Digital Habitats in Open Source Ecosystems: Uncovering Communities Using Social and Technical Digital Traces

# Introduction

The success of open source software (Lerner and Tirole 2002) and the trend away from monolithic to platform-based software systems (Hanseth and Lyytinen 2010) have created large-scale software ecosystems, which we refer to as *platform-based open source software ecosystems* (POSSE). POSSE are digital ecosystems consisting of digital artifacts (Kallinikos et al. 2013) that are open source software (OSS) modules (Eck and Uebernickel 2016). These modules interact with the digital ecosystem’s platform core through standardized interfaces and add functionality or value to the digital platform (de Reuver et al. 2018; Tiwana et al. 2010). Hence, digital ecosystems are a growing source of innovation, where capabilities shift from within an organization to third-party developers (Parker et al. 2017; Yoo 2013; Yoo et al. 2010), which has led to an increasing research interest on the phenomenon of POSSE (Jacobides et al. 2018).

In those ecosystems, developers make use of existing modules through a process of recombination and remixing when building new modules, which they then contribute to the POSSE. Out of this process, two interconnected networks arise: (1) a hierarchy of *technical dependencies* from one module to the next (MacCormack et al. 2006), and (2) a social network of *actors*, contributing their skill and time to multiple modules (Roberts et al. 2006). As actors (i.e., developers) form social relationships by contributing to various modules, they might also introduce new technical dependencies, therefore coupling the social and technical networks. Similarly, as new technical dependencies are introduced to a module, the social networks of the actors might change as well – for instance by reporting bugs or seeking help. Studies have investigated the dynamics of changes of technical dependencies over time and whether or not to upgrade a technical dependency (Bavota et al. 2013). Similarly, social aspects such as relations among developers have been investigated as well, for example, with regard to the effects of prior collaboration (Hahn et al. 2008) or how open source software teams are structured (Crowston and Howison 2006).

While some studies have shown that the technical architecture can influence participation in projects (e.g., Baldwin and Clark 2006; MacCormack et al. 2006), the direction and consequence of changes of social or technical networks in POSSE have yet to be uncovered. This understanding of the relationship between the technical and social layers, and how they influence each other’s structure, could explain the growth of those ecosystems and prove critical success factors of POSSE. Accordingly, we ask:

“*How do social networks and technical dependencies interact in platform-based open source software ecosystems?*”

To answer our research question, we propose to conduct a study of three of the most popular frontend development frameworks (i.e., Angular, React, and Vue) and their respective digital ecosystems. In doing so, we aim to analyze both the technical and social networks of those ecosystems with digital trace data (Berente et al. 2019) gathered from the projects’ GitHub and npm repositories. For our analysis, we will primarily use network analysis techniques (i.e., cluster, temporal, and network motif analyses). In doing so, the aim of our study is to uncover the underlying patterns and structures covering the social and technical perspective of POSSE.

These findings contribute to theory and practices by, first, proposing a socio-technical approach for analyzing POSSE, combining both the technical and social layer. Second, we advance our understanding of the growth and success of POSSE and identify important predictors. Combined, this research therefore benefits both research and practice related to POSSE and digital platform ecosystems in general.

The remainder of this paper is organized as follows: First, we define and explain the concept of POSSE and the social and technical interdependencies in open source projects in general and in POSSE in particular. Second, the proposed research design is introduced including the description of our data collection and analysis. Third, we present our expected results as well as our roadmap until ICIS 2019. Fourth, we conclude with a discussion about potential challenges, risks, and our expected contributions.

# Theoretical Background

## Platform-based Open Source Software Ecosystems

Popular OSS, such as the Linux kernel or Node, are platform-based, which means that they evolve around a *digital platform* that is the “extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate” (Tiwana et al. 2010, p. 675). When the platform orchestrator opens the digital platform for external developers, it transitions to a *software ecosystem* (Bosch 2009; Messerschmitt and Szyperski 2003). In software ecosystems, third-party developers contribute their time, knowledge, and skill to increase the value of the ecosystem by participating in it through the contribution of modules (Roberts et al. 2006), which are “add-on software subsystem that connect to the platform to add functionality to the platform” (Tiwana et al. 2010, p. 676). In this scenario, the platform functions as a hub with the complementary modules as spokes connected through application programming interfaces (APIs) or other technical standards (Jacobides et al. 2018). Accordingly, we define *platform-based open source software ecosystems* (POSSE) as “a collection of digital artifacts [i.e., modules] that co-evolve through mutual interference, and the social actors related to these artifacts that are linked by a common interest” (Eck and Uebernickel 2016, p. 2).

