SENG 435 Project Final Report

Alison Goshulak V00806939 | Jake Rothwell V00813277 | Jonathan Grandfield V00823227

Introduction

We begin by describing our vision for our problem domain and target users. In the early stages of our brainstorming sessions, we decided we wanted to create a mediator for increasing collaboration within teams in the form of a Slack bot. There are already several applications that similarly attempt to facilitate cooperative work, however they lack a certain aspect that our project has. Our Slack bot, S.P.A.R.K.Y., deviates from traditional collaborative applications by mimicking a living entity. When formulating our motivation for this project, we wanted to create something that breaks new ground in the CSCW field. S.P.A.R.K.Y. also uses different methods from other CSCW tools we have observed: the bot incorporates pathos (appeal to emotion) and gamification in order to encourage the members of the Slack channel to increase collaborative activities, such as group discussions and document sharing. The way that the bot is implemented, an increase in the overall activity in the Slack channel will translate to an increase in S.P.A.R.K.Y.'s well being. By encouraging the users of the channel to nurture the bot through pathos, S.P.A.R.K.Y. is ideally increasing the level of collaboration in the channel. The target user base for S.P.A.R.K.Y. is small project teams containing approximately five members. With a smaller group, we expected S.P.A.R.K.Y. to be more effective since it would be more noticeable in a channel with fewer members. In the following sections, we will discuss the various aspects of the S.P.A.R.K.Y. project and what we learned during the development of the bot.

Related Work

We have examined various online sources, including blog posts and a textbook on Organizational Behavior, discussing communication. One blog post, "4 ways to navigate different communication styles in your workplace," discussed how to effectively communicate with teammates that have certain communication styles [1]. This is not something we directly addressed in our bot, although we hope that the cooperative features and playfulness of S.P.A.R.K.Y. may help ease tensions caused by differing communication styles. Another post, "8 simple and effective ways to improve team communication," claims that increasing communication not only makes the team more effective, but also helps the members "build trust in each other, [strengthen] bonds and a sense of loyalty to each other" [2]. The textbook, "Organizational Behavior," also argues that poor communication can cost a team time and can mean losing the engagement of members in the project [3].This affirms for us that encouraging communication between collaborators makes them more productive and makes their work more effective.

We also found a Project Team Health Monitor, which is a framework provided by Atlassian to continuously assess the strengths and weaknesses of a team [4]. The Health Monitor involves regularly taking a survey (completed together by the team members) to rate how well desirable attributes are met by the team, such as a balance of skills or a shared understanding of the project goal. The results of the latest survey is also compared to previous results to check the progress on weak areas. Facilitating such surveys (especially if they are focused on attributes related to communication) would be a useful feature for S.P.A.R.K.Y., though this will have to be left for future work since there is not enough time to implement this before the end of the term.

There are several related Slack bots available on the Slack App Directory that we used to compare to S.P.A.R.K.Y. We were interested in the first two bots, Standuply and Standup Alice, to see how much choice the designers gave the users to schedule a time for the stand-up meetings. Both bots allow the user to pick the time for the meetings, but are strict on the meetings occurring daily [5,6]. Currently, our bot posts updates automatically after a certain period of time without any input from the users, though our intention is to allow the update schedule to be more customizable in the future.

The third bot we reviewed,Tettra, is used for building wikis to organize collective team knowledge [7]. This bot initially gave us an idea of how to solve one of our problems: we want our bot to link a question posted by a group member to a given answer. However, this feature turned out to be beyond the scope of our project and has been left for future work. Another bot we looked at, Niles, has also given us ideas for fulfilling a similar purpose. Niles organizes team knowledge to answer commonly asked questions [8]. This could be an even more useful feature for our bot for larger-sized teams than simply linking questions to answers, though this may be even further out of our scope and is still something we will need to investigate later.

The next bot, Leo (Officevibe bot), uses weekly “pulses” to evaluate group members’ engagement (similarly to the Health Monitor mentioned above) and allows members to give anonymous feedback to each other [9]. Using something like the second feature may encourage qualitative work more effectively than simply using a reputation system based on the number of thumbs up or thumbs down a user receives (as our bot currently has). Lastly, we also looked at the Weather Hippie Slack bot. Similarly to S.P.A.R.K.Y., Weather Hippie uses visual cues to relay information in a playful way [10], though to a further extent then our own bot. We may consider increasing the amount of visualization used by our bot, though is not currently a priority.

