Blurring Boundaries

Towards the Collective Team Grokking of Product Requirements

Rob Fuller(✉)

Electrical and Computer Engineering,
The University of British Columbia, Vancouver, B.C., Canada
rfuller@ece.ubc.ca

Abstract. Software development has become increasingly software 'product' development, without an authoritative 'customer' stakeholder that many requirements engineering processes assume exists in some form. Many progressive software product companies today are empowering cross-functional product teams to 'own' their product – to collectively understand the product context, the true product needs, and manage its on-going evolution – rather than develop to a provided specification.

Some teams do this better than others and neither established requirements elicitation and validation processes nor conventional team leadership practices explain the reasons for these observable differences. This research examines cross-functional product teams and identifies factors that support or inhibit the team's ability to collectively create and nurture a shared mental model that accurately represents the external product domain and its realities. The research also examines how teams use that collective understanding to shape development plans, internal and external communications, new team member onboarding, etc.

We are engaged with several software product companies using a constructivist Grounded Theory method towards the research question.

Early results are emerging as organisational factors, within and surrounding the teams. One emerging observation relates to the degree to which functional distinctions are treated as demarcations or blurred boundaries. The other observation is the impact an expectation of mobility has on an individual's sense of feeling part of the collective team versus solely being a functional expert. This also becomes a factor in the first observation.

The research is in-progress but early observations are consistent with a basic element of empathy that a certain blurring of the boundaries is necessary for a period of time in order to better understand the other context. Future research will examine whether the observed organisational factors are pre-conditions for the team being able to collectively understand the context of the product requirements, collectively and deeply.

Keywords: Empathy-driven development · Collective sensemaking · Design science · Requirements validation · Team learning · Knowledge management · Complex adaptive systems · Tacit knowledge · Product team organisation

C. Cappiello and M. Ruiz (Eds.): CAiSE Forum 2019, LNBIP 350, pp. 87–98, 2019.
https://doi.org/10.1007/978-3-030-21297-1_8

1 Introduction

For those unfamiliar with its popular culture roots, the Oxford English Dictionary [1] defines the verb 'grok' as "understand (something) intuitively or by empathy." Increasingly, the success of cross-functional software product development teams depends on the degree to which the team collectively groks, more than simply the product requirements, but the context for those requirements, the world for which their products are intended.

2 Historical Context

During the 1990s, three factors collectively contributed to a massive shift in the software landscape. The first was the continuing improvement in the price/performance ratio of computing which had brought PCs to every desktop in the workplace and to a great number of homes, creating a much more broad and diverse demand for software and carrying with it a broader range of needs and desires. The second contributing factor was the widespread introduction of graphical user interfaces on personal computers which caused the industry to examine human-computer interaction (HCI) in entirely new and vastly more complex ways. The third factor was the arrival of the Internet which affected everything from how we thought of using technology to business models themselves. These factors combined dramatically changed the 'art of the possible' in software.

By the late 1990s, a "model revolution" [2] began to emerge that took a different view on change, risk, and uncertainty. These 'agile' models typically embraced the possibility that requirements could change throughout the development effort in contrast to many earlier Software Development LifeCycles (SDLCs) that strived to lock down requirements in the specification and planning stages. They took the form of iterative and incremental approaches to solution development using cross-functional teams that attempt to 'discover' the needs throughout the development effort. This model viewed emergence as a fact of life rather than a failure of the requirements elicitation and analysis activities.

These process models had a greater focus on the software development 'team', usually cross-functional, as a critical success factor in delivering software. These teams often have the necessary collection of functional expertise and capacity in each functional area to be essentially self-sufficient. Many software development companies have gone even further, empowering their cross-functional teams to 'own' the product. This approach is now quite common, no longer adopted only by industry thought-leaders. It is these organisations and teams that are the main focus of this research.

