Measuring Tool and Resource Maturity in Developer Ecosystems

Koen van Baarsen, Slinger Jansen, Sergio España

Utrecht University, Princetonplein 5,
3584CC Utrecht, the Netherlands
koenvanbaarsen@gmail.com, slinger.jansen@uu.nl, s.espana@uu.nl

Abstract. Developer ecosystems are a set of software developers functioning as a unit and interacting with a shared market for software artefacts. Organizations that wish to develop their own developer ecosystems need to support developers with a comprehensive set of tools, to lower barriers to entry into the ecosystem and to raise productivity of developers in the ecosystem. In this paper a comprehensive tool framework is provided and used to evaluate eight developer ecosystems. The cases serve as an illustration for organizations that want to further improve the experience and productivity of its developers.

Keywords: Developer Ecosystem, Unified Developer Environment, Mobile Ecosystems, Data Integration Platforms, Application Platforms

1 Introduction

Research context. Software engineers working on modern connected apps and services require a large and varied set of resources to develop their applications. Luckily no software developer works in a vacuum; developer ecosystems provide developers with resources that they require for their work, thereby increasing developer productivity.

Developer ecosystems were previously defined by Jansen as *"a set of software developers functioning as a unit and interacting with a shared market for software artefacts.*" [1] This is the definition for developer ecosystems that is used in this study. Developer ecosystems are similar to software ecosystems, which are defined by Jansen as *"a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them."* [2] While different researchers have created different definitions of software ecosystems, they all agree that three core components are required in a software ecosystem: a social aspect that requires different actors to form a community, a technical aspect that allows ecosystem artefacts to influence each other, and an economical aspect that defines that they interact with a shared market. [3]

Just like software ecosystems have a social aspect, so do developer ecosystems. The difference between developer ecosystems and software ecosystems is that the community of a developer ecosystem consists of software developers, while the

community of software ecosystems contains more types of actors. For example, domain experts and users can also be part a software ecosystem's community. [3]

Examples of technical resources that can be available in developer ecosystems are flexible programming languages, IDEs, online documentation- and discussion portals, and tools and services for app distribution and monetization. All these technical resources together, form the unified developer environment of a developer ecosystem. The term unified developer environment is defined in this paper as "*the tools, services, and resources available to developers within a developer ecosystem*". Figure 1 shown below, illustrates the concept of the unified developer environment.

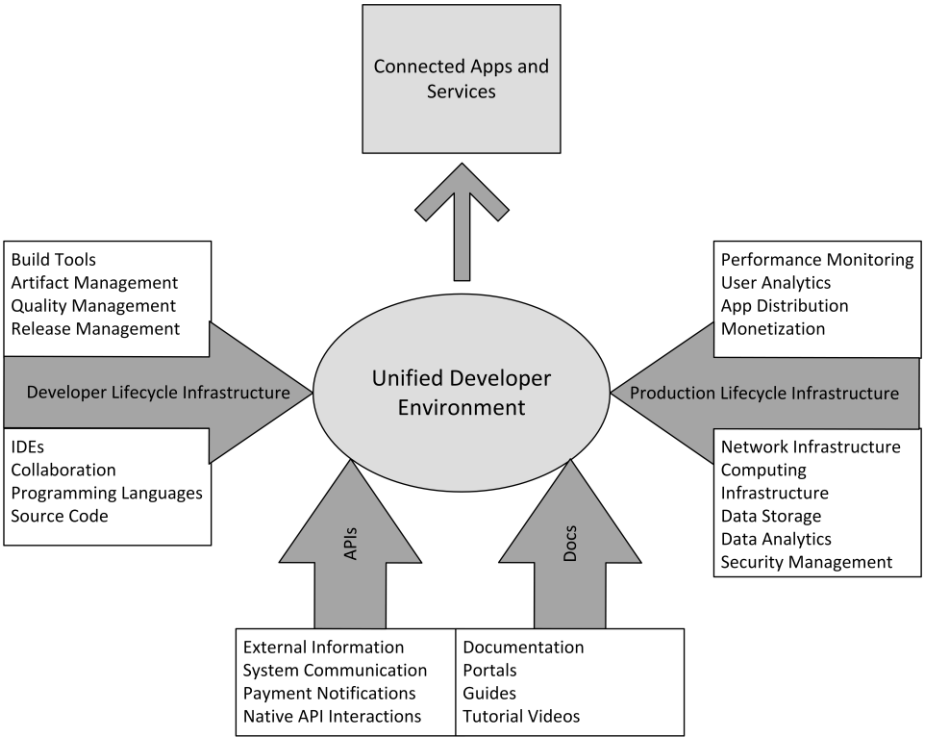

Figure 1. Diagram of the unified developer environment

Research problem analysis. There has been much research into how the quality of software ecosystems can be analysed, and how their quality has developed over time. Two years ago, a survey of 26 researchers in the software ecosystem domain identified this topic as one of the main areas in which more research needs to be done. [3]

