Bridging the Vendor-User Gap in Enterprise Cloud Software Development through Data-Driven Requirements Engineering

Short Paper

Philipp Hoffmann
University of Mannheim
Mannheim, Germany
hoffmann@uni-mannheim.de

Deborah Mateja
University of Mannheim
Mannheim, Germany
mateja@uni-mannheim.de

Kai Spohrer
University of Mannheim
Mannheim, Germany
spohrer@uni-mannheim.de

Armin Heinzl
University of Mannheim
Mannheim, Germany
heinzl@uni-mannheim.de

Abstract

The shift from on-premise to cloud software has fundamentally changed the interactions between enterprise software vendors and their users. Where user involvement has traditionally been a challenge, increasingly large amounts of user input now allow for data-driven requirements engineering (RE). Research has paid little attention so far to the changes entailed by data-driven RE and addressed neither technical nor empirical perspectives of data-driven RE in enterprise software development. We aim to understand how the increasing availability of large amounts of user input impact RE in enterprise cloud software development. We provide a conceptualization of the newly available user input and how it changes traditional RE. We collect and analyze rich data from multiple product units at a leading enterprise software company and examine the integration of user input into RE; specifically requirements discovery, prioritization, experimentation, and specification. We thereby aim to contribute to non-normative and empirical work on RE.

Keywords: Data-Driven Requirements Engineering, Enterprise Cloud Software, Software Development, Enterprise Software, Information Systems Development, Requirements Engineering

Introduction

In enterprise software development, user involvement has traditionally been a challenge. In the era of on-premise software, there has rarely been a direct interaction between enterprise software vendors and the users of their products. As such, on-premise software has been deployed on the systems of client organizations and maintained by client IT departments that acted as the point of contact for their respective users, independently of insourcing or outsourcing systems operations. Therefore both, users and enterprise software vendors, interacted with the customer-sided IT departments but not with each other. This yielded a prevailing disconnect between enterprise software vendors and the actual users of their products. Classical methods of requirement engineering (RE), e.g. focus groups, lead users, and workshops, aimed to bridge the gap but focused only on the attitudes and perspectives of few selected users. Consequently, this approach has led to severely imperfect requirements, even if RE methods have been applied rigorously. Systematically failing to meet the requirements of the users is, therefore, still one of the key development risks in enterprise software (Mathiassen et al. 2007).

The shift from on-premise software to enterprise cloud software products has fundamentally changed the interactions between vendors and users. Relying on agile development practices, enterprise software vendors have established shorter development iterations (Hoda et al. 2013) and most enterprise cloud software products are updated frequently for all users at the same time. These changes have led to shorter innovation cycles with incremental updates of cloud products that often consist of only a few, innovative features (Choudhary 2007). In contrast to on-premise solutions, enterprise cloud products thus usually start with basic core functionalities and evolve with additional features while users are already using the product. This has shifted much product development time from the pre-delivery phase to the post-delivery phase. At the same time, enterprise cloud software providers are technically able to measure nearly every user-interaction with their cloud product. This provides unprecedented insights into the actual usage of their software products. Furthermore, the increasing number of new feedback channels, like social media, forums, or reviews, provide additional user input. Yet it is unclear how these novel sources of user input are incorporated in the continuous development of enterprise cloud products and whether they support enterprise software vendors in enhancing the fit between their product and their users' needs.

Prior work on RE has referred to the integration of the massive amounts of newly available user input into RE as *data-driven RE*. However, research on RE has paid little attention to the changes that are entailed by data-driven RE. On the one hand, the majority of prior work on RE followed a normative stance and was less concerned with understanding or explaining actual RE practices in industry (Mathiassen et al. 2010). On the other hand, non-normative research on RE has primarily focused on the interactions between user representatives and development organizations to understand the various issues and socio-cognitive processes that make RE so challenging (Chakraborty et al. 2010). In spite of their tremendous importance, these lines of research do not provide a full understanding of how RE endeavors in industry integrate the newly available data related to enterprise software users. Prior work neither addressed the technical nor the empirical perspective of data-driven RE in enterprise software development. Against this backdrop, this research project aims to answer the question:

RQ: How does the increasing availability of large amounts of user input impact RE in enterprise cloud software development (ECSD)?

