

The African Journal of Information Systems

Volume 10

Issue 4 *Special Issue: Information Technology and the African Networked Society.*

Article 6


September 2018

Cultivating Third Party Development in Platform-centric Software Ecosystems: Extended Boundary Resources Model

Brown C. Msiska

University of Oslo, Department of Informatics; University of Malawi, Department of Computer Science, bmsiska@gmail.com

Follow this and additional works at: <https://digitalcommons.kennesaw.edu/ajis>

 Part of the [Computer and Systems Architecture Commons](#), [Software Engineering Commons](#), and the [Systems Architecture Commons](#)

Recommended Citation

Msiska, Brown C. (2018) "Cultivating Third Party Development in Platform-centric Software Ecosystems: Extended Boundary Resources Model," *The African Journal of Information Systems*: Vol. 10 : Iss. 4 , Article 6.
Available at: <https://digitalcommons.kennesaw.edu/ajis/vol10/iss4/6>

This Article is brought to you for free and open access by DigitalCommons@Kennesaw State University. It has been accepted for inclusion in The African Journal of Information Systems by an authorized editor of DigitalCommons@Kennesaw State University. For more information, please contact digitalcommons@kennesaw.edu.



Coles College of Business



Cultivating Third Party Development in Platform-centric Software Ecosystems: Extended Boundary Resources Model

Research Paper – Special Issue
Volume 10, Issue 4, October 2018, ISSN 1936-0282

Brown Msiska
University of Oslo, University of Malawi
bmsiska@gmail.com

(Received January 2018, Accepted May 2018)

ABSTRACT

Software ecosystems provide an effective way through which software solutions can be constructed by composing software components, typically applications, developed by internal and external developers on top of a software platform. Third party development increases the potential of a software ecosystem to effectively and quickly respond to context-specific software requirements. The boundary resources model gives a theoretical account for cultivation of third party development premised on the role of platform boundary resources such as application programming interfaces (API). However, from a longitudinal case study of the DHIS2 software ecosystem, this paper observes that no matter how good the boundary resources a software ecosystem provides, third party development remains a mere possibility until there exists adequate external generative capacity. Taking into consideration this observation, this paper contributes by extending the boundary resources model to foreground external generative capacity alongside boundary resources as factors that influence third party development.

Keywords

Software Platforms, Software Ecosystems, Third Party Development, Boundary Resources, Generative Capacity, Developing Countries

INTRODUCTION

Software ecosystems provide an effective way through which software solutions can be constructed by composing software components, typically applications, developed by internal and external actors on top of a software platform (Manikas and Hansen, 2013). By leveraging an existing software platform alongside applications specific to it, the time, cost and risk of implementing a software solution can be reduced. A software ecosystem consists of a software platform, a set of applications specific to the platform, a set of internal and external developers (also referred to as *third party developers*), a community of domain experts, and a community of end users whose information needs are met by the

software platform and associated applications (Bosch and Bosch-Sijtsema, 2010). Software ecosystems exist within a larger competitive environment, often competing with other rival ecosystems for end users as well as third party developers (Tiwana, 2013). Google's Android and Apple's iOS ecosystems are popular examples in this regard.

Core to a platform-centric software ecosystem is a software platform, a software system with an extensible codebase that provides core functionality shared by applications that interoperate with it, interfaces through which they interoperate, and resources with which derivative applications can be created (Eck et al., 2015; Ghazawneh and Henfridsson, 2013). The value of a software ecosystem to its end users is partly established by the range of applications specific to its platform. Applications, or apps, are add-on software subsystems that connect to the software platform and add functionality to it (Tiwana, 2013; Tiwana et al., 2010). Applications are a manifestation of "innovation on" (Grisot et al., 2014) whereby innovations are built on top of and without disrupting an underlying artefact.

Despite its benefits, the software ecosystem approach is not one without challenges. Challenges emanate from having a heterogeneous community of end users that are remote from platform developers. In this regard, one challenge faced by platform owners is lack of familiarity with the end user context which makes it difficult for them to make informed decisions on what services or applications to develop for the platform (Henfridsson and Lindgren, 2010). Another challenge is how to effectively respond to turbulent and often diverging demands for innovation from the end users (Ghazawneh and Henfridsson, 2013). Consequently, third party development which involves devolving the development of applications on top of a software platform to external developers in or close to the end user community is increasingly attractive for software platform owners (Ghazawneh and Henfridsson, 2013) as it allows them to focus on the platform core and defer satisfying specific end user requirements to third party developers. Familiarity and proximity to the end user context enables third party developers to make informed decisions on applications to develop in response to specific demands for innovation among end users.

Failure or delay responding to needs in the end user community can diminish the value of the software platform among its end users and give room to rival software ecosystems. The demise of once dominant Symbian and Blackberry ecosystems and rise of iOS and Android ecosystems is a good example in this regard. For platform owners, third party development is attractive because it increases the potential of a software ecosystem to effectively and quickly respond to specific end user needs. Third party development also works in favor of the end user community as it brings innovation close to the context of use. The proximity of third party developers to end users brings about agility in responding to innovation demands. As a result, enabling third party development is not only in the interest of platform owners but platform adopters as well.

Furthermore, third party development can accelerate innovation in a software ecosystem (Bosch, 2009) and be the basis for market leadership (Ghazawneh and Henfridsson, 2013). For example, in the year 2012 Apple's iOS ecosystem had about 200,000 external developers through which it was able to muster enough innovative third-party applications to usurp market leadership from the Blackberry ecosystem which only had 8000 external developers (Tiwana, 2013). The wealth of derivative applications needed for a software ecosystem to stay competitive is difficult to accomplish using in-house developers alone. Thus, to successfully build platform-centric software ecosystems third party development must be cultivated and this entails shifting design capability to external actors (Prügl and Schreier, 2006; von Hippel and Katz, 2002). This shift is necessary to build or enhance the generative capacity (Avital and Te'eni, 2009) of external developers.

This paper is based on a case study of the *District Health Information Software version 2* (DHIS2) software ecosystem. DHIS2 is an open source software platform that enables governments and organizations to collect, manage and analyses data in the health domain and beyond. It is currently in use by ministries of health in more than 60 countries. The DHIS2 platform is developed under the Health Information Systems Program (HISP) at University of Oslo in Norway. DHIS2 supports third party development by providing an application programming interface (API) with which third party applications can be constructed. Despite the provision of the API and associated platform boundary resources to facilitate third party development, the quality and number of third party applications in DHIS2 app store is still limited (Polak, 2015). A large part of DHIS2's end user community is in developing countries and the shortage of ICT skills in such countries are well documented (Kimaro, 2006; Kimaro and Nhampossa, 2005; Mutula and Van Brakel, 2007). The dearth of generative capacity in developing countries leaves a majority of DHIS2 end users dependent on core developers for the implementation of desired applications. This creates an innovation bottleneck which often results in failure or delays in responding to specific end user needs.

