Towards using Data to Inform Decisions in Agile Software Development: Views of Available Data

Christoph Matthies[a] and Guenter Hesse[b]
Hasso Plattner Institute, University of Potsdam, Germany

Keywords: Software Engineering, Agile Software Development, Data-driven Decision Making, Decision Support Systems.

Abstract: Software development comprises complex tasks which are performed by humans. It involves problem solving, domain understanding and communication skills as well as knowledge of a broad variety of technologies, architectures, and solution approaches. As such, software development projects include many situations where crucial decisions must be made. Making the appropriate organizational or technical choices for a given software team building a product can make the difference between project success or failure. Software development methods have introduced frameworks and sets of best practices for certain contexts, providing practitioners with established guidelines for these important choices. Current Agile methods employed in modern software development have highlighted the importance of the human factors in software development. These methods rely on short feedback loops and the self-organization of teams to enable collaborative decision making. While Agile methods stress the importance of empirical process control, i.e. relying on data to make decisions, they do not prescribe in detail how this goal should be achieved. In this paper, we describe the types and abstraction levels of data and decisions within modern software development teams and identify the benefits that usage of this data enables. We argue that the principles of data-driven decision making are highly applicable, yet underused, in modern Agile software development.

1 INTRODUCTION

The practice of software development includes series of decisions that must be made to ensure the success of a project. These decisions concern not only the scope, budget and feature set of the product being developed, but also how development teams are organized, what technologies and architectures are employed, how the customer is engaged and how requirements are elicited and prioritized. Making the right decisions for a given context, i.e. the decisions that have the highest chance of leading to project success, is of critical importance (Drury et al., 2012). Failing to do so is likely to result in overruns of budget and schedule, lost opportunities for the organization in need of the developed software, and ultimately project cancellation (Taherdoost and Keshavarzsaleh, 2018; Molokken-Ostvold and Jorgensen, 2005). As software is becoming pervasive and is being employed in all facets of life and in a large variety of contexts, few universal rules for decisions in software development, applicable to most situations and conditions, can be defined (Kuhrmann et al., 2018).

Modern software projects must be flexible, adapting to changing requirements and circumstances by analyzing available data, and making appropriate decisions. In Agile methods, such as Scrum and Kanban (Reddy, 2015), this idea is realized through short, iterative feedback cycles (Williams and Cockburn, 2003), relying on the capabilities of team members in self-organizing teams (Hoda et al., 2010; Matthies et al., 2016b) and making use of available data to enable "evidence-based decision making" (Fitzgerald et al., 2014). The Scrum Guide states: "Scrum is founded on empirical process control theory [...] knowledge comes from experience and making decisions based on what is known. [...]" (Schwaber and Sutherland, 2017). However, few details are given as to the concrete implementations of these concepts or what data and knowledge is available and relevant to teams employing Agile software development approaches. In this paper, we provide an overview of the described approaches that make use of data in informing decisions in Agile software development

[a] https://orcid.org/0000-0002-6612-5055
[b] https://orcid.org/0000-0002-7634-3021

Matthies, C. and Hesse, G.
Towards using Data to Inform Decisions in Agile Software Development: Views of Available Data.
DOI: 10.5220/0007967905520559
In *Proceedings of the 14th International Conference on Software Technologies (ICSOFT 2019)*, pages 552-559
ISBN: 978-989-758-379-7

and highlight challenges and possible conflicting aspects. Data-informed decision-making approaches are valuable for organizing and administering software projects. Through a better understanding of their role and impact, they can be applied more thoroughly in the future.

2 DATA-DRIVEN DECISION MAKING

Different approaches have previously been proposed for using data to inform and support individuals and project teams in making the most likely correct decisions. Related work reaching back multiple decades includes frameworks and theories that investigate this area. The terms *Data-Driven Decision Management*, *Data-Directed Decision Making* and *Data-Driven Decision Making* (DDDM) have been used to refer to the practice of basing decisions on the analysis of collected data.

