HARVARD | BUSINESS | SCHOOL

Building the Agile Enterprise: IT Architecture, Modularity and the Cost of IT Change

Alan MacCormack
David E. Dreyfus
Robert Lagerstrom
Carliss Y. Baldwin

Working Paper 15-060

Building the Agile Enterprise: IT Architecture, Modularity and the Cost of IT Change

Alan MacCormack
Harvard Business School

Robert Lagerstrom
KTH Sweden

David E. Dreyfus
Arete Analytics

Carliss Y. Baldwin
Harvard Business School

Working Paper 15-060

Building the Agile Enterprise: IT Architecture, Modularity and the Cost of IT Change

Alan MacCormack[1]

Robert Lagerstrom[2]

David E. Dreyfus[3]

Carliss Y. Baldwin[1]

[1] Harvard Business School

[2] KTH Sweden

[3] Arete Analytics

Keywords: IT Architecture, Modularity, Information Systems, Software, Agility, and Networks

Abstract

Recent contributions to information systems theory suggest the primary role of a firm's IT architecture is to facilitate agility, ensuring the continued alignment of a firm's capabilities with a changing business environment. Despite advances in our understanding of the role of IT however, we lack robust empirical data on the precise *mechanisms* by which IT architecture impacts agility. As a result, it is difficult to identify specific managerial actions that can improve a firm's architecture, and thereby lower the cost of responding to changing future demands.

We address this gap in the literature by exploring the relationship between IT architecture and IT agility at the level of the individual components in the architecture. In particular, we draw upon modular systems theory to develop and test a series of propositions about the impact of component coupling on the cost of IT change. Our methods make use of "Design Structure Matrices" (DSMs), a novel network analysis technique that addresses the limitations of prior work in this area. DSMs allow us to i) identify discrete *layers* in the IT architecture associated with different types of components, and; ii) capture data on the *dependencies* between components; hence to assess their relative levels of coupling and position within the architecture.

We test our propositions using empirical data from a large pharmaceutical firm, encompassing 407 architectural components and 1,157 dependencies between them. We show that different components occupy different positions in the IT architecture and possess different levels of coupling. We find measures of coupling predict IT agility – defined as the cost of making changes to software applications. The measure of coupling that best predicts agility is one that captures all *direct and indirect* connections between components (i.e., it captures the potential for changes to propagate via all possible paths between components). Our results reveal that different business groups within the firm face different costs of adaptation, deepening our understanding of the roots of organizational agility.

1. Introduction

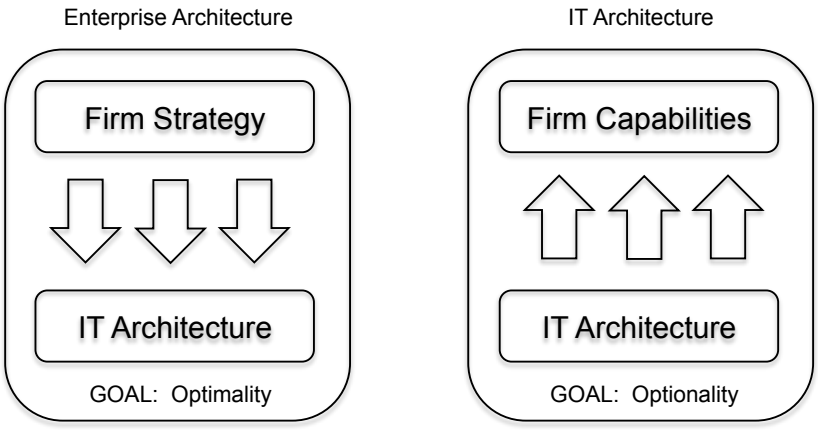
As information becomes more pervasive in the economy, information systems within firms have become more complex. Initially, these systems were designed to automate back-office functions and provide data to support managerial decision-making. Their role was expanded to coordinate the flow of production in factories and supply chains. The invention of the personal computer led to the creation of client-server systems, which enhanced the productivity of office workers and middle managers. Finally, the arrival of the Internet brought a need to support web-based communication, e-commerce, and online communities. Today, even a moderately sized business maintains information systems comprising hundreds of applications and databases, running on geographically distributed hardware platforms, and serving multiple clients. These systems must be secure, reliable, flexible, and capable of evolving to meet new challenges.

As the importance of information systems has grown, management scholars have developed frameworks to ensure these systems support a firm's business strategy. Enterprise architecture (EA) is the name given to a set of conceptual frameworks, processes and tools used to manage an enterprise's information system infrastructure. The aim of EA is to achieve alignment between the strategy of the firm and the configuration of its IT assets (Weill, 2007). In essence, *EA is the expression of a firm's business strategy, in terms of its choice of IT assets, the configuration of these assets and the relationships that exist between these assets and the formal organization*. The process of developing EA is inherently "top-down" in nature; for a given business strategy, the goal is to design an optimal enterprise architecture. For example, TOGAF®, the most-cited EA framework in this field, was developed by a consortium of firms to provide a standardized approach to the design and management of information systems within

and across organizations (TOGAF, 2009). It provides a method for visualizing, understanding and planning for the needs of organizational stakeholders in a seamless way.

While EA research emphasizes the need for alignment, a related but distinct stream of research adopts a sharper focus on the "IT Artifact" itself, and in particular, the way in which IT architecture facilitates the development of *new* capabilities (Orlikowski and Iacono, 2001; Tiwana and Konsynski, 2010). This work is motivated by the complex and dynamic nature of today's competitive landscapes, and the increasingly pervasive nature of digital platforms (Tanriverdi et al, 2010; Tiwana et al, 2010; Tilson et al, 2010). Given such trends, the view of EA as a static optimization problem is increasingly obsolete. Instead, modern theory emphasizes the need for an architecture that facilitates agility, through the use of layered, modular technologies (Yoo et al, 2010; Tiwana and Konsynski, 2010). Firms with modular architectures can quickly reconfigure resources to respond to new challenges, ensuring a continuous alignment of IT assets with changing business needs. In contrast to EA, IT Architecture research implies a more "bottom-up" approach to design; the aim is to create an architecture that can sense and respond to new challenges, the nature of which cannot be predicted in advance (Sambumurthy et al, 2003; Hanseth and Lyytinen, 2010). The goal is *optionality*, not optimality (see **Figure 1**).

Figure 1: Enterprise Architecture and IT Architecture Research Frameworks



Robust empirical work exploring the impact of information systems architecture on firm performance are scarce and produce mixed findings (Tamm et al, 2011). Studies of EA are often constrained by the difficulty of comparing architectures for firms that possess very different strategies and/or organizational structures. As a result, empirical work in this area has focused more on the quality of the EA planning process, the governance of this process or the documents output from it (e.g., Schmidt and Buxmann, 2011). By contrast, IT Architecture research places greater emphasis on measuring characteristics of the IT artifact itself, instead of the processes through which it is created. For example, several studies use firm-level measures of IT infrastructure flexibility to assess the impact of this construct on measures of firm performance (e.g., Duncan, 1995; Kim et al, 2011). Of particular note to this study, Tiwana and Konsynski (2010) show that firms with more modular IT architectures (i.e., a greater use of loosely-coupled, and standardized components) tend to have higher perceived levels of IT agility.

While these studies have increased our understanding of the role of IT in the modern firm, we still lack robust empirical data on the precise *mechanisms* by which IT architecture impacts IT agility, and as a consequence, firm performance. A major challenge relates to the fact that in many empirical studies, the firm is conceived of as a monolithic entity, hence measures of IT function and architecture are reduced to simple single item scales. But firms are not monolithic; their constituent parts do not move in lockstep. They comprise differentiated organizational units, with different objectives, requiring different levels of flexibility (Lawrence and Lorsch, 1967). Similarly, the components of a firm's IT architecture are heterogeneous (Orlikowski and Iacono, 2001). They are diverse in type, play different roles in the system, are connected in different ways, and vary significantly in their costs of adaptation (Sambumurthy and Zmud, 2000; Yoo et al, 2010). Research to better understand the roots of agility must

recognize this heterogeneity, and adopt methods to assess its impact on performance. *This suggests a need to study the impact of IT architecture at the level of the individual components in the architecture*. With such an approach, we can discriminate between parts of a firm's IT infrastructure that are amenable to change, versus parts that are rigid and inflexible. We can show how these relationships affect specific organizational groups within a firm, bridging the gap that exists between prior work on EA and IT Architecture. And we can identify specific managerial actions to improve a firm's IT architecture, thereby lowering the cost of responding to changing future demands. These goals serve as the motivation for our study.

In this paper, we explore the mechanisms through which IT architecture impacts IT agility, and as a consequence, firm performance. Our work addresses a well-defined gap in the literature, in particular, the lack of empirical data exploring the link between IT architecture and IT agility at the level of the individual components (i.e., the "microstructure" of IT architecture). Specifically, we use modular systems theory to develop propositions about the impact of different types/levels of component coupling on IT agility – defined as the cost to make changes to software applications. We use Design Structure Matrices (DSMs), a novel network-based methodology, to test these propositions (Steward, 1981; MacCormack et al, 2006). DSMs can reveal the different *layers* in an IT architecture associated with different types of component (e.g., applications, data, infrastructure) as well as the *relationships* between these layers and components (Yoo et al, 2010). Our data is drawn from a division of a large pharmaceutical firm. In contrast to prior work, which uses holistic measures of a firm's entire IT architecture, we capture fine-grained data on the actual dependencies between components in the architecture. Our dataset encompasses 407 components and 1, 157 dependencies between them.

We find significant differences in the cost to change different software components in the IT architecture in our focal firm. We show these differences in cost are associated with the level of coupling for components. Specifically, components that are tightly coupled to many other components in the IT architecture have significantly higher change costs than components that have relatively few connections to others. The measure of coupling that best predicts the cost of change is one that captures all *direct and indirect* connections between components. In sum, it is critical to account for all possible paths by which changes may propagate, when assessing the potential costs of change for a component. Other measures of coupling (e.g., direct coupling, closeness centrality) add no further power to our models. Our results suggest that different groups within the firm, by virtue of their use of different applications, data and infrastructure components, face different costs of change, and hence possess different levels of agility. Our work contributes to the literature by deepening our understanding of the roots of organizational agility, while helping to bridge the gap between EA and IT Architecture research.

The paper is organized as follows. In Section 2, we review the existing literature that motivates our work. In section 3, we develop theory and derive our research hypotheses. In Section 4, we describe our research methods, which make use of a novel network-based methodology called Design Structure Matrices. In Section 5, we introduce our empirical setting and describe how we develop measures of IT Architecture at our focal firm. In Section 6, we describe our empirical data, and provide the results of our statistical tests. Finally, in Section 7, we discuss the implications of our results for research and for managerial practice.

2. Literature Review

This section reviews the related but distinct streams of work on enterprise architecture and IT architecture, with the aim of explicitly defining the gap in the literature that we seek to

address. We then use Modular Systems Theory to develop a number of theoretical propositions about the relationship between measures of IT architecture and IT agility, focusing on the mechanisms that affect the individual components in the architecture.

