Feature Removal on Software Platforms

Discontinued Core Features on Browser Platforms – A Case Study on Mozilla Firefox

Benedict Bender, Andrzej Szadowiak
Chair of Business Informatics, esp. Processes and Systems
University of Potsdam
Potsdam, Germany
{bebender, szadowiak}@uni-potsdam.de

***Abstract*—Software platforms allow for the extension of features by third-party contributors. Thereby, platform innovation is an important aspects of platforms attractiveness for users and complementors. While previous research focused the introduction of new features, the aspect of feature removal and discontinued features on software platforms has been disregarded. To explore the phenomenon and motivations for feature removal on software platforms, a review of recent literature is provided. To illustrate the existence of and motivations for feature removal, a case study of the browser platform Mozilla Firefox is presented. The results reveal feature removal to regularly occur on browser platforms for user- and developer-related features. Frequent reasons for feature removal involve unused features, security concerns, and bugs. Related motivations for feature removal are discussed from the platform owner's perspective. Implications for complementors and users are highlighted.**

Keywords—Software Platforms; Discontinued Features; Feature Removal; Lean Core; Platform Innovation; Browser Platform; Mozilla Firefox

I. INTRODUCTION

Software platforms with their open paradigm allow for the extension of additional features from third parties [1]. Through contributions, third-party developers extend functionality beyond the platform owner's core [1]. Thereby, software platforms allow platform owners to deliver more features than a single entity could provide on its own [2]. Complementors provide functionality in form of packaged code fragments, also referred to as 'extensions' or 'add-ons' [1], [3].

Platform innovation is considered a success factor for digital platforms [4]. Platform innovation from a user perspective involves the introduction of new platform features. Platform users evaluate different platforms by their functionality [5]. Related functionality in the form of platform features can be either provided by the platform core or third-party complements. While the platform core usually provides functionality relevant for the general audience, third-party extensions focus on specialized functionality [6], [7]. In this regard, platform owners are highly interested to attract complementors to their platform to ensure its competitiveness [8].

Digital platforms are known to progress over time [9]. Typically, platform owners regularly release updates to their core platform, that introduce new features. Core updates thereby affect platform features as well as development resources [10]. While the aspect of feature introduction in the context of platform innovation on software platforms is covered by research studies [4], [8], [10], the research regarding the removal of software features has been limited to identifying unused and least profitable software features in general [11]. Similar to feature introduction, the removal of features may occur during platform updates that regularly occur and affect the platform ecosystem participants.

Feature removal is closely related to the idea of a lean core. Software platforms with their modular architecture have the inherent challenge to maintain an optimal infrastructure [6], [9], [12]. Following Olleros [6] the optimal core is the smallest core that allows for the greatest functionality of the platform.

While previous research studied approaches to achieve high levels of innovation in the form of contributions, the aspect of the lean core was not considered as a central aspect. In this regard, aspects such as platform openness [8], [13], [14], governance mechanisms [15], [16], and design of development resources [10], [17], [18] were discussed. To achieve greater levels of innovation with the smallest possible platform core, this study addresses the aspect of feature removal on software platforms. Platform owners are interested in how to achieve the smallest platform core that serves this purpose. Feature removal as part of platform optimization can contribute to that.

Achieving a lean core can be either realized by (a) not introducing additional features to the platform core or (b) by regular removal of features from the platform core. Stopping to introduce is considered as unlikely as innovation is an important factor of platforms competitiveness [4]. Moreover, many studies proved platform owners to regularly introduce new features to the platform core [3], [19]. Furthermore, platforms owners also introduce features similar to those provided by platform complementors [7], [20]. The removal of features on the other hand, can be used to exclude features that do not serve the purpose of the platform (anymore) and are of less importance [11]. For example, features that are provided in superior form by complementors may also be removed from the platform core.

Software platforms serve two sides. That is the group of users for which platforms offer functionality (user-related) and the group of developers for which boundary resources are provided to allow for complements in the form of platform modules (developer-related). The removal of features may occur for both types of features. The corresponding implications vary. For instance, the removal of user-related features may result in missing functionality for users. On the other hand, the removal of developer-related features may result in extensions to stop working. In this regard, feature removal is relevant for the platform ecosystem participants. As such, ecosystem

978-1-7281-3401-7/18/$31.00

2021 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)

stakeholders are interested whether platform owners remove features during software platforms evolution as this may affect their usage of and contribution to the platform [21]. To address this aspect this study aims to explore:

RQ1: Do software platforms remove core features during their evolution?

Assuming platform owners to remove features from their platform as part of their platform management, the question why platform owners do so is of interest. The underlying motivations are of interest for ecosystem participants given that different implications result for them. While feature removal from a maintenance perspective may lead to increased development efforts for complementors, removing features to foster innovation results in decreased competition and additional demand. As such this contribution aims to explore:

RQ2: Why do platform owners remove core features from software platforms?

To explore the research questions, two research methods are used. First, a literature survey is used to integrate findings from related literature. Second, a case-study approach is used to explore the occurrence of feature removal in the browser domain. Web browsers have become powerful platforms, offering value to a variety of customers, ranging from its main users to independent developers that create third-party add-ons. As such, they are a prime example of platform-centric ecosystems in modern software [7], [22], [23]. Modern web browsers can offer specialized functionality to its users via third-party addons, more generally described as 'modules' [1] that are clearly distinguishable from its platform's core. Concerning innovation activities, previous research has shown that web browser owners often expand their core by including features that have been previously only accessible via third-party addons [7], [20]. Concerning the research interest, it is to be explored if feature removal occurs on browser platforms. Mozilla Firefox is used for the case study analysis. Firefox was released in 2004 as an open-source and community-driven browser. The main competitor of Firefox is Google's Chrome and Chromium-based web browsers. Together, they account for the majority of users on the web.