3 The Problem

While agile models improve many of the issues that were breaking down during the crisis period of the 1990s, they still generally cling to the notion that there is a 'customer', an authoritative voice that the development team can interact with iteratively to clarify requirements and validate results. However, as software development has become less bespoke development and more 'product' development intended for a market, a new and critical challenge exists for software teams, especially those now empowered to own the product, and that is how to gain a deep understanding of the world for which their product is intended. Certainly, techniques to 'hear' from the market are helpful but, as Polyani [3] noted, this is tacit knowledge that these market participants have and people can know more than they can tell and they also know more than can be easily observed. A form of this problem commonly occurs with the popular 'user story' technique of communicating end-user requirements when it's later discovered that the story doesn't reflect an actual need but rather simply an articulation of what someone wants, resulting in, "I know that's what I said I wanted but that doesn't seem to be what I need." … they know more than they can tell or IKIWISI (I'll Know It When I See It).

It is also important that the entire team gain this deep understanding. Team members (individually and in sub-teams), in all functional roles, make decisions almost continually based on their understanding of the requirements and within their understanding of the context of those requirements. Much of this understanding is also tacit.

The problem exists in the midst of a current controversy. Some agile development thought-leaders such as Cohn are blunt: "The idea of eliciting and capturing requirements is wrong." [4, p. 52]. While many hold to prevailing views, believing that we just need better techniques to improve effectiveness, others are taking Cohn's critique even further and fundamentally suggesting that the notion of requirements itself may be counterproductive (e.g. Ralph [5], Mohanani et al. [6], Guinan et al. [7]) or even illusory (Ralph [8].)

This controversy aside, it still behooves teams to strive for a deep, collective understanding of the context of their product, that other world for which their product is intended, a shared mental model of the supra-domain since many large and small, conscious and unconscious design and implementation decisions are made within the team's understanding of the domain context. The success of the team, of their product, and often of the software company itself rests upon how well they do this. Teams do this with varying degrees of success. Some achieve reasonable success seemingly instinctively, while many struggle ineffectively. Software development leaders are often able to observe this phenomenon but have no theories that help explain why.

While some labels are being used to describe what they think development teams have to do to achieve a profound understanding (grokking) of that external world (e.g. "empathy-driven development"), there does not appear to be any clear definition of what that is, but rather simply a label of what some think may be happening but there lacks a true understanding of how this may be occurring.

4 Research Question and Importance

The purpose of this qualitative research study is to develop a substantive theory that answers the following general research question:

> *"what factors influence the degree to which cross-functional software product teams, empowered to own their product, collectively achieve a deep understanding of the environment for which their product is intended?"*

This theory will help industry practitioners explain why certain prevailing techniques and empirical approaches for understanding software solution needs are often inadequate, why some succeed while others do not. It is also likely to offer interpretive insights into how creativity and innovation occurs within software product teams and offer guidance for more effective software development approaches.

In addition to assisting practitioners in industry, this interpretive theory aspires to illuminate areas of potential further research. For example, how are technically-oriented people (primarily millennials) working in teams (typically cross-functional) and following a rational process to create software solutions able to develop, nurture, and incorporate 'squishier' skills into a process that strives to be as rational and deterministic as possible? What does this suggest regarding teaching of problem-solving skills for software engineers in the future? Or, how does that which cannot be easily observed nor expressed be equally understood and preserved within a team? Or, how does empathic appreciation of the context of a software solution translate across individuals, organisations, business domains, cultures?

5 Focus of the Research and Challenges

As the saying goes, "a fish doesn't know it's in water", thus the intended users of software solutions often cannot envisage an ideal (or, sometimes, even a conceptually different) solution nor clearly communicate the context in which they operate because they are trapped in that context. Thus, for software development teams to understand and define that which they cannot easily see, to understand 'why' more than 'what', to understand the functionality needs and the supra-functionality requirements and context, it is necessary to somehow become one of the people targetted to use a software solution, and to truly learn from that immersion. This is difficult because it involves somehow blurring the perceptual boundaries between the team members and the target environment. To be an outsider and obtain an insider's perspective and knowledge is not only difficult, it is messy logistically. This does not easily fit into established software engineering practices nor is it well-supported by software engineers' training. Considering that software solutions are a result of a collaborative cross-functional team effort, the messiness is even more acute.

