Original Research Article

BIG DATA & SOCIETY

Big Data & Society
July–December 2018: 1–18

DOI: 10.1177/2053951718790591
journals.sagepub.com/home/bds
SAGE

The social construction of technological stasis: The stagnating data structure in OpenStreetMap

Matthias Plennert

Abstract

The article aims for examining the 'technological stasis' of the data structure in OpenStreetMap – the successful global collaborative geodata project devoted to 'create and distribute free geographic data for the world'. Digital structures are strongly influenced by continuing stagnation. This technological stasis – the lack of change in technology – influences data in various ways, as demonstrated by the intensive discussion of the issue by computer scientists and software engineers. However, existing research describing stagnating software is often technic centred and fuzzy, while critical research is barely considering issues of technological stasis in the digital context at all. Therefore, this paper aims for enriching this body of knowledge in order to shed light on aging data structures. I reframe technological stasis with a social-constructivist perspective – using the approach of Social Construction of Technology – especially with the concept of technological frames. Based on the case example of OpenStreetMap, my findings suggest that the data structure – and its stasis – is the outcome of competing understandings and perspectives, shaped by power asymmetries. Although the data structure did not significantly change for more than 10 years, I demonstrate that this is not because of a lack of motivation, nor technological difficulties of carrying out such changes. The technological stasis is rather rooted in the dominant position of few project members who are able to change the software design; it is their perception of the project that defines how data should be stored and what features are dispensable.

Keywords

Social construction of technology, technological frames, OpenStreetMap, legacy software, technological stasis, data structure

Introduction

In the wider arena of Science and Technology Studies (STS), technological dynamics, innovation and the emergence of new technologies are traditionally predominant subjects of discussion. For example, Bijker (2012) analysed the emergence of Bakelite, a predecessor of plastic material; MacKenzie (2012) focused on developments of missile guidance systems in the US military; Law (1987) examined inter alia the innovation of sailing ships in the context of the colonial expansion of Portugal. All of them share the understanding that – to explain technological change – we have to scrutinize stability, obduracy and continuity of technological artifacts. Hommels acknowledges that '[. . .] it comes as no surprise that technology's obduracy and its effects on society have been a major concern in technology studies, if not the prevalent one' (2005: 329). However, when turning to current discussions dealing with emerging technologies, the exploration on stagnation seems missing. Especially in the context of digital technologies, the lack of a theoretical discussion regarding technological stasis is striking.

Friedrich-Alexander-University Erlangen-Nuremberg (FAU), Institute for Geography, Erlangen, Germany; Institute for Human Geography, Goethe-University Frankfurt am Main, Frankfurt am Main, Germany

Corresponding author:
Matthias Plennert, Department Geographie und Geowissenschaften, Friedrich-Alexander-University Erlangen-Nuremberg, Wetterkreuz 15, Erlangen 91054, Germany.
Email: matthias.plennert@fau.de

And yet, many software applications, algorithms or Big Data projects are mainly influenced by continuing stagnation. Taking the example of databases, old data structures are preserving outdated understandings and concepts, making it increasingly difficult for a project to adapt to new needs evolving in the broader environment. Furthermore, outdated data structures impede the development of other applications in the overall system, which build upon them, forcing them to compensate the deficiencies with cumbersome workarounds. This technological stasis is hardly a new issue, as it has been subject to software engineers in various studies (for example, see Bennett, 1995; Eick et al., 2001; Parnas, 1994). However, it has been mainly addressed from a problem-solving perspective (Chiang and Bayrak, 2006; Kim, 1997; Miller, 1998).

As existing literature addressing stagnating software is often technic centred and critical research barely considers issues of technological stasis in the context of digitization at all, this paper aims for enriching this body of knowledge by shedding light on stagnating data structures. Therefore, drawing on the debates of critical data studies, software studies and the social construction of technology (SCOT), I will reframe technological stasis with a social-constructivist perspective – utilizing the concept of technological frames (Bijker, 2012; Hommels, 2005). The objective of this paper is to address technological stasis on the case example of OpenStreetMap (OSM). I want to approach this objective from two opposing perspectives: first, from an engineering point of view, I will discuss the phenomenon of legacy software and its impact on technological change. Thereby I will show the insufficiency of this approach. Second, I will take a social-constructivist perspective by applying the theory of the SCOT, and the concept of technological frames in particular. In doing so, I want to reframe this concept by showing that it is a question of social context and power when talking about technological stasis.

The case example for this study is OSM, the 'free wiki world map' (OpenStreetMap contributors, 2017). I argue, that the data structure in OSM is the outcome of competing understandings and perspectives, shaped by power asymmetries. Although the data structure did not significantly change for more than 10 years, I will demonstrate that this is not because of a lack of motivation or willingness, nor technological difficulties of carrying out such changes. The technological stasis is rather rooted in the dominant position of few project members who are able to change the software design; it is their perception of the project that defines how data should be stored and what features are dispensable.

In the following sections, I will first give an overview of existing literature regarding Open Source Software (OSS) and understandings of technological stasis from an engineering perspective. I then turn to the case example of OSM, illustrating the perception of technological stasis. Based on my empirical findings, I will demonstrate that there are in this respect contradicting voices in the community. To dissolve the contradiction between the different perceptions of the very same phenomenon, I introduce the concept of technological frames, based on the SCOT. Following this theory, I will then argue, that there are different social groups involved in technological development of the digital infrastructure, divided along the line of their technical skills and field of action. This division also marks the power of the group in the community: if the group with the technical skills for manipulating the data structure won't see any problem with the existing infrastructure or won't see the benefits of adapting it to new needs, no change will happen.

OSS and OpenStreetMap

The OpenStreetMap software infrastructure builds on free and OSS. Thus, besides the open data objective, fundamental parts of OSM's software infrastructure also qualify as open source projects. But what are the particularities of OSS projects? A key aspect of OSS development are the diverging motivators in comparison with proprietary and for-profit software projects. While software development in a for-profit environment relies on extrinsic motivators such as monetary benefits, main motivators in OSS projects are intrinsic (Lakhani and von Hippel, 2003; Lerner and Tirole, 2002; Schweik, 2003; Shah, 2006). Furthermore, Achtenhagen et al. mentions fun, self-realisation or building a good reputation in the community as important driving forces in OSS communities (2003: 465 f.). Additionally, Bitzer et al. (2007), Feller and Fitzgerald (2000) and Nakakoji et al. (2002) highlight need-driven software development in OSS; or in the words Feller and Fitzgerald: 'scratching a developer's ''personal itch''' (2000: 5).

Due to the specific motivators and driving forces in OSS projects, power, governance and control differs from for-profit software development projects as well. Rajanen and Iivari refer to the 'onion-model' in order to describe the unequal participation within the community, referring to '[...] different layers representing levels of involvement in the community' (2015: 2). Feller and Fitzgerald (2000) and Mockus et al. (2002) share this perspective, when they describe a 'core team' of developers and a peripheral community in OSS projects. Focusing on the relation between those core developers and end-users, Iivari examines the power asymmetry (2009) by utilizing the theoretical framework of the Social Shaping of Technology (MacKenzie and Wajcman, 1999). In summary, it is a

common pattern for OSS projects that few contributors control vast parts of the software development (Rajanen and Iivari, 2015: 2).

Another key aspect of research on OSS are evolutionary dynamics: several authors refer to a 'life cycle' or 'evolution pattern' of OSS projects (Nakakoji et al., 2002; Schweik, 2003; Wynn, 2003). While the numbers of life cycle stages diverge, they mostly share an initiation-, growing-, maturing-stage or peak, and a decline or revive (Lattemann and Stieglitz, 2005: 4). However, the criteria for defining life cycle stages are to some degree overlapping and difficult to measure. Lattemann and Stieglitz mention several possible criteria, such as '(1) annual growth of participant base, (2) annual growth of user community, (3) growth in ''market share'', (4) user satisfaction with the product, and (5) peer recognition of the product' (2005: 4). Thus, the criteria for the life cycle of OSS projects mainly refer to the success of the product, its dissemination or user numbers.

Is OSM in decline in the sense of the OSS life cycle? Considering the database statistic of OSM, the number of active contributors, or users, is steadily increasing.[1] Accordingly, the amount of contributed geodata is growing as well.[2] Thus, the decline stage or even maturing stage in the sense of OSS life cycles isn't applicable. However, as I will further discuss in the following, OSM is in a stage of technological stasis, as the core of its technological infrastructure did not significantly change in the past 10 years.

Why did the core of OSM's software infrastructure stagnate for such a long time? There are several central concepts for explaining technological stasis in the computer sciences. The common approach is to focus on the technical artefact – old or outdated parts of software. In other words, to explain technical stasis, the main reason lies in the software or code itself. Thus, the common starting point of explanatory approaches is to analyse why it is hard to change old software or code. There are several different terms in this context which address the issue from different angles: code decay, code smell, technical debt, legacy software and software aging.

The term code decay stands for an understanding which argues that the reason for the difficulties of changing software is insufficiently designed and written code in reference to academic software development standards (see Sommerville, 2011: 732). Eick et al. define decayed code as 'harder to change than it should be [. . .]' (2001: 100). This understanding implies that, if the code was created or edited in a 'proper' way, there shouldn't be any problems in changing it afterwards. Similarly, technical debt addresses 'not-quite-right' code as immature coding (Cunningham, 1992). In other words, technical debt is characterized as incomplete coding which is postponed to a later point of time (Kruchten et al., 2012: 18). The term code smells is often used to refer to signs of 'improper' code (Tufano et al., 2015: 403); they are 'symptoms of poor design and implementation choices [. . .]' (Tufano et al., 2015: 403). It has become commonly accepted to see this phenomenon as an avoidable technical problem which subsequently causes difficulties in changing code (Olbrich et al., 2009).

