Information Systems Frontiers (2023) 25:241–257
https://doi.org/10.1007/s10796-020-10059-8

Sustaining Open Source Communities by Understanding the Influence of Discursive Manifestations on Sentiment

Denis Dennehy[1] · Kieran Conboy[1] · Jennifer Ferreira[2] · Jaganath Babu[1]

Published online: 15 September 2020

Abstract
Sustaining open source (OS) communities is fundamental to the long-term success of any open source software (OSS) project. An OSS project consists of a community of software developers who are part of a larger business ecosystem involving hardware and software companies. Peer review of software code, known as patch review comments, is an important quality assurance activity for OSS development that requires developers to provide feedback concerning their degree of satisfaction. Despite the importance of feedback, which can affect sentiment of OS communities, the underlying discourse has not been studied. In this study, we use Activity Theory to identify and categorise 20,651 discursive manifestations of contradictions that occurred in patch review comments of a large, evolving OS community. Unique community-specific expressions are identified and mapped to developers' sentiment during a software release cycle. The study contributes new insights concerning discursive manifestations of contradictions as a driving force for sustaining OS communities.

Keywords Activity theory · Contradictions · Sentiment analysis · Open source · Patch reviews

1 Introduction

Sustainability, an influential factor to creating competitive advantage (Berns et al. 2009; Hertel and Weisent 2013; Pappas et al. 2018) can be defined as the triple bottom line of economic, social, and environmental performance (Porter and Kramer 2006). To achieve this bottom line, companies will need to become 'sustainability-oriented' (Perrini and Tencati 2006), by (i) being aware of its responsibilities to various stakeholder groups, (ii) actively improving its ecological performance, (iii) contributing to sustainable social changes, (iv) and delivering value for society (Bednar and Welch 2020, Popovič et al. 2018, Pappas et al. 2018, Klievink et al. 2017). In this study, we focus on the sustainability of OS communities as it is fundamental to the long-term success and sustainability of any OS project (Gamalielsson and Lundell 2014).

✉ Denis Dennehy
denis.dennehy@nuigalway.ie

[1] National University of Ireland Galway, Galway, Ireland

[2] Victoria University of Wellington, Wellington, New Zealand

Sustainability in OSS projects largely depends on OSS developers maintaining healthy relationships with their peers in order to ensure their input and support (Ozer and Vogel 2015). Sustainability of OS projects largely depends on active, voluntary engagement from the OS community (Ho and Rai 2017; Germonprez et al. 2017; Xie and Matusiak 2016). Yet, research has shown that while processes, tools, and governance are important in enabling effective OSS development, sustaining the OS community is the most challenging (Appleyard and Chesbrough 2017; Ho and Richardson 2013; Sholler et al. 2019; Shaikh and Vaast 2016; Tourani et al. 2014). The reason sustainability is often such a challenge is that in traditional environments there is typically a formal reporting structure where one usually has a line manager or someone responsible for each developer. Developers can raise issues and concerns, and can get help if problems arise. However, OSS developers rarely raise issues and have no formal line manager to report to or who may be watching out for their well-being and motivation. Due to this, developers often simply disengage when not required to participate, when no communication regarding issues takes place, or when no support is provided to solve problems (Gamalielsson and Lundell 2014).

As OSS developers are usually geographically dispersed, 'mailing lists' are used to facilitate interactions between members of an OS community (Tourani et al. 2014; Mistrík et al.

2010; Bird et al. 2006). Mailing lists are communication channels where OSS developers conduct peer review of software code in the form of a patch[1] and provide feedback concerning their degree of satisfaction (Guzman et al. 2014). Previous research on mailing lists focused on discovering knowledge sharing practices (Sowe et al. 2008), information seeking behaviours among software developers (Sharif et al. 2015), identifying active contributors (Guzzi et al. 2013), and collective development (Hemetsberger and Reinhardt 2009). Other studies (Garcia-Cumbreras et al. 2013; Guzman et al. 2014; Pletea et al. 2014; Paul et al. 2018; Rousinopoulos et al. 2014; Sinha et al. 2016; Tourani et al. 2014; Ortu et al. 2015) applied sentiment analysis, which is essentially the task of identifying positive and negative opinions, evaluations, gestures, and cultural meanings organised around a relationship to a social object, usually another person or group (Wilson et al. 2005, Jongeling et al. 2015). While such studies have provided novel insights, they have largely focused on sentiment at a high level and not the underlying discourse, which can affect sentiment.

Our study is informed by Activity Theory (AT) as it anticipates discursive manifestations, which are types of contradictions (i.e. conflict, breakdown in communication) that interrupt the fluent flow of work (Hasan and Banna 2012; Helle 2000). Engeström and Sannino (2011) propose four distinct types of contradictions which they associate with discursive manifestations, namely, (i) double bind, (ii) conflict, (iii) critical conflict, and (iv) dilemma. Contradictions are historically accumulating structural tensions that occur within an activity and/or between multiple interrelated activities that generate disturbances and conflicts, as well as innovative attempts to change the activity (Karanasios et al. 2017; Engeström 2001). We argue that analysing patch review comments is important for three key reasons.

First, despite the long history of OS projects, patch review, a critical activity in sustaining OS projects has received relatively limited research effort (Wang et al. 2015). As AT deals with the purpose of information exchange (Valecha et al. 2019), it is a suitable lens to study the exchange of patch review comments. Our study extends previous research on sentiment analysis in OS projects by analysing the underlying discourse in patch review comments as discursive manifestations of contradictions, which can affect sentiment.

Second, data is now regarded as one of the most valuable resource to achieving a competitive edge and creating sustainable societies (Mikalef et al. 2020; Pappas et al. 2018; Chen et al. 2012; Gupta et al. 2018). As mailing lists provide a rich data source, this data can be used to advance our understanding of practices and social norms that contribute to the sustainability of OS communities (Bird et al. 2006; Shihab et al. 2009). We draw on a large, evolving OS community of 160 software developers from 25 companies that rely on a mailing list to conduct patch reviews.

Third, research has shown that sentiment affect quality, productivity, creativity, group rapport, and job satisfaction (De Choudhury and Counts 2013). It would therefore seem intuitive that for any OSS project, sentiment of OS communities plays an important role in sustaining an OS project (Tourani et al. 2014). Further, understanding the sentiment of software developers is important for project managers as it provides a better understanding of the social factors that affect the OS project and the corrective actions required to improve sentiment (Guzman et al. 2014; Rigby et al. 2008). This study shows how the language used by software developers in patch review comments can positively or negatively influence sentiment. To this end, the aim of this study is to advance knowledge on sustaining OS communities, by answering the following research questions:

1 How do discursive manifestations of contradictions manifest in OS patch reviews?
2 How do discursive manifestations of contradictions influence sentiment of OS communities?

The paper is structured as follows. First, we review literature on third generation AT and the types of discursive manifestations of contradictions. We then provide background to OSS development and the role of mailing lists and patch reviews. Next, the context of the OS project studied, and the analytical method used for data collection and analysis is outlined. In the discussion, we summarise key findings, followed by implications for practice and research. The paper ends with a conclusion, limitations and future action.

2 Theoretical Framework

2.1 Activity Theory and its Use in Information Systems Research

Contemporary thinking on AT, known as third generation AT emerged from the seminal work of Engeström (1987) who critically examined the classical lineage of semiotic and epistemological theories (cf. Peirce, Popper) and the lineage of symbol-mediated construction (cf. Mead, Trevarthen). Engeström (1987) concluded the former were too narrow in focus and the latter was conceived as 'construction-for-the mind' rather than 'practical material construction' (Ditsa 2003). Hence why it is possible to identify parallels between third-generation AT and other classical theories such as Mead's Symbolic Interactionism (Kuutti 1996).

AT is broadly defined as a "philosophical and cross-disciplinary framework for studying different forms of human practice as historically developing cultural systems" (Kuutti and Molin-Juustila 1998). It is a "reflexive theory that posits a dialectical relationship between theory and practice emerging

[1] Patches are sets of modifications to the existing codebase of a specific OS project.

in the context of history and culture" (Vermeulen et al. 2016, p. 1332). Strengths of AT are the interaction in a social context, and the notion of dynamics and development of a work activity (Mursu et al. 2007).

AT has inspired a number of theoretical reflections on what information systems (IS) and information systems development (ISD) are about (Allen et al. 2013; Beynon-Davies 2010; Bertelsen and Bødker 2000; Chen et al. 2013; Hasan et al. 1998, 2010; Korpela et al. 2001; Kuutti and Molin-Juustila 1998; Kuutti 1999; Vermeulen et al. 2016). AT has been used to examine Human Computer Interaction and IS design (Kuutti 1999; Nardi 1996), information systems development (Chen et al. 2013; Korpela et al. 2001; Dennehy and Conboy 2019; Igira 2008), technology mediated change (Chaudhury et al. 2001, Karanasios and Allen 2014; Ryu et al. 2005), patterns of use of IS (Wiredu and Sørensen 2006), and information and data sharing (Slavova and Karanasios 2018). A number of studies focused on contradictions associated with a new or changed mediating artefact, such as mobile technologies (Karanasios and Allen 2014; Kietzmann 2008), a pagination system, web channels (Chaudhury et al. 2001), software development tools (Dennehy and Conboy 2019), interaction systems, and enterprise systems (Malaurent and Karanasios 2020). Indeed such studies have made valuable contributions to knowledge; however, as they tend to overly focus on the material manifestation of contradictions (physical and virtual tools), they minimise, or possibly exclude the psychological tools (e.g. language, terminology) that influence the psyche and behaviour of people (cf. Hasan et al. 2010). Hasan et al. (2010) categorise the types of tools that mediate human activity and their influence on the object or subject (see Table 1). We map these tools to the context of this study.

During major transformations of their work activities, practitioners (i.e. software developers) are dealing with contradictions, but in expansive learning efforts, they must do much more than that, "they put themselves into imagined, simulated, and real situations that require personal engagement in actions with material objects and artifacts (including other human beings) that follow the logic of an anticipated or designed future model of the activity" (Engeström 2007, p. 37).