Coming from this background, this paper uses the DHIS2 software ecosystem as a case to address the question: how can third party development in a platform-centric software ecosystem be cultivated? The boundary resources model (Ghazawneh and Henfridsson, 2013) provides a theoretical account for cultivating third party development premised on the role played by boundary resources in enabling third party development. However, as observed in the DHIS2 software ecosystem, providing platform boundary resources alone is not enough as a catalyst for third party development. Besides platform boundary resources, third party development also depends on the generative capacity of external developers. With this in consideration, this paper extends the boundary resources model to foreground external generative capacity alongside boundary resources as factors that have influence on third party development.

BOUNDARY RESOURCES MODEL

Platform boundary resources are software tools and regulations that serve as the interface between platform owners and third-party developers facilitating the development of third party applications (Ghazawneh and Henfridsson, 2013). Such boundary resources typically consist of software development kits (SDKs) and application programming interfaces (APIs). Platform boundary resources have the power to stimulate or constrain generativity in a software ecosystem (*ibid.*). The boundary resources model, illustrated in Figure 1, is a set of related constructs that provides an intellectual structure with which the role of platform boundary resources in stimulating third party development can be understood.

In the model, *boundary resources design* is a process involving the platform owner developing new or modified boundary resources to enhance external contributions and address control concerns. The design of boundary resources can be reactive or proactive. The design of boundary resources to enhance opportunities for external contribution increases the scope and diversity of the software platform and is thus denoted *resourcing* the software platform. On the other hand, control concerns may emerge when third party developers launch applications that represent potential threats to the platform. As a result, boundary resources may be (re)designed to address the platform owner's control concerns, a process referred to as *securing*. Finally, *boundary resources use* is a process involving third party developers developing end user applications by leveraging the boundary resources offered by the platform owner.

Community Boundaries and the Boundary Resources Model

A software ecosystem has several communities of practice (CoPs). A community of practice is a group of people pursuing a common objective (a joint enterprise), sharing a repertoire of tools and ways to solve recurring problems (a practice) and having a shared domain of interest (an identity) that makes them different from other people (Wenger, 2011). With respect to software ecosystems, each of the communities of internal developers, external developers, domain experts and end users constitute a community of practice. CoPs are important social units through which competences and experiences can be exchanged and acquired.

The existence of a community of practices implicitly suggests the existence of boundaries. Boundaries are a separation between two groups of people arising from differences in enterprise, repertoires and capabilities (Wenger, 2000). Boundaries are channels through which competences, experiences and resources can be exchanged, allowing co-learning across communities and enriching capacities on each side of the boundary. The exchange across boundaries is facilitated by three bridges: boundary objects, boundary interactions and brokers (Wenger, 2000). Boundary objects are artefacts, for example tools and documents, linking or shared by communities of practice across a boundary. Boundary interactions are events or encounters (for example visits and meetings) that provide direct exposure to members of an external CoP. Lastly, brokers are human actors that operate between communities of practice and engaged in 'import-export' of competences, knowledge and resources.

The boundary resource model is based on the boundary object (Bergman et al., 2007; Carlile, 2002; Star and Griesemer, 1989; Wenger, 2000) construct. The model focuses on the role played by software artefacts, such as APIs and SDKs, in cultivating third party development. The model is, however, silent on the potential role-played human agents operating between internal developers and third-party developers in shaping third party development. Similarly, the role of boundary interactions such as events aimed at enhancing the capacity of third party developers is left in the background. Our understanding of how to cultivate third party development can, therefore, be further strengthened by drawing insights from boundary interactions and brokers in addition to those from boundary objects.

METHODOLOGY

As stated in the introduction, the aim of this paper is to address the question how can third development in a platform-centric software ecosystem be cultivated? To answer this question a longitudinal case study (Creswell, 2014; Yin, 2013; Zainal, 2007) involving the DHIS2 software platform was carried out. The transitions DHIS2 has undergone to allow and foster third party innovations make it an ideal case for addressing the research question above. DHIS2 traces its origin to DHIS 1.0 which was a desktop application based on proprietary Microsoft technologies such as Microsoft Access. Challenges with proprietary technology led to the release of DHIS2, an open source web based application. Growing worldwide usage of DHIS2 made it difficult for University of Oslo to effectively and timely address divergent user requirements. To relieve the burden on University of Oslo there was a need to support third party innovations as a result DHIS2 embraced a software platform approach. The ongoing quest to encourage the development of third party applications on top of it makes the DHIS2 software platform an interesting case in trying to understand how to cultivate third party development.

Different data collection strategies were employed during the case study running from 2014 to 2017. These included: interviews, field experiments (pilots), participant observation and document reviews. As part of observations, in 2014 the researcher attended four weeks of the open source software development masters course at the University of Oslo. During the course master students are taught the

fundamentals of DHIS2 application development. Going further with observations, between 2014 and 2017 the researcher attended three DHIS2 expert academies in Oslo (Norway) and 1 DHIS2 application development academy in Dar es Salaam (Tanzania). These observations were complemented by field experiments (pilots) in Malawi in form of two application development workshops carried out in Malawi. The first workshop took place in March 2016 and focused on development of web applications on top of DHIS2. The second workshop took place in October 2017 and focused on development of Android applications on top of DHIS2.

Over the period between 2014 and 2017, several interviews were carried involving organizers, facilitators, students and participants of DHIS2 related courses, academies and workshops. Participants in the academies and workshops were people interested in developing or already developing applications for the DHIS2 platform. These interviews centered around the various boundary resources, boundary interactions and human agents and how they are shaping third party software development in the DHIS2 ecosystem. Respondents in the interviews were given an informed consent form ensuring their privacy and confidentiality among other things. Altogether, 33 interviews have been conducted between 2014 and 2017. This is summarized in table below.

