Enterprise Architecture for Complex System-of-Systems Contexts

Philip J. Boxer, Suzanne Garcia
Research, Technology and System Solutions Program
Software Engineering Institute
Pittsburgh PA 15213-2612
pboxer@sei.cmu.edu

***Abstract*— An enterprise architecture is an accepted, widely used means for an organization to capture the relationship of its business operations to the systems and data that support them. Increasingly, enterprises are participating in complex system-of-systems contexts in order to meet changing customer demands that require them to collaborate with other enterprises in new and innovative ways. For a complex system-of-systems context, a shortcoming of enterprise architecture is that it presumes a single enterprise or a single, ultimate source of control.**

This paper explores an approach to reasoning about distributed collaboration in the complex system-of-systems, multi-enterprise context, in which this single, ultimate source of control does not exist. It outlines the ways in which the long-used Zachman Framework for enterprise architecture would need to be modified to account for multi-enterprise collaboration and decentralized governance. It proposes a concept of stratification to meet this need and puts forward the main characteristics of the methods needed to model the stratified relationships of complex systems-of-systems to their contexts-of-use.

Keywords-stratification; multi-enterprise systems of systems; distrubuted collaboration

I. THE DOUBLE CHALLENGE FACING ENTERPRISES

Enterprises face a double challenge when responding to changing demands in a complex system-of-systems context [1]. This double challenge involves preventing disparity between the enterprise's forms of *governance* and the forms of *relationship through which the enterprise creates value* for its customers.

Governance works through the way policies and procedures authorize members of the enterprise to act and can itself be described in terms of two dimensions (see the left-hand side of Fig. 1): the way members are held accountable for what they do ('up' an accountability hierarchy or 'out' towards the customer situation) and the responsibility members have for the resources through which they are enabled to act (directly under their span of control or indirectly across a span of complexity) [2]. These two dimensions define four canonical kinds of governance style, referred to in the diagram as *role*, *achievement*, *power*, and *support*. Note that like most schema of this sort, no enterprise lives completely in any single quadrant [3].

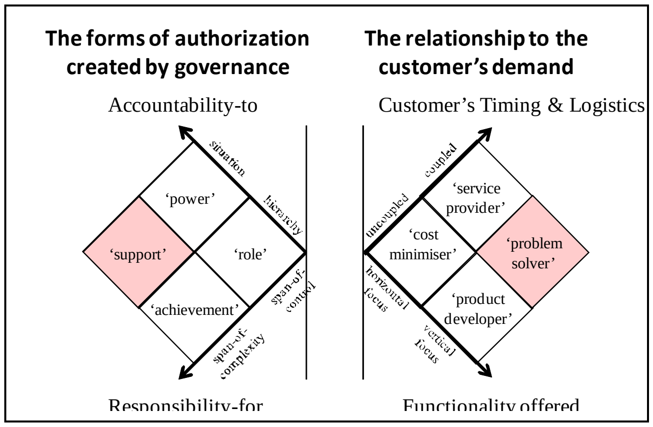


Figure 1. Aligning forms of governance to value propositions

The relationship to the customer's demand (the enterprise's value proposition) can also be expressed in terms of two dimensions (see the right-hand side of Fig. 1): the functionality offered by the enterprise to its customers (horizontally focused across customer segments or vertically focused on particular customers) and the way in which the enterprise's performance is coupled to the *where* and *when* of the customer's experience (coupled to or uncoupled from the customer's context-of-use). This too defines four canonical kinds of value proposition, three of which are organized around the supplier [4] and one of which is organized around the customer's experience [5].

In order to sustain a way of creating value for the customer through a chosen positioning on the right, the form of governance on the left needs to be aligned with that position [6]. Any lack of alignment will result either in difficulties sustaining the desired form of relationship to the customer or in costs of governance that are in excess of those needed to meet the customer's expectations.

The 'problem-solver' quadrant differs from the other three because the enterprise has to make possible a particular relationship to each customer's demand (i.e. with functionality that is vertically focused and behaviors that are coupled to the particular customer's context-of-use), with a correspondingly particular form of governance relevant to the customer's situation [7]. This form of governance can be referred to as *distributed collaboration* [8], being virtual or collaborative in nature, in order to distinguish it from the *directed* forms of governance in the other three quadrants [9]. It is the type of customer relationship that many enterprises find themselves in when they are participating in a complex system of systems, such as the regional health care networks funded by the US Department of Health and Human Services. Yet most of the governance structures of an enterprise were not conceived of with distributed collaboration in mind.