2.1 Enterprise Architecture Research

Enterprise Architecture is commonly defined as a tool for achieving alignment between a firm's business strategy and its IT infrastructure. For example, MIT's Center for Information Systems Research defines EA as "the organizing logic for business processes and IT infrastructure reflecting the integration and standardization requirements of the company's operating model" (Weill, 2007). EA analysis is not limited to IT systems, but encompasses the relationship with and support of business entities. Hence EA research tends to focus on the "strategic" implications of EA efforts (Aier, 2014, Boh & Yellin, 2007; Ross et al., 2006).

Prior EA work emphasizes conceptual models, frameworks and processes that seek to achieve alignment between business needs and IT infrastructure (e.g., Aier and Winter, 2009). As the name suggests, architectural descriptions are central to these efforts. These descriptions include entities that cover a broad range of phenomena, such as organizational units, business processes, applications, data and IT infrastructure (Lankhorst, 2009; Winter & Fischer, 2007; Jonkers et al., 2006). A number of frameworks have been proposed detailing the entities and the relationships between them that should form part of EA modeling efforts (e.g., DODAF, 2007; MODAF 2008; TOGAF, 2009). However, diversity exists in terms of the unit of analysis and terminology adopted within each (Zachman, 1987; TOGAF, 2009; Lagerström et al., 2009).

While there exists a wide range of EA frameworks, a theme they have in common is layering. Layers comprise the basic structure of an EA model, representing distinct conceptual domains which group together related entities (Simon et al, 2013). Given the goal of achieving

alignment between business strategy and IT resources, EA models include both business and IT layers. Typical business layers are business strategy, business processes and business groups. Typical IT layers are applications, data and infrastructure (Simon et al, 2013; Matthes et al, 2008; Winter and Fischer, 2006). While studies differ in the way that they classify different layers, they share common underlying assumptions. First, layering reflects a division of functions provided by a system into units, such that these units can be designed, developed, used and updated independently. Second, layering establishes a design "hierarchy" (Clark, 1985) such that each layer interacts only with layers immediately above or below, reducing complexity. Finally, the direction of dependencies between layers is such that higher layers "use" lower layers, but not the reverse, limiting the potential for changes to propagate (Gao and Iyer, 2006).

Several EA frameworks propose using *matrices* to display the relationships among various components of an information system, in an attempt to make EA more operational. For example, TOGAF (2009) recommends preparing nine separate matrices in the development of a firm's architecture, which are used to track the linkages and dependencies between various parts of the system. Unfortunately, it is not clear how these matrices should be combined to generate managerial insights (e.g., how to improve a system). Furthermore, the information used to construct them reflects an idealized view of how a system *should* function, and not how it actually functions. In essence, these matrices are primarily descriptive in nature, and do not yield quantitative measures of architecture that can be used to improve performance.

2.1.1 Empirical Studies in EA Research

Surprisingly, there has been little academic work to explore the *performance* benefits of EA, using empirical data on the outcomes achieved by firms. Tamm et al. (2011) found that of the top 50 articles on EA (ranked by citation count) only 5 provided data that sought to establish

a link between EA efforts and improved performance outcomes. Studies that do make claims about the performance benefits of EA typically cite a range of "enablers" that EA affects, which are likely to mediate important firm outcomes. Recurring themes include better organizational alignment, improved information quality and availability, optimized resource allocation, and increased complementarities between firm resources (Tamm et al., 2011).

Many authors note it is difficult to directly assess the "quality" of a firm's architecture. Hence empirical studies linking EA to performance have tended to focus on assessing the quality of the EA planning process, or the quality of the documents output from this process (Kluge et al., 2006; Aier et al., 2011; Lagerström et al., 2011; Schmidt and Buxmann, 2011). Unfortunately, this means that it is difficult to differentiate between firms that follow similar processes, but that implement different system designs. One exception is the work of Schmidt and Buxmann (2011), who show that higher quality enterprise architecture processes are associated with more "flexible" IT infrastructures (i.e., those with a greater level of connectivity, compatibility and modularity). Significantly, in this work, IT infrastructure is conceived of as a measure of *output*, whereas in studies of IT Architecture, it is typically viewed as an *input*. This further illustrates the gap that has emerged between EA and IT Architecture research.

From a practitioner perspective, the evidence that adopting EA frameworks like TOGAF (2009) improves outcomes is mixed (Buckl et al., 2009; Seppänen et al., 2009; Dietz & Hoogervorst, 2011). A 2008 study of EA adoption by firms reported that over two-thirds of EA projects fail to meet managers' expectations (Roeleven, 2010). EA frameworks have however, been shown to be a useful decision-support tool when focused on the needs of specific decision-makers (Johnson and Ekstedt, 2007). Narrow models help stakeholders to understand how EA efforts impact *specific* performance dimensions (e.g., security). They also generate insight into

current and possible future EA states (Sommestad et al., 2013; Ullberg et al., 2011; Franke et al., 2014; Lagerström et al., 2010; Närman et al., 2011).

In sum, operationalizing EA in a robust and reliable way, that allows firms to quantify, analyze and improve their information systems infrastructure, has proven an elusive goal. A wide range of EA frameworks exist, each employing differing units of analysis and terminology. Furthermore, most frameworks are conceptual in nature, generating few quantitative measures with which to analyze performance. Finally, these frameworks focus on "ideal" (i.e., planned) representations of a firm's architecture, as opposed to data on the actual architecture "in-use."

2.2 IT Architecture Research

In contrast to EA research, which focuses on the alignment of a firm's strategy to its IT assets, IT Architecture research adopts a sharper focus on the "IT Artifact" itself, and in particular, features of the architecture that facilitate the development of new firm capabilities (Tiwana and Konsynski, 2010). This line of work is motivated by critiques of EA research that note the overwhelming emphasis on processes and governance structures through which IT is managed, as opposed to features and characteristics of the technology itself (Orlikowski and Iacono, 2001; Tilson et al, 2010). In addition, the rapid rise of the Internet, World Wide Web and use of digital technologies has brought a need to revisit prior conceptions of the role of information systems within firms, to reflect the new dynamics of a digital age with its rapidly shifting competitive landscapes (Hansen and Lyytinen, 2010). As Sambamurthy and Zmud (2000) note, the "aim is to shift thinking… towards more complex structures that are reflective of contemporary practice. These structures are designed around important IT capabilities and network architectures."

Early work in the field of IT Architecture focused on understanding desirable features of IT technologies, and in particular, the antecedents of more flexible IT infrastructure. The works of Duncan (1995) and Byrd and Turner (2000), for example, were notable in establishing distinct constructs thought to underpin a more flexible IT infrastructure. They emphasized the need for IT systems with greater levels of compatibility and connectivity, through the use of system-wide standards and interfaces, to promote the sharing and reuse of applications, data and infrastructure components. They also emphasized the need for greater levels of modularity (i.e., the loose coupling of applications, data and infrastructure) so that components could be deployed, modified and updated with minimal impact on other system elements.

Subsequent work sought to deepen our understanding of how these concepts could be operationalized in firms competing in a dynamic, connected digital world. Sambamurthy and Zmud (2000) suggest the new organizing logic for IT architecture is the platform, which encompasses a "flexible combination of resources, routines and structures" to meet the needs of current and future IT-enabled functionalities. Platforms facilitate agility along multiple dimensions of performance, creating "digital options" and enhancing "entrepreneurial alertness" (Samburmathy at al, 2003). Adomavicius et al. (2008) introduce the concept of an IT "ecosystem," highlighting the different roles played by products, applications, component technologies and infrastructure technologies, including those *external* to the firm. Finally, Yoo et al. (2010) describe how pervasive digitization has given rise to a new "layered-modular" architecture, comprising devices, network technologies, services and content.

Firms with layered, modular IT architectures can quickly reconfigure resources to respond to new challenges, creating a continuous stream of new capabilities (Tanriverdi et al, 2010; Tiwana et al, 2010). In contrast to EA research, IT Architecture research therefore implies

a more "bottom-up" approach to design; the aim is to create an architecture that can sense and respond to new challenges, the nature of which cannot be predicted in advance (Sambumurthy et al, 2003). Yet layered, modular IT architectures are not easy to build, and not the norm in firms with complex information systems infrastructures (Baldwin et al 2014). Firms more often grapple with a mixture of systems of different vintages, designed using different conceptual frameworks, to meet different demands (Ross 2003). To move from the status quo to a layered, modular architecture, firms must adopt new models for the role of IT and new structures by which these technologies will work together (Ross, 2003; Ross and Westerman, 2004).

2.2.1 Empirical Studies in IT Architecture Research

Empirical studies of IT Architecture have also been scarce and limited in scope. Most work to date has been either case-based, or has used survey measures of IT infrastructure adapted from Duncan's work (1995) to demonstrate correlation with measures of performance (Salmela, 2015). For example, Kim et al (2011) show measures of a firm's IT infrastructure flexibility, as captured by the constructs of compatibility, connectivity and modularity, are correlated with a firm's ability to change existing business processes. Conversely, Liu et al (2013) find IT infrastructure flexibility is *not* associated with agility, but contributes to performance only via its association with increased levels of absorptive capacity. Schmidt and Buxmann (2011) measure IT infrastructure as an *output*, and find it is associated with higher quality EA planning processes. However, it is possible the implied causal relationship here operates in reverse. That is, IT infrastructure flexibility may in fact, promote more effective EA planning processes.

The most insightful empirical study in this stream of work is from Tiwana and Konsynski (2010), who characterize IT Architecture along two dimensions – loose coupling, and standardization – showing that these measures are associated with a more agile, adaptive,

flexible and responsive IT function. While this work has increased our knowledge of important features of IT Architecture in the modern firm however, we still lack robust empirical data on the precise *mechanisms* by which IT architecture impacts IT agility. The challenge relates to the fact that in this and other studies, the firm is conceived of as a monolithic entity; hence measures of IT function and architecture are assumed to be homogenous across the firm. But firms are not monolithic, they comprise differentiated organizational units, with different objectives and require different levels of flexibility (Lawrence and Lorsch, 1967). Similarly, the components of a firm's IT architecture are diverse in type, play different roles in the system, are connected in different ways, and vary in their costs of adaptation (Orlikowski and Iacono, 2001; Sambumurthy and Zmud, 2000; Yoo et al, 2010). Research to better understand the roots of agility must recognize this heterogeneity, and adopt methods to assess its impact on performance. This suggests a need to study the impact of IT architecture at the level of the individual components in the architecture.

3. Theory Development

3.1 Modular Systems Theory

A large number of studies contribute to our understanding of the design of complex systems (Holland, 1992; Kauffman, 1993; Rivkin, 2000; Rivkin and Siggelkow, 2007). Many of these studies are situated in the field of technology management, exploring factors that influence the design of physical or information-based systems (Braha et al, 2006). Complex technological systems comprise a large number of components with interactions between them. The scheme by which a system's functions are allocated to components is called its architecture (Ulrich, 1995; Whitney et al, 2004). Understanding how architectures are chosen, how they perform and how they can be adapted are critical topics for the study of complex technological systems.

Modularity is a concept that helps us to characterize different architectures (Sanchez and Mahoney, 1996; Schilling, 2000). It refers to the way that a system is "decomposed" into different parts or modules (Simon, 1962). Although there are different definitions of modularity, authors agree on its fundamental features: the interdependence of decisions *within* modules, and the independence of decisions *between* modules (Mead and Conway, 1980; Baldwin and Clark, 2000). The latter concept is commonly referred to as "loose-coupling." Modular systems are loosely coupled in that changes to one module have little or no impact on others (MacCormack et al, 2012). Just as there are degrees of coupling, there are also degrees of modularity.

