21st INTERNATIONAL DEPENDENCY AND STRUCTURE MODELING CONFERENCE, DSM 2019
MONTEREY, CA, USA, 23 – 25 September, 2019

Using Dependency Structure Models for Ranking Critical Components in Product Architectures

Raghvinder S. Sangwan, Satish Srinivasan, Colin Neill, Nil Kilicay-Ergin
School of Graduate Professional Studies
Pennsylvania State University
Malvern, PA, 19355 (USA)

Abstract: Dependency structure models, or their equivalent dependency network models, can be analyzed for identifying critical components of a product architecture – elements within the architecture that, if changed, can have a significant impact on the rest of the system. In this paper we propose a discrete-time Markov chain-based algorithm to analyze dependency structure models of several product architectures to not only identify but also rank such components in order of their criticality. Identifying and ranking these elements allows prudent allocation of resources on a project to mitigate any risks that result from changes made to an architecture as a product evolves. The results show that the proposed algorithm is more effective when compared to other algorithms, namely betweenness centrality, closeness centrality and eigenvalue centrality, that have been used on dependency network models to identify and rank critical components.

Keywords: Dependency Structure Model (DSM), Product Architecture, Component Ranking, Discrete-Time Markov Chain, Betweenness Centrality, Closeness Centrality, Eigenvalue Centrality

1 Introduction

Designing product architectures involves design decisions that address requirements which have an overall impact on the quality of the whole system. These architectural design decisions are embodied in a product's components, their relationship to each other and to the environment in which they operate (Eppinger and Browning, 2012). Because of their systemic nature, any changes to these architecturally significant components entails a risk that these changes can propagate to other parts of the system and can have far reaching consequences impacting the overall quality of the system.

As products evolve over time, however, they can become quite complex and so do their architectures. Managing this complexity involves maintaining an up to date model of the dependencies of product components to each other and to their operating environment. Dependency Structure Models or Matrices (DSM) have been used for this purpose (Steward, 1981). DSMs or their equivalent dependency network models have been further analyzed using clustering algorithms to understand the product modularity (Schaeffer, 2007), change propagation (Clarkson *et al*, 2004) and, reliability and availability (Wilschut *et al*, 2017). Browning (2016) provides a comprehensive survey of these extensions.

In this paper, we use DSMs and analyze their equivalent dependency network models to rank components based on how critical they are within a product architecture. There are a number of well-known ranking algorithms, such as betweenness centrality (BC), closeness centrality (CC), and eigenvector centrality (EC) for this purpose (Landherr *et al*, 2010). Each ranking algorithm takes a different approach to determining the criticality of a component, capturing different network characteristics and thus resulting in different ranking orders (Lu *et al*, 2016). For example, BC and CC are path-based centrality measures whereas EC is an iterative refinement centrality algorithm. BC ranks highly those components that are frequently located on as many of the shortest path between pairs of other components within a network; CC favors components that are closest to most of the other components in a network; and EC favors components if they are well-connected to other highly ranked components within a network. Due to their specific and narrow focus (supported by the results of our analysis presented later in the paper), however, no single measure was effective at identifying those elements within a product architecture that experts considered significant.

Consequently, we have developed a Discrete-time Markov chain (MC) based ranking algorithm that has a broader focus and more comprehensive approach to identifying important elements of complex networks. MC, gives more weight to components that have a high indegree, are connected to other components with high indegree, and frequently lie on the shortest paths between other components within the system. As such it detects components that have greater structural significance and, therefore, greater risk of propagating their changes and faults to other parts of the system.

MC creates a dependency network model by analyzing dependencies of a component of a system (for instance, from a dependency structure model) and treats the resulting network as a discrete-time Markov chain (DTMC), a concept used for stochastic modeling of complex systems (Srinivasan & Azadmanesh, 2007) that has been applied extensively to different areas of complex systems development (Davis 2018, Puterman 2005, Prowell 2005, Gomez *et al* 2010). Within a DTMC, every component has a weight that is propagated to its neighbors along its outgoing edges. Initially, all components are assigned equal weights. Once a component receives the weights from all its neighbors along the incoming edges it updates its weight. If the propagation of weights is performed over and over again in multiple rounds, then after several rounds the component graph will attain a stationary-convergence. Once the final weight of each component in the component graph is determined, MC ranks the components in the decreasing order of their weight *i.e.* a component having higher weight than other components is ranked higher thus determining the relative significance of the different components within the system under study. Detailed description of the MC algorithm can be found in Srinivasan *et al* (2017).

