DOI: 10.15439/2017F290
ISSN 2300-5963 ACSIS, Vol. 11

Fundamentals of a Components Sharing Network to Accelerate JavaScript Software Development

Daniel Souza Makiyama
Universidade Federal do ABC (UFABC)
Centro de Matemática Computação e Cognição (CMCC)
Santo André – SP - Brasil
Email: daniel.makiyama@gmail.com

Plinio Thomaz Aquino Jr.
Centro Universitário FEI
Fundação Educacional Inaciana Pe. Sabóia de Medeiros
Av. Humberto A. Castelo Branco, 3972 - 09850-901
Sao Bernardo Campo - SP – Brasil
Email: plinio.aquino@fei.edu.br

Abstract— **Based on a systematic review of empirical studies about software components selection and usability techniques applied to a functional prototype, this article maps the functional and non-functional requirements of a components sharing network that aims to accelerate JavaScript software development. Results point out that integrating the development environment to a component search mechanism with automated filters, ordered by quality criteria, and allow code snippets rank and improvements submission on a version control system are the path to accelerate the development and motivate IT students and professionals to participate in this network.**

I. Introduction

SINCE the 90th, known as the Dot-com bubble, many has changed regarding user experience on the web. In October 2014, W3C has released the fifth revision of HTML5, essential technology for multiple platforms [1]. As web scope increases, targeting almost every device, a plethora of frameworks and components have been released to simplify development on this platform. Bower (bower.io), a package manager created to store frameworks, libraries, assets, and utilities for HTML, CSS and JavaScript development has more than 60,000 packages on its database (data collected on January, 2017). Building systems through third party components reuse has being recognized as a crucial success factor in software industry, but only a few companies formalize their selection processes and employ any method to document their decisions [2].

Usually, API documentation is insufficient to assist programmers while coding. A survey [3] shows that the crowd can significantly enhance an existing API documentation, and indicates there is a strong association between API coverage on Stack Overflow and its usage in real software systems. Relevance data extracted from crowd participation can help narrowing down component options and preventing the YAFS syndrome [4], when developers tend to create new frameworks instead of using frameworks with the same features, because they could not afford evaluating a large number of options.

This paper aims to join the analysis of empirical studies and human-computer interaction techniques for eliciting functional and non-functional requirements of a components sharing network and understand how IT students and professionals could benefit from this approach and get motivated to participate in this network.

The following sections present the whole survey process that contemplates the analysis of papers on empirical methods and current industry practice for components selection, the creation of a prototype to elicit the possible requirements of this network, usage observation and focus group with 26 IT students, and a heuristic assessment on the prototype conducted by 3 specialists.

II. Background

In the academic context, [21] evaluates quality, validation and performance of 7 JavaScript web frameworks. On quality perspective, they analyzed size, complexity and maintainability using JSMeter (jsmeter.info), Cloc (cloc.sourceforge.net) and Understand (scitools.com). On validation perspective, critical and high severity errors were analyzed with Yasca (sourceforge.net/projects/yasca) and JSLint (javascriptlint.com) tools. Lastly, on performance perspective, SlickSpeed (github.com/kamicane/slickspeed) was used in 7 different browsers and 4 operational systems. As detected in [22], there is lack of studies to help professionals to select best JavaScript framework by its purpose and functionalities, as specific concerns on JavaScript frameworks are not addressed in more generic component selection methods.

In [21] they present important criteria that are missing in most academic studies, extracted from a questionnaire applied to 4 front-end developers: adequacy of the documentation to user needs, how many people contribute and use the code, and how fast it is for the component to bring value to user's application. All criteria listed vary according to user / project constraints. Besides these studies [21] and [22] approaches same language, another proximity between this study and [21] is the intention of reusing existing OSS tools to provide metrics on JavaScript code.

This study is focused on code snippets and components, and the other 2 are more focused on frameworks. The demand for organizational tools for JavaScript development is perceived by software community and has being addressed by package managers like Bower and scaffolding tools like

Yeoman (yeoman.io). Developers heavily use these tools, but still they keep continuously seeking new tools that will make them deliver faster and with a better quality. In this sense, there is room for new tools that addresses problems still not solved, e.g. how to classify JavaScript packages by the feature(s) they provide [22].

In this study, the agile requirements engineering process with prototypes is applied [20]: initial requirements are elicited, prototype is built / updated, submitted to end user revision, and prototype is refactored for next iteration. Prototypes increase motivation for requirements gathering and force users to discuss about requirements in less subjective terms [20]. Based on [5], a key question was defined: what are the empirical studies recently published related to the selection or evaluation of open-source (known as OSS) or commercial-off-the-shelf (known as COTS) components? Table I shows the search string generated.