However, this research only focussed on the analysis of software ecosystems. No studies focussing on the quality and historical development of developer ecosystems were done. This has resulted in a lack of knowledge about the characteristics of developer ecosystems and the lack of artefacts to analyse or manage them.

This study was started as an attempt to compare different developer ecosystems, based on the value that they provide to their developers. Developers do technical work, and therefore rely the most on the technical aspects that developer ecosystems provide, trough unified developer environments. For this reason, the different developer ecosystems are evaluated based on the completeness of their unified developer environments. In order to do this across varying domains, a framework was created. The framework splits up the unified developer environment into different tooling and resource needs that can be provided to developers by the developer ecosystems. This makes it possible to directly compare different developer ecosystems in varying domains. With the help of this framework, a case study was executed on the leading developer ecosystems, providing us with an overview of the strengths and weaknesses of these ecosystems with respect to the tools, services and resources that they contain.

Analysing the added value of developer ecosystems, is useful for the same reason as analysing the health of software ecosystems: it provides insight into where the ecosystems are going in the future, and gives feedback on changes that have been previously made in the ecosystem. [4] It also educates new developer ecosystem entrants on the unified developer environment that their newly established developer ecosystem needs to provide in order to competitive.

Research questions. In order to realize the research goals described above, three different research questions were identified and answered.

The first question was with regards to how the comparison framework could be created. A deeper understanding of the different elements the unified developer environment consists of was required for designing the framework. A list of tool and resource categories that the unified developer environment consists needed to be established.

The next question in the study, was which developer ecosystems the study needed to focus on. Because it was not possible to analyse all developer ecosystems, research was done on which ecosystems were to be included in the study. The main criteria for including developer ecosystems, was their size, and their market competitiveness according to the Gartner Magic Quadrants. [5,6,7].

The final research question of the study, was with regards to the strengths and weakness of each developer ecosystem that was included in the study. An easily comprehensible comparative overview of the strengths and weaknesses of the different developer ecosystems is what was desired as the end-result of the final phase of the study.

2 Research Methods

The study was structured using Wieringa's empirical cycle [8]. The research problem analysis was described in section 1. In this section, the research and inference design is explained. In sections 3 – 6, the results of the data are discussed and analysed.

This study is divided into three distinct phases, with different research goals and results for each. As such, the research methods are described by research phase.

Phase 1: Identifying the main tool categories. In the first phase of the study, categories for classifying specific tools, services and resources in the developer ecosystem were identified. The categories are generic, so that they are applicable regardless of the specific ecosystem being analysed. As an example, it allowed a comparison between the tools available to developers in mobile ecosystems like Apple's iOS or Android, to the tools available in cloud platforms like SAP HANA or the IBM Cloud.

A literature study provided the insight required to group the tools used by developers into different categories that met these requirements. Searching for literature online turned out to be the most effective way of obtaining information about software development. There is a large community of businesses and developers sharing their knowledge on the internet, through scientific publications, and personal or corporate websites and blogs. Although some of these sources are not completely scientific, they allowed us to analyse the way in which software developers work. The tool categories were generated based on the activities that developers do in their work. The reasoning behind this it that the task of tools offered in the developer ecosystems is to help software developers in their work. This means that the categories we generated for the different types of work developers do, can also be used to classify the tools that help them do those specific types of work.

Phase 2: Identifying the leading ecosystems. The goal of the second phase of the study, was to identify the leading developer ecosystems. Gartner's Magic Quadrants proved particularly useful, and were the main source of information for this phase of the study [5,6,7]. The Magic Quandrants are designed for analysing market trends, and show which competing products are leading in certain areas. Quartner has also created these Magic Quadrants for different types of developer ecosystems.

Sadly, it was not feasible to discuss all developer ecosystems identified in the Gartner literature. For this reason, we have chosen to do convenience sampling. The developer ecosystems that we rationally deemed most important were included in the study.

Phase 3: Case study for each important ecosystem. The purpose of the third phase of the study, was to compare the different ecosystems based on tool availability. The comparison is done based on what tools are available in each ecosystem, and how the developers can use these.

