A Method for Evaluating End-User Development Technologies

Full Paper

Claudia de O. Melo
University of Brasília
claudiam@unb.br

Jonathan H. M. de Moraes
University of Brasília
jonathanhmaiademoraes@gmail.com

Marcelo Ferreira
University of Brasília
marcelohpf@gmail.com

Rejane C. Figueiredo
University of Brasília
rejane@unb.br

Abstract

End-user development (EUD) is a strategy that can reduce a considerable amount of business demand on IT departments. Empowering the end-user in the context of software development is only possible through technologies that allow them to manipulate data and information without the need for deep programming knowledge. The successful selection of appropriate tools and technologies is highly dependent on the context in which the end-user is embedded. End-users should be a central piece in any software package evaluation, being key in the evaluation process in the end-user development context. However, little research has empirically examined software package evaluation criteria and techniques in general, and in the end-user development context in particular. This paper aims to provide a method for technology evaluation in the context of end-user development and to present the evaluation of two platforms. We conclude our study proposing a set of suggestions for future research.

Keywords

End-user development, EUD, Technology evaluation, Development tools.

Introduction

End-user development (EUD) aims at enabling end-users and non-specialists in application programming to develop and adapt systems according to their professional, educational or leisure needs (Lieberman, 2006). From the point of view of Software Engineering, EUD means, in general, the 'active participation of end users in the software development process' (Costabile, 2005).

EUD is a strategy that can reduce a considerable amount of business demand on IT departments, generating multiple benefits (McGill, 2004) as higher customer satisfaction with IT. However, empowering the end-user in the context of software development is only possible through technologies that allow them to manipulate data and information without the need for deep programming knowledge (Fischer, 2004). Failures in software package acquisition are not caused by the technology, but by the failure in choosing it in the right way (Misra, 2017), without prioritizing the end-user context and their capabilities.

In the absence of a quality system to evaluate software packages, vendors and users might play their role without any focus and relevance on the requirement of an IT project (Misra and Mohanty, 2003). The success and failure of end-user development within an organization ultimately depends on how effective software packages are used (Montazemi, Cameron, and Gupta, 1996). Therefore, end-users should be a central piece in any software package evaluation, being key in the evaluation process in the end-user development context. Little research has empirically examined software package evaluation criteria and techniques in general, and in the end-user development context in particular (Harnisch, 2014; Jadhav and Sonar, 2009; Jadhav and Sonar, 2011; Misra and Mohanty, 2003). This paper aims at investigating how to evaluate EUD technologies. We developed a model to evaluate end-user development software packages

that can be further extended to other types of technologies. We analyze and present the evaluation results from two platforms using the proposed method. We discuss the evaluation process and results, limitations, and possible future research paths.

Literature Review

According to Lieberman (2006), EUD has encompassed different fields of research, as Human-Computer Interaction (HCI), Software Engineering (SE), Corporate Work Supported by Computers (CSCW), and Artificial Intelligence (AI). We carried out a review of the literature on aspects related to the process of evaluating technologies for end-user developers. Considering the scope of our study (technology evaluation), we found during our review that three different areas have important, but partial, contributions to our research purpose: 1) software package acquisition research; 2) software quality models & CSCW/HCI research, and 3) technology acceptance research.

Software package acquisition research

There are multiple models available which attempt to increase the level of understanding of general software package acquisition processes (Jadhav and Sonar, 2009, 2011; Misra, 2017). Jadhav and Sonar, (2009) presented a systematic review that investigates methodologies for selecting software packages, software evaluation techniques, software evaluation criteria, and systems that support decision makers in evaluating software packages. They selected 60 papers published in journals and conference proceedings. They concluded that there is a lack of a common list of generic software evaluation criteria and its meaning, and that there is a need to develop a framework comprising of software selection methodology, evaluation technique, evaluation criteria, and system to assist decision makers in software selection. The same authors present the framework in a later study (Jadhav and Sonar, 2011).

Damsgaard and Karlsbjerg, (2010) presents seven principles for selecting software packages. First, when a given organization buys packaged software, they join its network. Secondly, they recommend organizations to take a long-term perspective, looking ahead but reasoning back. Then, when choosing packaged software, there is safety in numbers, but organizations should focus on compatibility and be wary of false gold. Finally, they recommend choosing a software package with accessible knowledge, with the right type of standardization, and that all journeys start with a first step. Harnisch, (2014) reviewed the literature on enterprise-level package software acquisition through the lens of IT governance to assess the state-of-the-art of software acquisition governance. His research aims at helping decision-makers to optimize the software procurement processes, governance, and behaviors.

