






Metadata Action Network Model for Cloud Based Development Environment

Mehmet N. Aydin¹ , Ziya N. Perdahci² , I. Safak¹,
and J. (Jos) van Hillegersberg³ 

¹ Kadir Has University, 34083 Istanbul, Turkey
mehmet.aydin@khas.edu.tr

² Mimar Sinan Fine Arts University, 34083 Istanbul, Turkey
nz.perdahci@msgsu.edu.tr

³ University of Twente, 34083 Enschede, The Netherlands
j.vanhillegersberg@utwente.nl

Abstract. Cloud-based software development solutions (entitled as Platform-as-a-Service, Low-Code platforms) have been promoted as a game changing paradigm backed by model-driven architecture and supported by various cloud-based services. With the engagement of a sheer number of platform users (experienced, novel, or citizen developers) these platforms generate invaluable data and that can be considered as user metadata actions. As cloud-based development solutions provide novice users with a new development experience (performing data actions that altogether leads to a successful software app), users often times face with uncertainty about development performance; how good or complete is app development? Thus, the issue addressed in this research is how to measure user performance by using digital trace data generated on the cloud platform from a Network Science perspective. This research proposes a novel approach to leveraging digital trace data on Platform-as-a-Service (PaaS) from a Network Science perspective. The proposed approach considers the importance of digital trace data as metadata actions on PaaS and introduces a network model (so-called Metadata Action Network), which is claimed to be the result of reconstruction of events of developer's actions. We show suitability of the proposed approach to better understanding of real-world digital trace data on PaaS solution and elaborate basic performance analytics on a PaaS solution with research and practical implications.

Keywords: Network science · Development performance · Digital trace data · PaaS · Analytics · Cloud computing

1 Introduction

Platform as a Service (PaaS) has been promoted as a panacea for the long-standing software development problem of delivering successful solutions with high performed or collaborative actors including developers, users. The premise behind PaaS is that it can foster user performance (novice developers) in terms of delivering better, faster, cheaper, and high-quality enterprise software solutions. Global enterprise solutions providers such as Salesforce, Mendix have adopted PaaS solutions and strived for

engaging greater developers' communities in contributing to cloud-based enterprise systems marketplaces. The research perspectives rooted in Information Systems and Software Engineering have made significant contributions to better understanding of PaaS [1, 2], but there a little attempt to put emphasis on PaaS from digital trace data point of view [3]. That is, with the engagement of sheer number of platforms users (experienced, novel, or citizen developers) these platforms generate invaluable data and that can be used to improve platform performance [4].

The very idea of PaaS has come to into play with the cloud-driven paradigm shift that has evolved over last two decades. Infrastructure, Platform and Service layers as three generic models are adopted by cloud-driven platform-based solutions [3]. This research focuses on PaaS which acts as a bridge between IaaS and SaaS – it is the runtime environment, the middleware of the cloud service architecture [6]. Metadata Application PaaS (aPaaS) is a particular type of PaaS, which provides cloud-based integrated development environment with a metamodel [9]. Simply this type (metadata aPaaS) embrace the very idea of model-driven software development [7] that employs a number of essential principles such as abstraction, model transformation and refinement, reusability [8]. Simply, the idea has existed for more than twenty years, but its realization is about to begin with global success stories such as Mendix, OrangeScape, ServiceNow For instance, German-based Siemens acquired Dutch software manufacturer Mendix for € 628 million (\$730 million) in June 2019, which reportedly makes it one of the largest acquisition of a Rotterdam-based company.

In traditional software development, development efforts and artefacts are structured (stages, phases, models, other artefacts depending on development methodologies) and are subject to certain quality of conduct, so actors involved have some feedback about development performance. As cloud-based development solutions provide novice users with a new development experience (create data actions that altogether leading to a successful software app), users face with uncertainty about development performance; how good or complete is development work underway? Thus, the issue addressed in this research is as follows: how can one measure performance of cloud-based application development by using digital trace data generated on the cloud platform? Can Network Science help in leveraging digital trace data as a network model and in turn provide insights in user performance on cloud-based development? This research proposes a novel approach to leveraging data on aPaaS from a Network Science perspective, which is acclaimed as the Science of the 21st Century [5]. The proposed approach considers the importance of digital trace data as metadata actions on aPaaS and introduces a network model which is claimed to be the result of appropriate events reconstruction on a platform. We show viability of the proposed approach with the real-world digital trace data on aPaaS solution and elaborate network analytics on aPaaS solution with research and practical implications.