By enabling contributions by external actors, digital ecosystems are seen as a growing source of innovation where capabilities shift to the outside of the organization (Parker et al. 2017; Yoo 2013; Yoo et al. 2010). The modular architecture allows for the coordination of heterogeneous third-party developers that contribute interdependent modules through an ecosystem (Jacobides et al. 2018) in which interoperability with the platform core is ensured through the usage of the platform’s interfaces (Tiwana 2015). This opens up POSSE for contributions from a wide range of developers and therefore raises the importance of understanding the interdependencies of social and technical layers involved.

## Social and Technical Interdependencies in Open Source Projects

OSS projects rely heavily on the communities surrounding these projects – without the contributions of time and effort of highly motivated developers, OSS projects cannot exist (Roberts et al. 2006). In general, the success of the open source model has led to an increase in OSS projects and new ideas on how innovation should be managed (Singh et al. 2011). OSS projects do not always produce software targeted at end users, but at developers and this software is “designed to be reused and to provide functionality to other software projects” (Haefliger et al. 2008, p. 180). Boland et al. (2007) make the point that technology (e.g., software) per se does not herald innovation in complex networks. Rather, diverse groups of actors with access to the technology are those putting their capabilities to innovative uses. In the context of OSS, voluntary software developers form project teams that emerge based on the developers’ social networks (Hahn et al. 2008). The success of OSS projects therefore is not only related to technical characteristics, but also to “the project production process and the broader social environment in which developers work” (Singh et al. 2011, p. 814). OSS projects can therefore be seen as socio-technical systems.

As a special type of OSS, POSSE are archetypical socio-technical systems as well and contain both technical and social aspects. From a technical perspective, the digital ecosystem of a platform consists of the platform itself and complementary applications build by third-party developers (Cusumano and Gawer 2002; Tiwana 2013). From a social perspective, the platform, its orchestrator, and all the complementary modules and their creators form the platform’s ecosystem (Jacobides et al. 2018). Hence, POSSE do not only consist of the platform core, its interfaces, and related modules, but also include those third-party developers that contribute to the software ecosystem and thereby generate additional value for the platform and its users through increasing usage of the platform and its complementary software (Gawer and Cusumano 2008).

Originating in biology, the term of a habitat is closely linked to ecosystems and has been defined as the area in which a community (in this case a group of individuals from various species) lives. We adopt this term to define the areas of the POSSE that are made up from technical and social relationships between the individual third-party developers and the respective modules. Thereby, we aim to uncover communities, their habitats, and their effect on the underlying structure and growth of POSSE. Figure 1 shows the components of POSSE and the connection between the social and technical layer.

Figure 1: The Social and Technical Layer in POSSE

To sum up, POSSE are the intersection of effects relating to digital platforms, ecosystems, and OSS. This intersection, combined with the effects emerging from the intertwined relationship of social and technical aspects, are uncharted territory. Due to the popularity and success of OSS—and especially POSSE—this combination of forces is a worthwhile area for research.

# Preliminary Research Design

## Research Method & Study Description

For this research project, we adopt a data-driven computational approach for theory development, following the recommendations for grounded theory methodology (Berente et al. 2019) and guidelines for qualitative research with digital trace data (Lindberg 2019). Here, we refer to digital trace data as “digital records of activities and events that involve information technologies” (Berente et al. 2019, p. 51).

We selected the three ecosystems by following a theoretical sampling strategy. Looking for similar and established POSSE, we identified three suitable ecosystems: (1) Angular[[1]](#footnote-1)(https://angular.io/), (2) React (https://reactjs.org/), (3) and Vue (https://vuejs.org/). All ecosystems are currently among the most popular front-end-frameworks for web- or web-app-development (Stack Overflow 2019). Further, they are suitable as all three ecosystems share similarities: they are written in JavaScript or derivatives (i.e., TypeScript) and aim at providing a basis for web- or app-development. While their respective implementations might differ, the ecosystems also remain similar due to their shared intent (i.e., front-end frameworks). Table 1 provides an overview of all three ecosystems.

