Configuring an Appropriate Team Environment to Satisfy Relevant Criteria

Charles Walter, Ian Riley, Rose Gamble
Tandy School of Computer Science
University of Tulsa
Tulsa, OK USA
charlie-walter@utulsa.edu, ian-riley@utulsa.edu, gamble@utulsa.edu

***Abstract*—Accredited computer science programs have a capstone course requiring a significant programming project. Generally, these projects are part of a software engineering class and require a team effort. To facilitate team interaction, instructor monitoring, and objective performance assessment, an appropriate team environment, or set of components and services that can be combined to form an environment, must be chosen, configured, and managed. Since many of the popular team environments have subscription or per-seat licensing, educational institutions could pay significant costs to keep pace with industry. However, instructors should be able to provide a work environment with modern tools to facilitate team interaction and foster productivity while at the same time minimizing cost and reducing their configuration and management effort. In this paper, we survey a wide range of team environments and individual services for team interaction. We compare and contrast them against a set of general, user, and instructor criteria, highlighting features that are relevant to a software engineering team in an educational setting. In addition to the survey, we discuss an open source, experimental platform we are developing to allow instructors to incorporate free services and create a collaborative team environment based on the criteria they want to target.**

Keywords—collaborative environments; software engineering; capstone course; computer science

I. INTRODUCTION

Computer science capstone courses often center on software engineering projects. Academics approach these team-based courses in different ways, yet share similar challenges. The main concerns with teaching such a course using a team-based approach are:

- having non-trivial projects appropriately scoped to skillsets and available resources
- promoting communication and productivity through the use of modern, collaborative tools
- standardizing activities and collaborative tool use to meet class, instructor, and team expectations
- providing transparency so the instructor can intervene when a project/team may be failing/dysfunctional
- providing metrics for team and individual performance

Software engineering classes use student teams to develop a non-trivial project over the course of a semester [1-4]. Some courses allow direct communication with industry mentors. There may be a dedicated lab that teams can access for development and face-to-face meetings. Digital communication, such as email, text, or social media posting, may be encouraged. Most courses use version control systems for code repositories.

Multiple studies focus on team performance as it relates to software development courses. Many of the studies are based on surveys that students are required to complete throughout the project timeline [5-8]. These surveys were given either at the end of each sprint or project milestone or at the end of the course. Depending on the study goals, surveys may include questions focused on the style of team communication and on evaluating members of the team, as well as self-reporting of team, tool, and product satisfaction. More objective criteria for individual and team performance can be established by analyzing event data that is captured by various collaborative tools used by the teams [2, 3, 9, 10].

To facilitate pedagogy, we assume that if establishing a collaborative environment that aligns with class expectations is straightforward, instructors would incorporate it, resulting in more successful projects and increased team satisfaction. However, there are numerous environments available, making a choice difficult. The challenge is to determine what environment is best suited to the class, how can it consistently be used for teams with distinct projects, and what it provides an instructor for performance evaluation.

To improve studies, proper data capture is needed throughout the product development lifecycle. The data should include team interaction and collaborative activities as part of normal team effort. For instance, video recording a meeting and uploading it to YouTube may require more effort than students are willing to provide, resulting on no recordings or potentially no remote meetings. When the same tools are used across semesters the instructor's learning curve is reduced, allowing for precise tool instruction, and thereby increasing student self-efficacy with tool usage. Thus, modern, but well-established tools, are preferable.

In this paper, we survey tools applicable to software development teams in an educational setting. We compare and contrast them against a set of general, user, and instructor criteria. General criteria include if the product is free to educators, wide range of functionality, small learning curve, and communication through social media. User

criteria include collaborative editing, notifications and alerts regarding team activity, integrated version control, and connects to additional (external) applications. Instructor criteria include ease of set up and management, ease of customization, capable of capturing event data for activity tracking, and the persistence of the accumulated data after the semester completes. If individual services are preferred over full project management environments, we show how an open source platform that can be customized.

II. Background

A. Team Communication

Team member communication in a senior computer science project course is a major contributing factor of project success. MacKellar [5] had students fill out weekly surveys on communication methods used and their outcomes. The communication methods consisted primarily of face-to-face meetings, emails, instant messaging, text messaging, and phone calls. Unsuccessful groups used less effective communication methods, such as short phone calls, while teams that employed consistent communication, such as face-to-face meetings, were more successful.

Pastel et al. [1] examined communication among multidisciplinary teams, in which students expressed that communication was easier with students from their discipline, reducing the frequency of intra-disciplinary communication. To increase overall communication, the respective classes were scheduled at the same time giving students dedicated time together. However, the communication patterns were not altered. Since the communication examined was face-to-face, it is inconclusive if online and asynchronous communication methods between disciplines could have provided an alternative style of discussion that crossed disciplinary boundaries.

Using social network analysis on larger distributed teams, Cataldo and Herbsleb [10] found that team members with a higher network centrality contributed more to project development. Marshall and Gamble [8] use social network analysis on small teams to determine if those students with the most influence were also the most productive. While they analyzed all forms of interactive activity, their results showed that influence as seen in forum posting had a positive correlation with productivity and team evaluations pointing to influential members.