Table 1. Data Collection Summary

Period	Data Collection Event	Interviews	Other Techniques
August 2014	Open Source Software Development course, MSc in Information Systems Program, University of Oslo, Norway	0	Participant observation
August 2015	DHIS2 Experts Academy, University of Oslo, Norway	2	Participant observation
March 2016	DHIS2 Web Apps Development Workshop, Zomba, Malawi	5	Field experiment, Participant observation
August 2016	DHIS2 Experts Academy, University of Oslo, Norway	3	Participant observation, Meetings
January 2017	DHIS2 Web Apps Development Workshop, Kampala, Uganda	7	Field experiment, Participant observation
August 2017	DHIS2 Experts Academy, University of Oslo, Norway	3	Participant observation, meetings
October 2017	DHIS2 Android Apps Development Workshop, Zomba, Malawi	2	Field experiment, Participant observation
November 2017	DHIS2 Web Apps Development Academy, Dar es Salaam, Tanzania	11	Participant observation

Document reviews were carried by going through documentation, reports, research accounts and other written material on DHIS2. Further data was collected through participation in projects that aimed at either building capacities of third party developers' capacity to build DHIS2 applications or actually developing a DHIS2 application. This included participating in the content development, facilitation and organization for the DHIS2 application workshops in Malawi. In addition, data was collected by participating in the mHealth4Afrika project which is an ongoing joint project involving University of Oslo (Norway), University of Gondar (Ethiopia), University of Malawi (Malawi), Strathmore University (Kenya), Nelson Mandela Metropolitan University (South Africa) and International Information Management Corporation (Ireland) building a maternal and child health records system on top of DHIS2.

The data collected using the various data collection techniques outlined above was recorded in various forms: field notes, google forms and sheets, audio recordings and photos. This data included among other things existing boundary resources in DHIS2 ecosystem and their generative attributes, proposed boundary resources, capacity building initiatives and respective factors for the success or failure of such initiatives. Where audio recordings were captured they were later transcribed to provide a corresponding textual account. Using the boundary resources model, generativity concept - generative capacity, and community of practice concepts - brokers, boundary interactions and boundary objects as analytical lens the data collected was thematically analyzed. The results of this analysis are presented in the discussion and concluding remarks sections below.

THE CASE OF DHIS2

DHIS2 (figure 2) is an open source software platform developed by the Health Information Systems Program (HISP) at the University of Oslo (HISP UiO). It is used primarily to collect, manage, aggregate, analyze and visualize routine health data by ministries of health and other stakeholders in developing countries. Although it is primarily built for use in the health domain, the generic nature of DHIS2 has seen its use in other domains grow over the years. Over the period of its existence, DHIS2 has evolved from a desktop application built around proprietary technologies to an open source and web based application, and now, to an open source software platform with a RESTful Web API that supports creation of third party innovations using ubiquitous web technologies such as JavaScript, CSS and HTML5.

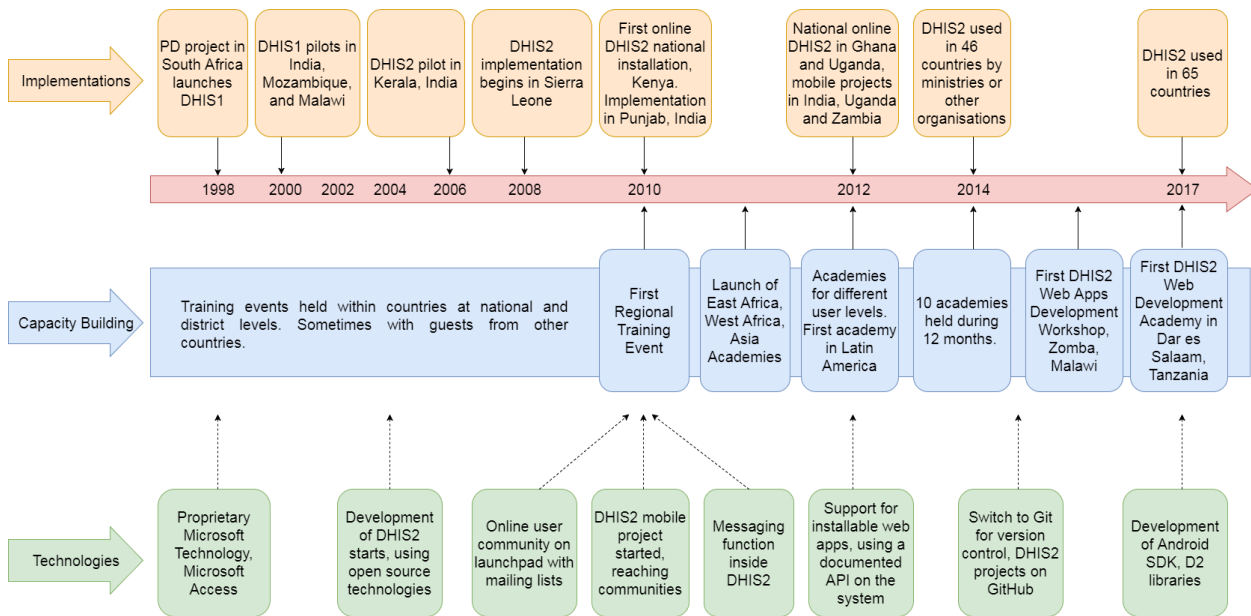


Figure 2: DHIS2 Timeline (adapted from a slide in the DHIS2 Online Academy)

DHIS2 traces its origin to the Reconstruction and Development program in post-apartheid South Africa under which HISP was initiated in 1995. One of the aims of the program was creating a unified HIS across the country. To fulfil this aim, a pilot project to develop a district-based HIS in the Western Cape Province was proposed. Between 1996 and 1998, the pilot project received financial backing from the

Norwegian Agency for Development Cooperation (NORAD) and resulted in a prototype of what was called District Health Information Software (DHIS) released in 1998. Successful implementations in the Western Cape led to the adoption of DHIS by the Eastern Cape Province in the same year and by the year 2000 several other provinces in South Africa had adopted DHIS.

From South Africa DHIS was later introduced to other developing countries including Mozambique, Malawi and India. At the time, DHIS was a desktop application built using Microsoft Access. This presented two challenges. First, as a desktop application it lacked support for sharing data between geographically distant stakeholders. Second, the use of proprietary Microsoft technology made scaling the solution costly as rolling out entailed paying license fees for each workstation where DHIS was installed. To address these two challenges, HISP UiO, with support from various international donors, embarked on a project to develop an open source and web based version of the software. This resulted in what is now known as DHIS2.

