Designing Platform Emulation

THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Designing Platform Emulation

Daniel Rudmark

Department of Applied Information Technology
University of Gothenburg

UNIVERSITY OF GOTHENBURG

Gothenburg 2021

Cover illustration: Catharina Jerkbrant

Designing Platform Emulation

daniel.rudmark@ri.se

ISBN 978-91-8009-392-7

Printed in Borås, Sweden 2021
Stema Specialtryck AB

To my family

Designing Platform Emulation

Daniel Rudmark

Department of Applied Information Technology
University of Gothenburg
Göteborg, Sweden

ABSTRACT

Many contemporary firms and public agencies seek to engage external third-party developers to supply complementary applications. However, this type of development sometimes occurs without organizational consent, which creates problems for subjected organizations at both the technical and organizational levels.

In this thesis, I have developed a theoretical perspective called *open platform emulation*. This perspective builds on emulation logics, where designers use an external model as a basis for developing compatible platform capabilities superior to the original model. In this thesis, this model has been external unsanctioned development. In open platform emulation, such capabilities include governance decisions enabling coherence with previously proven solutions, the flexibility to accommodate new development trajectories, and strategies for applying openness to a digital resource. The means to achieve these capabilities involves design rules' architecture, interfaces, and integration protocols, which convey the capabilities to third-party developers. This way, a platform owner can draw on governance and architectural configurations to emulate self-resourcing behavior through the platform core.

I generated the contributions from this thesis by materializing open platform emulation in a clinical setting. More specifically, I used action design research (ADR) together with the Swedish Transport Administration (STA). Starting in early 2012, I led a platform initiative that, in collaboration with the STA, sought to emulate self-resourcing to design an open platform. Here, I conducted two full ADR cycles that resulted in a currently active production platform used by both

the STA and external third-party developers. Before this engagement, I also conducted studies of related phenomena within the Swedish public transport industry, and I have continued to follow the STA's platform trajectory since its release in 2014.

The theoretical contributions from this thesis include design principles that seek to guide the designers of open platforms in situations where digital resources are subject to self-resourcing. These design principles cover both product and process aspects throughout the open platform's developmental trajectory. Also, I offer additional theoretical implications based on this work. These include extensions to current theories on open platforms, different types of platform emulation, an enunciated influence response to outlaw innovation, and methodological implications for guided emergence in ADR.

Keywords: open platforms, platform emulation, outlaw innovation, action design research, guided emergence

ISBN: 978-91-8009-392-7

SAMMANFATTNING PÅ SVENSKA

Många företag och offentliga aktörer försöker engagera externa tredjepartsutvecklare för att utveckla appar och andra digital tjänster. Ibland sker dock sådan extern utveckling utan organisationens medgivande, vilket kan innebära problem för utsatta organisationer på både teknisk och organisatorisk nivå.

I den här avhandlingen har jag utvecklat ett teoretiskt perspektiv, som jag kallar öppen plattformsemulering. Detta perspektiv bygger på emuleringslogik, där designers använder en extern modell som grund för att materialisera plattformsförmågor som blir överlägsna modellen. Öppen plattformsemulering inkluderar förmågor för att kan möjliggöra för externa utvecklare att återskapa populära lösningar, men också tillräcklig flexibilitet för att tillåta mer banbrytande innovation, tillsammans med strategier för att tillämpa öppenhet på en digital resurs. Medlet för att uppnå detta är plattformens designregler, d.v.s. arkitektur, gränssnitt och integrationsprotokoll som förmedlar funktionerna till tredjepartsutvecklare.

Empiriskt har jag använt mig av *action design research* (ADR) tillsammans med Trafikverket. Med start 2012 har vi tillsammans designat en öppen plattform, som till dags dato nyttjas som produktionsplattform av både Trafikverket och externa utvecklare. Före denna intervention genomförde jag också studier av relaterade fenomen inom den svenska kollektivtrafikbranschen, och jag har fortsatt att följa Trafikverket sedan plattformen lanserades 2014.

De teoretiska bidragen från denna avhandling inkluderar design-principer för öppna plattformar vars digitala resurser används i icke-sanktionerad extern utveckling. Designprinciperna täcker både produkt- och processaspekter i den öppna plattformens hela utvecklingscykel. Avhandlingen bidrar också till teorier om öppna plattformar, beskriver olika typer av plattformsemulering, hur man kan hantera s.k. *outlaw innovation* samt ger ett metodbidrag till ADR.

LIST OF PAPERS

I. Koutsikouri, D., Lindgren, R., Henfridsson, O., and **Rudmark, D.** 2018. “Extending Digital Infrastructures: A Typology of Growth Tactics,” *Journal of the Association for Information Systems* (19:10), pp. 1001–1019

II. **Rudmark, D.**, and M. Lind. 2011. "Design Science Research Demonstrators for Punctuation – The Establishment of a Service Ecosystem," in *Service-Oriented Perspectives in Design Science Research*, H. Jain, A. Sinha and P. Vitharana (eds.), Berlin: Springer, pp. 153–165.

III. **Rudmark, D.**, E. Arnestrand, and M. Avital. 2012. "Crowdpushing: The Flipside of Crowdsourcing," in *Proceedings of the 20th European Conference on Information Systems (ECIS 2012).*

IV. **Rudmark, D.** 2013. "The Practices of Unpaid Third-Party Developers – Implications for API Design," in *Proceedings of the 19th Americas Conference on Information Systems (AMCIS 2013)*.

V. **Rudmark, D**. 2021. "Designing Open Platform Emulation," *Under review at the 42nd International Conference on Information Systems (ICIS 2021).*

ACKNOWLEDGMENTS

Although completing a Ph.D. hinges on the student, the direction and content of this thesis have been anything but a solo project. In terms of my academic journey, I am forever thankful for the energy, perseverance, appreciation, and skillful guidance provided by my advisor, Rikard Lindgren. Thank you so much, Rikard. My sincere gratitude also goes to Mikael Lind for introducing me to the craft of research as well as the transportation setting, in which I am still active.

Besides this guidance, the present work would not have been possible without financial support. In this regard, the University of Borås, RISE Research Institutes of Sweden, Vinnova, the Swedish Transport Administration (STA), Region Västra Götaland, Sjuhärads kommunalförbund, and the University of Gothenburg have all helped to fund work related to this thesis. Thank you for your generous support!

Next, I would like to wholeheartedly thank all of the co-authors involved in the papers appended in this thesis: Mikael Lind, Elias Arnestrand, Michel Avital, Dina Koutsikouri, Ola Henfridsson, and Rikard Lindgren. It has been a true privilege to write with and learn from you all. I also thank my co-authors for the papers that were not included here but have still been instrumental in my learning. Thank you, Anders Hjalmarsson Jordanius, Ahmad Ghazawneh, Gustaf Juell-Skielse, Paul Johannesson, Workneh Ayele, Stefan Cronholm, Hannes Göbel, Amir Mohagheghzadeh, Per-Erik Holmberg, Johan Sandberg, and Magnus Andersson.

The academic side is but half of this journey. I want to thank two people that made working with real-world platform design possible for me. First, my sincerest gratitude goes to Elias Arnestrand. In 2010, you generously invited me to be part of what came to be Trafiklab. Ever since, you have helped me understand the public transport industry, been an invaluable sounding board, and a continuous source of insight. Second, my thanks go out to Lars-Olof Hjärp at the

STA. Without your continuous support, perceptivity, and teamwork, the content of this thesis would not have been possible.

I also want to thank the two ADR teams that made the development of various platforms possible. In 2012, I had the pleasure of working with Andreas Krohn, Lars Löfquist, Per Gidlund Montén, Henrik Hammarström, Lars-Olof Hjärp, and Elias Arnestrand. Then, in 2013/2014, the team consisted of Magnus Pettersson and Lars-Olof Hjärp. Thank you all! Also, extra thanks are due to Magnus Pettersson for his generosity in explaining and digging up data on what has happened since the platform was launched. Moreover, thanks to all of the transport organizations involved in this work (including especially helpful contact persons): the STA (Clas Roberg), Samtrafiken (Håkan Östlund and Vojislav Marinkovic), Västtrafik (Mikael Faleke), AB Storstockholms Lokaltrafik (Robert Fromell), and the City of Gothenburg (Noel Alldritt and Christer Erlandsson).

Importantly, thank you to all of the developers (72!) that so generously shared their expertise on what constitutes attractive platforms in a multitude of ways. Special thanks are due to Teodor Storm, Erik Eng, Rickard Nordström Pettersson, and Anders Granåker for their active and continuous participation in shaping the open API at the STA. I also want to thank the teams Kreativ Stuga, Mobisleapps, Hemliga Byrån, and Krawaller for allowing me to video record them for 24 hours. These recordings have been indispensable in understanding what does and does not make up useful APIs.

I want to thank the InnovationLab research group at the University of Borås for their important feedback and for providing a setting for conducting design-oriented IS research. Thanks also to Department of Applied Information Technology, University of Gothenburg, and their faculty, for hosting an excellent doctoral education. Thanks to Ulrike Schultze shedding new, insightful light on this research as discussant at my final seminar. Moreover, Mobility and Systems, and especially all Fellow Innovators of the Digital at RISE, thanks for the encouragement and being such great colleagues. Special thanks to Taline Jadaan for your support during (and before) my final write-up. Also, to Anders Hjalmarsson Jordanius, thank you for being such a great co-pilot in ISET (and after) and for going the extra mile for me

to finish this thesis. To Khruangbin and Eric Schüldt, thanks for providing the soundtrack for the writing of this thesis frame.

On a personal note, I would like to thank my parents Anders and Gunnel Rudmark for your unconditional support and encouraging me to both start and complete this journey—thank you! Also, thanks to my sisters Sara and Anna and their families for always being there for me. I also want to thank Siri and Edit for the blessing of having you as a part of my life. Frida, words cannot express the magnitude of your support—without you, there would simply be no thesis. Thank you for showing me that just as for a thesis, life can benefit from a new beginning. Axel and Albin, you mean everything to me, and you have grown into star models that I now can emulate. With this thesis done, I look forward to more time for us to laugh and be together.

Buskeröd, Kullahalvön

2021-05-14

CONTENT

1 INTRODUCTION

It's my fault
I never learned a trade
So I just scrape all day.

The Lemonheads

While innovation is imperative to surviving in today's fierce competition, external innovation sometimes occurs without organizational consent. A contemporary example concerns vehicles produced by Tesla. Although these cars are highly digitalized products, they currently lack official open application programming interfaces (henceforth API) that allow for external innovation. However, a vibrant community of technology enthusiasts has reverse-engineered Tesla's internal APIs and currently provides both hands-on instructions[1] and documentation[2] for how private individuals may go about using these unofficial interfaces. As a consequence, an array of innovative applications has been showcased. These include sending a text message as the car approaches a specific destination[3] and remotely unlocking the

[1] https://medium.com/@jhuang5132/a-beginners-guide-to-the-unofficial-tesla-api-a5b3edfe1467

[2] https://tesla-api.timdorr.com/ and https://www.teslaapi.io/ are two alternatives.

[3] https://medium.com/initial-state/how-to-build-a-tesla-data-dashboard-with-the-tesla-api-4ebee4b9827c

charger from the car's socket[4]. A more spectacular form of API usage includes integrating Amazon Alexa with unofficial Tesla APIs to enable the execution of a voice command that automatically moves a car out of a garage[5].

Although Tesla maintains all rights regarding the use of their software, they have not engaged in any legal action against these unsolicited uses to date. However, since Tesla's position on third-party developers remains unclear, Apple has banned most Tesla apps from its App Store in the spring of 2020[6]. To keep their apps published in the App Store, developers must provide written permission from Tesla.

1.1 Outlaw Innovation

This form of unsanctioned development has been coined *outlaw innovation* (Flowers, 2008). The term refers to innovation with "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). Outlaw innovation may thus infringe on an organization's intellectual property, which is governed by a product's terms of use or ruling laws. While outlaw innovation may take different forms, the outlaw innovator category of interest for this thesis is the *product hacker* (Flowers, 2008). These innovators typically seek to expand the boundaries of a product or service by reverse-engineering the underlying technology, as per the aforementioned Tesla API example[7].

[4] https://medium.com/@mattjeanes23/tesla-auto-charge-port-unlock-604101b25403

[5] https://www.teslarati.com/tesla-model-s-voice-command-amazon-echo/

[6] https://www.evword.com/2020/04/24/apple-bans-3rd-party-tesla-apps/

[7] Additional examples of such product hacking include modifying consumer products such as gaming consoles (Flowers, 2008; Kartas & Goode, 2012; Schulz & Wagner, 2008), video games (Mollick, 2005; Postigo, 2003), digital video recorders (Mollick, 2005), and toys (Lessig, 2004, p. 165). Product hacking has also been observed in more professional contexts, such as dentistry (Braun & Herstatt, 2008).

Since outlaw innovation may challenge existing and future revenue streams, brand image, and intellectual property governance, many organizations tend to take action against such unsanctioned hacking. According to Flowers (2008), there are several possible measures that organizations may take (often in combination) to mitigate outlaw innovation activities.

The most hostile response to outlaw innovators is an *attack*. Such a move is typically executed through legal measures, where the organizations subjected to outlaw innovation litigate either the outlaw users themselves, their distribution channels, or both (Braun & Herstatt, 2008).

However, organizations may instead take less confrontative measures against unsanctioned innovation. According to Flowers (2008), a typical response is to merely *monitor* these uninvited activities. Such monitoring may later be used to better understand flaws in a product's security architecture or possible unfulfilled customer demand. In other cases, a host organization may choose to *adapt* the outlaw innovation to their advantage. In this regard, Flowers (2008) refers to organizations incorporating their version of outlaw innovation into the product or service.

When the skills and capacities of the user innovation community are relevant to the company, Flowers (2008) described two remaining responses. The most far-reaching is to *absorb* the community by actively incorporating (parts of) the innovator ecology into the organization's offering. This response has been prevalent in the gaming industry, where many game users engage in developing derivatives, or mods (Schäfer, 2011). Moreover, Apple has exercised a far-reaching absorption response to jailbroken iPhones and succeeded to incorporate (and subsequentially further grow) the jailbreak developer ecology into the smartphone's offering (Eaton, Elaluf-Calderwood, Sørensen, & Yoo, 2015). However, as noted by Schäfer (2011) and Eaton et al. (2015), while absorption responses may funnel existing external development efforts and enable the substantial growth of additional innovators, such responses are often rife with tensions. Therefore, absorption responses are typically achieved in parallel with attack responses (e.g., through litigation)

and monitoring (e.g., where product or service is rearchitected to curtail future unsanctioned innovation).

Furthermore, an organization may seek to *influence* the outlaw innovators instead. This tactic aims to persuade underground innovators to modify their innovations and pursue activities in a sanctioned manner. Such innovator behavior can be achieved by softer measures such as recognizing outlaw innovator work and refraining from litigation against innovators. Other means may include revealing the source code of a hacked product more openly while offering different types of software tools that lower participation barriers and encourage alignment with organizational objectives, which represents the focus of this thesis.

1.2 Data Scraping

A prevailing challenge for the Swedish public transport industry involves providing timely and correct information to its passengers. Since traveling via public transport requires the traveler to be at a specific place at a particular time, travelers have a pressing need for relevant and accurate real-time information about route alternatives, delays, and departure platforms. Following the societal adoption of smartphones and wireless internet, the IT infrastructure mediating such business-critical information to travelers has undergone a drastic transformation. More specifically, this transformation moved a significant proportion of public transport users away from information services developed by public transport agencies to services developed by external (and mostly unknown) actors developing top-rated smartphone apps.

This development came as a surprise to most public transport actors since they did not provide third-party developer resources (e.g., APIs and associated administrative legislation). Instead, these external developers have relied on a technique known as *scraping* to fuel their apps. Scraping can be described as application development based on resources designed for purposes other than application development. Scraping can be directed toward a multitude of official sources of available information, such as web pages, PDF documents, or reverse-engineered programmable interfaces (as per the aforementioned case of Tesla). By using scraped data, third-party

developers may fuel applications in the absence of official third-party resources.

Since third-party development based on scraping occurs without organizational consent, some organizations view such development as malicious and infringing on their intellectual property rights. Thus, to safeguard against scraping, many public transport actors have implemented a technical layer of protection on top of their web servers to curtail such unsolicited data retrieval. Some public transport actors have gone even further and taken legal measures against third-party development based on scraping[8].

However, some organizations have taken less confrontative measures toward scrapers. In late 2011, I conducted two studies investigating scraping and related developer practices (Rudmark, 2013; Rudmark, Arnestrand, & Avital, 2012) and was subsequently offered to lead a team of experts in developing a new real-time railway data API platform at the Swedish Transport Administration (henceforth the STA). At that time, the 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, several rail-related apps that relied on scraping had emerged. These apps were written by independent developers and primarily driven by self-experienced needs. Notably, a handful of these apps gained a high number of downloads in app marketplaces (e.g., Google Play, Apple App Store).

At that point in time, the STA was interested in designing resources that would fit the needs of these developers. Early on in our cooperation, two central ideas stood out in their approach. First, in the spirit of the open data movement, the platform should be open for anyone to use. Second, the STA did not seek to coerce anyone to

[8] In 2010 the Belgian National Railway Company (NMBS/SNCB) sent a cease-and-desist letter to the non-profit initiative iRail urging them to stop scrape data from the NMBS/SNCB web site (https://yeri.be/stopping-irail-be). Moreover, New York's Metropolitan Transportation Authority (MTA) took legal actions towards the scraping-based iPhone app StationStops in 2009 (http://readwrite.com/2009/08/20/ny_transportation_authority_cites_schedules_as_cop)

use their resources. Instead, they sought to design resources so that third-party developers voluntarily chose to use these resources rather than any other forced measures. With these ideas as a starting point, we embarked on a joint journey resulting in the concepts presented in this thesis.

1.3 Self-Resourcing Emulation

Starting in early 2012, I began to develop and materialize an initial theoretical perspective using action design research (henceforth ADR) (Sein, Henfridsson, Purao, Rossi, & Lindgren, 2011). This perspective integrates and extends the existing platform literature by emulating self-resourcing[9] behavior through the platform core, which I gradually shaped through the execution of two full ADR cycles together with the STA. Since mid-2014, the platform has been fully operational and is currently serving external and internal API clients.

The theoretical perspective developed over these two ADR iterations has been coined *open platform emulation*. *Open* refers to the platform offering the same capabilities and restrictions to any user, including the platform owner (de Reuver, Sørensen, & Basole, 2018; Eisenmann, Parker, & van Alstyne, 2009). *Platform emulation* refers to when an organization uses an external model to design a compatible platform with capabilities superior to the model. In this thesis, these external cues originate from self-resourcing. Moreover, platform emulation entails that the improved capabilities, is being achieved via the reorganization of an organization's digital recourses (Teece, Pisano, & Shuen, 1997, pp. 524-525).

Congruent with current platform theories (Gawer, 2014; Saadatmand, Lindgren, & Schultze, 2019; Tiwana, 2014), open platform emulation recognizes the interplay between a platform's governance and architecture. In the context of open platform emulation, *governance* refers to the desired platform capabilities. In open platform emulation these capabilities include coherence with past, proven

[9] Self-resourcing refers to "third-party developers' act of developing new boundary resources as a response to perceived limitations in existing boundary resources" (Ghazawneh & Henfridsson, 2013, p. 186).

solutions, as well as the flexibility to accommodate new development trajectories (Brunswicker & Schecter, 2019), alongside the strategies for applying openness to a digital resource (Karhu, Gustafsson, & Lyytinen, 2018). Consequently, *architecture* constitutes the means to achieve these desirable capabilities by reorganizing incumbent digital resources and creating design rules (Baldwin & Clark, 2000) that conveys these capabilities to third-party developers.

1.4 Research Objective

Based on the problematic situation at hand, I developed and materialized design knowledge for open platforms in an authentic setting within the STA. Thus, the research presented in this thesis has sprung from the following research question:

> *How can organizations emulate self-resourcing activities of third-party developers to design open platforms?*

Answering this research question using ADR adds to theory and practice in three ways (Sein et al., 2011, p. 42; Westin & Sein, 2015, p. 24). First, this thesis generates *design knowledge*. Such knowledge should convey both the process and product aspects in a sufficiently generalized form to allow for usage in other similar design contexts. Second, this thesis should generate *ensemble-specific contributions*. This type of contribution concerns an actual, sustained ensemble encompassing the IT artifact (ingrained by initial theoretical hypotheses and contextual structures) as well as modified organizational structures in which the ensemble artifact resides. Finally, this research should generate *end-user utility*. In the context of this research, such utility concerns superior platform capabilities, compared to self-resourcing, that influences outlaw innovators to choose sanctioned resources over unsanctioned ones.

1.5 Thesis Structure

This thesis is structured as follows. Chapter 2 details the theoretical framework underpinning open platform emulation. In Chapter 3, I provide a contextualizing overview of the research method of this thesis. Chapter 4 provides a process view on how the platform

materialized through an intricate interplay between my guidance and emergent environmental responses. In Chapter 5, I briefly describe the included papers, while Chapter 6 presents the design principles that answers the research question of this thesis. Finally, in Chapter 7, I discuss the additional theoretical implications of this research.

2 OPEN PLATFORM EMULATION

The term *emulation* dates back to the late 16th century and is borrowed from Latin, where the original word—*aemulātus*—means to vie with, rival, or imitate. Hence, the Oxford Dictionary defines emulation as "the endeavor to equal or surpass others in any achievement or quality" (Oxford English Dictionary, 2019). To illustrate a more precise meaning of emulation—albeit in a different field to information systems—one may consider an experiment in developmental psychology conducted by Tennie, Call, and Tomasello (2010).

2.1 Emulation Logics

In their study of chimpanzee learning, Tennie et al. (2010) conducted *the floating peanut experiment*. In this experiment, a peanut was placed at the bottom of a plexiglass tube that was wide enough to fit a peanut but too narrow and deep for the test subjects (chimpanzees) to grasp the peanut by hand. However, by pouring liquid into the tube, the peanut would start to float and ascend the tube until a chimpanzee can grab it. This experiment compared two groups of chimpanzees that observed a human solving this intricate peanut problem. The first group watched a human demonstrator using their mouth to pour water into the tube. After several such liquid-dispensing iterations, the human was able to grasp the floating peanut by hand. In the second group, the human demonstrator used a bottle instead, and the vessel's water was poured into the tube until the same result was achieved. However, since no bottles were available for the test subjects, the chimpanzees in the second test group had to employ different learning mechanisms.

The first set of chimpanzees to successfully complete the floating peanut task simply observed and *copied the action itself* (i.e., filling

their mouths with water) to achieve the desired result (i.e., picking up the peanut). Developmental psychologists describe this strategy as *imitation learning*. However, to successfully obtain the peanut, the second group had to employ learning strategies that focused on *copying the environmental result of an action* rather than the action itself. In practice, this meant that these chimpanzees filled the tube by dispensing water using their mouths despite having only seen someone fill the tube using a bottle. Hence, the latter form of observation learning has been conceptualized as *emulation* learning.

As illustrated in this example, emulation is an activity conducted *vis-à-vis* a similar phenomenon that the emulator seeks to mimic or transcend. Additionally, this example illustrates another important aspect of this research: how emulation is related—to but inherently distinct from—imitation. These concepts are related since both tactics are driven by achieving a similar and desirable environmental results. However, in imitation, the desirable results emanates from *replicating the underlying mechanisms to cause the desired result*. In contrast, emulation relies on the subject *seeking alternative ways of achieving the same, desired result*.

Besides its applications in developmental psychology (e.g., the aforementioned floating peanut example), emulation has been used as an explanatory construct across a range of disciplines. Perhaps the most widely known application of emulation is found in computing. Here, emulation refers to the act of achieving software runtime compatibility on a different set of hardware or software specifications than what a software application was originally designed for. The key to software emulation lies in designing software (or hardware) to behave *similarly enough* to allow the execution of the original software. When the runtime environment behaves similarly enough while relying on different underlying mechanisms (e.g., hardware and operating systems), software emulation can allow older applications to run for long after the original runtime environment has become obsolete (Tucker, 1965).

In organizational sociology, researchers have used inter-organizational emulation to theorize how to "equal or surpass a comparison organization or organizations on a set of strategic qualities or features" (Labianca, Fairbank, Thomas, Gioia, &

Umphress, 2001, p. 313). In this stream of literature, authors home in on the forces that shape organizations, which neo-institutional theorists refer to as *isomorphic processes* (DiMaggio & Powell, 1983; Gioia & Thomas, 1996). Here, emulation can be considered an instance of *mimetic* isomorphism, where the organization seeks to both mimic and transcend a model organization.

Moreover, emulation has been used as a construct within strategic management to explain and conceptualize interfirm mimicry. In this type of research, the fundamental difference of causality between imitation and emulation is stressed, as noted by Teece et al. (1997):

> *"Imitation occurs when firms discover and simply copy a firm's organizational routines and procedures. Emulation occurs when firms discover alternative ways of achieving the same functionality."*
> Teece et al. (1997. p. 524-525)

Typically, this body of literature emphasizes how firms organize to prevent or decelerate competitors' emulation activities (Teece, 2007). This is often achieved by deeply embedding contextual knowledge into organizational routines (Coff, Coff, & Eastvold, 2006; Pil & Cohen, 2006; Rivkin, 2001).

2.2 Platform Emulation

In this research, I use the logic of emulation as a new strategy for platform design. I refer to platform emulation when designers use an external model as basis for materializing platform capabilities, compatible with, yet superior to the model. Moreover, platform emulation hinges on using alternatives ways of achieving superior platform capabilities, compared to the model (Hartman & Teece, 1990; Teece et al., 1997, pp. 524-525). In this way, it is distinct from platform imitation, or platform forking (Karhu et al., 2018), since platform emulation depends on resembling capabilities rather than the replication of another platform's resources, as in platform forking. Thus, platform emulation is thus contingent on the capability to resemble and outperform the capabilities of third-party development resources in the organizational ecology, by repartitioning assets that the platform owner controls.

In line with current platform theories (Gawer, 2014; Saadatmand et al., 2019; Tiwana, 2014, p. 47; Tiwana, Konsynski, & Bush, 2010), I argue that successful platform emulation requires paying close attention to platform governance (since this regulates the platform's capabilities) and architecture (since this constitutes the possible ways of achieving these desirable capabilities), as well as the interplay between these two factors. However, a platform's *openness* is fundamental to its governance and architecture decisions (de Reuver et al., 2018; Eisenmann et al., 2009; Ondrus, Gannamaneni, & Lyytinen, 2015; Parker & Van Alstyne, 2017; West, 2003). Since this thesis investigates *open* platforms, I next expand on the more precise meaning of this phenomena.

2.3 Open Platforms

In the digital platform context, openness is not a Boolean construct. Instead, it is a choice regarding the *extent* and *dimensions* to which the platform should be open (West, 2003). Notably, this decision entails a critical trade-off (Parker & Van Alstyne, 2017).

More restricted openness may increase the platform owner's capacity to appropriate rent from complementary innovation and deter competition. On the other hand, organizations opting for more full-fledged openness (Brunswicker & Schecter, 2019; Karhu et al., 2018) may value complementary innovation and application output over rent appropriation potential. This position can be beneficial for organizations within the public sector (Bonina & Eaton, 2020; Mukhopadhyay, Bouwman, & Jaiswal, 2019), scientific research communities (Brunswicker & Schecter, 2019), and commercial platforms in early, formative phases (Karhu et al., 2018; Parker, Van Alstyne, & Choudary, 2016). Henceforth, I focus on platforms that have chosen to be fully open.

Platform openness is a complex construct that applies to several dimensions of a platform (Eisenmann et al., 2009; Ondrus et al., 2015). When a platform is open in the *sponsor* dimension, this implies that any actor can influence the platform's roadmap and engagement rules, often through providing additional development resources. If a platform is open in the *provider* dimension, this means that any actor may erect an instance of a particular platform that allows for

user interactions. Finally, a platform can be open in the *user* dimension, which implies that any user can choose to use the platform in any way s/he chooses[10]. Notably, this thesis is concerned with openness in the user dimension. For the remainder of this thesis, I will refer to platforms open at the user level as *open platforms*.

In this research, I merge and augment two existing definitions of open platforms. First, de Reuver et al. (2018, p. 127) defined platform openness as "the extent to which platform boundary resources support complements." Using this definition of an open platform would approximately correspond to "a platform whose boundary resources are completely open to complements." Second, Eisenmann et al. (2009, p. 131) posited that "[a] platform is 'open' to the extent that: (1) no restrictions are placed on participation in its development, commercialization or use (access); and (2) any restrictions (authority) are reasonable and non-discriminatory regarding entry requirements, requirements to conform with technical standards or payment of licensing fees."

In their definition of open platforms, de Reuver et al. (2018) emphasize *boundary resources* (Ghazawneh & Henfridsson, 2013). Following the theoretical development of boundary resources by Ghazawneh and Henfridsson (2013), their definition includes both the restrictions (as emphasized by Eisenmann et al. (2009)) and the *design capabilities* transferred to users (von Hippel & Katz, 2002). In other words, I argue that the definition of de Reuver et al. (2018) highlights that platform openness does not only concern the scope of *permissible* innovation (restrictions) but also *possible* innovation (design capabilities). On the other hand, Eisenmann et al. (2009) stress the non-discriminatory aspects of platforms that are open at the user level, which represents a fundamental aspect of open

[10] In their typology of roles and platform openness, Eisenmann et al. (2009) also distinguish between demand-side users and supply-side users. While this division is pertinent to two-sided platforms, this research is concerned with product platforms (Boudreau & Lakhani, 2009, p. 73), where the third-party developer is also the (only type of) first-hand user from the platform provider perspective.

platforms that is lacking in the openness definition of de Reuver et al. (2018).

In addition to the non-discriminatory transfer of both design capabilities under the same restrictions, a currently overlooked—or at least not explicit—issue of platform openness concerns platform usage by the platform owner *vis-a-vis* the platform's complementors. Current definitions of open platforms do not explicitly recognize that there should be no difference in what the platform owner and external third parties are allowed to do in a truly open platform. In summation, given the definitions of de Reuver et al. (2018) and Eisenmann et al. (2009) alongside the hitherto unmentioned aspect of equal platform usage by the platform owner and third parties, I use the following definition of open platforms in this thesis:

> *A platform that offers the same capabilities and restrictions to any user, including the platform owner.*

As previously mentioned in this thesis, I follow current theories on platforms that argue for the need to pay close attention to both platform governance and architecture when designing platforms. Therefore, in what follows, I present the prevalent aspects of governance and architecture in open platform emulation[11].