Scholars including [9] have pointed out underutilized aspect of these platforms, which is about creation and possibly use of digital trace data (DTD) on aPaaS. It should be noted that the data on aPaaS deserve to be called digital trace if three criteria are satisfied:

- (i) it is found (not generated for research),
- (ii) it is event based (event reconstruction possible),
- (iii) and longitudinal (time stamp for tracing).

In this regard, validation of digital trace data is of particular importance to researchers if digital trace data is accepted as a credible data asset for generating valuable analytics. In this research to ensure validation of digital trace data for our examination we perform all necessary steps defined in [10]. Scholars in various research domains including online social media, information systems have already shown considerable interests in DTD and used terms like data analytics, business intelligence by using different approaches. We contend that the very idea of DTD surfaces recording of the relations of constituting elements of all relevant events. That is, it is about things and their relations forming a network. Even though the contexts for things and their relations are different so the networks are labeled differently, a number of common distinguishing network characteristics have been at the center of attention. This calls for a Network Science approach as a novel way of examining digital trace data as a complex system [11]. In the last two decades, there has been a significant interest in better understanding of real-world entities and their relations as complex systems. With 20th anniversary of Network Science [5], scholars have investigated the very nature of these complex systems in various contexts such as social, technological, health sciences, and political sciences [12, 13]. In the following we shall discuss how user actions, so-called metadata actions, generating digital trace data are considered as a network data. The basic trust of this research is that Network Science Approach to aPaaS can significantly contribute to better understanding of user actions in model-driven development and to progress of disruptive aPaaS technologies. Thus, this research is aimed to discover graph models which can describe best the digital trace data. Then the research question is to decide on the type of network representation of metadata action patterns (i.e., creating or connecting software artefacts). We have three important choices to make: the choice of nodes, the choice of edges, and the choice of metadata on the nodes and edges. The discovered graph model will be a significant contribution to the reference research domains (Software Engineering and Information Systems) and the cloud computing ecosystem.

2 Background

aPaaS in this context can be defined as a “complete application platform as a service” that offers independent software vendors (ISVs) as well as individuals the means to produce “multitenant SaaS solutions or various application integration solutions” in a fast and efficient way [6].

The critical role of Information Communications Technology (ICT) along with ubiquitous technologies and innovative software applications on digital transformation of nations, society, companies, and individuals cannot be denied in recent years. Apple’s motto “Everyone Can Code” at the beginning of 2018 is worth noticing in that it shows that “everyone”, rather than being just an end-user, can be involved in ICT development as a developer. Essentially, this discourse is a sign of manifesto for a need of paradigm shift from a conventional thinking and practice to a novel approach and cloud technologies (aPaaS, specific cloud computing layer) for ICT development, in particular software application development.

The history of research on software development methods, tools, technologies is rich and can go back to the origin of ICT. From 1970 to mid 1990s that we call a conventional paradigm in software development, the research endeavors and software development practice had witnessed a sheer number of ideas, methodologies, tools that failed to fulfill the promise of better, cheaper, high quality and fitness for purpose software development. In mid 1990s we saw a new era along with new ideas such as method engineering, model-driven, agile software development were proposed to overcome limitations of monolithic tools, jungle of heavy methodologies, constraining technologies leading to frustrating project results. More modern technologies coined as computer-aided software engineering (CASE) have been promoted to support this evolution [7]. The next era started in ICT development with the help of ubiquitous technologies (smart, integrated devices and cloud computing technologies) and innovative software applications such as social network application, mobile applications for end-users. Noticeably, compared to European companies, North American players such as Apple, Google, Facebook, Salesforce, WhatsApp have seized the opportunities of paradigm shift in software development for end-user applications at a global scale.