2.1 Application of DDDM

The application of DDDM concepts is contrasted with the process of coming to a decision by relying on "gut feeling", personal experience and intuition (Brynjolfsson et al., 2011; Provost and Fawcett, 2013), which, in the absence of capable analysis technologies, was the de facto standard throughout the history of commercial enterprises. One of the core ideas of DDDM is that decisions, as well as their estimated efficacy, can be deduced from key data sets. In a basic example, an employee in marketing, tasked with creating and selecting advertisements to be shown to website visitors, could base their designs and selections solely on their long experience of working in the advertising field and their intuitive understanding of the effects of different marketing campaigns. However, when applying DDDM, the employee could additionally analyze how website users have interacted with ads in the past in order to draw conclusions for the future. In line with our arguments, Provost and Fawcett point out that these two approaches are not mutually exclusive and can be combined (Provost and Fawcett, 2013).

Technology adoption in business scenarios has considerably increased in recent years, due to the continuing digital transformation (Hesse et al., 2018) and the reliance on Internet technologies (Afuah, 2002). DDDM has, therefore, become an influential part of a large variety of industries (Provost and Fawcett, 2013), including highly important sectors such as medicine (Hollis et al., 2015), transportation, and manufacturing (Hesse et al., 2019). Studies of the application of DDDM in companies have shown the benefits of the approach. Brynjolfsson et al. collected data on the business practices and information technology investments of 179 businesses and correlated this data with company performance measures such as productivity, profitability, and market value (Brynjolfsson et al., 2011). The authors find that companies that adopted DDDM show an increase of 5-6% regarding output and productivity than what would be expected given their investments and information technology usage.

While the introduction of digital networks and communication infrastructure in enterprises has improved efficiency, it has also intensified existing tensions and led to new challenges. Technology has enabled greater transparency and visibility throughout businesses. Stakeholders, e.g. customers or internal users, have instantaneous access to actionable information on the state of projects and real-time situational awareness is dramatically increased (Schrage, 2016). Therefore, the managerial and operational ability to act on the collected data and associated analyses must also scale.

2.2 Being Driven or Being Informed by Data

Recent work in the field of decision support for software projects has focused on *data-driven* approaches (Svensson et al., 2019; Olsson and Bosch, 2014; Provost and Fawcett, 2013). This term highlights the fact that data is in the idiomatic "driver's seat", being responsible for the direction a project is headed in. This approach is in direct opposition to relying solely on intuition and the experience of team members, which are classified as biased and unreliable. However, software engineering is still a quintessentially human task (Fernando Capretz, 2014), requiring creativity, problem-solving abilities, and empathy for the users and stakeholders who will eventually use, or be impacted by, the developed software. Disregarding or assigning little value to this aspect, i.e. concentrating on only the data that is produced by teams and software users, ignores the value that attention to human factors can provide in a project (Biddle et al., 2018; Sherdil and Madhavji, 1996). We, therefore, propose being *informed* instead of *driven* by data concerning decision making in the context of software development, taking into account analyses of the available data and highlighting the need for human interpretation. While important decisions that can affect project success should not only be based on intuition, they should also not solely rely

on data, which may likewise only portray a one-sided, biased view. Instead, as much project data as possible should be collected and analyzed, which can then be interpreted *or ignored as irrelevant* by practitioners and team members that have intricate knowledge of the data and the context in which it was gathered. This applies not only to software developers, who are experts concerning the developed application and the technical details, but also product managers, sales departments, and other roles that have knowledge related to the user and the context. Combining data and human interpretation can enable better *informed* decisions in software development.

3 AGILE SOFTWARE METHODS

Agile development methods are closely tied to the availability of data. It enables teams to implement the short feedback cycles that define "agility" (Schwaber and Sutherland, 2017).

