Ecosystem Health of Cloud PaaS Providers

Garm Lucassen, Kevin van Rooij, Slinger Jansen

Department of Information and Computing Sciences, Utrecht University,Princetonplein 5, 3584 CC Utrecht The Netherlands
{g.g.lucassen,k.a.vanrooij}@students.uu.nl,slinger.jansen@uu.nl

Summary. Customers of Platform as a Service providers are unable to evaluate the risk of their provider going bankrupt. Lacking this information, businesses are effectively putting their critical business services in jeopardy. In this paper, we present a method to evaluate the ecosystem health of eight different PaaS providers. The results of our research enable businesses and individuals to make an informed decision on what PaaS providers to do business with. Additionally, the PaaS providers themselves are given insight into the current state of their ecosystem compared to competitors.

Key words: Software Ecosystems, Cloud PaaS, Ecosystem Health, Open Source, Repository Mining, Platform as a Service

1 Introduction

Since 2009, academic and business interest in the Platform as a Service (PaaS) industry has grown exponentially. Yearly Google Scholar hits for the keyword "Platform as a Service" has grown from 83 in 2008 to 3320 in 2012.[1] Gartner expects enterprise public cloud services spending to reach $207 billion by 2016 [1]. Due to this surge in attention, the United States National Institute of Standards and Technology added a definition of PaaS in 2011:

> *"The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment."* [2]

Concretely put, PaaS providers host virtual systems that run application stacks facilitating easy deployment of a web-enabled program in a specific programming language, significantly reducing development time and effort for the customer. Key advantages include: dramatically lowering the cost of entry, almost immediate access to hardware resources, lowering IT barriers to innovation, flexibly scalable services to meet client demand and enabling new classes of applications

[1] http://scholar.google.com/scholar?q="platform+as+a+service"

and services [3]. Despite this explosive growth in both academic and business relevance, no publicly accessible information is available on the ecosystems health of public cloud PaaS providers. Ecosystem health is defined as *"long-term financial well-being of the business ecosystem and the long-term strength of the network"* [4]. Access to knowledge regarding ecosystem health is crucial for businesses researching the possibility to move their software to the cloud for two reasons [**?**]. To begin, businesses attempt to seek partners with a robust business ecosystem [5]. Second, the availability of their critical business applications relies on the robustness of their PaaS Provider. Unexpected bankruptcy of a PaaS provider can lead to loss of data or catastrophic downtime. Indications of these kinds of developments are preferably known beforehand.

Because PaaS providers are primarily software businesses, this paper leverages knowledge on the academic field of software ecosystems to study their ecosystem health. Several authors have defined the term software ecosystem [6, 7], but this research applies the following definition by Jansen et al., as it builds upon earlier definitions and further abstracts the concept [8]: *"a set of actors functioning as a unit and interacting with a shared market for software and services, together with the relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources and artifacts."* A high profile example of a strong software ecosystem is iOS. Apple benefits from the popular App Store, due to which the iOS ecosystem remains the market leader in terms of app availability, app sales and profit margins [9].

In the context of PaaS ecosystem health, the contributors to the long-term strength of the ecosystem are the direct users of the PaaS technology: developers. If a group of developers is actively contributing to the development of PaaS or its extensions, the PaaS itself is more likely to succeed in the long run. This paper uses data of the open source code hosting service GitHub to analyze the ecosystem health of eight different PaaS providers. First, metadata of all software projects or *repositories* that contribute to the ecosystems of each PaaS providers is collected. Next, specific data fields are aggragated into ecosystem health results based on open source ecosystem health measures introduced by Crowston et al. [10]. Finally, statistical analysis extracts relevant insights on the current growth expectations of a PaaS and expected health of their ecosystem.

With the results of our research, businesses will be able to make a more informed decision in choosing a cloud hosting PaaS provider. Moreover, the PaaS providers are presented with a method to gain insight into the state of their ecosystem. This can help them achieve their goals, make better use of available resources, reduce risks, increase revenues [11] and/or compare their business to competitors.

