WEIGHTING OF INTEGRATION QUALITIES IN IS ARCHITECTURES: A PRODUCTION CASE

Complete Research

Fuerstenau, Daniel, Freie Universität Berlin, Germany, daniel.fuerstenau@fu-berlin.de

Glaschke, Christian, Universität Potsdam, Germany, christian.glaschke@wi.uni-potsdam.de

Abstract

We suggest a method that conceptualizes a company's IS architecture—its set of information systems (IS) and integrations—as a weighted and directed graph. Going beyond an undifferentiated treatment of integration qualities, we present a scoring model to assign integration weights to integration links. Thereupon, we introduce weighted centrality measures from network analysis to identify important systems in the architecture with respect to their architectural embeddedness. Drawing on the case of a production company, we demonstrate our approach by concentrating on selected systems and integrations, such as the close and complex integration of an ERP and a product lifecycle management (PLM) system. Focusing on the technical challenges of enterprise transformations, our work contributes to enterprise architects' toolbox as it enables a more fine-grained understanding of critical systems with high change complexity.

Keywords: IS architectures; integration; weighted and directed graphs; centrality measures; change complexity

1 Integration as a Driver of Change Complexity

Integration in different flavours enables seamless interdepartmental and interorganizational communication. However, numerous, close, and diverse integration linkages often make difficult fundamental changes of IS architectures and block the innovative capability of a company (Dreyfus and Iyer, 2008; Lagerström et al., 2013; Lagerström et al., 2014; Ross et al., 2006).

One example is an IT transformation effort by *ProductionCo*, a German production company. Because its legacy enterprise system lacked functionality, the company aimed at migrating its core business processes to another, more powerful enterprise system. On its way, the company had to make tough decisions on how to maintain its operations while proceeding with the migration. Thus, multiple add-ons and extensions to the original system emerged—mostly in MS Excel and Access—to overcome serious functional gaps and limitations. Expert interviews showed that these workarounds were meaningful extensions of the original system. Nonetheless, while they were intended as temporary solutions, the company today is still held back by its deep integration in the architecture, and aims to overcome the inertia potential exhibited by these solutions.

This is one example from a class of problems linking the quality of integration in an enterprise's IS architecture to its change complexity. Integrations clearly come in different qualities and flavours, e.g. automated versus manual, close versus loose, online versus offline. Our main argument in this article is that different integration qualities produce different effects on the change complexity of an IS architecture. We refer to change complexity as the ease with which an existing IS architecture can be transformed from a certain *as-is* state to a desired *should-be* state. We believe that enterprise transformation research can profit from going beyond an undifferentiated treatment of integration qualities. We see a need to explore how different types of integration affect an IS architecture's change complexity.

We work toward our objective by conceptualizing IS architectures—a company's set of information systems and integrations—as directed and weighted graphs. In this graph, each node represents a logical IT application, such as "Hydra", a standard manufacturing execution system (MES), or an Excel-based "label printing application". Each link describes an integration, e.g. message passing, method invocations, function calls, or data sharing (Dreyfus, 2009). For example, the MES provides data to the label printing application. Therefore, the label printing application depends on the MES. In general, we assume that *if* system *A* provides data to system *B*, *then* system *B* depends on system *A*. While one may also consider the business process level, we find it a useful starting point to focus particularly on information systems and integrations as we aim to support enterprise transformations with respect to change complexity arising from technical integrations.

While previous work has focussed on *unweighted* graphs (Dreyfus and Iyer, 2008; Fuerstenau and Rothe, 2014; Lagerström et al., 2013; Lagerström et al., 2014), we introduce a method that supports IT architects and managers in evaluating the weight of these dependencies and the implications that arise for the inertia potential of an IS architecture by using weighted measures. Our method contributes to research on enterprise transformations in the following ways. Firstly, the method allows us to estimate the necessary effort and to assess the potential risks when migrating from one core enterprise system to another in a more comprehensible and systematic way. Secondly, our method allows for a definition of the necessary steps, when a company migrates a core system, earlier and in a more reliable manner. This has important practical implications, as it informs managers and architects and ensures avoiding a "criticality trap" in which dependencies within an architecture are underestimated or sensed too late.

To achieve these contributions, we choose a design science approach. In section 2, we introduce the relevant body of theoretical and practical knowledge on the effect of different qualities of integrations on the expected degree of change complexity in an IS architecture. Based upon the problem definition, section 3 presents the building blocks of our design artifact: the method and its foundational constructs in the form of measures from network analysis for weighted and directed graphs as well as respective visualizations for intelligible stakeholder discussions. Section 4 demonstrates the method's evaluation at the case site of ProductionCo. Finally, section 5 discusses the results' implications for the research question, concludes and points to fruitful future research areas.

2 Coping with Complexity: Visualizing and Measuring Integration

Every application integration project is a new challenge (Aier and Schönherr, 2006): It encompasses specifying syntactical and semantic definitions among different organizational units or departments, aligning organizational architectures, and committing to governance and change procedures. While recent integration technologies such as service-oriented architectures and web services (Erl, 2006), and cloud computing have largely eased the process of integrating new solutions into an IS architecture, the number of applications and integrations has increased at an ever higher pace, creating more and more distributed IS architectures with respect to the number and diversity of applications and integrations, and their rate of change (Schneberger and McLean, 2003; Schütz et al., 2013b).

Integration comes in different flavours (Mertens, 2013). Firstly, dependencies generally specify exchanges on multiple levels (Dreyfus, 2009): pragmatic, semantic, and syntactic. On a syntactic level, a dependency exists between two components that share some communication or physical interface. On a semantic level, common meanings are attached to these interfaces and, finally, on a pragmatic level, even purposes (e.g. business intentions, rules, policies) will be shared. Secondly, some integrations are uni-directional while others are bi-directional, which refers to the observation that often a change in one application affects the other one, while this is not true the other way around. Thirdly, integrations differ in their degree of closeness (Engels et al., 2008). While some applications are cobbled together in close ways, service-oriented paradigms suggest a loose coupling. Fourthly, integrations vary in their integration complexity. Various factors, such as the number of data fields or functional scope, drive an integration’s complexity (Mertens, 2013). Taken all together, these arguments suggest that it matters what qualities an integration exhibits in discerning between critical and non-critical ones.

To manage the resulting complexity in an emergent IS architecture, visual and quantitative approaches to enterprise architecture (EA) analysis of dependencies have been presented.

