2018 IEEE International Conference on Software Maintenance and Evolution

An experience report of the API evolution and maintenance for software platforms

Hobum Kwon*, Juwon Ahn*, Sunggyu Choi*, Jakub Siewierski†, Piotr Kosko†, and Piotr Szydelko†
*Samsung Research, Samsung Electronics, Seoul, South Korea**
Samsung R&D Institute Poland, Samsung Electronics, Warszawa, Poland†
{hobum.kwon, juwon.ahn, sunggyu.choi, j.siewierski, p.kosko, p.szydelko}@samsung.com

***Abstract*—Development and maintenance of software platform APIs are challenging because new APIs are constantly added in new software platforms. Furthermore, software platform API development requires a lot of stakeholders to work together on tight release schedules. Application developers use platform's APIs to create their applications and therefore providing a well-defined and comprehensive set of platform APIs may be the most basic requirement for software platforms. To provide such APIs, API usability should be secured and API backward compatibility should be guaranteed in subsequent platform releases. In these circumstances, sharing lessons learned from multiple years of experience of platform API development, maintenance, and releases using an integrated API development process can benefit API researchers and practitioners who have similar needs to create or adopt API development process for their projects. In this paper we share an API development and maintenance process for multi-device Tizen software platform, which we call the Tizen API Change Request (ACR) process. The process has been used among various Tizen API stakeholders for several years of Tizen platform and SDK releases to keep API usability and compatibility high. We believe the process can be further applied to various software platforms and projects to systematically develop and maintain their APIs.**

Keywords— API development, API maintenance, software platform APIs, API review, multi-device API

I. Introduction

Application developers for software platforms create their applications with APIs provided by the software platforms. Once platform APIs are released via SDKs, the APIs should not be changed in the following releases of software platforms to keep API backward compatibility. Practically, it is difficult to keep API backward compatibility because new APIs are continuously added in every release of a software platforms, and sometimes critical API design mistakes are found after release. Keeping API backward compatibility in software platforms enable applications developed on a current version of the platform to run on future versions of the platforms. Another important aspect of platform API development is API usability. If the platform provides well-defined APIs, application developers can easily create applications they want. On the contrary, if the APIs are not easy to use, application development will increase dissatisfaction with the software platforms and it will become a major obstacle for the success of the software platforms. Because of the API backward compatibility, APIs even badly designed should not be changed after release. Therefore, platform developers should be extremely careful when designing and releasing platform APIs.

This paper presents how the open source Tizen platform [1] has ensured API usability and backward compatibility for multiple target devices for more than six years. At the center of Tizen API development and maintenance, there is a process called the API Change Request (ACR). Tizen ACR is a process that defines the states required to develop and maintain APIs for software platforms. This process has enabled various stakeholders of APIs to cooperate; the stakeholders include API developers, API domain reviewers, API reviewers, SDK developers, API test case reviewers, document writers, and document reviewers. In section II, we present the overview of the Tizen ACR process. In section III, we share tools used for the Tizen ACR process to reduce human effort and error. Finally, we present related works in section IV, and conclusions and future works in section V.

II. Tizen ACR

At the beginning, Tizen ACR started as simple submissions and approvals by emails between API developers and API reviewers for Tizen Native APIs [2], which are based on the C programming language. When we were using the email-based ACR, we faced the following issues:

- An email is basically a conversation between a sender and a receiver, so other stakeholders can't be notified unless they are explicitly added in the recipients by API developers and reviewers.
- There was no system like a dashboard to see status of APIs, e.g. under development or completed.
- It is hard to retrieve the history of APIs since no query was allowed except text search in the ACR emails.

For those reasons, we decided to introduce a new ACR system. To do this, we analyzed the overall API development process in Tizen and we listed all API development states and stakeholders as listed in TABLE I and came up with a new API development process to efficiently communicate between the stakeholders. To implement the process, we adopted the JIRA [3] bug system as Tizen had already been using JIRA for its bug tracking system. The JIRA bug system satisfied most of our system requirements. Most notably it provided a submission-based and approval-based workflow, task search ability, status-change notifications, etc.