B. Team Collaboration

In an undergraduate human-computer interaction course, students worked and learned better in teams of two compared with working alone [11]. Thus, Feng and Luo [11] suggest paired students design improved applications, as each user can express their own design ideas, receive feedback, and improve upon them. This collaborative learning provides better concept understanding, implying that when implementing complex ideas, a team approach is appropriate.

Péraire and Sedano [12] examined the team reflections after completing a Scrum Sprint. These reflections occur in a team meeting at the end of the Sprint to review team progress toward goals, the process to achieve those goals, and what work products, issues, or impediments need addressing. Because not all of their teams were co-located, the reflection process had remote participants. The results indicated the value of collaborating in the reflection process on increased individual satisfaction because it included how the team as a whole was working together and what could be improved.

Evaluating teamwork and individual performance are the topics of multiple studies. Macek and Komárek [3] showed that an objective indicator of productivity appeared in issue trackers and source code managers. They recognized that many instructors rely on subjective estimations of productivity, based on the perceived quality of work products, team reports, and self-evaluation, which may not yield a clear picture of whether the team is working together. They discuss the need for methods to track all project management activities and to have communication logs available to ascertain the level of cooperation between team members. Gamble and Hale [2] relied on an in-house tool to perform this tracking, showing that the number of and size of activities performed during software development, including code commits, forum posting, and document editing have a direct relationship to individual performance as measured against product assessments and team member evaluations.

When teams are distributed, yet need to collaborate, training on collaborative activities may be warranted. Ocker, et al. [13] created a Drupal platform with third-party plugins for communication and content management to track the project progress of multiple distributed teams. Their in-house platform displayed milestones and deliverables, and provided social media, file sharing, and collaborative document editing. They found that when teams that were trained to avoid an "us vs. them" mentality, collaboration improved. Bruegge, et al. [14] examined synchronous and asynchronous collaboration on industrial projects. Participants used an environment with synchronous and asynchronous communication for meetings, along with modules for requirements specification, version control through Git, and issue tracking. They showed that collaboration tools that align with the style of project management can lead to an enjoyable development experience, which translates to as successful product.

Ellis et al. [4] focused on free and open source software projects that are often maintained or worked on by professionals. Their goal was to use this software to train students in determining when code is considered adequate for release. Projects relied on version control to allow users to modify sections of code and request their changes to be merged into the main project. These sections are then reviewed by other users for consistency and correctness before being merged. They found that participation in these collaborative projects had a positive impact on student motivation in computing and perceived learning.

III. Product Survey Criteria

In this section, we discuss the surveyed products for a team-oriented software development class. We define general, user, and instructor criteria, and how the tools satisfy

them. Many of these criteria stem from the use of a tool we developed in house for the software development class [15] and our subsequent studies [2, 8, 9]. With more modern services are being offered, it was necessary to search for the best environment or combination of services that could perform better in the classroom setting and increase opportunities for learning and productivity. We refer to the term *collaborative team environment* as (a) a single, feature-rich product, (b) a composition of products that can be accessed through a single interface, such as a portal, or (c) a set of distinct tools that are required but are not consolidated.

A. General Criteria

In this section we overview the expectations of the general criteria that a collaborative environment should have even if individual tools are used separately within the course.

Free to educators Instructors have to work with what is easily available and, most likely, free. Many collaborative tools provide free access to a certain level or free access to educational institutions, such as Lucidchart for modeling and diagramming. Other tools require monthly subscriptions, which can be dependent on the number of seats or users, which is generally prohibitive for educational use. In our survey, we excluded those that were not free to education.

Wide range of functionality. Functionality breadth focuses on what the tool allows the team to collaborate on. A tool with significant breadth may provide multiple forms of interaction, such as scheduling, planning, chat, version control, and history capture. A tool with minimal functional breadth may be dedicated to one style of collaboration, such as chat, file sharing, or time tracking.

Learnability. Today's digital natives can adapt relatively quickly to a tool with a well-constructed UI and video tutorials. If an online collaborative tool cannot be learned in this manner, it inhibits productivity, by decreasing self-efficacy [7]. In addition, the instructor must also be able to quickly gain expertise using the tool.

Communication through social media. As discussed earlier, communication among members of the same team, between the team and the instructor, and possibly across teams has been shown to directly correlate with product success and positive team satisfaction [1, 5, 10]. Thus, it is essential to have a communication environment that allows for discussion forums that are easy to create, manage, search, and respond.

B. User Criteria

In this section, we target the user's expectations for a tool that is required by the class for team interaction during software development.

Collaborative editing. This capability allows multiple users to edit work products simultaneously. These tools are preferable to the traditional model allowing only one user to edit a document at a given time.

Notifications and alerts regarding team activity. Today's students are accustomed to notifications on their smart devices and wearables. This reliance on alerts should be exploited in the tool to indicate that one or more team members are at work on the project.

Integrated version control. Software development requires a version control system to manage code commits by a collaborating team. It is preferable that access to the version control system is direct, and possibly integrated into the tool, so that the commits are displayed along with their comments.

Connects to additional applications. Similar to the discussion of version control, being able to access other services within a tool encourages fuller collaboration because students can multitask. It also directly relates activities occurring in one medium to those in another medium.