Software aging, originally introduced by Parnas (1994) in his eponymic article, describes maturing software by comparing it to an aging human body. In doing so, he takes a more comprehensive perspective by looking at the software system. He addresses insufficient and bad maintenance as well as initial design failures by using health metaphors. Parnas stresses the growth of software over time (1994: 280): '[T]he easiest way to add a feature, is to add new code. Modifying existing code to handle the new situations is often difficult because that code is neither well-understood nor well-documented'. Thus, he argues for 'preventive medicine' or 'design for success', in order to tackle the problem of software aging from the very beginning of the life-circle. The concept of legacy software shares this basic understanding: technological stasis does not originate from 'improper' code, but is related to old or even outdated software. Legacy software describes software with 'proper' code which was created with now outdated guidelines and principles (Bennett, 1995: 20). In the wider arena of software engineering, software legacy is often discussed in the context of 'reengineering' (Miller, 1998), 'modernizing' (Chiang and Bayrak, 2006) or 'maintaining' (Jelber et al., 2003).

In summary, OSS projects differ in their motivators and power structures from proprietary, for-profit software projects. Furthermore, there is a pattern of OSS going through a specific life cycle. According to these OSS life cycles, OSM is not stagnating at all. However, as I will discuss in the following, OSM is in technological stasis. As this issue is a well-established topic in the computer sciences, there are several concepts for explaining technological stasis. They share an understanding in which either the cause of stasis, its solution, or both, lies in the code. Furthermore, focusing on a smaller scale, code decay, code smell and technical debt argue that the reason for technical stasis lies in improper code. Thus, this perspective implies that 'proper' coding can prevent software from aging or at least from stasis. Legacy software and software aging take a wider perspective by focusing on how the software structure changes over time.

Research design

For examining the technological stasis in OSM, I build on a multi-perspective approach by employing

qualitative and quantitative methods. The qualitative approach utilizes a qualitative content analysis with a focus on content structuring (Mayring, 2010: 98). The data sources for this approach consist of eight semi-structured interviews, which were conducted via telephone and VoIP. The interview guideline includes questions regarding the general position and activity in the OSM community and the perception of and interaction with software, data structure and software development. The interviews were recorded, transcribed and anonymized. Furthermore, I examined 16 secondary interviews in 'The Book of OSM' which seeks to '[. . .] provide a variety of viewpoints [. . .] of those important to the project' (Coast, 2015: 2). Although the interviews weren't conducted by myself, the content is valuable since the data is still '[. . .] slightly raw' (Coast, 2015: 2). Additionally, I searched and examined OSM mailing lists and the wider environment of community communication, online project documentation and OSM conference data for software-related discussions and issues (see Table 1). The evaluation of the qualitative data is based on eight key themes which I derived from the theoretical framework and contextual focus. After a first reading of the data, I further differentiated them into more specific categories (see Table 2). In a second reading, I examined the data to assign fragments of data to according categories. These data informed my understanding of my theoretical framework and contextual focus from a qualitative perspective.

The quantitative approach builds on an evaluation of source code repositories of OSM (see Table 1). I aggregated the code revision per month and person to illustrate the overall development of OSM software over time and the impact of individuals. Furthermore, I utilize well-established concepts of the computer sciences to describe the relation of actors with software artifacts.

Legacy software in OpenStreetMap – The data structure

The objective of the following section is to address the perception of technological stasis in OSM: I will illustrate that some members of the community perceive a stagnating data structure while others neglect it, highlighting their contradicting assessment of the same issue. Since the founding of OSM in 2004, the project developed a comprehensive software-infrastructure. But at its core, it builds on principles and mechanisms which are mainly referred to as web 2.0 technologies (Haklay et al., 2008: 2026–2029). At the very beginning of the project in 2004 and 2005, OSM merely consisted of a database,[3] a mailing list as a main communication medium,[4] a very simple editor for entering geodata,[5] and a website which connected and hosted the infrastructure and services.[6] In the following years, various editors, mailing lists and forums with sundry foci were added. However, the core software of OSM is the website and the database, as all other tools rely on their structure and services. Figure 1 illustrates not only the complex software infrastructure of OSM, but also the centrality of the database for other system components.

Technological stasis

The software infrastructure is growing and stagnating at the same time: while new parts are constantly added to the project, the software core remains in its original design since 2007. Thus, the ongoing software development in OSM is characterized by new additional features and tools, which mostly access the old core of OSM via the database or the application programming interface (API). The design of the API reflects the design of the database: if the structure of database changes, the API must change as well to make new database features available. Though, as Table 3

Table 1. Data sources.

Data source	Description
Online project documentation	All publicly available information such as wikis, OSM websites, etc.
OSM conference data	Recorded videos and documentations of OSM-related conferences (SOTM, FOSSGIS)
Interviews	Total number of evaluated interviews: 24
Semi-structured interviews (average length: one hour)	8 interviews conducted, selected to cover central actors (founding members, OSM board members, active code contributors)
Secondary interviews	16 interviews in 'The Book of OSM' by Steve Coast (2015)
Mailing lists	Focus on mailing lists 'talk', 'dev', 'osmf-talk', 'osmf-announce', 'rails-dev' and 'rebuild'
Source code repositories	OSM SVN repository (https://svn.openstreetmap.org)
	OSM Github repositories (https://github.com/openstreetmap/)

Table 2. Key themes of the qualitative content analysis.

Theoretical framework: The social construction of technology		
Relevant social groups	Mapper	
	Geospatial professional	
	Developer	
Technological frames	GIS frame	
	Web 2.0 frame	Good enough is perfect
		Simplest useful thing
'Rules of play'	Decision-making	
	Driving forces	
Power to exclude	First level exclusion	
	Second level exclusion	
Contextual focus: legacy software in OSM – the data structure		
Technological stasis/legacy software		
Attempts/suggestions for change		
Obstacles for change		
Contradicting voices		

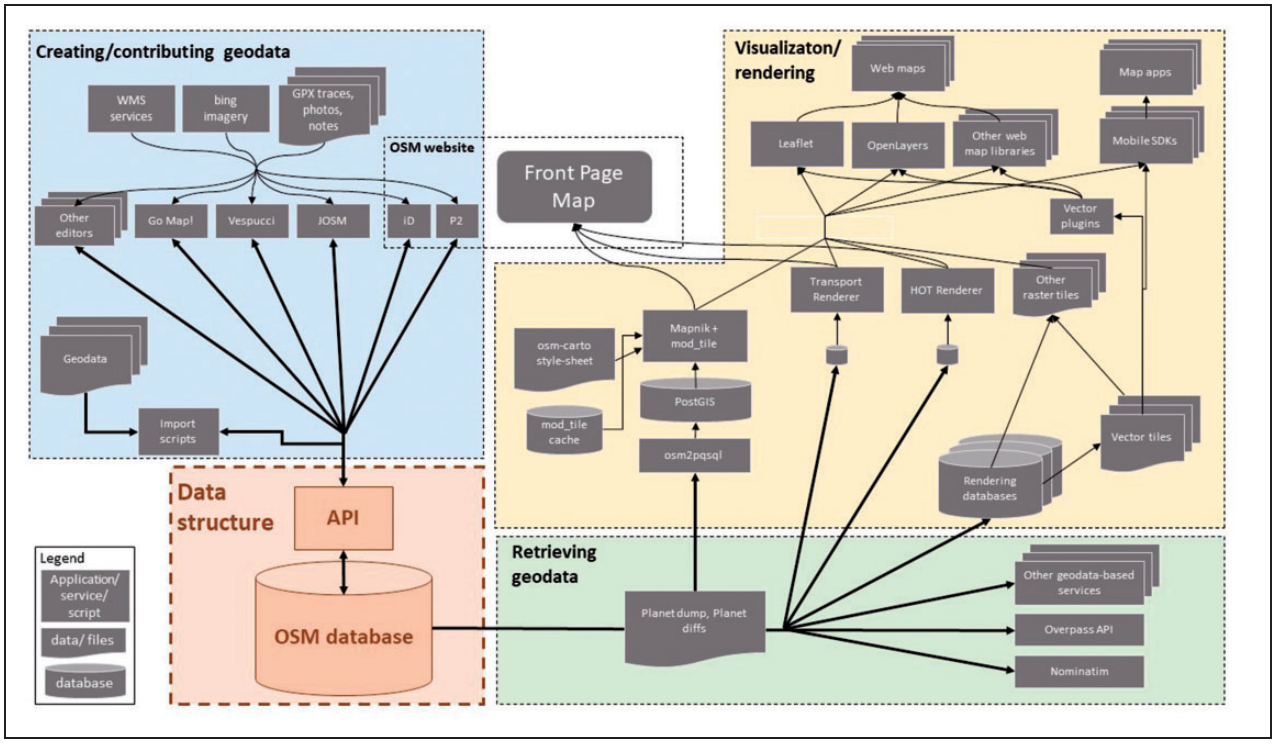

Figure 1. OSM system components (OpenStreetMap contributors, 2017).

shows, the API, as well as the strongly related database itself, were not significantly modernized, reengineered or adapted to new needs since the release of API version 0.5 in October 2007 (the API was last amended in April 2009, however this release included only minor changes). This lack of change becomes particularly obvious when looking at the OSM data structure, respectively, its data-model. From 2004 to 2007, the data-model underwent several changes: between version 0.3 in March 2006 and 0.5 in October 2007, major modifications were carried out. However, since the last documented API change to version 0.6 in April 2009 with minor adjustments, the data-model remains unmodified. This development is – to some

Table 3. API changes in OSM.