A first set of approaches has suggested analysing dependencies visually, drawing on software representation techniques such as UML or domain-specific description languages from the EA field such as ArchiMate (refer to Lankhorst 2009 for a discussion of different visualization approaches). To take one example, Matthes (2008) suggests multi-layered software maps to visualize the applications of a company and its dependencies. Various EA tools are available to support this task. As an extension to manually-created visualizations, Aier and Winter (2009) propose an approach to support semi-automatic domain mapping of applications by drawing on clustering techniques.

A second set of approaches has suggested a quantitative analysis of enterprise architectures. Schütz et al. (2013b), for instance, suggest measuring the complexity of an architecture by turning to diversity measures, such as the Herfindahl index from economics. Lankhorst (2009, p. 191ff.), in his book “Enterprise Architecture at work”, introduces another mechanism for quantitative analysis of dependencies by focusing on performance measures, such as workload, processing time, and utilization. Lagerström et al. (2013) and Lagerström et al. (2014) suggest mapping IS architectures to directed graphs, and thereupon, propose measures to identify cyclic groups and component characteristics to determine the flexibility of an architecture. The approach closest to our thinking is Fuerstenau and Rothe (2014). These authors suggest mapping IS architectures (applications and integrations) to unweighted and undirected graphs to identify critical shadow IT systems with respect to their architectural embeddedness by drawing on centrality measures from network analysis.

However, the overwhelming majority of approaches to EA analysis treat all dependencies equally. Thus, the quality of integration remains undifferentiated. This can strongly misrepresent the influence of critical applications within the architectures, i.e. those with closer and more complex integrations. Whereas loose couplings feature a flexible reconfiguration without affecting the other applications, close integration causes changes to ripple through the architecture. In addition, more complex integrations increase the likelihood that these paths are actually carrying important information.

Altogether, we showed that links among applications come in different flavours. We will demonstrate that it is beneficial to go beyond an undifferentiated treatment of integrations and to discern them by their quality. Our aim is to identify critical applications within an IS architecture with respect to their potential influence on change complexity within the architecture. Thus, we construct a method to weight integration qualities and to measure their impact on change complexity.

3 A Method to Weight Integration Qualities in IS Architectures

We have adopted a design science research approach for this study (Gregor and Jones, 2007; Hevner et al., 2004) which advances the value of utilizing design artifacts to advance knowledge and at same time provide a practical utility in solving an identified research problem (Goes, 2014). As our design artifact (Gregor and Hevner, 2013; Gregor and Jones, 2007), we develop a method for visualizing and assessing the criticality of an application in an organizational IS setup based on the quality of integration with other applications in the architecture. The development of artifacts in the form of reusable design patterns, principles, or guidelines, has been increasingly identified to be of great value to both IS research and practice (Kuechler and Vaishnavi, 2008; Winter, 2008). Fundamentally, design science advocates a systematic generation of useful knowledge through the construction and evaluation of design artifacts (Hevner and Chatterjee, 2010). Additionally, artifacts constructed in design science research present approaches, practices and capabilities which enable the effective and efficient analysis, design, application, and utility of information systems to be achieved (Denning, 1997; Hevner et al., 2004). The design research process that has been adopted follows the design science research methodology (DSRM) framework advanced by Peffers et al. (2007). This research process has been selected because it offers a synthesized general research model which has been well built on other approaches (Gregor and Hevner, 2013). The subsequent section describes the suggested method.

3.1 Weighting of Integration Qualities: Closeness and Complexity

As our objective is to evaluate an IS architecture's change complexity with respect to technical aspects, we focus on integration qualities among applications on a technical level as these qualities offer a more efficient model of the strength of dependencies than associations on a business process level (cf. Dreyfus and Iyer, 2008; Lagerström et al., 2014).

We suggest two dimensions along which we want to discern the quality of linkages among systems: (a) closeness and (b) complexity. We refer to *closeness* as the extent to which one system depends on another system to remain available, functional, and consistent (Figure 1). Closeness is an important dimension along which one needs to evaluate the quality of integration, as loosely-coupled applications can be treated as modules that are more flexible and robust when changes occur (Erl, 2006). In contrast, close integration results in changes rippling through the architecture in unpredictable ways.

The concept of loose and close coupling is defined here briefly (cf. Baumann et al., 2009). Let *B* be a system that depends on *A*, then *B* is loosely-coupled with *A*, if:

- Dependency on the availability: *B* is also available when *A* or the integration link to *A* is not available.
- Confidence: *B* does not trust that *A* satisfies preconditions or post-conditions of operations.
- Knowledge: *B* has only so much knowledge about *A* as necessary for the correct use of the operations. This includes the interfaces syntax and the structure and semantics of the passed and returned data.

We turn to *complexity* (*b*) as our second dimension to qualify integration linkages. By complexity, we refer to the non-triviality of the integrations regarding their information intensity and functional scope.

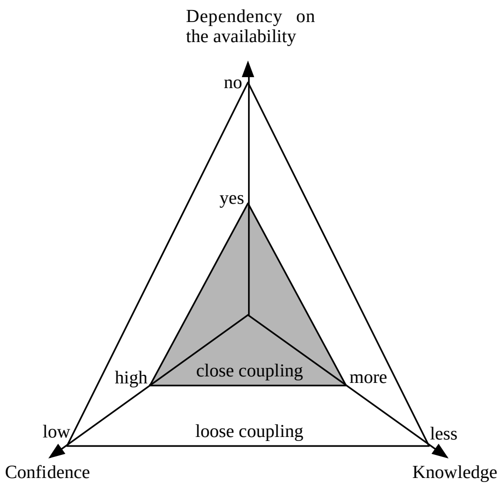

Figure 1. Close vs. loose coupling

Table 1 refers to possible indicators for the complexity dimension. For the evaluation of the complexity of an integration, we suggest using only a subset of these indicators. In particular, we believe that (*i*) the number of fields and (*ii*) number of message types are necessary and sufficient indicators. The *number of fields* serves as a useful indicator to reflect integration complexity, as it aptly captures different data and thus business objects transferred via an interface. The *number of message types* captures well the functional strength of the integration (e.g. whether one application depends on validation and rule-checking tasks performed in the other application).