TABLE I.
INDUSTRY PRACTICE THROUGH EMPIRICAL STUDIES

Libraries	Articles	Search string
IEEEXplore	59	(("empirical study") AND ("Open Source Software" OR "Off-The-Shelf") AND ("Software evaluation" OR "Software selection" OR "Component selection" OR "Component evaluation"))
ACM DL	24	
Science Direct	30	
Springer Link	16	
topics: title and abstract; language: English; when: 2005 to Jan. 2017; discipline: Computer Science; types: articles, conferences & chapters		

[Author] Objective	focus | target | tool | participants
[6] map reasons COTS or OSS are used in Norway, Italy & Germany	COTS, OSS | decision makers | questionnaires | 127 companies
[7] understand how researches can contribute to practice in Norway	OSS | developers | questionnaires | 16 software companies
[8] identify the principles of software packages selection	COTS | decision makers | interviews | 39 people
[9] investigate COTS selection practice in Jordan companies	COTS | decision makers | questionnaires | 10 companies
[10] understand components selection practice and emphasize underestimated topics in academy	COTS, OSS | developers | interviews | 23 people / 20 software companies
[11] challenges on OSS component selection, licensing and maintenance on Chinese software companies	OSS | decision makers | questionnaires | 43 companies
[12] Examine state-of-practice in OTS component-based development	COTS, OSS | decision makers | questionnaires | 127 companies

Industry Practice Characteristics [found in:]
ad-hoc and situational; generic selection methods not applied [7][9][10]
rely on developer team's previous experiences [7][9][10][11][12]
selection process, criteria and decisions are not registered [9][7]
search engine (google) is the source for new components; repositories rarely used [7][10][11][12]
market is continuously monitored [7][10][9]
selection happens in early development phases [10]
selection can happen in any phase, based on project context & flexibility [12]
evidences of real component usage matters on decision [10]
comply to functional requirements and project constraints [7][10][11]
future support assurance matters on decision [6][10][9]
bring less effort and take less time to apply matters on decision [6]
licensing terms matters on decision [11]

To filter these papers, the following criteria was defined: remove duplicated surveys, in-progress studies and outdated survey versions; and select papers approaching industry practice in component selection through an empirical study; or academic methods or criteria for component selection. This filter reduced the list to 46 articles, 38 focused on academic methods or criteria, which will not be cover on this article. Table I shows the remaining 7 articles that approaches industry practice through an empirical study.

In a research on component selection practices with architects and researchers that perform this activity, [23] concluded that 4 criteria are fundamental for component assessment: features, non-functional attributes, architecture compatibility and business considerations. Evaluation process is an iterative process interleaved with requirements engineering. Based on the analysis of the selected articles, focusing on their main conclusions from interviews and questionnaires, the key characteristics of the Industry practice on selection is summarized in Table I. These characteristics should be considered in a component selection process more connected to the industry practice.

III. PROTOTYPE, FOCUS GROUP AND HEURISTIC RESULTS

The main purpose of this prototype was to showcase a variety of possible features available on a components sharing network. The prototype focus was on user interface, a proposed taxonomy and selection criteria. The prototype was not linked to an IDE (Integrated Development Environment), but hosted in a web server. The prototype is the artifact that allows specialists and users to provide very early feedback on **requirement** elicitation, data **taxonomy** and terminologies, relevant selection **criteria**, possible **integrations** to bring value to the solution and detailed **use cases** that could address real problems.

The prototype was designed to be a repository of component bootstraps. Every component would contain a package with dependent files (scripts/resources/styles) and a code snippet that could be easily applied in user's code. These packages would be classified by feature and accessible through a search engine. The first criteria supported would be performance comparison through test cases evaluation and users rating. In search page, user can filter a feature by name and navigate to a list of components that implements this feature, referred in the prototype as techniques. Sample data was extracted manually from blogs, books and framework's documentation. Test cases were created for every feature, and a benchmark tool (benchmarkjs.com) was used to run them. For component rates, mocked data was used; the assumption is that when the final tool were delivered, users will start rating the components they use. After component selections, users would be able to generate an online documentation of their selections and packages with dependencies and snippets, named receipts. A simple reputation system was simulated, where users would earn points by adding snippets, ranking or generating receipts.