C. Instructor Criteria

An instructor of the CS capstone course in software engineering must wear multiple hats, e.g. lecturer, facilitator, manager, and grader. We outline some criteria that may be important to instructors when acquiring and adopting a tool or set of tools within the course.

Ease of set up and management. Tools should have minimal set up problems and intuitive management. Products may be locally hosted or web hosted, but should provide the instructor with methods to configure teams, milestones, templates for work products, and communication requirements. The ease with which these management functions can be executed directly translates into the time the instructor can spend on instruction and team interaction.

Ease of customization. Tools with significant functional breadth may require the ability to "turn off" certain functions so that teams do not get distracted by developing work products that are not part of the class. Invoicing and timesheets, for example, may be part of a tool, but unnecessary for the classroom experience.

Capable of capturing event data for activity tracking. Instructors may want to intervene on a project going in the wrong direction or mediate dysfunction emerging within a team. Thus, the instructor should have access to meaningful events and activities related to a team. A tool with transparent access to events may be very valuable.

Persistence of accumulated data. To run studies on past classes, instructors may need to keep the course data in its entirety after the conclusion of the course. Being able to maintain all of a semester's data or multiple years of accumulated data can impose additional costs.

IV. Summary of Tool Survey and Feature Comparison

We examined over 60 tools, many of which have a brief reference in [16] for use in project management, though not necessarily software project management. In this section, we overview 26 viable tools that satisfy the cost requirement, i.e. free for use in the classroom, and have persistent data. We segregate the tools into one of three classifications: project management platforms (14), communication platforms (5), and artifact creation platforms (7).

1) Project Management Platforms

We narrowed our consideration of project management platforms to those that contain a mechanism for tracking tasks that need to be completed to finish a project, since this is one of the main reasons to put forth the effort of learning an all-encompassing platform. The products we examine as self-hosted platforms may provide a paid service to host the project in the cloud. All projects surveyed are currently supported and persist accumulated data until manually deleted. Unless otherwise noted, all projects surveyed do not support collaborative editing. A summary of the products appears in Table I.

Taiga.io (*taiga.io*) is specifically designed for Agile software development. It has direct support for user stories, backlogs, tasks, and bug tracking. It has a fairly intuitive user interface, but requires significant understanding of Agile processes. For example, if instructors do not teach assigning points to development tasks, then the product cannot be used. Some users may have issues with navigation initially. Taiga also includes email notifications for discussion directed at a user and when a user is assigned to a task. Taiga integrates with version control services and online platforms using webhooks. It must be locally hosted to use for free, requiring a familiarity with Linux. Taiga can be customized using the templates provided. It includes the ability to generate reports with information from the user stories, tasks, and issues. Only tasks can be modified collaboratively.

Redmine (*redmine.org*) provides Gantt chart creation, issue tracking, activity tracking, wiki, and file uploads. The simple, well-defined user interface is easy to navigate. There is no social media, though plugin capability for one is provided. Redmine includes email notifications for any system event and an event feed. Version control repositories, such as SVN, Mercurial, Git, Bazaar, Darcs, and CVS are supported. Redmine is locally hosted, requiring familiarity with Linux. It allows instructors to fully customize fields for issues, projects, versions, users, and group. Users can also change the look and feel of the system with theme plugins. While there is no automatic report generator, instructors can download tracking data from the local database, though this is difficult and may not provide data in a useful format.

Trac (*trac.edgewall.org*) includes a version control system, bug database, wiki, a roadmap with milestones, issue tracking, and email notifications for system events. Trac's user interface has small and sometimes hard to find navigation links. It does not support any social media, but has a set of plugins that can connect to one. It includes email notifications related to issue tracking. Trac can integrate with existing version control repositories, either locally or externally hosted. Other plugins, such as a blog, file management system, and additional notification systems are supported. Trac requires familiarity with Linux. It tracks all user activities, which can be downloaded via a report generator. Trac allows users to discuss issues, but has no other collaboration options. Our survey indicates it currently supports only one project at a time. Apache Bloodhound (*bloodhound.apache.org*) uses the Trac source code, and subsequently improves upon Trac by allowing multiple projects at one time. It also adds a better and easier to use UI.

Apache Allura (*allura.apache.org*) includes a Wiki, issue tracking, social media, and documentation editing. It has a simple, easy to navigate user interface, with both discussion and chat features. Allura includes email notifications for all social media posts. It includes support for version control with Git, Mercurial, or SVN and can integrate with any platform that uses webhooks. Allura requires knowledge of Linux to host locally. There is support for customizing the user experience through themes. All user interaction events are stored and can be retrieved through the Allura API.

TACTIC (*community.southpawtech.com*) includes tasks with milestones, Gantt chart creation, report generation, file sharing, and version control for files only (not code). It has a cluttered but readable user interface by default, though this can be customized somewhat through theme plugins. It lacks a dedicated social media. Email notifications can be set up through a pipeline editor. This editor allows an instructor to define when notifications are needed, from as fine-grained as each action, to only at project completion. It allows for multiple versions of documents to be uploaded. TACTIC has plugins to connect to a limited number of apps, such as Photoshop, and requires knowledge of Linux to install and manage. It includes report generation to obtain project data, such as the number of times each user logs in, project and user schedules, and task status.