2.4 Governance Mechanisms

In platform emulation, platform owners focus on understanding and resembling the distinct capabilities that the platform must encompass. To successfully design emulated boundary resources, information on the non-negotiable features of used outlaw resources is critical. Here, the emulator must be cognizant of *de facto* usage and practices around these incumbent resources. Such issues include path dependence, the degree of compatibility with existing protocols

[11] Saadatmand et al. (2019) conducted a literature of papers taking a configurational perspectives, where Brunswicker and Schecter (2019) and Karhu et al. (2018) were the sole examples of open platforms. Since this publication, we identified O'Mahony and Karp (2020) to use a configurational perspective on open platforms. However, given their focus on collective governance, I did not include this study in our kernel theory.

and other technologies, and the community values surrounding third-party development. The emulator must also mindfully carve out room for superior capabilities that can convince complementors to switch platforms. A fundamental determinant of creating such capabilities is the associated governance principles.

In terms of the more precise meaning of governance, I follow the definition of Foerderer, Kude, Schuetz, and Heinzl (2019), who defined platform governance as:

> *"the fundamental decisions of platform owners with regards to the ecosystem of complementors." (Foerderer et al., 2019, p. 121)*

Among these, I elaborate on two governance aspects critical to platform emulation: a) platform capabilities in terms of solution search mechanisms and b) how an emulator chooses to open a platform to third-party developers.

2.4.1 Flexible and Coherent Searches

The first decision concerns how a platform can reconcile tensions that emerge from the need to both maintain stability (to decrease coordination and enable value capture by complementors) and simultaneously allow the platform to expand into new territories (Dattée, Alexy, & Autio, 2018; Kapoor & Agarwal, 2017; Saadatmand et al., 2019; Tilson, Lyytinen, & Sørensen, 2010; Wareham, Fox, & Cano Giner, 2014),

In their study of the platform nanoHub, Brunswicker and Schecter (2019) found promising paths to reconcile this dilemma on open platforms. They argued that individual developers may mitigate the stability-change tension in the platform periphery. By platform periphery, Brunswicker and Schecter (2019) refer to a platform ecosystem with a stable core and a periphery of complements. The platform core contains a set of central components with stable interfaces, while the complements should exhibit variety (Baldwin & Woodard, 2009). As an example, consider the setting of a traveler information platform ecosystem within the public transport industry (as per this research). Such a platform typically consists of core functions such as geocoding (e.g., the ability to transform an address or point of interest into geographical coordinates), travel planning

functionality (e.g., present travel routes based on origin and destination information), and real-time departure information (for a specific stop or station). These functions are likely to remain stable over time, and the interfaces can remain untouched even if the underlying traveler information system is replaced. However, the complements are likely to vary significantly over time. In this example, such variety could manifest itself in applications on different platforms (e.g., smartphones, watches, web pages), user groups (e.g., everyday travelers, tourists, riders with disabilities), and contexts (e.g., travel planning, waiting for a connection, *en route*). In some platform ecosystems, this periphery of complements is *open* (i.e., it is possible for developers to both contribute to and inspect the apps). Under such circumstances, Brunswicker and Schecter (2019) found that two types of developer search strategies unfold as third parties develop apps to meet user needs.

First, developers typically enact a *coherent search* strategy. By coherent search, Brunswicker and Schecter (2019) refer to a developer being guided by past experiences and known solutions to prevalent problems. By searching for solutions that are coherent with past experiences, developers are likely to identify solutions characterized by stability and reuse potential. Hence, to continue the public transport example, consider a tailored departure board as a complement. Such displays are typically found in venues like bus stops, shopping malls, and station restaurants. To achieve a consistent user experience, developers seek to present information in a similar fashion. If a hypothetical platform ecosystem within public transport was open in the *periphery*, this would entail having the most widely used departure board app(s) being licensed as open source. Consequently, any developer could simply copy that proven solution and rework the code into his/her app. Accordingly, together with the coherent search enactments, the open periphery helps to maintain the platform's stability. However, while such coherent searches can help maintain a platform's stability, an excessively one-sided focus on coherent searches will hamper changes in the platform ecosystems that are necessary for the platform to remain attractive.

Thus, Brunswicker and Schecter (2019) point to a *flexible search* strategy as a second strategy. This type of search strategy involves a

developer exploring unexploited solution spaces to meet anticipated user needs. Flexible searches may be triggered by new user needs, emerging technological capabilities, or market trends. While the flexible search strategy explores new territory, Brunswicker and Schecter (2019) posit that flexible searches branching out of solutions coherent with the past are more likely to be successful than those lacking such a connection with the past. To exemplify this type of flexible search in the public transport example, one could consider a developer attempting to bring the departure board concept to a much more technologically constrained environment, such as an LED/OLED display[12]. These types of displays typically require more low-level manipulation than a web-based equivalent. Consequently, to resolve this challenge, a developer cannot merely reuse an existing departure board layout code but will need to seek novel solutions. However, while code reuse may be limited, the developer may take inspiration from some of the layout used in existing complements and branch out from coherent searches.

Since coherent-flexible searches occur at the platform periphery, these search mechanisms unfold within the micro-architecture of an app. Therefore, the architectural implications from this work suggest that platform designers should facilitate developer movement across complements, maintain complement openness, and promote technology reuse across apps.

2.4.2 Access and Resource Openness

The second core aspect of open platform emulation governance concerns *how* to open up a platform for outside access. Here, the literature offers two principal strategies to achieve platform openness: *access openness* and *resource openness* (Boudreau, 2010; Karhu et al., 2018).

In access openness, a platform is opened by granting access to selected parts of the platform's core. Thus, Karhu et al. (2018) refer to access openness as:

[12] Some model railway enthusiasts like to incorporate live train departure information into their models.

> *"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)*

Through access openness, the platform host may thus choose what parts of, in what form, and under which intellectual property (IP) regime external users can use the platform. This way, access openness provides the platform owner with additional flexibility in the future use trajectory. When governing third-party development through access openness, boundary resources (Ghazawneh & Henfridsson, 2013) play a key role. Boundary resources constitute the thin layer of assets that both capacitate and confine third-party complements. These resources include the APIs, Software Development Kits (SDKs), license terms, and testing tools that enable third-party developers to interact with a platform alongside the design rules for doing so.

Within a multisided platform, access openness typically is instigated to allow for supply-side users to create complements for the platform's demand-side users. For instance, Apple provides access to a vast amount—albeit not all—of the capabilities of iPhones through the iOS SDK. While the SDK allows developers to access functions such as the current location and internet connection, Apple blocked access for third-party apps to use the phone's near field communication (NFC) chip long after the chip was installed on new phones. It is believed that this was done to help build the Apple Pay user base[13] without competition in the IOS platform ecosystem. Once sufficient momentum was created for Apple Pay, Apple initiated a stepwise strategy to also allow third parties to develop NFC-compliant apps. However, to date, Apple Pay remains the only NFC-compliant payment provider on the iPhone. In summation, by governing the platform through access openness, Apple has been able to mobilize third-party complements using NFC while maintaining a monopoly on NFC in-store payments.

[13] https://venturebeat.com/2017/05/02/apple-pay-transactions-rose-450-in-the-last-year/

The second principal method for opening a platform is through *resource openness*. This method implies the platform core being made available to users, and where particular importance is put on the governing *intellectual property rights* (IPR) of the platform. Karhu et al. (2018) refer to resource openness as:

> *"Opening the platform's valuable resources by forfeiting the IPR of the resource."*
> *(Karhu et al., 2018, p. 481)*

Hence, resource openness is closely associated with the platform governance that makes a platform core readily available for users. This way, a platform host may achieve greater uptake since the legal barriers to reusing code have been removed. However, the platform owner has limited means of controlling the continued evolution of the platform. As an example of resource openness, one can consider the Android mobile handset platform, which is the greatest competitor to iOS. In stark contrast to iOS, the Android operating system is open source, meaning that all operating system functionality is readily available for third-party developers. As a consequence, when Android implemented operating system support for NFC chips any third-party developer could immediately start exploiting this hardware innovation. As a result, there are many providers of NFC payment apps on Android besides Google, including tech giants such as Facebook, PayPal, and Samsung alongside an array of innovative start-ups.

2.5 Technology Architectures

Platform emulation is fundamentally contingent on transforming an organization's resources to outperform the existing ecosystem's capabilities. In this context, *platform architecture* constitutes the necessary means to reorganize incumbent digital resources and redistribute design capabilities to third-party developers.

As previously mentioned, a digital platform's architecture consists of a stable modular platform core, standardized visible interfaces, and peripheral applications (Baldwin & Woodard, 2009; Karhu et al., 2018; Saadatmand et al., 2019). To enable the seamless addition of new complements, the platform core must be modular and thus draw on the principle of information hiding (Parnas, 1972). Information

hiding posits that modular system designers should ensure that only necessary information is available to modules' users in order to reduce dependencies and better accommodate change.

2.5.1 Platform Design Rules

Since a platform core involves information hiding, platform module users can only act on a module's visible information. This visible information has been conceptualized as *design rules* (Baldwin & Clark, 2000) and constitutes the ways in which module developers can establish compatibility with a platform. As such, two important characteristics mark successful design rules (Tiwana et al., 2010). First, design rules should be stable over time to ensure that module developers, regardless of when they enter, can make the same assumptions around functionality and interface specifications. Second, design rules should be sufficiently versatile (e.g., not forestalling an ecosystem's variety and performance). Following (Baldwin & Clark, 2000, p. 77), a complete set of design rules has the following constituents:

- *Architecture* – A blueprint of existing modules within the systems, including their roles and relationships.
- *Interfaces* – Describe how a specific module behaves when, for example, a module's API is invoked. This includes what parameters are required to achieve this behavior.
- *Integration protocols and testing standards* – Allow a module designer to fit his/her app to the platform's interfaces and determine whether the app works sufficiently well.

Open platform emulation requires that existing modules are reorganized to achieve the desired capabilities. A part of this reorganization corresponds to the design rules *architecture* and constitutes the visible modules that external developers can interact with (Jha & Pinsonneault, 2016; Kapoor & Agarwal, 2017; Kazan, Tan, Lim, Sørensen, & Damsgaard, 2018). Within such modular systems, Baldwin and Clark (2000) suggest that *modular operators* play a key role. These operators act as a discrete set of possibilities through which designers may alter the architecture of modular systems. In this sense, a platform designer may draw on modular operators as:

> *"actions that change existing structures into new structures in well-defined ways."*
> (Baldwin and Clark (2000, p. 129)

Within such an architectural redesign, there are several such operators that a platform designer can apply to evolve a platform. Of interest for this research are the following operators[14]:

- By *inverting,* a designer may create modules that publishes information that is widely used or requested but has been previously hidden.
- Through *substituting*, a platform designer may replace existing modules with those having improved qualities.
- Finally, by *mutating* modules, designers can copy existing modules for usage in other application domains (Karhu et al., 2018; Tiwana, 2014, p. 195) [15].

Moreover, design rules also require visible *interfaces* specifying the behavior of modules in a platform. As such, the interfaces act as a description of what the platform provides for third-party developers (Parnas, Clements, & Weiss, 1985) and thus conveys the boundaries of possible platform innovation. From an architectural perspective, two important decisions stand out for interface design: the degree of app-platform decoupling and interface standards (Tiwana, 2014, pp. 106-114). Interface decoupling occurs when a designer minimizes the visible information by increasing a module's encapsulation of internal complexities (Ethiraj & Levinthal, 2004). Such designs decrease dependencies between the platform and its apps, thereby making integration and testing more straightforward—especially for new platform developers. However, a drawback from far-reaching decoupling is the risk of hampering third-party developer experiment opportunities (Tiwana, 2014, p. 105). Interface standards

[14] While the set of six original modular operators (splitting, substituting, augmenting, excluding, inverting, and porting) suggested Baldwin and Clark (2000) apply to any modular systems, additional modular operators for the digital platform context have been identified (Karhu et al., 2018; Tiwana, 2014, pp. 191-196).

[15] This same operator has been conceptualized as *cloning* by Karhu et al. (2018)

entail how interfaces materialize on the platform. In this vein, considerations concern communication protocols, compliance with existing industry standards, and versatility.

Finally, design rules hinge on the use of an *integration protocol and testing standards*. These aspects of the design rules concern additional information that allows third-party developers to connect a platform's core interfaces to those of an app's micro-architecture. Typically, these aspects of a platform's design rules manifest themselves as SDKs, integrated development environments (IDEs), or code examples. As such, these extensions target developers during the app design process by, for example, providing entry paths for new platform developers (e.g., code examples), simulating the runtime environment, and ensuring compatibility with specific devices (Evans, Hagiu, & Schmalensee, 2006; Tiwana, 2014). As such, platform complexities (Cennamo, Ozalp, & Kretschmer, 2018) can be encapsulated to minimize third-party developers' coordination costs (Tiwana, 2015).

3 RESEARCH METHOD

In this thesis, I have used ADR (Sein et al., 2011) to investigate and answer the research question presented in Chapter 1.4. This chapter details the more practical aspects of the included activities in this research.

In ADR, contributions come in a threefold package (Sein et al., 2011, p. 42; Westin & Sein, 2015, p. 24). As in other design science research approaches, the project should generate *design principles* that convey the necessary and sufficiently generalized design knowledge for use in other similar design contexts. The second part, which is more specific to ADR projects, is *ensemble-specific contributions*. This part of the contribution constitutes both the resulting artifact (ingrained by initial theoretical hypotheses and contextual structures) and the modified organizational structures where the ensemble artifact resides. The final ADR result relates to the *end-user utility* that emerges when the artifact is put into use.

Regarding design principles, I have opted for articulating design principles in accordance with the recommendations of Gregor, Kruse, and Seidel (2020). These authors stress the need for a schema that streamlines the explication of design principles while maintaining flexibility toward context-specific design situations. To this end, the schema includes four statements covering a total of six aspects of design principle formulation. These include:

1. The objective of the design principle, who is the intended designer, and who is the prospective user (aim, implementer, and user, respectively).
2. The boundary conditions for when the design principle is applicable (context).
3. The causal workings that help to accomplish the aim (mechanisms).

4. The theoretical and empirical justification for why the principle holds true (rationale).

To align with ADR's epistemological assumptions (Iivari, 2015) and deliver design principles, ensemble-specific contributions, and end-user utility, ADR researchers must situate artifacts within truly authentic settings. More specifically, ADR recognizes that since an IT artifact is always embedded in some context (Orlikowski & Iacono, 2001), it serves as a carrier of structures from its surrounding ecology. These structures are inscribed into the artifact by both designers and users, evolve over time, and add to subsequent design theorizing. Consequently, ADR's technology perspective requires that researchers possess sufficient contextual knowledge, resources, and legitimacy within the target ensemble environment to both initiate and continuously shape ensemble artifacts.

This chapter describes the more practical aspects of this journey. Chapter 3.1 details the research activities that gradually both increased my understanding of the study context but also helped me build sufficient credibility within the target environment. Then, in Chapters 3.2 and 3.3, I provide background information for two full ADR cycles that closely follows the ideal model of ADR presented by Sein et al. (2011) and builds on the converged insights from the previous activities. Finally, Chapter 3.4 describes how I followed the platform's continued trajectory after I had exited as an active designer.

3.1 Design Antecedent

3.1.1 The DART Group: Verifying Action Design Research

The starting point for the knowledge developed in this thesis emerged in August 2009 as I began investigating methods of publishing transport-related data for third-party developers. The organizational nexus of this investigation was a regional working group called "Regional deployment of traffic informatics" [16] (DART) in Gothenburg, Sweden. The group contained regional representatives from the city of Gothenburg, the regional public

[16] "Driftsättning av regional trafikinformatik" (Swedish)

transport authority (Västtrafik), and the Swedish Transport Administration. Although it was not a focus at that time, members of the DART group brought up scraping as an unresolved issue from the very start since the involved organizations had been subjected to different forms of self-resourcing. As part of a research initiative, I headed a work package responsible for creating a developer platform to support the emerging third-party developers identified in prior stages. The guiding vision was to create an inviting environment where extra-industrial actors could develop services supporting sustainable everyday travel. Since this approach was novel to the working group and undertheorized, we agreed that situated design research would be most appropriate for materializing a new ensemble artifact. In parallel to attending these monthly meetings, I interviewed both stakeholders within DART's organizations as well as third-party developers (some of which rely on self-resourcing). To make sense of the data, I imported that transcribed material into atlas.ti and coded the data inductively. First, data were analyzed inductively based on the methods of Strauss and Corbin (1990). Here, I sought to diagnose the current state of affairs in the local practice without forcing my own preconceptions onto the data. The relationships between codes were established and a detailed snapshot of the current struggles of the working group with respect to novel system development approaches emerged (see Appendix A for example). As the current situation became more articulated, I detailed the historical processes and events that led to the current gaps. To this end, I used follow-up interviews and reports and coded them using the theoretical raster of punctuated information systems (information systems) change (Lyytinen & Newman, 2008). Finally, the reconstruction of historical events was triangulated using interviews with former employees and external developers. Interviews with external developers also provided important cues for the upcoming developer platform design.

From this analysis (Rudmark & Lind, 2011), I found that the DART group primarily sought to share data that could help facilitate sustainable transportation. On the other hand, developers were seeking APIs that easily could be used in a mobile context to map to their problems and technological standards (e.g., APIs using the geographical coordinates native to smartphones). Based on these insights, I refined the developer platform design through two joint

workshops with various stakeholders representing both public transportation and various third-party developers. However, while the DART group indeed embraced these tentative results, they were not in a position to host the projected ensemble artifact. While the DART collaboration supported the notion of using design research as a basis for third-party developer platforms, the quest to find a more appropriate organizational context for situated design research continued.

EMPIRICAL DATA	N	SCOPE
Interviews with DART Group	10 interviews	Total minutes: 644 Total words: 95220
Interviews with third-party developers	12 interviews (20 people)	Total minutes: 711 Total words: 117309
Internal DART meetings	5 meetings	Total minutes: 440
Secondary data	7 reports	Total pages: 246 Total words: 65862
Workshops with DART and third-party developers	2 workshops	Total mins: 540

Table 1 - Empirical material related to the DART group (2009-09 - 2010-06)

3.1.2 Trafiklab.se: Industry Platform Openness

In the wake of the discontinued research collaboration with DART, I began to engage with Samtrafiken (the Swedish Association for Public Transport Companies). At this point (autumn 2010), Samtrafiken had advanced plans to deploy an industry-wide API platform for third-party developers, later named Trafiklab.se. In addition to being a data-sharing platform for its founding members (Stockholm Public Transport (SL) and Samtrafiken), Trafiklab.se had

a clear ambition to become a concerted effort for the entire Swedish public transport industry regarding data openness *vis-a-vis* external innovators. Therefore, in addition to hosting APIs from SL and Samtrafiken, the platform was a core component when the public transport industry sought to boost innovation and extend their digital infrastructure. Consequently, Trafiklab.se was used to innovate new services via innovation contests such as TravelHack (see Chapter 3.1.4).

Moreover, the industry employed Trafiklab.se when Google requested the underlying data resources (such as stops, routes, and schedules) since Google would not settle for indirect access through travel planning services. This series of events triggered a deeper investigation, which was analyzed using the procedures outlined in Koutsikouri, Lindgren, Henfridsson, and Rudmark (2018). Another important learning opportunity that emerged in this episode was that between September 2010 and February 2012, the product owner of Trafiklab.se was a part-time employee at the research institute where I worked. Thus, many insights gained about the industry emanated from a vast amount of informal and continuous communications throughout this period. Hence, through my continued close collaboration with Trafiklab.se, I was exposed to various configurations of platform governance and architecture within the industry.

EMPIRICAL DATA	N	SCOPE
Interviews with industry representatives	4 interviews	Total minutes: 206 Total words: 28939
Workshops with Trafiklab before release	2 workshops	Total minutes: 440
Meetings Trafiklab after release	11 meetings	Total minutes: 2370
Continuous and informal discussions with the product manager of Trafiklab.se		

Table 2 - Empirical material related to Trafiklab.se (2010-08 - 2011-12)

3.1.3 SL: Scraping trajectories

One of the leading actors in Trafiklab was SL. Accordingly, the collaboration with Trafiklab.se enabled me to map SL's trajectory regarding third-party development. By primarily using interviews with public transport representatives and self-resourcing third-party developers, I was able to position critical events on a timeline. Moreover, I took particular interest in analyzing SLs and third-party developer responses to these critical events. This inquiry (Rudmark et al., 2012) revealed that SL had been subjected to scraping for an extended period. Indeed, one of the primary reasons for launching a new platform was to resolve issues related to scraping. Under the sway of data scrapers and hundreds of thousands of end users, SL had started to open their internal systems for external innovators. Finally, in addition to inquiring into SL's past experiences, I was also able to follow the initial developer adoption of Trafiklab after the platform's launch, which took place in September 2011.

EMPIRICAL DATA	N	SCOPE
Interviews with the public transport industry	11 interviews	Total minutes: 491 Total words: 68171
Interviews with third-party developers	4 interviews	Tot mins: 206 Tot words: 28939
Database with registered users at Trafiklab.se	1 database	Tot users: 506 (per 2011-12-31)

Table 3 - Empirical material related to SL (2010-08 - 2011-12)

3.1.4 TravelHack: Exploring Developer Practices

The launch of Trafiklab.se materialized as a Digital Innovation Contest[17] designed by a team of researchers, of which I was part. The contest was called West Coast TravelHack 2011 and was designed as a development contest lasting 24 hours. The competition's goal was to generate prototypes for innovative digital services supporting citizens in their everyday travel. More specifically, the prototypes should help travelers make more sustainable choices in their daily journeys (e.g., choosing car-sharing over lone driving, public transport over car-sharing, or bicycling over public transport). Since the Swedish public transport industry backed this contest, it wielded significant legitimacy and attracted over 70 developers.

In addition to co-designing the overall contest, my task was to coordinate and consolidate APIs and other data sources available to the teams during the event. Through my previous engagement with the public transport industry in general, and SL in particular, it became clear that a more detailed understanding of technology use was necessary to address scraping. Consequently, I wanted to use TravelHack as a situated opportunity to observe and understand how API appropriation—and possibly self-resourcing—unfolded in real-

[17] See Hjalmarsson and Rudmark (2012) for more background information.

time. During the contest, third-party developers could choose from a wide array (17) of public transport-related APIs rooted in different design paradigms potentially yielding different adoption patterns among developers. Consequently, I contacted four teams before the contest and inquired whether I could video record their work. Of these four teams, three provided insights relevant to my objective. Afterward, I screened the approx. 57 hours of video I had gathered during the event. In this rich empirical material, I used the video research software Transana to identify and code 51 incidents where the developers used or attempted to use the available APIs, which shaped the future trajectory of the respective teams' API appropriation. More specifically, I created a dedicated clip for each incident that captured the videotaped incident and transcribed the dialogue. Next, I coded each incident with instances of related coding families, each of which captured important context and process characteristics connected to the incident. For example, this included which API caused the incident, how the incident manifested itself, and how it was resolved (see Appendix B). In this manner, I was able to observe and compare different API appropriations as they occurred rather than the retrospective accounts I had been gathering earlier through interviews. Also, I witnessed several instances of self-resourcing in certain cases, even in the presence of official APIs. As complementary data, I also collected the final source codes from two teams, alongside the server log files of commonly used APIs.

EMPIRICAL DATA	N	SCOPE
Video observations	3 teams	Tot mins: 3420 Tot incidents: 51
Data sources	17	n/a
Team submission application source code	2	Total lines of code: 29652
API log files	5	Tot log entries: 245829

Table 4 - Empirical material related to West Coast TravelHack 2011 (2011-10)

3.1.5 The Swedish Transport Administration: The Alpha version Platform

Together with the aforementioned institute colleague[18], I received a funding opportunity from Vinnova[19] in December 2011 to engage with other public transport industry actors in designing and launching APIs. At this point, we contacted the STA since they were subjected to scraping and would thus potentially benefit from participating in a study on resolving scraping. Moreover, they would provide an excellent research venue to develop corresponding design knowledge. At that time (January 2012), the STA could not commit to a full ADR ensemble artifact implementation upfront. Instead, the initial agreement allowed us to conduct a detailed problem formulation and present a solution blueprint (corresponding to an ensemble artifact alpha version (Sein et al., 2011)). After we had presented the alpha version, the project would enter a stage-gate. Here, the STA would decide whether to proceed and form a full-fledged ADR team and release a working platform or decline further collaboration.

In this phase, I conducted a round of interviews with key personnel at the STA to understand current strategies, challenges, and plans within the administration. Also, I started to investigate the most popular consumer-facing third-party apps that were available based on self-resourcing from third-party developers. Most of these developers also agreed to participate in in-depth interviews. For the analysis, I imported the transcribed interviews into Atlas.ti and coded the material according to the interview protocol categories (see Appendix C and Appendix D). This way, I was able to compare and highlight discrepancies between the current third-party developer program at the STA and the platform capabilities sought by self-resourcing third-party developers. A workshop concluded this part of the research and served as an *ex ante* formative evaluation (Venable, Pries-Heje, & Baskerville, 2016). In this workshop, we presented the alpha version to the participants, including critical stakeholders within the STA and prominent third-party developers (currently exercising self-resourcing). After compiling the workshop

[18] We conducted the activities collaboratively throughout this phase.

[19] Sweden's innovation agency.

results and presenting them for the STA, they decided to continue our collaboration and participate in beta version development.

EMPIRICAL DATA	N	SCOPE
Interviews with the STA	7	Total minutes: 382 Total words: 56204
Interviews with third-party developers	6	Total minutes: 313 Total words: 39925
Examining the existing data source of self-resourcing third-party developers	4	N/A
Workshops	1	Total minutes: 390

Table 5 - Empirical material related to the alpha version (Jan 2012–April 2012)

3.2 ADR Cycle 1

As a basis for the beta version artifact, I converged the rich insights gained from my previous thesis work. My initial work with DART provided sufficient evidence that situated design research (e.g., ADR) as a viable platform design method by working closely with both platform owners and third-party developers. The time I spent with Trafiklab.se provided me with a deeper understanding of third-party development and the impact of self-resourcing. Moreover, immersing in this particular emergent platform context convinced me that it was an excellent digital infrastructure for materializing ADR ensemble artifacts within the public transport industry. Moreover, the data collected from the API use experiment conducted at TravelHack allowed me to further develop my theoretical framework regarding platform emulation. Finally, our initial inquiry into the STA allowed for a refined design framework and yielded a substantial researcher-client agreement that ensured the necessary commitment from the STA.

Thus, I headed the team responsible for the first full cycle of ADR in May 2012. The ADR team consisted of a product manager, a systems manager, and a systems architect from Trafiklab.se alongside two systems managers and a systems developer from the STA. The overall solution architecture offered two APIs: one for known coherent searches and one for non-deterministic flexible searches. These APIs were delivered using the technical infrastructure of Trafiklab.se by providing access openness to an internal system within the STA called Orion[20].

First, we detailed the interface specifications based on the previous feedback and the team's domain knowledge. Consequently, we continued to implement the platform according to the specifications during the summer and autumn of 2012.

The ADR team officially released the solution on October 25, 2012. The solution allowed anyone to register for the API. In 3 months, a total of 59 developers had registered. To evaluate the ensemble artifact formatively *ex post* (Venable et al., 2016), I contacted these registered developers to inquire as to whether they would like to participate in an interview for evaluation purposes. Out of these 59 developers, 17 developers agreed. I then interviewed these 17 developers following an interview protocol (see Appendix E).

[20] Orion's API was subjected to self-resourcing since it had been made available through the STA website.

EMPIRICAL DATA	N	SCOPE
Project coordination meetings	18	Total minutes: 1650
Third-party application analysis	6	n/a
Interface specification online discussions	28	Tot words: 2991
Evaluation interviews with third-party developers	17	Total minutes: 604 Total words: 66930
Emails third-party developers (SL)	4	Total words: 1587

Table 6 - Empirical material related to the beta version (May 2012–January 2013)

3.3 ADR Cycle 2

Overall, the STA assessed the experiment as quite successful and thus decided to develop a permanent solution based on the findings from the ADR project. As a tangible result of the beta version, they altered their third-party developer strategy. More specifically, the strategy now included a new segment "other developers" that the new platform would target and serve. Consequently, the resources necessary for this iteration were provided solely by the STA and we reorganized responsibilities as a result. The STA systems manager that had been part of the beta version ADR team now became the team lead, and a systems architect was assigned to the group. I was contracted to this team through a direct award to ensure that the previous learnings were included in the platform's release version and was being responsible for conducting situated evaluations.

This second ADR cycle focused on creating a more sustainable solution while also addressing the shortcomings identified in the previous ADR iteration. During the beta version evaluation interview analysis, I found that existing developers had mostly not switched from scraping to open APIs. This surprising fact caused me to investigate the current scraping situations at SL and Trafiklab.se, given that this platform now had been operating for some two years.

To clarify what data sources third-party developers used to access SL data, I chose to investigate a particular subset of third-party services, namely real-time departure information for stations and bus stops used by smartphone apps. I opted for this subset since both of these services had caused the system breakdowns and were thus critical for SL to bind to sanctioned data sources. To find such services, I searched three major app marketplaces (Apple App Store, Google Play, and Windows Marketplace) for apps that used real-time information. By using commonly used keywords such as the name of the public transit company and other general transit-related keywords, I compiled a gross set of services (51 services). When investigating these services in greater detail, many fell outside of the criterion. Such services (which are outside the scope of my investigation) dealt with other aspects of public transit information, such as offline subway maps, travel planning capabilities, ticketing, and subway station maps. After this analysis, I found 19 services across the marketplaces using real-time departure information. As a next step, I wanted to trace the remote procedure calls that the apps made to obtain the data. To this end, I installed the 19 services on three different handheld devices (one for each marketplace).

Moreover, I installed an HTTP proxy called Charles Proxy on a laptop and configured the smartphones to access the internet through this proxy. This way, I was able to trace the requests made by each smartphone and could thereby determine the data source for each particular smartphone app. This procedure allowed me to determine the data sources for 14 of the 19 apps. Since the remaining 5 apps did not access SL directly and instead made their requests to app-specific servers, I could not determine the origin of the data displayed in these apps. Hence, I contacted these developers and inquired into the data sources used by their services. Through these email conversations, I determined an additional four of the five remaining services (since one developer did not respond to my request).

