Contents lists available at ScienceDirect

Information & Management

journal homepage: www.elsevier.com/locate/im

On the time lag of the effect of network position on service performance in software service networks

Kibae Kim[a], Jörn Altmann[b,*], Wonjoon Kim[a]

[a] Korea Advanced Institute of Science and Technology, Seoul, South Korea
[b] Seoul National University, Seoul, South Korea

ARTICLE INFO

Keywords:
Lagged position effect
Software service performance
Knowledge complementarity
Innovation study
Software-as-a-service
Platforms

ABSTRACT

We examine whether a time lag exists before the network position of a software service affects its performance. Moreover, we analyze different time lags, using empirical data about software services and their usage for creating composite services. Our results show that software services in central positions (i.e., high betweenness centrality) attract users the most. The highest effect exhibits, if the time lag is 26–32 months. Our findings are relevant, as they can guide developers in marketing their software services and are expected to impact innovation studies regarding the importance of considering time lags and analyzing complementary knowledge.

1. Introduction

In the software services industry, newly emerging business models not only attract users but also motivate third-party software developers to invent and offer new software services. This software services industry consists of platforms (i.e., an environment comprising of a number of interoperable software services, communication standards, and deployment tools), software services that use a platform, platform providers, software service providers that might also be platform providers or third-party software service developers, and users. An earlier study defined this software services industry as a "dynamic value-co-creation configuration of resources (i.e., people, organizations, shared information (language, laws, measures, methods), and technology) that are all connected internally and externally by value propositions" [1]. Within this industry, software services can also be reused for generating new software services. Any agent in the industry can employ tangible and intangible resources, to develop new software services that might be very specific to a user [2], making the agent a third-party software service developer. Although it is important to understand the software services industry from the perspective of a network of participating agents (i.e., the dependencies between platform providers, software developers, and users), it is difficult to find studies that examine software services from a network structural perspective.

In this study, we address this shortcoming and examine the effect of the network position on software service performance, especially focusing on the time lag of the effect. Fortunately, a great deal of literature on innovation studies examines network characteristics and their effect on firm or innovation performance, including academic collaboration networks [3], open-source project networks [4], and employee social networks within an organization [5]. They assume that an agent with a "good" position can gain useful knowledge. Moreover, the more an agent is embedded in a network, the more innovative the agent is [6,7]. Nonetheless, an agent, who has a low number of links but interconnects clusters of highly connected agents, could also be creative [8,9]. This agent has a weak tie position but bridges clusters.

However, the limitations of these previous studies are that they focus on performing cross-sectional analyses of the relationship between network position and innovation performance [5,10–15]. The analyses in those studies take the view that the network position affects the innovation performance instantaneously, although the relationship between network position and innovation performance changes over time in many scenarios and, due to the potential existence of a diffusion process, the effect of the network position might be noticeable with a time lag. Therefore, it is advised to check carefully whether a time lag between the network position and its effects on innovation performance exists. In other words, a more reasonable assumption is the existence of a diffusion process within a network [16]. That is, a new technology is adopted by a few adopters during the initial period, its adoption rate grows during the middle of its lifetime, and adoption declines after the technology reaches maturity. It has been observed in products [17], enterprises [18], and industries [19]. Moreover, the network position of a software service also inhibits a life-cycle behavior [20]. Therefore, without considering the time lag due to the diffusion within a network, cross-sectional analysis of network position and performance could be

* Corresponding author at: Gwanak-Ro 1, Gwanak-Gu, Seoul, 08826, South Korea.
E-mail address: jorn.altmann@acm.org (J. Altmann).

https://doi.org/10.1016/j.im.2019.02.004
Received 16 August 2016; Received in revised form 7 February 2019; Accepted 13 February 2019
Available online 14 February 2019

imprecise.

Therefore, in this paper, our research objective is to investigate whether a time lag between the effects of network positions on service performance in software service networks is noticeable. In other words, we examine how the position in a network affects service performance with a delay. To answer this research question, we gather software service network data from ProgrammableWeb for the period September 2005–April 2014 and performance data from Alexa for April 2014. In the software service network constructed, we measure the network position (i.e., degree centrality and betweenness centrality) of software services in the network for each month since its initiation [21]. Using these data, we perform multivariable regressions for different time lags, ranging from 0 to 36 months. By comparing the goodness-of-fit for each of these regression models, we find the best-fitting model and, consequently, the time lag, for which the effect of network position on service performance is the strongest.

Based on these regression analyses, two significant results can be identified. Our first result shows that, with high statistical significance, service performance increases as betweenness centrality increases. That is, network positions of software services, especially positions that mediate distant software services (i.e., a high normalized betweenness centrality), affect their service performance. Here, it is to be noted that our results do not indicate that service performance depends on degree centrality. This is plausible as our software service network is constructed based on the complementarity of software services and not the knowledge flow between software services. Our second result shows that the relationship between service performance and the network position varies depending on the time lag considered. Specifically, the effect of network position on service performance is most dominant if the time lag is 26–32 months. This reveals that a time lag exists before the effects of a network position of software services on service performance can be noticed.

Our findings are relevant, as they imply that, if explaining service performance through network positions, innovation studies should also consider the time lag of effects of factors. The time lag needs to be considered, if the diffusion theory and the life-cycle theory are considered. Furthermore, the findings will help software service providers to sell their services more effectively, as our study provides an indication for what kind of software services could be combined.

2. Theoretical background

2.1. Open innovation of software services

As software is provided as a service, the pattern of software innovation has changed. Although, at the outset, software is developed by a company with its own resources, third-parties now also participate in the software innovation process [22]. In this new paradigm of innovation, third-parties can combine existing software services and add their own special functions to create new software services. These new software services are called "composite services" [23] or "mashups" [24]. This new style of innovation can be seen as the software industry's version of open innovation. Open innovation is defined as an innovation process, in which an agent (e.g., company) shares its knowledge with other agents (e.g., companies) and utilizes the combined knowledge for the purpose of innovation [25]. In this process, a third-party software service developer can participate in the innovation of new products, services, and business models of a software company. At the same time, the third-party developer reduces the cost of developing its own software services from scratch by utilizing the innovation resources (i.e., platforms or software services) of other software service providers. Furthermore, a software service provider might provide third-party software service developers with an open platform, in which those developers can easily participate in the innovation process and interact with their users at a low cost [26].

If open innovation of software services works well, the variety of software services developed in this process will attract more customers. Then, both the software service provider and the third-party software service developers will benefit from the contact with users. This is part of the value creation of software service providers with an open innovation strategy. Third-party developers utilize the open innovation resources, and users use the software services and the software service platform [27–30]. As a software service provider also obtains benefit indirectly through its innovation partners, it needs to employ strategies that also support third-party developers in their efforts to achieve more innovation [26]. This is the reason that a lot of innovation studies investigate the structure of software service networks and the behavior patterns of third-party developers [28,31,32].

2.2. Effect of network position on innovation performance

Innovation is a process of reusing and "recombining past ideas, artifacts, and people" for solving problems [33,34]. An individual obtains knowledge through its relationships with other individuals (i.e., colleagues and friends), organizes the new knowledge, and applies the knowledge to resolve the problem that is at hand. This assumes that better knowledge leads to better performance. Previous innovation studies take this idea and investigate the relationship between the position of innovation agents in a social network and their innovation performance [4,7,9,15,35,36].

Intuitively, the best position is the one, in which an agent has high connectivity with its neighbors. That is, the innovation performance of an agent increases as it gains more connections with its neighbors [6,7]. The research in this stream assumes further that the agent obtains more knowledge through its deep embeddedness in its society and that the knowledge can be reused and recombined. However, the effect of direct connectivity does not always result in better innovation as the maintenance of a connection comes with a cost [37]. For example, an agent with a high connectivity can probably focus little on recombining what it learns from its neighbors. Moreover, the innovation of an agent deeply embedded in its society may not be so large, as this innovation might be similar to that of its neighbors [9]. Instead, an agent can achieve better innovation, if it takes a position, in which it can mediate its neighbors [8,35].

Not only people but also products and services can be connected [11,13,20,27,28,32]. The research on software services assumes that, if two software services are used together for developing a new composite software service, they are connected [20,28]. A connection between two software services means that they have been reused and recombined. Following this idea, software service networks can be analyzed using the same mechanism as for social networks. Therefore, it can be investigated whether the network position of a software service affects its service performance. In more detail, it could be investigated whether the service performance of a software service could depend on its embeddedness (i.e., its connection mediation between other software services) in a software service network.

