Exploring the concept of architecture in Technology and Organization studies

Polyxeni Vassilakopoulou [1], Miria Grisot [1]
[1] University of Oslo, Postboks 1080 Blindern, 0316, Oslo, Norway
{xvasil, miriag }@ ifi.uio.no

Abstract. In this paper we explore the use of the concept of architecture in Technology and Organization studies. We identify three discourses on architectures: one concerned with relationships among technical objects, one extended to cover sociotechnical relationships, and one where architectures themselves are the object of study (a discourse where there is an explicit strategic interest on the effect of architectures). Finally, we trace how different conceptualizations relate to different concerns related to change..

Keywords: Architecture, software, enterprise architecture, communication, control, complexity

1 Introduction

The notion of architecture conveys the idea that a set of elements are clustered into forms that are more or less stable and relate among themselves following a predefined logic. When thinking processes of change, "architectures" are frequently objectified as enabling or constraining factors: some architectures are viewed as better than others for accommodating change (robust architectures) or even for prompting change (generative architectures). However, the concept of 'architecture' is considered by many researchers as ill-defined. For instance, Scheil (2008) describes architecture as being '*a plastic concept ... a metaphorical idea that shapes the categories, discourse and language used*' [1]; Smolander et al (2008) identify multiple metaphors that describe different meanings of architecture as perceived by various actors, who participate in the creation and use of software, and they argue that "*Architecture, thus serves as a shared boundary object (Star, 1989; Star & Griesemer, 1989; Bowker & Star, 1999) between various stakeholder groups engaged in systems development, satisfying their varying informational needs during the systems development process*"[2]; similarly, Bidan et al (2012) stress how: "*The notion of an architecture is problematic in part because it seems to be usefully ambiguous and is often used at a high level of abstraction where anyone can agree that it is a useful concept*" [3].

Although conceptually elusive, architectures are central when discussing processes of stability and change. For example, the US National Research Council in a recent report argued: "*organizations should architect healthcare IT for flexibility to support disruptive change rather than to optimize today's ideas about healthcare*" (Committee on Engaging the Computer Science Research Community in Health Care

Informatics 2009). And, in the much sited book of Cummings and Worely on Organizational Development and Change [4] it is claimed that interventions need to address "*the organization's architecture" (*when discussing interventions that transform the way organizations relate to its environment or operate internally).

The aim of this paper is to explore the conceptualization of 'architecture' in technology and organization studies and to trace how different conceptualizations relate to different concerns related to change. Therefore we ask: how is architecture conceptualized and how it relates to change in technology and organization studies? By identifying and expressing the differences in the content of the word architecture and by tying these differences to distinct change concerns, we hope to contribute a sharper understanding of a notion that remains ambiguous although extensively used.

The paper is structured in the following way. First, we present our method. Then, we provide a brief overview of how the term architecture has been used in the literature we examined. More precisely, we identify three discourses on architecture: one concerned with relationships among technical objects, one extended to cover sociotechnical relationships, and one where architectures themselves are the object of study (a discourse where there is an explicit strategic interest on the effect of architectures). Finally, in the discussion section we point to the different roles of architecture in the three discourses.

2 Method

In order to investigate the use of the concept of architecture in technology and organization studies we have conducted a search in google scholar with the keyword architecture, information system, organization, technology. We have then ordered the search results according to their citation index. In the review we focus on how authors have conceptualized architecture, and how the term was used in the text for instance in association with other key terms. We have started grouping publications into two main categories, one dealing with technical issues of architectures, and one dealing with architecture in the context of enterprise and business models. These two ways of using the term architecture covered the majority of the studies. However, we also considered studies that come from literature thematising strategy, innovation and organizational change. While these studies may not strictly deal with implementation of technologies, we found they opened up a different discourse on architecture than the ones proposed by technical and business models studies. We labeled these studies 'strategic approaches'. We have then further elaborated the profile of the three conceptualizations and identified the most cited work (classics) within each.
Our subsequent analysis of the collected material is informed by a meta-level discussion on the use of the concept of architecture as a communication tool. For instance, much of the literature on software architecture we have reported has discussed issues of multiple views, problems of communication between stakeholders, and the lack of one 'single' understanding of architecture in software projects due to 'multiplicity of structures'. Such discussions seem to point to a function of architecture as '*boundary object*' [5] among different communities. As Bass et al. claim, an architecture is a set of structures *to reason about a system* [6], however such