Modular systems theory is the name given to a growing body of theory that explores the design of systems and the costs and benefits that arise from designs that are modular (Sanchez and Mahoney, 1996; Schilling, 2000; Baldwin and Clark, 2000). This theory has been broadly applied, to the study of biological systems, technical systems and organizational systems (Kauffman, 1993; Weick, 2001; Langlois 2002; Berente and Yoo, 2012). A central tenet of the theory is that modular systems (i.e., systems with components that are loosely-coupled) can be adapted with lower costs and with greater speed, given changes are isolated within modules. Conversely, in a system with tight coupling, change is more costly and takes longer, given the potential for changes to propagate and impact other system elements. We therefore assert:

Proposition: IT Architecture Modularity is associated with IT Agility; wherein IT Agility is defined as the cost of making changes to components in the architecture.

Several studies offer evidence to support this proposition, using holistic measures of a firm's IT architecture, and measures of the extent to which the IT function is perceived as agile, adaptive and flexible (e.g., Tiwana and Konsynski, 2010). However, studies that explore this dynamic using the firm as unit of analysis are not well suited to understanding the precise

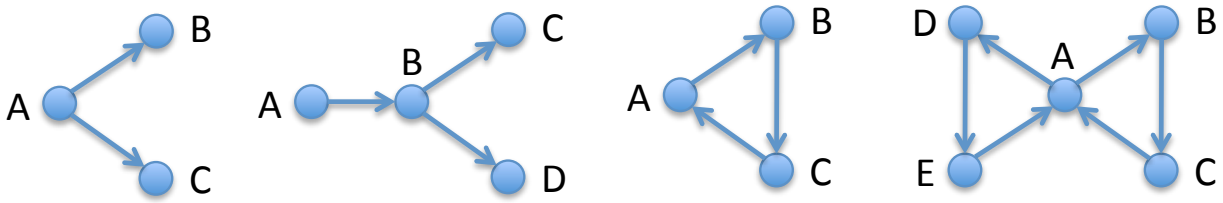
mechanisms through which architecture impacts agility. We need a more nuanced view of architecture, reflecting the complexity of modern day IT systems, which comprise hundreds of components, organized into different layers, with many relationships between them (Sambumurthy and Zmud, 2000; Yoo etal, 2010). Recent empirical studies confirm that heterogeneous levels of coupling are the norm in such systems (Lagerstrom et al, 2014; Baldwin et al, 2014). Some components belong to "Core" subsystems, critical for system functioning; whereas others belong to "Peripheral" subsystems, playing a less significant role (Tushman and Rosenkopf, 1992; Tushmann and Murmann, 1998). These studies and others suggest a need to focus on the *microstructure* of IT architecture; to identify different layers in the architecture associated with different types of component, to understand the patterns and levels of coupling between components, and to explore the impact of these measures on the cost of IT change.

3.2 Component Coupling and the Cost of Component Change

The computer scientist David Parnas argued that, in technical systems, the most important form of linkage between components is a directed relationship he calls "depends on" or "uses" (Parnas, 1972). If B uses A, then A fulfills a need for B. If the design of A changes, then B's need may go unfulfilled. B's behavior may then need to change to accommodate the change in A. Hence change propagates in the opposite direction to use. Parnas stressed that use is not symmetric. If B uses A, but A does not use B, then B might change with no impact on A. Our first step in theory building is to relate Parnas' concept of dependency to component coupling. To capture Parnas' insight that a dependency may be asymmetric, we define coupling as the property of being used (i.e., depended upon) by another component. Component A is coupled with component B if B uses A. Then, according to Parnas, if A changes, B may have to change as well. Hence we will state our hypotheses in terms of component coupling.

Modular systems theory predicts the more coupled a component is, the more difficult and expensive it will be to change (Simon, 1962; Sanchez and Mahoney, 1996). However, the components of a system can be coupled in different ways (Baldwin and Clark, 2000). They can be coupled directly or indirectly; and they can be coupled hierarchically or cyclically. Furthermore, components that are hierarchically coupled may be located at the top or the bottom of the design hierarchy (Clark, 1985); and components that are cyclically coupled may be members of a large or small cyclic group of components (Sosa et al, 2013). For the purposes of theory development, we therefore consider a single component (labeled "A") within a firm's IT architecture. A could represent any type of component in the IT architecture (e.g., an application, a database, or a piece of technical infrastructure). The question we must answer is how does the presence of coupling (or the lack thereof) affect the cost of change for this component? To answer this question, we examine four different types of coupling in a technical system, and consider how each type is likely to affect the cost of change (see **Figure 2**).[1]

Figure 2: Four Types of Coupling between Components



2.1: Direct 2.2: Indirect 2.3: Cyclical 2.4: Central

In Figure 2.1, component A is directly coupled with components B and C. Modular systems theory predicts that components with higher levels of direct coupling are more costly to change, given the need to consider the potential impact of changing the coupled component on

[1] We note the concept of studying different types of coupling is also a feature of studies of the social context within which technology is situated (e.g., Berente and Yoo, 2010). Here however, we focus only on *technical* coupling.

the dependent components (Simon, 1962). We therefore predict that component A would be more costly to change than a similar component with no coupling. Support for such a relationship is found in empirical studies of software systems, in which components are defined to be source files or classes, and dependencies denote relationships between them (Chidamber and Kemerer, 1994). We believe this relationship will also hold true for IT architecture:

Hypothesis 1: Within an IT architecture, the level of **direct** *coupling for a component will be positively associated with the cost of component change.*

Figure 2.2 depicts a more complex set of relationships between system components. Component A is directly coupled to B but is *indirectly* coupled to components C and D. This raises the prospect that changes may propagate between components that are not directly connected, via "chains" of direct couplings. Empirical work has shown that changes to one component in an IT system often create unexpected disruptions in distant parts of the system (Vakkuri, 2013). Furthermore, measures of indirect coupling have been shown to predict both the number of defects and the ease (or difficulty) with which a software system can be adapted (MacCormack, 2010; MacCormack and Sturtevant, 2016). In essence, when dealing with complex systems, changes often propagate in unexpected ways, significantly increasing the costs of adaptation. Hence we believe that the level of indirect coupling for a component will also be a significant predictor of the cost of component change:

Hypothesis 2: Within an IT architecture, the level of **indirect** *coupling for a component will be positively associated with the cost of component change.*

A related but distinct problem that complicates the design of complex systems is the presence of "cyclic groups" – groups of components that are mutually interdependent (Whitney, et al, 2004; Sosa et al, 2013). Figure 2.3 illustrates this challenge. In this system, A is coupled

with B, B is coupled with C, and C is coupled with A. Hence all three components are connected, either directly or indirectly, to all others. In such systems, there is no "hierarchy" or natural ordering of the components, such that one can be designed (or adapted) before the others. Rather, components in cyclic groups must often be designed (or changed) concurrently, to ensure that they continue to work together effectively. When cyclic groups are large, this presents a significant challenge, increasing the cost of change for components within the cyclic group (Baldwin et al, 2014). Hence we propose:

Hypothesis 3: Within an IT architecture, the level of **cyclic** *coupling for a component will be positively associated with the cost of component change.*

Components that are members of the same cyclic group possess the same levels of indirect coupling, given each component shares the dependencies of all others in the group. In this situation however, some components may still be more "central" than others, in terms of their position in the system, and their relative importance for the potential propagation of changes. Figure 2.4 illustrates this situation. In this system, the pattern of couplings is such that all components are mutually interdependent (there are indirect paths between all components). However, component A occupies a distinct position in that it is "closer" to all others, on average, and on more pathways by which changes might propagate between any two components. As a consequence, it is likely to be more costly to adapt than others:

Hypothesis 4: Within a cyclic group of components, the **centrality** *of a component will be positively associated with the cost of component change.*

The different types of coupling described above represent related, but conceptually distinct, mechanisms through which IT architecture might impact IT agility. We do not have an ex-ante view as to which will be stronger. Indeed, empirical evidence is found for each of them,

albeit in different contexts (i.e., for different types of system, and for different units of analysis). For this reason, we believe that the four mechanisms are likely to be complementary in their effect. Hence our final hypothesis can be stated as follows:

Hypothesis 5: Measures of component coupling, taken together, explain more of the variation in the cost of component change than any single measure.

We note that the types of coupling we describe above are likely to be correlated. Specifically, components with high levels of direct coupling are more likely to have high levels of indirect coupling. All else equal, components with high levels of indirect coupling are more likely to be members of cyclic groups. Finally, components that are members of cyclic groups with high levels of direct coupling are more likely to have higher measures of centrality. It will be important to be sensitive to these issues when conducting empirical tests.

4. Research Methods

Studies that attempt to measure modularity often employ network representations of a design (Barabasi, 2009). The objective is to identify the level of coupling that exists between different elements (Simon, 1962; Alexander, 1964). To develop measures of IT Architecture, we use Design Structure Matrices, an increasingly popular network-based method for analyzing technical systems (Steward, 1981; Eppinger et al., 1994; MacCormack et al., 2006; 2012; Sosa et al., 2007). A DSM highlights the structure of a complex system using a square matrix, in which rows and columns represent system elements, and dependencies between elements are captured in the off-diagonal cells. Baldwin et al. (2014) show that DSMs can be used to understand the "hidden structure" of complex systems, by capturing the level of coupling between components, and classifying components into categories based upon the direction and level of coupling.

Metrics that capture the level of coupling for components have been calculated from a DSM and used to understand system performance and evolution. For example, MacCormack et al. (2006) and LaMantia et al. (2008) use DSMs and the metric "propagation cost" to compare software system architectures, and to track the evolution of systems over time. MacCormack et al. (2012) show that the architecture of software systems tends to "mirror" that of the organizations in which they are developed. Sturtevant (2013) shows that software components with high levels of coupling experience more defects, cost more to adapt and are associated with higher staff turnover. Finally, Ozkaya (2012) shows that metrics derived from DSMs can be used to assess the value of "re-factoring" designs with poor architectural properties.

In a recent case study, Lagerström et al. (2013) use DSMs to study a firm's enterprise architecture, in which a large number of interdependent software applications have relationships with other types of components, such as business groups, schemas, servers, databases and other technology infrastructure elements. In this paper, we formalize and extend this approach, by showing how measures derived from an IT architecture DSM predict IT agility.

4.1 Design Structure Matrices and *Direct* Coupling

A DSM is a way of representing a network. Rows and columns of the matrix denote nodes in the network; off-diagonal entries indicate direct dependencies between the nodes. In the analysis of IT architecture, the rows, columns, and main diagonal elements of a DSM correspond to the components in the system—in our case, business groups and technical resources associated with the architecture (e.g., applications, databases, etc.). The main diagonal elements are set to one (indicating that components "depend on" themselves). Linkages between components are represented by off-diagonal entries (also set to one) and indicate that a direct

coupling relationship exists between two components, as defined previously. Specifically, if component A is coupled with component B, then any change made to A may affect B.

