The Design of Open Platforms: Towards an Emulation Theory

Daniel Rudmark
University of Gothenburg
RISE Research Institutes of Sweden
daniel.rudmark@ait.gu.se

Rickard Lindgren
University of Gothenburg
rickard.lindgren@ait.gu.se

Abstract

The enrolment of third-party developers is essential to leverage the creation and evolution of data ecosystems. When such complementary development takes place without any organizational consent, however, it causes new social and technical problems to be solved. In this paper, we advance platform emulation as a theoretical perspective to explore the nature of such problem-solving in the realm of open platforms. Empirically, our analysis builds on a 10-year action design research effort together with a Swedish authority. Its deliberate change agenda was to transform unsolicited third-party development into a sanctioned data ecosystem, which led to a live open platform that is still in production use. Theoretically, we synthesize and extend received theory on open platforms and offer novel product and process principles for this class of digital platforms.

Keywords: Open platforms, data ecosystems, emulation theory, action design research

1. Introduction

While contemporary organizations increasingly rely on digital innovation influenced by external sources, such value-creation sometimes occurs without organizational consent. Recent well-known examples include early versions of the Xbox console, the robot dog AiboPet, the iPhone, and Tesla cars. This type of unsanctioned development has been coined outlaw innovation, which refers to "non-cooperative, non-consensual relationships in which the user may be unknown to the supplier and in which there is likely to be no free flow of information between the two parties" (Flowers, 2008, p. 178).

In this paper, we report a 10-year action design research (ADR) (Sein et al., 2011) effort together with the Swedish Transport Administration (STA). At the time of its inception, STA did not grant third-party developers access to railway-related real-time data. In this situation, multiple rail-related apps relying on scraping had emerged and they were used by hundreds of thousands of travelers. Instead of trying to curtail the outlaw innovation that was going on, STA rather perceived the problem as an opportunity to cultivate and harvest an emerging pool of hackers with adequate technological expertise.

To assist STA in its problem-solving, we embarked on an ADR process to develop and materialize "platform emulation" as a new theoretical perspective. As such, our perspective integrates and extends existing open platform literature (Brunswicker & Schecter, 2019; Karhu et al., 2018; Tiwana, 2014) by applying emulation theory (Hartman & Teece, 1990; Teece et al., 1997) to benefit from self-resourcing activities of third-party actors (Ghazawneh & Henfridsson, 2013). Emulation deals with situations where an organization utilizes an external model to develop an open platform that is not only compatible, but also equipped with superior capabilities. Open here implies that the same capabilities are offered to any possible user including its owner. This is enabled by a technologically extensible core complemented by governance processes that mediate its interactions with autonomous complementors engaged in joint data-driven value creation (Saadatmand et al., 2019). Accordingly, we seek to answer the following research question: *How can organizations emulate self-resourcing activities of third-party developers to design open platforms?*

2. Initial Theoretical Base

We draw on emulation as a theoretical lens to inform the design of open platforms, i.e., a class of digital platforms that is fully open in terms of its possible appropriation by diverse users (Eisenmann et al., 2009). Our design agenda essentially entails a search process by which a platform owner discovers an alternative way to introduce a compatible platform that embeds superior capabilities (Hartman & Teece, 1990). As such, its logic differs from that of platform imitation (i.e., forking) (Karhu et al., 2018), simply because it depends on resembling behavior rather than replicating another platform configuration. Consistent with the latest platform research (Saadatmand et al., 2019), pursuing platform emulation as a design agenda

URI: https://hdl.handle.net/10125/103088
978-0-9981331-6-4

requires paying close attention to received theory on governance mechanisms and technology architecture as well as the complex interaction between them.

2.1. Governance Mechanisms

In platform emulation, a platform owner's primary objective is to resemble behavior and equip a platform with superior capabilities. A key determinant when building such desirable capabilities is the related governance mechanisms. These mechanisms concern a platform owner's decisions about managing an ecosystem of complementors (Foerderer et al., 2019; Saadatmand et al., 2019). In particular, we elaborate on 1) what the platform offers in terms of search opportunities and 2) how an emulator chooses to open a platform to third-party developers.

The former decision concerns how a platform can reconcile the tension that emerges from the need to both maintain stability (to decrease coordination and value capture by complementors) and at the same time allow the platform to expand into new territories (Saadatmand et al., 2019; Wareham et al., 2014). First, to maintain stability, developers may enact a coherent search strategy. Brunswicker and Schecter (2019) refer such search to a developer being guided by past experiences and known solutions to prevalent problems. In this way, developers are likely to identify solutions characterized by stability and re-use potential. While coherent searches promise to maintain a platform's stability, however, a too one-sided focus on these searches undermines changes in the ecosystem that help the platform to stay attractive. In contrast, Brunswicker and Schecter (2019) refer flexible searches to a developer exploring novel solution spaces that possibly meet anticipated future needs. Such flexible searches that branch out of solutions coherent with the past are more likely to impact than those that lack this connection with the past.