II. Enterprise Architecture and Beyond

The Zachman framework is often to be found in the roots of enterprise architecture frameworks e.g. DODAF [10]. The two axes of this framework originally related to creating and changing industrial products:

- **Abstractions**: Bills of Material (What), Functional Specs (How), Drawings (Where), Operating Instructions (Who), Timing Diagrams (When), and Design Objectives (Why); and
- **Perspectives:** Scoping Boundaries (Strategists), Requirements or Concepts (Owners), Schematics or Engineering descriptions (Designers), Blueprints or Manufacturing Engineering descriptions (Builders), and Tooling Configurations (Implementers) [11].

Zachman argues that we may use different language when describing information systems, but that the distinctions are the same. The important point he makes, however, is that the enterprise architecture describes the way the enterprise creates and changes its products – it does not define the entepríse itself [12]. How, therefore, are we to think about the architecture of the ‘problem solver’ enterprise[1], that needs to use different architectures to support different kinds of customer experience of services, frequently in conjunction with other enterprises involved with that same experience.

Recent work at the SEI explores the similarities and differences between various *genres* of architecture, whether enterprise architecture, system-of-systems architecture, system architecture, or software architecture. This work identifies significant commonalities across the genres, as well as emphasizing important complementarities between the different perspectives [14]. It also puts forward an engineering problem space ranging from the simple to the complex, in which the more complex the systems environment, the less a single source of authority is available through which to bring the different genres together under a unifying architecture [15]. This again presents us with the difficulty that arises in defining the architecture of a ‘problem solver’ enterprise: a single source of authority can define a particular collaboration the enterprise enters into, but cannot be assumed to apply to its other collaborations. This is a particular difficulty within the context of ultra-large scale systems [16], where multiple collaborations distributed across multiple sources of authority would appear to place us beyond the scope of an enterprise architecture.

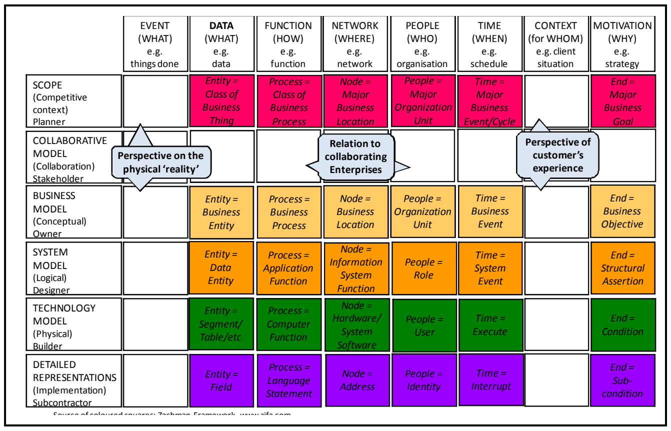
[1] i.e. an enterprise defined by multiple architectures, rather than by multiple instantiations within a single architecture, as with product-line architectures [13].

III. Gaps in the Zachman Framework

We can however use the Zachman framework to think about the relationship of a single enterprise architecture to this ‘beyond’ of distributed collaboration. The Zachman framework is represented by the colored squares in Fig. 2, from which the row representing actual instantiations has been omitted.

First, an extra abstraction can be added to describe the relation of a given collaboration to its participating enterprises (the second row in Fig. 2). This relation would describe the particular governance arrangements put in place with the other stakeholders in a collaboration to determine, for example, the nature of the quality attributes and economics expected of its systems (influence maps are an example of this form of analysis [17]).

Second, two perspectives can be added (the first and seventh columns in Fig. 2), the first extending the *what* to describe the model of physical “reality” from which data is being generated. For example, those aspects of physical reality attended to by a financial services collaboration will be very different from those attended to by a healthcare collaboration. And the second perspective adds a *for whom* description of the context to the customer’s experience that is the actual source of the customer’s demand on the enterprise. For example, the context shaping the nature of a householder’s demand for financial services will be very different from that shaping the demand from a business.



