TeamWorkFlo: Visualizing team performance and status

ABSTRACT

In software development workplaces it's standard practice for teams to adopt a software engineering process to help organize team efforts. This typically improves efficiency in a development team, but team performance metrics such as balance of effort and workload tend to go unnoticed by project managers who rely on feedback from their team to remain up-to-date. As new tasks emerge project managers must decide who will take them on while trying to avoid overburdening any one member. In this paper we introduce TeamWorkFlo, a novel system for visualizing teamwork metrics including balance of workload and effort, communications, and coordination. TeamWorkFlo uses data dynamically collected from systems typically found in software development environments including ticket systems, chat services and shared document stores. We evaluated the system with 14 computer science students and graduates and found that the system is successful in conveying workload balance but not in illustrating balance of effort, communications nor coordination. TeamWorkFlo showed a propensity to assist task allocation among team members through inference of developer capacities. Additionally, TeamWorkFlo shows utility in identifying potential process disruptions and can help project managers remain informed on progress between planning sessions.

INTRODUCTION

For decades researchers and entrepreneurs have looked for ways to increase team efficiency through the introduction of processes for categorizing and prioritizing tasks [2]. In a working environment, though, project dynamics add constraints on what tasks should be considered as well. Software engineering processes including those with Agile development principles help software developers to organize these tasks such that the team remains coordinated while individuals use the decisions to prioritize tasks agreed upon for the next milestone [1]. These practices strive to balance time spent planning and coordinating against actual development time.

Some environments aren't conducive to such practices, however, such as development centers that manage dynamic project loads. Environments like this balance multiple projects in parallel, often sharing the same finite pool of developers. These environments exist both in academic settings, such as where students enroll in multiple courses with group projects and changing curricula, and in industry such as contracting companies who survive fulfilling multiple projects concurrently. In both of these settings external forces can quickly and dramatically shift priorities and reallocate developers. Due to the dynamic nature, managers and developers can more easily fall out of sync on efforts spanning multiple projects. The usual mediation for this is for managers to be more proactive in checking on and remembering what developers are working on and making adjustments when necessary. This challenge increases with the number of developers that a project manager must handle, and much more so if the team members are shared between multiple project managers for different projects.

When new projects or tasks appear managers must determine who to allocate them to and when they should be completed, requiring considerations for backgrounds, experience, and of course their current workloads. This last point is critical as a preferred candidate may be unable to take on a new task given their current burdens and deadlines, so the manager must either assign it to another or reduce the burden by reallocating work to other developers. To do this effectively, a manager must have a high degree of awareness over all developers' workloads simultaneously. As such these environments require a different, though complementary, toolset that allows project managers to both track ongoing development efforts and better understand their developers' capacities.

In the following sections we present an overview of the research performed in this area and solutions proposed by academia and industry for similar issues. We then introduce our own system to address these needs. We will then focus on the study that we performed to assess our system's capabilities. Finally we will discuss the results of the study and the implications they have for future work.

PRIOR WORK

There are two main areas related to our research. The first is the theoretical framework for taskwork and teamwork as it relates to team projects. These guided our needs and requirements for the project, determining what we would need to capture and how we need to move forward. The latter is the approaches taken by industry for structuring and capturing engineering processes through task management systems. The following sections discuss each further.

Teamwork

Various researchers have investigated the effect of the aspects of teamwork on their overall output and the success of the project the teams are involved in. According to Bowers et al., team dynamics have a high impact on productivity in team-based settings [3]. Edwards et al. identified six dimensions for teamwork viz. coordination, communication, balance of member contributions, mutual support, effort, and cohesion [5]. Brannick et al. explore the evaluation of team performance as psychologists [4].

Marks et al. noted that time factors dramatically affect team functioning [9]. Whitehead argued that collaboration research in software engineering is different from broader collaboration research, which tends to address artifact-neutral coordination technologies and toolkits, due to the focus on model-oriented collaboration embedded within a larger process [13]. Hill et al. found that the chances of experts coming up with the better estimates was influenced by their ability to recall all the facets of historic projects [7].

