Automating Scrum Processes Through Chatbots

Morad E.
2465064E

07/04/2023

1 Background

Agile software development emerged as a response to the rigid, linear and inflexible waterfall model. A common agile methodology used in industries is scrum, which emphasises an iterative and incremental approach to developing working software while enabling clients to quickly and regularly provide developers with feedback. This approach to development became a standard in many industries due to its ability to satisfy clients through providing higher quality software efficiently, with lower risks, and at a pace fast enough to adapt to shifting market trends (Mishra et al., 2012).

In scrum, a typical team consists of a product owner, scrum master, as well as developers. Scrum team members work together to achieve set goals during short development periods called sprints. Throughout sprints, members attend scheduled scrum sessions which include planning meetings, daily standups, reviews and retrospectives. While scrum can be used to improve a team's efficiency, misunderstanding and misapplication of it can result in the reverse effect.

A complication within modern software development is disparity occurring within teams. Team members are likely to vary in levels of project experience (Serrador et al., 2012): some members of a team might be newly onboarding while others might be more senior and experienced with the project. Likewise, team members might view and understand the project from different perspectives. Thus, as organisations continue to grow and increase in complexity, refining and developing strategies to facilitate optimal team management to mitigate conflicts and blockers becomes increasingly relevant. Otherwise, inefficiency results.

Although the frequency and duration of scrum meetings allows teams more opportunities to communicate, when handled incorrectly, it can result in non-beneficial, excessive and ineffective communication. In a research paper by Akif et. al. 2012, 44% of participants reported a loss of concentration and distraction from their work due to interference by product owners and scrum masters. This disruption alongside the short term deadlines imposed on developers leads to less time for completing assignments. Hence there is an increased pressure to sacrifice code quality in order to deliver on time. In cases where meetings are dragged out unnecessarily due to incorrect management of meetings and poor communication within teams, the issue escalates further (Stray et al., 2013).

Effective scrum management within a team can help to reduce time wasted in meetings and thus reduce pressures on developers to deliver high quality software. This naturally improves the overall satisfaction of project stakeholders. With significant improvements having been made in natural-language processing and big data analytics, automation of some scrum events via the usage of chatbots - computer programs which aim to simulate conversation with human users (Bernardini et al., 2018) - has the potential to enhance team communication and focus by making them serve as asynchronous virtual scrum masters.

2 Problem Description

My industry partner follows the scrum methodology. The main platforms used to communicate and collaborate between my team included Slack for general communication, Microsoft Teams for scheduling meetings, Rally for project management and Azure for hosting the project source code and documentation. Listed below is my team's general meeting schedule:

> Each new sprint would start on a Wednesday and last for a period of two weeks.
>
> Prior to commencing a sprint, every second Tuesday featured an hour long iteration planning session to determine the team's capacity and to discuss what could be committed for the following sprint.
>
> 15 minute stand-up sessions would be held by the scrum master each day so that the team could discuss progress and to bring up potential blockers.
>
> Every second Tuesday and Thursday 30 minute long refinement sessions would be held to review the backlog.
>
> Every second Wednesday a catch-up meeting between developers and QA testers would be held.
>
> Finally, at the end of each sprint, a one hour retrospective would be held for the team to discuss what went well as well as what did not, and to curate a set of action plans to improve the team's overall process.

This arrangement is quite reasonable for a scrum time. However, the problem is that sometimes meetings tended to deviate beyond their scheduled times. In the case of standups, instances were observed where a blocker would be brought up and discussed in too much detail by a member, such that it would lead to the meeting running beyond the expected duration. And in cases where the scrum master intervened on updates which were becoming too complicated, it would happen in a delayed fashion. So ultimately this resulted in some members having to give rushed updates. Meanwhile, this resulted in time being lost for those who had finished providing their updates.

Due to my team consisting of members from different countries (UK and Canada), local hours differed by a span of five hours. In other words, this meant there were three hours available each day where all members were accessible to each other. So when time was lost due to meetings, a symptom of this would be developers working overtime (and in rare cases even until 10pm) to address issues, especially in the case of severe bugs or production releases whose deadlines needed to be promptly met.