The second core aspect of governance deals with how to afford outside access to a platform. Karhu et al. (2018) offer two key strategies to achieve such openness: access openness and resource openness. With regard to the former, a platform is opened by offering a controlled way in to selected parts of its core, i.e., "the granting of access to external complementors to participate and conduct business on a platform by providing them with dedicated resources to interact with the platform" (Karhu et al., 2018, p. 481). The platform owner simply chooses what parts of, in what form, and under which IP regime that external users can use the platform. As such, access openness offers the platform owner additional flexibility in terms of the future use trajectory. When governing third-party development in this way boundary resources play a crucial role (Ghazawneh & Henfridsson, 2013). These resources constitute the thin layer of assets that both capacitate and confine third-party complements, and include APIs, SDKs, license terms, and testing tools.

The other approach to open a platform is through resource openness, which involves making the platform core available to users, i.e., "opening the platform's valuable resources by forfeiting the IPR of the resource" (Karhu et al., 2018, p. 481). As such, it is closely related to platform governance that makes the platform core readily available to users. This means that a platform owner may achieve greater uptake because legal barriers to re-use have been removed. The owner still has also limited means of controlling the continued evolution of the platform.

2.2. Technology Architecture

Platform emulation depends on the transformation of an organization's resources to resemble the existing ecosystem's desired behavior. Here, platform architecture constitutes a necessary means to not only reorganize incumbent digital resources, but also redistribute design capabilities to third-party developers. Such architecture consists of a stable, modular platform core, standardized visible interfaces, and peripheral applications (Karhu et al., 2018; Saadatmand et al., 2019). To enable seamless additions of new complements to an open platform, the core needs to be modular and draw on the principle of information hiding, which posits that a designer of modular systems should ensure that only necessary information is available for the module users to reduce dependencies and enable change.

When the core builds on information hiding, users can only act on visible information. Such information has been conceptualized as design rules (Baldwin & Clark, 2000), which stipulate how module developers can establish compatibility with the platform. A complete set of design rules consists of a blueprint that describes the modules of the architecture, the interfaces of these modules, and accompanying integration protocols and testing standards that enable a developer to integrate his/her app with the platform.

To emulate by relying on design rules, a designer must reorganize the modules of a platform to achieve the desired capabilities. The resulting architecture constitutes what visible modules that external developers can interact with (Kazan et al., 2018). Baldwin and Clark (2000) suggest that modular operators play a crucial role within modular systems. These operators act as a discrete set of options by which a developer can alter such architectures. In our

research, three operators play crucial roles in achieving an open platform (Karhu et al., 2018; Tiwana, 2014). By inverting, a designer can create modules that implement widely used or requested functionality within a digital ecosystem. Through substituting, a platform developer can replace existing modules with improved qualities. By mutating, modules are copied for usage in other application domains (Karhu et al., 2018; Tiwana, 2014, p. 195).

However, design rules also require visible interfaces that specify behavior of platform modules. They serve as a description of what the platform affords to third-party developers, i.e., boundaries of possible platform innovation. From an architectural standpoint, there are two key decisions that concern interface design for open platforms: the degree of app-platform decoupling and interface standards (Tiwana, 2014, pp. 106-114). Decoupling occurs when a designer minimizes the visible information by increasing a module's encapsulation of internal complexities. Such designs decrease dependencies between the platform and its apps, thus making integration and testing more straightforward (especially for new platform developers). Though a drawback from too far-reaching decoupling is the risk of reducing third-party developer experiment opportunities (Tiwana, 2014, p. 105). Interface standards guide how boundary resources materialize on the platform. Key considerations here include communication protocols, compliance with existing industry standards, and versatility.

Finally, design rules address the use of integration protocol and testing standards to provide the necessary information for third-party developers to connect the platform's core interfaces with interfaces of the complementary app's micro-architecture. These types of design rules include SDKs, IDEs, or code examples. These extensions target developers during the design of their application, e.g., by providing entry paths for new platform developers (e.g., code examples), simulating the runtime environment, or ensuring compatibility with specific devices (Tiwana, 2014). In this way, platform complexities (Cennamo et al., 2018) can be encapsulated to minimize coordination costs for third-party developers (Tiwana, 2015).