Moreover, given the exhaustive beta version feedback from third-party developers, my role was to ensure that these requirements were considered throughout the implementation. To this end, I provided feedback in the project meetings and performed a formative expert evaluation during the design of the new platform.

Additionally, I planned the evaluation of the release version platform. First, and most importantly, wanted to understand whether experienced existing developers had switched (or were considering switching) platforms. Second, less experienced developers that were likely to implement coherent searches were also important. Therefore, before launching a tentative release version to the public, I codesigned a usability test with a human-computer interaction expert at my research institute to target novice users.

The test took 4 hours and was conducted in December 2013. The test subject group included second- and third-year information system majors, and the participants had not been given access to any materials before the test. The test aimed to capture a beginner's potential to use the interfaces as well as their impressions of this process. The participants were given three tasks to perform: locate data, call APIs, and build a crude application. Although not all students were able to complete the exercises, they independently stated that the coherent search examples were imperative to their productiveness. Additionally, the test yielded a handful of bugs and minor adjustments.

The platform was launched on February 10, 2014, as an open test environment. This launch meant that any user registered at Trafiklab.se could use the API if they applied for access by email. During this test period, 20 developers registered (including several existing railway data developers), and 6 of them agreed to be interviewed. Based on this feedback, the platform went live on March 18, 2014. As the final task in ADR loop 2, I ,in August 2014 interviewed another six developers that had registered as users of the platform in a summative evaluation *ex post* (Venable et al., 2016). All of these interviews were conducted following virtually the same protocol used for the beta version evaluations (see Appendix F). These interviews pointed clearly towards that the interviewed developers were content with the released version platform.

EMPIRICAL DATA	N	SCOPE
Project coordination meetings	18	Total minutes: 1140
Open feedback discussion forum posts	34	Total words: 3033
Evaluation interviews with third-party developers	12	Total minutes: 543 Total words: 69521
Novice user test	13	Total survey responses (before/after each task): 1255 Self-reported critical incidents:9 Post-test interviews: 3
Analyzed third-party apps using SL data	51(gross) 19 (net)	N/A

Table 7 - Empirical material related to the release version (April 2013-August 2014)

3.4 Design Outcome

After leaving my position as an active designer of STA's third-party developer platform, I continued to follow its trajectory in several respects.

In September 2016, I conducted a data source experiment following the same procedures used in ADR loop 1 (see Chapter 3.2) to investigate the actual data sources of the apps available in major app stores. Moreover, following the previous data source experiment, I contacted the app developers (5 developers managing a total of 11 apps) to inquire about data sources if an app's data source was unable to be determined programmatically.

Between August 2016 and March 2018, I was part of a project group tasked with designing a new infrastructure for open data targeting the entire Swedish public transport industry[21], in which the STA also participated. In this work, the open platform approach, as implemented by the STA, became a role model for resolving service level agreements (SLAs).

Third, I conducted a targeted follow-up study within the STA. This inquiry encompassed four follow-up interviews (two in October 2018 and two in May 2020) with key personnel within the STA. Here, I inquired into what had happened with the platform and its reception since its launch. More specifically, I questioned my informants about items in a changelog of the platform's public functionalities, the rationales behind them, and potential connections to the emulation approach. Finally, I collected the usage statistics of the platform.

In addition, this stage encompassed the formalization of learning stage in ADR (Sein et al., 2011), that yielded both the design principles in Chapter 6 and the additional theoretical implications presented in Chapter 7.

[21] See Arnestrand, Lundh, Rudmark, and Östlund (2017) for further details.

EMPIRICAL DATA	N	LENGTH
Analyzed third-party applications using STA data	35 (gross) 28 (net)	n/a
Emails third-party developers on data source	4	Tot words: 1587
Interview third-party developers on data source	2	Tot mins: 32
Workshops with the Swedish Public Transport Industry	6	Tot mins: 1800
Interviews with key STA personnel	4	Tot mins: 374 Tot words: 55787
Platform changelog entries	19	Tot words: 281
Usage statistics spreadsheet	1	Number of API calls between 2015 and 2020, separated on internal and external calls

Table 8- Empirical material related to the maintenance version (September 2014-April 2021)

While this chapter has outlined the overall structure of this research, it was not merely a series of data collection and analysis opportunities. Instead, it was a process that contained an intricate interplay between deliberate guidance from myself and my fellow ADR team members and emergent environment results. I expand on this process in the following chapter.

4 GUIDED EMERGENCE

The interventional design in this research was conducted between January 2010 and August 2014, with two full ADR cycles occurring between May 2012 and August 2014. The overarching objective was to design an open digital platform by emulating unsanctioned development and increase the STAs pool of potential innovators. An overview of these cycles and the concluding product design principles[22] can be found in Table 9[23].

[22] In chapter 6, the product (and process) principles are elaborated.

[23] While paper 5 includes the design interventions, the outlet space requirement did not allow for the full empirical narrative and supporting evidence. To this end, this narrative can be found in Appendix G.

	ARTIFICIAL PLATFORM DEMONSTRATION	AUTHENTIC PLATFORM DEVELOPMENT	TARGET PLATFORM IMPLEMENTATION	ENSEMBLE PLATFORM MANIFESTATION
VERSION	Alpha	Beta	Release	Maintenance
RESEARCHER-CLIENT AGREEMENT	Focus: Demonstrate the feasibility of and tentative principles for more open access to real-time railway data Roles: Researchers led the investigation and designed the artifact with limited feedback from third-party developers and the STA Client commitment: Support the alpha artifact, option to materialize a live beta artifact	Focus: Develop a fully functional platform enabling third-party development on real-time railway data Roles: Researchers as project managers, STA was part of design team, third-party developers were users Client commitment: Make the platform openly available during the project + 1 year	Focus: Implement an operational platform, addressing the shortcomings of the beta artifact and enabling third-party development using real-time railway data Roles: STA as project managers and designers, researchers as designers and responsible for evaluations Client commitment: create an open official real-time data platform for the STA	Focus: Follow the platform evolution Roles: STA as project managers and designers, researchers as observers Client commitment: Allow access to relevant usage statistics and qualitative inquiries
SPECIFIC PROBLEM	Developers missing shortcuts to frequently implemented use cases	Developers missing shortcuts to frequently implemented use cases + access to all data	Developers missing shortcuts to frequently implemented use cases and flexible data processing options	Ensure new datasets follow platform principles and find internal uses for the platform within the STA
GENERALIZED PROBLEM	Lack of platform capabilities for materialized self-resourcing	Lack of platform capabilities for materialized self-resourcing and additional experimentation opportunities	Lack of platform capabilities for both materialized and emergent self-resourcing	Sustaining external and exploiting internal emulated platform capabilities

	ARTIFICIAL PLATFORM DEMONSTRATION	AUTHENTIC PLATFORM DEVELOPMENT	TARGET PLATFORM IMPLEMENTATION	ENSEMBLE PLATFORM MANIFESTATION
INSTANTIATED GOVERNANCE	Access openness o Open, non-discriminatory API access Coherent search o Packaging observable app behavior	Access openness o Open, non-discriminatory API access Coherent search o Packaging observable app behavior Flexible search o Channel raw information objects	o Resource openness o Open platform Coherent search o Disclosing observable app behavior implementation Flexible search o Improved and opened internal interfaces	Resource openness o Open platform Coherent search o Disclosing observable app behavior implementation Flexible search o Improved and opened internal interfaces
INSTANTIATED ARCHITECTURE	Modular operator: o Inverting Interfaces o API key dispenser o Use-case-bound API	Modular operator: o Inverting Interfaces o Use-case-bound API o Full data model API Integration protocols o Coordinate conversion code o User tutorial o API console o User registration o Documentation	Modular operator: o Substituting Interfaces o Query engine API Integration protocols o Example queries for common use cases o User tutorial o API Console o User registration o Documentation	Modular operator: o Mutating o Interfaces o Query engine API Integration protocols o Example queries for common use cases o User tutorial o API Console o User registration o Documentation

	ARTIFICIAL PLATFORM DEMONSTRATION	AUTHENTIC PLATFORM DEVELOPMENT	TARGET PLATFORM IMPLEMENTATION	ENSEMBLE PLATFORM MANIFESTATION
CONCURRENT EVALUATION CONTEXT	*Ex ante*, formative evaluation Workshop with third-party developers and representatives from the STA	*Ex post*, formative evaluation Open discussion forum Platform open to the public, time-constrained	*Ex post*, summative evaluation Platform open to the public	*Ex post*, summative evaluation Platform open to the public and the STA
EVALUATION RESULTS	Coherent searches necessary Flexible searches missing Access openness sufficient	Coherent searches necessary and considered satisfactory Flexible searches necessary but considered unsatisfactory Access openness sufficient and considered satisfactory	Coherent searches necessary and considered satisfactory Flexible searches necessary and considered satisfactory Resource openness sufficient and considered satisfactory	Governance/architecture configuration persevered Scraping ceased STA emulation practices continued Open platform approach resolved SLA concerns Emulation as a platform design strategy led to internal adoption
CONCLUDING PRODUCT PRINCIPLE	The Principle of Platform Access to Externable Data and Functionality	The Principle of Platform Capability with Non-Deterministic Use Support	The Principle of Platform Growth By Experiment Flexibility	The Principle of Platform Equilibrium through Internal Integration

Table 9 – Overview of guided emergence in this research

4.1 Artificial Platform Demonstration

My engagement with the STA began in January 2012 after receiving funding to work with actors in the public transport industry to open up their internal systems for outside innovators. Based on this financial support and previous experiences from DART, Trafiklab.se, SL, and TravelHack, I searched for organizations that could be interested in resolving self-resourcing issues. Here, the STA stood out as a prime candidate for testing the emerging framework on open platform emulation since they were subjected to the extensive scraping of real-time railway data and had little knowledge of how to resolve this situation. Thus, the product owner of Trafiklab.se and I contacted the STA to determine their potential interest in engaging in a collaborative venture involving emulation as a means to design an open platform. After some initial discussions, they expressed a clear interest in engaging in this venture.

At this point in the investigation, I had immersed myself in third-party development, self-resourcing, and platforms within the Swedish public transport industry since autumn 2009. As a result, I had begun developing hypotheses on what configurations of platform governance and architecture were suitable for emulating self-resourcing. These hypotheses sprung out of observed disconnects between the platform capabilities offered (if any) by public transport actors and how developers designed and consumed resources for their use. Following the interactions with DART (Rudmark & Ghazawneh, 2011; Rudmark & Lind, 2011), we presented our ideas.

Additionally, the capabilities offered by public transport agencies were rooted in different design paradigms. One such paradigm was publishing previously internal interfaces publicly. For instance, Västtrafik, one of the agencies in the DART group, followed this tradition and outright published the APIs used for their webpage to developers. However, these interfaces were criticized by developers for a variety of reasons. They were considered verbose (response sizes were difficult to process by phones on the move), contained a non-standard use of XML, were stop-oriented (rather than *en route*-oriented), and used geographic positioning coordinates (RT90 coordinates) that were challenging to cast to those used by

smartphones (WGS84). For their part, Trafikverket designed data-sharing platforms rooted in another paradigm, where real-time data about road works, accidents, and severe weather conditions were based on the European standard Datex II (distributed free of charge)[24]. Incumbent industry actors such as traffic data aggregators, forestry companies, and traffic navigators used this data stream, and the STA's yearly customer surveys noted satisfaction with the service. However, when I interviewed a group of smartphone app developers outside of the established transport industry, they were critical of the STA's Datex II program. As in the case of Västtrafik's APIs, they found it impossible to cast this verbose data stream (initially developed for exchanging data between traffic management centers across Europe) into mobile use without substantial intermediate data processing. Moreover, obtaining access to this data involved signing a written agreement with the STA, which they found unnecessarily complicated.

The criticism of verbose (and to these developers) outdated technologies in tandem with the far-reaching written agreement repeated itself in my study of SL and their scraping trajectory (Koutsikouri et al., 2018; Rudmark et al., 2012). In a largely failed attempt to offer internal APIs conditioned by signed contracts, developers expressed their discontent in interviews and the initial developer program by SL suffered from low adoption rates. However, when SL offered less verbose APIs that were suitable for a mobile context and available through a simple registration process, adoption skyrocketed among external developers. Notably, the Trafiklab architecture enabled this turnaround redesign. SL could not offer the sought-after capabilities through their existing architectures due to outdated technologies and numerous dependencies. Instead, they published their new APIs through Trafiklab, which were more precise, less verbose, and allowed *en route*-oriented interfaces to be offered.

Finally, studying developers using platforms in real-time during TravelHack (Rudmark, 2013) allowed me to gather more information on the discrepancies between the software tools offered by public transport agencies and what third-party developers expected. This

[24] http://www.datex2.eu/

in-depth observation led me to understand the importance of considering seemingly minor details such as registration procedures, development environment integration, and the exact meaning of casting API calls into live use cases.

However, despite our previous experience in dealing with similar situations and the emerging theoretical framework, it was not feasible to directly instantiate a platform at that point. Given the STA's existing third-party developer programs, the organization did not have sufficient knowledge or capacity to directly fulfill the emerging requirements. Moreover, the current STA third-party developer strategy did not include the type "app hackers" under scrutiny in this project, which made more far-reaching commitments difficult at that time. Thus, to resolve the situation while still making progress, we entered a rather limited researcher-client agreement. This agreement stipulated that myself and the product manager of Trafiklab would investigate their situation and demonstrate an artificial version of a potential platform with support from both the STA and—to the extent possible—third-party developers. After this artificial platform demonstration, the STA would have the option, but not an obligation, to continue the collaboration.

As a first step, we formed a deeper understanding of the idiosyncratic situation at hand. At the outset, we identified that it was essential to detail current strategies at the STA and third-party developer preferences to describe the change nexus more precisely. The STA interviews highlighted their existing practices and policies concerning third-party development. These were marked by existing industry structures, where external data re-users were well known by the STA and operating within the transportation sector. Also, the data formats in use were rich and complex (i.e., focused on enabling *flexible searches*), while third-party relationships were based on signed contracts to ensure respective parties' responsibilities (not *open platforms*). I also reached out to developers that were using scraped data, who were surprisingly willing to share their views. These interviews with developers highlighted a rather different set of platform governance expectations. Rather than contracts, they wanted to work directly with APIs and were, at most, willing to submit to online registration (*access openness*).

Moreover, rather than searching for solutions in complex data structures, they wanted APIs that allowed for immediate recurrent usage (*coherent searches*). This preference meant that attractive APIs should support common use cases such as station searches based on names, coordinates, or departures and arrivals from platforms. The developer interviews also uncovered two categories of developers. The first type both scraped data and built the end-user services, while the other type focused on the end-user services and used unsanctioned APIs to build on scraped data that other developers had collected and shared.

At this point, my emulation framework[25] consisted of a specific configuration of architecture and governance. As argued in this thesis, emulation entails surpassing desirable compatible platform capabilities by rearranging an organization's resources. More specifically, the problem at hand concerned a lack of available platform capabilities that materialized existing self-resourcing.

Based on the problem formulation, I hypothesized that such capabilities involved two aspects in this context. First, there was a need to establish coherent search opportunities (i.e., a set of highly reusable capabilities that could cut across knowledge boundaries). This finding was corroborated by video observations from TravelHack (Rudmark, 2013), where I was able to study how developers appropriated APIs in their development in real time. The analysis showed that developers were attracted to platforms that facilitated casting of common use cases into platform calls (e.g., finding the nearest train station).

Second, my framework included access openness to govern the platform's openness. By drawing on access openness, an organization subject to outlaw innovation could retain control of what was shared with third-party developers (i.e., coherent search capabilities). Moreover, during fall 2011, my co-authors and I conducted a case

[25] During the ADR interventions, I ingrained the platform with theoretical ideas that were available at the time (e.g., Boudreau, 2010; Tilson et al., 2010). In retrospect, more contemporary platform theories (e.g., Brunswicker & Schecter, 2019; Karhu et al., 2018) that were built on earlier ones provided additional explanatory power and were thus included in the framework.

study (Rudmark et al., 2012) that investigated the trajectory of a similar organization (SL) and how they had struggled with (and now seemingly succeeded in) transforming unsanctioned third-party development involving scraping into development on sanctioned resources. A key conclusion from this work was that more far-reaching contracts were misaligned with the preferences of third-party developers that use scraping. Consequently, developer adoption skyrocketed when SL switched to non-discriminatory access openness.

On the architectural side, the means of ensuring such capabilities included inverting a new module on top of the existing resources. This new module was then used to emulate the desired behavior conveyed through specialized interfaces (e.g., APIs). I hypothesized that this architecture was an efficient way to provide access to incumbent digital resources and cast them into the type of reusable capabilities preferred by developers. An inversion-based architecture had been used to establish Trafiklab and was vital to enabling the rapid developer uptake of SL's open interfaces. This architectural pattern would also allow tentative platform experiments and release versions to unfold without affecting existing systems.

Given the input gathered in the problem formulation phase and my initial framework, the artificial platform materialized in the following manner. We proposed a dedicated API that was open to any external third-party developer (access openness) and conveyed the set of use cases hinted at during the interviews (coherent searches). In terms of architecture, these use case-bound APIs (interfaces) would be implemented in Trafiklab (inversion) and use API technologies relying on Representational State Transfer (REST) and simple URL parameters. The architecture also required an API key dispenser (interface) that developers would call to obtain API access tokens[26].

The alpha version evaluation involved a workshop held on April 19, 2012, where the platform's principles were presented. We concluded that even though we were in a pure building system mode, there was a need to sufficiently resemble the authentic tensions in the work

[26] In the subsequent beta version platform, this interface was exchanged for registration procedures using an integration protocol.

system. We achieved this by inviting representatives from the STA and third-party developers to get all actors in the same room[27] since we considered these developers an indispensable part of the work system ensemble. Until that point, these third-party developers had been mostly unknown to the STA personnel and were jokingly named "underwear hackers" (since these programmers presumably developed at home in their underwear). Notably, we considered it essential for these developers and key STA personnel to become more familiar with each other. Moreover, we considered it crucial to share more detailed knowledge regarding the impact of these third-party applications and their developers' genuine interest in rail-based public transport. By bringing these actors together, we determined that the ADR project would be better positioned to transition the artifact from the pure building ensemble (alpha version) toward the existing work system (beta version).

All workshop participants (i.e., STA developers and third-party developers) shared positive views of the artificial platform demonstration. The developers welcomed the coherent search capabilities and the suggested access openness. However, an unanticipated governance issue also emerged during the workshop. The more seasoned developers expressed their dislike for only exposing these shortcuts. For them, access to as many data points as possible was necessary to enable future innovative service development. Thus, only publishing the coherent searches was viewed as constraining future innovative uses of the real-time railway data. However, the developers also made it clear that the exact format or retrieval mechanism for these new datapoints were not essential.

[27] Although there was both an overall interest from the third-party developers to continue the dialogue and a rather positive response to the current project, recruiting the developers for a full day workshop was not easy. Since many of these were contractors, their participation would infer a financial investment on their part. Moreover, since we assessed their participation to be critical, we decided to offer some financial remuneration to compensate for the loss of income.

4.2 Authentic Platform Development

Given these largely positive signals from the workshop participants, the product owner of Trafiklab and I began to work on a proposal for creating a more authentic beta version. Here, we strived to balance the need for near-deployment authenticity while simultaneously recognizing that the beta version platform had to be developed in a way that the STA was able to host it. Some two weeks after the workshop, we sent the refined solution blueprint to the Head of Passenger Information at the STA (also a workshop participant). The blueprint used a system called Orion (some outlaw innovators fueled their apps with data using a "backdoor" to this system) as the underlying resource. Simultaneously, Trafiklab.se contained the architectural modules necessary to emulate the desired behavior. Our suggestion also included a request to engage personnel within the STA to become part of the ADR team that I would lead. Just one day after receiving our offer, the Head of Passenger Information at the STA gave the go-ahead to start the design and deployment of a live beta version and provided access to the required STA personnel.

After forming the ADR team, we started to reformulate the problem to develop the beta version. While many of the assumptions identified in the alpha version held true, the need for developers to also be able to experiment beyond these common use cases had surfaced. However, the participating developers noted that this missing feature could be a less complicated capability since the core issue was having all data points obtainable from the API. This assumption of a less sophisticated flexible search mechanism was highlighted by a unanimous statement from the developers participating in the workshop, who noted that they would be content with any format other than HTML. We thus embarked on resolving this problem of missing platform capabilities regarding already materialized self-resourcing but also to provide opportunities for additional experimentation.

As a next step, we began the Building, Intervention, and Evaluation (BIE) stage of ADR by addressing the more specific platform design aspects. Given the problem formulation, we decided to include the following elements.

Regarding governance, we found support from the developers in both interviews and the workshop to implement an *access openness* (as per the SL case). Thus, we concluded that we could mimic such governance concerning platform access. However, the *coherent search capabilities* were a bit more complex to construct. The alpha version interviews concerned integration capabilities and keeping data transfer to a minimum, a finding corroborated in the video observations from TravelHack. Besides catering for more general ease of integration, we concluded that the API needed quality-assured "shortcuts" to datasets with high developer demand. Thus, we decided to reverse-engineer the current app behaviors and "pirate" API designs and then offer these as beta platform shortcuts.

Given the unanticipated developer response to the constraining effect of merely publishing coherent searches, we concluded that the platform required a mechanism to channel *all* data to allow for *flexible searches*. However, based on the developer feedback. We also hypothesized that such an arrangement could be cruder. To this end, we decided to publish information objects in their original form. This change led to a shift in the design framework to include *flexible search* capabilities.

Although the data were readily available within the STA, their systems architecture at the time could not afford to support it within the project's resource boundaries. Instead, we used the infrastructure of Trafiklab for platform architecture since it could host the emulated capabilities. Moreover, its modular structure allowed for the resource-efficient inclusion of the STA's systems. In practice, the module facing developers was a cloud-based service hosted by ApiGee, a company selling platforms that host and scale APIs. This module handled access control, data caching (to relieve the underlying system of redundant queries), and provided the interfaces geared toward third-party developers that were decoupled from the underlying systems. Furthermore, we created a specification on how to extract data corresponding to the coherent and flexible searches within the ADR team, including how the ApiGee interfaces should offer these as REST APIs (the actual transformation was carried out by ApiGee personnel). Notably, the TrainInfo interface marshaled coherent searches while TrainExport facilitated flexible search activities.

Before we started to develop the beta version platform, the ADR team constructed a detailed specification of the expected externally available functionalities. Here, we opted for public feedback for our concurrent evaluation by openly publishing the specifications[28] and receiving feedback on a public web forum[29]. We did this because only a handful of developers could be present at the alpha version workshop and our team needed to make a few design decisions to transform the alpha version into actionable interface specifications.

Since we received few manageable suggestions from the public feedback, the specification remained virtually untouched. However, one unfulfilled request played a more critical role in the next version and surfaced twice in this group. This request involved the possibility to only retrieve records that had changed since the previous request.

Next, we carefully designed the environment in which the beta version platform would be deployed. While striving for maximum authenticity, we decided to launch the platform publicly to allow any interested developer to use the APIs. This way, the ensemble would allow for maximum structural influx. However, due to it being a beta version, there were limits to authenticity. In our case, these limits materialized as announcing that the APIs were deployed as a test. Thus, guarantees regarding their future operations was limited to the project's end, plus one year (effectively spring 2014).

Based on the specifications, the platform was developed between August and early October and was launched on October 25, 2012. Once the APIs were launched, I initiated a round of interviews with the developers. Through these interviews, I aimed to understand how both new and more seasoned developers experienced the platform in terms of its utility. New developers had almost exclusively opted for the coherent search interface (TrainInfo) and found it satisfactory. However, the more experienced developers often sought more flexible search capabilities and expressed disappointment in what was offered through the beta version platform. First, the assumption

[28] Available at https://docs.google.com/document/d/1qf-Cj18NGeDDkMjIRBSOmdQVGMmGu_Zi2UmkXAbwJDQ/ (in Swedish).

[29] https://groups.google.com/forum/#!forum/jarnvags-api-trafiklab (in Swedish).

(brought forward in the alpha version workshop) that any format except HTML would suffice fell short. Here, the developers highlighted the inherent versatility of the existing Orion interface (that some outlaw innovators had used). This interface is essentially non-deterministic and a general query interface (similar to SQL) that was sufficiently easy to tailor to the developers' different needs. On the other hand, our beta version platform allowed all raw data to be downloaded. However, all subsequent processing had to be performed in the ecosystem periphery by the individual developer.

Second, these developers also expressed the need for additional flexible search benefits to change their applications' data sources. When asked what such incentives might be, the developers' signals conveyed the need for only retrieving records that changed since the last request (this also surfaced in the open forum).

Regarding access openness, all developers were in favor of how the interfaces were offered.

4.3 Target Platform Implementation

As a direct consequence of the ADR project, the STA decided to revise its third-party developer strategy in early 2013 to include the type of mostly unpaid app developers that had been users of the beta APIs and used self-resourcing in the past. The subsequent need to implement a release version of the platform thus emerged, which facilitated my continued active participation. In this phase, our researcher-client agreement stipulated that the STA would lead the target platform's implementation project and that my role was to provide design input (especially regarding third-party developer needs) and be responsible for evaluations.

As a next step, we reformulated the problem. In summary, third-party developers that were new to the railway domain had used the coherent search interface TrainInfo, found it pertinent, and echoed a pleasant experience. However, existing and more seasoned third-party developers that had already implemented services expressed their dislike for the flexible search capabilities. For this reason, most of them had continued to use unsanctioned data access. Second, these developers were unhappy with the capabilities of TrainExport

compared to what certain scraped resources could achieve. They also expressed the need for additional flexible search benefits to motivate the effort of changing their data source. Consequently, we hypothesized that emergent flexible searches also had to be emulated and not only offered (as per the beta version) in tandem with those that had materialized across apps.

The surprising reception of the beta version platform by experienced third-party developers instigated a substantial release version platform redesign. Based on the feedback, we decided to implement a query language similar to that of Orion to cater to flexible searches. However, Orion's query language was designed for internal use, and the ADR team assessed that it was unsuitable for external publishing in its existing form. To this end, the query language was redesigned for reduced redundancy, syntax strictness, and data model congruence.

Moreover, in the beta version design and onwards, signals from developers conveyed the need for a functionality that facilitates the retrieval of records that changed since their last request. However, this feature would require a substantial redesign of the underlying system. Thus, implementing this feature had not been considered financially justifiable until that point. To further investigate whether this feature was necessary, I conducted a data source experiment on apps using SL's real-time data. This experiment corroborated the findings from the STA's beta version API, which indicates that the primary reason for developers continuing to scrape was the lack of perceived benefits related to switching. Consequently, this additional signal provided sufficient evidence for the STA to implement this improved flexible search functionality despite the large investment.

Given these various signals, we decided to apply a new governance regime for the platform's openness (*resource openness*)—a far-reaching decision that stemmed from several reasons. First, since the STA now planned to offer its internal (albeit refactored) query language for external developers, there were fewer incentives to encapsulate it behind a software layer offering access to the resource. Second, given our insights from the data source experiment, developers at SL mentioned capabilities not available in the official APIs as a reason for continued self-resourcing. Consequently, any

deviations between the interfaces offered to third-party developers and internally developed public apps risked introducing new self-resourcing. Finally, the STA did not want to maintain more interfaces than necessary. By providing improved Orion interfaces through DataCache, the STA could easily upgrade its own applications while still serving the needs of external third-party developers.

However, this decision involved challenges to the platform's architecture. First, Orion's interface (the system that was then explicitly being transformed and exposed to developers) was a non-deterministic query language and inherently supported only flexible searches. Hence, we concluded that it was no longer possible to use the interface level for coherent searches (unless introducing new interfaces, which is a solution that the STA rejected for system maintenance reasons). Instead, we opted for a revised architectural configuration. Here, we used the integration and testing protocols (i.e., predefined example queries) to implement the identified coherent searches. We then sought to replace the core resource (i.e., the Orion system, which was renamed DataCache in the release version) with a new third-party developer platform. This way, the necessary emulation activities (simplifying both the query language and internal data models while also introducing delta functionality) were implemented directly into the Orion system using the *Substituting* modular operator.

Although the coherent search implementation was successful in the beta version platform, we saw a need to validate the new implementation (using example queries). To this end, a more controlled test with novice users was conducted. In this test, university students were given a set of tasks to complete that involved them reusing the coherent searches to accomplish tasks. The students provided generous feedback on improvement opportunities (e.g., more informative names for the data model elements and example responses in addition to queries). A core signal from this test was that 10 out of the 13 students were able to perform the tasks (e.g., getting the train departures from a specific station) with the help of the queries. Since these students' application development experience was lower (according to the background information they provided) compared to the target group, we concluded that the coherent search solution would suffice.

The target platform was launched in a staged process. First, DataCache went live on February 10, 2014, as an open test environment. As such, any interested party would be granted access to the platform. However, since the platform was labeled as a beta version, this meant that it was subject to changes. During this beta period, I initiated a round of interviews with interested developers who had registered. The official production platform launch occurred on March 18, 2014[30]. An important decision made before the launch was to not require internal applications to migrate by the launch date. Instead, they could be migrated at will, either when they needed new platform functionality or when other necessary adjustments were due[31]. During this time, the old system (Orion) remained operational but was not upgraded with new functionalities

After the official launch, I continued my interviewing efforts with developers. In these interviews, I was especially interested in two aspects: a) whether the experienced developers now experienced sufficient incentives to switch from self-resourcing to the release version DataCache platform; b) whether the coherent search capabilities had maintained their qualities during the architectural change. The interview signals were generally positive from both experienced developers and those new to the railway domain.

4.4 Ensemble Platform Manifestation

Although these tentative ensemble signals were indeed positive, they provided somewhat conjectural evidence. Once the release version platform had been deployed, my active part in shaping the platform's future trajectory ceased. However, I continued to monitor its continued evolution in several ways. I was particularly interested in how the platform was adopted by external and internal clients and its possible industry effects.

