Ceiling and Threshold of PaaS Tools: The Role of Learnability in Tool Adoption

Rui Alves, Nuno Jardim Nunes

Ceiling and Threshold of PaaS Tools: The Role of Learnability in Tool Adoption

Rui Alves[1] and Nuno Jardim Nunes[1,2]

[1] Madeira Interactive Technologies Institute,
Polo Científico e Tecnológico da Madeira, 2nd floor, Caminho da Penteada
9020-105 Funchal, Madeira, Portugal
[2] Universidade da Madeira, Campus Universitário da Penteada,
9020-105 Funchal, Madeira, Portugal
rui.alves@m-iti.org, njn@uma.pt

Abstract. Cloud services are changing the software development context and are expected to increase dramatically in the forthcoming years. Within the cloud context, platform-as-a-service tools emerge as an important segment with an expected yearly growth between 25 to 50% in the next decade. These tools enable businesses to design and deploy new applications easily, thereby reducing operational expenses and time to market. This is increasingly important due to the lack of professional developers and it also raises a long standing issue in computer-aided software engineering: the need for easy to learn (low-threshold), functional (high-ceiling) tools enabling non-experts to create and adapt new cloud services. Despite their importance and impact, no research to date addressed the measurement of tools' ceiling and threshold. In this paper we describe a first attempt to advance the state of the art in this area through an in-depth usability study of platform-as-a-service tools in terms of their threshold (learnability) and ceiling (functionality). The measured learnability issues evidenced a strong positive correlation with usability defects and a weaker correlation with performance. Remarkably, the fastest and easiest to use and learn tool falls into the low-threshold/low-ceiling pattern.

Keywords: PaaS, threshold, ceiling, learnability.

1. Introduction

Within the next 30 years the demand for software applications will grow exponentially[1]. Yet, our capability to increase the number of professional software developers and/or their productivity will at best stabilize or grow linearly. Bezivin call it the impossible equation (see **Fig. 1**) stating that "How, with the same amount of people (code producer and managers, about 1% of the total population), to produce an order of magnitude more of software applications than now?"[2].

[1] http://semat.org
[2] https://modelseverywhere.wordpress.com/2013/02/15/one-percent-software-professionals-in-advanced-countries/

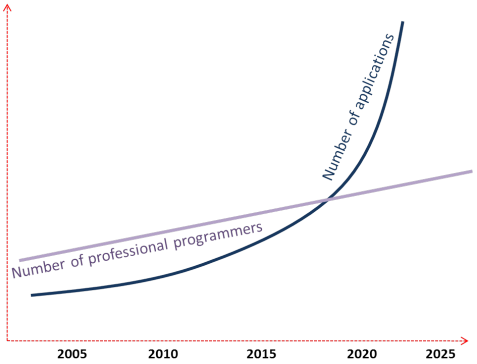

Fig. 1. The impossible equation (from http://modelseverywhere.wordpress.com)

This equation seems impossible to solve because, on one hand it is not practical to educate a larger percentage of the world population to develop professional software. On the other hand, lessons from the past show that whatever progress we may achieve in software tools, the productivity of software professionals will not improve by a major factor. As stressed by Bezivin, the key to solving these problems probably lies in the collaboration between professional developers and end-users.

This is a long-standing challenge in engineering interactive systems. In order to enable end-users to contribute to software development, we need to provide new languages and development environments that effectively suite their needs and are easy to learn and operate. These tools are known after the seminal paper about "the past, present and future of user interface software tools" [1] as low-threshold and high ceiling tools. In fact, a major trait in tools' acceptance is the role of threshold and ceiling (the threshold is how difficult it is to learn how to use a tool and the ceiling is how much can be done using the tool). Successful tools in the past 25 years are claimed to either be low-threshold/low-ceiling, or high-threshold/high-ceiling [1]. Despite all efforts we struggle to find tools that enable low-threshold and high-ceiling.