Overview of Selected Ecosystems

|  |  |  |  |
| --- | --- | --- | --- |
| \*\*Case\*\* | \_Angular\_ | \_React\_ | \_Vue\_ |
| \*\*Initial Release\*\* | September 2016 | May 2013 | February 2014 |
| \*\*Sponsor\*\* | Google | Facebook | \_Independent\_ |
| \*\*Contributors\*\* | 904 Contributors | 1,290 Contributors | 270 Contributors |
| \*\*Weekly Downloads\*\* (as of April 21, 2019) | 405,811 | 4,588,442 | 894,590 |
| \*\*Size of Ecosystem\*\* | 14,241 Modules | 40,225 Modules | 12,393 Modules |

## Data Collection

To collect data on our three ecosystems, we utilize two main data sources: GitHub (https://github.com), which all three use as source code repository, and npm (https://npmjs.com), which all three use as release management and distribution tool. If needed, we can extend our data collection to GH Archive (https://gharchive.org/), a publicly available data repository, recording and archiving timeline data from the GitHub API. However, as both GitHub and npm provide public access to their data via an API, we will prioritize a first-hand-access.

From GitHub, we are able to collect detailed information about the collaborators and the commits made for both the platform core) and for each module. Further, we are able to collect data on the lines of code (additions and deletions) for every commit and who committed these lines of code as well as comments made by contributors or other users.

From npm, we are able to collect detailed information about the releases related to the respective ecosystem – both for the platform core and for each module. Included in these details is a list of technical dependencies to other modules of the ecosystem for each release. The list of dependencies thereby includes all modules that are registered in the npm registry. Further, we are also able to get information on the download counts – again for the platform core and each module – for each single day.

To identify modules for each of the three ecosystems, we utilize the search API offered by the npm registry. Via a keyword search on the ecosystem’s name, we are able to identify every publicly available module for each of the ecosystems. While modules for Angular and React are tagged with “angular” and “react” respectively, modules for vue are tagged “vue”, “vue.js”, or “vuejs”. Including these variants, we are able to identify all needed modules.

Combining the data from GitHub and npm, we are able to aggregate measures such as the number of modules available in an ecosystem, the lines of code affected by a release, the lines of code per contributor per release, or the lines of code per download per day, giving us a wide range of measures for the growth and success of POSSE (cf. Lindberg 2019). Including temporal data (e.g., quarterly downloads from npm), we are able to calculate a “velocity” for the platform core and each module, further extending our options for later data analysis.

## Data Analysis

In a first step, based on the collected data on the social relationships as well as the technical dependencies of the modules, we are going to generate directed graphs for general network analyses and further cluster, temporal, and network motifs analyses. By adopting a network perspective on the social and technical relationships in the ecosystem, first we will calculate general network measures (i.e., degrees, centralities, modularity) (Wasserman and Faust 1994), followed by cluster analysis for both the social and technical network.

The main goal, especially of the cluster analysis, is to identify cliques of social actors (i.e., developers) and groups of technical dependencies between modules inside the networks, which we will later use for comparison. For our cluster analysis, we will use the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm introduced by Ester et al. (1996). We choose a density-based over a centroid-based clustering, which identifies clusters of arbitrary shape and accounts for outliers in the data, hence, no a priori specification of the number of clusters is necessary (Vaast et al. 2017).

In addition, we aim to conduct a temporal analysis of the whole network and its clusters to identify changes over time and potential key events that lead to those changes in the networks. In doing so, we can analyze the patterns that led to the evolution and growth of the ecosystems and are able to identify effects on micro and macro levels (cf. Lindberg 2019).

Furthermore, we plan on conducting a motif analysis with both networks. A motif analysis identifies “recurring, significant patterns of interconnections” (Milo et al. 2002, p. 824). Thus, a network motif can be described as a repeating sub-graph in a network or across networks. By detecting network motifs, we aim to identify patterns that might hint at underlying relations between the modules (technical layer) and/or the developers (social layer) not accounted for in the cluster analysis. For motif analysis, we plan to use the implementation of Wernicke (2006)’s RAND-ESU algorithm of the Python library graph-tools (https://graph-tool.skewed.de/).