We first delimit data-driven RE from traditional RE in ECSD. Second, we examine which data are becoming available to enterprise cloud software providers and categorize their utilization among different requirement development techniques. Our finings serve as a foundation for a subsequent study to investigate how this development alters the activities of RE in ECSD and what organizational factors influence the successful integration of user input data in RE for ECSD.

Data-driven versus Traditional Requirements Engineering

In the academic literature, RE is often described as a "staged sequence of activities and/or task objectives" (Chakraborty et al. 2010) based on a normative and deterministic perspective (e.g. Davis and Hickey 2002; Sommerville 2007). We follow a non-normative approach that perceives RE as encompassing the elicitation, validation, and documentation of desired properties and necessary constraints of a software, as well as the frequent validation and continuous management of the same (Pohl 2010). Mathiassen et al. (2007) derived a four-dimensional categorization of requirement development techniques to overcome the most prevalent risks: **Requirements Discovery (D)** techniques are user-centric and facilitate the prediction and identification of emerging requirements. **Requirements Prioritization (P)** addresses resource-centric analyses and decision support to focus on the elaboration of the relevant requirements. **Requirements Experimentation (E)** techniques apply software-centric designs as a key means for communicating with and iteratively involving users. **Requirements Specification (S)** describes the documentation-centric abstraction and textual/graphical representation of a requirement. We build on these techniques to structure our preliminary results in the later sections of this paper.

Data-driven RE essentially relies on the same high-level activities as other approaches to RE. However, it expands these activities by building on an extended set of user inputs. Data-driven RE has just recently emerged as both, a research stream and an organizational process. Hence, it has not yet been clearly defined as well as delimited from related RE approaches and contextualized to ECSD. There are only few studies directly referring to data-driven RE, which can be used to identify unique features of data-driven RE as compared to traditional RE. Maalej et al. (2016) motivate a data-driven RE approach for practitioners,

distinguishing explicit and implicit data as the novel source of user input and highlighting the need to systematically aggregate the new amounts of data. Q-Rapids, a data-driven approach to quality requirements engineering, was introduced and discussed in previous work (e.g. Franch et al. 2017). Nayebi (2018) provides a short summary on the motivation and existing literature of data-driven RE to introduce a conference panel, while Czarnecki (2018) describes data-driven RE in the specific context of cyber-physical systems. Regarding the enhanced user input in data-driven RE, the following common characteristics can be derived from previous research:

- **Context-awareness:** User input is no longer restricted to the explicit expression of their needs and preferences. Those traditional sources of feedback can be enriched with information about the actual context of the usage of a software system.
- **Continuity:** Requirements are no longer elicited before and after the development phase and monitored during commercial distribution. They are rather continuously re-elicited and re-prioritized allowing for continuous software improvement and adaption to user needs and preferences.
- **Automation:** In the light of increasing volumes of user input (feedback data and usage data), it is no longer suitable to analyze usage data manually and deduce requirements by hand. Also, new user input becomes increasingly available through software. This allows for automated acquisition, processing and smart interpretation of software requirements.

Data-driven RE constitutes a novel and relevant approach to RE which emphasizes the user as the central stakeholder in the context of large, distributed user bases that have yet remained outside an enterprise software vendor's reach. A key characteristic of data-driven RE is the specific type of comprehensive user input that becomes an integral part of the RE process.