2.2 Discursive Manifestations of Contradictions

A fundamental concept of AT is the notion of contradictions, which occur within an activity and/or between multiple interrelated activities and promote dialectical transformation (Karanasios et al. 2017; Engeström 2001). While the term 'contradiction' may be considered by some as a weakness, from the perspective of AT, they are a sign of richness and an opportunity to develop (Karanasios et al. 2017; Karanasios 2018). Contradictions are seen as the sources of learning and can become the driving force for change and development in a system, if they are addressed (Hasan and Banna 2012). Using contradictions, referred to as 'growth buds' by Foot (2001) to promote learning and change is referred to as "expansive learning" (Engeström et al., 1999). Essentially contradictions are 'motors of change' (Allen et al. 2013). Contradictions can occur either inside the key constructs (e.g. community) or between them, or they may occur in networks of activity systems (Engeström 2001; White et al. 2016). Contradictions can be identified through their manifestations, which include, disturbances, errors, problems, rupture of communication, breakdowns, and clashes (Helle 2000; Kuutti 1996; Engeström 2001). Contradictions, however, may not be obvious, openly discussed, or be culturally or politically challenging to confront (Allen et al. 2013). Researchers must therefore rely on indirect methods to make visible the contradictions and to explain the genesis of their development (Dionne and Bourdon 2018). The identification of

Table 1 Categories of tools in AT

Types of tools (Hasan et al. 2010)	Tool description (Hasan et al. 2010)	Relevance in this study
Artifacts, instruments, machines, computers, mobile phones	Tools can be physical or digital objects and are used to produce changes in the object. Enables the automation of a new routine or construction of a new tool	• [a]Data Plane Development Kit (DPDK) mailing list
Language, signs, ideas, models	Tools are psychological and influence the psyche and behaviour of subjects (internal and external artefacts)	• Terminology of DPDK community • Ideas in the form of patch review comments • Visualisation of DPDK community metrics (e.g. Top 10 contributors, Top 10 reviewers)
Cultural systems, scientific fiction, virtual realities	Tools are psychological and influence the psyche and behaviour of subjects (imagined, simulated, real)	• Culture of participating organisation and project teams • Culture of OS community

[a] DPDK is a framework for fast packet processing in data plane applications. Users may use the code to understand some of the techniques employed, to build upon for prototyping or to add their own protocol stacks. https://doc.dpdk.org/guides/prog_guide/overview.html

contradictions can help those involved in an activity to focus their efforts on the root cause of problems, which can lead to the creation of a shared vision for the solution of the contradictions (Engestrom 2000). More recently, discursive manifestations of contradictions in organisational change efforts have been studied (Engeström and Sannino 2011; Dionne and Bourdon 2018; Malaurent and Karanasios 2020). Engeström and Sannino (2011) identify four distinct types of contradictions associated with discursive manifestations and its resolutions (see Table 2). Although Engeström and Sannino (2011) acknowledge that their categorisation of manifestations is not exhaustive, it does provide a foundation for this study to build upon.

Double bind is typically expressed "first by means of rhetorical questions indicating a cul-de-sac, a pressing need to do something and, at the same time, a perceived impossibility of action" (Engeström and Sannino 2011). It occurs when a person or group engages in interactions that raise paradoxical and contradictory demands, which make it difficult to step back from their current activities, and consequently create feelings of helplessness. A double bind is typically a situation which cannot be resolved by an individual alone (ibid.). Resolution requires making practical changes that are transformative and collective actions that go beyond words but is often accompanied with expressions such as "let us do that", "we will make it" (Dionne and Bourdon 2018; Engeström and Sannino 2011).

Critical conflict are Pappassituations 'in which people face inner doubts that paralyse them in front of contradictory motives unsolvable by the subject alone' (Engeström and Sannino 2011, p. 374). These critical conflicts are very emotionally and morally charged, which makes it difficult, or even impossible, for them to be resolved solely by the subjects involved (ibid.). Discourse is also marked by vivid metaphors that describe an object or action in a way that isn't literally true, but helps explain an idea or make a comparison (Dionne and Bourdon 2018). Its resolution occurs through 'a renegotiation of meaning for the subject who was accompanied by the collective in order to allow the former to gain critical distance from their experience and to give it new meaning' (ibid., p. 282).

Conflict takes the form of resistance, disagreement, argument and criticism, and occurs "when an individual or a group feels negatively affected by another individual or group, i.e. because of a perceived divergence of interests, or because of another's incompatible behaviour" (De Dreu and Van De Vliert 1997, p. 1). Engeström and Sannino (2011) observed that people engaged in a conflict tend to argue and to criticise each other. Conflicts are resolved through compromise or submitting to authority or the majority (Dionne and Bourdon 2018).

Dilemma is an 'expression or exchange of incompatible evaluations, either between people or within the discourse of a single person' and is most often expressed in the form of hesitations, such as "yes, but" (Engeström and Sannino 2011). It is typically reproduced rather than resolved, often with the help of denial or reformulation (i.e. I didn't mean that).

Using AT as the theoretical lens is pertinent in this study for four reasons, namely (i) understanding context in which the words are used is important as it strongly influences accuracy (Aue and Gamon 2005; Turney 2002), (ii) AT is oriented at understanding the activity in context (Cole and Engeström 1993), (iii) AT acknowledges contradictions as a means of understanding and change (Engeström 2001; Ilyenkov 1974), a concept that is not explicit in other social theories (Karanasios and Allen 2014), and (iv) AT is a developmental theory that seeks to explain and influence changes in human practices over time (Engeström 1999).

3 Background to OSS Development

Founded in 1998, the Open Source Initiative protects and promotes open source development, standards, and communities of practice. Open source is a development method for

Table 2 Types of discursive manifestations of contradictions (Engeström and Sannino 2011)

Manifestation	Features	Linguistic Cues
Double bind	Facing pressing and equally unacceptable alternatives in an activity system: Resolution: practical transformation (going beyond words)	"We", "us", "we must", "we have to" pressing rhetorical questions, expressions of helplessness
Critical conflict	Facing contradictory motives in social interaction, feeling violated or guilty Resolution: finding new personal sense and negotiating a new meaning	Personal, emotional, moral accounts narrative structure, vivid metaphors "I now realise that. . ."
Conflict	Arguing, criticising Resolution: finding a compromise, submitting to authority or majority	"No", "I disagree", "this is not true", "this I can't accept"
Dilemma	Expression or exchange of incompatible evaluations Resolution: denial, reformulation	"On the one hand [. . .] on the other hand"; "yes, but" "I didn't mean that", "I actually meant"

software that harnesses the power of distributed peer review and transparency of process. The promise of open source is better quality, higher reliability, more flexibility, lower cost, and an end to predatory vendor lock-in. A unique characteristic of OSS development is the involvement of communities that engage general users who do not belong to typical software development roles (Wang et al. 2015). As the people who contribute code to an OS project are always users of the code produced, it means that software developers are a subset of the OS user, but not all users are software developers. Figure 1 illustrates the role of users and software developers in an OS project.

3.1 The Role of Mailing Lists in OSS Development

Mailing lists have a central role in OSS development (see Fig. 2) as software developers use them as a communication channel to discuss a variety of issues related to the source code and external factors such as the introduction of new features in competing products (Shihab et al. 2009). Mailing lists have been used to study social structure (Ogawa et al. 2007), identify architectural changes (Baysal and Malton 2007), code peer review process (Rigby et al. 2008, Weissgerber et al. 2008), knowledge sharing (Sowe et al. 2008), and the morale of software developers (Lakhani and von Hippel 2004).

In this study, we analyse the DPDK mailing list data as it enables us to analyse patch reviews of a large OS project.

3.2 Patch Review in OSS Development

Patch review is a practice in which members of an OS project review software code in order to discover defects as early as

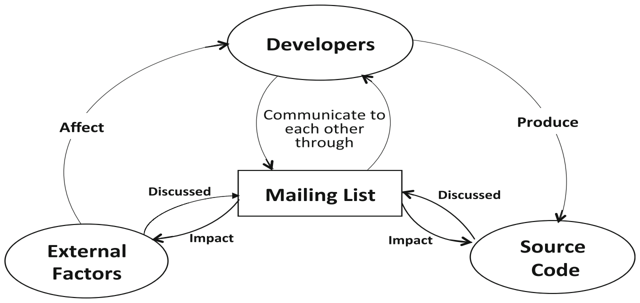

Fig. 2 Central role of mailing lists in OS projects. (source: Shihab et al. 2009)

possible during the software development release-cycle and to contribute and verify solutions that repair or improve them (Wang et al. 2015). The process of incorporating patches into the source code of an OS project is known as a 'patch contribution process', which effects both the quality of the OS system and the growth of the OS community (Sethanandha 2011; Sethanandha et al. 2010b). The main features of the DPDK patch review process include, (i) hosting software code in a public repository, (ii) a mailing list where registered members 'submit' code, (iii) code is reviewed publicly on the mailing list, and (iv), successfully reviewed code is merged into the main repository for scheduled releases.

