Geographically Distributed Enterprise Architecting: Towards a Theoretical Framework

J. Alberto Espinosa
Kogod School of Business
American University
alberto@american.edu

Frank Armour
Center for Information Technology in the Global Economy (CITGE)
American University
frank.armour@att.net

Abstract

Enterprise architecting is becoming critical for most modern organizations whose competitive strategies are tightly linked to the underlying information technology (IT) infrastructure. The reason for this is that an enterprise architecture takes a holistic view of the business processes and functions and the information technologies supporting them, rather than the more detailed perspectives provided with application-by-application views. Our understanding of effective enterprise architecting activities is still evolving and this practice is replete with challenges. These challenges are further compounded by the fact that organizations are often geographically dispersed. Furthermore, business processes, technology infrastructure components, information and the people involved may be distributed in different geographic configurations, making it very difficult to comprehend their organization. In this article, we make a first attempt at providing a theoretical framework to guide our thinking for practice and research in this area. We build on the foundations of coordination theory and geographically distributed collaboration research.

1. Introduction

As noted in prior research [25], Circular A130 of the U.S. Office of Management and Budget describes enterprise architecture as "the explicit description and documentation of the current and desired relationships among business and management processes and information technology." Armour and colleagues defined an enterprise architecture as the "blueprint for creating enterprise-wide information systems" [4]. They argued that when an organization's information management system is conceptualized and designed system by system, the result is an eclectic set of patched-up legacy systems commingled with new systems. This is an inadequate strategy in today's dynamic environments, which demand flexible and scalable systems that are responsive to rapidly changing business and technological environments. For example, studies have shown that implementing and adhering to enterprise architecture standards enhances IT project outcomes like fewer system redundancies and better integration of applications and data [6]. On the other hand, without an enterprise architecture IT managers have no guidance on how to develop, enhance and scale up their systems to meet business needs and no shared vision on how these systems should evolve.

While having a sound and flexible enterprise architecture (EA) is critical to success in information-intensive and fast paced competitive environments, the process of developing and maintaining this architecture—i.e., "enterprise architecting"—is very complex. As Kaisler and colleagues have articulated, there are three important areas in which enterprise architecting can encounter critical problems: (1) modeling—i.e., when the enterprise architecture is conceptualized and designed; (2) management—i.e., ensuring that individual systems are implemented in compliance with the enterprise architecture and that they interoperate properly; and (3) maintenance—i.e., ensuring that the enterprise architecture evolves consistently with organizational needs [25]. Furthermore, architecting activities in any of these three areas involve managing multiple and complex interdependencies among organizational goals, work processes, business functions, information needs, information technology (IT) infrastructure and people [4].

We argue that it is precisely because of this complex web of interdependencies that coordination is critical to successful enterprise architecting. Coordination theory views coordination as "the management of dependencies among task activities" [31]. Therefore, this body of theory provides a useful lens to better understand enterprise architecting and help us begin to articulate a research framework to study it. At the same time, we also argue that this theoretical lens is not sufficient because many of the enterprise architecture dependencies in organizations often need to be managed across geographic boundaries, thus presenting unique challenges. Two such boundaries that have a substantial

1530-1605/08 $25.00 © 2008 IEEE

effect on system implementations are distance and time [7]. There is a good amount of research about these two boundaries in fields like virtual teams [36], geographically distributed system development [1], and geographically dispersed communication and coordination [26, 33], which can inform research in geographically distributed enterprise architecting.

In this article we develop an initial theoretical framework of geographically distributed enterprise architecting by integrating three different bodies of research: enterprise architecture, coordination theory and geographically distributed collaboration research. The integration of these three research areas is very important because each area has been recognized as important in the respective literature. But because enterprise architecting is often conducted in larger organizations, operating in geographically distributed environments, and because the related processes are replete with complex dependencies, it is necessary that we develop an integrated view of enterprise architecting simultaneously from these three perspectives. As far as we know, no attempts have been made in the research literature to address these issues. This presents an excellent opportunity to develop a sound framework to guide further studies of enterprise architecting. In the next 3 sections we discuss each of these 3 perspectives in more detail. In section 5 we integrate these perspectives into a unified theoretical framework. In section 6 we provide some concluding remarks.

2. Enterprise Architecture Research

Enterprise architecture provides a blueprint that helps develop a shared vision for the development of enterprise-wide information systems [4]. This shared vision provides some common ground about how the various organizational stakeholders communicate about information system issues and a frame of reference to make IT investment and implementation decisions. The main goal of the architecting process is to align the IT vision with business goals, objectives, functions and processes. Armour et. al [4] suggested that there are 5 important views of an enterprise architecture: business, work, function, information and infrastructure views. The business, work and function views describe the business processes, where these processes need to take place, and the corresponding functional requirements. The work view also describes who needs to do what and how these people need to communicate and coordinate. The information view describes the various information entities necessary to support these processes and how these entities relate to each other. The infrastructure view describes the underlying IT necessary to support these processes. For the purposes of our theoretical development, these can be consolidated into four more general components previously suggested for information system development efforts [29]: business processes (i.e., business and function), information, technology (i.e., infrastructure) and people (i.e., work, locations).

