# An Additional Set of (Automated) Eyes: Chatbots for Agile Retrospectives

Christoph Matthies, Franziska Dobrigkeit, Guenter Hesse
*Hasso Plattner Institute, University of Potsdam, Germany*
christoph.matthies@hpi.de, franziska.dobrigkeit@hpi.de, guenter.hesse@hpi.de

***Abstract*—With the recent advances in natural-language processing, combined with the ability to analyze large amounts of data across various domains, software bots can become virtual team members, providing an additional set of automated eyes and additional perspectives for informing and supporting teamwork. In this paper, we propose employing chatbots in the domain of software development with a focus on supporting analyses and measurements of teams' project data. The software project artifacts produced by agile teams during regular development activities, e.g. commits in a version control system, represent detailed information on how a team works and collaborates. Analyses of this data are especially relevant for agile retrospective meetings, where adaptations and improvements to the executed development process are discussed. Development teams can use these measurements to track the progress of identified improvement actions over development iterations. Chatbots provide a convenient user interface for interacting with the outcomes of retrospectives and the associated measurements in a chat-based channel that is already being employed by team members.**

***Index Terms*—chatbot, agile software development, Scrum, retrospective, software process improvement**

## I. Introduction

Software tools that help software users and software developers in their daily tasks have been created for as long as code has been written [1]. Today, a countless number of such tools exist, from large and powerful to small and simple. They support various activities such as architectural design, test case generation [2] or collaboration within teams [3].

### A. Chatbot Definition

Traditionally, support tools which take actions on behalf of a user were referred to as *software agents* [4], from the Latin "agere", meaning "to take action". The importance of automation and autonomy involved with many of these systems is highlighted in the designation *softbots* [5], short for software robots. More recently, the terms *chatbot*, *chatterbot* or simply *bot* have been used to refer to support tools that employ a conversational-style user interface [6]. These types of text interfaces to software services have gained popularity due to the increasing role of instant messaging in the workplace as well as in social encounters [7]. Messaging platforms which are widely adopted include *Facebook Messenger* for social networking or *Slack* for work-related communication [8]. Therefore, these platforms, where users already frequently interact and collaborate, are where chatbots are often employed [9].

### B. Chatbot Operation

At its simplest, a (chat)bot is a computer program which performs various predetermined operations, receiving commands via chat messages and performing the requested action [10]. Figure 1 depicts the general interaction flow through such a system. Initially, a user sends a command as a chat message to the bot (1). The bot parses the message and performs operations based on the message's contents (2). To fulfill the request, the bot might need to request (3) and receive (4) additional information from third-party sources, such as knowledge bases or internet resources [1]. Finally, the bot computes a result (5), compiles a chat message and sends it as a reply to the user (6).

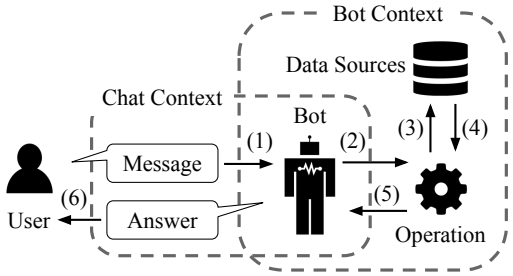


Fig. 1. Overview of Chatbot operations, adapted from Inokuchi et al. [10]

In this manner chatbots can provide a conduit between users and tools, integrating external services and additional information sources into already existent communication channels [6].

### C. Chatbots in Agile Teams

With the improvement of natural-language processing as well as big data analytics and their application in conversational bots [6], bots can grow from simple scripted tools to "virtual teammates" [11], informing and supporting teamwork.

> In this paper, we propose an approach for employing chatbots in teams with a focus on analyses and measurements of the data produced by the team, particularly in the domain of software development.

Modern software development teams employing agile methods spend a substantial amount of their time interacting with development tools, such as *Version Control Systems* (VCS), during regular development activities. The software project artifacts [12] produced using these systems, i.e. commits in a VCS containing code changes, represent a "gold-mine of actionable information" [13], i.e. knowledge on how a team

works and collaborates [14]–[16]. While analyses of this data can be used for a variety of purposes, they are especially relevant for software development teams in the context of their efforts to adapt and improve the executed development process. For example, relevant metrics such as burndown charts can be constructed by evaluating the software project artifacts of a development iteration [17]. Chatbots provide a convenient user interface for interacting with the project data and these types of analyses [6] in a chat-based channel already being used by teams on a daily basis.