User Input in Data-driven Requirements Engineering

In traditional RE, the elicitation of user feedback is limited to few inquiry techniques. The main techniques are interviews, workshops, focus groups, survey questionnaires, and field observations (Liu et al. 2017; Maalej et al. 2016; Olsson and Bosch 2015). RE can, therefore, be described as knowledge transfer where "(user) representative(s) have some knowledge and understanding of the system requirements, and the analysts use techniques (..) to arrive at a shared understanding of those requirements" (Chakraborty et al. 2010). These approaches have several disadvantages: Firstly, what users articulate within these formats of feedback inquiry does not necessarily match what they really need; in fact, it seldomly does (Olsson and Bosch 2015). Additionally, all articulations are retrospective, i.e. the problem situation and the context took place in the past (Maalej et al. 2009). The retrospective view causes a significant communication gap between users and the development teams which induces incomplete or wrong requirements. Moreover, these techniques require active customer participation and are only capable to involve a small subset of users, specifically those, that are within the organizational reach (Olsson and Bosch 2015; Oriol et al. 2018; Seyff et al. 2014). Moreover, users have only limited capabilities to provide feedback while productively using software. Concequently, they can only report very limited context without being distracted from their current activity (Seyff et al. 2014). Lastly, all of these methods require a lot of time to be executed, which becomes unsuitable for fast changing software products (Li et al. 2010).

Although many novel sources of user input promise to help overcome these deficiencies of traditional sources (Maalej et al. 2016), only few studies attempt to structure and test their potential (Fabijan et al. 2015; Pacheco et al. 2018). Thus, current research lacks a common understanding for data-driven RE and an alignment of the streams of literature regarding the divergent types of data.

User input is an umbrella term for any type of reflection of a user's behavior and experience with a software. The novel sources of user input comprise two categories which have been identified in previous work. However, this line of work has yielded a number of diverging definitions and terminologies to describe user input. For example, qualitative data versus quantitative data (Bosch-Sijtsema and Bosch 2015; Maalej and Pagano 2011; Olsson and Bosch 2015), explicit data vs. implicit data (Maalej et al. 2016; Sammaneh, 2018) and feedback versus monitoring (Oriol et al. 2018).

Albeit valuable, these conceptualizations have two shortcomings: First, they do not address the above-mentioned disadvantages of user input in traditional RE and, second, do not incorporate both categories

for requirement engineering. To overcome these deficiencies, we provide a more holistic definition of user input in data-driven RE. As such, we differentiate two categories of user input, namely:

Feedback Data*: The subjective user perceptions of a software reflected in explicit articulations of a user.*

Usage Data: *The usage of a software reflected in user behaviors and user interactions with a product.*

This distinction is expedient in data-driven RE regarding the different research streams on analytics: Feedback data analysis is mostly relying on text analytics and natural language processing techniques (e.g. Panichella et al. 2015, Iacob and Harrison 2013), whereas usage data analysis is often utilizing process mining approaches (e.g. Dąbrowski et al. 2017). Based on the two categories of user input, we define data-driven RE by extending a traditional RE definition by Maalej et al. (2016) as follows (depicted in Figure 1):

Data-driven RE *enriches the methods of collecting and analyzing user input in traditional RE with the automated and continuous analysis of novel feedback sources as well as with the analysis of context-aware usage data to identify, prioritize, document and manage requirements for a software product.*

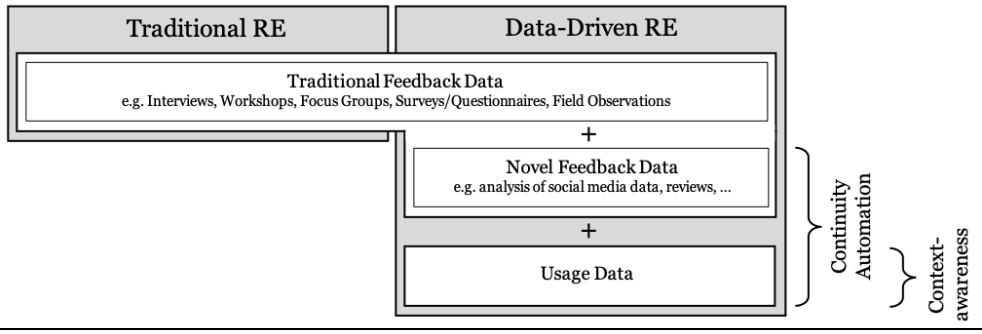

Figure 1. User Input in Data-driven RE compared to traditional RE

In summary, previous research on data-driven RE focused on motivating ideas and designing specific prototypes regarding the idea of data-driven RE only from a conceptional level. To the knowledge of the authors, no empirical research has been conducted in view of the novel and comprehensive concept of data-driven RE. Especially within the specific area of ECSD, it can be anticipated that the enterprise cloud software market experiences a fast transition towards data-driven RE on the basis of the described client-vendor relationship.