Thus, the focus of this research is practicing software product teams in action, specifically teams empowered to own their product. It examines the empirical adaptations these teams make to established software engineering practices and to methods of user interaction design to support empathic-based development towards an ever-growing and increasingly accurate understanding of the context in which their users

operate, the supra-domain - the business needs, technology, culture, and politics. The research also examines how software development individuals and teams, who are trained and encouraged to apply their best judgement, suspend those judgements and opinions in order to connect with and exercise empathy for the domain for which their solution is intended. Finally, it examines important organizational factors that either allow or inhibit a team's ability to collectively grok the domain.

One challenge identified early in the study was how to detect or measure a team's grokking ability. A team's ability to execute is not a suitable indicator since it is culturally and contextually dependent and that those factors may not be the same as the ones that influence the ability of a team to grok. Individual and collective engagement, however, is a necessary condition and the existence, or lack of, has been very easily detected in the field experience to-date, so I am using this as a first-level differentiator.

A second challenge has been the impact an organizational structure has on a team's collective understanding. To-date, my approach is to identify these as critical factors (intended or otherwise) and the insights as guidance for technical leaders.

6 Literature Review

A modest review of the literature was done determine if this topic was previously explored in a different context, perhaps with a different vocabulary. Nothing was found that referred to this research topic. Much was found that looked at intra-team dynamics within software development teams, but this does not get to the focus of my inquiry.

Literature was also reviewed in 3 main areas (requirements engineering (specifically elicitation), design science, and collective sensemaking) as well as certain tangential areas (management decision-making, information system success models, and cognitive and organisational models).

My research may be viewed as falling fully within the topic of software requirements engineering, specifically requirements elicitation (attempting to obtain and understand the true needs). I looked at all the accepted papers for the IEEE International Requirements Engineering Conference over the past 10 years, plus many related papers published in other publications. There are increasingly more views being expressed - consistent with the problem and views I reference in Sect. 3 - that acknowledge the shortcomings of prevailing approaches to requirements elicitation which have tended to focus on techniques and methods rather than deep practitioner understanding. This is evidence that some software product development efforts still operate in the 'process-driven' paradigm and are experiencing what Kuhn [2] described as the incommensurability across paradigms. While acknowledging that the 'techniques and methods' approach is entirely appropriate in certain domains, my focus is on problem domains that don't lend themselves well to clear specifications and, thus, I find myself firmly planted philosophically in the new paradigm and sharing the incommensurate view of 'requirements'. Setting labels and practices aside, I acknowledge that the intent of requirements elicitation has always been to understand the true software needs and, therefore, this research will contribute to the requirements engineering discipline, relabelled or not.

To establish a broader positioning of this research in the literature, I reviewed a significant amount of relevant literature in the design science field since design would appear to have some relationship to my research which is looking at teams trying to grok someone else's world (empathic ability). I also reviewed relevant literature of sensemaking as my research will examine the collective team effort to under-stand (collective sensemaking).

In the design science space, I found a considerable scholarship regarding empathy-driven design, e.g. (Koppen and Meinel [9], van Rijn et al. [10], Postma et al. [11], Wood-cock et al. [12], Dong et al. [13], Kourprie and Visser [14], Kolko [15]). However, this research falls short of addressing my inquiry questions in three critical respects: (1) the focus is solely on the design activity as part of an essentially sequential product development process rather than design as part of an on-going continuous product development effort, (2) it tends not to consider the whole development team, rather tends to focus on the design individual or design team, and, (3) when it does consider the design team, it is not viewed as a unit to consider regarding its empathic ability. There are design science models described by Wieringa [16] that acknowledge the challenge that empathy-driven requirements understanding attempts to address (using very different vocabulary) but he stops short of suggesting how those challenges are, or could be, addressed. I believe the results of my research could enrich those models and generally contribute to the design science field.

