# More than Code: Contributions in Scrum Software Engineering Teams

Frederike Ramin
frederike.ramin@student.hpi.de
Hasso Plattner Institute
University of Potsdam, Germany

Christoph Matthies
christoph.matthies@hpi.de
Hasso Plattner Institute
University of Potsdam, Germany

Ralf Teusner
ralf.teusner@hpi.de
Hasso Plattner Institute
University of Potsdam, Germany

## ABSTRACT

Motivated and competent team members are a vital part of Agile Software development and make or break any project's success. Motivation is fostered by continuous progress and recognition of efforts. These concepts are founding pillars of the Scrum methodology, which focuses on self-organizing teams. The types of contributions Scrum development team members make to a project's progress are not only technical. However, a comprehensive model comprising the varied contributions in modern software engineering teams is not yet established. We propose a model that incorporates contributions of all Scrum roles, explicitly including those which are not directly related to project artifacts. It improves the visibility of performed tasks, acts as a starting point for team retrospection, and serves as a foundation for discussion in the research community.

## CCS CONCEPTS

• **Software and its engineering → Agile software development**.

## KEYWORDS

Scrum, Agile Software Development, Teamwork, Contribution

**ACM Reference Format:**
Frederike Ramin, Christoph Matthies, and Ralf Teusner. 2020. More than Code: Contributions in Scrum Software Engineering Teams. In *IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20), May 23–29, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3387940.3392241

## 1 INTRODUCTION

Modern software engineering features collaborative, iterative development by teams following development processes and practices customized to their project contexts. The Agile Manifesto emphasizes the importance of teams, stating that "the best [...] designs emerge from self-organizing teams" [6]. Agile methods, based on these principles, such as Scrum, have become the de facto standard in professional software engineering [1, 19]. These processes highlight the importance of human factors [13]. They focus on teams that are cross-functional and which include team members with all the capacities and competencies required to accomplish the project work [18]. Therefore, the types of contributions that software engineers make to the progress and success of modern software projects are varied. They not only include technical aspects, e.g., source code changes, but also process improvement activities, meeting facilitation, and effective communication with colleagues [9]. For the remainder of this paper we rely on the following definition:

> *Contribution*: Any activity, demanding human resources, that adds to the fulfillment of project goals, by adding value to the developed product or the (future) effectiveness of the team.

Ford et al. characterized the actions of software engineers. They list tasks such as learning, knowledge dissemination, feedback, and networking, which are vital to team success [5]. Recent studies of software developers found that coding-related activities only took up one-fourth of their total work, with another fourth being used for collaborative activities [16]. The Scrum framework acknowledges these different task profiles, proposing specific roles, i.e. *Product Owner* (PO), *Scrum Master* (SM) and *Development Team* (Dev.), with distinct responsibilities [18].

Agile approaches rely on clear communication and visibility of progress to enable efficient collaboration and effective teams [11]. Capturing and categorizing the contributions of team members is a cornerstone of ensuring team awareness regarding accomplished work. It enables the appropriate valuation and appraisal of contributions to team efforts necessary for successful teamwork.

## 2 BACKGROUND

The characterization of contributions to software projects, and their assessment, is an ongoing field of research. Previous work in this domain includes research on models of software engineering *tasks*, i.e. specific activities, and the personality traits of developers [7, 22]. These studies are predominantly concerned with traditional software engineering approaches, categorizing contributions by the different phases of software development or the roles that perform them [17, 22]. Additional previous work focused on the technical aspects of software engineering [9], dividing engineer's tasks into activities such as IT support and database administration [21] or compilation and debugging [20]. While multiple models of teamwork have been proposed, the activities, tasks, and responsibilities specific to modern, Agile software engineering teams have not been the core focus of recent studies.