Currently, one major trend lies in what is usually known as cloud infrastructure, involving platforms and solution stacks as a service model of computing. These involve software-as-a-service (SaaS), infrastructure-as-a-service (IaaS) and platform-as-a-service (PaaS). The focus of this paper is the PaaS model in which the end-user controls software deployment and configuration settings, while the vendor provides everything required to host the application [2]. PaaS are characterized by providing low initial cost, incremental growing, self-service, resource sharing, automated deployment, management services, lifecycle management and reuse [3].

In this paper we attempt to advance the state of the art by increasing our understanding about what these concepts mean and how they can be effectively measured. We do this through: (1) a methodological approach to assess the usability and functionality of PaaS tools, (2) the results from a comparative study of three PaaS tools with 22 subjects and (3) statistical analysis looking for correlations between dimensions such as learnability issues, usability defects and performance. Lastly, in the discussion section, we contrast our findings with the prevailing literature and then present our conclusions.

1.1 State of the Art

Both software engineers and designers often complain that their tools are unsupportive and unusable [1]. While many studies have analyzed and tried to better support general software development practices [4, 5], qualitative studies on user interface related practices, in software development, are relatively rare. Furthermore, the work of Seffah and Kline, showed a gap between how tools represent and manipulate programs and the actual experience of the developer [5]. However, ceiling and threshold related issues were not specifically addressed.

Among the distinct sets of existing development tools, there is a group of literature on computer-aided software engineering (CASE) tools [1, 4, 6, 7], which we think worth assessing if the findings related to these tools prevail in the scope of this work. This prior research suggests that: (1) few organizations use CASE tools; (2) many organizations abandon the use of the tools; and (3) countless developers, working for organizations that own CASE tools, do not actually use them. Moreover, Jarzabek and Huang argued that CASE tools should be more user-oriented and use rigorous modeling, in order to better blend into the software development practice [7]. Yet, figures on the adoption of these tools seem to contradict the goals driving their development. Though, a new breed of tools has emerged, in particular Platform-as-a-Service (PaaS), which claim to bridge previous gaps and promise easier and faster development, even for non-experts. Intrigued by these apparently conflicting forces, we found room to contribute by researching the role of learnability in tools adoption.

2. Research Question

Since practitioners often complain about the suitability of tools for their work, we realized that threshold and ceiling are claimed as attributes that play a major role in tools adoption. Moreover, successful tools are said to either be low-threshold/low-ceiling or high-threshold/high-ceiling. The issue is that we lack knowledge on how to measure these concepts. Thus, we hypothesized that the threshold is associated with tool's adoption. So, if a tool is hard to learn then it's adoption is more likely to be low, since people will seek out easier solutions. The work presented here is part of a major project, which in the future will help us shed some light on tools adoption and we focus on how to define and measure the core concepts of the framework for tools' threshold and ceiling measurement.

In this paper we measure the initial learnability of tools and assess the relationships between the learnability issues and the usability defects, as well as eventual impacts on subjects' performance. Additionally, recognizing that the terminology is not consistent and before jumping into the details, we would like to clarify the core concepts used in this paper. For instance, learnability is the attribute of a system, which allows users to rapidly begin working with it [8]. In this paper a learnability issue is any problem preventing users from quickly understand how to use the system to do a task. A defect is a breakdown in usability, an obstacle to perform a desired task or a problem in the smooth functioning of the system. A usability defect is generally a mismatch between what the user wants, needs or expects and what the system provides. Finally, performance is the time a participant took to do a task.

3. Study

The research study described here was conducted over a period of six months (September 2013–February 2014) and expands the sample of the study described in a previous publication [9]. Participants were asked to play the role of a small shop owner willing to replace his spreadsheet to manage products with a web application. For that purpose, this owner is giving a try to PaaS tools and will try three tools (Knack [10], Mendix Business Modeler 4.7.0 [11] and OutSystems Studio 8.0 [12]).