The patch review process is an important activity for OSS development because it (i) is a primary quality assurance mechanism, especially for contributions from new contributors (Mockus et al. 2000; Rigby et al. 2008), (ii) enables learning and knowledge transfer (Nurolahzade et al. 2009; Sethanandha et al. 2010a, (iii) provides an opportunity for recruiting potential software developers into OS projects as contributors are able to gain influential roles in such projects (Jensen and Scacchi 2007; Sethanandha 2011), (iv) plays an important role in the integration and socialisation of new

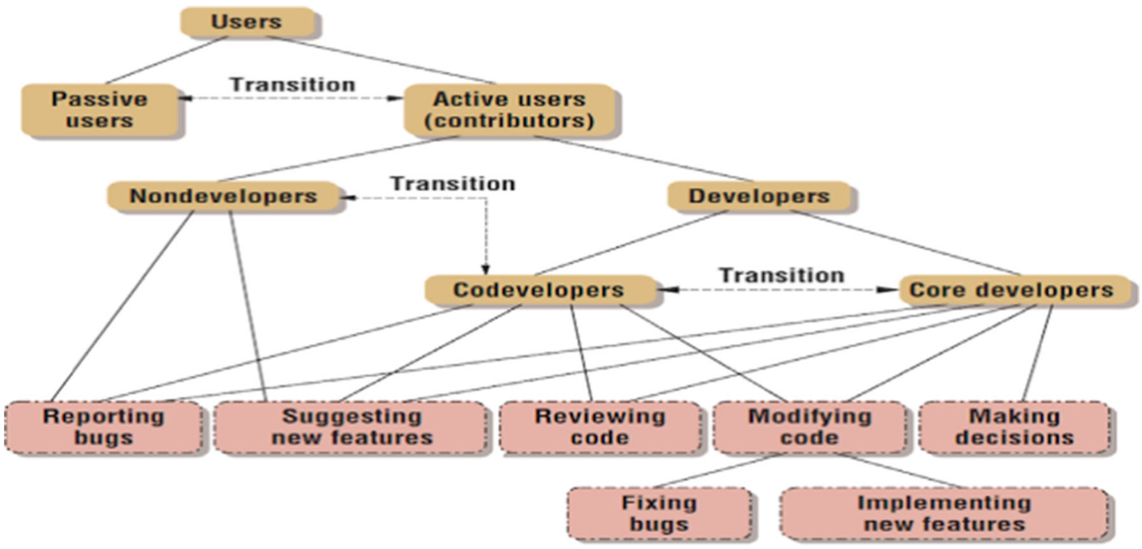

Fig. 1 Classification of OS users and software developers

members in an OS project (Ducheneaut 2005), and (v) raises the status of individuals and companies involved in the project, if the community involved recognise their contribution as being highly appropriate and of excellent quality (Gacek and Arief 2004).

Factors such as contributor experience, the capability of tools, and the quality of patches have significant influence on the amount of time required to complete a patch review (Sethanandha 2011). The patch review is widely regarded as the one that significantly benefits from the community involvement (Wang et al. 2015). Despite the importance of the patch reviews, there is limited research that examines software changes contained in email archives in the form of patches as opposed to analysing the change information stored in software repositories (Wang et al. 2015; Rigby et al. 2008; Weibgerber et al. 2008). Previous studies (e.g. Hackman et al. 2007) that focused on patch review activities provide evidence that the peer review practice is likely to vary across different OS communities, and that no one-size-fits-all model can ensure success (Barnham 2012).

3.3 Empirical Setting and Analytical Methodology

The study reported here is part of a larger 4-year research project. The over-arching goal of the project is to examine how analytics can be used to track changes, improvements, and transitions of the OS community and provide management with actionable insights. The case studied has a dedicated Research and Development (R&D) campus which is a Centre of Excellence for network transformation and cloud computing. The company's software developers participate in the DPDK OS project and also use the software in their work. DPDK was launched as commercial software in 2010 and made available under a permissive open source license in 2013. The DPDK OS community has since been continuously growing in terms of the number of contributors, patches, and contributions from software developers and organisations.

3.4 Overview of DPDK OS Community

This community has been steadily growing since its establishment in 2013. Figure 3 shows how the growth from 200 contributors with 20 commits in 2013 to 1,600 with 160 commits in version 18.05 in 2018. The growing and evolving OS community has two key implications for management of the OS project. First, as comments in the patch review are being provided by software developers who do not speak English as their first language, their meaning can then be lost in translation, as well as influence the sentiment of software developers. Second, management had the assumption that sentiment of the OS community was generally negative.

About DPDK It consists of data plane libraries and network interface controller drivers to accelerate the performance of telecommunications and data networks. DPDK is currently managed as an open-source project under the Linux Foundation and licensed under the Open Source License. As new networking hardware is developed for improved communications and connectivity, new software features are needed to enable those new hardware capabilities and support ongoing improvements in performance. New features are continually being added (committed) to the DPDK codebase by way of the community contributing, reviewing, and approving the software code for these features (https://www.dpdk.org/).

DPDK Patch Review The main features of the DPDK patch review process include, (i) hosting software code in a public repository, (ii) a mailing list where registered members 'submit' code, (iii) code is reviewed publicly on the mailing list, and (iv), successfully reviewed code is merged into the main repository for scheduled releases. All code contributions to the DPDK project are submitted to the dpdk-dev mailing list for community review. If the OS community approves (e.g. Ack = acknowledged/accepted) a patch, it is considered ready to be merged with the main codebase. If a patch requires improvements (e.g. NIT = minor problems with the code) to the code, the software developer will implement the suggestions and resubmit the patch to the mailing list for another round of reviews. If the OS community reject a patch (e.g. NACK = not acknowledged/rejected) then it is not merged with the main codebase.

DPDK Release Cycle The DPDK release cycle timeline for 2018 is presented in Fig. 4. The coding used for each release cycle (e.g. 18.02, 18.05, 18.08, and 18.11) is based on the current year and month of scheduled release. As there is a cut-off time for patches to be sent for review by the community during each release (18.05 V1), when patches have been approved (Ack'd) they must be added to the code base (see amber symbols). All validated patches are then released as scheduled (see green symbols). All validated patches are then released as scheduled (see green symbols).

4 Data Collection and Analysis

In order to rigorously analyse sentiment and discursive manifestations of contradictions in the DPDK mailing list, the data had to be prepared (extracted, validated, cleaned) using analytical techniques. We used the Cross Industry Standard Process for Data Mining (CRISP-DM) as it is an industry standard methodology, developed in 1996 by Daimler Chrysler, that prescribes a set of guidelines to guide the efficient extraction of information from data (Chapman et al. 2000; Shearer 2000). The CRISP-DM

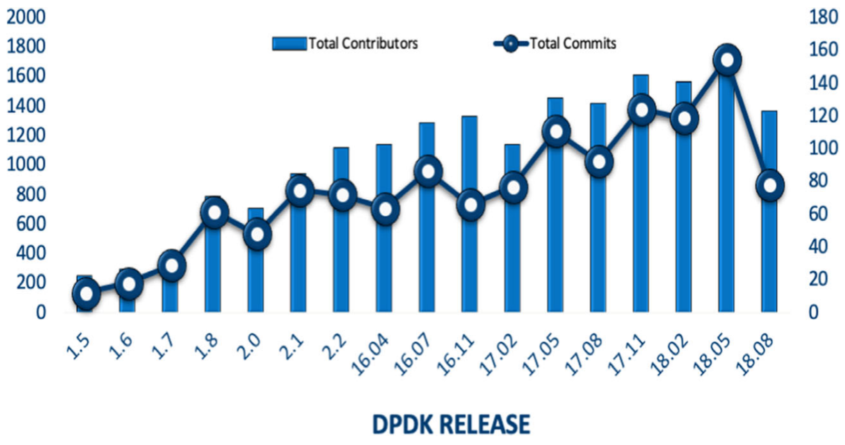

Fig. 3 Growth of the DPDK community relative to the number of commits

methodology consists of six cyclical steps, namely (i) Business Understanding, (ii) Data Understanding, (iii) Data Preparation, (iv) Modeling, (v) Evaluation, and (vi) Deployment. We adapt this methodology to suit the context of our research, which includes four cyclical phases (see Fig. 5).

The adapted model (Fig. 5) does not exclude any of the six phases of CRISP-DM, instead, it merges them into four interrelated activities, namely, (i) Business and Data Understanding, (ii) Modeling, (iii) Evaluation, and (iv) Actionable insights.

Business and Data Understanding In this phase, business understanding focused on the context, aim and business problem in order to align with the project objectives, and data understanding provided an understanding of the data that needed to be analysed, identify potential issues (i.e. quality) and prepare for modeling. Input data comprised of (i) dataset of 29,605 messages from the DPDK-dev mailing list archived at http://mails.dpdk.org/archives/dev/, and (ii) two popular sentiment analysis dictionaries (e.g. Opinion Lexicon, Comparative Words) that were customised for the business understanding (Lin et al. 2018; Novielli et al. 2018; Jongeling et al. 2015).

Data Preparation This phase determined what data should be included in the dataset, cleaning the data and all other activities that needed to be done to process data which served as an input to the modeling tool in the next step. Data extraction and integration using Python scripts whereby messages were converted from RAR file format into .CSV file format. Messages dated outside the four release cycles of 2018 were removed which resulted in 20,651 messages being included in this study. The message content was cleaned for analysis using regular expressions to ensure that only the message body and natural language remained. All message headers, code, file paths, and non-alphanumeric symbols/characters were removed. This activity was critical to reduce any instances of misclassification (Tourani et al. 2014). The remaining text was then converted into Pandas DataFrame format (a tabular data structure in Python) for compatibility purposes with the sentiment analysis algorithm.