	Prior to API 0.3	API 0.3	API 0.4	API 0.5	API 0.6
Date		Mar-06	Apr-07	Oct-07	Apr-09
Topological elements	Raw gpx data	Node, way, segment (temporary area)	No changes on the data-model	Segments removed	
Logical elements				Relation	
Metadata	Latitude, longitude, element ID	Timestamp, tag (key-value pair), visible	Username	Version	Changesets, user ID

degree – reflected in Figure 2: the number of code revisions in OSM's SVN repository peaked concurrently with the API changes; since then, the number of code revision is declining.[7]

Attempts/suggestions for change

With OSM evolving to one of the most prominent and successful volunteered geographic information (VGI) projects, many users placed increasing demands – also regarding the data structure. For example, a founding member noted that 'most people have agreed that [they] want [an area feature]' (IP2) – a feature of the data structure – as the current solution of creating area features is criticized as 'twice as complex as it needs to be, leaves room for inconsistencies (key placement), and overall is difficult for a new contributor to grasp'.[8] This perception is reflected on several wiki[9] pages and threads on the mailing lists, which are dedicated to ideas and suggestions for the further development of the data structure.[10] This is an ongoing debate, as a blogpost in 2018 illustrates:

> 'As an example, in 2012 there have been several proposals made to create a new datatype called an area that would greatly simplify the representation of certain types of geographic features. Despite this and the offer of technical help, the project has not made any significant progress on this or other important technical issues'[11]

Another example is the criticism by several users on the database design that it did not keep pace with the general technological progress.[12] This also touches the data structure and data-model, as the database is still designed in a monolithic architecture and the data exchange works with XML format, which doesn't meet the current state of the art. In summary, several users criticized the deficiencies regarding the database, data structure and data-model, as it forces users and software applications in the wider arena of OSM to adapt and to implement complex and cumbersome workarounds.

Obstacles for change

Why then was the data-model not changed over the last decade? Following the argumentation of the technical perspective on technological stasis, this might be because of the difficulties of changing it: bad maintenance, 'not-quite-right' (Cunningham, 1992: 279) code which make things 'harder to change than it should be [...]' (Eick et al., 2001: 100) and dependencies to countless software applications would cause technological stasis. Several voices of the OSM community – especially of contributors involved in the software development – confirm this perspective: '[...] forcing 50 + applications to change would be inappropriate' (OSM software developer, mailing list, 08 June 2009). An interviewee expressed the concern that

> '[...] if you want to get rid of one of these timeworn relics, you have to change countless subprojects at the same time [...] how do we manage that the entire software won't stop working?' (IP2)

Regarding the software infrastructure in OSM, it is indeed striking how many software applications are dependent on the data-model: besides more than 30 different editors,[13] there are very complex and cumbersome visualization processes (see Figure 1) for the OSM mapping services and more than 80 other OSM-based services.[14] It appears that the data-model qualifies as '[...] burdensome legacy from the past' (Parnas, 1994: 279). Thus the stasis of the data-model might 'imped[e] the further development of the system[...]' (Parnas, 1994: 279).

Contradicting voices

However, digging deeper in the discussion concerning technical changes, some of the interviews revealed a dissenting perspective on the stasis of the data-model.

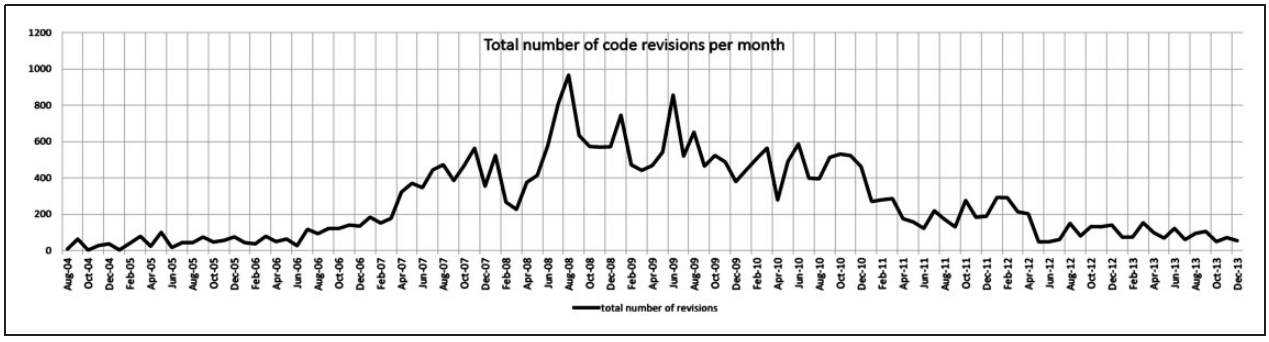

Figure 2. Code revisions per month in OSM's SVN repository (https://svn.openstreetmap.org).

For example, a member of the OSM board replied to the assessment, that changing 50+ application would be inappropriate: '[w]hy, we're doing it all the time;-)' (Member of OSM board, mailing list, 08 June 2009). A founding member of OSM stated that '[t]he technical constrains here are not very big' (IP4), referring to changing elemental parts of the database.

These contradicting voices on changing the data-model call the technical perspective on technological stasis into question. Is it solely the technical design, the strong entanglement with the overall system and the maintenance of the data-model that causes the lack of change? In the following, I want to argue that it is predominantly a question of social context and power when confronted with stagnation and aging software. Thus, I will reframe the technical stasis with the concept of technological frames. In doing so, I will approach this issue from a social-constructivist perspective: the SCOT.

The social construction of technology

Theory

In the 1980s, the perspective of the SCOT emerged from STS. It marks the 'turn to technology' (Woolgar, 1991) – the growing interest of STS researchers in examining technology. 'The Social Shaping of Technology' by MacKenzie and Waycman (1999) as well as 'The Social Construction of Technological Systems' by Bijker et al. (1987) are two of the pioneering publications in this field of study, evolving from the criticism on technological determinism, arguing that human action shapes technology. Starting from that 'turn to technology', several closely related theories emerged – among others the Actor Network Theory (Latour, 1988, 1994) or the Large Technical System approach (Hughes, 1987). However, a distinctive characteristic of SCOT is that it provides not only theoretical concepts, but also a strong methodology to analyse processes of technological development. In the following, I will briefly set out central principles of SCOT, then address major criticism on SCOT regarding its lack of structure and finally illustrate its understanding of technological stasis.

The focal point of SCOT is the identification and definition of relevant social groups. Relevant social groups are defined in their relation to an artifact: 'The key requirement is that all members of a certain social group share the same set of meanings, attached to a specific artifact' (Pinch and Bijker, 2012: 23). As an artifact is socially constructed, each relevant social group has its own perspective that constitutes their artifact with its own meanings and characteristics. By deconstructing the artifact – thus by pointing out the different meanings assigned to the same artifact – interpretative flexibility is demonstrated. According to Bijker, '[o]nce an artifact has been deconstructed into different artifacts, it is clear what has to be explained: how these different artifacts develop; whether, for example, one of them peters out while the other becomes dominant' (2002: 77).

However, Klein and Kleinman (2002: 29) complain that this approach is agency-centred when they state: 'Much criticism concerns an excessive emphasis on agency and neglect of structure'. They observe that '[i]mplicitly, SCOT assumes that groups are equal and that all relevant social groups are present in the design process. This fails to adequately attend to power asymmetry between groups' (2002: 30). In making this comment, they urge us to rethink why in the processes of closure and stabilization the meaning of an artifact of one specific social group dominates, while meanings of other social groups are forced out. According to them, there is a lack of '[. . .] structural features of social life' (2002: 31). Their point is that power and power asymmetries are undertheorized and there is a need of further investigation of structural features in SCOT.

As a solution, Klein and Kleinman suggest a '[. . .] discussion of groups' capacity or power. What enables

one group's interpretation to be embodied in the artifact?' (2002: 34). This capacity of groups or actors may affect the result of the construction process of technology. They further argue 'that group capacity should be understood in broadly organizational or structural terms, as it is such factors that fundamentally shape group capacity' (2002: 34). Hence groups are embedded in 'historically established structures' (2002: 35), defining '[. . .] specific formal and informal, explicit and implicit ''rules of play''' (Kleinman, 1998: 289). Furthermore, they suggest a focus on structural factors influencing the participation of groups to the design process. In order to capture these factors, they propose three concepts: design forum, rules of access and the power to exclude (2002: 37). That is, a description of the setting in which design occurs, the context of accessibility to design and decision-making, and power asymmetries regarding the capacity of using these structures as an unilateral advantage. In sum then, to compensate the lack of structure on SCOT, we must examine the characteristics and context of the design process, its 'rules of play' and how to participate in it.

To this point, I established the major characteristics of SCOT – or how SCOT describes technological development – as well as criticism on it due to its lack of structure. In the following, I will address SCOT's understanding of technological stasis. Bijker claims that it is necessary to understand the stability of technological artifacts to build a theory of technical change. Thus a strong aspect of SCOT is to explain '[. . .] how constancy and continuity exists in history' (Bijker, 2002: 14). He defines the resistance of artifacts as obduracy – a concept that describes why '[s]ome artifacts are [. . .] harder to get around and to change, than others' (Bijker, 2002: 4). Thus explaining obduracy is central for understanding technological change.