These areas have been identified by research by not examined and exploited by project managers who are mostly concerned about getting the job done in the short term without having the means for observing and analyzing visually hidden factors which influence long term quality and success of the projects that they manage.

Task Management Systems

When it comes to software team environments there are many software solutions to assist project managers and developers alike in establishing and maintaining software engineering processes [8, 10, 12]. Research has been in progress for over a decade on how to help users organize and prioritize tasks as well, presuming that users work in a static environment where their taskload is relatively stable and consistent from day-to-day. These efforts also tend to focus on individual task prioritization rather than taking into account team dynamics. Lacking, we found, is support for quickly understanding team dynamics such as coordination and balance of workload, especially in dynamic environments where developers work on multiple projects simultaneously and tasks are re-prioritized dynamically to adapt to ever-changing requirements.

SYSTEM DESIGN

To address these needs we developed an application, TeamWorkFlo (https://teamworkflo.herokuapp.com/, see Figure 1). TeamWorkFlo combines both taskwork and teamwork dimensions captured through monitoring services typical in software development environments. Namely, we have integrations with Slack for chat messages, Google Drive for its document revision history, GitHub for commit logs, and a task management system modeled through an online spreadsheet. These sources drive faceted visualizations to assist project managers in maintaining perspective on projects, teams, and tasks. For example, gaps or dips in work output for a team member combined with the lack of communication with the team members who could potentially help him/her would help the project manager discern that mentoring or cooperation is not in place, and which members he/she needs to talk to.

It was important for the system to work as an agent to collect and interpret data from the various sources as opposed to relying on manual entry by an end user. This removes a level of potential data degradation and fulfills the role that the end user would theoretically perform in its place, which is a tedious and time-consuming operation. Since the data is also already accessible in many cases it makes more sense to integrate the system with existing tools rather than to introduce another system to collect the same metrics in a more limited fashion.

For teamwork dimensions we targeted a subset of Edwards et al.'s taxonomy [5], namely effort, balance of member contribution, communication, and coordination. To illustrate these concepts we defined two main kinds of objects in our system: tasks and activities. Our task facets are based on several existing task management systems and include name; description; priority; estimated time; assignee; status; milestone; start date; and completion date, all pulled from our makeshift task management system. The activity facets we capture from Slack, Google Drive, and GitHub simply contain a contributor, timestamp, the source and, in the case of GitHub, a corresponding task ID to contextualize the activity.

The visualizations in our system cover both the temporal aspects [9] wherein a user can see the teamwork factors over a time-scale as well as allowing separation by artifacts [13]. This allows users to search effectively in a layered manner from a time range to a component and even explore further over features or users.

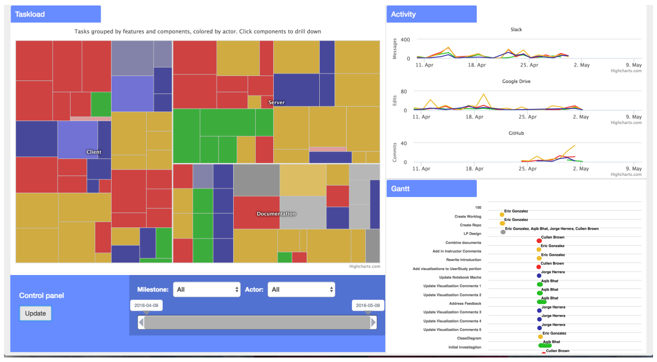

Figure 1. A screenshot of the TeamWorkFlo functional prototype. The system is arranged as three main visualizations – one primary and two accessory views, and a control panel with filters that manipulate all graphs in concert.

This section describes the motivation behind our design including aspects related to high level software architecture, data flow and overall user interaction.

Design overview

Our system design attempts to provide a unified interface for project managers to asses teamwork and effort facets including workload distribution, balance of effort and coordination. While designing our user interface we took the Schneiderman mantra as a guideline for our overall design: "Overview first, zoom and filer and details on demand"[11]. Following an iterative design process backed up with users feedback we got from user studies we ended up defining three visualizations that are provided at all times in our system to convey data related to team facets to the user and the data is filtered manually by interacting with the control panel. Our system gather data about activities from Slack, Google Drive, Github and also gather task information from a formatted spreadsheet created in Google Drive to keep track of our worklog.

