This is a near-final preprint of the paper: Sosa, Manuel E., Jürgen Mihm and Tyson R. Browning (2011) "Degree Distribution and Quality in Complex Engineered Systems," *Journal of Mechanical Design*, **133**(10): 101008 (15 pages).

Degree Distribution and Quality in Complex Engineered Systems

Manuel Sosa
Technology and Operations Management Area
INSEAD
Fontainebleau, France
manuel.sosa@insead.edu

Jürgen Mihm
Technology and Operations Management Area
INSEAD
Fontainebleau, France
jurgen.mihm@insead.edu

Tyson R. Browning
Department of Information Systems and Supply Chain Management
Neeley School of Business, Texas Christian University
Fort Worth, Texas, U.S.A.
t.browning@tcu.edu

Abstract

Complex engineered systems tend to have architectures in which a small subset of components exhibits a disproportional number of linkages. Such components are known as *hubs*. This paper examines the degree distribution of systems to identify the presence of hubs and quantify the fraction of hub components. We examine how the presence and fraction of hubs relate to a system's quality. We provide empirical evidence that the presence of hubs in a system's architecture is associated with a low number of defects. Furthermore, we show that complex engineered systems may have an optimal fraction of hub components with respect to system quality. Our results suggest that architects and managers aiming to improve the quality of complex system designs must proactively identify and manage the use of hubs. Our paper provides a data-driven approach for identifying appropriate target levels of hub usage.

Keywords: Complex networks, product architecture, design structure matrix, quality.

We appreciate support from Lattix, Inc., which provided us with the software used to document the architecture of the products and sequence our matrices to calculate some of our control variables. We thank Neeraj Sangal for insightful feedback throughout this research and Jeremy Berry for helpful research assistance. We also want to thank Bhavani Shanker and Aman Neelappa, who have spent months implementing web crawlers to collect raw data and preparing it for analysis. We would also like to thank two anonymous reviewers for their helpful comments that led us to improve the paper.

1 Introduction

This paper considers complex engineered systems as networks of interconnected components. In many such networks, some nodes are more connected than others [1-5]. The most highly connected components are called *hubs*. In this paper, we provide econometric analysis to support the argument that the presence of hubs has a significant effect on system quality (measured by the number of defects in the system). Thus, although hubs typically constitute only a small fraction of a system's components, they provide high-leverage points for designers and managers to design and maintain high-quality systems.

This paper integrates two streams of literature; we build on both the engineering design literature (both from hardware and software domains), which uses product architecture representations to capture systems as networks of interconnected components, and the literature on complex networks, which has focused on uncovering the underlying properties that characterize complex, real-world networks.

The literature on engineering design has characterized complex engineered systems as networks of interdependent components as a way to improve our understanding of the architecture of complex products [6-12]. A common theme in this literature stream is the use of a square matrix (i.e., a design structure matrix (DSM)) to represent the components comprising a system and how these components interconnect [13-15]. In this paper, we also rely on a DSM representation to capture the architecture of the systems in the sample we analyze.

The complex networks literature has primarily focused on uncovering the fundamental properties that govern network topology and dynamics (See [5, 16, 17] for reviews.) Recent studies in various disciplines from Physics to Engineering to Sociology have put considerable attention on finding a set of underlying network properties in complex physical, software, and social systems such as the world-wide web [18, 19], power grid networks [4, 20], scientific collaboration networks [21], and new product development task networks [1-3].

A salient property in complex network studies is the degree distribution. *Degree distribution* is defined by the probability, p_k, that a node in a network has k connections with other nodes in the network [5]. By examining the degree distribution, previous research has found that many complex networks exhibit a scale-free property, wherein a few nodes are heavily connected while most are not [3, 18, 19]. Examining the degree distribution of complex networks also led to the notion of hubs, the relatively small subset of components that are most highly connected (see [5] for review). Albert et al. [22] suggested that a pronounced hub-periphery structure can account for the resilience of scale-free networks to random failures. Since any randomly failing node probably has small degree, it is expendable. On the other hand, such networks are vulnerable to deliberate attacks on the hubs. These effects have been confirmed numerically (e.g., [22]) and analytically (e.g., [23]). In addition to resilience, Braha and Bar-Yam [3] explored dynamic aspects of system quality in the context of product development organizations. They showed that complex development networks are asymmetric in the degree distribution of incoming and outgoing links and confirmed, via simulation, the benefits of highly skewed degree distributions for the resolution speed of interdependent task networks. Thus, the presence of hubs can improve system performance.

We build on these literature streams to empirically show that there is an optimal presence of hubs which minimizes the number of defects in the system. First, we test whether the mere presence of hubs relates to lower levels of defects. Then, we test whether there is an optimal fraction of hub components that minimizes the levels of defects in a system. Our study provides an econometric-based approach for design managers to systematically evaluate the link between the fraction of hub components and system quality in their own contexts.

In the next section, we illustrate how to identify hubs through examination of the degree distribution of three mechanical systems examined in previous engineering studies. In Section 3, we build our theoretical argument to formulate two hypotheses; in Section 4, we conduct an econometric analysis based on a large, longitudinal data set of (software) systems. We conclude the paper by discussing the implications of our results for researchers, designers, and managers.

2 Degree distribution and hubs in complex engineered systems

In this section, we discuss how to examine the degree distribution of a system to formally identify "hubs". This paper builds upon the terminology and toolset developed in the context of the analysis of complex networks to establish a formal criterion for identifying the hub components of a system [1-4, 17].

In network and graph theoretic terms, the degree of a node in a network (or graph) is defined as the number of edges incident with it [24]. Hence, the degree of a system's component counts the number of other components (within the system) it connects to. As a result, the degree distribution captures how connections are distributed among the system's components (e.g., [1-4]).

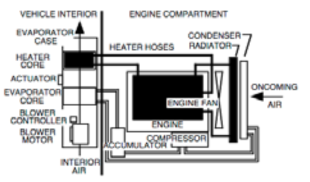
Informally, a hub is a component that is disproportionally more connected (i.e., that have a disproportionally higher degree) than most of the other components in the system [16, p. 169]. While the notion of a hub is intuitive, a formal criterion to identifying hubs is non-trivial because it requires drawing the line that separates the most connected components (hubs) from the less connected ones [25-27]. Next, we discuss three alternative ways of identifying the presence of hubs in a system.

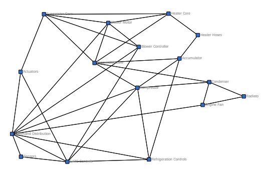
2.1 Identifying hubs based on the cumulative degree distribution

In line with previous work on complex networks (e.g., [3, 5]), we examine the degree distribution of a system to identify the presence of hubs. Based on an automobile climate control system [6], Figure 1 illustrates how to capture and examine the degree distribution of a system to identify its hub components.

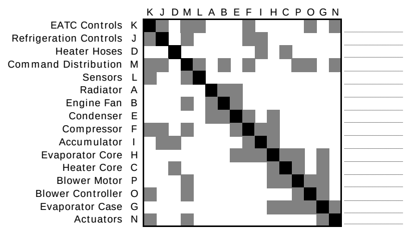
Starting with the product itself, the basis for identifying a hub is first to represent the product as a DSM by identifying its components and the linkages between them (see Step 1 of Figure 1). The automotive climate control system consists of 16 components and 68 interfaces ($N = 16, K = 68$). In a second step (see Step 2 of Figure 1), by first summing over the rows (or columns) of the DSM and then building the frequency distribution over the resulting degree vector ($\boldsymbol{k}$), we obtain the *degree distribution*. It shows the count of components that have a certain absolute number of interfaces (i.e., component degree, k). Specifically, the system in Figure 1 has, for example, three components with a degree of 2, two components with a degree of 3, and one component with a degree of 8. Clearly, the degree distribution depends on the size of the system. As a consequence, a definition of a hub based on the degree distribution cannot be universally applicable. So as to allow for a widely applicable criterion to identify hubs, it is imperative to normalize both the horizontal and vertical dimensions of the distribution. Thus, in step 3 the *normalized degree distribution* is obtained by expressing each value (whether it be the x-axis or the y-axis) as a percentage of its respective system maximum [28]. The horizontal axis is normalized by dividing the actual component degree by its theoretical maximum ($D = k/(N-1)$), and the vertical dimension is normalized by dividing each count of components by the total number of components in the system (N). Hence, all bars sum to one, yet the shape of the distribution is unaffected by the normalization method.

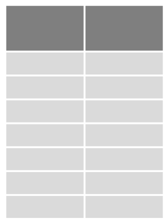
Since a formal method for quantifying the number of hubs has to contrast the subset of components that are highly connected against the other components that are only marginally connected, we split the normalized degree distribution shown in Step 4 of Figure 1 into two groups of components so that there is a highly connected group, the potential hub components, and a less connected group. Because we can split the normalized degree distribution arbitrarily at any point along its horizontal axis, there are as many alternative ways to define what potentially constitutes a system's set of hubs as there are valid splitting points. To illustrate this fact, consider Step 4 of Figure 1. Splitting the normalized degree distribution at $D = 0.40$ yields a percentage of hubs of $Q_{0.40} = 0.25$. That is, the top 25% of most connected components have a normalized degree of 0.40 or higher. For the split at $D = 0.33$, the percentage of hubs is $Q_{0.33} = 0.38$. We can repeat this procedure of splitting for any value of D. Doing so, in the last line of Figure 1 we plot the cumulative distribution (Q_D for each D, in steps of D of 0.05). Q_D measures the fraction of (potential) hubs with normalized degree D.