The contribution is structured in six sections. The following section briefly covers related literature. In section three, the literature survey approach and analysis regarding feature removal is presented. Section four presents the case study on feature removal. Section five discusses the results and section six concludes the paper.

II. Related Literature

This section briefly presents central concepts and literature for this contribution. The aspect of feature removal can be situated in the research domain of software platforms. Especially, the group of external software platforms [24] are known for their extensibility by complementors that result in dependencies [9]. Following the notion of Tiwana et al. [1] a software platform is understood as: "The extensible codebase of a software-based system that provides core functionality shared by the modules that interoperate with it and the interfaces through which they interoperate".

Therefore, software platforms make use of the associated platform ecosystem. The platform ecosystem being related to an open platform infrastructure is of importance for its growth and competitiveness [25], [26]. Following Gawer and Cusumano [27], a platform ecosystem is understood as: "The network of innovation to produce complements that make a platform more valuable". A software platform ecosystem mainly consists of three roles: the platform owner, complementors, and customers. The platform owner as the central entity serves both sides, i.e. complementors and users. Complementors build their modules on top of the existing platform. To do so, platform owners provide boundary resources for application development. In this regard the notion of Ghazawneh and Henfridsson [15] is followed whereby, boundary resources "[…] typically consist of a software development kit (SDK) and a multitude of related APIs" [10]. For instance, browsers platforms can provide add-on developers with specialized APIs reducing their cost of development [28].

For the group of platforms users, platforms offer functionality through the platform core that can be extended by modules provided by third-party developer. The close relationship between the three main players results in mutual dependence of the key stakeholders. In this regard, the importance of network effects, is to be mentioned [29]. As such, the competition among platforms largely depends on their platform ecosystem and the ability to attract complementors and users to their platform [8], [26].

The provision of third-party functionality in the context of software platforms is realized achieved through capsuled code-fragments in the form of modules. Following [1] a module is understood as „An add-on software subsystem that connects to the platform to add functionality to the platform". In this regard, the term of functionality can be further separated in a group of features that is combined through a software module.

As defined by ISO/IEC/IEEE26515:2011 a "*software feature* is a functional or non-functional distinguishing characteristic of a system, often an enhancement to an existing system". Since software platforms are comprised of a stable core surrounded by variable peripheral components [30], these value-providing features can be part of the core, or the surrounding modules [26]. Mozilla Firefox as an open software platform allows independent third-party developers to create these modules [22], that serve as distinct parts of the platform, and can be designed and implemented independently [1], [31]. Furthermore, the architecture of the platform enables the platform owner to add new features, modify existing ones, or remove them completely [30]. The paper focusses on features as functional characteristics that were part of the platform core and removed afterwards.

Differentiating between user features and developer features, user features in web browsers include UI-elements, accessibility options, translations, cross-device syncing, or third-party addons. Examples for developer features would mainly include the access to developer tools in form of debuggers, APIs, or interfaces, but also the support of external APIs and protocols.

In this regard, the *removal of features* is meant as the complete removal from the platform's core, rather than just disabling it temporarily, changing its position, or obscuring its

978-1-7281-3401-7/18/$31.00

existence from the user. The sole announcement of a feature deprecation will not be counted as feature removal.

III. Literature Review

As a first step towards a better notion regarding the occurrence of and rationale for feature removal on software platforms, a literature review is conducted.

A. Methododical Approach

A systematic literature review following the guidelines of [32], [33], and [34] is conducted. As a first step, the review scope is defined [33]. The topic of feature removal is situated in the research area of software platforms [1]. Moreover, the focus might be narrowed to the group of extensible open software platforms, since these allow for contribution of third parties [24]. In contrast to closed software platforms the implications of feature removal may lead to different reactions of platform ecosystem participants.

The literature screening revealed that the aspect of feature removal and discontinued features has not been considered so far. As there was little to none studies on the removal of software platform features and its reasons, it was necessary to approach the topic from a broader theoretical angle.

To identify potentially relevant articles four databases were queried for literature retrieval. These are: Scopus, Google Scholar, JSTOR, and Microsoft Academic. The search terms composed of the keyword combination were designed more broadly, given that the aspect of feature removal is not present as the literature screening revealed. To query the four databases the search term: *"software platform" AND ("third-party" OR "extensions" OR "third party") AND ("functions" OR "features")* was used. As the domain of software platforms allowing for third-party contributions emerged around 2008, literature studies from 2008-2020 were included in the literature survey.

Following the initial search, 396 records were identified (JSTOR: 71, Google Scholar: 232, Scopus: 58, Microsoft Academic: 35). After duplicate removal, the exclusion criteria were applied (excluding thesis, pre-prints). Resulting in 279 papers. The remaining paper were screened for their relevance (using a title and abstract review) which resulted in 23 papers for detailed screening. Finally, 13 papers were identified to be relevant for this research. During literature analysis and screening, nine additional papers were added to the group of potentially relevant studies. These include highly influential papers, as well as the results of the for- and backward search.

B. Results

The literature analysis revealed the aspect of feature removal to be uncommon among software platform literature. Therefore, the identified studies were reviewed for closely related topics. Table 1 illustrates the concepts that cover aspects of relevance feature removal. Among these are the aspect of platform openness as central foundation for complementors contribution.

TABLE I. Literature Analysis Results

Source	Concepts			
	Platform Innovation	***Platform Openness***	***Comple-mentors***	***Platform Core***
Bender et al. 2019 [7]	X	X	X	X
Boudreau 2010 [8]	X	X	X	-
Choi et al. 2019 [13]	X	X	X	-
Khomh et al. 2012 [36]	-	X	-	X
Olleros 2008 [6]	X	X	X	X
Parker et al. 2018 [14]	X	X	X	X
Tåg 2009 [37]	-	X	X	-
Tan et al. 2020 [28]	X	X	X	-
Tiwana et al. 2010 [1]	X	X	X	X
Rickmann et al. 2014 [29]	-	-	X	-
Zhou et al. 2018 [23]	X	X	X	X
Zhu 2019 [20]	X	-	X	X

Most studies consider the relationship between platform owners and complementors, as well as the platform openness and innovation, which are all interconnected [6], [8], [35]. Barely half of them touched upon the concept of the platform core. Regarding feature removal, the platform core as the most central element, is of great importance.