With the notion of cloud computing this paradigm has made a significant but inadequate leap for software applications for enterprises including small-medium ones. A growing number of enterprises are facing up to digital reality and yet to fully harness the power of cloud computing as a disruptive technology. The attempts made by global tech companies (IBM and Apple, Oracle, SAP) to enable SMEs for digital transformation have made limited progress compare cloud-born niche players providing software as a service (SaaS) (e.g., Salesforce) and platform as a service (PaaS) (e.g., Mendix, OrangeScape) solutions. aPaaS indeed opens up a new ecosystem where the three key stakeholders (independent software vendors-ISV), platform providers, and customers, especially SMEs have an opt for a new way of developing, managing, using software applications. A novel way of developing software for enterprises bring on disruptive changes on the developer side that one can use a motto “everyone can develop app for enterprises”. In fact, these platforms provide higher abstraction with modeling actions instead of code development, making it possible for business logic and user to develop software products instead of code. More precisely, without any code writing knowledge, business and user requirements can be transformed into software products in a unique way. This process can all be recorded digitally.

We contend that aPaaS is the signature of a paradigm shift in software development. Simply it embraces the very idea of model-driven software development that employs a number of essential principles such as abstraction, model transformation and refinement, reusability. That is, the idea has existed for more than 20 years, but its realization is about to begin!

Although these game-changing technology companies have made significant progress in terms of software paradigm shift, recent research and field experience show that an extent to which the aforementioned principles are not fully harnessed in existing platforms. aPaaS platforms are evolving as research and practice goes hand-in-hand. On the research side, analyzing of these platforms in terms of key principles is available in market intelligence report or scientific papers that remain non-empirical, method, model artefacts. On the industry side, technologies, techniques and features on the

platforms are evolving as more experimental and real-life platform use experience driven studies carried out.

3 Network Construction from Digital Trace Data

One of the challenges with DTD is to validate found digital trace data. The researchers and the company worked together to fix several issues with the digital trace resulted from user actions. For instance, creating a Plain Menu Item produces:

```

<<82494,user1@abc.com,bb1ecc8c-9473-4322-8fb2-
221a6ea2d41c,CREATE_NEW_MENU,null,null,null,null,2018-11-16
10:06:09.0,null,null,null,null,null,null,null,null,null,null,null,Mars,null,null,null
82495,user1@abc.com,8cdadf96-e81a-4c8e-87c8-
d1399f4aede2,ADD_TRANSIENT_ENTITY_TO_MENU,null,null,null,null,2018-
11-16
10:06:09.0,null,null,null,null,null,null,null,null,null,null,null,Mars,null,null,null
>>
    
```

aPaaS platform records both the digital trace data of user actions as metadata actions that occur at particular points in time as longitudinal data and attributes of the software artifacts some of the actions, such as the kind of abstract data type the action creates. One converts digital trace data into a network by determining what corresponds to a node and an edge, as well as what corresponds to node/edge attribute. We consider that there is only one edge type. Edges represent binding between two software artifacts. On the other hand, we classify nodes as Model (M), View (V), or Controller (C) (see the Appendix). The node classifications are incorporated into the network model as node attributes. Figure 1 depicts an overall structure of the services provided to users and furthermore shows that the generated digital trace data that can be mapped to architectural views, which is Model-View-Controller (MVC). MVC is a useful pattern for separations of concern in software engineering. It helps developers to partition the application as early as the design phase and especially is applied to web-based applications.

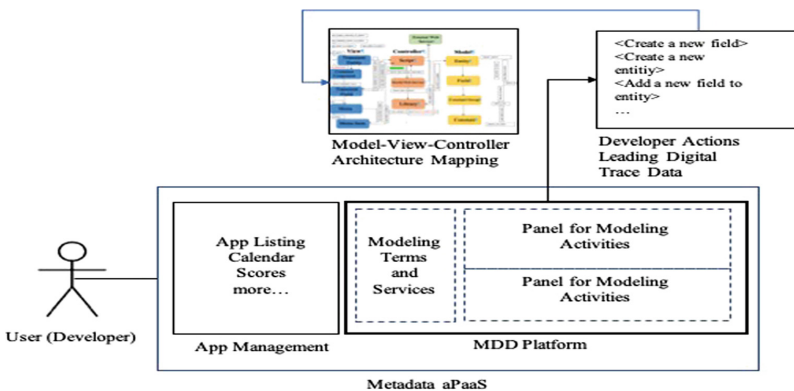


Fig. 1. Overall structure of the services provided on Metadata aPaaS.