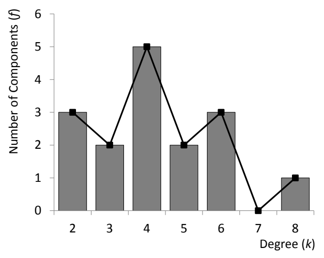
1. Identify components
2. Identify dependency structure between components
3. Represent in a DSM form

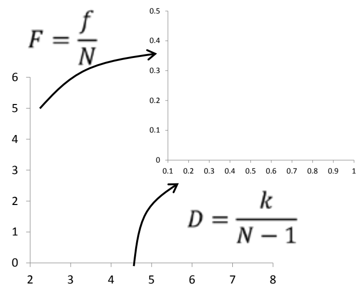




1. Determine component degree by summing marks across rows (or) columns in DSM
2. For each degree, count the number of components
3. Plot resulting frequency distribution

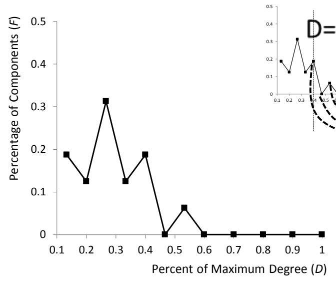
Step 3: Degree Distribution to Normalized Degree Distribution

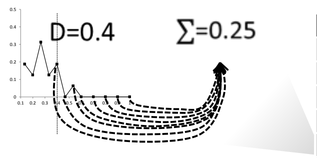




1. Normalize x-axis by dividing it by the theoretical maximum degree (*N*-1)
2. Normalize y-axis by deviding it by the total number of components
3. Plot the normalized degree distribution

Step 4: Normalized Degree Distribution to Cumulative Distribution

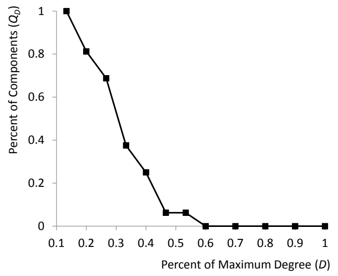




D	Q_D
0.13	1.00
0.20	0.81
0.27	0.69
0.33	0.38
0.40	0.25
0.47	0.06
0.53	0.06
0.60	0.00
0.70	0.00
0.80	0.00
0.90	0.00
1.00	0.00

1. For each normalized degree (*D*) sum up the percentage of components equal to or larger than the degree (Q_D)
2. Plot the normalized cumulative distribution

Cumulative Distribution



Sources: For schematic and DSM of automotive climate control system see [6]

Figure 1: Schematic, product DSM, and degree distributions of an automotive climate control system

From the cumulative distribution plot shown in the bottom line of Figure 1 it is clear that, when considering $D \leq 0.30$, the majority of components (i.e., more than 50% of the components) are in the group with high degree. For instance, considering $D = 0.27$ will result in $Q_{0.27} = 0.69 > 0.50$. That is, the top 69% of the components have a normalized degree of 0.27 or more. In such a split, more than 50% of the components in the system would be considered hubs, which would counteract the intuition that hubs comprise a minority of the components.[1] This suggests that we need to establish a criterion to define an acceptable lower bound for D that can be used to identify hubs.[2]

A lower bound for D can be defined in several ways. A first approach could be based on identifying a significant change in regime of the distribution. Such an approach is used in social networks to identify core-periphery structures [25] and in engineering design to identify modular and integrative subsystems [27]. The drawback with such an approach is that for certain degree distributions, such as the power law distribution, changes in regime may not exist [4]. A second approach would involve imposing a fixed limit, which in itself can be a function of the mean degree in the network (e.g., hubs must have a degree x times larger than the mean degree in the network) or can be externally defined (e.g, Braha and Bar-Yam [26] defined hubs as the top 1000, or 1.7%, most connected nodes in the networks they analyzed). A naïve approach would advocate that it is counter-intuitive to accept that the majority of components could be hub components, so an acceptable D would be one that guarantees Q_D < 50%. A Pareto law-based approach (e.g., [29]) with its focus on the 20% most significant elements of a distribution would suggest that, at most, 20% of components could be hubs. A radically agnostic view would not impose any lower bound on D *a priori* and leave such a decision to the decision maker for the particular situation.

We deliberately do not define a hub based on any specific (arbitrary) D; rather, we allow the decision maker to choose a lower bound of D (D_o). We only require that D be chosen such that the fraction of hub

[1] Although the intuitive notion of hubs suggests that they constitute a set that is both highly connected and small, the set does not necessarily need to be small. For instance, Strogatz ([5], p. 274) referred to hubs as simply highly connected components, even if they constitute the majority of the components in the system.

[2] Here we assume that a system has, at least, one hub unless its degree distribution gives positive probability weight to only one degree k_o (i.e., all components have the same degree, k_o). In such degree distributions, we cannot identify hubs because we cannot split the degree distribution at any point.

components, Q_D, is smaller than a pre-defined threshold, Q_{Do}. ($D \geq D_o$ such that $Q_D \leq Q_{Do}$.) We use a Pareto-based criterion to identify hubs in our analysis (Q_{Do}= 0.20). Such a threshold ensures that hub components in the systems we analyze always have a normalized degree significantly greater than the mean normalized degree. This is consistent with the notion of right-skewed distributions put forward in the complex network literature [3]. Formally,

Definition *A component is a hub if both of the following conditions hold: (i) its normalized degree, D_i, is greater than or equal to a predefined D ($D_i \geq D$), and (ii) the fraction of components in the system for which $D_i \geq D$,Q_D, is less than or equal to a predefined threshold Q_{Do} ($Q_D \leq Q_{Do}$).*

Our approach for identifying hubs is not specific to the automotive climate control system shown in Figure 1. For comparison, Figure 2 applies this approach to two considerably larger mechanical systems: (a) an automobile climate control system [6], (b) a large commercial aircraft engine [11, 27], and (c) the Mars Pathfinder [30, 31]. These systems are modeled at the level of 16, 54, and 43 components respectively. Although the DSMs of the systems in Figure 2a and 2c are symmetric, which result in identical in-degree and out-degree distributions, the depiction of the aircraft engine (Figure 2b) shows a non-symmetric DSM, because it acknowledges that some components depend on others but not vice-versa (see [11, 27] for details). In such a case we could plot the degree distribution corresponding to the incoming and outgoing dependencies of each component. For simplicity we plot the in-degree distribution of the engine. Yet, considering the asymmetry of in- and out- degree distributions is important, because such asymmetric distributions exist in complex engineered systems, and because such asymmetry may have a significant impact on system quality [1-3].

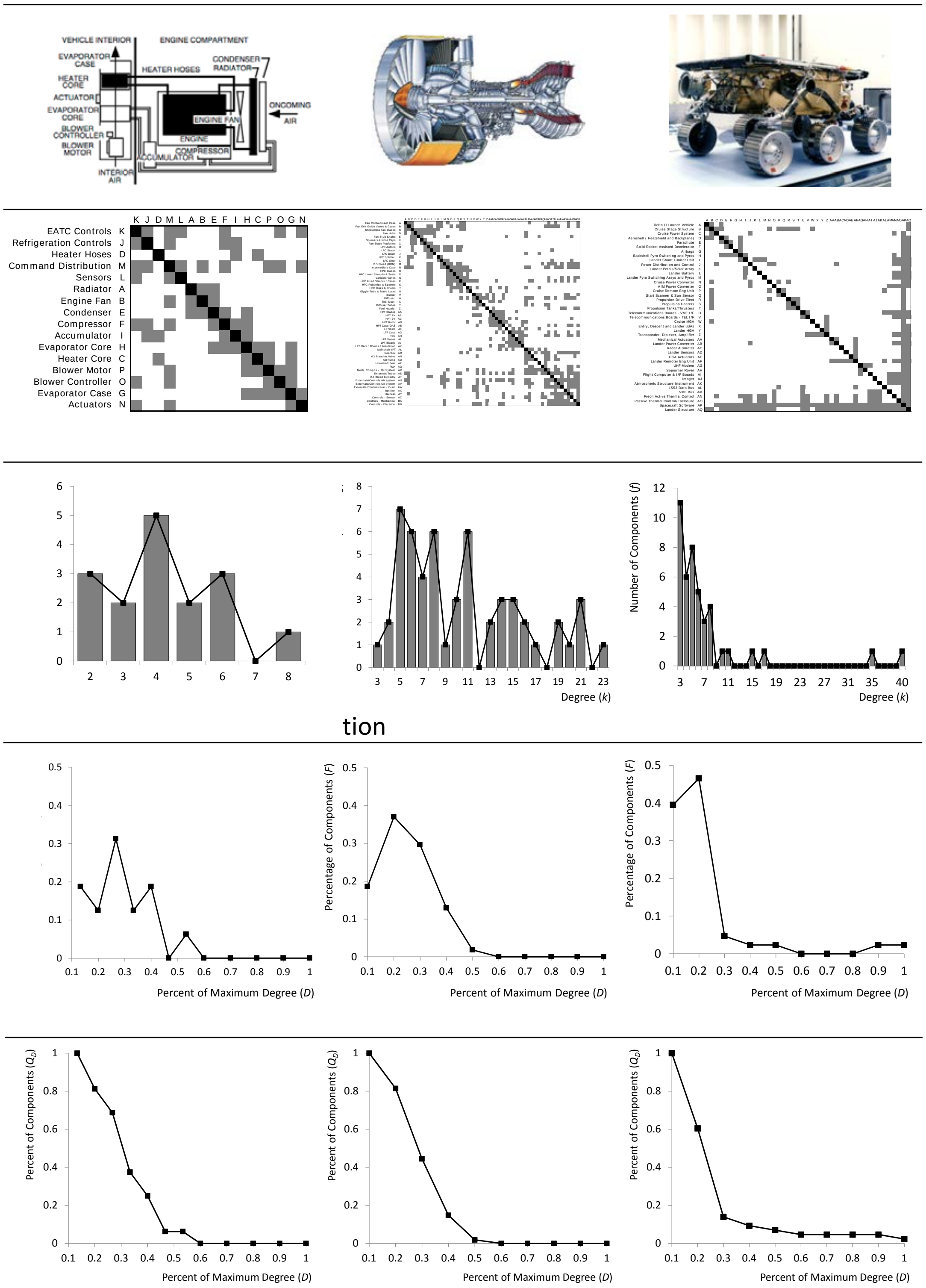


Figure 2: Product depiction, DSM and degree distributions for the climate control system (left hand panel), an aircraft engine (middle panel) and the Mars Pathfinder (right hand panel)

2.2 Two alternative approaches to identify hubs

There are alternative approaches to identify hubs. By examining the degree distribution in log-log scales, as is typically done in the complex network literature, we can check if the system under consideration has a degree distribution that follows a power law [3, 32, 33]. Power law distributions have received significant attention in the literature because they have "fat" (or heavy) tails [5, 16]. "Fat tails" imply that there are components (hubs) that are substantially more connected than others. Power law distributions appear in a log-log plot as a straight line whose slope (measured by γ) defines the distribution and thus the "fatness" of the right tail. Thus, if the distribution follows a power law, parameter γ can serve as a measure for the number of hubs. The less negative γ, the larger the right tail of the distribution, and thus the larger the number of hubs in the system [5].

Figure 3 plots in log-log format the cumulative distributions of the three systems shown in Figure 2. Both the climate control and the aircraft engine systems have degree distributions that do not follow a power law, as indicated by the poor fit of the best fitted line in their log-log plots (the R^2 of the fitted lines are 0.68 and 0.63, respectively; both failed a Pearson Chi-square test for a power law governed degree distribution). The degree distribution of the Mars pathfinder offers a better fit ($R^2 = 0.93$), yet it also failed the Pearson Chi-square test. This is consistent with previous work showing that the degree distributions of complex hardware and software systems often do not follow a "pure" power law but a power law with "cut-offs" [3]. ("Cut-offs" can be associated with long-tailed degree distributions in complex product development networks [3].) Some systems follow no power law regime at all [5, 21]. For such systems a power law may only be a very rough approximation for the "fat tail." Therefore, using γ to estimate the extent to which hubs are present in a system has the drawback of sometimes unrealistically imposing a specific functional assumption on the form of the degree distribution.