3.1 Evolution of Agile Software Development

Iterative and incremental development (IID) approaches for building software are not new. Larman and Basili trace these concepts back to the 1930s, identifying the 1970s and 80s as the most active but least known part of their history (Larman and Basili, 2003). They describe the evolution of Agile methods, starting with the Rational Unified Process (Kruchten, 2004) and the Dynamic Systems Development Method (Stapleton, 1997) from the 1990s. Several key ideas followed, such as Extreme Programming (Beck and Gamma, 2000), the Crystal family of methods (Cockburn, 2004) and finally Feature-Driven Development (Palmer and Felsing, 2001), Lean (Mary and Tom, 2003) and Scrum (Schwaber and Sutherland, 2017)

Agile software development methods represent a set of best practices for software development in teams, created by experienced practitioners (Fowler and Highsmith, 2001; Dybå and Dingsøyr, 2008). These methods highlight communication, adaptation to change, innovation, and teamwork, emphasizing productivity rather than process rigor (Ågerfalk and Fitzgerald, 2006). They are often portrayed as the antithesis to traditional, more plan-based approaches, which feature rationalized and planned decision making, with work progressing in planned, successive stages, with little feedback built into the system (Lei et al., 2017). Williams and Cockburn claim that software development cannot be considered a "defined process", as change is inevitable, and requirements will be adjusted during the time the product is being developed (Williams and Cockburn, 2003). Software development is instead recognized as a flexible and "empirical (or nonlinear)" process (Williams and Cockburn, 2003). In the context of engineering software, development processes, therefore, require short "inspect-and-adapt" cycles and continuous feedback loops to enable process improvements based on empirical evidence. These ideas are part of the *Agile Manifesto* (Fowler and Highsmith, 2001), written by the practitioners who proposed many of the modern Agile development methods. It presents four core values, among them "responding to change over following a plan". The importance of adapting to changing circumstances through feedback loops is also highlighted in the accompanying Agile principles, which state that "at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly" (Fowler and Highsmith, 2001). The Scrum Guide (Schwaber and Sutherland, 2017), the seminal text for the currently most popular Agile development method (VersionOne Inc., 2018), likewise incorporates these concepts, stating that "transparency, inspection, and adaptation" are prerequisites for making decisions in teams based on data.

3.2 Agile Decision-making

The shift from a more plan-driven approach to Agile development methods in an organization also requires a change in how decisions are made (Moe et al., 2012). It implies converting classical "command and control" attitudes to approaches relying on shared decision-making involving stakeholders and development teams (Moe et al., 2009). A central point of Agile methods is the focus on working in small teams that have access to customers as well as the product's users and those affected. The responsibility of establishing the priority of features and work items falls jointly on the development team and the stakeholders, who have different backgrounds and goals (Nerur et al., 2005). This shared, flexible, decision-making represents a "barely sufficient" command and control approach (McAvoy and Butler, 2009). The decision-making process in Agile methods has been described as *naturalistic*, compared to the *rational* decision-making of plan-driven approaches (Moe et al., 2012). In these frameworks, a rational decision complies with a set of rules that govern behavior, which are applied in a logical fashion to generate decisions with acceptable consequences for the decision makers. Rational decisions are based on

the assumption that the set of solution possibilities and the probability of outcomes is known (Zannier et al., 2007). In comparison, a naturalistic decision-making process is characterized by situational behavior, the diminished importance of conscious analytical evaluation and application of context-dependent rules (Klein, 2008). The principles of Agile software development more closely align with the definition of naturalistic decision-making, as the importance of awareness and project context are highlighted (Zannier et al., 2007). The types of decisions and plans that have to be made in a software business can be split into three main domains: strategic, tactical, and operational. Figure 1 gives an overview of these levels and their assignments to different team roles in plan-driven and Agile software development approaches.