1) Platform Innovation

Platform innovation is acknowledged to be a critical success factors for software platforms [4]. Software platforms in their role as innovation platform serve as the basis for complements to be built upon the platform [24]. As such platform innovation is composed of the functionality provided by the platform core as well as contributed third-party complements [7]. Thereby, platforms are able to provide more innovation than a single entity could achieve alone [2]. Innovation from the platform owner, as well as the complementors standpoint is essential for the growth of a software platform.

Following Nambisan [25] digital innovation management is understood as "the practices, processes, and principles that underlie the effective orchestration of digital innovation" [25]. Platform owners are responsible to conduct platform innovation management. Effective orchestration includes the decision which entity (platform owner through the core or third parties through contributions) should provide which functionality. While the introduction of functionality in terms of innovation has been discussed [4], [8], [10], the aspect of feature removal has not. The aspect of feature removal is part of platforms innovation activities.

2) Platform Openness

Just as much as innovation, the openness of a platform plays an important role, as it has been shown to be linked to a platform's innovation and growth [8] (Boudreau 2010). Open software platforms allow for contributions from third parties. In contrast to that, closed software platforms limit or restrict the

possible external innovation as a driver for growth [13], [24], [38].

Besides the strict distinction between open and closed, more granular levels of openness are distinguished [8], [39]. Software platforms allow to access platform resources through related boundary resources (see complementors section). Previous studies reveal more open platforms to show more growth and variety in their offerings, while more closed platforms show less and less diverse offerings [8]. Yet, platform openness doesn't always need to go hand in hand with platform growth, as is the case with Apple's iOS [35].

Concerning discontinued features, platform openness on a general level is important since external platforms may allow to replace discontinued features through contributed modules. In this regard, discontinuing features may open a market for third-party contributors. This signal may motivate third-party contributors to offer complements to replace the discontinued features. As such, feature removal may lead to increased innovation and competition among developers [40]. While entering fields in which the platform owner is active is less attractive for contributors [7], [20], [23], domains which platform owners leave may be of interest. However, this may only apply for features that are still relevant for platform users. From a complementors perspective, feature removal may be seen as a signal from the platform owner [41].

3) Software Platform Complementors

Software platforms as multi-sided platforms are typically surrounded by the platform ecosystem. Important stakeholders are: the platform owner as the provisioning entity, the platform user as the demanding entity, and the complementors as the group of contributors [26]. This close relationship between the three main players results in network effects, meaning that the value and attractiveness of the platform depends on a good relationship between the players, and rises the more it is used by complementors and users [29]. As such, software platforms themselves serve at least two sides. For the group of platform users, platforms offer core functionality which can be extended by the available third-party applications.

For the group of developers, platforms provide boundary resources that allow for external innovation in the form of contributions [10]. Boundary resources refer to "the software tools and regulations that serve as the interface for the arm's-length relationship between the platform owner and the application developer" [15]. They note that in "software platform settings, such (boundary) resources typically consist of a software development kit (SDK) and a multitude of related APIs" [10]. While SDKs are used during development, the group of APIs allows complements to access platform (core) resources during application usage.

In the realm of web browsers providing APIs for third-party developers is a popular strategy, since providing "a good API makes the platform more modular by providing well-defined interfaces for the application to the platform, hence reducing development costs for the third-party content provider" [28]. Moreover, contributors are known to consider SDK quality during platform selection [21]. Moreover, previous studies found increased usage of boundary resources to allow for enhanced success and customer satisfaction [18].

4) Platform Core

The platform core serves as the main centralized part of a software platform and should, by design, provide value for its customers and allow complementors to build their previously mentioned 'modules' on top of it [35]. [6] describes the optimal core as follows: "The optimal platform core is the leanest core capable of eliciting from an innovative market or community all the missing elements to bring the platform to its highest degree of functionality".

While some call for a lean core to maximize scalability and evolvability [6], [31], studies identified the platform core to expand over time. Increased functionality may be the result of innovation activities of the platform owners as well as entering complementary markets [20]. This entry into complementary spaces and expansion of a software platform's core is also referred to as platform coring and can have positive effects for the platform. Depending on the type of functionality user-focused or developer-focused different implications emerge [3]. Still, platform owners must think about the balance between features provided by the platform core and those offered by third-party developers, as such entry into complementary spaces can often be seen as hostile [7] and cause complementors to be less willing to innovate [20], [35]. From a theoretical angle, the core features that platforms provide should be designed to be relevant for the mass of users [3], [6].

Through complementary market entry platform owner enter contributors market spaces. As such, feature removal can be considered as the opposite. For user-related features in specific, feature removal may open markets for complementors as core functionality is reduced.

C. Motivations for Feature Removal

As a result of the literature analysis, different motivation could be identified for feature removal from a theoretical standpoint.

First, maintaining a *lean core* may be the result of *architectural considerations* on platform architecture [6]. Feature can be removed over time to achieve a lean core. Following [6], the optimal core allows for the highest degree of functionality while being as small as possible to serve this purpose. Following this idea, the platform core should serve as the foundations for complements while not containing all the features themselves. More specifically, the focus should be on developer-related boundary resources to allow for complements and access of platform resources rather than extensive user features and a closed platform.

Second, *platform hygiene* might be a motivation for feature removal [3], [42]. Similar to the platforms themselves, user requirements change over time. As such, features may be outdated from a functional perspective or receive less attention from users. Thereby, platform owner may be interested to remove features that are not employed by users given that their maintenance is relatively costly for a small user base.