But enterprise architecting is more than just developing the architecture. According to Armour et al [5], it involves: (1) establishing a target vision for the architecture and other initial activities (e.g., scoping the project, building the team, etc.); (2) describing the baseline architecture—i.e., the current architecture we want to change—in terms of its business processes, information, technology and people; (3) describing the target architecture—also in terms of business processes, information, technology and people; (4) planning the transition from the baseline architecture to the target architecture; (5) planning the implementation of the architecture; and (6) the implementing individual systems according to the architecture.

Establishing the target vision for the enterprise architecture is important because it helps develop a shared understanding of the architecture at a high level of abstraction among the stakeholders. Developing such a vision is a complex process because it involves a wide variety of stakeholder groups—e.g., customers, users, company managers, system architects, system developers, etc.—each with a different view of what the architecture should look like. A coordinated high level vision of the architecture needs to emerge from these various views. Good communication is of paramount importance in this activity and this communication is likely to be frequent and interactive.

Describing the baseline architecture involves developing an inventory of: the current business processes; the information that needs to be maintained to carry out these processes; the current IT infrastructure and applications supporting the processes; and the people and locations involved in the processes. This activity will require a substantial amount of modeling and written documentation, but the communication and gathering of the necessary information is more likely to be structured and planned (compared to the vision development).

The artifacts (i.e., models and written documents) necessary to describe the target architecture are similar to those of the baseline architecture, except that the target architecture describes where the enterprise will be after the implementation of the architecture, rather than where it is now. However, the process of developing the target architecture is likely to be more complex because it will need to be agreed upon and validated by the stakeholders, which will require a substantial amount of frequent interaction. This validation will need to be done against the target vision discussed earlier.

Planning the transition from a baseline architecture to the target architecture is also a complex undertaking because it involves multiple projects and is affected by uncertainties and changing conditions in the business and technical

environments [3]. Furthermore, transition planning must carefully consider the interdependencies across each system implementation project and among the new systems being implemented and between the new and current systems that will not be updated. For example, one needs to evaluate how legacy systems being retained will interoperate with new systems being deployed. Another important aspect of transition planning is deciding on how and when each old system will be phased out and when each new system will be phased in so that the current business processes are not affected during the transition. The plan needs to describe how each current system will be changed and how new ones will be rolled out. This will depend on which system is being updated, migrated to a new infrastructure platform, or totally replaced [3]. In addition, if the enterprise architecture effort is part of a larger business process re-engineering effort, then the architecture transition effort needs to be synchronized to meet the needs of the re-engineering goals and plans. Naturally, all stakeholders need to be involved in this process and communication is likely to be frequent and interactive. In addition, there needs to be a substantial amount of technical and domain knowledge sharing during this process.

The target architecture implementation plan differs from the transition plan in that it contains more specific details about resources and activities necessary for the implementation. Once the transition plan is in place and agreed upon by stakeholders, the implementation plan needs to specify how individual systems will be implemented in compliance with the target architecture. This master implementation plan focuses on a general allocation of budgets, personnel, and technical resources for the various projects identified in the transition plan. The communication required during this phase is more structured and will require regular meetings of stakeholders to review these allocations and substantial written documentation. The last activity in the enterprise architecting process derives from the implementation plan and it involves the individual implementation of each specific system. In essence, this is not a single activity, but it involves a number of activities bundled as individual system development projects. These projects will proceed in the same fashion as any other system implementation project, but the overall system requirements are subordinate to the target architecture, and the implementation plans are subordinate to the enterprise architecture transition and implementation master plans. Communication and coordination requirements for each project is no different than any other system implementation project, except that all enterprise architecture stakeholders need to participate in this communication even if the specific system being implemented does not affect them directly.

In sum, enterprise architecting requires developing a high level vision of an architecture that is aligned with business goals, describing the baseline architecture, developing the target architecture, planning the transition to and implementation of the target architecture, and planning of individual information system development in accordance with the architecture. The description of baseline and target architectures require that those involved in the architecting effort develop accurate views of business processes, information, technology and people associated with the architecture.

3. Coordination Theory

Naturally, defining the baseline and target architecture will require the input of and agreement among many people and business units. As such, the process will be fraught with complex dependencies, which will need to be managed effectively, thus the importance of incorporating coordination theory into our research framework. In the next subsections we discuss the basic concepts behind coordination theory. In later sections we integrate this theory into our framework.