Endeavour (*endeavor-mgmt.sourceforge.net*) is specifically for software project management, supporting use cases, iterations, change requests, issue tracking, task assignment, document management, a wiki, email notifications, SVN integration, and a forum. Endeavour's user interface is a bit cluttered. There are video tutorials to explain how to use it. Email notifications are for task assignments and forum posts, as well as SVN commits. It can connect to external apps for additional version control options. Endeavour generates reports of user activity, such as member assignment and defect reports. Occasionally tasks will take a few seconds before being displayed, potentially causing user confusion.

FusionForge (*fusionforge.org*) includes facilities for document upload and storage, a survey mechanism for user self-reporting, issue tracking, task assignment, and a wiki. The interface is somewhat unintuitive making it difficult to navigate initially. FusionForge provides a forum for team communication, and email notifications for activities such as forum posts, when issues are added and completed, news, commits, and release notes. It supports Git, plugins to connect external apps, and plugins to partially customize the platform. It requires a knowledge of Linux to set up and does not include an automated reporting system, though tracking data, such as forum data, task information, and commits, may be acquired through a dump of the database.

MantisBT (*mantisbt.org*) includes bug tracking, email notifications, version control support, task status, changelogs, a wiki, and plugins for project management. It can be installed on any operating system and provides a simple, intuitive interface. Email notifications can be set up for issue updates, issue resolution, or comments on an issue. It integrates with external repositories, such as Github or

Gitlab. MantisBT allows users to customize issue fields and notifications as needed. Plugins allow reports to be generated detailing the number of issues submitted, resolved issues, and the number of issues resolved by each user. Plugins are also available for detailed customization of the system.

TABLE 1. SUMMARY OF TOOL SURVEY FOR PROJECT MANAGEMENT ENVIRONMENTS AND FEATURE COMPARISON

		QUALITIES											
		Free to educators	Wide range of functionality	Small learning curve	Communication through social media	Collaborative editing	Notifications and alerts	Integrated version control	Connects to additional apps	Ease of set up and management	Ease of customization	Capable of capturing event data	Persistence of accumulated data
TOOLS	*Taiga*	Y	Y	P	P	P	Y	Y	Y	P	P	Y	Y
	Redmine	Y	Y	Y	N	N	Y	Y	Y	P	Y	P	Y
	Trac	Y	Y	P	N	P	Y	Y	Y	P	Y	Y	Y
	Apache Alura	Y	P	Y	Y	N	Y	Y	Y	P	P	P	Y
	Apache Bloodhound	Y	Y	Y	N	P	Y	Y	Y	P	Y	Y	Y
	TACTIC	Y	Y	P	N	N	Y	P	Y	P	Y	Y	Y
	Endeavour	Y	Y	P	Y	N	Y	Y	P	P	N	P	Y
	FusionForge	Y	P	P	Y	N	Y	Y	Y	P	P	Y	Y
	MantisBT	Y	Y	Y	N	N	Y	P	N	P	P	Y	Y
	OpenProject	Y	Y	Y	Y	N	Y	Y	Y	N	P	P	Y
	ProjectQtOr	Y	Y	N	Y	N	Y	P	N	P	P	Y	Y
	Plandora	Y	P	P	N	N	P	N	N	P	N	P	Y
	Collabtive	Y	N	Y	Y	N	Y	N	N	Y	N	Y	Y
	LibrePlan	Y	P	P	N	N	Y	N	Y	P	N	P	Y

OpenProject (*openproject.org*) includes user stories, task assignments with milestones, issue tracking, bug tracking, a wiki, document upload and storage, version control, and social media. It has an easy to learn interface with email notifications that can be set up for any event within the project. It includes plugins to further customize the experience for each project, as well as to connect to outside apps. It does include integrated version control. OpenProject automatically installs itself, but does not work correctly unless given a high-level domain name, and can only be installed on 64-bit systems and servers.

ProjectQtOr (*projectqtor.org*) provides task assignment, a Gantt chart creation, and tracking systems for resources, product quality (workflow, reports, and alerts), tickets, risks, perimeter (meetings, decision information, document management), and commitments (requirements and test cases). It does not have an intuitive user interface, making navigation difficult. It cannot connect to outside applications. It does include a messaging system for team communication, email notifications for system events, integrated version control and report generation as it relates to system events. The risk management section allows ratings of severity, likelihood, criticality, and status, which can then be linked to tasks to help mitigate the risks. ProjectQtOr is recommended to be run on a Linux server, but this is not required.

Plandora (*plandora.org*) includes an Agile board, to-do lists, Gantt charts, SVN, and task lists. The interface is cluttered and unintuitive, making it hard to understand which tasks are most important. Tasks can have a defined priority, project, estimated and actual time tracking, as well as be assigned to individual users. Plandora allows users to respond to survey questions and can send notifications crafted by an admin to users by email. Charts can be generated based on any data collected within the tasks. Plandora can be installed on any operating system. It does not include integrated version control, any form of social media, or the ability to connect to external applications.

Collabtive (*collabtive.o-dyn.de*) includes task assignment, file management, and report generation. The interface is understandable. Collabtive provides social media through messaging with the entire team or select users. Email notifications are available for any change in a task assigned to a user. Tasks can be associated with specific milestones, allowing the system to track progress towards project completion. It includes the ability to pay for plugins to customize the platform, but does not integrate with version control. Collabtive has a fairly simple setup and must be installed on a Linux system.