## II. Related Work on Chatbots

A wide variety of application scenarios for chatbot usage in teams has been described, ranging from information processing and sharing to detecting and monitoring activities in team communications and even providing recommendations regarding possible next tasks [6]. Of course, they have also been used to search for and send animated cat gifs [18] to team members [9].

### A. Software Engineering

Software developers belong to the early adopters of automation and bot use as they are familiar with automated tools increasing code quality and team productivity [6]. A particularly interesting approach is the *BuildBot* by Ablett et al., a chatbot in the true nature of the term: it communicates with developers using sound [19]. The bot uses a robotic interface, a small Sony AIBO robot dog, which monitors the build status of software within an agile continuous integration approach. Should the build be broken, e.g. because tests fail, the robot walks to the developer whose code is responsible for the failure and notifies them (in a playful way). The authors argue that the system increased awareness of the software's build status and helped agile teams with self-supervision [19]. In modern software development, chat solutions have disrupted previous software development processes and have replaced email in some cases [9], [20]. Fitzpatrick et al. attempted to combine the informal discussions developers have about code with version control information reporting code changes in a ticker tape form [21]. They point out that the studied software developers used the close integration of versioning information with chat functionality for such varied tasks as growing team culture, marking phases of work or managing work interruptions. Similarly, Lebeuf et al. report on industry teams that use chatbots to provide instructions for development procedures, e.g. merging feature branches, to monitor website outages, or to manage code deployments [6], [11]

### B. Human Factors

In addition to uses in communication and data access, chatbots have previously also been employed to tackle issues related to human factors in software engineering. By issuing commands to bots via chat messages in group chats, every operation is shared with the team as well as logged and persisted in the chat log [10]. This enables transparency and awareness of other team members interactions with the chatbot and its functions. It enables nontechnical team members to engage with the bot's capabilities without explicitly needing domain expertise [6]. Having human team members and bots share the same chat context enables use cases such as adopting the chat solution as a distributed command line or a collaborative debugger [7]. Especially relevant to the topic of employing chatbots for process improvement approaches within agile teams is the idea of using bots to regulate individual and team tasks and goals [22]. For collaboration to succeed in a team, all members must understand and share the goals set for the team as well as the actions necessary to achieve these goals [11]. Bots can initiate and track reminders set in earlier meetings as well as help to monitor and visualize progress towards certain team goals [22].

## III. Chatbots in Agile Retrospectives

In order to improve the executed software development process within a team, the current status has to be measured, so that changes in the future can be detected. The agile process framework currently most popular in industry [23], *Scrum*, calls for a specific meeting with the goal of process improvement: the *retrospective meeting* [24].

### A. The Retrospective Meeting

As the name suggests, in the dedicated retrospective meeting the team looks back at the most recent development iteration and decides which aspects of the process should be kept and what should be changed in the future. Issues that should be improved are recorded as action items as outcomes of the meeting. In the next retrospective, the development team then decides whether headway has been made on the previously defined actions items [25]. While this decision relies on team members understanding of their executed process, the Scrum Guide also specifies that "decisions to optimize value and control risk are made based on the perceived state of the artifacts." It goes on to define a task which is to "detect incomplete transparency by inspecting the artifacts, sensing patterns [...] and detecting differences between expected and real results" [24]. Similarly, Derby and Larsen suggest to "start with the hard data" in retrospectives [17], including metrics such as velocity, defect count, number of stories completed or amount of refactored code.

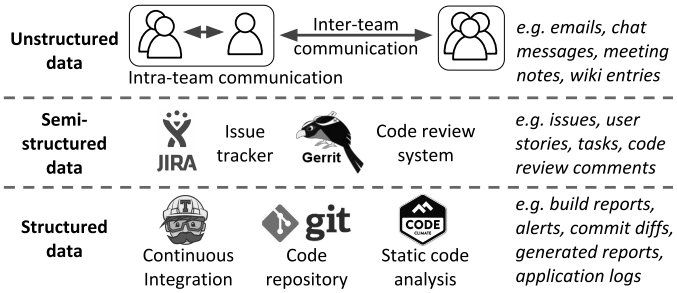


Fig. 2. Sources and examples of project artifacts in the software development domain providing data of different levels of structure.