We have examined a few research papers and other materials on gamification of collaborative work. The first paper, “Gamify Employee Collaboration - A Critical Review of Gamification Elements in Social Software,” provides a critical analysis of popular products that appeared in the authors’ market review. The authors argue that the analyzed products favor quantitative work over qualitative work, which ultimately discourages users by shifting their focus to intangible rewards and achievements instead of the intrinsic value of the work itself [11]. We have attempted to counter this bias towards quantitative work by adding a reputation system to

S.P.A.R.K.Y. Each user has a reputation with the bot based how the other users react to their contributions. The effectiveness of this feature reflected in the evaluation of S.P.A.R.K.Y. is discussed later in the paper.

The second paper, “Using Gamification as a Collaboration Motivator for Software Development Teams: A Preliminary Framework” [12], did not give us many ideas on how to evaluate our bot, but it did introduce us to the Behavior Platform by Badgeville, a gamification platform that appeared several times in the market review of the first paper [13]. The platform includes descriptions of game mechanics that the company considers effective, such as missions and reputation mechanics [13]. The Behavior Platform has proven to be useful in the evaluation stage of our project. Our pretesting survey includes questions on past experiences with gamification employed in a workflow, and the platform has helped us to create common ground with testers experienced with gamification by using familiar terms (such as “badges”, “achievements”, and “narratives”). This will help us judge more clearly the popular or unpopular gamification concepts that we can try to emphasize or avoid in any future versions of S.P.A.R.K.Y.

Finally, we looked at the paper “Designing Cooperative Gamification: Conceptualization and Prototypical Implementation”, which argues that cooperative gamification features are more beneficial for collaborative work than the more commonly used competitive gamification features [14]. The paper draws on the Social Interdependence Theory, which states that individuals are more likely to promote others’ efforts to achieve goals if they perceive that doing so will help them reach their own goals (as in cooperative gamification) [15]. This validates our design of S.P.A.R.K.Y., which progresses faster when all the users are engaged and cooperate with each other.

Approach

We spent a significant amount of time in brainstorming meetings creating a concise list of features that would match with our problem description. During our brainstorming, we took an iterative approach to implementing the project. By creating MVP prototypes in the early stages of the project, we were able to test out subsets of proposed features in order to detect whether certain features would fit as solutions to our problem description. With such an open-ended project choice, there was a great danger of “feature creep.” In the beginning of development, we came up with a large list of features; however we ended up cutting a subset of features due to them not meeting the problem description or due to time constraints. For example, we came up with an idea of linking questions to answers through natural language processing (NLP), but the implementation turned out to be more advanced than we anticipated as well as being out of the scope of our problem statement.

As an alternative approach, we had considered assigning each member in the Slack channel their own personal bot rather than one bot for the entire channel. This would have given the members a more personalized analysis of their activity and collaborative contributions.

However, we decided to have one instance of S.P.A.R.K.Y. for each group so that the bot could act as a kind of heart monitor for the channel, which would give the users an idea of the “health” (i.e. amount of collaboration) of the group as a whole.

The core of S.P.A.R.K.Y.’s feature set is represented in the bot’s set of states. By triggering a certain state through a specific criteria involving channel activity, S.P.A.R.K.Y. will enter that state and change appearance. S.P.A.R.K.Y.’s states and corresponding triggers are described as follows:

- **Emotive** - emojis/reactions detected in channel
- **Busy -** links detected in the channel
- **Silly** - many memes/GIFs detected in channel
- **Inquisitive** - many questions detected in channel
- **Sleepy -** no activity in channel
- **Content -** default state, begins in Content state

S.P.A.R.K.Y. also maintains a value representing its current level. While S.P.A.R.K.Y.’s state essentially describes the *type* of activity happening in the channel, the bot’s level describes the *magnitude* of activity. S.P.A.R.K.Y.’s level increases based on the amount of messages sent in the channel across all users. Our aim for these two main attributes is to capture the relationship between quality versus quantity in measuring a certain metric - where state represents quality and level represents quantity. S.P.A.R.K.Y. levels up from 1 - 3 when there is a high frequency of messages in the channel, and subsequently level down when there is a low frequency. Message frequency is measured by the average number of messages sent over a given time frame. Note that S.P.A.R.K.Y. holds both level and state attributes at any given time.