Treemap

The treemap visually depicts the user taskload for our team across the project (see Figure 2). Each individual node in the tree is a project task, as defined on a Worklog spreadsheet in our group google drive account. Task node sizes are determined by listed task importance (low priority tasks are small, while high priority tasks are large), and are grouped into three tiers - first by project component (like the server backend, the client frontend, or general documentation), then by feature (like tasks pertaining to the taskload visualization, or tasks involved in implementing a connector for Github activity). Through semantic zooming (or as described on the chart, drilling in), users can zoom in on a component or feature to see more details of the tasks within each grouping. Tasks are also color coded by the team members assigned to them (i.e., red for Cullen, yellow for Eric, green for Aqib, and blue for Jorge, with tasks performed by multiple actors depicted in gray). The saturation of the color in the task describes the current status of the task;tasks not started are low saturation, in progress tasks are medium saturation, and completed tasks are fully saturated. The data for the tree is programmatically pulled from the Worklog spreadsheet by a Google Drive connector. Hovering over the different layers displays popovers providing varying degrees of information about tasks, such as the assignee(s) and description on the task nodes.

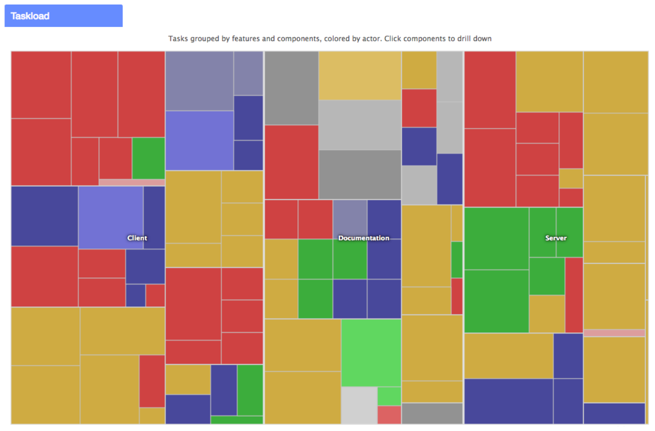

Figure 2. Treemap visualization of user taskload across the project.

Gantt charts

The Gantt chart shows when various tasks were performed over a timeline (see Figure 3). This visualization pulls in data from the Worklog spreadsheet described above to determine when users were working on specific tasks and how often they pushed code to fulfill those tasks. Entries are color-coded to show who was performing each task, using the same color scheme described in the taskload visualization. Users can scroll the window to view taskwork by team member over time, where hovering over individual entries provides additional information for each task similar to the treemap described above.

Line graphs

A set of line graphs designed to show the individual activity of multiple users (see Figure 4). These graphs chart Google Drive document revisions, Slack chat messages, and Github commits over time to show how active users were on given days. Again, user data is color coded and uses the same scheme as in the taskload visualization. Hovering over each data point provides a context showing the day, contributor, and how many contributions they made that day. Each of these visualizations can be filtered over specific actors, project milestones (which deliverable we were working on), and within a range of time. When the filter is updated each visualization responds as they are able (the activity chart does

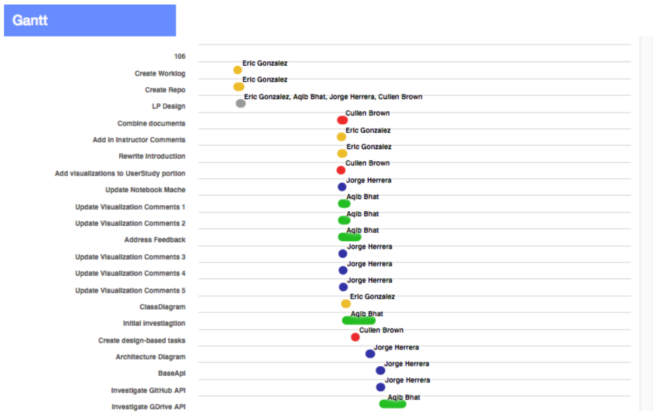

Figure 3. Gantt chart visualization of tasks performed over a timeline.

not display milestone information, and so is unresponsive unlike the other two charts).