Third, *stimulating innovation* may be a motivation for feature removal. Given that platform owners are privileged with regard to access to platform resources on the one hand and customer access on the other, competing with core functionality may be unattractive for complementors [20]. Given that, core

functionality is usually available for users directly, extended functionality through complements requires search and setup effort [18], [43]. Through feature removal, platform owners open up a formerly unattractive market for complementors. Through feature removal platform owner might signal that related product spaces are not covered by platform owner anymore [41]. This might stimulate innovations by complementors. Even though they might compete amongst them, related competition is among equal developers. As such, discontinuing features can stimulate platform innovation.

Fourth, feature may be removed if *better alternatives* exist. While that the platform owner has to provide a wide range of features, contributors add highly specialized domain features through their contributions. Complementors were found to be highly innovative [4]. Moreover, many complementors may compete among customers by providing similar features through their contributions [40]. This may result in the situation, that complements provide better features than the platform core. As a result, the platform owner may consider giving up related functionality. Especially, if the features are of minor strategical importance. In this case, feature removal may be the result of better alternatives.

Fifth, feature removal may be motivated by *focussing resources*. Platforms owners have to handle many tasks in their role as platform leader [27]. Moreover, the importance of features may vary over time. While especially innovative features may be introduced for showcase purposes, the motivation to provide related features may diminish over time. This may lead to feature removal. Furthermore, more attractive other features may be of interest for the platform owner which may lead to a focus of resources on related attractive fields [44]. Assuming powerful, but still limited resources for the platform owner, this may lead to feature removal to focus on more attractive domains.

Table 2 provides an overview of motivations for feature removal and related rationales.

TABLE II. MOTIVATIONS FOR FEATURE REMOVAL

Motivation	Rationale	Concepts
Lean core	Maintain a lean core as a basis for innovation.	[6], [31]
Platform hygiene	Remove less important features from the platform core.	[3], [42]
Stimulate innovation	Stimulate innovation by opening up market for contributors	[41]
Better alternatives	Remove features that are not necessary, since better alternatives exist.	[40]
Resources focus	Focussing resources on attractive and important domains.	[44]

IV. CASE STUDY

In addition to the results of the literature review, a case study is used to approach the research questions. To identify if feature removal occurs (RQ1) and if so for which reasons features are removed (RQ2), the example of Firefox is used.

A. Method

The method of a descriptive single case study about Mozilla Firefox was chosen, as it will allow to gather data on an appropriate level of detail and interpret it given the results of the literature review [45].

Mozilla Firefox was chosen for the case-study as it fulfils the requirement of a software platform [22]. The extensible browser allows for the contribution of third-party modules termed add-ons [1]. Users are able to gather related add-ons through the platform marketplace [38].

The browser used for the case study is required to be actively developed in order to identify feature changes. In this regard, Firefox is known for their regular update policy that frequently introduces new features. Moreover, Firefox is well-suited for the case study as Mozilla is known for their active communication to and engagement with the developer community. An open information policy is useful to gather insights regarding the reasons for feature removal (RQ2). Considering the browser market share, Firefox, Safari and Chrome account for the majority of desktop users. Prior studies already considered Firefox as a software platform [7], [22].

B. Data and Classification

To ensure the validity of the collected data (Baxter et al. 2008), only official statements from Mozilla were used to identify removed features and classify the reason for feature removal. These include: the newest release notes, Mozilla's own blog, or threads on Bugzilla regarding suggestions and explanations for the removal of certain features.

All Mozilla Firefox stable desktop releases, up to the current (as of 31.07.2020) version 79.0 were systematically searched for removed features using the official release notes, as well as Bugzilla threads that resulted in the removal of a feature. Removal of one or multiple features may, but not have to, occur with every new version. 45 removed features could be identified. For four removed features no explanation for their removal provided by Mozilla was given and will thus be ignored.

To address the research interest concerning the reasons for feature removal, the removed features were classified in eight different categories. Those were inductively developed by screening and classifying related announcements. Features were removed for security concerns (Security), a previously deprecated feature that has already been replaced by another (Deprecation), a feature deemed obsolete due to other features fulfilling the same purpose or the feature not having any more advanced value and thus providing the user with no additional value (Obsolete). Further features are removed that have not been used by complementors or users of the browser (Unused, measured by Mozilla in *x%* of the entire user base), features slowing down the performance of the browser (Performance), features that are too costly to maintain (Costs), features that are not working as intended (Bugs), and competitor pressure, stemming from Google Chrome. As there are several channels that Mozilla uses to announce these removals (Blog, Bugzilla, Support, Release Notes), some features can have more than one reason for its removal named. Still, if only a single reason was provided, it should be assumed that this doesn't exclude any other underlying reasons for the removal of a certain feature.

Additionally, data was gathered from Bugzilla when it was found to more accurately explain the rationale behind the removal of certain features, as well as showcasing the decision-

making process of Firefox' main contributors. These threads mostly start with suggestions to remove a certain feature, after which multiple contributors discuss, whether such removal is justified. Alternatively, threads start with contributors or community members asking about already removed features and get the explanation retroactively.

C. Results

All removed features were documented with the respective version number and the reason for its removal. The feature list with the classification can be found in Table 3. Overall, almost every full release cycle (for Firefox being at around 4 weeks), Mozilla removes a feature it deems to not be needed anymore using the reasons listed in Table 3 and Figure 1. The three most frequently named reasons for removing a feature are security concerns, deprecated, or unused features, accounting for over half of all the reasons.

Concerning *unused user features*, Mozilla's has been shown to wait until the usage for a given feature is low enough to not impact their overall user base, before deciding to remove a feature. Still, this approach is only used if there are no other reasons for the removal. As far developer-related features go, supplying complementors with alternative tools to those that were deprecated is an important part of Mozilla's community management and usually included in the removal announcements as helpful information for third-party developers.