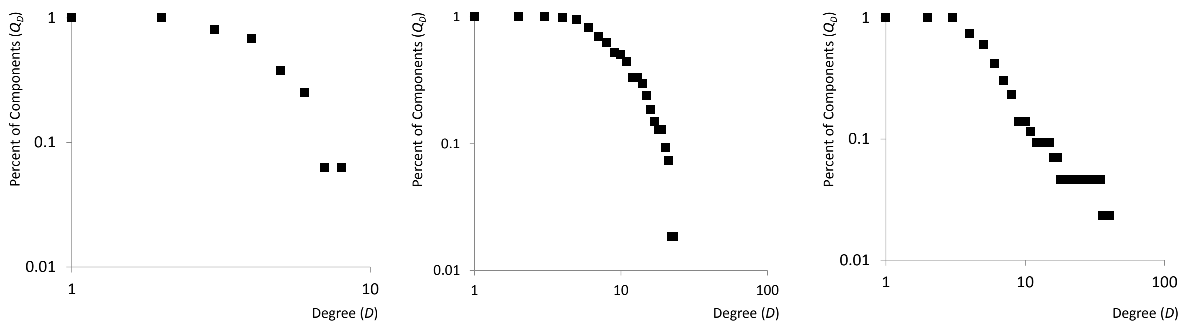


Figure 3: Log-Log Plot of the cumulative distributions for the climate control system (left hand panel), an aircraft engine (middle panel) and the Mars Pathfinder (right hand panel)

Another alternative way to capture the presence of hubs, without making any assumptions regarding the functional form of the degree distribution, is by considering its skewness (the third moment of the degree distribution). Because right-skewed distributions (positive values of skewness) indicate a small set of components with large values of degree, we can assume that the more right-skewed the distribution (the more positive its skewness) the more hubs it may have. For instance, the degree distributions of the three systems shown in Figure 2 have skewness values of 0.4469, 0.6879, and 3.3439 respectively. These values indicate that the presence of hubs is more pronounced in the Mars Pathfinder than in the engine and the climate control system.

The skewness measure has its limitations though. While the number of hubs in a distribution is likely to increase with the value of skewness, this relation is not guaranteed. For example, a skewness of zero and even negative skewness does not rule out the existence of a hub, because such distributions can still have a few highly connected components [5]. Thus, it is hard to translate specific values of skewness to a specific number of hubs, because the (right) skewness of a distribution can be increased in different ways.

Because skewness aggregates in a single measure the overall tendency of the degree distribution to have hubs, and because Q_D quantifies precisely the fraction of hub components for a given D, we use both indicators to examine empirically how the presence of hubs relates to system quality.

2.3 Degree distribution and hubs of a sample of software systems

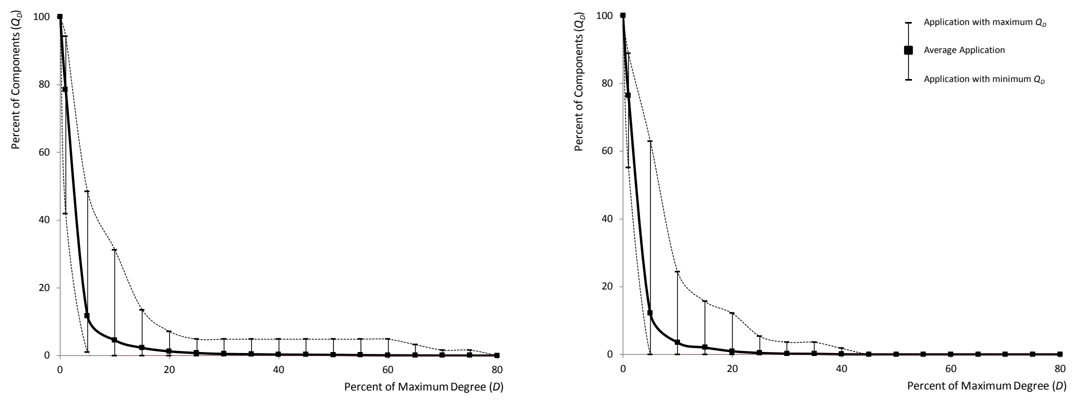
To study the relationship between the fraction of hub components in a system (Q_D) and the system's quality, we examine the degree distribution of 105 software systems from the Apache foundation. Figures

4a and 4b show the minimum, average, and maximum cumulative degree distribution curves for the systems in our sample. Because the connections in our sample are directional, we distinguish between in-degree (Figure 4a) and out-degree (Figure 4b) distributions (see also [1-3]). First, we remark that there is significant variation in the degree distributions in our sample for any given D. Such variation allows us to test empirically the relationship between the fraction of hub components, Q_D (for any given D), and system quality. Second, for both in- and out- degree perspectives, the top 20% most-connected components ($Q_D = 0.20$) in all the systems in our sample have normalized degree greater than D = 0.15. To put the value of D = 0.15 in perspective, note that the mean of the average normalized degree of all the applications in our sample is 0.02. Hence, the normalized degree (in any direction) of a potential hub component is at least seven times larger than the mean normalized degree in the average system in our sample (and more than twice the mean normalized degree of the system with the largest average normalized degree in our sample).

We also fitted power law forms to the degree distributions and found that about 90% of the systems did not follow a "pure" power law (i.e., about 90% of the systems did not pass a Chi-square test, at the 5% level, testing whether their degree distribution followed a power law). Hence, using the power-law coefficient, γ, as a measure for the number of hubs was inappropriate in our sample. In contrast, the skewness values of all the distributions indicate our systems are all right-skewed; hence we use skewness as an alternative indicator of the presence of hubs.

The patterns of degree distributions observed in our sample are consistent with [3], which found evidence of power law regimes with cut-offs in the degree distributions of a single release of two large open-source software systems (the Linux operating system kernel and the MySQL relational database). They found that the distributions of incoming links (equivalent to outgoing "function calls") had significantly smaller cut-offs (faster decaying tails) than the distribution of outgoing links (equivalent to incoming "function calls").[3]

[3] The degree distributions of these systems were based on directed graphs that represent calling relationships among subroutines within each of these systems [3]. The convention used by [3] with respect to the directionality of the link between two nodes is reversed from the one used here. Our convention is similar to the one used in previous work that models software architectures using DSM representations [13,34,47] by which the direction of the link follows the



(a) In-degree **(b) Out-degree**

Figure 4. Cumulative degree distribution curves for our sample of 105 systems

3 Hypotheses

In this section we integrate previous findings in engineering design, complex networks and software development to formulate two hypotheses that relate characteristics of a system's degree distribution to its level of defects. Previous research (e.g., [3, 17, 22]) has suggested that system architectures characterized by the presence of hubs are likely to benefit the design process in various ways and thus have fewer defects than architectures barely using hubs.

Hubs can have two types of integrative roles depending on whether their components "use" or "get used by" other components in the system [1, 11, 34]. Hence, the *in-degree* of component *i* counts the number of other components that use it, and the *out-degree* of component *i* counts the number of other components that get used by it. Previous research in the complex network literature has not only shown that complex product development networks exhibit significant asymmetry in their degree distributions [1-3] but also that such an asymmetry can moderate the effects of hubs on the generation of defects in a system [3].

In-degree hubs can serve as a "platform" upon which many other components build their functionality. Examples include the manifold that distributes material flows to many other components in a chemical plant, the memory chip in a computer which is used to process signals requested by many other components, and the basic software functions (such as memory management, scheduling, and

direction of the "function call" (i.e., in-degree counts the number of incoming function calls). Such a difference in convention does not affect our analysis.

communication/networking) that get called by many other functions in a software application. An out-degree hub provides a "common-entry point" to the system because it provides a path to many other components. Examples of out-degree hubs are the control panel of an automobile, the "black box" of an airplane, and the user-interface components of software applications.

While in-degree and out-degree hubs are functionally different, they both serve as integrative mechanisms in the product architecture domain. A set of components playing a platform role (in-degree hub) tends to force adherence to a common protocol for interface management. This is consistent with the benefits of bus modularity [8, 35]. The presence of a platform-type hub routes function requests through the same set of components, which reduces the need for component-to-component coordination [27]. This reduces the chances of omissions or coordination pitfalls that could potentially lead to defects in the system [27, 36]. Similarly, the presence of "single-entry point" components (out-degree hub) may facilitate the organization of functional requirements into different segments [37]. Hence, from an engineering design perspective, the use of this type of hub can facilitate the internal definition of modules, which hinders the generation of defects by enabling the creation of more modular architectures [37].

Our arguments pertaining to system architecture are reinforced by organizational considerations. The organization often mirrors the technical structure of systems [36, 38-40]. Thus, systems that make a considerable use of hubs are likely to have development actors that act as coordination brokers and facilitate critical information flows among other developers [2]. Consistent with the notion of betweenness centrality, developers of the hub components centralize and distribute information during the development, which enables coordination among the otherwise disconnected actors [1, 39]. Moreover, developers responsible for hub components tend to invest greater efforts in interface management and standardization [41] to reduce the chances of critical interfaces going unattended, which can lead to costly rework or quality issues [36, 39].

In addition to being integrative mechanisms, hubs are also salient candidates for component reuse [42]. Because hubs access (or get accessed) by many components in the system, they are considered important elements when designing the platform for a family of products. Hence, components that form part of a hub are more likely to be part of a product family, which attracts additional organizational attention and resources to scrutinize their quality [43-45]. Even within a single long-lived system, such as software products, hubs are more likely to be reused than components with low degree [34]. Again, in such a case the more highly connected components (hubs) are more likely to be scrutinized and debugged over several product generations [34, 46].

Having highly scrutinized and reliable hubs is important for those systems in which hubs may build efficient redundancy: if a component fails, then hubs can re-route the dependency from the failed component to prevent overall system failure [3, 22]. This is consistent with previous work in complex networks, which has suggested that complex engineered systems showing a right-skewed degree distribution are more robust (both from a static and dynamic perspective) to random node failures, because such a local failure is less likely to propagate and bring the whole system down [22]. Such resiliency is attributed to the presence of hubs and is particularly salient in new product development networks. As shown by [3], new product development networks with inhomogeneous degree distributions of tasks are less likely to have unresolved tasks, which could lead to product defects.

Note that we are not arguing for a positive relationship between component degree (i.e., mere direct connectivity) and lower levels of defects. Rather, grounded in previous findings, we suggest that the presence of hubs provides a net benefit (reflected in lower levels of defects for the system) because they facilitate the design of other components in the system, some of which might be critical to deliver specific product functionality. (For evidence of this concerning the aircraft engine in Figure 3, see [27]; for evidence of this concerning software products, see [47].)

Considering the integrative and reusability benefits associated with the presence of hubs, we posit the following baseline hypothesis to test empirically:

H1*: Systems architectures characterized by the presence of hubs are likely to be less defect-prone. That is:*

a) Systems with right skewed in-degree and out-degree distributions are more likely to have a below-average number of defects.

b) Systems with above-average fractions of in-degree and out-degree hub components ($Q_{D,in}$ and $Q_{D,out}$) are more likely to have a below-average number of defects.

