2014 IEEE 38th Annual International Computers, Software and Applications Conference Workshops

An API Design Process in terms of usability

A case study on building more usable APIs for Smart TV Platform

Sunghoon Lee, Sanghee Lee, Sumi Lim,
Jiyoung Jung, Sangho Choi
CTO Software Capability Development Center
LG Electronics
Seoul, Korea
{sunghoon.lee, sanghee3.lee, sumi.lim,
jeeyoung.jung, sangho109.choi}@ lge.com

Neunghoe Kim, Jung-Been Lee
Department of Computer Science and Engineering
Korea University
Seoul, Korea
{nunghoi, jungbini}@korea.ac.kr

***Abstract*— Products are released based on various platforms. An Application programing interface (API) is important to develop platform based applications effectively. Previously, we had some difficulties in developing applications using our platform APIs. Their name was ambiguous, and their functions were not primitive, and even their documentation was not enough to refer to. Therefore, we had to maintain our platform APIs. In this paper, we propose an API design process and API evaluation guidelines. "API Analysts" elicit functions from requirement documents. "API designer" design APIs for the functions following the guidelines. "Technical writers" produce documentation for the APIs. "API reviewers" evaluate the APIs and API documentation conforming to the proposed guidelines. Proposed process made our platform APIs more convenient to use for developing applications.**

Keywords—applicaton programming interface; api; software; usability; requirements; process; platform product;

I. INTRODUCTION

Recently, products are released based on various platforms such as Android, iOS and webOS. Most Platform based applications depend on APIs [1]. Well-defined APIs provide consistent level of functions and forms. They are easy to use and hard to misuse. Documentation is also a part of them [2]. The APIs for a complex system are difficult to use without documentation [3].

Previously, we had some difficulties in using our platform APIs. An API provided various functions. There were many APIs providing similar functions. Documentation was hardly found nor updated. There were many defects caused by misuses of APIs. Therefore, we have to maintain our APIs to reduce misuses of them and make them more convenient to use for developing applications.

In our study, we propose an API design process. Firstly, "API analysts" analyze requirement documents to elicit minimal set of functions to provide. Secondly, "API designers" design APIs to fulfill functions using the guidelines that we proposed. Then, "Technical writers" document them. Finally, "API reviewers" evaluate the APIs whether they conform to the guidelines.

II. RELATED WORK

M. Henning, in [2], provides some guidelines for a good API design. After his more than 25 year experience as a software engineer, many problems in programming tasks have dealt with badly designed APIs. Once APIs had been designed and provided, their design flaws have a tendency to get magnified out of all propositions because they are called many times.

The guidelines that the author provides are as follows:

- An API must provide sufficient functionality for the caller to achieve its task.
- An API should be minimal, without imposing undue inconvenience on the caller.
- APIs cannot be designed without an understanding of their context.
- General-purpose APIs should be policy-free, special-purpose APIs should be policy-rich.
- APIs should be designed from the perspective of the caller.
- Good APIs don't pass the buck.
- APIs should be documented before they are implemented.
- Good APIs are ergonomic.

User/system interfaces are essential components of any interactive software. The quality of interface is a major part of overall quality of interactive software. In [4], author presents a set of 10 quality criteria for user/system interfaces in usability. There are some advantages of satisfying usability criteria. Firstly, users can use software without any documentation. Secondly, interfaces provide necessary information. Finally, terminologies and definitions in the application area can be used by users.

The usability criteria for interactive software are listed in the sequel:

978-1-4799-3578-9/14 $31.00 © 2014 IEEE
DOI 10.1109/COMPSACW.2014.95

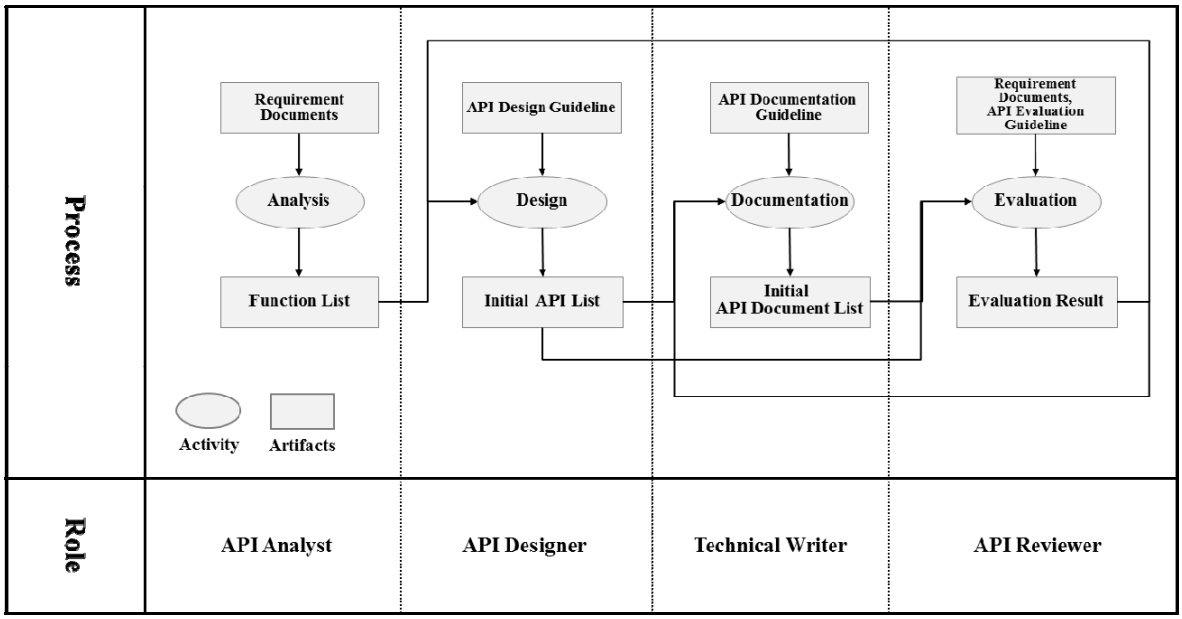

Fig. 1. API Design process

- Convenience of the language
- Convenience of the terminology
- Convenience of the metaphor
- Convenience of the inputs
- Functionality
- Simplicity
- Consistency
- Minimum memory load
- Navigability
- Least training

III. API DESIGN PROCESS

We have developed an API design process in terms of usability.

The roles in this process are "API analysts", "API designers", "Technical writers" and "API reviewers". Their responsibilities are defined as follows:

- API analysts: Analyze application requirements to refine functions that platform need to provide and require necessary APIs.
- API designers: Design APIs that "API analysts" required using the proposed design guidelines.
- Technical writers: Produce documentation for the APIs that "API designers" designed using the proposed documentation guidelines.
- API reviewers: Evaluate APIs and API documentation conforming to the evaluation guidelines. If they do not follow them, "API reviewers" would request API re-design or re-documentation.

The API design process comprises analysis, design, documentation, evaluation.

Fig. 1 shows the roles, artifacts and activities in each stage.

A. Analysis

In an analysis stage, "API analysts" elicit functions to be designed as APIs. They can use any requirement documents such as software requirement specification (SRS) documents or user experience scenario (UX Scenario). Previously, most of our APIs were designed based on not the user requirements but the functions that platform wanted to provide. Therefore, the APIs were hard to use for application developers and the applications began to possess complex logics which were unable to reuse.

When "API analysts" finish finding functions, they request for "API designers" to design APIs to provide the functions.

B. Design

In design stage, "API designers" design APIs for the functions from "API analysts" using the proposed design guidelines.

We propose design guidelines by referencing the criteria in the usability part of [4]. We select the criteria which are easy to apply and quantitatively measurable.

TABLE I. API DESIGN GUIDELINES

Metric	Description
Unambiguousity	The function of an API is figured out by its name. The name conforms to the coding conventions.
Convenience	The parameters in an API are minimally defined, and order and type of parameters keep consistency.
Primitiveness	An API provides a single function.

There are three metrics and the descriptions in Table 1.

"Unambiguousity" leads "API Designers" to make the name of APIs and their parameters in consistent forms. The name should clearly express that which function it provides. The abbreviation should not be used for names except for the general terminologies that every developer knows. The terminology should be unified. For example, if there is a terminology like "start" and "begin" in APIs, they should use one terminology only.

To meet "Convenience", "API designers" should define parameters considerably. Less than three parameters are recommended. When the number of parameters is more than three, it is difficult to set values correctly [5]. Consistent type and order of parameters makes APIs more convenient to use [1].

An API meets "Primitiveness" when it provides a single function. If the function can be provided by the combination of other APIs, it does not meet primitiveness.