Bijker introduces the notion of technological frames in order to describe and explain obduracy: 'it is used by the analyses to order data and to facilitate the interpretation of the interactions within a relevant social group' (Bijker, 2002: 125). Hommels summarizes technological frames as '[. . .] conceptions of technology's obduracy that focus on the role and strategies of actors involved in the design of technological artifacts, and the constraints posed by the sociotechnical frameworks within which they operate will be addressed in particular' and that they manifest in '[. . .] fixed ways of thinking and interacting' or '[. . .]meanings and values they attribute to technologies' (2005: 331). However, '[t]echnological frames are not purely cognitive, but also comprise social and material elements' (Bijker, 2002: 126).

How do technological frames constrain specific actors? Bijker defines the involvement into technological frames as inclusion: '[t]he degree of inclusion

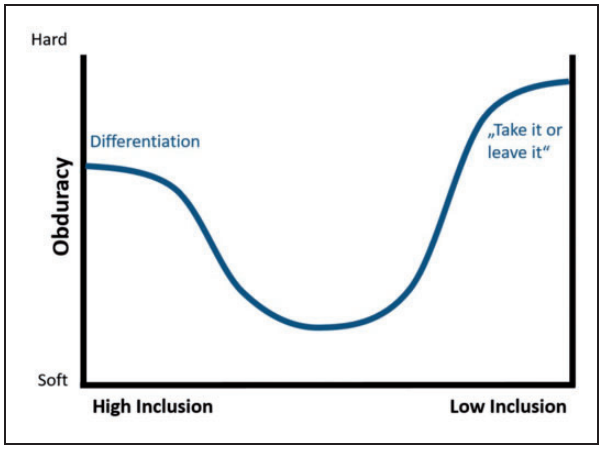

Figure 3. Inclusion into technological frames, degree and characteristics of obduracy (Bijker, 2002: 285).

of an actor in a technological frame indicates to what extent the actor's interactions are structured by that technological frame' (Bijker, 2002: 143). He examined the notions of technological frames at the example of Bakelite: 'Bakelite engineers knew how many variables had to be controlled to produce specific forms of the artefact, they knew how many different shades could be produced, and they knew how tricky the manufacturing process could be. As shown in Figure 3, an artifact was then ''hard'' for these actors with high inclusion, but in the very specific sense of enabling and constraining interaction and thinking' (Bijker, 2002: 284). In his example, he shows how engineers with a high degree of inclusion into the technological frame of Bakelite production are not able to think 'outside the box' of Bakelite engineering. Thus, they know how complicated the process of Bakelite-production is and they also knew the difficulties of changing it.

However, a low degree of inclusion into a technological frame does not mean fewer constraints (see Figure 3). Interactions of actors with low inclusion are not strictly structured by the technological frames, but the sociotechnical ensemble of that frame '[. . .] does have a relatively undifferentiated, monolithic meaning' (Bijker, 2002: 284). Accordingly, an artefact presents a 'take it or leave it' decision: on the one hand, the actor is not able to modify it if he 'takes' it, on the other hand he can 'go on quite well' if he 'leaves' it. Thus there is no flexibility or differentiated insight into the technology (Bijker, 2002: 284).

In summary, the concept of technological frames provides an explanatory framework for technical stasis and their diverse manifestations: the diverging degree of inclusion determines the perception of socio-technical and cognitive elements and thus the sundry perceived obduracies of the very same.

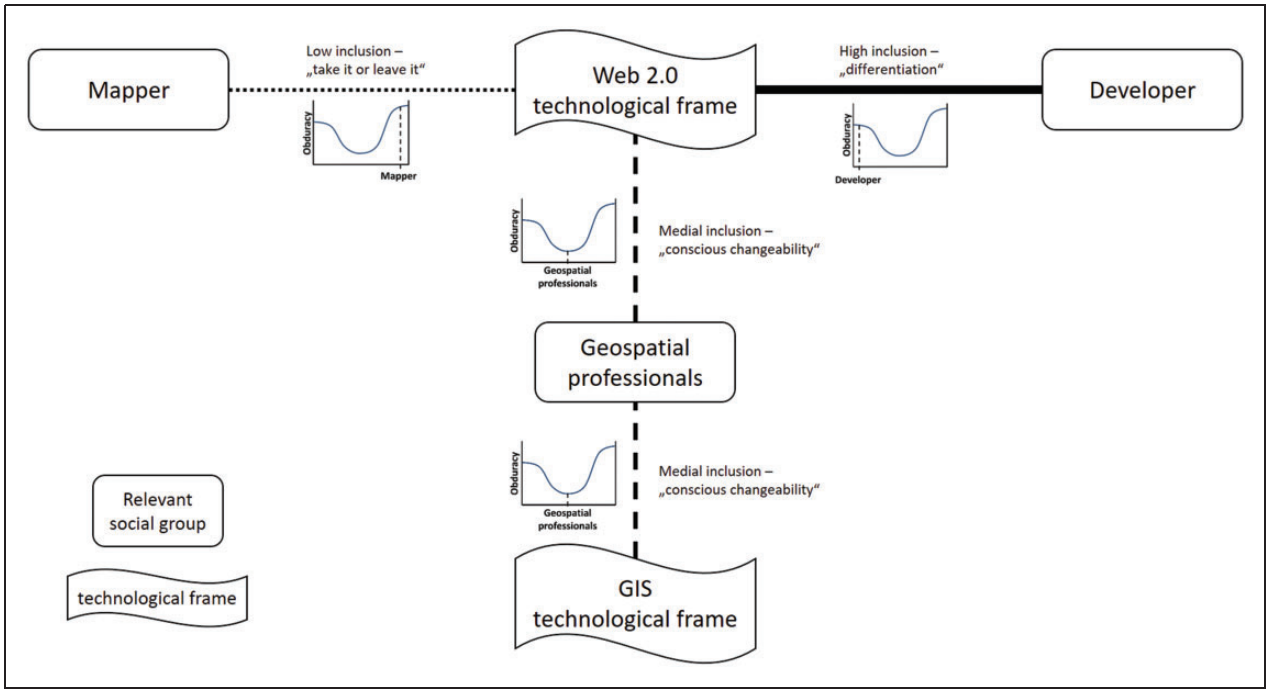

Figure 4. Relation between technological frames, relevant social groups, their degree of inclusion and the corresponding obduracy of technology.

Furthermore, the criticism of Klein and Kleinman (2002) urges us to seek for the groups who have the power and structural advantages not only of changing technology, but also of determining continuity and stasis of technology.

The social construction of technological stasis in OSM

The empirical application of SCOT (see Figure 4) is structured as follows: first, I address the relevant social groups in relation to the data-model in OSM. I subsequently describe their technological frames and their degree of inclusion. Finally, I lay out the relation between the relevant social groups, their technological frames and the technological artefact.

The identification and definition of relevant social groups is based on the methodological suggestion of Bijker as they are defined in their relation to an artifact – in this case the data-model/ structure of OSM. Thus, the distinction of groups is based on whether they '[. . .] share the same set of meanings attached to [. . .]' the data-model (Bijker, 2002: 23). Therefore, there is not necessarily an actual joint appearance in the community, strong social relations within the group or general homogeneity of social characteristics which defines relevant social groups; it is rather a shared understanding of a specific technological artifact. But how can we determine the relation to the data structure of OSM? My empirical analysis for identifying and delineating the relevant social groups utilizes three approaches: the first approach examines the general relation and interaction of community members and the OSM project, respectively, geodata: are OSM contributors predominantly involved in creating geodata and thus active in the 'open data' part of the project or do they rather 'use' geodata? Or are they solely involved in technical or infrastructural tasks – the 'open source' part of OSM? Second, based on the previous delineation, I set out the prevalent field of action of each group in reference to the software tiers of OSM's software infrastructure. Software tiers, tier-architecture, or 'the stack' (Solomon, 2013) is a common concept in the computer science, dividing software in different number of tiers to provide a model for modularity and interchangeability; the most common concept is the three-tier architecture (Fischer and Hofer, 2008: 732). Although OSM's software infrastructure wasn't strictly designed under the principles of the three-tier architecture – it rather emerged to its current design – this concept is beneficial for understanding the diverging technological centrality, complexity and plasticity of system components (see Figure 5). Straube notes that '[. . .] the stack [respectively tier-architecture – author's note] clearly establishes hierarchy: each layer depends on the one below to function, and adds a dimension of

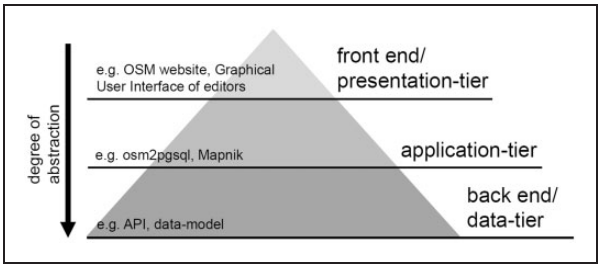

Figure 5. Software tiers.

abstraction that is in turn the base for the layer above' (2016: 6). Upper tiers – often referred to as 'front end' – are close to the end-user and little abstracted, with few interdependencies to the overall system. Lower tiers are subject to greater abstraction, very close to the database and, consequently, more complex to handle. In the case of OSM, this concept is beneficial as it highlights the centrality of the data structure and thus how other software parts depend on them. It is also useful for illustrating to what extent contributors encounter the technological infrastructure of OSM and therefore describes their field of action in reference to OSM software. The last approach aims at describing the perception of the data structure by each group: in which way is the artefact of the data structure present within a group?