Modeling This phase performed the following three prerequisite tasks, (i), classification of discursive manifestations, (ii) identification of community-specific expressions e.g. 'NIT' (e.g. minor problems with the code), 'NACK' (e.g. a patch not accepted by the community), 'SELF-NACK' (e.g. a patch removed by its author), and (iii) sentiment analysis across four release cycles of 2018. Task one involved the customisation of the sentiment analysis dictionaries (e.g. Opinion Lexicon, Comparative Words), which enabled us to classify the types and frequency of discursive manifestations. This involved augmenting the natural language dictionary with community-specific expressions (e.g. NACK, NIT) of the OS community. In doing so, the linguistic cues unique to the OS community studied are central in the analysis of discursive manifestations of contradictions, namely 'NIT' (e.g. Dilemma), and 'NACK' (e.g. Critical conflict, conflict). Task three involved analysing the sentiment in the message body content (see Table 3). We followed a similar approach to Rousinopoulos et al. (2014) where the message body was split

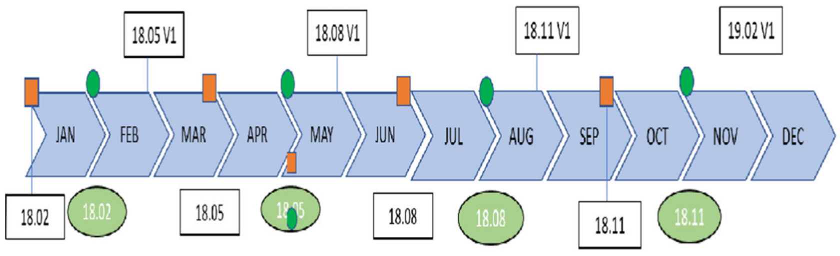

Fig. 4 DPDK patch review and release cycle timeline

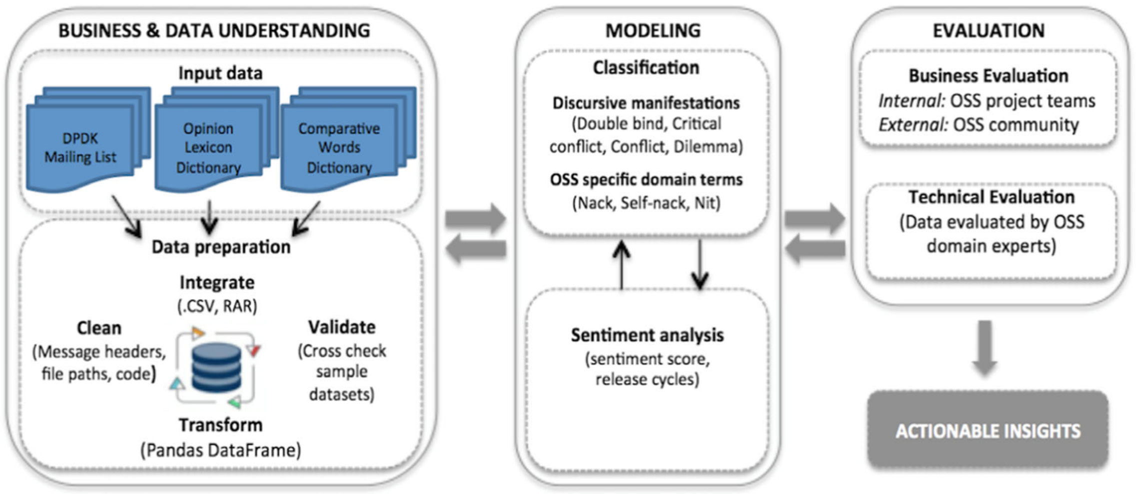
BUSINESS & DATA UNDERSTANDING
Input data
DPDK Mailing List
Opinion Lexicon Dictionary
Comparative Words Dictionary
Data preparation
Integrate (.CSV, RAR)
Clean (Message headers, file paths, code)
Validate (Cross check sample datasets)
Transform (Pandas DataFrame)
MODELING
Classification
Discursive manifestations (Double bind, Critical conflict, Conflict, Dilemma)
OSS specific domain terms (Nack, Self-nack, Nit)
Sentiment analysis (sentiment score, release cycles)
EVALUATION
Business Evaluation
Internal: OSS project teams
External: OSS community
Technical Evaluation (Data evaluated by OSS domain experts)
ACTIONABLE INSIGHTS

Fig. 5 Research methodology

into tokens and using a rule-based algorithm in combination with the two dictionaries, assigned a positive, neutral, or negative score. The assigned sentiment scores ranged from 'Strong negative' (-20), Weak negative (-10), Neutral (0), Positive (+ 10), and Strong positive (+ 20). A token was assigned a score according to the matching word found in the dictionaries and the overall sentiment of a message was computed as the sum of all scores assigned to the tokens contained in that message.

Evaluation In this iterative phase, the model, data, and emerging findings were analysed in relation to the business and data understanding (e.g. aims of the business and research). This involved collaborating with experienced software developers and managers involved in a large OS project and who were involved in the patch review process. This iterative process ensured that the emerging findings and actionable insights supported the OS project. These findings are presented in the next section.

5 Findings and Analysis

This section presents key findings under three interrelated subsections, namely, (i) discursive manifestations of contradictions, (ii) frequency and effect of the community-specific expressions (e.g. NIT, NACK) on sentiment, and (iii) a statistical analysis of sentiment throughout 2018. Our analysis draws on the analytical approach that was scaffolded by the CRISP-DM methodology to provide an analytical and explanatory chain of evidence (e.g., word frequencies). The excerpts that we present in this section are those we consider to be representative of the OS community studied.

5.1 Discursive Manifestations of Contradictions

Analysing the natural language contained in the DPDK mailing list provides rich insights into the internal dynamics of the OS community. In the context of this study, Fig. 6 presents the frequency of the four distinct types of discursive manifestations of contradictions as proposed Engeström and Sannino (2011). Most noticeable is the high frequency of 'conflict' (arguing, criticising) and 'double blind' (unacceptable alternatives) manifestations, specifically in the 18.02 and 18.08 release cycles. This finding would suggest that software developers have established strong working relationships and are comfortable with challenging their peers and vice versa to

Table 3 Sample of DPDK patch review comments with time stamp

Index	Time Stamp	Patch review comment
13,674	2018-03-31 17:04:08	on XXX 2018 at 11 37 am, XXX wrote uhmm, not sure about this. it is specific to the nfp pmd and i do not see it as a new feature, at least dpdk users will not be aware of it. i forgot to remove this reference. it is not needed now, but it was with the initial internal work. about the other build errors, i do not get them and i have used a couple of different systems, XXX and XXX. this is, of course, a serious concern. can you give me more information about the system you are using i remember i got some build error with other patches, from automatic builds made just after those patches were sent. i did not get any this time, just those warning for checkpatch which i was aware of. is this automatic build not happening
15,587	2018-04-10 15:28:48	i am sorry, i have to nack because the change is not explained.

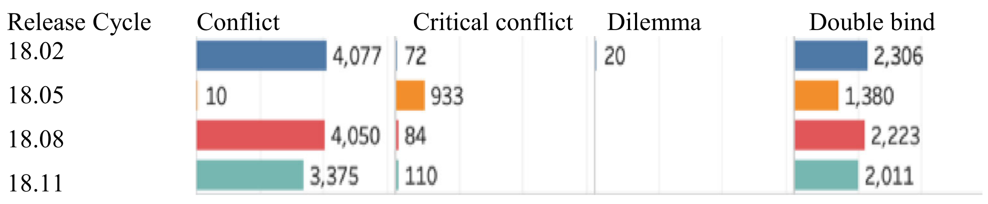

Fig. 6 Frequency of discursive manifestations

ensure that patches are defect free before being merged with other code prior to the release.

There was a much lower frequency of 'critical conflict' manifestations across all four-release cycles, except the 18.05 release cycle which accounted for the majority of these. The low number of critical conflicts suggests there is a high degree of professionalism and accepted language used by the software developers as they have established working relationships over a number of years. It could also suggest that software developers have a shared understanding of the patch and the code is consistent across the OS community.

No 'dilemma' manifestations were identified in three release cycles (e.g. 18.05, 18.08, 18.11) and only twenty in the 18.02 release cycle. The low number of dilemmas would suggest that the OS community generally use language that is unambiguous in their patch reviews and that the OS community have developed a common language that is understood by the experienced software developers. Possibly new members (e.g. graduate software developers) to the OS community write comments that contribute to these dilemmas but over time, assimilate into the social norms of the OS community.

Table 4 lists excerpts of the patch review comments that are categorised as discursive manifestations of contradictions, as proposed by Engeström and Sannino (2011). By extracting the underlying discourse used in the patch review comments and categorising it as discursive manifestations of contradictions, we gain a new perspective about the OS community and how language can influence sentiment and success of the OS community.

The value of Table 4 is that it shows how a 'NACK' can manifest as different types of contradictions (e.g. critical conflict, conflict, dilemma), depending on how it is framed by the person reviewing the patch. This would suggest that there are subtle differences around instances of 'NACK' that require further investigation. For example, in the following excerpt from an email message (2nd Mar), *"In my opinion the fact test is allowed on a closed port is fishy. The proposed patch is a workaround that doesn't address the underlying issue, thus NACK unless proven otherwise:J"* we start to understand why sentiment around 'NACK' is not strongly negative. Firstly, a smiley emoji at the end of the sentence indicates that the author is not adversarial with this comment. Secondly, the author rejects the patch, but leaves it to the community to prove that this patch is still useful for solving the 'underlying issue', which implies this is a conditional 'NACK' and the author is willing to retract it. In another excerpt (12th Apr), *"It's a NACK from me, but let's work together on something better"* a positive sentiment is displayed by the author who encourages the community to work towards a better solution, despite the rejection of the patch.

We acknowledge that discursive manifestations of contradictions can indeed overlap with other categories (i.e. NACK), depending how (i) the sender frames the discourse, (ii) the recipient receives it, and (iii) it is categorised by the researchers.

5.2 Frequency and Effect of Community-specific Expressions on Sentiment

As the community-specific expressions (e.g. NIT, NACK, and SELF NACK) are unique to the DPDK community, identifying their frequency and effect on sentiment is important. Figure 7 lists the frequency of the community-specific expressions for each release cycle of 2018. Most noticeable is the high frequency of 'NACKS' (#15), NITS (#17), and specifically 'SELF NACKS' (#7), during the 1805 release cycle. By identifying the frequency of these expressions, it provides a rich contextual understanding of the OS community, and to influence change in the OS community over time. For example, code that may be considered to be too innovative and not adhering to the traditional practices of the OS community may be prematurely rejected by members of the community.