S.P.A.R.K.Y. also has a feature that is similar to the karma system that Reddit uses. Each user in the channel holds a reputation level - which can be positive or negative - that is measured by the ratio between “thumbs up” and “thumbs down” reactions from other members towards the user’s messages. While also keeping track of a numerical metric of this reputation, S.P.A.R.K.Y. will have a speech style that changes based on your reputation. Therefore, when a user requests details on their personal activity, the tone of S.P.A.R.K.Y.’s response will differ based on the user’s reputation. This reputation feature, like the bot state, is meant to capture the quality of each user’s contributions.

Implementation and Development

In this section we will discuss how we went about implementing the list of features for S.P.A.R.K.Y. We implemented the bot in Python 3, using the Slack Client Library [16] and referring to the Slack API documentation for guidance [17]. The groundwork for nearly all of S.P.A.R.K.Y.’s features resides in a constantly running while loop by utilising Slack’s Real Time Messaging (RTM) API. Every second, new messages are detected in the Slack channel and corresponding information is tracked and passed on to metrics for S.P.A.R.K.Y.’s state, level, as well as user information such as total messages, questions, reactions, and reputation. For instance, when a question is detected in the channel, the question and message count for the user who asked the question is incremented as well as the metric for S.P.A.R.K.Y.'s inquisitive state.

All of our version control and code management during development was done using GitHub. Along with the use of GitHub for code, we also used issues as a ticketing system to help keep track of, categorize, and assign tasks that needed to be completed.

There are two main commands available to users: the *status* and *details* commands. With the *status* command, users are able to check in on S.P.A.R.K.Y.'s current state, level, and appearance. Note that S.P.A.R.K.Y. will automatically output its status to the channel every three hours in order to keep the team updated.

While the *status* command is mainly concerned with S.P.A.R.K.Y.'s information, the *details* command relates to individual user information. When a user prompts S.P.A.R.K.Y. with the details command, the user will receive a private message sent from the bot, which includes all user activity metrics tracked by S.P.A.R.K.Y. These metrics include:

- number of messages sent
- number of media shared (GIFs and other types of files)
- number of links sent
- number of reactions made
- number of questions asked
- reputation score

The details command provides the user with a short report of their activity in the channel, as well as the bot's disposition towards that user encompassed in the reputation feature.

In order to accurately reflect user activity in the channel (for both the *details* command as well as S.P.A.R.K.Y.'s level and state logic) we needed a reliable method for measuring the metrics we were concerned with. Some of the metric tracking was fairly simple to implement by using the Slack API (i.e. total messages, GIFs and other files, reactions) but other portions proved to be more than what could be handled easily with the tools available (i.e. questions asked, links shared), leading us to search for alternatives.

Wit.ai [18] was discovered almost by accident while searching for the solution to another problem but quickly became an attractive option for dealing with our problem of question and link detection. Wit uses natural language processing (NLP) to determine to varying degrees of confidence what the intent of a user's message may be. The intent of "Question" had to be trained into the Wit app manually in order to guarantee more and more reliable detection of questions within a given message. There were some hiccups with Wit struggling to identify the intent of one word messages but overall we were pleased with the result. Luckily for us, Wit comes with a URL detection entity readily available for use which made detection of links shared within a user message a much simpler task than previously expected. After setting up and testing the Wit app online, all that was left to do was implement the message-passing and result-evaluation in our bot's Python code which proved slightly more difficult due to lack of consistent and thorough documentation.