Platformisation of DHIS2

With DHIS2 now a free, open source and web-based application the problems of shared access and cost of scaling were alleviated. This has since led to increased adoption of DHIS2 by ministries of health and non-governmental organizations in developing countries. It is currently in use in more than 60 countries. This includes, among others, countries in the Eastern and Southern Africa region such as Uganda, Kenya, Rwanda, Tanzania, Malawi, Zambia, Zimbabwe, South Africa and Mozambique.

The increased adoption of DHIS2 came with its own challenges. First, because of increased adoption there was also an increase in user requirements that the developers at HISP UiO had to address to satisfy the end user community. At the same time, the different context under which the end users operate meant that the user requirements were divergent and often competing. Effectively and timely addressing such divergent end user requirements became a challenge. This resulted in complaints and queries from the end-user community regarding how user requirements are prioritized and addressed.

With the switch to being open source, it had been envisaged that the innovation burden on DHIS2 will be shared between HISP UiO developers and external developers in the end user community. However, the core of DHIS2 was developed using Java and requisite skills were not as ubiquitous among external developers in the end user community. As a result, a large part of the end user community remained dependent on HISP UiO for the implementation of requirements specific to their context. This created an innovation bottleneck against developers at HISP UiO resulting in delays and sometimes failure to respond to some end user requirements.

The core of DHIS2 is developed using Java and a stack of related software development frameworks. For most external developers, developing for DHIS2 meant learning a new programming language as well as a stack of associated technologies used and therefore not as attractive. To effectively and timely address end user requirements it became necessary to devise a way to allow external developers develop solutions on top of DHIS2 without requiring them to learn Java and a stack of software frameworks used by HISP UiO. This has seen DHIS2 evolving from a mere open source software to now an open source software platform.

DHIS2 Boundary Resources

As a software platform DHIS2 (figure 3) provides a range of boundary resources to enable third party development. First, within its core DHIS2 supports the creation of custom modules that can co-exist with core modules. Such modules must be written in Java alongside frameworks such as Spring and Hibernate. Second, DHIS2 now has a RESTful Web API with which external developers can develop third party applications on top of DHIS2. The API continues to evolve with each new version of DHIS2 with the aim of offering better support to third party developers.

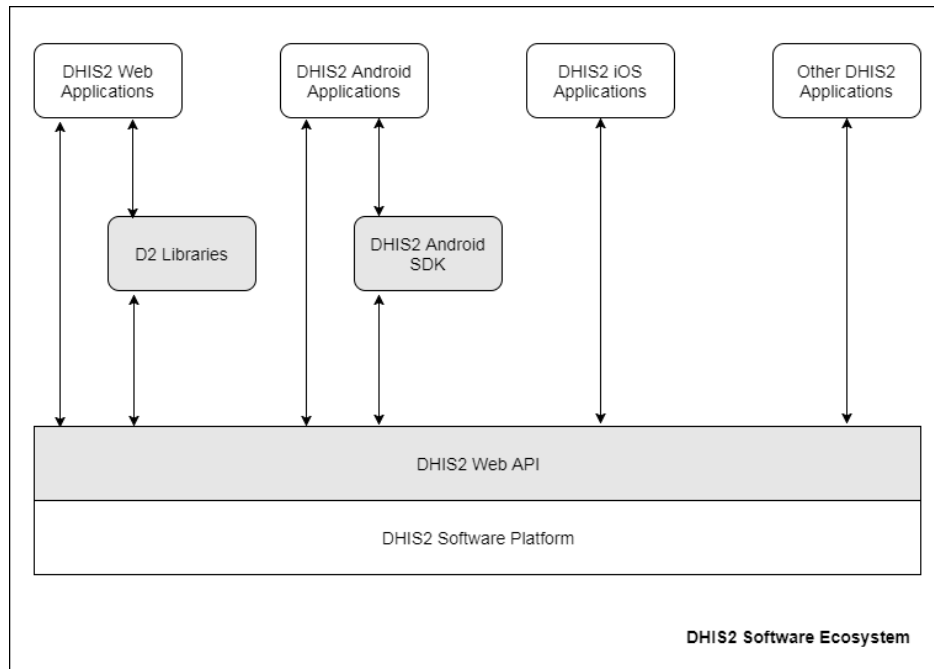


Figure 3: DHIS2 Software Ecosystem

Using the Web API, external developers do not necessarily need to learn a new programming language just to develop applications for DHIS2. As long as the programming language they are familiar with has features to handle web-based data they can use it to develop applications for DHIS2. Typically, third party DHIS2 applications are web applications written using JavaScript, CSS and HTML5. However, the flexibility of the Web API allows for creation of different kinds of applications as well. For example, the popularity of Android as a computing platform in developing countries has attracted interest in DHIS2 Android applications.

In addition to the Web API, HISP UiO has developed other auxiliary boundary resources to support specific developer needs. One of such resources is the d2 library, a JavaScript library which provides a level of abstraction above the Web API allowing third party developers to develop web applications without requiring in depth knowledge of the API. Another of such auxiliary boundary resources is the DHIS2 Android SDK, currently in beta, which abstracts the Web API when building Android applications on top of DHIS2. Besides boundary resources created by HISP UiO a number of community-driven boundary resources. For example, some third-party developers have created “seed applications” which are dummy applications containing boiler plate code that allows new developers to quickly get started building DHIS2 applications.

To facilitate distribution of third party applications, a dedicated online app store for DHIS2 web applications (figure 4) was created and can currently be accessed on <https://play.dhis2.org/appstore/>. There is also a dedicated app store for DHIS2 Android applications which can currently be accessed on <https://www.dhis2.org/appstore-android>. The number of available applications in both app stores is however limited. At the time of writing, there were 5 applications in the DHIS2 Android app store and 13 applications in the DHIS2 Web Applications app store. Most of these apps, particularly on the DHIS2 Android app store, are developed by HISP UiO.