The next section details the research approach. Subsequently, section 3 explicates the data collection methods, while section 4 introduces all indicators which we use to extract insights from the collected data. Section 5 and 6 present and analyze the results themselves. These findings are discussed in section 7. The paper finishes with a conclusion and future research possibilities in section 8.

2 Research Approach

The PaaS providers included in this research are: Azure, Cloud Foundry, dotCloud, Engine Yard, Google App Engine, Heroku, Nodejitsu and OpenShift. This list was created based on two criteria: (1) The PaaS should support easy deployment with one or more development frameworks such as PHP, .Net or Ruby. (2) The solution has to be mature and actively used. PaaS providers that adhere to this criterion have public customers and their share of PaaS modules on the Open Source code hosting service is larger than 1% of all repositories. Four PaaS that meet the first, but not the second criterion are omitted from this research: AppHarbor, AppFog, Amazon Elastic Beanstalk and Nodester.

Relevant comparative measures are necessary to evaluate the health of a PaaS ecosystem. In 2006, a multitude of indicators were collected by Crowston et al. to measure the health of open source ecosystems [10]. We consider four indicators from this research based on three earlier publications [12, 13, 14] to be appropriate in the context of PaaS ecosystem health:

1. Number of active developers
2. Spin offs
3. Interest in the project
4. Download count

Due to the focus on open source development of large projects, the other indicators collected by Crowston et al. are unsuitable in the context of PaaS or require unobtainable data. For example, the perceived ease of use is not applicable to projects that require integration with another piece of software and code quality can not be measured for ten-thousands of projects accross more than a hundred programming languages. Instead, the selected indicators measure interest in the PaaS Providers of their users, the developers. If developers create applications for a PaaS, its ecosystem will flourish. On the other hand, if developers are not interested in developing for that platform, this indicates that the ecosystem is unhealthy. This paper is based on the assumption that developer interest has a direct correlation to the customer size of the PaaS provider and subsequently the long term health of the ecosystem that belongs to it.

Data was collected from GitHub with a Ruby program developed by the authors of this research paper. Once collected, the data was prepared for analysis by validating completeness, correcting errors and redundancy, transforming to a uniform format and storing it in a database. Statistical analysis was subsequently conducted with the R programming language for statistical computing[2]. Alternative open source code hosting services to GitHub with a publicly accessible API such as Bitbucket, Tigris and Launchpad were excluded. These alternatives host orders of magnitude less repositories than GitHub and their data is less clean. A manual search for the largest PaaS, Heroku, returned 30,748 results on GitHub, compared to at most 247 repositories at the alternatives. Moreover, their databases contain many redundant records and significantly less rich metadata.

[2] http://www.r-project.org/

3 Data Gathering

Multiple methods to extract data from GitHub are available, including the official API or screen scraping. For this research, a small Ruby application leveraging the octokit gem[3] was written that conducts repository searches through keywords of PaaS provider names. Because developers use a multitude of different names to refer to the same PaaS, multiple searches with variations of each name were conducted. For instance, while Heroku only requires 'Heroku', Google App Engine has repositories with names as: 'GAE', 'GoogleAppEngine', 'Google App Engine' or just 'App Engine'. Additionally, GitHub hides private repositories from search results, restricting data collection to public repositories. The data collection code can be found on GitHub[4]. All data was collected on January 4, 2013.

3.1 Key Data Elements

The Ruby application collects the following data elements for every repository:

1. Created at date
2. Description
3. Number of followers
4. Fork boolean
5. Number of forks
6. Programming Language
7. Name of repository
8. Owner
9. Private boolean
10. Push date (latest update)
11. Size
12. Type

Unfortunately, users are unable to retrieve the number of downloads of a repository with the API due to technical limitations[5]. On top of that, GitHub does not display any download count data on the website, ruling out the possibility of screen scraping. As a result, it is impossible to assess PaaS ecosystem health with the download count indicator of Crowston et al [10].