However, many of those features can be considered 'dead on arrival', only increasing the core's complexity and increasing costs. A prime example is the 'asynchronous plugin initialization' that was developed since 2014 and added to Mozilla Firefox in 2015 in version 40.0, yet didn't gain popularity due to several bugs, and was ultimately removed three years later, while providing almost no value to users during that time.

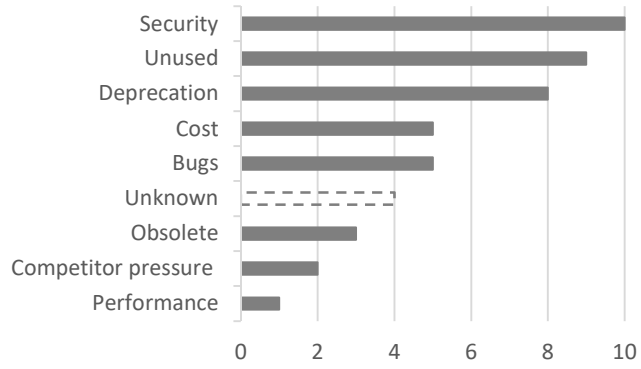

Fig. 1. Overview of Reasons for Feature Removal on Firefox

The security aspect is a major concern for the Mozilla Foundation. Security is part of their mission statement to making the web an accessible resource. As such, the focus on security concerns regarding removed features are reasonable.

TABLE III. FEATURE REMOVE FROM MOZILLA FIREFOX

Ver.	Feature	Reason	Type
3.0	predefined add-ons whitelist	Unknown	User
21.0	E4X-Support	Deprecation	Dev
23.0	<blink> element	Obsolete	Dev
23.0	"Enable JavaScript", "Load images automatically" & "always show the tab bar" options	Obsolete	User
24.0	Support for loading Sherlock files	Deprecation	Dev
24.0	"Revocation Lists" feature	Performanc e	User
28.0	SPDY/2 Support	Deprecation	other
29.0	tabs-on-bottom	Costs	User
31.0	CAPS infrastructure removed for specifying site-specific permissions	Unknown	User
32.0	trust bits for 1024-bit root certificates	Security	User
33.0	JavaScript Debugger Service	Obsolete	Dev
33.0	Proprietary window.crypto properties/functions	Security	User
35.0	Proprietary window.crypto properties/functions	Security	User
36.0	"-remote" option	Unused	Dev
38.0	Support for autocomplete=off for username/password fields	Competitor pressure	User
39.0	"-remote" option	Unused	Dev
39.0	Support for insecure SSLv3 for network communications	Security	User
39.0	Support for RC4 except for temporarily whitelisted hosts	Security	User
41.0	Support for XPCOM components in extensions	Deprecation	Dev
44.0	"ask me everytime" cookie option	Bugs	Dev
44.0	Support for RC4 decipher	Security	User
45.0	Tab Groups (Panorama)	Unused	User
47.0	Firefox User Extension Library	Unused	Dev
47.0	"click-to-activate" plugin whitelist	Deprecation	User
49.0	Firefox Hello	Unused / Costs	User
49.0	Support for OS X- 10.6-10.8 and SSE processors	Costs	User
50.0	Support for libavcodec older than 54.35.1	Security	User
51.0	SPDY/3.1 Support	Competitor pressure	other
51.0	(be) locale	Unused	User
52.0	Battery Status API	Security	Dev
52.0	Netscape Plugin API-Support	Deprecation	User
53.0	Support for 32-bit Mac OS X	Costs	User
53.0	Support for processors older than Pentium 4 & AMD Opteron on Linux	Costs	User
56.0	Asynchronous plugin initialization	Bugs	User
57.0	Toolbar Share button	Unknown	User
62.0	Description field for bookmarks	Unused / Bugs	User
63.0	Never check for updates" option	Security	User
63.0	"Open in Sidebar" feature	Unknown	User
67.0	Upload and Sharing of screenshots via Firefox Screenshots server	Bugs	User
68.0	unmaintained translations	Unused	User
70.0	Always active" feature for flash plugin content	Security	User
70.0	Aliased theme properties	Deprecation	Dev
77.0	Support for blocking images from individual domains	Unused / Bugs	User
77.0	browser.urlbar.oneOffSearches preference	Unused	User
78.0	TLS 1.0 and TLS 1.1	Security / Deprecation	User

Moreover, features were classified regarding their intended target group being either platform user or third-party developer. The related classification can be found in Table 3. Most features (32) removed are targeted towards the group of users.

Multiple examples exist of planned or already delivered feature removals which were later rolled back due to causing bugs on the platform (e.g., proprietary window.crypto properties/functions). This illustrates the need for a lean core with loosely coupled modules around it as described in [6]. Software platforms should thus leverage the power of modularity in their design, to prevent changes in one part of the program to cause problems in another [46], and enable the seamless exchange of existing modules and the integration of new ones.

TABLE IV. MOTIVATIONS FOR FEATURE REMOVAL

Motivation	Rationale	Example
Lean core	Maintain a lean core as a basis for innovation.	v24.0 Support for Sherlock files v52.0 NPAPI support
Platform hygiene	Remove less important features from the platform core.	v23.0 features except <blink> (see table 3)
Stimulate innovation	Stimulate innovation by opening up market for contributors	
Better alternatives	Remove features that are not necessary, since better alternatives exist.	v45.0 Tab Groups Feature
Resources focus	Focussing resources on attractive and important domains.	v21.0 E4X-Support

Among the removed features, different motivation could be identified in Firefox case study. Removing features to obtain and maintain a *lean core* was identified as a frequent motivation. For instance, the support for Sherlock files in version 24[1]. More recent examples involve the NPAPI plugins in version 52[2].

Mozilla removed different features in version 23 as the incorrect use of the features may cause the browser to stop working properly[3]. Easy usage and proper function is important for Mozilla [7]. By removing potential conflicting features, Mozilla maintained platform hygiene.

