CASE Tools to Aid Agile Software Development

Project Summaries submitted by student groups in

CS 440 - Introduction to Software Engineering

at the

University of Illinois, Chicago

Spring 2016

John T. Bell, Instructor
Tanim Chatterjee, TA
During the Spring 2016 semester the students enrolled in CS 440, Introduction to Software Engineering, at the University of Illinois Chicago were asked to work together in groups on two different software engineering projects, representing roughly the first and second halves of a complete project. The goal was to give the students experience working in groups on a (relatively) large software engineering project, for which they alone were responsible for the specification, design, implementation, and validation of the project. Splitting the complete software engineering experience into two half-projects allowed students to begin the implementation phase of the coding project early in the semester, thereby allowing time for thorough testing and inspection of significant amounts of real code. As an added benefit, students were faced with the realistic challenge of specifying and designing their new original systems in sufficient detail that future students that they had never met would be able to implement their visions, while simultaneously working to implement the vision created by former students.

For the coding project, their overall task was to implement and thoroughly test (at least a working prototype of) an object-oriented software design that had been previously developed by former students. This work represented the final two stages of a complete software engineering project, (implementation and testing), and gave the students the perspective of completing a project that had been conceived and initially planned by other software engineers. For this work, the students utilized agile development methods, conducting two sprints of development to yield two versions of a working prototype, and one sprint of code testing and inspections.

For the development project, their task was to conceive an original software product, and to develop it through the requirements, system design, and object design phases, using a more traditional waterfall approach, for eventual implementation by other students in future semesters of CS 440. This work represented the initial stages of a complete software engineering project, and gave the students the perspective of developing a complete software specification and design to be eventually implemented by other software engineers.

For Spring 2016, the development project was to develop A CASE (Computer-Aided Software Engineering) tool to aid agile software development. The basic idea was to develop a tool that would somehow help a group of Software Engineers to collaborate and coordinate their efforts, when employing agile methodologies. (Students used one such tool, IceScrum, for several semesters and complained about it, so this was their chance to design something better.)

In developing the initial project descriptions, detailed system requirements, system and object designs and acceptance test plans for these software products, students were asked to not limit themselves to what a small group could accomplish in a few months, but rather to go ahead and plan a project as large as it needed to be to get the job done. Major reports were due every few weeks, and at the end of the semester, students submitted a final overall report of their project results.

Overall the students completed both their projects admirably, producing some very good working prototypes for their coding projects and some very thorough software requirements, designs, and test plans for their development projects. This document includes the two-page summaries of the development projects submitted by each of the groups. Anyone interested in learning more about the systems described here may either contact the course instructor or the students involved directly. In the meantime, I only hope that the learning experience turns out to be at least as valuable as the finished software designs.

John T. Bell
May 2016
Part I: Project Description
The DevTools CASE tool is for all agile developers that want to maintain their proficiency in software development using CASE tools while taking advantage of all the unique features DevTools has. What DevTools has in contrast to other relevant CASE tools are its integrated file host for development teams, UML and diagram creation suite, native apps on mobile and desktop OSs, support from major IDEs and message boards with instant messaging capabilities. These features make it enticing for existing agile developers to utilize DevTools to their own needs while still maintaining the same sprint/release plan as most tools have. The user will be a range of individuals, such as developers, team leaders, clients or students. This is the primary goal of DevTools: to create a universal CASE tool that lends itself to both new and seasoned developers as well as any parties interested in software development. The expected date for release is the first quarter of 2017.

Part II: Requirements
The product uses functional and nonfunctional requirements planned out and implemented by the development team. A large use case diagram is shown in this section to show the reader all of the requirements of features that they can utilize. The functional requirements diagram depicts all of the essential features that the product must contain, so the user can view them and see exactly what options DevTools has to offer. The rest of the report section deals with DevTools nonfunctional requirements.

