A Gestão de API Realmente Melhora a Relação com Desenvolvedores?

Filipe D. Coelho[1], **Cleidson R. B. de Souza**[1], **Emanuel A. Rodrigues**[1]

[1]Universidade Federal do Pará
Tv. Augusto Côrrea, n. 1
66.075-110 – Belém – PA – Brasil

filipe.bcdc@gmail.com, cleidson.desouza@acm.org, emanorodrigues@gmail.com

Abstract. *The development and adoption of Application Programming Interfaces (APIs) has several obstacles, among them is their acceptance by the developers. Several causes may be linked to this problem, such as lack of documentation, learning difficulties, low number of developers using the API, etc. API management attempts to address these problems. Its goal is to facilitate developers' engagement, which is one of the risk factors in API adoption.*

Resumo. *O desenvolvimento de Application Programming Interface (APIs) possui diversos entraves, entre eles está a aceitação por parte dos desenvolvedores. Isto é, desenvolvedores podem adotar ou não uma API. Diversas causas podem estar atreladas a este problema, incluindo a falta de documentação sobre uma API, a dificuldade no aprendizado da mesma, o baixo número de desenvolvedores em comunidades, etc. Para tanto, a gestão de APIs visa tratar alguns destes problemas. Ela visa facilitar o engajamento dos desenvolvedores que é considerado como um dos fatores de risco em uma API.*

1. Introdução

Uma *Application program interface* (API) é um mecanismo de software que permite a disponibilização de serviços, através de interfaces, para usuários externos, ou seja, uma API é um conjunto de rotinas e padrões de programação para acesso a um serviço de software qualquer. Frequentemente, as APIs oferecem serviços baseados na *Web*. Assim, um sistema de software moderno não é mais independente, este funciona através de funcionalidades disponibilizadas por outras organizações. Estas soluções se mostram eficazes, devido sua especialização [Robillard 2009].

O uso de APIs durante o desenvolvimento de software é cada vez mais imprescindível. Diversos sistemas de software apoiam-se diretamente em APIs de terceiros [Bogart et al. 2016]. As APIs deram as organizações a possibilidade de crescimento muito além do que teriam se estivessem trabalhando por conta própria. Assim, diferentes organizações passaram a comunicar-se entre si e gerar serviços umas às outras.

Entretanto, o uso de APIs tem se tornado cada vez mais complexo, haja vista que a organização detentora da API, frequentemente desconhece como, quem e onde as aplicações que consomem serviços da API estão sendo usadas. Surge então a necessidade de conhecer como os serviços de uma API estão sendo utilizados para planejar a evolução desta API e para evitar danos aos seus usuários[Bogart et al. 2016].

O uso de APIs web se dá através de requisições, em sua maioria *REST-Full* com o protocolo HTTP. Ao seguir os padrões definidos pela organização detentora da API, o usuário é capaz de propiciar a comunicação entre sistemas especializados. Esta prática aumenta a produtividade e facilita a solução de diversos problemas. Por exemplo, a distribuição de tarefas entre diversos sistemas especializados, através de APIs, proporciona um ganho de desempenho e coesão ao sistema que utiliza estas APIs[Brajesh 2017].

A necessidade de gerir uma API se dá devido às falhas constantes e potenciais que podem acarretar no mau uso de suas funcionalidades. Por exemplo, a evolução (mudanças) desordenada de uma API pode acarretar problemas de segurança, documentação, confiabilidade dos dados, entre outros [Brito et al. 2016, Linares-Vásquez et al. 2013, Robbes et al. 2012] [Wijayarathna et al. 2017] [Stylos and Clarke 2007, Stylos and Myers 2006].

Existem diversas causas para problemas em APIs [Bogart et al. 2016], incluindo o desconhecimento da forma como uma API é utilizada e das necessidades dos desenvolvedores que a utilizam. A ineficiência causada por desbalanceamentos de carga, falta de priorização de requisições, ataques maliciosos, a falta de certificados de autenticidade e auditoria de usuários e, finalmente, a falta de portais e documentação adequada para os desenvolvedores [Bogart et al. 2016]. A gestão de APIs visa tratar destes fatores, entre outros aspectos, facilitando o a tomada de decisão em relação às APIs.

Este trabalho discute o tema de gestão de APIs e, especificamente, o impacto desta gestão na comunicação entre a organização que desenvolve a API e os desenvolvedores que utilizam a API. Este trabalho analisa se as ideias de gestão de APIs minimizam os problemas encontrados. Este é um trabalho em andamento a nível de mestrado.