"API designers" request to "Technical writers" to produce documentation when the designs are finished.

C. Documentation

"Technical writers" produce API documentation to make APIs convenient to use. The documentation includes necessary contents in terms of usage. When "Technical writers" finish documentation, "API reviewers" evaluate both APIs and API documentation whether they conform to the evaluation guidelines.

D. Evaluation

"API reviewers" evaluate APIs and API documentation conforming to the proposed evaluation guidelines. The evaluation guidelines possess the design guidelines and documentation guidelines.

The API design and documentation process finishes when the API marked "Pass" or "Accepted" on "Primitiveness" and the average score of other metrics is over the score that all the people related to this process agreed. In case of APIs marked "Fail" on "Primitiveness" or earned the lower average score than the agreed score, the API would return to design or documentation stage.

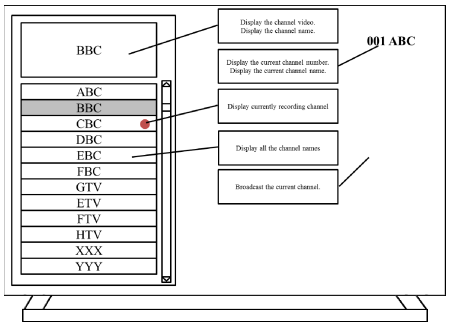

Fig. 2. Channel list UX Scenario analysis

TABLE II. FUNCTIONS ON DISPLAYING CHANNEL LIST

Functions
Display the channel video
Display the channel name
Display the current channel number
Display the current channel name
Display currently recording channel
Display all the channel names
Broadcast the current channel

IV. CASE STUDY

We adapted the proposed process to our new platform from October 2012 to September 2013. More than 20 "API analysts", 8 "API designers", 8 "Technical writers" and 28 "API reviewers" are involved. All members in the process agreed with the target score as 4.0/5.0.

In analysis stage, "API analysts" had analyzed requirements of previous platform because the requirements for new platform were not fixed at that time. They focused only on the common scenarios which were hardly changed for several years. After the requirements of new platform were released, they could elicit new functions.

Fig. 2 shows an example of the analysis result of a UX scenario for displaying channel list. The functions on Fig.2 are given in Table 2.

After "API analysts" finished eliciting functions, "API designers" designed APIs for the functions. They referred to the guidelines that we proposed. In some cases, they communicated with the "API analysts" to completely understand what the functions were and why they were needed in the scenario.

"Technical writers" produced documentation for the APIs. Some of them were "API designers" also. Since they had much knowledge about APIs that they designed, the documentation worked efficiently.

We had chosen the set of necessary contents in term of usability. The target contents were from Java Platform API [6]

and Android API [7]. The contents that API documentation includes are listed in the sequel:

- Function description
- Parameter names, types and value ranges
- Return value names, types and value ranges.
- Example usage code

We needed detailed guidelines to evaluate APIs and API documentation.

TABLE III. EVALUATION GUIDELINES

Metric	Description	Score
Unambiguousity	The function of an API is difficult to be figured out by its name.	-2
	The name does not conform the existing coding conventions	-1
	The terminology is not unified	-1
	Specific abbreviation is used.	-1 per abbreviation
Convenience	More than three parameters exist in an API.	-1 per parameters
	The order and type of parameters does not keep consistency.	-1
Primitiveness	An API does not provide a primitive function.	Fail
Documentation	Function description does not match to API	-1
	No information for parameters	-1
	No information for return values	-1
	No information for example usage code	-1

Table 3 describes how to score each APIs and API documentation.

An API can be marked with “Pass”, “Fail” or “Accepted” in “Primitiveness”. When the API provides a primitive function, it marked with “Pass”. If there are another APIs that provide a similar function or the function can be replaced by the combination of other APIs, it marked with “Fail”. When the possibility of security or performance issues exists, the API could mark with “Accepted” though it is not primitive.

The score of other metrics range from 0 to 5. The higher an API earns, the better it designed. The score subtract from 5 when each item is applicable to APIs or API documentation.