My empirical findings suggest three relevant groups: I will refer to them as 'mappers', 'geospatial professionals' and 'developers'. Although there are certainly more social groups involved, I focus on the above mentioned as they are dominating the discussions and represent the major contradicting as well as competing perspectives on the technological artefact of the data structure. It is also important to note that the groups are not static and to some degree overlapping, thus their description focuses on ideal-typical members of a group. Furthermore, I will address two distinct technological frames with respect to the same technological category – the data structure.

Mapper. The social group of mappers consists of a large number of geodata contributors who are not involved in the software development process. This becomes obvious when comparing the number of code contributors with the number of geodata contributors: while from 2004 to 2008 a total of only 155 persons contributed to the software infrastructure of OSM (see Figure 6), there were more than 10,000 active geodata contributors per month in 2008.[15] This discrepancy illustrates that there is a large group of geodata contributors, who are not involved in the software development. Most of the studies which focus on the OSM community are in fact focusing on the latter. Thus this group was object for extensive research on motivation (Budhathoki and Haythornthwaite, 2013; Haklay and Weber, 2008). Members of this group joined the project to contribute geodata to the open data part of the project. Mappers differ from other relevant social groups by their field of action being predominantly within the first software tier: the presentation tier. They interact with the software infrastructure on the front end, respectively, with graphical user interfaces. In general, their relation to the artifact is determined by indifference: since they remain on the presentation tier, the highly abstracted data structure of OSM is hard to identify as a distinctive element. However, this group is hardly homogeneous: depending on their involvement in OSM, very active mappers can completely exploit the functionality of software tools and sometimes even take advantage of them. Thus, their field of action may sometimes reach beyond the presentation tier into the application tier. However, it will stay outside the data tier and therefore their relation to the data structure remains mainly characterized by indifference.

Data users/Geospatial professionals. The geospatial professionals or data users represent a group which is mainly interested in using OSM geodata; furthermore, they often have conventional geo-informational expertise or are experienced in working with GIS. One contributor stated that '[t]here was a wider community I guess of geospatial experts and geospatial amateurs who were mainly interested in the map but who came with a whole bunch of knowledge of the way other geospatial systems worked [. . .]' (IP3). Their meaning attached to the data structure is based on their professional expertise and thus aversion towards the 'chaotic', 'dirty' and unstructured geodata of OSM (Carden, in Coast, 2015: 25). A common perspective on OSM data was that '[. . .] everyone could work on it, everyone could change it; there is no authority which checks the data and approves it with a quality seal' (IP7). Additionally, the group of geospatial professionals mainly appears as data consumers, meaning their interaction with OSM is shaped by tools for visualizing/rendering and retrieving geodata (see Figure 1). Thus, their field of actions lies primarily within the application tier.

GIS frame. The first technological frame – the GIS frame – is characterized by conventional approaches for structuring geodata in the context of cartography, geoinformatics or other geoinformation-based sciences and professions. This frame aims for an optimized way of digitally recording, manipulating, storing, organizing, modelling, analysing and representing spatial data. 'When you start a career in GIS, it's not so much about making maps but it's a lot about doing stuff with maps that other people make and you

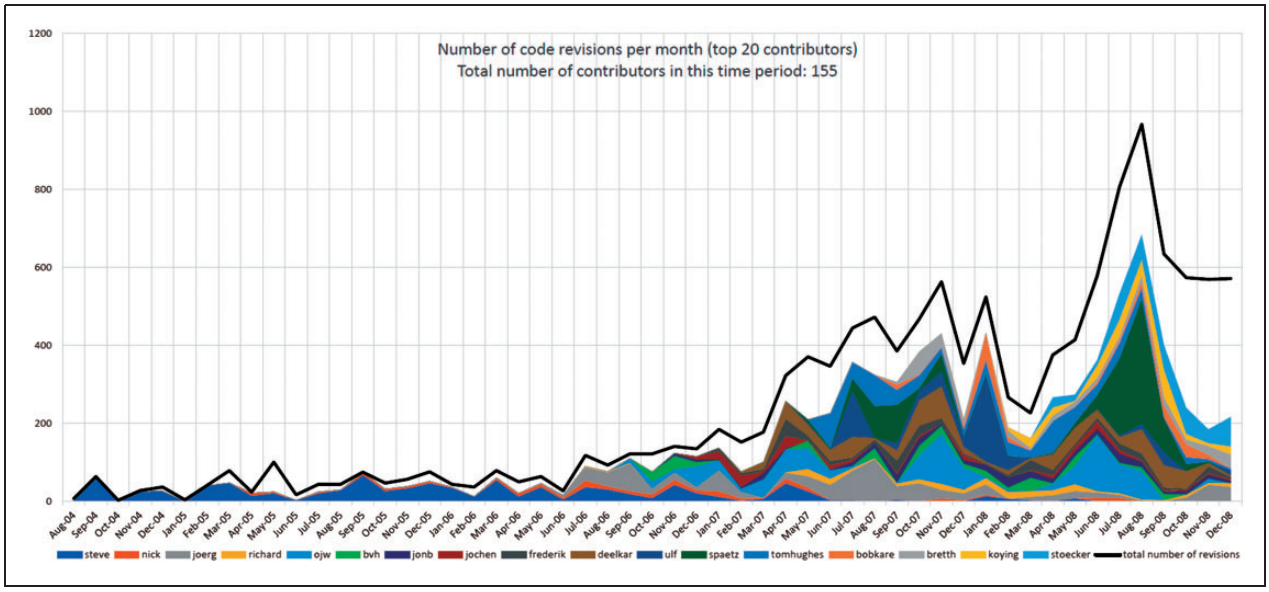

Figure 6. Code revisions per month and person in OSM's SVN repository (https://svn.openstreetmap.org).

never really get exposed to the making part' (van Exel, in Coast, 2015: 100). Thinking and interacting in this frame is therefore shaped by this knowledge: rigid ontologies, strong data structure, organized in a strict hierarchy often appear imperative. Thus, in professional GIS contexts there are well thought out data structures which are optimized in this regard: '[e]verything needs to be very structured, professional, and official, with a lot of pedigree in order for it to be GIS. It is a continuation of that whole professionalization, accreditation culture meme that surrounds GIS' (Gorman, in Coast, 2015: 151).

Developers. The group of developers is characterized by a professional background in software development, system administration or similar fields. Some of them are also active in Linux user-groups or involved in other open source projects (for example, see short biographies of Tom Carden, Mikel Maron or Grant Slater in 'The Book of OSM' (Coast, 2015), or personal websites of developers[16]). A founder of the project noted: 'Well, in the very early days, it was essentially open source people. Right, so people who liked Linux, people who liked Wikipedia, those were the two main crowds, because they could see the potential without having any functioning websites or tools, when it was just an idea' (IP4). Their relation to the data structure is characterized by the approach of finding 'the simplest useful thing that works'. However, it is important to note that this simplicity refers to the software design, in the sense that the software is easy to develop; it doesn't necessarily refer to easy-to-use software. Although this group consists mostly of volunteers and is, in comparison to the other two groups, particularly small, they contribute most of the code and are almost exclusively active in the data-tier of OSM. Figure 6 illustrates that until August 2008, when most of OSM's data structure was implemented, only 20 persons consistently contributed about 80% of all revisions. Thus they created the infrastructural backbone and the data-structure within the back end of OSM (see Glasze et al., 2018). For them, OSM is rather an open source project than an open data project.

Web 2.0 frame. The web 2.0 technological frame is defined by the approach of using web 2.0 technologies for creating geodata. Following this approach, the idea was to utilize crowdsourcing to produce a 'wiki world map'. The necessary digital infrastructure was shaped by a very practical approach of software development as the software shouldn't be 'over-engineered'.[17] In this context, terms like 'good enough is perfect' (Carden, in Coast, 2015: 24) or 'simplest useful thing' (IP1, IP2) are frequently used and illustrate a culture of workarounds and technical improvisations. Tom Carden noted that '[we] would rather have a messy system with a low barrier to entry and worry about cleaning it up later than having a perfect system with no data because it's too difficult to contribute' (Coast, 2015: 24).