2. Gestão de APIs

A gestão de APIs trata essencialmente de fatores associados ao uso das mesmas incluindo segurança, usabilidade, evolução, análise de utilização, etc. Estes fatores serão brevemente explanados nesta seção.

[Brajesh 2017] apresenta um modelo de visualização da gestão de API. Esta gestão é dividida em 4 quadrantes que cobrem por completo diferentes características das APIs. Os quadrantes apresentam uma visão de todas os aspectos de uma boa gestão de API, além de agrupar os requisitos principais de uma API, os quais são necessários para a sua continuidade. A figura 1 apresenta os 4 quadrantes propostos por [Brajesh 2017] com suas características macros e micros.

Habilitação do desenvolvedor para APIs Esta característica é responsável por buscar, facilitar e engajar os desenvolvedores em relação às APIs levando em consideração a documentação das mesmas, mecanismos de comunicação e suporte aos desenvolvedores entre outros aspectos. Um exemplo de um mecanismo de comunicação são os fóruns onlines que frequentemente existem associados às APIs. Este quadrante tem papel fundamental no crescimento e consolidação de uma API. Por exemplo, através de mecanismos como *hackathons* para a divulgação de uma API [Komssi et al. 2015], ou através de outras formas de divulgação. Em suma, este quadrante é extremamente importante para permitir uma comunicação clara e fácil com os desenvolvedores.

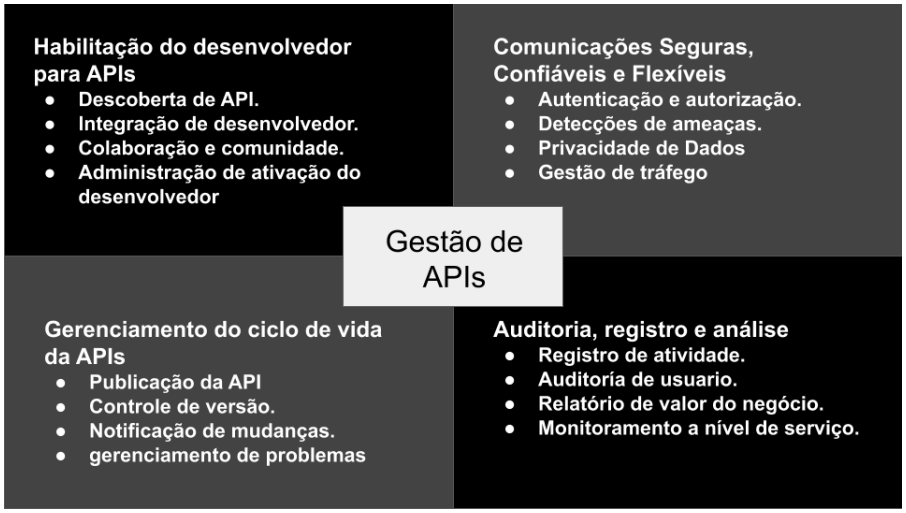

Figura 1. Divisão da Gestão de APIs em Quadrantes
Fonte: [Brajesh 2017]

Comunicações Seguras, Confiáveis e Flexíveis Este quadrante é responsável pela análise da estrutura e disponibilidade de uma API. Ele engloba também os níveis de acesso, identificação de usuários e requisitos de privacidade de dados, detecções e correções de vulnerabilidades. O plano de contingência deve estar descrito neste quadrante, desta forma, avaliando e minimizando os riscos da API.

Gerenciamento do ciclo de vida da API O ciclo de vida de uma API diz respeito à continuidade de funcionamento da mesma. Este ciclo começa no planejamento do desenvolvimento da API, passando pelas evoluções e melhorias nas documentações, chegando até o momento de descontinua-la. Durante o ciclo de vida, existe a responsabilidade de sinalizar aos desenvolvedores sobre evoluções (mudanças), melhorias e a remoção de funcionalidades.

Auditoria, registro e análise Para o fornecimento de informações e dados para tomadas de decisão é necessário o registro de todas a requisições feitas à uma API e a forma como tais requisições foram realizadas, em que condições e por quem, e finalmente, se tais requisições tiveram sucesso ou não. Este quadrante sugere a necessidade de se realizar este registro. A partir do mesmo é possível encontrar e corrigir problemas na documentação, desbalanceamento de carga, mudanças para atender necessidades, etc.

3. Gestão de APIs e sua relação com os desenvolvedores

A gestão de APIs, como descrito na seção 2, tem papel fundamental no alcance dos desenvolvedores. Assim, o desenvolvimento de uma API não é uma tarefa simples [Cataldo and d. Souza 2014]. Há também a dificuldade no uso correto das funcionalidades disponibilizadas.