Qualities	**Definition**
Number of fields	Total number of fields that are defined in an interface
Number of message types	Total number of message types that are defined in an interface
Standardization	Extent to which industry-, domain-, or company-wide conventions are used to ensure compatibility among applications
Usage frequency	If the interface is used frequently or infrequently
Data volume	How much data is transferred via the interface
Data security	Authentication, trust, and validation can provide increased complexity

Table 1. Potential indicators to reflect integration complexity

We next describe the rationale for not considering the other indicators mentioned in Table 1. To begin, a rating of usage frequency and volume of data did not seem meaningful to us, because these properties allow no statement about the strength of the dependency. In particular, a rarely used interface (e.g. master data) can be as dependent on an application as one that operates continuously and produces a lot of data. Data security is often seen in projects as a complexity driver but carries—from a purely technical point of view—no potential to assess the strength of a dependency. In many cases, data security is a major issue in inter-organizational integration. As our main focus is on inter-company integration, however, we did not consider data security as an indicator for the complexity of an integration. In addition, we also did not explicitly consider standardization of an integration, as it has a bright side and a dark side. Standardization enables compatibility among actors, i.e. among different information systems (David and Greenstein, 1990). If an interface is highly standardized, the change complexity can be lower as modules can be exchanged more easily and flexibly (Baldwin and Woodward, 2008). However, standardization on an interface level may also result in rigidity as particular standards follow a path-dependent trajectory and become locked in (Arthur, 1989; David, 1985).

Table 2 shows our scoring model mapping these two dimensions to an integration weight. This score is then assigned to the directed link, describing the extent to which system *B* depends on system *A*.

		Closeness of integration (a)	
		Loosely coupled	Close
Complexity of integration (b)	Non-complex	1	5
	Complex	5	10

Table 2. Scoring model to assign integration weights to links in an architecture network

3.2 Network-Analytic Visualizations and Weighted Measures for Integration

We conceptualize an IS architecture as a network of *N* nodes and *g* links. In this network (*N*,*g*), a node designates a logical IT application, such as an SAP CRM customer management system or an Excel-based controlling cockpit. Depending on the organizational context, an enterprise system such as the SAP business suite falls apart into a set of nodes, such as financials, controlling, and materials management. Those nodes may be coupled together in close and complex ways, which is expressed by the links any node has to other nodes in the architecture network (Dreyfus and Iyer, 2008).

The network (*N*,*g*) can be represented by an adjacency matrix, which is integer-valued and potentially asymmetric. Integers ensure that links can have assigned more than two values (i.e. not only binary existence). An asymmetric adjacency matrix accounts for the directionality of links. Accordingly, we portray an IS architectures as a weighted and directed graph (refer to Barrat et al., 2004 for a general overview).

As a valuable initial vantage point to measure change complexity we draw on centrality measures that work on the level of individual nodes. Centrality measures are useful to identify critical systems in an IS architecture as they allow to identify influential nodes in a network. In their paper, Dreyfus and Iyer (2008) found that "[a]pplications with high positional value may be important because they influence many other applications." Following suit, we focus on finding metrics for the influence of applications on others. We begin by turning to simple, unweighted measures of centrality and we enrich them step-by-step by bringing in more complex, weighted measurements.

In network analysis, a variety of centrality measures has been proposed. We focus on three of them, which are degree, betweenness, and closeness centrality. We chose them by two means. Firstly, they are broadly applied in many areas of application. Secondly, for these three measures generalizations exist from the unweighted original measure to a weighted version.

To begin, degree centrality captures the importance of an actor on the micro level most easily and comprehensively. In an undirected network, a node's degree centrality $d_i(g)$ is simply written as the sum of the links from node i to its direct neighbours $j_1, ..., j_n$. Often times, degree centrality is normalized by dividing it by (n – 1), so that it ranges from 0 to 1. By n, we refer to the number of nodes in the network. Alongside Fuerstenau and Rothe (2014), we believe that degree centrality is a comprehensible measure that could support attainable and realistic decisions in IS architectures, because it straightforwardly counts the number of integration linkages.

The directed version of degree centrality accounts for situations in which $ij \neq ji$ by discerning a node's *in-degree*—the number of incoming links—and its *out-degree*—its number of outgoing links. As we are interested in how many systems depend on a particular system, a node's out-degree is an important baseline for our approach.

The sum of the weights assigned to all linkages of an application indicates its total integration weight. In network terminology, one refers to this number as the *weighted out-degree* (Jackson, 2008). The *weighted out-degree* extends the classical notion of centrality by taking into account the weight that has been assigned to any link. The weighted out-degree $C_D^w(i)$ can be written as the sum of those weights. It is formalized as follows (Opsahl et al., 2010):

$$C_D^w(i) = \sum_{j \text{ } in \text{ } N} w_{ij} \tag{1}$$

where w is the weighted adjacency matrix, in which w_{ij} is > 0 if node i is connected to all other nodes j, and the value represents the weight of the link.

However, degree centrality is limited as it only takes into account a node's total involvement in the network, and not its position in the network. We thus introduce betweenness and closeness centrality as alternative means to measure the criticality of individual nodes in a network. Closeness centrality refers to the relative distance from a particular node i to all other nodes $j_1, ..., j_n$ in the network. It thus captures the proximity from one node to any other node. Closeness can be written as follows (Jackson, 2008):

$$C_C \ (i) = \frac{n-1}{\sum_{j \text{ } in \text{ } N} l(i,j)} \tag{2}$$

where $l(i,j)$ refers to the length of the path from node i to the other nodes $j_1, ..., j_n$ in the network. As the scale is normalized by (n – 1), twice as far means half as close. The maximum value of closeness centrality is 1 for the centre of a star as any other node in the network can be reached via one step. Accordingly, weighted closeness centrality C_C^w can be formalized as follows (Opsahl et al., 2010):

$$C_C^{w\alpha}(i) = \left[\sum_{j \text{ } in \text{ } N} l^{w\alpha}(i,j)\right]^{-1} \tag{3}$$

where $l^{w\alpha}$ is the weighted path length from i to j, w is the weighted adjacency matrix, and α is a tuning parameter. For $\alpha < 1$, a shorter path composed of weak ties (e.g. *AB*) is favoured over a longer path with strong ties (e.g. *ADEB*). Conversely, $\alpha > 1$ features strong links with more intermediaries. Within IS architectures that are suspected to feature core-periphery structures (Baldwin et al., 2014), closeness centrality describes well whether a particular node is located in the centre group of strongly integrated systems or in the periphery.

