Usability of Educational Technology APIs: Findings and Guidelines

Evangelos Kapros(✉) and Neil Peirce

Unit 28, Learnovate Centre, Trinity College Dublin,
Trinity Technology and Enterprise Campus, Pearse Street, Dublin 2, Ireland
{evangelos.kapros,neil.peirce}@scss.tcd.ie
http://www.learnovatecentre.org

Abstract. This paper describes a project that reviewed the usability of existing Educational Technology Application Programming Interfaces (EdTech APIs). The focus was on web-based APIs and the portals through which these are offered to developers. After analysing the state of art with regard to existing EdTech APIs and after conducting a literature review on API usability, a survey was circulated among developers and CTOs of EdTech organisations. The results of the aforementioned three steps were triangulated and resulted in usability guidelines for EdTech APIs. The contribution of this project is twofold: firstly, the production of a concrete set of EdTech API usability guidelines and, secondly, their implementation in a proof-of-concept a portal for two different EdTech offerings.

Keywords: Usability · API · Programming

1 Introduction

Application Programming Interfaces (APIs) are an important component in software development. They allow for code modularity, reuse, and separation of concerns. With the advancement of web-based applications, web-based APIs have an even more prominent role, and have even been described as "the glue of the web". While traditionally API usability has been studied in the context of development environments (IDEs) for desktop computers, research on web-based API usability is, to date, rather scarce. Moreover, APIs for Educational Technology (EdTech APIs) are even less often the subject of research with regard to their usability.

Web-based APIs are different to traditional APIs, not necessarily in structure, but primarily in the way they are offered. Dedicated portals are built to offer access to and document an API for developers. Online documentation and offering of APIs present unique usability challenges.

In addition, EdTech APIs present challenges specific to the domain of Education. For example, the variable utilisation of the hosting servers of an API is different in the case of EdTech as typically peak load will occur during a brief

P. Zaphiris and A. Ioannou (Eds.): LCT 2015, LNCS 9192, pp. 249–260, 2015.
DOI: 10.1007/978-3-319-20609-7_24

period of time, such as an examinations period. Other challenges include the increased need for transparency and data privacy. Another important aspect is that educational software applications are usually being developed by a combination of software *and* content developers; the case of subject-matter experts participating in the development process is not common in many fields, but in education content developers have to either prototype an application of verify its pedagogical appropriateness.

For the aforementioned reasons, it is necessary to treat EdTech API usability specially, as the domain-specific usability issues in a web context call for separate investigation.

2 EdTech APIs

In the Learnovate Centre, we partner with around 50 well-established EdTech organisations[1]. With regard to APIs that are EdTech domain-specific, we have identified the following recurring issues:

High scalability for short periods of time. Some of our partners need to scale their API usage for a brief period of time, usually during state exams. In some occasions the server utilisation has reportedly reached $200 * n$ of the required number of servers n for a period of only two days, while the utilisation during the rest of the year was ranging from n to $2 * n$.

Geo-location of cloud servers. Other Learnovate partners have a legal obligation to locate student data within the country they reside, which is not always possible with large sparsely located cloud server farms.

IP Protection. Other requirements concerning Intellectual Property and Data Protection may consist of a combination of institutional policies and legal obligations. Example of such requirements are the following:

— Universities require restriction on cloud usage.
— Requirements for ISO 27001 and other security certification.
— U.S. Safe Harbor agreement.

Overall, the aforementioned requirements consist barriers to API adoption. Moreover, the often requirement of a combination of two or all three of the above in EdTech enhances the domain-specific barriers to EdTech web-based API adoption.

3 Traditional API Usability

This section presents previous work around the usability of APIs. Since web-based APIs are a more recent advancement, most previous work assumes that developers have typically downloaded a Software Development Kit (SDK) and are using an IDE to write their code. Thus, this section describes usability issues around the use and documentation of APIs in IDE environments and not in interactive web consoles. Interactive web consoles will be discussed in the following section.