	EVENT (WHAT) e.g. things done	**DATA** (WHAT) e.g. data	FUNCTION (HOW) e.g. function	NETWORK (WHERE) e.g. network	PEOPLE (WHO) e.g. organisation	TIME (WHEN) e.g. schedule	CONTEXT (for WHOM) e.g. client situation	MOTIVATION (WHY) e.g. strategy
SCOPE (Competitive context) Planner		*Entity = Class of Business Thing*	*Process = Class of Business Process*	*Node = Major Business Location*	*People = Major Organization Unit*	*Time = Major Business Event/Cycle*		*End = Major Business Goal*
COLLABORATIVE MODEL (Collaboration) Stakeholder								
BUSINESS MODEL (Conceptual) Owner		*Entity = Business Entity*	*Process = Business Process*	*Node = Business Location*	*People = Organization Unit*	*Time = Business Event*		*End = Business Objective*
SYSTEM MODEL (Logical) Designer		*Entity = Data Entity*	*Process = Application Function*	*Node = Information System Function*	*People = Role*	*Time = System Event*		*End = Structural Assertion*
TECHNOLOGY MODEL (Physical) Builder		*Entity = Segment/ Table/etc*	*Process = Computer Function*	*Node = Hardware/ System Software*	*People = User*	*Time = Execute*		*End = Condition*
DETAILED REPRESENTATIONS (Implementation) Subcontractor		*Entity = Field*	*Process = Language Statement*	*Node = Address*	*People = Identity*	*Time = Interrupt*		*End = Sub-condition*

Perspective on the physical ‘reality’
Relation to collaborating Enterprises
Perspective of customer’s experience
Source of coloured squares: Zachman Framework, www.zifa.com

Figure 2. Gaps in the Zachman Framework

The extra abstraction and two perspectives are implicit in the way an enterprise applies the framework to describe its own architecture: it is the only stakeholder, and both the “reality” and the context relationship to the customer’s experience are part of how the enterprise has arrived at its choice of the product to be created by its architecture. By making them explicit, however, we can describe these background assumptions defining how the architecture of the collaboration is related to its larger context. By adding the abstraction and two perspectives to the modeling of collaboration, we can consider how the multiple collaborations in which an enterprise participates relate to one another through the ways overlapping stakeholder and customer contexts link their architectures. Thus, we can begin to think about the enterprise architecture needed to support the fourth quadrant in Fig. 1.

IV. Modeling the relation to the larger context

Our approach to modeling the relation of a collaboration to its larger context has been used successfully in both industry and government, giving an expanded account of the issues facing enterprises involved in complex system-of-systems situations beyond those that are typical of an analysis using a framework such as Zachman [18, 19]. As such, it starts with the views normally used for describing the architectures of systems of systems:

- **Structure-functioning**: the physical and logical structures and their relationship to processes and functional outputs.
- **Trace/Data**: the data representations of the states of the physical and logical structures and the relationship of those states to the structure-functioning.

To this are added three additional views that can describe the enterprise contexts in which the collaboration is formed, the way processes of collaboration cut across the accountability hierarchies of those enterprise contexts, and the relationships of those processes to demand:

- **Hierarchy**: the accountability hierarchies under which structure-functioning and trace/data elements are held accountable to governance processes.
- **Synchronization**: the social relationships and the data fusion relationships between structure-functioning and trace/data elements that cut across the accountability hierarchies.
- **Demand/Drivers**: the customer situations generating demand, their drivers, and the contexts within which the customer situations emerge.

Each of these views is modeled visually, with each view generating a knowledge-base of triples in the form subject-verb-object, in which the types of subject, verb, and object are particular to each view. (This is the same form of knowledge-base as is used by IBM in its Brownfield development approach [20], the differences being in the ontology of the views and therefore the forms of analysis that are possible.)

Various forms of analysis of the resultant combined knowledge-base are then possible, including the extraction of design dependencies across the structure-functioning and trace/data views. We can derive an architectural description from these dependencies, as in, for example, modularity analysis [21]. This architecture, which may be more or less emergent, constrains the uses to which systems can be put within the larger context. Understanding these constraining effects within a complex system of systems can help in deciding whether to re-architect interoperabilities between systems or to re-architect the constituent systems themselves.

Another form of analysis probes the nature of the different accountability hierarchies across which collaborations are being established, each one of which may have differing expectations for the quality attributes that any underlying architectures should exhibit. Understanding the differences in expectations is a key to effectively managing the agreements between parties for whom there is no sole source of control.