According to Yin, a research design based on case studies is the best approach for researching "how" questions like how the different ecosystems provide value for developers, because the different cases can be studied in-depth and within their real-life context. [9]

Again, online data sources like websites were the best source of information for this phase of the study. The developer documentation for the different platforms described all the first-party tools available to developers. Other sources described the third-party tools that were available to developers for cases where first-party tools were not available in specific tool categories. References to the websites of the owners of developer ecosystems are not included in this paper due to space constraints. Many different webpages on the sites of the ecosystem owners were used in the case studies.

3 Research Execution: Designing the Unified Developer Environment

In this section, the results of the study will be discussed phase-by-phase.

Recurring development phases. *"The number of phases within a project's lifecycle is based on the characteristics of a project and the employed project management methodology"* [10]. These development phases can be completed once, or multiple times through different iterations. However, they are different steps with different tasks.

Different sources indicate that there are 6 or 7 distinct phases in the software development lifecycle, depending on whether initiation and planning are considered as one phase or as two separate phases in the software development lifecycle. [10,11]. In this study, the initiation and planning phase are grouped together. The resulting development phases are illustrated in Figure 2, shown below.

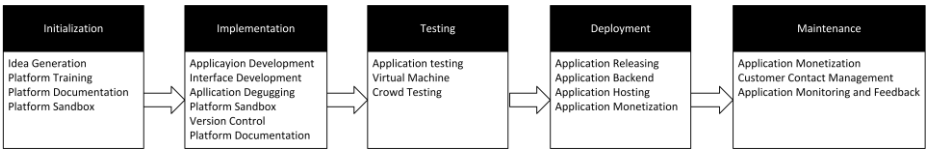

Figure 2. The recurring development phases.

In the *idea generation phase*, the idea of the application is figured out. What should the application do? Who will be using the application? How will they be using the application? What are the business requirements? These are questions that can be answered in the requirement gathering and analysis phases of the software development process.

In the *design phase*, the development process will start to get more technical. Developers can for example think about what the technical requirements of the application will be, or what a good software architecture would be like for the application.

In the *implementation phase* of the software development process, the application will be built. The way developers can build their applications, depends on the specific ecosystem being analysed. As an example, developers creating an iOS app can code their app in Swift. By contrast, developers working in the Oracle Cloud Integration ecosystem, can build their applications through a point-and-click software development tool.

The *testing phase* of the software development process ensures that the application meets the requirements set out in the idea generation and design phases. Ensuring that the requirements are met is done in a broad sense with varying types of tests. For example, this phase includes unit tests that are done to ensure small components of the application work correctly for all possible input but also includes the final user acceptance testing.

In the *deployment phase*, the application is made available for the end-user. From this point onwards, customers can actually interact with the application.

The *maintenance phase* includes tasks like bug fixing or application monetization.

Identifying the tool categories. Having established broad categories for the different tools based on the different phases of the development process they fall into, the next step is to identify specific tasks that developers need to be able to complete within these categories. With these tasks, binary comparisons can be made between the different platforms. Either a platform provides the tools required to for a developer to complete a given task, or it does not.

For every category, we have listed tasks based on our own experiences in the software development industry. These tasks are included in diagram 2 of the previous section.

The *initialization phase* is a creative aspect of the software development process. Different developers take different approaches to completing this phase of the development cycle. As such, it is difficult to establish a set list of tasks that need to be completed in order to mark this phase as done. When comparing different developer ecosystems, it is more useful to look at whether they provide tools for idea generation than what specific approach is taken. Therefore, this category is not split up into specific tasks that an ecosystem can provide tools for. Any type of idea generation & design tool provided by a developer ecosystem will be perceived as providing tools for this category.

In the *implementation phase* of the development cycle, differences in the tasks that need to be completed start to emerge. The task of developing an application is significantly different from the task of designing a visual interface or the task of searching for documentation. With this task-based approach of identifying tool subcategories, we have identified the following subcategories: application development; interface development; application debugging; platform sandbox; version control; platform documentation; platform training.

With regards to the *testing phase*, there is also a large difference in the tasks tools aim to help developers with. Applications can be tested as their own independent entity, on machines, and as they are being used by users. In the same order, the following tool subcategories help developers work on those tasks: application testing; virtual machine; crowd testing.

It is also possible to identify three different tool categories for the *deployment phase* of the application development cycle: application releasing services; application backend services; application hosting services.