Additionally, betweenness centrality captures well an actor's role as intermediary, boundary spanner, or gatekeeper (Freeman, 1977). In contrast to degree and closeness, betweenness centrality takes into account the overall positioning of a node in the network. In particular, it captures the number of shortest paths running through node i from any other node $j_1,\ldots,j_n$. Following Jackson (2008), betweenness centrality is defined as

$$C_B \ (i) = \sum_{i,k \neq j} \frac{P_k(i,j)/P(i,j)}{(n-1)(n-2)/2} \tag{4}$$

where $P_k(i,j)$ is the number of shortest paths from node i to node j running through node k and $P(i,j)$ is the overall number of shortest paths from i to j. The term under the fraction line normalizes this value. Betweenness centrality is 0 if k doesn't lie on any shortest path; it is 1 if k occupies a position on any shortest path. Weighted betweenness centrality C_B^w takes into account the weights along these paths (as indicated by enriching $P_{(k)}$ with the tuning parameter α). Drawing on Dijkstra's shortest path algorithm, weighted betweenness centrality can be written as follows (Opsahl et al., 2010):

$$C_B^{w\alpha}(i) = \frac{P_k^{w\alpha}(i,j)}{P^{w\alpha}(i,j)} \quad (5)$$

Altogether, we believe that the weighted out-degree serves as a useful starting point to tap into the direct dependency of other systems on an individual system in an IS architecture. As betweenness and closeness centrality overlap (i.e. both measures capture a node's positional importance in a network), we focus on the former (betweenness centrality) in our further analysis.

3.3 Procedural Model

Figure 2 shows the procedure model that comprises the steps to applying the method. We turn to a brief discussion of each of these steps.

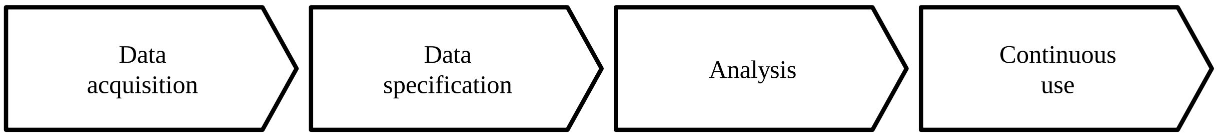

Figure 2. Procedural model to weight integration qualities and to assess change complexity

In the **data acquisition**, firstly an overview of the information systems is created. Thereby, initially data from various sources, such as deployment systems and EA tools, is collected and reviewed. Additionally, interviews should be used to verify and complete the data. They should be conducted with both domain and technical experts. To support a systematic screening of the organization regarding the applications, one may turn to the business process structure as a starting point (Fuerstenau and Rothe, 2014). This step results in a *list of applications*. Based thereupon, data on interfaces must be collected, drawing on the expertise of operations specialists and IT architects. This step produces an *edge list*, a list of all integrations among applications. Consistent with Lagerström et al. (2014), one should thereby consider the directionality of each integration. The *lists of applications* and the *edge list* are then merged and transformed into an *adjacency matrix*. The final result of this work package is a directed graph that includes the information systems (nodes) and integrations (directed links). The data acquisition should consist of several iterations in which the network visualizations are continuously refined.

In the **specification phase**, the collected data is then processed by specifying the quality of the integration. In particular, one needs to apply the scoring model as presented in section 3.1. To gain insights into the quality of the integration, we believe it is useful to triangulate data from expert interviews, especially to identify the closeness of the integration, with the documentation of the interface (i.e. to assess the complexity). As integrations will involve highly specific and tacit knowledge, it is often critical to identify the qualified humans with technical and/or domain expertise.

In the **analysis phase**, the key visualizations and measures are calculated and interpreted. In this phase it is particularly important that the user of the method is equipped with business analytics and network analysis skills. An essential part of the analysis is also to validate the results with experts.

To ensure its **continuous use**, the data collected during the project must be integrated into the enterprise's architecture management (EAM) initiative. A maintenance procedure is useful to incorporate changes to the integration linkages quickly and to proliferate the benefit of the network visualization.

4 ProductionCo: Method Demonstration and Evaluation

Within the next section, we apply our approach at the case site of *ProductionCo*. We will demonstrate its usefulness to support the company's migration from one enterprise system to another.

4.1 The Case Site: Research Context and Data

ProductionCo is a small and centralized production company with about 180 employees. While it is based at one production site in Germany, it sells its products (i.e. for the optical industry) on a global scale. The product range includes the manufacture of products and services such as development, storage, and approval. The company is currently going through a major transition of its application landscape. The company's legacy enterprise system *Sage*—a small and medium enterprise solution—is out-phased and replaced by an integrated, demand-based ERP, *abas*. Accounting for the company's growing demands, functional gaps are closed by complementary standard software (e.g. MES and PLM). Thereby, the company builds in parts on the technology stack of the new ERP vendor.

To mitigate risks with regards to the transformation, the company's main focus in applying the method was to develop a more grounded understanding of the change complexity involved in the switch. Since the functional coverage increases with the new system, the company assumed a higher level of dependence from the new ERP system, as the company is about to change all existing systems into an integrated and validated solution. Several systems—i.e. the product lifecycle management (PLM) system *Teamcenter*, the business intelligence (BI) system *Jedox*, and the financial accounting module of *abas*—have already been transformed. Other areas such as order management will follow. In order to manage the migration while maintaining operations, several additional solutions have been developed, e.g. an integration between *abas* and *Sage* and a master data management application.

The first step of our method evaluation (data acquisition) took about four weeks in 2014 and involved mainly key users and the IT department. The IT department delivered data for the major applications (PLM, ERP, and MES). Another application that is important for our analysis is the *Active Directory (AD)*, which controls the user access to the standard software applications, such as the PLM system. As a means to gather the remaining data for our approach, we conducted several expert interviews. In part, this was motivated by the need to appropriate the documentation of the transformation project, i.e. with respect to interface descriptions and spreadsheet solutions. To complete our records, we turned to key users from production planning, purchasing, and development. An evaluation was made based on a first network visualization. Thereupon, the business unit and IT department staff recognized numerous missing integration linkages, some of which had not been documented previously. In comparison, significantly fewer integration linkages' documentations for the new ERP system were missing. When collecting the data, we took into account the direction of each integration linkage by specifying whether system *A* depends on system *B* (and/or, vice versa).