In the organisational sensemaking field, the focus of many researchers is mainly on the social process of individual identity in successive spheres of membership through interactions with others. The collective (team) is usually considered only insofar as its relationship to the organisation, not to its understanding of a specific domain outside of the organisation. Some researchers, notably Russell [17] from a Human-Computer Interaction (HCI) perspective, look at sensemaking for a broader purpose - to collect and organise information in order to gain insight, to analyse, to transfer. However, although his view establishes sensemaking in a collective location (an information world), he describes a style of engagement of sensemaking that is essentially personal, not collective. The Cynefin framework (Kurtz & Snowden [18]) is a sensemaking framework that is particularly useful for collective sensemaking in that it is designed to allow shared understandings to emerge which could be insightful with respect to how teams ingest, socialise, and collectively store insights. As with other collective sensemaking models, it has resonance in early problem-solving stages and for formal and finite periods of time. Other researchers (Klein et al. [19], Naumer et al. [20], Kolko [21]) elaborate further by bringing data-framing into the picture and defining design synthesis as a process of sense-making, trying to make sense of chaos. The data-framing activity of sensemaking lends itself to being part of a long-term collective effort to understand and therefore may have some relevance to this research.

In the more tangential areas, there is scholarship in the management decision-making field that have parallels to this inquiry, e.g. Isabella [22] work on how management team interpretations evolve should be compared and contrasted with how software development teams evolve their understanding of needs, and Weick and Roberts [23] work on the collective mind could help frame how software teams socialise learning and maintain a relevant team memory. My research is about obtaining understanding upon which decision-making would be based and not about

decision-making per se, so I have not surveyed this space thoroughly although I intend to do so once core categories have emerged in my study.

Finally, once the core categories do emerge, I intend to compare my findings to information systems success models such as: Technology Acceptance Model (Davis [24]) based on the Theory of Reasoned Action, DeLone and McLean Information Systems Success Model (DeLone and McLean [25]), and the IS-Impact Model (Gable et al. [26]). Also models from other disciplines are likely to have comparative value such as the Expectation Confirmation Theory (Oliver [27, 28], Bhattacherjee, [29] and theories from Organisational Learning – mental models, shared vision, team learning, and systems thinking.

7 Method of the Research

As the primary interest is on substantive theory generation, rather than extending or verifying existing theories, I am taking an interpretive epistemological stance, employing a Grounded Theory approach, as developed by Glaser and Strauss [30], and using the Constructivist Grounded Theory methodology described by Charmaz [31]. Grounded Theory is highly applicable in research such as this because it is explicitly emergent. I am interested in generating theory relating to a specific research situation and this research calls for a qualitative approach. This is an area that is a relatively new, where there has been limited research, and where field data will come from observations and interviews, conditions for which Grounded Theory is particularly well suited.

More specifically, Grounded Theory is applicable for this research because the current Agile paradigm for software development focusses on people and interactions and Grounded Theory, as a qualitative research method, allows for the study of complex, multi-faceted social interactions and behaviour. Grounded Theory has been used success-fully as a research method to study Agile software development teams: Adolph et al. [32], Dagenais et al. [33], Coleman and O'Connor [34], Martin [35], Hoda [36].

The research uses theoretical sampling (Charmaz [32]) where the analysis of the data collected prior informs the selection of and inquiry with the next participants. Individual participants and corporate sites selected are ones involved with software product development (teams developing software for market) and that claim to have cross-functional product development teams. The primary data collection method is semi-structured interviews with open-ended questions that will allow real issues to emerge. I con-duct observations of team meetings and team interactions to enrich interview data.

I carefully recruit participants through my professional networks, from product study groups, and via direct outreach to select software product organisations. Where per-mitted, I hold interviews in the participant's workplace to allow for record review to enrich the interview data. Also, where I have approval from the organisations involved, I locate myself as unobtrusively as possible in the workplace to allow for direct observation as an additional data source and for those observations to direct further data collection and analysis. The interviews conducted are primarily with individuals and recorded whenever permitted. Group interviews may be held if data

analysis suggests, although, to date, this need has not surfaced. My many years of leader-ship with the types of people that are participants affords me considerable comfort, understanding, and rapid rapport with them.