Relation between technological frames and the relevant social groups. The relation of the group of mappers to the web 2.0 frame is characterized by low inclusion. Their interaction with the project manifests itself mainly in

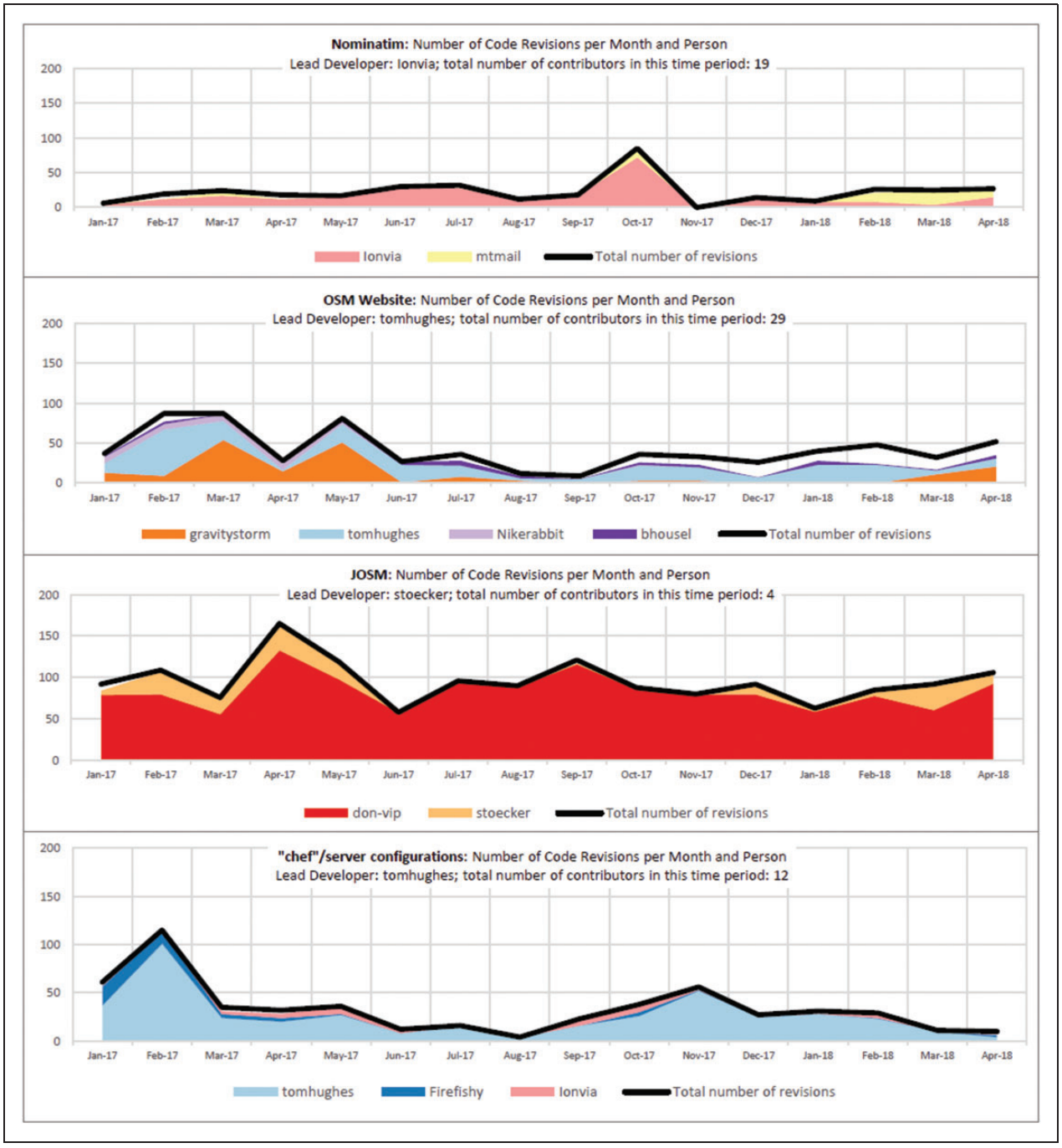

Figure 7. Recent code revisions per month and person of a selection of OSM subproject on Github (https://github.com/openstreetmap/).

crowdsourcing geodata. Thus, in respect to the web 2.0 frame, their relation is characterized by a 'take it or leave it' decision: either they participate in OSM with its current technological infrastructure, or they leave it and carry on. This group has a limited understanding of the project in the sense that it is hard for them to imagine technological alternatives to the system in use. Therefore, they have difficulties in relating to struggles or benefits of changing it; if change happens, they accept it, otherwise they leave: 'from a mapper-perspective, I didn't care about new [software] features. I thought it was fun to map; thus, new features were [always] fun and good' (IP7). In summary, because of their low inclusion, the technological infrastructure of

OSM is very obdurate to them as they are unable to see technical alternatives.

The perception of the group of developers is mainly shaped by their high inclusion into the web 2.0 frame. Thus, advantages of changing the data-model are not very present in their thinking. In their point of view, the data structure is 'good enough' as they can generate maps; their main goal of creating generic geodata to play with is accomplished: 'We [relevant social group of developers] didn't know about mapping quality, we just wanted a map' (IP1). In their perspective, modernizing or reengineering the data structure has no point. On the contrary: for them, the struggle of changing it is even disproportionally high, as they have a much more differentiated perspective on the technical details: due to their deep technical understanding of OSM's software infrastructure and their contributions to its development, they see all the complex adjustments and difficulties involved. It was this context, which provoked comments such as '[. . .] forcing 50+ applications to change would be inappropriate' (OSM software developer, mailing list, 08 June 2009). Furthermore, in their perception, there is a reasonable alternative: extending the existing workarounds by using the tagging system. For instance, one core member noted in response to a requested new software-feature that 'all of it can already be done client-side using appropriate tags' (OSM software developer, mailing list, 08 June 2009). For them, this approach is well-proven and established in the community. Hence, in their perception, the difficulties of changing the data structure appear very complex, and at the same time, the benefits of doing so seem very little. It follows, then, that the data structure is very obdurate for the group of developers.

The thinking and interacting of the group of geospatial professionals is equally shaped by their medial inclusion in the GIS frame and web 2.0 frame. Despite OSM being a very attractive data source, as it provides a huge amount of free and open geodata ready to be used, their inclusion into the GIS frame influences their perception of OSM geodata as having a weak data structure and being inconvenient and cumbersome for conventional GIS approaches. These problems could be solved with a modernization or reengineering of the data structure in OSM. Thus, they attacked social tagging as an annotation system and argued that they needed a 'proper structure to get a useful map' (IP1). Therefore, 'if you want a high-quality map, you really need to have control over the process of making the map and you need a good structure, making sure everyone understand what the structure means, how it should be used' (IP1). Due to their equally high inclusion into the web 2.0 frame, this group is aware of the general advantages of OSM being a crowdsourcing project and the technical consequences of web 2.0 technologies. They see the technical difficulties of changing the data structure of such a complex software infrastructure; however, they also see the benefits of doing so. In addition, they are well-aware of technical alternatives, as there are established solutions in the GIS frame. Since the group of geospatial professionals is medial included in both, the GIS frame and the web 2.0 frame, the data structure is the least obdurate to them. Especially their relation to the web 2.0 frame is characterized by a 'conscious changeability', as they see possible alternatives and the benefits of change.

Power, structure and driving forces

After setting out the relevant social groups, technological frames and their relations, the next step is to address power and structure as suggested by the criticism of Klein and Kleinman (2002). In the following, I will examine OSM's design forum, its 'rules of play' and power to exclude in relation to the previous findings.

Design forum. In OSM, there are several settings in which software design is negotiated by the community: the main communication media and the software development infrastructure. These settings differ in their technical entry barriers: while it is relatively easy to sign up for a mailing list[18] or to participate in a forum,[19] the usage of IRC chats,[20] OSM trac[21] or SVN repositories[22] makes a more sophisticated understanding of web technologies necessary. Furthermore, we can distinguish these settings from being discussion-oriented or implementation-oriented. For one thing, previous findings by Glasze et al. suggest that the main communication media – the mailing lists in which most of the community participates – are not the setting where design occurs, respectively, where design decisions are made. They argue that '[. . .] the mailing list is a platform for discussing ideas, explaining implementations and thus legitimizing decisions that were made elsewhere' – thus mailing lists are clearly discussion-oriented (Glasze et al., 2018). On the other side, due to the software-related do-ocratic decision-making in OSM (Glasze et al., 2018), design decisions are linked to their implementation – the process of deciding upon a specific software design and implementing it. Accordingly, the setting in which design decisions are made is also the same in which they are implemented. That leaves the actual design forum to the software development infrastructure, such as the SVN repository, which was predominantly used in the first four to five years, and OSM's most common software repository today: Github.[23] As a result, for accessing the design forum, a specific technical knowledge is

necessary, rising the technical entry barrier. Admittedly, there are also settings in between those extremes with medial technical entry barriers, like OSM trac, a bug tracking system: they enable users to describe and discuss problems with the current software or even make own suggestions for future features; OSM trac[24] enables developers and non-developers to closely define feature suggestions or software problems and discuss those issues at the same time. However, the final decision is still made by a developer within a repository.

Rules of play/rules of access. The 'rules of play' and rules of access in OSM comprise decision-making and driving forces. As mentioned before, the decision-making in OSM is based on do-ocracy and thus favours those who have the technical skills, understanding and time for carrying out changes in the software. Although this decision-making was openly criticized by some members of the community[25] as undemocratic, this mechanism is still commonly accepted by the community.

The driving forces of community members to contribute and participate in OSM are essential to understand the technological stasis. The first driving force – social costs – refers to the great importance of the community. Today, each attempt for change poses a potential threat to the community in the sense that community members who don't like the change might turn away from the project. In the first instance, it doesn't matter if that attempt for change is technical, social or legal in nature. Referring to technical change regarding one of OSM's central editor-tools, Haklay noted that '[i]t's the social aspect of the way everyone danced around everyone else in the change from Potlatch to iD [...] nobody wanted to offend [the lead developer of Potlatch] in any case [...] and those sorts of things appearing again and again in the history of OpenStreetMap' (Coast, 2015: 175). However, in the beginning of OSM, the fear of damaging the community was rather a principle of community-building: with the necessity of building a large and functioning community, each design decision had to be optimized in this regard. A good example for this principle is the implementation of folksonomies, which was clearly a decision in favour of community-building (Glasze et al., 2018). Thus, the driving force of social costs has a long history in OSM and still plays a significant role in each major decision.

Another driving force is the treatment of companies and monetization with distrust and suspicion. Steven Coast experienced this suspicion when he founded CloudMade. He said: 'When I started CloudMade, I was evil. They literally use the word evil. It's not shorthand for something. And when we got venture capital, I was evil. When I went to Microsoft, I was evil. So I mean, it's just levels of evilness' (Coast, 2015: 91). Randy Meech, the CEO of Mapzen, agrees when he notes that '[t]here is definitely like a strain of suspicion about corporations in general [...] it's a concern about taking all the data and monetize it somehow and not giving back' (Coast, 2015: 227).