LibrePlan (*libreplan.com*) includes calendars, task assignment, Gantt chart creation, and report generation. The user interface is simple but difficult to navigate. It connects to other Libre applications, including LibreOffice. LibrePlan includes email notifications for system events, and can generate reports of hours worked and progress of the project. It is focused more toward industry, since it includes expense calculation and resource management. It does not support any form of social media or version control.

2) Communication Platforms

Communication platforms are services designed primarily to facilitate communication among teams, such as Internet Relay Chat (IRC). We surveyed in detail those commonly used services that were stand-alone (i.e., not part of another platform) and free for educators (i.e., usage does not exceed the free allotment of resources). The five most accessible communication platforms surveyed are Slack, Jabber, Reddit, Google Hangouts, and HipChat. We discuss them below. A summary of their comparative features appears in Table II.

Slack (*slack.com*) is free for most users in an educational setting. It only stores the 10,000 most recent messages on free accounts, potentially causing a loss of data if too much communication occurs. It provides mobile device support along with a desktop app with alerts to team postings. To manage conversations, channels can be created on specific topics. Private channels can be used to segregate teams. The drawback to using private channels in this manner is that Slack only allows exporting conversations (to JSON) from public channels. Thus, to use Slack and preserve all of the conversations for later study, each team must have a separate Slack account. Slack also includes the ability to allow other web apps to post to it through the use of webhooks.

Reddit (*reddit.com*) is a popular social networking site, billing itself as "the front page of the internet." It can be locally hosted, allowing a class to use a familiar tool to facilitate communication. An admin can set up separate sub-reddits for each project. Links to outside tools can be posted, and important notifications or links can be included on the side of the main pages of each sub-reddit. One drawback of having it served locally is that the mobile app is only integrated with the official Reddit site. When deployed, students complained that it was not how they used Reddit, which is mainly used for issues and help. So they did not feel comfortable having virtual meetings using the platform.

TABLE II. COMMUNICATION-BASED SERVICES AND THEIR FEATURES

		QUALITIES											
		Free to educators	Wide range of functionality	Small learning curve	Communication through social media	Collaborative editing	Notifications and alerts	Integrated version control	Connects to additional apps	Ease of set up and management	Easy of customization	Capable of capturing event data	Persistence of accumulated data
TOOLS	*Slack*	Y	Y	Y	Y	N	Y	N	Y	Y	P	Y	P
	Reddit	Y	P	P	Y	N	P	N	N	P	P	Y	Y
	Jabber	Y	Y	Y	Y	N	P	N	P	P	P	N	N
	Google Hangouts	Y	Y	Y	Y	N	P	N	N	Y	N	P	Y
	HipChat	Y	Y	Y	Y	N	Y	N	Y	Y	P	Y	Y

Jabber (*jabber.com*) is an XMPP service allowing instant messaging capabilities between users. There are a number of clients that can connect to Jabber servers, and Jabber can use either public servers or a private server, if one is set up. Because Jabber is a text chat system, it has a very small learning curve. It is easy to set up with a public server, but could be difficult to set up a private server for use by only a single class. Jabber, by default, does not include any form of alert, though, because it is a XMPP messaging system, alerts could be added through a third party application.

Google Hangouts (*hangouts.google.com*) allows video recording and chat for prearranged meetings. Thus, it does not provide functionality for asynchronous communication. Using Hangouts is preferable to Skype because there is less overhead to setup and record the meeting, which can then be privately posted to YouTube. Hangouts is best used within other products, such as LucidChart (see Section IV.3) for direct chat about a specific artifact. The chat dialog is not persistent. Monitoring a chat cannot be performed unless the instructor is working on the same artifact.

HipChat (*hipchat.com*) is a free chat service which includes file sharing and integration with external services. It includes an API to retrieve data from the service and, for a fee, can be hosted on a local server. When hosted locally, it provides video calling and screen sharing. It is simple and intuitive to use, and includes free desktop and mobile apps.

3) Artifact Creation Platforms

Artifact creation platforms have the primary purpose of developing a work product or artifact, such as a report, spreadsheet, presentation, model, diagram or code. Seven artifact creation platforms were surveyed, with three that focused on code artifact creation and three that focused creating type of documents. A summary of their comparative features appears in Table III.

Outsystems Community Edition (*outsystems.com*) integrates with Eclipse to allow for code and models creation. It provides rapid iteration with one-click testing and running of code. It has no notification system, is not open source, and is not intuitive to use. It does not include a mobile app or chat.

Fossil (*fossil-scm.org*) and The Bug Genie (*thebuggenie.com*) are both bug trackers which include a wiki. Both must integrate with version control software, are open source, and include a simple, easy to understand user interface. Fossil does not include alerts. Bug Genie has alerts, but it is not as easy to acquire tracking data as it is in Fossil. Both Bug Genie and Fossil lack a mobile app, do not include chat, and must be hosted locally.