Figure 2 provides an exemplary overview of the different sources of development artifacts available in an agile

software development team. In teams with large code bases and therefore a large number of development artifacts to inspect, collection and analysis is a challenge, especially for human actors. On the other hand, a bot, equipped with human instruction on what patterns to detect and sense, e.g. desired outcomes from retrospective action items, can analyze large quantities of data.

### B. Chatbots for Process Improvement

Developers have built or customized chatbots to support their daily lives, also outside of work and development activities. Early examples of this are bots which have been used to assist with information retrieval [26]. More recent examples include bots that can help decide where to go to lunch or help keep a grocery list as well as provide entertainment, i.e. to "search for images and to then add moustaches to them" [9]. With the varied background of bot tasks in mind, we propose having a bot track the progress of retrospective action items, which can come from a variety of contexts. As agile software development teams already spend significant amounts of time communicating in chat solutions and sometimes also already use these during retrospectives [27], this means no context switch is necessary. Instead of the Scrum Master or an agile coach analyzing project data and providing a different perspective to team members during retrospectives, it could be a bot that retrieves and processes the information, leaving humans more time to interpret it.

### C. Integration into the Agile Processes

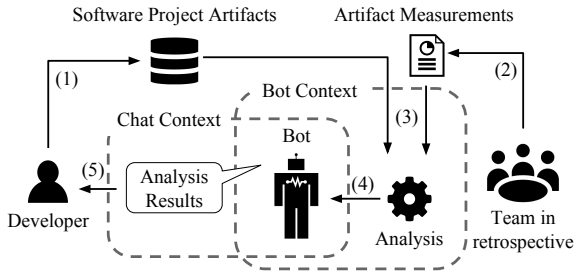


Fig. 3. Overview of using a chatbot for analysis of software project artifacts

Figure 3 describes the operation of the envisioned chatbot and its integration into the agile process framework. Agile software developers produce software project artifacts in their daily development activities during a development iteration (1). At the end of an iteration, they hold a retrospective meeting, focusing on those practices that went well and should be continued as well as defining action items for those that should be changed. To track these action items measurements based on the created project artifacts are created by the team (2), e.g. the number of commits into the VCS which increase code complexity but do not provide tests. The bot applies these measurements to the collected project data and stores the analysis values over time (4). Using the chat service, it communicates the results, i.e. the change of the measurement, to the development team members (5). This information can be used in the next retrospective to initiate discussions on the state of an action item, on the basis of concrete data points, in addition to—and possibly in contrast to—the perceptions of team members.

## IV. Integration into Existing Tools

The use of tools is vital to collaboration within modern software development teams, especially for geographically distributed teams. Tools enable the automation, and control of the entire development process [3]. Automation is also key to many common practices of modern agile software development, such as frequent product deliveries, which would otherwise not be feasible [28]. There is a range of tools available, specifically aimed at supporting the retrospective meetings of agile teams through automation, by setting reminders, archiving action items [29] as well as facilitating activities [30]. The chat solution Slack, which is popular for work-related instant messaging [9], features extension points and APIs for third-party applications and bots to interact with users via a conversation [31]. These possibilities have already been used to create initial chatbots on the Slack platform that support agile teams in their retrospectives [27], [32], [33]. These bots remind team members of the retrospective meeting and can record the individual statements of developers, summarizing results and archiving outcomes. While they automate the process, this approach still fully relies on team members' perceptions to provide the inputs for discussion. Currently available bots can automate the tedious organizational tasks, but do not provide an additional perspective based on the available project data. In order to provide this additional perspective, the chatbot must be enabled to assess a team's situation by measuring the team's development data. This can be achieved through various metrics designed for agile practices [34]–[36] or by using tools, such as the git command line [37], that developers are already familiar with from their daily development activities.