3.1 Foundations

Coordination theory defines coordination as "the management of dependencies among task activities" [30]. This definition is very useful because it implies that if task activities can be carried out independently, then coordination is less necessary or not necessary at all for collective performance. Malone and colleagues [31] extended this concept in a more interdisciplinary way by arguing that dependencies exist, not only among people, but also among processes and resources. For example, the work of a software developer depends on the work of a software architect, thus the need to coordinate their activities. Similarly, two pieces of software code that need to be integrated have a technical dependency with each other at their interface, which needs to be carefully managed. Finally, certain activities may be dependent on shared resources like budgets, personnel and equipment, such that their use needs to be properly coordinated.

To better understand the nature of these dependencies it helps to think in terms of the topology of dependencies formulated by Thomson [38], which includes, in increasing order of complexity: (1) independence—i.e., task activities can be carried out individually without coordination; (2) pooled dependency—i.e., task activities can still be carried out individually, but depend on and compete for the same pool of resources (e.g., shared budget, equipment, people); (3) sequential dependency—i.e., one task activity depends on another, but not the other way around, thus only the dependent activity needs to be coordinated; and (4) reciprocal dependency or interdependency—i.e., both activities are dependent on each other, thus requiring tight coordination.

One other distinction that is important to understand the concept of "coordination" is that this word can have a dual interpretation, depending on whether one is referring to the

process of coordinating—i.e., managing dependencies—or coordination outcomes—i.e., extent to which an activity was effectively coordinated. For example, when collaborators communicate about task issues they are coordinating—a process. But this doesn't ensure that the team will be well coordinated in the end. Once and if their dependent activities have been carried out in a synchronized fashion in accordance with project schedules and plans the activity can be said to have been successfully coordinated—an outcome. We now discuss coordination processes and outcomes in more detail.

3.2 Coordination Processes

The process of coordinating is an old problem in the classic organizational literature. March and Simon discussed coordination processes in detail way back in the 50's [32] and argued that coordination can be accomplished *"mechanistically"* or *"organically"*.

- *Mechanistic coordination,* also referred to as task programming or coordination by plan [32, 38, 39] is useful for task activities that are more routine and certain, which can be more easily programmed. The implementation and use of mechanisms like project schedules, interface specifications, plans, procedure manuals, and workflow automation are a few examples of mechanistic coordination processes. A PERT chart is a classic project scheduling tool that exemplifies this type of coordination.
- *Organic coordination,* also referred to as coordination through communication, by feedback or by mutual adjustment [32, 38, 39] is more effective with uncertain and non-routine task activities that cannot be coordinated mechanistically because conditions are often changing. This communication can take place in a number of ways, including informal and spontaneous interaction, formal and planned [28, 35], interpersonal or group interaction [39], and verbal or non-verbal interaction

More recent research on team cognition in the psychology and organizational literatures suggest that collaborators also coordinate implicitly through collective or shared knowledge [8, 9, 27]. Therefore, we also incorporate this type of coordination into our framework.

- *Shared Knowledge.* "Implicit coordination" has been described as "unspoken assumptions of what others in the group are likely to do" [41]. Individuals in a group develop these unspoken assumptions based on their knowledge of each other and the shared aspects of the collective task. Team cognition research suggests that when members of a group share knowledge about each others' task activities they can better plan their own activities in coordination with the activities of others. Similarly, when collaborators have knowledge about each other they can anticipate each other's actions more accurately and locate and access expertise more effectively. Research with software teams has provided evidence that knowledge sharing is critical to coordination success [13, 14, 19]. This body of research is quite extensive, but the main implication for our research is that shared knowledge helps members coordinate in two ways: directly because individuals can program their actions more effectively; and indirectly because this knowledge helps them coordinate organically in a more effective way—i.e., members have more common ground in their communication. Similarly, their use of mechanistic coordination artifacts becomes more effective because individuals better understand how to interpret and use plans, schedules, etc.

3.3 Coordination Outcomes

Coordination outcomes are the results of coordination processes. Thus, coordination outcomes represent the "state" of coordination, which need to be distinguished from the acts of coordinating employed to achieve that state. Coordination outcomes are easier to identify when coordination is lacking because coordination problems are often very evident. There are many types of coordination outcomes (or problems), but research in collaborative software development has identified three main types of coordination outcomes, which are important in IT projects: technical, temporal and process coordination [20].

- *Technical coordination* problems arise when different parts of a system don't integrate or interoperate well—i.e., technical dependencies have not been managed effectively. For example, if two systems work well individually and comply with the enterprise architecture but exhibit severe problems at their interface during system operation.
- *Temporal coordination* problems arise when the deliverables of different interdependent parts of the project are not completed on schedule. For example, if the two systems in the example above integrate and interoperate well but one of them was completed way behind schedule it may throw the entire architecting process into disarray and future system development activities may need to be re-planned.
- *Process coordination* problems arise when there are parts of the system or architecture development process that depend on other processes and these dependencies are not adhered to. In the case of enterprise architecting, process coordination problems can originate either because some established processes for the architecting process are not followed—e.g., priority conflicts, activities starting out of sequence, escalation issues, etc.—or because the architecture itself is out of synch with the business processes it needs to handle.