In the second phase (specification), we scored each integration linkage with respect to its complexity and closeness. Table 3 shows examples for complex and non-complex integrations. As put forward in section 3.1, we used the number of fields and message types for assessing the complexity of an interface. For instance, *MES-Hydra* features a standardized legacy interface that is served by *abas*, the new ERP. The integration of ERP and MES is complex. The complexity of the integration is driven by a lack of standardization on the ERP-side (i.e. self-developed, proprietary interface), usage of rigid legacy integration technology (flat files), and the scope of the integration (ten message types and more than 1,000 fields). In contrast, the PLM and *AD* integration is non-complex as it incorporates only a limited number of message types (i.e. user authentication and password validation) and takes into account only few fields. To arrive at this conclusion, we evaluated each integration linkage with regards to defined threshold values and drew on the existing interface documentation where applicable.

Provider	Consumer	Number of fields	Number of message types	**Complexity of integration**
MES	ERP	High (1,000)	High (10)	**Complex**
AD	PLM	Low (10)	Low (2)	**Non-complex**

Table 3. Example for the assessment of integration complexity

Table 4 illustrates the evaluation of integration closeness. For instance, the ERP is highly-dependent on the availability of the MES as shipment date calculation in the ERP depends on up-to-date information from the MES. The ERP furthermore trusts the data from the MES blindly. As the interface is standardized on the MES side, the ERP needs to perform various transformations to process the data. Thus, the ERP has more knowledge on interface structure and semantics than absolutely necessary. The strength of the dependency as well as the high level of trust (i.e. dependence on preconditions) result in a close integration. For the PLM integration with the *Active Directory,* only the necessary fields are exchanged (e.g. user name and password), which results in limited knowledge on the interface semantics. However, the high dependency on the availability and the high level of trust still result in a close integration. The decision whether an interface was classified as loose or close was discussed with a small group of IT managers and system administrators.

Provider	Consumer	Dependency on the availability	Confidence	Knowledge	**Closeness of integration**
MES	ERP	High	Yes	More	**Close**
AD	PLM	High	Yes	Less	**Close**

Table 4. Example for the assessment of integration closeness

For the analysis (step 3), we drew on *Gephi* and *R* (*igraph*/*tnet*), two widely applied network analysis tools. We visualized the weighted and directed graph based on the adjacency matrix. For instance, we assigned a high integration weight (10) to the incoming link to the ERP from the MES and a medium weight (5) to the incoming link to the PLM from the *Active Directory*. We then calculated centrality measures to identify critical systems within the architecture. We present and interpret results in the next section.

4.2 Results

Figure 3 shows a plot of the network that comprises ProductionCo's IS architecture. Within the network, each node represents an information system, such as *abas*—a comprehensive enterprise system—or an Excel-based label printing application. We filtered the network for its largest connected group (giant component), as several isolated nodes had no integration in the IS architecture at all. The remaining network encompasses 42 nodes (information systems) and 70 links (integration linkages). These links are directed. Depending on whether the integration is bi-directional, the links will have arrows on both ends or not. The thickness of the lines represents the scoring value from our assessment of the complexity and closeness of the integration. As can be seen from the figure, we use three grades according to the scoring value as detailed in Table 2.

The color of a node designates to which of three clusters, identified by a method called "modularity clustering", the application belongs. From a contextual point of view, these automatically identified clusters overlap widely with three areas that have a meaningful interpretation in ProductionCo's IS architecture: (1) the *Sage* cluster (as-is), (2) the *Active Directory* cluster (infrastructure), and (3) the *abas* cluster (should-be).

To begin, we turn attention to the axis of PLM-ERP-MES, as it is the backdrop of the future *should-be* landscape. These three complex systems (PLM-Teamcenter, ERP-abas, and MES-Hydra) will provide large parts of the required business functionality. In particular, the trio maps the entire information flow from PLM (product idea, development, and revision), via ERP (materials management, purchasing, sales, logistics, production planning) to MES (manufacturing control, detail planning, quality management, capturing of production data) for all produced products. In addition, the production planning system (PPS) ensures, in close interaction with the ERP, the planning in the light of limited resources. Shipment dates from the PPS are processed in the ERP and further transferred to the MES.

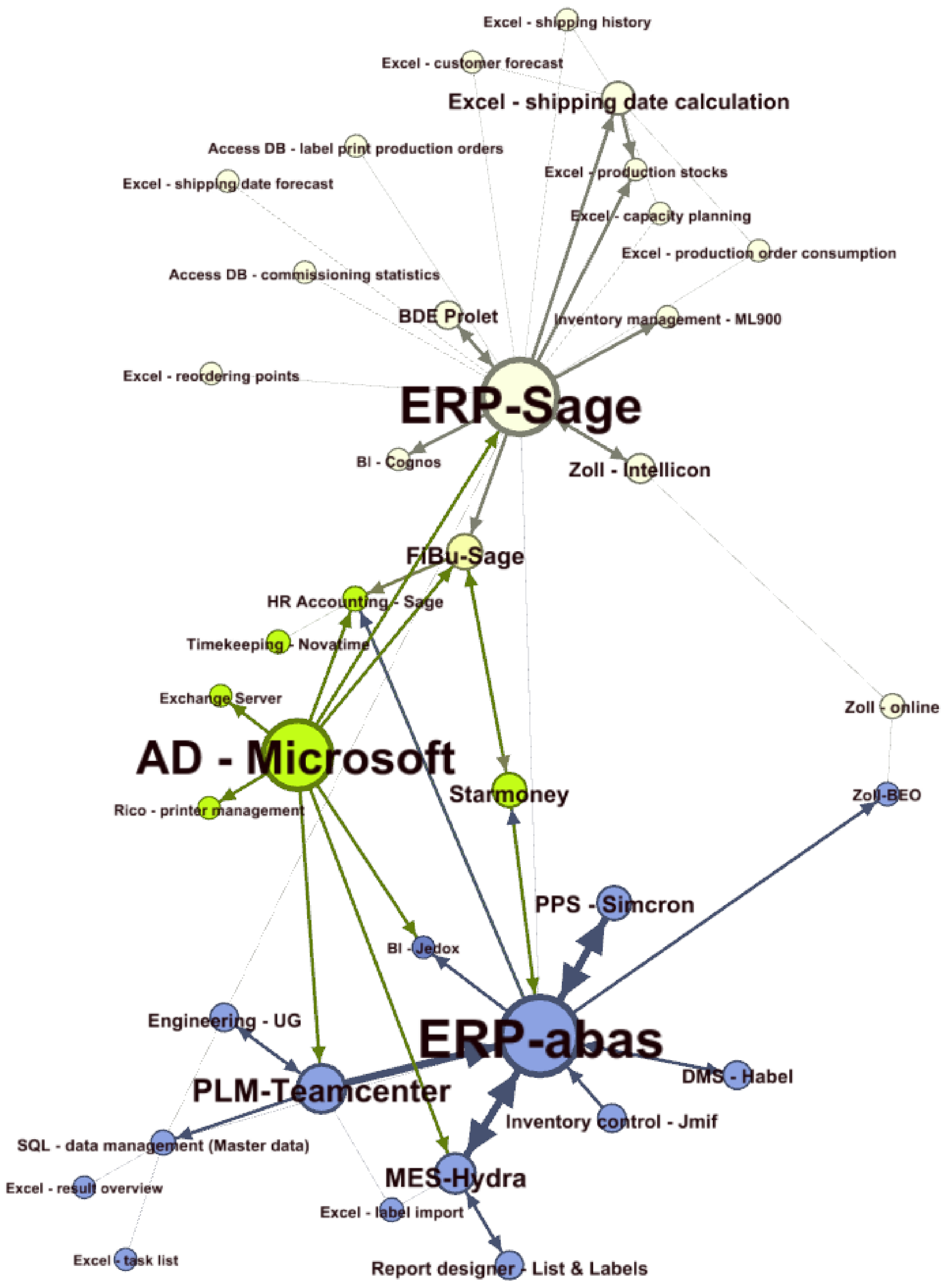
Excel - shipping history
Excel - customer forecast
Excel - shipping date calculation
Access DB - label print production orders
Excel - shipping date forecast
Excel - production stocks
Excel - capacity planning
Excel - production order consumption
Access DB - commissioning statistics
BDE Prolet
Inventory management - ML900
Excel - reordering points
ERP-Sage
BI - Cognos
Zoll - Intellicon
FIBu-Sage
HR Accounting - Sage
Timekeeping - Novatime
Exchange Server
Zoll - online
AD - Microsoft
Starmoney
Rico - printer management
Zoll-BEO
PPS - Simcron
BI - Jedox
Engineering - UG
ERP-abas
PLM-Teamcenter
DMS - Habel
Inventory control - Jmif
SQL - data management (Master data)
MES-Hydra
Excel - result overview
Excel - label import
Excel - task list
Report designer - List & Labels