aPaaS provides the developer with a cloud-based integrated development environment (Cloud-IDE) that fosters rapid application development via a glossary and services (Fig. 2a and Fig. 2b). To each term in the glossary or service there corresponds a metadata action that compiles a software artifact (executable code). Similarly, to each service there corresponds a built-in function to perform a specific task. Moreover, developers are able to add their own services to enhance the functionality of the via a simple scripting language (e.g. MVEL). Some of the metadata actions provide the developer with creating an artifact, while others enable the developer to link the artifacts.

As a specific example, imagine that the developer is given the task to develop a simple post form that contains several text fields and buttons. One button linked to a service allows the user to retrieve their own contact information from the server, one button allows the user, for example, to upload an image, and a text box is provided to type a post they want to send. Clicking on another button sends the post data to a server. If we think about the metadata actions involved in developing the form, the developer may invoke a few actions or around a dozen actions at most, as well as a few functions, and there are hundreds of metadata actions and services to choose from on a typical aPaaS. This means that almost every metadata action that the developer could potentially select to create an artifact and/or link artifacts the developer does not select. Or, almost every possible link between the artifacts is non-link, it is not registered in the digital trace data; so, the data collected by aPaaS platforms in this respect is sparse; as opposed to dense data in a world where every software artifact is linked with each other.

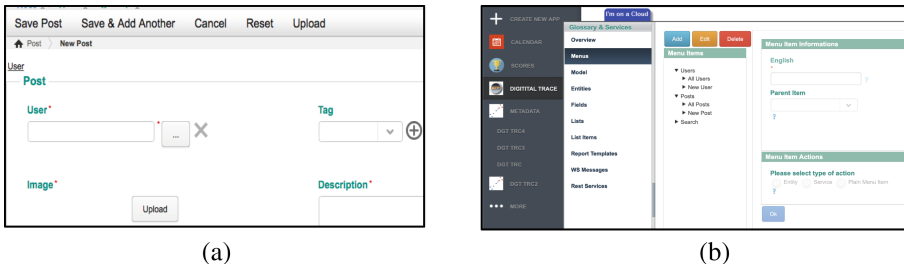


Fig. 2. a. A simple form generated on aPaaS. b. User interface on aPaaS

It is this sparseness [18] that calls for Network Science approach to analysis of metadata actions on aPaaS. Many systems can be regarded as networks, sets of things and their interactions. In a graphic representation of a network, nodes (or vertices) are the things of interest, and interacting nodes are joined in pairs by arcs (or links). A network is a mathematical object specifically designed to represent sparse data, and network science is concerned with analyzing and modeling such systems. Figure 3 depicts an overall network construction process from the raw data creation (users' actions) to network visualization and analytics.