All these measures will then be used as predictors for our current main independent variable: growth of the ecosystem. In this specific context, we define growth as a combination of four indicators: first, we will utilize the number of downloads – for the platform core and associated modules – as an indicator for popularity. Second, we will include the number of collaborators for the platform core as an indicator for the growth of the core team (e.g., Setia et al. 2012). Third, we will use the number of lines of code added or deleted as an indicator for growth regarding the code base. Utilizing this multi-indicator approach enables us to observe an ecosystem’s growth on multiple dimensions (cf. Lindberg 2019). Fourth, we analyze the rate at which modules are added to the ecosystem, which not only is an indicator for its growth, but also for the rate of innovation in the ecosystem (Parker et al. 2017). With the help of correlation analysis or regression analysis, we plan then on demonstrating which of the above mentioned measures regarding social or technical networks predicts growth of an ecosystem. Especially by including a temporal dimension, we are able to identify cause and effect more clearly (cf. Lindberg 2019).

# Expected Results

Due to the explorative nature of this study, we remain open to different results and different explanations (cf. Lindberg 2019). However, based on existing work, we have some prior assertions, which we expect to see.

Answering our research question as stated above, we expect social networks, derived from the collaboration networks of modules, to have an effect on technical networks, derived from the dependencies of modules within ecosystems. In detail, we expect to see that if collaboration networks are of less modularity, dependency networks are as well. We expect this to occur because developers bring their own technical experience and knowledge into the projects they are working on – including modules they have used before. This means that if developers collaborate on different modules together, they might use the same dependencies across multiple modules. Hence, these choices made by one third-party developer might create trajectories for the remaining third-party developers (Boland et al. 2007), especially if they share the same digital habitat.

Regarding the interrelation of structure and technical dependencies, we see a priori no indication for the structure (i.e., collaborators) to follow the technical dependencies – we would, as done above, argue for an inverse effect (i.e., the technical dependencies to follow the collaborators). In our specific context, we imagine only marginal cases in which the social network (i.e., collaborators) changes due to changes in the technical network (i.e., dependencies) – for instance, only due to a help request from the collaborators of the including module towards collaborators of the dependency.

Further, we expect number of modules to correlate with download numbers, as a higher number of available modules might increase the value of an ecosystem for developers, as more modules might increase reuse and the chance to find a suitable module for a problem at hand.

# Future Plan

Until the International Conference on Information Systems in December 2019, we plan to accomplish multiple further steps towards our final goal. First, we plan to have completed data collection (i.e., have collected all data from GitHub and npm for all three ecosystems). This step includes cross referencing data across GitHub and npm (e.g., releases from npm and commits included in this commit from GitHub). As the data crawler has already been written, we expect this step to be completed around July 2019. Second, we plan to create an internal research memo on initial observations from the collected data. This step is important to have a clearer understanding of the structure of the data and to prepare for detailed quantitative analysis. We expect this step to be completed by October 2019. Third, we plan on conducting a detailed quantitative analysis before December 2019, which contents have been laid out in the previous section. Finaly, if time permits, we hope to have derived preliminary implications for research and practice based on the detailed quantitative data analysis.

# Discussion

## Risks and Challenges

Regarding risks and challenges for the described research project, we identified three main areas: technical challenges, analytical challenges, and challenges for contribution.

First, as we have a completed the programming of a data crawler, we only see one technical challenge. As GitHub is throttling and limiting requests to their API, we cannot download all the data we need in a single request or as fast as our Internet connection would allow. While this might be annoying, it is not a threat to the research project – it clearly delays the completion date of this research project, but does not threaten the success in general. Further, it is technically possible to use modules without downloading releases via npm by downloading the source code directly from GitHub. However, this is unlikely to occur often due to the ease-of-use and added benefits (e.g., automatic updates of dependencies) by using npm.