Diversos fatores podem ser causas de desmotivação dos desenvolvedores na utilização de uma API: a documentação pouco explicativa, a falta de comunicação en-

tre a comunidade de desenvolvedores e a organização detentora, a forma de monetização com a API, a dificuldade no aprendizado dos padrões usados na API, etc [Robillard 2009].

Existem diversos trabalhos descrevendo como uma API deveria ser desenvolvida e planejada para facilitar o seu entendimento e proporcionar o menor risco do uso incorreto de suas funcionalidades [Bloch 2006].

Diversos estudos [Bloch 2006, Bogart et al. 2016, Cataldo and d. Souza 2014, Komssi et al. 2015] apontam diferentes mecanismos de atração de desenvolvedores para utilizar os serviços disponibilizados por uma API, entre eles: *hackthons* [Komssi et al. 2015], apoio a documentação, custos reduzidos ou nulos para aprender uma API, comunidades e fóruns de discussões online. Estes mecanismos são usados pelas organizações proprietárias das APIs para apresentar os benefícios do uso das mesmas, permitindo assim a atração e manutenção de desenvolvedores externos utilizando estas APIs.

4. Resultados Parciais

Conforme mencionado anteriormente, APIs são um mecanismo de programação que permitem que serviços de software, e consequentemente, organizações colaborem entre si. Isto é, APIs são um importante mecanismo de colaboração entre organizações, especialmente aquelas de de desenvolvimento de software. Assim, o objetivo principal deste trabalho é entender as estratégias utilizadas por organizações para permitir que desenvolvedores de software facilmente adotem e permaneçam engajados usando suas APIs.

Este é um trabalho que está em andamento. Para entender as estratégias utilizadas pelas organizações, conduzimos uma revisão da literatura utilizando a técnica *snowballing*, que foi usada devido ao seu comprometimento com as sementes () e a facilidade de busca. Este projeto usou como sementes (*seeds*) iniciais os seguintes artigos: [Sawant et al. 2018, Cataldo and d. Souza 2014, Stylos and Clarke 2007, Stylos and Myers 2006, Mosqueira-Rey et al. 2018]. Foi dada enfase em evolução de APIs e usabilidade e, posteriormente, acrescentou-se outras vertentes vinculados à gestão de APIs. Um total de 89 trabalhos foram analisados até o momento.

Através destes resultados pode-se notar que pesquisas em diversas áreas poderiam ser complementares. Por exemplo, a literatura cita diversas pesquisas na areá de documentação de APIs [Stylos and Myers 2006, Robillard 2009] que podem facilitar o entendimento dos usuários. Em contra-partida as plataformas de perguntas e resposta são um auxilio à documentação de APIs [Wang et al. 2013]. Por exemplo, o design de documentação mais claro unido a uma comunidade organizada, pode ajudar nas dificuldades de entendimento de uso em uma API [Wang et al. 2013, Bloch 2006] . Além disso, não restam dúvidas sobre a necessidade da atração e gestão de desenvolvedores de software para o sucesso de uma API [Stylos and Clarke 2007]. A baixa complexidade de uso [Bloch 2006], o custo de aprendizagem [Robillard 2009], as comunidades bem organizadas[Bogart et al. 2016], a criação de eventos direcionados [Komssi et al. 2015], uma API consolidada com um bom controle de versão [Sawant et al. 2018], entre outros, são aspectos de extrema importância para a atração e retenção de desenvolvedores. Através dos estudos na literatura pode-se perceber alguns pontos estão ainda sem resposta, sendo assim o projeto levantou alguns questionamentos:

1. Quais os fatores que influenciam um desenvolvedor na escolha de uma API?

2. Dentre estes fatores, qual o fator mais importante?
3. Qual destes fatores tem menor relevância?
4. Aspectos sociais, como comunidades e fóruns online, amigos, etc, influenciam os desenvolvedores na escolha de uma API?
5. Até que ponto a gestão de API proporciona a atração de desenvolvedores? E quais são os benefícios aos usuários (organização detentora e desenvolvedores)?

Todas as características estudadas são imprescindíveis para a consolidação de uma API. Entretanto, conhecer o que agrega mais valor à uma API é de extrema importância. Este poderá direcionar para um plano de maturidade no desenvolvimento e ciclo de vida de APIs.