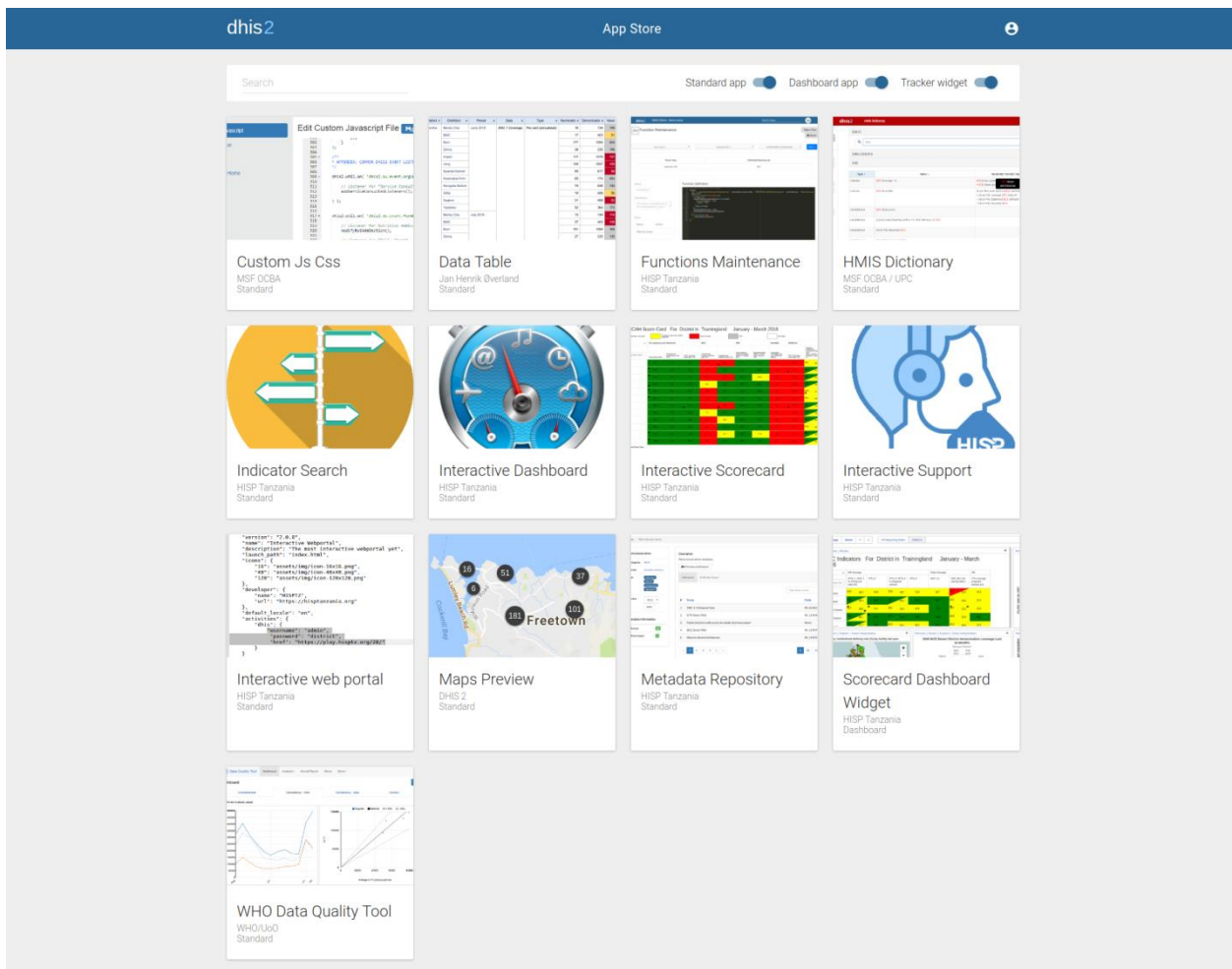


Figure 4: DHIS2 App Store

Building Third Party Development Capacity

Despite the existence of a range of boundary resources to support development of third party applications on top of the DHIS2 platform, the range of applications available in its app stores is however limited. Furthermore, several applications in the app stores have been developed by HISP UiO itself. Providing the boundary resources has not necessarily resulted in distributing the innovation

burden as much as it was envisaged in the platformisation of DHIS2. This is partly attributed to lack of awareness and requisite development capacity among potential third-party developers.

To address the capacity and awareness gap, HISP UiO, in collaboration with its global partners, has several initiatives aimed at enhancing the generative capacity of prospective third-party developers. Since the inception of DHIS2, University of Oslo has been running a master course on Open Source Software Development centered on the software. The course has evolved over the years to embrace the platform approach taken on DHIS2. In the course, students work on different projects developing third party applications on top of DHIS2. In Malawi, the University of Oslo is working in collaboration with University of Malawi where a similar course was introduced as part of a Master of Science in Informatics program.

Within the DHIS2 community, DHIS2 academies have been a popular vehicle for building different kinds of capacity among end users and technical personnel (e.g. as shown in figure 5). Topics at such academies have ranged from information use through to customization and implementation of DHIS2. The DHIS2 community is now exploiting the same vehicle to cultivate generative capacities of prospective third-party developers. With funding from UNICEF, in March 2016, HISP UiO in collaboration with the University of Malawi held a DHIS2 Application Development Workshop in Zomba, Malawi as a pilot for a potential DHIS2 Application Development academy. The workshop drew participants from Malawi, Kenya, Ethiopia and Zambia. A similar pilot workshop was carried out in January 2017 in Kampala, Uganda and attracted participants from Rwanda, Kenya, Uganda, Tanzania, Malawi, and South Africa. Following feedback from the pilots a full scale DHIS2 Application Development Academy took place in November 2017 in Dar es Salaam, Tanzania.