Thus, in September 2016, I performed a more technical follow-up study to investigate the actual data sources used by apps displaying data from the STA. I surveyed the apps using the same method

[30] See https://api.trafikinfo.trafikverket.se/

[31] The last internal application migrated in late fall 2017, followed by Orion's operations being discontinued.

utilized for the previous scraping follow-up for Trafiklab (i.e., intercepting the apps' API calls). If the data source for an app was unable to be determined, I contacted the app's developers to inquire about the data source by either email or phone. This investigation revealed that development toward unsanctioned interfaces was virtually extinct. At that time, 28 services for smartphones using real-time information were available in the application marketplaces for Apple iPhone, Google Android, and Windows Phone. Out of the 28 real-time services, 19 used the open API, 6 used interfaces connected to other STA third-party development segments (a system called UT/IN), and 3 were not functioning (where it appeared as though the app was no longer maintained). Moreover, usage statistics from the platform showed that not only existing developers had adopted the API. These statistics conveyed that external API calls had increased from approximately 20 million per month in 2016 to approximately 100 million per month in 2020.

Concerning the organizational reception of the STA, I found signals pointing toward emulation activities persevering on the open platform after the ADR project had ended. The platform's website hosted a changelog where all changes to the platforms had been recorded. When I interviewed STA personnel about what triggered these changes, emulation was an essential rationale. For instance, although the platform initially only hosted railway data, it was not long until it also hosted roadside data such as accidents, road works, and road weather, which also became available through the platform. These data were published by drawing on emulation as a design strategy to better fit third-party developers' practices. Also, STA personnel began to monitor discussion boards and similar outlets to find cues for additional improvements to the data models, which was triggered by the previous ADR project.

Another essential trigger for altering platform functionalities was the chosen open platform approach's consequences. When an internal information object was required for public digital services developed by the STA, this information object was published in the DataCache platform and made available to external third-party developers. One such example concerned road ferries, whose timetables and real-time updates were published in DataCache for this reason.

Although the effects above were desired—and thus somewhat possible to predict if the platform met the ensemble needs—two more surprising outcomes of the DataCache manifestation emerged in the follow-up studies. First, since the STA had chosen to keep the platform open in the user dimension, they used this approach to solve a dilemma connected to third-party developer service level agreements. This dilemma was related to how the provider of a "free" platform provides sufficient assurance of platform uptime so that external innovators risk investing in their services. The open DataCache platform was able to satisfy developers' demands for availability and quality because third-party developers could enjoy a "shadow SLA" of the STA's services. Since the services provided by the STA were mission-critical, they were secured under an SLA between the STA and their systems suppliers, which the third-party developers could indirectly enjoy. Therefore, the platform owner had not received third-party inquiries to sign SLAs with the STA.

Another surprising aspect that surfaced in the follow-up study was related to how the platform was used within the STA. Until 2015, DataCache had only been deployed as one instance within the STA. This instance was the open platform for both external third-party developers and public end-user services catered for by the STA. However, in 2015, the systems development team responsible for DataCache suggested that DataCache could also be used for internal projects. Since then, through internal word-of-mouth, by *mutating* the platform, copies of the platform had been increasingly used within the STA for integration development projects. According to the DataCache team members, the reason for this was the development speed that the platform provided due to its roots in emulation. A core activity involved in enabling this greater development speed was guiding potential data publishers in constructing data models that were understandable outside the publishing group, using well-known standards, and including workable examples. Following its increased popularity within the STA, the DataCache platform was chosen as the official integration platform to be used across the entire STA in mid-2020.

Ultimately, the STA's open platform approach affected the Swedish public transport industry. In 2017, the industry's members ratified a new strategic plan for Sweden's open public transport data. In this

blueprint, Samtrafiken was to host an open platform following the resource openness principle used by the STA for their DataCache platform. In the report (Arnestrand et al., 2017), the benefits of the STA's open platform approach were brought forward as an essential rationale for having a national public transport data platform that is open in the user dimension.

5 PAPER CONTRIBUTIONS

In this chapter, I summarize the papers included in this thesis. In addition to these paper summaries, I detail the roles of these papers in relation to this thesis as well as my role in collecting evidence, analyzing the data, and writing up the individual papers.

5.1 Paper 1

Koutsikouri, D., Lindgren, R., Henfridsson, O., and **Rudmark, D.** 2018. "Extending Digital Infrastructures: A Typology of Growth Tactics," *Journal of the Association for Information Systems* (19:10), pp. 1001–1019.

Summary: Digital infrastructures enable the delivery of information services in functional areas such as health, payment, and transportation by providing a socio-technical foundation for partnership governance, resource reuse, and system integration. However, to effectively serve emerging possibilities and changing purposes, a key question concerns how infrastructure can be extended to cater to future services in its functional area. This paper approaches such digital infrastructure growth as a challenge related to aligning new partners whose digital capabilities spur innovative services that attract more users. The paper advances an initial typology that covers four growth tactics (i.e., adding services, inventing processes, opening identifiers, and providing interfaces) with the potential to set the extension of infrastructures in motion. The paper subsequently explores the proposed typology by investigating how its particular tactics successfully extended the scope of a digital infrastructure for public transportation in Stockholm, Sweden.

Relation to thesis: This paper presents four tactics that can be used to extend an organization's digital infrastructure. One of these tactics is coined "opening interfaces" and becomes the focal tactic that is further elaborated on in this thesis. As such, this paper helps position the contribution from this thesis into the larger context of digital infrastructures.

Contribution as author: In this paper, I was invited by the other co-authors since they were developing a previously accepted conference proceedings paper (Koutsikouri, Lindgren, & Henfridsson, 2017). I collected the data and analyzed it with my co-authors to determine the tactic opening identifiers as well as complementary data for the tactic-adding services. Additionally, I reanalyzed already collected data (corresponding to that presented in Chapter 3.1.3 and 3.1.4 above) with my co-authors. augmenting the data collected by Koutsikouri et al. (2017). Additionally, I co-wrote the paper with the other authors.

5.2 Paper 2

Rudmark, D., and M. Lind. 2011. "Design Science Research Demonstrators for Punctuation – The Establishment of a Service Ecosystem," in *Service-Oriented Perspectives in Design Science Research*, H. Jain, A. Sinha and P. Vitharana (eds.), Berlin: Springer, pp. 153–165.

Summary: Design science research (DSR) is concerned with demonstrating design principles. To prove the utility of these principles, design ideas are materialized into artifacts and put into an environment sufficient to host the testing of these principles. When DSR is used in combination with action research, environmental constraints may prevent researchers from fully inscribing or testing design principles. In this paper, it is argued that scholars pursuing DSR have paid insufficient attention to the type of change required in the local practice. We draw upon theories on IS change (e.g., punctuated equilibrium) to illustrate when DSR demonstrators can be used to make substantial contributions to local practice and the scientific body of knowledge.

Relation to thesis: This paper is derived from interactions with the DART group, as described in Chapter 3.1.1[32]. In the context of this thesis, this paper contributed in two ways. First, it established that interventional design methods (e.g., ADR) are suitable to address the type of situation that later emerged at the STA. Second, it proposes the punctuated socio-technical IS change (PSIC) model of Lyytinen and Newman (2008) as a theoretical lens to understand how interventional design methods can be used to address organizational problems in situations where little guidance exists. A more comprehensive version of this reasoning can be found in Chapter 7.3.

Contribution as author: I planned the study, collected the data, performed the analysis, and wrote the paper with support and feedback from Lind.

5.3 Paper 3

Rudmark, D., E. Arnestrand, and M. Avital. 2012. "Crowdpushing: The Flipside of Crowdsourcing," in *Proceedings of the 20th European Conference on Information Systems (ECIS 2012)*.

Summary: Activities and initiatives pertaining to co-creation are traditionally viewed as a way for organizations to gain value through the involvement of certain actors in their environment. This paper highlights the implicit assumption in current theoretical conceptualizations that co-creation is exclusively initiated and driven by organizations. However, it appears that co-creation activities may also be driven by third-party actors outside of organizations. Based on interviews and secondary data from a public transport company in Stockholm, Sweden, we noted that third-party developers of services that gained large and diverse user bases were driving co-creation activities with their respective organizations. Based on our findings, we introduced the term "crowdpushing" to denote externally driven co-creation activities and frame four propositions to describe how co-creation activities are motivated and

[32] Although not part of this thesis, Rudmark and Ghazawneh (2011) also described the self-resourcing and countermeasures taken by one of DART's members (Västtrafik).

driven. Our findings contribute to a broader understanding of co-creation and have implications for its design and deployment.

Relation to thesis: This paper contributes to the present thesis in two distinct ways. First, the paper demonstrates that while self-resourcing on unsanctioned resources may serve heterogenous user bases, it also poses a threat to systems operations. Second, it demonstrates that developer adoption among outlaw innovators fueled by self-resourcing requires significant degrees of openness.

Contribution as author: I planned the study, collected the data with Arnestrand, performed the analysis, and wrote the paper together with Arnestrand and Avital.

5.4 Paper 4

Rudmark, D. 2013. "The Practices of Unpaid Third-Party Developers – Implications for API Design," in *Proceedings of the 19th Americas Conference on Information Systems (AMCIS 2013)*.

Summary: To draw on the innovation capabilities of third-party developers, many organizations are currently deploying open APIs. While third-party services may offer commercial opportunities for independent software firms, a large proportion of existing third-party software were undertaken without any financial compensation. Although unpaid developers offer a potential source of innovation in end-user services, the current literature has largely overlooked how these unpaid actors use and appropriate the technology provided by organizations. To this end, this research focuses on the specific practices of unpaid developers. The data used for analysis were collected through a programming contest—a hackathon—where unpaid developers gather to craft end-user services. Through an ethnographic lens, we present a number of recurrent activities and patterns of action employed by developers. Based on this analysis, we present implications for API designers seeking to attract unpaid developers.

Relation to thesis: The findings in this paper emanate from TravelHack, as previously described in Chapter 3.1.4. As such, this paper summarizes the practices identified during the innovation

contest. Notably, some of these practices (especially those relating to API use) were important to better understand self-resourcing developers when designing the alpha and beta versions of the open platform.

Contribution as author: I planned the study, collected the data, performed the analysis, and wrote the paper.

5.5 Paper 5

Rudmark, D. 2021. "Designing Open Platform Emulation, " *Under review at the 42nd International Conference on Information Systems (ICIS 2021).*

Summary: The successful engagement of third-party development has been instrumental in establishing contemporary platform leaders. However, complementary application development sometimes occurs without organizational consent. Notably, such unsolicited development can pose severe problems for organizations at both technical and organizational levels. In this paper, we advance platform emulation to leverage such unsanctioned development when designing open platforms. We base our contributions on a 10-year collaboration with a Swedish authority subjected to extensive unsanctioned development. Here, we applied the ADR method to develop a live open platform for third-party developers that is currently in use. From this work, we synthesize and extend current theories on open platforms and offer design principles encompassing a set of product and process principles throughout the open platform's developmental trajectory.

Relation to thesis: In this paper, I detail and provide empirical evidence supporting the design principles presented in Chapter 6. In relation to the research journey presented in Chapter 3, this paper is primarily concerned with the empirical information presented in Chapters 3.1.5, 3.2, 3.3, and 3.4.

Contribution as author: I planned the study, collected the data, performed the analysis, analyzed the design process, and wrote the paper

6 DESIGN PRINCIPLE DEVELOPMENT

This chapter presents a design principles that has been developed from the following research question:

> *How can organizations emulate self-resourcing activities of third-party developers to design open platforms?*

To address this research question, I have used ADR. The contributions from ADR are threefold (Sein et al., 2011, p. 42; Westin & Sein, 2015, p. 24), with two types of practice contributions and one generalized knowledge contribution. The first practice contribution encompasses *ensemble-specific contributions*. This contribution constitutes both the resulting artifact (ingrained by initial theoretical hypotheses and subsequent contextual structures) along with the modified organizational structures where the ensemble artifact resides. This first type of practice contribution corresponds to the DataCache platform (both the original open platform and subsequent internal instances) alongside STA's new strategic developer segment, third-party developers, and the STA's organization surrounding the platform. The second practice result concerns *end-user utility*. The ADR project described in this thesis includes utility for both third-party developers and internal developers at the STA (using the open platform). Finally, and at the center of this chapter, the *design principles*, following the formalization of learning stage of ADR (Sein et al., 2011), and conveying the necessary and sufficiently generalized design knowledge for use in other similar design contexts.

The first type of design knowledge contribution concern product-centric design knowledge. In this regard, I use situated platform design decisions and their environmental response to derive more generalized design principles addressing the class of problems under

scrutiny (Gregor et al., 2020; Sein et al., 2011). Second, I follow a design-theoretical tradition emphasizing that it may not be sufficient to merely describe product properties but that it is necessary to also provide process-oriented guidance to help designers meet their aims (Li, Sun, Chen, Fung, & Wang, 2015; Markus, Majchrzak, & Gasser, 2002; Walls, Widmeyer, & El Sawy, 1992, 2004). I base this decision on the experiences from this research, which highlight that deploying an open platform based on emulation is a rather confounding intervention for an organization. Here, the *process* in which such implementation is conducted is critical to meeting the desired aims. Consequently, in parallel to product principles, I am also offering generalized process principles to help designers design open platforms using emulation.

Thus, I present these concluding product and process principles in the following sections based on the schema suggested by Gregor et al. (2020)[33].

[33] The schema presented by Gregor et al. (2020) inherently supports design principles about an artifact's properties (the way I am applying the schema), user activities as well as user activities and artifacts. Since process aspects are not inherently supported, I present product and process principles in tandem in this chapter.

6.1 Alpha Version Principles

The focus of the alpha version is showcasing a blueprint for a designed ensemble environment. Detailed product and process principles are presented in Table 10.

	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 via a novel, abstract software layer with dedicated interfaces offering such emulated functionality	Execute an artificial demonstration of the alpha version platform blueprint by 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 reusable 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 to enable further development

Table 10 - Product and process principles for the alpha version platform

6.2 Beta Version Principles

The beta version involves developing a production-use platform for testing in an authentic development setting. Detailed product and process principles are presented in Table 11.

	PRODUCT ASPECT	PROCESS ASPECT
Principle title	**The Principle of Platform Capability with Non-Deterministic Use Support**	**The 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-like capabilities encapsulating product hackers' frequently implemented functionalities while 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 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 should retain the option to alter release version design rules, or even to withdraw from further development

Table 11 - Product and process principles for the beta version platform

6.3 Release Version Principles

The release version involves transforming and implementing desired capabilities into live production systems for both external third-party developers and internal application developers. The product and process principles for this phase are detailed in Table 12.

	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 capabilities are preserved in the target platform implementation while assuring 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 emulating the desired capabilities.	Because transforming internal digital resources to an open release version platform may infer altered design rules compared to both the beta version and substituted release versions

Table 12 - Product and process principles for the release version platform

6.4 Maintenance Version Principles

The maintenance version involves upholding desired capabilities for new information objects and creating options to harness emulation capabilities for internal purposes. The product and process principles for this phase are detailed in Table 12

	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 for 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 capabilities 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 qualities, 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 with support for desired capabilities 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

Table 13 - Product and process principles for the maintenance version platform

7 DISCUSSION

The research presented in this thesis has been undertaken over more than a decade. As such, it has encompassed a wide array of activities and analyses along the way. Each of the appended papers presents individual results but has also been critical in building the cumulative knowledge leading up to the contributions of this thesis. The first paper (Koutsikouri et al., 2018) discusses the evolution of digital infrastructures and platforms, and presents one tactic I have followed through, namely providing interfaces. While not addressed in the paper, this study informed me about the potency of an architectural configuration using inversion. By adding a new module exposing previously hidden information with high demand, SL and Samtrafiken was able to substantially expand their digital infrastructures into smartphone apps and Google maps. In paper two (Rudmark & Lind, 2011), I investigated the feasibility of using interventional design research method as a device to develop design knowledge for the type of platforms under scrutiny in this thesis. This paper also served as a starting point for the reasoning regarding the molding of early tentative demonstrators deeper into the organizational fabric. A more developed argument regarding this aspect can be found in Chapter 7.3. In the third paper (Rudmark et al., 2012), I investigate scraping and organizational consequences in more detail. Here, I conclude that it is first when organizations align their platform governance with the external thrust from third-party developers that equilibrium is possible. As such, this paper lays the foundation for the emulation approach used in this thesis. Moreover, the study underpinning the paper provided clear pointers that non-discriminatory access openness would stall self-resourcing incentives. Paper four (Rudmark, 2013) provided an opportunity for me to connect so far unconnected dots regarding how self-resourcing developers use APIs. This study thus enriched my design understanding and provided evidence of the importance of

emulating coherent searches. Finally, in paper five (Rudmark, 2021), I started by testing the first version of the emulation framework (consisting of governance pertaining to coherent search capabilities plus access openness and an architecture relying on inversion), building on the cumulative insights from the previous papers. Following a continuous shaping over four platform versions, the final design principles stemming from this process have been presented in Chapter 6.

While a core contribution from this thesis indeed are the design principles presented in in the previous chapter, such principles are not the only type of possible contributions from ADR. As argued in Sein et al. (2011, p. 44), ADR ventures may come with additional theoretical implications. In this chapter, I discuss three such implications. First, I discuss the implications of bringing emulation logic into the digital platform realm. Second, I offer suggestions regarding outlaw innovation pertaining to the influencing response. Finally, I reflect on guided emergence in ADR, before ending with the limitations of this thesis.

7.1 Platform Emulation

7.1.1 Architectural and Governance Openness

In chapters 6.1 through 6.4, I synthesized how different configurations of architecture and governance can be used to design open platforms using emulation throughout a platform's developmental trajectory. Based on these design insights, I will also offer extensions to the existing literature on open platforms.

Regarding governance pertaining to *solution search mechanisms* on open platforms, Brunswicker and Schecter (2019) offer two powerful constructs that shape platform ecosystems' trajectories, coherent and flexible searches. However, while Brunswicker and Schecter (2019) show how such searches occur in the platform ecosystem periphery, this research demonstrates that such searches may also play out in the platform core. By incorporating such attractive capabilities into the platform core, coherent searches can be re-used across apps, and new innovative derivatives may emerge through flexible searches without the requirement of having to keep the periphery open.

Architecturally, these search mechanisms can play out in one of two ways:

- These capabilities can be materialized by coherent and flexible search mechanisms being inverted into the platform and executed at the interface level.
- These capabilities can be achieved by substituting the resource subject to self-resourcing while implementing flexible searches at the interface level and coherent searches at the integration protocol level.

Moreover, this research extends the openness theory of Karhu et al. (2018). They conceptualize the platform owner's important governance decisions regarding whether to provide platform openness through access or resource openness. However, while Karhu et al. (2018) show how resource openness can be applied on the provider dimension (by forfeiting the IPR of the platform source code) in a platform context, this research demonstrates that such resource openness also applies to the platform's user dimension. This type of resource openness in the user dimension is of particular interest when a platform owner seeks to share their platform with the general public but the platform resources are not transferable via source code. Such situations may emerge when the desirable resources are bound to the platform owner's physical and digital infrastructure, such as the real-time railway and roadside data studied in this research. Architecturally, such resource openness is obtained by making internal resources used for public services visible to outsiders and using the very same resources when the platform owner develops public services.

To the best of my knowledge, this research is the first to introduce emulation logics into the digital platform realm. As such, I have also identified two distinct ways of achieving emulation, which I denote *high-level* and *low-level* emulation[34].

[34] These concepts are borrowed from the gaming emulator scene. Here, high-level emulation refers to when an emulator creates runtime compatibility at the system kernel level. Low-level emulation refers to

7.1.2 High-level emulation

In the context of emulation on digital platforms, high-level emulation refers to emulation that occurs by decoupling the underlying resource and visible third-party developer design rules (as in the alpha and beta versions of the DataCache platform). This loosens dependencies on the existing resource used (e.g., by outlaw innovators) and enables the platform owner to implement the emulated capabilities through interfaces without affecting the underlying resource. This type of emulation architecture is achieved by inverting the system with a new software layer. Through this new layer, the emulator may encapsulate and abstract specific aspects of the necessary transition of its incumbent resources into specialized interfaces that imitate or surpass the behavior of those subjected to self-resourcing (see Figure 1)

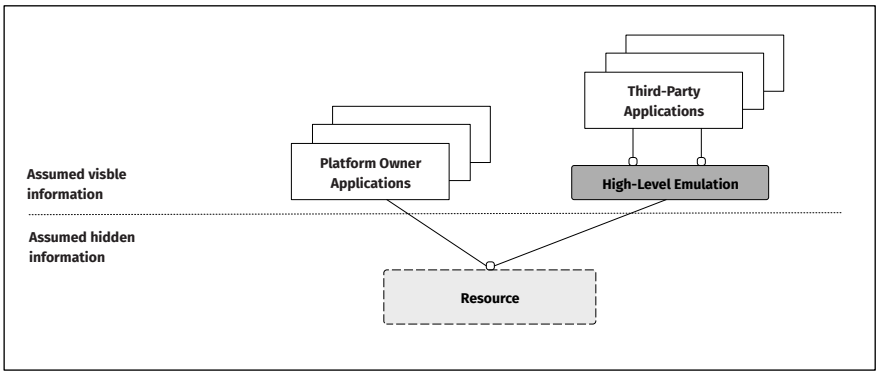


Figure 1 – High-level Emulation

I speculate that this type of emulation may be of interest when organizations wish to draw on emulation as a design strategy to design platforms for third-party developers but do not seek to deploy open platforms. Such situations include where only portions of the underlying resource are suitable for third-party innovation given security considerations or when the platform owner aims to avoid creating the underlying resource interface's dependencies. Other reasons for such limited emulation can include protecting existing

when the emulator creates a more complex and comprehensive abstraction layer for the emulated hardware.

revenue streams or issues related to underlying IP restrictions (that may not allow third-party innovation). However, given the empirical insights from this thesis, high-level emulation may also entail continued self-resourcing activities.

7.1.3 Low-level emulation

In platform emulation, I refer to low-level emulation when an emulator replaces existing modules with such that exhibit the capabilities desired by external innovators. Through low-level emulation, the emulator implements necessary changes within an organization's existing resource collection and effectively emulates directly into its internal platforms (see Figure 2). Consequently, this type of emulation suggests that the platform owner considers the emulated behavior as beneficial to the organization's own applications and triggers the modification of its applications to the new emulated platform design rules.

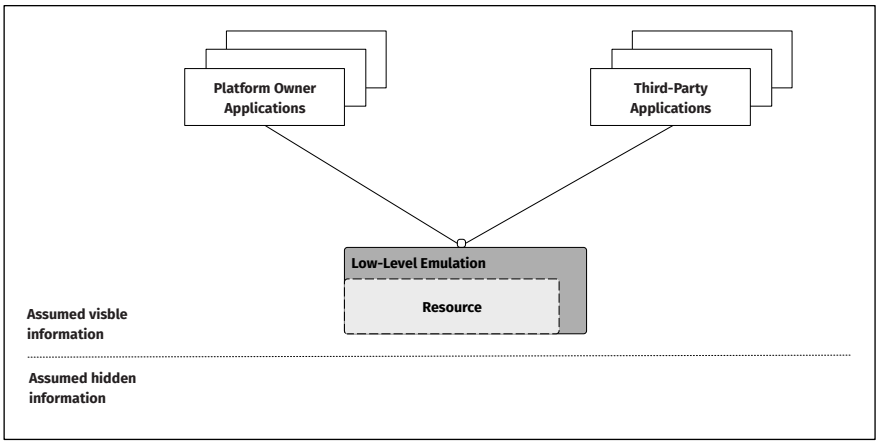


Figure 2 – Low-level Emulation

As exhibited in the present thesis, this approach to emulation is of interest when the platformer seeks to establish a platform open in the user dimension, and where such improved capabilities is of interest to the platform owner. Although this thesis has focused on third-party developer resources provided through emulation, I speculate that low-level emulation also may be used for internal platforms where outlaw innovation is used to find cues for improving existing systems and services.

7.1.4 Self-resourcing integration

Another benefit of using emulation as an approach to platform design (in cases of external self-resourcing) concerns the option to integrate potentially valuable capabilities into the platform owner's organization for internal purposes. As shown in this thesis, the platform capabilities linked to coherent, and flexible searches had implications beyond third-party application development. Indeed, within the STA, these capabilities were proven to significantly increase internal development velocity to the extent that the STA appointed the DataCache platform as the official integration platform to be used across the organization. However, this is not the same instance of DataCache used as an open platform among third-party developers. Instead, the STA has mutated the external platform to replicate the functionality for information models different to those of the open platform. Hence, self-resourcing integration refers to when a platform owner copies the open platform and uses it for purposes other than an external platform. In this way, the platform host can integrate the emulated capabilities, rooted in self-resourcing, for data and information that are not shared with the general public (see Figure 3).

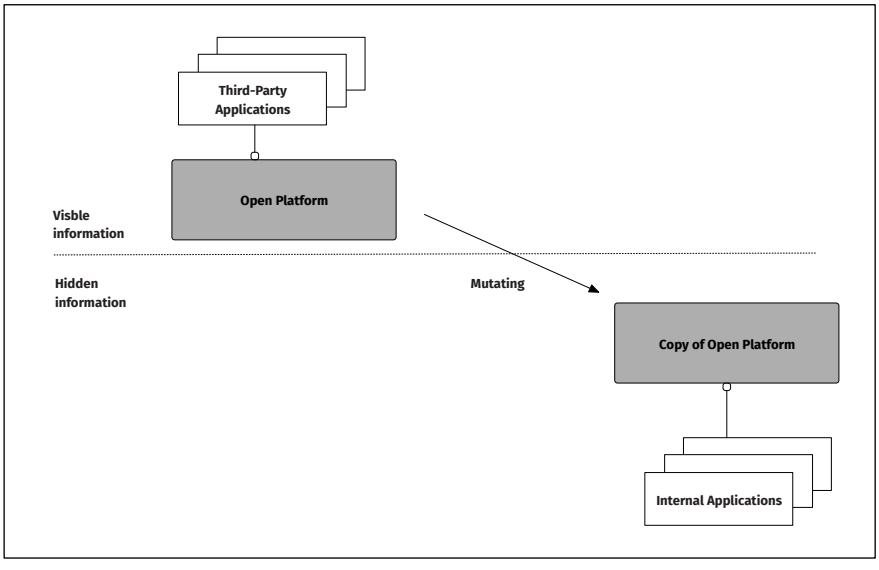


Figure 3 - Mutating for Internal Integration of Emulation

7.2 Outlaw Innovation

The next theoretical implication of this thesis more broadly concerns innovation beyond the platform domain. More specifically, this contribution adds to existing theories on outlaw innovation (Braun & Herstatt, 2008; Flowers, 2008; Mollick, 2005; Postigo, 2003; Schulz & Wagner, 2008; Schäfer, 2011), where an organization is subjected to product hacking. In these situations, organizations may respond in several ways, which include attacking the innovator, monitoring outlaw activities, or adapting outlaw innovations into the hacked product. Previous in-depth studies have investigated the absorption response when organizations seek to engage external outlaw innovators (Eaton et al., 2015; Schäfer, 2011). Such responses are typically a blend of increased yet selective openness paired with monitoring and attacking actors who do not comply with administrative legislation related to third-party developer programs.

In this thesis, I have explored the *influencing* response to product hacking. Notably, I argue that the findings from this research can be used to provide influencing responses with more nuanced content. When an organization seeks to convince product hackers to choose sanctioned resources, their outlaw innovation strategy must entail such innovators having access to publicly available data and functionalities. First, an organization should investigate and offer common functionalities with high developer demand by analyzing or reverse-engineering available applications to find such patterns and regularities. Second, while such common functionalities will likely satisfy most external developers, it is imperative to offer opportunities to experiment beyond these commonly implemented functions. This means that organizations should provide the same (although improved) data and functionalities that are publicly available in some form (e.g., in proprietary apps and web pages). Third, once such coherent searches and improved flexible searches have been identified, a concluding activity could involve having proprietary apps and web pages use this new and improved platform. Moreover, this use should be governed by the same restrictions and capabilities applied to external third-party developers to effectively establish an open platform.

In cases where such complete openness is not possible, another identified method of influencing some (or even most) product hacking is to offer more limited access to requested capabilities. In such a case, potential solutions would be similar to the one described as high-level emulation in Chapter 7.1.2. Here, the most desirable and reused capabilities can be implemented for third-party developers, possibly alongside some opportunities for experimentation. Moreover, rather than offering an entirely open platform, the organization subject to product hacking can instantiate a specific third-party developer platform that is not in use by proprietary applications.

7.3 Guided Emergence

In this section, I offer my methodological reflections on using ADR to design platforms. More specifically, I address the principle guiding reflection and learning in ADR—*guided emergence*.

Taking ADR's theoretical perspective of the ensemble artifact seriously involves ADR teams delivering significant contributions in all three dimensions. In this case, there is a need to design ensemble artifacts that are deeply embedded into its structures. I argue that an essential methodological key to unlocking this level of integration is paying closer attention to the process how the artifact evolves throughout its life cycle. Within ADR, this evolution follows the principle of guided emergence.

The basis for guided emergence is the BIE phase's ensemble signals. Thus, the BIE phases are contingent on embedding the design into an organizational context where evaluation is characterized by authenticity and concurrency (Sein et al., 2011). For ADR teams, authenticity is challenging since the research often questions organizational assumptions and structures, while the BIE may require confounding interventions to materialize into an authentic ensemble. Hence, in one of the papers of this dissertation (Rudmark & Lind, 2011), the PSIC (Lyytinen & Newman, 2008) is brought forward to conceptualize this dilemma.

In an information system change situation, PSIC posits that it is necessary to distinguish between a building and a work system

analytically. In the PSIC model, a *work system* constitutes the *de facto* IS structures enabling ongoing IS operations. It is characterized by "low malleability due to path dependencies, habitualization, cognitive inertia, and high complexity" (Lyytinen & Newman, 2008, p. 592). Hence, directly altering work systems is typically not feasible unless minor incremental changes suffice. For this reason, organizations instead establish *building systems* when other than trivial changes in work systems are necessary. In contrast to work systems, building systems are time delimited and have relative autonomy toward the work system when addressing a specific problem. Building systems can thus exhibit rather different properties to work systems, such as increased agility, specialized resource configurations, and the use of more flexible systems architectures.

An example of such a building system could be an enterprise resource planning (ERP) system implementation project that tailors the system to existing processes and simultaneously revamps existing processes to better fit the ERP system's logic. Over time, the building system needs to transition into the work system to achieve the anticipated change. In the case of ERP systems, such a transition may be performed by rolling out selected functions across the enterprise or incrementally deploying the full system to additional departments in an organization, or perhaps combining the two. Since knowledge development through ADR is an organizational intervention at its core (as per an ERP project), I argue that it is of particular importance for researchers to understand the roles that ADR phases play—including the crucial transitions between them—in the journey from building to work system.