Iterative data collection and analysis (formulation, testing, and redevelopment of propositions) allows the sample of participants and questions to purposefully evolve as patterns emerge in the data until I reach a theory. I use the NVivo software tool to analyse the unstructured qualitative data collected. The current expectation is to interview 25–30 team members representing 5–8 different teams, more if the analysis suggests. Data collection will stop once the analysis indicates the achievement of theoretical saturation, the point at which gathering more data reveals no new properties nor yields any further theoretical insights about the emerging grounded theory (Charmaz [31]). This ensures a certain degree of consistency in the analysis.

I recognise that my professional experience allows for a certain considered positionality and that this shapes my objectivity and subjectivity of many aspects of perspective in this study. While acknowledging the challenges, I consider this experience, and the bias it creates, to be an asset to this research. As Eisner [37] suggests, the expert ability to "see what counts" – the sensitivity to tacit elements of the data, meanings and connotations – will guide the research, supported fully by the collected data, towards questions that matter.

Quality in research of this nature is generally assessed in terms of validity and generalizability, which, together, determine some measure of usefulness. During the research, I employ various strategies (Maxwell [38]) to mitigate threats to validity (credibility, dependability, reliability). Intensive, on-going involvement (extended participation, the ability to 'live' in the participants' workplace) provides richer types of data, more direct and less dependent on inference, opportunity for repeated observations and interviews, all which will help rule out spurious associations and premature theories. The collection and use of rich data (transcribed inter-views, thick descriptive note-taking of observations) help provide a more complete and revealing picture of what is going on. Participant checks (obtaining participant and peer feedback on the data collected and conclusions drawn) help rule out possibilities of misinterpretation. Triangulation (collection from a range of participants and settings) reduces the risk of chance associations and systematic biases. Finally, I will be transparent with any discrepant evidence or negative cases. In short, applying disciplined rigor to the grounded theory methodology. I intend to assess transferability of the results within the context of software product development primarily via peer reviews of the resulting theory with software product development leaders and, further, to draw comparisons with non-product software development teams to further refine the specificity of transferability claims.

8 Status of the Research

Fieldwork began in 2017 and, to-date, I have been working with 4 software companies, all of which produce commercial software products, are leaders in their product markets, and range in size and maturity from early-stage to well-established (>12 years). With 8 participating teams across these companies, I have conducted 15 individual,

semi-structured interviews and 17 team planning observation sessions. More interviews and observation sessions are scheduled and more organizations and teams are being actively recruited.

9 Emerging Observations

The first emerging observation is that whether or not there is a functional organisation model surrounding the cross-functional team, the team dynamics, individual participation and sense of primary allegiance are significantly impacted. Where there is, e.g., a software engineering department, a design department, and a product management department, all contributing resources into cross-functional product teams, the inter-team dynamics are often strikingly different than when there is no functional organisation surrounding the teams. In the former case, team members are more likely to temper their contributions, identifying more with their functional affiliation than with the product mandate. The analogy I use here is that they're wearing a functional tee-shirt (e.g. I'm wearing the software engineering department t-shirt with a small insignia that says I happen to be assigned to this particular product at the moment). In addition to observing this in team interactions, this also appears in the language, "*I just do my job and they do theirs*", "*I trust them*", "*I think someone else is looking after that*", "*I just do what Product Management (or Product Design) says*", "*I'm on this team for now*". A software engineer in this environment is much more likely to care about the 'how' and defer to others on 'what' and 'why'. In contrast, organisations that do not have a functional structure surrounding the cross-functional product teams tend to see the teams have a more complete sense of ownership for their product and richer inter-team interactions. The tee-shirt analogy is that they're all wearing the same product tee-shirt with perhaps an insignia that identifies their functional competency. On these teams, sense of team is much stronger, thus the language does not refer to 'them'. All team members are more likely to care about 'what', 'why', and 'how' because they feel a stronger sense of ownership for the product overall, not just their particular contribution to it. I plan to probe this phenomenon further and look at definitions of success and how they may be defined similarly or not across these two models.