3.2 Data preparation

Some repositories contain two or more different keywords referring to a single PaaS in their description or name fields. As a result, multiple returned results of the same repository for different keywords create redundant records in our data set. On the other hand, 574 repositories support multiple PaaS platforms, necessitating some redundant records. For example, if a repository supports both Heroku and Google App Engine a second entry of the record is appropriate. Instead of leveraging the unique GitHub id assigned to every repository, a unique primary key was composed by combining the PaaS platform keyword, username of the repository owner and the repository name itself. GitHub enforces a strict

[3] http://rubygems.org/gems/octokit
[4] http://github.org/gglucass/seco
[5] http://stackoverflow.com/questions/6198194/how-to-see-count-of-project-downloads-on-github

unique naming protocol which ensures that no false duplication positives are raised. Next, we removed all repositories with a size of 0 from the data set. We consider these repositories as false positives because they are empty and thus do not indicate a concretized developer interest.

Finally, a 1% sample of all repositories was used to determine the amount of remaining false positives in the data-set. In this sample, a repository is considered a false positive when the project itself is not directly related to that specific PaaS. Examples are repositories that mention the PaaS, but in a different context, e.g. 'this repository provides an alternative to PaaS X'. The sample was gathered by performing a SQL query on the data-set which randomly selected 1% of all repositories. All repositories from this sample were subsequently manually verified. Although a random sample has its limitations, the sample gives an overall indication for the total data-set. We believe that this sample is representative for the total data-set and therefore should be taken in consideration when reviewing our results.

The sample resulted in 29 (5.8%) false positives, of which the majority are from Google App Engine (13) and Heroku (9). CloudFoundry, dotCloud and Nodejitsu had no false positives. However, due to their smaller sample size a smaller number of false positives is expected. These results do not indicate that repositories of these PaaS providers are free of false positives. Based on the low percentage of false positives found in the sample, we are confident that the collected data-set provides an accurate representation of the contemporary ecosystems of the selected PaaS providers.

4 Indicators

This section explicates three indicators and what types of sub-indicators each is comprised of. Each sub-indicator is accompanied by a short explanation of how it is calculated.

4.1 Active Developers

Because the development of a project first and foremost relies on voluntary contributions of developers, Crowston et al. state that one indicator of success is the absolute number of developers involved in an open source project [10]. By looking at the number of active developers in the past year as well as per week, a more balanced representation is generated.

Active developers in the past year The total number of active developers developing on top of a PaaS provides a direct measure of developer interest in the past year. Crowston et al. propose to measure this by collecting the number of developers who are **formally** associated with a project. To adhere to this requirement, we calculate this measure as the sum of all unique owners of repositories updated in the past year.

Active developers of unique repositories in the past year Not all repositories are created equal. Many are copies of original projects or slight derivates. This sub-indicator only takes into account non-fork repositories, i.e. unique projects that were started from scratch instead of based on another project. Likewise, this sub-indicator is calculated as the sum of all unique owners of unique repositories updated in the past year.

Active developers per segment of time The week to week variety in number of actively contributing developers measures group activity in time between releases or *cycle time* [10]. For each week in the past year, the sum of all unique owners of repositories is calculated. These numbers are subsequently divided by the total number of developers in the past year and in turn presented in a line-graph. With this graph, the relative weekly activity of developers for the PaaS is visualized.

4.2 Spin offs

Crowston et al. mention spin offs as an element of recognition, which in turn is a measure for project success [10]. A spin off is a derivative of a previous projects or a new project entirely. In the context of this research, the two simplest measures of spin offs are the number of forks (derivative of previous project) and projects (repositories). However, on GitHub a fork is also considered a repository. To not let this skew the results, a separate sub-indicator representing the number of repositories which are not forks is included.

Total repositories Repositories are a direct indicator of the number of spin offs for a PaaS. The total number of repositories which contain the PaaS keyword is the most basic method to measure the contribution of spin offs to project success.