For the purposes of this paper, we are interested in the ways collaborations constrain how the modularity of functionality underlying an architecture can be defined [22]. This involves looking for layered relationships across the combined knowledge-base that can be aligned to particular demands and analyzing the particular ways in which the collaborations can be made to cohere. These relationships are not hierarchical (i.e., “is a part of”) but rather of a service type (i.e., “is used by”) [23], and they generate a *stratification* of embedded layers that relate underlying technologies to the ultimate customer’s contexts.

V. Analysing a stratification

The analysis of a stratification describes how particular demands can be aligned to underlying technologies made available through the infrastructures providing systems, equipment, people, and materials. The analysis is not only of the content of the different layers, but also of the way the different layers can be aligned in the particular forms demanded. The stratification distinguishes six layers, at the bottom of which are the technological capabilities available, and at the top of which are the customer situations in which demands are to be satisfied. Each layer supports the layer above it, so that the layers define a set of nested contexts. What makes the analysis valuable is its ability to identify gaps in the way lower layers are aligned to higher layers.

Fig. 3 shows an example of a stratification within the context of treating a medical patient’s condition. This is a schematic representation of matrices with common rows and columns, the content of which define the particular relationships in the stratification. The matrices are generated from an analysis of the composite knowledge-base.

At the bottom of the stratification are the medical technologies (matrix 1). The design dependencies constraining how they can be used are represented in matrices 0 and 1b. At the top are the patient situations (matrix 6), within the larger context of the patient’s demand defined by the condition and its drivers (matrices 7 and 7b).

The stratification is then defined by the zig-zag of matrices in between. Thus medical technologies (matrix 1) are combined to form medical services (matrix 2) that are themselves brought together to form treatment protocols (matrix 3). These treatment protocols are then combined to form care plans (matrix 4) that, when combined across multiple episodes of care (matrix 5), provide care through the life of the patient’s condition (matrix 6). Given the organization of these layers, the off-diagonal matrices (2b, 3b, 4b, 5b and 6b) then represent the ways in which the lower layers are aligned to the higher layers.

Overall, the hierarchies (matrix 1c) organize the way medical services are made available to doctors, while the doctors align the upper layers of the stratification to the particular needs of the patient. Thus the stratification describes the governance of the collaborations addressing the needs of patients, within the context of the hierarchies from which the constituent medical services are drawn. The analysis of stratification, therefore, provides a framework within which to determine the kinds of alignment needed between the engineering of the underlying technologies and the governance of the collaboration processes, in ways that can span the variety of collaborations needed. For the theoretical roots of this particular form of stratified analysis, see [24].

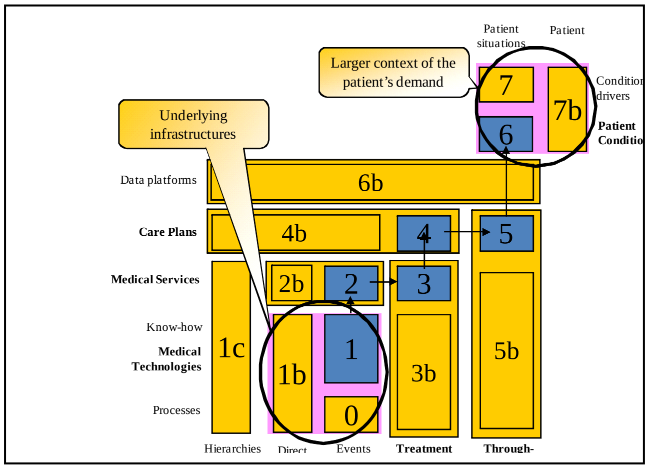


Figure 3. Stratification of the treatment of a patient's condition

So what is different about the enterprise architecture needed to support virtual or collaborative forms of governance? Directed systems of systems will continue to have their architectures, but in the complex systems-of-systems contexts supporting distributed collaboration, the architecture of the collaborative enterprise has to be approached as emergent, created through an alignment of individual architectures. The processes that engineer this alignment have therefore to be driven by the nature of the demands needing to be satisfied and the particular system-of-systems interoperability risks that must be mitigated. The approach to modeling and analysis outlined in this paper extends the enterprise architecture ideas promulgated via the Zachman framework in order to provide a framework within which to meet this challenge.

VI. Conclusion

An enterprise architecture needs the ability to align its governance processes to the varieties of value it chooses to create for its customers. The paper argues that this involves being able to align underlying infrastructures through processes of collaboration that cut across the existing accountability hierarchies of collaborating enterprises, spanning multiple architectures.