To understand the impact of community-specific expressions, the 18.05 release cycle, which has the highest frequency of manifestations (39 in total), is used to illustrate the impact of these 39 manifestations on sentiment of the OS community (see Fig. 8). The sentiment score is first plotted against the time period of the 18.05 release cycle (28th Feb – 5 May) during which three key activities, (i) scoping, (ii) pre-merge code, and (iii) bug fix, test, and release are completed by the OS community. Each of the thirty-nine reported community-specific expressions is then mapped to the actual date recorded in DPDK (see Fig. 8). Figure 8 shows that the frequency of community-specific expressions during the 18.05 release cycle is balanced, with a slight increase towards the end of the release. Most concerning though, is the number of 'NACK' and 'SELF NACK' expressions that occur close to the release date. When compared to similar instances that occur much earlier in the release cycle these have a high negative impact on sentiment. While a possible explanation may be that software developers are more confident in resolving such instances early in the release cycle, such instances clearly impact

Table 4 Examples of discursive manifestations of contradictions

Manifestation	Examples in the context of the OS community studied
Double bind	• It is not friendly to multi-process model as it leads to port ID contention issue if two processes both find the data entry is free. We must allocate from the pre-defined array so that we can find it.
	• I think we must guarantee no port allocation for the same port ID in the callback time.
	• Unfortunately, it doesn't meet the basic quality criteria… we must not add compile time device option if not well justified. I guess we can work around with a direct include.
	• We must consider a solution as you propose below, but my proposal could easily be implemented for v18.05. Whereas your patch is quite a big change and I think it's a bit too late as integration deadline.
	• We must allocate from the pre-defined array so that we can find it.
Critical conflict	• What I dislike is having inconsistencies in the code layout. Paragraphs, for lack of a better word, are highly subjective and a matter of taste. Given that subjectivity is not helpful in review and taste is hard to debate, I prefer to have a single terse rule; this makes a patch context as information rich as possible. Sorry that I made it more difficult than necessary.
	• This is irritating when one is compiling with warnings as errors in order to catch more serious bugs. One potential fix might be to define page size but as I'm not clear on the implications, I can't comment further.
	• Looks like the bind mechanism should be managed directly by the Project Managers rather annoying application developers to deal with it.
	• I'm very unhappy about the logging hack.
	• I am sorry but I have to ***NACK*** because the change is not explained.
	• I can't agree with this statement, the essence of DPDK is to give a good alternative to managing network devices. I disagree with this final assessment.
Conflict	• I bet your teacher would disagree with that statement with one single paragraph in your book reports - taste is hard to debate, but you have gone the extreme route with only the bare minimum blank lines and that is not good. A silly script does not read code or understand code, we the humans have to make the code readable.
	• I've seen this kind of approach implemented before to add additional memory types to DPDK and I don't like it. I've outlined some of my thoughts on this before. You're welcome to continue that discussion, and make sure whatever comes out of it is going to be useful for all of us. Now that the memory hot plug is merged, I'll hopefully get more time prototyping, it's a ***NACK*** from me, but let's work together on something better.
	• You did not reply to my questions and did not address everything. It deserves an explanation. Please Mr. X; think about commit explanations more often.
Dilemma	• Small ***NIT***, duplicating the title in the body is not useful and the returned value is positive.
	• Some ***NITS***, why is the whole item not under 'xxx' as it's more accurate to keep consistency among the errors.
	• Two ***NITS***, I think we could add a note in commit log that it only applies to primary and secondary request. To make the change simpler, maybe we can just put a declaration of function.
	• On second thoughts, ***SELF-NACK***… without this patch the original problem still exists and we need to find an alternative workaround.
	• We would like 2 or 3 more days on this before we can ***'ACK' 'NACK'*** this patch.
	• ***NACK***, this is breaking the native Linux compilation. I am looking into it.

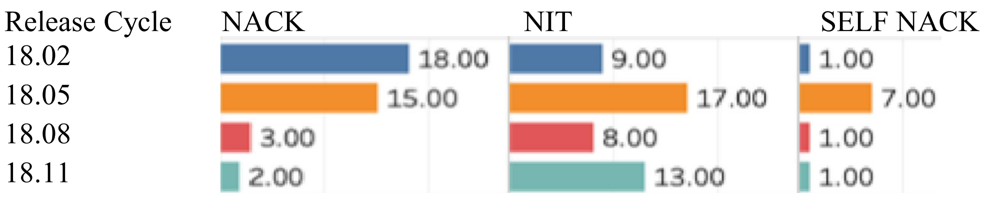

Fig. 7 Frequency of OSS community-specific expressions

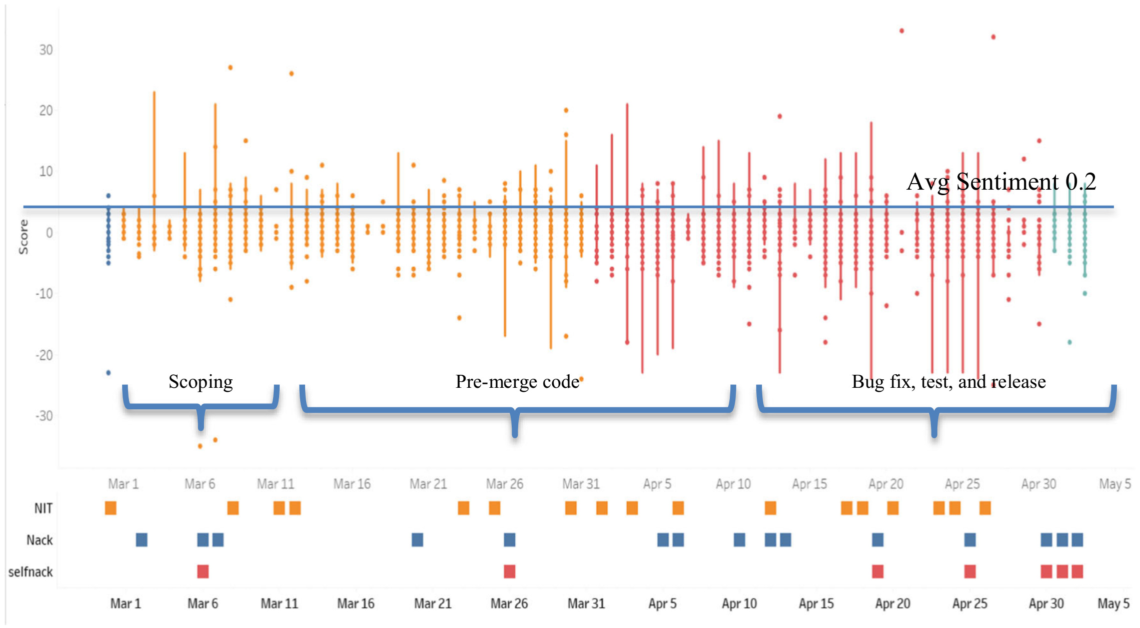

Fig. 8 Sentiment score of 18.05 release cycle

their sentiment as the release date draws nearer and their professional reputation and that of their company can be questioned if a patch is rejected in the days preceding a release.

Analysis of the sentiment reveals that the overall sentiment is minimally positive (0.210). A number of positive and negative outliers are present at the start and end of release cycle. The underlying reason for these is that initially a patch will have errors/defects but following a series of reviews and revisions, the quality of the patch improves, as does sentiment of the community. As overall sentiment is minimally positive, these findings challenge two assumptions. First, that sentiment of the DPDK OS community is negative. Sentiment is only negative during certain periods of the release cycles, i.e. towards the end of the pre-merge code phase and during the bug fix, test, and release phase. This is understandable given the complexity of OSS development and that the software developers' work under pressure due to fixed timelines and release dates that cannot be adjusted. Second, that patch review comments containing a 'NACK' should have strong negative sentiment. A patch review comment containing a 'NACK' can also contain positive sentiment, which can have a neutralising effect on the overall sentiment score.

5.3 Statistical Analysis of Sentiment

The findings are supported by the distribution of sentiment scores represented in Fig. 9 that shows the sentiment score distribution that is normally distributed and the mode is zero. This indicates that the majority of discussions were neutral due to the technical nature of the conversations for each review.

Table 5 provides a summary of statistics for each release cycle during 2018. The table shows that mailing list sentiment changes over the course of the year as evidenced by the variations in the mean and standard deviations of the sentiment scores for each release cycle. The mean sentiment score for the entire year is 0.2 (marginally positive). Although sentiment remains marginally positive throughout the year, the mean scores steadily decrease in positivity from the first release cycle (18.02), to the second (18.05), and into the third release cycle (18.08), before increasing again in the fourth release cycle (18.11). The 18.02 release cycle exhibits the highest mean sentiment score of 0.31 and is therefore the most positive of all the release cycles for 2018. While the 18.08 release cycle exhibits the lowest mean sentiment score of 0.06 and is therefore the closest to neutral sentiment of all release cycles in 2018.

The standard deviation of sentiment scores for the year is 2.2, which is sentiment scores for mailing list messages sent in 2018 differ from the mean by 2.2 points. In this instance, the 1805 release cycle has the largest standard deviation, indicating that mailing list messages exhibit stronger positive or stronger negative sentiment than in other release cycles.

The 18.02 release cycle has the lowest standard deviation, indicating that the mailing list messages exhibit weaker positive or negative sentiment. The strength of sentiment is low in the first release cycle (e.g. 18.02) and then it peaks in the second release cycle (e.g. 18.05), before steadily declining throughout the rest of the release cycles. These scores indicate that more emotion is expressed in the two mid-year release cycles (e.g. 18.05, 18.08) than in the first (e.g. 18.02) and last (e.g. 18.11) release cycles. These scores support the previous analysis such as the 'mean' progressing from − 0.12 in Feb to + 0.21 in April.

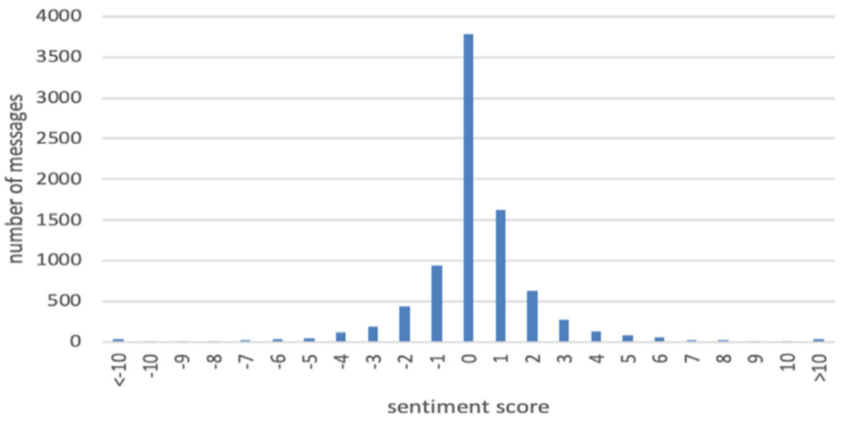

Fig. 9 Frequency distribution of sentiment scores

6 Discussion and Implications

From the outset, the aim of this study was to advance knowledge on sustaining OS communities, by understanding how discursive manifestations of contradictions manifest in OS patch reviews and its influence on sentiment of OS communities. In doing so, this study makes important theoretical contributions to sustainability in the 21st century (Bednar and Welch 2020, Popovič et al. 2018, Pappas et al. 2018, Klievink et al. 2017), and activity theory (Engeström and Sannino 2011; Dionne and Bourdon 2018; Karanasios et al. 2017; Malaurent and Karanasios 2020), and OSS development (Appleyard and Chesbrough 2017; Gamalielsson and Lundell 2014; Germonprez et al. 2017; Shaikh and Vaast 2016).