Figure 3. ProductionCo‘s IS architecture

Subsequently, we discuss the centrality measures suggested in section 3.2. In doing so, we focus on six important applications within the IS architecture (PLM, ERP-Sage, ERP-abas, MES, PPS, and AD) as they are illustrative for the kinds of insights we can gain with our approach.

Table 5 shows results. Within the table, the columns designate the applications and the rows show important centrality measures. To enable a better comparison of the state-of-the-art (unweighted measures) with our approach, we have always included the unweighted and weighted measure.

	PLM-Teamcenter	ERP Sage	ERP abas	MES Hydra	PPS Simcron	Active Directory
Out-degree (unweighted)	5	17	10	2	1	8
Out-degree (weighted)	22	45	48	15	10	40
Betweenness (unweighted)	36	315	320	74	0	0
Betweenness (weighted)	39	304	330	76	0	0

Table 5. Centrality measures for core enterprise and collaboration systems at ProductionCo

Turning to the PPS system and its integration linkages, we see that it provides only one outgoing interface, in network terminology its out-degree is 1. However, this one link is essential for the ERP system, as no statement on delivery dates would be possible without detailed planning in the PPS. In addition, it would be impossible to create a useful ordering of the production program in the MES. Thus, an unweighted approach would largely misrepresent the criticality of this application. From the second row in Table 5, we see that weighted measures emphasize the importance of the PPS.

Turning to the MES, as can be seen from Figure 3, the *MES-Hydra* provides two outgoing interfaces, one of which is important for our argument, as discussed above (cf. Table 3-4). As can be seen from the figure, a thick line between *Hydra* and *abas* indicates a close and complex integration. In Table 5, we see that *Hydra's* high criticality is pronounced by the weighted centrality measure.

A comparison of weighted and unweighted out-degree for the PLM illustrates that its criticality for the company would be grossly misrepresented by unweighted measures. As can be seen in Table 5, its out-degree of 5 is greatly increased by the weighting, resulting in a weighted out-degree of 22. A main reason for this proliferation is the high functional dependency of the ERP system *abas* on the PLM. In particular, master data transferred from the PLM is of critical importance for the ERP and incorporates several message types: new product creation, product changes, updates, blockings, and deletions. Thereby, the ERP system trusts in the PLM's output as it performs particular functions (i.e. approvals, inventory blockings) directly without cross-checks. To highlight the importance of the integration, we refer to the fact that no article could be created in the ERP systems without the integration.

We next turn to a brief comparison of *Sage* and *abas*, as this raises a question that is of critical importance for the company: Will a new enterprise system result in reduced levels of change complexity? Our results in Table 5 indicate that the reliance on the new ERP will remain almost on the same level. While *Sage* offered limited base functionality, change complexity was driven by the sheer number of interfaces (out-degree = 45). In contrast, *abas* draws on fewer but more intense integrations.

We turn to *Active Directory*, which highlights another useful finding. In contrast to the aforementioned applications that are involved in complex relationships, the integration of an LDAP service to authenticate login data from a user is non-complex. However, we can see from Table 5 that it is still critical. Firstly, the mere number of services relying on the authentication results in a considerable change complexity (e.g. if one considers a switch to a Linux directory). Furthermore, the analysis shows that the directory is closely coupled with numerous applications, as a system failure would propagate to the adjacent systems and would cut down the accessibility of those applications.

Regarding betweenness centrality, Table 5 shows that the *PPS Simcron* and the *Active Directory* do not lie on any of the shortest paths. Thus, both applications have a betweenness centrality of 0. These systems cannot be described as gateways. In contrast, *Sage* (315) and *abas* (320) are not only important with respect to the number of integrations but also as important gateways. These systems bridge different parts of the landscape (e.g. MES-ERP-PLM). Regarding weighted betweenness, we do not find a qualitative difference between weighted and unweighted measures when using a tuning parameter that takes into full account the weight of each link ($\alpha = 1.0$).