reasoning is a collective endeavor of an heterogeneous community. Accordingly, Smolander et al claim that research on software architecture may benefit from the work in fields such as CSCW where the topic of reconciling different views is discussed, and the concept of boundary object is one of the core concepts of the research field [2]. Gorton [7] stresses how the architecture is an abstract description of the system that "has to" employ abstractions in order to be understandable by the team members and project stakeholders. Abstractions allow to black box components in order to focus on their external properties, and allow flexibility in composition and decomposition work. Similarly, boundary objects are organic infrastructures that arise due to the 'information and work requirements' of the group cooperating [8].
A similar point is raised by Scheil when he argues that IT architecturing is a learning process as "*one or more meaningful interpretations of the continuous organizing, emerging from the interrelationships between a sociotechnical system's parts*" [1]. Here, the emphasis of architecture/ing is not on the relationships between various parts of a larger system, but on the human interpretation and understanding of these relationships (a sense making process with reference to [9]). Further Scheil argues that this view recognizes technology as "*intentional and subjective in its nature, and not an objective instrument, where the tangible use determines its position*" [1]. He sees architecturing as indicating the necessary role that humans play when mediating between various architectural semantics such as business architecture, and software architecture.

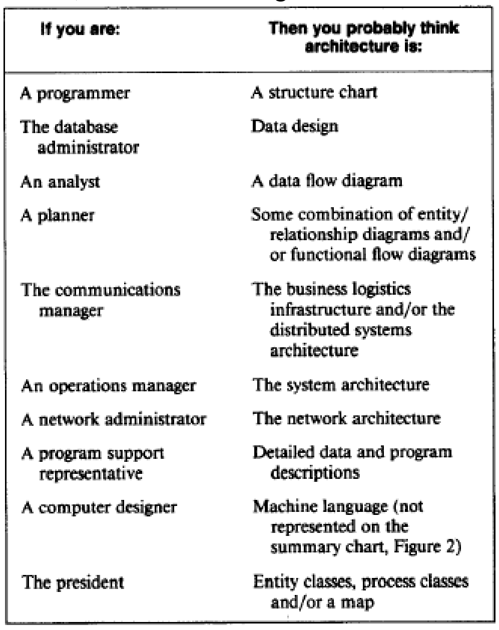
In the line with this meta-level discussion, we have approached the reviewed literature on architecture with attention to how the concept of architecture identified in the three profiles conveys the understanding of change.

3 A Technical View: Components and their Interrelations

The common understanding of 'software architecture' is that of structure describing systems' components, their operational principles and their interconnections. An example is the classic definition given by Bass et al. (1998): "*The software architecture of a program or computer system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them"[10].* In this understanding the term architecture has a structural and technical connotation and indicates the components of a system and their arrangement. The basic syntactic elements of an architectural description are components, connectors, and configurations of components and connectors. *Components* are the active, computational entities of a system that accomplish tasks through internal computation and external communication with the rest of the system via a collection of interaction points defined as *ports*. *Connectors* define the interaction between components and each connector provides a way for a collection of ports to come into contact in addition to defining the protocol through which a set of components will interact. A *configuration* is a collection of component instances which interact by means of connector instances [11].

In a more recent work, Bass et al define software architecture as "*the set of structures needed to reason about the system, which comprises software elements, relations among them, and properties of both*" [6]. It follows that software systems are composed by many structures, and no single structure is *the* architecture of the system. It also follows that a structure is architectural if it supports reasoning about the system and the system's properties.