Fun and commitment is another key driving force for contributors. OSM is mainly based on volunteers; thus, the clear majority of tasks is made in a voluntary effort. Although having spare time is a necessary precondition for this driving force, it is ultimately fun and commitment that motivates contributors to perform those tasks. A key developer of the JOSM editor stated: 'It's basically like those who enjoy it, those will participate, and the others won't [...] it's just fun to code' (IP5). Thus, if a developer considers a suggestion for a feature as 'a good idea and fun', it's probable that he or she will implement it. These statements are in line with the general perception of motivation in open source projects. Important figures of the open source community stated that fun is a major driving force (Bitzer et al., 2007: 163; Torvalds and By-Diamond, 2001).

At this point it is important to note that money as driving force – in the sense of getting paid for contributing code – is not a major issue for core parts of OSM's software infrastructure, especially not in the context of the data structure. This is because paid developers appear relatively late in the history of OSM (CloudMade, the first company which interfered in the software development of OSM was founded in 2007, when major parts of the data structure were already implemented). Additionally, an analysis of the SVN repository from August 2004 to December 2008 (see Figure 6) reveals that developers with potential monetary interests (i.e. steve, frederik, jochen), who were involved in the development of central software parts, make up only a minor part of the overall code revisions. Finally, except for CloudMade, most companies which were involved in software development did so on the upper software tiers of OSM (for example Mapbox and the iD editor, Geofabrik and geodata processing or Mapzen and providing Web Mapping Services).

Another, more implicit driving force falls in the broad range of power and control. Andy Robinson, a former member of the board of the OpenStreetMap Foundation (OSMF) stated that '[t]here are some people who get themselves into positions that give them a degree of ability – power if you want to call it in politics' (Coast, 2015). Referring to his power in software development, a lead developer stated: 'And yet, I have to say, I reserve to be – in this respect – a dictator, if there's something I absolutely don't like' (IP5).

Therefore, there is an interest in gaining and retaining power, or to put it differently, a fear of losing power and control.

In summary, the 'rules of play' in OSM can be characterized by do-ocratic decision-making and the driving forces social cost, suspicion of monetization, fun and commitment, and rather implicitly, gaining and retaining power and control. Money or profit as incentive is not a relevant factor in respect to the data structure. Thus, software development in OSM is subject to very specific framework conditions. This is important since SCOT builds on empirical evidence that is based on a capitalist environment and thus money and profit as a key driving force. As a result, the power to exclude within this system must be adjusted to these specific framework conditions.

Power to exclude. Considering the previous findings, I want to argue that there are two levels of 'power to exclude'. The first refers to exclusion on the level of relevant social groups. The SVN repositories of OSM have a high technical entry barrier and do-ocracy enables individuals with specific technical skills and understanding to just implement their desired software design. Furthermore, the data structure lies within the data-tier, outside of the field of action of several groups. Thus, a major reason of technical stasis in OSM is that the relevant social groups differ in their access to relevant resources: the group of developers have the deepest technical understanding, in the sense that they can develop or manipulate highly abstracted and complex software in the data-tier; geospatial professionals have a technical understanding which predominantly limits them to the application-tier and the group of mappers remain on an application level of the software. This asymmetric access in combination with each group's predominant field of action explains the limitation of the design process to the perspective of the group of developers, and thus to the web 2.0 frame. Therefore, the data structure is highly obdurate to the only group with the necessary specific technical understanding to change it: the developers. It is their perception of the data structure that determines software development in the data-tier. If they don't see the necessity and benefits of change, nothing will happen. Furthermore, this implies that many ideas and initiatives for change – although maybe seen as vital by a majority of the community – don't even clear the hurdle of being considered as a necessary, beneficial and legitimate suggestion by the group of developers.

The second level refers to 'power to exclude' within the group of developers, on an individual level. Within the group of developers, power is decentralized; a pattern which was already discussed by Haklay and Coast (Coast, 2015). Haklay argues that '[...] it's a lot of people that are ''kings in the castle'' of OSM [...] if someone can be the person with a log-in to the whole system and the root access and someone else can be the root access to the database, that enables two people to be the technical side and to do different things and be their own king in the castle' (Coast, 2015). Figure 7 illustrates this pattern: the software infrastructure is divided into small subprojects with small communities, who do the daily work of maintaining the software. Each subproject has a system administrator with root access who is often the lead developer as well. In other words, the software infrastructure of OSM is decentralized into smaller entities – such as the website, the server-software, editors or map-styles – with individuals who 'hold the keys' in the sense that they have the root access and can determine changes.

How does decentralized power affect software development in OSM? To understand the effects, it is necessary to describe the way in which those individuals can wield their power: since there are predominantly intrinsic motivators for contributing, it is hardly possible to force other developers to carry out changes. Thus, lead developers in gatekeeping positions can either keep control over the software design and implement changes on their own, or give up control and accept code contributions of other developers with their do-ocraticly implemented design decisions. As this threatens another driving force in OSM – retaining power and control – they may try to prevent change they don't feel confident about. Therefore, the second level exclusion potentially paralysis the group of developers from the inside, impeding internal attempts for change as well.

In summary, significant technical change in OSM – such as reengineering parts of the data-tier – is not impossible, but due to the circumstances, it is highly unlikely. Any new idea of technical change must clear the hurdle for being considered as legitimate and beneficial suggestion by the group of developers. Additionally, there are several driving forces within the community – social coasts, fun and commitment, and gaining and retaining power and control – which oppose change on a second level.

Conclusion and discussion

Although OSM is a very heterogeneous project with a diverse community, its software infrastructure shares significant similarities with OSS projects. Back end-related software development in OSM is, inter alia, driven by intrinsic motivators such as fun and enjoyment, while monetary benefits play a less significant role. Also, the structure of power, control and governance appear to be in line with OSS findings, as there is a core team of developer dominating software

development. Regarding evolutionary dynamics of OSS, OSM is in a growing stage, which contrasts its technological stasis. This phenomenon of stagnating software is a well-known issue in the computer sciences, however existing literature addresses stagnating software often from a technic-centred or problem-solving perspective.

By framing this issue with a social-constructivist perspective, I focused on social context and power. The SCOT, informed by Klein and Kleinman, is a useful framework for understanding technological change and stasis and how specific expertise can influence the perception of technological possibilities. Klein and Kleinman's critique is beneficial for identifying crucial points in the process of technological development and how asymmetric access to resources can exclude groups from it. My results indicate that the technological stasis in OSM is rooted in a combination of the diverging perception and thus obduracy of the data structure and the access to the design process. While the largest group perceives the data structure as very obdurate because of their low inclusion into the web 2.0 frame, it is the same case for the smallest group, which is dominating the software development, though due to their high inclusion. The main group advocating change can't access the design process, as it lies beyond their field of action. I described these results in the first level exclusion. Since SCOT is focusing on a group level, individuals in gatekeeping positions with a major impact on the overall system are easily overlooked. Thus, I introduced a second level in which I focused on individuals within the group of developers. My findings suggest that there are several developers in key positions who further paralyse the group of developers from the inside.

But how did these levels of exclusions emerge in the first place? One of the major reasons lies in the do-ocratic decision-making, as it excludes a significant part of the community and thus their ideas for change. Furthermore, OSM's driving forces for (code) contributions cause an adverse environment for fundamental technological change. A combination of suspicion towards external for-profit actors and monetization, a fear of damaging the community, the implicit interest in having power and control and a hobby-like motivation of fun and enjoyment is not an environment for constructive discussions or productive software development.

This leads to another question: is this kind of environment still suitable for a project that has grown so much? Do-ocracy is a beneficial decision-making principle for a young and evolving open source project based on volunteers – it advantages rapid growth since there is no need for votes on specific software designs. But is this still the case for a project with more than four million registered users, who are mostly unable to carry out technological changes and thus are excluded from the design process? Similarly, should such a large project rely – in its core – on the voluntary work of very few people? Or did OSM reached the point, where the stakes are too high for do-ocracy and hobbyists maintaining the main infrastructure? Although it is difficult to find an optimal environment for software development in such a diverse and complex project as OSM, it seems inevitable to include all community members in the design process and take their needs into account.

Acknowledgements

I would like to thank the anonymous reviewers as well as the editors for their helpful suggestions and comments.

Declaration of conflicting interests

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The author was supported by a grant of the Hanns-Seidel-Stiftung and acknowledges support by Deutsche Forschungsgemeinschaft and Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU) within the funding programme Open Access Publishing.