Software quality models & HCI/CSCW research

Software package acquisition methodologies usually compare user requirements with the packages' capabilities. There are different types of requirements, as managerial, political, and quality requirements (Franch and Carvallo, 2003). In their systematic review of software package evaluation and selection, Jadhav and Sonar (2009) found that quality characteristics such as functionality, reliability, usability, efficiency, maintainability, and portability have been used as evaluation criteria in several studies. They also found that, among the ISO/IEC standards related to software quality, ISO/IEC 9126-1 specifically addresses quality model definition and its use as a framework for software evaluation. ISO/IEC 9126 was replaced by ISO/IEC 25010.

In the context of end-user development technology evaluation, we consider the ISO an important guide to define characteristics, attributes, and metrics. However, to be able to capture the needs of an end-user developer, which is not the same as a professional developer nor a final end-user, further refining is probably required.

In addition, we argue that usability is a central characteristic for any model focusing on evaluating software packages. We have noticed that both software package acquisition research and software quality models research did not pay enough attention on investigating better usability evaluation techniques. There is widely accepted well-documented evaluation method for diagnosing potential usability problems in user interfaces, such as Nielsen's heuristic evaluation (Nilsen, 2012). Based on Nielsen's work, Baker, Greenberg, and Gutwin, (2002) developed an evaluation method that looks for groupware-specific

usability problems. Their results suggested that an evaluation using their groupware heuristics can be an effective and efficient method to identify teamwork problems in shared workspace groupware systems.

Technology acceptance research

In general, technology acceptance research focuses on the perceived usefulness, ease of use, and user acceptance of Information Technology. Davis (1989) presented a seminal work in this field, the Technology Acceptance Model (TAM), aimed at explaining user behavior across a broad range of end-user computing technologies and user populations.

TAM has already evolved into an unified theory of acceptance and use of technology (UTAUT) (Venkatesh, Morris, Davis, and Davis, 2003). UTAUT has four independent variables as performance and effort expectancy, social influence, and facilitating Conditions. It is a useful tool for managers to assess the likelihood of success for new technologies introduced and the drivers of acceptance. Thus, they can design conditions to facilitate their adoption.

Along the same line as TAM and UTAUT, Doll and Torkzadeh, (1988) developed a model called End-User Computing Satisfaction (EUCS) to measure the affective attitude from an user while interacting with a specific computer application. The model contains dimensions such as content, accuracy, format, ease of use, and timeliness.

Despite the fact that this body of research is extremely valuable for understanding end-users in general, we argue that end-user developers have a different role when interacting with software packages. They develop working software through them, so technology acceptance research might not answer what is needed for an organization to decide which software package is more suitable for end-user developers.

A method for evaluating EUD Technologies

Evaluating and selecting software packages that meet organizational and end-user requirements is a non-trivial Software Engineering process (Jadhav and Sonar, 2009). To the best of our knowledge, there is no *technology evaluation methodology* particularly focused on *end-user developers and their context*. Therefore, we propose a method based on a new extended literature review on the three aforementioned research areas. In addition, we evaluate two platforms to test and refine our method. In the context of EUD, a technology evaluation model may consider elements from the three research fields we identified in the literature review: software package acquisition models, software quality models and CSCW/HCI models, and finally technology acceptance models. Based on our interpretation, the model should have:

• Essential qualities that enable the end-user developer to manipulate the tool and produce useful results in a certain application domain (from software quality models and CSCW/HCI models);

• General qualities inherent to software packages (from software package acquisition models and technology acceptance models);

• Essential qualities for management and technological governance (from software package acquisition models);

• An evaluation method based on already-established and tested techniques, even if they come from a different context (from all models).

Evaluation criteria, characteristics, sub characteristics, and attributes

To be able to evaluate EUD technologies, we designed a structured quality model (Franch and Carvallo, 2003) that provides a taxonomy of software quality features and metrics to compute their value. Based on the specific EUD domain and our literature review, we selected appropriate quality **characteristics**, determined quality sub characteristics, decomposed them into **attributes** and finally developed **questions and metrics** from different **points-of-view**.