3. Research Method

We have pursued an ADR effort to answer our research question. In ADR, guided by its principles and stages, researchers collaborate with practitioners to resolve prevalent problems through artifact design and at the same time generate generalized design knowledge for re-use in other design situations. Successful ADR efforts should generate three types of contributions: Two practice-oriented contributions (ensemble-specific contributions and end-user utility) and one theory-building contribution (design principles) (Sein et al. 2011).

We conducted this research in close collaboration with the Swedish Transport Administration (STA) between January 2012 and August 2020. STA has the overall responsibility for both physical and digital transport infrastructure in Sweden. As such, the administration is responsible for communicating traveler-critical information like passage times, departure platforms, and potential delays for passenger trains. At the outset, STA did not grant third-party developers access to railway-related real-time data. However, despite this lack of official third-party resources for train data, multiple railway apps relying on scraping had emerged. These apps were written by independent developers, primarily driven by self-experienced needs, and a handful of these applications had gained a high number of downloads in application marketplaces. STA therefore sought our guidance in designing a platform for railway data that would satisfy needs of third-party developers.

Consistent with ADR, the platform was developed and refined over several versions, each drawing on various signals to guide our research efforts (see Table 1). While the three first iterations

Table 1 - Data collected per phase

Version	Interviews	Workshops / meetings	Other vital empirical material
Alpha 2012-01-2012-05	N=13 (Tot:695 mins, 76129 words)	N=1 (390 mins)	Apps' functionality (Tot: 6 apps)
Beta 2012-05-2013-01	N=17 (Tot:604 mins, 66930 words)	N=18 (Tot: 1650 mins)	Interface specification discussions (Tot: 28 posts, 2991 words)
Release 2013-04-2014-08	N=12 (Tot: 543 mins, 69521 words) Emails developers (Tot 4 emails, 1587 words)	N=18 (Tot: 1140 mins)	Apps' data sources (Tot: 51 gross, 19 net)
Maintenance 2014-03-2020-08	N=6 (Tot: 406 mins, 55787 words) Emails developers (Tot 4 emails, 1424 words)	-	Apps' data sources (Tot: 51 gross, 19 net), Usage statistics, API Changelog (Tot 19 items, 281 words)

closely followed the ideal model of ADR, the final version (maintenance) unfolded without researcher interventions. In this phase, we instead relied on extracting case study techniques to extract the final design principle (Van Aken, 2004).

4. Design Theory Development

Our ADR process together with STA generated alpha (4.1), beta (4.2), release (4.3), and maintenance (4.4) versions of the sought for open platform.

4.1. Alpha Version

4.1.1 Problem Formulation. We started our investigation by examining the existing, unsanctioned apps revealing that they typically implemented a standard set of use cases. These included searching for a station based on a search string, getting departures/arrivals from a specific station and platform, and getting a particular train's status. Moreover, we found that developers scraped data from a variety of interfaces. Some relied on an obscure web page designed for mobile use that, due to its minimalistic use of HTML, made the page less complex to parse and re-process. Another common way of accessing data was through a JavaScript interface at the STAs web page. This interface provided an unsanctioned API (albeit without developer documentation) to a system named Orion. On top of Orion, the STA had developed a flexible query language (similar to SQL) to retrieve information. When interviewed about what they would like to see in an official API from the STA, developers stressed capabilities focusing on simplicity and immediate problem-solving.

Based on this background material, we assessed that the primary problem for the STA was missing coherent search capabilities that had emerged during unsanctioned app development and experimentation. Moreover, given the uncertainty regarding how, if at all, the STA would offer real-time railway data to third-party developers, there was a need to provide these data through access openness. This way, the STA could investigate what data and in what form the potentially increased openness could be implemented.

4.2.2 Building, Intervention, and Evaluation. To address this problem, our alpha version platform as a solution blueprint, included a new software layer that effectively inverted observed third-party developer behavior through an interface offering access to coherent searches.

In April 2012, the project held a joint workshop summoning nine representatives from STA, two from Trafiklab.se, and the first author of this paper. This workshop's idea was to bring different stakeholders together for the first time and test the design on both third-party developers and more stakeholders within STA. We introduced the suggested solution blueprint (in the form of a PowerPoint presentation, presenting both capabilities and overall implementation structures). Regarding access openness, developers were quite content with this type of openness regime. In terms of solution search mechanisms, our idea was to package these recurrent use cases as REST endpoints to minimize developers' need to invest in