2.3. Effect of life cycle and diffusion process on network position

Previous innovation studies on the relationship between network position and performance are cross-sectional [4,7,15,35,36]. This is usually caused by the limitation of the methodology applied, especially if the data are gathered using a questionnaire survey. For example, Sasidharan et al. [15] used a questionnaire survey to gather information on social relationships among people and their adoption of new technologies. The model does not consider that the social relationship changes over time and that it takes time to adopt a technology. Another example is the model of Everard and Henry [35], which is also cross-sectional. The authors collected information on interlocked directorates of firms and firm performance (i.e., the visibility of firms in the *Wall Street Journal*) in 2000. However, they did not consider that it takes time before the actions of directorates show an effect. As these examples of previous studies suggest, a time lag might exist before the effect of network position on service performance can be noticed. The time lag is the result of a diffusion process, which might be at work before an improvement of the performance can be observed. In other words, a more reasonable assumption is the assumption of the potential existence of a diffusion process of a technology within a network, following the theory of diffusion [16].

As the network position and the service performance vary over time, the relationship between the network position and the service performance could also change. This statement is supported by the well-known fact that life cycles can be identified for products [17], enterprises [18], and even industries [19]. A life cycle can always be described in terms of phases, in which rapid growth occurs in the beginning, prosperity is maintained over an intermediate period, and then prosperity declines. For example, sales quantity shows a bell-shaped curve over time [16]. With respect to software service networks, the position of a node can also change in an evolving network. For example, Kim et al. [20] found that the centralities of software services change over time. That is, nodes approach the center of a network and, then, retreat to the periphery of the network.

Therefore, considering that the network position of software services changes over time and that a diffusion process might exists, it is worth to investigate whether an effect of the network position of software services on service performance exists with a time lag. Without this knowledge, it could happen that, if the network position of an agent declines and its service performance increases, a cross-sectional analysis overestimates their relationship. Following this idea, we suggest investigating the following hypothesis with respect to software services:

Hypothesis 1. *A time lag exists before the network position of a software service affects its service performance.*

3. Methodology

3.1. Research model

Our research model, presented in Fig. 1, captures the effect of the network position of a software service on the software service's performance over a certain time period. The network position is determined according to either the number of direct connections of a node with its neighbors (i.e., degree centrality) or the level of its connection mediation between distant neighbors (i.e., betweeness centrality).

As the revelation of the service performance of a software service could be delayed after the network position of a node has been fixed, we consider different time lags, ranging from 0 to T time periods. For example, if the service performance reveals itself at the same time period as the network position, the time lag will be $t = 0$. If it appears one time period later, the time lag will be $t = 1$. If it appears two time periods later, the time lag will be $t = 2$. We consider all time lags up to $t = T = 36$.

3.2. Data

The data on the service performance of software services were collected from Alexa (www.alexa.com), which provides commercial web traffic data analysis. Alexa estimates the traffic of each domain according to a global traffic panel, which is sampled from millions of Internet users. It shows the ranking of a website according to a variety of indicators, including "reach." Reach denotes the number of users that visit a website per day. We gathered the traffic data on different subdomains for April 2014.

The software service network data were collected from ProgrammableWeb (www.programmableweb.com), a website that publishes information about software services and composite services (mashups). The data collected include the names of composite services developed between September 2005 and April 2014 and the software services used to develop the composite services. Using the data, we define a service network as a set of nodes, which represents software services, and a set of links between these nodes. A link between a pair of software services indicates that the two software services have been used jointly in a composite service. A service network is weighted, and the value of a link between two software services represents the number of composite services developed with these two software services. Although a composite service can also be re-used through opening its API to third parties, the data set considered does not contain such a case.

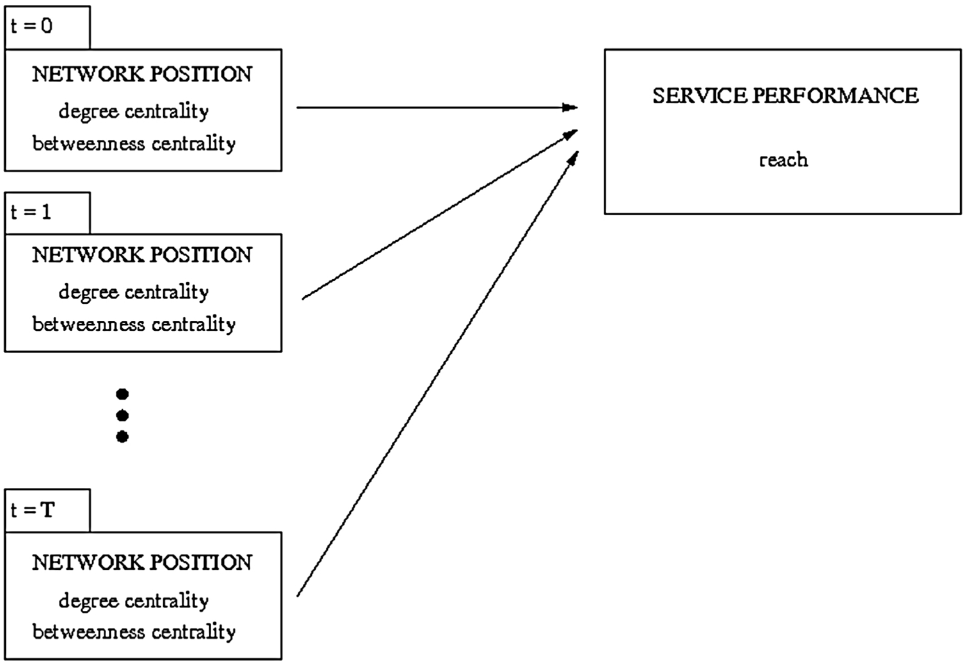

Fig. 1. Research model.

3.3. Dependent variables

Prior research proposes a variety of ways to measure service performance, such as financial performance, degree of affiliation, technical or commercial success, and task performance of individuals or teams [4,15]. In this paper, the service performance of a software service is measured through the indicator "reach," which is defined as the number of people visiting a software service within a certain time period. When those visiting people use a composite software service, they also pass through those software services that are reused by the composite software service. In this case, a visiting user is counted for the composite service and the specific software services reused. This measure represents the value of the software service to users. For our analysis, the "reach" of each software service has been measured for the month of April 2014.

3.4. Independent variables

The independent variables are two indicators for network position: degree centrality and betweenness centrality [21]. Degree centrality is defined as the number of links of a node [38], which indicates how many neighbors are directly connected to it. The mathematical definition of this indicator is given in the Appendix A. This definition of degree centrality is slightly modified, if the links are weighted. That is, degree centrality for a weighted graph is used to estimate the strength of collaborations with neighbors [39] and is also called "strength" [38]. Furthermore, as the degree centrality is likely to be dependent on the network size and, in order to avoid a bias according to the network size, the degree centrality is also normalized by the maximum possible number of links of a node (see Appendix).

In addition, the betweenness centrality measures the extent, to which a node interconnects other nodes in the network by being on the shortest path between any two nodes in the network [35,38]. The mathematical definition of betweenness centrality is given in the Appendix. Betweenness centrality indicates how many neighbors a node mediates. Like degree centrality, betweenness centrality of a node is normalized by the maximum possible number of paths that crosses the node.

3.5. Control variables

In order to be able to focus on the effect of the network position of a software service on the performance of a software service, we control three variables that might affect the service performance. First, the performance of software services could depend on the age of software services, as indicated in theories on the business life cycle [16–19]. To capture the age of software services, we use the time since the software service has been introduced the first time.

Second, software service providers often offer their own development and deployment environments (i.e., open platforms) for service developers, in order to promote the reuse of the provider's software services by third-party service developers [32]. Both the network position of a software service and its service performance could be biased, if the software service is part of a large platform. In order to avoid this bias, we also use the size of the software service platform (platform size) as a control variable. The platform size defines the number of interoperable software services that a provider releases on its platform [40]. For the purpose of our analysis, interoperable software services are only those software services that have been released by the same software service provider.

Third, software services are grouped into service categories. Popular service categories (e.g., mapping and social networking) attract an extraordinarily large number of users, whereas other categories are used by only a few users. Therefore, we also use the software service categories as a set of dummy control variables to control for their effects.