Mozilla also removed features such as the tab grouping feature and directly mentions the availability of third-party add-ons to provide that functionality. Moreover, a dedicated help page presents different options[4].

Regarding the motivation of resource focus, the example of the E4X-Support is to be mentioned. The following discussion statement exemplifies the motivation "E4X slows down our development and increases the security attack surface"[5].

V. DISCUSSION

A. Theoretical Contributions

Given that feature removal is unexplored in terms of practical evidence on software platforms, the results are among the first to indicate the actual occurrence of feature removal. By use of the practical case-study the paper addresses the first research question regarding the occurrence of feature removal. The results indicate that the Firefox platform regularly removes features from the platform core. Moreover, the results indicate that user-related as well as developer-related features are removed. In this regard, the study contributes by discussing the different effects depending on which type of functionality (user or developer) is removed.

As a second contribution, the study provides insights in the reasons and motivations for feature removal. By means of a literature survey, motivations for the removal of platform core features are identified and systematized. From a theoretical standpoint, different motivations were identified. These include motivations being related to the modular structure of software platforms (lean core), the importance of features over time (platform hygiene, resource focus), as well as competition-related considerations (alternatives). Moreover, platform owners may use feature removal to foster external contributions (stimulate innovation).

Apart from theoretical considerations, reasons for feature removal were identified through the case study. Addressing RQ2, the paper contributes by highlighting different motivations and their actual occurrence as well as their relative importance. The analysis reveals that unused features, security issues as well as feature deprecation is among the most frequent reasons for feature removal.

As a third contribution, the contribution points out the different implications depending on the type of functionality removed. Table 5 provides an overview of the different effects depending on the type of functionality being removed. Related implications and the parties affected vary. The removal of user-related features may be substituted through third-party complements being available on the platform. In contrast, the removal of developer-related features is not similarly substitutable given that only the platform owner is able to allow for the access of platform core features and is responsible to provide related boundary resources. In this regard the removal of developer-related features is to be handled with care given that the demotivation of developer might have negative consequences for further contributions [17].

TABLE V. FEATURE REMOVAL EFFECT DEPENDING ON FEATURE TYPE

Feature	User-related	Developer-related
Description	Features employed by users directly	Features that serve developer to realize complements (e.g. extensions, modules)
Example	Firefox Hello (audio and video calls directly in the browser)	Battery Status API (information on the power status of the main battery)
Effect	Features are no longer available to user in the core application directly	Features are no longer available to be used by complements
Implication (if used / required)	Seach for and use of third-party alternatives (if exists)	Need to realize similar feautes within their module context (if possible)

[1] https://mail.mozilla.org/pipermail/firefox-dev/2013-April/000329.html
[2] https://blog.mozilla.org/futurereleases/2015/10/08/npapi-plugins-in-firefox/
[3] https://bugzilla.mozilla.org/show_bug.cgi?id=851702
[4] https://support.mozilla.org/en-US/kb/tab-groups-removal
[5] https://groups.google.com/g/mozilla.dev.tech.js-engine/c/yYQyMCcMf-0/discussion

B. Practical Implications

The implications are differentiated according to the platform ecosystem stakeholders. For the group of platform owner, the case study revealed to remove features that are of minor importance. Given that feature removal results in decreased value from a functional perspective, the effect is not assumed to be perceived directly positively. Considering however that in most cases Mozilla usually ensures only features with either a low user base or security vulnerabilities are removed from the core, it is unlikely that the removal of features would be met with much disapproval by the respective participants on the platform. In this regard, platform owners are well-advised to focus on features of minor importance during their removal.

Given the structural similarities to complementary market entry and platform coring, related implications can be partly adopted for feature removal [3], [42]. Platform owners were found to focus on complements of high popularity for platform inclusion [7], [20]. Considering feature removal as an inversion to complementary market entry and platform coring, as features are removed from the core, the focus on less popular features is likely. Moreover, the results apply from a content perspective for instance for the aspect of security concerns. It is to be expected, that keeping security vulnerabilities as part of the core would negatively impact the relation to the platform's users and complementors. Furthermore, as less complex and more generic features tend to be cored [7], higher complexity of core features is more likely to be removed due to bugs or the associated costs, just like more specialized features tend to result in lower user numbers that in turn result in the removal of said features.

The group of third-party developers are most likely to be affected by feature removal. The results reveal several developer-related features are to be removed from the Firefox platform. Complementors are required to adapt their extensions accordingly to ensure their operation. Given that platforms are highly competitive environments [40], developers are well-advised to adapt their extensions prior to upcoming platform changes, e.g. feature removal to ensure to retain their user base [47]. Especially within a platform environment, switching costs are considered to be relatively low [48].

C. Limitations

This study is subject to several *limitations*. Regarding the case study, the focus on a single, yet prominent platform limits generalizability. While a single case study is assumed to be reasonable as a first approach towards exploring the phenomenon of feature removal, the platform chosen is specific in some ways. Firefox as an open-source browser is known for their engagement and collaboration with the developer community [7], [22]. As such, it is questionable in how far the results concerning the reasons for feature removal are applicable to commercial platform environments such as for example mobile device platforms [9], [10]. The literature review conducted was rather focused in the scope of literature considered. A broader focus, for instance by considering research from software development and economics literature might have revealed additional reasons for feature removal.

Finally, the combination of reasons for feature removal and the underlying motivation were hypothesized by the authors based on the results of the literature review. While the actual motivation for feature removal remains to be further explored, likely motivations were assumed for illustration purposes.