PaaS is a growing and highly competitive sector with more than 70 vendors, making it hard to select which PaaS tools to test. We surveyed the web for comparisons and picked two tools that are popular and comparable to the tool proposed by the vendor that asked us to evaluate their tool. Additionally, prior to deployment, we performed three pilots to improve the test instructions, assess the duration, completeness and suitability of proposed tasks. Furthermore, we faced two major challenges to run this study: (1) how to record subjects' actions so that they did not felt observed and (2) the test duration. To address both, upon subjects' informed consent, we recorded video and audio and captured screen actions using 15" laptop built-in features. Having no extra hardware helped subjects perceive it naturally. Additionally, the amount of tools tested was constrained by the fact that subjects will do the same set of tasks on each tool. As such, it proved unfeasible to add more tools (due to subjects' engagement and motivation) since the test duration roughly spanned from 2,5h to 3h per subject. Because of this plus the fact that we required participants to be completely new (initial learnability) to all tested tools, volunteers were scarce.

3.1 Sample

We used a sample of 22 subjects from both genders, divided into two even groups: (G1) comprised people who run small businesses and (G2) IT experts. The rationale for choosing these groups was to contrast their results and assess if PaaS is ready for non-IT experts, as claimed by some vendors. On average, subjects were 30 years old (minimum 22 and maximum 43). Circa 30% had worked for less than one year and 40% had between four and nine years of experience, while 20% had already worked for more than ten years. The vast majority (60%) holds a bachelor degree and 35% hold a master's degree. Additionally, none of the G1 subjects had any experience building web applications or had ever used the tools of the trade, whereas all, but one, G2 subjects were experienced in building web apps. Among the G2 subjects, IDEs were popular tools used to create web apps (50% uses Netbeans, 40% Dreamweaver) but no one had used or even knew of PaaS tools.

3.2 Methodology

As previously stated, this paper expands (from ten to 22 subjects) and builds on top of a previous publication [9]. A detailed explanation of the methodology used was already provided in the methodology section of [9], please refer to this reference for further detail on the methodology used. Here we focus on the results and provide a statistical analysis of the results to complement the claims and findings stated in [9].

4. Results

Three dependent variables were measured in this study, specifically (i) learnability issues, (ii) usability defects, and (iii) performance. Additionally, based on subjects' level of success to complete each task, learnability scores were computed also.

4.1 Learnability Issues

Grossman's et al classified learnability issues into five categories: (i) awareness, (ii) locating, (iii) transition, (iv) task-flow and (v) understanding. For detailed explanation on each category please refer to [16]. **Table 1** summarizes each category measured weight. The most frequent categories are highlighted in bold and account for 50%+ of issues per group/tool. The transition category is the less frequent, except for G1 in Knack. We used the transition category to compute a ratio between all categories and transition, to learn what categories hinders users the most. **Table 2** stresses the proportionality among categories and we found evidence that understanding and task-flow issues hinder users the most, being three times more prevalent than transition issues. In brief, these figures provide empirical evidence that Knack was easier to learn than Mendix and Outsystems. Moreover, all tested tools seem to have a lot of technicalities since all fail to make it easier for non-technical users to quickly learn how to use the tool, risking their putative success in this niche.

Table 1. Learnability issues weight per tool.

Category	Knack				Mendix				Outsystems			
	G1		G2		G1		G2		G1		G2	
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%
Awareness	5	8	11	26	16	17	9	15	16	16	8	17
Locating	11	18	9	21	26	27	20	33	14	14	7	15
Transition	6	10	5	12	9	9	6	10	3	3	6	13
Task-flow	10	17	5	12	33	35	12	20	25	26	13	27
Understanding	28	47	13	30	11	12	13	22	40	41	14	29
Total	60		43		95		60		98		48	

Table 2. Relative impact of each learnability issue category.

Category	G1	G2	Knack	Mendix	Outsystems	Total
Awareness	2	2	1	2	3	2
Locating	3	2	2	3	2	2
Task-flow	4	2	1	3	4	3
Understanding	4	2	4	2	6	3
Transition	1	1	1	1	1	1

4.2 Usability Defects

The usability defects were categorized according to their relation to the interface. The icons/graphics category are related to a graphical design issue (similar icons). The bars/windows category is directly related to using bars or high level commands. When the defect was related to buttons or input fields then it was categorized as menus/commands, whereas defects classified as interaction are, for instance, double clicking creates something without questioning the user). Finally, the text/feedback category are issues on the textual terminology and text feedback on the interface.