Although H1 suggests that increasing the fraction of hub components will improve system quality, we argue that there are limits to the benefits associated with the presence of hubs, which would imply an optimal fraction of hub components that minimizes the number of defects in a system.

The limits to using hubs are associated with the limited capacity of development actors to process information [1-3, 48]. Because humans have a limited capacity to provide information to and receive information from others, designing components with disproportionally connected components tests the bounded rationality of certain people in the organization [1-3, 21, 48]. Moreover, because developers are likely to have a greater capacity to provide information to others (broadcasting information) than to receive and process it, Braha and Bar-Yam [1-3] suggested that the significant asymmetry in degree distributions observed in new product development networks is (in part) a consequence of the bounded rationality argument originally put forward by Simon [48].

There are also limits to the number of hubs. From an organizational viewpoint, since using more hubs implies the inclusion of more components with significantly more interfaces to coordinate, they add a significant coordination cost to the organization [49]. Although the organization can afford to have some people specialized in handling interfaces across organization and system boundaries [39], there is a limit to the number and size of such highly communicative groups that are focused on integration at the expense of people focused on implementing specific product components.

The reusability argument associated with hub components also gets diminishing returns as the use of reusable (standardized) hub components becomes excessive. With a given level of system functionality, the use of more reusable (standardized) hub components increases the risk of them fulfilling system-specific functions [50]. However, because such reusable components are designed to fulfill generic (rather than specific) functions, their "misuse" leads to unnecessary complexity in the system that may in turn cause defects [43].

In sum, because there are limits to the benefits associated with the presence of hubs as the fraction of hub components increase, we posit our second set of hypotheses to test:

H2a: *There is an optimal fraction of in-degree hub components ($Q_{D,in}$) that minimizes the expected number of defects in the system.*

H2b: *There is an optimal fraction of out-degree hub components ($Q_{D,out}$) that minimizes the expected number of defects in the system.*

4 Empirical Evidence

To test our hypotheses, we studied open-source, Java-based software applications from the Apache Software Foundation (http://www.apache.org/), one of the largest, best-established, and most widely studied open-source communities of developers and users who share values and a standard development process [51]. We examine the architecture of software products for three reasons: (a) they are complex, (b) they exhibit fast change rates (enabling longitudinal study), and (c) they offer (through their source code) an efficient, reliable, and standardized medium to capture their architecture. Thus, as fruit flies facilitate studies of biological evolution, software systems allow for the study of complex systems and the relation between their inner characteristics and quality.

Initially, we identified 69 Java-based development projects at the Apache Software Foundation in mid-2008. An effective examination of the causal relationship between architectural characteristics and quality requires a longitudinal data set, so we only considered products for which we could obtain related product and bug data for successive major releases. Such filtering left us with a set of 105 releases representing 16 products with an average of 6.6 major releases (or versions) each. (Our sample can be extended to 122 releases if we exclude two organization-related control variables from the analysis. Our overall results hold with this extended sample.) The product sizes in our sample of 105 releases range from 29 to 1,282 components ($\bar{x} = 279$, $s = 216$).

We capture the architecture of each product release with a readily available tool called LDM (by Lattix, Inc., www.lattix.com), which automatically extracts the dependency structure of each system from its source code. We represent each product release in a product DSM and use this as the basis for calculating the degree distribution and other architectural variables (see [14, 15] for details).

Concerning the types of dependencies captured in our product DSMs, if the source code in a Java class (i.e., a product component) refers to another class, then we say that the class is dependent on the class being referred to. Specifically, we capture the following types of syntactic dependencies: *invocations* (static, virtual and interface), which allow for various types of method calls; *inheritances* (extensions and implementations), which allow a class to extend or define new behaviors; *data member references*, which refer to the field of a class; and *constructs*, a method call for creating an object. We include these dependencies because they are generally integral to the design of the system and because developers create them deliberately. Although our approach is consistent with the majority of prior research in relying on static dependencies (e.g., [13, 52]), we acknowledge that we do not capture some types of data or logical dependencies, in particular dynamic runtime dependencies (such as memory use), which are also likely to affect software quality [53, 54]. Moreover, the automated methods for extracting dependencies are highly reliable and replicable for the types of static design-related dependencies they do address when compared to methods for models of hardware systems and design processes (e.g., [6, 11, 27, 36, 55]), where the dependencies must be gathered manually and/or subjectively.

4.1 Variables

To test our hypotheses we need to measure for each product in our sample a key dependent variable of interest (namely, the number of defects per product release), a predictor variable (for instance, the fraction of hub components of each product release from either an in-degree ($Q_{D_{in}}$) or out-degree ($Q_{D_{out}}$) perspective), and a set of control variables that help us prevent confounding effects and spurious regressions. Based on these variables our regression model estimates the effect of the predictor variable on the dependent variable in the presence of the control variables.

4.1.1 Dependent Variable: Number of defects per product

Number of defects associated with version s of product i (y_{is}). Our dependent variable counts all the defects (bugs) that have been formally logged into the project's bug tracking systems and attributed to version s of product i. Typically, the identification of a formally reported defect is carried out by developers or users (with confirmation by developers) after the product has been released. However, in some occasions bugs are attributed to version s before its official release. Our dependent variable counts all of these bugs as long as they are explicitly assigned to version s.

4.1.2 Predictor Variables

Skewness ($skewness_{is}$). We use the most common measure of skewness used in probability and statistics [56], the distribution's third standardized moment:

$$skewness_{is} = \frac{\frac{1}{N}\sum_{j=1}^{N}\left(x_j-\mu\right)^3}{\sigma^3} \tag{1}$$

where x_j is the degree of component j of version s of product i; μ is the mean degree of version s of product i, σ is the corresponding standard deviation around the mean degree, and N is the number of components in version s of product i. The measure is positive when the right tail is more pronounced than the left tail, indicating that the distribution is right-skewed and negative if the left tail is more pronounced than the right. In our sample, all the product releases exhibited various degrees of positive skewness, indicating that they all were to some extent right-skewed.

Fraction of hub components ($Q_{D,is}$). Based on the cumulative normalized in-degree (or out-degree) distribution, we measure the fraction of hub components in version s of product i that have a normalized in-degree (or out-degree) greater than or equal to D. Since normalized in-degree (D_{in}) and out-degree (D_{out}) can be theoretically defined in the range [0,1], we determine $Q_{D_{in},is}$ (and $Q_{D_{out},is}$) for each 0.05 increment of D_{in} (and D_{out}) provided that $Q_{D_{in},is}$ (and $Q_{D_{out},is}$) is equal to (or smaller than) 0.20 (which is the Pareto law threshold we exogenously imposed to identify hubs).

4.1.3 Control Variables

Including a relevant set of control variables ensures that our hypothesis testing rules out possible alternative explanations that could be confounded with the effect of our dependent variables. We include two sets of control variables. First, we control for exogenous, non-architectural features of the product that are likely to affect the generation of defects. Our non-architectural controls, described in Table 1, neutralize the potentially confounding effects of program size and complexity (e.g., source lines of code and cyclomatic complexity) as well as effects that influence the discovery process of defects (e.g., the time to the next release). Second, we control for architectural characteristics as functions of the interaction patterns in each product release (see [57] for various measures of complexity in engineered systems). Table 2 provides a description of our architectural controls.

Table 1: Description of non-architectural control variables

Non-architectural Controls	Description
Source lines of code ($SLOC_{is}$)	Since both the number of defects as well as the degree distribution may be a function of the overall system complexity, we need to control for it. The overall complexity of a system is a function of the amount of information it carries. One of the most widely used metrics to capture the raw complexity of a software product is its number of source-code lines [58, 59]. We measure the number of source lines of code (in "kilolines") with the readily available JHawk tool (www.virtualmachinery.com), which counts the number of statements (excluding comments) in each method of each component of version s of product i.
Average cyclomatic complexity (AVG_CC_{is})	Since the presence of defects in the system can be a function of the internal complexity of the components in the system, we control for the average component complexity with one of the most widely accepted measures in the software domain. Cyclomatic complexity is the minimum number of linearly independent paths in the control flow graph of a software program [60]. Cyclomatic complexity is typically used to identify the methods of a program that would be harder to test and build [58, 60]. We use the JHawk tool to calculate the cyclomatic complexity of all methods in our sample. We then average cyclomatic complexity of all methods in version s of product i.
Age of product at version s (AGE_{is})	The time a component has been in existence may influence both bugs (as it may be debugged) and degree distribution (as it may attract more links). The age of the product is measured by the number of days since its development began. This assumes that the application is officially "born" on the date of the first release available (as indicated in the release notes) and then ages with successive releases.
Days since last release ($DAYS\_BEFORE_{is}$)	The time between successive releases varies within and across products and directly influences the likelihood a bug is discovered, so it is important to control for the time span between the previous release and the release of version s.
Days to next release ($DAYS\_AFTER_{is}$)	This is the time between the current version s and the next release ($s + 1$). The longer this time, the higher the probability that bugs will be discovered, because it corresponds to when the application is most actively scrutinized by testers and users.
Newness of application at version s ($NEWNESS_{is}$)	Both *new features* (added functionality) and *incremental improvements* (modifications to existing functionality) add uncertainty and complexity to the structure of a product, influencing the generation of bugs. Using information from the release notes of each product release, we count the "new features" and "improvements" items in such notes as an indicator of the newness of version *s* of application *i*.
Implicit bugs ($IMPLICIT\_BUGS_{is}$)	Some bugs reported in the bug-tracking system are not explicitly assigned to a specific version, so they are not accounted for by our dependent variable. We control for the existence of these "implicit" bugs because their discovery may influence the discovery of bugs that are explicitly assigned to version s. We assign an "implicit" bug to the version that was most recently released when the bug was entered into the bug-tracking system.
Cumulative number of changes ($CUM\_CHANGES_{is}$)	Because previous work in information systems has found positive association between "churn" metrics (such as the number of changes made to product components) and software failures [61, 62], we control for the cumulative number of changes associated with components of version s *prior to* its release.

Average breadth of a change ($AVG\_BREADTH\_CHANGE_{is}$)	Because our sample is longitudinal, it is important to control for the "decay" of the source code over time (i.e., changes in the source code tend to involve more components as the product gets older) [63]. We control for this by measuring the average number of components in version *s* that have been modified in version *s* due to either bug fixing, product improvements, or product new features.
Average interface classes usage ($AVG\_INTERFACE\_USAGE_{is}$)	In object-oriented programming, as enabled by Java, the use of "interface-type" components is customary to decouple modules. We aim to control for the ability of developers to properly use "interface-type" classes to handle dependencies across modules. To do so, we use the normalized metric of "distance" proposed by Martin [64, p. 267], which assesses developers' ineffectiveness at grouping interface-type components into potentially stable modules. See [64, p. 264] for a detailed description. We again use the LDM tool to calculate this metric directly from the JAR files of each product release in our sample.