3.6. Method of analysis applied

We implement linear regressions of software service reach using the degree centrality and the betweenness centrality with different time lags and control variables. To express this in a mathematical formula, we define $Y_{i,0}$ as the logarithm of reach of software service i at time $t = 0$, and $X_{i,t}$ and $Z_{i,t}$ as the logarithm of the normalized degree centrality and the logarithm of the normalized betweenness centrality, respectively. They are measured for software service i at time period t, which denotes t time periods in the past.

Moreover, we consider three control variables, which are invariant across different time lags. We define $\Omega_{i,c}$ as the control variables of software service i. For our analyses, we consider the platform size $\Omega_{i,1}$ (i.e., the number of software services that the provider of a focal software service had released) and the age of a software service $\Omega_{i,2}$ (i.e., the number of time periods in months since the first release of the software service). $\Omega_{i,3}^{j}$ defines a set of dummy variables that represents different software service categories j (e.g., shopping, social, and search).

Depending on the combinations of those variables, four different models (Model 1–4) are considered for the analysis and are represented with the following four equations (Eqs. (1)–(4)). Model 1 contains only control variables $\Omega_{i,1}$, $\Omega_{i,2}$, and $\Omega_{i,3}^{j}$ (Eq. (1)). Degree centrality $X_{i,t}$ and betweenness centrality $Z_{i,t}$ are, respectively, added in Model 2 (Eq. (2)) and Model 3 (Eq. (3)). Model 4 involves all variables.

$$Y_{i,0} = \alpha + \delta_1\Omega_{i,1} + \delta_2\Omega_{i,2} + \delta_3\Omega_{i,3}^{j} + \varepsilon_{i,t} \tag{1}$$

$$Y_{i,0} = \alpha + \beta X_{i,t} + \delta_1\Omega_{i,1} + \delta_2\Omega_{i,2} + \delta_3\Omega_{i,3}^{j} + \varepsilon_{i,t} \tag{2}$$

$$Y_{i,0} = \alpha + \gamma Z_{i,t} + \delta_1\Omega_{i,1} + \delta_2\Omega_{i,2} + \delta_3\Omega_{i,3}^{j} + \varepsilon_{i,t} \tag{3}$$

$$Y_{i,0} = \alpha + \beta X_{i,t} + \gamma Z_{i,t} + \delta_1\Omega_{i,1} + \delta_2\Omega_{i,2} + \delta_3\Omega_{i,3}^{j} + \varepsilon_{i,t} \tag{4}$$

where α is a constant, β, γ, and δ_c are coefficients of variables $X_{i,t}$, $Z_{i,t}$, and $\Omega_{i,c}$, respectively, and $\varepsilon_{i,t}$ is the error term.

4. Results of analysis

4.1. Descriptive analysis

As of April 31, 2014, ProgrammableWeb (www.programmableweb.com) listed more than 2000 software services, of which around 1000 were used to develop composite services. Of these software services, 179 software services were most frequently used (i.e., more than 10 uses) for composite service development and were selected for the analysis. However, traffic data on "reach" for 54 software services were not available at Alexa (www.alexa.com). Therefore, only 125 software services could be used for our analysis.

The dependent and independent variables are highly skewed (Fig. 2a and b). That is, most of the software services have low values for reach, degree centrality, and betweenness centrality. Only a few software services have very high values. This is not unusual for these types of networks. For example, the degree distribution decays by a power function in many large complex networks [9,35]. However, this highly skewed distribution of variables makes any statistical analysis difficult. It means that the results of our analysis are likely to depend on a few outliers in the low-value area. In order to reduce this problem, the variables are transformed with a logarithmic function. Through the logarithmic transformation, the data are distributed in log–log scales over a large area (Fig. 2c and d), making outliers less influential. Therefore, our regression based on these logarithmic transformed variables becomes useful.

Fig. 3 depicts the descriptive statistics for some variables: the number of available values (solid line, right-hand side y-axis), the mean of the logarithm of the normalized degree centrality (dashed line, left-

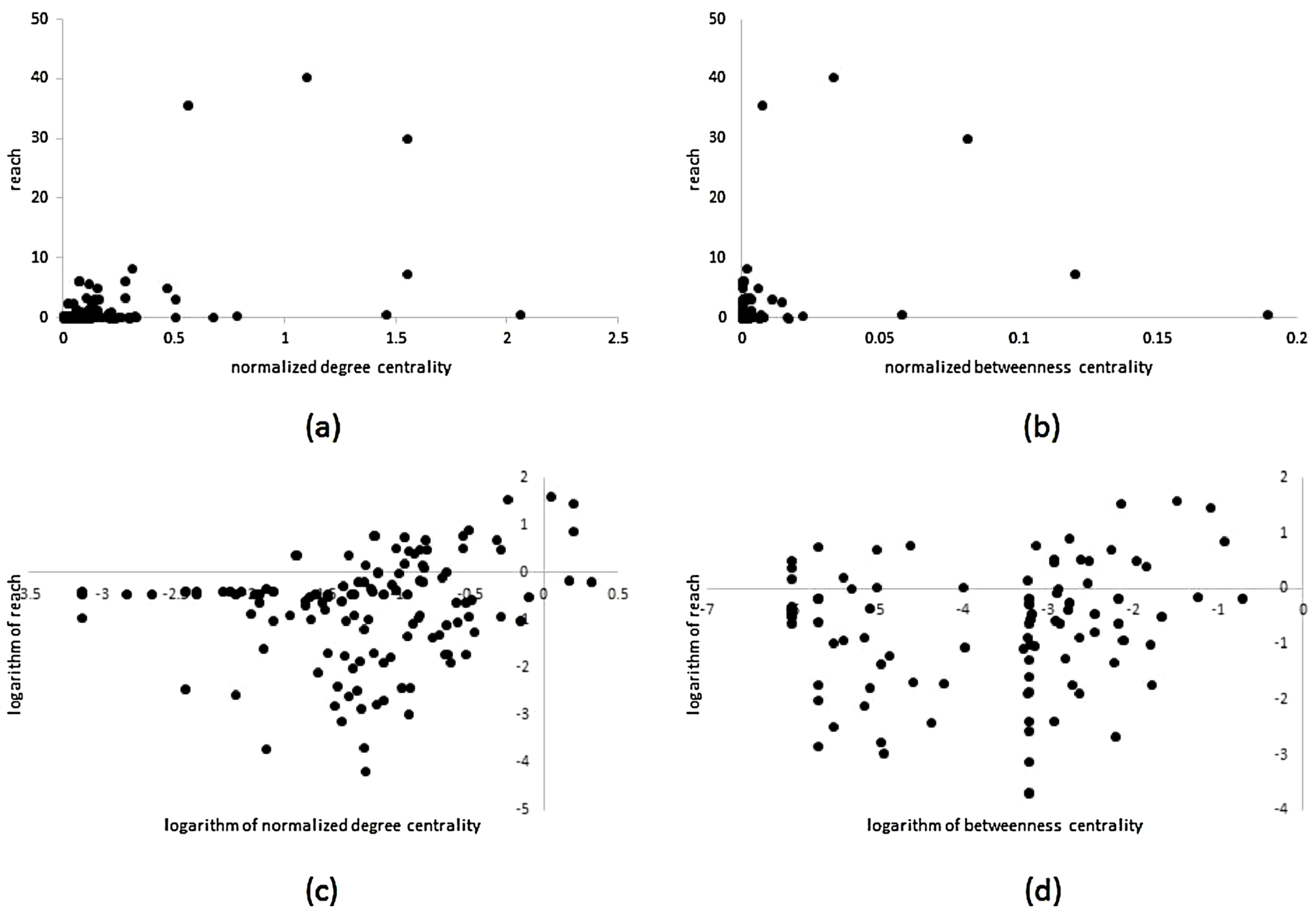

Fig. 2. Scattogram (a) between reach (i.e., the number of people visiting the web site on April 2014) and normalized degree centrality in linear–linear scales, (b) between reach and normalized betweenness centrality in linear–linear scales, (c) between reach and normalized degree centrality in log–log scales, and (d) between reach and normalized betweenness centrality in log–log scales.