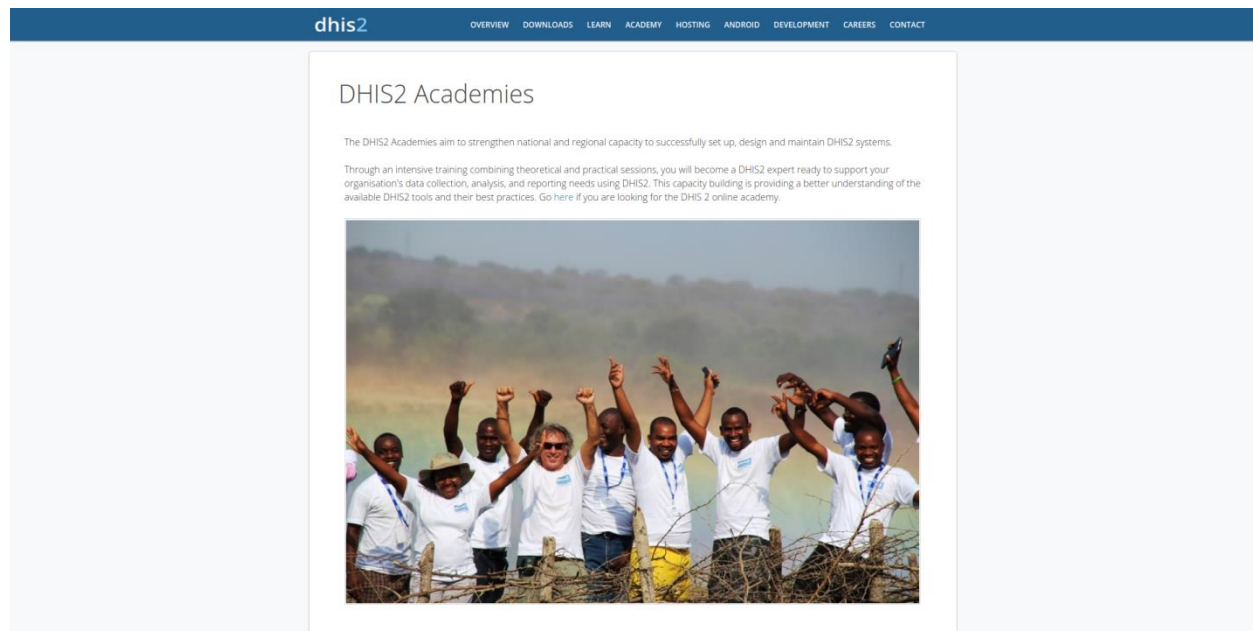


Figure 5: DHIS2 Academies Website

In addition to academies, other avenues for cultivating the generative capacity of third party developers are being explored. For example, recently the University of Oslo entered an agreement with University of Malawi and the Eduardo Mondlane University in Mozambique for a staff and student exchange program which will among other things allow HISP UiO to host technical personnel from Malawi and

Mozambique as a form of internship. In Tanzania, HISP Tanzania has an internal developer mentorship program where experienced DHIS2 developers work with new recruits to build their capacity. This has led to HISP Tanzania amassing adequate local capacity which has enabled them to develop several third-party applications on top of DHIS2. All these efforts are being done in complement to existing efforts on the boundary resources described in the earlier section.

DISCUSSION

The platformisation of DHIS2 and the provision of various boundary resources has made it possible for external developers to develop third party applications on top of DHIS2. The aim has been to usher in an era of distributed innovation whereby the innovation burden on DHIS2 is distributed between internal developers at HISP UiO and third-party developers within the ecosystem. HISP UiO sees the provision of boundary resources as critical to the achievement of this aim. This agrees with the intellectual account provided by the boundary resources model (Ghazawneh and Henfridsson, 2013). Without boundary resources, third party developers cannot extend DHIS2 functionality without the risk of forking the software. With appropriate boundary resources, useful third-party applications can be developed while at the same time circumventing forkability.

However, empirical data from the DHIS2 ecosystem indicate that provision of boundary resources is not in itself an endgame in cultivating third party development. Besides providing boundary resources, it was observed that third party development requires building generative capacities of third party developers. This agrees with the argument that third-party development demands shifting of software design and development capabilities from internal to external developers (Prügl and Schreier, 2006; von Hippel and Katz, 2002). This calls for the extension of the boundary resources model to foreground generative capacity alongside boundary resources as factors having impact on third party development. Sections 5.1 and 5.2 present two dimensions that can be used to extend the boundary resources model. Section 5.3 concludes the discussion by providing an extended boundary resources model and is followed by concluding remarks.

Software Development and Capacity Building Boundary Resources

The boundary resources model is based on the boundary object concept (Ghazawneh and Henfridsson, 2013, 2010). Drawing from this concept the model defines boundary resources as software tools and regulations that serve as an arm's length interface between platform owners and third-party developers facilitating the development of third party applications (Ghazawneh and Henfridsson, 2013). With this definition, our understanding of boundary resources is limited to technical artefacts, such as SDKS, APIs and libraries that are provided alongside a platform to facilitate third party development. However, the boundary between platform owners and external developers is not characterized by boundary objects alone. Besides boundary objects, boundary interactions and human agents as brokers are deployed to facilitate third party development. For example, the DHIS2 ecosystem has application development academies and workshops as boundary interactions aimed at cultivating third party development. Such boundary interactions rely on facilitators and mentors entrusted with the task of instituting a shift in software design capabilities from internal developers to third party developers. Just like the boundary objects, these boundary interactions and brokers can be regarded as boundary resources facilitating third party development. This calls for extending our understanding of boundary resources to include brokers and boundary interactions in addition to boundary objects.

Extending our understanding of boundary resources to include brokers and boundary interactions allows for a distinction between software development and capacity building boundary resources. Software development boundary resources are software artefacts that are used to develop third party applications. Such boundary resources include SDKs, API, seed applications and libraries. On the other hand, capacity building boundary resources are boundary objects, boundary interactions and brokers that are used to build the third-party development capacity of external developers. Within the DHIS2 ecosystem, such boundary resources include capacity building events such as application development academies and workshops, boundary objects such as developer documentation, and brokers such as academy and workshop facilitators and experienced developers acting as mentors for new third-party developers. To cultivate third party development, both software development and capacity building boundary resources are required.

While this paper uses the distinction between software development and capacity building boundary resources to extend the boundary resources, further distinction can be made between endo-platform boundary resources and exo-platform boundary resources, and between platform-owner driven boundary resources and community driven boundary resources. Endo-platform boundary resources are boundary resources that come bundled as part of the software platform. These include software artefacts such as APIs, SDKs and libraries. On the other hand, Exo-platform boundary resources are boundary resources that exist outside the software platform. These include software artefacts such as documentation, boundary interactions such as academies and brokers such as facilitators and mentors. Going further with this discussion, platform-owner driven boundary resources are those that are put in place by the platform owner to facilitate third party development. On the other hand, community-driven boundary resources are those that are derived through efforts by third party developers or other stakeholders other than the platform owner.

Internal and External Generative Capacity

The case of the DHIS2 reveals that third-party development requires more than just providing software development resources such as SDKs and APIs. There is a need to propagate software design and development capabilities to external developers to enable them to develop third party applications. Therefore, provision of software development boundary resources must be complemented by mechanisms aimed at building the generative capacities of external developers. In the DHIS2 ecosystem such mechanisms include application development academies and workshops. In literature on software platform generativity, the term generative capacity has been used to describe a collection of competences that enable an individual to produce something innovative in a particular context. With respect to third party development, generative capacity relates to a developer's ability to develop third party applications on top of a given software platform.

In this regard, distinction can be made between internal generative capacity and external generative capacity. Internal generative capacity refers to platform owners' software development and related competencies that enable them design, develop and maintain the software platform and platform owner driven boundary resources. On the other hand, external generative capacity refers to third party developers' software development and related competences that enable them to design, develop and maintain third party applications and community-driven boundary resources. No matter how good the software development boundary resources a software ecosystem has, third party development remains a mere possibility until there exists adequate external generative capacity.