As a matter of convention, usage (i.e., dependency) proceeds from row to column in our DSMs, hence coupling proceeds from column to row. That is, reading down the column of a component in a DSM reveals all of the other components that directly depend upon it. The focal component is directly coupled with all of the dependent components. Hence the levels of direct coupling (and direct dependency) for a component can be read directly from the DSM. Specifically, for the *i*th component in a system, the level of direct coupling (DC) can be found by summing the entries in the *i*th column. The level of direct dependency (DD) can be found by summing the entries in the *i*th row.[2] In general, the measures of DC and DD will be different unless all of the dependencies for a focal component are symmetric. If usage *is* symmetric (i.e., A uses B *and* B uses A), then off-diagonal entries in the DSM will be symmetric around the main diagonal, and measures of direct coupling and direct dependency will be the same.

4.2 Design Structure Matrices and *Indirect* Coupling

Using a DSM, we can also analyze the *indirect* dependencies between components, which reflect the potential for changes to propagate in a system via a "chain" of dependencies. Suppose, for example, component A is coupled with component B and component B is coupled with component C. Then a change to A may affect C *through* its impact on B. To identify the indirect couplings and dependencies in a system, we apply the procedure of transitive closure to the direct DSM and set all positive entries equal to one. The result is called the "visibility" matrix (MacCormack et al., 2006; Baldwin et al., 2014). The visibility matrix captures all *direct and indirect* dependencies between components. In a similar fashion to the direct dependency

[2] In prior work, we used the symbol DFI (Direct Fan-In) to denote the level of direct coupling for a component, and DFO (Direct Fan-Out) to represent the level of direct dependency for a component.

DSM, the level of direct and indirect coupling for a component (AC) is captured in the column sums of the visibility matrix. The level of direct and indirect dependency (AD) for each component is captured in the row sums of the visibility matrix.[3]

The density of the visibility matrix, called *Propagation Cost*, measures the level of indirect coupling for the system as a whole. Intuitively, the greater the density of the visibility matrix, the more ways there are for changes to propagate in the system, and thus the higher the potential cost to change it. Large differences in propagation cost have been observed across systems of similar size and function (MacCormack et al., 2006). These differences are predicted, in part, by the structure of the developing organization (MacCormack et al, 2012). Empirical evidence suggests that refactoring efforts aimed at making a design more modular can lower propagation cost substantially (MacCormack et al., 2006; Akaikine, 2009). In combination, these findings suggest that for many systems, architecture is not dictated solely by system function, but is affected by both organizational constraints and the objectives of the system designers.

4.3 Design Structure Matrices and *Cyclic* Coupling

Measures of visibility can be used to identify "cyclic groups" of components, each of which is directly or indirectly connected to all others in the group. Mathematically, members of the same cyclic group all have the same AC and AD measures, as defined above, given they are all connected directly or indirectly to each other. Thus we can identify cyclic groups in a system by sorting on these measures after performing a transitive closure on the direct dependency DSM and setting all positive entries equal to one (Baldwin et al., 2014). Large cyclic groups are problematic for system designers, given that changes to a component may propagate via a chain of dependencies to many other components. In such a structure, the presence of cyclicality

[3] In prior work, we used the symbol VFI (Visibility Fan-In) to represent all direct and indirect couplings for a component and the symbol VFO (Visibility Fan-Out) to represent all direct and indirect dependencies.

means that there is no guarantee that the design process (or a design change) will converge on a globally acceptable solution that satisfies all components.

The components in a system can be classified into groups according to the levels of coupling and dependency they exhibit, as captured by AC and AD. Specifically, Baldwin et al. (2014) use DSMs to analyze the structure of 1286 releases from 17 distinct software applications. They find the majority of systems exhibit a "core-periphery" structure, characterized by a single dominant cyclic group of components (the "Core") that is large relative to the system as a whole as well as to other cyclic groups. They show the components in such systems can be divided into four groups – Core, Peripheral, Shared and Control – that share similar properties. Core components are members of the largest cyclic group, and have high levels of direct and indirect coupling (AC) and dependency (AD). Shared components have only high levels of direct and indirect coupling (AC), i.e., they are "used", directly or indirectly, by many other components. Control components have only high levels of direct and indirect dependency (AD), i.e., they "use", directly or indirectly, many other components. Peripheral components have low levels of both direct and indirect coupling (AC) and dependency (AD). In such systems, dependencies flow from Control components, through Core components, to Shared components. This is the main "flow of control" in a system. Peripheral components lie outside the main flow of control, thus are loosely coupled to the rest of the system.

4.4 Design Structure Matrices and *Centrality*

The asymmetry of dependency relationships between components in many technical systems makes it difficult to use typical measures of network centrality, which are based upon the notion of the "path length" between, or the "closeness" of, two components. For example, suppose A is coupled with B, but not vice versa. The path length from A to B is 1; however, the

path length from B to A is undefined or infinite. Thus, the most commonly applied measures from social network theory must be adapted for use in networks with directional dependencies.

In contrast, a DSM allows us to determine the flow of control in a system and hence the direction in which changes are likely to propagate. In particular, we can distinguish between systems that are hierarchical in nature (i.e., there exists a strict ordering of coupled components) versus those that are cyclical in nature (i.e., the coupled components are mutually interdependent). Hierarchy and cyclicality are critical constructs for understanding how changes might propagate in complex systems. DSMs reveal these characteristics.

Given the preceding discussion, in this study we examine measures of centrality *only* for those components that are part of the same cyclic group, and for which the dependency structure is symmetrical. As a consequence, our analysis of centrality as a coupling mechanism is limited only to specific parts of the firm's IT Architecture where these assumptions are met.

4.5 Design Structure Matrices and *Layering*

In a layered architecture, convention determines the ordering of layers from top to bottom. One can place the "users" in higher layers and the objects of use in lower layers or vice versa. Most EA layer diagrams, for example, display users at the top. In constructing DSMs, however, we depart from this practice, and place users below the components that they use. Our reason for doing this is based upon the concept of "design sequence" as described below.

When used as a planning tool in a design process, a DSM indicates a possible sequence of design tasks, i.e., which components should be designed before which others (Steward, 1981; Eppinger et al, 1994). In general, it is intuitive and desirable to place the first design tasks at the top of a DSM, with later tasks below. In sum, the first components to be designed should be those that other components depend upon. For example, suppose that component B uses

component A. Then A's design should be complete before B's design is begun. Reversing this ordering runs the risk that B will have to be redesigned to comply with unforeseen changes in A.

Reflecting this discussion, we place the "most used" layers at the top of the DSM and the "users" of these layers at the bottom of the DSM. This convention ensures that design rules and requirements, which affect many subsequent design choices, will always appear at the top of the DSM (i.e., they should be completed early in a design process). Hence components at the top of the DSM will have high levels of direct and indirect coupling (AC) and components at the bottom of the DSM will have high levels of direct and indirect dependency (AD). Critically, in cases where the system layers are not known *a priori*, the visibility matrix can be sorted using AC and AD to reveal the hierarchical relationships among different layers and components.

5. Measuring IT Architecture

Our study site is the research division of a US biopharmaceutical company with over 1,000 employees ("BioPharma") founded in the 1990's.[4] At this company, "IT service owners" are responsible for divisional information systems, and provide project management, systems analysis, and limited programming services to the organization. Data on IT architecture components and the dependencies between them were collected through a combination of manual and automated methods. To identify components, we examined strategy documents and interviewed the IT director and IT service owners. To identify dependencies, IT service owners were asked to input architectural information into a repository. This information was supplemented by the use of automated open-source and custom tools that monitored server and network traffic. The resulting data on processes and communication links was aggregated to the

[4] The data for this study was gathered as part of a doctoral dissertation. However, only part of the data collected was described and analyzed in the dissertation. Hence this paper uses both i) previously unreported data, and ii) data described and published in the dissertation, albeit to test a different set of research hypotheses (see Dreyfus, 2009).

level of individual components, and added to the repository. Importantly, many links discovered using these automated tools had been overlooked by, or were unknown to, IT service owners. This indicates that the theoretical (i.e., documented) IT architecture can deviate substantially from the actual architecture "in use," validating the broader motivation for our work.

We captured data on 407 IT architecture components. The components are divided into: eight "business groups;" 191 "software applications;" 92 "schemas;" 49 "application servers;" 47 "database instances;" and 20 "database hosts". These components form a layered architecture, typical of modern information systems, as we show later. Note that "business groups" are organizational units not technical objects. The dependence of particular business groups on specific software applications and infrastructure is integral to studies of enterprise architecture. While our focus in this paper is on IT architecture, we capture data on business groups so that we can analyze the implications of our findings for specific organizational units. (We did *not* capture data on business strategy or processes, which are also features of enterprise architecture.)

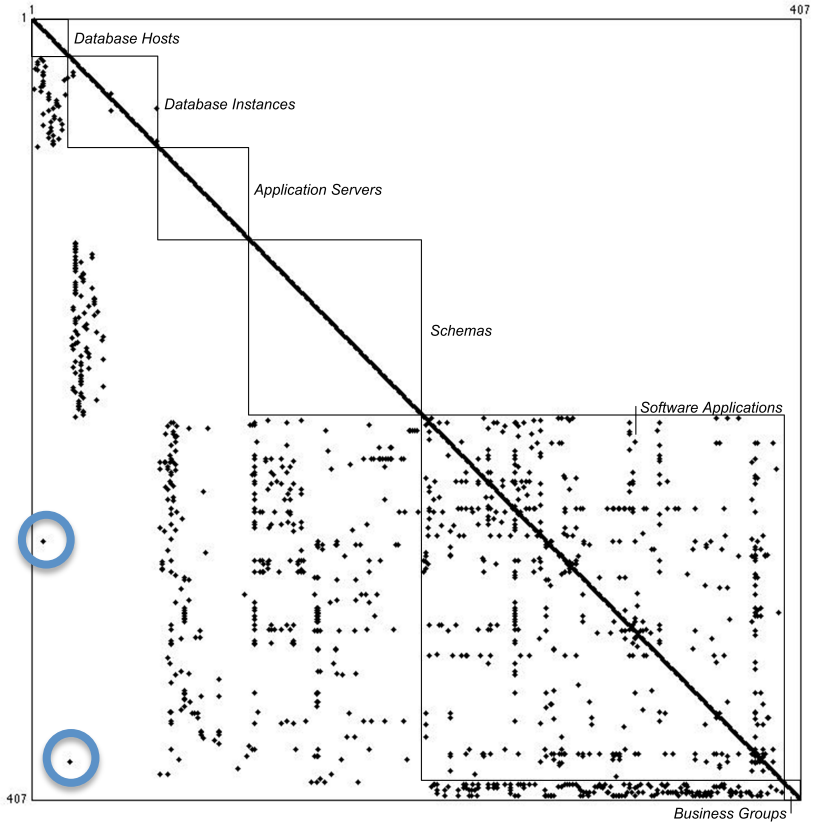
We captured data on 1,157 dependencies between components. In particular, this firm gathered information on four types of dependency between components – uses, communicates with, runs on, and instantiates. Business units *use* applications; Applications *communicate with* each other, use schemas, and *run on* application servers. Schemas in turn *instantiate* database instances that run on database hosts. Importantly, of these dependency types, "uses", "instantiates" and "runs on" possess a specific direction (i.e., they are asymmetric). In contrast, "communicates with" is a bi-directional (i.e., symmetric) dependency.