Table 3 summarizes the weight of each usability defect category, per tool and group. In the same table we have highlighted, in bold, the top two categories per group and tool, which account for more than 50% of all defects. Similarly to learnability issues, we have also computed a ratio between all categories and bars/windows, to grasp an idea of what categories have a stronger impact on subjects **Table 4**. In brief, the trend found on learnability issues is also true here and provides empirical evidence that Knack was easier to use than Mendix and Outsystems. Moreover, all tools seem to suffer from usability defects and fail to make it fully usable for non-technical users, which PaaS vendors claim to be their target.

Table 3. Usability defects weight per tool.

Category	Knack				Mendix				Outsystems			
	G1		G2		G1		G2		G1		G2	
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%
Bars/Windows	6	8	3	5	13	13	8	12	22	16	9	12
Icons	3	4	5	8	21	21	9	13	27	20	10	14
Interaction	2	3	22	37	12	12	5	7	27	20	18	24
Menu/Commands	2	2	14	24	32	32	27	40	20	14	10	14
Text/Feedback	2	3	15	25	21	21	18	27	25	18	20	27
Total	7		59		99		67		13		74	

Table 4. Relative impact of each usability defect category.

Category	G1	G2	Knack	Mendix	Outsystems	Total
Icons	1	1	1	1	1	1
Interaction	2	2	5	1	1	2
Menus/Command	2	3	4	3	1	2
Text/Feedback	2	3	4	2	1	2
Bars/Windows	1	1	1	1	1	1

4.3 Performance

Table 5 shows the performance, in hours, per group/tool. Curiously, the tasks that took longer in each tool, took longer for both groups. Additionally, we found that the amount of time spent in each task ranged a lot per tool, although the tasks were the same for all tools tested. For instance, while in Knack the T4 and T5 were the tasks that took more time to do, in Outsystems these two tasks were the quickest ones. These differences could evidence the intrinsic difficulty to accomplish a task in one tool, which in turn could be associated with high, or low frequency of learnability issues, thus impacting tools' threshold. In brief, our results provide evidence that the longer it takes to finish the task, the harder it is to learn how to do it. Performance figures follow the trend found in 4.1 and 4.2, providing empirical evidence that Knack is faster to use than Mendix and Outsystems.

Table 5. Performance (aggregated time spent) in hours.

Task	Knack				Mendix				Outsystems			
	G1		G2		G1		G2		G1		G2	
	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%	Σ	%
T1	0.81	17	0.55	14	1.08	16	0.55	12	0.67	10	0.66	13
T2	1.08	23	0.85	22	2.56	39	2.05	43	2.49	39	2.01	38
T3	0.44	9	0.39	10	0.78	12	0.43	9	2.18	34	1.67	32
T4	1.21	26	1.00	26	0.53	8	0.44	9	0.69	11	0.47	9
T5	1.16	25	1.08	28	1.68	25	1.26	27	0.40	6	0.46	9
Total	4.70		3.86		6.64		4.72		6.42		5.27	

4.4 Statistical Analysis

The results highlighted a difference between tools where Knack (classified as low-threshold/low-ceiling) was the fastest, easiest to learn and use tool being tested, whereas the other two tools produced similar performance results. In this section we investigated if there is statistical evidence, which could ground the previous claims. The tools used in this study were encoded in one independent variable and we measured the differences for each dependent variable (learnability issues, usability defects and performance). Additionally, we also sought for possible correlations.