Taken together, our approach supports visualizing dependencies in an application landscape in a simple yet efficient manner. Data collection and assessment could be performed with little effort. The suggested measures have meaningful interpretations in the context of a real migration planning project, which increased the participants' acceptance of the approach. The company has indicated interest in using the measures as a base for effort planning in the migration project. As a result, demands can be estimated and selected on a more grounded basis. In addition, our approach has been useful to support migration planning, as it tapped into undetected potential threats, such as the variety and diversity of add-ons and extensions to the *Sage* enterprise system.

5 Discussion and Conclusion

Our aim in this article has been to suggest a method that supports IS architecture planning (i.e. migration projects) by assessing the impact of integrations and their qualities on change complexity. As enterprise systems' value is more and more dependent on their integration potential, our method maps a company's IS architecture to a weighted and directed graph in which nodes represent applications and links designate qualified integrations. Our main argument is that critical applications with impact on change complexity are better identified by taking into account integration qualities. This is achieved by assigning integration weights to the network's links.

Our procedure uses two qualifications of integrations: closeness and complexity. By scoring the integrations and assigning weights to the links, one can observe that some applications importance becomes reinforced while others decline in importance. This allows for a more nuanced portrayal of "key players" than possible otherwise. Consider, for instance, the close cohesion of MES, PLM, and ERP.

5.1 Discussion

We turn to a discussion of the suspected change complexity "before" and "after" ProductionCo's transition to the new enterprise system. To begin, we remark that such comparison must be treated with reservation as the functional scope of the new system is significantly larger; in particular, many of the Excel solutions are already integrated in the new system's standard. Nonetheless, the following tentative conclusions can be drawn from applying the method at the case site.

Firstly, we emphasize that many small spreadsheet solutions can be a significant driver of change complexity. Think, for instance, of the Sage-ERP system and its numerous Excel- and Access-based extensions. Secondly, we find that a migration of an enterprise system will not necessarily results in a decrease of change complexity. To illustrate that point, refer, for instance, to the weighted out-degree of the new enterprise system *abas* (48) versus the old system *Sage* (45). One may suspect that the weighted out-degree of the new enterprise system would decrease as Excel-based and proprietary solutions are merged into a new system. While the number of integrations may be reduced, new possibilities to interconnect may open up ("open ends") and a closer and more complex integration of complementary systems may thus result in an increase in overall integration strengths. In connection to that point, we find that an integrated technology stack is often a main driver of change complexity as various information systems must be treated as a set that may only be changed or replaced in association. Consider, for instance, the abas-ERP system that enables an integrated process flow bridging several applications. A well-aligned technology stack has been installed, e.g. ERP and fitting PPS. Replacing on of the applications in this set (e.g. by an SAP ERP) would require setting up a different integration architecture almost from sketch. Altogether, the weighted out-degree has served as a useful starting point to illuminate different effects of numerous, close, and diverse integrations on change complexity.

Regarding betweenness centrality, we conclude that (weighted) betweenness is a valuable extension to the out-degree. We emphasize a phenomenon which we refer to as “data hopping”. As mentioned earlier, we often stated a direct dependency of a system *B* from another system *A*. However, information is often passed over several stages in the architecture (e.g. PLM → ERP → MES). This results in an indirect dependency in which also the MES depends on the PLM. In parts, betweenness centrality captures this phenomenon by incorporating how many shortest paths run through a particular node.

5.2 Limitations

Before sketching broader implications for a theory of enterprise transformation and practice, we emphasize conditions that restrict the perspective we have presented. Firstly, we have focussed on a one-time assessment of the IS architecture without visualising explicitly the *as-is* and *should-be* states of the architecture. Relatedly, another extension to the approach presented here would focus on tracking an architecture’s development over time. Secondly, we have focused on how integrations affect change complexity while lacking a consideration of the properties of systems themselves. Obviously, an application such as PLM-Teamcenter has an inherent complexity, which has been out of scope in our approach. Thirdly, we have not yet validated the indicators that were used to measure integration qualities as our main focus in this paper has been a first exploration of weighting integration qualities.

5.3 Future Directions

In addition to demonstrating the usefulness of our approach in other settings, especially those in which more dispersed IS structures are expected, we see three particularly promising ways to proceed further. Firstly, one can employ our method to compare *as-is* and *to-be* architectures. What implications do transformation efforts have on change complexity? Secondly, one can build on our approach to go beyond the level of individual systems and to identify and compare the change complexity among different clusters, segments of enterprise systems, or the entire architecture. For instance, we draw on the example of the migration from *Sage* to *abas*, which illustrates the value of comparing clusters of enterprise systems. Thirdly, the weighted out-degree misrepresents the criticality of systems that pass information over *n* steps as it limits attention to direct neighbours. Consider, for instance, the axis of PLM, ERP, and MES. Assigning a high weight (10) to the PLM’s single close and complex integration lacks consideration of the reinforced importance from further integration with the MES on the next level. However, a measure tapping into such *n*-tier propagation requires context-dependent knowledge as it must qualify the objects running through particular paths. Relatedly, it is useful to take a close look into what we referred to as “data hopping” as the assumption of betweenness centrality that information takes shortest paths may be a too strong in architectural settings.

5.4 Theoretical and Practical Implications

Turning in conclusion to further implications for theory and practice, our approach contributes to a recent and fruitful stream of EA research that conceptualizes IS architectures as networks (Aier and Winter, 2009; Fuerstenau and Rothe, 2014). Our contribution to quantitative EA approaches, such as Lagerström et al. (2014), is as follows. We conceptualize IS architectures as weighted networks and apply weighted measurements to them. Our results advocate for the crucial role of taking into account integration qualities in determining the change complexity of an IS architecture. From a practical point of view, we suspect that our approach equips IT managers and architects with a valuable method when assessing risks in enterprise system migrations. In addition, we believe it could be used to support other enterprise transformation use cases such as mergers and acquisitions, carve-outs, restructurings, consolidations and system decommissioning. A management dashboard, integrated into EAM and strategic IT planning, can build upon our approach.

References

Aier, S. and Schönherr, M. (2006), “Status quo geschäftsprozessorientierter Architekturintegration”, *Wirtschaftsinformatik*, Vol. 48 No. 3, pp. 188–197.

Aier, S. and Winter, R. (2009), “Virtual Decoupling for IT/Business Alignment -- Conceptual Foundations, Architecture Design and Implementation Example”, *Business & Information Systems Engineering*, Vol. 1 No. 2, pp. 150–163.