In sum, coordination is important for enterprise architecting because of the pervasiveness and complexity of interdependencies involved in the process. The process of coordinating enterprise architecting activities can be mechanistic, organic and/or cognitive, resulting in technical, temporal and process coordination outcomes. As we discuss in the next section, the effectiveness of coordination processes and the resulting outcomes are strongly affected by geographic dispersion.

4. Geographically Distributed Collaboration Research

The conventional wisdom about which coordination processes are more adequate for a particular task activity needs to be re-evaluated when architecting collaborators are separated by distance and time, and their ability to communicate (i.e., coordinate organically) effectively is impaired. We now discuss relevant aspects of geographically distributed collaboration research and we later integrate these concepts into our framework.

4.1 Bridging Global Boundaries

Information system development projects are increasingly carried out across global boundaries because of proximity to clients and skilled personnel [10] and offshore outsourcing resources [11]. This trend also applies to most enterprise architecting efforts. At the very least, many individual project implementations will be carried out across geographic locations, often across the globe, and in many cases the stakeholders, implementation teams, information, business processes and IT infrastructure will be spread out across several locations. Geographic boundaries create barriers that collaborators need to bridge to get the architecting job done. There are many types of geographical boundaries that can divide large groups of collaborators (e.g., cultural, organizational), but we are particularly interested in two boundaries—time and distance—which are most important in system implementations [7] because they have a substantial effect on how people communicate.

4.2 Geographic Distance

Much has been written about geographic distance in collaboration. The main effect of geographic distance is that it eliminates or severely impairs the benefits of co-presence. Even small distance separations can have strong negative effects on communication frequency [2]. Distance separation also affects the timeliness [21, 40] and richness [15] of communication because the media through which it takes place are less interactive and have less shared contextual references [12]. Geographic dispersion also reduces shared understanding [24] and common ground in communication [33] among collaborators. These arguments about the difficulties of coordinating work across geographic distance are consistent with the research literature on geographically distributed software teams showing that geographic separation causes increased coordination overhead and more substantial delays, partly because it takes longer to initiate contacts and resolve difficult issues [10, 23]. Because most of the effects of geographic distance have to do with communication, distance will have a strong influence on how collaborators coordinate organically in enterprise architecting activities. While the Internet has made it easier to collaborate across distance, lack of co-presence continues to present substantial barriers and challenges in large scale collaborative work [33].

4.2 Time Separation

Time separation affects primarily the synchronicity of communication [18]. Time separation may occur for several reasons (e.g., differences in work schedules, holidays, vacation time, telecommuting, etc.) the primary form of time separation in geographically dispersed collaboration is due to time zones [17]. In contrast to geographic distance—in which once individuals are separated the magnitude of the distance makes little difference—the magnitude of time separation strongly affects how collaborators can interact and manage their workflow. Larger time zone differences reduce the window of opportunity for synchronous interaction. Furthermore, when collaborators are scattered across multiple time zones, simply figuring out when to meet can become daunting tasks [17]. At the same time, and in contrast to geographic distance, time zone differences can be advantageous because work may be carried out in one site while members in other sites sleep overnight. As with distance, time zone differences affect how collaborators coordinate their work organically. In contrast to distance, the extent to which time separation will impair the ability to coordinate activities across sites will vary depending on how the workflow is organized, how predictable are the dependencies, and how much work is non-routine.

5. Integrated Framework

The discussion we have presented above leads us to formulate the research framework depicted in Figure 1. In this section we describe, elaborate and support our framework with a series of propositions. Because, to the best of our knowledge, there are no other research frameworks for geographically distributed enterprise architecting, we propose this framework as preliminary for discussion. Additional work will need to be done to further develop theory to guide research in this field, but we view this as a very important first step. We develop our framework in a backwards fashion, starting with coordination outcomes, followed by enterprise architecting process, then by coordination processes, and ending with task context factors that influence the effectiveness of coordination processes.