Differences

We have used a repeated measure analysis of variance (ANOVA) to test if there are differences between conditions (Knack, Mendix and Outsystems), where the same participants are being measured on the same variable (learnability issues, usability defects or performance). The groups are related because they contain the same cases (the subjects) and each group is a repeated measurement on a dependent variable. This led us to the null hypothesis: H0: all related group means are equal. The alternative hypothesis is Ha: at least one related group mean is different, which would provide evidence on statistical differences between tools, if verified. We verified all the assumptions and found no outliers, in learnability issues, by inspecting the boxplots for values greater than 1.5 box-lengths from the edge (dark grey boxplots on **Fig. 2**). Data is normally distributed in all three conditions, as assessed by Shapiro-Wilk's test ($p > 0.05$) and Mauchly's Test of Sphericity indicated that the sphericity assumption has not been violated, $\chi 2$ (2) = 1.048, $p = 0.592$. Having satisfied all assumptions, we computed the repeated measures ANOVA, which yielded statistically significant changes in the amount of learnability issues found per tool, $F(2, 42) = 8.868$, $p < 0.001$, partial $\eta 2 = 0.297$. Although this result rejects the null hypothesis and retains the alternative hypothesis, we cannot directly determine where exactly the differences between groups lie. To clarify this uncertainty, a post hoc analysis with a Bonferroni adjustment revealed that the amount of learnability issues significantly decreased from Outsystems to Knack (1.68 (95% CI, 0.25 to 3.12) issues per subject, $p < 0.05$) and from Mendix to Knack (2.64 (95% CI, 0.97 to 3.76) issues, $p < 0.01$), but not from Mendix to Outsystems (0.68 (95% CI, -0.98 to 2.34) issues, $p = 0.895$). As such, these results provide statistical evidence that Knack offers less learnability issues for first time users than the other tested tools.

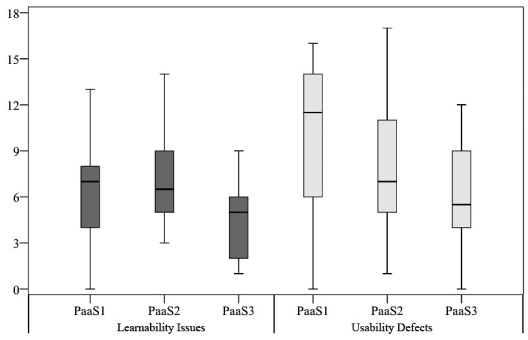

Fig. 2. Learnability issues and usability defects

The same procedure was applied to usability defects and we found no outliers (light grey boxplots on **Fig. 2**). Yet, we realized that Knack data was not normally distributed ($p = 0.041$), although it is normally distributed for Mendix and Outsystems, $p > 0.05$. Nevertheless, we run the test because the repeated measures ANOVA is fairly robust to deviations from normality. Mauchly's Test of Sphericity indicated that the assumption of sphericity was met ($\chi2\ (2) = 2.46$, $p = 0.292$) and the repeated measures ANOVA test elicited a statistically significant change in the amount of usability defects per tool, $F(2, 42) = 12.519$, $p < 0.0005$, partial$\eta2 = 0.373$. A post hoc analysis with Bonferroni's adjustment revealed that the amount of usability defects significantly decreased from Mendix to Knack (2.091 (95% CI, 0.01 to 4.2) defects per subject, $p < 0.05$) and from Outsystems to Knack (3.455 (95% CI, 1.7 to 5.2) defects per subject, $p < 0.0005$), but not from Mendix to Outsystems (1.364 (95% CI, 0.2 to 2.93) defects per subject, $p = 0.102$). Given these results, there is statistical evidence that Knack has less usability defects than the other tested tools.

In what concerns performance, the first assumption was violated since there are outliers in Knack and Mendix datasets. Moreover, although the Shapiro-Wilk's test yielded $p > 0.05$ for Knack and Outsystems, the normality test returned a negative result for Mendix ($p = 0.038$). Thus, contrarily to what happened with learnability issues and usability defects, we cannot make any statement about statistical evidence on performance differences between the three tools by using this test.

To sum up, we found statistical evidence that (1) Knack was easier to learn (for first time users) than Mendix and Outsystems, (2) Knack was easier to use than Mendix and Outsystems. These findings are in line with the results section. However, we found no statistical evidence that subjects were faster in Knack than in Mendix or Outsystems, although the figures in the results section point towards that direction.