Table 2. Product and Process Principles for the Alpha Version Platform

	Product aspect	**Process aspect**
Principle title	Principle of Platform Access to Externable data and functionality	Principle of Artificial Platform Demonstration
Aim, implementer, and user	To allow designers to emulate external development activities into alpha version open platforms targeting external developers	
Context	In a situation where external development is based on self-resourcing	
Mechanism	Design a blueprint exhibiting access to frequently self-resourced functionalities together with other data available through self-resourcing through a new, abstract software layer with dedicated interfaces offering such emulated functionality	Execute an artificial demonstration of the alpha version platform blueprint including both external self-resourcing third-party developers and managerial decision makers
Rationale	Because platform ecosystems are largely dependent on the stability that tested and re-usable knowledge entails, but also need to be able to evolve beyond such functionality. Existing systems can remain untouched when offering designated access openness to these platform capabilities by inverting existing systems architectures	Because deploying an open platform requires a substantial resource investment, and long-term commitment that require alignment with developer preferences as well as managerial anchoring enabling further development

industry-specific domain knowledge and be suitable for mobile clients' direct consumption.

While more experienced developers confirmed the value of having the coherent search interface as a natural entry point for novel developers, they were surprisingly critical towards having such a design as the only approach. More specifically, they wanted access to all data points to enable the design of novel services. However, the more precise formats for such flexible searches appeared less critical. We assigned developers to break out of the entire group during the workshop and discuss what formats would be of interest for such capabilities. Amid these discussions, a joint position among developers emerged, where data could be pretty crude.

Despite this criticism of missing flexible search mechanisms, the blueprint's reception was overall positive. All developers agreed to participate in potential further development activities by providing feedback and input on how the STA could make real-time railway data available for third-party developers. Similarly, the STA appreciated the format and the ideas brought forward, but primarily meeting these developers in person. Thus far, they had mainly remained anonymous to the STA.

4.1.3 Reflection and Learning. Based on the intervention, we reflected on the solution and theories used. Regarding governance, we found support from the developers in both interviews and the workshop to implement non-discriminatory access openness. Moreover, developers embraced the blueprint's coherent search capabilities, and we concluded that the API needed quality-assured "shortcuts" to use-case-bound datasets with high developer demand. Moreover, given the unanticipated developer response on the constraining effect of merely publishing coherent searches, we concluded that the future platform also needed some mechanism to channel all data to allow for flexible searches. Architecturally, we assessed that inverting such requested functionality would be beneficial, and to use integration protocols and testing standards to facilitate integration.

4.2. Beta Version

Given these overall positive signals from workshop participants, we started to draft a more authentic beta version. The head of passenger information at the STA (also a workshop participant) corroborated this interpretation. In May 2012, she gave the go-ahead to start designing and deploying a live beta version, alongside access to the necessary personnel from the STA.

4.2.1 Problem Formulation. After forming the ADR team, we started to reformulate the problem to develop the beta version. While many of the assumptions addressed in the alpha version held, we concluded that access to data beyond these common use cases was necessary. However, the participating developers also expressed that this missing feature could be a less sophisticated capability; the core issue was to have all data points attainable in some way from the platform.

4.2.2 Building, Intervention, and Evaluation. As a next step, we started to address the more specific platform design aspects. Given the problem formulation, we decided to include the following elements:

First, we decided to largely reverse-engineer the current app behaviors and "pirate" API designs. Second, we decided to include another interface for all datapoints. However, since participating developers stated that they would be content with any format other than HTML, we also hypothesized that interface could be cruder.

In the alpha version workshop, we demonstrated the blueprint to the participants. Still, it had only been shown to four developers, and the details were not fully specified. To this end, the interface specifications were made publicly available on an open internet forum to gather input. Of the received replies, the feedback was overall positive. One feature request, however, appeared twice. This request concerned a task that developers currently struggled with, detecting changes since their last API call. Although seemingly simple, the STA was not able to implement this due to underlying architectural constraints. Orion was only a cached layer of information, and the entire dataset of Orion was replaced periodically, not just the records that had changed since Orion's last update. Consequently, this seemingly simple field addition required a significant redesign of the underlying system that was not feasible under the project budget constraints.

From the STAs systems perspective, their architecture was inverted through a new module facing application developers. This module was a cloud-based service hosted by an external cloud provider. This module handled access control and caching of data. Also, this module provided the two new interfaces, facing third-party developers and decoupled these from the STAs underlying systems.

The beta solution interfaces could not translate between the STA internal geographical coordinates (SWEREF99) into developer-friendly formats (WGS84). To this end, we included conversion code libraries as integration protocols to facilitate this translation. We also included an API console (that allowed developers to execute API calls without a development environment) and user registration functionality (dispensing API keys). Finally, we

created data model documentation and a tutorial to expedite the development process.