Unique repositories Popular repositories on GitHub have many forks, which are often direct copies of the original repository. Multi-platform repositories that are exceptionally popular increase the total number of repositories for all PaaS providers, although one or more of these might not be that popular at all. To discount this phenomenon, this sub-indicator is calculated as the total number of original repositories.

Forks In theory, total repositories includes forks, but in practice this data is incomplete because of two factors. First, GitHub provides private repositories to premium members; effectively shielding us from accessing that data. Second, some repositories are forked but deleted sometime later. Luckily, for each repository GitHub counts the number of forks made regardless of these restrictions. For each PaaS the sum of all forks is taken to include potentially lost data.

4.3 Interest

General interest in a project is different from the first two indicators, which are focused on the quantity of developers and spin-offs. Instead, this indicator focuses on *level of activity*. Crowston et al. recommend to examine development logs for evidence of software being written and released. In absence of this data, we measure interest as the combination of passive interest, interest longevity, number of unique programming languages and multi-platform projects.

Total number of followers Followers are GitHub members who have starred a repository, a mechanism that is used to be kept up-to-date with changes. A follower does not actively contribute to development. Thus a follower is an indicator of passive interest in the project. Calculated as the sum of all follower counts for each repository of a PaaS.

Number of unique programming languages The diversity of programming languages that are used to develop for a PaaS indicate how broad of an interest there is in leveraging the strength of that PaaS. The metric is the total number of unique languages used for each PaaS.

Number of multi-platform repositories the PaaS is a part of Some popular repositories support more than one PaaS. Counting all repository name + owner combinations which also occur for other PaaS providers measures the interest of high profile development projects.

Number of repositories updated at least once Many repositories and forks of repositories are created but never updated. Counting the number of repositories that are updated at least once after creation provides an indication of the interest longevity of developers in the PaaS.

Created at date smaller than push date / created at date > = than push date Many repositories are forked or created and subsequently never updated, effectively rendering these projects as dead. Calculating the proportion of these repositories in relation to repositories that are updated, indicates the longevity of developer interest in developing on top of a PaaS. This ratio is calculated by dividing the number of repositories that were updated after creation by the number of repositories that have the same updated as created date.

5 Results

The data collection program yielded a dataset of 55,927 repositories, 35% (19,859) of these are unique repositories, i.e. repositories that are not forked from another repository on GitHub. In total, 50,057 forks were made, of which 85% are forked from a unique repository. In the last year 24,987 developers have contributed to any of these repositories. The average PaaS provider has 6991

repositories, 2482 unique repositories and 4945 developers. On average, a repository has 0.895 forks and 4.579 followers. Table 1 summarizes these findings. Only taking in account unique repositories, these numbers more than double to 2.16 forks and 10.8 followers. Compare this to the numbers of strictly forked repositories, which on average has 0.1983 forks and 1.154 followers. The difference in popularity is close to a factor of ten for both values, indicating that a spin-off of a project rarely is an interesting new project itself. Instead, most forks are likely duplicates of the original project, which developers further develop to contribute to the expansion of that project.

Number of ...	**Minimum**	**Median**	**Mean**	**Maximum**
Repositories per PaaS	924	2785	6991	29,980
Unique repositories per PaaS	318	720.5	2482	10,400
Developers per PaaS	618	1948	4945	20,320
Forks of repositories	0	0	0.895	1212
Followers per repository	0	1	4.579	7567

Table 1. Descriptive statistics

5.1 Indicator results

The raw results of the indicators presented in section 2 are presented in Table 2 and Figure 1. Aside from the updated vs. non-updated repository, all metrics confirm the expectation that Heroku and dotCloud are the most and least popular PaaS providers, respectively. Furthermore, the total number of forks exceeded the number of repositories minus the number of unique repositories for every single PaaS. This validates our expectation that data is lost due to private and deleted repositories, as well as the necessity of inclusion of this sub-indicator. The next section further analyzes these results.