In 1989, Goldstein classified software project team members according to the tasks they regularly perform [8]. He identified four distinct groups: *programmers*, *analysts*, *maintainers*, and *supporters*, each contributing in different ways. Similarly, Glass et al. presented an early model of software engineering contributions, listing tasks


*ICSEW'20, May 23–29, 2020, Seoul, Republic of Korea*

ACM ISBN 978-1-4503-7963-2/20/05…$15.00
https://doi.org/10.1145/3387940.3392241

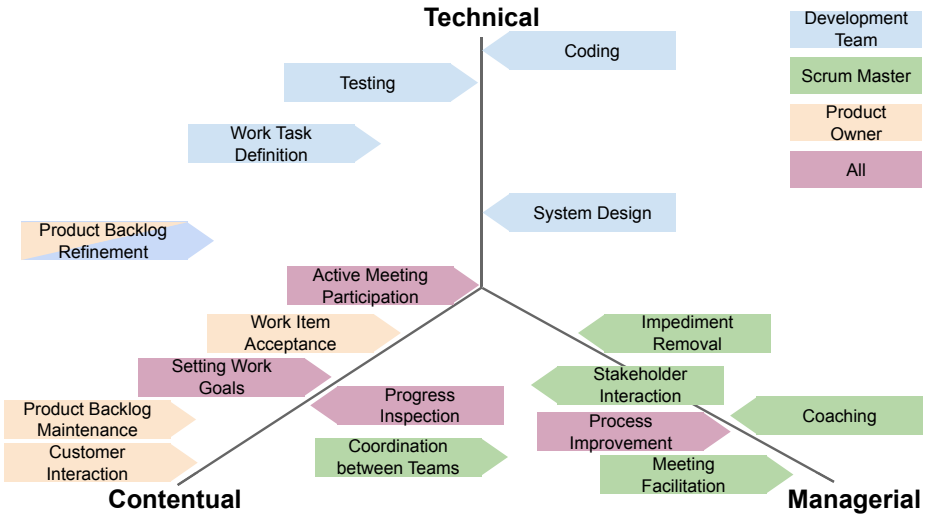


**Figure 1: Visual representation of our initial contribution model regarding contributions to Scrum teamwork by participant roles (color coded). The closer a contributions is placed at the ends of a dimension, the more it is related to this category.**

such as debugging, reviews or user training [7]. The authors tag these activities as either routine or non-routine. They conclude that in the domain of software engineering the intellectual, i.e. non-routine, tasks dominated routine tasks by a ratio of almost 4 to 1. More recently, Gousios et al. have pointed out, that, even in the field of Software Repository Mining, "no clear definition" of software engineering contributions existed [9]. However, the authors state that source code should not be the only contribution metric, especially in the context of Agile development. They identified technical contributions through a hierarchical, top-down approach, based on project assets and the actions that can be performed on them. Nonetheless, contributions not directly resulting in asset changes, e.g. facilitating meetings, are disregarded in this model.

## 3 A MODEL FOR CONTRIBUTIONS IN SCRUM

Little previous research on contributions to teamwork has focused on the human aspects that Agile methods and the Scrum process framework stress. We, therefore, constructed an initial model from first principles, based on the seminal Scrum Guide [18] by Ken Schwaber and Jeff Sutherland, the originators of the method. We successively coded the guide's text, particularly focusing on the sections dealing with Scrum roles, artifacts, and meetings. For every paragraph, we extracted the passages which mentioned work items, project requirements and responsibilities that the process stipulates for the different Scrum participants. These items, after deduplication and clustering, represent the contributions of Scrum team members to the development process. For each identified contribution, we assigned a short name and noted the topic as well as which role was designated for it. Table 1 contains examples of this process.

Table 2 presents the 17 individual Scrum teamwork contributions we extracted, assigned to the Scrum roles: *Development Team*, *Scrum Master* and *Product Owner*. We explicitly included the role of *All*, to highlight contributions that involve a high level of collaboration between all roles involved in the development process.

**Table 1: Examples of annotated Scrum Guide [18] extracts**

| Text | Name | Role |
|---|---|---|
| "The SM serves the Development Team [...], including: [...] Removing impediments to the Development Team" | Impediment Removal | SM |
| "The Development Team consists of professionals who do the work of delivering a [...] product at the end of each Sprint." | Coding | Dev. |
| "Each [Product] Increment is [...] thoroughly tested, ensuring that all Increments work together." | Testing | Dev. |

We relied on previous work in the area of project management to identify the main traits and topic areas that teamwork contributions are classified by. Ebert and de Man [3] grouped software engineering knowledge into the three areas: *Project* (e.g. requirements, budget, timing, milestones), *Product* (e.g. product features, relations to other products or standards) and *Process* (e.g. business processes, workflows, responsibilities). Similarly, Wynekoop and Walz [23] divided the traits of top-performing software developers into three related categories: (i) traits of those who best "make things work", (ii) traits of those who best communicate with users, and (iii) traits of those "destined for management". We employ a combination of these categories as dimensions to characterize contributions, based on the language of the Scrum Guide:

- **Technical**: How much does the contribution add to the product increment, utilizing technical knowledge and skills?
- **Contentual**: To what degree does the contribution influence the product's prospect and direction?
- **Managerial**: To what degree is the contribution concerned with adapting the work process, not the product?