Table 2: Description of architectural control variables

Architectural Controls	Description
Number of nominal modules ($NUM\_NOM\_MODULES_{is}$)	This measure counts the number of component-based modules in each product release. The products in our data set are complex systems formed by interrelated components. To manage this complexity, system architects group the components into hierarchically organized modules. Typically, modules aggregate components that collectively perform certain functions.
Propagation cost ($PROPAGATION_{is}$)	The presence *or* absence of direct and indirect dependencies between the components of a complex system can create defects [7, 11, 13, 57]. We control for the overall connectedness of the components in a product release by calculating its propagation cost as defined by [13].
Intrinsic cyclicality ($P_{I,is}$)	In complex systems in which the dependencies are directional (such as the function calls in software products, and the energy, material, or information flow in hardware products), some components might be connected in a cyclical manner forming "component loops." Sosa *et al.* [15] showed that one particular type of component loop called intrinsic loops influences product quality most significantly. We measure intrinsic cyclicality as the fraction of components in version *s* of product *i* involved in intrinsic component loops**.**
Number of component loops (NUM_LOOPS_{is})	Because intrinsic cyclicality does not explicitly control for the number of intrinsic component loops in the system, we include a control for it.
Average clustering coefficient ($CLUSTERING_{is}$)	Because previous research in information system has found that the density of connections surrounding components (i.e. the extent to which components directly connected to a focal component are connected among themselves) may influence coordination breakdowns and software failures in various ways, we include a control for it [53]. We calculate a measure equivalent to the local clustering coefficient used in the small world literature [20], which measures the fraction of a component's neighbors that neighbor each other. We then average these component-level coefficients over the components in the application. Observe that because we are interested in whether two components are connected or not (neighboring components), we symmetrize our DSMs for the purpose of calculating this measure. Doing so allows us to calculate the clustering coefficient using non-directed links as done by [20]. Nonetheless, one can also calculate a clustering coefficient as a fraction of transitive triplets for the directed DSMs.
In-Out degree correlation ($DEGREE\_CORRELATION_{is}$)	Pearson correlation between the in- and out- degree of the components of each product release in our sample. Because simulation-based research in complex networks has shown that the correlation between in- and out-degree distribution may influence the risk of development tasks to be unresolved (which can also lead to the generation of defects) [3], we control for two types of correlations, of which this is the first.
Out-out neighboring degree correlation ($OUT - OUT\_NEIGHBOR\_DG\_CORRL_{is}$)	Second, and also consistent with [3], we control for the out-out degree correlation between neighboring components (i.e., pairs of interconnected components). This measure correlates the number of outgoing links (equivalent to incoming design dependencies) between connected nodes in a network. (Note that because our convention of link directionality is reversed from the one used by [3], our measure corresponds to the *in-in* neighboring degree correlation proposed by [3].) For robustness, we also tested our hypotheses with a similar measure based on in-in neighboring degree correlation and obtained a similar pattern of results.

4.2 A statistical model to predict defects in the system

We estimate a regression model that predicts the number of defects in each product release. Because our dependent variable (y_{is}) exhibits skewed count distributions (which take only nonnegative values), a linear model (estimated with standard ordinary least-squares regressions) could lead to inefficient and biased estimates and is thus inappropriate. Poisson-like regression models, in contrast, have been developed to explicitly model the count nature of dependent variables [65]. Because the variance of our dependent variable is significantly larger than its mean, negative binomial regression models provide a more accurate estimate of the standard errors of the coefficient estimates than a basic Poisson regression [65, 66]. Finally, because of the panel structure of our data, we use panel data regression procedures. We therefore estimate a model of the form [65, p. 279].

$$E[y_{is}|x_{is}, \alpha_i] = \alpha_i \, e^{(x'_{is}\beta)} \qquad (2)$$

For estimation purposes, we use the *Stata* command *xtnbreg* with both application- and year-specific fixed effects. The β-coefficients for our predictor variables as well as for the control variables are shown in Tables 3, 4, and 5 below. The coefficient β_j quantifies the change in the expected number of defects if the jth regressor changes by one unit. A significantly positive (negative) β_j coefficient indicates that, all else being equal, an increase (decrease) in regressor j increases (decreases) the expected number of defects in the system. For example, a significantly *negative* coefficient $\beta_{D_{in}}$ of variable $Q_{D_{in},is}$ would indicate that, the expected number of defects of a system with above-average percentage of components with normalized in-degree D_{in}, is significantly below the average number defects in the sample (all else equal). This result would be in line with our hypothesis H1.

The α_i are product-specific fixed effects. These effects permit observations of the same product to be correlated across versions, thereby allowing the model to incorporate serial correlation. These fixed effects capture any time-invariant, unobserved, product-specific features and thus effectively control for any such factors which we could not explicitly measure such as the “culture” or “baseline experience” of the development team of each product. Finally, because software development technologies may change significantly from year to year, and such developments might affect the generation of defects across all of the products, we include indicator variables for the year of each release.

4.3 Testing our hypotheses

Table A1 (in the appendix) shows descriptive statistics and pair-wise correlations between the variables included in the analysis. As expected, some of the architectural controls (such as propagation cost, intrinsic cyclicality, and clustering) show significantly high levels of positive correlation. In-out degree correlation is not only positively correlated with architectural variables but also negatively correlated with out-out neighboring correlation. Both in- and out-degree skewness are positively correlated, indicating that the variations in asymmetry of the in- and out-degree distributions are positively associated. Finally, in- and out-degree skewness are not significantly correlated with $Q_{D,in}$ and $Q_{D,out}$, respectively, for values of D = 0.25 (which is in the mid-range of D values used in our analysis). This suggests that skewness and Q_D capture complementary (rather than redundant) aspects associated with the presence of hubs in a system.[4]

We estimate two sets of regression models to test our hypotheses. Tables 3 and 4 show the standardized coefficient estimates ($\widehat{\beta_J}$) of the models predicting the expected number of defects.[5] Table 3 presents two baseline models as well as the models used to test hypothesis H1. Table 4 reports the models used to test hypothesis H2.

Model 1, in Table 3, includes all our non-architectural controls and a first set of architectural controls. Model 2 adds the other three architectural control variables (clustering and two degree correlation variables). Because architectural variables are highly correlated among themselves, we estimate variance

[4] The lack of significance in correlation between Q_D and skewness can be explained by the following thought experiment. Assume a network with non-directed dependencies and a certain (right-skewed) degree distribution. In addition, choose a D to define Q_D. How are both skewness and Q_D likely to change if a new node is added to the network? If the newly added node has a degree that is smaller than the average mean, then skewness will decrease and Q_D will decrease (or will stay at zero if the original Q_D = 0). If the newly added node has degree greater than the mean degree but smaller than D, then skewness will increase whereas Q_D will decrease (or will stay equal to zero if the original Q_D = 0). Finally, if the newly added node has degree greater than or equal to D, then skewness will increase and Q_D will also increase. Hence, the actual relationship between skewness and Q_D is an empirical question, which in our dataset does not exhibit a significant correlation for a wide range of D.

[5] Because the variables are measured using very different unit scales, and to facilitate the interpretation of the estimated coefficients, we standardize each variable to its z-score before entering it into the regressions [67]. The z-score of a variable is obtained by subtracting its mean value and dividing it by its standard deviation.

inflated factors (VIF) and found that intrinsic cyclicality exhibited VIF superior to the acceptable threshold of 10 [68]. Hence, we exclude it from our baseline model, although our results are substantially similar if including intrinsic cyclicality in the baseline model. Moreover, although we use Model 2 as the baseline model, the results with respect to our hypotheses are similar using Model 1 as the baseline model. Model 2 is also consistent with [3] by showing that both in-out degree correlation and out-out neighboring degree correlation are positively associated with defects.

Models 3, 4, and 5 test hypothesis H1, which predicts that the presence of hubs in a system is associated with fewer defects. Models 3 and 4 show a significant, negative coefficient for the skewness measure of the in- and out-degree distributions, respectively. This indicates that the more right-skewed the degree distributions of a system, the fewer the defects they are likely to exhibit. Model 5 includes the two measures of skewness and shows that out-degree skewness is significant (-0.999, $p < .001$), whereas in-degree skewness is not (0.046, $p < .867$). This is not entirely surprising given the significant, positive correlation between these two measures (0.46, $p < .0001$). Most of the effect of in-degree skewness on the dependent variable coincides with part of the effect of out-degree skewness. As a result, in our sample, it is not possible to fully separate out the effect of in-degree skeweness from the effect of out-degree skewness.

Because the fraction of hub components (Q_D) can be defined with respect to different normalized degrees (D), we test hypothesis H1 for many possible definitions of (Q_D) by systematically varying D in increments of 5% ($Q_{D,is} \in \{Q_{0.95,is}, \dots, Q_{0.10,is}, Q_{0.15,is}\}$. We consider our predictor variable Q_D for values of D greater than 0.15 in order to ensure that hubs comprise no more than the top 20% most connected components in a system. Models 6 and 7, in Table 3, show the coefficient estimates for in-degree hubs corresponding to $D_{in} = 0.15$ and $D_{in} = 0.25$, whereas models 8 and 9 show coefficient estimates for out-degree hubs corresponding to $D_{out} = 0.15$ and $D_{out} = 0.35$. These values of D define the range of D in which the Q_D show a significant linear relationship with the number of defects. Because Q_D is uncorrelated with skewness, we include both in- and out-degree skewness in all of these models. These models all exhibit negative and significant coefficient estimates for Q_D, indicating that applications in our sample with above-average fractions of hub components are significantly associated with a below-average number of defects. Considering Model 6, for instance: the significant, negative coefficient of $Q_{D_{in}}$ (-1.103, $p < 0.001$) indicates that a one standard deviation increase in the fraction of components with normalized degree greater than or equal to 0.15 correlates with 67% fewer defects ($1 - e^{-1.103}$ = 0.67). Overall, models 6-9, in Table 3, provide support to hypothesis H1b. The models in Table 4 provide further support for hypothesis H1 over a wider range of D by also finding an optimal fraction of hub components (Q_D) that is above the average value of Q_D.