D. Future Research

Several aspects remain open for future research. To achieve greater levels of generalizability, future studies may focus on *more and different* types of software *platforms* to explore the phenomenon of feature removal. Aside from similarities and differences among platforms, general evolution patterns as for instance the removal of features within platform domains could be identified. Apart from theoretical considerations, the *perspective of the stakeholders* involved is to be explored. First and foremost, this includes the perspective of the platform owner as the entity to remove features. Moreover, the position of the affected parties, i.e., platform user (for user-related features) and third-party developer (for developer-related features) is to be explored to gain a better understanding on the effects of feature removal. Moreover, specific types of feature removal are to be identified during future explorations. While feature removal was considered on a functional level, platforms were found to discontinue large parts of boundary resources. For instance, Firefox switched their development resources to the WebExtensions standard (with version 57 in November 2017). While this allows for cross-platform availability of related add-ons, existing add-ons needed to be adopted to the new standard in order to be further available on the browser platform.

VI. Conclusion

Software platforms allow for the extension of features by third-party complementors. Innovation is an important success factor regarding platform attractiveness for users and contributors. While previous research focused the introduction of new features as part of innovation activities, the phenomenon of feature removal and discontinued features on software platforms was not addressed.

To explore the phenomenon and motivations for feature removal on software platforms, a review of recent literature is provided which identified different motives for feature removal, i.e., lean core, platform hygiene, stimulate innovation, better alternatives, and resources focus. To illustrate the existence of and motivations for feature removal, a case study of the browser platform Mozilla Firefox is conducted.

The results reveal feature removal to regularly occur on browser platforms for user- and developer-related features. During the 16 years of operation, Mozilla Firefox has continuously removed features from its platform's core. Mozilla removed 45 features that were either deemed unused, deprecated, too costly to maintain, obsolete, posing a security risk for its users, not working as intended, or resulting from competitor's pressure. Implications for complementors and users are discussed. In this regard, the aspect in how far feature removal contributes to platform success and which degree of feature removal is optimal to foster innovation, remains to be explored.

References

[1] A. Tiwana, B. Konsynski, and A. Bush, 'Platform evolution: coevolution of platform architecture, governance, and environmental dynamics', Inf. Syst. Res., vol. 21, no. 4, pp. 675–687, 2010.