The rest of the paper is organized as follows. In section 2, current studies relevant to analysis of dependency network models and ranking of important elements within a dependency network model are reviewed. Section 3 describes the ranking of components by the different ranking algorithms using a simple dependency network model as an example. Section 4 compares the merits of the different ranking algorithms using various product architecture case studies explored in detail in Eppinger and Browning (2012) discussing the implications of each algorithm for ranking components. Finally, Section 5 concludes this study.

2 Related Work

SPARS system uses analysis of dependency networks to rank components for Java-based systems (Inoue *et al.*, 2005). Baker *et. al.* (2006) use greedy and simulated annealing algorithms to address a component ranking problem to support identification of components that best suits a target system. Neuhaus *et. al.* (2007) have proposed ranking of software components to investigate their vulnerability. Garvey and Pinto (2009) represented engineering systems as a directed graph where nodes depict the direction, strength, and criticality of supplier-provider dependency relationships. They proposed Functional Dependency Network Analysis (FDNA) method which analyzes criticality of nodes and dependencies in order to understand ripple effects of service failures. Guariniello and DeLaurentis (2013) have adapted the FDNA to System of Systems (SoS) analysis where architecture of SoS is modeled as a directed graph to study critical capabilities in operational SoS networks. Development Dependency Network Analysis (DDNA) is another directed graph model which quantifies the effect of dependencies on SoS development time. Analysis of dependency networks has been used to rank functionally important methods in complex software systems such as Google Chrome (Pakhira and Andras, 2012).

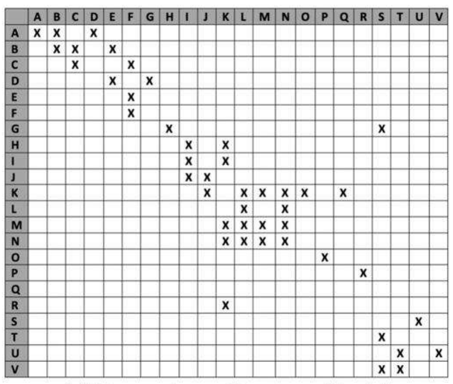
There are numerous measures and methods for identifying critical nodes of a network. Lu *et al.* (2016) have provided a systematic review of the metrics and methods for identifying important nodes of a network. Network metrics have been used to determine critical components of system architectures as well (Okami *et al*, 2017, Bartolomei *et al*, 2012, Santana *et al*, 2016). Yang and Xie (2016) have reviewed various measures for ranking social networks and proposed a multi-attribute ranking method which combines various measures for a weighted evaluation of social network nodes.

Whereas centrality measures have been shown effective at identifying a network's most influential nodes, they are not as useful in quantifying the impact on a network of those nodes that are not highly influential, yet make up the vast majority of the network. This has given rise to node influence metrics including expected force (Lawyer, 2015) and accessibility (Travencolo and Costa, 2008). These metrics are based upon epidemiological models of infection and spreading that likely have meaning in engineered systems also.

While there are different network metrics for identification of critical nodes, it is important to understand the implications of using these metrics for different network structures. This study extends current studies by utilizing DSM as means to construct Markov chain dependency networks for identification of architecturally significant components.

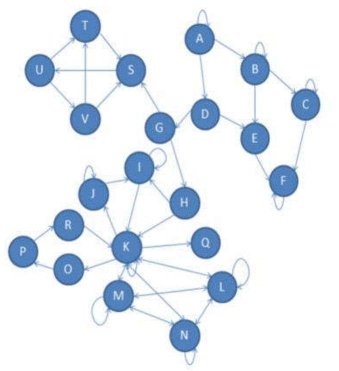
3 Component Ranking Strategies

Using the DSM shown in Figure 1(a) and its equivalent dependency network model in Figure 1(b), we illustrate the ranking of the components in the network using MC, EC, CC and BC algorithms. Through this contrived example, we demonstrate the differences in the ranking strategies these algorithms use.