While all the features described make S.P.A.R.K.Y. an effective bot for keeping track of user activity metrics, they do not sufficiently support the element of pathos in the bot. This is what S.P.A.R.K.Y.'s "encourage" notification system aims to achieve. By calculating the users who have the lowest activity in the channel by a metric of lowest message frequency, S.P.A.R.K.Y. will periodically send private messages to users with low activity. This is a crucial feature of the bot, because it provides a true connection between the user and S.P.A.R.K.Y. through private encouraging messages. We hope that this feature gives users who normally communicate infrequently the encouragement they need to contribute to the discussion, as well as an appeal to emotion through fun, personalized messages.

The appeal to emotion was also achieved through careful design and use of the state GIFs. Due to limitations in artistic talent within our group we decided to use very simplistic pixel art created in the online tool Piskel [19]. These images were important as they acted both as a reward for the user to unlock (gamification) as well as a conduit for developing a deeper emotional connection and investment in the well-being of the bot. This meant that the GIFs needed to be interesting and novel enough to drive user motivation to unlock, while also friendly and familiar enough to facilitate an emotional connection.

When it came time to conduct user testing we decided to conduct a pretesting survey, followed by the testing itself, finishing off with a post-testing survey. The questions for the surveys were workshopped in Slack and Google Docs then implemented using SurveyMonkey [20]. We decided due to our target audience of small groups of students working together on a project, the class would be a perfect place to search for testing candidates. Unfortunately, due to the amount of time it took to have a working implementation of the bot we were content to test with, we were only able to secure one group of three users for testing of S.P.A.R.K.Y. Following completion of the pretesting questionnaire, the group was left with S.P.A.R.K.Y. in a private channel for approximately 24 hours, after which the post-testing survey was conducted. Both surveys were completely unsupervised and semi-anonymous, using only the last 4 digits of each tester's UVic ID to allow comparison of the pre- and post-testing results of each individual. It is important to note that we used Heroku [21] in order to deploy our application to a web server so that it would run constantly for our testing group in a private slack channel.

Questions in the pretesting survey were concerned mainly with gauging each user's opinion of gamification and appeal to emotion, as well as their experience with Slack bots and average participation in group conversations. The post-testing survey questions were all geared more towards the testers' opinions and comments from their time spent with S.P.A.R.K.Y. in the channel.

During the development of our project, we evenly distributed the tasks that needed to be done. In Table I below we provide a breakdown of individual contributions to the project.

Table I
Project Task Involvement

Task	Individual Involvement
Conceptualization of bot features and characteristics	All were involved equally
Project milestones (presentations and reports)	All were involved equally
Implementation of features	All were involved equally
Wit application	Jonathan
Research into related work	Alison
Design and Artwork	Jonathan
Evaluation strategy development	All were involved equally
Evaluation Realization (communicating with and supporting testers, creation of surveys)	All were involved equally
Recording and editing of final presentation video	Jake

Findings from Evaluation

Although the extremely small sample size of a single group of three users does warrant need for more user testing in the future, our pre and post-testing surveys yielded some very interesting and helpful results. Due to this small sample size, our results suffer from having a fairly large error if we were to use them to draw very general conclusions, hence why we decided to summarize each tester’s general results and feedback on a more individual level before attempting to draw any conclusions from apparent patterns.

We will be using User A, User B, and User C for this section in lieu of the IDs used during the surveys themselves as these IDs were based off of the individuals UVic student numbers.

One of the questions asked users to give a rating from 1 (low) to 5 (high) for each of several listed features included in S.P.A.R.K.Y.'s design. The results of this question are stored in Table II below for ease of access and understanding, and include the average rating given for each feature.

User A had no prior experience with a Slack bot or any other messaging-based bot before performing the testing with us. They strongly liked games, and had used gamification in the past, finding it added a lot to the workflow at the price of being somewhat distracting. They had also seen appeal to emotions in software before and actually felt positively about these interactions. User A did not self-report any difference in their own activity within conversations in the channel (somewhat active) and felt that S.P.A.R.K.Y. brought a small positive change to the conversations, stating "It encouraged communication in order to get S.P.A.R.K.Y to level up but it ended up with us sending messages just for that reason as opposed to meaningful messages." Overall they said the productivity in the channel remained unchanged and that normally quiet users did not see any significant changes in behaviour. User A said the bot was somewhat intrusive. They thought the user-rating system was fair, noting that "It shows you a numerical value as to how much you are contributing", and found all features to be fairly functional and effective except for dynamic states. Their positive comments were "I think the design of it is great and I like that you can check your own participation in the channel" with the constructive criticism of "I didn't like that it would send updates in the middle of the night.".