Two sessions were conducted with 26 IT students in computer labs of FEI University Center in São Bernardo do Campo, São Paulo, Brazil. Programmers composed the majority of the group: 73% of the group works with IT, 92%

read and write in English, 38% code some days every week and 31% code on a daily basis; 69% informed that are familiar to JavaScript language and CSS, and 92% are familiar to HTML. Activities contemplated a questionnaire to map group expertise level, user observation, where pairs had to complete a list of tasks while being observed, a post questionnaire and a focus group to discuss post questionnaire answers. Post questionnaire topics were: how people should use this tool (Q1); if they agree a code snippets database would play a crucial role in component selection (Q2); if they recognize any differential between this tool and other tools available in the market (Q3); and if the recognition simulation is seen as relevant (Q4). Researcher role was to moderate discussion and not influence group [14].

On Q1, group agreed users would use this tool to rank the best snippets and receipts (88%), and when tool had more access, use it to extract component development patterns (85%), find solutions recognized by development community (81%) and share them in Question and Answer (Q&A) sites (81%). Group agreed the receipts concept was not clear so they would not use it. On Q2, group agreed code snippets available on the web influence JavaScript, HTML and CSS development (85%). 96% agree code snippets that work are the information source that most helps when adopting a framework, confirming [10] results. The second main information source is technical blogs (85%) followed by Q&A sites (81%).

On Q3, groups disagreed. First group agree this tool has potential, but it should be integrated with existing tools like GitHub (github.com). The other group argued they would only use this tool if it could compete with tools like Stack Overflow in performance and search engine quality. On Q4, recognition mechanisms are considered positive by 69% of the group, but group pointed out its relevance depends on how it prevents people from cheating.

Heuristic evaluation was conducted by 3 specialist during three days. Specialists recommended organization, quality, communication and integration changes.

Table II shows the fundamentals of a Component Sharing Network based on the software engineering dimensions: Requirements, Integrations, Taxonomy, Criteria and Use Cases. This list integrates data from academic background, user observation and heuristic evaluation.

TABLE II.
FUNDAMENTALS OF A COMPONENT SHARING NETWORK

Structure Requirements: invest on search engine; User area should list project's snippets, components & versions; Component and snippet's rank should be per criteria; Search should be contextualized by project metadata; Component comparison should be by criteria (adherence, performance), not code; Components already used in user's project should have precedence in search results; IDE results should be ordered by best option based on project metadata; allow users to add test cases to an existing snippet; tool should support storing private data for companies; allow running performance reports for the entire project
Quality Requirements: pay special attention to search engine and package dependency resolver performance; performance analysis should be done in background; tool should reduplicate code; recognition system should stimulate user interaction and avoid cheating; a tag system should be used to classify snippets; avoid anonymous user to submit content to the tool; moderation of abusive content; free text restrictions should be applied; moderation program should be stablished, with moderators chosen by their reputation on the network
Communication Requirements: clearly inform languages available and supported; tool features should be well documented; content should be in English; allow user to change criteria used to select a component in a given context;
GitHub Integration: create repositories, branches, forks, pull requests
Atom, VS Code and Cloud9 Integration: code pre-analysis to speed up contextualized snippets suggestion; search snippets per feature & component (best option and list); resolve component dependencies on selection; allow rating inside IDE
GitHub and Package Managers Integration: extract component reputation data
Google Integration: define search engine optimization strategy
Bower Integration: resolve components dependencies of code snippets
JsMeter, Cloc, Understand, Benchmark.js, Jslint and SlickSpeed Integration: use to run component evaluation metrics
Taxonomy: features, techniques (code snippets), components; package dependencies instead of receipts
Criteria: performance; constraints adherence (components/frameworks in use, architecture); code complexity (lines of code, cyclomatic complexity, maintainability index); vulnerability and conformance (critical and severity errors in component)
Use Cases: choose a list of components that matches a list of features before starting a new project; find a list of components that matches a specific feature and user project metadata; infer project constraints from code analysis to generate search metadata; find a list of components that matches your project metadata; find the fastest component for a feature disregarding project metadata; apply code snippet to an existing project and resolve dependencies automatically; find most popular components;

IV. CONCLUSION

This research aims to go beyond a new rational method for component selection, aggregating info on how this activity is done, a fundamental step to design a successful tool with this purpose [7] [10]. There are only a few studies focusing on gathering Industry feedback, which is one of the purposes of this study, and bring more evidences of real component selection practice. On internal and external validity [18], the 26 participants and 3 heuristic specialists formed a heterogeneous group of people directly or indirectly involved in software development, only 35% of them with a more active role in development community, in observation to 90-9-1 rule of [15] for online communities. We strictly followed rules described in [17] and [14] for conducting user observation and focus groups. Interaction between researcher and users were as low as possible, they had no previous contact with the prototype and questionnaires before the session and answered questions individually at the same time. Sessions were recorded, transcript and analyzed. During focus group, researcher read the questions, clarified that an agreement of the group was expected for every question, helped on doubts and controlled time available for each discussion.