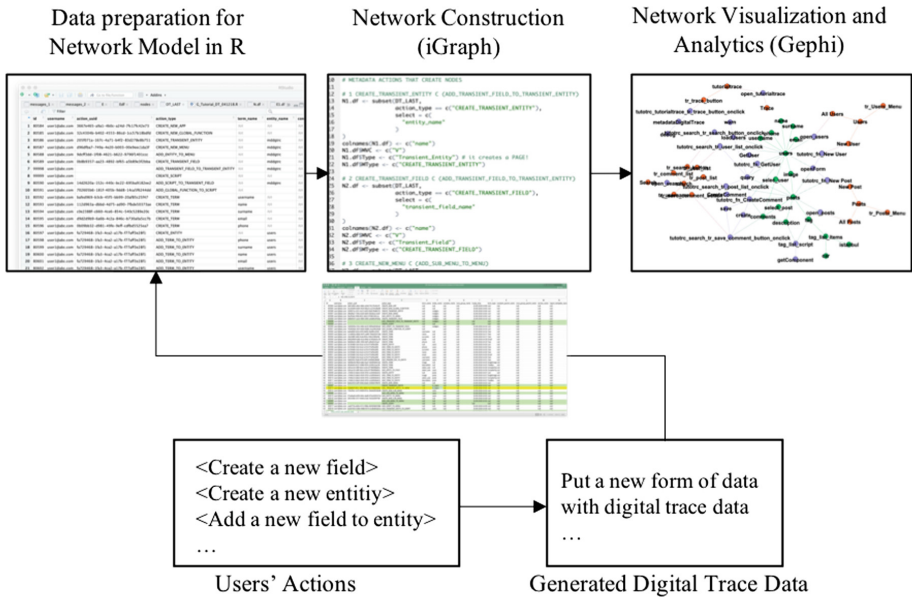


Fig. 3. Network construction: from digital trace data to network analytics

We can regard the sparse data of recorded metadata actions on the aPaaS data base as a network. To emphasize this, from now on, we will refer to the network produced by the developers who develop applications as the “metadata action network” (MAN). In MAN the objects of interest are the software artifacts created by metadata actions, and the artifacts are joined in pairs by arcs if the developer opts to link them.

4 Demonstration of Network Analytics on aPaaS

The digital trace data under examination is found on imona.com, which is an application development platform (Application Platform as Service: aPaaS) where developers can not only create new applications, but it also offers the possibility of extending the functionality of any application already placed in its marketplace [4]. Imona.com is a one type of aPaaS, called metadata aPaaS [4]. Metadata aPaaS provides visual tools to customize data models, application logic, workflow, and user interface. The underlying metadata model for this aPaaS is essential to this research as it provides us a meta-model [9] to reflect on network models of digital trace data to be discussed later on.

The dataset includes the list of metadata actions that were created by developing a tutorial app by seven developers. This tutorial app is chosen as the name suggests it is used for a training purpose, simple enough to monitor all development activities and all steps with visual guidance in a 42-minutes video are provided. It should be noted that

even the tutorial video provides a clear guidance on how to develop app, there is no single pathway to follow while developing the app. This gives us an opportunity to compare development activities from the proposed network model. Another interesting point is that in advance only a verbal brief on where and what to develop is given and no any conceptual support (such as use cases or any documents) is provided during the development activities. The application is based on three conceptual entities: User, Post, Comment. It is similar to typical online user-content sharing apps where a new user is to be created so that the user can create a post and comment on a post. Additionally, a user can search posts or comments. This app essentially consists of three distinct pages, which is referred as transient entity in the metadata action description.

While presenting MAN, we use the tutorial app and refer to MVC to describe metadata actions. We distinguish metadata actions that create nodes, actions that create edges between nodes, and furthermore actions that create both nodes and edges. For each node we use an MVC label as metadata. That means, the node can be model or view or controller. An application in general is composed of several screens. For example, a developer can add a new screen to an application by using a given service (in our case, it is using “an add button with a label of “add””). This act invokes the metadata “CREATE_ENTITY”, which we model as a creation of a new network node, type M(Model). Instantiation of this metadata can be “user”. Another example would be “ADD_TERM_TO_ENTITY”, which we model as a creation of a new network edge. Instantiation of this metadata can be a form field, such as “name” of the “user” entity. One can find descriptions of the rest of all metadata actions as a MAN Catalog in the Appendix.

5 Discussion and Conclusion

Table 1 summarizes basic network statistics of the cases under investigation. The research question was to decide on what should constitute elements of a graph (what is node and what is edge) and graph representation (directed or undirected, multigraph, weighted, bipartite) itself. We have demonstrated that MAN proposed is viable to model applications developed on an aPaaS environment with promising outcomes.

Regarding viability of the proposed network, MAN of an application is able to reveal the underlying MVC architectural paradigm [15]. That is, three network layers (colored as black, white, and grey) we observed are in accordance with what the underlying architecture of the platform provides (Fig. 4). The outer most layer has to be the View because this is what an end user interacts with. The middle layer corresponds to the Controller because this is the layer that bridges the gap between the Model and the View layers. The inner most layer indicates the Model where the data reside.