The solution was officially released in October 2012. Anyone could register for access, and in three months, 59 developers had registered. For evaluation, we contacted developers who had signed up for the API, inquiring into whether they would like to participate in an interview. Out of the 59 registered developers, 17 agreed to participate in an interview. Summarizing their impressions, users that had focused on the coherent interface found it utile. In this category, two developers of existing apps were found. The first had previously been using the pirate APIs and now investigated transitioning their apps data source to the new interface. Overall, they found such a transition straightforward and appreciated the interface's official status. The second type of developer who had focused on TrainInfo was new to the railway domain but could still use the API to match their needs and when asked to summarize their overall views from using the API, all 8 users of TrainInfo echoed a pleasant experience, as stated by a user:
TrainInfo is excellent. It was quick to get started and find the information you needed to find a solution to your problem. I don't think that STA needs to change a thing. Developer B13

Third-party developers that had used TrainExport conveyed a more complex picture. Users (2 developers) without live apps were quite content with the flexible search interface. However, those users (2 developers) that had existing, popular applications based on scraping expressed disappointment due to the lack of benefits of switching and had, for this reason, stuck with unsanctioned data access.

4.1.3 Reflection and Learning. As a next step we reflected on the mixed results of the beta version results . The primary benefit of implementing common use cases had been the enrollment of developers new to the railway domain. The solution proved to expand the number of developers quickly, both regarding minimized platform access negotiation and by lowering the barrier for extra-industry actors by inverting coherent searches into dedicated REST interfaces. However, developers with existing services expressed dislike for the flexible search capabilities and stuck with unsanctioned data access. Second, not only were these developers discontent with TrainExport capabilities vis-à-vis what some scraped resources could afford. They also expressed the need for additional flexible search benefits to motivate the effort of changing the data source, as commented by a user during the beta version evaluations:
I won't stop scraping, and that's mostly because I see no reason to, "if it ain't broken, don't fix it," something like that. There is nothing there that attracts me; I will stick to the current solution as long as there is no real reason to switch. Developer B14

4.3. Release Version

The beta version was a large-scale pilot project to inform a potential release version platform design, and

Table 3. Product and Process Principles for the Beta Version Platform

	Product aspect	**Process aspect**
Principle title	The Principle of Platform Capability with Non-Deterministic Use Support	Principle of Authentic Platform Development
Aim, implementer, and user	To allow designers to emulate external development activities into beta version open platforms targeting external developers	
Context	In a situation where external development is based on self-resourcing	
Mechanism	Offer access to production-use capabilities encapsulating product hackers' frequently implemented functionalities as well as offering non-deterministic use support by adding a new software layer conveying emulated capabilities through its interfaces	Execute the development of the beta version platform in an environment that concurrently allows authentic third-party development to unfold and under constraints that does not bind the platform owner to the beta version platform design rules.
Rationale	Because an open platform requires capabilities for both coherent and flexible searches, and existing systems can remain untouched when offering access to designated, production-mimicking platform capabilities by inverting existing systems architectures	Because the identification of improvement opportunities and non-negotiable capabilities for an open platform are facilitated by third-party developers assessing platform capabilities in perceived release circumstances, yet a platform owner need to retain the option to alter release version design rules, or even to withdraw from further development

the overall outcome of the trial convinced the STA to create a more persistent solution. To this end, the STA revised its third-party developer strategy. A new developer segment was added and denoted *Basic*. This segment should include general terms of use, rudimentary support in the form of FAQ and web-based support, and "simple, basic information products". However, while many insights on the more precise design of the boundary resources had been gained from the last ADR loop, the more exact design for the Basic segment still needed refinement.

4.3.1. Problem Formulation. To resolve these platform design issues, a new ADR project was formed. In the permanent solution, the solution should be implemented within the realm of the STA's systems rather than through Trafiklab.se. A new ADR team was formed, consisting of a project manager (participating in the previous iteration) and a systems architect/developer from the STA, and the first author of this paper. The project was funded internally and ran from August 2013 through March 2014. In contrast to the previous iteration (which was researcher-led), this iteration was led by the STA and had a researcher (the first author of this paper) as an ADR team member.

As a next step, we reformulated the problem. Given the beta version findings, our previous position concerning the need for coherent searches was reified. However, considering the disappointing result for existing users, we hypothesized that flexible searches also needed to be *emulated*, not just offered, as in the beta version.

4.3.2. Building, Intervention, and Evaluation. This somewhat surprising reception by experienced third-party developers instigated a substantial release version platform redesign. Based on these findings, we decided to implement a query language similar to that of Orion to cater to flexible searches. This design afforded more precise, flexible searches, as requested by developers.