Second, and related to the first challenge, we expect the collected data set to be decently large – both in terms of database rows, due to the amount of releases and commits, and in terms of size (i.e., gigabytes), due to the included textual descriptors. The size of the data set is a challenge as it is more difficult to handle during analysis. However, as only limited manual analysis is needed (see section “Data Analysis”) and as computational resources are available to us, we do not expect the size of the dataset to threaten the success of this research project. Similar to the previous challenge, we—at worst—only expect a delay.

Third, we see challenges for our contribution. Without a qualitative analysis, meaning has to be derived by application of extant theories. We cannot rely on context-giving clues as can be done with qualitative approaches. Depending on our findings—for instance, if we cannot explain a finding—we might only be able to report some findings, without providing reasons or explanations. However, this may provide also a source for future contributions and in-depth studies.

## Expected Contribution

Based on the above described approach, we expect to contribute to the research on POSSE as follows: having insight into how social and technical networks (i.e., collaborators and dependencies) are interconnected, reveals interdependencies and enables conclusions on who drives which aspects in the development in and ultimately the growth of POSSE. Being able to explain why some ecosystems are more successful or are growing faster than others, based on the social and technical networks underlying these ecosystems, enables developers of such ecosystems to take corrective actions. In addition, we contribute to the growing stream of research analyzing digital platforms and their ecosystems by proposing a methdological approach for analyzing the socio-technical nature of these phenomenons using digital trace data. Further, this insight creates an avenue for future research, explaining in more detail, how such more successful networks can be created and maintained. Since firms are increasingly choosing “orchestration over production” when it comes to software code (Parker et al. 2017), we contribute to research on a growing phenomenon.

# Conclusion

Coming back to the research question posted at the beginning of this paper, we target the interactions of social and technical networks in platform-based open source ecosystems as our main objective. We plan on investigating these interaction with the help of the study of three popular ecosystems. In this paper, we argued why the social and technical networks might interact and described our plan on how to gather and analyse data to test our hypothesis.

# References

=1 ==10000 =10000

Baldwin, C. Y., and Clark, K. B. 2006. “The Architecture of Participation: Does Code Architecture Mitigate Free Riding in the Open Source Development Model?” *Management Science* (52:7), pp. 1116–1127.

Bavota, G., Canfora, G., Penta, M. D., Oliveto, R., and Panichella, S. 2013. “The Evolution of Project Inter-Dependencies in a Software Ecosystem: The Case of Apache,” in *2013 IEEE International Conference on Software Maintenance*, Eindhoven, Netherlands: IEEE, pp. 280–289.

Berente, N., Seidel, S., and Safadi, H. 2019. “Research Commentary—Data-Driven Computationally Intensive Theory Development,” *Information Systems Research* (30:1), pp. 50–64.

Boland, R. J., Lyytinen, K., and Yoo, Y. 2007. “Wakes of Innovation in Project Networks: The Case of Digital 3-D Representations in Architecture, Engineering, and Construction,” *Organization Science* (18:4), pp. 631–647.

Bosch, J. 2009. “From Software Product Lines to Software Ecosystems,” in *Proceedings of the Thirteenth International Software Product Line Conference (SPLC)*, San Francisco, pp. 1–10.

Crowston, K., and Howison, J. 2006. “Hierarchy and Centralization in Free and Open Source Software Team Communications,” *Knowledge, Technology & Policy* (18:4), pp. 65–85.

Cusumano, M. A., and Gawer, A. 2002. “The Elements of Platform Leadership,” *MIT Sloan Management Review* (43:3), pp. 51–58.

de Reuver, M., Sørensen, C., and Basole, R. C. 2018. “The Digital Platform: A Research Agenda,” *Journal of Information Technology* (33:2), pp. 124–135.

Eck, A., and Uebernickel, F. 2016. “Reconstructing Open Source Software Ecosystems: Finding Structure in Digital Traces,” in *Proceedings of the Thirty Seventh International Conference on Information Systems*, Dublin, pp. 1–13.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. 1996. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise,” in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, Association for the Advancement of Artificial Intelligence, pp. 226–231.

Gawer, A., and Cusumano, M. A. 2008. “How Companies Become Platform Leaders,” *MIT Sloan Management Review* (49:2), pp. 28–35.

Haefliger, S., von Krogh, G., and Spaeth, S. 2008. “Code Reuse in Open Source Software,” *Management Science* (54:1), pp. 180–193.