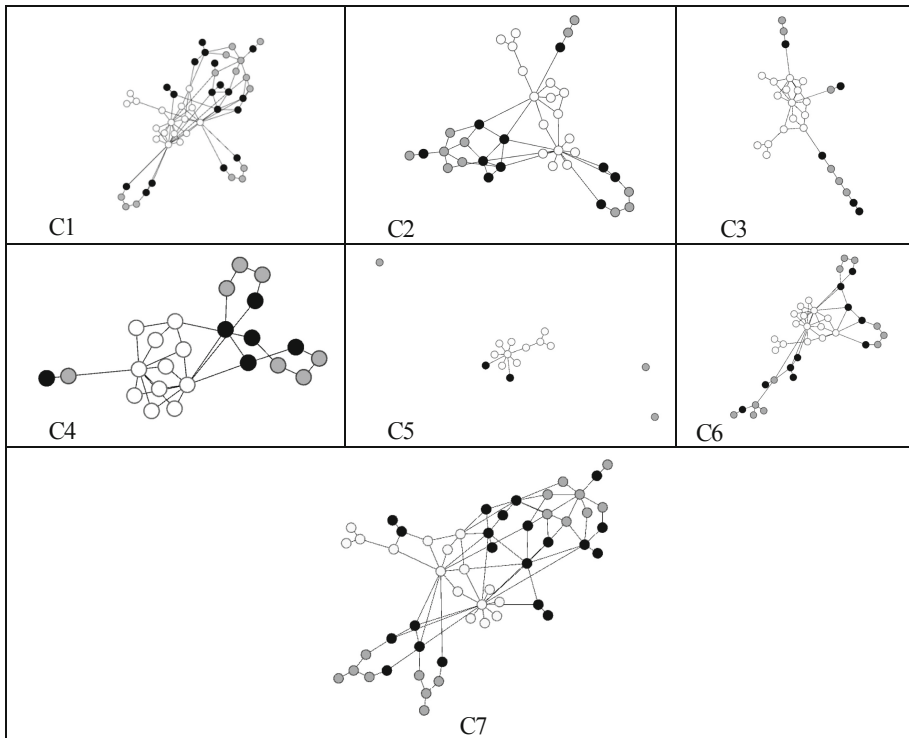


Fig. 4. MAN of the application developed by each group. For graph layout, force Atlas2 is used in Gephi [14]

Table 1. Comparison of different cases based on connected components and network diameter.

Cases	Network measures			Status of release candidate
	#of CC	Diameter	Radius	
C1	1	8	5	Beta
C2	1	8	5	Beta
C3	1	11	6	Alpha
C4	1	7	4	Alpha
C5	4	4	0	Premature
C6	1	9	5	Release candidate
C7	1	9	5	Release candidate

Regarding promising outcomes, one of the challenges the platform owner would face is to provide developers with a revision control system (RCS) [16]. Although it may not be possible to provide RCS in the same way as traditional IDE we contend that MAN could be employed to provide a new kind of RCS. In conventional software development, the user can visually check whether all software components are

integrated whereas on an aPaaS this is not the case. The MAN graph consisting of a single connected component indicates that all software artefacts are interconnected. We suggest that the total number of components can be used as a revision control system, if it is more than one it means that some artefacts are still not interconnected (so-called premature, Alpha or Beta) and the user should make additional metadata actions to complete a version. If there is only one component and if other analytics results are fulfilled, then the application developed may deserve to be a Release Candidate.