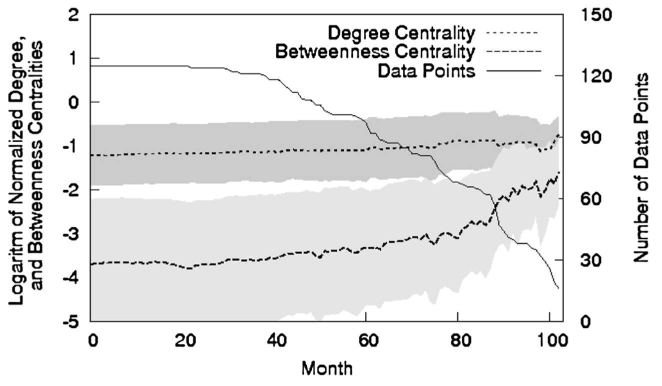

Fig. 3. Trend of the descriptive statistics.

hand side y-axis), and the mean of the logarithm of the normalized betweenness centrality (dotted line, left-hand side y-axis). Gray areas surrounding the mean of the logarithm of the normalized degree centrality and the mean of the logarithm of the normalized betweenness centrality are one standard deviation from the mean of the values. The number of available data points per period remains stable at a value of 125 from the most recent time period ($t = 0$) to the time period of 29 months ago ($t = 29$). Afterward, it decreases gradually from 123 in 30th month to 16 in 102nd month (right-hand side y-axis in Fig. 3). This diminution of the number of variables is because of the fact that many software services did not exist in earlier periods. Another fact for the diminution of the number of data points is that its logarithm is not defined, if the normalized betweenness centrality is zero. For this reason, the number of available data points is 95 in the 0th month instead of the total number of surveyed software services (125). Considering the diminution of the number of cases for the different time lags, we only choose the most recent 36 months as our study period ($0 \leq t \leq 36$). During that time period, at least 80% of the cases are available.

Table 1 shows the descriptive statistics (i.e., mean and standard derivation) of the variables used for the analysis. The results show that, in average, software services were registered approximately 78 months ago. The average value of the logarithm of degree centrality is −1.17, and for the average logarithm of betweenness centrality, it is −2.68. The value for platform size is higher than the 125 software services considered for the regressions, as the platform size is calculated based on the entire population of more than 2000 software services.

The values of Pearson's correlations of variables of our model are

Table 1
Descriptive statistics of the variables used.

	Mean	SD	Min	Max
Logarithm of reach	−0.75	1.12	−4.16	1.6
Platform size	36.56	45.94	0	115
Age of software service	77.78	20.17	25	103
Logarithm of degree centrality (for most recent 36 months)	−1.17	0.65	−3.05	0.36
Logarithm of betweenness centrality (for most recent 36 months)	−2.68	2.85	−6	0

Table 2
Pearson correlation of variables used.

	a.	b.	c.	d.	e.
a. Logarithm of reach	–	–	–	–	–
b. Platform size	0.41**	–	–	–	–
c. Age of software service	−0.03	−0.11	–	–	–
d. Logarithm of normalized degree centrality	0.22*	−0.15	0.21*	–	–
e. Logarithm of normalized betweenness centrality	0.02	0.15	−0.08	−0.3**	–

* p < 0.05.
** p < 0.01.

Table 3
Distribution of services in service categories.

Service Category	No. of Services	Service Category	No. of Services
Shopping	19	Tools	2
Social	12	Utility	2
Internet	10	News	2
Search	9	Blog search	1
Music	8	Project management	1
Mapping	7	Answers	1
Video	5	Job search	1
Other	4	Events	1
Weather	3	Bookmarks	1
Storage	3	Payment	1
Email	3	Enterprise	1
Security	3	Recommendations	1
Telephony	2	Media management	1
Shipping	2	Wiki	1
Travel	2	Real Estate	1
Blogging	2	Medical	1
Advertising	2	Messaging	1
Reference	2	Education	1
Widgets	2	Entertainment	1
Personal information management	2	Sports	1

shown in Table 2. Although several pairs of variables (i.e., logarithm of reach and platform size; logarithm of reach and logarithm of degree centrality; age of software services and logarithm of degree centrality; logarithm of degree centrality and logarithm of betweenness centrality) have significant correlations at a 5% of significance level, the absolute values of their coefficients are less than 0.5 or larger than −0.5.

Table 3 describes the distribution of services into service categories in descending order. The service category with the highest number of software services is the shopping category (19 services), followed by the categories social services (12 services), Internet services that support the connection to the Internet (e.g., matching IP addresses to host names) (10 services), search services (9 services), music services (8 services), and mapping services (7 services). In addition to these 6 categories, there are 34 other software service categories comprising 1–5 services.

4.2. Analysis of existence of immediate effect of network position on service performance

For the analysis, we used 125 software services. However, because of the fact that the logarithm is not defined, if the betweenness centrality is zero, the number of available values is 95 in the 0th month instead of the total number of software services (125). With respect to the F-test statistics, the degree of freedom 1 represents the number of model variables (e.g., 42 for Model 1) except for the intercept. The degree of freedom 2 represents the sample size (125) minus the number of model variables (including the intercept) (e.g., 82 for Model 1).

As the results of the multivariable regressions for our four research models with no time lag (i.e., t = 0 in Eqs. (1)–(4)) show (Table 4), reach is positively associated with the control variable platform size (i.e., the number of software services that a provider releases). The coefficients of platform size are positive for all four models, and the linear relationship between reach and platform size is statistically significant ($p < 0.05$). The relationship between reach and the age of software service (i.e., the second control variable) is not statistically significant for Models 3 and 4. Among the service category dummy variables, only three category variables are significant. The entertainment services category variable is significant for all models. The weather services category variable and wiki services category variable are significant for Models 1 and 2. All coefficients are negative.

Based on these results, we can state that the performance of a software service is likely to increase, if the software service is released by a large provider, independent of how old the software service is. Moreover, the performance of software services in some service categories (i.e., entertainment services, weather services, and wiki services) could be lower than other software services due to their service category characteristics.

The results show, equivalent to previous studies [6,7], the service performance of a software service depends on its embeddedness in the software service network. In our models, the embeddedness is represented through the logarithm of normalized degree centrality and through the logarithm of normalized betweenness centrality. The service performance is represented through the logarithm of reach. In detail, our results for Model 2 (Table 4) show that the coefficient of the logarithm of normalized degree centrality is positive, and its relationship with the logarithm of reach is statistically significant ($p < 0.05$). Similarly, the coefficient of the logarithm of normalized betweenness centrality is significantly positive ($p < 0.01$) for Model 3. However, those two centralities are insignificant if they are considered in one model (Model 4). The adjusted R^2 values for the models are between 0.32 and 0.41, and their F-tests are all significant ($p < 0.01$). We can state that these measured adjusted R^2 values are acceptable, as our constructed software service network is similar to research fields that accepted even lower R^2 values [41]. In these research fields, constructs such as attitude, beliefs, and judgements of individuals are measured. Falk and Miller [42] even argued that R^2 values that are equal or greater to 0.10 are adequate for particularly endogenous structures. Therefore, the four models explain the service performance well.

The results suggest that only one of the normalized degree centrality and the normalized betweenness centrality might affect the service performance and that, as they correlate with each other (Table 2), the interference between the two variables results in eliminating their effect on service performance. However, a positive joint effect of both normalized centralities could be present if a time lag exists. Therefore, when investigating a time lag before either one of the centralities affects the service performance (i.e., $t \neq 0$ in Eqs. (2)–(4)), Models 2, 3, and 4 need to be analyzed as well. That is, it will reveal the centrality variables that influence the service performance with a time lag.

4.3. Analysis of existence of a time lag before network position affects service performance

In order to test whether the network position affects the service performance with a time lag, we conduct regressions with the same variables as in the previous section but with different time lags. We use Models 2, 3, and 4 (Eqs. (2)–(4)) with 37 different time lags, ranging from t = 0 to t = 36. That is, the regression equations include the age of software service, the platform size, the dummy control variables, as well as the logarithm of the normalized degree centrality and the logarithm of the normalized betweenness centrality for the network accumulated from the beginning until *t* months before the final time period (April 2014). In detail, time *t* specifies the time lag (i.e., the number of months) before the normalized degree centrality and the normalized betweenness centrality affect the performance of the

Table 4
Cross-sectional analysis results considering no time lag.