[1] http://www.learnovatecentre.org/membership/our-members/.

3.1 Identifying Usability Issues

Many different approaches have been taken in the study of API usability. One study [28] has correlated posts on bug-tracking systems with different API usability factors; the percentage of posts for each factor is presented below:

1. Missing feature — 43.5 %
2. Correctness — 31.1 %
3. Documentation — 27.3 %
4. Exposure of elements — 10.3 %
5. Memory management <9 %
6. Function parameter and return <9 %
7. Technical mismatch <8 %
8. . . . similar smaller issues include Naming, Callers perspective, constructor parameter.

While the percentage of each category of bugs does not necessarily imply its importance, it is shown sufficiently that API usability affects the final software product. Thus, it is of concern to the developer *and* the end user alike. Apart from requests for missing features, the high percentages of bugs around the correctness and the documentation of an API show that features that are poorly communicated are causing poor development despite their usefulness.

A different approach exists in [11]. The team of this research project conducted interviews with developers about the barriers to adopting an API, in order to identify any usability issues that may occur. Since *reported* usability issues may differ from *actual* ones, workshops were conducted with developers in order to validate the reported issues while developing a piece of software. Below is an ordered list with encountered usability problems:

1. Documentation
2. Conceptual correctness
3. Callers perspective
4. Complexity
5. Data types
6. Leftovers for client code
7. Error handling/Exceptions
8. Consistency and conventions
9. Method parameters and return type
10. Factory pattern.

While results are similar in both the case of workshops and interviews, some differences exist. Specifically, interviews also mentioned Naming, but it was not encountered in workshops.

Another finding was that consistent error messages across various error types can be beneficial for developers and enhance greatly the usability of an API. Eight error types have been grouped and identified in [1,2], e.g. Structure error, Consistency error, etc.

Apart from identifying API usability factors, previous work has also investigated ways to improve the usability of APIs, and is presented below.

3.2 Improving API Usability

While the best way to design a usable API is to produce and follow guidelines that ensure its usability [12][2], it is also important to provide to developers the appropriate tools that will help them understand and use an API.

A number of API classes and methods usability factors, which focus on API improvement, has been identified in [19] and is listed below.

- High number of classes is negative, re-structuring in sub-packages is beneficial
- Constructors are easier than factory methods
- Distinctive names should be used
- Parameters per method should be as low as possible
- Knowledge of domain and programming exp equally useful
- Few concepts (classes/methods) lead to high learnability.

One proposed way to improve API usability is to evaluate it automatically using Complexity Metrics and Visualizations [7]. To achieve this, a tool called `Metrix` has been built in order to evaluate Bandi *et al.* metrics, like the sum of Bandis complexity metrics for a class and the number of classes with a particular complexity.

Other approaches use text analytics to improve API usage within IDEs; for example, `Jadeite` uses a wordcloud-like experience to improve documentation by showing which calls are more frequent [22,25]. This helps identify calls that are otherwise similar. Consider for example the call `/system/student` and the call `/system/class/student`: it is highly probable that a new developer would want to use the one that has already been used the most by other developers.

Finally, other ways to make APIs more developer-friendly are colour coding, tooltips in code editors/IDEs [8,20], or diagrams [27] that help developers understand the mental model of the API.

However, while these tools improve greatly the usability of APIs that are typically part of an SDK (so that metadata to produce tooltips or diagrams are readily available), web APIs have additional challenges that need to be addressed.

4 Findings

4.1 API Usability Survey

Literature Review Findings. The table below presents a triangulated list of usability factors, compiled by combining the factors listed in the literature of the previous section. We anticipate that these issues are not specific to EdTech. Documentation, correctness, and low complexity, are the three factors that were persistent throughout (Table 1).

[2] For example, see: https://github.com/WhiteHouse/api-standards and http://guides.rubyonrails.org/api_documentation_guidelines.html.