Despite this importance, the existing literature conveys little guidance on how the ADR phases and their transitions should be managed. Sein et al. (2011) believe that the early alpha versions of IT-dominant artifacts should be lightweight, evaluated in a limited organizational setting, and subject to formative evaluations. Since beta versions are more mature, they can be tested in a broader environment and are typically evaluated in a more summative fashion. However, there is no explicit mention of release versions in the seminal article by Sein et al. (2011). In addition to this original article, Barrett and Holeman (2017) suggest that a set of activities

rooted in theories of sociomateriality can facilitate guided emergence (i.e., the implementation and use of prototypes, practice breakdowns, investigating breakdowns, accommodating material back talk, reconfiguring artifacts and routines, and the use of new practices).

Given this scarcity, I have studied how existing ADR research projects have dealt with guided emergence in practice. More specifically, I have focused on research published in the most influential IS journals[35] since these are more likely to contain substantial contributions to the IS field (Webster and Watson 2002) and can be expected to adhere to a high level of scientific rigor. A full list of these articles can be found in Table 14. Notably, in this collection, only the paper by Gregor, Imran, and Turner (2014) explicitly mentions how guided emergence impacted research.

7.3.1 Alpha Version

In ADR, the research team commences by using problem formulation as an entry point (Sein & Rossi, 2019). This order is essential since ADR departs from the problems in a client system (rather than using a clinical setting as an expository instantiation (Iivari, 2015)). Thus, the focus of the researchers will initially be placed on the work system. Hence, ADR research commences by analyzing the work system and establishing that it favors the introduction of a new ensemble artifact (Asatiani, Hämäläinen, Penttinen, & Rossi, 2020; Ebel, Bretschneider, & Leimeister, 2016; Giesbrecht, Schwabe, & Schenk, 2017; Giessmann & Legner, 2016; Gregor et al., 2014; Hustad & Olsen, 2014; Mettler, 2017). In some cases, signals from similar environments are collected to further contextualize the problem (Ebel et al., 2016; Giessmann & Legner, 2016; Gregor et al., 2014).

Next, during the alpha version BIE cycle, an ADR team formulates theoretically ingrained hypotheses concerning the artifact properties necessary to trigger desired effects in the observed work system (Mandviwalla, 2015). These first hypotheses materialize as the *ensemble artifact's alpha version* (Sein et al., 2011). However, since the artifact is in a genuinely formative stage where both utility potential

[35] https://aisnet.org/general/custom.asp?page=SeniorScholarBasket

and organizational legitimacy still are uncertain, the ADR project is set up as a building system[36].

Examples of such activities may include the incorporation of representative end users (Giesbrecht et al., 2017; Giessmann & Legner, 2016; Gregor et al., 2014; Mettler, 2017), the entire ensemble setting (Asatiani et al., 2020; Hustad & Olsen, 2014), or those who are likely to create legitimacy for a future artifact (Gregor et al., 2014).

7.3.2 Beta Version

In the following reflection and learning stage, the researchers enter a more analytical mode of thinking to make sense of the alpha version intervention. Here, the focus is to assess whether the alpha version artifact substantially affected the situation. The ADR team must also establish what additional structures from the ecological milieu need to be inscribed into the artifact. To grasp the alpha version's effect, ADR teams can use a variety of signals. The most common signals include user perceptions, which are provided in workshop-like settings (Asatiani et al., 2020; Giesbrecht et al., 2017; Giessmann & Legner, 2016; Gregor et al., 2014) or in written form (Asatiani et al., 2020; Ebel et al., 2016; Gregor et al., 2014; Hustad & Olsen, 2014). To make sense of these signals and articulate a new set of revised principles for the design of the next version, researchers use workshops (Asatiani et al., 2020; Giessmann & Legner, 2016; Mettler, 2017) as well as more formal quantitative (Ebel et al., 2016) and qualitative (Giesbrecht et al., 2017) research methods. However, given that the alpha version is deployed into a pure building system, Ebel et al. (2016) also noted that researchers must consider the extent to which the cues from an artificial ensemble potentially differ from those in a work system.

If there is sufficient consensus within the ADR team that the alpha version intervention results were successful, the next step is to materialize a *beta version*. Auspicious properties from the alpha version, alongside resolutions to identified structural misalignments,

[36] In the analyzed literature, Hustad and Olsen (2014) surfaced as an exception to this. Their ensemble artifact concerned university courses, an environment in which the authors had sufficient control over enabling them to start in the work system.

are inscribed into the beta version by analyzing the signals from the alpha version ensemble. Besides investigating a more refined artifact, the BIE environment at this stage starts to transition from a pure building system toward incorporating more facets from the work system. In the reviewed articles, beta version environments involved more "genuine" users and use situations (Asatiani et al., 2020; Ebel et al., 2016; Giesbrecht et al., 2017; Giessmann & Legner, 2016; Gregor et al., 2014; Mettler, 2017). However, all studied beta version target environments come with authenticity constraints such as indicating formative artifact status (Asatiani et al., 2020; Mettler, 2017), expert (as a proxy for actual market/user) assessment of utility (Ebel et al., 2016; Giessmann & Legner, 2016), impersonation of clients (rather than interaction with real ones) (Giesbrecht et al., 2017), and/or only deploying the solution to selected parts of the work system (Gregor et al., 2014; Mettler, 2017).

7.3.3 Release version

Given the artifact's continued molding into the work system, the evaluation becomes less controlled, while the reflection and learning phases gain potential access to a broader spectrum of ensemble signals. However, most of the reviewed research relied on user perceptions to assess the beta version's utility (Asatiani et al., 2020; Ebel et al., 2016; Giesbrecht et al., 2017; Giessmann & Legner, 2016; Mettler, 2017). In addition to experiential cues such as user perceptions, researchers may examine the actual results of artifact use (Ebel et al., 2016). Signals may also be collected from actors that can legitimize the release version's deployment in the client organization (Asatiani et al., 2020; Giessmann & Legner, 2016; Gregor et al., 2014).

In the studied research, six out of seven artifacts were refined through an explicit alpha and beta stage (Asatiani et al., 2020; Ebel et al., 2016; Giesbrecht et al., 2017; Giessmann & Legner, 2016; Gregor et al., 2014; Mettler, 2017). Out of these six, three eventually molded their ensemble artifact into the work system (Asatiani et al., 2020; Giessmann & Legner, 2016; Gregor et al., 2014). Since Sein et al. (2011) only implicitly deals with a final live version of the artifact, I

considered this stage's ensemble as the *release version*[37]. When making the transition into the release version, two articles included the client organization's management to wield resources and authority to deploy a live version of the artifacts (Asatiani et al., 2020; Giessmann & Legner, 2016). Another essential activity in this stage is the reciprocal shaping of design knowledge into production-ready artifacts, which can include aligning the culture code with the organization's visual identity (Asatiani et al., 2020) and modifying app marketplaces to fit new business models (Giessmann & Legner, 2016)[38]. Notably, three of the reviewed projects that were ready for real-world deployment had not yet been implemented in their respective client systems (Ebel et al., 2016; Giesbrecht et al., 2017; Mettler, 2017).

7.3.4 Maintenance Version

If ADR projects manage to refine artifacts from initial hypotheses to deployed full-fledged solutions, understanding the continued maintenance of released artifacts is of great interest for researchers using an ensemble view of technology (Mullarkey & Hevner, 2019). Since this artifact version is not explicitly defined within the ADR literature, I am suggesting the term *maintenance version* to refer to ADR artifacts that are put into work system use. In the analyzed literature, four studies managed to deploy artifacts into their respective work systems, while three conducted some form of follow-up study (Asatiani et al., 2020; Giessmann & Legner, 2016; Gregor et al., 2014).

To understand the ensemble trajectory after researchers exited the cooperation, Gregor et al. (2014) and Asatiani et al. (2020) captured the perspectives of users and practitioners through interviews. Additionally, Gregor et al. (2014) complemented such perspectives by collecting more structural evidence. This material included observing the continued refinement of interventional artifacts (not guided by researchers), performing an independent external

[37] This terminology is consistent with later publications addressing ADR projects leading to actual uses (Asatiani et al., 2020; Sein & Rossi, 2019).

[38] Since the other two studies reporting on work system implementation (Gregor et al. (2014); Hustad and Olsen (2014) did not follow the beta release sequence, this type of release version modification was not evident.

maturity assessment regarding e-Government, and even altering national budget priorities that could be traced to the ADR project.

7.3.5 Guided Emergence Revisited

ADR seeks to conduct interventions to design innovative artifacts that resolve organizational problems. While working directly with real-world problems has much potential, it also poses a dilemma for ADR researchers: since innovation often requires profound organizational change, the change must start in a relatively artificial organizational environment (i.e., a building system). However, as ADR draws on technology as structure, valid ADR research must also reflect essential structures from the organization's daily operations (i.e., the work system). Therefore, under the ADR team's supervision, this type of research seeks to manage a gradual movement from the building system toward the work system. This movement is at the core of the guided emergence principle, which is a core tenet of ADR. Unfortunately, as shown in the literature review earlier in this chapter, this principle's more specific content is largely overlooked by existing research.

Based on both existing research and the ADR conducted in this thesis, I argue that ADR researchers must acknowledge two principal parts: the role of *ADR phases* and *signal variance*.

7.3.6 ADR Phases

In this section, I argue that ADR researchers need to be more attentive to the roles of individual phases and the transitions between them. While the phases themselves are partially explained by Sein et al. (2011), the critical role that transitions play in an artifact's gradual movement into the work system (Lyytinen & Newman, 2008) has yet to be articulated. In what follows, I offer a set of recommendations to help ADR teams manage these transitions.

ADR is a research method that aims to resolve organizational problems through artifact design. However, resolving prevalent problems at sufficient depth typically requires that significant resources are made available, trust among the involved parties is established, and credible evidence supporting that a proposed solution will indeed resolve the problematic situation at hand exists. In early formative phases such as alpha version materialization, these

necessary preconditions are seldom present. Thus, researchers must design arenas that allow for theoretical ideas to be artificially demonstrated and problem owners must make proportional resource commitments.

Following the chronology of ADR, researchers start by framing a problem in the work system for epistemological reasons (Iivari, 2015). Once the most prominent structural misalignments have been identified and appropriate remedying theories have been selected, the BIE form (Sein et al., 2011, p. 43) can be appointed. As such, the target environment for evaluation must be chosen. I argue that ADR teams make a critical transition from the work system to the building system. As such, ADR researchers should design the building system evaluation context to resemble the essential traits of the "real" ensemble (Kock, 2003; Lee, 2007). Here, the ADR team must mindfully design the target building environment in a way that allows for the problem's cardinal structural contours to determine the fitness of the artifact. In the reviewed research, ADR teams typically took some (implicit) measures to accommodate this. For instance, Giessmann and Legner (2016) illustrated the business model through a canvas for current problems to surface, which allowed for a detailed discussion regarding improvements. Furthermore, Mettler (2017) exposed actual users to social network mockups in the alpha version to detect essential tensions. In the present research, we ensured that third-party developers were present at the workshop evaluating the alpha version. This way, the environmental structures became more evident, which represents a crucial objective in the knowledge development process.

While an alpha version is tentative by nature, the ADR team's goal (i.e., to achieve the trifecta of ADR contributions) should be to deploy the artifact into the work system. An essential aspect of this transition is organizational actors' involvement, which legitimizes and sanctions this process. Hence, during alpha version development, ADR teams should also consider involving actors that can authorize release version implementations into the ensemble. For example, Gregor et al. (2014) chose early intervention in target environments containing influential officials, while Asatiani et al. (2020) decided to run a series of workshops with the leadership team during the early formative phases. In the present study, we chose to

include the Head of Passenger Information into our target environment to increase knowledge and commitment to our knowledge development venture.

If the alpha version demonstration is successful, an ADR project enters the beta stage. I argue that the core of the beta version phase involves materializing premises that facilitate authentic artifact experimentation. To the greatest extent possible, researchers should embrace and develop their artifacts based on the possibilities and restrictions that authentic situations present.

Given the focus on increased authenticity, the beta version target environment is fundamentally different since the alpha version environment is contingent on having an ADR team *design* the most important structural element in the alpha target environment. However, the beta version target environment is mostly identical to the work system. For instance, Ebel et al. (2016) offered a business model development tool to expected users within the ERP system manufacturer SAP SE. Moreover, Giesbrecht et al. (2017) offered their service encounter thinklets to actual advisors for use in citizen interactions. This way, the beta version environment offers a much richer opportunity for ensemble artifact theorization. However, being a beta version, the evaluation environment comes with deliberate limitations to ensemble authenticity. Here, I argue that ADR researchers should choose beta version constraints that allow for the broadest spectrum of organizational structures to appear. To this end, the business models produced in the beta phase of Ebel et al. (2016) were subject to expert implementation (rather than being put to use within the company). Additionally, the citizen interactions that occurred in the beta phase of Giesbrecht et al. (2017) involved citizens impersonating real issues. In this research, we wanted to use the beta version to examine whether developers appropriated our platform's resources. Given this objective, it was instrumental to have these developers invest a similar amount of time, energy, and commitment into their work as they would if the platform were to be sustained over time. To this end, we publicly launched the new APIs into the Trafiklab.se platform with a time constraint that we assessed as sufficiently distant for developers to consider constraining.

If the beta version development results are sufficiently positive, an ADR team approaches the release version phase. I argue that the core of the release version phase is the implementation of design knowledge into an artifact that can be used to deliver end-user utility in a work system.

While this implementation step is not explicitly mentioned in Sein et al. (2011), it has been vital to the theorization in two of the analyzed research papers (Asatiani et al., 2020; Giessmann & Legner, 2016) as well as this dissertation. This step is critical since ADR teams must handle the realities that shape operational systems. For example, Asatiani et al. (2020) had to refine their handbook with graphical designers based on the organization's graphical profile to mold its way into the work system. Similarly, the business models created by Giessmann and Legner (2016) required the software vendor's application marketplace to be redesigned. In the present research, moving from the beta to the release version implied a radically different architectural implementation. Rather than having one interface for coherent searches and another for flexible searches (as per the beta version), the STA only wanted to support one interface in their production system. This change was a requirement from STA to enable continued platform use for their purposes. However, this requirement meant that catering for coherent searches needed to mutate. After lengthy discussions, we arrived at publishing the common use cases as example code instead of the fixed endpoints (as per the beta version). I argue that in this beta-to-release transition, it was important for the ADR team to ensure the knowledge gained during the alpha and beta versions was not "thrown out with the bathwater," but instead fortified in a possibly mutated form in the release version. From a theorization standpoint, the ADR team now has access to a broader spectrum of ensemble signals since the beta version was deployed to a hybrid work system. These new signals allow for new types of analysis regarding artifact utility and fit that could not be addressed in earlier phases.

Even though a release version has been deployed to the work system, this does not imply that a functioning ensemble has been established. The materialization of a structural arrangement requires that the artifact's material properties be put into—and possibly reshaped by—subsequent use. While researchers may have limited influence

over this continued post-release trajectory, it can have important implications for the ensemble theorization.

As previously mentioned, this fourth and concluding step in ADR is not explicitly defined within the existing ADR literature. However, while this phase has not been treated explicitly, some of the aforementioned empirical studies have conducted follow-up studies to assess its impact, use patterns, and other ensemble characteristics. In the present research, this fourth phase has been instrumental for the resulting theorization. First, by following the actual uptake by third-party developers, I concluded that self-resourcing as a practice appeared to have come to an end. Second, the emulation established by the ADR projects had persevered long after the ADR project had finished. Finally, and perhaps most importantly, the emulated platform had begun a different journey within the STA. Since the capabilities sought by external third-party developers were also of interest to internal developers within the STA, the platform started to gain traction internally. Here, different projects consequently mutated the open platform internally for other purposes to the point that the STA appointed DataCache as the default integration engine of the STA. This surprising turn allowed me to suggest that emulation can be used also to harness external requirements for internal purposes.

7.3.7 Signal Variance

The second process contribution concerns signals that are captured when conducting ADR. These signals are critical since the reflection and learning as prescribed by ADR are essentially fueled by the signals that the ADR team capture during the preceding BIE cycle (or possibly during the reflection and learning). In this research, I argue that signal variance was crucial to guiding the artifact from the problematic work system into a pure building system and then gradually back into the work system again. Moreover, I argue that signal types could be demarcated across the emic-etic dimension (Barley, 1986; Brooks & Alam, 2015; Morris, Leung, Ames, & Lickel, 1999).

Signals subscribing to the *emic* perspective emphasize inside perspectives and lived experiences from the ensemble under scrutiny. Such signals may be captured in workshops, interviews,

user diaries, or written comments. Conversely, signals subscribing to the *etic* perspective focus on the environmental results of ensemble interactions. Within ADR, signals in this vein could concern the product of actual uses, the number of registered users, and the digital traces of workarounds.

Within the reviewed research, there is a strong tendency to rely entirely on emic signals (Asatiani et al., 2020; Giessmann & Legner, 2016; Hustad & Olsen, 2014; Mettler, 2017). Signals more strongly connected to the etic perspective include video observations of user interactions (Giesbrecht et al., 2017), careful external analysis of the *output* of the ensemble artifact (i.e., business models) (Ebel et al., 2016), and the external assessment of an ensemble after the intervention (Gregor et al., 2014). However, the literature review did not identify any deliberate combination of the two.

In the present research, several complementary emic and etic signals helped transition the artifact into the work system.

1. In the alpha version problem formulation phase, the emic signals highlighted a structural misalignment between existing contractual routines at the STA and developers' preference for non-discriminatory access openness. However, based on etic cues from SL and Trafiklab (Rudmark et al., 2012), we could be relatively comfortable in our assumption that developer adoption would increase should more open platform governance be employed.
2. In the beta version platform, we could operationalize the need for coherent search capabilities that developers expressed in interviews. We did this by collecting the actual use cases supported by smartphone apps and unsanctioned APIs.

3. When more experienced developers did not adopt the beta version API despite earlier emic signals pointing in this direction ("anything but HTML will suffice"), we collected etic cues from SL by capturing the data sources that were actually used some two years after the launch of Trafiklab.se. We then collected complementary emic signals to understand why these unsanctioned resources were still used.

While the importance of emic and etic perspectives in ADR has previously been highlighted by Brooks and Alam (2015), their argument was made to define an *elaborated* version of ADR known as action design ethnographic research (ADER). While such research is applicable in many circumstances, I argue that the use of complementary emic and etic signals in the context of guided emergence is quite useful in any ADR venture.

7.3.8 Concluding thoughts

In Chapter 7.3, I have sought to provide reflections on guided emergence based on the different ADR phases and the variance of ensemble signals. Since ADR draws on technology as structure (Orlikowski & Iacono, 2001), the mindful assessment of ensemble cues and their relationships to surrounding structures is at the heart of any ADR endeavor.

At one end of this continuum, an ADR project may be classified as a pure building system. By this, I refer to an intervention that is performed in a researcher-practitioner collaboration where the BIE is conducted in a temporal, artificial ensemble, outside the daily operations, and never infused into the work system. As a result, important *in situ* experimentation opportunities that were nearly impossible to achieve within the work system can open up. On the other hand, the building system may lack the necessary structures from the work system due to the building system's deliberately external placement.

At the other end of this continuum, an ensemble artifact may be classified as an institutionalized work system. By this, I refer to the ensemble as being equivalent to the clinical setting in which the artifact is intended to work. Thus, the end of the continuum reflects

the full range of ecological factors that must be inscribed for the ensemble artifact to function and solve the problem at hand. Since work systems are inherently difficult to modify, many ADR ventures will never be fully infused into a work system. Moreover, those that do eventually succeed will need to be able to gradually transform the ensemble from a building system to a work system. In the wake of such a truly authentic ensemble, researchers have suitable opportunities to both note and theorize anticipated and unanticipated behaviors.

I posit that such awareness is an essential trait of guided emergence since it will determine the types of traces that the surrounding ecology can inscribe into the ensemble artifact. This way, building system ensembles can only resemble the ecological factors that ADR teams included. Consequently, they may lack the structural traces of elements left out of such ensembles. While this mode of research holds potential for truly innovative and disruptive research, researchers must recognize the artificial nature of ensembles (although clinical) in their reflections and learning.

Consequently, I argue that work system ensembles will allow for more solid theorizing due to their full-fledged authenticity. Once an ensemble artifact is deployed into a truly authentic setting, new unforeseen use trajectories may open up (as per the internal integration of self-resourcing at the STA). Consequently, ADR theorizing efforts are strengthened by the type of traces that work system implementations provide.

7.4 Limitations and Future Research Opportunities

As with any research, this thesis comes with limitations. The first limitation, concerns that this research has been conducted within a single setting (albeit with more than one organization). As such, the principles presented in thesis should not uncritically be transferred to another setting before their mutability has been proven in other settings, or their scope has been more properly defined (Gregor & Jones, 2007).

To clarify this with an example, currently (May 2021), within the Swedish context, there is a publicly discussed case of outlaw

innovation occurring on top of the City of Stockholm's systems. This example concerns a digital school platform where parents, students, and teachers e.g., can interact and record study progress. Due to a perceived lack of usability, external developers have reverse-engineered the platform's internal APIs and built new end-user applications with a presumably more delightful user experience[39], using these unsanctioned interfaces. This action has prompted the City of Stockholm to first investigate[40] and later litigate the outlaw innovators. The city has chosen to engage in an attack response as they consider this product hacking an infringement on the city's data policies. Should the city reconsider this position and opt for the influencing response, perhaps even using an open platform, this could constitute an opportunity for testing the viability of these principles.

However, this context also differs in terms of ecological factors that may influence the ensemble design. For instance, the response from Stockholm city was exercised prior to outlaw innovations reached any substantial user base penetration. Consequently, since use patterns still is in a formative phase, what constitutes stable use cases with reuse potential may at the current time be difficult to determine. Second, given the attack response by the city, there is currently a high degree of conflict between the city and the outlaw innovators being portrayed in media. As such, the starting point from a relational standpoint might be different compared to the empirical setting in this research, and thus influence the process aspects of emulation.

Another important limitation and opportunity for future research, concerns elaborating more on the design principles in Chapter 6. To further develop these principles into a design theory following the recommendation per Walls and associates (Walls et al., 1992, 2004), further work is necessary on establishing formal meta requirements alongside supporting kernel theories for the process principles.

[39] See https://skolplattformen.org/ (In Swedish)

[40] https://start.stockholm/globalassets/start/forskola-och-skola/skolplattformen/pm---rattsutredning-oppna-skolplattformen-2021-04-14-final.pdf (In Swedish)

Moreover, more research to understand the intricate details of what constitutes a flexible search is called for. Currently, my principles for the beta version, suggests conducting evaluation authentically to untangle the more precise meaning of flexible searches in a given context. More research, and possibly additional sub-principles (Gregor et al., 2020), may prove such a stage in the beta version development obsolete.

Article	Ensemble artifact	Explicit use of Guided Emergence	Ensemble signals guiding transitions			Deployed release version and followed-up study
			Problem to Alpha version	*Alpha to Beta version*	*Beta to release version*	
Asatiani et al. (2020)	Organizational culture handbook	No	- Workshops - Interviews with the client	- Open group discussions - Anonymous written feedback	- Anonymous written feedback - Acceptance from leadership team	Deployed version, follow-up study conducted
Ebel et al. (2016)	Business model development tool	No	- Reviewing client product portfolios - Interviews with external domain experts	- Questionnaires from testers	- User-generated business models rated by researchers and external experts	No deployed version (deployment initiated)
Giesbrecht et al. (2017)	Service Encounter Thinklets	No	- Observations of service encounters client organization	- Evaluations with end-users - workshops with ADR team	- Video observations of simulated encounters - User Questionnaires/ Interviews	No deployed version
Giessmann and Legner (2016)	PaaS Business Models	No	- Explicating current business model - Analysis of competitor business models	- Workshops with ADR team	- Qualitative and quantitative assessment by client business models leadership roles	Deployed version, follow-up study conducted
Gregor et al. (2014)	Sweet spot change strategy e-government in least developed countries	Yes	- Focus groups with client organizations - Interviews with external stakeholders	- Roundtable discussions with external stakeholders and course participant feedback		Deployed version, follow-up study conducted
Hustad and Olsen (2014)	Teaching framework Enterprise Systems for IS graduates	No	- Student course evaluations and learning outcomes were used during the entire study as the teaching framework was being subjected to three revisions. All revisions were made directly to the work system.			Deployed version, no follow-up study conducted

Article	Ensemble artifact	Explicit use of Guided Emergence	Ensemble signals guiding transitions			Deployed release version and followed-up study
			Problem to Alpha version	*Alpha to Beta version*	*Beta to release version*	
Mettler (2017)	Professional social networks	No	- Interviews with relevant professionals	- User feedback on mockup screens	- Focus group feedback on beta - Interview feedback on beta	No deployed version

Table 14 – Transitions from building to work systems in extant ADR Research, published in AIS Senior Scholars' Basket of Journals

REFERENCES

Arnestrand, E., Lundh, A., Rudmark, D., & Östlund, H. (2017). *Kraftsamling Öppna Trafikdata - en målbild för Sverige*. Samtrafiken. Retrieved from https://samtrafiken.se/wp-content/uploads/2017/04/Slutrapport-_-Kraftsamling-%C3%96ppna-Trafikdata-en-m%C3%A5lbild-f%C3%B6r-Sverige-v-1.0-_-Diarienummer-Vinnova-2016-03467.pdf

Asatiani, A., Hämäläinen, J., Penttinen, E., & Rossi, M. (2020). Constructing Continuity Across the Organisational Culture Boundary in a Highly Virtual Work Environment. *Information Systems Journal, 31*, 62 - 93. doi:10.1111/isj.12293

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

Baldwin, C., & Woodard, J. (2009). The architecture of platforms: a unified view. In A. Gawer (Ed.), *Platforms, Markets and Innovation* (pp. 19-44). Cheltenham, UK: Edward Elgar Publishing.

Barley, S. (1986). Technology as an Occasion for Structuring: Evidence from Observations of CT Scanners and the Social Order of Radiology Departments. *Administrative science quarterly, 31*(1), 78-108. doi:10.2307/2392767

Barrett, M., & Holeman, I. (2017). Insights from an ICT4D Initiative in Kenya's Immunization Program: Designing for the Emergence of Sociomaterial Practices. *Journal of the Association for Information Systems, 18*(12), 900-930. doi:10.17705/1jais.00476

Bonina, C., & Eaton, B. (2020). Cultivating Open Government Data Platform Ecosystems Through Governance: Lessons From Buenos Aires, Mexico City and Montevideo. *Government Information Quarterly, 37*(3), 101479. doi:https://doi.org/10.1016/j.giq.2020.101479

Boudreau, K. (2010). Open Platform Strategies and Innovation: Granting Access vs. Devolving Control. *Management Science, 56*, 1849-1872. doi:10.1287/mnsc.1100.1215

Boudreau, K., & Lakhani, K. (2009). How to Manage Outside Innovation. *MIT Sloan Management Review, 50*(4), 69-75.

Braun, V., & Herstatt, C. (2008). The Freedom-Fighters: How Incumbent Corporations are Attempting to Control User-Innovation. *International Journal of Innovation Management, 12*(03), 543-572. doi:10.1142/S1363919608002059

Brooks, L., & Alam, M. S. (2015). Designing an Information System for Updating Land Records in Bangladesh: Action Design Ethnographic Research (ADER). *Information Systems Frontiers, 17*(1), 79-93. doi:10.1007/s10796-014-9512-7

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. doi:https://doi.org/10.1016/j.respol.2019.03.016

Cennamo, C., Ozalp, H., & Kretschmer, T. (2018). Platform Architecture and Quality Trade-offs of Multihoming Complements. *Information Systems Research, 29*(2), 461-478. doi:10.1287/isre.2018.0779

Coff, R., Coff, D., & Eastvold, R. (2006). The Knowledge-Leveraging Paradox: How to Achieve Scale without Making Knowledge Imitable. *Academy of Management Review, 31*(2), 452-465. doi:10.5465/amr.2006.20208690

Dattée, B., Alexy, O., & Autio, E. (2018). Maneuvering in Poor Visibility: How Firms Play the Ecosystem Game when Uncertainty is High. *Academy of Management Journal, 61*(2), 466-498. doi:10.5465/amj.2015.0869

de Reuver, M., Sørensen, C., & Basole, R. (2018). The Digital Platform: A Research Agenda. *Journal of Information Technology, 33*(2), 124-135. doi:10.1057/s41265-016-0033-3

DiMaggio, P., & Powell, W. (1983). The Iron Cage Revisited: Institutional Isomorphism and Collective Rationality in Organizational Fields. *American Sociological Review, 48*(2), 147-160. doi:10.2307/2095101

Eaton, B., Elaluf-Calderwood, S., Sørensen, C., & Yoo, Y. (2015). Distributed Tuning of Boundary Resources - The Case of Apple's iOS Service System. *MIS Quarterly, 39*(1), 217-244.

Ebel, P., Bretschneider, U., & Leimeister, J. M. (2016). Leveraging Virtual Business Model Innovation: A Framework for Designing Business Model Development Tools: Leveraging Virtual Business Model Innovation. *Information Systems Journal, 26*(5), 519-550. doi:10.1111/isj.12103

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). Cheltenham, UK: Edward Elgar Publishing.

Ethiraj, S., & Levinthal, D. (2004). Modularity and Innovation in Complex Systems. *Management Science, 50*(2), 159-173. doi:10.1287/mnsc.1030.0145

Evans, D. S., Hagiu, A., & Schmalensee, R. (2006). *Invisible Engines: How Software Platforms Drive Innovation and Transform Industries*. Cambridge, MA: The MIT Press.

Flowers, S. (2008). Harnessing the Hackers: The Emergence and Exploitation of Outlaw Innovation. *Research Policy, 37*(2), 177-193. doi:10.1016/j.respol.2007.10.006

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. doi:10.1111/isj.12186

Gawer, A. (2014). Bridging Differing Perspectives on Technological Platforms: Toward an Integrative Framework. *Research Policy, 43*(7), 1239-1249. doi:https://doi.org/10.1016/j.respol.2014.03.006

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.