A main problem discussed in software architecture literature is that of architectural representations. In one of the classic works, Zachman (1987) is concerned with developing awareness on the use of representations of software architecture for improving professional communication among owners, designers, and builders in the process of building complex information systems[12]. He shows how in this process a variety of representations are created at different levels of detail, are different in nature, content and semantics, forming a set of multiple architectural representations depending on different roles, as illustrated in figure 1.



If you are:	Then you probably think architecture is:
A programmer	A structure chart
The database administrator	Data design
An analyst	A data flow diagram
A planner	Some combination of entity/ relationship diagrams and/ or functional flow diagrams
The communications manager	The business logistics infrastructure and/or the distributed systems architecture
An operations manager	The system architecture
A network administrator	The network architecture
A program support representative	Detailed data and program descriptions
A computer designer	Machine language (not represented on the summary chart, Figure 2)
The president	Entity classes, process classes and/or a map

Figure 1: Different architectural representations for different roles [12]

Another classic work is the one by Krutchten on architecture views [13]: he identifies a way of describing an architecture based on four views: logical, process, physical, development views. These views describe the system from different perspectives according to different users' needs, and a 'scenario' view overlaps the others and relates the design to its context. These views represent different stakeholders' interests as a set of coherent, logical, harmonized descriptions. However, the discussion in the field has been directed mainly at representing and documenting a system's architecture from different perspectives without really offering a detailed description of the rationale that guides the architecting process [14]. For instance, Smolander et al (2008) argue for broadening the focus of current approaches to representing, designing and communicating software architectures in a way to support the role that different stakeholders play in the creation and use of software architecture [2]. Based on a study on three-software producing organizations, they have focused on understanding how the different stakeholders generate, represent, use and share knowledge regarding software architectures. Their findings stress the coexistence of different views on software architecture, and recognize the need for addressing the communication needs of stakeholders. Bosch (2004) argues that one of the key challenges of the software architecture community has been that software architectures need to be designed carefully because changes after the initial design are typically very costly. Software architectures change independent of how carefully they are designed. Thus, he argues that "*Rather than components and connectors, we need to model and represent a software architecture as the composition a set of architectural design decisions, concerning, among others, the domain models, architectural solutions, variation points, features and usage scenarios that are needed to satisfy the requirements"*[15]. Thus, software architectures should be represented in several phases of the lifecycle. Finally he redefines software architecture as the *composition of architectural design decisions*, and not a set of components and connectors (Idem.).

4 Extended Conceptualizations: Enterprise Architectures

Going beyond the technical focus, the architecture notion has also been used to map holistically relationships among heterogeneous components that together constitute purposeful work systems. The term "enterprise or business architecture" has been used to encompass sociotechnical arrangements of software, hardware, organizational structures, human competences and incentive schemes, [16-19]. These architectures address enterprise-level objectives like efficiency and effectiveness and mirror managerial choices for the desired level of standardization and integration within large scale work systems: "*The enterprise architecture is the organizing logic for business process and IT capabilities reflecting the integration and standardization requirements of the firm's operating model"[19].*