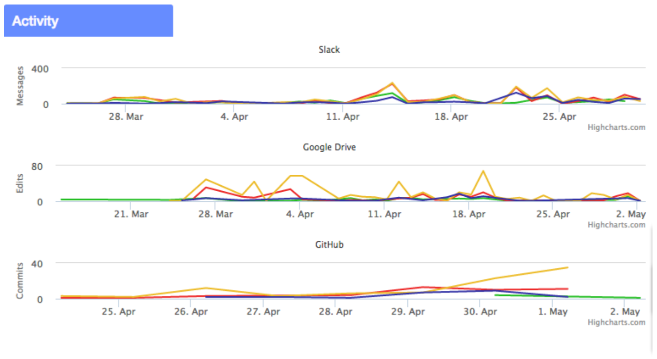

Figure 4. Line charts showing activity on Slack, Google Drive and Github.

Control Panel

We added a control panel section so that the user can interact with the data by applying filters such as: Milestones, Team Member/Actor and Date (see Figure 5). We used a slider bar for the date filter so that the user can drag the handles and select the date window he wants to see. After that the user would have to click the "Update" button to update all three visualizations with only the data that satisfy the selected filters.

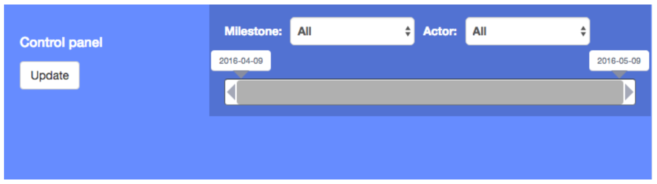

Figure 5. Control panel used to filter and update the data showed on the visualizations.

Software architecture

As you can see in Figure 6 the system follow a Client-Server architecture. The server component is the one in charge of all the processing and API calls. We defined an Activity Interpreter that connects to each one of the services we needed to get information about activities (Slack and Github). The API are represented in the diagram as artifacts and they connect to services outside our server. In the same way we defined a Task Interpreter to connect to the task service we needed (Google Drive). After we get the information we need from those Interpreters we pass that data to their correspondent aggregators. The Activity and Task Aggregators are in charge of formatting the information the way the client needs in order to work. Finally we have a TeamWorkFlo API component who is in charge of collecting all the formatted data from Tasks and Activities and send them to the client in JSON encoding when required. The client side catches the formatted data from the server and updates the tasks and activities visualizations.

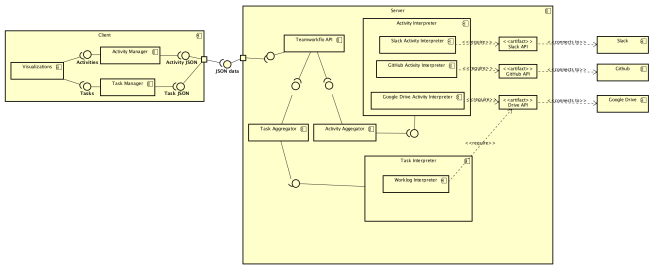

Figure 6. Software Architecture of TeamWorkFlo modeled as an UML component diagram.Each box represent a component. Each component can be represented as a set of other components. The Circles represent interfaces that they provide and the half circles represent that they require something. We use them to model the calls and data flow of our system.

EVALUATION

Our study consisted of four main components; a short pre-study survey, a briefing with training questions, a few open-ended exercises to discover teamwork facets, and a final post-study questionnaire. In the next few sections, we will discuss each component of our study, before moving on to general testing procedures.

Pre-Study Questionnaire

The pre-study survey was designed to collect simple demographic information on users, like user age, sex, experience managing teams, and experience using project management systems (see Figure 7). This data was collected using a Google Forms document linked to a private Google Drive account owned by the authors. The majority of of users were male (12 out of 14) and between 18-25 years old (8 out of 14). The occupation of the users varied from 5 computer science undergraduate students, 3 software engineers, 3 computer science graduate students, 2 professional team leads and 1 participant that did not fall into any such category.