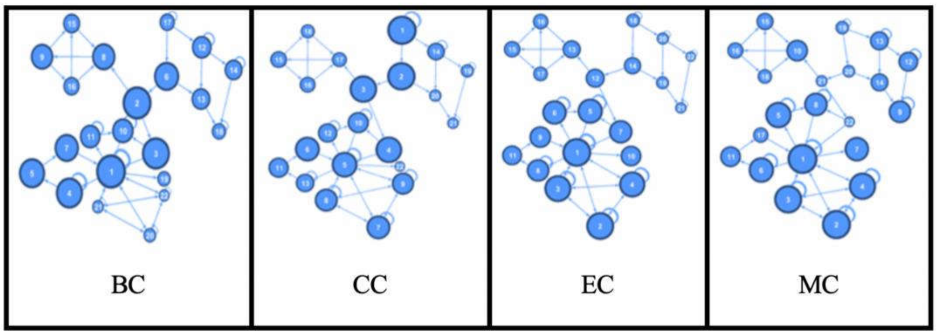


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
A	X	X		X																		
B		X	X		X																	
C			X			X																
D					X		X															
E						X																
F						X																
G								X											X			
H									X		X											
I									X		X											
J									X	X												
K										X		X	X	X	X		X					
L												X		X								
M											X	X	X	X								
N											X	X	X	X								
O																X						
P																		X				
Q																						
R											X											
S																					X	
T																			X			
U																				X		X
V																			X	X		

(a) Dependency Structure Model



(b) Dependency Network Model



(c) Component Rankings

Figure 1. Rankings of components in a dependency network model using BC, CC, EC and MC

Figure 1(c) shows the rankings of components in the dependency network model using the different raking algorithms. The sizes of the components are in proportion to their relative rank with the largest sized component achieving the highest rank and the smallest sized component achieving the lowest rank. We also label a node with its corresponding rank.

As the figure shows, BC ranks those components highly that appear on most of the shortest paths between any pair of components within the network. For instance, components K, G and H are ranked highly because most components in the lower half of the network must communicate through them in order to reach the nodes in the upper half of the network. CC ranks components A, D and G highly since they have close access to a significant proportion of the other components in the network. EC ranks K, N and M highly because they are connected to other components in the network that also have a high ranking. While MC also ranks K, N and M highly its ranking strategy is multi-faceted. It tends to favor those components that have a high indegree and also receive connections from other components that have a high indegree. It also gives more weight to those components that lie frequently on the shortest paths among of other components in the network. Therefore, its overall ranking is different from EC. For instance, I has a higher indegree compared to

J, O, and Q but is ranked lower because the latter have an incoming link from K (a node ranked highly because of high indegree).

4 Comparative Analysis

Due to the diverse approaches these ranking algorithms take, we used each of them to analyze the AW101 helicopter system described in Eppinger and Browining (2012) to understand which algorithm may be most effective in identifying and ranking highly, the critical components within that system. The systems described in this reference serve as great examples for such studies because the authors have extensively explored each system and provide an independent analysis of critical elements based upon the expertise of senior engineers familiar with those systems. This forms the basis of the ideal ranking against which the automatic approaches are assessed shown in Table 1. We call this ranking ideal since this case study provides the rationale for why one component may be more important than the other based on how influential they are in propagating the changes to other components in the system.

Table 1. Ideal rankings of components of the AW101 helicopter model

Ranking	Components	Description
1	Bare fuselage	*System Description*: Agusta Westland produces helicopters, such as AW101, for civil and military applications. Their aircraft designs are often based on existing model, but they also support redesigning a product based on the specific needs of their customers. In that process, a change to a part of the product, however, results in changes to other parts. The AW101 model is comprised of 19 key components and subsystems. A change any one may impact other components. *Rationale behind ranking its components*: The 19 components of the AW101 helicopter model were ranked in such a way that a change introduced within a highly ranked component has a high risk of propagating changes to many other components in the system.
2	Cabling and piping	
3	Avionics	
4	Flight control systems	
5	Auxiliary electrics	
6	Air Conditioning	
7	Hydraulics	
8	Transmission	
9	Equipment and furnishings	
10	Main rotor head	
11	Fuel	
12	Ice and rain protection	
13	Tail rotor	
14	Fire protection	
15	Engine auxiliaries	
16	Weapons and defensive systems	
17	Main rotor blades	
18	Fuselage additional Items	
19	Engines	

