Poster: Towards a Formal API Assessment

Amir Zghidi
CES Lab
Sfax, Tunisia
zghidi.amir@gmail.com

Imed Hammouda
Mediterranean Institute of Technology, South Mediterranean University, Tunisia
Chalmers and University of Gothenburg, Sweden

Brahim Hnich
CES Lab
Sfax, Tunisia
hnich.brahim@gmail.com

ACM Reference Format:
Amir Zghidi, Imed Hammouda, and Brahim Hnich. 2018. Poster: Towards a Formal API Assessment. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, Article 4, 2 pages. https://doi.org/10.1145/3183440.3195026

1 INTRODUCTION

Assessing the quality of an API is important in many different aspects: First, it can assist developers in deciding which API to use when they are faced with a list of potential APIs to choose from, by comparing the benefits and drawbacks of each option [1]; we refer to this as the *API selection problem*. Second, it can help guide the design process and expose problem areas in early stages of API design, even before implementing the actual API [2]; we refer to this as the *API design problem*. In order to assess the quality of an API, various evaluation methods have been used: some are based on empirical laboratory studies, gathering feedback from API users; others are based on inspection methods where experts evaluate the quality of an API based on a list of design guidelines [3][4] such as Nielsen's heuristics and the cognitive dimensions framework [2][5]. In this paper, we are particularly interested in extending Steven Clarke's approach of measuring API usability based on the cognitive dimensions framework [5]. The usability of an API is assessed by comparing the API (what it actually offers) with the profiles of its potential users (what they expect out of it).

2 API ASSESSMENT FACTORS

When assessing an API, we plan to incorporate several new factors as discussed below:

2.1 Fitness Dimensions

As a result of an empirical study conducted [6], we obtained a list of 23 fitness dimensions which included all the factors defined in the cognitive dimensions framework in addition to other new factors such as API latency, synchrony, compatibility, model complexity, evolvability, testability, etc.

2.2 Classification of Fitness Dimensions

When assessing an API, we find it important to distinguish between technical and cognitive fitness dimensions. Technical dimensions are those that do not depend on the user's persona. Cognitive dimensions, however, are dependent on the user's persona. The importance of such classification rises from the fact that when dealing with cognitive dimensions, what makes an API good for one stakeholder would probably make it less appealing from another stakeholder's perspective. However, for technical fitness dimensions, regardless of who the stakeholders are, they should have similar preferences.

2.3 Priorities of Fitness Dimensions

When assessing an API, not all fitness dimensions are equally important to a given stakeholder. A study we conducted indicated that, on a scale of 1 to 5 (1 being extremely unimportant), some fitness dimensions such as *easiness of use* and *consistency* scored 5. However, other dimensions such as *atomic setting* and *evolvability* were of less importance and only scored 2 [7]. We extend Steven Clarke's API assessment framework by adding the notion of priorities or weights to fitness dimensions and therefore introduce a way to specify which dimensions have higher influence in assessing an API.

2.4 Importance of stakeholders

API stakeholders can be broadly classified into three main groups: designers or architects (those who build the API); API users or consumers (software developers working with the API); product consumers (consumers of software products that have been developed using the API) [2]. Unfortunately, it is not always feasible to satisfy the needs of all API stakeholders because they have different and possibly conflicting preferences. One possible solution for API designers is to "aim to displease everyone equally"[8] as suggested by Joshua Bloch. Another solution, which we advocate, is to prioritize users based on their weights; i.e: their importance to API owners.

3 FORMAL API ASSESSMENT MODEL

3.1 Definitions

When assessing an API, an expert would have to assign for each fitness dimension d_i a value v_i defined on a measurement scale $E_i \subset \mathbb{N}$.

By assigning values to each fitness dimension, we define the *API's profile* which can be formally presented as a vector of n integers; where n is the number of fitness dimensions, and the i^{th} member of the vector represents the API's evaluation score with respect to

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195026

dimension d_i.

$$API\ \ Profile = (v_1, v_2, .., v_n); \forall i \in [1..n]\ \ v_i \in E_i$$

Similarly, a stakeholder's profile is defined in the same manner, except that not all dimensions have to be evaluated. If a stakeholder does not care about a given dimension, he would indicate so by assigning an asterisk ($*$) to such dimension. The profile of stakeholder a can be formally defined as follows:

$$Profile_a = (v_{a_1}, v_{a_2}, .., v_{a_n}); \forall i \in [1..n]\ \ v_i \in E_i \cup \{*\}$$

Since, for a given stakeholder a, different fitness dimensions do not always have the same importance, we propose to use a *weight* value wd_{a_i} to represent the weight or relevance of fitness dimension d_i to stakeholder a. To model the priority of fitness dimensions, we define, for each stakeholder a, a dimension weight vector Wd_a as follows, where the sum of all weights is equal to one:

$$Wd_a = (wd_{a_1}, wd_{a_2}, .., wd_{a_n});\ \ \forall i \in [1..n]\ \ wd_{a_i} \in [0, 1]$$

Similarly, we propose to use a *weight* value ws_a to represent the importance of stakeholder a to API owners. Let $m \in \mathbb{N}$ be the number of stakeholders. Each stakeholder is assigned a weight value ws between zero and one. The sum of weights of all stakeholders is equal to one.