User B did have previous experience with bots in Slack or an alternative messaging environment. They also strongly liked games and found that gamification added a moderate amount to their workflow in the past without being distracting at all. They had also used software that implemented pathos and felt neutral about it. Their self-reported participation in conversations went from being somewhat active to very active but unfortunately felt that S.P.A.R.K.Y. brought a large negative change to the conversations, elaborating that "I felt compelled to write more messages for the sake of boosting SPARKY and my own stats. Often, these messages were unnecessary and added noise to the chat". User B felt that productivity overall suffered from the inclusion of S.P.A.R.K.Y., but did find that quiet users had actually become slightly more active in their communication. They thought the bot was very intrusive and that ratings were an unfair way to give or receive feedback with your team. They found most features fairly functional and effective, with status images and messages receiving a slightly lower rating, and user ratings an even lower one. They said: "I enjoyed the fun and quirky style.", but also really wanted it to "be less intrusive".

User C also had prior messaging bot experience. They did not like nor dislike games and found that gamification they used in the past was somewhat distracting and did not add anything to their workflow. After their experience with appeal to their emotions in past software they were left feeling negative about it, selecting the following negative qualities to describe it: manipulative, passive-aggressive, unethical, insincere, and annoying. User C's self-reported participation in channel conversations went from somewhat active to not at all active and they felt that S.P.A.R.K.Y. brought no change to the channel. This was reaffirmed by their belief that the productivity in the channel was unchanged, as was the activity of quiet users. Additionally, they did find S.P.A.R.K.Y. to be very intrusive to the channel activity. User C found that the rating system was an unfair way to give feedback to their team and gave no further comments on the matter. They found all features to be fairly functional and effective, with the exceptions of dynamic states which they really liked, and rating systems and levelling which they found to be less beneficial than the rest. They said S.P.A.R.K.Y. excelled in that “He's very cute!! Personable.” while also noting that “Sparky's gifs are particularly large which offset the chat greatly which I found bothersome, especially when the icons are pixel art and don't have to display so large.”

The user feedback on S.P.A.R.K.Y.’s impact on the communications in the channel seems to be fairly mixed in both magnitude and positivity/negativity, however, the effect on productivity is more apparent. Two of three users found there was no change in productivity while the third found the impact on productivity to be a negative one. Another consistent piece of feedback is S.P.A.R.K.Y.’s overall level of intrusiveness, which is higher than we had hoped for but not completely unexpected. The users cited some of the interactions that contributed to this intrusion as the fact that S.P.A.R.K.Y. remains active through the night, as well as the GIFs being too large and dominating the channel’s screen space unnecessarily. Both of these could be easily fixed and implemented as part of future work on S.P.A.R.K.Y. Another shared sentiment that came up is that the users felt that S.P.A.R.K.Y. was encouraging them to message more frequently, but not necessarily in a meaningful way. This problem is a bit harder to solve, but may be helped with the addition of customizability of the features by a chosen admin in the channel (or potentially every user if desired), as well as some more tweaking to ensure that users are not being rewarded for spamming the system, possibly even finding some way to discourage this type of behaviour without having too much of a negative impact on the detection of very active meaningful communication. All of the users seemed to enjoy the charm and personality of S.P.A.R.K.Y. and it may be beneficial in future work to analyze what specifically is so likeable about the bot in order to take further advantage of it and develop it further.

Table II
Features rated from 1(low) to 5(high) on their general functionality and efficacy from the post-testing survey results

Feature	User A	User B	User C	Average
Personal User Details	4	4	4	4
User Rating System	4	2	3	3
Dynamic States	2	4	5	3.67
Levelling	4	4	3	3.67

System				
Status Messages	4	3	4	3.67
Status Images	4	3	4	3.67

Discussion

We discuss the strengths and weaknesses of our solution, the lessons we learned from building it, and further work we may do with S.P.A.R.K.Y. in the future.