For a subset of components – specifically, software applications – we collected survey data on the cost of change, to test propositions about the impact of component coupling on IT agility. This data and our empirical tests are described in section six.

5.1. BioPharma's IT Architecture

A DSM showing BioPharma's IT Architecture is shown in **Figure 3**. The matrix is binary with marks in the off-diagonal cells indicating a direct dependency from row to column, hence a direct coupling from column to row. White space indicates there is no direct dependency between elements. To set the order of layers we use knowledge of the logical relationships between components. Usage flows from business groups (at the bottom) to applications, from applications to schemas and application servers, from schemas to database instances and from database instances to database hosts (at the top). Within layers, we order components using the component ID, an arbitrary numbering scheme. Note that "communicates with," the dependency captured between software applications in our data, is bi-directional, hence the marks in the rows and columns of this layer are symmetric around the main diagonal.

Figure 3: DSM Displaying BioPharma's Layered IT Architecture



As noted above, measures of direct coupling can be read directly from this DSM. Summing column entries gives the direct coupling (DC) of a component. Summing row entries gives the direct dependency (DD) of a component. White space to the right of a layer indicates that components in this layer do not depend on layers below. White space to the left of a layer indicates that components in this layer do not depend on layers above.

On the whole, this DSM suggests that the IT architecture displays a good separation of concerns: for the most part, schemas act as an interface between applications and the database instances and hosts. Schemas are also efficiently managed: one schema may serve several applications and one application may make use of several schemas. There are two exceptions, however, as indicated by the two circles, where specific applications appear to directly use a database instance or a database host. These exceptions may indicate poor encapsulation, non-standard practices, or measurement errors, and hence would be worth investigating further.

We note that this DSM combines several diagrams and matrices that are part of the TOGAF enterprise architecture framework (TOGAF, 2009). The mapping from business groups to applications (at the bottom of the DSM) corresponds to the "Application/Organization Matrix." The square submatrix of applications corresponds to the "Application Interaction Matrix" (AIM). The mapping from applications to schemas and servers (to the left of the AIM) corresponds to the "Application Technology Matrix." Finally, the mapping from schemas to database instances and database instances to database hosts contains the information needed to construct the "Application/Data Matrix," while also showing how the use of data by applications operates through particular schemas and database instances. Hence DSMs represent an analytical technique by which EA frameworks such as TOGAF can be made more operational.

The Application Interaction Matrix (AIM) is the largest submatrix in this DSM. It shows the dependencies that are related to interactions between software applications. In this dataset, dependencies between applications are captured by the term "communicates with," which does not possess directionality (i.e., we do not know which application requests a computation and which performs it). Hence the AIM is symmetric. In general however, capturing information on directionality is desirable. In particular, one application may always ask for a computation, and another may always supply the result. This distinction is lost if dependencies are assumed symmetric. However, if applications switch roles, sometimes requesting computational services and sometimes supplying results, a symmetric dependency would, in fact, be warranted.

5.2 Measuring Component Coupling

Figure 3 displays the layered structure of the IT architecture, but does not reveal other important architectural characteristics such as indirect coupling, cyclic coupling, hierarchy, or the presence of "Core" and "Peripheral" components. Matrix operations can be applied to the DSM to analyze these additional features. As noted earlier, the transitive closure of the matrix reveals indirect dependencies among components in addition to the direct dependencies (Sharman et al., 2002; Sharman and Yassine, 2004; MacCormack et al., 2006).

We constructed the Visibility Matrix for BioPharma, and applied the classification methodology described in Baldwin et al. (2014) to the resulting data for AC and AD (**Table 1**). We find that the firm's IT architecture has a core-periphery structure, with 132 "Core" components (i.e., 132 members in the largest cyclic group). Furthermore, all of the Core components in the system are software applications (however, not all software applications are Core; some are in the Periphery). Each of the layers in the IT architecture displayed in **Figure 3** has some components that are part of the "main flow" and others in the Periphery. In total, two-

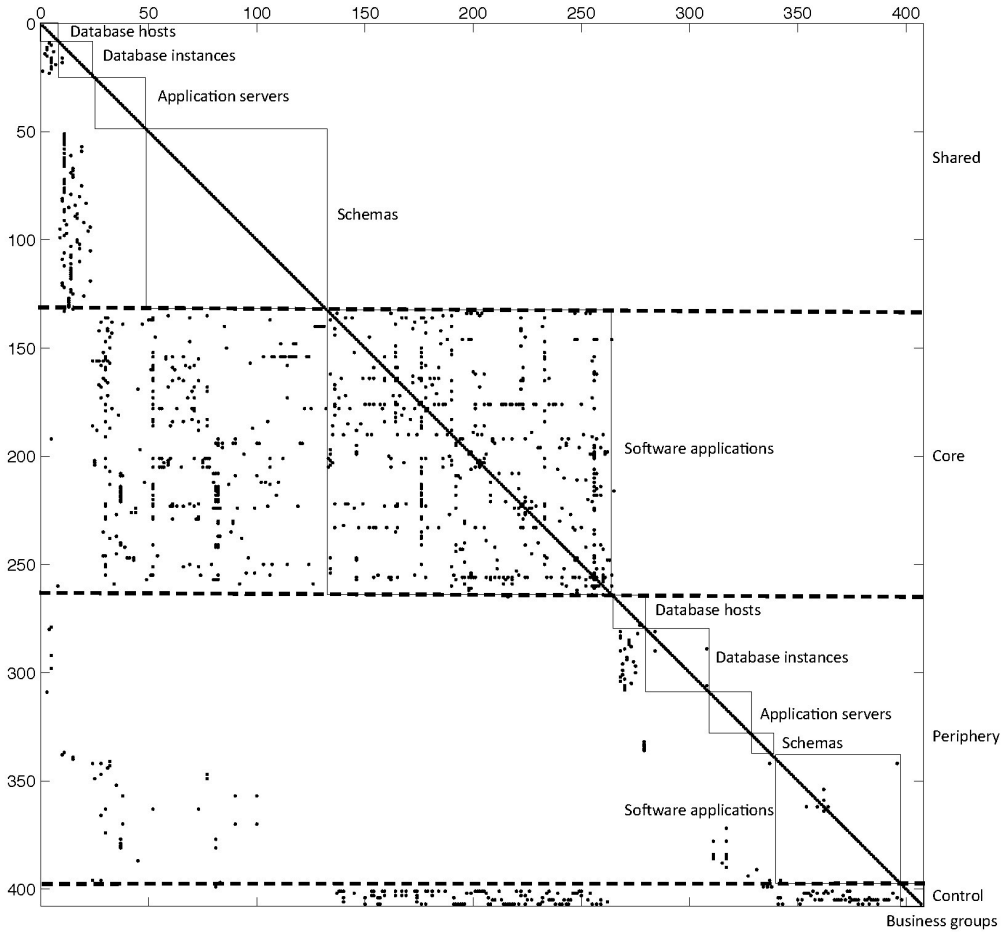
thirds of the components are part of the main flow and one-third is in the Periphery. We believe managers will find this type of classification scheme useful to set priorities, allocate resources, analyze costs, and understand potential differences in resource productivity.

Table 1: Distribution of Components in the Architecture by Layer and by Category.

	Shared	Core	Control	Periphery
Database hosts	8	0	0	12
Database instances	15	0	0	32
Application servers	27	0	0	22
Schemas	83	0	0	9
Software application	0	132	0	59
Business groups	0	0	7	1
TOTAL	**272**			**135**
Percent of Total	**66%**			**34%**

Figure 4 shows a reorganized view of BioPharma's DSM, organized first, by type of component (i.e., Shared, Core, Periphery, Control) and second, by layer. We call this the "core-periphery" view of the IT architecture. Components in Shared, Core or Control categories are directly or indirectly connected to all Core components (and potentially, to other components not in the Core). This represents the main flow of control in the system. Thus each main-flow component is connected to at least 132 other components. In contrast, the highest level of indirect connectivity for *any* peripheral component is only 7. Assuming component coupling is related to the cost of change, main-flow components will cost more to change, on average, than Peripheral components. We investigate this argument below.

Figure 4: Reorganized DSM showing Main Flow and Peripheral Components.



6. Predicting IT Agility

In the previous section, we show that BioPharma's IT architecture is comprised of components with different levels of coupling. In complex systems, heterogeneous levels of component coupling are the rule, not the exception (e.g., Lagerström, et al. 2014; Baldwin et al., 2014; Akaikine, 2010; Sturtevant, 2013). However, little empirical evidence exists about how different measures of coupling relate to the costs of change for a system's components. These costs determine the agility of a firm, with respect to evolving and adapting its IT systems.

In this section, we test our propositions about the relationship between measures of component coupling and IT agility – defined as the cost of IT change. Our analysis focuses on a

subset of the components for which information on the cost of change was available. Specifically, we predict the cost of change for *software applications*, using measures of coupling derived from the DSM. Focusing on one layer in the firm's IT architecture (i.e., as opposed to multiple layers) allows us to i) identify a specific respondent for data collection, ii) request quantitative data from respondents, and iii) ensure the data is comparable across components.

As we note earlier, dependency relationships between software applications are defined to be symmetric in our dataset. Hence it is not possible to explore the impact of differences between direct coupling (DC) and direct dependency (DD), nor differences in the hierarchical classification of components (a symmetric DSM contains only Core and Peripheral components, and no Shared or Control components). In general however, our method allows for the exploration of these issues, where the dependencies between components are asymmetric.

6.1 Dependent Variable: The Cost of Change for Software Applications

The cost to change each application was assessed via a survey sent to IT service owners. Respondents were asked to estimate the time, in person-years, to perform four IT operations: deploy, upgrade, replace and decommission. Operations were defined as follows: A component is *deployed* when it is put into production for the first time; a component is *upgraded* when it is replaced by a new version of the same component; a component is *replaced* when the existing component is removed from the information system and a new component with similar functionality is added to the information system; a component is *decommissioned* when it is removed from the information system (Baldwin and Clark, 2000). Given the uncertainty in estimating point values, respondents were asked to consider whether the effort for each operation fell within specific ranges, predetermined in a pilot-test with participants. [5]

[5] Specifically, respondents were asked to estimate whether the effort (in person-years) required for each operation fell into one of six buckets: <0.10, 0.10-0.249, 0.25-0.49, 0.50-0.99, 1.00-1.99, and > 2.00. (Dreyfus, 2009).

We received survey responses for 99 of the 191 software applications. The change cost estimates ranged from less than one-person-month to over two-person-years. Respondents could also indicate that the time to perform a given operation was unknown. Applications for which all change costs were unknown were removed from the dataset, resulting in a final sample of 77 applications.[6] For these 77 applications, we combined the change cost estimates for all four operations into a single measure, by calculating the mean change cost for those operations where a response was provided. The Cronbach's alpha for this aggregate measure was 0.78.[7]

6.2 Independent Variables: Measures of Component Coupling

All measures of coupling were calculated from a subset of the IT Architecture DSM containing only the 191 software applications. Measures of direct coupling (DC) for each component were calculated by summing entries in the columns of the direct dependency DSM. Measures of direct and indirect coupling (AC) for each component were calculated by summing entries in the columns of the visibility DSM. (Given dependencies between software applications are symmetric, the measures for coupling and dependency are identical.) Components that are members of cyclic groups were determined by inspecting the visibility DSM to identify those with equal levels of AC. We identified one very large cyclic group, containing 132 applications, which we call the "Core" (Baldwin et al, 2014) and two very small cyclic groups, comprising four and two applications respectively. The remaining 53 applications were not coupled to any others (i.e., they were in the "Periphery"). Of the 77 applications for which we have change cost data, 58 were part of the Core, and 19 were part of the Periphery.