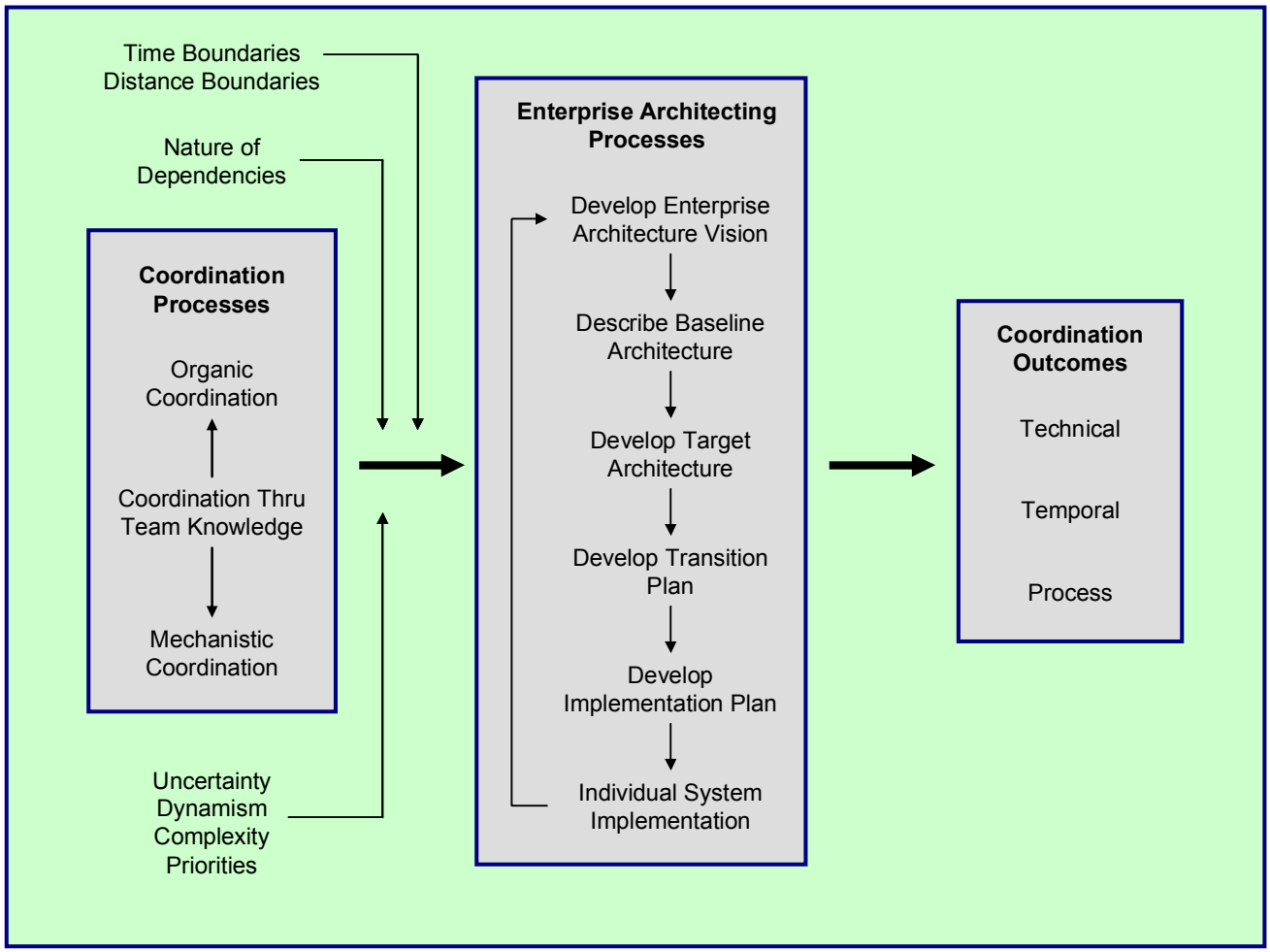

Figure 1: Geographically Distributed Enterprise Architecture Research Framework

5.1 Coordination Outcomes

From the perspective of our framework, the goal of the enterprise architecting process is to develop an enterprise architecture with minimal or no technical, temporal and process coordination outcome problems. Prior research has found that different types of task activities are more prone to some type of coordination problems than others. For example, a prior study found that in large scale software projects, technical personnel are more sensitive to technical coordination problems, whereas managers are more sensitive to temporal and process coordination problems [20]. We posit these findings can be extended to enterprise architecting work:

Proposition 1a: the importance of the difference types of coordination outcomes will vary for each of the six enterprise architecting activities presented in the framework.

The type of coordination outcomes that will matter the most for each enterprise architecting activity will depend on the type of dependencies that are more critical for that activity. Therefore, we argue the following: (1) technical coordination will be more important for implementation activities, whereas process coordination will be more important for conceptual activities; (2) temporal coordination is important for all activities, but its importance will be affected by the priority and time pressures of the architecting effort, more than the activities per se, and (3) temporal coordination is important between activities because each activity needs to start and end according to a schedule to avoid delaying the next activity. For example, temporal coordination is necessary between the activities involved in developing the transition and implementation plans (due to their sequential dependency) so that the start of the implementation plan's activities is not delayed. Thus we posit:

Proposition 1b: technical coordination is most important for implementation activities—i.e., describing the baseline architecture, developing the target architecture, and developing the implementation plan.

Proposition 1c: process coordination is most important for conceptual activities involving stakeholders—i.e., developing the enterprise architecture vision and developing the transition plan.

Proposition 1d: temporal coordination is most important between activities with sequential dependencies, so that one activity does not delay subsequent activities.

Proposition 1e: beyond proposition 1d, the importance of temporal coordination is dictated by the priority and time pressures of the architecting process.

Proposition 1f: because each individual system implementation is different, the importance of technical, temporal and process coordination will vary for each individual system project, depending on which type of dependencies are more salient in the respective project.