TABLE III. ARTIFACT CREATION SERVICES AND THEIR FEATURES

		QUALITIES											
		Free to educators	Wide range of functionality	Small learning curve	Communication through social media	Collaborative editing	Notifications and alerts	Integrated version control	Connects to additional apps	Ease of set up and manage	Ease of customization	Capable of capturing event data	Persistence of accumulated data
TOOLS	*Outsystems platform CE*	Y	N	P	N	N	N	P	N	N	N	P	Y
	Fossil	Y	N	Y	N	N	N	Y	P	Y	N	Y	Y
	The Bug Genie	Y	N	Y	N	N	Y	Y	N	P	N	P	Y
	Google Drive	Y	P	Y	Y	Y	P	P	Y	Y	N	P	Y
	Lucidchart	Y	N	P	Y	Y	P	N	Y	Y	N	N	Y
	draw.io	Y	N	P	N	N	N	N	Y	Y	N	N	Y
	GanttProject	Y	N	P	N	Y	N	N	N	P	N	P	Y

Google Drive (*drive.google.com*) provides multiple document creation options, including text files, spreadsheets, and presentations, all of which can be collaboratively edited. Email alerts can be set up when changes occur. Editing history can be acquired through an API by the document owner. If multiple users are editing a document concurrently, what each contributed cannot be discerned. Google Drive includes a mobile app, embeds chat for users that are collaborating, and is hosted in the cloud. It is free for all Google accounts.

LucidChart (*lucidchart.com*) offers a range of modeling templates, including network components, mind maps, wireframes, and UML. It provides a mobile app and can input artifacts from other products, such as Microsoft Visio. It preserves a history of changes to all artifacts that can be accessed while in the page or obtained through their API by the artifact owner. Once documents are shared, collaborative editing can be performed. LucidChart provides Hangouts as a chat feature associated with each page so that team members can converse directly while altering the artifact.

Much like LucidChart, draw.io (*draw.io*) is a service offering modeling templates, mind maps, wireframes, and UML diagram creation, though it is primarily used for flowchart creation. It can connect with some existing project management platforms. It can save files locally or in Dropbox, Google Drive, or OneDrive. It does not provide a mobile app, though it does provide support for external plugins. It does not allow for collaborative editing.

GanttProject (*ganttproject.biz*) is a desktop application that includes a Gantt chart and report generation detailing tasks. It does not include a social media, version control system, or additional app connections. It is not customizable. It does, however, support collaborative editing of the chart.

V. Platform Experimentation

As more collaborative services become freely available, constructing an open source platform where instructors can incorporate the services of choice may be preferable. Larger environments may have unnecessary functionality that can confuse students or increase the learning curve. From the student perspective, task identification, assignment, and management are important, though chat, collaborative editing, and version control are most critical. From the instructor perspective, ease of deployment, transparency, and performance evaluations are critical.

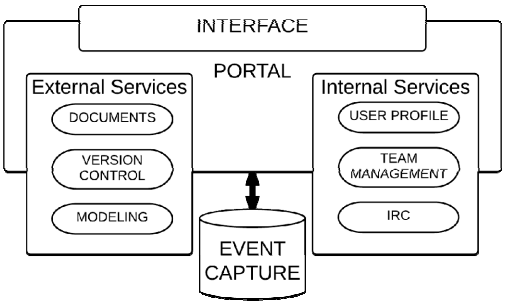

Fig. 1. Platform Architecture

Given our success with moving from a single environment to a service based environment [7-9], [15], we have developed an experimental platform using the Angular 2 web application framework to provide a portal that interacts with Slack.com, Google Drive, LucidChart, and GitLab, to provide functionality for administrative and user activities. The interaction allows the instructor to own and grant permission to shared artifacts, which is necessary to capturing the activity history in Google Drive or LucidChart. The platform is designed to easily define wrappers for service incorporation so that instructors can control the environment and standardize team interaction expectations.

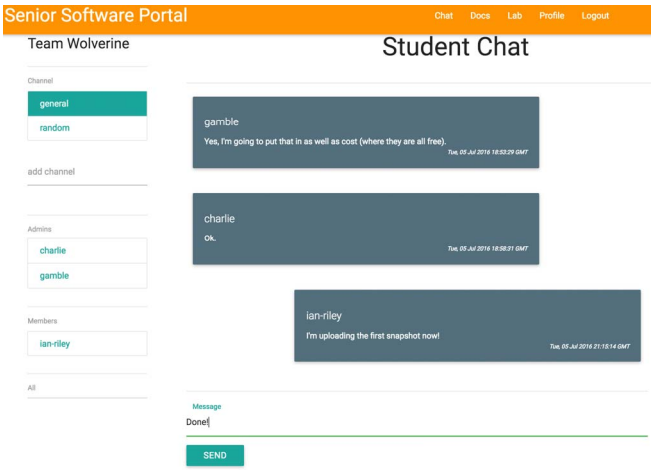

Fig. 2. IRC Interface

The platform architecture appears in Fig. 1. Internal services include team management, user profile creation, and chat using Slack.com (see Fig. 2). User activity is recorded in the Event Capture database. External services are integrated for artifact creation, including Sprint backlogs, modeling, and version control. These services are incorporated using the *injectables pattern* provided by Angular 2. If an instructor wants to provide users with access to shared documents in Google Drive, the instructor would write a software wrapper for Google Drive's RESTful API. Given that the user is already authenticated using the OAuth protocol, the API is implemented by subclassing the abstract class provided in Fig. 3, and then overwriting all of its methods.

```
lib/docs.service.ts
export interface Doc {
 link : string;
 name : string;
}

export abstract class DocsService {
 abstract checkDocs(teamID : string) : Promise<Doc[]>;
 abstract watchDocs(teamID : string) :
Observable<Doc[]>;
}
```

Fig. 3. DocsService Abstract Class

The checkDocs method returns a Promise, which makes a request to fetch all Docs pertaining to the current user. The documents are returned as an array of Doc tuples containing the document name and a URL that is linked inside the page for users to access the document, as displayed in Fig. 4. The Google Drive RESTful API provides all documents in JSON. For services that do not provide JSON, a parser must be included in the wrapper. For example, the LucidChart API provides an XML response which must be parsed separately.