Consequently, our research in progress focuses on the comprehensive in-depth investigation of available user input and investigates how it can bridge the gap between user needs and elicit requirements in the RE process in ECSD. It aims to contribute to the existing literature on information systems development by empirically identifying changes in RE in the light of the emerging user input data in ECSD.

Research Design

To understand data-driven RE in ECSD, we have been able to gain access to one of the world's leading enterprise cloud software vendors. We are in the process of conducting an explorative multi case study into the incremental development of cloud products following Yin (2009). The empirical study tackles prevailing gaps in research, especially the lack of a comprehensive understanding of the changes involved in data-driven RE that go beyond ideas and narrow prototypes. As these gaps remain untouched by literature, our study is explorative by nature. A qualitative approach is considered appropriate because RE often differs tremendously from product to product, even within single organizations (Fernández and Wagner 2015). It is consequently necessary to take the product context into account when studying RE. Thus, the unit of analysis in our study is a single product organization (Yin 2009).

We have sampled cases with a necessary degree of comparability and a sufficient level of contrast to investigate how development teams from different enterprise cloud products engage in data-driven RE. We draw all cases from a single company because this company provides excellent conditions to study RE in ECSD as it is a leading provider of enterprise cloud software products that has its roots in on-premise software business. We sampled for theoretical replication (Yin 2009) such that the cases are comparable

but offer enough contrasts to identify those influences that can be generalized to understand the successful integration of newly available user input in RE.

To assure that a multi case-study research design within our target company is able to yield results from comparable but nonetheless diverse settings, we reviewed company internal documentations of the methodologies and processes of RE and interviewed the central process manager for RE. This procedure has shown that comprehensive topics requiring joint efforts of multiple teams (e.g. tool support for RE) are dealing with the organizational level of the case company which provides a significant degree of comparability between the different product units. However, daily RE practices in the case company are dealt with on product unit level. Hence, also we find substantial contrasts along the product units.

Case Company Description

During the last decade, the case company underwent a transition from an on-premise enterprise software vendor towards a cloud software vendor. The company is industry-representative in multiple ways due to the large software product portfolio ensuring the coverage of different types of software architecture, varying target industries, and the diverse size and complexity of software products. Due to various acquisitions of other enterprise software vendors including start-ups, multiple perspectives on RE methods exist throughout the company. For example, the case company is currently integrating a recently acquired company into its set of product units that offers services focusing on the elicitation of feedback and usage data from their users. The case company provides a highly diversified setting for the study at hand. It allows us to observe cases at both extremes, i.e. those still relying solely on traditional RE and those eliciting and managing requirements in a more data-driven manner.

Collected Data and Analytical Procedures

Besides the exclusive use of usage data and feedback data for requirements elicitation, we derived three additional categories to combine the two data types from the literature review: (1) Usage data enriching/validating feedback data (e.g. Dąbrowski et al. 2017), (2) feedback data enriching/validating usage data (e.g. Olsson and Bosch 2015) and (3) collectively joint feedback and usage data (e.g. Oriol et al. 2018). For reasons of conciseness, we only present an aggregated perspective of the three categories in this paper and derived a more generic conceptual framework (see Figure 2). Rather than as a causal model, we utilize this conceptual framework as a tool to guide our data collection efforts within the cases as well as by structuring our questions during the interviews. We focus on the two types of user input, namely traditional and novel feedback data and usage data, as well as how they are introduced into and processed within the RE process of a product unit to elicitate requirements by following the four-dimensional categorization of requirement development techniques of Mathiassen et al. (2007). Following prior work on socio-cognitive processes in RE (Chakraborty et al. 2010), we assumed that creating aligned perspectives on potential requirements would constitute a key challenge and effortful process in all cases and be influenced by feedback and usage data. We find that product units in our sample differ in the degree of context-aware collection of user input, the continuity of their data collection efforts, and the degree of automation regarding collection and analysis of user input for RE. We have collected detailed data about challenging events as well as successful and failed uses of user input in all cases. A particularly fruitful angle for data collection has so far been how more objective usage data compared to more subjective (traditional-) feedback methods (c.f. Chakraborty et al. 2015) helped development teams align their perspectives on single requirements that had diverged based on feedback data only.