Heuristic specialists had previous experience in heuristic evaluation and strictly followed Nielsen heuristics and severity ratings [18] [19]. They did not participate at the user observation and focus group sessions.

This study do not attempt to make universal generalizations, it is concerned with characterizing and aligning its own solution to the practice under the context of this study [16]. As evidenced in [10], evidence of real usage are the source of information that most help in the adoption of a JavaScript, CSS or HTML component, and this was proven to influence the result of this type of software.

Two perspectives were identified in component selection process, one when user is studying component options for his new project, more open to new options and the second one when user is in the middle of a project and needs a technical solution for a specific problem, looking for compatible components. This tool should address both use cases, supporting project bootstrap with best components available under initial project constraints, and suggesting the best component to solve technical problems or missing features in an existing code. The main differential identified on prototype was the capacity to run performance tests and rate snippets.

Performance and usability problems on the prototype disturbed user perception, but most participants think that, over time, when integrated to a version system and IDE, this tool can be used to map component patterns. The reward mechanism showed moderated relevance, which can be consequence of the limited usage period. Participants down voted the receipts feature. A more practical approach would be to rely on an existing package management tool. Users did not report major problems navigating in features, techniques and criteria, which suggests that the taxonomy defined was considered natural.

The results of this study are a stimulus to early user involvement on software projects, a key resource on the design phase, and the use of prototypes to help increasing the capacity to share the envision of features and requirements. Future studies will focus on applying the fundamentals gathered in this study to create a new prototype that will be validated for a longer time (some months). The focus will be on IDE integration and code analysis with metadata generation to provide contextualized search results.

Acknowledgment

Special thanks to professor Gordana Manic, PhD. that conducted the initial research and providing important information. We thank FAPESP (São Paulo Research Foundation) for financial support.

References

[1] W3C. "Open Web Platform Milestone Achieved with HTML5 Recommendation," in *http://www.w3.org/2014/10/html5-rec.html.en*, October, 2014.

[2] Ayala, C.; Hauge, Ø.; Conradi, R.; Franch, X.; Li, J. "Selection of third party software in Off-The-Shelf-based software development—An interview study with industrial practitioners," in *Journal of Systems and Software*, vol. 84, 4 ed., pp. 620-637, Apr. 2010 https://doi.org/10.1016/j.jss.2010.10.019.

[3] Delfim, F.; Paixão, K. V. R.; Cassou, D.; Maia, M. A. "Redocumenting APIs with crowd knowledge: a coverage analysis based on question types," in *Journal of the Brazilian Computer Society*, vol. 22:9, 1 ed., Dec. 2016 https://doi.org/10.1186/s13173-016-0049-0.

[4] Osmani, A. "Yet Another Framework Syndrome (YAFS)," in *https://medium.com/tastejs-blog/yet-another-framework-syndrome-yafs-cf5f694ee070*, Jan. 2015.

[5] Petersen, K.; Feldt, R.; Mujtaba, S.; Mattsson, M. "Systematic mapping studies in software engineering," in *Proc. of the 12th Intl. Conference on Evaluation and Assessment in Software Engineering*, Swinton, United Kingdom, pp. 68-77, Jun. 2008.

[6] Li, J.; Conradi, R.; Slyngstad, O.P.N.; Bunse, C.; Torchiano, M.; Moriso, M. "An empirical study on decision making in off-the-shelf component-based development," in *Proc. of the 28th Intl. Conference on Software engineering*, Shangai, China, pp. 897-900, May 2006 https://doi.org/10.1145/1134285.1134446.

[7] Hauge, Ø.; Østerlie, T.; Sørensen, C.-F.; Gerea, M. "An Empirical Study on Selection of Open Source Software – Preliminary Results," in *ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development*, Vancouver, British Columbia, Canada, pp. 42-47, May 2009 https://doi.org/10.1109/floss.2009.5071359.

[8] Damsgaard, J.; Karlsbjerg, J. "Seven Principles for Selecting Software Packages," in *Communications of the ACM*, vol. 53, 8 ed., pp. 63-71, Aug. 2010 https://doi.org/10.1145/1787234.1787252.