Briefing and Training Exercises

During this phase of the study, users were first instructed on how the system worked. We described to them the structure of each visualization in the system, as well as basic information on how to interpret the data they were seeing and how to manipulate each visualization. The user was then instructed on how to filter visualizations by task actor, by milestone, or by time window.

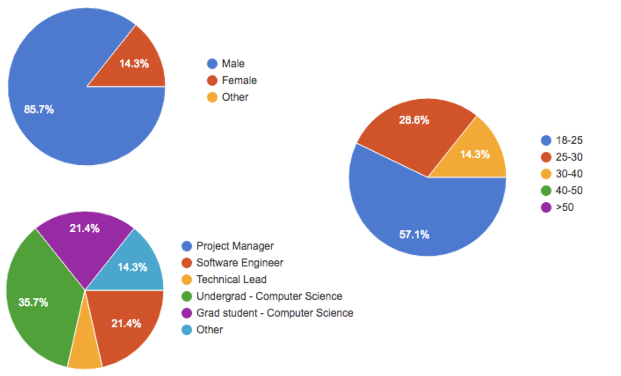

Figure 7. An overview of the demographics we were able to identify from our users.

To verify that the user had learned how to use the system, we then asked the user to answer a number simple questions that would require the use of one or more visualizations and our filters. As an example, one question was "Who completed the most tasks during the Lightweight Prototype milestone?"; for this question, we intended for the user to filter all graphs by the "Lightweight Prototype" milestone, and then use the Taskload Visualization to count up tasks assigned and completed to determine a response (the answer in this case being "Jorge").

There were 5 training questions, with each one asking the user to try a new filter or focus on different visualizations than previously asked. The final question reset the filters to prepare for the next phase of testing. Answers to these questions were again recorded in a Google Forms document linked to a private Google Drive account owned by the authors. User commentary was also recorded in an audio log and stored on this same Google Drive.

Open-Ended Exercises

Having completed the previous set of questions, users were given a number of more open-ended exercises about the activities of the team as a whole. These questions were each framed to utilize a teamwork facet we attempted to capture and to determine a managerial response from the test participant. Each of these exercises, along with their corresponding teamwork facets and the visualizations we expected the user to use to find answers, are listed below.

The first exercise asked the user to determine which of their four team members to assign the task of creating a Slack-based application, based on the data stored in the current visualization. This exercise was intended to determine how a user would allocate tasks based on past taskload balance and on team member expertise; in particular, we expected the user to closely analyze the taskload visualization to determine who had performed the most tasks relating to Slack to make a decision.

The second exercise asked user to assign a leadership role to one of their members while the user would be away for two weeks. This leadership role would require the selected team member to keep everyone on task and stay in communication with other team members. This exercise was intended to focus on team communication, so we expected use of the activity visualization (which includes information on Slack communications) would be used here.

The third exercise asked the user to determine who, if anyone, in the group was contributing less than the rest of the group. From a managerial point of view, we expected this exercise might show potential weaknesses in team work processes and elicit responses on how to correct them. This again involved looking at balance of workload and effort, as well as team coordination. For this task, we expected users to look at all three visualizations.

Lastly, we asked the user to determine how to correct an imbalance of tasks if they saw one. The task was intended to determine if tasks had been well organized with regards to importance; we expected this task would involve the use of the taskload and Gantt visualizations, tapping into the facets of balance of workload and coordination.

Once again, we had users record their responses in a Google Forms document linked to a private Google Drive account owned by the authors. User commentary was also recorded in an audio log and stored on this same Google Drive.

Post-Study Questionnaire

In this section, users were asked to give their thoughts on how the system worked, through both qualitative methods (recorded in an audio log) and through quantitative methods (1-10 Likert scales). Users were asked if any particular component of the system confused them, how well they thought the system informed them about their team's performance, how much they believed the visualizations aided them in completing study tasks, whether or not they noticed the presence or lack of communication and coordination, whether or not they saw an imbalance in team efforts, if there was additional data they wanted to see visualized, and what concerns they would have using the system in the future. Like in previous phases, results were recorded using Google Forms and through audio recordings stored on the aforementioned Google Drive account.