Arthur, W.B. (1989), “Competing Technologies, Increasing Returns, and Lock-In by Historical Events”, *The Economic Journal*, Vol. 99 No. 394, pp. 116–131.

Baldwin, C., MacCormack, A. and Rusnak, J. (2014), *Hidden Structure : Using Network Methods to Map System Architecture* (No. 13-093), HBS Working Paper. Boston, Mass.

Baldwin, C.Y. and Woodward, J. (2008), *The Architecture of Platforms: A Unified View* (No. 09-034), HBS Working Paper Series. Boston, Mass, doi:10.2139/ssrn.1265155.

Barrat, A., Barthélemy, M., Pastor-Satorras, R. and Vespignani, A. (2004), “The architecture of complex weighted networks”, *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 101 No. 11, pp. 3747–3752.

Baumann, A., Engels, G., Hofmann, A., Sauer, S. and Willkomm, J. (2009), “A Holistic Software Engineering Method for Service-Oriented Application Landscape Development”, in Proper, E., Harmsen, F. and Dietz, J. (Eds.), *Advances in Enterprise Engineering II*, Lecture Notes in Business Information Processing, Springer Berlin Heidelberg, Vol. 28, pp. 1–17.

David, P.A. (1985), “Clio and the Economics of QWERTY”, *The American Economic Review*, Vol. 75 No. 2, pp. 332–337.

David, P.A. and Greenstein, S. (1990), “The Economics of Compatibility Standards: An Introduction To Recent Research”, *Economics of Innovation and New Technology*, Vol. 1 No. 1-2, pp. 3–41.

Denning, P.J. (1997), “A new social contract for research”, *Communications of the ACM*, Vol. 40 No. 2, pp. 132–134.

Dreyfus, D. (2009), *Digital Cement: Information System Architecture, Complexity, and Flexibility*, Dissertation, Boston, MA.

Dreyfus, D. and Iyer, B. (2008), “Managing architectural emergence: A conceptual model and simulation”, *Decision Support Systems*, Vol. 46 No. 1, pp. 115–127.

Engels, G., Hess, A., Humm, B., Juwig, O., Lohmann, M. and Richter, J.P. (2008), *Quasar Enterprise: Anwendungslandschaften serviceorientiert gestalten*, Dpunkt-Verl., Heidelberg, 1st ed.

Erl, T. (2006), *Service-oriented architecture: Concepts, technology, and design*, Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, 6th ed.

Freeman, L.C. (1977), “A set of measures of centrality based on betweenness”, *Sociometry*, Vol. 40 No. 1, pp. 35–41.

Fuerstenau, D. and Rothe, H. (2014), “Shadow IT Systems: Discerning the Good and the Evil”, *ECIS 2014 Proceedings*.

Goes, P.B. (2014), “Editor’s Comments: Design Science Research in Top Information Systems Journals”, *MIS Quarterly*, Vol. 38 No. 1, pp. iii-viii.

Gregor, S. and Hevner, A.R. (2013), “Positioning and Presenting Design Science Research for Maximum Impact”, *MIS Quarterly*, Vol. 37 No. 2, pp. 337–355.

Gregor, S. and Jones, D. (2007), “The Anatomy of a Design Theory”, *Journal of the Association for Information Systems*, Vol. 8 No. 5, Article 2.

Hevner, A.R. and Chatterjee, S. (2010), “Design science research in information systems”, in Hevner, A.R. and Chatterjee, S. (Eds.), *Design Research in Information Systems*, Springer, eBook, pp. 9–22.

Hevner, A.R., March, S.T., Park, J. and Ram, S. (2004), “Design science in information systems research”, *MIS Quarterly*, Vol. 28 No. 1, pp. 75–105.

Jackson, M.O. (2008), *Social and Economic Networks*, Princeton Univ. Press, Princeton and NJ.

Kuechler, B. and Vaishnavi, V. (2008), “Theory Development in Design Science Research: Anatomy of a Research Project”, in Vaishnavi, V. and Baskerville, R.L. (Eds.), *Proceedings of the Third International Conference on Design Science Research in Information Systems and Technology*, Atlanta, Georgia, pp. 1–15.

Lagerström, R., Baldwin, C., MacCormack, A. and Aier, S. (2014), “Visualizing and measuring enterprise application architecture: An exploratory telecom case”, *HICSS 2014 Proceedings*, pp. 3847–3856.

Lagerström, R., Baldwin, C., MacCormack, A. and Dreyfus, D. (2013), *Visualizing and Measuring Enterprise Architecture: An Exploratory BioPharma Case* (No. 13-105), HBS Working Paper, Boston, Mass.

Lankhorst, M. (2009), *Enterprise architecture at work: Modelling, communiation and analysis*, Springer, Dordrecht and and New York, 2nd ed.

Matthes, F. (2008), “Softwarekartographie”, *Informatik-Spektrum*, Vol. 31 No. 6, pp. 527–536.

Mertens, P. (2013), *Integrierte Informationsverarbeitung 1: Operative Systeme in der Industrie*, Gabler, Wiesbaden, 18th ed., p. 389.

Opsahl, T., Agneessens, F. and Skvoretz, J. (2010), “Node centrality in weighted networks: Generalizing degree and shortest paths”, *Social Networks*, Vol. 32 No. 3, pp. 245–251.

Peffers, K., Tuunanen, T., Rothenberger, M.A. and Chatterjee, S. (2007), “A Design Science Research Methodology for Information Systems Research”, *Journal of Management Information Systems*, Vol. 24 No. 3, pp. 45–77.

Ross, J.W., Weill, P. and Robertson, D. (2006), *Enterprise architecture as strategy : creating a foundation for business execution*, HBS Press, Boston, Mass.

Schneberger, S.L. and McLean, E.R. (2003), “The complexity cross: implications for practice”, *Communications of the ACM*, Vol. 46 No. 9, pp. 216–225.

Schütz, A., Widjaja, T. and Gregory, R.W. (2013a), “Escape from Winchester Mansion – Toward a Set of Design Principles to Master Complexity in IT Architectures”, *ICIS 2013 Proceedings*, Milan, Italy, pp. 1–19.

Schütz, A., Widjaja, T. and Kaiser, J. (2013b), “Complexity in Enterprise Architectures - Conceptualization and Introduction of a Measure from a System Theoretic Perspective”, *ECIS 2013 Proceedings*.

Winter, R. (2008), “Design science research in Europe”, *European Journal of Information Systems*, Vol. 17 No. 5, pp. 470–475.