Figure 1 shows the overall workflow of the Tizen ACR process. At the first state, API developers get a request for API changes (add, remove, deprecate, or change) and API developers then start preparing the ACR. It could be a request for adding a simple getter API or whole new framework-level large number of APIs. At this state, peer reviews from co-workers who have developed the platform APIs can give valu-

2576-3148/18/$31.00 ©2018 IEEE
DOI 10.1109/ICSME.2018.00034

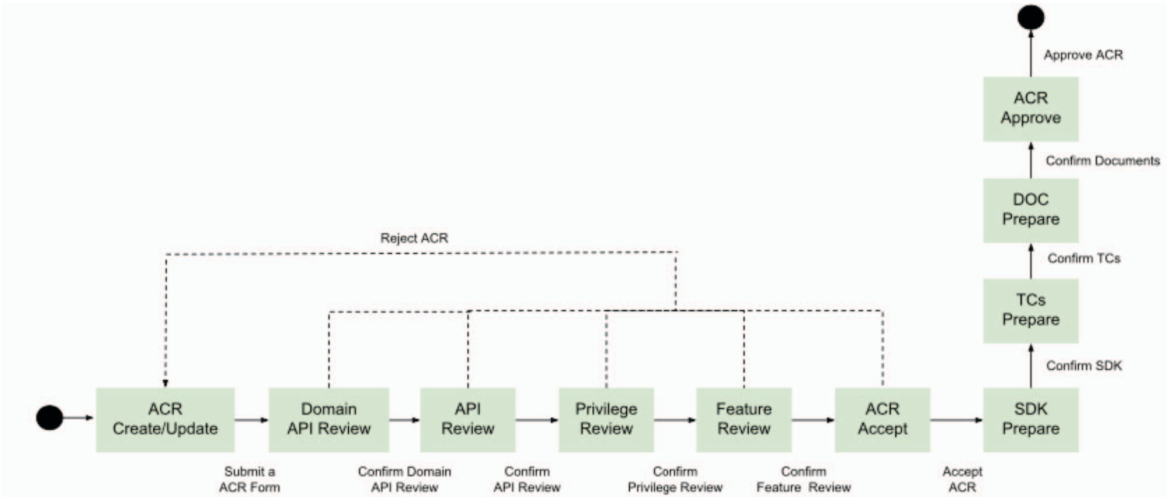

Fig. 1. The overall workflow of Tizen ACR

TABLE I. DESCRIPTION OF TIZEN ACR STATES

State	Stakeholder	Result (Output)
ACR Create / Update	API developers	The ACR form written by API developers
Domain API Review	Domain reviewers	The ACR domain reviewed by domain reviewers
API Review	API reviewers	The API reviewed by API reviewers
Privilege Review	Security reviewers	The API security privilege reviewed by security reviewers
Feature Review	Feature reviewers	The API feature reviewed by feature reviewers
ACR Accept	API reviewers	ACR is accepted. Now all API related tasks are completed, ready to implement.
SDK Prepare	SDK developers	The APIs in the ACR are now SDK prepared.
TCs Prepare	API test case reviewers	The test cases of Tizen compatibility tests for the APIs are now ready.
Documentation Prepare	DOC writers / reviewers	Required developer guides for the APIs in the ACR are now completed.
ACR Approve	--	The APIs are officially added to the public API scope of Tizen platform and SDK.

able feedbacks as they are already accustomed to the API style and are knowledgeable about the underlying programming languages and the foundation APIs. In the ACR form, mandatory fields include the API specification via a Gerrit [4] link for the API review since the Tizen project uses Gerrit to review code changes, Git [5] path for the APIs, target SDK [6] or API version, additional description of the APIs which includes; contact of the ACR submitter, purpose, required API privileges [7], and features [8].

After submitting the ACR form, API developers need to get an approval from the domain reviewers of the API. In Tizen, APIs are divided into domains based on technologies such as multimedia, graphics, and so on, and most API domains consist of 10 to 30 developers. In this state, developers in the domain, which the ACR requested APIs belong to, review the APIs and a designated approver, often a leader of the domain, decides approval or rejection. The Domain API Review state was not included when the ACR process was originally defined. It was added because of the following reason: first, domain API reviewers have the most expertise in the domain so they can evaluate whether the API changes in an ACR are correctly exposing the underlying framework features and the APIs are well-defined from the view of domain experts. Such reviews are hard to get from general API reviewers who review in the next state. Second, it is important to get an agreement with the domain reviewers since the domain reviewers are responsible for developing and maintaining the APIs after all. After approval, the next state is API Review. The API Review state should include the following reviews:

- Check whether the submitted ACR breaks API backward compatibility and the APIs in the ACR fully follow API style guides.
- Check whether API descriptions are understandable from a 3rd party application developer's perspective.

- Check whether similar APIs already exist in the platform and all required information used to generate API specification are fully prepared.

In the Tizen project, the API reviewers consist of 3~10 members for each Tizen application model because the review process requires knowledge on underlying programming languages and the foundation APIs which are often originated from open source projects, for example, CoreFX APIs [9] in Tizen .NET APIs [10]. Thus, it is reasonable to have different API reviewers for each Tizen application model. The API reviewers are also authors of the API style guide for the application model, so you can rely on the API reviews from the API reviewers since they have a clear idea about how the APIs should be defined.