General Testing Procedure

Each study was conducted in-person, except for the study with a participant who was in India. We started off by presenting the consent form to the study participants and getting their signatures (initials in place of signature entered in the Word document format of the consent form, sent over email by the participant in India). After obtaining the consent, we sent the participant a link to our Google Forms document. Upon completion of the initial survey, we began recording an audio log of the user's reactions to our system. We requested the users to think aloud so as to accurately know what there were doing and also thinking while interacting with our system. During the study, members of our group functioned as limited active observers, frequently asking questions about user decisions, and answering questions users had about the system when asked, but otherwise allowing the user to function on their own. We facilitated the users in case they asked for any kind of help and also answered their queries. We took notes of our observations and also recorded the audio from our sessions. After testing, we transcribed the recorded audio logs, and then encoded user comments from the logs and our observational notes using grounded theory (see Figure 8).

Results

Nine of the fourteen participants expressed difficulty in understanding the taskload visualization, with just as many experiencing difficulties with the filtering controls. After the initial tasks working with the tree map, however, they became much more confident and tended to utilize it more than the other graphs to answer questions. We found that users expected the size of tasks to be dependent on either importance or time needed in that task. One participant used both metrics at different times, first perceiving the size as based on importance and then interpreting it as time required when comparing effort.

Issues Noted by Participant

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Totals
Treemap Ambiguity	x	x	x			x		x	x	x		x	x	x	10
Difficulty Filtering by Time	x	x	x	x	x					x	x	x		x	9
Color Ambiguity		x	x			x		x	x	x		x		x	8
System Ease of Use (Issues?)	x			x	x	x		x	x	x					7
Activity Legibility Issues		x		x		x	x			x	x				6
Gantt Ambiguity	x		x			x		x	x	x	x				7
Coordination not visible				x	x	x			x			x			5
Scalability Concerns						x	x			x				x	4
Insufficient Data for Coordination					x	x					x				3
Insufficient Data for Effort							x					x		x	3
Insufficient Data for Communication		x									x				2
Difficulty Filtering by Actor							x							x	2
Difficulty Filtering by Milestone							x			x					2
Privacy or Intimidating Concerns				x								x			2

Figure 8. Issues participants noted while interacting with the system. The highlighted columns represent the participants with 1 year or more of team leadership in a professional setting.

Absent in our system was a legend of any sort to describe the visual elements. This turned out to lead to much confusion for participants who needed to confirm what the different variables meant in each visualization. We found that during the study participants would resort to using the Gantt chart as a makeshift color legend as it displays both the team members color and name next to each other. Color was the most mentioned concern by all participants, with six of the participants explicitly singling it out when asked which portions of the system caused confusion. One participant responded that the system asked too much of color.

Another problem with the use of color was in cases of task overlap; each team member had a specific hue, but where multiple team members were assigned the same task a neutral gray was used. This caused confusion amongst the participants who were unable to distinguish at a glance who actually contributed to any of such tasks. The use of color was also cited as a concern for scalability, where a larger team may exhaust distinctive colors and reduce legibility in the current layout of tasks.

One question asked the participants to identify the least active day for the team. For this question participants mostly used the Activity visualization, but surprisingly they used different metrics to determine team members activity levels. Our initial expectation would be that participants would identify this from a cumulative approach to the three activity metrics captured. Instead, they tended to choose only one aspect of the graph, most using code commits as their metric. Nine of the fourteen used this approach exclusively, while two exclusively used messages sent. Only one participant used all three metrics in answering this question.

We discovered that part of this at least had to do with the general utility of the visualization. Several participants explained that they were unable to get basic information out of the graph without hovering their cursor over each data point. Instead they were hoping that the graph had more room, and that they could get summative information for a selected time-frame that would indicate total contributions. Hovering was also problematic for several users. One participant complained that for the size of the graph the lines are a little too close to tell apart.

DISCUSSION

Based on our results, we found a few common trends worth discussing. Each is mentioned in it's own subsection below.