Evaluating our Approach

Our iterative approach to development saved us a lot of potentially wasted time and energy in avoiding features that quickly became apparent to be too ambitious or outside of our scope. It also aided in the generation of new ideas and tweaks to current ones, allowing for a more natural and dynamic evolution of S.P.A.R.K.Y. into the bot and tool we wanted it to be.

The use of GitHub as version control proved invaluable as we were constantly making small tweaks and bug fixes and definitely facilitated the iterative approach mentioned above. GitHub issues as a ticketing system was a new idea to us but ended up working very well, ensuring we each knew what feature of S.P.A.R.K.Y. needed to be completed next, and who would be working on it. The availability of labels for issues was very helpful as it helped us prioritize certain critical issues over others such as “enhancement” features that were more for quality of life than the strict functionality of the bot.

With the features we currently have in place, we unfortunately ended up with a skewed imbalance of quality versus quantity in the channel activity metrics that S.P.A.R.K.Y. gathers. Most of the attributes that S.P.A.R.K.Y. keeps track of are simple, incremental integers - for instance the number of messages or reactions sent. Looking back on S.P.A.R.K.Y.’s implementation, we would have liked to capture more of the quality metrics, as we have an abundance of quantifiable data but not a lot of metrics that measure quality.

One aspect that we thought S.P.A.R.K.Y. was truly effective in was capturing an overview or snapshot of the level of collaboration happening in the channel. The bot ends up being larger than the sum of its parts - in that all the metrics that S.P.A.R.K.Y. gathers come together to create a summary of how much the team members are collaborating with each other.

When working on our bot implementation, we tried to relate to an important CSCW concept that is “space vs place”. Throughout the project’s lifespan, we regularly posed the question: “Does S.P.A.R.K.Y. help transform a Slack channel from a space to a place even more so than it already is?” Being that a space is simply where we put things, and a place is where things actually happen and activities can occur, we wanted to make S.P.A.R.K.Y. able to facilitate the sharing of the information space that is the Slack channel. For instance, by logging links shared, files shared, and user activity, the bot helps keep users accountable within the space, thus transforming it into a place where activities occur versus a plain old repository.

The articulation work involved with setting up S.P.A.R.K.Y. was something that was initially a bit of an issue on our end, but thanks to the Slack API’s facilitation of the creation and use of bot users it didn’t end up being an issue on the user end during testing. Behind the scenes multiple tokens had to be imported each time the bot was started up and this may have been made much easier if we had started out by setting up a simple script to handle it all for us.

We did not run into any issues with users abusing any of the features, but a future concern that we did not consider enough during our implementation may be the co-opting of the user rating system which leaves the potential for a majority of users to discredit and bully a minority, leading to the bot mistreating them for “poor” content. We could avoid this by investing further in S.P.A.R.K.Y.’s NLP to confirm this undesirable content, as well as keep track of which users tend to downvote specific other users.

Bounded transparency is also an issue in the fact that a user’s details are only ever available to themselves, which under certain circumstances may not be desirable. It may have been more appropriate for a user to be able to decide who else can check their details, and potentially decide at the beginning of use of S.P.A.R.K.Y. whether a single administrator should always be able to check (although not without the user being made aware of this fact first).

One of our main pillars of S.P.A.R.K.Y. from the beginning was the incorporation of game-like elements to bring about better collaboration. While the gamification was successful in engaging the users, as we’ve found in some of our research beforehand, it may be too successful in a quantitative measure while being potentially harmful qualitatively. We were hoping that by including states to track the type of interactions occurring in the channel we may be encouraging certain desirable types of communication, however, there are currently no incentives to push users away from “bad” interactions like only sending GIFs - only the silly state to inform them that they are doing so. The rating system is meant to act as a user regulated check of this but does not work if everyone is participating in and upvoting the behaviour. This of course may be acceptable in some contexts, regardless, as a CSCW tool the bot should do more to encourage productive collaborative behaviour (e.g. perhaps too long spent in the silly state would lead to the bot acting out).

The levelling system seemed to be effective but only having three levels available may be too limited in scope to encourage true long-term engagement, as once users achieve the third level and realize that there are no more higher levels, they lose a lot of the incentive that comes from growing the bot as opposed to simply maintaining it. Perhaps even implementation of a high-score or longest-streak feature would help with this issue. The number of states does not seem to be an issue at this point in time, and should continue to be fine unless more useful metrics were to be discovered for incorporation and representation.