	a	b	c	d	e	f	g	h	i	j
Azure	2298	681	3607	1251	4187	19,455	29	1327	27	0.65
CloudFoundry	1244	316	2857	684	2762	11,820	18	1410	54	0.55
dotCloud	496	193	924	418	839	6175	12	328	44	0.69
Engine Yard	1124	193	2713	320	2141	10,805	13	1079	44	0.56
Google App Engine	4660	2243	12,319	5709	7538	39,967	32	4594	87	0.70
Heroku	15,384	5633	29,982	10,402	29,137	148,766	38	10619	228	0.62
Nodejitsu	957	214	1446	318	1596	12,766	7	601	38	0.5
OpenShift	1327	498	2079	757	1857	6361	18	742	51	0.72

Legenda

a Active developers in the past year	f Followers
b Active developers of unique repos	g Unique programming languages
c Number of repositories	h Repositories updated at least once
d Number of unique repositories	i Multi-platform repositories
e Number of forks	j Updated vs. Non-updated ratio

Table 2. Indicator results

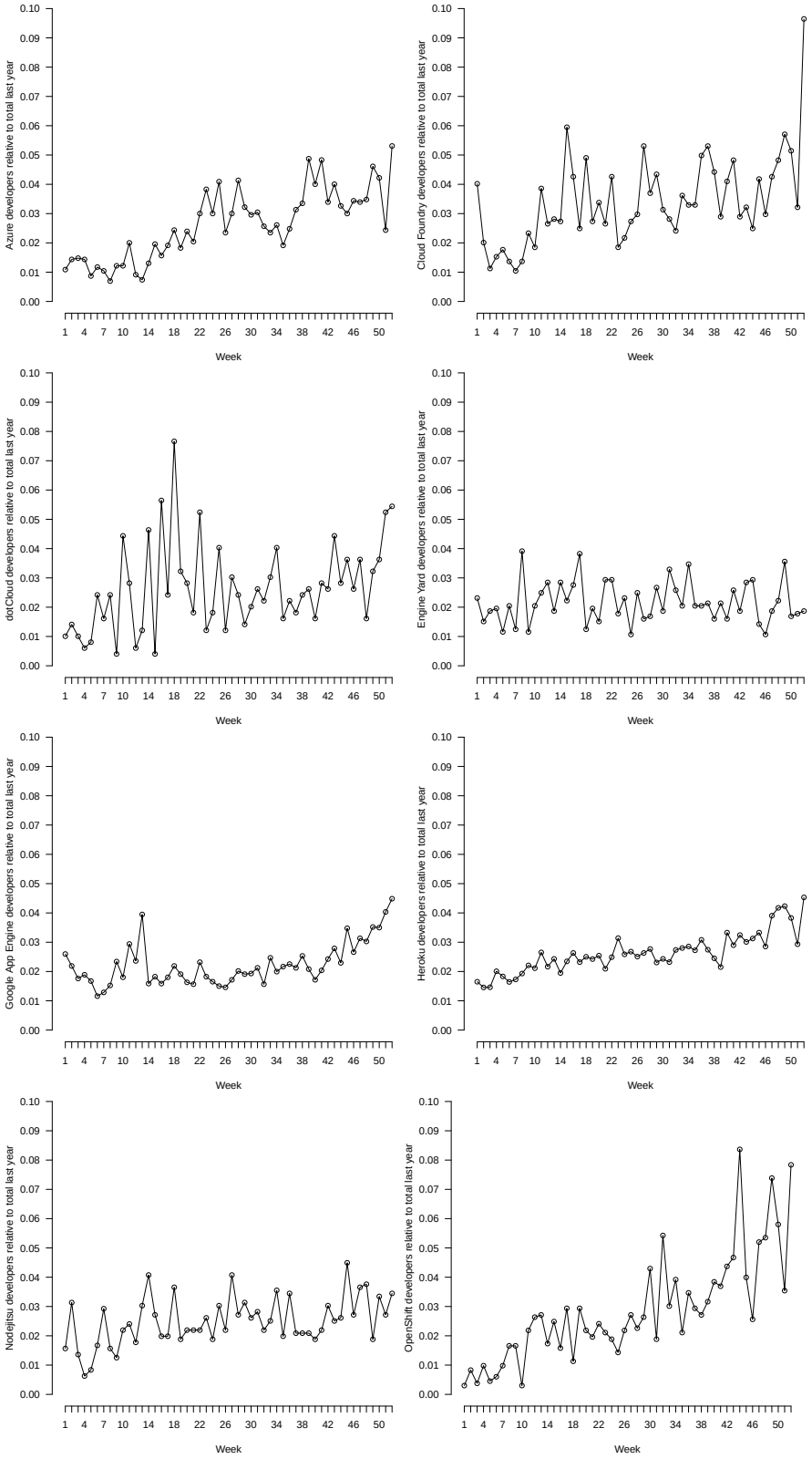

Fig. 1. Graphs displaying relative number of active developers per week

6 Analysis

In this section, the results are further analyzed by looking at the distribution of all indicator metrics and of actively contributing developers in the past year. Table 2 demonstrates Heroku its dominance in the PaaS industry, with the best results for 9 out of 10 sub-indicators. Google App Engine and Azure are in similar comfortable positions, with second place for all sub-indicators and third place for 8 out of 10 sub-indicators, respectively. Although it should be noted that Heroku accounts for more than half of all results for 7 out of 10 sub-indicators. Furthermore, development for Azure includes the least amount of unique programming languages. dotCloud and Nodejitsu are at the bottom of the results, together they have collected the bottom scores for all but one indicator. Moreover, the scores of Heroku are more than 20 times as high for each indicator. The results of the remaining three PaaS providers; CloudFoundry, Engineyard and OpenShift, contain some intriguing patterns relative to one another. CloudFoundry strictly has first and close second positions. EngineYard is a close second for half of the indicators and the overwhelming third for the other half. OpenShift has a mix of all three places. However, its low scores for f (number of followers) and h (repositories updated at least once) are higher than both CloudFoundry and Engineyard relative to the number of repositories these metrics are derived from. This is further confirmed by OpenShift having the highest score for indicator j, updated vs. non-updated ratio.