[9] Tarawneh, F.; Baharom, F.; Yahaya, J.H.; Zainol, A. "COTS Software Evaluation and Selection: a pilot Study Based in Jordan Firms," in *Int. Conf. on Electrical Engineering and Informatics*, Bandung, Indonesia, pp. 1-5, Jul. 2011 https://doi.org/10.1109/iceei.2011.6021821.

[10] Ayala, C.; Hauge, Ø.; Conradi, R.; Franch, X.; Li, J. "Selection of third party software in Off-The-Shelf-based software development—An interview study with industrial practitioners," in *Journal of Systems and Software*, vol. 84, 4 ed., pp. 620-637, Apr. 2010 https://doi.org/10.1016/j.jss.2010.10.019.

[11] Weibing C.; Jingyue, L.; Jianqiang, M.; Reidar, C.; Junzhong, J.; Chunnian, L. "A Survey of Software Development with Open Source Components in Chinese Software Industry," in *Software Process Dynamics and Agility*, Minneapolis, USA, pp. 208-220, May 19-20 2007 https://doi.org/10.1007/978-3-540-72426-1_18.

[12] Li, J.; Torchiano, M.; Conradi, R.; Slyngstad, O. P. N.; Bunse, C. "A State-of-the-Practice Survey of Off-the-Shelf Component-Based Development Processes," *in Reuse of Off-the-Shelf Components. Lecture Notes in Computer Science*, vol. 4039, pp. 16-28, Springer, Berlin, Heidelberg, 2006 https://doi.org/10.1007/11763864_2.

[13] Teixeira, L.; Saavedra, V.; Ferreira, C.; Santos, B.S. "Using Participatory Design in a Health Information System," in *Proc. of IEEE Annual Int. Conference of Engineering in Medicine and Biology Society*, Boston, Massachusets, EUA, pp. 5339-5342, Ago./Set. 2011 https://doi.org/10.1109/IEMBS.2011.6091321.

[14] Morgan, D. "Focus group as qualitative research," in *Qualitative Research Methods Series*, Sage Publications, London, England, vol.16, 2 ed., Out. 1996 http://dx.doi.org/10.4135/9781412984287.

[15] Nielsen, J. "The 90-9-1 Rule for Participation Inequality in Social Media and Online Communities," in http://www.nngroup.com/articles/participation-inequality/, Oct. 2006.

[16] Robson, C., "Real World Research: A Resource for Social Scientists and Practitioner-researchers," 2nd ed., Blackwell Publishers Inc., 2002.

[17] Stone D., Jarrett C., Woodroffe M., Minocha S. "User Interface Design and Evaluation," Morgan Kaufmann, pp. 29-37, Apr. 2005.

[18] Nielsen, J. "Severity Ratings for Usability Problems," in https://www.nngroup.com/articles/how-to-rate-the-severity-of-usability-problems/, Jan. 1995.

[19] Nielsen, J. "10 Usability Heuristics for User Interface Design," in https://www.nngroup.com/articles/ten-usability-heuristics/, Jan. 1995.

[20] Käpyaho, M.; Kauppinen, M. "Agile Requirements Engineering with Prototyping: A Case Study," IEEE 23rd Intl. Requirements Engineering Conference (RE), Ottawa, Ontario, Canada, pp. 334-343, Ago. 2015 https://doi.org/10.1109/re.2015.7320450.

[21] Gizas, A.B.; Christodoulou, S. P.; Papatheodoru, T.S. "Comparative evaluation of JavaScript frameworks," in *Proc. of the 21st Intl. Conference Companion on World Wide Web*, Lyon, França, pp. 513–514, Apr. 2012 https://doi.org/10.1145/2187980.2188103.

[22] Graziotin, D.; Abrahamsson, P. "Making Sense Out of a Jungle of JavaScript Frameworks – Towards a Practitioner-Friendly Comparative Analysis," in *Proc. of the 14th Intl. Conference on Product-Focused Software Process Improvement*, Pafos, Chipre, pp. 334-337, Jun. 2013 https://doi.org/10.1007/978-3-642-39259-7_28.

[23] Land, R.; Blankers, L.; Chaudron, M.; Crnković, I. "COTS Selection Best Practices in Literature and in Industry," in *Proc. of the 10th Intl. Conference on Software Reuse* , Beijing, China, pp. 100-111, May. 2008 https://doi.org/10.1007/978-3-540-68073-4_9.