We analyzed the AW101 dependency network model, created from the dependency structure model, using the BC, CC, EC and MC. The results of their rankings and their statistical significance are shown in Table 2. MC ranked most of the components highly consistent with the ideal ranking list (Column 2 of Table 2). However, the rankings by MC (Column 3 of Table 2) were not exactly in the same order as listed in the ideal ranking list.

Table 2. Ideal ranking of the components of the AW101 helicopter model

Components	**Ideal Ranking**	**MC**	**BC**	**CC**	**EC**
Bare fuselage	1	1	1	19	1
Cabling and piping	2	2	4	12	4
Avionics	3	3	8	8	3
Flight control systems	4	5	7	4	8
Auxiliary electrics	5	6	3	16	2
Air Conditioning	6	9	12	17	7
Hydraulics	7	4	9	7	19
Transmission	8	7	16	13	12
Equipment and furnishings	9	8	10	3	6
Main rotor head	10	11	13	1	13
Fuel	11	14	19	18	16
Ice and rain protection	12	18	6	15	18
Tail rotor	13	10	11	9	5
Fire protection	14	16	17	10	11
Engine auxiliaries	15	13	5	11	10
Weapons and defensive systems	16	12	15	6	9
Main rotor blades	17	17	14	14	17
Fuselage additional Items	18	15	18	5	15
Engines	19	19	2	2	14
Spearman Rho	**Correlation coefficient**	0.903	0.414	-0.321	0.6052
	p-value	2.2e-16	0.039	0.9113	0.00355
Kendall Tau b	**Correlation coefficient**	0.766	0.333	-0.2514	0.4152
	p-value	1.488e-07	0.0245	0.9376	0.00641
Precision@k	**Precision@5**	0.8	0.6	0.2	0.8
	Precision@10	0.9	0.7	0.5	0.7
	Precision@15	0.87	0.8	0.73	0.8
	MAP	0.86	0.70	0.47	0.77

For statistical analysis, we employed two nonparametric ranking correlations, Spearman's Rho and Kendall's Tau-b, to measure the strength of relationship between two ordered variables, the ideal ranking and the ranking by one of the other algorithms. A high value of associativity (close to 1) indicates a strong relationship between the two rankings. The results show that MC has the highest correlation coefficient and p-values were significant (α=0.01).

We also drew upon an information retrieval evaluation statistic, Precision@k commonly used to compare ranked search results. Precision@k measures the ratio of the total count of the number of components that are ranked in the exact same order by a ranking algorithm when compared with the ideal ranking, to the total number of ranked components. So, while the ranking correlations assess the similarity of the rankings with the desired ideal rank, the Precision@k metric or Mean Precision@k (MAP) assesses the relative positions of the components in each ranking. As shown in Table 2, the precision@k (k = 5, 10, 15) and MAP values for MC are superior when compared to the other algorithms.

We repeated this study with three other systems, namely, Pratt & Whitney's commercial aircraft jet engine, Xerox's digital printing system, and Kodak's single use camera, using their dependency structure models from Eppinger and Browning (2012). For all three product architectures, both MC and EC resulted in a p-value less than 0.01 for both the Spearman's Rho and Kendall's statistic thus indicating that the null hypothesis can be rejected. Therefore, it can be concluded that the ranking for MC and EC are positively correlated with the ideal ranking of the components in all the three product architectures. Although the p-values were significant for EC, the Spearman Rho and Kendall Tau correlation coefficients were relatively less when compared to MC, indicating that MC's performance was superior. MC also produced superior precision@k and MAP values for all but Xerox's digital prinitng system for which EC recorded better performance. Table 3 shows a summary of these results.