Resultingly, some members of the team, particularly experienced developers, would decide to skip meetings and instead provide a text update so as to maintain focus with their work. A partial reason as well for skipping meetings was due to the fact that additional meetings would likely be scheduled anyway. Essentially, while it may sound like a good idea to dedicate more time to addressing an issue, my team wasn't so successful at this because recurring complaints would be brought up and mentioned during retrospectives. Yet, more often than not, it felt like the meetings were affecting overall productivity and performance. The following two screenshots taken from the team's retrospective board illustrate previous comments made by developers.

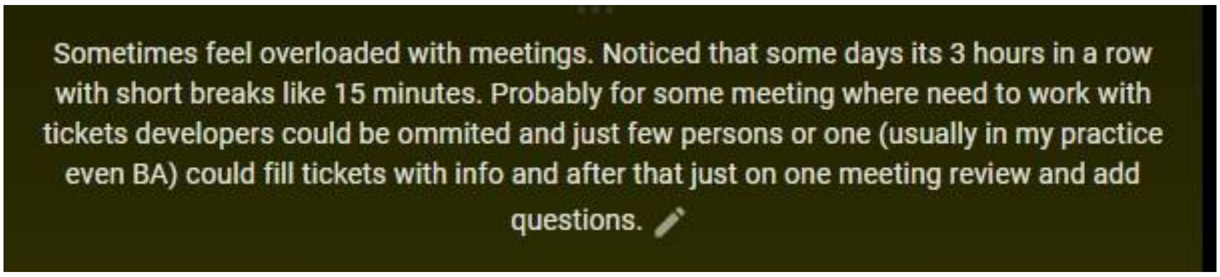
Sometimes feel overloaded with meetings. Noticed that some days its 3 hours in a row with short breaks like 15 minutes. Probably for some meeting where need to work with tickets developers could be ommited and just few persons or one (usually in my practice even BA) could fill tickets with info and after that just on one meeting review and add questions.

Figure 1: Retrospective Screenshot 1 - Demonstrating issue of overload felt by developers.

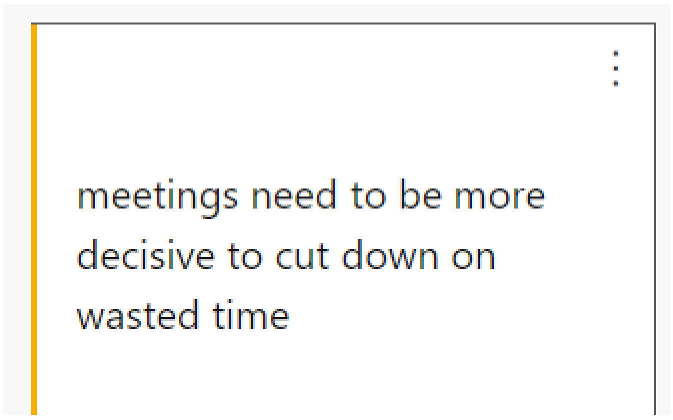
meetings need to be more decisive to cut down on wasted time

Figure 2: Retrospective Screenshot 2 - Mentioning a need to cut down on wasted time.

3 Objectives

The objective for this project is to reduce wasted time resulting from excessively long meetings by building a Microsoft Teams chatbot that can help to automate certain processes and behave as an asynchronous virtual scrum master for the team.

The bot will:

Provide functionality to run asynchronous standup meetings so that team members can provide text or audio updates without live meetings needing to be arranged.

Run scheduled surveys for the team, asking members about their mood and whether or not they feel blocked.

Facilitate asynchronous retrospectives in the form of short, quick surveys, allowing team members to report about what things are currently going well (or the reverse) during sprints, with data gathered to be used later for faciliting more quickly the official retrospective.

Integrate with Rally to allow developers to quickly ask for relevant information about their assigned tasks and request to view what current blockers team members are being faced with so that they can be addressed swiftly.