Finally, we have identified the following tool subcategories for the *maintenance phase*: application monetization; customer contact management; application monitoring and feedback.

4 Research Execution: Identifying the leading ecosystems

Establishing ecosystems for the case study. Having established the tool categories that can be used, the next step is to determine the developer ecosystems that this study should focus on.

The selection of ecosystems was made using a long-list short-list system. We started by looking for sources citing the leading ecosystems. These were filtered later

based on whether they were still relevant at the time of writing and whether they were large enough to offer the kind of tools we are looking at in this study.

Currently, the development of software platforms is mainly focussed on two main areas: mobile platforms and cloud platforms. Therefore, these two areas are also the ones that will be analysed in this study.

Mobile ecosystems. According to research by Gartner, 99.6% of all new smartphones ran either Android or iOS [12]. The same source also shows that Blackberry has a (rounded) 0.0% market share making that platform irrelevant. Finally, Windows Phone – the third biggest mobile platform – is facing a quarter over quarter decline of 45.2%. This decline is expected to continue, as Microsoft is focusing more on business users and less on the consumer market [13].

Based on this data, it is plain to see that the ecosystems of Android and iOS are the leading mobile ecosystems. Therefore, these are also the ecosystems that this study is focused on.

Cloud ecosystems. There is a wide variety of services available to developers trying to utilize the power of the cloud. This makes it difficult to create a list of the leading ecosystems for the developers of cloud application. Depending on the requirements of the application, different services can be useful. For one application, an ecosystem focused on providing infrastructure resources might be best, while for others ecosystems specialized in data integration services might the better option. Finally, sometimes requirements can be fulfilled using an off-the-shelf application platform service.

In order to not generalize this complex market, it is best to look at each of these cloud service categories individually. The Gartner Magic Quadrants provide a quick overview of the dominant players in each developer ecosystem category [5,6,7].

The easiest elimination from the magic quadrants are the cloud infrastructure service providers. While the services they provide are vital to developers, these services to not form a comprehensive cloud ecosystem. Rather they provide practical solutions for running applications.

With regards to Application Platform services, Salesforce and Microsoft are the two only leaders in that Magic Quadrant. As they are the leaders, their aPaaS products should be included in the comparison that will be done in this study. Salesforce offers two main types of services on their two websites: salesforce.com and force.com. Force.com is the platform where developers can develop their own applications. Salesforce.com hosts CRM applications that are developed and running on the Force.com platform. This study will focus on the force.com platform in the Salesforce ecosystem, as this is the platform where developers can actually develop custom applications.

Gartner identified more leaders for Data Integration Tools in the Magic Quadrant of that service discipline [3]. The main criteria whether those service providers should be included, is whether they offer developers access to a real ecosystem. SAP, IBM, and Oracle provide the biggest ecosystem in this area of the market. Therefore, the scope of the Data Integration services section of this study, is limited to the products of those companies.

Industry overview. Before analysing the selected ecosystems further, it is useful to develop an understanding of the size of the enterprises that own the discussed platforms. Based on data gathered from press releases from the different platform

owners, the industry overview in table 1 was generated. One of the key observations that can be made from this overview, is that all platforms owners are large enterprises with large revenues to match. Another easily-made observation is that most successful developer ecosystems have been released in the last ten years. Only Salesforce has released a version of their platform earlier than 2007.

	Mobile ecosystems		**Integration Platform ecosystems**			**Application Platform ecosystems**	
	Apple iOS	Android	SAP HANA	IBM Cloud	Oracle Cloud (Integration)	Salesforce	Microsoft Azure
Platform owner	Apple	Alphabet	SAP SE	IBM	Oracle Corporation	Salesforce.com	Microsoft
Revenue over FY 2016	$215.639B	$90.27B	$22.1B	$79.919B	$37.04B	$8.39B	$85.32B
Number of employees	80.000	75.606	84.183	380.000	136.263	25.178	124.293
Founded	1976	1998	1972	1911	1977	1999	1975
Platform initial release	2007	2008	2010	2007	2015	2000	2010

Table 1. Industry Overview of the discussed ecosystems.

5 Research Execution: Case-study on the leading ecosystems

The goal of the final phase of the study was analysing the different selected ecosystems, based on the tooling and resources provided in their unified developer environments. These case studies focussed on discovering which tools were available in the ecosystems, what made them unique, and whether they were developed by the ecosystem owners or by third party developers.