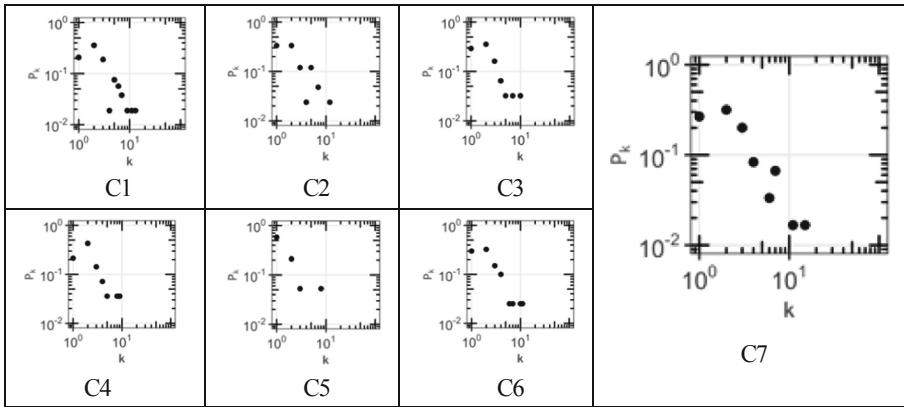


Fig. 5. Degree distribution for all cases

Yet another promising outcome is that basic network statistics such as degree distributions, network radius, and network diameter can be related to aPaaS user performance analytics. Figure 5 depicts degree distributions of each case. Visually one can see that the degree distributions of Case 7, Case 6 and to some extent Case 1 and Case 3 exhibit an approximate straight line on doubly logarithmic scales which is typical of real-world networks, whereas the degree distribution of Case 5 is clearly distinct from a power-law [19]. MAN for MVC architectural paradigm constraints the Diameter of the developed application to a certain size, which we believe is the number of screens developed for an aPaaS application multiplied by the software architecture layers (MVC), which is 3. So, for example, for the case at hand it has to be 9. When we compare 7 cases, two of them satisfy this result, which would be another indicator for them to be Release Candidates. The Radius provides us with another salient analytics, which indicates whether MVC layers are interconnected with shorter pathways. The middle layer of the software architecture (the Controller layer) and the other two layers it bridges, namely the View and the Model layers should be equally spaced. In line with this argument, we suggest yet another formula: The Radius of MAN for a Release Candidate has to be approximately half the Diameter. So, for the app examined, the Radius should be approximately equal to the number of screens developed, which is three, divided by two (half the Diameter), which is 4.5. The observed value of the Diameters for Release Candidates and Beta conform to this formula, which are 5.

Further exploratory network analysis is done to identify structure of a network where one can ask the key question: What structural shape does the metadata action network take? The next step would be to search for common patterns of network correlations for MAN of different applications where one can ask the key question: Is there a typical correlation pattern the metadata action networks of developed aPaaS exhibit? Promising correlation metrics to be examined would be as follows: degree-to-degree correlation, node attribute-to-node attribute correlation [20]. Specifically, it looks promising to search for a pattern of assortative mixing by MVC. More ambitious analysis is going beyond structural and correlation pattern analysis is to predict a network structure based on a statistical network model. Essentially this is about answering the why question by fitting a mechanism to observed network data.

It is vital for the platform provider to use the metamodel and the meta language that are at the heart of the platform. These analyses of the digital trace data help the platform provider understand how fully and correctly the software developer performs. For example, the magnitude of network correlation may be a critical criterion for demonstrating the level of reusability (in-app and across-app) of application software. From the developer perspective [17], network metrics can be used that provide understandability, the version of the software, and the extent to which the software has been completed. How much has the software completed in the first version, and what is the situation in terms of upcoming releases from the network point of view and how can the software evolve?

Appendix

METADATA ACTIONS	NETWORK CONSTRUCTION		
	NodeCreation	Edge Creation	NodeType
ADD_ENTITY_TO_FIELD		x	
ADD_ENTITY_TO_MENU		x	
ADD_ENTITY_TO_SCRIPT		x	
ADD_ENTITY_TO_LIBRARY	x	x	C
ADD_GLOBAL_FUNCTION_TO_SCRIPT		x	
ADD_ITEM_TO_LIST		x	
ADD_LIBRARY_TO_SCRIPT	x	x	C
ADD_LIST_TO_FIELD		x	
ADD_PRIMARY_KEY_TO_ENTITY		x	
ADD_SCRIPT_TO_MENU		x	
ADD_SCRIPT_TO_TRANSIENT_FIELD		x	
ADD_SCRIPT_TO_REST	x	x	C
ADD_SUBMENU_TO_MENU		x	
ADD_TERM_TO_ENTITY		x	
ADD_TERM_TO_FIELD		x	
ADD_TRANSIENT_ENTITY_TO_MENU		x	
ADD_TRANSIENT_ENTITY_TO_SCRIPT		x	

(continued)

(continued)