Points-of-view are particularly important as they indicate who should answer the question, or the most important stakeholder interested for that question. Moreover, as the model focuses on technology evaluation, sometimes the software packages are the main subject of the evaluation, and sometimes the output of the software packages (that is also software) is the main subject.

All characteristics, sub-characteristics and related attributes were defined after a thorough analysis of the reviewed literature and a pilot testing where two platforms were evaluated. We also organized the quality characteristics into 4 criteria presented on the systematic review of Jadhav and Sonar, (2009): 1) Functional, 2) Cost and benefit, 3) Vendor, and 4) Software quality. Tables 1 and 2 present our evaluation model, that comprises criteria, characteristics, sub characteristics, literature references, attributes, and points-of-view. The complete model, also containing the questions and metrics is available at https://itrac.github.io/eud_technology_evaluation. From our literature review, we selected 11 fundamental characteristics to evaluate EUD technologies, which in turn are refined into 20 sub-characteristics, and finally 30 attributes. These attributes are measured by 300 questions, all of them initially collected from already established and tested techniques. We detail the model in the following sections.

Functional characteristics

Functionality is the capability of the software product to provide functions that meet stated and implied needs when the software is used under specified conditions. There are many possible related sub-characteristics that are already covered by our model. So we selected the main target application domain, that is the functional area(s) for which the software is especially oriented or strong (Jadhav and Sonar, 2009). We used the taxonomy provided by Richardson and Rymer (2016). *Collaboration* is the ability to edit documents synchronously (Iacob, 2011). When the tool meets the heuristics, it indicates that there are few or no usability errors for collaborative development (Baker et al., 2002). Despite that, end-user developers develop solutions for themselves and less frequently for their peers, and reuse software in an unplanned fashion. It is also expected that technologies support collaboration between professional software developers and end-user developers, or among end-user developers (Ko, 2011). Thus, we selected Collaboration Technology as a sub-characteristic, further derived into 5 attributes to understand to what extent the technology provides support (Baker et al., 2002; Iacob, 2011): 1) Shareability; 2) Coordination of actions; 3) Consequential communication; 4) Finding collaborators and establishing contact; and 5) Concurrent protection.

Data management is the business function of planning for controlling and delivering data and information assets (Cupoli, Earley, and Henderson, 2009). For the context of end-user development, it is important to ensure the platform's data and database capabilities evolve while the application is developed (Sadalage and Fowler, 2016) and the platform capabilities to send and retrieve data from external systems and databases (Mika, 2006). Data processing is one of the common tasks performed by an end-user and the data management should be simplified in technical terms, to ensure simplicity in the development (Doll and Torkzadeh, 1988).

Cost and benefit & Vendor characteristics

We adopted all cost attributes from the literature review conducted by Jadhav and Sonar, (2009). Benefits are covered by other characteristics from our model. The selected cost attributes are: 1) License cost of the product in terms of number of users; 2) Cost of training to users of the system; 3) Cost of installation and implementation of the product; 4) Maintenance cost of the product; and 5) Cost of upgrading the product when new version are launched.

General vendor characterization Jadhav and Sonar, (2009) found a number of vendor attributes, some of them are already covered in our method (e.g., user manual and tutorial are covered by the usability characteristic). We thus selected three essential attributes for characterizing a vendor: 1) Experience of vendor about development of the software product, 2) Popularity of vendor product in the market, and 3) Number of installations of the software package. *Vendor dependency/Switching costs* are a consequence of buyer switching between alternative suppliers of essentially the same product. Large switching costs can make buyers reluctant to switch suppliers (Greenstein, 1997). The more dependent on a vendor, the higher is the probability to incur into large switching costs. Long terms and being dependent imply more cost and less ability to innovate through Information Technology as the company is locked-in with a specific IT supplier. We argue that anticipating the vendor dependency analysis will increase an

organizations' ability to understand possible future switching costs and reflect on the trade-offs that dependency brings to a company innovativeness.

Software quality characteristics

Compatibility is the degree with which two or more systems or components can exchange information and/or perform their required functions while sharing the same hardware or software environment (ISO/IEC 25023, 2011). Four compatibility attributes were considered: 1) Technical knowledge requirement; 2) Data exchangeability; 3) Connectivity with external component/system; and 4) Reusability. *Maintainability* is the capability of the software product to be modified. Modifications may include corrections, improvements or adaptation of the software to changes in the environment, and in requirements and functional specifications (ISO/IEC 25023, 2011). Two main attributes were considered: 1) Modifiability and 2) Reusability.