The use of architectural maps to associate diverse components of complex work-oriented sociotechnical arrangements is not new. The increasing complexity of work settings has triggered the development of numerous reference models and design frameworks for sociotechnical architectures during the last 30 years (see figure 2). These are either industry specific like: "Computer Integrated Manufacturing Open System Architecture- CIMOSA" (aims to support the enterprise integration of machines, computers and people in computer integrated manufacturing settings) and, "Purdue Enterprise Reference Architecture-PERA" (provides a consolidated view of production facilities, people/organization, and information systems) or generic such as GERAM (Generalized Enterprise Reference Architecture and Methodology), ARIS (Architecture of Integrated Information Systems) that covers process design, management, work flow, and application processing, or TOGAF (The Open Group Architecture Framework) that aims to encompass simultaneously: business strategy, governance, organisation, processes, data structures, applications, hardware capabilities and standards [20-25]. All these architectural models and frameworks support the figurative definition of relationships but at the same time they aim to serve as control tools. They aim to put in place specific rigid regulatory setups that will guide the development of novelty into specific directions. Following this fully planning ideal, the role of the "architect" within specific project teams is then stripped from ingenuity and resourcefulness connotations and is reduced to a role similar to the one that compliance officers have. In their book "Enterprise Architecture as Strategy", Ross, Weill and Robertson write: "*project architects are responsible for ensuring that individual projects are compliant with technology standards and that related projects reuse technologies as appropriate. If a project architect feels an exception to the standards is warranted, he or she either seeks approval from one of the assistant vice presidents authorized to grant exceptions or refers the request to the architecture committee*" [19]. Segars and Grover 1994 argue: "*unlike bottom up approaches which typically emphasise the information needs of traditional functional areas, the architecture approach is 'top down' in nature" [26].*

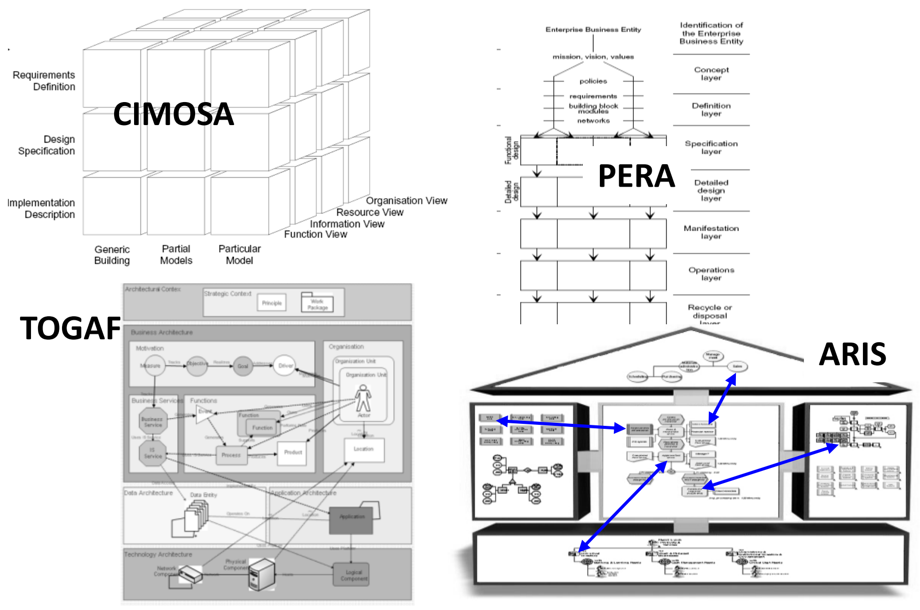


Figure 2: Schematic of different frameworks for enterprise architectures

Today, within informatics there is a lot of discussion on "ecosystems" within which different technical components evolve following trajectories that are not dictated by traditional hierarchies and are not mere instantiations of fully pre-planned courses [27-29]. To cater for this new situation, a recent Gartner report puts forward the new concept of panarchitecture: "*Network-centric and biologically based dynamic models are needed in order to design solutions, enterprises and industries that are more resilient in the face of transformative change, especially unforeseen transformations. Such models include "panarchy" from ecological science, "hyperconnected networks" from network science and "relationship coordination"* *from organizational science. These emerging models will be integrated using hybrid thinking to complement enterprise architecture with a new paradigm of renewal known as panarchitecture*". [30]

Table 1. Definitions of architecture.