The use of Slack and its API were overall a good choice both conceptually for what we wanted S.P.A.R.K.Y. to achieve, as well as developmentally for the suite of tools and options that were made available to us. However, it was a bit of a double-edged sword at some points in that although the Slack API is obviously very powerful, it lacked a reliable source of documentation for many of the features, with some that we considered to be very basic needs being nearly impossible to find (e.g. the sending of events back to the channel). The Wit API suffered from a similar problem of documentation but the online application setup was so seamless and easy to use that it definitely outweighed any of the smaller negatives incurred. The training of the Wit app itself had to be done manually and could be improved much further with S.P.A.R.K.Y.'s detection only getting better and better the more it gets used. More user testing and further adoption by channels would allow us to crowdsource this problem and definitely be beneficial to the overall usefulness and accuracy of Wit's NLP.

Lessons Learned

One lesson we have learned in developing our bot is that fully identifying and articulating the problem it addresses, and the need for a solution, is an essential step before building the solution. Skipping this step could result in building a solution to a problem that does not exist or has already been sufficiently solved. Thankfully, this turned out not to be the case for us since we eventually found sources to back our reason for building the bot. We also have not found any similar applications to ours in the Slack App Directory, which further demonstrates the need for our product.

We intended to have multiple evaluation stages during development, but we instead ended up only evaluating S.P.A.R.K.Y. with one group at the very end of the term. We mainly attribute this to the number of features we implemented in the bot, despite being aware of the risk of feature creep. It is also due in part to our inexperience with the Slack API which was more complex than we anticipated (though the lack of detail in the API documentation is partially responsible for some of our confusion during implementation). In hindsight, we should have restricted the scope of our bot further, at least so a more manageable MVP could have been developed in time to allow for more extensive testing. This is also something to keep in mind for any future work we may consider doing on the bot.

Future Work

Some additions to our bot that we would like to consider in the future and that were inspired by other sources were discussed previously in the Related Work section. In addition, we would like to make the bot more customizable for teams with differing sizes and work styles. In particular, we would like users to have more choice on how often the bot posts automatic updates and what activities are tracked. These choices would be given within some set limits so the bot will still serve its purpose and not be completely ignored. This would also allow for teams to have a bit more freedom in deciding how much communication in a channel is the right amount for them, avoiding any possible “unhealthy” level of communication that may occur.

As mentioned in the Related Work section, we have not managed yet to link questions from users to given answers from other users. Achieving this would allow the bot to more effectively evaluate the quality of communication between team members. Alternatively, we may consider a similar approach to the Slack application Niles to use natural language processing to organize team knowledge so that commonly asked questions can be answered by the bot itself. Both approaches could be very difficult to implement, and the second approach may be redundant since Niles already solves the problem. However, investigating the problem may lead us to a simpler solution or to a more relevant problem that we could address instead.

Some of the tweaks to help the intrusiveness of the bot should be addressed. From our evaluations we learned that it would be better if S.P.A.R.K.Y. only messaged at night while other users in the channel were active, and had slightly smaller state GIFs to avoid taking up so much space in the channel feed. Overall, making the bot less easily “gamed” would be very beneficial to the application of the bot as a lack of truly meaningful communication in the channel is a critical failure of S.P.A.R.K.Y. that does need to be addressed if it is ever to be a viable CSCW tool.

Any further work on S.P.A.R.K.Y. will also involve further evaluations. If we do conduct more user-testing in the future, we would like to do so with more general target users (students working in small group projects). This further testing could also help us determine if there is in fact a “sweet spot” for a general healthy amount of communication to encourage within a channel, which would make for good default settings for the bot if and when customization was implemented. Although our peers involved in our first evaluation did fit this description, being in the CSCW course may have given them certain biases towards our bot and other collaborative tools. Including students outside of this course will therefore give us new perspectives on our solution. For these potential evaluations, we would refer to the paper “Bots as Virtual Confederates: Design and Ethics,” which discusses the ethics of employing bots as virtual confederates in online experiments, and also provides guidelines for using bots in this way [22]. Although S.P.A.R.K.Y. would not act as a virtual confederate in our tests, the paper gives good insights into reducing psychological harm and discomfort to testers by, for example, avoiding unexpected behavior in the bot’s interactions with users within the Slack environment.