Extended Boundary resources model

Taking into consideration the foregoing discussion the paper presents, an extended boundary resources model in figure 6 is shown below. Below the figure descriptions of each of the construct used in the model are provided. In the extended boundary resources model the paper introduces the constructs *internal generative capacity*, *external generative capacity*, *generative capacity use* and extends the *boundary resources use* construct to include the use of capacity building boundary resources by platform owners and other stakeholders in the ecosystem to build external generative capacity. In addition, constructs *software development boundary resources* and *capacity building boundary resources* replace the boundary resources construct. Adding these constructs helps foreground other factors critical for third party development besides provision of software development boundary resources.

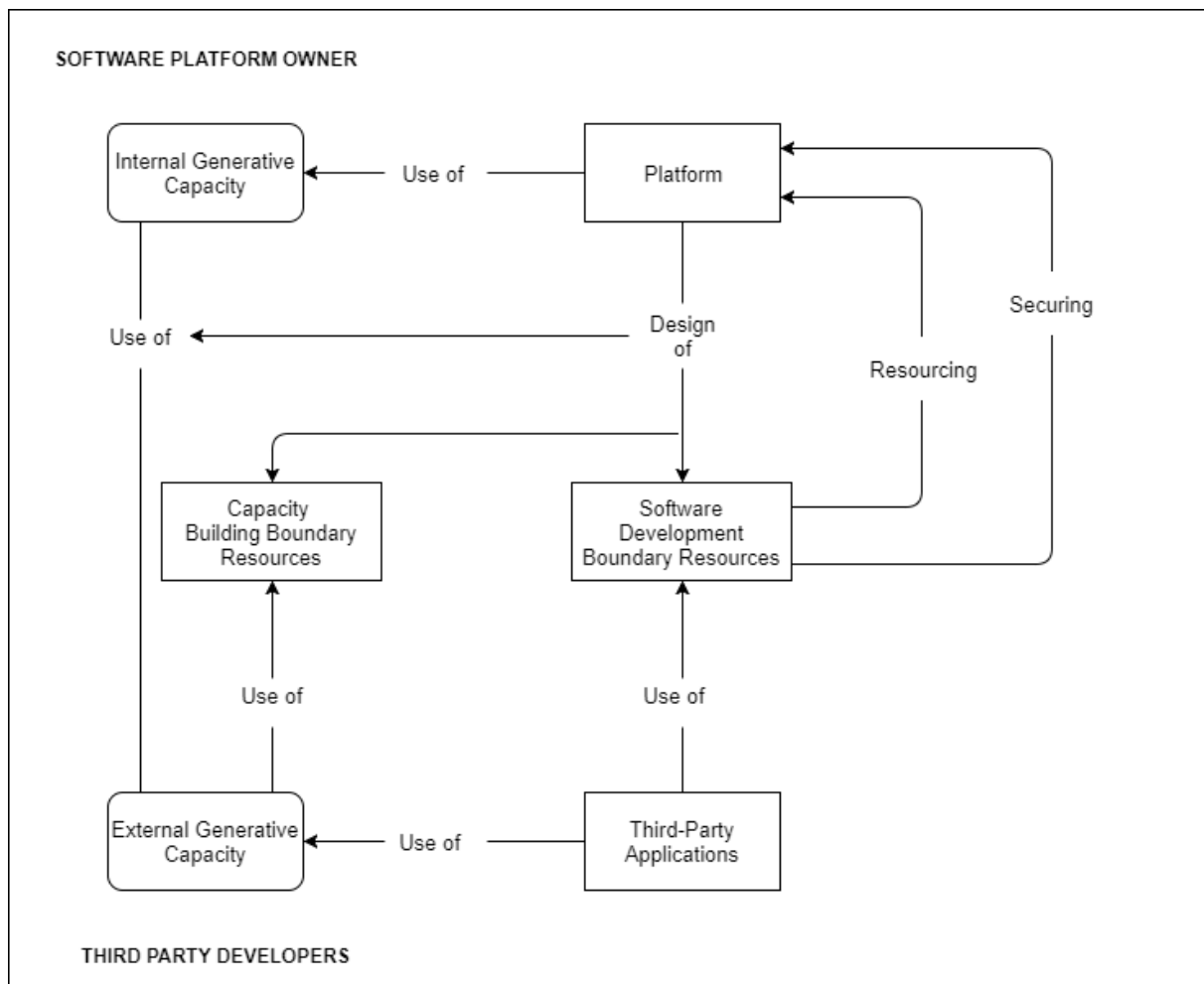


Figure 6: Extended Boundary Resources Model

Constructs

Platform: “The extensible codebase of a software based system that provides core functionality shared by modules that interoperate with it and the interfaces through which they interoperate” (Tiwana et al. 2010, p. 676)

Software Development Boundary Resources: boundary objects in form of software artefacts that are used to develop third party applications. Such boundary resources include SDKs, API, seed applications and libraries.

Capacity Building Boundary Resources: boundary objects, boundary interactions and brokers that are used to build the third-party development capacity of external developers. Such boundary resources include, amongst other things, capacity building events such as application development academies and workshops.

Third Party Applications: Executable pieces of software that are offered as applications, services, or systems to end users of the platform.

Internal Generative Capacity: platform owners' software development and related competencies that enable them design, develop and maintain the software platform and platform owner driven boundary resources

External Generative Capacity: third party developers' software development and related competences that enable them to design, develop and maintain third party applications and community-driven boundary resources

Generative Capacity Use: the use of internal generative capacity by platform owners to design, develop and maintain the software platform and platform owner driven boundary resources, and the use of external generative capacity by third party developers to develop third party applications and community-driven resources

Boundary Resources Use: the use of software development boundary resources by third party developers to build third party applications and the use of capacity building boundary resources by platform owners and other stakeholders to build external generative capacity.

Resourcing: The process by which the scope and diversity of a platform is enhanced.

Securing: The process by which the control of a platform and its related services is increased.

CONCLUSION

The boundary resources model aims to provide an intellectual account for cultivating third party development in platform-centric software ecosystems. Empirical data from renowned platform centric software ecosystems such as Google's Android and Apple's iOS renders credence to the argument advanced by the model that boundary resources are critical to third party development. In this study, empirical data from the DHIS2 software ecosystem also supports this argument. However, further observations reveal that boundary resources are not in themselves an endgame in cultivating third party development. Analyzing the data with respect to constructs from generativity literature reveals that external generative capacity has a role to play in cultivating third party development.