Term	Definition	Reference/Source
Software architecture	The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.	ISO/IEC 42010
Software architecture	Software architecture involves the structure and organization by which modern system components and sub-systems interact to form systems, and the properties of systems that can best be designed and analyzed at the system level.	[14]
System architecture	The underlying structure of a system, such as a communication network, a neural network, a spacecraft, a computer, major software or an organization.	[31]
Information systems architecture	Architecture is relative depending on 'who' you are (see also figure 1).	[12]

5 Strategic Approaches

In the context of inter-organizational innovation, Andersson, Lindgren and Henfridsson study a process of architectural knowledge development [32]. Architectural knowledge is defined as "*knowledge developed and enacted in innovation processes of aligning heterogeneous business and technical elements*". This definition draws from n Henderson and Clark's theory of architectural innovation. According to Henderson and Clark "*the essence of architectural innovation is the reconfiguration of an established system to link together existing components in a new way*" [33]. Thus, in this view innovation is the result of changes in assembling already existing components that remain the same, while radically improving customer value and satisfaction. It refers to those: "*innovations that use* *many existing core design concepts in a new architecture and that therefore have a more significant impact on the relationships between components that on the technologies of the components themselves*" (idem.). Andersson et al. (2008) define architectural knowledge as a specific type of knowledge concerning how to draw linkages between each of the components. In the following paragraphs we elaborate on three important architectural knowledge concepts, namely: modularity, layering and end- to-end topologies.

Henderson & Clark have identified modularity as a key concept for architectural innovation. This is a concept initially introduced by Simon who identified the property of "near decomposability" of complex systems [34]: "*Such systems consist of a hierarchy of components, such that, at any level of the hierarchy, the rates of interaction within components at that level are much higher than the rates of interaction between different components. Systems with this property are called nearly completely decomposable, or more briefly, nearly decomposable (ND). The explanation for the ubiquity of the ND property is that, under the usual conditions of mutation and/or crossover and natural selection, ND systems will increase in fitness, and therefore reproduce, at a much faster rate than systems that do not possess the ND property*". Nearly decomposable systems tend to be better in adapting to their environment than systems that do not exhibit this property. The importance of architectural modularity has been identified repeatedly by design theorists from various fields such as: civil architecture [35, 36], product design [37], software development [38], innovation studies [39], developmental economics [40, 41]. Aiming for architectural modularity is a key strategic direction in the design of robust heterogeneous arrangements that include significant technological components, but this is far from straightforward. The aim for Service Oriented Architectures [42, 43] where the modular structure consists of services is an exemplary case of applying the modularity concept .

Layering is an architectural concept frequently used complementarily to modularity when developing strategies for complex interconnected systems. Layering has been traditionally employed by software engineers that came up with the concept of multi-tier architectures to disentangle the complexity of multiple interconnected components. Layering can also help reconcile different timeframes and lifecycles of various components within the same system. The "shearing layers" idea from traditional architecture conceptualizes buildings as a set of components that evolve in different timescales [44]. Similarly, complex interconnected systems can be viewed as sets of components with qualitatively different rates and scales of change [45, 46] and be constructed in shearing layers, with a clear demarcation between parts that should change at different rates.

Finally, a third concept that contributes to our architectural knowledge is the topological "end-to-end" concept. An "end-to-end" architecture allows control devolution to the periphery: "the ends". This allows multiple interconnected components to be gradually replaced, expanded or eliminated without threatening the survival of the whole system. The "end-to-end" architecture is widely held to be the key reason for Internet's flexibility and strong generative capacity [47-49]. What's more, applying an "end-to-end" architecture means acknowledging that no centralized authority and no well-prepared plan could possibly synthesize all of the knowledge required and anticipate all possible needs.