Giesbrecht, T., Schwabe, G., & Schenk, B. (2017). Service Encounter Thinklets: How to Empower Service Agents to Put Value Co-Creation Into Practice: Service Encounter Thinklets. *Information Systems Journal, 27*(2), 171-196. doi:10.1111/isj.12099

Giessmann, A., & Legner, C. (2016). Designing Business Models for Cloud Platforms: Designing Business Models for Cloud

Platforms. *Information Systems Journal, 26*(5), 551-579. doi:10.1111/isj.12107

Gioia, D., & Thomas, J. (1996). Identity, Image, and Issue Interpretation: Sensemaking During Strategic Change in Academia. *Administrative science quarterly, 41*(3), 370-403. doi:10.2307/2393936

Gregor, S., Imran, A., & Turner, T. (2014). A 'Sweet Spot' Change Strategy for a Least Developed Country: Leveraging E-Government in Bangladesh. *European Journal of Information Systems, 23*(6), 655-671. doi:10.1057/ejis.2013.14

Gregor, S., & Jones, D. (2007). The Anatomy of a Design Theory. *Journal of the Association for Information Systems, 8*, 312-335.

Gregor, S., Kruse, L., & Seidel, S. (2020). Research Perspectives: The Anatomy of a Design Principle. *Journal of the Association for Information Systems, 21*, 1622-1652. doi:10.17705/1jais.00649

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. doi:10.1080/10438599000000009

Hjalmarsson, A., & Rudmark, D. (2012). Designing Digital Innovation Contests. In K. Peffers, M. Rothenberger, & B. Kuechler (Eds.), *Design Science Research in Information Systems. Advances in Theory and Practice* (Vol. 7286, pp. 9-27): Springer Berlin.

Hustad, E., & Olsen, D. (2014). Educating Reflective Enterprise Systems Practitioners: A Design Research Study of the Iterative Building of a Teaching Framework: Educating Reflective ES Practitioners. *Information Systems Journal, 24*(5), 445-473. doi:10.1111/isj.12032

Iivari, J. (2015). Distinguishing and Contrasting Two Strategies for Design Science Research. *European Journal of Information Systems, 24*(1), 107-115. doi:10.1057/ejis.2013.35

Jha, S., & Pinsonneault, S. (2016). The Evolution of an ICT Platform-Enabled Ecosystem for Poverty Alleviation: The Case of eKutir. *MIS Quarterly, 40*(2), 431-445. doi:10.25300/MISQ/2016/40.2.08

Kapoor, R., & Agarwal, S. (2017). Sustaining Superior Performance in Business Ecosystems: Evidence from Application Software Developers in the iOS and Android Smartphone Ecosystems. *Organization Science, 28*(3), 531-551. doi:10.1287/orsc.2017.1122

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. doi:10.1287/isre.2018.0786

Kartas, A., & Goode, S. (2012). Use, Perceived Deterrence and the Role of Software Piracy in Video Game Console Adoption. *Information Systems Frontiers, 14*(2), 261-277. doi:10.1007/s10796-010-9236-2

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. doi:10.1080/07421222.2018.1440772

Kock, N. (2003). Action Research: Lessons Learned From a Multi-Iteration Study of Computer-Mediated Communication in Groups. *IEEE Transactions on Professional Communication, 46*(2), 105-128. doi:10.1109/TPC.2003.813164

Koutsikouri, D., Lindgren, R., & Henfridsson, O. (2017). Building Digital Infrastructures: Towards an Evolutionary Theory of Contextual Triggers. In *Proceedings of the 50th Hawaii International Conference on system Sciences*.

Koutsikouri, D., Lindgren, R., Henfridsson, O., & Rudmark, D. (2018). Extending Digital Infrastructures: A Typology of Growth Tactics. *Journal of the Association for Information Systems, 19*(10), 1001-1019. doi:10.17705/1jais.00517

Labianca, G., Fairbank, J., Thomas, J., Gioia, D., & Umphress, E. (2001). Emulation in Academia: Balancing Structure and Identity. *Organization Science, 12*(3), 312-330. doi:10.1287/orsc.12.3.312.10101

Lee, A. (2007). Action is an Artifact. In N. Kock (Ed.), *Information Systems Action Research* (Vol. 13, pp. 43-60). Boston, MA: Springer US.

Lessig, L. (2004). *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity*. New York: Penguin Press.

Li, X., Sun, S., Chen, K., Fung, T., & Wang, H. (2015). Design Theory for Market Surveillance Systems. *Journal of Management Information Systems, 32*(2), 278-313. doi:10.1080/07421222.2015.1063312

Lyytinen, K., & Newman, M. (2008). Explaining Information Systems Change: A Punctuated Socio-Technical Change Model. *European Journal of Information Systems, 17*(6), 589-613. doi:10.1057/ejis.2008.50

Mandviwalla, M. (2015). Generating and Justifying Design Theory. *Journal of the Association for Information Systems, 16*(5), 314-344.

Markus, L., Majchrzak, A., & Gasser, L. (2002). A Design Theory for Systems That Support Emergent Knowledge Processes. *MIS Quarterly, 26*, 179-212.

Mettler, T. (2017). Contextualizing a Professional Social Network for Health Care: Experiences From an Action Design Research Study. *Information Systems Journal, 28*. doi:10.1111/isj.12154

Mollick, E. (2005). Tapping Into the Underground. *MIT Sloan Management Review, 46*(4), 21.

Morris, M., Leung, K., Ames, D., & Lickel, B. (1999). Views from Inside and Outside: Integrating Emic and Etic Insights about Culture and Justice Judgment. *The Academy of Management Review, 24*(4), 781-796. doi:10.2307/259354

Mukhopadhyay, S., Bouwman, H., & Jaiswal, M. P. (2019). An Open Platform Centric Approach for Scalable Government Service Delivery to the Poor: The Aadhaar Case. *Government Information Quarterly, 36*(3), 437-448. doi:https://doi.org/10.1016/j.giq.2019.05.001

Mullarkey, M., & Hevner, A. (2019). An Elaborated Action Design Research Process Model. *European Journal of Information Systems, 28*(1), 6-20. doi:10.1080/0960085X.2018.1451811

O'Mahony, S., & Karp, R. (2020). From Proprietary to Collective Governance: How Do Platform Participation Strategies

Evolve? *Strategic Management Journal, n/a*(n/a). doi:https://doi.org/10.1002/smj.3150

Ondrus, J., Gannamaneni, A., & Lyytinen, K. (2015). The Impact of Openness on the Market Potential Of Multi-Sided Platforms: A Case Study of Mobile Payment Platforms. *Journal of Information Technology, 30*(3), 260-275. doi:10.1057/jit.2015.7

Orlikowski, W., & Iacono, S. (2001). Research Commentary: Desperately Seeking the "IT" in IT Research—A Call to Theorizing the IT Artifact. *Information Systems Research, 12*(2), 121-134. doi:10.1287/isre.12.2.121.9700

Oxford English Dictionary. (2019). "Emulation, n.". Retrieved from http://www.oed.com/view/Entry/61461

Parker, G., & Van Alstyne, M. (2017). Innovation, Openness, and Platform Control. *Management Science, 64*(7), 3015-3032. doi:10.1287/mnsc.2017.2757

Parker, G., Van Alstyne, M., & Choudary, S. (2016). *Platform Revolution: How Networked Markets Are Transforming the Economy and How to Make Them Work for You*. New York, NY: WW Norton & Company.

Parnas, D. (1972). On the Criteria to Be Used in Decomposing Systems Into Modules. *Communications of the ACM, 15*, 1053-1058. doi:10.1145/361598.361623

Parnas, D., Clements, P., & Weiss, D. (1985). The Modular Structure of Complex Systems. *IEEE Transactions on Software Engineering, SE-11*(3), 259-266. doi:10.1109/TSE.1985.232209

Pil, F., & Cohen, S. (2006). Modularity: Implications for Imitation, Innovation, and Sustained Advantage. *Academy of Management Review, 31*(4), 995-1011. doi:10.5465/amr.2006.22528166

Postigo, H. (2003). From Pong to Planet Quake: Post-Industrial Transitions from Leisure to Work. *Information, Communication and Society, 6*(4), 593-607. doi:10.1080/1369118032000163277

Rivkin, J. (2001). Reproducing Knowledge: Replication without Imitation at Moderate Complexity. *Organization Science, 12*(3), 274-293.

Rudmark, D. (2013). The Practices of Unpaid Third-Party Developers – Implications for API Design. In *Proceedings of the 19th Americas Conference on Information Systems (AMCIS 2013)*.

Rudmark, D. (2021). *Designing Open Platform Emulation*. Paper presented at the Under review at the 42nd International Conference on Information Systems (ICIS 2021).

Rudmark, D., Arnestrand, E., & Avital, M. (2012). Crowdpushing: The Flipside of Crowdsourcing. In *Proceedings of the 20th European Conference on Information Systems (ECIS 2012)*.

Rudmark, D., & Ghazawneh, A. (2011). Third-Party Development for Multi-Contextual Services: On the Mechanisms of Control. In *Proceedings of the 19th European Conference on Information Systems (ECIS 2011)*.

Rudmark, D., & Lind, M. (2011). Design Science Research Demonstrators for Punctuation – The Establishment of a Service Ecosystem. In H. Jain, A. Sinha, & P. Vitharana (Eds.), *Service-Oriented Perspectives in Design Science Research* (Vol. 6629, pp. 153-165): Springer Berlin Heidelberg.

Saadatmand, F., Lindgren, R., & Schultze, U. (2019). Configurations of Platform Organizations: Implications for Complementor Engagement. *Research Policy, 48*(8), 103770. doi:https://doi.org/10.1016/j.respol.2019.03.015

Schulz, C., & Wagner, S. (2008). Outlaw Community Innovations. *International Journal of Innovation Management, 12*(03), 399-418. doi:10.1142/S1363919608002084

Schäfer, M. (2011). *Bastard Culture! How User Participation Transforms Cultural Production*. Amsterdam: Amsterdam University Press.

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

Sein, M., & Rossi, M. (2019). Elaborating ADR While Drifting Away From Its Essence: A Commentary on Mullarkey and Hevner. *European Journal of Information Systems, 28*(1), 21-25. doi:10.1080/0960085X.2018.1527189

Strauss, A., & Corbin, J. (1990). *Basics of Qualitative Research: Grounded Theory Procedures and Techniques.* Newbury Park: Sage.

Teece, D. (2007). Explicating Dynamic Capabilities: The Nature and Microfoundations of (Sustainable) Enterprise Performance. *Strategic Management Journal, 28*(13), 1319-1350. doi:10.1002/smj.640

Teece, D., Pisano, G., & Shuen, A. (1997). Dynamic Capabilities and Strategic Management. *Strategic Management Journal, 18*(7), 509-533. doi:10.1002/(SICI)1097-0266(199708)18:7<509::AID-SMJ882>3.0.CO;2-Z

Tennie, C., Call, J., & Tomasello, M. (2010). Evidence for Emulation in Chimpanzees in Social Settings Using the Floating Peanut Task. *PLOS ONE, 5*(5), e10544. doi:10.1371/journal.pone.0010544

Tilson, D., Lyytinen, K., & Sørensen, C. (2010). Research Commentary—Digital Infrastructures: The Missing IS Research Agenda. *Information Systems Research, 21*(4), 748-759. doi:10.1287/isre.1100.0318

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

Tiwana, A. (2015). Platform Desertion by App Developers. *Journal of Management Information Systems, 32*(4), 40-77. doi:10.1080/07421222.2015.1138365

Tiwana, A., Konsynski, B., & Bush, A. A. (2010). Research Commentary--Platform Evolution: Coevolution of Platform Architecture, Governance, and Environmental Dynamics. *Information Systems Research, 21*, 675-687. doi:10.1287/isre.1100.0323

Tucker, S. (1965). Emulation of large systems. *Communications of the ACM, 8*(12), 753-761.

Venable, J., Pries-Heje, J., & Baskerville, R. (2016). FEDS: a Framework for Evaluation in Design Science Research. *European Journal of Information Systems, 25*(1), 77-89. doi:10.1057/ejis.2014.36

von Hippel, E., & Katz, R. (2002). Shifting Innovation to Users via Toolkits. *Management Science, 48*, 821-833. doi:10.1287/mnsc.48.7.821.2817

Walls, J., Widmeyer, G., & El Sawy, O. (1992). Building an Information System Design Theory for Vigilant EIS. *Information Systems Research, 3*(1), 36-59. doi:10.1287/isre.3.1.36

Walls, J., Widmeyer, G., & El Sawy, O. (2004). Assessing Information System Design Theory in Perspective: How Useful Was Our 1992 Initial Rendition? *JITTA: Journal of Information Technology Theory and Application, 6*(2), 43-58.

Wareham, J., Fox, P., & Cano Giner, J. (2014). Technology Ecosystem Governance. *Organization Science, 25*(4), 1195-1215. doi:10.1287/orsc.2014.0895

West, J. (2003). How Open Is Open Enough? Melding Proprietary and Open Source Platform Strategies. *Research Policy, 32*(7), 1259-1285. doi:10.1016/S0048-7333(03)00052-0

Westin, S., & Sein, M. (2015). The Design and Emergence of a Data/Information Quality System. *Scandinavian Journal of Information Systems, 27*(1), 3-26.

APPENDIX A. CODE EXAMPLES DART GROUP

TRANSCRIPT EXCERPT	OPEN CODE	AXIAL CODE
"I'll just finish, I will not talk more about it - but with this Philips TV as information service SL bought the service from a contractor, because they don't know how to do it, you have a function procurement in some way instead"	Insufficient End-user Technology Development Capabilities	**Difficulties Creating End-user Services with Extensive Coverage**
"For a while we implemented a service for Nokia with a java client so you could get a map exactly where the train was, a very cool service but with the next java version it was gone because we did not get money to upgrade it – it caused much frustration!"	Application Adaptation to Specific Devices Costly	
"It reads as follows: In the so-called PSI directive - PSI stands for public service information - the EU has decided that authorities must provide unprocessed raw data at self-cost price. Sweden considered for a long time that the Swedish agreement would be amended to comply with the directive, but after the EU Commission initiated an investigation into Sweden's breach of the directive, they will initiate an investigation that proposes that PSI adapt - a PSI adaptation of Swedish law."	Legislative pressures	**Pressure to release data more publicly**
"But what has happened is that a number of pirate services have been created where there are clever boys and girls who have hacked their mobile service against our web service and created services that they had on the agenda. They really slapped us on the wrist!"	Pressures from existing unsanctioned developers	

APPENDIX B. TRAVELHACK CODE EXAMPLE

INCIDENT TRANSCRIPT	INCIDENT CODE INSTANCES
(D1) Wasn't there something where you could find distances in road traffic. The Google Maps Road Traffic API? Västtrafik? *(D2)What did you say? For streets or?* *(D1)Yes, for distance to work? Västtrafik API, it did not work at all or? Labs?* *(D2)I can't even find the base URL. You should obviously log in with e-mail then, but once you get in there, there are no links to type but only method names, so you sit and "yeah, now what?"* *(D1) You have managed to log in but nothing happens after that?* *(D2)No, no, there is no page that says "this is how you do it", how-to. Nothing.* *(D1) Should you use e-mail?* *(D2)Email, I also thought it was name I should use so I "argh!" I'm thinking about whether I should try with Trafiklab services instead where they have existing API keys* *(D1)Yes. "For developers. Join this group to take part in Västtrafik's APIs". "APIs and documentation". "Traffic disruptions". "Documentation of new API". Here we have it!* *(D2)Did you find it? I clicked around like crazy.* *(D1) Pdf file is there and sample file to generate correct calls* *(D2)I clicked on "for developers" and then I just got to my login box. Strange!* *(D2)This is what it looks like for me when I click on "For developers"* *(D1)And then you click on the developer group* *(D2)It still looks like this* *(Ju) You click on the API documentation* *(D2)Ah... [Sigh]*	API :: TrafficInformation Incident Manifestation :: Asking for peer support IncidentCause :: Base URL Location Use Trajectory :: Continued Use User Feelings :: Frustration

APPENDIX C. INTERVIEW GUIDE THE STA

- Intro
  - Recording
  - Background us
  - Background project

- Background and third-party development
  - Describe your background and role at the Swedish Transport Administration?
  - How does the Swedish Transport Administration support external parties who want to develop innovative services on STA data?
  - What needs do you think these have?
  - How well are you familiar with what third-party developers exist? Are there others in the organization who are aware of this?
  - What does it mean that many third-party developers use unsanctioned data deliveries?
  - What does it mean that many travelers use services based on unofficial data deliveries?
  - Have you acted against any third party actor?

- Current data deliveries
  - What types of data deliveries for rail are available today?
  - What data is available?

- Who are the recipients of the information?
  - Depending on the person: Describe UTIN / Lastkajen?
    - Purpose
    - History
    - What works well / needs improvement
  - What deliveries are missing?

- In an ideal world: How would the Swedish Transport Administration's deliveries of traffic information for railways work and look like?
- How does the Swedish Transport Administration view Open Data? The PSI Directive? In what way does the PSI directive affect the Swedish Transport Administration's information supply?

- Check-out

- How do you view the goal of this project?
- In an ideal world, what would you like to achieve with the project?
- How does it relate to organizational goals at the Swedish Transport Administration?
- Contact again, for example on Skype.

APPENDIX D. INTERVIEW TEMPLATE DEVELOPERS ALPHA VERSION

- Intro
    - Recording
    - Background myself
    - Background project
- The service
    - What service (s) have you created?
    - When did the development begin?
    - How did you come up with the idea?
    - Why did you create the service? Motivations
    - Long-term plan, maintenance - when and how did the service become more than a prototype?
    - How did the service spread and when?
    - What dialogue / contact / feedback do you have with end users of the service?
        - In such cases, how have your users expressed a need for information from the Swedish Transport Administration?
        - How did you communicate this to the Swedish Transport Administration? In what way did you experience the STA attitude towards this?
    - What contact have you had with the Swedish Transport Administration and the actors with data / information?
        - What do you think have been critical issues for the STA to release information to external actors?

        - Do you see a change in the response to these issues from the Swedish Transport Administration?
        - In what way has the STA attitude influenced your / your work and development of services?
    - What data source do you use today?
- Technology
    - How does the service work? How do you retrieve information?
    - Changes in the course of development?
    - In an ideal world - how would information be delivered from the Swedish Transport Administration? What support or other help / support would you receive (or receive)?
    - Do you make money from the service? What does the business model look like?
    - The future of the service? Development ideas? Other projects / services?
    - What other services are on the market that we should contact

APPENDIX E. EVALUATION INTERVIEW PROTOCOL BETA VERSION

Interview template new developers

Below are suggestions for questions to third-party developers. The text in parentheses is so-called " probes " which are extra interesting and should be asked in a suitable context .

Background and service

Tell me about the service you have developed

(Target group: yourself / others, etc.)

(why build the service: solve a problem, learning, mission, commercial service)

(What have you spent the most time on?)

(how do you develop: leisure, service: at home / at work)

Tell me about previous experience of development

(programming, APIs, mobile / web services)

(More projects?)

(In those projects, what do you work on the most?

What APIs have you used?

(how did you execute the selection?)

Simple and inviting registration and access

How did you experience the process of accessing the API (" time to first request ")?

(Long / short, smooth / frustrating, simple / cumbersome)

Do you have any idea about what's in the user agreement?

(If so, what did you think of it?)

Which ev. possible improvements could be made to access the API?

Understand content, possibilities, and limitations

For the API (s) you used, how did you go about understanding what data was available and what could be done?

(Documentation, code sample, sample response, API console)

What is your experience of understanding what the API contains, what can and cannot be done with the API?

(smooth / frustrating, simple / cumbersome, inspiring / disappointing)

How did your assessment of the content of the API affect your work with the service?

(Did not affect, had to change (what?), Did not want to continue)

Which ev. could improvements be made to better understand the API 's content, capabilities and limitations ?

Working with the API

Can you describe for which environment the service was developed?

(This may have been described in the previous question about the service)

(Development environment / language, user platform, integrated services (eg map services))

Given the environment, and the service you wanted to develop, how did you experience the API?

(Did anything change in the service? What in that case .?

What is your overall experience of working with the API?

Production set-up

Do you think the service will go into production?

(When in that case .?)

Isf ., Would you need to make any changes to your service (regarding the API)?

(Own server environment)

(More calls at trafiklab.se)

How do you experience the work that must be done to take a job in production?

(Point out that this applies to work that is linked to the API)

(The process of getting more calls)

(Lack of written agreements - good or bad)

Scraping

In the past, the services developed have been based on scraped data. Would that be an option for you?

(Why / why not?)

(Describe the advantages / disadvantages of scraping / APIs)

If so, what, if anything, would need to be changed for you to use official APIs instead?

Summary

What is your overall impression of the Swedish Transport Administration's APIs at Trafiklab ?

Want to add something else we haven't covered?

Interview template existing developers

Background and service

Can you briefly describe your service and why you created it?

How do you retrieve data today?

How has this mechanism worked so far?

(Possible problems, more degrees of freedom)

What APIs have you used?

(how did you execute the selection?)

(how did you find out what APIs existed?)

Get access

How did you experience the process of accessing the API (" time to first request ")?

(Long / short, smooth / frustrating, simple / cumbersome)

Do you have any idea about the user agreement?

(If so, what did you think of it?)

How do you see the API being provided via the traffic lab ?

(Together with other traffic APIs etc. ?)

Which ev. possible improvements could be made to access the API?

Understand content, opportunities and limitations

For the API (s) you used, how did you go about understanding what data was available and what could be done?

(Documentation, code sample, sample response, API console)

What is your experience of understanding what the API contains, what can and cannot be done with the API?

(smooth / frustrating, simple / cumbersome, inspiring / disappointing)

How did your assessment of the API's content affect your work with your existing service?

(Matched in terms of content, not possible to move)

Which ev. could improvements be made to better understand the API 's content, capabilities and limitations ?

Working with the API

Can you describe for which environment the service was developed?

(This may have been described in the previous question about the service)

(Development environment / language, user platform, integrated services (eg map services))

Given the environment, and the service you wanted to develop, how did you experience the API?

(Was there something in the service that was not compatible with the API? What in that case .?

What is your overall experience of working with the API?

Production set-up

Do you think you will move the service towards the official APIs ?

(When in such case .?)

(If not, why? What would need to change for this to happen?)

What do you need to do to take the service in production against the official API (regarding the API)?

(Dedicated server environment)

(Changes in the service)

(More calls at trafiklab.se)

How do you experience the work that needs to be done to move a service from scraping to official APIs?

(Point out that this applies to work that is linked to the API, after development of the service)

(The process of getting more calls)

(Lack of written agreements - good or bad)

Summary

What is your overall impression of the Swedish Transport Administration's APIs at traffic labs ?

Comments on your participation in the project?

(interview 1, workshop, spec , launch, interview 2)

Want to add something else?.

APPENDIX F. EVALUATION INTERVIEW PROTOCOL RELEASE VERSION

New developers

Below are questions to third-party developers. The text in parentheses is so-called "probes" which are extra interesting and should be asked in a suitable context.

Background and service

Tell us a little about the service you have developed

(Target group: yourself / others, in that case which)

(why build the service: solve a problem, learning, mission, commercial service)

(What have you spent the most time on?)

(how do you develop: leisure, service: at home / at work)

Tell about previous experience of development

(programming, APIs, mobile / web services)

(More projects?)

(In those projects, what do you work on the most?

Registration and access

How did you experience the process of accessing the API ("time to first request")?

(Long / short, smooth / frustrating, simple / cumbersome)

Do you have any idea about the user agreement?

(If so, what did you think of it?)

Which ev. possible improvements could be made to access the API?

Content, opportunities and limitations

For the API (s) you used, how did you go about understanding what data was available and what could be done?

(Documentation, code sample, sample response, API console)

Were these supports useful? How?

What could be better to understand what the API contains?

What is your experience of understanding what the API contains, what can and cannot be done with the API?

(smooth / frustrating, simple / cumbersome, inspiring / disappointing)

How did your assessment of the API's content affect your work with your existing service?

(Matched in terms of content, not possible to move)

Which ev. could improvements be made to better understand the API's content, capabilities, and limitations?

Working with the API

Can you describe for which environment the service was developed?

(This may have been described in the previous question about the service)

(Development environment / language, user platform, integrated services (eg map services))

Given the environment, and the service you wanted to develop, how did you experience the API?

(Was there something in the service that was not compatible with the API? What in that case?

Was it harder or easier than expected?

Did you use the examples?

How did you experience these as support?

Could you develop what you wanted?

If not why?

Did it take a reasonable amount of time to solve what you wanted?

If not, what took too long?

What is your overall experience of working with the API?

Production set-up

Do you think the service will go into production?

(When in such case?)

In such case, would you need to make any changes to your service (regarding the API)?

(Own server environment)

How do you assess the work that must be done to take a job in production?

(Point out that this applies to work that is linked to the API)

(Lack of written agreements - good or bad)

Scraping

In the past, the services developed have been based on scraped data. Would that be an option for you?

(Why / why not?)

(Describe the advantages / disadvantages of scraping / APIs)

If so, what, if anything, would need to be changed for you to use official APIs instead?

Summary

What is your overall impression of the Swedish Transport Administration's Open API?

Would you recommend the API to others? Why, why not?

Want to add something else?

Interview template for existing developers

Background and service

Can you briefly describe your service and why you created it?

How do you retrieve data today?

How has the data retrieval worked so far?

(Possible problems, more degrees of freedom)

Get access

How did you experience the process of accessing the API ("time to first request")?

(Long / short, smooth / frustrating, simple / cumbersome)

Do you have any idea about the user agreement?

(If so, what did you think of it?)

How do you see the API being provided via the traffic lab?

(Together with other traffic APIs etc.?)

Which ev. possible improvements could be made to access the API?

Understand content, opportunities and limitations

For the API (s) you used, how did you go about understanding what data was available and what could be done?

(Documentation, code sample, sample response, API console)

Were these supports useful? How?

What could be better to understand what the API contains?

What is your experience of understanding what the API contains, what can and cannot be done with the API?

(smooth / frustrating, simple / cumbersome, inspiring / disappointing)

How did your assessment of the API's content affect your work with your existing service?

(Matched in terms of content, not possible to move)

Which ev. could improvements be made to better understand the API's content, capabilities, and limitations?

Working with the API

Can you describe for which environment the service was developed?

(This may have been described in the previous question about the service)

(Development environment / language, user platform, integrated services (eg map services))

Given the environment, and the service you wanted to develop, how did you experience the API?

(Was there something in the service that was not compatible with the API? What in that case?

Was it harder or easier than expected?

Did you use the existing examples?

How did you experience these as support?

Could you develop what you wanted?

If not why?

What is your overall experience of working with the API?

Production set-up

Do you think you will move the service towards the official APIs?

(When in that case?)

(If not, why? What would need to change for this to happen?)

What do you need to do to take the service in production against the official API (regarding the API)?

(Own server environment)

(Changes in the service)

How do you experience the work that needs to be done to move a service from scraping to official APIs?

(Point out that this applies to work that is linked to the API, after development of the service)

(Lack of written agreements - good or bad)

Summary

What is your overall impression of the Swedish Transport Administration's Open API?

Comments on your participation in the project?

(interview 1, workshop, spec, launch, interview 2)

Want to add something more?

APPENDIX G. DESIGN INTERVENTIONS AND OUTCOME

1.1. Artificial Platform Demonstration

Following the launch of Trafiklab.se and its relatively quick success, Sweden's Innovation Agency (Vinnova) was interested in funding projects that would lead to more actors publishing public transport data to third-party developers. One of Sweden's most important actors was the Swedish Transport Administration's, particularly their passenger train data. Trafiklab.se, together with researchers, approached STA and discussed whether publishing train data on Trafiklab.se was a viable option. The discussions led to a mutual agreement on engaging in a joint problem formulation phase, and in the case the results were positive, a pilot API would be developed and tested on Trafiklab.se. However, at this point, no promises on more permanent APIs were given.

At the outset of this investigation, third-party developers were not granted access to rail-related data, while data stemming from roads (such as accidents, road works, and traffic flows) were distributed freely. The primary rationale for the difference in third-party development on the rail and the road data was both due to 1) historical organizational factors[41], 2) uncertainties whether ownership of data was with the STA or the train operators, and, with mutual researcher-practitioner interest 3) ***how*** train data should potentially be made available to third-party developers, as commented by an STA strategist in charge of compiling a new third-party development strategy:

[41] The Swedish Transport Administration was the result of merger between The Swedish Road Administration, The Swedish Rail Administration and parts of the Swedish Maritime Administration, the Swedish Civil Aviation Administration and the Swedish Institute for Communications Analysis. In this context the Swedish Road Administration had a history of working closely with third-party developers while the Swedish Rail Administration did not.