Part III: Design
DevTools was designed with a few goals in mind: keep the software easy to use for new and experienced users, maintain functionality to compete with other existing tools, and create an all-encompassing tool for the user base. The idea behind an easy-to-use software is to encourage new users to join the DevTools team without a steep learning curve in order to quickly develop them into experienced users. The reasoning behind an all-encompassing tool is to keep users on the DevTools team so they continue to use the product, so that they will have no need to access other tools on the market. Finally, maintaining a competitive product is to always stay relevant in the industry. An agile development tool should constantly stay updated with all of the latest features that a potential competitive product may have in order to stay as a top-tier software option.
Part IV: Test Plans

Testing for DevTools is important to the implementation of functional and nonfunctional requirements, and to the product as a whole. The testing will follow all functional requirements added to the code of the projects. This is done to make sure all prior requirements and code functions correctly with the newly added function. The testing criteria is either 100% correct and functioning as planned with all other code and receives a passing mark or else it fails the test. This strict pass/fail testing was chosen for many reasons but a few of them would be the absolute need for all functional requirements to be able to be utilized no matter what happens in the CASE tool. An extensive list of test cases is provided in the report. The test cases were chosen through a flow-chart approach to testing each part of DevTools. Other than testing each functional requirement as they are added, the testing schedule for the product will be occasional during development as well as testing after release to see any update needs to be made to DevTools.

Part V: Project Issues

Even with careful planning and extensive testing, the product will run into several issues when being developed. The development team details these extensively in the Project Issues section in the product report. The main issues highlighted are the fact that not all hardware may be able to run DevTools the way intended or at least as a smooth experience for the user. Another possible issue is the fact that the main DevTools interface is through its dedicated website, so compatibility of older browser and computers may pose a problem of slowdown or inconsistency compared to better machines and software. Solutions to the problems that may arise using this software are also detailed in the report, such as tutorial videos to ensure users are completely ready and knowledgeable of the tool before use. A final look at the project is given as well. DevTools main focus was being able to implement its unique features first and foremost, then all the basics and necessities of a typical agile CASE tool. If these were goals were not reached, they were not disregarded before reconsideration and may even come up in future updates of DevTools.
Group 2: Software Engineering – fireScrum Report Summary
Dawid J. Zawislak; Oliver San Juan; Jakub Borowski; Prabhunathan Gnanasekaran

Software developers often have to work together on medium-to-large projects in various companies. These projects often contain tens of thousands of lines of code and have complex documentations. For this reason, splitting projects of this caliber is never easy, and different teams tackle this problem in different ways. One particularly popular way of tackling large software projects is through a method known as agile development, where teams accept that requirements may change, even late in development, and take advantage of the changes for the customer's competitive advantage. One particular method of agile development is known as the scrum method, which encourages teams to organize through frequent communication either in-person or online.

The goal of fireScrum is to enable remote collaboration between software developers/software engineers that will allow the efficient completion of medium-to-large projects. fireScrum shall consist of a multitude of product scenarios. These scenarios include setting up a project group, proposing/submitting ideas for the project, taking decisions on submitted ideas, assigning responsibilities, editing user profiles, editing task plans, setting up group chat, uploading code and code snippets for tasks/ideas, creating meeting notifications, managing all user related data, UML diagramming tools, viewing reports, and managing user data.

The data requirements for fireScrum require a remote database and some local hard drive space. fireScrum will require remote access to a database in order to store all user credentials, user information, code, and progress in projects. This database will store all unit tests, tasks, scenarios, sprints, discussions, and so forth, with the client only being an interface for this data. Access should be efficient and reliable, with user login taking no longer than 2-5 seconds on a reliable (and fast) internet connection. The desktop and mobile application of fireScrum will require some hard drive space. This hard drive access is used to save a person’s login and/or password and their interface preferences. The browser version of the client also requires the use of some hard drive space in order to store cookies that mark the user as logged in. Without this cookie, the user would be automatically logged out after a page change.

Deployment on a cloud platform will allow fireScrum to be scalable for a large variety of team sizes, anywhere from a couple people up to millions, depending on the size of the project. fireScrum uses cookies and session variables for maintaining its high speed and reducing the request-response time. Data integrity is maintained through logged modifications and user access control. With privacy and security being a great concern for fireScrum, the application will transmit all confidential data over HTTPS to provide an extra layer of security.

With the designing of fireScrum, it was decided to segregate the application into 4 major groups: Accounts & Security, Projects & Tasks, Chat System, and Source Control & IDEs, each built with a system of classes.
Accounts will be created in order to prevent unauthorized users from accessing data or making changes to what they are not supposed to. Each project can consist of scenarios, which can be based off of user or technical stories. The scenarios can be divided up into sprints that can span a specified timeframe. Tasks are the individual pieces that compose a sprint. Each task can be assigned to a project collaborator and must be completed in order to progress a sprint forward.