Treemap Usability

Though there were many shortcomings in the presentation of the taskload visualization, most participants placed higher value on it than the other two visualizations. Those who resorted to other visualizations for task-related investigations did so after discovering that desired information was not readily available through the tree map. In particular, the graph best served inquiry into balance of workload and effort questions. Of the 14 participants, 11 answered such questions using the taskload visualization through comparison of the volume and frequency of the different colored blocks. This plays well on the natural ability for people to recognize spatial discrepancies rather than textual ones [6].

New Understandings

A new understanding that dawned upon us was that people use multiple visualizations to answer the same question, sometimes in tandem. We also found some unexpected use of visualizations by the users such as the use of Gantt chart for determining who completed the most tasks for a milestone, the overall taskload distribution or even the task completion rate.

In some cases, and particularly among those with occupational teamwork leadership roles, participants justified apparent discrepancies in the visualizations reports on user activity. One participant in particular said in response to Aqibs apparent imbalance of tasks, the connector for Google Drive activity, who knows, maybe thats a huge deal. He went on to explain that if he knew the team and was familiar with the tasks and could make a qualitative assessment on their difficulty then he would be able to make a more accurate determination on balance of effort. Several other participants shared the same opinion that to make informed decisions would require more knowledge of the developers and the tasks they were working on. As another participant pointed out, you cannot compare senior with junior developers.

Implications for Design

The various participants made clear where limitations in variable mappings existed. As mentioned above, users tended to assume that the size of tasks in the tree map corresponded to either importance or a component of time or effort. Our representation is limited in efficacy as it only makes use of one dimension in a two-dimension space.

Our use of undocumented color assignments in the three visualizations caused confusion in the participants to the degree that they needed to frequently consult the Gantt chart as a makeshift legend. Legends must not be underestimated in their value to the end user as a map to navigate visual data. Intersecting colors should also be more obvious than our simple desaturation technique, as it loses context of which team members were involved. An alternative representation for contributor altogether or a textured pattern for overlap including each contributor's color would help to clarify these situations.

Another point that surfaced was the difficulty in understanding the activity visualization. The line graphs ran close and thus were hard to differentiate for the users. Though pie-charts or bar graphs could be used to show the relative participation of actors, fluctuations are more easily visualized with line graphs. Two participants wanted more space to compare data series in the charts either through more screen space or through a form of focus+context that could help distinguish regions of the graphs. Both of these same participants additionally requested summary statistics that would provide aggregate reports for the activity metrics, to avoid the need to compare the various data points individually when analyzing each team member's activity performance.

Lastly, a number of users mentioned that they would have preferred automatic redrawing of the visualizations upon changing a filter. In our current implementation, we had a button mapped with the ability to redraw; however, a handful of users kept forgetting that they needed to push the button to get the visualizations to actually change, and became frustrated when they thought they had fully applied a filter and the visualizations stayed the same. One participant in particular was unable to complete one of the orientation tasks due to this oversight, convinced that the data he needed simply did not exist.

CONCLUSION

The results of our study have given us a course of action to further motivate future research. We will be sure to use this information for any further work and studies related to the TeamWorkFlo system. We set out to investigate visualizations that could reflect balance of effort and workload, communication and coordination. In the end our results show that the tree map layout of tasks successfully demonstrated the first two facets in concert if not perfectly. Communication was reported with mixed success, where some participants needed additional contextual information such as who was speaking with whom and topicality in order to make accurate qualitative assessments of the communication ongoing.

Coordination measures did not come through our visualization according to the participants. This is likely partly due to the varying definitions that participants gave for coordination and more from the difficulty in capturing enough context to assess when coordination activities are ongoing. This becomes especially difficult as coordination tends to subsume different parts of the communication and workload facets as team members collaborate with each other or rely on dependent tasks before being able to begin their own tasks.

FUTURE WORK

Management System Plugins

Currently, our system relies on a Google Drive spreadsheet to compile the Taskload visualization; while this was sufficient for our purposes, future work on the system might focus on supporting a number of alternative task input stream (for example, Github's Issue system and Trello's task listings). Adding plugins for popular task management systems should be relatively simple could potentially make it easier for participants to adopt our system.