Correlations

There is no statistical test to determine the association between three continuous variables, which is our case and Pearson's correlation was not suitable since there was no linear relationship between variables, as assess by inspecting a scatterplot. As such, we opted for Spearman's correlation. Thus, the null hypothesis is H0: there is no association between the variables and the alternative hypothesis is Ha: there is an association between the variables. Two of the three assumptions of the Spearman's correlation relate to the study design, which we met both (to have two continuous or ordinal variables and the two variables represent paired observations). The third assumption requires to determine whether there is a monotonic relationship between two variables. A monotonic relationship does one of the following: (i) as the value of one variable increases, so does the value of the other variable or (ii) as the value of one variable increases, the other variable value decreases. The Spearman's correlation test unveiled that there was a strong positive correlation between the amount of identified learnability issues and usability defects, $rs(20) = 0.724$, $p < 0.01$. Preliminary analysis showed the relationship to be monotonic, as assessed by visual inspection of a scatterplot. In the other two cases, although the monotonic relationship was not fully clear, there was a remarkably positive correlation between the amount of learnability issues and subjects' performance ($rs(20) = 0.466$, $p < 0.01$) and a moderately positive correlation between the amount of usability defects found and subjects' performance, $rs(20) = 0.352$, $p < 0.01$.

5. Threshold and Ceiling

As successful tools are said to either be low-threshold/low-ceiling, or high-threshold/high-ceiling, we plotted the tested tools within this framework [1]. We achieved the threshold by measuring the learnability of tools. As aforementioned, Knack is easy to learn whereas Mendix and Outsystems scored as regular. Yet, the other attribute i.e. ceiling, was missing. Since ceiling shows how much can be done using the system, there are two major approaches one can take to assess it: (1) compare tools usage over time or (2) compare the features each tool offers [1]. The first option, which could yield results based on empirical data and could be grounded on actual tool usage, proved impractical due to time constraints and lack of actual users to whom we could refer. Thus, we selected the second option and measured each tool ceiling by comparing the features offered by the tool. To do this we identified another tool (www.force.com), which is supposedly one of the most complete in the PaaS context, and compared it to the tested tools feature set. This approach yielded that from a total of 31 features, Knack matches with seven, Mendix matches 29 features and Outsystems 31 features. Therefore, the computed ceiling was 23% for Knack, 94% for Mendix and 100% for Outsystems.

Additionally, we ranked it according to this ceiling scale: (i) very low [0 to 19%], (ii) low [20 to 39%], (iii) regular [40 to 59%], (iv) high [60 to 79%] and (v) very high [80 to 100%]. This means that according to our measure Knack is low ceiling and Mendix and Outsystems are both very high ceiling. **Table 6** summarizes the threshold and ceiling classification based on aggregated data (for initial learnability). Threshold is a measure for how difficult it is to learn a tool and is obtained by inverting the learnability scores. Upon surveying the literature, we found several learnability metrics from which we selected two: (M1) the percentage of users who complete a task without any help and (M2) the percentage of users who complete a task optimally [16]. In M1 users could not ask task-flow related questions. In M2 users must complete the task without help [17]. The aggregated results based on all 330 tasks (22 subjects did five tasks each in three distinct tools) performed, provide empirical evidence that Knack is the easiest tool tested, which reached the easy to learn score with 63%, followed by Outsystems with a score of regular (53%) and Mendix ranked last with 46%. This means that if a tool is easy to learn then the threshold is low or if a tool is very difficult to learn then the threshold is very high. Inversely, the ceiling is determined directly (100% matching of features is very high ceiling). As such, only one tool (Knack) matches the low-threshold/low-ceiling or high-threshold/high-ceiling pattern identified in [1]. The other tools do not match this pattern. The difference in Outsystems was only 7%, which made us wonder if a bigger sample would make a difference or if it will make the gap between tools even bigger.

Table 6. Overall threshold and ceiling classification.

Threshold	**Ceiling**				
	Very	Low	Regular	High	Very High
Very High					
High					
Regular					**Mendix/Outsystems**
Low		**Knack**			
Very Low					

6. Discussion

The work presented here is part of a major project addressing other tools and development environments, here we focused only on the building blocks of the framework for tools' threshold and ceiling measurement, because they were reported as key attributes in tools adoption [1]. As such, we measured the learnability issues and usability defects found per subject and tool, as well as subject performance.