	Model 1	Model 2	Model 3	Model 4
Degree	–	0.376 (0.160) **	–	0.422 (0.429)
Betweenness	–	–	0.235 (0.077) ***	0.144 (0.120)
(Intercept)	−0.031 (0.870)	0.623 (0.892)	0.514 (1.211)	0.868 (1.264)
Platform size	0.008 (0.003) ***	0.008 (0.003) ***	0.010 (0.004) ***	0.009 (0.004) **
Age of software service	−0.010 (0.005) **	−0.012 (0.005) **	−0.008 (0.006)	−0.010 (0.006)
Cat_Answers	1.042 (1.127)	0.836 (1.101)	1.548 (1.283)	1.234 (1.323)
Cat_BlogSearch	−0.209 (1.146)	−0.525 (1.124)	−0.341 (1.320)	−0.578 (1.342)
Cat_Blogging	−1.012 (0.921)	−1.223 (0.901)	−1.246 (1.084)	−1.246 (1.084)
Cat_Bookmarks	0.162 (1.146)	−0.215 (1.127)	−0.141 (1.323)	−0.385 (1.346)
Cat_Education	−1.911 (1.168)	−1.679 (1.141)	−1.850 (1.347)	−1.506 (1.392)
Cat_Email	0.525 (0.847)	0.517 (0.825)	–	–
Cat_Enterprise	0.105 (1.141)	0.095 (1.110)	−0.096 (1.308)	0.042 (1.315)
Cat_Entertainment	−3.268 (1.195) ***	−3.154 (1.164) ***	−3.152 (1.380) **	−2.939 (1.397) **
Cat_Events	−0.628 (1.146)	−0.862 (1.120)	−0.434 (1.317)	−0.709 (1.347)
Cat_Internet	−1.085 (0.749)	−1.050 (0.730)	−0.742 (1.073)	−0.819 (1.076)
Cat_JobSearch	0.652 (1.146)	0.514 (1.117)	0.429 (1.320)	0.421 (1.321)
Cat_Mapping	−0.356 (0.747)	−0.588 (0.734)	−0.331 (0.964)	−0.445 (0.972)
Cat_MediaManage.	1.230 (1.147)	1.086 (1.118)	1.716 (1.324)	1.436 (1.355)
Cat_Medical	−0.590 (1.132)	0.010 (1.131)	–	–
Cat_Messaging	−1.174 (1.179)	−1.499 (1.156)	−1.207 (1.363)	−1.443 (1.384)
Cat_Music	−0.804 (0.768)	−0.900 (0.749)	−0.929 (1.053)	−0.999 (1.056)
Cat_News	0.609 (0.956)	0.398 (0.935)	0.958 (1.175)	0.677 (1.209)
Cat_Other	−0.345 (0.802)	−0.205 (0.783)	−0.150 (1.139)	−0.077 (1.142)
Cat_PIM	0.200 (0.930)	0.463 (0.912)	0.769 (1.298)	0.929 (1.308)
Cat_Payment	1.306 (1.135)	1.140 (1.107)	0.881 (1.299)	0.938 (1.300)
Cat_Photos	−0.302 (0.752)	−0.461 (0.735)	−0.409 (1.021)	−0.656 (1.051)
Cat_ProjectManage.	0.151 (1.146)	0.299 (1.118)	0.573 (1.322)	0.644 (1.325)
Cat_RealEstate	0.588 (1.148)	0.515 (1.118)	0.467 (1.323)	0.507 (1.324)
Cat_Recommend.	0.775 (1.152)	0.513 (1.127)	0.712 (1.329)	0.528 (1.342)
Cat_Reference	0.406 (0.950)	0.102 (0.934)	0.285 (1.171)	0.073 (1.191)
Cat_Search	0.550 (0.722)	0.314 (0.710)	0.536 (0.957)	0.389 (0.969)
Cat_Security	0.237 (0.844)	0.357 (0.823)	0.787 (1.269)	0.732 (1.271)
Cat_Shipping	0.577 (0.944)	0.554 (0.919)	0.818 (1.160)	0.760 (1.162)
Cat_Shopping	−0.447 (0.755)	−0.525 (0.735)	−0.227 (1.026)	−0.348 (1.034)
Cat_Social	0.269 (0.728)	0.088 (0.713)	0.831 (1.028)	0.573 (1.061)
Cat_Sports	−0.883 (1.158)	−0.303 (1.154)	–	–
Cat_Storage	−0.334 (0.868)	−0.518 (0.849)	−0.533 (1.103)	−0.568 (1.104)
Cat_Telephony	−0.377 (0.939)	−0.505 (0.916)	−1.324 (1.353)	−1.507 (1.366)
Cat_Tools	−0.571 (0.930)	−0.510 (0.906)	−0.133 (1.112)	−0.055 (1.115)
Cat_Travel	−0.273 (0.949)	−0.152 (0.926)	−0.363 (1.169)	−0.111 (1.197)
Cat_Utility	−0.001 (0.922)	0.145 (0.900)	0.118 (1.255)	0.159 (1.256)
Cat_Video	0.868 (0.785)	0.606 (0.772)	1.008 (1.019)	0.767 (1.048)
Cat_Weather	−1.796 (0.860) **	−1.834 (0.838) **	−1.693 (1.139)	−1.731 (1.140)
Cat_Widgets	0.105 (0.936)	0.300 (0.915)	−0.065 (1.322)	0.013 (1.325)
Cat_Wiki	−1.995 (1.152)*	−2.054 (1.122)*	−1.401 (1.333)	−1.611 (1.351)
adj. R^2	0.323	0.358	0.408	0.408
F-static	2.41***	2.611***	2.622***	2.58***
Degree of freedom1	42	43	40	41
Degree of freedom2	82	81	54	53

* $p < 0.10$.
** $p < 0.05$.
*** $p < 0.01$.

software service (i.e., reach). As we measured the reach of software services in April 2014, a value of $t = 4$, for example, states that the normalized degree centrality value and the normalized betweenness centrality value of December 2013 are used.

The results of the regressions are shown in Table 5. For readability, we display the regression results (coefficient, standard error, and significance) of the logarithms of the normalized degree centrality and the normalized betweenness centrality only. The corresponding values for the control variables are not displayed, as they are similar to the regressions with a zero time lag. Furthermore, the coefficients and standard errors of the control variables platform size and age of software service change only slightly as the time lag varies for all models. Moreover, the significance level and the sign of the coefficients change even less. This stability of these values of the control variables is an indication that the control variables are independent of any time lag. For example, the platform size of Flickr is 57, as Flickr belongs to Yahoo and Yahoo released 57 software services. This platform size value has been considered for the regressions with 37 different time lags. With respect to the third type of control variable, the effect of the dummy control variables (i.e., software services category variables) is also almost invariant for all time lags. The entertainment service category has a significantly negative effect on the service performance for time lags $0 \leq t \leq 29$, and the weather services category and the wiki services category have a significantly negative effect for time lags $24 \leq t \leq 31$. The effects of other software service categories are insignificant for all time lags. That is, the service performance is not dependent on service categories, to which software services belong, except for the three categories (i.e., entertainment services category, wiki services category, and weather services category). This means that the explanation of service performance with respect to software service categories does not show any stability. There is no specific characteristic of those categories that could explain these results. Moreover, the weather services category comprises only three software services. The entertainment services category and the wiki services category even comprise only one

Table 5
Trend of coefficients for Models 2, 3, and 4 in dependence of the time lag. The value in the parenthesis denotes the standard error, and the number of stars represent the level of significance.