5.2 Enterprise Architecting Processes

A coordination strategy involves employing a particular mix of coordination processes. A specific coordination outcome can be accomplished with different coordination strategies. For example, two architecting subgroups may be effectively coordinated, with one of them using little communication and more mechanistic coordination with project planning tools, and the other relying on team knowledge and frequent communication. But we argue that an optimal mix of coordination processes will be one that is best suited for the particular nature of the task activities. Generally speaking, more interactive, uncertain and less routine activities will require more organic coordination. Similarly, more structured, certain, and routine activities will require more mechanistic coordination. Finally, activities that require more integration of knowledge from more stakeholders will benefit from more team knowledge. Thus we posit:

Proposition 2a: the optimal mix of coordination processes for different enterprise architecting activities will vary depending on the routine-ness, certainty, and equivocality of the architecting activity.

Proposition 2b: the optimal coordination mix for more equivocal tasks—i.e., developing the enterprise architecture vision, developing the target architecture, and developing the transition plan—will benefit more from organic coordination—i.e., communication.

Proposition 2c: the optimal coordination mix for more defined, certain tasks—i.e., describing the baseline architecture and developing the implementation plan—will benefit more from mechanistic coordination.

Proposition 2d: the optimal coordination mix for activities that require more integration of domain and technical knowledge—i.e., developing the enterprise architecture vision, developing the target architecture, and developing the transition plan—will benefit more from coordination through team knowledge.

Proposition 2e: the optimal coordination mix for individual system implementations will depend on the volatility and certainty of requirements of the individual systems.

5.3 Coordination Processes and Geographic Boundaries

The propositions we just articulated are generally applicable to enterprise architecting activities. However, because distance and time boundaries impair communication, we also posit that:

Proposition 3a: the optimal mix of coordination processes for any enterprise architecting activity will be affected by the presence of geographic boundaries—i.e., distance and time.

Geographic distance reduces or eliminates many of the benefits of co-presence—e.g., members of the architecting group don't know each other that well, interaction is less frequent and less spontaneous, it is more difficult to ascertain who is where and what is going on with the task, etc. Therefore, the reduced effectiveness of organic coordination due to distance separation can be offset with increased mechanistic coordination and team knowledge. Thus, we posit:

Proposition 3b: as more geographic locations are involved in the enterprise architecting process, the optimal coordination mix will benefit from less organic coordination and more mechanistic coordination and team knowledge. Team knowledge will help members coordinate implicitly and make their limited communication more effective.

Time separation changes the synchronicity of the communication of those involved in the enterprise architecting effort. As more time zones are represented in the group and as the time zone difference spanned by the group increases, the window of opportunity for synchronous interaction is reduced and members need to be more careful about structuring and timing their communication to be better synchronized with workflow activities in each location. Collaborators need to make conscious choices about when and how to communicate—i.e., wait to communicate synchronously with other sites during the overlapping time or communicate asynchronously outside of the overlapping period. The most predictable activities can be programmed so that task requests are sent by site A to site B to be completed while A sleeps. Conversely, if a task request is sent by A while B is sleeping it will introduce delay because nothing will be done until B comes to work. For less predictable activities, time zone differences often introduce delay because the

collaborators can't communicate as frequently and spontaneously. Thus, we posit:

Proposition 4: as more time zones and larger time zone differences are represented among those involved in the enterprise architecting effort, the effectiveness of organic coordination:

a) will not be severely affected for activities that require periodic communication that can be timed and programmed to meet workflow needs.

b) can be enhanced if dependencies are sequential and communication for task requests can be timed such that one site advances the work while the other site sleeps.

c) will be progressively and more severely affected for more equivocal and uncertain activities that contain reciprocal interdependencies, which require frequent and spontaneous interaction. The optimal mix of coordination processes for these activities will need to be much more heavily weighted on mechanistic coordination and team knowledge to offset this deficiency.

5.4 Other Context Factors

Not every enterprise architecting project is the same and the coordination needs of different projects will vary depending on the context of the architecting task. We have discussed two such factors to some extent throughout this paper, but it is important to discuss more specifically how these factors may affect the coordination mix. The first task context factor has to do with the nature of dependencies found in enterprise architecting work. As dependencies become more complex and tightly coupled, the importance of coordination increases. Thus, we posit:

Proposition 5a: as the enterprise architecting activity dependencies move from independent, to pooled, to sequential, to interdependent, the effect of the coordination processes on the effectiveness of the enterprise architecting process will be amplified—i.e., an interaction effect.

The second factor has to do with the uncertainty, dynamism, complexity, priorities and time pressures of the implementation, all of which will affect the importance of coordinating and the effectiveness of the respective coordination processes. Based on our arguments throughout the paper, we posit:

Proposition 5b: the higher the uncertainty of the enterprise architecting task, dynamism and volatility of the requirements of the task, complexity of the target architecture, and priorities and time pressures, the stronger the effectiveness of organic coordination processes will be, thus making communication more important and mechanistic coordination less important, which is typical of agile methodologies of system development.