Include Table 3 here

4.4 Finding an optimal fraction of hub components

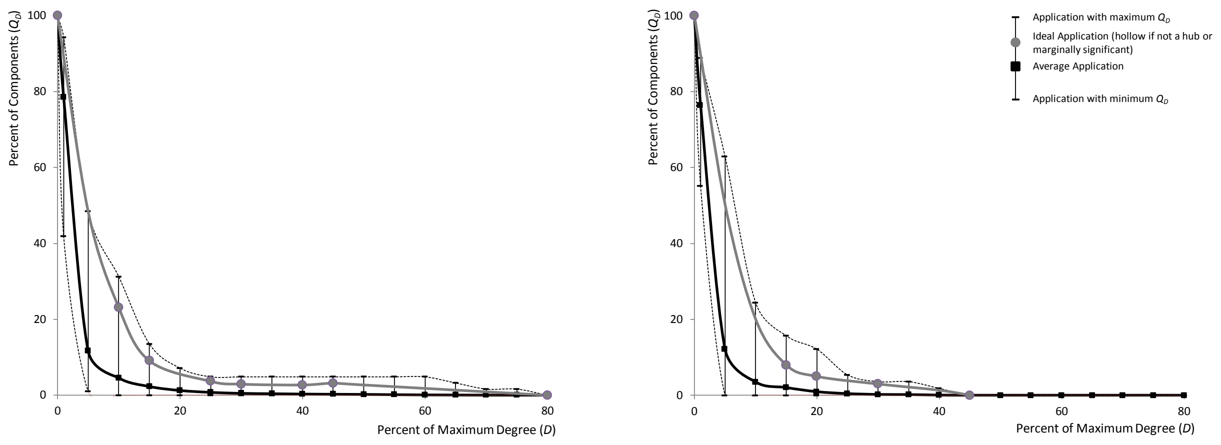
According to hypothesis H2, it would be unrealistic to expect that increasing either $Q_{D_{in}}$ or $Q_{D_{out}}$ without limits would yield a system with ever-fewer defects. Rather, we expect to find an optimal level of $Q_{D_{in}}$ (or $Q_{D_{out}}$) above which further increases would be detrimental. We test for the existence of an optimal level of Q_D by adding a quadratic term of Q_D to our regression models. Hence, a model that shows both a negative and significant coefficient of Q_D and a positive and significant coefficient of ${Q_D}^2$ would indicate that there is a level of Q_D (within the range of data in our sample) that minimizes the expected number of bugs. Such a model would suggest that the relationship between Q_D and our dependent variable is U-shaped (all else constant). Table 4 shows the coefficients for the models that improve their goodness of fit significantly after adding the quadratic term of $Q_{D_{in}}$ and $Q_{D_{out}}$, respectively.

Include Table 4 here

The first five models in Table 4 provide empirical evidence for a U-shape relationship between $Q_{D_{in}}$ and our dependent variable for values of D_{in} in the range $D_{in} \in \{0.15, 0.25, 0.30, 0.40, 0.45\}$. This provides empirical support to hypothesis H2a. For instance, Model 1 suggests that those systems with normalized in-degree $D_{in} \geq 0.15$ are likely to have the least number of defects if 9.0% of components are involved in hubs.[6] To put the value of 9.0% in perspective, note that the average value of $Q_{0.15_{in}}$ in our sample is 2.3% and the maximum value of $Q_{0.15_{in}}$ is 13.5%.

To illustrate better the regression results shown in Table 4, Figure 5a plots $Q_{D_{in}}$ as a function of D_{in}. In addition to the minimum, average, and maximum application of $Q_{D_{in}}$, we plot an "ideal" curve which joins the optimal values of $Q_{D_{in}}$ for the values of D_{in} for which we could find a significant optimal value of $Q_{D_{in}}$. The existence of the ideal curve above the average curve is fully in line with the prediction of hypothesis H2a. Note that the ideal curve is above the average curve even for those D for which we could not find a significant linear effect in our first analysis, implying that having an above-average fraction of components with high in-degree improves the quality of the system, which further bolsters H1a.

As for the out-degree perspective, we found a significant optimal level of $Q_{D_{out}}$ for values of $D_{out} = 0.15$, $D_{out} = 0.20$ and $D_{out} = 0.30$ (see Models 6-8 in Table 4). This provides empirical evidence supporting hypothesis H2b. Figure 5b plots the optimal $Q_{D_{out}}$, or the "ideal" curve, as a function of D_{out}. Again, the ideal curve is above the average curve, which is line with H1b.



(a) In-degree **(b) Out-degree**

Figure 5. Graphical representation of regression results

[6] To see this, note that equalizing to zero the partial derivative of y_{is} with respect to $Q_{0.15_{in}}$ in Model 1 yields the z-score of $Q_{0.15_{in}}$ = 2.105, which minimizes y_{is}, all else constant. To obtain the value of 9.0%, multiply such a z-score by the standard deviation and add its mean value, $2.105 * 0.032 + 0.023 = 0.09$.

Overall, our statistical analysis provides substantial empirical evidence that systems with in-degree and out-degree distributions with "thicker than average" right tails (i.e., above-average fraction of hub components) are more likely to have fewer defects. Interestingly, we found that there is a limit on the improvement that increasing the fraction of hub components (i.e., the thickness of the right tail of the degree distribution) has from both in-degree and out-degree perspectives. The fact that for certain values of normalized degree *D* we did not find an optimal level does not imply that such optimal levels do not exist; it only implies that the range of observations available in our sample was not large enough to provide statistical significance.

5 Discussion

This paper integrates work from the engineering design and complex network literatures to study how the degree distribution of a complex system relates to its quality. In particular, our work relates the fraction of hub components and system quality in an empirical (econometric-based) study.

Our first set of results implies that systems with above-average fraction of hub components have a below-average number of defects. These results reinforce to system architects, designers, and managers the crucial role that hubs play in determining the quality of large, engineered systems: If better systems require a larger fraction of components playing the roles of hubs, then managers of complex system design projects should ensure the allocation of sufficient resources and attention to the design of such a crucial set of components. This is not trivial, because hubs continue to be a very small fraction of the components in the overall system and are therefore at risk of being allocated too few resources and too little attention, incommensurate with the crucial integration role they play and their effect on system quality.

Our second set of results suggests that there is an optimal fraction of hub components with respect to system quality. Hubs provide net benefits reflected in better quality products, but using hubs excessively is detrimental to quality. Providing empirical support to that notion critically underlines the need to carefully deploy resources assigned to the design and maintenance of hubs. The good news is that the approach we use to find such an optimal level is generally applicable to any family of complex systems for which one can capture both architectural and quality features.

5.1 The directionality of interfaces: In-degree versus out-degree

Our analysis distinguished between the in-degree and out-degree distributions [1]. This is consistent with the fact that the interfaces between components in software systems are mostly directional, because they represent information flows [13, 69]. Although hardware systems have many symmetrical interfaces (most of which are spatial dependencies between components), they typically also contain many directional interfaces that are determined by energy, material, and/or information flow between components (see [11, 27] for examples of directional design interfaces among the components of the large aircraft commercial engine shown in Figure 1).

From a degree distribution perspective, the distinction between in-degree and out-degree is important because they define two different types of hubs, platform and common-entry-point hubs. We found that the presence of both types of hubs seems to prevent defects in the system, presumably because they play integrative roles. This is interesting because previous work in the engineering design and management science literatures has recognized the benefits of using platform-type hubs when designing complex systems [43-45], but very little attention has been given to the benefits of using "common-entry point" hubs. Our results not only confirm the benefits of using platform-type hubs but also suggest that increasing the number of components playing "common-entry point" hubs can be a useful lever for designers and managers to improve the quality of complex systems.

5.2 A general approach to determine the number of hubs in a complex system

Our approach to test our hypotheses provides the basis for a more general, structured approach to find the appropriate number of hubs which minimizes the expected number of defects in a family of comparable systems. This general approach can be summarized in four major steps:

1) Capture the system architecture of several comparable systems in terms of their components and interfaces. This step results in a network representation of each system in the sample to analyze.

2) Determine the cumulative normalized degree distribution of each system in the sample as illustrated in Figure 1. If interfaces are directional (like in our empirical study), then distinguish in- and out-degree distributions.

3) For each system in the sample, capture the number of defects (or any other measurable quality metric) as well as any other potentially relevant metric that could explain system quality. Such metrics function as control variables to rule out spurious results.

4) Estimate a regression model (analogous to the one defined in equation 2 that predicts system quality as a function of a set of relevant control variables as well as the fraction of hub components (Q_D). Then, for each value of normalized degree D (in increments of 5% or 10%), estimate the fraction of hub components (Q_D) that would minimize the number of defects. The result of this step would be summarized in an "ideal" curve analogous to the ones shown in Figures 6 and 7. Such curves can provide specific guidance to managers about the defect-minimizing number of hubs.

This paper has shown how to apply this approach for a sample of complex software systems. But it could also apply to any sample of hardware systems, as long as both their architectures and quality have been measured for several versions of a sample of systems. Such practice is customary, for instance, in the auto industry where companies keep records of both architectural and quality measures of major subsystems for several versions of their car models. (See [36] for a study that relates the connectivity of automobile subsystems to their number of quality complaints).

5.3 Research questions for future work

Our findings raise a series of questions related to the notion of having hubs in a system. For example, in this paper, we treat all of the component interfaces equally. Yet we know from previous work [6, 11, 27, 53] that interfaces vary in type and strength. Hence, we could define various types of hubs. Does hub type affect system quality? In software systems, for instance, previous work has identified important differences between syntactic and semantic/logical dependencies [53]. A promising area for future research is to investigate whether these differences moderate the hub-quality relationship.

Our analysis is relevant for the design of complex, long-lived, adaptive engineered systems. We took advantage of the rapid evolution of software products as well as an objective and automated way to document their architecture and quality features to carry out our analysis in a sizable longitudinal sample. While we expect similar patterns of results with other complex systems in the automotive, aerospace, and computer industries, in which the use of hubs is common, a verification of such claims is still outstanding. Future work could validate our approach for other types of complex systems.

As for our main dependent variable, we measure quality by the absolute number of defects in the system. As such, we do not distinguish defects according to their type or severity. Is there a taxonomy of defects that helps designers of complex systems focus? Is the effect of the number of hubs on system quality moderated by the severity of the defect? Moreover, once a defect is discovered, how long does it take to fix it? Does the number of hubs influence the time to fix a defect? Further research is needed to improve our understanding of how precisely hubs influence system quality. We hope this paper stimulates future research efforts in this promising area relevant for the design of complex systems.

6 References

[1] Braha, D., and Bar-Yam, Y., 2004, "The Topology of Large-Scale Engineering Problem-Solving Networks," Physical Review E, 69(016113).

[2] Braha, D., and Bar-Yam, Y., 2004, "Information Flow Structure in Large-Scale Product Development Organizational Networks," Journal of Information Technology, 19(4), pp. 234-244.

[3] Braha, D., and Bar-Yam, Y., 2007, "The Statistical Mechanics of Complex Product Development: Empirical and Analytical Results," Management Science, 53(7), pp. 1127-1145.

[4] Barabási, A.-L., and Albert, R., 1999, "Emergence of Scaling in Random Networks," Science, 286, pp. 509-512.

[5] Strogatz, S. H., 2001, "Exploring Complex Networks," Nature, 410, pp. 268-276.

[6] Pimmler, T. U., and Eppinger, S. D., 1994, "Integration Analysis of Product Decompositions," *Proceedings of the ASME Int. Design Eng. Tech. Conferences (Design Theory & Methodology Conference)*, Minneapolis.

[7] Clarkson, P. J., Simons, C., and Eckert, C., 2004, "Predicting Change Propagation in Complex Design," Journal of Mechanical Design, 126, pp. 788-797.

[8] Sharman, D. M., and Yassine, A. A., 2004, "Characterizing Complex Product Architectures," Systems Engineering, 7(1), pp. 35-60.

[9] Chen, L., and Li, S., 2005, "Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition," ASME Journal of Mechanical Design, 127(4), pp. 545-557.