**Table 2: Overview of the 17 Scrum project work contributions included in the model, grouped by roles. Based on [18].**

| Name | Roles | Description |
|---|---|---|
| Coding | Dev. | Producing a product increment which satisfies the Sprint Goal |
| Testing | Dev. | Writing and executing software tests, ensuring that product increments work together |
| Work Task Definition | Dev. | Creating implementation tasks from Sprint Backlog items and a plan for delivering them |
| System Design | Dev. | Planning the software architecture and fundamental structure of the product |
| Product Backlog Refinement | Dev., PO | Adding details and estimates to work items and adjusting their priorities |
| Product Backlog Maintenance | PO | Modifying the Product Backlog by adding work items and their priorities |
| Customer Interaction | PO | Extracting product requirements and prioritization by communicating with the customer |
| Work Item Acceptance | PO | Checking the produced product increment for compatibility with the product vision |
| Meeting Facilitation | SM | Scheduling and leading through meetings, preparing an agenda and keeping the time-box |
| Impediment Removal | SM | Resolving identified problems that hinder project progress, improving team workflows |
| Stakeholder Interaction | SM | Collecting feedback from stakeholders and changing the interactions with the outside world |
| Coaching | SM | Leading and passing on process knowledge to team members |
| Coordination between Teams | SM | Communicate and distribute work between teams in multi-team settings |
| Setting Work Goals | All | Structuring the work of the next iteration (Sprint Planning) or the next day (Daily Scrum) |
| Active Meeting Participation | All | Being involved and contributing to the meeting goals, supporting a positive attitude |
| Process Improvement | All | Inspecting and adapting the employed process (Sprint Retrospective) |
| Progress Inspection | All | Examine the work performed in the latest product increment |

These three model dimensions are non-exclusive. Project contributions may involve technical, contentual and managerial aspects simultaneously, though to different extents. For example, a Scrum Master solving identified issues as part of *Impediment Removal* is most likely dealing with adapting work processes. However, depending on the specific issue, these changes may also require some technical skills or may affect the developed product and how it is built[1]. The proposed model dimensions thus create a gravity field to locate and categorize project and teamwork contributions.

Figure 1 presents our visualized model of contributions to project work in Scrum teams. The closer a specific contribution is to the edges of the graph, the more it is related to only one or two dimensions. For example, the contribution of *Coding*, i.e. producing source code and "delivering a potentially releasable increment" [18], is at the far end of the *Technical* dimension.

## 4 DISCUSSION

The proposed model represents a structured exploration of the project member's contributions described by the Scrum Guide [18]. We explicitly constricted the scope of this initial model to promote clarity and traceability of construction. The model represents the state of the Scrum Guide in its latest version of November 2017. The related literature on Scrum is vast and contains many more collections of tasks and roles of team members in the software engineering domain [5, 11]. Since the Scrum process model needs to be adapted to the context in which it is used, as well as to the team which uses it, many different implementations exist in practice. As such, the presented model serves as a basic structure, representative of key contributions in unmodified, theoretical, "vanilla Scrum" [10].

[1]Conway's Law: "organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations." [2]

The Scrum Guide itself lists adaptation as one of the core elements of the underlying theory and states that "specific tactics for using the Scrum framework vary and are described elsewhere" [18]. While Scrum contains prescriptions for team organization, it contains few specifics on how software development activities should be performed. Our model, therefore, reflects this approach. In practice, Scrum is often employed in conjunction with additional, complementary methods, most notably XP, which suggests specific tactics on how work is to be performed, e.g. Pair Programming [11].