Moreover, from the beta version design and onwards, ensemble signals conveyed a need for functionality that allowed them to retrieve records that changed since their last request. At this point, developers had to download a complete snapshot of all running trains in Sweden and then write an algorithm that detected any potential changes since their last request. To further investigate whether this feature was necessary, we triangulated our findings towards apps using Stockholm Public Transport real-time data that had a history of scraping but offered non-discriminatory access openness since September 2011. For those still using unsanctioned data access (4/21), one important rationale given to use scraping over the Open API was the lack of incentives to switch to the open API. Hence, to influence such developers to desert unofficial interfaces, the ADR team decided to implement new functionality that the current solution did not include – the ability to deliver changes since the previous request.

Moreover, the ADR team decided to apply a new governance regime for the platform's openness, resource openness, a far-reaching decision that came about for several reasons. In this platform context,

Table 4. Product and Process Principles for the Release Version Platform

	Product aspect	**Process aspect**
Principle title	The Principle of Platform Growth by Experiment Flexibility	The Principle of Target Platform Implementation
Aim, implementer, and user	To allow platform designers to emulate external development activities into release version open platforms targeting external and internal developers	
Context	In a situation where external development is based on self-resourcing	
Mechanism	Offer the improved capabilities to both external and internal users under the same conditions, including shortcuts to product hackers' frequently implemented functionalities as well as non-deterministic experiment flexibility by substituting the digital resource subject to self-resourcing with modules providing non-deterministic interfaces and common functionality through integration protocols.	Ensure that both desired third-party developer behavior is persevered in target platform implementation as well as warranting a flexible upgrade plan for internal applications in their adoption of the release version platform
Rationale	Because an open platform requires coherent and flexible search capabilities for both internal and external users, and such resource openness requires that the underlying system is substituted with a resource implementing the desired emulated behavior.	Because transforming internal digital resources to an open release version platform may infer altered design rules compared to both the beta version, as well as substituted release versions

resource openness entailed offering the same platform to external as well as internal clients. First, since the STA now planned to provide its internal (albeit improved) query language for external developers, there were little incentives to encapsulate it behind a software layer providing access to the resource. Second, two of those still scraping Stockholm Public Transport did this as the data was not available through the API. Consequently, any data access deviations between the interfaces offered to third-party developers and public applications risked introducing new self-resourcing. Finally, the STA did want to maintain more interfaces than necessary. By providing improved interfaces similar to those of Orion, through the new platform, DataCache, the STA could easily upgrade its own applications to use the new platform while still serving external third-party developers' needs.

However, this resource openness decision entailed challenges for the platform's architecture. At this point, the ADR team decided to *substitute* functionality that had been residing in Orion. This redesign enabled both the STA and third-party developers to use the new platform when developing end-user services. However, Orion's query language was designed for internal usage, making it unsuitable for publishing in its current form. To this end, the query language was redesigned for reduced redundancy, syntax strictness and clearness, and data model congruence

Data was retrieved through a query interface – where developers could construct their own data retrieval composition based on three underlying information objects (one for trains, one for passenger announcements, and one for stations). The interface required an authentication token, what information objects and fields the user intended to query, and optional selection criteria (such as a given station). Moreover, all these information objects now included ModifiedTime signifying the most recent update of a given data post. This field enabled developers to retrieve only the records that had been changed since their last request and resolved the previously tedious work of sorting out changes to real-time information. Finally, the information objects now included the WGS84 geographic coordinate system, effectively scrapping developers' need to perform the conversion between SWREF99 and WGS84.

The query interface was non-deterministic and thus inherently supported flexible searches. Consequently, it was no longer possible to use the interface level to implement coherent searches. Instead, we opted for a revised architectural configuration. Here, we used integration and testing protocols, i.e., predefined example queries, to implement the coherent searches in previous ADR iterations. This way, the exact syntax of the question, e.g., the departures from a given train station, was provided by STA but simultaneously served as a starting point for those who wanted to develop the query further. Moreover, given the developers' positive reception regarding the API console, documentation, and tutorial/example API calls, we also implemented those as integration protocols.

The platform went live on March 18. For evaluation, we interviewed another 12 developers that had registered as users of the platform. Although minor technical shortcomings were identified by the interviewees, when developers were asked to summarize their experience of the APIs, they were all positive and stated that they would recommend this API to other developers interested in developing railway services.

4.3.3 Reflection and Learning. At this point, our evaluations showed that developers appeared to be content with platform capabilities., based on emulation. However, for the ensemble platform to persevere after release we decided to continue the observe the design moves by the STA.