After the API Review state, the next states are checking API privileges (Privilege Review) and features (Feature Reviews) which are additional information required for the Tizen APIs. In both states, API security reviewers and API feature reviewers need to understand the purpose and usages of APIs to decide appropriate API privileges and features for the APIs. This state may be used for reviewing security issues of APIs. Then the APIs are now in the ACR Accept state. Now the ACR requested APIs' signature and descriptions, privileges, and features are all fixed and the API developers can finalize their API implementation. After they complete their API implementation, the APIs should be properly added to the platform binaries.

The following state is SDK Prepare, which adds the APIs to the SDKs so application developers can use the APIs in the SDKs to compile and execute on the emulators or devices connected to the SDKs. The next state is TC Prepare. In this state, API developers create API verification test cases to guarantee the API functionality and compatibility. API test case reviewers must approve this state only if required test cases are provided for the APIs. It is desirable to have 100% unit test case coverage for the APIs.

The next state is Document Prepare, in which document writers and reviewers check whether API specifications are properly generated and whether additional developer guides need to be prepared. After Documentation Prepare, the Tizen ACR process is completed and the APIs are now officially part of Tizen public APIs.

The Tizen ACR process has helped evolving and managing Tizen APIs conveniently among various API stakeholders. Tizen ACR has provided the following benefits to Tizen API stakeholders. First, the API stakeholders can develop and maintain APIs systematically and cooperatively by knowing state of API development. Second, the history of API changes is now easily searchable in various query conditions and accessible to all API stakeholders. Third, API developers' awareness of the importance for API usability and backward compatibility is has been increased and it helped Tizen platform APIs improved significantly in API consistency and resulted in keeping Tizen APIs backward compatible. As a result, TABLE II shows statistics of added APIs compared to deprecated and removed APIs for Tizen Mobile Native APIs from Tizen platform version 2.3 to 4.0. Only few APIs were deprecated and removed from Tizen platform 2.3 to 4.0 for Tizen Native APIs. In the last row in TABLE II, the number of reviewers for the ACR requested APIs are increasing as more engineers are actively participating in API reviews, which is a positive signal to make Tizen APIs better. Fourth, more accurate API development time estimation for Tizen platform release is feasible by referring to previous ACR history. Milestones such as API freeze, in other word announcing no more ACR accepted in this platform release, have been added to the Tizen platform development process to ensure that all API development for the platform release should be ended at least one and half month before the platform release based on our ACR experience. Finally, the Tizen ACR process could be applied to other similar API development and management required projects beyond Tizen since it gives a clear insight for overall API development and maintenance workflow and the states in the process are general enough to be adapted or customized for any platforms or projects to deploy APIs via SDKs.

TABLE II. ACR STATISTICS OF TIZEN NATIVE APIS

Measurement item	Count				
	Tizen ***2.3***	Tizen ***2.3.1***	Tizen ***2.4***	Tizen ***3.0***	Tizen ***4.0***
ACR count	-	69	297	443	225
Added API count	4759	451	3028	3524	659
Deprecated API count	-	24	74	237	275
Removed API count	-	-	-	-	6
ACR count including ABI Break	-	-	-	-	2
ABI Break count after release	-	0	0	0	0
Average elapsed time of Review (unit: day)	-	10	14	15	16
Average involved reviewer	-	6.9	7.4	9.1	9.3

III. TOOLS USED FOR TIZEN ACR

In this section, we share the tools we have created to facilitate the ACR process and ensure API backward compatibility. At the beginning, the entire ACR process was manually performed. To make the review easier, more precise and efficient, a few tools were created to automate parts of the review and to detect API backward compatibility breaks. First, we have created API review scripts for each of three Tizen application models [11][12][13]. The review scripts help API developers to find self-checking errors before submitting ACR and API reviewers to detect errors automatically. The API review scripts were created to detect two types of errors, certain errors and probable errors. The certain errors are the type of errors that can be detected with 100% certainty. An example of certain errors in Tizen Native API review script is missing parameter description or missing mandatory API tags used to generate API specification document. Probable errors are more complicated to detect than certain errors. For instance, if function's output parameter is a C string (char **), then the API

reference is required to state whether the string should be freed by the caller or not, and if not, when the string should be freed by platform. The tool attempts to detect the phrases related to freeing the data. If no such phrase is found in the function's description, then the tool issues a question, alerting the user to a potential issue. It's recommended that the tools are not the only way of reviewing the API, and that human supervision is used.

In addition to API review script tools, we also created a code skeleton generator for Tizen Web APIs [14]. The code skeleton generator can be used to prepare the skeleton code from the API description for ACR process. When the API specification is ready using WIDL [15] file for Tizen Web APIs, the developers can run the script to generate skeleton code and to get an auto-generated API implementation.