We have been collecting data from various sources, including company internal documentations, dashboards, and system reporting, as well as semi-structured interviews. Interviews were scheduled for 60 minutes with 15 persons, mostly with face-to-face meetings or via video conference. Interviewees were selected at the interface between the ECSD teams and customers. Similar to most large development organizations that rely on Scrum or scalable agile development methods, RE tasks at the case company are primarily conducted by two specific job roles: product owners and the product managers. The former are the functional leads in a development team who define the backlog to guide the development efforts of a team. They can directly decide what kind of user input they consider during the definition of the backlog. Product managers drive direct customer and user interactions and are, hence, responsible for capturing requirements of customers and users and for transferring these requirements to the development teams' product owner. Despite the above distinction between product owners and product managers conveying

the impression of clearly distinguishable responsibilities, this is not the case in the daily operation of the case company and varies across product units. Hence, depending on the interaction of the two roles in each product unit, the number of interviewees and job roles that we took into account varies. However, we could ensure that the interviewees were not responsible for multiple products in our selected cases.

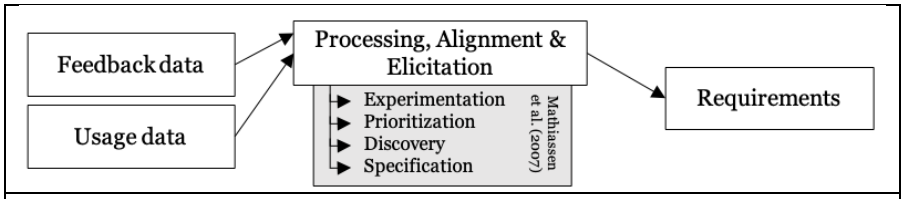

Figure 2. Conceptual Framework

Our semi-stuctured interviews are conducted as follows: After a short introduction, the interviewees are asked how they currently elicit requirments in the post deployment phase. Afterwards they are asked which new sources of feedback and usage data they currently integrate as well as how and for what they make use of them. Furthermore, we also asked for current and planned projects of integrating further user input. During the interviews, we added further, more specific sub-questions to deepen our understanding.

In a two-fold approach, consisting of a within-case and a cross-case analysis, as proposed by Miles and Huberman (1994), the product units are analyzed while data collection is still under way. Each product unit is individually investigated in the light of their utilization of feedback data and usage data. Specifically, the approach to eliciting and processing feedback and usage data are investigated. Afterwards, the results are analyzed between cases to identify common enablers, inhibitors, and shared patterns in the respective utilization and processing of the feedback and usage data. The aim is to comprehensively capture data-driven RE within its organizational context of the ECSD.

Preliminary Results

In order to investigate the impact of the increasing availability of large amounts of user input on RE in ECSD, so far nine interviews across four product units have been conducted, and additional documentation suggested by the interviewees such as dashboard manuals and articles from the internal knowledge sharing platform has been reviewed. At the moment we are planning to conduct in total of 15 interviews across seven product units within this study.

The selected cases provide a broad variety of approaches to RE. While some product teams are pioneering the integration of additional user input for data-driven RE, others stick to more traditional RE practices. Table 1 introduces preliminary evaluations of four cases and examples of current collection of usage data and feedback data. Feature usage describes how often a feature is used by a user, e.g. measured through clicks on the respective button representing the feature, and measuring failure refers to records of errors occuring while a user is engaged in an operation using a feature, e.g. through response codes of an API. Discussion forums and ticketing systems are analysed for feedback data regarding software features.