Usability is the capability of the software product to be understood, learned, used, and being enticing to the user when used under specific conditions (ISO/IEC 25023, 2011). Fifteen attributes were considered in terms of usability: 1) Visibility of system status; 2) Match between system and the real world; 3) User control and freedom; 4) Consistency and Standards; 5) Help for users to recognize, diagnose, and recover from errors; 6) Error prevention; 7) Recognition rather than recall; 8) Flexibility and minimalist design; 9) Aesthetic and minimalist design; 10) Help and documentation; 11) Skills; 12) Pleasurable and respectful interaction with the user; 13) Privacy; 14) Accessibility; and 15) Localization.

Reliability is the capability of the software product to maintain a specified level of performance when used under specified conditions (ISO/IEC 25023, 2011). Two reliability attributes were considered: 1) Availability; and 2) Vendor support. *Performance Efficiency* is the capability of the software product to provide appropriate performance, relative to the amount of resources used, under stated conditions (ISO/IEC 25023, 2011). Two attributes were considered: 1) Response time; and 2) Turnaround time. *Security* is the capability of the software product to protect information and data so that persons or other products or systems have the degree of data access appropriate to their types and levels of authorization (ISO/IEC 25023, 2011). Six security attributes were considered: 1) Access behaviors; 2) Security behaviors; 3) Update behaviors; 4) File upload Security, 5) Report behaviors; and 6) Security algorithms.

Points-of-view

We defined two variables to define the points-of-view: stakeholder and technology. From a stakeholder perspective, our model has questions related to end-user developers and to the organization's governance team. From a technology perspective, the questions focus on the software package being evaluated (platform) or on its output (the application generated).

Evaluating two platforms with the proposed method: Results and Discussion

We applied our technology evaluation method for analyzing two platforms. The research team has a Software Engineering background, both in academia and industry. To report the overall steps, we structure the stages using the sequence proposed by Jadhav and Sonar, (2009). We explain how we conducted each step in the EUD context. The last two stages proposed - negotiating a contract and purchasing and implementing the software package - are outside the scope of this study.

Determining the need, including high-level investigation of software features and capabilities provided by vendors Literature and market research helped to obtain the available EUD technologies. The first criterion to form a list of candidate tools was market analysis. Reports such as the one provided by Richardson and Rymer, (2016) help to get an overall picture of the market and of the needs to be updated, as the software market is always changing. Between August/2016 and October/2016 we carried out a literature review and contacted the leaders of public and private organizations to build a general list of tools. The literature review on tools was comprehensive, iterative, and incremental. We looked for technologies associated with the following search strings:

"EUAD" OR "EUD" OR "citizen development" OR "end-user development" OR "end-user software engineering" OR "Low code" OR "Shadow IT" OR "User-centric" OR "RAD" OR "customer-facing applications" OR "End-user computing" OR "End-user programming".

Criteria	Characteristic	Sub characteristic	References	Attributes	Point-of-View
Functional	Functionality	Main target	(Richardson and Rymer, 2016)	Application domain (6 items)	Governance/platform
Functional	Collaboration	Collaboration Technology	(Andriessen, 2012) (Baker, Greenberg, and Gutwin, 2002)	Shareability (1 item) Coordination of actions (1 item) Consequential communication (5 items) Finding collaborators and establishing contact (4 items) Concurrent protection (2 items)	End-user developer/platform
Functional	Data Management	Data Management	(Sadalage and Fowler, 2016) (Mika, 2006)	Data management process (2 items) Data input and output (3 items) Required technical knowledge (3 items)	End-user developer/platform Governance/platform
Cost and benefit	Cost	Cost	(Jadhav and Sonar, 2009)	License cost (5 items) Maintenance cost (1 item)	Governance/platform
Vendor	Vendor	Vendor Characterization Vendor Dependency/ Switching costs	(Greenstein, 1997) (Lichtenstein, 2004)	Vendor length of experience (1 item) Product history (1 item) Number of product installations (1 item) Contract dependency (2 items) Technology dependency (2 items)	Governance/platform Governance/application

Table 1: Criteria related to functional, cost & benefits, and vendor characteristics

After a high-level investigation of the results, we narrowed the search space to software packages. This is because there are is wide variety of solutions available to support the end-user and thus a large combination of analytical methods needed to evaluate them.