Table 1. This table shows the difference between the *perceived* API usability issues, as described in interviews, and the *actual* ones, identified in a workshop.

Workshops	Interviews
1. Documentation	1. Documentation
2. Correctness	2. Naming
3. Complexity	3. Correctness
4. Data types	4. Complexity
5. Error handling	5. Data types
6. Factory pattern/Constructor	6. Leftovers for client code
7. . . . Others, encountered <5 % of the time	

Developer Survey Results. In order to specify EdTech-specific usability issues and factors, we conducted a survey across CTOs and developers of our industry partners ($n = 11, k = 49$). The ease of integration into an organisation's existing technology stack, the API performance, and its availability, were the technical factors that were described as essential (see Fig. 1). Concerning non-technical factors (see Fig. 2), as most important were described the API meeting the organisation's needs, transparency as *how data are stored and processed*, and the *data privacy policy* of the service.

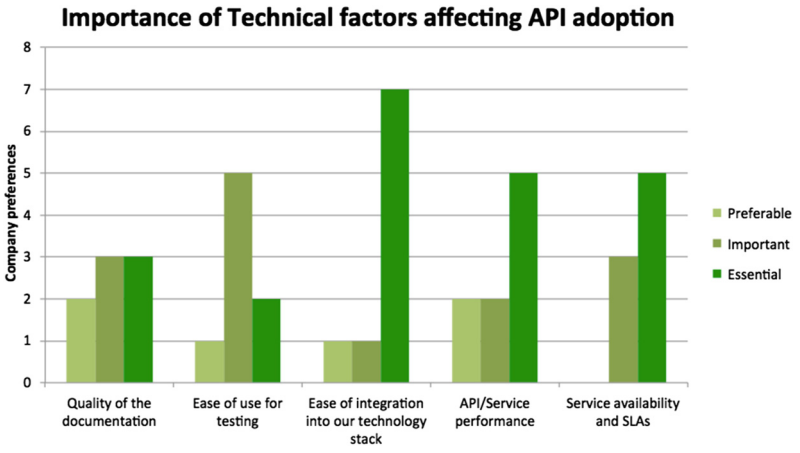

Fig. 1. The importance of technical factors that affect the adoption of APIs according to replies from CTOs and developers of EdTech industry partners of Learnovate. The survey was sent to Learnovate's k partners, and n replied ($n = 11, k = 49$).

From the results of this survey, we conclude that policy issues around Data Protection are a *sine qua non* for EdTech API adoption. As such, these policy issues should be considered as tightly coupled with API usability and

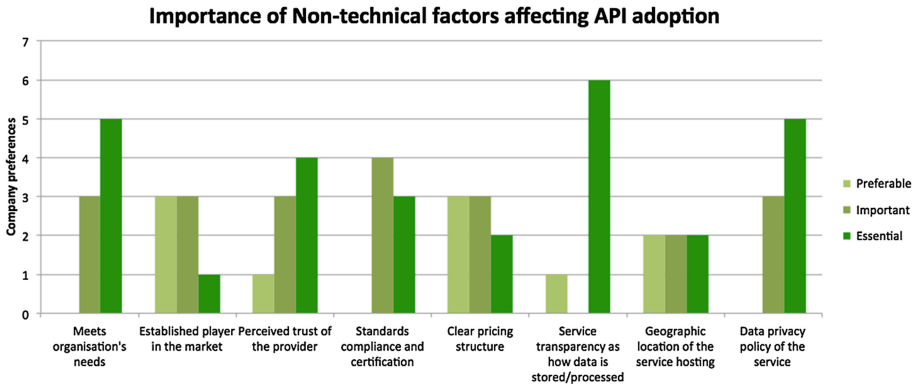

Fig. 2. The importance of non-technical factors that affect the adoption of APIs according to replies from CTOs and developers of EdTech industry partners of Learnovate. The survey was sent to Learnovate's k partners, and n replied ($n = 11, k = 49$). Policy issues around data protection are especially important in Educational Technology.

investigated under this prism during the design and development of an API. In addition, it is important that portals offering EdTech APIs are transparent in describing these policies—this and other portal-related usability issues are described in the following section.