6.1 Implications for Research

First, we advance knowledge on sustainability in the context OS projects by understanding the language and community-specific expressions, which are essentially psychological tools (cf. Hasan et al. 2010) that influence the sentiment of OS communities and impact its long-term sustainability. Previous studies that examined sentiment in OS projects they tend to overly focus (and rely) on the statistical meanings of sentiment which limits understanding of the personal relations between members of the OS communities. By understanding the language of the OS community our studies highlights how the use of language is both 'a product of activity and a key determining factor of human behaviour, which affects the activity itself' (cf. Sannino 2008). This is critically important to the long term evolution and sustainability of the OS community.

Second, we make a methodological contribution to Activity Theory by applying discursive manifestations of contradictions (Engeström and Sannino 2011)to a new domain (e.g. OS communities). Previous studies (e.g. Dionne and Bourdon 2018; Malaurent and Karanasios 2020) have examined discursive manifestations of contradictions in the context of organisational change efforts but to the best of our knowledge, this is the first study to apply the four types of contradictions (e.g. double bind, double conflict, conflict, dilemma) as proposed by Engeström and Sannino (2011).

Third, we contribute to OSS development by theorising about the influence of discursive manifestations of contradictions on sentiment of OS communities, which can influence the sustainability of an OS project. By using a robust theoretical framework such as AT, it provides researchers with the opportunity to analyse and conceptualise complex real-world situations where the interrelationship between communities of people (OS community), mediating tools (online forum), and a cultural-historical setting co-evolve (new members join or leave the OS community). While the *"global process of digitalising and digitalised mediation of every aspect of human practice and activity is the hardest challenge activity theory has ever met"* (Rückriem 2009, p. 30), this study shows that AT is a robust theory that provides a powerful

Table 5 Summary statistics of release cycles

Statistical Analysis	18.02 Release cycle	18.05 Release cycle	18.08 Release cycle	18.11 Release cycle	Full Year Cycle (2018)
Number of messages	7,413	8,585	6,232	7,375	[a]29,605
Mean sentiment score	0.31	0.21	0.06	0.21	0.20
Standard deviation	2.13	2.55	2.29	2.18	2.20

[a] During the data preparation phase (cleaning) this figure was reduced to 20,651 messages in the actual analysis reported in this study as messages that were 'forwarded', or 'follow-ups' were excluded to ensure a more accurate representation of the OS community.

analytical lens to study digitised mediated environments and ecosystems. This is an important contribution as there is a noticeable absence of research that progress from simply applying sentiment analysis (e.g. Tourani et al. 2014; Guzman et al. 2014; Rigby et al. 2008) to advancing the accumulative body of knowledge via theoretical development. This lack of cumulative tradition resonates with the issue of 'fragmented adhocracy' (Metcalfe 2004; Weick 1989) which has previously overshadowed IS research (Fitzgerald and Adam 2000; Banville and Landry 1989; Hirschheim et al. 1996).

Fourth, we advance knowledge by highlighting that events that are generally perceived as 'bad' in an OS project are indeed opportunities for (i) innovation, (ii) improved dialogue within the OS community, and (iii) better collaboration between all stakeholders of the OS project. For example, rather than view 'NACK' as a waste of time, resources, and finances, it can be used as an opportunity to create events (on/offline) that can build cohesion in the OS community and contribute to the overall health and sustainability of community. Existing OS research and associated theories examine methods for eliciting and sustaining behaviour, but all are built on the assumption that (i) issues that create negative sentiment are known, and (ii) there is time to take corrective action. Researchers can now use sentiment analysis to get an earlier determination of negative issues, and so research interventions can then be more targeted and will have more time to take effect, hopefully increasing their success.

6.2 Implications for OS Practice

This study has three critical implications for the management of OS projects and associated work practices.

First, by illuminating the community-specific expressions and mapping them to the release cycle, it provides managers and software developers the opportunity to reflect and reassess their current work practices in order to reduce unnecessary time pressures. Indeed, such reassessment could challenge the 'adherence based' approach to using a specific software development method or project management tool as it may be interrupting the fluent flow of work (Helle 2000) or restricting the interactions of the OS community (Kuutti 1996).

Second, in contrast to sentiment analysis, which by itself, does not provide rich contextual data to drive change (i.e. OS community), AT is a developmental theory that seeks to explain and influence changes in human practices over time (Engeström 1999). By using AT to analyse patch review comments, this study provides rich data in the form of discursive manifestations of contradictions that can influence the long term sustainability of an OS community. This is particularly valuable, given that sustaining an OS community is one of the biggest challenges in OSS practice (Appleyard and Chesbrough 2017; Sholler et al. 2019; Tourani et al. 2014) and that software developers rarely raise concerns or issues of negativity - they just stop contributing. Therefore, early detection of negative sentiment is critical.

Third, it provides insights into the well-being of software developers and holds much promise for better management of people involved in software development projects in general. Sentiment is a useful indicator of the social well-being of individuals and teams, as well as maintaining the social structure of communities (Pappas et al. 2018). Further, in order for sustainability to be meaningful, it must be achievable and measurable (Solow 1993), sentiment as an indicator is therefore valuable to sustaining OS communities. For example, it can provide companies the opportunity to develop interventions that improve the quality of life and well-being of its software developers, which in turn would reduce health care costs, as prevention is better than a cure (Rogers et al. 2012).

7 Conclusion, Limitations, and Future Action

Sustaining a large, evolving OS community is indeed challenging, as these communities by their nature are exigent, in the sense of sustaining positive sentiment in a dynamic, continuously changing pressurised environment. This study demonstrates that it is feasible to extract data from a mailing list and analyse with high accuracy the sentiment and discursive manifestations of contradictions of an OS community.

There are limitations to this work that warrant further investigations. *First*, we categorised discursive manifestations of contradictions into four neat categories (e.g. double bind, conflicts, critical conflicts, and dilemmas) by following a similar approach to Rousinopoulos et al. (2014) where the message body of the patch review was split into tokens and using a rule-based algorithm in combination with the two dictionaries, assigned a positive, neutral, or negative score. We believe that focusing on academic rigour was critical due to the novelty of the study but acknowledge that using a different analytical approach or less rigor, some tokens may fit into more than one category or none at all. Future studies could adopt other analytical approaches (i.e. machine learning) or conduct a manual analysis of the same data. Future studies could also extend the four categories proposed by Engeström and Sannino (2011). *Second*, since all sentiment analysis tools have limitations, researchers need to assess the suitability of such tools for their research project and to carefully understand the social context of the research in order to draw meaningful and actionable insights. While we used two popular tools (e.g. Opinion Lexicon, Comparative Words), researchers should carefully evaluate their chosen tools in the specific context of usage before building something on top of them (Lin et al. 2018). *Third*, analysis of sentiment is a snapshot in time and therefore is not always representative of the OS community being studied. This study does however present opportunities for future work in order to gain a deeper understanding of the relationship between discursive manifestations of contradictions and sentiment, as well as the propensity of individual patch reviewers over time. Future work could include a longitudinal study over a number of years and/or compare sentiment across multiple OS projects. This study highlights the importance of not only considering sentiment as statistical values but to take into consideration the context of the sentiment and how discourse can directly and indirectly have a positive or negative impact on people within an OS community.

Acknowledgements This work was supported with the financial support of the Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero - the Science Foundation Ireland Research Centre for Software (www.lero.ie).

References

Allen, D. K., Brown, A., Karanasios, S., & Norman, A. (2013). How Should Technology-Mediated Organizational Change Be Explained? A Comparison of the Contributions of Critical Realism and Activity Theory. *MIS Quarterly, 37*, 835–854.

Appleyard, M. M., & Chesbrough, H. W. (2017). The dynamics of open strategy: from adoption to reversion. *Long Range Planning, 50*(3), 310–321.

Aue, A., & Gamon, M. (2005). Customizing sentiment classifiers to new domains: A case study. In Proceedings of recent advances in natural language processing (RANLP) 1(3), 1–2.

Banville, C., & Landry, M. (1989). Can the Field of MIS be Disciplined? *Communications of the ACM, 32*, 48–60.

Barham, A. (2012). The impact of formal QA practices on FLOSS communities–the case of Mozilla. In Proceedings of 2012 *IFIP International Conference on Open Source Systems* (pp. 262–267). Berlin: Springer

Baysal, O., & Malton, A. J. (2007, May). Correlating social interactions to release history during software evolution. In Proceedings of the *Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007)* (pp. 7–7). IEEE.

Bednar, P. M., & Welch, C. (2020). Socio-Technical Perspectives on Smart Working: Creating Meaningful and Sustainable Systems. *Information Systems Frontiers, 22*(4), 281–229. https://doi.org/10.1007/s10796-019-09921-1.

Berns, M., Townend, A., Khayat, Z., Balagopal, B., Reeves, M., Hopkins, M. S., & Kruschwitz, N. (2009). The business of sustainability: what it means to managers now. *MIT Sloan Management Review, 51*(1), 20–26.

Bertelsen, O. W., & Bødker, S. (2000). Introduction: Information technology in human activity. *Scandinavian Journal of Information Systems, 12*(1), 3.