The second emerging observation relates to expectation of mobility. I've observed two pressures that inhibit an individual's inclination to be 'all-in'. One pressure is where there is a high degree of staff churn that impacts product development team resourcing. After a certain length of time, people in these environments come to expect they will be reassigned soon and thus have a certain tentativeness to their commitment to the product and the product team and tend to apply their focus to functional excellence only. The other pressure is similar, however, intended, and this is where an HR policy exists that encourages a high degree of mobility with respect to team assignment, e.g. 20% of technical staff should change teams every year. This seems to stem from a belief that this is healthy for the individual and/or adds to corporate robustness. A telling quote from an engineering manager, "I don't know how a true 'team' can emerge this way."

10 Discussion

Product development is a social process, thus the organizational dimension is the 'elephant in the room', a critical factor for success or failure of software product teams. The two observable phenomena surfacing strongly in the analysis thus far both fall into a category of what an organisation may do, consciously or otherwise, to support or inhibit a cross-functional team to be all it can be.

In the context of requirements engineering, I use the definition of empathy as *the ability to imaginatively step into another domain, understand the perspectives of those in that domain, and use that understanding to guide decisions* [39]. Stepping into that other domain involves a certain temporarily ***'blurring of the boundaries'*** in order to truly understand perspectives in that domain.

Although these observations point to internal conditions that impact a team's ability to perform, it appears that both these observations point to a certain *blurring of the boundaries* that may be a pre-condition for a cross-functional team to collectively grok (have deep, empathic understanding for) the world for which their product is intended.

This is consistent with a basic notion of empathy, namely that, in order to truly understand another world, one has to blur the boundaries somewhat for a period of time in order to better understand. Further work is needed to explore this, particularly as it applies to the collective cross-functional product development team.

References

1. Grok: Oxford English Dictionary. Oxford University Press, Oxford (1989)
2. Kuhn, T.S.: The Structure of Scientific Revolutions, 4th edn. University of Chicago Press, London (2012)
3. Polanyi, M.: The tacit dimension. In: Knowledge in Organisations (1997)
4. Cohn, M.: User Stories Applied: For Agile Software Development. Addison-Wesley Professional, Boston (2004)
5. Ralph, P.: The illusion of requirements in software development. Requirements Eng. **18**(3), 293–296 (2013)
6. Mohanani, R., Ralph, P., Shreeve, B.: Requirements fixation. In: Proceedings of the 36th International Conference on Software Engineering, pp. 895–906 (2014)
7. Guinan, P.J., Cooprider, J.G., Faraj, S.: Enabling software development team performance during requirements definition: a behavioral versus technical approach. Inf. Syst. Res. **9**(2), 101–125 (1998)
8. Ralph, P., Mohanani, R.: Is requirements engineering inherently counterproductive?. In: Proceedings - 5th International Workshop on the Twin Peaks of Requirements and Architecture, TwinPeaks 2015 (2015)
9. Koppen, E., Meinel, C.: Knowing people: the empathetic designer. Des. Philos. Pap. **10**(1), 35–51 (2012)
10. Van Rijn, H., Sleeswijk Visser, F., Stappers, P.J., Özakar, A.D.: Achieving empathy with users: the effects of different sources of information. CoDesign **7**(2), 65–77 (2011)
11. Postma, C., Zwartkruis-Pelgrim, E., Daemen, E., Du, J.: Challenges of doing empathic design: experiences from industry. Int. J. Des. **6**(1) (2012)