4.2 EdTech Portal Survey

The adoption and use of new software can be positively influenced through early experimentation with the software, the internalising of structures and cultures surrounding its use, and progress validation as it is being used [5]. In scenarios where the software is not already in use within an organisation the additional factors of software discoverability and clarity in software capabilities need to be addressed. In the case of web-based APIs these challenges are addressed in part through discovery services such as the API Directory provided by `ProgrammableWeb`[3]. However, increasingly there is a growth in API or developer portals to enhance the enrolment and on-boarding of developers.

Although API portals do exist for educational technology it was found that the majority of them were either simple documentation pages (`Engrade`[4], `Khan Academy`[5]) or were only open to select partners and affiliates (`TurnItIn`[6], `Knewton`[7]). In light of this in order to understand common trends across API portals 18 portals were identified through web searches that offered web-based APIs (Table 2).

[3] http://www.programmableweb.com/apis/directory.
[4] https://wikis.engrade.com/engradeapi.
[5] http://api-explorer.khanacademy.org/.
[6] http://turnitin.com/en_us/integrations/overview.
[7] http://www.knewton.com/partners/.

Table 2. API Portals Surveyed.

Organisation	Industry
`Aylien`	Software/Text analysis
`Berkeley`	Education
`Clever`	Education
`Edgar Online`	Finance
`Klout`	Social media
`Marvel`	Entertainment
`Mendeley`	Education
`MusicGraph`	Entertainment
`Nike+`	Fitness/Clothing
`Overdrive`	Media/Publishing
`Pearson`	Education
`Rovi`	Software/Entertainment
`Rubix`	Software/Computer vision
`Tomtom`	Automotive/Mapping
`Transport for London`	Transport
`USA Today`	Media/News
`Yellow Pages` (Canada)	Advertising/Directory Services
`Yummly`	Nutrition

Within these portals we assessed how they facilitated early experimentation and the internalisation of structures. The assessment of progress validation has more relevance within large software teams where mentors with prior experience exist [5]. Its consideration in the context of API portals is a future direction for this research.

Early experimentation with web-based APIs requires readily available access to the functioning APIs and access to documentation. The following factors were identified as potential barriers to early experimentation:

- Access to documentation without signup
- Complexity of the signup process
- Availability of a cost-free pricing tier
- Availability of interactive documentation.

Of the portals surveyed all provided access to API documentation allowing developers to begin planning integrations without committing to a particular API. Although documentation goes part of the way to enable experimentation it still relies on the developer technically integrating with the API before realistic queries can be made and responses received. In order for this to happen in all cases except one (`Clever`) the developer has to signup to the portal in question.

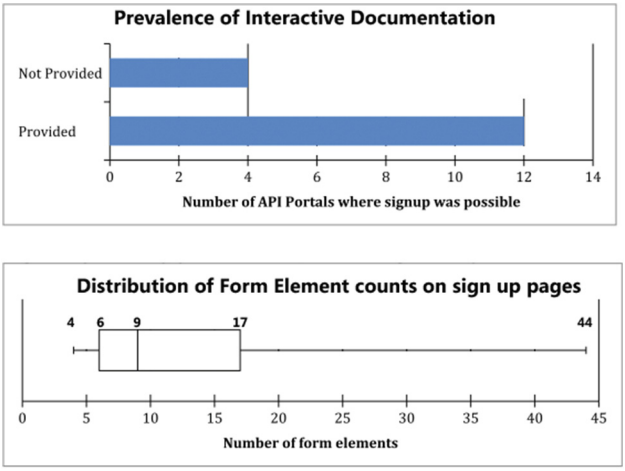

Fig. 3. Usability metrics of the surveyed API portals. Above, interactive documentation is increasingly common. Below, the number of required fields to sign up varies greatly; complexity can be a barrier to adoption.