6. Concluding Remarks and Future Research

We have proposed and formulated a research framework to guide us in our studies of geographically distributed enterprise architecting work. Our framework is by no means complete, but we hope that it will spark much needed discussion in this area. This framework has limitations because it has not been validated empirically and because it only incorporates distance and time boundaries. Further research is necessary to expand our framework to include other global boundaries. Despite this limitation, our proposed framework is soundly grounded on well established bodies of research—i.e., coordination theory and geographically distributed collaboration—and practice—i.e., enterprise architecting. This framework is a first step to providing theoretical foundations for studies in this area.

In order to further refine our framework and begin to validate parts of it we intend to conduct empirical studies, which will help us understand the coordination issues and challenges in geographically distributed enterprise architecting. We plan to carry out various studies using multiple methods so that we can acquire different perspectives of these issues. Because there is so little theoretical or empirical research in this area at this time, we intend to conduct our first study using qualitative methods based on observations and semi-structured interviews. We plan to collect and analyze our data using Grounded Theory method, which can provide rich descriptions of the phenomena observed [22, 37]. Grounded Theory has been widely as a qualitative method in information systems research [16, 34], particularly when the study is exploratory, and the theoretical development of the topic is in its early stages [34].

Our goal in this next phase of our research is to discover the main challenges that need to be overcome to develop, implement and maintain a geographically distributed architecture. Therefore, we will seek to interview several enterprise architects from organizations in multiple industries and sectors. We will use the theoretical framework proposed in this paper as a starting base to develop our semi-structured interview instruments and protocols. After we analyze the data, we will use the findings to verify which parts or our framework can be validated empirically and further refine the framework as needed. We then plan to use our findings and revised framework to help us develop hypotheses, constructs and variables to further validate our framework. We will then plan to use our revised framework and variables to test and further refine it with quantitative methods based on survey and archival data sources.

7. Acknowledgements

This study was supported by the Center for Information Technology in the Global Economy (CITGE) at the Kogod School of Business, American University.

8. References

[1] Agerfalk, P.J. and B. Fitzgerald, Flexible and Distributed Software Processes: Old Petunias in New Bowls? Communications of the ACM, 2006. 49(10): p. 27-34.

[2] Allen, T., Managing the Flow of Technology. 1977, Cambridge, MA: MIT Press.

[3] Armour, F.J. and S.H. Kaisler, Enterprise Architecture: Agile Transition and Implementation. IT Professional, 2001. 6(3): p. 30-37.

[4] Armour, F.J., S.H. Kaisler, and S.Y. Liu, A Big-Picture Look at Enterprise Architectures. IT Professional, 1999. 1(1): p. 35-42.

[5] Armour, F.J., S.H. Kaisler, and S.Y. Liu, Building an Enterprise Architecture Step by Step. IT Professional, 1999. 1(4): p. 49-57.

[6] Boh, W.F. and D. Yellin, Using Enterprise Architecture Standards in Managing Information Technology. Journal of Management Information Systems, 2007. 23(3): p. 163-207.

[7] Bullen, C. and J. Bennett, Groupware in Practice: An Interpretation of Work Experiences, in Groupware and Computer-Supported Cooperative Work: Assisting Human-Human Collaboration, R. Baecker, Editor. 1993, Morgan Kaufman Publishers: San Francisco, CA. p. 69-84.

[8] Cannon-Bowers, J.A. and E. Salas, Reflections on Shared Cognition. Journal of Organizational Behavior, 2001. 22(2): p. 195-202.

[9] Cannon-Bowers, J.A., E. Salas, and S. Converse, Shared Mental Models in Expert Team Decision-Making, in Individual and Group Decision-Making: Current Issues, J. Castellan, Editor. 1993, Lawrence Erlbaum Associates, Inc.: Hillsdale, NJ. p. 221-246.

[10] Carmel, E., Global Software Teams. 1999, Upper Saddle River, NJ: Prentice Hall.

[11] Carmel, E. and P. Tjia, Offshoring Information Technology: Sourcing and Outsourcing to a Global Workforce. 2005, Cambridge, U.K.: Cambridge University Press.

[12] Cramton, C.D., The Mutual Knowledge Problem and Its Consequences for Dispersed Collaboration. Organization Science, 2001. 12(3): p. 346-371.

[13] Crowston, K. and E.E. Kammerer, Coordination and Collective Mind in Software Requirements Development. IBM Systems Journal, 1998. 37(2): p. 227-245.

[14] Curtis, B., H. Krasner, and N. Iscoe, A Field Study of the Software Design Process for Large Systems. Communications of the ACM, 1988. 31(11): p. 1268-1286.

[15] Daft, R. and R. Lengel, Organizational Information Requirements, Media Richness and Structural Design. Management Science, 1986. 32(5).