Table 3. Correlation Coefficients, Precision@k for different ranking algorithms

Correlation Coefficient (p-value)	**MC**	**BC**	**CC**	**EC**
	Commercial aircraft jet engine, Pratt & Whitney, USA			
Spearman Rho (p value)	0.964 (2.2e-16)	0.320 (0.009)	0.191 (0.08)	0.527 (2.75e-05)
Kendall Tau (p value)	0.853 (2.2e-16)	0.235 (0.005)	0.127 (0.08)	0.403 (8.36e-06)
Precision@5	1.0	0.4	0.2	0.6
Precision@10	1.0	0.6	0.5	0.8
Precision@15	0.93	0.53	0.33	0.6
MAP	0.98	0.51	0.34	0.67
	Digital printing system, Xerox, USA			
Spearman Rho (p value)	0.836 (2.2e-16)	0.290 (0.003)	0.310 (0.002)	0.721 (2.2e-16)
Kendall Tau	0.650	0.223	0.218	0.554

(p value)	(2.2e-16)	(0.001)	(0.001)	(4.21e-14)
Precision@5	0.4	0.4	0.2	0.6
Precision@10	0.4	0.3	0.4	0.8
Precision@15	0.66	0.4	0.33	0.6
MAP	0.49	0.37	0.31	0.67
	Single use camera, Kodak, USA			
Spearman Rho (p value)	0.965 (2.2e-16)	0.418 (0.01)	0.086 (0.324)	0.910 (9.311e-08)
Kendall Tau (p value)	0.857 (6.66e-16)	0.314 (0.007)	0.089 (0.250)	0.756 (1.17e-11)
Precision@5	0.8	0.4	0.2	0.8
Precision@10	0.9	0.5	0.4	0.8
Precision@15	0.933	0.6	0.533	0.866
MAP	0.877	0.5	0.377	0.822

We do not provide the ideal ranking of the components for these product architectures as they have components that range from 19 to 84. The rationale behind ranking of their components are summarized below based on the DSMs and description of the product in Eppinger and Browning (2012).

1. *Commercial aircraft jet engine, Pratt & Whitney, USA*: 54 components across different subsystems were ranked based on how many interfaces it had with other components in the system. Components with higher number of interfaces were ranked higher. The components belonging to the subsystems that interfaced with other components across different subsystems were also ranked higher.
2. *Digital printing system, Xerox, USA*: The 84 components of the Xerox iGen3 printing system were ranked based upon the number of interfaces it has with other components in the system. Also, a component that interfaces with many components in the system were ranked higher.
3. *Single use camera, Kodak, USA*: The components that have many common interactions with other components were ranked higher. Following the components with common interactions are the components that have variant and unique interactions.

5 Conclusion

This paper explored four different ranking algorithms, including one that we have developed, each offering a different perspective on ranking components within a given system. BC gives more importance to components that lie more frequently on the shortest paths between components of a system, CC deems components that have access to most other components within a system as significant, and EC ranks highly those components that are well connected to other highly ranked components. Our proposed ranking algorithm, MC, gives more weight to components that have a high indegree, are connected to other components with high indegree, and frequently lie on the shortest paths between other components within the system. As such it detects components that have greater structural significance and, therefore, greater risk of propagating their changes and faults to other parts of the system.

In our future work we will expand the analysis to a larger set of product architectures and explore alternative network algorithms including node influence metrics such as accessibility and expected force, both of which have meaning with respect to change propagation and network resilience.

References

Baker, P., Harman, H., Steinhofel, K., Skaliotis, A., 2006. Search Based Approaches to Component Selection and Prioritization for the Next Release Problem", in the proceedings of the 22nd IEEE International Conference on Software Maintenance, pp. 176-185.

Bartolomei, J.E., Hastings, D.E., de Neufville, R., Rhodes, D.H., 2012. Engineering Systems Multiple Domain Matrix: An organizing framework for modeling large-scale complex systems" Systems Engineering, 15(1), pp.41-61.

Browning, T.R., 2016. Design Structure Matrix Extensions and Innovations: A Survey and New Opportunities," in IEEE Transactions on Engineering Management, vol. 63, no. 1, pp. 27-52.

Clarkson, P.J., Simons, C., Eckert, C., 2004. Predicting change propagation in complex design, J. Mech. Des., vol. 126, pp. 788–797.

Davis, M.H., 1993. Markov Models and Optimization, New York, NY, USA: Taylor & Francis.