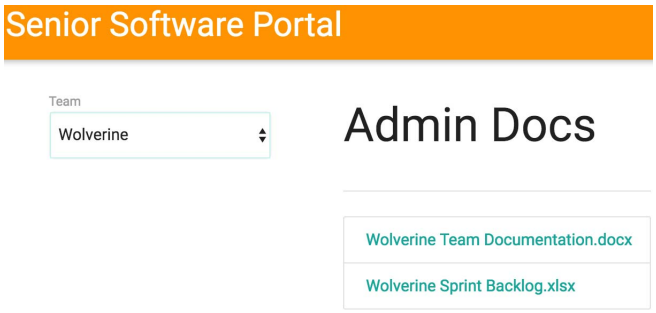

Fig. 4. Integration with Documents

Overriding the watchDocs method in Fig. 3 is similar except where checkDocs uses a Promise, watchDocs uses an Observable, which polls the service for changes. The easiest way to implement an Observable is to establish a timer and call checkDocs at every timer interval. An appropriate interval to check the list of documents could be every 30 seconds, while an appropriate interval for Chat would be about every half second. An example of a completed wrapper for Google Drive can be seen in Fig. 5.

```
lib/docs.service.ts
export class GoogleDriveService extends DocsService {
  checkDocs(teamID : string) : Promise<Doc[]> {
    return new Promise(function(resolve : Function, reject :
Function) {
      var xhr : XMLHttpRequest = new XMLHttpRequest();
      xhr.open('GET',
'https://www.googleapis.com/drive/v2/files');
      xhr.onreadystatechange = function() {
        if (xhr.status == 200) {
          var files : File[]=
JSON.parse(xhr.responseText).items;
          var docs : Doc[] = files.map(function(file : File) {
            return { link : file.selfLink, title : file.title };
          };
          resolve(docs);
        }
      };
      xhr.send();
    };
  }
}
  watchDocs(teamID : string) : Observable<Doc[]> {
    return Observable.create(function(observer :
Observer<Doc[]>) {
      var poll = function() {
        this.checkDocs(teamID).then(function(docs : Doc[]) {
          observer.next(docs);
          setTimeout(poll, 30000);
        };
      };
      setTimeout(poll, 30000);
    };
  }
}
```

Fig. 5. Completed Wrapper for Google Drive

Lastly, the service needs to be provided to the Senior Software Portal application through the main module. This is called an injection and is how the wrapper will be provided throughout the application. An example implementation of injection can be seen in Fig. 6.

Writing the wrapper as described above, we were able to provide a direct connection to our local Gitlab service (Fig. 7). The connection shows a partial listing a team's project files, code commits, pull requests, and open issues.

```
main.ts
bootstrap(AppComponent, [
  ... ,
  GoogleDriveService,
]);
```

Fig. 6. Implementation of an Injection

VI. Discussion and Conclusion

In this paper, we discuss issues that instructors have with tools for CS capstone courses that focus on team-based software development. We outline general, user, and instructor criteria for evaluating tools and survey a wide range of free tools based on the criteria. We overview an approach to create an open source, customizable platform that incorporates desired online services. The goal of the platform is to show that instructors have options when it comes to choosing between a large tool with functions that may not be useful and configuring their own to act as a service portal.

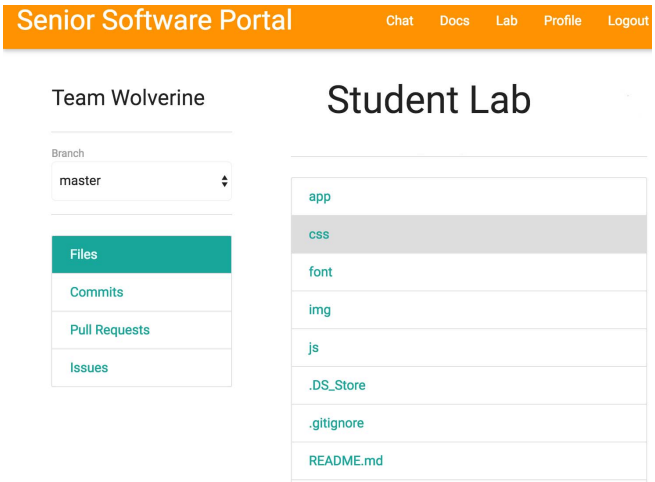

Fig. 7. Integration with GitLab

There are many popular tools targeted toward team projects that are not free to education. PivotalTracker [17] supports Agile development, while Jira [18], Cloudforge [19] and VersionOne [20] support multiple types of software development processes. Other tools targeting project planning are Rally [21] and Teamwork [22]. Jazz [23] also provides tools for project planning and includes source control. Basecamp [24] adds collaborative discussion and activities to centralized scheduling using Microsoft Project.