12. Woodcock, A., McDonagh, D., Osmond, J., Scott, W.: Empathy, design and human factors. In: Advances in Usability and User Experience, pp. 569–579 (2018)
13. Dong, Y., Dong, H., Yuan, S.: Empathy in design: a historical and cross-disciplinary perspective. In: Baldwin, C. (ed.) AHFE 2017. AISC, vol. 586, pp. 295–304. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-60642-2_28
14. Kouprie, M., Sleeswijk-Visser, F.: A framework for empathy in design: stepping into and out of the user's life. J. Eng. Des. **20**(5), 437–448 (2009)
15. Kolko, J.: Well-Designed: How to Create Empathy to Create Products People Love. Harvard Business Review Press, Boston (2014)
16. Wieringa, R.: Design Science Methodology for Information Systems and Software Engineering. Springer, Berlin (2014). https://doi.org/10.1007/978-3-662-43839-8
17. Russell, D., Pirolli, P.: An Overview of Sensemaking: A View from the Workshop CHI 2009. Sensemaking Work. CHI, pp. 1–2 (2009)
18. Kurtz, C.F., Snowden, D.: The new dynamics of strategy: sense-making in a complex-complicated world. IBM Syst. J. **42**(3), 462–483 (2003)
19. Klein, G., Moon, B., Hoffman, R., Associates, K.: Making Sense of Sensemaking 2: a macrocognitive model. IEEE Intell. Syst. **21**(5), 88–92 (2006)
20. Naumer, C., Fisher, K., Dervin, B.: Sense-Making: a methodological perspective. In: CHI 2008 Work. Sense-Making Florence (2008)
21. Kolko, J.: Sensemaking and framing: a theoretical reflection on perspective in design synthesis. In: 2010 Design Research Society Conference, pp. 1–9 (2010)
22. Isabella, L.A.: Evolving interpretations as a change unfolds: how managers construe key organisational events. Acad. Manag. J. **33**(1), 7–41 (1990)
23. Weick, K.E., Roberts, K.H.: Collective mind in organizations: heedful interrelating on flight decks. Adm. Sci. Q. 357–381 (1993)
24. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Q. **13**(3), 319–340 (1989)
25. DeLone, W.H., McLean, E.R.: The DeLone and McLean model of information systems success: a ten-year update. J. Manag. Inf. Syst. **19**(4), 9–30 (2003)
26. Gable, G., Sedera, D., Taizan, C.: Re-conceptualizing information system success: the IS-Impact measurement model. J. Assoc. Inf. Syst. **9**(7), 1–32 (2008)
27. Oliver, R.L.: Effect of expectation and disconfirmation on post exposure product evaluations - an alternative interpretation. J. Appl. Psychol. **62**(4), 480 (1977)
28. Oliver, R.L.: A cognitive model of the antecedents and consequences of satisfaction decisions. J. Mark. Res. **17**, 460–469 (1980)
29. Bhattacherjee, A.: Understanding information systems continuance: an expectation - confirmation model. MIS Q. **25**(3), 351–370 (2001)
30. Glaser, B.G., Strauss, A.L.: The Discovery of Grounded Theory: Strategies for Qualitative Research. Aldine Transaction, Piscataway (1967)
31. Charmaz, K.: Constructing Grounded Theory: A Practical Guide Through Qualitative Analysis. Sage, London (2006)
32. Adolph, S., Hall, W., Kruchten, P.: Using grounded theory to study the experience of software development. Empirical Softw. Eng. **16**(4), 487–513 (2011)
33. Dagenais, B., Ossher, H., Bellamy, R.K.E., Robillard, M.P., De Vries, J.P.: Moving into a new software project landscape. In: ICSE 2010 Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering, pp. 275–284 (2010)
34. Coleman, G., O'Connor, R.: Using grounded theory to understand software process improvement: a study of Irish software product companies. Inf. Softw. Technol. **49**(6), 654–667 (2007)

35. Martin, A.M.: The role of customers in extreme programming projects. Ph.D. thesis. Victoria University of Wellington, New Zealand (2009)
36. Hoda, R.: Self-organizing agile teams : a grounded theory. Ph.D thesis. Victoria University of Wellington, New Zealand (2011)
37. Eisner, E.W.: The Enlightened Eye: Qualitative Inquiry and The Enhancement of Educational Practice. Prentice-Hall, Upper Saddle River (1998)
38. Maxwell, J.A.: Qualitative Research Design: An Interactive Approach. SAGE Publications, Thousand Oaks (2012)
39. Krznaric, R.: Empathy: Why It Matters, And How to Get It. Penguin Random House, New York (2014)