Beynon-Davies, P. (2010). The enactment of significance: a unified conception of information, systems and technology. *European Journal of Information Systems, 19*(4), 389–408.

Bird, C., Gourley, A., Devanbu, P., Gertz, M., & Swaminathan, A. (2006). Mining email social networks. In Proceedings of the 2006 International Workshop on Mining Software Repositories (pp. 137–143). ACM.

Carver, J., Capilla, R., Penzenstadler, B., Serebrenik, A., & Valdezate, A. (2018). Gender, sentiment and emotions, and safety-critical systems. *IEEE Software, 35*(6), 16–19.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. Chicago: SPSS Inc, 16.

Chaudhury, A., Mallick, D., & Rao, H. R. (2001). Web channels in e-commerce. *Communications of the ACM, 44*(1), 99–104.

Chen, H., Chiang, R. H., & Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS Quarterly, 36*(4), 1165–1188.

Chen, R., Sharman, R., Rao, H. R., & Upadhyaya, S. J. (2013). Data Model Development for Fire Related Extreme Events: An Activity Theory Approach. *MIS Quarterly, 37*, 125–147.

Cole, M., & Engeström., Y. (1993). *A cultural-historical approach to distributed cognition* (pp. 1–46). Distributed cognitions: Psychological and educational considerations.

De Choudhury, M., & Counts., S. (2013). Understanding affect in the workplace via social media. In Proceedings of the 2013 Conference on Computer Supported Cooperative Work (303–316). New York: ACM

De Dreu, C. K. W. & Van De Vliert., E. (1997). Introduction: Using conflict in organizations.

Dennehy, D., & Conboy, K. (2019). Breaking the flow: a study of contradictions in information systems development (ISD). *Information Technology & People, 33*(2), 477–501. https://doi.org/10.1108/ITP-02-2018-0102.

Dionne, P., & Bourdon, S. (2018). Contradictions as the driving force of collective and subjective development group employment programmes. *Journal of Education and Work, 31*(3), 277–290.

Ditsa, G. (2003). Activity theory as a theoretical foundation for information systems research. *Information Management: Support Systems & Multimedia Technology*,192–231.

Ducheneaut, N. (2005). Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work (CSCW), 14*(4), 323–368.

Engestrom, Y. (1987). Learning by expanding. *Helsinki: Orienta-Konsultit Oy.*

Engeström, Y. (1999). Activity theory and individual and social transformation. *Perspectives on activity theory, 19*(38), 19–30.

Engestrom, Y. (2000). Activity theory as a framework for analyzing and redesigning work. *Ergonomics, 43*(7), 960–974.

Engeström, Y. (2001). Expansive learning at work: Toward an activity theoretical reconceptualization. *Journal of education and work, 14*(1), 133–156.

Engeström, Y., & Kerosuo, H. (2007). From workplace learning to inter-organizational learning and back: the contribution of activity theory. *Journal of Workplace Learning, 19*(6), 336–342.

Engeström, Y., & Sannino, A. (2011). Discursive manifestations of contradictions in organizational change efforts: A methodological framework. *Journal of Organizational Change Management, 24*(3), 368–387.

Fitzgerald, B., & Adam, F. (2000). The status of the IS field: historical perspective and practical orientation.

Foot, K. A. (2001). Cultural-historical activity theory as practice theory: Illuminating the development of conflict-monitoring network. *Communication Theory, 11*(1), 56–83.

Gamalielsson, J., & Lundell, B. (2014). Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved? *Journal of Systems and Software, 89*, 128–145.

García-Cumbreras, M., Montejo-Ráez, A., & Díaz-Galiano, M. C. (2013). Pessimists and optimists: Improving collaborative filtering through sentiment analysis. *Expert Systems with Applications, 40*(17), 6758–6765.

Germonprez, M., Kendall, J. E., Kendall, K. E., Mathiassen, L., Young, B., & Warner, B. (2017). A theory of responsive design: A field study of corporate engagement with open source communities. *Information Systems Research, 28*(1), 64–83.

Gupta, A., Deokar, A., Iyer, L., Sharda, R., & Schrader, D. (2018). Big data and analytics for societal impact: Recent research and trends. *Information Systems Frontiers, 20*(2), 185–194.

Guzman, E., Azócar, D., & Li, Y. (2014). Sentiment analysis of commit comments in GitHub: an empirical study. *In Proceedings of the 11th Working Conference on Mining Software Repositories* (352–355). New York: ACM.

Guzzi, A., Bacchelli, A., Lanza, M., Pinzger, M., & Deursen, A.-V. (2013). Communication in open source software development mailing lists. *In Proceedings of the 10th Working Conference on Mining Software Repositories (277–286)*. Piscataway: IEEE Press.

Hasan, H., & Banna, S. (2012). The unit of analysis in IS theory: The case for activity. Information Systems Foundations, 191.

Hasan, H., Gould, E., & Hyland, P. (1998). *Information systems and activity theory: tools in context*. Wollongong: University of Wollongong Press.

Hasan, H., Kazluaskas, A., & Crawford, K. P. (2010). Blending complexity and activity frameworks for a broader and deeper understanding of IS. *In Proceedings of the Thirty First International Conference on Information Systems (ICIS), St. Louis, USA*.

Helle, M. (2000). Disturbances and contradictions as tools for understanding work in the newsroom. *Scandinavian Journal of Information Systems, 12*(1), 7.

Hemetsberger, A., & Reinhardt, C. (2009). Collective development in open-source communities: An activity theoretical perspective on successful online collaboration. *Organization Studies, 30*(9), 987–1008.

Hertel, M., & Wiesent, J. (2013). Investments in information systems: A contribution towards sustainability. *Information Systems Frontiers, 15*(5), 815–829.

Hirschheim, R., Klein, H. K., & Lyytinen, K. (1996). Exploring the intellectual structures of information systems development: a social action theoretic analysis. *Accounting, Management and Information Technologies, 6*(1–2), 1–64.

Ho, S. Y., & Rai, A. (2017). Continued voluntary participation intention in firm-participating open source software projects. *Information Systems Research, 28*(3), 603–625.

Ho, S. Y., & Richardson, A. (2013). Trust and distrust in open source software development. *Journal of Computer Information Systems, 54*(1), 84–93.

Igira, F. T. (2008). The situatedness of work practices and organizational culture: implications for information systems innovation uptake. *Journal of Information Technology, 23*(2), 79–88.

Ilyenkov, E. V. (1974). Activity and knowledge. Philosophy and culture.

Jensen, C., & Scacchi, W. (2007). Role migration and advancement processes in OSSD projects: A comparative case study. *In Proceedings of the 29th international conference on Software Engineering* (pp. 364–374). Washington, D.C.: IEEE Computer Society.

Jongeling, R., Datta, S., & Serebrenik, A. (2015). Choosing your weapons: On sentiment analysis tools for software engineering research. *In Proceedings of the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME) (531–535)*. https://doi.org/10.1109/ICSM.2015.7332508.

Karanasios, S. (2018). Toward a unified view of technology and activity: The contribution of activity theory to information systems research. *Information Technology & People, 31*(1), 134–155.

Karanasios, S., & Allen, D. (2014). Mobile technology in mobile work: contradictions and congruencies in activity systems. *European Journal of Information Systems, 23*(5), 529–542.

Karanasios, S., Riisla, K., & Simeonova, B. (2017). Exploring the use of contradictions in activity theory studies: An interdisciplinary review.

Kietzmann, J. (2008). Interactive innovation of technology for mobile work. *European Journal of Information Systems, 17*(3), 305–320.

Klievink, B., Romijn, B. J., Cunningham, S., & de Bruijn, H. (2017). Big data in the public sector: Uncertainties and readiness. *Information systems frontiers, 19*(2), 267–283.

Korpela, M., Mursu, A., & Soriyan, H. A. (2001). Information systems development as an activity. *Computer Supported Cooperative Work (CSCW), 11*(1–2), 111–128.

Kuutti, K. (1996). Activity theory as a potential framework for human-computer interaction research. *Context and consciousness: Activity theory and human-computer interaction, 1744*.

Kuutti, K. (1999) Activity theory, transformation of work, and information systems design. *Perspectives on activity theory*: 360.

Kuutti, K., & Molin-Juustila, T. (1998). Information System Support for 'Loose'Co-ordination in a Network Organisation: an Activity Theory perspective. *Information Systems and Activity Theory: Tools in Context*: 73–92.

Lakhani, K. R., & Von Hippel, E. (2004). How open source software works:"free" user-to-user assistance. In Porceedings of the *Produktentwicklung mit virtuellen Communities* (pp. 303–339). Wiesbaden: Gabler Verlag.

Lin, B., Zampetti, F., Bavota, G., Di Penta, M., Lanza, M., & Oliveto, R. (2018). Sentiment analysis for software engineering: How far can we go? *In Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)* (94–104). Piscataway: IEEE.

Malaurent, J., & Karanasios, S. (2020). Learning from workaround practices: The challenge of enterprise system implementations in multinational corporations. *Information Systems Journal, 30*(4), 639–663.

Metcalfe, M. (2004). Theory: Seeking a plain English explanation. *JITTA: Journal of Information Technology Theory and Application, 6*(2), 13.

Mikalef, P., Pappas, I. O., Krogstie, J., & Pavlou, P. A. (2020). Big data and business analytics: A research agenda for realizing business value. *Information & Management, 57*(1), 103237.

Mistrík, I., Grundy, J., Van der Hoek, A., & Whitehead, J. (2010). Collaborative software engineering: challenges and prospects. In Collaborative Software Engineering (389–403). Berlin: Springer.

Mockus, A., Fielding, R. T., & Herbsleb, J. (2000). A case study of open source software development: the Apache server," ICSE '00: *In Proceedings of the 22nd International Conference on Software Engineering* (pp. 263—272). New York: ACM Press.

Mursu, A., Luukkonen, I., Toivanen, M., & Korpela, M. (2007). Activity Theory in information systems research and practice: theoretical underpinnings for an information systems development model. *Information Research: An International Electronic Journal, 12*(3), 3.