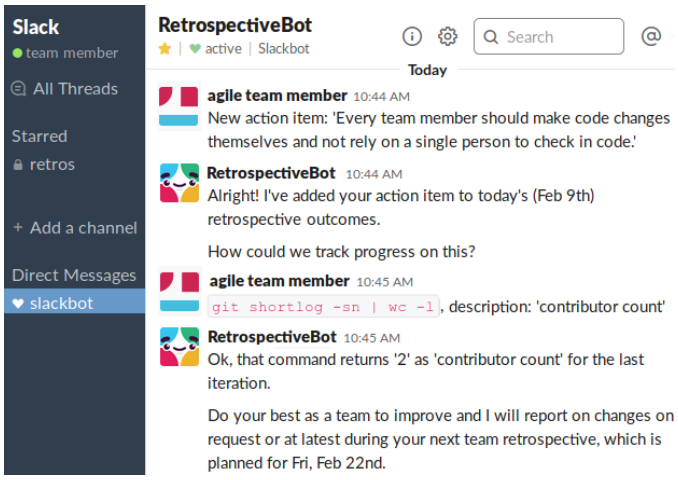


Fig. 4. User interface mockup of the interaction between a developer and a retrospective bot tasked with tracking retrospective action items.

Figure 4 shows an exemplary interaction during a retrospective between a development team member and the envisioned retrospective bot. A team member has identified an improvement for the next development iteration, i.e. that everyone should check in code. They notify the retrospective bot of this new action item and provide a short command line call, measuring the number of unique contributors in the VCS. This measurement acts as a proxy for assessing progress on the new action item. At this point, the bot can take over, repeatedly taking measurements using the provided code statement, and can inform team members of the status of the action item. In the next retrospective the results can then be discussed and interpreted by the team.

## V. Conclusion

It seems inevitable that software developers will see more automation and bots being introduced to support their workflow and development-related activities [6]. This includes both coding activities as well as tending to and improving the development process executed in teams. This proposal for using chatbots in agile retrospectives as an additional set of automated eyes on the software project data of teams is a step in this direction.

## References


[1] E. Paikari and A. van der Hoek, "A framework for understanding chatbots and their future," in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering - CHASE '18*. ACM Press, 2018, pp. 13–16.

[2] A. I. Wasserman, "Tool integration in software engineering environments," in *Software Engineering Environments*, F. Long, Ed. Springer Berlin Heidelberg, 1990, pp. 137–149.

[3] F. Lanubile, C. Ebert, R. Prikladnicki, and A. Vizcaíno, "Collaboration Tools for Global Software Engineering," *IEEE Software*, vol. 27, no. 2, pp. 52–55, 2010.

[4] H. S. Nwana, "Software agents: an overview," *The Knowledge Engineering Review*, vol. 11, no. 03, p. 205, sep 1996.

[5] S. R. Hedberg, "Intelligent agents: The first harvest of softbots looks promising," *IEEE Intelligent Systems*, no. 4, pp. 6–9, 1995.

[6] C. Lebeuf, M.-A. Storey, and A. Zagalsky, "Software Bots," *IEEE Software*, vol. 35, no. 1, pp. 18–23, jan 2018.

[7] S. Chan, B. Hill, and S. Yardi, "Instant Messaging Bots: Accountability and Peripheral Participation for Textual User Interfaces," *Proceedings of the International ACM SIGGROUP Conference on Supporting Group Work (GROUP '05)*, pp. 113–115, 2005.

[8] Jeffrey M. Perkel, "How scientists use Slack," *Nature News*, vol. 541, no. 7635, p. 123, 2014.

[9] B. Lin, A. E. Zagalsky, M.-A. Storey, and A. Serebrenik, "Why Developers Are Slacking Off: Understanding How Software Teams Use Slack," in *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion - CSCW '16 Companion*. ACM Press, 2016, pp. 333–336.

[10] A. Inokuchi, H. Tamada, H. Hata, and M. Tsunoda, "Toward Obliging Bots for Supporting Next Actions," in *2016 4th Intl Conf on Applied Computing and Information Technology/3rd Intl Conf on Computational Science/Intelligence and Applied Informatics/1st Intl Conf on Big Data, Cloud Computing, Data Science & Engineering (ACIT-CSII-BCD)*. IEEE, dec 2016, pp. 183–188.

[11] C. Lebeuf, M.-A. Storey, and A. Zagalsky, "How Software Developers Mitigate Collaboration Friction with Chatbots," *arXiv preprint arXiv:1702.07011*, feb 2017.

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

[13] J. Guo, M. Rahimi, J. Cleland-Huang, A. Rasin, J. H. Hayes, and M. Vierhauser, "Cold-start software analytics," in *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*. ACM Press, 2016, pp. 142–153.