The preliminary results suggest that the increasing availability of large amounts of user input is impacting the nature and execution of the major RE techniques that constitute the building blocks of RE processes in enterprise software development. However, the three characteristics of data-driven RE (context-awareness, continuity and automation) are only partly observable in practice: Although context-awareness, continuity are already observable using novel forms of feedback and usage data at the case company there are still struggels to automate the feedback analysis. We structure the presentation of our findings using the framework of the major RE techniques as proposed by Mathiassen et al. (2007):

Requirements Experimentation (E): Our first findings indicate that requirements experimentation can act as an entry point to utilizing usage date in data-driven RE. Yet, neither A/B-Tests nor canarying (e.g. Harman et al. 2013), were extensively applied for testing software features deployed to a limited group of users. Instead, feature usage was analyzed to measure the adoption of new features and to adapt features to increase the adoption rate (cases A, C). Furthermore, experiments with response times of UIs were conducted to optimize interfaces (case C). In-app feedback was used to get direct user feedback for new features (cases A, C).

Requirements Prioritization (P): Based on a central usage monitoring tool, several product units acquired simple measures of feature usage (e.g. usage frequency) as a starting point for further feature refinement. By combining this information with user feedback, especially problems and ideas known from the customer support function, the product owners used this information to prioritize the necessary improvments and refinements for existing features (cases B, C). Further, licencing information e.g. (trial or paying user) were analyzed (case C) as well as frequencies of specific errors (case B). One observed example is the planning of investments in software features based on their actual usage where improvements of frequently used features were prioritized over changes to rarely used features. The data-driven prioritization of requirements encouraged fact-based decisions compared to more subjective decisions based on occasional user feedback.

Requirement Discovery (D): All cases used some new user input for the requirement discovery, especially from public online forums, blogs and communities (cases B, C) as well as information from support systems (cases A, B, C, D). Furthermore, individual in-app feedback was analyzed (cases A, C). Several cases measured and tracked failures in executing features during user operations (cases A, B, C). Error data was analyzed to identify and overcome reasons for their appearance. Additionally, we found one product unit experimenting with click streams to measure the time of using process-oriented software features (end-to-end workflows) and identify workarounds of users (case A). The integration of click stream data allowed the team to discover which software features were complicated to use and needed to be redesigned.

Requirements Specification (S): Until now, no implementations of data-driven requirements specification could be observed in the case company. Related research on feedback analytics (e.g. Panichella et al. 2015; Iacob and Harrison 2013) leads us to assume that the automated analysis of customer incidents for eliciting requirements should already be in place. However, we have so far not been able to find any instantiations of this technique in the case company.

		Product A	**Product B**	**Product C**	**Product D**
Number of customers		Few	Moderate	Many	Few
Point in time of initial release		Young	Young	Established	Moderate
Industry focus		Independent	Independent	Independent	Digital Products
Type of software		Application	Platform	Application	API-based Service
Usage data	Feature usage	Yes	Yes	Yes	No
	Measuring failure	No	Yes	Yes	No
Feedback data	Discussion forums	No	Yes	Yes	No
	Ticketing Systems	Yes	Yes	Yes	Yes
RE techniques according to Mathiassen et al. (2007)		E, D	P, D	P, E, D	D

Table 1. Cases with examples of usage and feedback data usage

Expected Contributions

We expect this study to make two major contributions. First, we aim to provide a detailed account of the technical and interactional changes involved in the transformation from traditional RE to data-driven RE in ECSD. In doing so, we extend prior non-normative and empirical work on RE that primarily focused on the social and socio-cognitive interactions between development organizations and user representatives (e.g. Chakraborty et al. 2010). We provide a first attempt to capturing the influences of newly available amounts of user input on RE as a whole, going beyond conceptual ideas and overly specific prototypes to rather emphasize data-driven RE as a comprehensive organizational practice. Second, we aim to contribute to practice by providing a guide for product units on their journey to data-driven RE in ECSD that allows companies to identify and work on enabling and inhibiting factors of the different approaches. We thus identify relevant aspects to focus on while steering data-driven RE endeavors.

Conclusion and Next Steps

This project synthesizes knowledge from different streams of research, including requirements engineering, analytics, and ECSD as the specific domain. It constitutes a starting point for conceptualizing and eventually implementing prototypes for data-driven approaches to RE in the specific context of ECSD. It tackles prevailing challenges, namely the involvement of users in the RE for enterprise software development.