Eppinger, S.D., Browning, T.R., 2012. Design Structure Matrix Methods and Applications. Cambridge, MA, USA: MIT Press.

Inoue, K., Yokomori, R., Fujiwara, H., Yomamoto, T., Matsushita, M., Kusumoto, S., 2005. Ranking Significance of Software Components Based on Use Relations, IEEE Transactions on Software Engineering, vol. 31, No. 3, pp. 213-225.

Garvey, P., Pinto, C., 2009. Introduction to functional dependency network analysis", Second International Symposium on Engineering Systems, MIT, Cambridge, Massachusetts, June 15-17.

Gomez, S., Arenas, A., Borge-Holthoefer, J., Meloni, S., Moreno, Y., 2010. Discrete-Time Markov Chain approach to contact-based disease spreading in complex networks, Europhysics Letters Association, Vol. 89 No. 3.

Guariniello, C., DeLaurentis, D., 2013. Dependency Analysis of System-of-Systems Operational and Development Networks" Conference on Systems Engineering Research, Atlanta GA, March 19-22.

Lu, L., Chen, D., Ren, X., Zhang, Q., Zhang, Y., Zhou, T. 2016. Vital nodes identification in complex networks, Physics Reports, vol. 650, pp. 1-63.

Landherr, B., Friedl, J., Heidemann, J., 2010. A Critical Review of Centrality Measures in Social Networks, Discussion Paper WI-282, Business and Information Systems Engineering, vol. 2, no.6, pp. 371-385, 2010.

Lawyer, G. 2015. Understanding the influence of all nodes in a network. Scientific Reports 5, p. 8665.

Neuhaus, S., Zimmermann, T., Holler, C., Zeller, A., 2007. Predicting Vulnerable Software Components, Proceedings of the 14th ACM conference on Computer and Communication Security (CCS '07), pp. 529-540, 2007.

Okami, S., Kohtake, N., 2017. Transitional Complexity of Health Information System of Systems: Managing by the Engineering Systems Multiple-Domain Modeling Approach" IEEE Systems Journal.

Pakhira, A., Andras, P., 2012. Using Network Analysis Metrics to Discover Functionally Important Methods in Large-Scale Software Systems, Proceedings of the 3rd International Workshop on Emerging Trends in Software Metrics (WETSoM), pp. 70-76.

Puterman, M.L., 2005. Markov Decision Processes: Discrete Stochastic Dynamic Programming, Hoboken, NJ, USA: John Wiley & Sons.

Santana, A., Kreimeyer, M., Clo, P., Fischbach, K., de Moura, H., 2016. An Empirical Investigation of Enterprise Architecture Analysis Based on Network Measures and Expert Knowledge: A Case from the Automotive Industry" Modern Project Management, Vol. 46.

Schaeffer, S.E. 2007. Graph clustering, Comput. Sci. Rev., vol. 1, pp. 27–64.

Srinivasan, S.M., Azadmanesh, A.H., 2007. Exploiting Markov Chains to Reach Approximate Agreement in Partially Connected Networks, International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2007), 2007.

Srinivasan, S.M., Sangwan, R.S., Neill, C.J., 2017, On the measures for ranking software components, Innovations in Systems and Software Engineering, vol.13, no.2-3, pp. 161-175.

Steward, D.V., 1981, The design structure system: A method for managing the design of complex systems, IEEE Trans. Eng. Manage., vol. EM-28, no. 3, pp. 71–74.

Travençolo, B. A.N. and Costa, L. da F., 2008, Accessibility in complex networks, Physical Letters A, Vol. 373, Iss. 1, pp. 89-95.

Wilschut, T., Rooda, J.E., Etman, L.P., and Vogel, J.A., 2017, A DSM based method for the ranking of system components wrt system reliability and availability, 19th International Dependency and Structure Modeling Conference (DSM 2017), pp. 1-10.

Yang, Y., Xie, G., 2016. Efficient Identification of Node Importance in Social Networks", Information Processing and Management, vol. 52, pp. 911-922.

Contact: Raghvinder S. Sangwan, School of Graduate Professional Studies, Pennsylvania State University, Malvern, PA, 19355 (USA), +1 (610) 725-5354, rsangwan@psu.edu