These distributed collaborations across complex systems of systems are necessary in order to respond to heterogeneous demands from diverse customers' contexts. Identifying the limitations of Zachman in supporting this form of analysis helps us to define additional abstractions and perspectives that can overcome those limitations. The analysis of these extended models in terms of their stratification then provides the means of identifying the distributed forms of collaboration needed to respond to heterogeneous demands.

References

[1] Philip Boxer, Edwin Morris, Dennis Smith, and Bill Anderson, "The Double Challenge in Engineering Complex Systems of Systems." news@sei, 2007, 5. http://www.sei.cmu.edu/news-at-sei/columns/eye-on-integration/2007/05/eye-on-integration-2007-05.htm (2007)

[2] E. Jaques, "In Praise of Hierarchy," Harvard Business Review, vol. 68 (1), pp. 127-33, Jan-Feb 1990.

[3] Roger Harrison and Herb Stokes, Diagnosing Organizational Culture. San Francisco: Pfeiffer, 1992.

[4] M. Treacy and F. Wiersema, The Discipline of Market Leaders; Choose your customers, narrow your focus, dominate your market. London: Harper Collins, 1995.

[5] C. K. Prahalad and V. Ramaswamy, The Future of Competition: Co-Creating Unique Value with Customers. Boston: Harvard Business School Press, 2004.

[6] Philip Boxer, The Double Challenge, March 2006. http://www.asymmetricdesign.com/archives/26.

[7] S. L. Goldman, R. N. Nagel, and K. Preiss, Agile Competitors and Virtual Organizations: strategies for enriching the customer. New York: Van Nostrand Reinhold, 1995.

[8] Philip Boxer, et al. SoS Navigator 2.0: A Context-Based Approach to System-of-Systems Challenges (CMU/SEI-2008-TN-001). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008.

[9] M. W. Maier, "Architecting Principles for Systems-of-Systems," Systems Engineering, vol. 1 (4), pp. 267 - 284.

[10] D.B. Robi, Enterprise DoD Architecture Framework and the Motivational View, CrossTalk April 2004.

[11] John A. Zachman, "A Framework for Information Systems Architecture," IBM Systems Journal, vol. 26 (3), pp. 276-292, 1987.

[12] J. Zachman, "Architecture is Architecture is Architecture," EIMI Archives, vol. 1 (1), March 2007.

[13] P. Clements and L. Northrop, Software Product Lines: Practices and Patterns. New York: Addison-Wesley, 2002.

[14] S. Blanchette, P. Clements, M. Gagliardi, and J. Klein, U.S. Army Workshop on Exploring Enterprise, System of Systems, System, and Software Architectures (CMU/SEI-2008-TR-023). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2008.

[15] P. Clements, Exploring Enterprise, System of Systems, and System and Software Architectures, SEI Webinar, January 2009.

[16] L. Northrop, et al., Ultra-Large-Scale Systems: The Software Challenge of the Future. Pittsburgh, PA; Software Engineering Institute, Carnegie Mellon University, June 2006.

[17] J. D. Smith, The use of influence maps to understand systems-of-systems programmatics, SSTC 2009 (in press)

[18] William Anderson, Philip Boxer, and Lisa Brownsword, An Examination of a Structural Modeling Risk Probe Technique (CMU/SEI-2006-SR-017). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University, 2006.

[19] JFSP Software Tools and Systems Study Approach, Joint Fire Science Program, September 2007. http://www.firescience.gov/documents/Software_Tools_Systems_Study/Brief%20JFSP%20STS%20Approach%20September%2020%202007.pdf

[20] R. Hopkins and K. Jenkins, Eating the IT Elephant: Moving from Greenfield Development to Brownfield. Upper Saddle River, NJ: IBM Press, 2008.

[21] Y. Cai and K. J. Sullivan, Modularity Analysis of Logical Design Models, Proceedings of 21th IEEE/ACM International Conference on Automated Software Engineering. Tokyo, JAPAN, September 18-22, 2006.

[22] William Anderson and Philip Boxer, "Modeling and Analysis of Interoperability Risk in Systems of Systems Environments," CrossTalk, pp. 18-22, November 2008.

[23] M. W. Maier, "System and software architecture reconciliation," Systems Engineering, vol. 9 (2) , pp. 146-159, May 2006.

[24] P. J. Boxer, "The stratification of cause: when does the desire of the leader become the leadership of desire?" Psychanalytische Perspektieven, vol. 32/33, pp. 137-159, 1998.