[6] In prior work, Dreyfus (2009) and Dreyfus and Wyner (2011) use different screening criteria, resulting in a smaller sample of 62 responses for analysis. We discuss the sensitivity of our results to different screening criteria later.
[7] Where the estimate of change cost for an operation was missing, we substitute the mean level of change cost for that operation from all respondents to calculate Cronbach's alpha. Other ways of treating missing values result in a minimum value for alpha of 0.66 (acceptable) to a maximum value of 0.89 (extremely good).

As noted earlier, our final measure of component coupling, Centrality, is only defined for components that are members of the same cyclic group (i.e., components in the Core). For our empirical tests, we used a common social network measure called "Closeness Centrality," found by calculating the minimum path length from each component to all other Core components, summing the path lengths, and taking the inverse of this sum (Bounova and de Weck, 2012). The higher this metric, the more "central" is the component within the Core.

6.3 Control Variables

IT change costs may be affected by a number of factors that are unrelated to architecture, including the source of the component, the users of the component, its internal structure, and whether it was the focus of active development at the time of the survey. In addition, the respondent's experience with a given component might affect the appraisal of change cost in a systematic way. Hence data on the following variables were collected and included as controls:

(1) *Vendor* indicates whether an application is developed by a vendor (1) or in-house (0). One component missing data for this variable was assigned a value of 0.5.[8]

(2) *Client* indicates whether an application is accessed by end-users (1) or not (0).

(3) *Comp* indicates whether an application is focused on computation (1) or not (0).

(4) *N-Tier* indicates whether an application has an N-tier architecture (1) or some other type of architecture, such as client-server or monolithic (0).

(5) *Active* indicates whether, at the time of the survey, the component was being actively enhanced (1) or was in maintenance mode (0).

(6) *Res-Exp* measures the respondent's experience with the application in question (less than one year = 1; 1 – 5 years = 2; More than 5 years =3).

[8] Omitting the one application with no data provided about vendor did not change the results.

6.4 Descriptive Statistics

Table 2 presents descriptive data and a correlation matrix for our sample of software applications. Of note, *indirect and cyclic coupling could not be identified as separate constructs in this dataset*, *given they are 100% convergent*. This dynamic occurs because our final sample of 77 applications is divided strictly into two groups: 58 members of the Core (i.e., the largest cyclic group), which have the same level of indirect coupling; and 19 members of the Periphery, which have no coupling to other components. This empirical finding is driven by the particular properties of our dataset. Most significantly, dependencies for software applications are symmetric in our data, increasing the overall level of coupling, and the probability that a component is a member of a cyclic group. While indirect coupling and cyclic coupling are often correlated (Baldwin et al, 2014) they represent distinct theoretical constructs, possess distinct measures, and will not, in general, converge to such a degree.

Returning to **Table 2**, consistent with our hypotheses, direct coupling (DC), indirect coupling (AC) and cyclical coupling (CORE) are correlated with component change cost. Furthermore, direct coupling is correlated with indirect and cyclic coupling (r = 0.52). Significantly, for applications in the CORE (i.e., that are mutually dependent, with n=58), we find closeness centrality (CENT) is *not* correlated with change cost.[9] But as expected, closeness centrality and direct coupling are strongly correlated (r = 0.73). Among the control variables, Active components have higher change costs. Vendor provided components have lower centrality, are more likely to be Active and perform computations, and less likely to have N-tier architectures. Finally, components with N-tier architectures have higher levels of coupling.

[9] We note that some authors use a value of 0 to capture the centrality of components that are not part of the same network (i.e., that are not in the Core). The motivation is that this denotes an infinite path length to other components (the inverse of infinity being zero). If we adopt this approach, assigning a centrality measure of 0 to all *Peripheral* components, we unsurprisingly find a strong correlation between CORE and Centrality (r=0.96). In essence, centrality becomes a construct that is indistinguishable from indirect and cyclic coupling.

Table 2 Descriptive Statistics and Correlation Matrix

	Mean	*St.Dev*	*N*	COST	DC	AC	CORE	Vendor	Client	Comp	N-Tier	Active	Res-Exp	CENT
COST	2.35	1.33	77	1	0.32**	0.32**	0.32*	-0.21	-0.02	-0.09	0.16	0.33**	0.09	0.13
DC	3.58	2.83	77	0.32**	1	0.52***	0.52***	-0.19	0.1	0.01	0.39***	0.21	-0.08	0.73***
AC	99.68	56.48	77	0.32**	0.52***	1	1.0***	-0.17	0.11	-0.04	0.37***	0.06	-0.19	*n/a*
CORE	0.75	0.43	77	0.32**	0.52***	1***	1	-0.17	0.11	-0.04	0.37***	0.06	-0.19	*n/a*
Vendor	0.41	0.49	77	-0.21	-0.19	-0.17	-0.17	1	-0.06	0.39***	0.52***	0.35**	0.1	-0.4**
Client	0.71	0.45	77	-0.02	0.1	0.11	0.11	-0.06	1	0.15	0.27*	0.05	-0.11	0.16
Comp	0.39	0.49	77	-0.09	0.01	-0.04	-0.04	0.39***	0.15	1	-0.21	-0.05	0.12	0.00
N-Tier	0.53	0.5	77	0.16	0.39***	0.37***	0.37***	-0.52***	0.27*	-0.21	1	0.08	-0.14	0.43***
Active	0.26	0.44	77	0.33**	0.21	0.06	0.06	0.35**	0.05	-0.05	0.08	1	0.18	0.23
Res-Exp	2.04	0.84	77	0.09	-0.08	-0.19	-0.19	0.1	-0.11	0.12	-0.14	0.18	1	0.11
CENT	2.29	0.34	58	0.13	0.73***	*n/a*	*n/a*	-0.4**	0.16	0.00	0.43***	0.23	0.11	1

* p<0.05, ** p<0.01, and ***p<0.001

6.5 Empirical Results

We test our hypotheses by running OLS regressions predicting the impact of measures of coupling on component change costs. The results of our models are presented in **Table 3**. Model 1 contains only controls, showing two are significant: Vendor provided applications tend to have lower change costs and Active applications tend to have higher change costs. The control variables alone explain 19% of the variation in change cost across the 77 applications.

Our first hypothesis, H1 predicts that direct coupling is associated with change cost, a relationship supported by the correlations reported above. In Model 2 however, which includes control variables, we find that DC is only a marginal predictor of change cost (p= .06). This model explains 21% of the variation in change cost across applications. Our second and third hypotheses (H2/H3), as noted above, use measures that cannot be distinguished from each other. Hence we explore these hypotheses using only the binary measure of CORE membership. In Model 3, we find that CORE is a strong predictor of change cost (p = .005). This model explains 25% of the variation in change cost across components, providing support for both H2 and H3. Finally, in Model 4, we include both DC and CORE, but find only CORE is significant. This

model again explains 25% of the variation in change cost across applications, the same as in Model 3. However, we note that the F-statistic declines from 4.71 to 4.13. In sum, *H1* is only partially supported by our results; *H2/H3* are strongly supported but cannot be distinguished separately; and *H5* is not supported. In this dataset, indirect and/or cyclic coupling appears to be the dominant mechanism that explains the relationship between IT Architecture and IT Agility. Adding direct coupling to a model that includes CORE only makes the model worse.

Table 3 Regression Models predicting Change Cost

Sample	All Software Applications					Core Applications	
Model	**1**	**2**	**3**	**4**		**5**	**6**
Constant	2.58***	2.29***	1.90***	1.86***		3.21***	4.96***
Vendor	-1.33**	-1.21**	-1.30***	-1.26**		-1.51**	-1.80***
Client	-.18	-.15	-.15	-.15		-.62	-.64†
Comp	.31	.20	.25	.21		.28	.41
N-Tier	-.25	-.40	-.52	-.55		-.49	-.38
Active	1.58***	1.40***	1.50***	1.44***		1.96***	2.20***
Res-Exp	.02	.05	.10	.10		.02	.04
DC		.09†		.04			
CORE			.91**	0.81*			
CENT							-.80
Adj. R-square	0.19	0.21	0.25	0.25		0.27	0.28
F-Statistic	3.89**	3.85***	4.71***	4.13***		4.44***	4.15***
Observations	77	77	77	77		58	58
† p<0.1, * p<0.05, ** p<0.01, and ***p<0.001							

In models 5 and 6, we analyze only the 58 components in the Core, for which a value of closeness centrality can be calculated. All components in this group are members of the same cyclic group, and have the same level of direct and indirect coupling (AC). Hence we do not include measures of indirect or cyclic coupling in these models. Model 5 contains only control variables, and produces results consistent with model 1. Model 6 includes the measure of closeness centrality, which is not significant. We therefore reject *H4*.

6.6 Robustness Checks

We performed a number of checks to assess whether our results were sensitive to other assumptions or specifications of variables. First, we note that our basic specification did not control for the size of components, a variable that could plausibly affect the cost of changes. Data on component size (measured by the number of lines of code and files in each) was available for a subsample of 60 applications. We ran our models on this smaller sample, including size variables as controls. In these tests, the control variables including size were insignificant, while the results for our explanatory variables were consistent with those reported above. (The significance levels of all variables declined as a result of the decrease in sample size and statistical power.) We next conducted a test to explore the possibility that transformations of direct coupling might better predict change cost, given this variable has a skewed distribution and is truncated at zero. Specifically, we included the log of direct coupling in models, instead of its raw value. We found the transformed variable had more explanatory power than the raw variable (i.e., its use improved the results in Model 2). However, it still explained less of the variation in change cost than CORE, and was insignificant when included in a model with CORE. Finally, we explored whether direct coupling, or the log of this measure, contribute to explaining the variation in change cost among Core components (as we did for centrality). **Appendix A** reports the results of three models predicting change cost, the first being a model with controls, the second adding direct coupling, and the third adding the log of direct coupling. Direct coupling is not statistically significant in any model. This suggests that in this dataset, CORE is the most parsimonious and powerful variable that predicts the cost of change. Neither direct coupling, nor centrality, contributes additional explanatory power in our models.

7. Discussion

The main contribution of this paper is in developing and testing theory about the underlying mechanisms through which IT Architecture impacts IT Agility. In particular, we focus on exploring this relationship at the level of the individual components in the architecture. We develop measures for the amount of coupling between components, and show that these measures predict the cost of change for software applications. Our final model explains over 25% of the variation in the cost of change for the components we examine.