Time Lag t	Model 2	Model 3	Model 4	
	Degree Centrality	Betweenness Centrality	Degree Centrality	Betweenness Centrality
0	0.376 (0.160) **	0.235 (0.077) ***	0.422 (0.429)	0.144 (0.120)
1	0.375 (0.160) **	0.233 (0.077) ***	0.435 (0.430)	0.139 (0.121)
2	0.377 (0.160) **	0.233 (0.077) ***	0.440 (0.429)	0.138 (0.120)
3	0.375 (0.160) **	0.234 (0.077) ***	0.439 (0.425)	0.140 (0.120)
4	0.376 (0.160) **	0.231 (0.077) ***	0.455 (0.425)	0.133 (0.120)
5	0.376 (0.160) **	0.232 (0.077) ***	0.448 (0.426)	0.136 (0.120)
6	0.375 (0.160) **	0.233 (0.077) ***	0.452 (0.419)	0.136 (0.118)
7	0.376 (0.161) **	0.229 (0.078) ***	0.439 (0.422)	0.135 (0.119)
8	0.374 (0.161) **	0.227 (0.078) ***	0.441 (0.421)	0.134 (0.118)
9	0.373 (0.161) **	0.228 (0.077) ***	0.421 (0.421)	0.139 (0.118)
10	0.374 (0.161) **	0.228 (0.078) ***	0.421 (0.421)	0.139 (0.118)
11	0.375 (0.161) **	0.230 (0.078) ***	0.422 (0.420)	0.140 (0.118)
12	0.376 (0.161) **	0.235 (0.078) ***	0.412 (0.421)	0.147 (0.119)
13	0.368 (0.161) **	0.233 (0.078) ***	0.406 (0.421)	0.144 (0.121)
14	0.366 (0.161) **	0.231 (0.078) ***	0.416 (0.418)	0.140 (0.120)
15	0.368 (0.161) **	0.238 (0.078) ***	0.378 (0.418)	0.155 (0.120)
16	0.368 (0.161) **	0.239 (0.077) ***	0.355 (0.419)	0.160 (0.121)
17	0.365 (0.162) **	0.247 (0.076) ***	0.267 (0.421)	0.189 (0.120)
18	0.361 (0.162) **	0.252 (0.076) ***	0.301 (0.402)	0.185 (0.118)
19	0.367 (0.162) **	0.267 (0.074) ***	0.203 (0.404)	0.221 (0.118) *
20	0.365 (0.162) **	0.253 (0.075) ***	0.253 (0.408)	0.196 (0.118)
21	0.368 (0.162) **	0.265 (0.073) ***	0.301 (0.387)	0.199 (0.112) *
22	0.448 (0.165) ***	0.267 (0.073) ***	0.288 (0.382)	0.204 (0.111) *
23	0.408 (0.166) **	0.296 (0.074) ***	0.042 (0.374)	0.287 (0.112) **
24	0.410 (0.166) **	0.299 (0.075) ***	0.039 (0.374)	0.290 (0.112) **
25	0.399 (0.166) **	0.297 (0.078) ***	−0.082 (0.402)	0.316 (0.123) **
26	0.400 (0.166) **	0.305 (0.075) ***	−0.245 (0.391)	0.364 (0.120) ***
27	0.398 (0.166) **	0.294 (0.076) ***	−0.121 (0.384)	0.322 (0.117) ***
28	0.399 (0.166) **	0.310 (0.075) ***	−0.229 (0.374)	0.364 (0.115) ***
29	0.400 (0.165) **	0.319 (0.076) ***	−0.443 (0.389)	0.415 (0.113) ***
30	0.400 (0.165) **	0.332 (0.077) ***	−0.432 (0.381)	0.424 (0.112) ***
31	0.411 (0.164) **	0.282 (0.084) ***	−0.450 (0.411)	0.378 (0.121) ***
32	0.420 (0.164) **	0.277 (0.084) ***	−0.418 (0.409)	0.367 (0.121) ***
33	0.439 (0.170) **	0.265 (0.085) ***	−0.314 (0.422)	0.331 (0.124) **
34	0.430 (0.171) **	0.246 (0.088) ***	−0.350 (0.425)	0.319 (0.126) **
35	0.409 (0.171) **	0.243 (0.084) ***	−0.303 (0.384)	0.305 (0.115) **
36	0.410 (0.171) **	0.240 (0.085) ***	−0.326 (0.392)	0.308 (0.118) **

* $p < 0.10$.
** $p < 0.05$.
*** $p < 0.01$.

software service each.

The regression results of Model 2 depict that the logarithm of the normalized degree centrality affects the logarithm of reach (Table 5). The coefficients are positive at a level of significance of 5%. However, with respect to Model 4, the dependence of the logarithm of reach on the logarithm of the normalized degree centrality is insignificant though positive for all time lags ($0 \leq t \leq 36$). Although the normalized degree centrality could be the network position measure that explains the service performance, the results are not consistent between Model 2 and Model 4. The normalized degree centrality is inconsistent for all time lags, if we insert the normalized betweenness centrality in the model (Model 4). Therefore, we can hardly suggest that degree centrality affects the service performance for any time lag. Moreover, we suspect that the negative relationship observed in Model 4 is likely to be caused from its correlation with the normalized betweenness centrality, which affects the service performance (Model 3).

The regression results of Models 3 and 4 suggest that the effect of the normalized betweenness centrality on service performance is significantly different from the one of the normalized degree centrality. If the regression model does not include the normalized degree centrality (Model 3 in Table 5), the effect of the logarithm of the normalized betweenness centrality is positive at a significance level of 1% for all time lags ($0 \leq t \leq 36$). If we consider both normalized degree and normalized betweenness centralities in one regression model (Model 4), the logarithm of betweenness centrality is also insignificant for small time lags (i.e., $t \leq 18$) but reaches a 10% level of significance for $t \geq 19$, a 5% level of significance for $t \geq 23$, and a 1% level of significance for $26 \leq t \leq 32$. Considering these results, it is suggested that the normalized betweenness centrality strongly affects the service performance with a time lag between $t = 26$ and $t = 32$.

In order to decide on the quality of the three models considered, we measure the models' goodness-of-fit with the adjusted R^2 for each regression model [43]. Fig. 4 illustrates the trend of the adjusted R^2 of the regression Models 2, 3, and 4 for each time lag t. The adjusted R^2 of Models 3 and 4 remains stable until around 0.4 for the short time lag region ($t \leq 18$) and slightly fluctuates for the long time lag region ($t > 18$) between 0.4 and 0.45. The adjusted R^2 of Model 2 remains stable around 0.35 for the short time lag region ($t \leq 20$) and slightly increases for the long time lag region ($t > 20$). Consequently, considering the time lags does not undermine the relationship between the network position and the service performance. However, because of the high R^2 of Models 3 and 4, we can also state that Models 3 and 4 are superior to Model 2. Moreover, as Model 3 is simpler than Model 4 (i.e., it comprises one variable less) but achieves the same R^2 values, Model 3 is to be preferred to Model 4.

In summary, the trend of adjusted R^2 suggests that the effect of the normalized betweenness centrality with a time lag of around 29 months is the strongest. Moreover, we can state that Hypothesis 1 is supported.

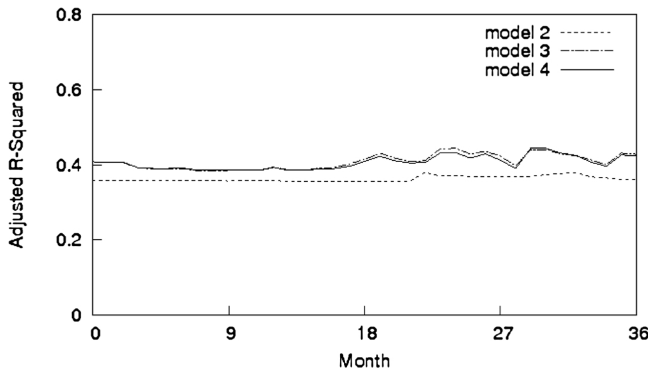

Fig. 4. Adjusted R^2 of Models 2, 3, and 4 for different time lags, ranging from 0 to 36 months.

Specifically, software services with a high betweenness centrality are likely to yield a high service performance around 26–32 months later and affect the service performance with R^2 of 0.45.

5. Discussion and conclusion

5.1. Academic implications

Although our results also show that the network position of a software service affects its service performance immediately, our study also indicates that previous network positions have an effect on the service performance. Specifically, with a time lag of around 26–32 months, the betweenness centrality of a software service affects its service performance.

Innovation studies show that if new knowledge is created on the basis of existing knowledge and experience, an agent, who has a network position where it can gather a variety of existing knowledge, can be more innovative than agents at the periphery of the network. In particular, Burt [8] and Granovetter [9] find that the network position that is most advantageous for innovation is the one in which an agent mediates other agents (i.e., combines fragmented knowledge of those agents). Besides, Krackhardt [6] and Tsai [7] show that an agent with strong connectivity could be innovative if complex problems have to be solved. Moreover, the impact of the network position in an innovation network also depends on the context (e.g., the type of knowledge considered [36], the structure of an organization [44], and the diversity of people [45]). Although the effect of network positions on service performance is not new in innovation studies, the analysis of the network position of a software service in a software service network is new. Moreover, the significant difference between existing innovation studies and our study of the software service network is that existing innovation studies analyzed knowledge flows, whereas our study reveals the complementarity of existing knowledge (i.e., software services) that can be used for creating new knowledge (i.e., composite software service). A composite software service always identified and uses two complementarities of existing software services and combines those with its own extra knowledge. This way, the composite software service can attract end users, as its own service combines the complementarities. Without any complementarity, software services would not be combined in a composite service. This also provides an explanation for our analysis results that only the effect of the normalized betweenness centrality (and not the normalized degree centrality) can be observed. The normalized betweenness centrality is a very good measure for capturing the complementarity of existing knowledge.