The use of interactive documentation is increasingly common in API portals (see Fig. 3). Such documentation allows developers to make live API calls from within the portal where the submitted fields and responses are documented. By allowing developers to test real world queries to an API without the need to write any code, early experimentation is encouraged through interactive documentation. Examples of common interactive API documentation are `Swagger UI`, `Mashery IO Docs`, `3Scale Active Docs`, and `Mulesoft RAML API Console`.

The complexity of the signup process is seen as a barrier to experimentation and it is desirable to simplify the process. As a simple indicator, the complexity of signup was determined by the number of form elements (e.g. text fields, check boxes, drop downs, etc.) that had to be read and completed (see Fig. 3). The median number of form elements was nine with the minimum being four as required by `Mendeley` (first name, last name, email, password).

Four portals also provided `OAuth` sign-in for using other services, removing the need to complete a registration form and further easing signup. However, only one portal (`Marvel`) offered several service logins (`Facebook`, `Google`, `Yahoo`, `Twitter`, `Marvel`).

Two of the portals did not allow online signup as they only provided access to approved partners or affiliates (`Nike+`, `Yellow Pages` (Canada)).

5 EdTech API Usability Requirements and Guidelines

As an output of our research we produced the following guidelines, and designed a portal which incorporates these guidelines.

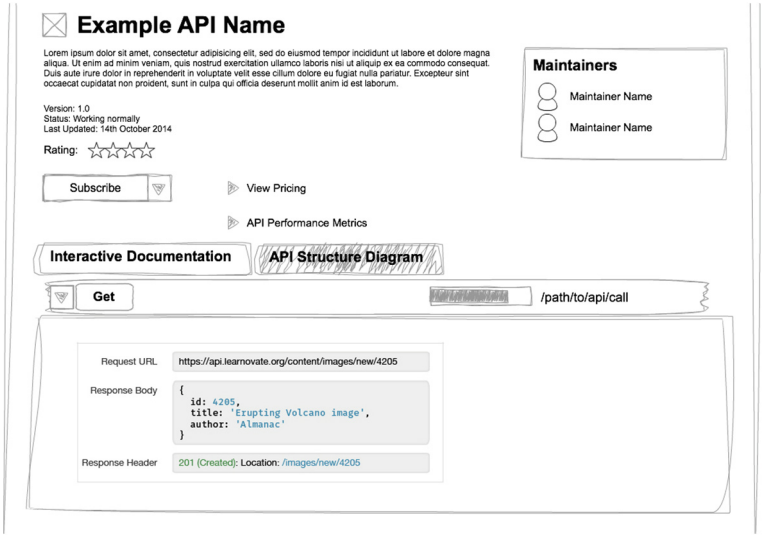

Fig. 4. (a) Colour-coding the structure of a call at an interactive console can improve the legibility of the documentation and, thus, the learnability of an API. (b) A bar that visualises the amount of calls on that method helps distinguish it from similar ones (Color figure online).

API Packaging Related. These guidelines are related to API Packaging. API designers and developers are encouraged to use them; please note that guidelines marked with an asterisk {∗} are EdTech domain specific (Fig. 4).

- * Domain knowledge balances lack of programming experience
- * Data Processing should be transparent
- APIs should be generic enough, that is decoupled from specific applications
- Sub-packaging is better than large categories
- Fewer method parameters is better
- Categorise error messages consistently.

API Portal Related. Other usability aspects that affect API usage and are related to the portal that offers the API are listed below. As above, guidelines marked with an asterisk {∗} are EdTech domain specific (Fig. 5).

- * Explain the API's Data Protection policy (or how it may facilitate an application's policy)
- Provide an interactive console documentation system
- Visualise the popularity of a method call
- Colour code to improve learnability
- Represent visually the API's structure.