Based on metric a, Figure 1 was created. For each PaaS, it displays the relative number of active developers in a weekly period compared to the total number of active developers in the past year. The general trend of the industry is positive. Developers of most PaaS providers have created or updated more repositories in recent weeks than in weeks at the start of the year. OpenShift in particular has a steep graph upwards throughout the year, with just one third (34.29%) of developers in the first half of 2012. However, Google App Engine and EngineYard display a negative pattern. More than a quarter (26.37%) of Google App Engine developers have not updated their repository since the first quarter of 2012. EngineYard is in a similar position, with the majority of their developers (52.04%) in the first six months of 2012.

7 Discussion of PaaS Ecosystem Health

The analysis of all indicators introduced Heroku as the dominant, leading PaaS. The scores of the runner-up PaaS, Google App Engine, further confirms this dominance. Only two sub-indicators exceed half the scores of Heroku.

The passive interest shown by the number of followers is four times greater for Heroku than the passive interest of Google App Engine and an order of magnitude greater than all the others. This indicates that interest in Heroku is not limited to active developers, but includes a large number of passive GitHub users that are interested in the progress of these projects. Based on these observations, we expect Heroku to maintain its dominance in the coming years.

However, absolute size is not an indicator of a great ecosystem per se. OpenShift its good performance on indicators a, i and j illustrates this. OpenShift has results close to Azure for many indicators, even though Azure boasts a larger customer-base and a three times as large community. Moreover, the active developers for OpenShift in the past shows the most promising slope of all providers, indicating a good position to grow in the PaaS industry in the future. An explanation for this growth in success is that Openshift is a subsidiary of Red Hat, which has a large existing customer base and a devoted community.

The general positive distribution of active developers per week implies positive future growth expectations for the PaaS industry as a whole. However, the shift in the graphs of Google App Engine and EngineYard indicates that developers may already be abandoning these PaaS providers, resulting in a future with a compounding loss of developers and ultimately end of business.