Nardi, B. A. (1996). Activity theory and human-computer interaction. *Context and consciousness: Activity theory and human-computer interaction* (Vol. 436, pp. 7–16). Cambridge: MIT Press.

Novielli, N., Girardi, D., & Lanubile, F. (2018). A Benchmark Study on Sentiment Analysis for Software Engineering Research. *In Proceedings of the 15th International Conference on Mining Software Repositories* (pp. 364–375). New York: ACM. https://doi.org/10.1145/3196398.3196403.

Nurolahzade, M., Nasehi, S. M., Khandkar, S. H., & Rawal, S. (2009). The role of patch review in software evolution: an analysis of the mozilla firefox. *In Proceedings of the joint international and annual ERCIM workshops on Principles of software evolution (IWPSE) and software evolution (Evol) workshops* (pp. 9–18). New York: ACM.

Ogawa, M., Ma, K. L., Bird, C., Devanbu, P., & Gourley, A. (2007). Visualizing social interaction in open source software projects. *In Proceedings of the 2007 6th International Asia-Pacific Symposium on Visualization* (pp. 25–32). Piscataway: IEEE.

Ortu, M., Destefanis, G., Adams, B., Murgia, A., Marchesi, M., & Tonelli, R. (2015). The jira repository dataset: Understanding social aspects of software development. *In Proceedings of the 11th international conference on predictive models and data analytics in software engineering* (pp. 1–4).

Ozer, M., & Vogel, D. (2015). Contextualized relationship between knowledge sharing and performance in software development. *Journal of Management Information Systems, 32*, 134–161.

Pappas, I.-O., Mikalef, P., Giannakos, M.-N., Krogstie, J., & Lekakos, G. (2018). Big data and business analytics ecosystems: paving the way towards digital transformation and sustainable societies. Berlin: Springer.

Paul, R., Bosu, A., & Sultana, K. Z. (2018). Expressions of Sentiments During Code Reviews: Male vs. Female. *In Proceedings of the 2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)* (pp. 26–37). Piscataway: IEEE.

Perrini, F., & Tencati, A. (2006). Sustainability and stakeholder management: the need for new corporate performance evaluation and reporting systems. *Business Strategy and the Environment, 15*(5), 296–308.

Pletea, D., Vasilescu, B., & Serebrenik, A. (2014). Security and emotion: sentiment analysis of security discussions on GitHub. *In Proceedings of the 11th working conference on mining software repositories* (pp. 348–351). New York: ACM.

Popovič, A., Hackney, R., Tassabehji, R., & Castelli, M. (2018). The impact of big data analytics on firms' high value business performance. *Information Systems Frontiers, 20*(2), 209–222.

Porter, M. E., & Kramer, M. R. (2006). Strategy & Society: The Link Between Competitive Advantage and Corporate Social Responsibility. *Harvard Business Review, 84*(12), 78–92.

Rigby, P. C., German, D. M., & Storey, M. A. (2008). Open source software peer review practices: a case study of the apache server. *In Proceedings of the 30th international conference on Software engineering* (pp. 541–550). New York: ACM.

Rogers, D. S., Duraiappah, A. K., Antons, D. C., Munoz, P., Bai, X., Fragkias, M., & Gutscher, H. (2012). A vision for human well-being: transition to social sustainability. *Current Opinion in Environmental Sustainability, 4*(1), 61–73.

Rousinopoulos, A., Robles, G., & González-Barahona, J. (2014). Sentiment Analysis Of Free / Open Source Developers: Preliminary Findings From a Case Study. *Electronic Journal of Information Systems, 13*(2), 1.

Rückriem, G. (2009). Digital technology and mediation: A challenge to activity theory. *Learning and expanding with activity theory* (pp. 88–111).

Ryu, C., Kim, Y. J., Chaudhury, A., & Rua, H.-R. (2005). Knowledge acquisition via three learning processes in enterprise information portals: Learning-by-investment, learning-by-doing, and learning-from-others. *MIS Quarterly, 29*, 245–278.

Sannino, A. (2008). Experiencing conversations: Bridging the gap between discourse and activity. *Journal for the Theory of Social Behaviour, 38*(3), 267–291.

Sethanandha, B. D. (2011). Improving open source software patch contribution process: methods and tools. *In Proceedings of the 33rd International Conference on Software Engineering* (pp. 1134–1135). New York: ACM.

Sethanandha, B. D., Massey, B., & Jones, W. (2010a). Managing Open Source Contributions For Software Project Sustainability. Management of Engineering & Technology, 2010. *In Proceedings of the Technology Management for Global Economic Growth* (pp. 1–9). IEEE. Portland International.

Sethanandha, B. D., Massey, B., & Jones, W. (2010b). On the need for OSS patch contribution tools. *In Proceedings of the Second International Workshop on Building Sustainable Open Source Communities (Notre Dame, IN, USA, June 2010)..*

Shaikh, M., & Vaast, E. (2016). Folding and unfolding: Balancing openness and transparency in open source communities. *Information Systems Research, 27*(4), 813–833.

Sharif, K. Y., English, M., Ali, N., Exton., C., Collins, J. J., & Buckley, J. (2015). An empirically-based characterization and quantification of information seeking through mailing lists during open source developers' software evolution. *Information and Software Technology, 57*(3), 77–94.

Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of Data Warehousing, 5*(4), 13–22.

Shihab, E., Bettenburg, N., Adams, B., & Hassan, A. E. (2009). On the central role of mailing lists in open source projects: An exploratory study. *In Proceedings of the JSAI International Symposium on Artificial Intelligence* (pp. 91–103). Berlin: Springer.

Sholler, D., Steinmacher, I., Ford, D., Averick, M., Hoye, M., & Wilson, G. (2019). Ten simple rules for helping newcomers become contributors to open projects. *PLoS Computational Biology, 15*(9), e1007296.

Sinha, V., Lazar, A., & Sharif, B. (2016). Analyzing developer sentiment in commit logs. *In Proceedings of the 13th International Conference on Mining Software Repositories* (pp. 520–523). New York: ACM.

Slavova, M., & Karanasios, S. (2018). When Institutional Logics Meet Information and Communication Technologies: Examining Hybrid Information Practices in Ghana's Agriculture. *Journal of the Association for Information Systems, 19*(9), 4.

Solow, R. M. (1993). Sustainability: An economists perspective. Published in Dorfman, R. & Dorfman, NS (eds.) Selected readings in environmental economics.

Sowe, S. K., Stamelos, I., & Angelis, L. (2008). Understanding knowledge sharing activities in free/open source software projects: An empirical study. *Journal of Systems and Software, 81*(3), 431–446.

Tourani, P., Jiang, Y., & Adams, B. (2014). Monitoring sentiment in open source mailing lists: exploratory study on the apache ecosystem. I*n Proceedings of 24th Annual International Conference on Computer Science and Software Engineering* (pp. 34–44). Armonk: IBM Corp.

Turney, P. D. (2002). Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. *In Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 417–424). Stroudsburg: Association for Computational Linguistics.

Valecha, R., Rao, R., Upadhyaya, S., & Sharman, R. (2019). An activity theory approach to modeling dispatch-mediated emergency response. *Journal of the Association for Information Systems, 20*(1), 2.

Vermeulen, H., Gain, J., Marais, P., & O'Donovan, S. (2016). Reimagining gamification through the lens of Activity Theory. *In Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)..*

Wang, J., Shih, P. C., Wu, Y., & Carroll, J. M. (2015). Comparative case studies of open source software peer review practices. *Information and Software Technology, 67*(1), 1–12.

Weick, K. E. (1989). Theory construction as disciplined imagination. *Academy of management review, 14*, 516–531.

Weißgerber, P., Neu, D., & Diehl, S. (2008). Small patches get in! *In Proceedings of the 2008 international working conference on Mining software repositories* (pp. 67–76). Leipzig: ACM.

White, L., Burger, K., & Yearworth, M. (2016). Understanding behaviour in problem structuring methods interventions with activity theory. *European Journal of Operational Research, 249*(3), 983–1004.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. *In Proceedings of the Human Language Technology and Empirical Methods in Natural Language Processing* (pp. 347–354). Stroudsburg: Association for Computational Linguistics.

Wiredu, G. O., & Sørensen, C. (2006). The dynamics of control and mobile computing in distributed activities. *European Journal of Information Systems, 15*(3), 307–319.

Xie, I., & Matusiak, K. (2016). *Discover digital libraries: Theory and practice*. Amsterdam: Elsevier.

Denis Dennehy is Director of the master's in business analytics programme at NUI Galway and funded investigator at the Lero research group. His research broadly encompasses information systems and emerging technologies, with specific interest in viewing technology-mediated human activity through the lens of Activity Theory. This research has been published in ranked journals such as Information Systems Frontiers, Information Technology & People, and Journal of Systems and Software.

Kieran Conboy is a Professor in Information Systems and leader of the Lero research group at NUI Galway. He previously worked for Accenture Consulting and the University of New South Wales in Australia. He has collaborated with many organisations such as Accenture, Atlassian, Cisco Systems, and Fidelity Investments, as well as many SMEs. He is on the board of the Irish Research Council and advisor to the EU Commission on funding programmes and calls. Kieran has published over 200 articles in leading international journals and conferences including Information Systems Research, the European Journal of Information Systems, IEEE Software and IEEE Computer.

Jennifer Ferreira is a senior lecturer at Te Herenga Waka - Victoria University of Wellington, New Zealand. She conducts inter-disciplinary research in partnership with organisations based in New Zealand, the UK, Canada, and Ireland. She regularly publishes on topics that span Design, Software Engineering, and Social Science perspectives. Her research focuses on the human aspects of software development, from investigating ways of enhancing collaborative work on diverse software teams, to the user experience design of software products.

Jaganath Babu is a technical/analytical research assistant with the Lero research group and a postgraduate student on the MSc. Business Analytics program at NUI Galway, Ireland. He holds an MBA from Anna University, Chennai, India, and BSc in Electrical and Electronics Engineering from Anna University, Chennai, India. He has worked on a number of technical projects involving data science, computer vision, natural language processing, and machine learning.