6 Discussion

In this paper, we have explored the concept of architecture and its use in IS literature. The software architecture literature mainly uses the term to indicate technical objects (components) and their combination. In this view it is possible to predefine relations between components by defining the 'architecture' of the system. Encompassing both technical and non-technical components, enterprise architecture literature discusses how to map and structure relations among heterogeneous components. In this approach components are of sociotechnical nature: hardware but also organizational processes including people performing them. The last stream we have described deals with an even higher level of complexity and relates architecture/ing to technical elements, sociotechnical elements and strategic interests such as innovation or generativity. However, moving beyond a mere understanding of the elements characterizing each architectural view, we now trace how different conceptualizations relate to different concerns related to change. .

1. Architecture as a way of determining relations – Design. Much of the literature on software architecture sees architecture as a prescriptive concept that defines how artifacts should be realized [50]. Gorton sees a software architecture representing a *complex design artifact*[7]. The term architecture is used to signify a coherent set of principles and rules that guide design. All these principles and rules are about the relationships among components that constitute an identifiable bounded technical system with a certain operational purpose. A requirement for the system itself might be that it will handle turbulences and variation (external change). This is a reactive stance on change, centered on the designed capabilities of the technological object.

2. Architecture as a way to control evolution – Regulation. In the enterprise architecture literature 'the architect' is responsible for controlling the compliance of novel endeavors to technical standards, business objectives and operational archetypes and architecture has a 'top-down' regulatory role. Architectures are exhaustive, fully inclusive blueprints that cover all aspects of purposeful work systems. Even though this "control ideal" has been proved unrealistic in practice for large scale systems with significant social components, it is not abandoned. Recent literature that adopts "ecosystem" views still pursues ways to put in place mechanisms and enabling/constraining structures that can make systems to have desirable behaviors (e.g. be more resilient). These new approaches draw from network theories and biological archetypes in order to pursue new types of control [30, 51].

3. Architecture as a way of handling complexity – Sensemaking. In the literature that focuses on the design of multilevel, heterogeneous and large-scale systems (within economy, society, and technology) architectural knowledge is put forward as means for complexity handling. Architectural notions such as *modularity*, *layering* and *end-to-end* topological arrangements have been proposed as ways to conceptualize disentanglement and power devolution for complexity containment. This literature discards the "control ideal" [52, 53] but acknowledges the influence of architectural choices. The emphasis is not on preparing complete maps of components and interrelationships but on applying architectural principles and developing an awareness of the choices that are already in place: "*An explicit understanding of the underlying architecture is a prerequisite for the design, evolution and maintenance of modern information systems that must complement today's complex business processes spread across internal divisions and external partners*" [2]. Smolander et al. go as far as proposing that multiple dissimilar views of architecture can be maintained simultaneously to accommodate complex relationships and multiple perspectives: "*the work by Ciborra (2000) may provide another pointer to deriving architectural specifications that take into account their emergent and improvized nature as opposed to carefully planned common road maps (i.e., maintaining multiple and even conflicting views of architecture simultaneously.*" (idem).

7 Conclusion

In conclusion, in this paper we have explored the concept of architecture and its use in technology and organization studies. We have identified three discourses on architectures: one concerned with technical objects, one extended to cover both technical and non-technical components of enterprises, and one developing an explicit strategic interest in architectures. Based on an understanding of architecture as communication tool, we have then examined the different views with a focus on understanding change. In the first view, architectures are used in order to bring into existence something that did not exist previously: prescriptive architectures are used to build new technological artifacts with predefined behavior. In the second view, architectures are related to a regulatory concern. The purpose here is to control changes within complex sociotechnical arrangements (such as enterprises). Finally, in the third view, architectures serve as sensemaking instruments that can support complexity handling. Taken together, the identified range of architecture related conceptualizations can help developing a sharper understanding of a notion that remains ambiguous although extensively used.