Our multi-case study focuses on the increasingly large amounts of available user input within data-driven RE in ECSD. It will outline influencing factors underlying the approach to data-driven RE across multiple product units in the case company. During the next months, we aim to complete our data collection and extend our preliminary results by engaging in more detailed within-case and cross-case analyses. At ICIS, we aim to present results of the completed data collection, including a variety of archived documentations and a total of 15 interviews across seven product units. We will be able to outline the utilized feedback and usage data and how they were integrated into RE activities including requirements discovery, prioritization, experimentation, and specification. We aim to draw on literature on traditional RE, such as Chakraborty et al. (2010), that considers social and interactional activities as the central aspects of RE processes. We will identify how data-driven RE alters these activities and will provide a model that integrates the multiple activities into a process-driven understanding of RE in ECSD.

We expect our study on data-driven RE in ECSD to be a springboard for future research. Follow-up studies may want to examine to which degree data-driven RE is more effective in enterprise software development than traditional RE and how this influences the quality of enterprise software products and enterprise cloud software subscriptions. Whereas this study specifically focuses on refined user input and its integration in RE for incremental enterprise software development, data-driven RE can eventually also incorporate data that goes beyond the user, opening up an even greater space of options. As Bosch and Olsson (2016) propose, data-driven RE may even go as far as allowing for autonomous and dynamic software evolvement, which shifts the development effort from software developers to the system itself, which would constitute a new paradigm to software engineering.

References

Bosch, J., & Olsson, H. H. 2016. Data-Driven Continuous Evolution of Smart Systems. 2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS), pp. 28–34. (doi: 10.1109/SEAMS.2016.011)

Bosch-Sijtsema, P., & Bosch, J. 2015. User involvement throughout the innovation process in high-tech industries. Journal of Product Innovation Management (32:5), pp. 793-807. (doi: 10.1111/jpim.12233)

Chakraborty, S., Sarker, S., & Sarker, S. 2010. An exploration into the process of requirements elicitation: A grounded approach. Journal of the association for information systems (11:4).

Chakraborty, S., Rosenkranz, C., & Dehlinger, J. 2015. Getting to the shalls: facilitating sensemaking in requirements engineering. ACM Transactions on Management Information Systems (TMIS) (5:3).

Choudhary, V. 2007. Software as a service: Implications for investment in software development. In 2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07), pp. 209a-209a. IEEE. (doi: 10.1109/HICSS.2007.493)

Czarnecki, K. 2018. Requirements engineering in the age of societal-scale cyber-physical systems: the case of automated driving. In 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 3-4, IEEE. (doi: 10.1109/RE.2018.00-57)

Dąbrowski, J., Kifetew, F. M., Muñante, D., Letier, E., Siena, A., & Susi, A. 2017. Discovering requirements through goal-driven process mining. 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), pp. 199-203. (doi: 10.1109/REW.2017.61)

Davis, A. M., & Hickey, A. M. 2002. Requirements researchers: Do we practice what we preach?. Requirements Engineering (7:2), pp. 107-111. (doi: 10.1007/s007660200007)

Fabijan, A., Olsson, H. H., & Bosch, J. 2015. Customer Feedback and Data Collection Techniques in Software R&D: A Literature Review. In J. M. Fernandes, R. J. Machado, & K. Wnuk (Eds.), Software Business (210), pp. 139–153.

Fernández, D. M., & Wagner, S. 2015. Naming the pain in requirements engineering: A design for a global family of surveys and first results from Germany. Information and Software Technology (57), pp. 616-643. (doi: 10.1016/j.infsof.2014.05.008)

Franch, X., Ayala, C., López, L., Martínez-Fernández, S., Rodríguez, P., Gómez, C., Jedlitschka, A., Oivo, M., Partanen, J., Räty, T. & Rytivaara, V. 2017. Data-driven requirements engineering in agile projects: the Q-rapids approach. In 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW), pp. 411-414. (doi: 10.1109/REW.2017.85)

Harman, M., Lakhotia, K., Singer, J., White, D. R., & Yoo, S. 2013. Cloud engineering is search based software engineering too. Journal of Systems and Software (86:9), pp. 2225-2241. (doi: 10.1016/j.jss.2012.10.027)

Hoda, R., Noble, J., & Marshall, S. 2013. Self-organizing roles on agile software development teams. IEEE Transactions on Software Engineering, 39(3), pp. 422-444. (doi: 10.1109/TSE.2012.30)

Iacob, C., & Harrison, R. 2013. Retrieving and analyzing mobile apps feature requests from online reviews. In Proceedings of the 10th Working Conference on Mining Software Repositories, pp. 41-44. IEEE.