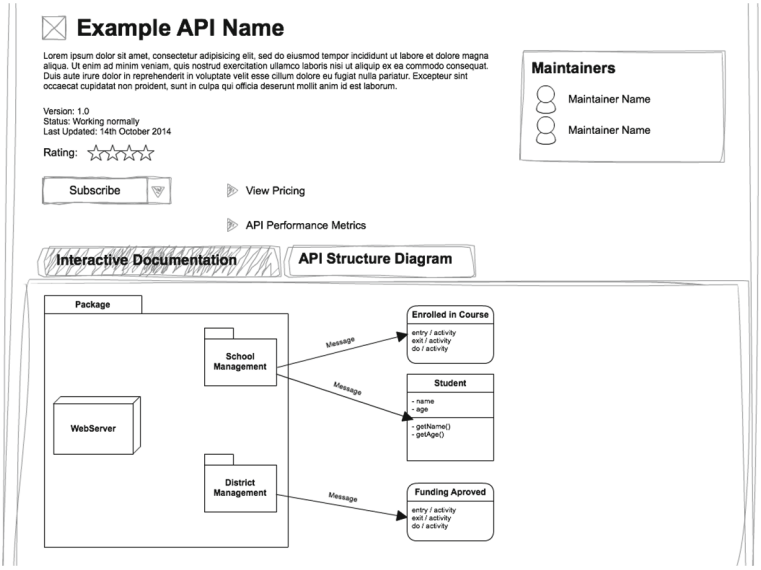

Fig. 5. A diagram that represents the structure of an API helps developers understand its mental model. Visual learners benefit especially from this representation, as API documentation is typically in text form.

6 Conclusions

In conclusion, after reviewing the API usability literature, surveying EdTech CTOs and developers, and surveying API portals, we created a concrete set of eleven guidelines that facilitate usable APIs for EdTech applications. While most of the guidelines hold for each and every API, some are specific to the EdTech field.

Moreover, we have designed and implemented a portal that offers EdTech APIs and implements the above guidelines. However, other aspects of the portal are still under development, thus public access is still not available (contact us for details).

Acknowledgments. This research is supported by the Learnovate Centre at Trinity College, the University of Dublin. The Learnovate Centre is funded under the Technology Centre Programme through the Government of Ireland's state agencies Enterprise Ireland and IDA Ireland.

References

1. Beaton, J.K., Myers, B.A., Stylos, J., Jeong, S.Y., Xie, Y.: Usability evaluation for enterprise SOA APIs. In: International Workshop on Systems Development in SOA Environments, pp. 29–34. ACM Press, New York (2008)