4.4. Maintenance Version

The API platform persevered long after its release and is at the time of writing (2022-05-18) still in production. In the following, we summarize the evolutionary trajectory after the platform's launch by paying specific attention to developer adoption, continued emulation activities since the ADR interventions, and finally, how the platform has been received within the STA and the Swedish public transport industry.

In September 2016, the first author of this paper investigated the actual data sources used for the apps. The review was performed in the same way as the scraping follow-up for SL: by intercepting the apps' API calls. The investigation revealed that development towards unsanctioned interfaces had ceased. At the time, 28 services for smartphones using real-time information were available in the application marketplaces for Apple iPhone, Google Android, and WindowsPhone. Out of the 28 real-time services, 19 used the open API, 6 used interfaces connected to other STA third-party development segments, and 3 were not functioning (no longer maintained).

Moreover, API usage statistics (see Table 5) from the platform showed that not only existing developers seemed to have adopted the API, but external clients are also currently generating more calls than internal clients. Also, the STA continued to incorporate new data fields in the railway data, based on developer

requests and feedback. One such illustrative example concerned "ViaToLocation". Typically, a train is announced by several stations the train is passing during a trip (the significant stations along the line). However, the order in which these stations are passed was not explicit in the API response. While it was possible to derive the order by examining the estimated/actual passing time along line, the STA decided to incorporate a clear indicator of the order of the location a particular train passes. Such cognizant changes to the data models had become more institutionalized after the ADR project.

Table 5. New user registrations, external and internal API calls

Period	New users	External calls millions/month	Internal calls millions/month
2014	338	No data	No data
2015	422	No data	No data
2016	639	22,2	78,7
2017	702	41,1	95,6
2018	1466	69,7	83,5
2019	1377	90,7	63,2
2020	783	100,5	63,2

The final aspect that surfaced in the follow-up study concerned how the emulated platform was adopted within the STA. Until 2015, DataCache had only been deployed once within the STA. This instance was the open platform for both external third-party developers and end-user services catered for by the STA. However, in 2015, the systems development team responsible for DataCache suggested using DataCache codebase for an internal project. After this first usage, the platform had continuously grown in popularity. These new, internal instances contained the same functionality, included a test console, API documentation, relevant query examples, and required internal developers to register to get access tokens. The only thing that differed was that the data objects were different from those present in the open platform. When asked what helped the DataCache team to embrace this approach, they pointed to the increased understanding of third-party developer needs through the ADR project.

In 2020, the STA performed an internal investigation to appoint an official integration platform to be used throughout the agency. After going through existing solutions at the STA and other external products, the inquiry recommended management at the STA to choose and appoint the DataCache platform. This recommendation was primarily based on the teams' experiences using the platform and their reported development velocity. In August 2020, STA IT Management did decide in favor of this investigation, thus making DataCache the official integration platform of the entire STA.

5. Concluding Remarks

The trajectory of the maintenance version suggests that our ADR project delivered both end-user utility and ensemble-specific contributions. Given our agenda to advance new design theory, we here discuss

Table 6. Product and Process Principles for the Maintenance Version Platform

	Product aspect	Process aspect
Principle title	The Principle of Platform Equilibrium through Internal Integration	The Principle of Ensemble Platform Manifestation
Aim, implementer, and user	To allow platform designers to maintain open platforms targeting external and internal developers	
Context	In a situation where external development based on self-resourcing has been emulated	
Mechanism	Offer new public datasets with the same capabilities and restrictions to both external and internal users, including shortcuts to projected frequently implemented functionalities as well as non-deterministic experiment flexibility and mutate the open platform for internal usage.	Maintain the platform in a way that ensures that both sides of the ensemble are content, by conditioning publishing of new datasets with having support for desired behavior and by encouraging internal use of emulated capabilities.
Rationale	Because continual offering of data ex-post open platform release with coherent and flexible search capabilities for both internal and external users will maintain platform integrity, and mutating the open platform allows for the emulated capabilities to be used in internal settings	Because publishing new data ex-post open platform release having support for desired behavior will facilitate platform usage and stall new self-resourcing, and by encouraging internal use in new contexts, the platform owner may harness emulated capabilities for proprietary organizational purposes

our developed design principles as a third type of ADR contribution. To develop them, we have followed the design principle schema suggested by Gregor et al. (2020). We have also embraced a design-theoretical tradition that emphasizes the importance of not only articulating product properties, but also providing process-oriented guidance to help designers to effectively meet their goal (Walls et al., 1992). Indeed, developing an open platform by relying on our emulation theory requires a deliberate process intervention capable of accumulating relevant design knowledge over time. In the same vein, our collaboration with STA has generated both product and process insights in the realm of designing open platforms as a particular class of digital platforms.