Future work will focus on this aspect, including contributions from additional sources, further Agile methods, and industry practice in the model. The base model presented here can then also be employed to show differences between the project contributions expected in different Agile methods. Furthermore, we envision the proposed model to be helpful in software engineering projects, e.g. in the following use cases:

*Scrum Team Status Check.* The model can be used as a means of conformance analysis, contrasting a team's process and self-identified teamwork contributions to those designated by the Scrum Guide. In an initial step, after identifying their own roles, team members can investigate whether they make all of the contributions listed for their specific role. Any mismatches in this process represent starting points for discussion in the team. This allows retrospection on the chosen process adaptations and their rationales.

*Team Contribution Analysis.* The list of Scrum team contributions, see Table 2, can be reviewed, discussing for every item whether it is clear who currently is or should make a specific contribution. This can identify project contributions that have previously been overlooked in a team.

*Contribution Awareness.* The proposed teamwork model contains many contributions of an interpersonal and non-technical nature, which are core to Scrum but receive less focus in more traditional software development approaches. A visual representation of these contributions as presented in Figure 1, can improve developers' awareness of their own behavior [15] and their contributions besides writing code.

*Agile Process Coaching.* The Scrum contributions to project work represent the daily activities and responsibilities of Scrum team members. They define what a developer, Scrum Master, or Product Owner will likely spend a significant portion of their work time on. While the proposed model focuses solely on the Scrum process framework, it can be used to highlight the differences in work activity between different Agile methods. By highlighting which contributions would be impacted by changes in the development process flow, shifts in daily work can be anticipated. This approach thus allows an easier transition from one method to another by focusing on the changes in daily contributions that need to happen.

Our model makes the often implicit contributions of Scrum team members to project progress explicit. It allows comparisons of one's own team state to the "by the book" Scrum process. These types of analyses are suitable to foster self-reflection and retrospection regarding teamwork processes and contributions in teams. They may, therefore, prove particularly useful in *Retrospectives*, Scrum's implementation of software process improvement [12]. In Retrospective Meetings, the team reflects on what aspects of the last development iteration were beneficial and should be kept, and what aspects should be improved in the future [18]. To structure these meetings, encourage active participation and to break the usual routine, interactive *Retrospective Games* have been introduced by Agile practitioners [4]. These games often make use of brainstorming, visualizations or metaphors to generate process improvement ideas. The proposed model of Scrum project contributions can provide a default view of Scrum for teams to compare themselves against. It complements the array of existing tools to facilitate productive team reflection and retrospection.

## 5 CONCLUSION

Software development project teams are often characterized as homogeneous groups, without taking the varied tasks and roles of team members into account [5]. However, contributions to a software development project are not constricted to coding, with developers routinely spending only about half of their workday working on their computers [16]. This fact is especially relevant for teams employing the Scrum process framework, which highly depends on effective self-organization, communication, and collaboration [14]. This reality of modern software engineering teams is reflected only inadequately in previous models of teamwork contributions. We, therefore, present a model based on project contributions defined in the Scrum Guide [18] to address this research gap. This model, categorizing the teamwork contributions of Scrum roles using the dimensions *Technical*, *Contentual* and *Managerial*, is an initial proposal. It is open to refinement, discussion, and enrichment using additional sources and Agile method descriptions. Our model fosters retrospection and represents a first step towards a more complete view and understanding of the types of contributions Scrum team members make to successful projects.

## REFERENCES


[1] CollabNet Inc. 2019. *13th Annual State of Agile Report.* Technical Report. https://www.stateofagile.com/#ufh-i-521251909-13th-annual-state-of-agile-report

[2] M E Conway. 1968. How do committees invent. *Datamation* 14, 4 (1968), 28–31.

[3] Christof Ebert and Jozef De Man. 2008. Effectively utilizing project, product and process knowledge. *Information and Software Technology* 50, 6 (may 2008), 579–594. https://doi.org/10.1016/j.infsof.2007.06.007

[4] Derby Esther and Diana Larsen. 2006. *Agile retrospectives: Making Good Teams Great.* Pragmatic Bookshelf. 200 pages.

[5] Denae Ford, Tom Zimmermann, Christian Bird, and Nachiappan Nagappan. 2017. Characterizing Software Engineering Work with Personas Based on Knowledge Worker Actions. In *ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 394–403. https://doi.org/10.1109/ESEM.2017.54

[6] Martin Fowler and Jim Highsmith. 2001. The Agile Manifesto. *Software Development* 9, 8 (2001), 28–35.