Additional Information In Visualizations

In at least two of our visualizations, we had users request more (or more direct) information; users wanted the Taskload visualization to display task ID numbers, the Activity visualization to display who communicated with whom at what times and over which tasks, additional git commit information in the Gantt visualization, and many users requested a legend to identify which color represented each team member in our visualizations. This information needs to be provided within our visualizations, with the ability to filter it out if deemed unnecessary.

Customizable Visualizations

One of the issues with our current set of visualizations is that, by their nature, they were designed to attempt to answer questions and identify trends we felt were important to teamwork. As we discovered through user studies, however, a number of participants (in particular those with real management experience) noted they did not have sufficient information to determine teamwork facets. Additionally, a number of users misrepresented the size of tasks within the Taskload treemap to represent the amount of time spent completing the task, when instead it was showing task importance.

We could solve these problems by providing users with customizable visualizations; if we allow the user to determine which information they deem relevant at a given time for a given visualization (as an example, allowing a user to map task sizes in the Taskload treemap by time spent, or by lines of code/text revision, instead of importance) would allow users to determine what they feel is important to make a management decision and act upon it. Other possible customizations might include, but are not limited to, allowing the user to determine whether they wanted to see the average number of words sent over Slack in a day instead of total number of messages, allowing the user to view the average number of lines of code in a day's worth of git commit as opposed to purely the number of commits. We will want to provide these options, and more, to users to better equip them for teamwork measuring tasks.

Visualization Switching

During development, we initially intended to allow the three visualizations to switch cells; that is, we intended to allow the user to select a visualization to focus on, which would swap into the large, leftmost visualization slot. We intended for this switch to allow for more focus on the selected visualization, which would then grow to fill the space provided. Unfortunately, due to time constraints, we were not able to implement this feature; future studies should be performed on the effectiveness of the system with the addition of task switching, versus the lack thereof.

Additional Testing

For this project, we were only able to test 14 individuals, and we only showed participants progress statistics once the project being visualized was basically finished. To get more trustworthy results, our system will need to be tested by more users, and with test data for multiple stages of a given project. A way to potentially address both of these issues would be to find a real software development firm willing to adopt our system for several months, with regular reporting on effectiveness and ease of use from managers and employees alike.

REFERENCES

1. Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., et al. Manifesto for agile software development.

2. Bellotti, V., Dalal, B., Good, N., Flynn, P., Bobrow, D. G., and Ducheneaut, N. What a to-do: Studies of task management towards the design of a personal task list manager. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, ACM (New York, NY, USA, 2004), 735–742.

3. Bowers, C. A., Braun, C. C., and Morgan, B. Team workload: Its meaning and measurement. *Team performance assessment and measurement: Theory, methods, and applications* (1997), 85–108.

4. Brannick, M. T., Salas, E., Prince, C. W., et al. *Team performance assessment and measurement: Theory, methods, and applications*. Psychology Press, 1997.

5. Edwards, B. D., Bell, S. T., Arthur Jr, W., and Decuir, A. D. Relationships between facets of job satisfaction and task and contextual performance. *Applied psychology 57*, 3 (2008), 441–465.

6. Glenberg, A. M., and Langston, W. E. Comprehension of illustrated text: Pictures help to build mental models. *Journal of memory and language 31*, 2 (1992), 129–151.

7. Hill, J., Thomas, L., and Allen, D. Experts' estimates of task durations in software development projects. *International journal of project management 18*, 1 (2000), 13–21.

8. Jira. `https://www.atlassian.com/software/jira`.
9. Marks, M. A., Mathieu, J. E., and Zaccaro, S. J. A temporally based framework and taxonomy of team processes. *Academy of management review 26*, 3 (2001), 356–376.
10. Microsoft Project. `https://products.office.com/en-us/project/project-and-portfolio-management-software/`.
11. Shneiderman, B. The eyes have it: a task by data type taxonomy for information visualizations. In *Visual Languages, 1996. Proceedings., IEEE Symposium on* (Sep 1996), 336–343.
12. Waffle.io. `https://waffle.io/`.
13. Whitehead, J. Collaboration in software engineering: A roadmap. In *2007 Future of Software Engineering*, FOSE '07, IEEE Computer Society (Washington, DC, USA, 2007), 214–225.