Short listing candidate packages and eliminating the candidate packages that do not have the required feature. In this stage, we decided to consider only the most solid market offers. We thus selected two tools based on the market leadership scenario (Richardson and Rymer, 2016): Oracle Apex (Oracle Application Express) and OutSystems (Outsystems Platform).

Using the proposed evaluation technique to evaluate remaining packages and obtain a score We applied our technology evaluation method to the two selected platforms. Table 3 describes a summary of the evaluation results obtained. We present in this study only the results consolidated by each attribute. Figure 1 illustrates the usability evaluation results separately, since it has 15 attributes. A complete evaluation can be found at https://itrac.github.io/eud_technology_evaluation. We chose the heuristic evaluation to perform this analysis and used four scenarios that consist basically of creating an application, either using the platform's predefined templates or not.

To perform a heuristic evaluation, 3 evaluators should inspect the interface separately, so 3 members of our research team participated. Only after completing their evaluations, they communicated and aggregated their findings. This procedure is important to ensure independent and unbiased evaluations

(Nielsen and Mack, 1994). During the evaluation session, the evaluator went through the interface several times, inspected the many dialogue elements and compared them with the list of questions from our model.

Criteria	Characteristic	Sub characteristic	References	Attributes	Point-of-View
Software quality	Compatibility	Interoperability	(Sherman, 2016) (Srivastava, Sridhar, and Dehwal, 2012) (ISO/IEC 25023, 2011)	Technical knowledge requirement (1 item) Data exchangeability (1 item) Connectivity with external component/system (3 items) Reusability (1 items)	End-user developer/application generated
Software quality	Maintainability	Modifiability Reusability	(ISO/IEC 25023, 2011)	Modifiability (3 items) Reusability (3 items)	End-user developer/application generated End-user developer/platform
Software quality	Usability	Appropriateness recognizability Learnability Operability User error protection User interface aesthetics Accessibility	(ISO/IEC 25023, 2011) (Weiss, 1994) (Nielsen and Mack, 1994) (Pierotti, 2004) (IBM Corporation, 2016A) (IBM Corporation, 2016B) (*Localization Testing Checklist - A Handy Guide for Localization Testing*)	Visibility of System Status (20 items) Match Between System and the Real World (12 items) User Control and Freedom (19 items) Consistency and Standards (26 items) Help Users Recognize, Diagnose, and Recover from Errors (16 items) Error Prevention (8 items) Recognition Rather Than Recall (25 items) Flexibility and Minimalist Design (9 items) Aesthetic and Minimalist Design (10 items) Help and Documentation (18 items) Skills (11 items) Pleasurable and Respectful Interaction with the User (6 items) Privacy (5 items) Accessibility (12 items) Localization (15 items)	End-user developer/platform
Software quality	Reliability	Availability	(Banerjee, Srikanth, and Cukic, 2010) (Gray and Siewiorek, 1991) (Lehman, Perry, and Ramil, 1998)	Availability (2 items) Vendor support (5 items)	Governance/platform
Software quality	Performance efficiency	Time behavior	(ISO/IEC 25023, 2011)	Response time (2 items) Turnaround time (4 items)	End-user developer/platform
Software quality	Security	Integrity Confidentiality	(Stanton, Stam, Mastrangelo, and Jolton, 2005) (Hausawi, 2016) (ISO/IEC 25023, 2011)	Access behaviors (5 items) Security behaviors (2 items) Update behaviors (2 items) File Upload Security (2 items) Report behaviors (1 item) Security algorithms (3 items)	End-user developer/application generated End-user developer/platform Governance/platform

Table 2: Criteria related to software quality characteristics

From the evaluation results, it is possible to contrast characteristics of both platforms and, depending on the organization's priorities, to rank them. Without any prioritization, we can interpret that Outsystems has the best chance to fulfil an end-user developers' requirements as it scored better than Oracle Apex.