[10] Hölttä, K. M. M., and Otto, K., 2005, "Incorporating Design Effort Complexity Measures in Product Architectural Design and Assessment," Design Studies, 26(5), pp. 463-485.

[11] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2007, "A Network Approach to Define Modularity of Components in Product Design," Journal of Mechanical Design, 129(11), pp. 1118-1129.

[12] Giffin, M., Weck, O. D., Bounova, G., Keller, R., Eckert, C., and Clarkson, P. J., 2009, "Change Propagation Analysis in Complex Technical Systems," Journal of Mechanical Design, 131(8), pp. 081001-081001-14.

[13] MacCormack, A., Rusnak, J., and Baldwin, C. Y., 2006, "Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code," Management Sci., 52(7), pp. 1015-1030.

[14] Sosa, M. E., Browning, T. R., and Mihm, J., 2007, "Studying the Dynamics of the Architecture of Software Products," *Proceedings of the ASME 2007 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2007)*, Las Vegas, NV, Sep. 4-7

[15] Sosa, M. E., Mihm, J., and Browning, T. R., 2010, "Product Architecture and Quality: A Study of Open-Source Software Development," Technical Report No. 2010/53/TOM, INSEAD, Fontainebleau, France.

[16] Newman, M. E. J., Barabási, A.-L., and Watts, D. J., 2006, *The Structure and Dynamics of Networks*, Princeton University Press, Princeton, NJ.

[17] Albert, R., and Barabási, A.-L., 2002, "Statistical Mechanics of Complex Networks," Reviews of Modern Physics, 74(1), pp. 47-97.

[18] Barabási, A.-L., Albert, R., and Jeong, H., 1999, "Mean-Field Theory for Scale-Free Random Networks," Physica A, 272, pp. 173-197.
[19] Huberman, B. A., and Adamic, L. A., 1999, "Internet: Growth Dynamics of the World-Wide Web," Nature, 401(6749), pp. 131.
[20] Watts, D. J., and Strogatz, S. H., 1998, "Collective Dynamics of 'Small-World' Networks," Nature, 393, pp. 440-442.
[21] Amaral, L. A. N., Scala, A., Barthélémy, M., and Stanley, H. E., 2000, "Classes of Behavior of Small-World Networks," Proceedings of the National Academy of Science (PNAS) USA, 97, pp. 11149-11152.
[22] Albert, R., Jeong, H., and Barabási, A.-L., 2000, "Error and Attack Tolerance of Complex Networks," Nature, 406, pp. 378-382.
[23] Callaway, D. S., Newman, M. E. J., Strogatz, S. H., and Watts, D. J., 2000, "Network Robustness and Fragility: Percolation on Random Graphs," Physical Review Letters, 85, pp. 5468-5471.
[24] Harary, F., 1969, *Graph Theory*, Addison-Wesley, Reading, MA.
[25] Borgatti, S. P., and Everett, M. G., 1999, "Models of Core/Periphery Structures," Social Networks, 21, pp. 375-395.
[26] Braha, D., and Bar-Yam, Y., 2006, "From Centrality to Temporary Fame: Dynamic Centrality in Complex Networks," Complexity, 12(2), pp. 59-63.
[27] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2003, "Identifying Modular and Integrative Systems and Their Impact on Design Team Interactions," Journal of Mechanical Design, 125(2), pp. 240-252.
[28] Wasserman, S., and Faust, K., 1994, *Social Network Analysis*, Cambridge University Press, Cambridge, UK.
[29] Persky, J., 1992, "Pareto's Law," Journal of Economic Perspectives, 6(2), pp. 181-192.
[30] Muirhead, B. K., 1996, "Mars Pathfinder Flight System Design and Implementation," *Proceedings of the IEEE Aerospace Applications Conference*, pp. 159-171.
[31] Brady, T. K., 2002, "Utilization of Dependency Structure Matrix Analysis to Assess Complex Project Designs," *Proceedings of the ASME 2002 Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2002)*, Montreal, Canada, Sep 29 - Oct 2.
[32] Adamic, L. A., and Huberman, B. A., 2000, "Power-Law Distribution of the World Wide Web," Science, 287, pp. 2115.
[33] Faloutsos, M., Faloutsos, P., and Faloutsos, C., 1999, "On Power-Law Relationships of the Internet Topology," *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '99)* Cambridge, MA, Aug 30 - Sep 3.
[34] Maccormack, A., Baldwin, C. Y., and Rusnak, J., 2008, "The Impact of Component Modularity on Design Evolution: Evidence from the Software Industry," Technical Report No. 08-038, Harvard Business School, Boston, MA.
[35] Ulrich, K. T., 1995, "The Role of Product Architecture in the Manufacturing Firm," Research Policy, 24(3), pp. 419-440.
[36] Gokpinar, B., Hopp, W. J., and Iravani, S. M. R., 2010, "The Impact of Misalignment of Organizational Structure and Product Architecture on Quality in Complex Product Development," Management Science, 56(3), pp. 468-484.
[37] Yu, J. S., Gonzalez-Zugasti, J. P., and Otto, K. N., 1999, "Product Architecture Definition Based Upon Customer Demands," Journal of Mechanical Design, 121, pp. 329-335.
[38] Henderson, R. M., and Clark, K. B., 1990, "Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms," Administrative Science Quarterly, 35, pp. 9-30.
[39] Sosa, M. E., Eppinger, S. D., and Rowles, C. M., 2004, "The Misalignment of Product Architecture and Organizational Structure in Complex Product Development," Management Science, 50(12), pp. 1674-1689.
[40] Cataldo, M., Wagstrom, P., Herbsleb, J. D., and Carley, K. M., 2006, "Identification of Coordination Requirements: Implications for the Design of Collaboration and Awareness Tools," *Proceedings of the ACM Conference on Computer-Supported Cooperative Work*, Banff, Alberta, pp. 353-362.
[41] Baldwin, C. Y., and Clark, K. B., 2000, *Design Rules: The Power of Modularity*, MIT Press, Cambridge, MA.
[42] Braha, D., and Maimon, O., 1998, *A Mathematical Theory of Design: Foundations, Algorithms and Applications*, Kluwer Academic Publishers, Norwell, MA.
[43] Robertson, D., and Ulrich, K., 1998, "Planning for Product Platforms," Sloan Management Review, 39(4), pp. 19-31.
[44] Gonzalez-Zugasti, J. P., Otto, K. N., and Baker, J. D., 2001, "Assessing Value in Platformed Product Family Design," Research in Engineering Design, 13(1), pp. 30-41.

[45] Suh, E. S., Weck, O. L. D., and Chang, D., 2007, "Flexible Product Platforms: Framework and Case Study," Research in Engineering Design, 18(2), pp. 67-89.
[46] Von Krogh, G., Stuermer, M., Geipel, M., Spaeth, S., and Haefliger, S., 2008, "How Component Dependencies Predict Change in Complex Technologies," Technical Report No. ETH Zurich, Zurich, Switzerland.
[47] Maccormack, A., Baldwin, C. Y., and Rusnak, J., 2010, "The Architecture of Complex Systems: Do Core-Periphery Structures Dominate?," Technical Report No. 10-059, Harvard Business School, Cambridge, MA.
[48] Simon, H. A., 1996, *The Sciences of the Artificial*, MIT Press, Cambridge, MA.
[49] Baldwin, C. Y., 2008, "Where Do Transactions Come From? Modularity, Transactions, and the Boundaries of Firms," Industrial and Corporate Change, 17(1), pp. 155-195.
[50] Ulrich, K., and Ellison, D. J., 1999, "Holistic Customer Requirements and the Design-Select Decision," Management Science, 45(5), pp. 641-658.
[51] Roberts, J. A., Hann, I.-H., and Slaughter, S. A., 2006, "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects," Management Science, 52(7), pp. 984-999.
[52] El-Emam, K., 2002, "Object-Oriented Metrics: A Review of Theory and Practice," in H. Erdogmus, et al., Eds., *Advances in Software Engineering*, New York, NY: Springer, pp. 23-50.
[53] Cataldo, M., Mockus, A., Roberts, J. A., and Herbsleb, J. D., 2009, "Software Dependencies, Work Dependencies, and Their Impact on Failures," IEEE Transactions on Software Engineering, 35(6), pp. 864-878.
[54] Selby, R. W., and Basili, V. R., 1991, "Analyzing Error-Prone System Structure," IEEE Transactions on Software Engineering, 17(2), pp. 141-152.
[55] Smith, R. P., and Eppinger, S. D., 1997, "Identifying Controlling Features of Engineering Design Iteration," Management Science, 43(3), pp. 276-293.
[56] Groeneveld, R. A., and Meeden, G., 1984, "Measuring Skewness and Kurtosis," Journal of the Royal Statistical Society. Series D (The Statistician), 33(4), pp. 391-399.
[57] Summers, J. D., and Shah, J. J., 2010, "Mechanical Engineering Design Complexity Metrics: Size, Coupling, and Solvability," Journal of Mechanical Design, 132(2).
[58] Henry, S. M., and Selig, C., 1990, "Predicting Source-Code Complexity at the Design Stage," IEEE Software, 7(2), pp. 36-44.
[59] Sommerville, I., 2007, *Software Engineering*, Addison-Wesley, New York, NY.
[60] Mccabe, T., 1976, "A Complexity Measure," IEEE Transactions on Software Engineering, 2(4), pp. 308-320.
[61] Nagappan, N., and Ball, T., 2007, "Using Software Dependencies and Churn Metrics to Predict Field Failures: An Empirical Case Study," *Proc. of the Empirical Software Eng. and Meas. Conf. (ESEM),* Madrid, Spain.
[62] Zimmermann, T., and Nagappan, N., 2009, "Predicting Defects with Program Dependencies," *Proceedings of the ACM-IEEE Empirical Software Engineering and Measurement Conference (ESEM)*, Orlando, FL,
[63] Eick, S. G., Graves, T. L., Karr, A. F., Marron, J. S., and Mockus, A., 2001, "Does Code Decay? Assessing the Evidence from Change Management Data," IEEE Transactions on Software Engineering, 27(1), pp. 1-12.
[64] Martin, R. C., 2002, *Agile Software Development*, Prentice Hall, Englewood Cliffs, NJ.
[65] Cameron, A. C., and Trivedi, P. K., 1998, *Regression Analysis of Count Data*, Cambridge University Press, Cambridge, U.K.
[66] Hausman, T., Hall, B. H., and Griliches, Z., 1984, "Econometric Models for Count Data with an Application to the Patents-R&D Relationship," Econometrica, 52(4), pp. 909-938.
[67] Neter, J., Wasserman, W., and Kutner, M. H., 1990, *Applied Liner Statistical Models Regression, Analysis of Variance, and Experimental Design*, Richard D. Irwin, Inc., Burr Ridge, IL.
[68] Kennedy, P. E., 2003, *A Guide to Econometrics*, MIT Press, Cambridge, MA.
[69] Shaw, M., and Garlan, D., 1996, *Software Architecture*, Prentice Hall, Upper Saddle River, NJ.