[2] T. Eisenmann, G. Parker, and M. Van Alstyne, 'Platform envelopment',

Strateg. Manag. J., vol. 32, no. 12, pp. 1270–1285, 2011.
[3] B. Bender and N. Gronau, 'Coring on Digital Platforms-Fundamentals and Examples from the Mobile Device Sector.', presented at the International Conference on Information Systems, Seoul, South Korea, 2017.
[4] A. Kankanhalli, H. Ye, and H. H. Teo, 'Comparing Potential and Actual Innovators', MIS Q., vol. 39, no. 3, pp. 667–682, 2015.
[5] N. Haile and J. Altmann, 'Value creation in software service platforms', Future Gener. Comput. Syst., vol. 55, pp. 495–509, 2016.
[6] X. Olleros, 'The lean core in digital platforms', Technovation, vol. 28, no. 5, pp. 266–276, 2008.
[7] B. Bender, C. Thim, and F. Linke, 'Platform Coring in the Browser Domain-An Exploratory Study', presented at the International Conference on Information Systems, Munich, Germany, 2019.
[8] K. Boudreau, 'Open platform strategies and innovation: Granting access vs. devolving control', Manag. Sci., vol. 56, no. 10, pp. 1849–1872, 2010.
[9] M. de Reuver, C. Sørensen, and R. C. Basole, 'The digital platform: a research agenda', J. Inf. Technol., vol. 33, no. 2, pp. 124–135, 2018.
[10] B. Eaton, S. Elaluf-Calderwood, C. Sørensen, and Y. Yoo, 'Distributed Tuning of Boundary Resources: The Case of Apple's iOS Service System', MIS Q., vol. 39, no. 1, pp. 217–243, 2015.
[11] A. Janes and V. Lenarduzzi, 'Towards an Approach to Identify Obsolete Features based on Importance and Technical Debt', in Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2020, pp. 389–393.
[12] A. Gawer and R. Henderson, 'Platform owner entry and innovation in complementary markets: Evidence from Intel', J. Econ. Manag. Strategy, vol. 16, no. 1, pp. 1–34, 2007.
[13] G. Choi, C. Nam, and S. Kim, 'The impacts of technology platform openness on application developers' intention to continuously use a platform: From an ecosystem perspective', Telecommun. Policy, vol. 43, no. 2, pp. 140–153, 2019.
[14] G. Parker and M. Van Alstyne, 'Innovation, Openness, and Platform Control', Manag. Sci., vol. 64, no. 7, pp. 3015–3032, 2018.
[15] A. Ghazawneh and O. Henfridsson, 'Balancing platform control and external contribution in third-party development: the boundary resources model', Inf. Syst. J., vol. 23, no. 2, pp. 173–192, 2013.
[16] T. L. Huber, T. Kude, and J. Dibbern, 'Governance practices in platform ecosystems: Navigating tensions between cocreated value and governance costs', Inf. Syst. Res., vol. 28, no. 3, pp. 563–584, 2017.
[17] L. Xue, P. Song, A. Rai, C. Zhang, and X. Zhao, 'Implications of Application Programming Interfaces for Third-Party New App Development and Copycatting', Prod. Oper. Manag., vol. 28, no. 8, pp. 1887–1902, 2019.
[18] B. Bender, 'The Impact of Integration on Application Success and Customer Satisfaction in Mobile Device Platforms', Bus. Inf. Syst. Eng., no. 62, pp. 515–533, 2020.
[19] J. Foerderer, T. Kude, S. Mithas, and A. Heinzl, 'Does platform owner's entry crowd out innovation? Evidence from Google photos', Inf. Syst. Res., vol. 29, no. 2, pp. 444–460, 2018.
[20] F. Zhu, 'Friends or foes? Examining platform owners' entry into complementors' spaces', J. Econ. Manag. Strategy, vol. 28, no. 1, pp. 23–28, 2019.
[21] H. J. Kim, I. Kim, and H. Lee, 'Third-party mobile app developers' continued participation in platform-centric ecosystems: An empirical investigation of two different mechanisms', Int. J. Inf. Manag., vol. 36, no. 1, pp. 44–59, 2016.
[22] A. Tiwana, 'Evolutionary Competition in Platform Ecosystems', Inf. Syst. Res., vol. 26, no. 2, pp. 266–281, 2015.
[23] G. Zhou, P. Song, and Q. Wang, 'Survival of the fittest: understanding the effectiveness of update speed in the ecosystem of software platforms', J. Organ. Comput. Electron. Commer., vol. 28, no. 3, pp. 234–251, 2018.
[24] A. Gawer and M. A. Cusumano, 'Industry platforms and ecosystem innovation', J. Prod. Innov. Manag., vol. 31, no. 3, pp. 417–433, 2014.
[25] S. Nambisan, K. Lyytinen, A. Majchrzak, and M. Song, 'Digital Innovation Management: Reinventing innovation management research in a digital world.', MIS Q., vol. 41, no. 1, 2017.
[26] P. Song, L. Xue, A. Rai, and C. Zhang, 'The ecosystem of software platform: A study of asymmetric cross-side network effects and platform governance', MIS Q., vol. 42, no. 1, pp. 121–142, 2018.
[27] A. Gawer, M. A. Cusumano, and others, Platform leadership: How Intel, Microsoft, and Cisco drive industry innovation, vol. 5. Harvard Business School Press Boston, MA, 2002.
[28] B. Tan, E. G. Anderson Jr, and G. G. Parker, 'Platform pricing and investment to drive third-party value creation in two-sided networks', Inf. Syst. Res., vol. 31, no. 1, pp. 217–239, 2020.
[29] T. Rickmann, S. Wenzel, and K. Fischbach, 'Software ecosystem orchestration: The perspective of complementors', presented at the American Conference on Information Systems, Savannah, USA, 2014.
[30] C. Y. Baldwin and C. J. Woodard, 'The architecture of platforms: A unified view', in Platforms, markets and innovation, vol. 32, Edward Elgar Cheltenham, 2009.
[31] C. Y. Baldwin, K. B. Clark, K. B. Clark, and others, Design rules: The power of modularity, vol. 1. MIT press, 2000.
[32] A. P. Siddaway, A. M. Wood, and L. V. Hedges, 'How to do a systematic review: a best practice guide for conducting and reporting narrative reviews, meta-analyses, and meta-syntheses', Annu. Rev. Psychol., vol. 70, pp. 747–770, 2019.
[33] J. vom Brocke et al., 'Reconstructing the giant: On the importance of rigour in documenting the literature search process', presented at the European Conference on Information Systems, Verona, Italy, 2009.
[34] J. Webster and R. T. Watson, 'Analyzing the Past to Prepare for the Future: Writing a Literature Review', MIS Q., vol. 26, no. 2, pp. xiii–xxiii, 2002, doi: 10.1.1.104.6570.
[35] G. Parker, M. Van Alstyne, and X. Jiang, 'PLATFORM ECOSYSTEMS: HOW DEVELOPERS INVERT THE FIRM', MIS Q., vol. 41, no. 1, pp. 255–266, 2017.
[36] F. Khomh, T. Dhaliwal, Y. Zou, and B. Adams, 'Do faster releases improve software quality? an empirical case study of mozilla firefox', in IEEE Working Conference on Mining Software Repositories (MSR), 2012, pp. 179–188.
[38] J. Taag, 'Competing platforms and third party application developers', Commun. Strateg., no. 74, Art. no. 74, 2009.
[38] A. Ghazawneh and O. Henfridsson, 'A paradigmatic analysis of digital application marketplaces', J. Inf. Technol., vol. 30, no. 3, pp. 198–208, 2015.
[39] A. Benlian, D. Hilkert, and T. Hess, 'How open is this Platform? The meaning and measurement of platform openness from the complementers' perspective', J. Inf. Technol., vol. 30, no. 3, pp. 209–228, 2015.
[40] S. Kajanan, N. Pervin, N. Ramasubbu, K. Dutta, and A. Datta, 'Takeoff and sustained success of apps in hypercompetitive mobile platform ecosystems: An empirical analysis', presented at the International Conference on Information Systems, Orlando, USA, 2012.
[41] P. Hukal, O. Henfridsson, M. Shaikh, and G. Parker, 'PLATFORM SIGNALING FOR GENERATING PLATFORM CONTENT.', MIS Q., vol. 44, no. 3, 2020.
[42] W. Wen and F. Zhu, 'Threat of platform-owner entry and complementor responses: Evidence from the mobile app market', Strateg. Manag. J., vol. 40, no. 9, pp. 1336–1367, 2019.
[43] R. M. Müller, B. Kijl, and J. K. Martens, 'A comparison of inter-organizational business models of mobile app stores: There is more than open vs. closed', J. Theor. Appl. Electron. Commer. Res., vol. 6, no. 2, pp. 63–76, 2011.
[44] F. Zhu and Q. Liu, 'Competing with complementors: An empirical look at Amazon.com', Strateg. Manag. J., vol. 39, no. 10, pp. 2618–2642, 2018.
[45] R. K. Yin, 'Case study research: design and methods (ed.)', Appl. Soc. Res. Methods Ser., vol. 5, 2003.
[46] D. S. Evans, A. Hagiu, and R. Schmalensee, Invisible engines: how software platforms drive innovation and transform industries. Cambridge, Mass: MIT Press, 2006.
[47] L. Selander, O. Henfridsson, and F. Svahn, 'Capability search and redeem across digital ecosystems', J. Inf. Technol., vol. 28, no. 3, pp. 183–197, 2013.
[48] Y. Park and Y. Koo, 'An empirical analysis of switching cost in the smartphone market in South Korea', Telecommun. Policy, vol. 40, no. 4, pp. 307–318, 2016.