References

1. Scheil M. IT Architecturing: Reconceptualizing Current Notions of Architecture in IS Research. ECIS 20082008.
2. Smolander K, Rossi M, Purao S. Software architectures: Blueprint, Literature, Language or Decision? European Journal of Information Systems. 2008;17(6):575-88.
3. Bidan M, Rowe F, Truex D. An empirical study of IS architectures in French SMEs: integration approaches†. Eur J Inform Syst. 2012;21(3):287-302.
4. Cummings TG, Worley CG. Organization development & change: South-Western Pub; 2009.
5. Star SL, Griesemer JR. Institutional ecology,translations and boundary objects: Amateurs and professionals in Berkeley's Museum of Vertebrate Zoology, 1907-39. Soc Stud Sci. 1989;19(3):387-420.
6. Bass L, Clements P, Kazman R. Software architecture in practice: Addison-Wesley Professional; 2012.

7. Gorton I. Essential software architecture: Springerverlag Berlin Heidelberg; 2011.
8. Star SL. This is not a boundary object: Reflections on the origin of a concept. Science, Technology & Human Values. 2010;35(5):601-17.
9. Weick KE. Making Sense of the Organization. Oxford: Blackwell Publishing; 2001.
10. Bass L, Clements P, Kazman RSAiP. Software Architecture in Practice. Reading, MA: Addison Wesley; 1998.
11. Shaw M, DeLine R, Klein DV, Ross TL, Young DM, Zelesnik G. Abstractions for software architecture and tools to support them. Software Engineering, IEEE Transactions on. 1995;21(4):314-35. doi: 10.1109/32.385970.
12. Zachman JA. A framework for information systems architecture. IBM systems journal. 1987;26(3):276-92.
13. Kruchten P. The 4+ 1 view model of architecture. Software, IEEE. 1995;12(6):42-50.
14. Kruchten P, Capilla R, Dueas J. The decision view's role in software architecture practice. Software, IEEE. 2009;26(2):36-42.
15. Bosch J. Software architecture: The next step. Software architecture: Springer; 2004. p. 194-9.
16. Gharajedaghi J. Systems thinking: Managing chaos and complexity: A platform for designing business architecture. Burlington, MA: Elsevier Science; 1999.
17. Nevo S, Wade MR. The Formation and Value of IT-Enabled Resources: Antecedents and Consequences of Synergistic Relationships. Mis Quart. 2010;34(1):163-83.
18. Uhl-Bien M, Marion R, McKelvey B. Complexity Leadership Theory: Shifting leadership from the industrial age to the knowledge era. The Leadership Quarterly. 2007;18(4):298-318. doi: 10.1016/j.leaqua.2007.04.002.
19. Ross JW, Weill P, Robertson D. Enterprise architecture as strategy: Creating a foundation for business execution: Harvard Business Press; 2006.
20. Kosanke K. CIMOSA--Overview and status. Comput Ind. 1995;27(2):101-9.
21. Doumeingts G, Vallespir B, Darricau D, Roboam M. Design methodology for advanced manufacturing systems. Comput Ind. 1987;9(4):271-96.
22. Williams TJ. The Purdue enterprise reference architecture. Comput Ind. 1994;24(2-3):141-58.
23. Schmidt C, Buxmann P. Outcomes and success factors of enterprise IT architecture management: empirical insight from the international financial services industry. Eur J Inform Syst. 2011;20(2):168-85.
24. Kozina M. Evaluation of Aris and Zachman frameworks as enterprise architectures. Journal of Information and Organizational Sciences. 2006;30(1):115-36.
25. Josey A, Harrison R, Homan P, Rouse M, Sante van T, Turner M, et al. TOGAF Version 9: A Pocket Guide. Zaltbommel, NL: Van Haren Pub; 2009.
26. Segars A, Grover V. Communications architecture: towards a more robust understanding of information flows and emergent patterns of communication in organizations. European journal of information systems. 1994;3(2):87-100.
27. Bosch J, editor From software product lines to software ecosystems. Proceedings of the 13th International Software Product Line Conference 2009; San Francisco, CA: Carnegie Mellon University.
28. Messerschmitt DG, Szyperski C. Software ecosystem: understanding an indispensable technology and industry. MIT Press Books. 2005;1.
29. Jansen S, Finkelstein A, Brinkkemper S, editors. A sense of community: A research agenda for software ecosystems2009: Ieee.
30. Gall N. From Hierarchy to Panarchy: Hybrid Thinking's Resilient Network of Renewal. Stamford, CT: Gartner; 2010. 27 p.
31. Rechtin E. Systems Architecting: creating and building complex systems. Upper Saddle River, NJ: Prentice Hall 1991.