Provide scheduled reminded on work article due dates and time remaining

3.1 Deliverables

A set of requirements provided by the team

User stories highlighting the team’s needs

Mocks and prototypes of the chatbot

Installation of the Power Virtual Agents app in Teams

An internal project repository housing code, including a web scraper for pulling data from Rally

A deployment pipeline

A finalised version of the chatbot available on Teams

A user manual

Documentation for maintenance and development of the project

An analysed report of results obtained from a user study at the end of the project

4 Work Plan

4.1 Collecting user requirements from the team

User requirements are to be collected from the team, then categorised according to level of priority and importance using the MoSCoW technique.

Definition of Done: A set of requirements are retrieved and distinguished according to degree of project relevance.

Dua Date: Week 2

Asignees: Student, scrum team

4.2 Creating User Stories

Based on the gathered requirements, users stories are to be created and stored in a document.

Definition of Done: A set of user stories are created and made accessible for the team to provide feedback on.

Due Date: Week 4

Assignees, Student, scrum team

4.3 Designing mock-ups and prototypes of the chatbot

Prototypes are to be designed on how the bot should be displayed, when scheduled messages should be set to be delivered, dialog prompts and sensory feedback (eg. sound notification), and the how the bot’s tone of language when communicating should be implemented (eg. calm, serious, joking)

Definition of Done: A set of prototypes are designed and shared as a document for the team to provide feedback on.

Due Date: Week 6

Assignees: Student, scrum team

4.4 Installing the Power Virtual Agent (PVA) app in Teams and creating a project repository

A project repository is to be set up and the tool for create MS Teams chatbots installed.

Definition of Done: When a new repository is created in Azure and PVA is installed on Teams.

Due Date: Week 7

Assignees: Student (might require admin access from the lead developer)

4.5 CI/CD setup

A deployment is to be set up.

Definition of Done: When a deployment pipeline is set up in Azure

Due Date: Week 8

Assignees: Student

4.6 Researching technical documentation

Time is to be set aside to learn necessary languages and technologies involved in the design of the bot as well as prototyping a solution to scrape data from Rally.

Definition of Done: A document containing a table of things to be learned with details on the progress made and outcome.

Due Date: Week 10

Assignees: Student (however might ask for assistance from team lead in certain bits)

4.7 Writing the web scraper logic and designing the chatbot interface

The web scraper script is to completed and the UI for the chat finalised

Definition of Done: A fully functional chatbot on Teams is created and deployed and data is able to retrieved from Rally.

Due Date: Week 16

Assignees, Student

4.8 Quality Assurance

The designed bot is to be tested and checked for any bugs present, as well as feedback given for improvements.

Definition of Done: Identified bugs are addressed and any relevant feedback used to improve the chatbot.

Due Date: Week 18

Assignees: Student, scrum team

4.9 Documentation write-up and User Study Analysis

Several documentations are to be created, including a user manual, technical documentation for maintaining and developing the chatbot further, and an analysed report conducted from a user study

Definition of Done: Complete documentation is provided to understand and maintain the project, as well as analysed results from the introduction of the chatbot

Due Date: Week 22

Assignees: Student, scrum team

5 References

Bernardini, A.A., Sônego, A.A. and Pozzebon, E., 2018, October. Chatbots: An analysis of the state of art of literature. In Anais do I Workshop on Advanced Virtual Environments and Education (pp. 1-6). SBC.

Mishra, D., Mishra, A. and Ostrovska, S., 2012. Impact of physical ambiance on communication, collaboration and coordination in agile software development: An empirical evaluation. Information and software Technology, 54(10), pp.1067-1078.

Serrador, P. and Pinto, J.K., 2015. Does Agile work?—A quantitative analysis of agile project success. International journal of project management, 33(5), pp.1040-1051.

N. B. Moe, T. Dingsøyr and T. Dybå, ”Understanding Self-Organizing Teams in Agile Software Development,” 19th Australian Conference on Software Engineering (aswec 2008), Perth, WA, Australia, 2008, pp. 76-85, doi: 10.1109/ASWEC.2008.4483195.

Akif, R. and Majeed, H., 2012. Issues and challenges in Scrum implementation. International Journal of Scientific & Engineering Research, 3(8), pp.1-4.

Stray, V.G., Lindsjørn, Y. and Sjøberg, D.I., 2013, October. Obstacles to efficient daily meetings in agile development projects: A case study. In 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 95-102). IEEE.