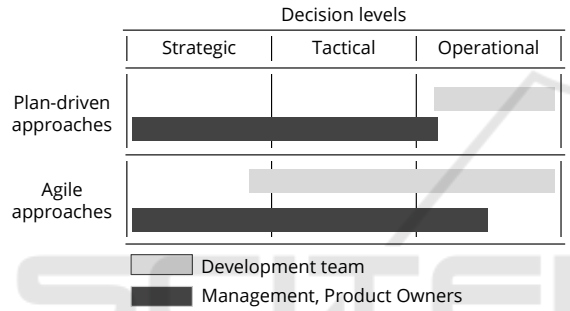

Figure 1: Differences between plan-driven and Agile approaches regarding responsibilities of different levels of decisions, adapted from Moe et al. (Moe et al., 2012).

Strategic planning is employed to define an organization's direction and the goals that should be pursued in the long run. Strategic decisions generally concern the whole business instead of individual business units or teams. In an Agile development context, many strategic decisions are assigned to the Product Owner, which require detailed knowledge of the software product and are primarily concerned with software release plans and product road maps. However, as the development team possesses deep technical knowledge of the product they should also be involved in defining strategy.

Tactical plans are involved with the management of projects, translating strategic directions into actionable plans for specific organization areas, i.e. how to best structure teams and resources to achieve the given goals.

Operational planning describes the lowest, most concrete level of decisions within an organization. It defines milestones and success conditions and can detail how, and what portion of, projects and proposals will be tackled in a given operational period. In an Agile software organization, operational decisions are related to implementing features and ensuring that tasks are carried out efficiently. An ideal, efficient, Agile development team is involved in shared decision-making not only on the operational but also on the tactical levels (Moe et al., 2012).

4 DATA IN (AGILE) SOFTWARE DEVELOPMENT

An ever-increasing number of companies, in the domain of software development or otherwise, are relying more on the analyses of collected data to make informed decisions.

4.1 Business Data

Businesses have gathered extremely detailed statistics, regarding not only their consumers and users, but also suppliers, alliance partners, and competitors (Brynjolfsson et al., 2011). This development is driven by the widespread adoption of enterprise information systems, which collect and store large amounts of business data and enable analyses of these collections. Examples for these systems are Enterprise Resource Planning (ERP) applications, which present a unified view of the business and contain databases of all business transactions (Umble et al., 2003). Recently, analyses and utilization of this collected business data have become more relevant with the introduction of Business Intelligence software solutions, which apply an extensive set of data analytic tools to the operational data of an enterprise (Chen et al., 2012). Furthermore, the continuing digitization of society has allowed new possibilities for data collection outside operational business systems (Brynjolfsson et al., 2011). Cars, smartphones, home automation systems, and other smart devices that are interacted with on a daily basis are routinely instrumented to capture information regarding their current status and ongoing activities (Svensson et al., 2019). This data, collected from "reality mining", can be used to recognize social patterns in daily user activity, infer relationships, and, especially relevant for software process improvement, model organizational procedures (Eagle and (Sandy) Pentland, 2006). Additionally, clickstream data, user interface interactions and keyword searches from websites and software applications can reveal insights into customer behavior without the need for long-running and costly customer studies (Brynjolfsson et al., 2011). These types of detailed as well as diversified data sets are not only generated internally in software-intensive or software development companies but can also be acquired or

purchased from third parties, be they public or proprietary (Provost and Fawcett, 2013). Businesses focused on software development can make use of the available data in these various systems to make informed decisions in their development processes and to stay competitive. The recent resurgence of machine learning approaches (Svensson et al., 2019), based on the large amount of available data and promising more automated and powerful data analysis, accelerates these trends.