32. Andersson M, Lindgren R, Henfridsson O. Architectural knowledge in inter-organizational IT innovation. The Journal of Strategic Information Systems. 2008;17(1):19-38.
33. Henderson RM, Clark KB. Architectural innovation: the reconfiguration of existing product technologies and the failure of established firms. Admin Sci Quart. 1990;35:9-30.
34. Simon HA. The architecture of complexity. Proceedings of the American philosophical society. 1962;106(6):467-82.
35. Alexander C. Notes on the synthesis of form. . Cambridge, MA: Harvard University Press.; 1964.
36. Alexander C, Coplien JO. The origins of pattern theory: The future of the theory, and the generation of a living world. Ieee Software. 1999;16(5):71-82. PubMed PMID: ISI:000082851000017.
37. Schilling MA. Towards a General Modular Systems Theory and its Application to Interfirm Product Modularity. Acad Manage Rev. 2000;25(2):312-4.
38. Parnas DL. On the Criteria to Be Used in Decomposing Systems into Modules. Commun Acm. 1972;15(12):1053-8. PubMed PMID: ISI:A1972O176500009.
39. Hippel von E. Task Partitioning: An Innovation Process Variable. Res Policy. 1990;19:407-18.
40. Langlois RN. Modularity in Technology and Organization. Journal of Economic Behavior and Organization. 2002;49:19-37.
41. Langlois RN, Robertson PL. Networks and Innovation in a Modular System: Lessons from the Microcomputer and Stereo Component Industries. Res Policy. 1992;21(4):297-313.
42. Channabasavaiah K, Holley K, Tuggle E. Migrating to a service-oriented architecture. IBM DeveloperWorks. 2003;16.
43. Vassiliadis B, Stefani A, Tsaknakis J, Tsakalidis A. From application service provision to service-oriented computing: A study of the IT outsourcing evolution. Telematics and Informatics. 2006;23(4):271-93.
44. Brand S. How buildings learn: What happens after they're built. New York, NY: Penguin Books; 1994.
45. Papantoniou B, Nathanael D, Marmaras N. Moving Target: Designing for Evolving Practice. In: Stefanidis, editor. HCI: Inclusive Design in the Information Society; Heraklion, Crete: Lawrence Erlbaum Associates, Mahwah; 2003.
46. Simmonds I, Ing D. A shearing layers approach to information systems development. In: Huhns M, Gasser L, editors. The structure of ill-structured solutions: boundary objects and heterogeneous distributed artificial intelligence 2000.
47. Zittrain JL. The Future of the Internet and How to Stop It. New Haven, CT: Yale University Press; 2008.
48. Agre P. P2p and the promise of internet equality. Communications of the ACM. 2003;46(2):39-42.
49. Abbate J. Inventing the internet: MIT press; 2000.
50. Hoogervorst J. Enterprise architecture: Enabling integration, agility and change. International Journal of Cooperative Information Systems. 2004;13(03):213-33.
51. Kephart J, Chess D. The vision of autonomic computing. Computer. 2003;36(1):41-50.
52. Garud R, Jain S, Tuertscher P. Incomplete by design and designing for incompleteness. Organ Stud. 2008;29(3):351-71.
53. Pollock N, Williams R. e-Infrastructures: How Do We Know and Understand Them? Strategic Ethnography and the Biography of Artefacts. Comput Support Coop Work. 2010;19:521-56.