Pilot testing the tool in an appropriate environment We performed this stage in parallel with the previous stage as we used four scenarios to create applications across platforms, simulating the behavior of an end-user developer. This stage was fundamental to refine the model proposed, and to remove, rewrite, and add questions/metrics to it. The evaluation model and the platform evaluation results presented in this work are already the result from a second evaluation iteration. The successful selection of tools and technologies for end-user developers is highly dependent on the context in which the end-user is embedded, such as business domain characteristics, the organization's culture, and the end-user motivation to apply or develop technical skills (Fischer, 2004). One limitation in our study is that we did not evaluate the platforms in a real world scenario. To address this limitation, we developed four common scenarios of simple information systems that enable create, read, update, and delete information (CRUD scenarios).

Characteristic	Attributes	Oracle Apex	OutSystems
Functionality	Application domain	**Database** data analysis; graphics generation; employer's control; calendar; data mining; spatial database; responsive interfaces	**General-purpose** Process-based development; Web and mobile applications; Library with more than 100 base interfaces; Deploy control tool
Compatibility	Technical knowledge requirement	Advanced	Advanced
	Data exchangeability	N/A	100%
	Connectivity with external component/system	RPC; Service Oriented Integration; Messaging	RPC call; Messaging passing; Software service
	Reusability	100%	100%
Maintainability	Modifiability	100%	100%
	Reusability	Possible	Possible
Reliability	Availability	N/A	N/A
	Vendor support	Avg time for new release: 179,58	Avg time for new release: 20.30
		Avg fixes in each release: 73.46	Avg fixes in each release: 8.76
Performance efficiency	Response time	100%	100%
	Turnaround time	50%	100%
Security	Access behaviors	80%	60%
	Security behaviors	100%	100%
	Update behaviors	100%	50%
	File Upload Security	100%	100%
	Report behaviors	100%	100%
	Security algorithms	67%	100%
Cost	License cost	$ 164,839.00	$ 2,072,601.74
	Maintenance cost	N/A	N/A
Vendor	Contract dependency	100%	100%
	Technology dependency	100%	0%
Collaboration	Shareability	100%	100%
	Coordination of actions	100%	100%
	Consequential communication	60%	60%
	Finding collaborators and establishing contact	75%	0%
	Concurrent protection	0%	50%
Data Management	Data input and output	input: TXT, XML, CSV, SQL, REST, SOAP. output: CSV, REST.	input: CSV, TXT, XML, XLS, JSON, SQL, SOAP, REST. output: CSV, REST
	Required technical knowledge	33%	33%

Table 3: Summary of Oracle Apex and OutSystems evaluation results

Conclusion and Future Work

We propose a method for evaluating end-user development (EUD) technologies here, based on an extensive work of literature research. We also presented the evaluation of two platforms using four scenarios. The evaluations have improved and refined the model. This paper sheds light on under-researched questions related to the end-user development context in general, and in the EUD technology

evaluation in particular. The major original contributions of this paper are (1) a detailed method for evaluating EUD technologies