We have evaluated APIs for three well-known platforms and our platform with the proposed evaluation guidelines to check feasibility of our guidelines. We assumed APIs for well-known platforms would get more scores than our previous platform. The four platforms are as follows:

- APIs for our previous platform
- iOS APIs
- Android APIs
- Open Internet Protocol television Form(OIPF)

TABLE IV. API EVALUATION RESULT FOR IOS, ANDROID, OIPF, OUR PLATFORM

Metric	iOS	Android	OIPF	Our platform
Unambiguousity	4.6	5.0	4.9	3.1
Convenience	4.8	5.0	4.6	3.4
Primitiveness	Pass	Pass	Pass	Fail
Documentation	5.0	5.0	4.0	3.7

Since the functions that four platforms provide are different, we evaluated common functions only which are about “media”.

The evaluation result is shown in Table 4. As we expected, the usability of our previous platform APIs was far from the other platform APIs. The APIs for other platforms earned more than 4.0 for most of metrics. However, our previous platform APIs scored around 3.0. There were many abbreviations, parameters on them, and documentation hardly found nor was synchronized with the APIs. They marked with “Fail” on “Primitiveness” though those for other platforms marked “Pass” since there were many APIs which provide duplicated functions.

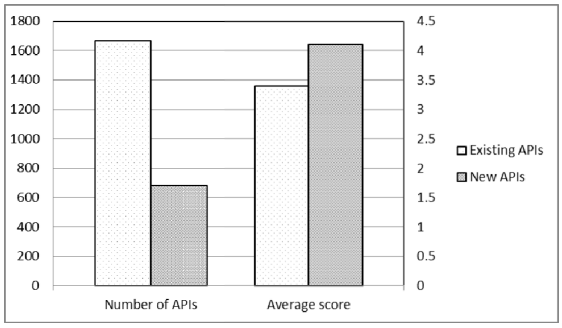

Fig. 3. Comparision between Existing APIs and New APIs

Fig. 3 shows the comparison between existing APIs and new APIs. We have designed 680 new APIs, and the average score was 4.1. Previously, the number of APIs is approximately 1,600 and its average score was 3.4. Comparing to the existing APIs, the score for new APIs increased by about 17%, but the number of APIs decreased by about 40% with full functions covered. Since we designed APIs based on requirements, unnecessary and duplicated APIs were not designed. Additionally, the functions for our platform are precise and specified.

V. CONCLUSION

The purpose of using APIs for developing software is to reuse existing code effectively [2]. The application development based on our platform can be effective when the platform provide APIs that application developers need to use and they are well defined.

We provide an API design process and guidelines in terms of usability. We produced 680 APIs through the process. Comparing to the previous APIs, their usability increased by approximately 17%. The number of APIs decreased by about 40% with full functions covered.

In the next research, we would add the metrics for performance and backward compatibility to our process. Performance and backward compatibility is important criteria as well as usability in designing APIs [1]. There were many design changes in our new APIs because of performance which makes many of them being marked “Accept” on “Primitiveness”. Since it was the first time to design overall APIs, backward compatibility is not essential for us. However, those metrics are inevitable to maintain the APIs for next version of our platform.

REFERENCES

[1] Martin Reddy. (2011). API Design for C++. Morgan Kaufmann Press.

[2] M. Henning. API design matters. Queue, 5 (4): 24 - 36,2007.

[3] J.Bloch. How to design a good API and why it matters. In OOPSLA '06: Companion to the 21st ACM SIGPLANconference on Object-oriented programming systems,languages, and applications, pages 506 - 507, New York,NY, USA, 2006. ACM.

[4] Ören, T.I., Cetin, S. (1999). Quality Criteria forUser/System Interfaces. In: Proc. of the Symposiumon Modelling and Analysis of C3I Systems, NATOR&T Agency, 12-14 January 1999, Fort d’Issy lesMoulineaux, near Paris, France, pp. 18-1 – 18-8.

[5] David Flanagan.(2011).Javascript: The Definitive Guide. O’Reilly.

[6] Oracle, Java Platform, Standard Ed. 7 API Specification, Retrieved January 17th, 2013 , from: http://docs.oracle.co/javase/7/docs/api/

[7] Google, Reference Android Developers, Retrieved April 1st, 2013 , from: http://developer.android.com/reference/classes.html

[8] Apple, IOS Developer Library, Retrieved April 1st, from: https://developer.apple.com/library/ios/navigation/

[9] OIPF, Volume 6 - Procedural Application Environment R2 v2.3, Retrieved April 1st, from: http://www.oipf.tv/specifications/login/com-sef-users-download?gid=117

[10] Jaroslav Tulach. (2008). Practical API Design. Apress