4.2 Business Data for Software Development

While large amounts of data are available within software businesses, which could be used to augment decision-making processes, selection and prioritization of product features to be shipped in the next iteration is commonly based on stakeholders' previous experiences, perceptions, and opinions (Olsson and Bosch, 2014; Svensson et al., 2019). Decisions based on these factors may be inconsistent and, more crucially, lack explanation and links to the evidence and the data that led to them. These factors make it easier to base decisions on politics or gains for individual teams than to focus on customer value. Analyses of metrics captured from users of deployed software products, however, enable short feedback cycles between customers and developers. Instead of guessing and assuming how users will interact with, e.g. a given graphical user interface, developers and stakeholders can observe the actual (mis)usage and base further developments and changes on these insights. The quality of analytical frameworks and tools being employed is directly related to the quality of decisions derived from them. However, the characteristics of the descriptions and visualizations that decision makers use, are equally important (Janssen et al., 2017; Matthies et al., 2016a). In recent research Svensson et al. have pointed out, that while there has been a growing interest in the tools used for data processing, little work has focused on the practitioners' perspective and the context of Agile development (Svensson et al., 2019). However, as is the case in the enterprise domain, there is an increased interest in applying machine learning methods and techniques in software engineering contexts to enable higher efficiency (Feldt et al., 2018). Supporting the important decisions that are required during software development activities in teams is one of the major application areas for these concepts in the realm of software development processes.

4.3 Software Project Data

In addition to data produced by customers, internal business units or third parties, the software development process itself represents a source of valuable data, which is intrinsically highly relevant to making decisions within software companies. Modern Agile software development relies on the creation, management, and delivery of digital development artifacts (Fernández et al., 2018). Not only is software, i.e. code, produced during regular development activities, but also a range of supporting documents and structures, such as work item descriptions, documentation, and version control information (Noll et al., 2012). Software engineers continuously produce data points on their current work and development process (Ying et al., 2005). As such, the employed development processes are "inscribed" into the produced software artifacts (de Souza et al., 2005). The version control system (VCS) which is ubiquitously employed for collaboration in teams, keeps track of the individual changes by developers as well as when the changes were made. It also captures information on who authored and committed the changes and the goal of the commit is recorded in the commit message (Santos and Hindle, 2016). More detailed information on work items might be contained in an issue tracker, such as Jira, with issues detailing the rationale, background, and context of a requested feature (Ortu et al., 2015). Frequently employed tools such as Continuous Integration servers or static analysis tools provide data points on the current status and health of the developed software project (Beller et al., 2017; Embury and Page, 2019) All of these tools are present in modern Agile teams as necessary prerequisites for efficient communication and collaboration. However, they also present "a gold-mine of actionable information" (Guo et al., 2016), especially in the domain of data-informed decision making in software producing organizations. There is little overhead in collecting and analyzing this sort of data, as it is already being produced by software teams. In the case of open.source software development, much of this data is even available publicly (Linstead et al., 2009; Zampetti et al., 2017). Especially of note for data-informed decision-making is the fact that software project data not only comprehensively documents progress but also provides evidence for failures and problems of the developed product (Ziftci and Reardon, 2017).

4.4 Decision-making and Agile Self-organization

The concepts of self-organization and self-management underpin the issue of decision-making in Agile teams (Moe et al., 2009). Members of a self-organizing Agile team, are ideally responsible not only for working on the tasks they choose to work on, but are also compelled to manage and track their own performance. Equally, the responsibility for decision-making should be distributed among team members rather than being centralized with few parties (Moe et al., 2012), see Figure 1. In Agile teams, the project manager's role a the main source of decision-making power is reduced, and developers and product owners—and even customers— may be involved in decisions. The concept of self-organization directly influences the effectiveness of a team as the authority for making decisions is directed to the lowest level of operations, which increases the speed of addressing problems and adapting to changes (Moe et al., 2012). While data is available that can inform decisions regarding development processes, research in this area has also shown that Agile team members rely on their experiences for evaluating design decisions (Zannier and Maurer, 2006). Current empirical research offers little clarity on how and why Agile software development teams make business and product-related decisions and whether the team autonomy emphasized in Agile methods leads to teams that can make both strategic and tactical decisions in practice (Drury et al., 2012). We hope for and would like to encourage more research and exploration of this promising research field in the future.