Our work suggests that *the dominant mechanism through which IT Architecture influences IT Agility is related to the level of indirect and/or cyclic coupling in the system*. Once the variation in change cost explained by this measure is accounted for, other measures of coupling add no further power. This implies a firm's agility to adapt its IT infrastructure is driven mainly by the potential for changes to propagate from one component to others via chains of dependencies. Critically, such data is not visible from inspection of a component's "nearest neighbors" in the IT Architecture. It represents "hidden structure" that can only be revealed by a comprehensive analysis of the coupling that exists throughout a system (Baldwin et al, 2014).

This finding lends support to the methods we employ, which reveal all pathways between system components along which potential changes may propagate. Specifically, we used Design Structure Matrices (DSMs), a novel network-based analytical technique, which allows us to i) identify discrete *layers* in the IT architecture associated with different types of component, and; ii) capture data on the *dependencies* between components; hence to assess their relative levels of coupling and position within the architecture. DSMs help us to characterize the layered, modular IT architectures required in the modern digital-age firm (Yoo et al, 2010; Tilson et al, 2010).

Our work helps to bridge the gap between EA and IT Architecture research. In particular, by including organizational units in our analysis, we can explore the link between the heterogeneous levels of coupling observed across the IT architecture, and the impact of this heterogeneity in terms of agility. Specifically, in the firm we study, seven of the eight business groups "depend upon" applications that are part of the Core, which tend to be more costly to change (see **Table 1**). In contrast, the remaining business group is not coupled to any application in the Core, and hence can adapt its IT systems at lower cost and with greater speed. Prior work suggests that agility has multiple dimensions, the relative importance of each which depends on the context (Sambamurthy et al, 2003). Our work disentangles which parts of an IT Architecture facilitate agility and which do not, critical for assessing whether a firm can consistently develop the new capabilities that it needs to compete in an increasingly dynamic business environment.

For managers, our methodology provides a picture of the actual *instantiated* architecture that they possess, as opposed to high level conceptual representations found in documents depicting a firm's IT architecture. The insights thereby generated should prove useful in several ways, including i) helping to plan the allocation of resources to different components, based upon predictions of the relative ease/difficulty of change; ii) monitoring the evolution of an architecture over time, as new components and/or dependencies are introduced (e.g., when a new firm is acquired) and; iii) identifying opportunities to improve the architecture, for example, by reducing coupling, and hence reducing the cost to change specific components.

Ironically, in this era of "big data," the lack of appropriately granular data may be the largest barrier to the systematic investigation of IT architecture using our methodology. At a minimum, firms need to capture data on the coupling and dependencies between different components in the architecture, and the way that these linkages evolve over time. To fully use

this data, they must also systematically capture data on the cost of change for components over time. *In most organizations with which we have worked, this type of data does not exist.* In some, efforts have been made to collect this data manually. However, there are many challenges associated with this approach, including a lack of incentives to provide accurate and timely information. In this study, we found substantial omissions in the data collected via survey, in comparison to the automated tools used to uncover system dependencies. In essence, firms may not know the "real" IT architecture that they possess. Eppinger and Browning (2012) state: "for most product DSM models, the data collection requires at least some amount of direct discussion with subject matter experts in order to draw out the tacit and system-level knowledge that may not be captured in the documentation." However, manual methods of dependency extraction are labor-intensive, and limit the scale, precision and accuracy of analyses. The solution is to develop automated ways to detect and capture relationships between components in the IT architecture. This implies the need for investment by firms that wish to adopt these methods.

Our work opens up the potential for further research to explore the relationship between IT architecture and performance. In this study, features of our dataset made it difficult to disentangle the effects of indirect and cyclic coupling, given these were convergent. However, in other settings, this will not always be true. We believe it important to study these mechanisms further, to fully understand the relationships they have with measures of IT change. Further, while we focused only on software applications in our analysis of change cost, our methods can be extended to look at the cost of change for other system components. For example, with the increasing importance of data in IT systems, it would be valuable to explore the impact of coupling on the cost of change for databases. We also believe work might usefully focus on the relationship between coupling and other (i.e., non-cost) performance measures. For example,

does modularity require trade-offs? And if so, what measures may be negatively impacted? Finally, studies across different organizations might reveal how measures of IT architecture impact *firm-level* performance. This area is most promising, given prior literature argues there is a strong linkage between certain types of IT architecture and firm-level agility. One might ask, for example, whether loosely coupled architectures, in general, facilitate a rapid response to unpredictable business challenges? Or are there subtle nuances to account for, with respect to different *layers* in the architecture (e.g., is the use of shared databases – with the patterns of coupling that this demands – a best practice)? Our methodology allows us to answer such questions with a robust approach that can be replicated consistently across studies.

Of course, our study is subject to a number of limitations that must be considered when assessing the generalizability of results. In particular, while our unit of analysis is a component in the IT architecture (of which we have 400+ observations in this dataset), the data to test our theoretical propositions comes from a single firm. Hence additional empirical work is needed to validate that our results hold for other firms. We may find that different types of firm, or different managerial processes within similar firms, can influence the results that we find. Furthermore, questions remain as to the different types of layers and components that should be included in an analysis of IT architecture, as well as the types of coupling between them. For example, different types of dependency (e.g., "uses" versus "communicates with") may have more or less importance in predicting different dimensions of performance and agility. And there may be important intermediate variables (e.g., the use of standards; see Tiwana and Konsynski, 2010) that mediate the relationship between component coupling and change cost. Ultimately, we hope our work provides a platform to enable future research to answer these and other important questions about the relationship between IT architecture and IT agility.

References

Adomavicius, G., Bockstedt, J. C., Gupta, A., and Kauffman, R. J. 2008. Making sense of technology trends in the information technology landscape: A design science approach. *MIS Quarterly* 32, 4, 779-809.

Aier, S. 2014. The role of organizational culture for grounding, management, guidance and effectiveness of enterprise architecture principles. *Information Systems and E-Business Management 12*, 1, 43-70.

Aier, S., Gleichauf, B., and Winter, R. 2011. Understanding Enterprise Architecture Management Design: An Empirical Analysis. In *Proc. of Wirtschaftsinformatik,* Association for Information Systems.

Aier, S. and Winter, R. 2009. Virtual decoupling for IT/business alignment–conceptual foundations, architecture design and implementation example. *Business & Information Systems Engineering 1,* 2, 150-163.

Akaikine, A. 2010. The Impact of Software Design Structure on Product Maintenance Costs and Measurement of Economic Benefits of Product Redesign. System Design and Management Program Thesis, Massachusetts Institute of Technology.

Alexander, C. 1964. *Notes on the Synthesis of Form*. Harvard University Press.

Baldwin, C. and Clark, K. 2000. *Design Rules, Volume 1: The Power of Modularity*. MIT Press.

Baldwin, C., MacCormack, A., and Rusnack, J. 2014. Hidden structure: Using network methods to map system architecture. *Research Policy*, Article in Press. Accepted May 19 2014.

Barabási, A. 2009. Scale-free networks: A decade and beyond. *Science* 325, 5939, 412-413.

Berente, N. and Yoo, Y., 2012. Institutional contradictions and loose coupling: Postimplementation of NASA's enterprise information system. *Information Systems Research*, *23*(2), pp.376-396.

Boh, W. F., and Yellin, D. 2007. Using enterprise architecture standards in managing information technology. *Journal of Management Information Systems* 23, 3, 163-207.

Bounova, G., de Weck, O.L. 2012. Overview of metrics and their correlation patterns for multiple-metric topology analysis on heterogeneous graph ensembles, *Phys. Rev. E* 85.

Braha, D., Minai, A.A., and Bar-Yam, Y., 2006. Complex Engineered Systems: Science Meets Technology. Springer: New England Complex Systems Institute, Cambridge, MA.

Buckl, S., Ernst, A. M., Matthes, F., Ramacher, R., and Schweda, C. M. 2009. Using enterprise architecture management patterns to complement TOGAF. In *Proc. of the Enterprise Distributed Object Computing Conference (EDOC'09). IEEE International* 34-41. IEEE.

Byrd, T.A., and Turner D.E. 2000. Measuring the flexibility of information technology infrastructure: Exploratory analysis of a construct. *Journal of Management Information Systems* 17, 1, 167-208.

Chidamber, S. R., and Kemerer, C. F. 1994. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering* 20, 6, 476-493.

Clark, K.B. 1985. The interaction of design hierarchies and market concepts in technological evolution. *Research Policy* 14, 5, 235–251.

DODAF, Department of Defense Architecture Framework. Version 1.5, 2007, Technical report, Department of Defense, USA.

Dietz, J. L., and Hoogervorst, J. A. 2011. A critical investigation of TOGAF: based on the enterprise engineering theory and practice. In *Advances in Enterprise Engineering V, Lecture Notes in Business Information Processing Volume 79, Proc. of the 1st Enterprise Engineering Working Conference (EEWC),* 76-90, Springer Berlin Heidelberg.

Dreyfus, D. 2009. *Digital Cement: Information System Architecture, Complexity, and Flexibility*. PhD Thesis. Boston University Boston, MA, USA, ISBN: 978-1-109-15107-7.

Dreyfus D. and Wyner, G. 2011. Digital cement: Software portfolio architecture, complexity, and flexibility. In *Proc. of the Americas Conference on Information Systems (AMCIS)*, Association for Information Systems.

Duncan, N. 1995. Capturing Flexibility of Information Technology Infrastructure: A Study of Resource Characteristics and Their Measure. *Journal of Management Information Systems* 12, 2, 37-57.

Eppinger, S.D., and Browning. T.R. 2012. *Design structure matrix methods and applications*. MIT press.

Eppinger, S. D., Whitney, D.E., Smith, R.P., and Gebala, D. A. 1994. A model-based method for organizing tasks in product development. *Research in Engineering Design* 6, 1, 1-13.

Franke, U., Johnson, P., and König, J. 2014. An architecture framework for enterprise IT service availability analysis. *Software & Systems Modeling* 13, 4, 1417-1445.

Gao, L. S, and Iyer, B. 2006. Analyzing Complementarities Using Software Stacks for Software Industry Acquisitions. *Journal of Management Information Systems* 23, 2, 119-147.

Hanseth, O. and Lyytinen, K., 2010. Design theory for dynamic complexity in information infrastructures: the case of building internet. *Journal of Information Technology*, *25*(1), pp.1-19.

Holland, J. H. 1992. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*, seconded. MIT Press, Cambridge, MA.

Johnson, P. and Ekstedt, M. 2007. *Enterprise Architecture: Models and analyses for information systems decision making*. Studentlitteratur.

Jonkers, H., Lankhorst, M. M., ter Doest, H. W., Arbab, F., Bosma, H., and Wieringa, R. J. 2006. Enterprise architecture: Management tool and blueprint for the organisation. *Information Systems Frontiers* 8, 2, 63-66.

Kauffman, S.A. 1993. *The Origins of Order*. Oxford University Press, New York.

Kim, G., Shin, B., Kim, K.K., and Lee, H.G. 2011. IT capabilities, process-oriented dynamic capabilities, and firm financial performance. *Journal of the Association for Information Systems* 12, 7.

Kluge, C., Dietzsch, A., and Rosemann, M. 2006. How to realize corporate value from enterprise architecture. In *Proc. of the European Conference on Information Systems* (*ECIS),* 1572-1581, Association for Information Systems.

Lagerström, R., Franke, U., Johnson, P., and Ullberg, J. 2009. A method for creating enterprise architecture metamodels: applied to systems modifiability analysis. *International Journal of Computer Science and Applications* 6, 5, 89-120.