Notes

1. https://wiki.openstreetmap.org/wiki/Stats
2. https://wiki.openstreetmap.org/wiki/Stats
3. The first code for the database was contributed in December 2004.
4. The first mail on the mailing list was sent in October 2004.
5. The first code for an editor was contributed in April 2005.
6. The domain for the website www.OpenStreetMap.org was registered in October 2004.
7. It is notable that the decrease of code revision beginning in 2010 may also partly be caused by the successive change to another source code repository system.
8. https://www.openstreetmap.org/user/ThePromenader/diary/13610
9. For example see http://wiki.openstreetmap.org/wiki/Contributors_functionalities_wishlist, http://wiki.openstreetmap.org/wiki/User:Frederik_Ramm/Ideas_for_API_0.7
10. For example see https://lists.openstreetmap.org/pipermail/dev/2012-July/025256.html, https://lists.openstreetmap.org/pipermail/dev/2011-April/022313.html
11. https://blog.emacsen.net/blog/2018/02/16/osm-is-in-trouble/

12. For example see https://lists.openstreetmap.org/pipermail/dev/2012-July/025256.html, https://lists.openstreetmap.org/pipermail/dev/2008-November/012463.html
13. http://wiki.openstreetmap.org/wiki/Editors
14. http://wiki.openstreetmap.org/wiki/List_of_OSM-based_services
15. https://wiki.openstreetmap.org/wiki/Stats
16. http://www.ostertag.name/JoergOstertag/index.shtml, https://compton.nu/about/
17. https://lists.openstreetmap.org/pipermail/talk/2004-November/000078.html
18. http://wiki.openstreetmap.org/wiki/Mailing_lists
19. https://forum.openstreetmap.org/
20. https://trac.openstreetmap.org/
21. https://irc.openstreetmap.org/
22. https://svn.openstreetmap.org
23. https://github.com/openstreetmap/
24. https://trac.openstreetmap.org/
25. http://tomchance.org/2010/07/16/political-philosophy-in-openstreetmap/, https://blog.emacsen.net/blog/2018/02/16/osm-is-in-trouble/

References

Achtenhagen L, Müller-Lietzkow J and zu Knyphausen-Aufseß D (2003) Das open source-dilemma: Open Source-Software zwischen freier Verfügbarkeit und Kommerzialisierung. *Schmalenbachs Zeitschrift für betriebswirtschaftliche Forschung* 55(5): 455–481.

Bennett K (1995) Legacy systems: Coping with success. *IEEE Software* 12(1): 19–23.

Bijker WE (2002) *Of Bicycles, Bakelites, and Bulbs: Toward a Theory of Sociotechnical Change*. Cambridge, MAMIT Press.

Bijker WE (2012) The social construction of bakelite: Toward a theory of invention. In: Douglas DG, Hughes TP, Pinch TJ, et al. (eds) The social construction of bakelite: Toward a theory of invention. *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology* 155–182.

Bijker WE, Hughes TP and Pinch T (1987) *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press.

Bitzer J, Schrettl W and Schröder PJH (2007) Intrinsic motivation in open source software development. *Journal of Comparative Economics* 35(1): 160–169.

Budhathoki NR and Haythornthwaite C (2013) Motivation for open collaboration. *American Behavioral Scientist* 57(5): 548–575.

Chiang C-C and Bayrak C (2006) Legacy software modernization. In: *2006 IEEE international conference on systems, man and cybernetics*, Taipei, Taiwan, 8–11 October 2006, pp. 1304–1309.

Coast S (2015) *The Book of OSM*. Denver, CO: Steve Coast.

Cunningham W (1992) *The WyCash Portfolio Management System*. Available at: http://c2.com/doc/oopsla92.html (accessed 21 August 2017).

Eick SG, Graves TL, Karr AF, et al. (2001) Does code decay?: Assessing the evidence from change management data. *IEEE Transactions on Software Engineering* 27(1): 1–12.

Feller J and Fitzgerald B (2000) A framework analysis of the open source software development paradigm. In: *Proceedings of the twenty first international conference on information systems* (ed Association for Information Systems), Brisbane, Australia, 10–13 December 2000, pp. 58–69.

Fischer P and Hofer P (2008) *Lexikon der Informatik*. Berlin: Springer.

Glasze G, Schlieder C and Plennert M (2018) The socio-technical background of an unconventional software architecture in OpenStreetMap: Understanding the implementation of 'folksonomy'. *Computational Culture*..

Haklay M, Singleton A and Parker C (2008) Web Mapping 2.0: The neogeography of the GeoWeb. *Geography Compass* 2(6): 2011–2039.

Haklay M and Weber P (2008) OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing* 7(4): 12–18.

Hommels A (2005) Studying obduracy in the city: Toward a productive fusion between technology studies and urban studies. *Science, Technology & Human Values* 30(3): 323–351.

Hughes T (1987) The evolution of large technological systems. In: Bijker WE, Hughes TP and Pinch T (eds) *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press, pp. 51–82.

Iivari N (2009) Empowering the users? A critical textual analysis of the role of users in open source software development. *AI & Society* 23(4): 511–528.

Jelber SS, Lethbridge TC and Matwin S (2003) Mining the maintenance history of a legacy software system. In: *International conference on software maintenance* (ed IEEE), Amsterdam, Netherlands, 22–26 September 2003, pp. 95–104.

Kim Y-G (1997) Improving legacy systems maintainability. *Information Systems Management* 14(1): 7–11.

Klein HK and Kleinman DL (2002) The social construction of technology: Structural considerations. *Science, Technology & Human Values* 27(1): 28–52.

Kleinman DL (1998) Untangling context: Understanding a university laboratory in the commercial world. *Science, Technology & Human Values* 23(3): 285–314.

Kruchten P, Nord RL and Ozkaya I (2012) Technical debt: From metaphor to theory and practice. *IEEE Software* 29(6): 18–21.

Lakhani KR and von Hippel E (2003) How open source software works: ''Free'' user-to-user assistance. *Research Policy* 32(6): 923–943.

Latour B (1988) The prince for machines as well as for machinations. In: Elliot B (ed.) *Technology and Social Process*. Edinburgh: Edinburgh University Press, pp. 20–43.

Latour B (1994) On technical mediation – Philosophy, sociology, genealogy. *Common Knowledge* 3(2): 29–64.

Lattemann C and Stieglitz S (2005) Framework for governance in open source communities. In: *Proceedings of the 38th annual Hawaii international conference on system*

sciences, Hawaii, 3–6 January 2005, pp. 1–11. Washington, DC.

Law J (1987) Technology and heterogeneous engineering: The case of Portuguese expansion. In: Bijker WE, Hughes TP and Pinch T (eds) *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press, pp. 111–134.

Lerner J and Tirole J (2002) Some simple economics of open source. *The Journal of Industrial Economics* 50(2): 197–234.

MacKenzie DA (2012) Missile accuracy: A case study in the social processes of technological change. In: Douglas DG, Hughes TP, Pinch TJ, et al. (eds) *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press, pp. 189–216.

MacKenzie DA and Wajcman J (1999) *The Social Shaping of Technology*. Buckingham / Philadelphia, PA: Open University Press.

Mayring P (2010) *Qualitative Inhaltsanalyse*. Weinheim: Beltz.

Miller H (1998) *Reengineering Legacy Software Systems*. Boston, MA: Digital.

Mockus A, Fielding R and Herbsleb J (2002) Two case studies of open source software development: Apache and Mozilla. *ACM Transactions on Intelligent Systems and Technology* 11(3): 309–346.

Nakakoji K, Yamamoto Y, Nishinaka Y, et al. (2002) Evolution patterns of open-source software systems and communities. In: Aoyama M, Inoue K and Rajlich V (eds) Evolution patterns of open-source software systems and communities. *Proceedings of the international workshop on principles of software evolution*, Orlando, USA, 19–20 May 2002, pp. 76–85.

Olbrich S, Cruzes DS, Basili V, et al. (2009) The evolution and impact of code smells: A case study of two open source systems. In: IEEE (ed.) *3rd international symposium on Empirical Software Engineering and Measurement (ESEM)*. pp. 390–400.

OpenStreetMap contributors (2017) *OpenStreetMap*. Available at: http://www.openstreetmap.org (accessed 01 August 17).

Parnas DL (1994) Software aging. In: *Proceedings of the 16th international conference on Software engineering* (ed IEEE Computer Society Press), Sorrento, Italy, 16–21 May 1994, pp. 279–287. Los Alamitos, CA.

Pinch TJ and Bijker WE (2012) The social construction of facts and artifacts: Or how the sociology of science and the sociology of technology might benefit each other. In: Douglas DG, Hughes TP, Pinch TJ, et al. (eds) *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. Cambridge, MA: MIT Press, pp. 11–44.

Rajanen M and Iivari N (2015) Power, empowerment and open source usability. In: Begole B, Kim J, Inkpen K, et al. (eds) *Proceedings of the 33rd annual ACM conference on human factors in computing systems – CHI '15*. New York, NY: ACM Press, pp. 3413–3422.

Schweik C (2003) The institutional design of open source programming: Implications for addressing complex public policy and management problems. *First Monday*. 8: p. 1.

Shah SK (2006) Motivation, governance, and the viability of hybrid forms in open source software development. *Management Science* 52(7): 10000–11014.

Solomon R (2013) Last in, first out: Network archaeology of/as the stack. Available at: http://amodern.net/article/last-in-first-out/ (accessed 15 April 2018).

Sommerville I (2011) *Software Engineering*. Boston, MA: Pearson.

Straube T (2016) Stacked spaces: Mapping digital infrastructures. *Big Data & Society* 3(2): 1–12.

Torvalds L and By-Diamond DR (2001) *Just for Fun: The Story of an Accidental Revolutionary*. New York: Harper Collins.

Tufano M, Palomba F, Bavota G, et al. (eds) (2015) "When and why your code starts to smell bad". *Proceedings of the 37th international conference on software engineering – Volume 1: First international workshop on big data software engineering proceedings*. Florence, Italy, 23 May 2015, pp. 403–414. Los Alamitos, CA / Washington / Tokyo: IEEE Press.

Woolgar S (1991) The turn to technology in social studies of science. *Science, Technology & Human Values* 16(1): 20–50.

Wynn DE (2003) *Organizational structure of open source projects: A life cycle approach: Abstract for 7th Annual Conference of the Southern Association for Information Systems*. Georgia.