With regard to the learnability issues, the Grossman et al classification [16] was helpful for clustering issues. The raw data and the statistical analysis provided evidence that Knack was easier to learn than Mendix and Outsystems. This is arguably of little interest for generalization purposes. Yet, the fact that, in general, task-flow and understanding related issues are three times more prevalent than transition issues, presents a clear hint for aspects that vendors need to observe in order to increase their tools learnability. This previous claim is moderately in line with the findings from [9] where understanding and task-flow related issues hinders users the most, although with distinct proportionality. One fact that could justify this difference is the sample size which comprised ten subjects in [9] and 22 here.

Regarding the usability defects, which were reported as a major issue preventing adoption [18], we found statistical evidence that Knack was easier to use than Mendix and Outsystems. Additionally, the differences between categories of defects was not as evident as the learnability issues, yet, defects related to interaction, menus/commands and text/feedback happen twice as often as defects related to bars and windows. Again, tools creators should take this evidence into account so that they can diminish the existing usability barriers and make these tools fully usable for non-technical users, which some PaaS vendors target for. Moreover, the performance data shows that Knack was quicker to use than Mendix and Outsystems, although no statistical evidence was found (not all needed assumptions to run the repeated measures ANOVA test were met).

Remarkably, the correlation tests revealed a strong positive correlation between the number of learnability issues and the usability defects, which stresses the visceral relationship between these two attributes. Noteworthy is also the positive correlation between the number of learnability issues and performance issues (this result hints that the more learnability issues, the longer it takes to do the task) yet, correlation is not an implication. Lastly, a moderately positive correlation between usability defects and performance should also be highlighted.

Additionally, regarding the ceiling/threshold, our findings are in line with Myers et al, who claim that successful tools are either low-threshold/low-ceiling or high-threshold/high-ceiling [1]. The fact is that the fastest, easiest to learn and use tool was the only low-threshold/low-ceiling tool tested (Knack). Yet, the success and adoption of tools is far more complex than simply measuring a tool's threshold and ceiling (social and motivational factors, for instance, are claimed to play a major role). As such, since our sample comprises only three tools we cannot make any claim based on this finding. Having said that, we came to realize that both academia and industry are still failing, to build tools that allow us to create sophisticated systems easily (low-threshold and high ceiling) [19], and that this is a demanding challenge which requires further research to address it.

Jeng claims that learnability is inferred from the amount of time required to achieve user performance standards [20]. Yet, in our study, we had no access to real users' performance figures. To do that we would need to perform an extended test with more tools and users. Nevertheless, our goal in this phase was to measure initial learnability, and was not intended to be a longitudinal study.

A note of caution when interpreting these results should be observed due to the intrinsic subjectivity of these evaluations. The fact that all measures were obtained from analyzing video and audio recordings is prone to subjective interpretation. For instance, when identifying learnability issues, one researcher can consider a fact as an issue whereas another person could skip it. However, if there is no perfect way to do this, it does not mean that we should give up. Instead, to know and understand the underlying facts that impact tool adoption is an objective worth pursuing, and one which could benefit the entire community. As such, we suggest that a pool of analysts should review the recordings and doubtful cases should be pair reviewed to reduce bias. Another hypothetical weakness of this work is the sample size, which is arguably small. However, we cannot ignore how complex this kind of study is. Knowing that (1) subjects should have no experience at all in all three tools, and that (2) each subject spent roughly 2,5 to 3 hours using these tools and got no incentives to do so, we can easily understand why did it took us half a year to reach 22 subjects, which heavily limited our efforts to have a bigger sample, with a larger set of tools.

7. Conclusion

The cloud infrastructure is pushing the requirements of development tools in the face of the increasing needs of software and the growing importance of cloud infrastructure. PaaS is becoming an important because it facilitates the deployment of applications or services reducing cost and complexity. PaaS are required to handle tasks from editing code to debugging, deployment, runtime, and management. The prevailing literature on development tools hints at several factors contributing to PaaS adoption: (1) software engineers and designers often complain that their tools are unsupportive and unusable [1], (2) there is a demand for tools more user-oriented tools, which better blend into the software development practice [7] and (3) a gap between how tools manipulate programs and the developers' actual experiences [1].

