieve a uniform and consistent architecture. This is a field of tension every architect has to face. On one hand he needs to enforce his architecture in his domain and on the other hand the needs to reconcile the different types of architecture in order to find the right compromise for the best overall system.

4. ENTERPRISE ARCHITECTURE

Looking at the distinction in scope and issues of the different architectural views, Enterprise Architecture requires also distinct methodologies and skills.

4.1 Tasks and Methodology

4.1.1 Abstraction

As said, due to the cross-system and cross-generation scope, Enterprise Architecture must be technology as well as implementation agnostic, i.e. the structures and constructs of Enterprise Architecture must be independent of certain technologies or implementations. This implies abstraction. Abstraction means, the separation of the generic, underlying terms from concrete embodiments, e.g. implementations. One of the most powerful methods in this direction is the introduction of *roles* and *services*. A role or service describes tasks ("components") independent of a specific system or implementation. The IHE-initiative (Integrating the Healthcare Enterprise), see (IHE, 2001) is one of the best examples within healthcare proving the potential of such an approach on a cross-system, cross-vendor scale. Abstraction is sometimes confused with "vaporware". While abstraction is necessarily in-concrete in the sense of independent of specific embodiments, it is necessary to describe and solve existing problems, which are otherwise not visible and therefore not solvable in an explicit and deterministic manner.

4.1.2 Growing versus Design

Traditional software architecture is based on the "design" paradigm: the ingenuity of the developers creates the appropriate artifacts. Enterprise Architecture cannot be designed in this sense, because in general, Enterprise IT spans independent businesses, driven by independent business authorities, see Section 3.1. This implies that there is no single point of authority to *enforce* decisions. Instead, a consensus process between the stakeholders must achieve agreements and coordination. Instead of a top down design approach, there is a networking of stakeholders and an evolution of results. The effectiveness and consensus of this network defines and limits the possible uniformity on this level, as all standards can cover only minimal consensus – not bold goals. For issues, where no consensus is possible, diversity and difference must be handled.

4.1.3 Anticipation

While the definition and design of a concrete system focuses on current customer requirements, it is the task of Enterprise Architecture to provide some cross-generation continuity. This can only be achieved by anticipation of future requirements and trends. In this sense, Enterprise Architecture fulfills a similar task as Innovation Management – however with the focus to prepare the structures for the necessary flexibility and agility. (However, it is outside the scope of Enterprise Architecture to provide the business rationale to select concrete business opportunities.)

4.1.4 Patterns

Within traditional Software Architecture, the use of Design Patterns has been proven very useful, both as a design methodology as well as communication tool, e.g. see (Gamma, 1996). It's assumed, that such Patterns might be even more useful in the field of Enterprise Architecture. However, to the knowledge of the authors, there is only limited material published so far (Buschmann, 1996), (Brown, 1998). Further research in this direction might be fruitful.

4.2 Role

Any architect needs some form of a client for whom the architecture is developed, i.e. a role or person, who profits from the work of the architect. While this is very straightforward for traditional Software Architecture – the client is the business owner of the system to be developed – the issue is much more complicated for Enterprise Architecture. While there is the hope that all stakeholder might profit in the long-term, in usual business environments such a statement must be made much more concrete in order to get the necessary commitment. There are three principle situations:

1. There exists a cross-system business authority, which is able and willing to enforce some overall optimization, e.g. the corporate level at large IT-companies.
2. The business is structured such that mid-term coordination and collaboration are required and rewarded for *all* stakeholders in a measurable way, e.g. because of perceptible customer requirements. In this situation, there is a win-win situation for coordination on all sides which is the base for an architectural networking
3. None of the above.

The role of the enterprise architect needs to adapt to these situations. It is obvious that in situation 1 the preferred solution is a central role reporting directly to the central business authority. In this role, Enterprise Architecture can function as a central monitoring and controlling instance, which enforces the generic principles and minimal requirements. However, this is a rare situation because in many companies business responsibility is delegated to independent sub-units.

Situation 2 calls for a networking of interested experts. Such a network can achieve decisions, which lead to win-win results for all sides. It is an important task for Enterprise Architecture to identify, shape and promote such win-win opportunities. Of course, this approach is limited in case of real conflicts of interests.

However, there are also business situation, in which a general need might be felt, but which cannot be translated in a measurable profit for all involved stakeholders. In this situation, Enterprise Architecture might be desirable in principle, however it will not be successful in practice. It is of prime importance to recognize and admit this situation. ***If there is no client, there is no role for an enterprise architect!***

4.3 Skills

From the previous discussion, it is obvious that an Enterprise Architect needs very broad skills. It is tempting to describe this skill sets in terms of the overall hero: combining most advanced technological knowledge – to use and anticipate state-of-the-art possibilities – with deep insight into the application domain – to understand the customer use cases – and a bold entrepreneurship – to consider all business implications. All this is definitely necessary.

While it is obvious, that an enterprise architect needs to be a generalist in this triangle between technology, business and application, in order to bring the demands back to human level, it might be worthwhile to state the limitations:

- An enterprise architect is not the "senior specialist" or "senior implementer". It is not required that the architect has the skills to implement or realize his plans. This is obvious for most other domains (e.g. plumbing or electro-installation should rather not be done by an architect) – but is still very often debated in software development.
- An enterprise architect needs to understand the main customer requirements, but without the level of detail and concreteness to enumerate use cases like a product manager.
- An enterprise architect needs to understand basic business mechanism, but without the skill to fill in detailed accounting sheets or profitability checks.

This generalist approach makes an enterprise architect susceptible to the challenges by the respective specialists. Maybe, it is the most important skill of an enterprise architect to have the stature and authority to stand these challenges.

5. REFERENCES

1. *Abowd, G. et al.; Recommended Best Industrial Practice for Software Architecture Evaluation. Pittsburgh, Pa.: Software Engineering Institute, CMU, 1996.*
2. *Barbacci, M.; et al.; Steps in an Architecture Tradeoff Analysis Method: Quality Attribute Models and Analysis. Pittsburgh, Pa.: Software Engineering Institute, CMU, 1997.*
3. *Barroca L., Hall J., Hall P.; Software Architecture; Springer Verlag, 1999*
4. *Bredemeyer D., Malan R.; The Role of the Architect in Software Development; http://www.bredemeyer.com/role.pdf, 1999.*
5. *Brown W.J., Malveau R.C., McCormick H.W.; AntiPatterns; 1998; Wiley.*
6. *Buschmann, F., et al.; Pattern-Oriented Software Architecture; Wiley, 1996.*
7. *Gamma, E., el. al.; Design Patterns, Elements of Reusable Object-Oriented Software, Addison-Wesley, 1995.*
8. *IHE - Integrating the Healthcare Enterprise; http://www.rsna.org/ihe, 2001*
9. *R. Kazman, G. et.al.; Scenario-Based Analysis of Software Architecture; IEEE Software November 1996, 47-55.*
10. *Kruchten, P. B.; The 4+1 View Model of Architecture; IEEE Software November 1995, p. 42-50.*