Furthermore, previous innovation studies implemented cross-sectional analyses, ignoring the time lag between network position and innovation performance [4,7,9,35,36]. General life-cycle theory states that a product of an agent (i.e., a firm or a person) and a technology (i.e., technology embedded in a product, a service, or a document) depict a life cycle consisting of three stages: emerging, maturity, and declining [16–19]. With respect to social network analyses, existing research demonstrates that network positions also change over time. Wagner and Leydesdorff [3] observe that a scholar approaches a central position in an academic collaboration network over time. Kim et al. [20] show that within a software service network, a software service approaches the center and retreats to the periphery according to the stage of its life cycle. Because of these three aspects, it is reasonable to assume that the relationship between network position of software services and their service performance changes over time.

In addition to this, another academic implication from our study concerns the time lag before the network position of a software service affects its service performance. According to our analysis, the highest effect of a network position (i.e., betweenness centrality) in a software service network on service performance can be noticed with a time lag of around 26 to 32 months. Prior research on innovation from a network perspective misses the time lag aspect of the dependence, although the diffusion process has been considered [17,16,18]. The effect of network position is not only immediate but rather also manifests itself after a time lag. In the context of software services, the time lag might be caused through the time that a software service provider needs for identifying and for generating revenue from the value of its software service through the complementarity that its service provides to other software services. It also takes time before users of software services become aware of the composite services within the software service network, recognize their value, and become familiar with them.

Therefore, based on our empirical results, we conjecture that life-cycle theory and the diffusion process (e.g., [17,16,18]) are at work, affecting service performance in software service networks. The consequence of our research is that social network-based innovation studies need to consider not only the structure of the network but also the evolution of the network structure over time and the time lag before the network position affects service performance.

5.2. Managerial implications

Our findings could be applied for designing strategies for optimally positioning a software service of a software service provider in a software service network and for designing revenue sharing schemes between software service developers. As the network position determines its service performance, the business model of the software service provider is to motivate third-party software service developers to create new composite services using the software services of the provider.

By considering the network position, the revenue sharing scheme can reflect the value contributions of the different software services of the composite software service. Therefore, the network position can be the basis for the revenue for both the software service provider and the third-party service developer [27–29]. Furthermore, the revenue sharing scheme should consider the number of users, who visit the website of a software service (i.e., "reach"). That is, the higher the number of users, who visit a software service, is, the higher the value contribution of the software service is.

Our results show that the normalized betweenness centrality affects the service performance for all time lags, from $t = 0$ to $t = 36$ (Model 3). The strongest effect can be observed at a time lag of about 26–32 months with the highest adjusted R^2 of 0.45 at a time lag of 29 months (Models 3 and 4). The normalized degree centrality does not confer a significant advantage. Therefore, a desirable innovation strategy for a software service provider would be to incentivize composite services that combine its service with software services that are distant within the software service network. Furthermore, although the network position of a software service is determined through the businesses of the composite service developers utilizing the software services, the software service provider can directly promote its software services by advertising its software service as being complementary to a certain set of software services categories. The complementarity characteristics of software services exhibit the value creation opportunities.

Moreover, a software service provider can also develop composite services directly, so that it becomes easier for third-party developers to use a set of software services, as mentioned in Baek et al. [27]. This aspect has been shown in our empirical data analysis. The platform size is significant.

5.3. Limitations

Our research has two limitations, which need further research. First, the explanatory power of our models suggests that our models might miss other factors affecting service performance. Although our Models 3 and 4 suggest that network positions (i.e. degree centrality and betweenness centrality) explain already 40% of the service performance (Fig. 4), they still suggest that the explanatory power can still be increased. Therefore, although our models are controlled with the platform size and service categories, we suspect that the relationship between the network position and service performance might depend on the business context of service providers. Therefore, future studies need to explain the time lag between network position and service performance on the ground of the competitive advantages, business models, and strategies of service providers.

Second, the general social network research describes knowledge flows within networks, whereas the software service network that we use reveals the complementarity of fragmented knowledge. Therefore, it is difficult to state, in general, whether our time lag related results can be observed in social networks that are analyzed with respect to knowledge flows.

Acknowledgements

This research was partly supported by the Basic Science Research Program (2016R1A6A3A11932416), the Social Science Korea program (NRF-2018S1A3A2075175) of the National Research Foundation of Korea (NRF), and not by the 2019 Jung-Gyeon Yeongu Program of the National Research Foundation of Korea (NRF) funded by the Korea government (MSIT). The Institute of Engineering Research at Seoul National University provided research facilities.

Appendix A

Suppose that the maximum possible number of connections that a node can have is *g – 1* in a weighted network of *g* nodes (i.e., the network size is *g*), then the normalized degree centrality D_i^w of node *i* is the sum of weights w_{ij} of all the links to node *j* of the neighbor set, *N*(*i*), of node *i*:

$$D_i^w = \sum_{j \in N(i)} w_{ij}/(g-1)$$

Suppose that the number $\sigma_{kl}^w(i)$ of the shortest paths between two nodes (node *k* and *l*) passing through node *i* and the total number T_{kl}^w of shortest paths between the pair of nodes *k* and *l* is given, then the betweenness centrality B_i^w of node *i* can be calculated as

$$B_i^w = \sum_{k,l \neq k} \frac{\sigma_{kl}^w(i)}{T_{kl}^w} \bullet \frac{2}{(g-1)(g-2)}$$

where $(g-1)(g-2)/2$ is the maximum possible number of shortest paths between any pair of nodes and a specific third node on the path in a network of size *g* with $g > 2$.