Since tools ceiling and threshold are among the key factors impacting tools adoption [2] and, due to the lack of existing knowledge on how to measure these attributes, we researched how to measure the threshold by assessing tools initial learnability. For that purpose, we have used a hybrid protocol by making use of think-aloud and question-asking. In brief, all tested tools seem to fail to make it easy to learn, use and perform for non-technical users, which some PaaS vendors claim to be their major target market. Additionally, although threshold and ceiling measurement proved to be challenging and the findings must be thoughtfully analyzed (for instance, high learnability may be necessary for a low threshold, but high learnability may not guarantee a low threshold), we think this is an effort worth pursuing, in light of contributing to our understanding on how to design the tools of the future.

The authors would like to thank Claudio Teixeira, Amanda Marinho and Monica Nascimento for their help to conduct the study described in this paper.

References

1. Myers, B., Hudson, S.E., Pausch, R.: Past, present, and future of user interface software tools. ACM Trans Comput-Hum Interact. 7, 3–28 (2000).
2. Rimal, B.P., Choi, E., Lumb, I.: A taxonomy and survey of cloud computing systems. In: INC, IMS and IDC, 2009. NCM'09. Fifth International Joint Conference on. pp. 44–51. Ieee (2009).
3. Lawton, G.: Developing software online with platform-as-a-service technology. Computer. 41, 13–15 (2008).
4. Iivari, J.: Why are CASE tools not used? Commun ACM. 39, 94–103 (1996).
5. Seffah, A., Metzker, E.: The obstacles and myths of usability and software engineering. Commun. ACM. 47, 71–76 (2004).
6. Campos, P., Nunes, N.J.: Practitioner tools and workstyles for user-interface design. Softw. IEEE. 24, 73–80 (2007).
7. Jarzabek, S., Huang, R.: The case for user-centered CASE tools. Commun ACM. 41, 93–99 (1998).
8. Holzinger, A.: Usability engineering methods for software developers. Commun ACM. 48, 71–74 (2005).
9. Alves, R., Teixeira, C., Nascimento, M., Marinho, A., Nunes, N.J.: Towards a measurement framework for tools' ceiling and threshold. In: Proceedings of the 2014 ACM SIGCHI symposium on Engineering interactive computing systems. pp. 283–288. ACM, Rome, Italy (2014).
10. Knack- the easiest online database builder and web app builder., https://www.knackhq.com/?ref=getapp.
11. App Platform for The Enterprise | Mendix, http://www.mendix.com/.
12. OutSystems Platform - High Productivity Platform as a Service - PaaS, http://www.outsystems.com/platform/.
13. Hanington, B., Martin, B.: Universal Methods of Design: 100 Ways to Research Complex Problems, Develop Innovative Ideas, and Design Effective Solutions. Rockport Publishers (2012).
14. Lewis, C., Rieman, J.: Task-centered user interface design. Pract. Introd. (1993).
15. Kato, T.: What "question-asking protocols" can say about the user interface. Int. J. Man-Mach. Stud. 25, 659–673 (1986).
16. Grossman, T., Fitzmaurice, G., Attar, R.: A survey of software learnability: metrics, methodologies and guidelines. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 649–658 (2009).
17. Linja-aho, M.: Evaluating and Improving the Learnability of a Building Modeling System. Hels. Univ. Technol. (2005).
18. Malavolta, I., Lago, P., Muccini, H., Pelliccione, P., Tang, A.: What Industry Needs from Architectural Languages: A Survey. (2013).
19. Myers, B.A., Rosson, M.B.: Survey on user interface programming. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 195–202. ACM, Monterey, California, USA (1992).
20. Jeng, J.: Usability assessment of academic digital libraries: effectiveness, efficiency, satisfaction, and learnability. Libri. 55, 96–121 (2005).