> *We need to understand what needs developers have regarding things like formats, delivery qualities, and content. We also need to know why they need this to understand the value of actually delivering it in a better way, not just that they want something free of charge.*
> *Strategist at the STA*

Despite this lack of an official third-party developer program for train data, many rail-related apps relying on scraping had emerged. These apps were written by independent developers, primarily driven by self-experienced needs. A handful of these applications had gained a high number of downloads in application marketplaces[42]. The developer of one the leading smartphone applications explained why he started and persevered in his efforts:

> *In the beginning, I was only developing to meet my own needs. I do a lot of these little experiments out of curiosity and without a commercial goal. It is only when I see that it is being used and that there is a demand that I start to think commercially about it. But before the app made it to the top-ten list on App Store, I didn't really believe that so many other people had the same combination of being both a control freak and frustrated that they actually would search for an app that solved this for them. But apparently, there were… And this is still nothing you get rich by doing, but it's a service that is enjoyable to manage since it is so appreciated. You get in direct contact with other people in a way that I haven't experienced previously. You get thank-you-emails, it's quite bizarre but also makes it very rewarding to manage this kind of service.*
> *Developer A1*

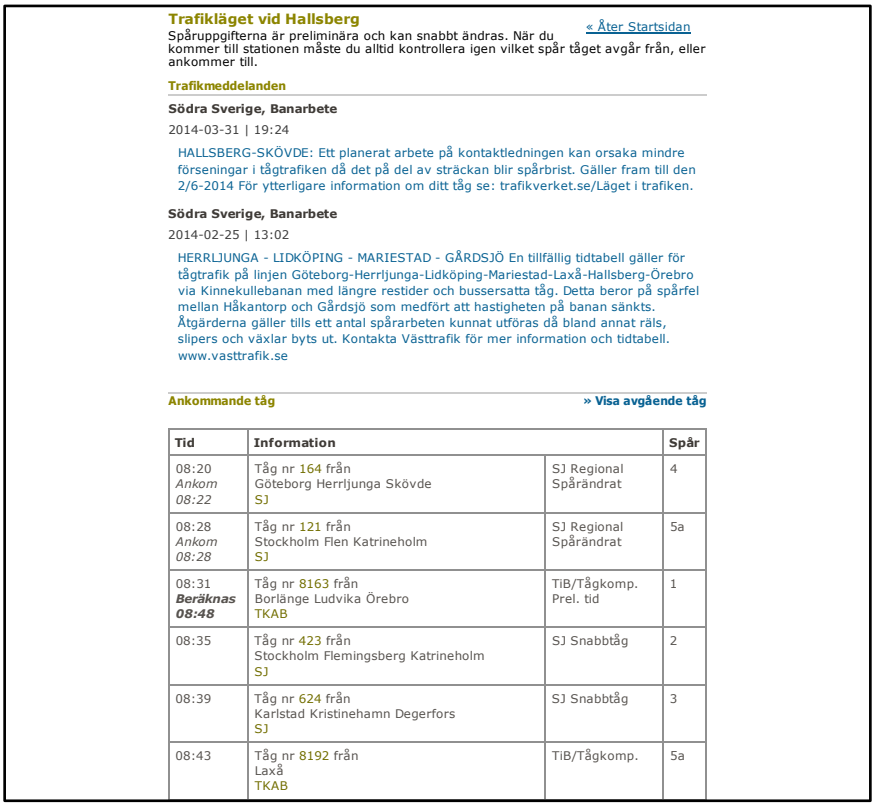
A more careful investigation of the existing apps revealed that the apps typically implemented a standard set of use cases. These included searching for a station based on a search string, getting

[42] Tågtavlan (using scraped data) was even installed by default on all of the STAs smartphones

departures/arrivals from a station and platform, and getting a particular train's status.

The data were scraped 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 (see

Figure 4).



Trafikläget vid Hallsberg

« Åter Startsidan

Spåruppgifterna är preliminära och kan snabbt ändras. När du kommer till stationen måste du alltid kontrollera igen vilket spår tåget avgår från, eller ankommer till.

Trafikmeddelanden

Södra Sverige, Banarbete

2014-03-31 | 19:24

HALLSBERG-SKÖVDE: Ett planerat arbete på kontaktledningen kan orsaka mindre förseningar i tågtrafiken då det på del av sträckan blir spårbrist. Gäller fram till den 2/6-2014 För ytterligare information om ditt tåg se: trafikverket.se/Läget i trafiken.

Södra Sverige, Banarbete

2014-02-25 | 13:02

HERRLJUNGA - LIDKÖPING - MARIESTAD - GÅRDSJÖ En tillfällig tidtabell gäller för tågtrafik på linjen Göteborg-Herrljunga-Lidköping-Mariestad-Laxå-Hallsberg-Örebro via Kinnekullebanan med längre restider och bussersatta tåg. Detta beror på spårfel mellan Håkantorp och Gårdsjö som medfört att hastigheten på banan sänkts. Åtgärderna gäller tills ett antal spårarbeten kunnat utföras då bland annat räls, slipers och växlar byts ut. Kontakta Västtrafik för mer information och tidtabell. www.vasttrafik.se

Ankommande tåg **» Visa avgående tåg**

Tid	Information		Spår
08:20 *Ankom* *08:22*	Tåg nr 164 från Göteborg Herrljunga Skövde SJ	SJ Regional Spårändrat	4
08:28 *Ankom* *08:28*	Tåg nr 121 från Stockholm Flen Katrineholm SJ	SJ Regional Spårändrat	5a
08:31 ***Beräknas*** ***08:48***	Tåg nr 8163 från Borlänge Ludvika Örebro TKAB	TiB/Tågkomp. Prel. tid	1
08:35	Tåg nr 423 från Stockholm Flemingsberg Katrineholm SJ	SJ Snabbtåg	2
08:39	Tåg nr 624 från Karlstad Kristinehamn Degerfors SJ	SJ Snabbtåg	3
08:43	Tåg nr 8192 från Laxå TKAB	TiB/Tågkomp.	5a

Figure 4 - Web page scraped by several developers

Another common way of accessing data was through a JavaScript interface at the STAs web page. This JavaScript interface was introduced when the STA deployed a new web page where the JavaScript interface was used to create more dynamic web service. In parallel to launching more dynamic services, the STA provided an unsanctioned API (albeit without developer documentation) to a system named *Orion*. Orion was designed to supply a range of end-

user services with data and therefore fused a broad range of transport-related datasets (such as accidents, train departures, weather forecasts, and ferry operations) into this data lake. On top of Orion, STA had developed a flexible query language (similar to SQL) that could be used to retrieve all information from Orion, accessible through JavaScript. As developers quickly discovered through trial-and-error, Orion contained a wealth of useful information. Still, given that Orion's interfaces were not intended for external use, developers needed to single-handedly figure out the underlying information model's workings and query language through trial-and-error exploration. Eventually, this lack of documentation prompted a more experienced third-party developer to reverse-engineer the API and provide instructions on how to support everyday rail traveler use cases[43] and thereby paving the way for inexperienced developers to use this resource more efficiently.

Two of the leading app developers, however, had not only created applications based on the scraped data. They also created "pirate APIs" on top of the unsanctioned data to use in their applications. These APIs were in some cases also offered to other third-party developers that hence did not have engage in time-consuming data retrieval activities, as commented by one of the "pirate API" developers:

> *I have published this API based on the massive effort I have put in to get some useful data out of this messy, underlying dataset, so that no one else will have to do again. So, I want to share what I have done, so that others may do something fun or useful or whatever it may be. My basic frustration is that, as a traveler, I do not get the information I think I deserve, not before, not during, or after my train ride. But I am just a single individual, and I can't possibly do all apps for all platforms or services or whatever it may be that people need. Developer A2*

[43] See http://tagtider.net/blogg/tjanster/trafikverket-exponerar-api/ and https://gist.github.com/RickardPettersson/1247081

These unsanctioned APIs had a very similar structure and corresponded to the use cases implemented in popular apps (see **Table 15**). Also, these interfaces were marked by quite limited data models, only conveying the essential data points to implement a specific use case (see Figure 5)

```
GET /stations/9.xml

<?xml version="1.0" encoding="utf-8"?>
<response>
    <stations>
        <station>
            <id>9</id>
            <name>Arlanda C</name>
            <code>74,arnc</code>
            <slug>arlanda-c</slug>
            <lat>59.6496</lat>
            <lng>17.9292</lng>
        </station>
    </stations>
</response>
```

Figure 5 - Tågtider API, Retrieving Arlanda C station

Coherent search	**Tåg.info API**	**Tågtider API**
Get all stations	http://api.tagtider.net/v1/stations.xml	http://tåg.info/stationer.xml?apikey=[api key]
Get a given station	http://api.tagtider.net/v1/stations/[station id].xml	http://tåg.info/[station_name].xml? apikey=[apikey]
Get departures and arrivals from a given station	http://api.tagtider.net/v1/stations/[station id]/transfers/	http://tåg.info/[station name].xml? apikey=[apikey]
Get departures from a given station	http://api.tagtider.net/v1/stations/[station id]/transfers/departures	-
Get arrivals from a given station	http://api.tagtider.net/v1/stations/[station id]/transfers/arrivals	-
Get all current trains	-	http://tåg.info/tag.XML?apikey=[apikey]
Get a given train	-	http://tåg.info/[train number].xml?apikey=[api key]
Get train operators	http://api.tagtider.net/v1/operators.xml	-
Get a given train operator	http://api.tagtider.net/v1/operators/[operator id].xml	-

Table 15 - Coherent search manifestations in unsanctioned APIs

When asked about what they would like to see in an official API, developers stressed capabilities focusing on simplicity and immediate

problem-solving. The developer of the most downloaded app for WindowsPhone expanded on this matter:

> *Well, simplicity is super-important – although it's OK if you can choose whether to get JSON or XML. I really prefer simple functions that actually work over more advanced stuff. Some companies expose their entire domain model to third parties, and I'm sure their domain is super clear to the company but not really understandable to anyone else. So, I would prefer companies who design their APIs for someone who doesn't know anything about their domain*
> *Developer A3*

Another developer described the characteristics of an attractive API to support common use cases:

> *What is important to me is whether the API reflects the use case; if you start from what most users want to do, such as travelers who need to travel from point A to point B, be able to download those data as quickly as possible and get it via a single API call is ideal, instead of a call that just returns a lot of information about a terminal and then you have to look further from there - is not as attractive.*
> *Developer A4*

The STA, on their part, experienced unsanctioned third-party developers as problematic from two perspectives:

First, as this development was anonymous, the STA could not establish regulated third-party developer relationships – a stark contrast to the STA's experiences from road data. For road data, the STA did periodic surveys and arranged conferences to understand third-party developers' satisfaction with the DATEX II feed from the STA. The STA wanted to achieve similar types of relationships with third-party developers using rail data.

> *Since they already have our data, it must be much better for both them and us that we agree on the terms and liabilities so that we can communicate when we change our interfaces. We are not able to get in touch with our customers, and we don't know who they are; we don't get any feedback. It is much better to have a relationship for both parties; we believe this, where we can negotiate each party's liabilities and resolve issues as they occur.*
> *Head of Traffic Information services, STA*

Second, the scraped interfaces were fragile and subject to change that periodically caused third-party applications to malfunction, which in turn, given the unsanctioned apps' popularity had invoked traveler annoyances. To this end, and the fact that the access was unregulated, the STA started to contact third-party developers of popular applications before launching redesigns of resources known to be used by scrapers. This interaction was necessary since hundreds of thousands of travelers would be affected if any of these applications malfunctioned, as commented by one developer:

> *They have contacted me before web server updates to check that nothing breaks on my side, that it works as intended, and as well as providing me warnings when changes are in progress.*
> *Developer A1*

Based on this background material, the product manager of Trafiklab.se and I assessed that the primary problem for the STA was the lack of access to emulation capabilities regarding the coherent searches that had emerged during app development. 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 by providing access to them. This way, the STA could decide what data, in what form, and under which terms and conditions the potentially increased openness could be implemented.

To materialize the foreseen solution, we drew on Trafiklab.se and its capabilities. More specifically, our solution blueprint included a new software layer residing at Trafiklab.se's cloud software provider, ApiGee. This layer would be used to effectively emulate the

capabilities that third-party developers desire through an interface offering access to coherent searches (like those exhibited in **Table 15**). Integration protocols would be a bare minimum, containing the coordinates and station name strings necessary to present correct traveler information.

On 2012-04-19, the project held a joint workshop summoning nine representatives from STA, two from Trafiklab.se, and myself. This workshop's idea was to bring different stakeholders together for the first time and test the design principles towards both third-party developers and more stakeholders within STA.

During the workshop, the suggested solution blueprint (in the form of a PowerPoint presentation, presenting both capabilities and overall implementation structures) was introduced to the audience. Regarding access openness through Trafiklab.se, developers were quite content with this type of openness regime. Regarding solution search mechanisms, the problem formulation phase revealed that only a limited number of use cases were implemented recurrently across third-party applications (such as list stations by name, recent departures and arrivals for each station and platform, and the status of a given train). Our idea was to package these recurrent use cases as dedicated REST endpoints to minimize developers' need to invest in industry-specific domain knowledge and create interfaces that would be suitable for direct consumption from mobile clients. This idea was also corroborated by the unsanctioned APIs that had emerged and had a very similar structure.

While the 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 to have access to all data points to design new types of services. One developer explained this position to attendees of the workshop:

> *I want an API to present more exhaustive data that don't have to be easy to understand – instead, I would like a focus on correctness and structure, and this goes for things like complete timestamps, not just hours and minutes but complete timestamps including dates. Also, I've seen 'train groups' in Orion, and this is something that would help immensely.*
> *Developer A1*

As illustrated by the viewpoint above, the more precise formats for such flexible searches appeared less critical. The developers were assigned to break out of the entire group and discuss what formats would be of interest for such capabilities. During these discussions, a joint position started to emerge, where data could be pretty crude, as put by one of the developers:

> *I would be super happy if that freakin' HTML is just transformed into XML so that everything looked exactly the same. That would be enough for me. I do not have higher requirements than that for an API at the moment. Of course, they can develop this further as much as they like and fine-tune and gold-spray it, but I don't think it has to be so damn advanced.*
> *Developer A2*

After these developer-internal discussions, one of the developers summarized the talks to the other participants this way:

> *Regarding formats, we believe that there should be a dedicated API with all data, but the exact format is much less important. But HTML is not a preferred format in our group; we don't want to scrape web pages.*
> *Developer A3*

The reception of this event and the blueprint were 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. One participant from the STA was in charge of compiling a new third-party developer strategy for the STA. After the workshop, he provided feedback by email stating that:

> *This was the best workshop of the year. I felt that that the meeting gave a clearer picture of the needs of the represented developer group and what possibilities their engagement may lead to.*
> *Third-Party Development Strategist, the STA*

1.2. Authentic Platform Development

Given these overall positive signals from workshop participants, the product owner of Trafiklab and I started to work on a suggestion on how to materialize a more authentic beta version. We sent a refined solution blueprint suggestion some two weeks after the workshop to the head of passenger information at the STA (also a workshop participant).

1.2.1. Problem formulation

The blueprint suggested the above-mentioned system, Orion, as the underlying resource. Moreover, Trafiklab.se contained the architectural modules necessary to emulate the desired capabilities. Our suggestion also included a request to engage personnel within the STA to become part of the ADR team that I would lead. Just one day after receiving our offer, she gave the go-ahead to start the design and deployment of a live beta version, alongside access to the required personnel from the STA.

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 true, the need for developers to also be able to get platform access to beyond these common use cases had surfaced. 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 from the API.

1.2.2. 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:

Regarding governance, we found support from the developers in both interviews and the workshop to implement an *access openness* like in the SL case. Thus, we concluded that we could mimic such governance concerning platform access. However, the *coherent search capabilities* were a bit more complex construct. The interviews hinted at the concept of both concerned integration capabilities and extracting the correct data. This finding was corroborated and detailed in the video observations from TravelHack. Besides catering for more general ease of integration, we concluded that the API needed quality-assured "shortcuts" to datasets with high developer demand. Thus, we decided to largely reverse-engineer the current app behaviors and "pirate" API designs and offer these as *interfaces* in beta platform architecture (see **Table 16**).

Coherent search	**TrainInfo API https://api.trafiklab.se/trafikverket/traininfo**
Get all stations	/stations/listAllCurrentlyUsed]?key=[api_key]
Search station by name or coordinates (bounding box)	/stations/stations/search?name=[search_string]&key=[api_key] /stations/stations/search?lat_start=[SWEREFF99_lat]&lat_end=[SWEREFF99_lat]&lon_start=[SWEREFF99_lon]&on_end=[SWEREFF99_lon]&key=[api_key]
Get a given station by name or signature[44]	/stations/stations/[name]/?key=[api_key] /stations/stations/[sign]/]&?key=[api_key]
Get departures from a given station	/stations/stations/[name]/departures/key=[api_key] /stations/stations/[sign]/departures/key=[api_key]
Get arrivals from a given station	/stations/stations/[name]/track/[track_id]/arrivals/key=[api_key] /stations/stations/[sign]/track/[track_id]/arrivals/key=[api_key]
Get all operational tracks at a given station	/stations/stations/[name]/listCurrentUsedTracks?key=[api_key] /stations/stations/[sign]/listCurrentUsedTracks?key=[api_key]
Get departures from a given track	/stations/stations/[name]/track/[track_id]/departures/key=[api_key] /stations/stations/[sign]/track/[track_id]/departures/key=[api_key]
Get arrivals from a given track	/stations/stations/[name]/arrivals/key=[api_key] /stations/stations/[sign]/arrivals/key=[api_key]
Get a given train	/trains/[train_id]/?key=[api_key]

Table 16 - Coherent search implementations

Given the unanticipated developer response on the constraining effect of merely publishing coherent searches, we concluded that the platform also needed some mechanism to channel *all* data to allow

for *flexible searches*. However, based on the unanimous statement from the developers participating in the workshop that they would be pretty content with any format other than HTML, we also hypothesized that such an arrangement could be cruder and, to this end, decided to publish information objects in their original form, channeled through an interface. This change consequently led to a shift in the design framework to include *flexible search* capabilities.

Flexible search	**TrainInfo API** **https://api.trafiklab.se/trafikverket/trainexport**
Get all messages	/messages?key=[api_key] (*retrieves all train traffic messages, regarding, e.g., track work, train disturbances, or facility malfunctions)*
Get all stations	/stations?key=[api_key] *(retrieves all stations in Sweden, including those non-operational)*
Get all traffic information	/traffic?key=[api_key] *(retrieves all current timetable information, e.g., information on trains at traffic junctions (stations, stops). Each item corresponds to a specific train at a specific traffic location).*

Table 17 - Flexible search implementations

In the alpha version workshop, the blueprint was indeed demonstrated to the participants. However, the ADR team saw two reasons to evaluate the most important parts of the foreseen solution further before realizing it further. First, the workshop had not outlined the interfaces in detail, and TrainExport had not been part of the prepared workshop material. Second, only a handful of developers participated, and it was necessary to seek feedback from a

[44] Signature is an institutionalized way of assigning all stations 2-4 letters, used as identifier for that station. For instance, the signature of Stockholm Central Station is **CST**.

wider circle of developers. To this end, the interface specifications were made publicly available on an open internet forum[45] to gather input.

Of the received replies, the feedback was overall positive. There were individual suggestions to use additional technical standards, such as GeoJSON, JSON Schemas, and HTTP caching headers (that the ApiGee platform did not support and thus could not be implemented). Another request, however, appeared twice. This request concerned a task that developers currently struggled with, detecting changes since their last API call. One developer elaborated this request:

> *It would be great if each line could have a timestamp that says when the line was last modified, corresponding to "UpdatedTime" in KartDB.messages. You are often interested in what has happened since the last known time, and today there is no reliable way to make such a selection from Orion. The field must therefore be assigned the current time when the line is created, and then updated each time the line is changed - e.g., when RealTimeArrival / RealTimeDivision updated, new estimated times are entered, status messages change, etc. Hopefully, it's a pretty simple thing to add, and one such field would probably save a lot of bandwidth and server capacity for the STA because it allows developers to download only delta/differences instead of the entire train traffic model at each call.*
> *Theodor Storm*[46]

More specifically, this request concerned adding a timestamp when each data item was updated. This way, developers would only need to retrieve items updated since their last request (or any other arbitrary point in time). Although seemingly simple, the STA was not able to implement this due to underlying architectural constraints. Orion

[45] https://groups.google.com/forum/#!forum/jarnvags-api-trafiklab

[46] This user's statement is not anonymized since it is posted on open discussion group

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.

Given the otherwise generally positive reception, the ADR team started to materialize the architecture of the outlined solution. While the data itself was readily available within the STA, their current systems architecture could not afford to support it within the project's resource boundaries. For this reason, we, as described above, used the architecture of Trafiklab that could host the emulated capabilities.

From STAs system architecture perspective, their architecture was *inverted* through a new module facing application developers. This module was a cloud-based service hosted by ApiGee, a company selling platforms that host and scale APIs. This new module handled access control, caching of data (to relieve the underlying system of redundant queries). In addition, this module provided the two new *interfaces*, TrainInfo and TrainExport, facing third-party developers, yet decoupled from the STAs underlying systems. Based on the functional specification displayed on the open web forum, the ADR team's Orion expert from the STA, together with the Trafiklab architect, crafted a technical specification targeting ApiGee's engineers. This document specified how to extract data corresponding to the coherent and flexible searches, including how the ApiGee interfaces should offer these interfaces as REST APIs (the actual transformation was carried out by ApiGee personnel).

While the use cases were possible to implement, the solution could not provide geographical coordinates in developer-friendly formats but instead used the SWEREF99 grid for geographical positioning. SWEREF99 is an official Swedish positioning system used by national and local authorities and has thus become a de facto standard for publicly administered digital geographic data. While the widespread usage of SWEREF99 among Swedish authorities enables systems operability on a national level, most modern technology platforms instead use the American WGS84 standard for geographical positioning. This fact meant that the beta version's use of SWEREF99 required all developers to resolve this conversion, a non-trivial task.

To this end, we included references to existing conversion code libraries as *integration protocols* that could help resolve this translation.

Overall, the solution would also use other, existing integration protocols of Trafiklab.se to enable third-party development. This infrastructure included an API console at Trafiklab.se (that allowed developers to execute API calls without a development environment) and the user registration functionality (where API keys could be dispensed). Finally, we also created a small tutorial that allowed developers opting for coherent searches to expedite their development process, alongside documentation of the data models.

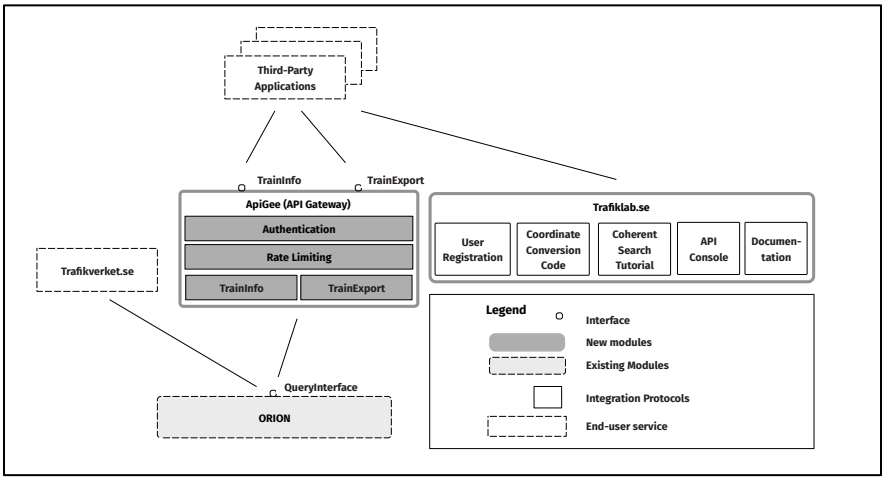


Figure 6 - -The Beta Version Architecture

The solution was officially released in the late autumn of 2012. Anyone could register for the API, and in three months, 59 developers had registered. For evaluation, I 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. Among these, 8 developers had primarily used the TrainInfo interface, 3 had focused on TrainExport, 2 used both, and 4 had registered but not used the APIs to the extent that they could provide evaluation feedback.

Summarizing their impressions, users that had focused on the TrainInfo 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 looked into transitioning their apps data source to TrainInfo. Other than finding a few bugs, they found such a transition straightforward and appreciated the official status of TrainInfo. 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. Regarding negative experiences from using the API, it concerned minor technical aspects, such as fields having fluctuating positions of data points in the resulting data structures (one developer), the effort required to execute the SWEREF99-to-WGS84-conversion mentioned above (one developer), and difficulties understanding how to retrieve the API key (three developers). However, when asked to summarize their overall views from using the API, all users of TrainInfo echoed a pleasant experience:

> *I'm positively surprised; I think TrainInfo works very well; it was straightforward to get started. Two words describe it well, quick and easy…if you only have a little knowledge of the world of APIs and development, the rest will follow quickly.*
> *Developer B4*

> *The road to getting the API to work was very straight. I made a test call from the API console to see how the XML was structured, and then I wrote my API client that called TrainInfo. I didn't even look at documentation until last week; I understood the API anyway. […] I think the developer experience is bang on.*
> *Developer B2*

> *I thought it was fantastic with the examples so that you didn't have to write your API calls directly. You could immediately get data through the console and then convert the coordinates through the PHP examples[47] to build your web pages. I was able to create an API request and get the data really fast.*
> *Developer B7*

> *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*

However, third-party developers that had used TrainExport conveyed a more complex picture. These users (two developers) that had tried TrainExport but did not have any implemented services based on scraping were quite content with the functionality of TrainExport, although they would have preferred the possibility to retrieve smaller batches of data, as were possible with the query language of Orion. However, those users (two developers) that had existing, popular applications based on scraping expressed disappointment and had, for this reason, stuck with unsanctioned data access:

> *I've tried TrainExport, but I have not started to use it. Unfortunately, there's no way to tell what's happened at the last minute, but you need to download the whole batch every time. I would like to see some sort of timestamp, and I am well aware that this isn't possible today; it's just not how Orion works; the STA seems to load everything into their database every minute. It's the most significant disadvantage with TrainExport.*
> *Developer B14*

[47] https://github.com/gnucifer/CoordinateTransformationLibrary

> *We are still using the unofficial API that STA exposed, so we haven't switched to these other TrainInfo and TrainExport. The reason was that we already set up our services to get that data, so we were already kind of tied up towards that API. It would just mean more work to switch.*
> *Developer B3*

1.3. Target Platform Implementation

The first ADR iteration, the beta version, was a large-scale pilot project to inform a potential release version platform design. Although the problems had not been overall resolved, the overall outcome of the trial convinced STA to create a more persistent solution, as described in the official decision by the STAs director of Business Area Society:

> *The Swedish Transport Administration has developed and decided on a strategy for traffic information as well as a strategy for service provision, capacity allocation and pricing within the railway operations, which will provide guidelines on the Swedish Transport Administration's operations in these areas. Regarding travel information based on railway data, demand has increased from market participants who develop services. Today, there are actors who "scrape" information from the Swedish Transport Administration's websites, as it is not available to them in any other way. The Swedish Transport Administration sees a need to provide information via an established interface, also to these actors in order to be able to ensure quality in a better way and to start the development of requested services. Which in turn contributes to the fulfillment of the transport policy goals and to more satisfied customers.*
> *Excerpt from decision signed by the Director of Society, the STA*
> *(Registration number TRV 2012/87434)*

Thus, the STA revised their third-party developer strategy that hitherto had contained three segments, targeting different actors in

the surrounding society. The new, fourth segment was 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*."[48] 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 was still debated within the STA.

1.3.1. 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 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.

The overarching rationales from the previous ADR iteration were intact, yet the beta version results had yielded mixed results. The primary benefit of implementing common use cases had been the enrollment of *new* developers. This way, the solution could expand the number of developers quickly, both regarding minimized platform access negotiation (through online registration and general terms of use) and by lowering the barrier for extra-industry actors by inverting common uses into dedicated REST interfaces.

As a next step, we reformulated the problem. In summary, third-party developers that were new to the railway domain had used the coherent search interface TrainInfo, found it pertinent, and echoed a pleasant experience. However, existing and more seasoned third-party developers that already had implemented services instead

[48] Previously STA had three segments: *Complete* (for rail operators and transport agencies); *Societal* (for society-critical functions); and *Extended* (for larger software houses and information brokers). These segments were more complex regarding both the administrative legislation and the information products.

expressed dislike for the flexible search capabilities. Most had, for this reason, stuck with unsanctioned data access. Second, not only were these developers discontent with TrainExport capabilities *vis-à-vis* what some scraped resources could afford. In addition, these developers also expressed the need for *additional* flexible search benefits to motivate the effort of changing the data source, as commented by one developer during the beta version evaluations:

> *No, 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*

Consequently, we hypothesized that flexible searches also needed to be emulated, not just offered in the beta version.

1.3.2. Building, Intervention, and Evaluation

This somewhat surprising reception by experienced third-party developers instigated a substantial release version platform redesign. Based on the feedback, we decided to implement a query language similar to that of Orion to cater to flexible searches, as this provided more precise flexible searches, as demanded by developers.

Moreover, from the beta version design and onwards, developers' 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. Such change-detections were a challenging task, as explained by one developer:

> *We've spent quite a bit of time on the part where we're detecting differences in the data and pass it on to an internal real-time API that we are then using throughout our service. I guess it's a necessary evil to achieve what we're aiming for.*
> *Developer B3*

However, as explained previously, offering this feature would require a substantial redesign of the underlying system, and implementing this feature had to this point not been considered financially justifiable.

To further investigate whether this feature was necessary, I conducted a data source experiment on apps using SLs real-time data. In September 2013, 19 services for smartphones using real-time data from SL were available in the application marketplaces for Apple iPhone, Google Android, and WindowsPhone. Out of the 19 real-time services, 14 used the official API as the only data source, 2 used both scraping and the Open API, 1 one relied solely on scraping, and one was not possible to determine. The rationale given to use scraping over the Open API was either 1) they had deployed their app before the launch of the open API and did not see enough incentives to move data retrieval to the open API and 2) there was currently data available on the web site missing in the open API (where scraping hence was the only way to get that data). To influence these developers to desert unofficial interfaces, the STA thus decided to implement new functionality that the current solution did not include – the ability to deliver changes since the last 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. First, since the STA now planned to offer its internal (albeit refactored) query language for external developers there were less incentives to encapsulate it behind a software layer offering access to the resource. Second, given the data source experiment, developers at SL brought forward capabilities not available in the official APIs as one reason for continued self-resourcing. Consequently, any deviations between the interfaces offered to third-party developers and for internally developed public application 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, but through the new platform, DataCache, the STA could easily upgrade its own applications while still serving the needs of external third-party developers.