3.2 Problem Formulation

In this section, we try to solve the API selection problem. We do this by modeling our problem as a Multi Criteria Decision Making (MCDM) problem [9], where fitness dimensions represent the criteria based on which a choice has to be made, and different APIs represent possible alternatives.

Let $U = \{a, b, c, ...\}$ be a non empty set of stakeholders; and let $A = \{\alpha, \beta, \gamma, ...\}$ be a finite non empty set of alternatives (possible APIs that we need to choose from; or candidate API profiles that we are going to develop). As a first step to finding a solution to the *API selection problem*, we assume that we are working with a single stakeholder.

Let $S^{\alpha}_{a_i}$ be the satisfaction value of stakeholder a with respect to fitness dimension d_i of alternative α (API_α).

A value between 0 (completely unsatisfied) and 1 (fully satisfied) should be assigned to S_{a_i}. Such value could be calculated in the following manner.

For cognitive fitness dimension we have $S^{\alpha}_{a_i} = 1 - \frac{|v_i - v_{a_i}|}{max_i - min_i}$.

For technical fitness dimension we have set $S^{\alpha}_{a_i}$ to 1 if $v_i \geqslant v_{a_i}$, else to $1 - \frac{v_{a_i} - v_i}{max_i - min_i}$ Note that when $S^{\alpha}_{a_i} = *$, v_{a_i} is set to zero.

Once the partial scores indicating the stakeholder's satisfaction degrees with respect to different fitness dimensions are calculated, we proceed to calculating the global score indicating the global satisfaction degree of a given stakeholder regarding the examined API. This can be obtained by summing up the products of partial satisfaction degree and the corresponding fitness dimension's priority. Let D^{α}_{a} be the set of fitness dimensions related to API_α that $stakeholder_a$ cares about. $|D^{\alpha}_{a}|$ denotes the cardinality of the set (the number of its elements). Let GS^{α}_{a} be the global evaluation score of API_α by $Stakeholder_a$; it can be formally expressed as follows:

$$GS^{\alpha}_{a} = \sum_{\substack{i=0 \\ d_i \in D^{\alpha}_{a}}}^{n} S^{\alpha}_{a_i} \times pr^{\alpha}_{a_i} \quad (1)$$

Once the global scores of all possible alternatives are computed, they can be used to rank the alternatives and choose the one that best meets the stakeholder's needs.

So far, we have found a way to select, from a set of APIs, the one that best meets the needs of a given stakeholder. However, it is uncommon to find multiple stakeholders with conflicting preferences interacting with the same API. In this case, we should use a second aggregation operation in order to come up with a final global score for each alternative taking into considerations the weights of stakeholders and the global scores of each alternative from the stakeholders perspective.

For each alternative, the final global score or weighted global score can be calculated by summing up the product of the global scores assessed by stakeholders and the corresponding stakeholders weights. The obtained number provides an indication of how well a given API meets the preferences of different stakeholders.

$$WGS^{\alpha} = \sum_{a \in U} GS^{\alpha}_{a} \times w_a \quad (2)$$

One possible way of selecting the API that "best" satisfies the need of all interacting parties taking into considerations the importance of stakeholders is to select the API that has the highest weighted global score.

4 CONCLUSION

In this paper, we built on top of Clarke's API assessment framework and introduced additional factors that should be considered when deciding how well an API meets the expectations of its stakeholders. We developed a formal modal that can be used in assessing the design of an API by quantitatively comparing what the API is actually providing to what its stakeholders are expecting out of it, taking into considerations the weights of stakeholders and those of fitness dimensions.

REFERENCES

[1] Andrew J. Ko, B. A. Myers, and H. H. Aung. Six learning barriers in end-user programming systems. In *2004 IEEE Symposium on Visual Languages - Human Centric Computing*, pages 199–206, Sept 2004.

[2] Brad A. Myers and Jeffrey Stylos. Improving api usability. *Commun. ACM*, 59(6):62–69, May 2016.

[3] Thomas Grill, Ondrej Polacek, and Manfred Tscheligi. *Methods towards API Usability: A Structural Analysis of Usability Problem Categories*, pages 164–180. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[4] Umer Farooq and Dieter Zirkler. Api peer reviews: A method for evaluating usability of application programming interfaces. In *Proceedings of the 2010 ACM Conference on Computer Supported Cooperative Work*, CSCW '10, pages 207–210, New York, NY, USA, 2010. ACM.

[5] Steven Clarke. Measuring api usability. *Dr.Dobb's*, 2004.

[6] E. Knauss and I. Hammouda. Eam: Ecosystemability assessment method. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 319–320, Aug 2014.

[7] A. Zghidi, I. Hammouda, B. Hnich, and E. Knauss. On the role of fitness dimensions in api design assessment - an empirical investigation. In *2017 IEEE/ACM 1st International Workshop on API Usage and Evolution (WAPI)*, pages 19–22, May 2017.

[8] Joshua Bloch. How to design a good api & why it matters, 2007.

[9] Michael D Constantin Z. *Multiple Criteria Decision Making*. Springer International Publishing, 2017.