5 CONCLUSION

Agile methods, with their focus on self-organized teams and empirical process control, require customized decision-making processes and procedures in organizations (Moe et al., 2012). Agile software development teams, due to the requirement of having to deliver software increments in short iterations, are by definition involved in short-term decision-making (Svensson et al., 2019). Managers, in an Agile context, are expected to create an environment that allows team members to make decisions based on the best information available (Schuh, 2004). However, the question of what attributes define *best information* for a given team is still unanswered and requires additional research. Furthermore, it is accepted that even if good quality data is available, individuals can ignore analyses or even their own preferences due to rules, traditions or the influence of others (Dybå and Dingsøyr, 2008). In this paper, we argue that the principles of data-driven decision making, while still underused, are highly applicable and beneficial to Agile software development. The combination of data analysis and interpretation by Agile teams of humans can enable better informed decisions, both in the domain of business as well as software development processes.

REFERENCES

Afuah, A. (2002). *Internet Business Models and Strategies: Text and Cases*. McGraw-Hill, Inc., New York, NY, USA, 2nd edition.

Ågerfalk, P. J. and Fitzgerald, B. (2006). Flexible and distributed software processes: old petunias in new bowls? *Communications of the ACM*, 49:27–34.

Beck, K. and Gamma, E. (2000). *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Beller, M., Gousios, G., and Zaidman, A. (2017). Oops, My Tests Broke the Build: An Explorative Analysis of Travis CI with GitHub. In *2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR)*, pages 356–367. IEEE.

Biddle, R., Meier, A., Kropp, M., and Anslow, C. (2018). Myagile: Sociological and Cultural Effects of Agile on Teams and their Members. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE '18*, pages 73–76, New York, New York, USA. ACM Press.

Brynjolfsson, E., Hitt, L. M., and Kim, H. H. (2011). Strength in Numbers: How Does Data-Driven Decisionmaking Affect Firm Performance? *SSRN Electronic Journal*.

Chen, H., Chiang, R. H. L., and Storey, V. C. (2012). Business intelligence and analytics: From big data to big impact. *MIS quarterly*, 36(4).

Cockburn, A. (2004). *Crystal clear: a human-powered methodology for small teams*. Pearson Education.

de Souza, C., Froehlich, J., and Dourish, P. (2005). Seeking the Source: Software Source Code as a Social and Technical Artifact. In *Proceedings of the 2005 international ACM SIGGROUP conference on Supporting group work - GROUP '05*, page 197, New York, New York, USA. ACM, ACM Press.

Drury, M., Conboy, K., and Power, K. (2012). Obstacles to decision making in Agile software development teams. *Journal of Systems and Software*, 85(6):1239–1254.

Dybå, T. and Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology*, 50(9-10):833–859.

Eagle, N. and (Sandy) Pentland, A. (2006). Reality mining: sensing complex social systems. *Personal and Ubiquitous Computing*, 10(4):255–268.

Embury, S. M. and Page, C. (2019). Effect of Continuous Integration on Build Health in Undergraduate Team Projects. In Bruel, J.-M., Mazzara, M., and Meyer, B., editors, *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 169–183, Cham. Springer International Publishing.

Feldt, R., de Oliveira Neto, F. G., and Torkar, R. (2018). Ways of applying artificial intelligence in software engineering. In *Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, RAISE '18, pages 35–41, New York, NY, USA. ACM.

Fernández, D. M., Böhm, W., Vogelsang, A., Mund, J., Broy, M., Kuhrmann, M., and Weyer, T. (2018). Artefacts in Software Engineering: What are they after all? *International Journal on Software and Systems Modeling*.

Fernando Capretz, L. (2014). Bringing the Human Factor to Software Engineering. *IEEE Software*, 31(2):104–104.

Fitzgerald, B., Musiał, M., and Stol, K.-J. (2014). Evidence-based decision making in lean software project management. In *Companion Proceedings of the 36th International Conference on Software Engineering - ICSE Companion 2014*, pages 93–102, New York, New York, USA. ACM Press.

Fowler, M. and Highsmith, J. (2001). The agile manifesto. *Software Development*, 9(8):28–35.

Guo, J., Rahimi, M., Cleland-Huang, J., Rasin, A., Hayes, J. H., and Vierhauser, M. (2016). Cold-start software analytics. In *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*, pages 142–153, New York, New York, USA. ACM Press.