Other free tools were difficult to install, which could potentially overburden an instructor. Web2Project [25] would not work with our dedicated Apache install, while Project.net Community Edition [26] required Oracle to be installed. EGroupWare Community Edition [27] had serious database issues with its installation and dotProject [28] took an inordinately long time to install.

REFERENCES

[1] R. Pastel, M. Seigel, W. Zhang, and A. Mayer, "Team Building in Multidisciplinary Client-Sponsored Project Courses," *Trans. Comput. Educ.,* vol. 15, pp. 1-23, 2015.

[2] R. Gamble and M. Hale, "Assessing Individual Performance in Agile Undergraduate Software Engineering Teams," in *Frontiers in Education*, Oklahoma City, OK, 2013.

[3] O. Macek and M. Komárek, "Evaluation of Student Teamwork," in *Software Engineering Education and Training (CSEE&T), 2012 IEEE 25th Conference on*, 2012, pp. 130-133.

[4] H. J. C. Ellis, G. W. Hislop, S. Jackson, and L. Postner, "Team Project Experiences in Humanitarian Free and Open Source Software (HFOSS)," *Trans. Comput. Educ.,* vol. 15, pp. 1-23, 2015.

[5] B. MacKellar, "A Case Study of Group Communication Patterns in a Large Project Software Engineering Course," in *2012 IEEE 25th Conference on Software Engineering Education and Training*, 2012, pp. 134-138.

[6] L. P. Robert Jr., "Monitoring and Trust in Virtual Teams," presented at the *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, San Francisco, California, USA, 2016.

[7] K. Wilson and A. Narayan, "Relationships among individual task self-efficacy, self-regulated learning strategy use and academic performance in a computer-supported collaborative learning environment," *Educational Psychology: An International Journal of Experimental Educational Psychology,* vol. 36, pp. 236-253, 2016.

[8] A. Marshall and R. Gamble, "Gauging Influence in Software Development Teams," in *Frontiers in Education*, El Paso, TX, 2015.

[9] M. Hale, R. Gamble, K. Wilson, and A. Narayan, "Collaborative Learning in Software Engineering Teams," in *17th Americas Conference on Information Systems*, 2011.

[10] M. Cataldo and J. D. Herbsleb, "Communication patterns in geographically distributed software development and engineers' contributions to the development effort," presented at the *Proceedings of the 2008 International Workshop on Cooperative and Human Aspects of Software Engineering*, Leipzig, Germany, 2008.

[11] G. Feng and B. Luo, "An Experience of Teaching HCI to Undergraduate Software Engineering Students," in *2012 IEEE 25th Conference on Software Engineering Education and Training*, 2012, pp. 125-129.

[12] C. Péraire and T. Sedano, "Essence reflection meetings: field study," presented at the *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering,* London, England, United Kingdom, 2014.

[13] R. Ocker, M. B. Rosson, D. Kracaw, and S. R. Hiltz, "Training Students to Work Effectively in Partially Distributed Teams," *Trans. Comput. Educ.,* vol. 9, pp. 1-24, 2009.

[14] B. Bruegge, S. Krusche, and L. Alperowitz, "Software Engineering Project Courses with Industrial Clients," *Trans. Comput. Educ.,* vol. 15, pp. 1-31, 2015.

[15] N. Jorgenson, M. Hale, and R. Gamble, "SEREBRO: Facilitating Student Project Team Collaboration," in *33rd International Conference on Software Engineering, Demonstration Track*, 2011.

[16] (4/24/2016). *Comparison of Project Management Software*. Available: https://en.wikipedia.org/wiki/Comparison_of_project_management_software

[17] (5/9/2016). *Pivotal Tracker | Agile Project Management*. Available: https://www.pivotaltracker.com/

[18] (5/9/2016). *JIRA Software - Issue & Project Tracking for Software Teams | Atlassian*. Available: https://www.atlassian.com/software/jira

[19] (5/9/2016). *Free Subversion and Git Hosting | Bug and Issue Tracking | CloudForge*. Available: http://www.cloudforge.com/

[20] (5/9/2016). *Agile Project Management Software for Agile Development | VersionOne*. Available: https://www.versionone.com/

[21] (5/9/2016). *Rally | Agile Development Software for Business Agility*. Available: https://www.rallydev.com/

[22] (5/9/2016). *Teamwork.com - Cloud Productivity Software*. Available: https://www.teamwork.com/

[23] (5/9/2016). *Jazz Community Site*. Available: https://jazz.net/

[24] (5/9/2016). *Basecamp 3: Manage projects, groups, and client work*. Available: https://basecamp.com/

[25] (5/9/2016). *Welcome - web2Project*. Available: http://web2project.net/

[26] (5/9/2015). *Community | Open Source Project Management Software - Project.net*. Available: http://www.project.net/community

[27] (5/9/2016). *community.egroupware.org: Community wiki*. Available: http://community.egroupware.org/

[28] (5/9/2016). *dotproject - Open Source Software :: Open Source Project and Task Management Software*. Available: http://www.dotproject.net/