[16] de Vreede, G.-J., N. Jones, and R.J. Mgaya, Exploring the Application and Acceptance of Group Support Systems in Africa. Journal of Management Information Systems, 1998-99. 15(3): p. 197-234.

[17] Espinosa, J.A. and E. Carmel, The Impact of Time Separation on Coordination in Global Software Teams: a Conceptual Foundation. Journal of Software Process: Practice and Improvement, 2004. 8(4): p. 249-266.

[18] Espinosa, J.A. and C. Pickering. The Effect of Time Separation on Coordination Processes and Outcomes: A Case Study. in 39th Hawaiian International Conference on System Sciences. 2006. Poipu, Kauai, Hawaii: IEEE.

[19] Espinosa, J.A., et al., Familiarity, Complexity and Team Performance in Geographically Distributed Software Development. Organization Science, 2007. 18(4): p. 613-630.

[20] Espinosa, J.A., et al., Team Knowledge and Coordination in Geographically Distributed Software Development. Journal of Management Information Systems, 2007. 24(1): p. 135-169.

[21] Gittell, J.H., Supervisory Span, Relational Coordination, and Flight Departure Performance: A Reassessment of Post-Bureaucracy Theory. Academy of Management Journal, 2001. 12(4): p. 468-483.

[22] Glaser, B.G. and A. Strauss, The Discovery of Grounded Theory: Strategies for Qualitative Research. 1967, Hawthorne, NY: Aldine de Gruyter.

[23] Herbsleb, J.D. and A. Mockus, An Empirical Study of Speed and Communication in Globally Distributed Software Development. IEEE Transactions on Software Engineering, 2003. 29(6): p. 481-494.

[24] Hinds, P. and S. Weisband, Knowledge Sharing and Shared Understanding in Virtual Teams, in Virtual Teams that Work: Creating Conditions for Virtual Team Effectiveness, S.G. Cohen and C.B. Gibson, Editors. 2003, Jossey-Bass: San Francisco, CA. p. 21-36.

[25] Kaisler, S.H., F.J. Armour, and M. Valivullah. Enterprise Architecting: Critical Problems. in 39th Hawaiian International Conference on System Sciences. 2005. Poipu, Kauai, Hawaii: IEEE.

[26] Kiesler, S. and J.N. Cummings, What Do We Know About Proximity in Work Groups? A Legacy of Research on Physical Distance, in Distributed Work, P. Hinds and S. Kiesler, Editors. 2002, MIT Press: Cambridge, MA. p. 57-80.

[27] Klimoski, R.J. and S. Mohammed, Team Mental Model: Construct or Metaphor. Journal of Management, 1994. 20(2): p. 403-437.

[28] Kraut, R.E. and L.A. Streeter, Coordination in Software Development. Communications of the ACM, 1995. 38(3): p. 69-81.

[29] Lee, G., W. Delone, and J.A. Espinosa, Ambidexterous Coping Strategies in Globally Distributed Software Development Projects. Communications of the ACM, 2006. 49(10): p. 35-40.

[30] Malone, T. and K. Crowston. What is Coordination Theory and How Can it Help Design Cooperative Work Systems. in Computer Supported Collaborative Work. 1990. Los Angeles, CA: ACM Press.

[31] Malone, T. and K. Crowston, The Interdisciplinary Study of Coordination. ACM Computing Surveys, 1994. 26(1): p. 87-119.

[32] March, J. and H.A. Simon, Organizations. 1958, New York: John Wiley and Sons.

[33] Olson, G.M. and J.S. Olson, Distance Matters. Human-Computer Interaction, 2000. 15(1): p. 139-179.

[34] Orlikowski, W., CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development. MIS Quarterly, 1993. 19(3): p. 309-340.

[35] Perry, D.E., N.A. Staudenmayer, and L.G. Votta, People, Organizations, and Process Improvement. IEEE Software, 1994. 11(4): p. 36-45.

[36] Powell, A., G. Piccoli, and B. Ives, Virtual Teams: A Review of Current Literature and Directions for Future Research. Data Base for Advances in Information Systems, 2004. 35(1): p. 6-36.

[37] Strauss, A. and J. Corbin, Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Second Edition ed. 1998, London: Sage Publications, Inc.

[38] Thompson, J., Organizations in Action. 1967, New York, NY: McGraw-Hill.

[39] Van de Ven, A.H., L.A. Delbecq, and R.J. Koenig, Determinants of Coordination Modes Within Organizations. American Sociological Review, 1976. 41(2): p. 322-338.

[40] Waller, M.J., The Timing of Adaptive Group Responses to Non-Routine Events. Academy of Management Journal, 1999. 42(2): p. 127-137.

[41] Wittenbaum, G.M. and G. Stasser, Management of Information in Small Groups, in What's Social about Social Cognition? J.L. Nye and A.M. Brower, Editors. 1996, Sage Publications: Thousand Oaks, California. p. 3-27.