Hesse, G., Matthies, C., Sinzig, W., and Uflacker, M. (2019). Adding Value by Combining Business and Sensor Data: An Industry 4.0 Use Case. In Li, G., Yang, J., Gama, J., Natwichai, J., and Tong, Y., editors, *24th International Conference on Database Systems for Advanced Applications*, pages 528–532. Springer International Publishing.

Hesse, G., Reissaus, B., Matthies, C., Lorenz, M., Kraus, M., and Uflacker, M. (2018). Senska – Towards an Enterprise Streaming Benchmark. In Nambiar, R. and Poess, M., editors, *Performance Evaluation and Benchmarking for the Analytics Era (TPCTC 2017)*, pages 25–40. Springer International Publishing.

Hoda, R., Noble, J., and Marshall, S. (2010). Organizing self-organizing teams. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - ICSE '10*, volume 1, page 285, New York, New York, USA. ACM Press.

Hollis, C., Morriss, R., Martin, J., Amani, S., Cotton, R., Denis, M., and Lewis, S. (2015). Technological innovations in mental healthcare: harnessing the digital revolution. *The British Journal of Psychiatry*, 206(4):263–265.

Janssen, M., van der Voort, H., and Wahyudi, A. (2017). Factors influencing big data decision-making quality. *Journal of Business Research*, 70:338–345.

Klein, G. (2008). Naturalistic decision making. *Human factors*, 50(3):456–460.

Kruchten, P. (2004). *The rational unified process: an introduction*. Addison-Wesley Professional.

Kuhrmann, M., Diebold, P., Munch, J., Tell, P., Trektere, K., Mc Caffery, F., Vahid, G., Felderer, M., Linssen, O., Hanser, E., and Prause, C. (2018). Hybrid Software Development Approaches in Practice: A European Perspective. *IEEE Software*, PP:1–1.

Larman, C. and Basili, V. (2003). Iterative and incremental developments. a brief history. *Computer*, 36(6):47–56.

Lei, H., Ganjeizadeh, F., Jayachandran, P. K., and Ozcan, P. (2017). A statistical analysis of the effects of Scrum and Kanban on software development projects. *Robotics and Computer-Integrated Manufacturing*, 43:59–67.

Linstead, E., Bajracharya, S., Ngo, T., Rigor, P., Lopes, C., and Baldi, P. (2009). Sourcerer: mining and searching internet-scale software repositories. *Data Mining and Knowledge Discovery*, 18(2):300–336.

Mary, P. and Tom, P. (2003). Lean software development: an agile toolkit.

Matthies, C., Kowark, T., Richly, K., Uflacker, M., and Plattner, H. (2016a). ScrumLint: Identifying Violations of Agile Practices Using Development Artifacts. In *Proceedings of the 9th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE '16*, pages 40–43. ACM Press.

Matthies, C., Kowark, T., and Uflacker, M. (2016b). Teaching Agile the Agile Way — Employing Self-Organizing Teams in a University Software Engineering Course. In *American Society for Engineering Education (ASEE) International Forum*. ASEE.

McAvoy, J. and Butler, T. (2009). The role of project management in ineffective decision making within Agile software development projects. *European Journal of Information Systems*, 18(4):372–383.

Moe, N. B., Aurum, A., and Dybå, T. (2012). Challenges of shared decision-making: A multiple case study of agile software development. *Information and Software Technology*, 54(8):853–865.

Moe, N. B., Dingsøyr, T., and Dybå, T. (2009). Overcoming barriers to self-management in software teams. *IEEE software*, 26(6):20–26.

Molokken-Ostvold, K. and Jorgensen, M. (2005). A comparison of software project overruns - flexible versus sequential development models. *IEEE Transactions on Software Engineering*, 31(9):754–766.

Nerur, S., Mahapatra, R., and Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM*, 48(5):72–78.