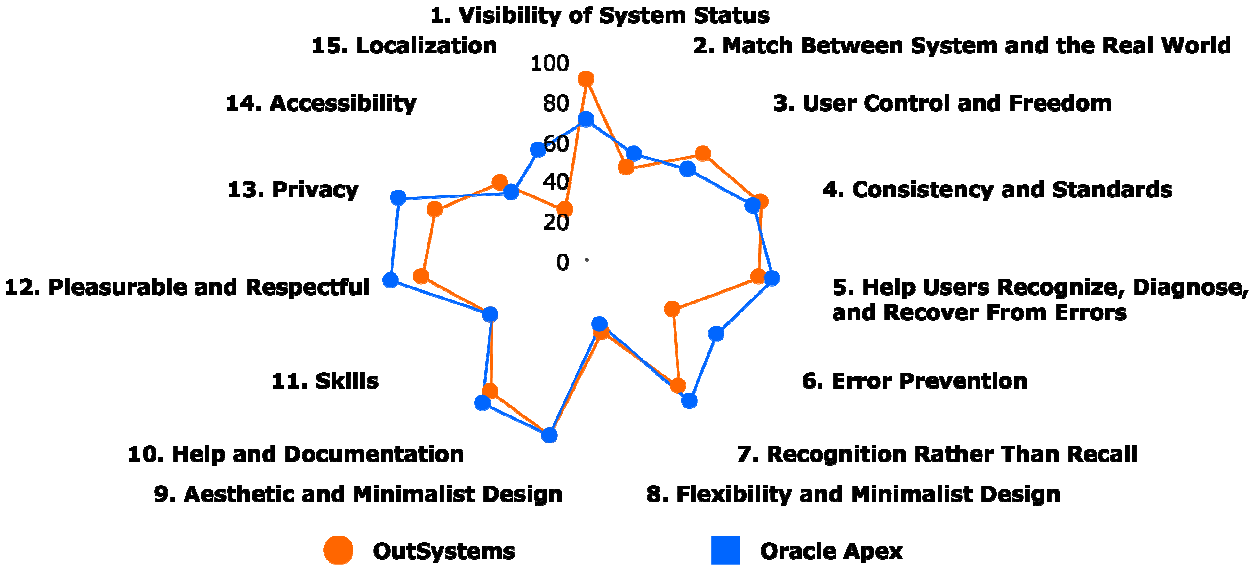

Figure 1: Usability analysis result of the two platforms

that comprises 11 characteristics, 20 sub-characteristics, 30 attributes, and 300 questions/metrics, and (2) examples of two evaluations using our method against leading EUD platforms in the market.

This work points to the need for some interesting future studies. The next step is to refine the method through real-world scenarios, and to evolve the model to assist decision makers with the evaluation and selection of software packages, using evaluation techniques such as analytic hierarchy process or weighted scoring method (Jadhav and Sonar, 2011). There is an avenue for exploring evaluation automation, for instance with the use of dynamic application security testing tools to support security characteristics' evaluation. We also plan to explore existing research on user requirements determination to improve end-user developer acceptance, their success, and consequently the success of the software package acquisition. In addition, and given the proliferation of technologies, we defined that, in the context of a first analysis of tools, it would be appropriate to select tools with a higher degree of maturity. The higher the maturity, the lower the risk of a given technology. Emerging technologies, however, are riskier and potentially more innovative. Researchers could look at them in future studies.

REFERENCES

Andriessen, J. (2012). Working with Groupware: Understanding and Evaluating Collaboration Technology. Computer Supported Cooperative Work. Springer-Verlag London.

Baker, K., Greenberg, S., and Gutwin, C. (2002). "Empirical development of a heuristic evaluation methodology for shared workspace groupware." *Proceedings of the 2002 ACM conference on Computer Supported Cooperative Work, CSCW '02*. ACM Press, USA, 96.

Banerjee, S., Srikanth, H., and Cukic, B. (2010). "Log-Based Reliability Analysis of Software as a Service (SaaS)." *IEEE 21st International Symposium on Software Reliability Engineering*. IEEE, 239–248.

Costabile, M. F. et al. (2005). "A Meta-Design Approach to End-User Development." Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VLHCC '05), pp. 308-310.

Cupoli, P., Earley, S., and Henderson, D. (2009). DAMA - Data Management Book of Knowledge. 1st ed. Technics Publications, LLC, Post Office Box 161 Bradley Beach, NJ 07720.

Damsgaard, J. and Karlsbjerg, J. (2010). "Seven Principles for Selecting Software Packages." *Communications of the ACM* 53.8, 55–62.

Davis, F. D. (1989). "Perceived Usefulness, Perceived Ease of Use, and User Acceptance of Information Technology." *MIS Quarterly* 13.3, 319.

Doll, W. J. and Torkzadeh, G. (1988). "The Measurement of End-User Computing Satisfaction." *MIS Quarterly* 12.2, 259–274.

Fischer, G. et al. (2004). "Meta-design." *Communications of the ACM* 47.9, 33.