Another example of the tools is the code formatting tool to ensure a consistent code style in API implementation. A common style helps code review and future maintenance.

Finally, API binary compatibility checker tools are used to detect violation of API compatibility, such as changed API signatures. Overall, the automated tools help improve the efficiency and accuracy of the review process.

IV. Related Works

There has been API related research but those works have mainly focused on improving API usability frameworks using cognitive dimensions [16][17], human-centered methods [18], measuring API usability [19], and interview [20]. However, no experience report of API process especially designed and used for large-scale software platforms targeting multiple type of devices was found. In fact, the methods in the previous research may be applied within the states in the ACR process to improve API usability or to measure API usability improvement after the ACR reviews. On a few occasions we have tried to interview API developers to figure out API usability changes before and after the ACR in quantitative way. However, it was hard to obtain an actionable result from some questions in the cognitive dimension framework because of the prior work experiences of the interviewees. For example, preferred API styles are different between close-to-system level developers who have used C mostly and close-to-application level developers who had used object-oriented programming styles. Therefore, more effective and widely applicable questionnaires for API developers should be developed in the future research. Another approach we may consider is applying more systematic ways of peer reviews [21] to enhance Domain API Review and API Review state in Tizen ACR. Also, the API style guides used for Tizen application models can be improved by applying similar analysis approach in the style guide analysis of REST API [22].

V. Conclusion and Future Works

The Tizen ACR process helped API stakeholders to work efficiently and systematically while keeping API consistency for API usability and API backward compatibility for multi-device Tizen platform and it can be easily applicable to any API release required software platforms or frameworks. For the future works, we noticed that the following effort can make the current the ACR process better for API usability and compatibility. First, inconsistency between API specification and API implementation should be examined in the ACR process. Second, in the API review, research on how other platforms or widely-used libraries expose such features via APIs should be performed by API developers and should be examined by API reviewers to provide more usable APIs and also to detect missing APIs for the future works. Third, the test coverage of API unit tests for the ACR requested APIs should be forcedly 100% to make API backward compatible. Finally, in the future AI and machine-learning can be applied to the tools in the ACR process especially for API review process to give better and consistent review to API developers.

Acknowledgment

The authors thank Sidharth Gupta at the University of Illinois for helpful comments on our draft.

References

1. https://developer.tizen.org
2. https://developer.tizen.org/development/api-references/native-application
3. https://www.atlassian.com/software/jira
4. https://www.gerritcodereview.com
5. https://git-scm.com
6. https://en.wikipedia.org/wiki/Software_development_kit
7. https://developer.tizen.org/development/training/native-application/understanding-tizen-programming/security-and-api-privileges
8. https://developer.tizen.org/development/guides/native-application/device-settings-and-systems/system-information#feature
9. https://github.com/dotnet/corefx
10. https://developer.tizen.org/development/api-reference/.net-application
11. https://developer.tizen.org/development/training/native-application/tizen-application-model
12. https://developer.tizen.org/development/training/web-application/tizen-application-model
13. https://developer.tizen.org/development/training/.net-application
14. https://developer.tizen.org/development/api-references/web-application
15. https://www.w3.org/TR/NOTE-widl-970922
16. Steven Clarke, "Using the cognitive dimensions framework to reason about API usability", Visual Studio Usability Group, Microsoft Corporation, OSU EECS Colloquium June 1st 2004.
17. Steven Clarke, "Describing and measuring API usability with the cognitive dimensions", in Cog. Dimensions of Notations: 10th Ann, Workshop, 131-132.
18. Myers BA, Stylos J "Improving API Usability", Communications of the ACM, vol. 59 issue 6, pp 62-69 June 2016.
19. Steven Clarke, "Measuring API Usability", Dr. Dobb's. Journal, pp S6-S9. May 2004.
20. Marco Piccioni, Carlo A Furia, and Bertrand Meyer, "An Empirical Study of API Usability", Empirical Software Engineering and Measurement, 2013 ACM/IEEE international symposium on. IEEE, 5-14.
21. Umer Farooq, Leon Welicki, Dieter Zirkler, "API Usability Peer Reviews: A Method for Evaluating the Usability of Application Programming Interfaces", Proceedings of the 28th international conference on Human factors in computing systems, CHI '10, pp. 2327-2336, New York, NY, USA, 2010. ACM.
22. Lauren Murphy, Tosin Alliyu, Andrew Macvean, Mary Beth Kery, Brad A. Myers, "Preliminary Analysis of REST API Style Guidelines", 8th Workshop on Evaluation and Usability of Programming Languages and Tools (PLATEAU'17) at SPLASH 2017, October 23, 2017, Vancouver, CA.