In addition, analyzing the data with respect to constructs from community boundaries literature reveals that boundary resources are more than technical artefacts – they are socio-technical artefacts comprising of not only boundary objects but also boundary interactions and brokers. Extending our understanding of boundary resources to include boundary interactions and brokers, allow a distinction between software development and capacity building boundary resources both of which are required to cultivate third party development. The paper makes further distinction between exo-platform boundary resources and endo-platform boundary resources, and between platform-owner driven boundary resources and community driven boundary resources.

Based on these analytical results, the paper has proposed an extended boundary resources model presented in the discussion above. In the extended boundary resources model the paper introduces the constructs internal generative capacity, external generative capacity, generative capacity use and extends the boundary resources use construct to include the use of capacity building boundary resources by platform owners and other stakeholders in the ecosystem to build external generative capacity. In addition, constructs software development boundary resources and capacity building boundary resources replace the boundary resources construct. Adding these constructs helps foreground other factors critical for third party development besides provision of software development boundary resources.

In doing this, it is worth noting that every research has its limitations. Further knowledge can be obtained by augmenting this research with further studies. In the future, for example, research could look at the interplay between factors that influence third party development as put across in the extended boundary resources model and developer motivation factors. There are a lot of accounts on developer motivation factors in the open source literature which can provide a good starting point (Aknouche and Shoan, 2013; Fogel, 2005; Hars and Ou, 2002). Notwithstanding the importance of such interplay, for the sake of brevity and simplicity, such an analytical account is beyond the scope of this paper.

REFERENCES

- Aknouche, L., Shoan, G., 2013. Motivations for Open Source Project Entrance and Continued Participation.
- Avital, M., Te'eni, D., 2009. From generative fit to generative capacity: exploring an emerging dimension of information systems design and task performance. *Information Systems Journal* 19, 345–367.
- Bergman, M., Lyytinen, K., Mark, G., 2007. Boundary objects in design: An ecological view of design artifacts. *Journal of the Association for Information Systems* 8, 546.
- Bosch, J., 2009. From software product lines to software ecosystems, in: *Proceedings of the 13th International Software Product Line Conference*. Carnegie Mellon University, pp. 111–119.
- Bosch, J., Bosch-Sijtsema, P., 2010. From integration to composition: On the impact of software product lines, global development and ecosystems. *Journal of Systems and Software, SI: Top Scholars* 83, 67–76. <https://doi.org/10.1016/j.jss.2009.06.051>
- Carlile, P.R., 2002. A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development. *Organization Science* 13, 442–455.
- Creswell, J.W., 2014. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE.
- Eck, A., Uebernickel, F., Brenner, W., 2015. *The Generative Capacity of Digital Artifacts: A Mapping of the Field*.
- Fogel, K., 2005. *Producing Open Source Software*, 1st ed. O'Reilly.
- Ghazawneh, A., Henfridsson, O., 2013. Balancing platform control and external contribution in third-party development: the boundary resources model. *Information Systems Journal* 23, 173–192. <https://doi.org/10.1111/j.1365-2575.2012.00406.x>
- Ghazawneh, A., Henfridsson, O., 2010. Governing third-party development through platform boundary resources, in: *The International Conference on Information Systems (ICIS)*. AIS Electronic Library (AISeL), pp. 1–18.
- Grisot, M., Hanseth, O., Thorseng, A., 2014. Innovation Of, In, On Infrastructures: Articulating the Role of Architecture in Information Infrastructure Evolution. *Journal of the Association for Information Systems* 15.
- Hars, A., Ou, S., 2002. Working for Free? Motivations for Participating in Open-Source Projects. *International Journal of Electronic Commerce* 6, 25–39.
- Henfridsson, O., Lindgren, R., 2010. User involvement in developing mobile and temporarily interconnected systems. *Information Systems Journal* 20, 119–135.
- Kimaro, H.C., 2006. Strategies for Developing Human Resource Capacity to Support Sustainability of ICT Based Health Information Systems: A Case Study from Tanzania. *The Electronic Journal of Information Systems in Developing Countries* 26.

- Kimaro, H.C., Nhampossa, J.L., 2005. Analyzing the problem of unsustainable health information systems in less-developed economies: Case studies from Tanzania and Mozambique. *Information Technology for Development* 11, 273–298. <https://doi.org/10.1002/itdj.20016>
- Manikas, K., Hansen, K.M., 2013. Software ecosystems – A systematic literature review. *Journal of Systems and Software* 86, 1294–1306.
- Mutula, S.M., Van Brakel, P., 2007. ICT skills readiness for the emerging global digital economy among small businesses in developing countries: Case study of Botswana. *Library Hi Tech* 25, 231–245. <https://doi.org/10.1108/07378830710754992>
- Polak, M., 2015. Platformisation of an Open Source Software Product: Growing up to be a generative software platform. University of Oslo, Oslo, Norway.
- Prügl, R., Schreier, M., 2006. Learning from leading-edge customers at The Sims: opening up the innovation process using toolkits. *R&D Management* 36, 237–250.
- Star, S.L., Griesemer, J.R., 1989. Institutional Ecology, “Translations” and Boundary Objects: Amateurs and Professionals in Berkeley’s Museum of Vertebrate Zoology, 1907-39. *Social Studies of Science* 19, 387–420. <https://doi.org/10.2307/285080>
- Tiwana, A., 2013. *Platform Ecosystems: Aligning Architecture, Governance, and Strategy*, 1 edition. ed. Morgan Kaufmann, Amsterdam; Waltham, MA.
- 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.
- von Hippel, E., Katz, R., 2002. Shifting Innovation to Users via Toolkits. *Management Science* 48, 821–833. <https://doi.org/10.1287/mnsc.48.7.821.2817>
- Wenger, E., 2011. *Communities of practice: A brief introduction*.
- Wenger, E., 2000. Communities of practice and social learning systems. *Organization* 7, 225–246.
- Yin, R.K., 2013. *Case Study Research: Design and Methods*, 5 edition. ed. SAGE Publications, Inc, Los Angeles.
- Zainal, Z., 2007. Case study as a research method. *Jurnal Kemanusiaan* 5.
- Zittrain, J., 2008. *The future of the Internet and how to stop it*. Yale University Press, New Haven, [Conn.].
- Zittrain, J., 2006. The Generative Internet. *Harvard Law Review* 119, 1974–2040.