References

[1] P.P. Maglio, S.L. Vargo, N. Caswell, J. Spohrer, The service system is the basic abstraction of service science, Inf. Syst. E-Bus. Manag. 7 (2009) 395–406, https://doi.org/10.1007/s10257-008-0105-1.
[2] A.J. Lopes, R. Pineda, Service systems engineering applications, Procedia Comput. Sci. 2013 Conference on Systems Engineering Research vol. 16, (2013) 678–687, https://doi.org/10.1016/j.procs.2013.01.071.
[3] C.S. Wagner, L. Leydesdorff, Network structure, self-organization, and the growth of international collaboration in science, Res. Policy 34 (2005) 1608–1618, https://doi.org/10.1016/j.respol.2005.08.002.
[4] R. Grewal, G.L. Lilien, G. Mallapragada, Location, location, location: how network embeddedness affects project success in open source systems, Manag. Sci. 52 (2006) 1043–1056, https://doi.org/10.1287/mnsc.1060.0550.
[5] R.S. Burt, Structural holes and good ideas, Am. J. Sociol. 110 (2004) 349–399, https://doi.org/10.1086/421787.
[6] D. Krackhardt, The strength of strong ties: the importance of philos in organizations, Networks and Organizations: Structure, Form, and Action, Harvard Business School Press, Boston, MA, 1992, pp. 216–239.
[7] W. Tsai, Knowledge transfer in intraorganizational networks: effects of network position and absorptive capacity on business unit innovation and performance, Acad. Manage. J. 44 (2001) 996–1004, https://doi.org/10.2307/3069443.
[8] R. Burt, Structural Holes: The Social Structure of Competition, Harvard University Press, Cambridge, MA, 1995.
[9] M.S. Granovetter, The strength of weak ties, Am. J. Sociol. 78 (1973) 1360–1380.
[10] A. Abbasi, J. Altmann, On the correlation between research performance and social network analysis measures applied to research collaboration networks, 2011 44th Hawaii International Conference on System Sciences (HICSS) (2011) 1–10, https://doi.org/10.1109/HICSS.2011.325.
[11] A. Abbasi, J. Altmann, J. Hwang, Evaluating scholars based on their academic collaboration activities: two indices, the RC-index and the CC-index, for quantifying collaboration activities of researchers and scientific communities, Scientometrics 83 (2009) 1–13, https://doi.org/10.1007/s11192-009-0139-2.
[12] A. Abbasi, J. Altmann, L. Hossain, Identifying the effects of co-authorship networks on the performance of scholars: a correlation and regression analysis of performance measures and social network analysis measures, J. Informetr. 5 (2011) 594–607, https://doi.org/10.1016/j.joi.2011.05.007.
[13] J. Altmann, A. Abbasi, J. Hwang, Evaluating the productivity of researchers and their communities: the RP-index and the CP-index, Int. J. Comput. Sci. Appl. 6 (2009) 104–118.
[14] O. Cimenler, K.A. Reeves, J. Skvoretz, A regression analysis of researchers' social network metrics on their citation performance in a college of engineering, J. Informetr. 8 (2014) 667–682, https://doi.org/10.1016/j.joi.2014.06.004.
[15] S. Sasidharan, R. Santhanam, D.J. Brass, V. Sambamurthy, The effects of social network structure on enterprise systems success: a longitudinal multilevel analysis, Inf. Syst. Res. 23 (2012) 658–678, https://doi.org/10.1287/isre.1110.0388.
[16] E.M. Rogers, Diffusion of Innovations, 5th ed., Free Press, New York, 2003.
[17] F.M. Bass, A new product growth for model consumer durables, Manag. Sci. 15 (1969) 215–227, https://doi.org/10.1287/mnsc.15.5.215.
[18] F. Bellone, P. Musso, L. Nesta, M. Quéré, Market selection along the firm life cycle, Ind. Corp. Change 17 (2008) 753–777, https://doi.org/10.1093/icc/dtn025.
[19] J.W.B. Bos, J.W. Kolari, R.C.R. van Lamoen, Competition and innovation: evidence from financial services, J. Bank. Finance 37 (2013) 1590–1601, https://doi.org/10.1016/j.jbankfin.2012.12.015.
[20] K. Kim, W.-R. Lee, J. Altmann, SNA-based innovation trend analysis in software service networks, Electron. Mark. (2014) 1–12, https://doi.org/10.1007/s12525-014-0164-8.
[21] L.C. Freeman, Centrality in social networks conceptual clarification, Soc. Netw. 1 (1978) 215–239, https://doi.org/10.1016/0378-8733(78)90021-7.
[22] S.S.C. Shang, E.Y. Li, Y.-L. Wu, O.C.L. Hou, Understanding Web 2.0 service models: a knowledge-creating perspective, Inf. Manage. 48 (2011) 178–184, https://doi.org/10.1016/j.im.2011.01.005.
[23] M.N. Haines, M.A. Rothenberger, How a service-oriented architecture may change the software development process, Commun. ACM 53 (2010) 135–140, https://doi.org/10.1145/1787234.1787269.
[24] M. Ogrinz, Mashup Patterns: Designs and Examples for the Modern Enterprise, Addison-Wesley Professional, Upper Saddle River, NJ, 2009.
[25] H.W. Chesbrough, Open Innovation: The New Imperative for Creating and Profiting from Technology, Harvard Business Press, Boston, MA, 2005.
[26] A. Gawer, M.A. Cusumano, Platform Leadership: How Intel, Microsoft, and Cisco Drive Industry Innovation, Harvard Business Press, Boston, MA, 2002.
[27] S. Baek, K. Kim, J. Altmann, Role of platform providers in service networks: the case of Salesforce.com App Exchange, 2014 IEEE 16th Conference on Business Informatics (CBI) (2014) 39–45, https://doi.org/10.1109/CBI.2014.58.
[28] K. Kim, J. Altmann, Evolution of the software-as-a-service innovation system

through collective intelligence, Int. J. Coop. Inf. Syst. 22 (2013) 1340006, , https://doi.org/10.1142/S0218843013400066.

[29] N. Haile, J. Altmann, Structural analysis of value creation in software service platforms, Electron. Mark. 26 (2) (2015) 129–142, https://doi.org/10.1007/s12525-015-0208-8 Springer.

[30] N. Haile, J. Altmann, Value creation in software service platforms, Future Gener Comp Sy, Elsevier, 2015, https://doi.org/10.1016/j.future.2015.09.029.

[31] J. Hwang, J. Altmann, K. Kim, The structural evolution of the Web 2.0 service network, Online Inf. Rev. 33 (2009) 1040–1057, https://doi.org/10.1108/14684520911010990.

[32] K. Kim, J. Altmann, J. Hwang, An analysis of the openness of the Web 2.0 service network based on two sets of indices for measuring the impact of service ownership, 2011 44th Hawaii International Conference on System Sciences (HICSS) (2011) 1–11, https://doi.org/10.1109/HICSS.2011.47.

[33] A.B. Hargadon, Brokering knowledge: linking learning and innovation, Res. Organ. Behav. 24 (2002) 41–85, https://doi.org/10.1016/S0191-3085(02)24003-4.

[34] J.A. Schumpeter, J.E. Elliott, The Theory of Economic Development: An Inquiry into Profits, Capital, Credit, Interest, and the Business Cycle, Transaction, New Brunswick, NJ, 1982.

[35] A. Everard, R. Henry, A social network analysis of interlocked directorates in electronic commerce firms, Electron. Commer. Res. Appl. 1 (2002) 225–234, https://doi.org/10.1016/S1567-4223(02)00014-5.

[36] M.T. Hansen, The search-transfer problem: the role of weak ties in sharing knowledge across organization subunits, Adm. Sci. Q. 44 (1999) 82–111, https://doi.org/10.2307/2667032.

[37] M.O. Jackson, Social and Economic Networks, Princeton University Press, Princeton, NJ, 2010.

[38] T. Opsahl, F. Agneessens, J. Skvoretz, Node centrality in weighted networks: generalizing degree and shortest paths, Soc. Netw. 32 (2010) 245–251, https://doi.org/10.1016/j.socnet.2010.03.006.

[39] M.E.J. Newman, Clustering and preferential attachment in growing networks, Phys. Rev. E 64 (2001) 25102, https://doi.org/10.1103/PhysRevE.64.025102.

[40] N. Haile, J. Altmann, Evaluating investments in portability and interoperability between software service platforms, Future Gener Comp Sy, Elsevier, 2017, https://doi.org/10.1016/j.future.2017.04.040.

[41] I. Newman, C. Newman, A discussion of low r-squares: concerns and uses, Educ. Res. Q. 24 (2) (2000) 3.

[42] R.F. Falk, N.B. Miller, A Primer for Soft Modeling, University of Akron Press, 1992.

[43] J. Johnston, Econometric Methods, 4th ed., McGraw Hill Higher Education, New York, 1997.

[44] J. Walter, C. Lechner, F.W. Kellermanns, Knowledge transfer between and within alliance partners: private versus collective benefits of social capital, J. Bus. Res. 60 (2007) 698–710, https://doi.org/10.1016/j.jbusres.2007.01.026.

[45] S. Boehm, H.S. Schröder, F. Kunze, Comparative age management: theoretical perspectives and practical implications, The Sage Handbook of Aging, Work and Society, (2013), pp. 211–237.

Kibae Kim is a research assistant professor at Moon Soul Graduate School of Future Strategy of the Korea Advanced Institute of Science and Technology. Prior to this position, he was a BK assistant professor at the Technology Management, Economics and Policy Program of Seoul National University, South Korea. His research focuses on the empirical and numerical analysis of innovation networks, including service networks, as well as on nonlinear dynamic behavior of agents in complex socio-economic systems.

Jörn Altmann is Professor for Technology Management, Economics, and Policy at the College of Engineering of Seoul National University. Prior to this, he taught computer networks at UC Berkeley, worked as senior scientist at Hewlett-Packard Labs, and has been a postdoctoral fellow at EECS and ICSI of UC Berkeley. Dr. Altmann's research centers on Internet economics with a focus on economic analysis of Internet services and on integrating economic models into Internet infrastructures.

Wonjoon Kim is associate professor at the School of Business and Technology Management, KAIST. He has been conducting and publishing numerous researches on the strategic management of innovation in high-tech industry such as mobile, pharmaceutical, energy, and creative industry. Before he joined KAIST, he has been adjunct-assistant professor at the Department of Economics, NYU and a research fellow at the Yale School of Management. He holds a PhD degree in economics from Seoul National University focusing on economics and management of technology innovation.