For every ecosystem and every tool category, online developer documentation was analysed in order to determine whether there were tools or resources available that fit into the different categories. This documentation was sourced from both the developer documentation portals provided by the platform owners, and websites of third party developers and businesses. Using both first and third party data sources was required, because in developer ecosystems and software ecosystems, many actors interact by providing software artefacts. They are not just provided by the platform owners.

Many different websites were required in order to do this analysis. Due to space constraints, these references are not listed. Examples of first party developer documentation portals that were used are *"developer.android.com"* or *"msdn.microsoft.com"*.

This information gathered in this case study was aggregated in a single table, that represents an abstract view all the results of the case study.

The different categories can be fulfilled in three ways. If a platform owner has developed tools or resources that fulfil the developer need, this is represented in the table as a 1st party need fulfilment. If third party developers not affiliated with the platform owners have developed the tools or resources, this is represented in the table as a 3rd party need fulfilment. Finally, if no tools or resources are available in a certain need category, this is illustrated with a dash in the table.

The needs belonging to a specific category can be fulfilled by one tool, but also by multiple. Only that the need is fulfilled is represented in the result of the case study.

Finally, when both third and first party tools and resources are available, the table only shows that first party tools are offered.

The results of the case study are displayed in table 2 shown below. The results are analysed in-depth in the discussion in section 6.

		Mobile ecosystems		**Integration Platform ecosystems**			**Application Platform ecosystems**	
		Apple iOS	Android	SAP HANA	IBM Cloud	Oracle Cloud	Salesforce	Microsoft Azure
Initialization	Idea generation	1st	3rd	-	1st	1st	1st	1st
	Platform training	1st	1st	1st	1st	1st	1st	1st
	Platform documentation	1st	1st	1st	1st	1s	1st	1st
	Platform sandbox	-	-	1st	1st	-	1st	1st
Development	Application development	1st	1st	1st	1st	1st	1st	1st
	Interface development	1st	1st	1st	3rd	-	1st	-
	Version control	3rd	3rd	3rd	1st	-	3rd	1st
	Application debugging	1st	1st	1st	1st	1st	1st	1st
	Application testing	1st	3rd	3rd	1st	N.A.	3rd	1st
	Virtual machine	1st	1st	N.A.	1st	N.A.	N.A.	1st
	Crowd testing	1st	1st	N.A.	N.A.	N.A.	N.A.	-
Deployment	Application releasing	1st	3rd	1st	1st	1st	1st	1st
	Application backend	-	1st	1st	1st	1st	1st	1st
	Application hosting	-	1st	1st	1st	1st	1st	1st
Live apps	Application monetization	1st	1st	-	-	-	1st	1st
	Customer contact management	-	1st	-	-	-	1st	-
	Application monitoring and feedback	1st	1st	1st	1st	1st	1st	1st

Table 2. Developer ecosystem case-study results

6 Data Analysis

6.1 Mobile ecosystem analysis

Focus on user experience. The largest observed difference between the mobile- and cloud ecosystems, was that there is a stronger focus on design and user experience on the mobile platforms. Previous research by Microsoft has established that user interface inefficiencies are the leading reason for users to abandon mobile apps [14]. Additionally, previous research has proven that properties like user

interface complexity have an impact on the user-perceived quality of mobile applications [15]. Indeed, before the creation of mobile ecosystems, user satisfaction was already identified as one of the 5 attributes of usability [16].

When placed in this scientific context, it easy to see that actors in the mobile developer ecosystems have a vested interest in making apps that provide their users with an enjoyable experience. Apple and Google have also made this observation, and provide developers with tools and resources that enable them to rapidly create applications that provide a consistent user experience with the rest of the ecosystem. Examples of such tools and resources that were identified are graphical interface builders and comprehensive design guidelines.

Monetization opportunities. Another key observation about the mobile ecosystems, is that monetization opportunities are offered through the app stores. App stores allow developers to sell and distribute their products to actors within one or more multisided software platform ecosystems [17]. As the app stores matured and developers and businesses discovered how to utilize them, new App Business Models were developed based on the different avenues in which developers can monetize their apps, providing businesses with new ways of generating revenue [18].

Facilitation of developer-customer relationship. Another similarity between both analysed mobile ecosystems, is that they are designed to bring developers and app users closer together. Customers can for example use app reviews to give feedback to developers. In the Google Play Store, developers can even reply to reviews in order to better respond to customer feedback. Pagano and Maleej have done research on the types of feedback that users give with their reviews. They concluded that while part of the feedback is superficial, some reviews include useful comments, bug reports, user experience and feature requests. However, they also observed that it is difficult to use reviews for deriving requirements or prioritizing development goals, because the data cannot be systematically filtered, aggregated or classified [19].