Lagerström, R., Johnson, P., and Höök, D. 2010. Architecture Analysis of Enterprise Systems Modifiability: Models, Analysis, and Validation. *Journal of Systems and Software* 83, 8, 1387-1403.

Lagerström, R., Sommestad, T., Buschle, M., and Ekstedt, M. 2011. Enterprise architecture management's impact on information technology success. In *Proc. of the 44th Hawaii International Conference on System Sciences (HICSS-44)*, IEEE.

Lagerström, R., Baldwin, C., MacCormack, A., and Dreyfus, D. 2013. Visualizing and Measuring Enterprise Architecture: An Exploratory BioPharma Case. In *Proc. of the 6th IFIP WG 8.1 Working Conference on the Practice of Enterprise Modeling (PoEM)*. Springer.

Lagerström, R., Baldwin, C., MacCormack, A., and Aier, S. 2014. Visualizing and Measuring Enterprise Application Architecture: An Exploratory Telecom Case. In *Proc. of the Hawaii International Conference on System Sciences (HICSS-47)*, IEEE.

LaMantia, M., Cai, Y., MacCormack, A., and Rusnak, J. 2008. Analyzing the evolution of large-scale software systems using design structure matrices and design rule theory: Two exploratory cases. In *Proc. of the 7th Working IEEE/IFIP Conference on Software Architectures (WICSA7)*.

Langlois, R.N. 2002. Modularity in technology and organization. *Journal of economic behavior & organization* 49, 1, 19-37.

Lankhorst, M. 2009. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. The Enterprise Engineering Series. Springer.

Lawrence, P.R., and Lorsch, J.W. 1967. Differentiation and integration in complex organizations. *Administrative science quarterly*.

Liu, H.,Ke, W., Wei, K.K., and Hua, Z. 2013. The impact of IT capabilities on firm performance: The mediating roles of absorptive capacity and supply chain agility. *Decision Support Systems* 54, 3, 1452-1462.

MacCormack, A., Rusnak, J., and Baldwin, C. 2006. Exploring the structure of complex software designs: an empirical study of open source and proprietary code. *Management Science* 52, 7, 1015–1030.

MacCormack, A. 2010. The Architecture of Complex Systems: Do “Core-Periphery" Structures Dominate?. In *Proc. of Academy of Management*.

MacCormack, A., Baldwin, C., and Rusnak, J. 2012. Exploring the duality between product and organizational architectures: A test of the "mirroring" hypothesis. *Research Policy* 41, 8, 1309-1324.

MacCormack, A., and Sturtevant, D. 2016. Technical Debt and System Architecture: The Impact of Coupling on Defect-Related Activity. *Journal of Systems and Software*, accepted for publication.

Matthes, F., Buckl, S., Leitel, J., and Schweda, C.M. 2008. *Enterprise architecture management tool survey 2008*. Techn. Univ. München.

Mead, C. and Conway, L. 1980. *Introduction to VLSI Systems*. Addison-Wesley Publishing Co.

MODAF, Ministry of Defence Architecture Framework, Version 1.2.003 (2008). Technical report, Ministry of Defence, UK.

Närman, P., Holm, H., Johnson, P., König, J., Chenine, M., and Ekstedt, M. 2011. Data accuracy assessment using enterprise architecture. *Enterprise Information Systems* 5, 1, 37-58.

Orlikowski, W.J., and Iacono, C.S. 2001. Research commentary: Desperately seeking the “IT” in IT research - A call to theorizing the IT artifact. *Information systems research* 12, 2, 121-134.

Ozkaya, I. 2012. Developing an architecture-focused measurement framework for managing technical debt. In *Software Engineering Institute blog*. http://blog.sei.cmu.edu/post.cfm/developing-an-architecture-focused-measurement-framework-for-managing-technical-debt, accessed Sept. 2013.

Parnas, D. L. 1972. On the criteria to be used in decomposing systems into modules. *Communications of the ACM* 15, 12, 1053-1058.

Rivkin, J.W. 2000. Imitation of complex strategies. *Management Science* 46,824–844.

Rivkin, J.W., and Siggelkow, N. 2007. Patterned interactions in complex systems: implications for exploration. *Management Science* 53 (7), 1068–1085.

Roeleven, S. 2010. Why Two Thirds of Enterprise Architecture Projects Fail: An explanation for the limited success of architecture projects. Whitepaper, Software AG.

Ross, J.W. 2003. Creating a strategic IT architecture competency: Learning in stages. MIT Sloan School of Management Working Paper No. 4314-03.

Ross, J.W., and Westerman, G. 2004. Preparing for utility computing: The role of IT architecture and relationship management. *IBM systems journal*, 43, 1, 5-19.

Ross, J.W., Weill, P., and Robertson, D. 2006. *Enterprise Architecture As Strategy: Creating a Foundation for Business Execution*. Harvard Business School Press.

Salmela, H., Tapanainen, T., Baiyere, A., Hallanoro, M., and Galliers, R. 2015. IS Agility Research: An Assessment and Future Directions. *ECIS 2015 Completed Research Papers*. Paper 155.

Sambamurthy, W. and Zmud, R. 2000. The Organizing Logic for an Enterprise's IT Activities in the Digital Era: A Prognosis of Practice and a Call for Research. *Information Systems Research* 11, 2, 105-114.

Sambamurthy, V., Bharadwaj, A., and Grover, V. 2003. Shaping Agility through Digital Options: Reconceptualizing the Role of Information Technology in Contemporary Firms. *MIS Quarterly* 27, 2, 237-263.

Sanchez, R.A., Mahoney, J.T. 1996. Modularity, flexibility and knowledge management in product and organizational design. *Strategic Management Journal* 17, 63–76.

Schilling, M.A. 2000. Toward a general systems theory and its application to interfirm product modularity. Academy of Management Review 25 (2), 312–334.

Schmidt, C. and Buxmann, P. 2011. Outcomes and success factors of enterprise IT architecture management: empirical insight from the international financial services industry. *European Journal of Information Systems* 20, 168–185.

Seppänen, V., Heikkila, J., and Liimatainen, K. 2009. Key issues in EA-implementation: case study of two Finnish government agencies. In *Proc. of the IEEE Conference on Commerce and Enterprise Computing (CEC'09),* 114-120, IEEE.

Sharman, D., Yassine, A., and Carlile, P. 2002. Characterizing modular architectures. In *Proc. of the ASME 14th International Conference on Design Theory & Methodology*, DTM-34024, Montreal, Canada, September.

Sharman, D. and Yassine, A. 2004. Characterizing complex product architectures. *Systems Engineering Journal* 7, 1.

Simon, H. A. 1962. The architecture of complexity. *American Philosophical Society* 106, 6, 467-482.

Simon, D., Fischbach, K., and Schoder, D. 2013. An Exploration of Enterprise Architecture Research. *Communications of the Association for Information Systems* 32, 1, 1-72.

Sommestad, T., Ekstedt, M., and Holm, H. 2013. The Cyber Security Modeling Language: A Tool for Assessing the Vulnerability of Enterprise System Architectures. *IEEE Systems Journal*, Online-first.

Sosa, M. E., Mihm, J., and Browning, T. R. 2013. Linking Cyclicality and Product Quality. *Manufacturing & Service Operations Management 15*, 3, 473-491.

Sosa, M., Eppinger, S., and Rowles, C. 2007. A network approach to define modularity of components in complex products. *Transactions of the ASME* 129, 1118-1129.

Steward, D. 1981. The design structure system: A method for managing the design of complex systems. *IEEE Transactions on Engineering Management* 3, 71-74.

Sturtevant, D. J. 2013. *System design and the cost of architectural complexity*. Diss. Massachusetts Institute of Technology.

Tamm, T., Seddon, P. B., Shanks, G., and Reynolds, P. 2011. How does enterprise architecture add value to organisations. *Communications of the Association for Information Systems* 28, 1, 141-168.

Tanriverdi, H. A. Rai and N Venkatraman. 2010. Reframing the dominant quests of information systems strategy research for complex adaptive business systems, *Information Systems Research*, Vol 21 No 4, 2010.

Tilson, D., Lyytinen, K. and Sørensen, C., 2010. Research commentary-digital infrastructures: the missing IS research agenda. *Information systems research*, *21*(4), pp.748-759.

Tiwana, A. and B. Konsynski. 2010. Complementarities between organizational IT architecture and governance structure. Information Systems Research, Vol 21, No 2, 2010.

Tiwana, A., B Konsynski and A Bush. 2010. Platform Evolution: Coevolution of platform architecture, governance and environmental dynamics, Information Systems Research. Vol 21. No 4, 2010.

TOGAF, The Open Group Architecture Framework, Version 9, 2009. http://pubs.opengroup.org/architecture/togaf9-doc/arch/ (viewed 11/3/14).

Tushman, M.L., Murmann, P.J. 1998. Dominant designs, technological cycles and organizational outcomes. In: Staw, B., Cummings, L.L. (Eds.), *Research in Organizational Behavior*, vol. 20. JAI Press, Greenwich, CT.

Tushman, M.L., Rosenkopf, L. 1992. Organizational determinants of technological change: toward a sociology of technological evolution. *Research in Organizational Behavior* 14, 311–347.

Ullberg, J., Johnson, P., and Buschle, M. 2012. A Language for Interoperability Modeling and Prediction. *Computers in Industry* 63, 8, 766-774.

Ulrich, K. 1995. The role of product architecture in the manufacturing firm. *Research Policy* 24, 419–440.

Vakkuri, E. T. 2013. *Developing Enterprise Architecture with the Design Structure Matrix*. Master Thesis. Tampere University of Technology, Finland.

Weill, P. 2007. Innovating with Information Systems: What do the most agile firms in the world do. In *Proc. of the 6th e-Business Conference*, Barcelona.

Whitney, D.E. (Chair) and the ESD Architecture Committee. 2004. The Influence of Architecture in engineering Systems. Engineering Systems Monograph,

Winter, R. and Fischer, R. 2007. Essential Layers, Artifacts, and Dependencies of Enterprise Architecture. Journal of Enterprise Architecture 3, 2, 7-18.

Yoo, Y., Henfridsson, O., and Lyytinen, K. 2010. Research Commentary—The New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research. *Information Systems Research* 21, 4, 724-735.

Zachman, J. A. 1987. A Framework for Information Systems Architecture. *IBM Systems Journal* 26, 3, 276-292.

Appendix A: Models Predicting Change Cost only for Core Components

Sample	Core Applications			
Model	**1**	**2**	**3**	**4**
Constant	3.21***	3.15***	3.16***	4.96***
Vendor	-1.51**	-1.49**	-1.50**	-1.80***
Client	-.62	-.61	-.61	-.64
Comp	.28	.26	.27	0.41
N-Tier	-.49	-.50	-.49	-.38
Active	1.96***	1.93***	1.94***	2.20***
Res-Exp	.02	.03	.03	.04
DC		3.15		
Ln (DC)			.05	
CENT				-.80
Adj. R-square	0.27	0.25	0.25	0.28
F-Statistic	4.44***	3.75**	3.74**	4.15***
Observations	58	58	58	58
† p<0.1, * p<0.05, ** p<0.01, and ***p<0.001				