Li, H., Zhang, L., Zhang, L., & Shen, J. 2010. A user satisfaction analysis approach for software evolution. In 2010 IEEE International Conference on Progress in Informatics and Computing (2), pp. 1093-1097 IEEE. (doi: 10.1109/PIC.2010.5687999)

Liu, L., Zhou, Q., Liu, J., & Cao, Z. 2017. Requirements cybernetics: elicitation based on user behavioral data. Journal of Systems and Software (124), pp. 187-194. (doi: 10.1016/j.jss.2015.12.030)

Maalej, W., Nayebi, M., Johann, T., & Ruhe, G. 2016. Toward Data-Driven Requirements Engineering. IEEE Software (33:1), pp. 48–54. (doi: 10.1109/MS.2015.153)

Maalej, W., & Pagano, D. 2011. On the socialness of software. In 2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, pp. 864-871, IEEE. (doi: 10.1109/DASC.2011.146)

Mathiassen, L., Rossi, M., Saarinen, T., & Tuunanen, T., 2007. A contigency model for requirements development. Journal of the Association for Information Systems, (8:11), pp. 569-597.

Miles, M. B., & Huberman, M. A. 1994. Qualitative Data Analysis: An Expanded Sourcebook. SAGE.

Nayebi, M. 2018. Data Driven Requirements Engineering: Implications for the Community. In 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 439-441. IEEE. (doi: 10.1109/RE.2018.00058)

Olsson, H. H., & Bosch, J. 2015. Towards Continuous Customer Validation: A Conceptual Model for Combining Qualitative Customer Feedback with Quantitative Customer Observation. In J. M. Fernandes, R. J. Machado, & K. Wnuk (Eds.), Software Business (210), pp. 154–166.

Oriol, M., Stade, M., Fotrousi, F., Nadal, S., Varga, J., Seyff, N., Abello, A., Franch, X., Marco, J., Schmidt, O. 2018. FAME: supporting continuous requirements elicitation by combining user feedback and monitoring. In 2018 IEEE 26th International Requirements Engineering Conference (RE), pp. 217-227, IEEE. (doi: 10.1109/RE.2018.00030)

Pacheco, C., García, I., & Reyes, M. 2018. Requirements elicitation techniques: a systematic literature review based on the maturity of the techniques. IET Software, 12(4), pp. 365–378. (doi: 10.1049/iet-sen.2017.0144)

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. 2015. How can i improve my app? classifying user reviews for software maintenance and evolution. 2015 IEEE international conference on software maintenance and evolution, pp. 281-290. (doi: 10.1109/ICSM.2015.7332474)

Pohl, K. 2010. Requirements Engineering: Fundamentals, Principles, and Techniques (1st ed.). Springer Publishing Company.

Sammaneh, H. 2018. Requirements Elicitation with the Existence of Similar Applications: A Conceptual Framework. In 2018 International Conference on Computer and Applications (ICCA), pp. 444-9, IEEE. (doi: 10.1109/COMAPP.2018.8460407)

Seyff, N., Ollmann, G., & Bortenschlager, M. 2014. AppEcho: A user-driven, in situ feedback approach for mobile platforms and applications. In Proceedings of the 1st International Conference on Mobile Software Engineering and Systems, pp. 99-108, ACM.(doi: 10.1145/2593902.2593927)

Sommerville, I. 2007. Software Engineering (8th ed.). Addison Wesley

Yin, R. K. 2009. Case study research: Design and methods (4 ed.). Los Angeles, CA: Sage.