Another area where the close developer-user relationship is visible, is in the possibility of crowd testing. With crowd testing, developers can release new versions of their applications to a select group of customers, who can provide feedback on what they like and dislike about the new app version. Researchers have recently proposed an App Store 2.0, which further exploits crowdsourcing to increase the overall quality of mobile apps. Their *"key insight is that app stores should leverage the different types of crowds to which they have access – crowd of apps, crowd of devices and crowd of users."* The app store that they envisioned is able to do 5 important tasks*: "tell the developer about performance issues and regressions happening in the wild"*, *"automatically synthesize tests that reproduce issues happening on specific devices or configurations only"*, *"automatically infer the root causes of crashes such as those related to permissions"*, *"warn app store administrators about bad apps that harm the app store reputation"* and *"patch apps to prevent the occurrence of previously observed crashes."* [20] It is important to note that in this proposed app store, all users are part of this crowd-source, rather than a select group as is the case with current crowd testing techniques.

6.2 Data Integration ecosystems analysis

Case-studies instead of visual idea generation. The type of applications that can be developed in the data integration ecosystems are not suited for visual brainstorming; their value stems from the insight that can be generated by combining data from different data sets. To inspire developers with ideas for new applications, the platform owners are showcasing how other organizations have created value with their tools and resources. A common way in which this was done, was with the help of case studies. Rather than visually demonstrating what types of applications can be developed, the added business value is highlighted in these case studies.

Lack of app stores. After seeing how app stores can provide value within a developer ecosystem, it was disappointing to observe that there were no app stores or data markets for the studied data integration platforms. Various developers might want to gather data from the same types of datasets or do similar types of data analysis, but in the current data integration landscape they are not able to collaborate on those tasks. More research should be done on how developers can share and monetize the data integration applications or application components that they have developed. Such exchanges might lead to new sources of revenue, and time savings during application development.

Cloud computing. All studied data integration ecosystems were extensively integrated with cloud infrastructure services. In order to understand this phenomenon, it is important to understand what data integration is, and how it creates value for businesses. Lenzerini has defined data integration as *"the problem of combining data residing at different sources, and providing the user with a unified view of these data"*. [21] The datasets that businesses want to analyse are continuously growing larger, resulting in increasingly computationally intensive data analysis tasks. Doing these computations on on-site appliances, requires business to purchase and maintain large server farms.

6.3 Application Platform ecosystem analysis

B2B application stores. The two biggest application platform ecosystems – Microsoft Azure and Salesforce – both include a business to business app store. Salesforce already allowed developers to exchange application components back in 2006 [22].

According to research by Baek and Altmann, the collective intelligence that app exchanges enable, allow third party developers to gradually become innovation leaders within a developer ecosystem [23]. As a result, third party developers can increase the value of the platform, which is a positive development for the platform owners. It is a win-win situation for both the platform owners and third party developers.

Cloud computing. Just like the data integration platforms, the Application Platform ecosystems that were studied also ran on the cloud. This is a core aspect of PaaS ecosystems, as is evident by Hashem et al.'s definition of PaaS: *"PaaS, such as Google's App Engine, Salesforce.com, Force platform, and Microsoft Azure refers to*

different resources operating on a cloud to provide platform computing for end users." [24] These developments allow developers in the application platform ecosystems to benefit from the same advantages of cloud computing that were observed in the other developer ecosystem categories.

6.4 Similar developer tool maturity levels

For all discussed developer ecosystems, we observed that the tool maturity levels were similarly high. There was similar tool availability for the different tool categories in the previously designed framework. This is in line with previous findings by Jansen and Bloemendaal from a case-study on mobile app-stores [17]. They did a case-study on 6 different app-stores in order to create a definition and conceptual model of app-stores, and observed that the app stores were similar with regards to what features have been implemented. The same observation can be made for the tools and services available in the different developer ecosystems analysed in this study. All analysed ecosystems provide developers with similar features in their unified developer environments.

More research into why competing software products seem to offer similar features would be valuable. It is possible that there is a reason why both different app stores and different developer ecosystems have such small differences between them with regards to feature or tool availability.

7 Threats to the Validity

Two issues affecting the validity of this research were identified.

Lack of developer input. No input from developers was received during this study. However, such input from developers could have taught us valuable lessons. It is for example interesting to see if other developers agree with the tool categories that we have established, or to learn that we have missed key tool categories. Another example where developer feedback would be appreciated is with the selection of developer ecosystems that was made. Other developers might have different insights on which developer ecosystems are market leaders in the current software development landscape.