Noll, J., Seichter, D., and Beecham, S. (2012). A qualitative method for mining open source software repositories. *IFIP Advances in Information and Communication Technology*, 378 AICT:256–261.

Olsson, H. H. and Bosch, J. (2014). From Opinions to Data-Driven Software R&D: A Multi-case Study on How to Close the 'Open Loop' Problem. In *2014 40th EUROMICRO Conference on Software Engineering and Advanced Applications*, pages 9–16. IEEE.

Ortu, M., Destefanis, G., Adams, B., Murgia, A., Marchesi, M., and Tonelli, R. (2015). The JIRA Repository Dataset. In *Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering - PROMISE '15*, pages 1–4, New York, New York, USA. ACM Press.

Palmer, S. R. and Felsing, M. (2001). *A practical guide to feature-driven development*. Pearson Education.

Provost, F. and Fawcett, T. (2013). Data Science and its Relationship to Big Data and Data-Driven Decision Making. *Big Data*, 1(1):51–59.

Reddy, A. (2015). *The Scrumban [R]evolution: getting the most out of Agile, Scrum, and lean Kanban*. Addison-Wesley Professional.

Santos, E. A. and Hindle, A. (2016). Judging a commit by its cover. In *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*, MSR '16, pages 504–507, New York, New York, USA. ACM Press.

Schrage, M. (2016). How the big data explosion has changed decision making. *Harvard Business Review Digital Articles*, pages 2–5.

Schuh, P. (2004). *Integrating Agile Development In The Real World*. Charles River Media, Inc., Rockland, MA, USA, 1st edition.

Schwaber, K. and Sutherland, J. (2017). The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game. Technical report, scrumguides.org.

Sherdil, K. and Madhavji, N. H. (1996). Human-Oriented Improvement in the Software Process. In *European Workshop on Software Process Technology*, pages 144–166. Springer.

Stapleton, J. (1997). *DSDM, dynamic systems development method: the method in practice*. Cambridge University Press.

Svensson, R. B., Feldt, R., and Torkar, R. (2019). The Unfulfilled Potential of Data-Driven Decision Making in Agile Software Development. In *Agile Processes in Software Engineering and Extreme Programming*, pages 69–85. Springer International Publishing.

Taherdoost, H. and Keshavarzsaleh, A. (2018). A Theoretical Review on IT Project Success/Failure Factors and Evaluating the Associated Risks. In *Mathematical and Computational Methods in Electrical Engineering*.

Umble, E. J., Haft, R. R., and Umble, M. M. (2003). Enterprise resource planning: Implementation procedures and critical success factors. *European journal of operational research*, 146(2):241–257.

VersionOne Inc. (2018). 12th Annual State of Agile Report. Technical report, Collab.net, versionone.com.

Williams, L. and Cockburn, A. (2003). Agile software development: it's about feedback and change. *Computer*, 36(6):39–43.

Ying, A. T. T., Wright, J. L., and Abrams, S. (2005). Source code that talks: an exploration of Eclipse task comments and their implication to repository mining. *ACM SIGSOFT Software Engineering Notes*, 30(4):1.

Zampetti, F., Scalabrino, S., Oliveto, R., Canfora, G., and Di Penta, M. (2017). How Open Source Projects Use Static Code Analysis Tools in Continuous Integration Pipelines. *IEEE International Working Conference on Mining Software Repositories*, pages 334–344.

Zannier, C., Chiasson, M., and Maurer, F. (2007). A model of design decision making based on empirical results of interviews with software designers. *Information and Software Technology*, 49(6):637–653.

Zannier, C. and Maurer, F. (2006). Foundations of Agile Decision Making from Agile Mentors and Developers. In Abrahamsson, P., Marchesi, M., and Succi, G., editors, *Extreme Programming and Agile Processes in Software Engineering*, pages 11–20, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ziftci, C. and Reardon, J. (2017). Who broke the build? Automatically identifying changes that induce test failures in continuous integration at google scale. In *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track, ICSE-SEIP 2017*, pages 113–122.