References

[1] A. Blanche, “4 ways to navigate different communication styles in your workplace“: Atlassian Blog, May 31, 2017.
[2] M. Le Cren, “8 simple and effective ways to improve team communication”: Azendoo, Apr. 8, 2016.
[3] “8.2 understanding communication,” *Organizational Behavior*: University of Minnesota Libraries Publishing, Jan. 24, 2017.

[4] Atlassian. (2018). *Project Team Health Monitor*. [online] Available at: https://www.atlassian.com/team-playbook/health-monitor/project-teams [Accessed 18 July 2018].
[5] Slack App Directory. (2018). *Standuply*. [online] Available at: https://cscwuvic2018.slack.com/apps/A355V71K7-standupiy [Accessed 22 Jun. 2018].
[6] Slack App Directory. (2018). *Standup Alice*. [online] Available at: https://cscwuvic2018.slack.com/apps/A454FNE64-standup-alice [Accessed 22 Jun. 2018].
[7] Slack App Directory. (2018). *Tettra*. [online] Available at: https://cscwuvic2018.slack.com/apps/A0H4VM9FG-tettra [Accessed 22 Jun. 2018].
[8] Slack App Directory. (2018). *Niles*. [online] Available at: https://cscwuvic2018.slack.com/apps/A32DRG3ED-niles [Accessed 16 July 2018].
[9] Slack App Directory. (2018). *Leo (Officevibe Bot)*. [online] Available at: https://cscwuvic2018.slack.com/apps/A0GU27WR1-leo-officevibe-bot [Accessed 16 July 2018].
[10] Slack App Directory. (2018). *Weather Hippie*. [online] Available at: https://cscwuvic2018.slack.com/apps/A8FH3KKT3-weather-hippie [Accessed 17 July 2018].
[11] C. Meske *et al*, "Gamify Employee Collaboration - A Critical Review of Gamification Elements in Social Software," 2016.
[12] F. Steffens *et al*, "Using Gamification as a Collaboration Motivator for Software Development Teams: A Preliminary Framework," 2015.
[13] Enterprise-gamification.com. (2018). *Badgeville - Enterprise Gamification Wiki*. [online] Available at: http://www.enterprise-gamification.com/mediawiki/index.php?title=Badgeville [Accessed 28 Jun. 2018].
[14] B. Morschheuser *et al*, “designing cooperative gamification: conceptualization and prototypical implementation,” *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, New York, NY: ACM New York, Feb. 25, 2017.
[15] A. Aubrey et al, “social interdependence theory - johnson and johnson,” *Aligning Collaborative Learning Theory with Technology*: Centre for Excellence in Enquiry-Based Learning, Dec. 9 2009.
[16] Python Slack Client Library (2018). *Slack Developer Kit for Python*. [online] Available at: https://github.com/slackapi/python-slackclient [Accessed 20 Jun. 2018].
[17] Slack API (2018). *Slack API Documentation*. [online] Available at: https://api.slack.com/ [Accessed 20 Jun. 2018].
[18] Wit.ai. (2018). *Wit — Docs Home*. [online] Available at: https://wit.ai/docs [Accessed 23 Jun. 2018].
[19] Descottes, J. (2018). *Piskel - Free online sprite editor*. [online] Piskelapp.com. Available at: https://www.piskelapp.com/ [Accessed 6 Jun. 2018].
[20] Surveymonkey.com. (2018). *SurveyMonkey: The World’s Most Popular Free Online Survey Tool*. [online] Available at: https://www.surveymonkey.com/ [Accessed 29 Jul. 2018].
[21] Heroku Documentation (2018). *Documentation - Heroku Dev Center.* [online] Available at: https://devcenter.heroku.com/categories/reference [Accessed 20 July. 2018].
[22] P. Krafft *et al*, "Bots as Virtual Confederates: Design and Ethics," 2016.