Ecosystem selection. The case-study design used in this study is not suitable for analysing a large collection of developer ecosystems. As such, the focus was on the leading developer ecosystems. Focussing on the smaller selection of ecosystems might have prevented us from observing innovations in ecosystems hat were not analysed. Smaller, less successful ecosystems, could still contain new and innovative tools. Based on the selection criteria that we have made these innovations would not be picked up on. We consider the exploration of different ecosystems as future work.

8 Future Work

Factors affecting the success of developer ecosystems. In this study, the focus was on creating a comprehensive framework that provides a deeper understanding of the current software development landscape. Because the study only included the leading developer ecosystems, there is a strong survival bias – developer ecosystems only become large if they are successful. Another study, focussing on what criteria played a role in the success or failure of developer ecosystems would be valuable, and could lead to a more comprehensive framework with which to benchmark developer ecosystems by.

App stores for data integration platforms. It would be valuable to do research on what app stores for data integration platforms could be like, because all discussed data integration ecosystems do not have app stores. Are there any data integration platforms that have successfully introduced an app store for example? Or if not, how would such an app store add value for the developers in the ecosystem?

The lack of app stores for the data integration platforms is different to the pattern that we have observed for other tool categories in this study, and differences in features in app stores by another study done by Jansen [17]. It would be valuable to do more research into why competing developer ecosystems tend to be so similar, and what causes the data integration platforms to be an outlier.

Expanding the framework for software ecosystems. In this work, a software ecosystem is defined as: *"a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them."* [25]

The framework generated in this study, was designed for comparing developer ecosystems. However, it might be possible to make the framework applicable to software ecosystems by expanding it. The framework is largely technical in nature, and a large part relates to the quality of the software available to developers.

Earlier research has established that the health of a software ecosystem is determined by its actors, its software, and its orchestration [26]. The framework is already useful for comparing the state of software within an ecosystem. If criteria for actor health and orchestration health are also included, the framework can also be used to do case studies on software ecosystem health.

9 Conclusion

The main conclusion of this study is that all successful developer ecosystems have an extremely mature developer toolset. However, it is difficult to speculate on whether these tools have led to the success of the ecosystem, or were developed because of the success of the ecosystem. It does however make sense that the most mature developers ecosystems, also have the most mature tooling and resource availability.

Furthermore, it was interesting to see that there were no application stores or data stores for the studied data integration platforms, despite the type of applications offered in this app store being suited for such an exchange.

The results of this study are interesting to the owners of currently active developer ecosystems, or parties interested in starting a new developer ecosystem. The designed framework is a useful benchmark for analysing the selection of available tools and services within the ecosystem. Actors active within a developer ecosystem can for example see in which areas competing developer ecosystems have flourished and where competing developer ecosystems are lacking. Developers interested in starting a new developer ecosystem, can see beforehand what tools and services are required for the ecosystem to be considered competitive. They can use this information to decide whether establishing a new developer ecosystem is worth the cost, and what tools and services must be invested in. The cost of creating a comprehensive developer ecosystem might be prohibitively expensive to most software developers.

The overview can also be useful for developers choosing or switching developer ecosystems. Different ecosystems might provide different types of tools and services. Using the overview that will be created in this study, developers can identify which developer ecosystem is most suitable for the types of applications that they want to develop.

Finally, software development businesses can also choose to focus on obtaining expertise in certain ecosystems, based on which ecosystems fit their business strategies best. Businesses focusing on enterprise application development for example, might benefit from certain enterprise grade tools or services that are not available in all platforms.

With this paper we are closer to achieving a comprehensive insight into the current development landscape in different developer ecosystems. There is a long road ahead, but the results for this small selection of developer ecosystems is promising.