[7] Robert L. Glass, Iris Vessey, and Sue A. Conger. 1992. Software tasks: Intellectual or clerical? *Information & Management* 23, 4 (1992), 183–191. https://doi.org/10.1016/0378-7206(92)90043-F

[8] David K. Goldstein. 1989. The Effects of Task Differences on the Work Satisfaction, Job Characteristics, and Role Perceptions of Programmer/Analysts. *Journal of Management Information Systems* 6, 1 (1989), 41–58. https://doi.org/10.1080/07421222.1989.11517848

[9] Georgios Gousios, Eirini Kalliamvakou, and Diomidis Spinellis. 2008. Measuring developer contribution from software repository data. In *Proceedings of the 2008 international working conference on Mining software repositories*. ACM, 129. https://doi.org/10.1145/1370750.1370781

[10] Lucas Gren, Richard Torkar, and Robert Feldt. 2017. Group development and group maturity when building agile teams: A qualitative and quantitative investigation at eight large companies. *Journal of Systems and Software* 124 (2017), 104–119. https://doi.org/10.1016/j.jss.2016.11.024

[11] Henrik Kniberg. 2015. *Scrum and XP From the Trenches* (2nd ed.). C4Media.

[12] Christoph Matthies. 2019. Feedback in Scrum: Data-informed Retrospectives. In *Proceedings of the 41st International Conference on Software Engineering: Companion Proceedings*. 198–201. https://doi.org/10.1109/ICSE-Companion.2019.00081

[13] Christoph Matthies, Johannes Huegle, Tobias Dürschmid, and Ralf Teusner. 2019. Attitudes, Beliefs, and Development Data Concerning Agile Software Development Practices. In *IEEE/ACM 41st International Conference on Software Engineering: Software Engineering Education and Training*. IEEE, 158–169. https://doi.org/10.1109/ICSE-SEET.2019.00025

[14] Christoph Matthies, Thomas Kowark, and Matthias Uflacker. 2016. 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. https://peer.asee.org/27259

[15] André N Meyer. 2018. Fostering software developers' productivity at work through self-monitoring and goal-setting. In *Proceedings of the 40th International Conference on Software Engineering Companion Proceedings*. ACM Press, 480–483. https://doi.org/10.1145/3183440.3183446

[16] André N Meyer, Gail C Murphy, Thomas Fritz, and Thomas Zimmermann. 2019. *Developers' Diverging Perceptions of Productivity*. Apress, 137–146. https://doi.org/10.1007/978-1-4842-4221-6_12

[17] Adesina S. Sodiya, Olumide Babatope Longe, S. Adebukola Onashoga, Oludele Awodele, and L. O. Omotosho. 2007. An Improved Assessment of Personality Traits in Software Engineering. *Interdisciplinary Journal of Information, Knowledge, and Management* 2 (2007), 163–177. https://doi.org/10.28945/107

[18] Ken Schwaber and Jeff Sutherland. 2017. *The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game.* Technical Report. scrumguides.org. 19 pages. http://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[19] Scrum Alliance. 2018. *State of Scrum 2017-2018: Scaling and Agile Transformation.* Technical Report. http://info.scrumalliance.org/State-of-Scrum-2017-18

[20] Janice Singer, Timothy Lethbridge, Norman Vinson, and Nicolas Anquetil. 2010. An examination of software engineering work practices. In *CASCON First Decade High Impact Papers*. ACM, 174–188. https://doi.org/10.1145/1925805.1925815

[21] Lori Anderson Snyder, Deborah E. Rupp, and George C. Thornton. 2006. Personnel Selection of Information Technology Workers: The People, the Jobs, and Issues for Human Resource Management. In *Research in personnel and human resources management*. Research in Personnel and Human Resources Management, Vol. 25. Elsevier JAI, 305–376. https://doi.org/10.1016/S0742-7301(06)25008-4

[22] Manuel Wiesche and Helmut Krcmar. 2014. The relationship of personality models and development tasks in software engineering. In *SIGMIS-CPR'14*. ACM, 149–161. https://doi.org/10.1145/2599990.2600012

[23] Judy L. Wynekoop and Diane B. Walz. 2000. Investigating traits of top performing software developers. *Information Technology & People* 13, 3 (2000), 186–195. https://doi.org/10.1108/09593840010377626