[14] C. Rosen, B. Grawi, and E. Shihab, "Commit guru: analytics and risk prediction of software commits," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering - ESEC/FSE 2015*. ACM Press, 2015, pp. 966–969.

[15] C. Matthies, R. Teusner, and G. Hesse, "Beyond Surveys: Analyzing Software Development Artifacts to Assess Teaching Efforts," in *IEEE Frontiers in Education Conference (FIE)*. IEEE, 2018.

[16] E. A. Santos and A. Hindle, "Judging a commit by its cover," in *Proceedings of the 13th International Workshop on Mining Software Repositories - MSR '16*. ACM Press, 2016, pp. 504–507.

[17] D. Esther and D. Larsen, *Agile retrospectives - Making Good Teams Great*. Pragmatic Bookshelf, 2007, vol. 24, no. 5.

[18] J. Eppink, "A brief history of the GIF (so far)," *Journal of Visual Culture*, vol. 13, no. 3, pp. 298–306, 2014.

[19] R. Ablett, E. Sharlin, F. Maurer, J. Denzinger, and C. Schock, "BuildBot: Robotic Monitoring of Agile Software Development Teams," in *RO-MAN 2007 - The 16th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2007, pp. 931–936.

[20] V. Käfer, D. Graziotin, I. Bogicevic, S. Wagner, and J. Ramadani, "Communication in open-source projects-end of the e-mail era?" in *Proceedings of the 40th International Conference on Software Engineering Companion Proceeedings - ICSE '18*. ACM Press, 2018, pp. 242–243.

[21] G. Fitzpatrick, P. Marshall, and A. Phillips, "CVS integration with notification and chat," in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work - CSCW '06*. ACM Press, 2006, p. 49.

[22] M.-A. Storey and A. Zagalsky, "Disrupting developer productivity one bot at a time," in *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering - FSE 2016*, vol. 13, no. 5. ACM Press, oct 2016, pp. 928–931.

[23] Scrum Alliance, "The State of Scrum Report 2017 Edition," Scrum Alliance, Tech. Rep., 2017. [Online]. Available: https://www.scrumalliance.org/scrum/media/ScrumAllianceMedia/FilesandPDFs/StateofScrum/State0fScrum_2016_FINAL.pdf

[24] K. Schwaber and J. Sutherland, "The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game," Tech. Rep., 2017. [Online]. Available: http://scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf

[25] H. Kniberg, *Scrum and XP from the Trenches*. C4Media, 2007.

[26] E. M. Voorhees, "Software Agents for Information Retrieval," *Association for the Advancement of Artificial Intelligence (AAAI) Technical Report SS-94-03*, pp. 126–129, 1994.

[27] Standuply, "Retrospective Meeting Slack Bot," 2019. [Online]. Available: https://standuply.com/retrospective-meeting

[28] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," *IEEE Software*, vol. 33, no. 3, pp. 94–100, 2016.

[29] GoReflect, "goReflect - Continuous Retrospectives for Agile Improvement," 2019. [Online]. Available: https://www.goreflect.com/

[30] Retrium, "The era of boring retrospectives is over," 2019. [Online]. Available: https://www.retrium.com

[31] Slack Technologies Inc., "Enabling interactions with bots," 2019. [Online]. Available: https://api.slack.com/bot-users

[32] R. Sharp, "Retrobot - a slack bot for retrospectives," 2019. [Online]. Available: https://github.com/remy/retrobot

[33] K. McAuliffe, "Retrobot - a Slack bot to record retrospectives!" 2019. [Online]. Available: https://github.com/PebbleKat/retrobot

[34] A. Ju and A. Fox, "TEAMSCOPE: measuring software engineering processes with teamwork telemetry," in *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education - ITiCSE 2018*. ACM Press, 2018, pp. 123–128.

[35] C. Matthies, T. Kowark, M. Uflacker, and H. Plattner, "Agile metrics for a university software engineering course," in *IEEE Frontiers in Education Conference (FIE)*. IEEE, oct 2016, pp. 1–5.

[36] M. Perkusich, K. C. Gorgônio, H. Almeida, and A. Perkusich, "Assisting the continuous improvement of Scrum projects using metrics and Bayesian networks," *Journal of Software: Evolution and Process*, vol. 29, no. 6, sep 2017.

[37] Git Community, "Git Documentation," 2019. [Online]. Available: https://git-scm.com/docs/git