However, this resource openness decision entailed challenges for the platform's architecture. At this point, the ADR team instead decided to substitute and promote functionality that had been residing in Orion. This way, both the STA and third-party developers would use the new platform to construct new end-user services (see **Figure 8**). 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 (see **Figure 7**).

```
<ORIONML version="1.0">
  <REQUEST plugin="WOW" version=""
                locale="SE_sv"
                authenticationkey="{apikey}">
       <PLUGINML
           table="LpvTrafiklagen"
           filter="TrafikplatsNamn = 'Borlänge C'
                       AND ((AnnonseradTidpunktAnkomst >
                           datetime('now','localtime','-15
                       minute') AND (datetime('now','+24
                           hour') >
                       AnnonseradTidpunktAnkomst) OR
                       BeraknadTidpunktAnkomst >
                       datetime('now','localtime')) AND
                       VisaAvgangVidStationSokning =
                       true)"
           orderby="AnnonseradTidpunktAvgang"

            selectcolumns="TrafikInfoAgareNamn,TrafikInfoAgar
           eUrl,TrafikInfoAgareMobilUrl,Fran,Till,StatiskInfo
           rmationTrafikplatsVisning,StatiskInformationTagVis
           ning,InstalldAvgang,AnnonseradTidpunktAvgang,Verkl
           igTidpunktAvgang,BeraknadTidpunktAvgang,Sparangive
           lseAvgang,AnmarkningarAvgang,ArAvgangTag,Annonsera
           tTagId" limit="50" />
  </REQUEST>
</ORIONML>
```

```
<REQUEST>
  <LOGIN authenticationkey="{apikey}" />
  <QUERY objecttype="TrainAnnouncement"
        orderby="AdvertisedTimeAtLocation"
        limit="50">
    <FILTER>
      <AND>
        <EQ name="ActivityType" value="Avgang" />
        <EQ name="FromLocation.LocationName"
                  value="Borlänge C" />
        <OR>
          <AND>
            <GT name="AdvertisedTimeAtLocation"
                        value="$dateadd(-00:15:00)" />
            <LT name="AdvertisedTimeAtLocation"
                        value="$dateadd(14:00:00)" />
          </AND>
          <AND>
            <LT name="AdvertisedTimeAtLocation"
                        value="$dateadd(00:30:00)" />
            <GT name="EstimatedTimeAtLocation"
                        value="$dateadd(-00:15:00)" />
          </AND>
        </OR>
      </AND>
    </FILTER>
    <INCLUDE>Operator</INCLUDE>
    <INCLUDE>WebLink</INCLUDE>
    <INCLUDE>FromLocation.LocationName</INCLUDE>
    <INCLUDE>ToLocation.LocationName</INCLUDE>
    <INCLUDE>OtherInformation.Description</INCLUDE>
    <INCLUDE>Canceled</INCLUDE>
    <INCLUDE>AdvertisedTimeAtLocation</INCLUDE>
    <INCLUDE>TimeAtLocation</INCLUDE>
    <INCLUDE>EstimatedTimeAtLocation</INCLUDE>
    <INCLUDE>TrackAtLocation</INCLUDE>
    <INCLUDE>Deviation.Description</INCLUDE>
    <INCLUDE>ActivityType</INCLUDE>
    <INCLUDE>AdvertisedTrainIdent</INCLUDE>
  </QUERY>
</REQUEST>
```

Figure 7 - Query language and data model example in Orion (left) and DataCache (right)

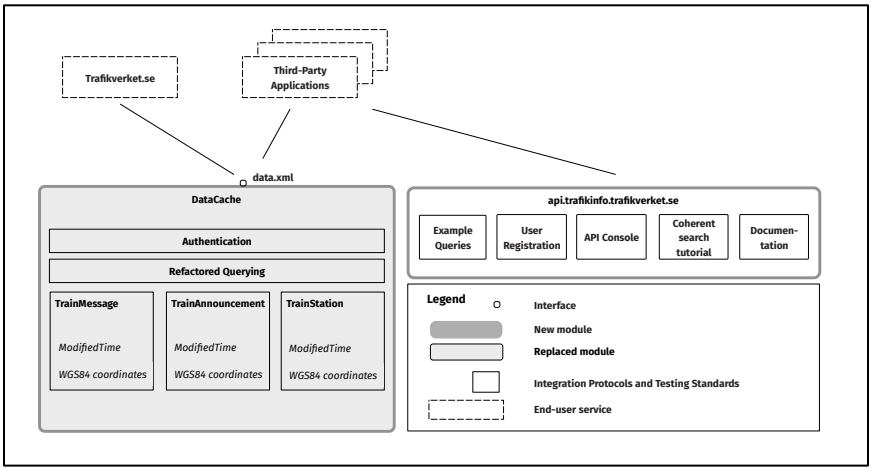


Figure 8 - The Release Version Architecture

The solution comprised the following constituents:

A query *interface* (data.xml) – where developers could construct their own data retrieval composition (right-hand side in **Figure 7** above) based on three underlying information objects:

TrainMessage – Announcements around track works, track and train dysfunctions, and other types of disturbances.

TrainAnnouncement – real-time train information, i.e., information about train traffic locations (stations, stops).

TrainStation – train station information including name, its location's geographic coordinates, and whether passengers board trains at that station

The query interface required an authentication token, what information objects and fields the user intended to query, and optional selection criteria (such as a given station) (see Figure 6). Moreover, all these information objects included the field *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. This way, the tedious work of sorting out changes to real-time information was resolved. Finally, the information objects now included the WGS84 geographic

coordinate system, effectively scrapping developers' design tasks to perform the conversion between SWREF99 and WGS84.

The query interface at data.xml was a non-deterministic query language and thus inherently supported only flexible searches. Hence, we concluded, it was no longer possible to use the interface level for coherent searches (unless introducing new interfaces, a solution that the STA rejected, for system maintenance reasons). 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 (see Figure 9). 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 positive reception from developers regarding the API console, documentation, and tutorial/example API calls, we also implemented those as integration protocols.

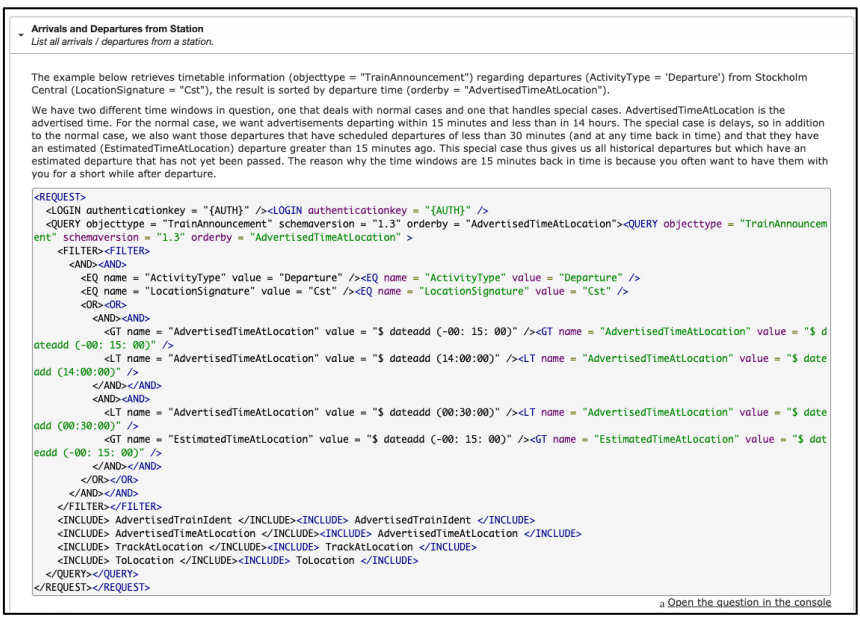


Figure 9 - Coherent search through integration protocols

While the coherent search implementation had been successful in the beta version platform, we saw a need to validate the new

implementation (using example queries). To this end, a more controlled test with novice users was conducted. Here, university students were given a set of tasks to complete where they needed to reuse the coherent searches to accomplish the tasks. These students provided generous feedback on improvement opportunities (such as more informative names of the data model elements and example response, not just queries). A core signal from this test was that 10 out of the 13 students were able to perform the tasks (such as getting the train departures from a specific station) with the queries' help. Since these students' application development experience was lower (according to the background information they provided) compared to the target group, we concluded that the coherent search solution would suffice.

The platform was pre-launched on February 10, 2014, as an open test environment. This launching meant that any registered user at Trafiklab could use the API if they applied for access by email. During this test period, 20 developers registered (among them several of the existing railway data developers), and 6 of these agreed to be interviewed. Based on this feedback, the platform went live on March 18. In August 2014, I interviewed another 6 developers that had registered as users of the platform.

Among these, 9 had used the beta version solution from the previous ADR iteration, and all but one preferred the release version design. This preference was primarily due to more flexible ways of retrieving data (six developers), improved response times (four developers), and the possibility of using the ModifiedTime functionality (two developers), as expanded by one developer that tried both solutions:

> *Understanding the information in the old TrainExport was difficult at first, but in the new API, I must stress that it has been an incredible improvement, and it's just fantastic to use this test console and try out different queries and execute them…it has eased development amazingly much. So, I spent more time figuring out what data to use than writing code to fetch it. […] It has been very, very neat compared to other APIs I have been working with.*
> ***In what way has it been standing out compared to other APIs?***
> *Well, first of all, that there is a console where you can test queries quickly, and even more important are the example queries, it is documented, yes all these simple things, it returns simple responses, JSON objects, I've worked so much with these weary, clunky SOAP APIs*
> *Developer R3*

The developer that preferred the TrainInfo API motivated this preference by preferring domain-agnostic identifiers (GUIDs) for stations in TrainInfo but not in DataCache, and that those REST calls were more straightforward to construct than the XML queries of DataCache.

8 developers had existing services consuming railway data. Among these, none said they would continue to scrape, given the release version, simply because there now were no real benefits of retrieving in another way. One developer expanded on this matter:

> *Especially, I found it very positive that I could choose exactly what data I wanted…which is one of the reasons that I had a web service that sat and collected all data from Orion and then just sent out the data I needed to the client. Because even though I had some middleware, it was still faster to download all the data to my mobile phone at once. But now, with the new APIs, the data will go directly to the mobile phone and make it go even faster.*
> *Developer R7*

The 4 developers that did not have existing services but developed from scratch were also overall content with their experience, as explained by one novel developer:

> *I think it went very smooth, I have started developing from those parameters that exist, and I have used quite basic parameters. I haven't done anything advanced, so I don't think it has been very difficult. I think it has been much easier than I initially thought. I haven't thought much about it because all the things I have tried to do I have been able to do because I have always been able to go back to the examples because everything I have been trying to do had examples. I have always been able to go back there and see" OK, how do I do this" since the things I was doing weren't advanced. It was very easy to limit my selection, search for a county, municipality, a particular train stop or station, so it was very neat.*
> *Developer R9*

Among the issues found across interviewed developers, the most common objection (eleven developers) was that trains' real-time positioning had deteriorated (or was missing, in case they hadn't tried the beta version). In the previous beta version APIs, developers were able to get the latest passage points, not only where the trains stopped for passenger exchange but also closed stations and other official passage points along the route. However, for security reasons, this information had been removed from the release version API, making it more difficult for developers to, e.g., create maps plotting train movements[49]. In addition to this comment, some mentioned integration possibilities with related datasets (i.e., the use of the identifiers) (four developers), introducing technical identifiers of trains (two developers), learning barriers of the query language (two developers), difficulties specifying correct HTTP request header information (one developer), and what exact unit the term "radius" was referring to (when doing geographical searches) (one developer).

[49] This information was re-introduced into the DataCache data model in December 2015, due to multiple developer requests.

However, when developers were asked to summarize their experience of the APIs, they were all positive:

> *Easy to get started, easy to understand the syntax, easy to register, and get API keys, I simply give it a high rating.*
> *Developer R1*

> *Well worked through, good data model, easy to get started, very complete when it comes to the public information. It lacks a bit when it comes to the integration possibilities since there is very little underlying technical information to tie the data towards other services—something like that.*
> *Developer R6*

> *Definitely a nice surprise. It wasn't, how do you put it, it wasn't my perception of what the STA was doing. So that it actually exists made very pleasantly surprised. The API meets my needs.*
> *Developer R10*

> *Very good. I wished all agencies did it this way; it has been good working with it. I hope that more agencies see what the STA is doing and that more will follow their example.*
> *Developer R3*

All the respondents also stated that they would recommend this API to other developers interested in developing railway services.

1.4. Ensemble Platform Manifestation

The API platform persevered long after its inception and is at the time of writing (2021-03-18) 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.

1.4.1. Third-party developer adoption

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 on Trafiklab: by intercepting the API calls. In case the data source was unable to determine, the developers were contacted to inquire about the data source. The investigation revealed that development towards unsanctioned interfaces was virtually extinct. 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 (a system called UT/IN), and 3 were not functioning (where it seemed as if the application was no longer maintained).

Name	Retrieved	Platform	Data source
Info Tracker	2016-08-16	iOS	Open API*
Pend.la	2016-08-16	iOS	Open API*
Railor	2016-08-16	iOS	Open API
Tåg	2016-08-16	iOS	Open API
Nästa Avgång	2016-08-16	iOS	Open API
Pendelprognos	2016-08-16	iOS	UT/IN*
pendla - reshjälp för pendlare	2016-08-16	iOS	Open API
SJ	2016-08-16	iOS	UT/IN*
Stationen Plus	2016-08-16	iOS	Did not work
SJ Labs	2016-08-16	iOS	UT/IN*
Tågkoll	2016-08-16	iOS	Open API
Tågtavlan	2016-08-16	iOS	Open API*
Tågtavlan	2016-08-16	Android	Open API
Tågtavlan v2	2016-08-16	Android	Open API*
Tågtider	2016-08-16	Android	Open API*
Tåginfo Sverige	2016-08-16	Android	Open API
Railor	2016-08-16	Android	Open API
SJ	2016-08-16	Android	UT/IN*
Tågkompaniet	2016-08-16	Android	UT/IN*
Tåghjälpen	2016-08-16	Android	Did not work
Pendelkollen	2016-08-16	Android	Open API
PendelPal Sverige	2016-08-16	Android	Open API
SJ Labs	2016-08-16	Android	UT/IN*
STHLM Traveling	2016-08-17	Android	Open API*
Travelplan Sweden	2016-08-16	WindowsPhone	Open API
Tågtrafik	2016-08-16	WindowsPhone	Open API
Tågtid	2016-08-16	WindowsPhone	Did not work
Tågläget	2016-09-13	WindowsPhone	Open API
* Interview response since the data source was unable to determine from API request interception			

Table 18 - Smartphone apps and their data sources

Moreover, usage statistics from the platform showed that not only existing developers seemed to have adopted the API. These statistics conveyed that new developer registrations had persevered since the launch (**Table 19**) and the platform had in August 2020 5727 registered developers. Moreover, more detailed usage statics conveyed that external API calls had increased over time, from some 20 million in early 2016 to some 100 million per month in 2020 (**Table 20**). As such, external clients are now generating more calls than internal clients.

Period	**Number of new developer registrations**
2014-02 – 2014-12	338
2015-01 – 2015-12	422
2016-01 – 2016-12	639
2017-01 – 2017-12	702
2018-01 – 2018-12	1466
2019-01 – 2019-12	1377
2020-01 – 2020-08	783

Table 19 - Number of new developer registrations per year

Period	**External calls (avg millions/month)**	**Internal calls (avg millions/month)**
201601 – 201612	22,2	78,7
201701 – 201712	41,1	95,6
201801 – 201812	69,7	83,5
201901 – 201912	90,7	63,2
202001 – 202008	100,5	63,2

Table 20 - Eternal and Internal DataCache API calls

1.4.2. Continued Emulation Activities

Although the platform initially was designated for railway data, it was not long after the DataCache launch until the STA decided that the platform also should host road data:

> *At the time when DataCache was launched, a project on how to publish road data to third parties was undertaken in parallel. The report from this work described that there was currently little developer activity on road data and suggested to also include road data in the DataCache platform. This was to lower the barriers since there was no easily available API for road data. DATEX II existed, but it required signed agreements, and the data model was too complex for any external hacker.*
> *Systems Owner, DataCache Platform*

In mid-2014, the DATEX II feed was serving a total of 91 clients. While some of these (9) were private individuals, the vast majority were corporations or universities/research institutes. The STA hypothesized that the current DATEX II feed suffered from several of the deficits that the undertaken emulation approach had discovered for railway data. These shortcomings included

4. the need for greater autonomy – as the onboarding routine for DATEX II required a written agreement
5. A way to minimize the work of discovering relevant data – by making the DATEX II data model less complex
6. Find ways to decrease the work necessary to integrate with the data – currently, the developer needed to set up his/her polling server, and figure out how to extract relevant data, e.g., roadworks in a certain geographical area – a non-trivial task.

These hypothesized deficits would be addressed by streamlining the data models for road data (as has been done for railway data) and plugging the road data systems into DataCache. In January 2015, the STA consequently launched road data into DataCache.

Besides road activity, the STA continuously incorporated 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 projects. The systems architect of DataCache commented on this:

> *We are always refining the data model to make it more useable, and the input can be something from a forum or bulletin board: "How do we know how to sort ViaToLocationName?" Well, that is not trivial to figure out, so we'll add it!*
> *Systems Architect, DataCache*

```
<ViaToLocation>
      <LocationName>Gms</LocationName>
      <Priority>4</Priority>
      <Order>0</Order>
</ViaToLocation>
<ViaToLocation>
      <LocationName>Ml</LocationName>
      <Priority>2</Priority>
      <Order>1</Order>
</ViaToLocation>
<ViaToLocation>
      <LocationName>Ök</LocationName>
      <Priority>3</Priority>
      <Order>2</Order>
</ViaToLocation>
<ViaToLocation>
      <LocationName>Thn</LocationName>
      <Priority>1</Priority>
      <Order>3</Order>
</ViaToLocation>
```

Figure 10 - Explicit Ordering of ViaToLocation

Besides the platform's continued evolution, an important issue unresolved in the ADR project concerned the developers' assurances that the platform would be up and running. The platform has since its inception remained open, meaning that the STA had access to the same resources to develop public end-user services in DataCache as third-party developers. This strategy turned out to be very useful regarding the service-level agreement (SLA). SLAs typically regulate the platform owner's responsibility to maintain a certain level of service availability and quality of the service. The client pays a

predetermined price for such a service level, and in the case, the platform provider fails to meet the service levels, the platform owner compensates the client through indemnities. However, when the platform is open, the client (third-party developers) pays no access fees, and thus compensations become irrational. Therefore, there were internal discussions on how to assure third-party developers that the service would have a sufficient service level for external innovators to invest in production-ready services. One possibility was setting up separate and optional SLAs for a fee:

> *We were not sure about what level of quality that was necessary, what requirements third parties have. If they are dependent on 100% levels, then it's going to start to cost a lot, and we may have to start charging for the information if it's stricter requirements than what we need for ourselves.*
> *Traffic Information Strategist, the STA*

However, according to the STA, these risks never effectuated. The reason why the open DataCache platform was able to satisfy developers' demands for availability and quality was that third-party developers were able to enjoy a "shadow SLA" of STAs services. Since the services provided by the STA were mission-critical, they were secured under an SLA between the STA and their systems suppliers.

> *This thing with SLA was such a hot topic when we started that you needed an SLA for "What do you promise?", "What is your availability?" and those types of questions. We said this is based on our best effort. But we use the exact same platform for our own services, and we used this fact as a reputation capital, of sorts. As it turns out, this model works very well. The APIs are practically uninterrupted. We measure uptime and quality through an external service, and it's incredibly high. And then, of course, this issue fades away…*
> *System Manager, DataCache*

1.4.3. Organizational reception

The final last aspect that surfaced in the follow-up study concerned how the emulated platform was embraced 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.

> *And from that project onwards, we have been using this platform that was shaped during the ADR project as a general component internally for the integration of all kinds of stuff. For instance, we integrate between the traffic information systems at the STA, sharing data to enable internal systems to consume train timetables and whatnot. So, the platform has grown into something else, and there are several instances deployed nowadays, using the exact same code base as the original platform.*
> *System Manager, DataCache*

After this first usage, the platform had continuously grown in popularity, as elaborated by the systems architect of DataCache:

> *People in the corridors are saying, "We heard rumors about this API platform," and then "we would also like to use it." So, it's all based on mouth-to-mouth, something like" This is something good, this is something we'd like to use." And then, if anyone has the need to publish data in any way, we can say to them, "we'll solve all your problems." They get so incredibly much free. They come to us and say, "Here's our data," then we do our magic, and all of a sudden, they have a service up and running in a couple of days that they previously estimated would take half a year to build.*
> *System Manager, DataCache*

When asked what differs in the DataCache team compared to other groups within the STA, the systems architect argued that they had a sharper user focus than what is prevalent within technology products such as integration platforms:

> *I believe what sets us apart from many other teams internally at the STA, but also in many other organizations, is that we really put the user into focus. Typically, everything is so technically oriented, especially in internal projects. But once you step into our door and say," I have a need to publish data," we answer, "well, is it really you that have this need? Isn't someone out there who have this need, those who wanted the data, they have the need!" And then they come dragging with an extremely complicated data model with obscure coordinates, strange field names, and whatnot, and then we say, "We cannot publish this; no one will ever be able to use it." We must start by knocking together an understandable data model, and once that is done, we say, "And now it's time to document it," and they will say, "What – must we also document this?" and we say "Yes." So, we are a little tough on our internal clients, but that is all to provide a great experience for those developers who will be using it.*
> *Systems Architect, DataCache*

These new, internal instances contained the same functionality, with a test console, API documentation, some examples, and required even 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:

I would say the ADR project is 100% of the explanation. The reason the platform turned out so great is that we learned how always to have the user in focus. Usually, you do something for your colleague; in the next room, you typically just meet that need. You get it to work, and when another need shows up, you'll just mend something different for that. When we designed the open API platform, we collaborated closely within the ADR team. "How do we get in touch with the third-party developers? How can we make things easier for them?" This collaboration gave us a general solution that works really well. And that is the reason why we are using it more and more internally, is because the interfaces are so good and easy to use.
Systems Architect, DataCache

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 decided in favor of this investigation thus making DataCache the official integration platform of the STA.

1.4.4. DataCache influence on the Swedish Public Transport Industry

Learnings from the STAs SLAs became increasingly important within the Swedish public transport industry from 2016 and onwards. In 2016, through the "Forum for Transport Innovation," the Government Offices of Sweden ignited a redesign of open public transport data in Sweden called "mobilization of open traffic data." This initiative's primary reason was to create a more comprehensive and harmonized open data delivery from the public transport industry. For instance, real-time data were only available in a few regions, and the datasets from different jurisdictions were challenging to combine. From a policy perspective, more comprehensive data from the public transport industry was a necessity to enable new mobility services.

Therefore, the project's goal was to develop five strategic objectives anchored in the industry, summarizing the principles of essential facets of industry-wide publication of open data. One of these objectives was directly connected to the principles of the DataCache platform at the STA.

Here, the new industry platform should be used by third-party developers and the industry itself. The new open platform should be used to scrap existing integrations between public transport authorities and, the Swedish public transport industry should use the new platform in their end-user services. This principle was referred to as "eat your own dog food." During the project, one out of six workshops were dedicated to "customer value propositions," where aspects such as service levels and third-party developer assurance were handled. Here, the model chosen by the STA was brought forward as an alternative way of circumventing the public transport industry to sign legally binding SLAs. If the public transport themselves would use the new national public transport open data platform as their primary resource, third-party developers could rely on a sufficient "shadow SLA." In the final document that was ratified by the Swedish public transport industry, the principle of "eat your own dogfood" was highlighted[50].

[50] See https://samtrafiken.se/wp-content/uploads/2017/04/Slutrapport-_-Kraftsamling-%C3%96ppna-Trafikdata-en-m%C3%A5lbild-f%C3%B6r-Sverige-v-1.0-_-Diarienummer-Vinnova-2016-03467.pdf#page=51

Gothenburg Studies in Informatics

ISSN 1400-741X (print), ISSN 1651-8225 (online)

1. Ulf Sundin. A Logic Programming Approach to Information Modelling and Database Design, May 1990. (Licentiate Thesis).

2. Thanos Magolas and Kalevi Pessi. En studie om informationssystems- arkitekturer (in Swedish), February 1991. (Licentiate Thesis).

3. Peter Nordenstam. Individbaserade relativt öppna informationssystem (in Swedish), February, 1990. (Licentiate Thesis).

4. Bo Dahlbom and Lars Mathiassen. Struggling with quality; The Philosophy of Developing Computer Systems, August 1991. (Revised edition: Computers in Context. The Philosophy and Practice of Systems Design, Oxford: Blackwell, 1993.)

5. Börje Langefors. Essays on infology. Summing up and Planning for the Future, Edited by Bo Dahlbom, August 1993.

6. Bo Dahlbom (ed.). The Infological Equation. Essays in honor of Börje Langefors, March 1995.

7. Bo Dahlbom, Frederik Kämmerer, Fredrik Ljungberg, Jan Stage and Carsten Sørensen (eds.). Designing in Context. Proceedings of the 18th Information Systems Research Seminar in Scandinavia, June 1995.

8. Bo Dahlbom, Fredrik Ljungberg, Urban Nuldén, Kai Simon, Jan Stage and Carsten Sørensen (eds.). The Future. Proceedings of the 19th Information Systems Research Seminar in Scandinavia, June 1996.

9. Agneta Ranerup. Användarmedverkan med representanter (in Swedish), August 1996. (Doctoral Thesis).

10. Ole Hanseth. Information Technology as Infrastructure, November 1996. (Doctoral Thesis).

11. Fredrik Ljungberg. Networking, September 1997. (Doctoral Thesis).

12. Jan Ljungberg. From Workflow to Conversation, October 1997. (Doctoral Thesis).

13. Thanos Magoulas and Kalevi Pessi. Strategisk IT-management (in Swedish), March 1998. (Doctoral Thesis).

14. Fredrik Ljungberg (ed.). Informatics in the Next Millennium. Essays in honor of Bo Dahlbom, June 1999.

15. Urban Nuldén. e-ducation, May 1999. (Doctoral Thesis).

16. Lars Erik Holmquist. Breaking the Screen Barrier, May 2000. (Doctoral Thesis).

17. Nina Lundberg. IT in Healthcare - Artifacts, Infrastructures and Medical Practices, May 2000. (Doctoral Thesis).

18. Henrik Fagrell. Mobile Knowledge, October 2000. (Doctoral Thesis).

19. Staffan Björk. Flip Zooming - The Development of an Information Visualization Technique, October 2000. (Doctoral Thesis).

20. Johan Redström. Designing Everyday Computational Things, May 2001. (Doctoral Thesis).

21. Dick Stenmark. Designing the new Intranet, March 2002. (Doctoral Thesis).

22. Pouya Pourkomeylian. Software Practice Improvement, March 2002. (Doctoral Thesis).

23. Rikard Lindgren. Competence Systems, June 2002. (Doctoral Thesis).

24. Ulrika Lundh Snis. Codifying Knowledge, October 2002. (Doctoral Thesis).

25. Lars Svensson. Communities of Distance Education, December 2002. (Doctoral Thesis).

26. Kai Simon. BPR in the Pharmaceutical Industry, April 2003. (Doctoral Thesis).

27. Per Dahlberg. Local Mobility, May 2003. (Doctoral Thesis).

28. Alexandra Weilenmann. Doing Mobility, June 2003. (Doctoral Thesis).

29. Carina Ihlström. The Evolution of a New(s) Genre, September 2004. (Doctoral Thesis).

30. Antonio Cordella. Information Infrastructures in Action, November 2004. (Doctoral Thesis).

31. Helena Holmström. Community-Based Customer Involvement for Improving Packaged Software Development, November 2004. (Doctoral Thesis).

32. Christian Hardless. Designing Competence Development Systems, March 2005. (Doctoral Thesis).

33. Andreas Nilsson. Sport Informatics – Exploring IT Support for Spectators at Sporting Events, November 2005. (Doctoral Thesis).

34. Johan Lundin. Talking about Work – Designing Information Technology for Learning in Interaction, November 2005. (Doctoral Thesis).

35. Agneta Nilsson. Contextual Implementation of Organizational Networking Systems, August 2006. (Doctoral Thesis).

36. Mathias Klang. Disruptive Technology – Effects of Technology Regulation on Democracy, October 2006. (Doctoral Thesis).

37. Ulrika Josefsson. Coping Online – Patients' Use of the Internet, February 2007. (Doctoral Thesis).

38. Magnus Holmqvist. Developing And Implementing IS/IT in Aftermarket Logistics, June 2007. (Doctoral Thesis).

39. Jonas Landgren. Designing information Technology For Emergency Response, September 2007. (Doctoral Thesis).

40. Magnus Andersson. Heterogeneous IT Innovation. Developing industrial architectural knowledge, October 2007. (Doctoral Thesis).

41. Nicklas Lundblad. Law in a Noise Society, February 2008. (Doctoral Thesis).

42. Maria Åkesson. Digital Innovation in the value networks of newspapers, September 2009. (Doctoral Thesis).

43. Marie Eneman. Developing Effective Child Protection Strategies: Critical Study of Offenders' Use of Information Technology for Sexual Exploitation of Children, December 2010. (Doctoral Thesis).

44. Elisabeth Frisk. Evaluating as Designing, March 2011. (Doctoral Thesis).

45. Ann Svensson. Kunskapsintegrering med informationssystem i professions orienterade praktiker (cover paper in Swedish), May 2012. (Doctoral Thesis).

46. Maria Bolin. Narrativer i förändringsarbete – från projekt till Athenas plan. September 2014. (Doctoral Thesis).

47. Tomas Lindroth. Being Multisituated – Characterizing Laptoping in Networked Situations, April 2015. (Doctoral Thesis).

48. Wanda Presthus. Business Intelligence Utilisation through Bootstrapping and Adaptation, September 2015. (Doctoral Thesis).

49. Jesper Lund. Digital Innovation: Orchestrating Network Activities. September 2015. (Doctoral Thesis).

50. Soumitra Chowdhury. Service Logic in Digitalized Product Platforms – A Study of digital service innovation in the Vehicle Industry. September 2015. (Doctoral Thesis).

51. Asif Akram. Value Network Transformation – Digital Service Innovation in the Vehicle Industry. January 2016. (Doctoral thesis).

52. Fatemeh Saadatmand. Shared Platform Coopetition: The Paradoxical Tension between Stabilized Cooperation and Intensified Competition. November 2016. (Licentiate thesis).

53. Fatemeh Saadatmand. Shared Platform Evolution: An Imbrication Analysis of Coopetition and Architecture. March 2018. (Doctoral Thesis).

54. Fahd Omair Zaffar. The Value of Social Media – What Social Networking Sites afford organizations. June 2018. (Doctoral Thesis).

55. Stefan Nilsson. Designing for technology-mediated collaboration. December 2018. (Doctoral Thesis).

56. Taline Jadaan. The Emergence of Digital Institutions. Oktober 2019. (Doctoral Thesis).

57. Hannes Göbel. Designing Digital Resourcing. Januari 2020. (Doctoral Thesis).

58. Hawa Nyende. Maternal Healthcare in Low-Resource Settings: Investigations of IT as a resource. June 2020. (Doctoral Thesis).

59. Michael Kizito. Enacting ambidextrous IT governance in healthcare. June 2020. (Doctoral Thesis).

60. Daniel Rudmark. Designing Platform Emulation. June 2021. (Doctoral Thesis).