2. Beaton, J., Jeong, S.Y., Xie, Y., Stylos, J., Myers, B.A.: Usability challenges for enterprise service-oriented architecture APIs. In: Proceedings - 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, pp. 193–196. IEEE (2008)
3. Burns, C., Ferreira, J., Hellmann, T.D., Maurer, F.: Usable results from the field of API usability: a systematic mapping and further analysis. In: Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC, pp. 179–182. IEEE (2012)
4. Clarke, S.: Measuring API usability. Dr. Dobb's J. Windows/.NET Suppl. **9**(5), S6–S9 (2004)
5. Dagenais, B., Ossher, H., Bellamy, R.K.E., Robillard, M.P., de Vries, J.P.: Moving into a new software project landscape. In: Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE 10, vol. 1, p. 275. ACM Press (2010)
6. Daughtry, J.M., Farooq, U., Myers, B.A., Stylos, J.: API usability: report on special interest group at CHI. SIGSOFT Softw. Eng. Notes **34**, 27–29 (2009)
7. De Souza, C.R.B., Bentolila, D.L.M.: Automatic evaluation of API usability using complexity metrics and visualizations. In: 31st International Conference on Software Engineering - Companion Volume, ICSE 2009, pp. 299–302. IEEE (2009)
8. Dekel, U., Herbsleb, J.D.: Improving API documentation usability with knowledge pushing. In: Proceedings - International Conference on Software Engineering, pp. 320–330. IEEE (2009)
9. Ellis, B., Stylos, J., Myers, B.: The factory pattern in API design: a usability evaluation. In: Proceedings - International Conference on Software Engineering, pp. 302–311. IEEE (2007)
10. Farooq, U., Zirkler, D.: API peer reviews: a method for evaluating usability of application programming interfaces. In: Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work, pp. 207–210. ACM Press, New York (2010)
11. Grill, T., Polacek, O., Tscheligi, M.: Methods towards API usability: a structural analysis of usability problem categories. In: Winckler, M., Forbrig, P., Bernhaupt, R. (eds.) HCSE 2012. LNCS, vol. 7623, pp. 164–180. Springer, Heidelberg (2012)
12. Henning, M.: API design matters. Commun. ACM. **52**(5), 46–56 (2009)
13. Ko, A.J., Riche, Y.: The role of conceptual knowledge in API usability. In: Proceedings - 2011 IEEE Symposium on Visual Languages and Human Centric Computing, VL/HCC 2011, pp. 173–176. IEEE (2011)
14. Martens, A.: Usability of web services. In: 2003 Proceedings of Fourth International Conference on Web Information Systems Engineering Workshops, pp. 182–190 (2003)
15. Nelson, A.J., Dinolt, G.W., Michael, J.B., Shing, M.T.: A security and usability perspective of cloud file systems. In: Proceedings of 2011 6th International Conference on System of Systems Engineering: SoSE in Cloud Computing, Smart Grid, and Cyber Security, SoSE 2011. pp. 161–166. IEEE (2011)
16. Plunkett, L., Solow-Niederman, A., Gasser, U.: Framing the Law & Policy Picture: A Snapshot of K-12 Cloud-Based Ed Tech & Student Privacy in Early 2014 (2014)
17. Potter, T.C.: An evaluation methodology for the usability and security of cloud-based file sharing technologies (2012)
18. Robillard, M.P., Deline, R.: A field study of API learning obstacles. Empir. Softw. Eng. **16**, 703–732 (2011)

19. Scheller, T., Kuhn, E.: Influencing factors on the usability of API classes and methods. In: Proceedings - 2012 IEEE 19th International Conference and Workshops on Engineering of Computer-Based Systems, ECBS 2012, pp. 232–241. IEEE (2012)
20. Stylos, J.: Making APIs more usable with improved API designs, documentation and tools (2009)
21. Stylos, J., Clarke, S.: Usability implications of requiring parameters in objects constructors. In: Proceedings - International Conference on Software Engineering, pp. 529–538. IEEE (2007)
22. Stylos, J., Faulring, A., Yang, Z., Myers, B.A.: Improving API documentation using API usage information. In: 2009 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2009, pp. 119–126. IEEE (2009)
23. Stylos, J., Graf, B., Busse, D.K., Ziegler, C., Ehret, R., Karstens, J.: A case study of API redesign for improved usability. In: Proceedings - 2008 IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC 2008, pp. 189–192. IEEE (2008)
24. Stylos, J., Myers, B.: Mapping the space of API design decisions (2007). http://repository.cmu.edu/hcii/169
25. Watson, R.B.: Improving software API usability through text analysis: a case study. In: 2009 IEEE International Professional Communication Conference, pp. 1–7. IEEE (2009)
26. Winckler, M., Forbrig, P., Bernhaupt, R. (eds.): Human-Centered Software Engineering. Springer, Heidelberg (2012)
27. Xiong, P.X., Fan, Y.F., Zhou, M.Z.M.: A petri net approach to analysis and composition of web services. IEEE Trans. Syst. Man Cybern. A Syst. Humans **40**, 376–387 (2010)
28. Zibran, M.F., Eishita, F.Z., Roy, C.K.: Useful, but usable? Factors affecting the usability of APIs. In: Proceedings - Working Conference on Reverse Engineering, WCRE, pp. 151–155. IEEE (2011)