Hahn, J., Moon, J. Y., and Zhang, C. 2008. “Emergence of New Project Teams from Open Source Software Developer Networks: Impact of Prior Collaboration Ties,” *Information Systems Research* (19:3), pp. 369–391.

Hanseth, O., and Lyytinen, K. 2010. “Design Theory for Dynamic Complexity in Information Infrastructures: The Case of Building Internet,” *Journal of Information Technology* (25:1), pp. 1–19.

Jacobides, M. G., Cennamo, C., and Gawer, A. 2018. “Towards a Theory of Ecosystems,” *Strategic Management Journal* (39:8), pp. 2255–2276.

Kallinikos, J., Aaltonen, A., and Marton, A. 2013. “The Ambivalent Ontology of Digital Artifacts,” *MIS Quarterly* (37:2), pp. 357–370.

Lerner, J., and Tirole, J. 2002. “Some Simple Economics of Open Source,” *The Journal of Industrial Economics* (50:2), pp. 197–234.

Lindberg, A. 2019. “Developing Theory Through Integrating Human & Machine Pattern Recognition,” *Journal of the Association for Information Systems*.

MacCormack, A., Rusnak, J., and Baldwin, C. Y. 2006. “Exploring the Structure of Complex Software Designs: An Empirical Study of Open Source and Proprietary Code,” *Management Science* (52:7), pp. 1015–1030.

Messerschmitt, D. G., and Szyperski, C. 2003. *Software Ecosystem: Understanding an Indispensable Technology and Industry*, Cambridge, MA: MIT Press.

Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. 2002. “Network Motifs: Simple Building Blocks of Complex Networks,” *Science* (298:5594), pp. 824–827.

Parker, G., Alstyne, M. V., and Jiang, X. 2017. “Platform Ecosystems: How Developers Invert the Firm,” *MIS Quarterly* (41:1), pp. 255–266.

Roberts, J. A., Hann, I.-H., and Slaughter, S. A. 2006. “Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects,” *Management Science* (52:7), pp. 984–999.

Setia, P., Rajagopalan, B., Sambamurthy, V., and Calantone, R. 2012. “How Peripheral Developers Contribute to Open-Source Software Development,” *Information Systems Research* (23:1), pp. 144–163.

Singh, P. V., Tan, Y., and Mookerjee, V. 2011. “Network Effects: The Influence of Structural Capital on Open Source Project Success,” *MIS Quarterly* (35:4), pp. 813–829.

Stack Overflow. 2019. *Developer Survey 2019*, https://insights.stackoverflow.com/survey/2019/.

Tiwana, A. 2013. *Platform Ecosystems: Aligning Architecture, Governance, and Strategy*, Waltham, MA: Morgan Kaufmann.

Tiwana, A. 2015. “Evolutionary Competition in Platform Ecosystems,” *Information Systems Research* (26:2), pp. 266–281.

Tiwana, A., Konsynski, B., and Bush, A. A. 2010. “Research Commentary—Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics,” *Information Systems Research* (21:4), pp. 675–687.

Vaast, E., Safadi, H., Lapointe, L., and Negoita, B. 2017. “Social Media Affordances for Connective Action: An Examination of Microblogging Use During the Gulf of Mexico Oil Spill,” *MIS Quarterly* (41:4), pp. 1179–1205.

Wasserman, S., and Faust, K. 1994. *Social Network Analysis: Methods and Applications*, Structural Analysis in the Social Sciences, Cambridge, MA: Cambridge University Press.

Wernicke, S. 2006. “Efficient Detection of Network Motifs,” *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (3:4), pp. 347–359.

Yoo, Y. 2013. “The Tables Have Turned: How Can the Information Systems Field Contribute to Technology and Innovation Management Research?” *Journal of the Association for Information Systems* (14:5), pp. 227–236.

Yoo, Y., Henfridsson, O., and Lyytinen, K. 2010. “Research Commentary—the New Organizing Logic of Digital Innovation: An Agenda for Information Systems Research,” *Information Systems Research* (21:4), pp. 724–735.

1. Note: Angular needs to be differentiated from AngularJS, as Angular has emerged from a complete rewrite of AngularJS. [↑](#footnote-ref-1)