Franch, X. and Carvallo, J. P. (2003). "Using Quality Models in Software Package Selection." *IEEE Software* 20.1, 34–41.
Golota, H. *Localization Testing Checklist - A Handy Guide for Localization Testing*. URL: https://www.globalme.net/blog/localization-testing-checklist (visited on 03/01/2017).
Gray, J. and Siewiorek, D. (1991). "High-availability computer systems." *Computer* 24.9, 39–48.
Greenstein, S. M. (1997). "Lock-in and the Costs of Switching Mainframe Computer Vendors: What Do Buyers See?"*Industrial and Corporate Change* 6.2, 247.
Harnisch, S. (2014). "Enterprise-level Packaged Software Acquisition: a Structured Literature Review Through the Lens of IT Governance." *European Conference on Information Systems (ECIS) 2014*.
Hausawi, Y. M. (2016). "Current Trend of End-Users' Behaviours Towards Security Mechanisms." Springer, Cham, 140–151.
Iacob, C. (2011). "Design Patterns in the Design of Systems for Creative Collaborative Processes." *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 359–362.
IBM Corporation (2016B). *IBM Accessibility Checklist for Software*.
IBM Corporation (2016A). *IBM Accessibility Checklist for Web and Web-Based Documentation*.
ISO/IEC 25023 (2011). *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality*. Technical Report.
Jadhav, A. S. and Sonar, R. M. (2009). "Evaluating and Selecting Software Packages: A Review." *Information and Software Technology* 51.3, 555–563.
Jadhav, A. S. and Sonar, R. M. (2011). "Framework for Evaluation and Selection of the Software Packages: A Hybrid Knowledge Based System Approach." *The Journal of Systems and Software* 84.8, 1394–1407.
Ko, A. J. et al. (2011). "The State of the Art in End-user Software Engineering." *ACM Computing Surveys (CSUR)* 43.3,21:1–21:44.
Lehman, M., Perry, D., and Ramil, J. (1998). "Implications of evolution metrics on software maintenance." *Proceedings. International Conference on Software Maintenance (Cat. No. 98CB36272)*, 208–217.
Lichtenstein, Y. (2004). "Puzzles in Software Development Contracting." *Communications of ACM* 47.2, 61–65.
Lieberman, H., Paterno, F., Klann, M. and Wulf, V. (2006). End-user development: An emerging paradigm. End User Development. (2006), 1–8.
McGill, T. (2004). Advanced Topics in End User Computing. Vol. 3. IGI Global.
Mika, S. (2006). *Five Steps to Database Integration*. URL: http://www.government-fleet.com/article/story/2006/03/five-steps-to-database-integration.aspx (visited on 12/12/2016).
Misra, H. (2017). "Managing User Capabilities in Information Systems Life Cycle: Conceptual Modelling." *International Journal of Information Science and Management* 15.1, 39–58.
Misra, H. and Mohanty, B. (2003). *The IT-acquisition models and user's perspective: a review*. Tech. rep.
Montazemi, A. R., Cameron, D. A., and Gupta, K. M. (1996). "An Empirical Study of Factors Affecting Software Package Selection." *Journal of Management Information Systems* 13.1, 89–105.
Nielsen, J. and Mack, R. L. (1994). Usability inspection methods. Wiley. Nilsen, J. (2012). *Usability 101: Introduction to Usability*.
Oracle. *Oracle Application Express*. URL: https://apex.oracle.com/en (visited on 03/01/2017).
Outsystems. *Outsystems Platform*. URL: https://www.outsystems.com (visited on 03/01/2017).
Pierotti, D. (2004). *Heuristic Evaluation - A System Checklist*. URL: ftp://ftp.cs.uregina.ca/pub/class/305/lab2/example-he.html (visited on 03/01/2017).
Richardson, C. and Rymer, J. R. (2016). *The Forrester Wave ™: Low-Code Development Platforms*. URL: http://agilepoint.com/wp-content/uploads/Q2-2016-Forrester-Low-Code.pdf(visited on 12/2016).
Sadalage, P. and Fowler, M. (2016). *Evolutionary Database Design*. URL: https://martinfowler.com/articles/evodb.html (visited on 02/20/2017).
Sherman, R. (2016). *How to evaluate the features of data integration products*. URL: http://searchdatamanagement.techtarget.com/feature/How-to-evaluate-the-features-of-a-data integration-product (visited on 10/24/2016).
Srivastava, K., Sridhar, P. S.V. S., and Dehwal, A. (2012). "Data Integration Challenges and Solutions: A Study." *International Journal of Advanced Research in Computer Science and Software Engineering* 2.7, 34–37.
Stanton, J. M., Stam, K. R., Mastrangelo, P., and Jolton, J. (2005). "Analysis of end user security behaviors." *Computers & Security* 24.2, 124–133.
Venkatesh, V., Morris, M. G., Davis, G. B., and Davis, F. D. (2003). "User Acceptance of Information Technology: Toward a Unified View." *MIS Quarterly* 27.3, 425–478.
Weiss, E. (1994). Making Computers People-Literate. 1st. Jossey-Bass Inc., Publishers.