METADATA ACTIONS	NETWORK CONSTRUCTION		
	NodeCreation	Edge Creation	NodeType
ADD_TRANSIENT_FIELD_TO_TRANSIENT_ENTITY		x	
CREATE_ECONTAINER_COMPONENT	x		M
CREATE_ENTITY	x		M
CREATE_LIST	x		M
CREATE_LIST_ITEM	x		M
CREATE_NEW_GLOBAL_FUNCTION	x		C
CREATE_NEW_MENU	x		V
CREATE_NEW_SUB_MENU	x		V
CREATE_REST_SERVICE	x		C
CREATE_SCRIPT	x		C
CREATE_TERM	x		M
CREATE_TRANSIENT_ENTITY	x		V

References

1. Armbrust, M., Fox, A., Griffith, R., et al.: A view of cloud computing. *Commun. ACM* **53** (4), 50–58 (2010)
2. Beimborn, D., Miletzki, T., Wenzel, S.: Platform as a service (PaaS). *Bus. Inf. Syst. Eng.* **3** (6), 381–384 (2011)
3. Teixeira, C., Pinto, J.S., Azevedo, R., et al.: The building blocks of a PaaS. *J. Netw. Syst. Manag.* **22**(1), 75–99 (2014)
4. Aydin, M.N., Perdahci, N.Z., Odevci, B.: Cloud-based development environments: PaaS. In: *Encyclopedia of Cloud Computing*, p. 62 (2016)
5. Vespignani, A.: Twenty years of network science. *Nature* **558**, 528 (2018)
6. Bezemer, C.P., Zaidman, A., Platzbeecker, B., et al.: Enabling multi-tenancy: an industrial experience report. In: *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, September 2010, pp. 1–8. IEEE (2010)
7. Premkumar, G., Potter, M.: Adoption of computer aided software engineering (CASE) technology: an innovation adoption perspective. *ACM SIGMIS Database: DATABASE Adv. Inf. Syst.* **26**(2–3), 105–124 (1995)
8. Henkel, M., Stirna, J.: Pondering on the key functionality of model driven development tools: the case of mendix. In: *International Conference on Business Informatics Research*. Springer, Heidelberg (2010)
9. Aydin, M.N., Kariniauskaite, D., Perdahci, N.Z.: Validity issues of digital trace data for platform as a service: a network science perspective. In: *World Conference on Information Systems and Technologies*, pp. 654–664. Springer, Cham (2018)
10. Howison, J., Wiggins, A., Crowston, K.: Validity issues in the use of social network analysis with digital trace data. *J. Assoc. Inf. Syst.* **12**(12), 767 (2011)
11. Barabási, A.L.: *Network Science*. Cambridge University Press, Cambridge (2016)
12. Borgatti, S.P., Mehra, A., Brass, D.J., Labianca, G.: Network analysis in the social sciences. *Science* **323**(5916), 892–895 (2009)
13. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: structure and dynamics. *Phys. Rep.* **424**(4), 175–308 (2006)

14. Bastian, M., Heymann, S., Jacomy, M.: Gephi: an open source software for exploring and manipulating networks. In: The Proceedings of the Third International ICWSM Conference ICWSM, San Jose, California, pp. 361–362. AAAI Press, Menlo Park (2009)
15. Leff, A., Rayfield, J.T.: Web-application development using the model/view/controller design pattern. In: Proceedings of the Fifth IEEE International Enterprise Distributed Object Computing Conference, pp. 118–127. IEEE, September 2001
16. Karsai, G., Sztipanovits, J., Ledeczi, A., Bapty, T.: Model-integrated development of embedded software. *Proc. IEEE* **91**(1), 145–164 (2003)
17. Giessmann, A., Stanoevska-Slabeva, K.: What are developers’ preferences on platform as a service? An empirical investigation. In: Forty-Sixth Hawaii International Conference on System Sciences, January 2013, pp. 1035–1044. IEEE (2013)
18. Demaine, E.D., Reidl, F., Rossmanith, P., Villaamil, F.S., Sikdar, S., Sullivan, B.D.: Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *J. Comput. Syst. Sci.* **105**, 199–241 (2019)
19. Clauset, A., Shalizi, C.R., Newman, M.E.: Power-law distributions in empirical data. *SIAM Rev.* **51**(4), 661–703 (2009)
20. Newman, M.E.: Assortative mixing in networks. *Phys. Rev. Lett.* **89**(20), 208701 (2002)