With regard to the alpha version, we stress the importance of designing a blueprint that exhibits coherent and flexible searches without necessarily binding them to the design rules. On the process side, despite the artificial nature, we emphasize the value of involving self-resourced developers and decision-makers in the client organization (see Table 2).

As for the beta version, we assert that it should include capabilities for coherent and flexible searches suitable for live use, but without binding the platform to the current design rules. In terms of our process insight, we point to the importance of performing authentic evaluations to ensure that emulated capabilities materialize (see Table 3).

The release version should include emulated capabilities for both coherent and flexible searches that are suitable for both internal and external clients. What we learned about the process is that it is key to not only preserve third-party developer preferences, but also prepare internal clients for the changed design rules (see Table 4).

Finally, speaking of the maintenance version, we recognize the centrality of both introducing new datasets with emulated capabilities and mutating the platform to leverage emulation also internally. Our process insight here is that a platform owner should take actions that help to ensure that both internal and external clients are satisfied (see Table 6**Error! Reference source not found.**).

6. References

Baldwin, C., & Clark, K. (2000). *Design Rules*. MIT Press.

Brunswicker, S., & Schecter, A. (2019). Coherence or Flexibility? The Paradox of Change for Developers' Digital Innovation Trajectory on Open Platforms. *Research Policy*, *48*(8), 103771

Cennamo, C., Ozalp, H., & Kretschmer, T. (2018). Platform Architecture and Quality Trade-offs of Multihoming Complements. *Information Systems Research*, *29*(2), 461-478

Eisenmann, T., Parker, G., & van Alstyne, M. (2009). Opening Platforms: How, When and Why? In A. Gawer (Ed.), *Platforms, Markets and Innovation* (pp. 131-162). Edward Elgar Publishing.

Flowers, S. (2008). Harnessing the Hackers: The Emergence and Exploitation of Outlaw Innovation. *Research Policy*, *37*(2), 177-193

Foerderer, J., Kude, T., Schuetz, S., & Heinzl, A. (2019). Knowledge Boundaries in Enterprise Software Platform Development: Antecedents and Consequences for Platform Governance. *Information Systems Journal*, *29*(1), 119-144

Ghazawneh, A., & Henfridsson, O. (2013). Balancing Platform Control and External Contribution in Third-Party Development: The Boundary Resources Model. *Information Systems Journal*, *23*(2), 173-192

Gregor, S., Kruse, L., & Seidel, S. (2020). Research Perspectives: The Anatomy of a Design Principle. *Journal of the AIS*, *21*, 1622-1652

Hartman, R., & Teece, D. (1990). Product Emulation Strategies in the Presence of Reputation Effects and Network Externalities: Some Evidence From the Minicomputer Industry. *Economics of Innovation and New Technology*, *1*(1-2), 157-182

Karhu, K., Gustafsson, R., & Lyytinen, K. (2018). Exploiting and Defending Open Digital Platforms with Boundary Resources: Android's Five Platform Forks. *Information Systems Research*, *29*(2), 479-497

Kazan, E., Tan, C.-W., Lim, E., Sørensen, C., & Damsgaard, J. (2018). Disentangling Digital Platform Competition: The Case of UK Mobile Payment Platforms. *Journal of Management Information Systems*, *35*(1), 180-219

Saadatmand, F., Lindgren, R., & Schultze, U. (2019). Configurations of Platform Organizations: Implications for Complementor Engagement. *Research Policy*, *48*(8), 103770

Sein, M., Henfridsson, O., Purao, S., Rossi, M., & Lindgren, R. (2011). Action Design Research. *MIS Quarterly*, *35*(1), 37-56

Teece, D., Pisano, G., & Shuen, A. (1997). Dynamic Capabilities and Strategic Management. *Strategic Management Journal*, *18*(7), 509-533

Tiwana, A. (2014). *Platform ecosystems: aligning architecture, governance, and strategy* (1st ed.). Morgan Kaufman.

Tiwana, A. (2015). Platform Desertion by App Developers. *Journal of MIS*, *32*(4), 40-77

van Aken, J. (2004). Management Research Based on the Paradigm of the Design Sciences: The Quest for Field-Tested and Grounded Technological Rules. *Journal of Management Studies*, *41*, 219-246

Walls, J., Widmeyer, G., & El Sawy, O. (1992). Building an Information System Design Theory for Vigilant EIS. *Information Systems Research*, *3*(1), 36-59

Wareham, J., Fox, P., & Cano Giner, J. (2014). Technology Ecosystem Governance. *Organization Science*, *25*(4), 1195-1215