Do these results commend or discourage doing business with certain PaaS providers? Assumptions need to keep the origin of the data in mind. Data collected from GitHub might not be representative for commercial private cloud providers, e.g. Azure, as those developers are more inclined to use proprietary solutions or private repositories to collaborate. Furthermore, should PaaS providers with negative indicators be neglected? Although e.g. EngineYard scored low, they reported revenues of 28$ million in 2011. This shows an obvious market interest and a potential for healthy future growth. However, business aspects as revenues or customer size are not publicly available or verifiable for the majority of PaaS providers. As a consequence, we are unable to evaluate the *long-term financial well-being of the business ecosystem* and are restricted to the *long-term strength* aspect of the ecosystem health definition on page 2.

8 Conclusion and Future work

This exploratory research provides businesses with a method to evaluate PaaS providers. The current ecosystem health is skewed to two major players, with Heroku far ahead. If OpenShift can maintain its growing trend, this will change in the future. The PaaS industry is still young, new businesses will enter the market and others will exit. Additionally, PaaS providers can evaluate the current state of their ecosystem and adjust their corporate strategy accordingly.

Future research could validate our method by confirming whether weak PaaS providers went out of business or by applying this model to other players. To assist academics and businesses in this process, we provide the data extraction methods and analysis code on GitHub. Furthermore, retrospective studies with a broad scope can document the developments within the PaaS industry as a whole.

References

1. Gartner Says Worldwide IT Spending On Pace to Surpass 3.6 Trillion in 2012, http://www.gartner.com/it/page.jsp?id=2074815

2. Mell, P., Grance, T.: The NIST Definition of Cloud Computing (draft). NIST Special Publication 800-145 (2011)
3. Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., Ghalsasi, A.: Cloud Computing, The Business Perspective. Decision Support Systems 51, 176–189 (2001)
4. Hartigh, E., Tol, M., Visscher, W.: The Health Measurement of a Business Ecosystem. In: Proceedings of the European Network on Chaos and Complexity Research and Management Practice Meeting (2006)
5. Iansiti, M., Levien, R.: Strategy as ecology. Harvard business review 82-3, 68–81 (2004)
6. Kittlaus, H.B., Clough, P.N.: Software Product Management and Pricing: Key Success Factors for Software Organizations. Springer, New York (2009)
7. Bosch, J.: From Software Product Lines to Software Ecosystems. In: Proceedings of the 13th International Software Product Line Conference, pp. 111–119. ACM, New York (2009)
8. Jansen, S., Finkelstein, A., Brinkkemper, S.: A Sense of Community: A Research Agenda for Software Ecosystems. In: 31st International Conference on Software Engineering, New and Emerging Research Track, pp. 187–190. IEEE Press, New York (2009)
9. Spriensma, G.J.: 2012 Year in Review, http://www.distimo.com/publications/archive/Distimo%20Publication%20-%20Full%20Year%202012.pdf
10. Crowston, K., Howison, J., Annabi, H.: Information systems success in free and open source software development: Theory and measures. Software Process: Improvement and Practice 11-2, 123–148 (2006)
11. Baars, A., Jansen, S.: A Framework for Software Ecosystem Governance. In: Proceedings of the Third International Conference on Software Business, pp 168–180. Springer-Verlag Berlin, Heidelberg (2012)
12. Stewart, K.J., Gosain, S.: The impact of ideology on effectiveness in open source software development teams. MIS Quarterly 30-2, 291–314 (2006)
13. Krishnamurthy, S., Cave or community? An empirical examination of 100 mature open source projects. First Monday 7-6, http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/view/1477/1392 (2002)
14. Crowston, K., Annabi, H., Howison, J., Masango, C.: Effective work practices for software engineering: free/libre open source software development. In: Proceedings of the 2004 ACM workshop on Interdisciplinary software engineering research, pp. 18–26. ACM Press, New York (2004)