References

1. Jansen, S.: Opening the Ecosystem Flood Gates: Architecture Challenges of Opening Interfaces within a Product Portfolio. In: European Conference on Software Architecture. Springer, Cham, 2015. p. 121-136. (2015)
2. Jansen, S., Finkelstein, A., Brinkkemper S.: A sense of community: A research agenda for software ecosystems. In: Software Engineering-Companion Volume, 2009. ICSE-Companion 2009. 31st International Conference on. IEEE, 2009. p. 187-190. (2009)
3. Serebrenik, A., Mens, T.: Challenges in software ecosystems research. In: Proceedings of the 2015 European Conference on Software Architecture Workshops. ACM, 2015. p. 40. (2015)
4. Manikas, K., Hansen, K.M.: Software ecosystems–A systematic literature review. Journal of Systems and Software, 2013, 86.5: 1294-1306. (2013)
5. Vincent, P., Natis, Y.V., Iijima, K., Thomas, A., Dunie, R., Driver, M.: Magic Quadrant for Enterprise Application Platform as a Service, Gartner (2016)
6. Beyer, M.A., Thoo, E., Zaidi, E., Greenwald, R.: Magic Quadrant for Data Integration Tools, Gartner (2016)
7. Leon, L., Petri, G., Gill, B., Dorosh, M.: Magic Quadrant for Cloud Infrastructure as a Service, Gartner (2016)
8. Wieringa, R.J.: Design Science Methodology. pp. 109--119. Springer (2014)
9. Yin, R.K.: Case Study Research Desgin and Methods (1994)
10. Ragunath, P.K., Velmourougan, S., Davachelvan, P., Kayalvizhi, S., Ravimohan, R.: Evolving A New Model (SDLC Model-2010) For Software Development Life Cycle

(SDLC), In: International Journal of Computer Science and Network Security, 2010, 10.1: 112-119. (2010)

11. ISTQBExamCertification: What are the Software Development Life Cycle (SDLC) phases?, http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/ (Retreived 2017)
12. Gartner: Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016, http://www.gartner.com/newsroom/id/3609817 (2017)
13. IDC: Smartphone OS Market Share, 2016 Q3, http://www.idc.com/promo/smartphone-market-share/os (2017)
14. Karlson, A.K., Iqbal, S.T., Meyers, B., Ramos, G., Lee, K., Tang, J.C.: Mobile Taskflow in Context: A Screenshot Study of Smartphone Usage. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, 2010. p. 2009-2018. (2010)
15. Taba, S.E.S., Keivanloo, I., Zou, Y., Ng, J., Ng, T.: An Exploratory Study on the Relation between User Interface Complexity and the Perceived Quality. In: International Conference on Web Engineering. Springer, Cham, 2014. p. 370-379. (2014)
16. Nielsen, J.: Usability Engineering. Morgan Kaufmann Publishers. Inc.–1993, 1994. (1994)
17. Jansen, S., Bloemendaal, E.: Defining App Stores: The Role of Curated Marketplaces in Software Ecosystems. In: International Conference of Software Business. Springer, Berlin, Heidelberg, 2013. p. 195-206. (2015)
18. Tang, A.K.Y.: Mobile app monetization: App business models in the digital era. International Journal of Innovation, Management and Technology, 2016, 7.5: 224. (2016)
19. Pagano, D., Maleej, W.: User Feedback in the AppStore: An Empirical Study. In: Requirements Engineering Conference (RE), 2013 21st IEEE International. IEEE, 2013. p. 125-134. (2013)
20. Gomez, M., Adams, B., Maleej, W., Monperrus, M., Rouvoy, R.: App Store 2.0: From Crowd Information to Actionable Feedback in Mobile Ecosystems. In: IEEE Software, Institute of Electrical and Electronics Engineers, 2017, IEEE Software - Theme Issue on Crowdsourcing for Software Engineering, 34, pp.81-89 (2017)
21. Lenzerini, M.: Data Integration: A Theoretical Perspective. In: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. ACM, 2002. p. 233-246. (2002)
22. Barros, A.P., Dumas, M.: The rise of web service ecosystems. IT professional, 2006, 8.5: 31-37. (2006)
23. Baek, S., Kim, K., Altmann, J.: Role of Platform Providers in Service Networks: The Case of Salesforce. com App Exchange. In: Business Informatics (CBI), 2014 IEEE 16th Conference on. IEEE, 2014. p. 39-45. (2014)
24. Hashem, I.A.T., Yaqoob, I., Anuar, N.B., Mokhtar, S., Gani, A., Khan, S.U.: The rise of "big data" on cloud computing: Review and open research issues. Information Systems, 2015, 47: 98-115. (2015)
25. Jansen, S., Brinkkemper, S., Finkelstein, A.: Business Network Management as a Survival Strategy: A Tale of Two Software Ecosystems. In: IWSECO@ ICSR. 2009. (2009)
26. Manikas, K., Hansen, K.M..: Reviewing the Health of Software Ecosystems – A Conceptual Framework Proposal. Proceedings of the 5th International Workshop on Software Ecosystems (IWSECO). (2013)