5. Conclusões

A pesquisa atual se mostra relevante para o entendimento, como um todo, da atuação da gestão de APIs no contexto de desenvolvimento colaborativo de software. Os resultados iniciais sugerem que já existem estudos que indicam importantes aspectos para facilitar a adoção e utilização de APIs [Wang et al. 2013, Ellis et al. 2007, Bogart et al. 2016, Cataldo and d. Souza 2014]. Isto também sugere que o uso de gestão de APIs poderia proporcionar aos desenvolvedores uma experiência mais adequada ao integrar diversas abordagens diferentes. A comunicação entre a organização detentora da API e os usuários (desenvolvedores) é essencial para que uma API possa continuar evoluindo e gerando resultados para os seus usuários e também para a organização. Fóruns, *hackathons*, comunidades, documentações, etc são diferentes mecanismos que permitem esta comunicação. Assim, há a necessidade de entender qual a melhor forma de comunicar-se com os desenvolvedores.

Um estudo através da literatura e estudos empíricos devem ser conduzidos para o entendimento do melhor mecanismo de comunicação entre as partes grupos, bem como qual o momento do ciclo de vida da API mais adequado para utilizar um dado mecanismo.

Porém, o trabalho apresenta diversas dificuldade no estudo da comunicação entre as partes interessadas. A literatura apresenta algumas soluções como: os fóruns de perguntas e respostas os quais podem dar a explicação equivocada de algumas funcionalidades ou modelos de documentações mais claros que não são usados largamente pelas organizações dificultando o estudo de sua eficácia [Wang et al. 2013, Ellis et al. 2007] .

Agradecimentos

Os autores gostariam de agradecer ao CNPq pelo apoio financeiro através dos processos 420801/2016-2 e 311256/2018-0.

Referências

Bloch, J. (2006). How to design a good api and why it matters. In *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '06, pages 506–507, New York, NY, USA. ACM.

Bogart, C., Kästner, C., Herbsleb, J., and Thung, F. (2016). How to break an api: cost negotiation and community values in three software ecosystems. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 109–120. ACM.

Brajesh, D. (2017). *API Management*, pages 15–28. Apress, Berkeley, CA.

Brito, G., Hora, A., Valente, M. T., and Robbes, R. (2016). Do developers deprecate apis with replacement messages? a large-scale analysis on java systems. In *2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)*, volume 1, pages 360–369.

Cataldo, M. and d. Souza, C. R. B. (2014). Exploring the impact of api complexity on failure-proneness. In *2014 IEEE 9th International Conference on Global Software Engineering*, pages 36–45.

Ellis, B., Stylos, J., and Myers, B. (2007). The factory pattern in api design: A usability evaluation. In *Proceedings of the 29th International Conference on Software Engineering*, ICSE '07, pages 302–312, Washington, DC, USA. IEEE Computer Society.

Komssi, M., Pichlis, D., Raatikainen, M., Kindström, K., and Järvinen, J. (2015). What are hackathons for? *IEEE Software*, 32(5):60–67.

Linares-Vásquez, M., Bavota, G., Bernal-Cárdenas, C., Di Penta, M., Oliveto, R., and Poshyvanyk, D. (2013). Api change and fault proneness: A threat to the success of android apps. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 477–487, New York, NY, USA. ACM.

Mosqueira-Rey, E., Alonso-Ríos, D., Moret-Bonillo, V., Fernández-Varela, I., and Álvarez Estévez, D. (2018). A systematic approach to api usability: Taxonomy-derived criteria and a case study. *Information and Software Technology*, 97:46 – 63.

Robbes, R., Lungu, M., and Röthlisberger, D. (2012). How do developers react to api deprecation?: The case of a smalltalk ecosystem. In *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, FSE '12, pages 56:1–56:11, New York, NY, USA. ACM.

Robillard, M. P. (2009). What makes apis hard to learn? answers from developers. *IEEE Software*, 26(6):27–34.

Sawant, A. A., Aniche, M., van Deursen, A., and Bacchelli, A. (2018). Understanding developers' needs on deprecation as a language feature. In *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, pages 561–571.

Stylos, J. and Clarke, S. (2007). Usability implications of requiring parameters in objects' constructors. In *Proceedings of the 29th International Conference on Software Engineering*, ICSE '07, pages 529–539, Washington, DC, USA. IEEE Computer Society.

Stylos, J. and Myers, B. A. (2006). Mica: A web-search tool for finding api components and examples. In *Visual Languages and Human-Centric Computing (VL/HCC'06)*, pages 195–202.

Wang, S., Lo, D., and Jiang, L. (2013). An empirical study on developer interactions in stackoverflow. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 1019–1024, New York, NY, USA. ACM.

Wijayarathna, C., Arachchilage, N. A. G., and Slay, J. (2017). A generic cognitive dimensions questionnaire to evaluate the usability of security apis. In Tryfonas, T., editor, *Human Aspects of Information Security, Privacy and Trust*, pages 160–173, Cham. Springer International Publishing.