Table 3. Negative Binomial Regressions Predicting Expected Number of Defects per Product (*N* = 105)

	Model 1 Controls	**Model 2 Baseline**	**Model 3 IN-Skewness**	**Model 4 OUT-Skewness**	**Model 5 IN & OUT Skewness**	**Model 6 D_{in} = 0.15**	**Model 7 D_{in} = 0.25**	**Model 8 D_{out} = 0.15**	**Model 9 D_{out} = 0.35**
SLOC$_{is}$	–0.291	-0.448	-0.381	0.116	0.102	-0.010	0.112	0.221	-0.049
AVG_CC$_{is}$	0.217***	0.335***	0.337***	0.259***	0.266***	0.152	0.203***	0.192*	0.121*
AGE$_{is}$	0.224	0.252	0.017	0.001	-0.022	0.004	0.032	-0.132	-0.109
DAYS BEFORE$_{is}$	–0.010	0.026	0.079	0.153	0.151	0.107	0.123	0.062	0.133
DAYS AFTER$_{is}$	0.362***	0.294***	0.442***	0.469***	0.486***	0.368***	0.428***	0.417***	0.376***
NEWNESS$_{is}$	0.323***	0.259**	0.240**	0.248***	0.247***	0.279***	0.260***	0.276***	0.220***
IMPLICIT_BUGS$_{is}$	–0.175	-0.297**	-0.320***	-0.319***	-0.324***	-0.319***	-0.335***	-0.309***	-0.358***
CUM_CHANGES$_{is}$	–0.149	-0.152	0.052	-0.160	-0.124	-0.207	-0.272*	0.076	-0.041
AVG_BREADTH_CHANGE$_{is}$	0.171	0.118	0.104	0.212	0.207*	0.303***	0.380***	0.378***	0.306***
AVG_INTERFACE_USAGE$_{is}$	–0.112	0.071	0.272	0.389**	0.407*	0.174	0.131	0.094	0.054
NUM_MODULES$_{is}$	0.132	0.246	0.227	-0.290	-0.258	-0.238	-0.381	-0.378	-0.345
PROPAGATION COST$_{is}$	–0.470*	-0.274	-0.446*	-0.352**	-0.374*	-0.157	-0.260*	-0.415**	-0.555***
Intrinsic *NUM_LOOPS*$_{is}$	–0.063	-0.033	-0.050	-0.206*	-0.207*	-0.298**	-0.272**	-0.320***	-0.241***
Intrinsic *CYCLICALITY* ($P_{I,is}$)	0.552*								
CLUSTERING$_{is}$		0.026	0.267	-0.188	-0.119	-0.250	-0.336	0.073	-0.008
In-Out degree correlation$_{is}$		0.576**	0.644**	0.732**	0.713**	0.354	0.955***	0.396	0.992***
Out-out neighboring degree correlation$_{is}$		0.372**	0.302*	0.052	0.067	-0.081	-0.222	0.159	-0.332
In-degree skewness$_{is}$			-0.544**		-0.092	0.079	0.063	-0.271	-0.107
Out-degree skewness$_{is}$				-0.882***	-0.843***	-0.962***	-0.928***	-0.781***	-0.857***
$\boldsymbol{Q_{Din,\ is}}$						-1.103***	-0.808***		
$\boldsymbol{Q_{Dout,\ is}}$								-1.183***	-1.943***
Log likelihood	–386.212	-381.457	-378.935	-374.460	-374.404	-368.924	-369.562	-367.900	-360.810

Standardized coefficient estimates. *< .1 ** < .05 *** < .01 (two-tailed).
All models include product and year specific fixed effects.

Table 4. Negative Binomial Regressions Predicting a U-shape Relationship Between Fraction of Hub Components and Expected Number of Defects (N = 105)

	Model 1 D_{in} **= 0.15**	**Model 2** D_{in} **= 0.25**	**Model 3** D_{in} **= 0.30**	**Model 4** D_{in} **= 0.40**	**Model 5** D_{in} **= 0.45**		**Model 6** D_{out} **= 0.15**	**Model 7** D_{out} **= 0.20**	**Model 8** D_{out} **=0.30**
$SLOC_{is}$	-0.168	-0.047	-0.012	0.165	0.145		0.273	0.305	0.137
AVG_CC_{is}	0.249**	0.195***	0.270***	0.318***	0.348***		0.208***	-0.042	0.128*
AGE_{is}	-0.012	0.039	0.024	0.103	0.147		-0.375*	-0.033	-0.123
$DAYS\ BEFORE_{is}$	0.112	0.105	0.091	0.139	0.091		0.174*	0.257**	0.087
$DAYS\ AFTER_{is}$	0.371***	0.426***	0.437***	0.373***	0.390***		0.414***	0.465***	0.394***
$NEWNESS_{is}$	0.307***	0.251***	0.251***	0.264***	0.260***		0.226***	0.265***	0.272***
$IMPLICIT\_BUGS_{is}$	-0.406***	-0.348***	-0.362***	-0.345***	-0.345***		-0.128	-0.452***	-0.374***
$CUM\_CHANGES_{is}$	-0.245*	-0.268*	-0.279*	-0.293*	-0.284**		0.172	-0.292**	-0.039
$AVG\_BREADTH\_CHANGE_{is}$	0.375***	0.338***	0.300**	0.349***	0.324***		0.399***	0.459***	0.472***
$AVG\_INTERFACE\_USAGE_{is}$	0.206	0.116	0.162	0.272	0.256		-0.252	0.194	0.041
$NUM\_MODULES_{is}$	-0.124	-0.315	-0.244	-0.402	-0.387		-0.543**	-0.690***	-0.509*
$PROPAGATION\ COST_{is}$	-0.181	-0.169	-0.303*	-0.413**	-0.399**		-0.162	-0.604***	-0.572***
Intrinsic NUM_LOOPS_{is}	-0.398***	-0.249**	-0.187	-0.167	-0.197		-0.321***	-0.292***	-0.347***
$CLUSTERING_{is}$	-0.296	-0.407	-0.044	-0.284	-0.349		-0.055	-0.025	-0.060
In-Out degree correlation$_{is}$	0.652*	1.062***	0.772**	0.826***	0.789***		-0.006	1.298***	0.928***
Out-out neighboring degree correlation$_{is}$	0.030	-0.216	0.004	0.064	0.177		-0.535**	-0.315	-0.372
In-degree skewness$_{is}$	-0.048	0.098	0.126	0.278	0.299		-0.399*	0.167	-0.066
Out-degree skewness$_{is}$	-0.873***	-0.875***	-0.700**	-0.832***	-0.850***		-0.717***	-1.186***	-0.891***
$\boldsymbol{Q_{D,is}}$	-1.790***	-1.266***	-0.991***	-0.942**	-1.098***		-4.234***	-2.079***	-2.305***
$\boldsymbol{Q_{D,is}\ SQ}$	0.425**	0.257**	0.186**	0.157**	0.142***		1.494***	0.549***	0.261*
Log likelihood	-365.932	-367.671	-370.679	-371.544	-370.086		-351.251	-360.743	-356.838

Standardized coefficient estimates. *< .1 ** < .05 *** < .01 (two-tailed)
All models include product and year specific fixed effects.

Appendix A. Table A1. Descriptive Statistics and Pairwise Correlations (N=105)

	Mean	STD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1. Number of defects (y_{is})	82.8	127.6	1.00																				
2. $SLOC_{is}$	19.4	17.9	.27	1.00																			
3. AVG_CC_{is}	2.6	2.7	.03	-.07	1.00																		
4. AGE_{is}	700.7	673.5	.26	.24	.07	1.00																	
5. *DAYS BEFORE*$_{is}$	212.1	232.2	.36	.07	.03	.57	1.00																
6. *DAYS AFTER*$_{is}$	259.4	272.3	.36	.06	.17	.36	.35	1.00															
7. $NEWNESS_{is}$	31.5	43.0	.53	-.04	.05	.10	.32	.17	1.00														
8. $IMPLICIT\_BUGS_{is}$	20.0	43.2	.25	.25	.01	.02	-.01	-.02	.11	1.00													
9. $CUM\_CHANGES_{is}$	1438.6	4532.4	.62	.17	-.04	.22	.36	.29	.66	.08	1.00												
10.$AVG\_BREADTH\_CHANGE_{is}$	3.9	3.2	.14	.06	-.15	.17	.18	.12	-.15	.15	.01	1.00											
11.$AVG\_INTERF\_USE_{is}$	0.1	0.1	.27	.66	-.12	.19	.14	.17	-.04	.20	.12	.16	1.00										
12. $NUM\_MODULES_{is}$	26.6	25.9	.14	.75	-.08	.11	.02	.09	-.03	.44	.08	.08	.47	1.00									
13. $PROPAGATION_{is}$	16.3	9.3	.00	-.07	-.07	-.17	-.16	-.11	.14	-.18	.00	-.03	.04	-.22	1.00								
14. Intrinsic NUM_LOOPS_{is}	3.9	2.6	.29	.65	-.08	.42	.07	.11	-.09	.28	.13	.22	.31	.45	-.14	1.00							
15. Intrinsic cyclicality ($P_{I,is}$)	19.1	10.6	.30	.30	-.01	.02	-.03	.00	.17	-.02	.03	.03	.27	.01	.78	.21	1.00						
16. $CLUSTERING_{is}$	0.2	0.0	.32	.05	.03	-.11	-.10	-.08	.29	-.01	.19	-.01	.10	-.22	.57	-.02	.67	1.00					
17. In-Out degree correlation$_{is}$	0.0	0.1	.39	.34	.05	-.02	-.02	.04	.29	.15	.07	.07	.28	.22	.47	.19	.75	.51	1.00				
18. Out-out neighboring degree correlation$_{is}$	-0.0	0.1	-.06	.11	-.06	.13	.07	-.01	-.02	.16	-.02	.01	-.19	.13	-.33	.09	-.30	-.35	-.37	1.00			
19. In-degree skewness$_{is}$	4.8	2.5	.27	.38	-.12	.00	.17	.29	.18	.21	.40	.24	.38	.34	-.11	.28	.05	.09	.18	.07	1.00		
20. Out-degree skewness$_{is}$	3.1	2.3	-.08	.19	-.14	-.05	.14	.09	-.20	-.04	-.08	.15	.48	.07	.01	.03	.08	-.13	.09	-.29	.46	1.00	
21. $\boldsymbol{Q}_{25\ IN\,is}$	0.8	1.2	-.12	-.37	-.04	-.33	-.17	-.13	-.02	-.17	.03	.04	-.14	-.31	.40	-.30	.05	.37	-.10	-.37	-.02	-.05	1.00
22. $\boldsymbol{Q}_{25\ OUT,is}$	0.9	2.1	-.25	-.36	-.07	-.31	-.21	-.20	-.15	-.12	-.13	-.07	-.31	-.26	.29	-.23	.02	.03	-.13	-.42	-.29	.02	.49

Correlations > |0.23 | are significant at p < .01