The chat system will be based on the Account class and tied into projects or sprints. The purpose of this is to allow developers and collaborators to directly communicate with one another on a specific topic. This should also prevent discussion of other topics outside of a specific chat, keeping people focused on the tasks at hand.

Since fireScrum aims to be user friendly, it will be integrated with a wide variety of third-party software. It allows users to select which form of source control they would like (Git, SVN, etc) and also allows users to select which IDE they would like to integrate with their selection of source control.

fireScrum aims to reach as many people as possible, and as such, it will be available on the web and on mobile platforms’ app stores. The software’s full features will work whenever a user can connect to the internet. If not, users can still access some limited software features. fireScrum will also be available in several languages and ensures that the content, such as symbols, are not offensive to any religious or ethnic group. Part of fireScrum’s attractability is its ease of use. This comes with all of the usability requirements that have been put into place for the development of fireScrum. One of fireScrum’s big focus is to create an interface that is intuitive for all users to pick up and easily use. It will also provide an easy way to customize the application through configuration files, allowing for users to modify and move features where they please.

The user interface that fireScrum shall sport will also be a quintessential part of the entire application, serving only the best of experiences. One of the key aspects that FireScrum will address is the organization of all objects, stringing down scenarios, to sprints, to the individual tasks within the sprints themselves. The user interface will follow trending material designs to provide a more familiar feeling to many of the existing applications users may already use and feel comfortable with, but allowing users to modify that theme at their will, providing for a truly personal experience. This user interface will be consist from the web application, to the mobile application, down to the desktop program that can be installed on all operating systems, providing the most seamless integration and interaction. Through all of the enhancements, fireScrum will be a top contender to many of the existing CASE tools.

**Test Plan:** fireScrum will be tested thoroughly before deployment. All testable Requirements (Functional and Nonfunctional) will be mapped to a corresponding Test case and validated. Unit testing will be done by developer team followed by System Integration Testing and Client Acceptance testing by a dedicated testing team and Client respectively. fireScrum will be tested for Data Integrity, functionalities, User Interface, Performance, Load, Security and Access Control, Configurations.
Project Description Summary
Group #3 – Vinit Kumar, Omaid Khan, Filip Variciuc, Jesus Solorzano

Introduction

Ozil is another product in the long list of “A CASE Tool to Aid Agile Software Development” type of product. It is aimed to be a multi-platform tool with a modern user interface. Users can use this product as a means of networking and organizing efforts towards designing a new piece of software. It is mainly targeted for the audience who want to develop their products/or provide IT services and follow agile methodology for their development procedure.

Requirements

The user can log in to the product as an administrator and have full access to the project or can log in as one of the team member and have limited accessibility to the project, with functions like creating user stories, creating tasks for those stories, assigning/taking up those tasks, updating the task status to different states, logging in defects/assigning the defect to the respective resource, triaging the defect. The user can also log into the product as a test manager and check for the reports such as the “burndown” and “burnup” charts.

On top of the basic functionalities that a CASE tool must have, Ozil brings to the table a self-contained package of services used throughout the software development cycle. This includes, but is not limited to, a text processor, a spreadsheet manager, video conferencing and chatroom capabilities so that the user does not have to use other pieces of software to work on a project. On top of this, Ozil offers support and assistance for the newer, less technologically capable users. Tooltips, video tutorials, and customer support are some of the things that will help all members of the team- be it a developer, a manager, or a client- to be able to use the application without having difficulties.

Testing:

A test will be considered a pass if and only if the actual results of the test match the expected results specified in the associated test case. Some features to be tested include:

1. New user sign up.
2. Returning user login.
3. User creates user story.
4. User creates task for the user stories.
5. User assigns tasks to different user(team members).
6. Users create test case.
7. User execute the testcase and saves the result in the GIT repository.
8. User logs defect for failed test cases and pushes the defect in the GIT repository.
9. User is able to do video chat with the entire team.
10. User saved changes is universal within the team.
11. DownTime access results in appropriate message rather than “error404” message.
12. User is able to install the local binary (installable) as well web version of the product.

**Design and Subsystems:**

There will be several interfaces. Some of them include:

*User* - *Manager, Developer/Tester, Client*

*Sprint* - *Tasks, Test Cases, Defects*

*Report* - *Individual/Team Reports*

*Video and Telecommunications* - *Webcam chat, Meeting - Sprint planning, Review, Retrospect*

*Deployment* - *Installer or Web-Application*

*Support* - *Tooltips, Video Tutorials, Customer support*

There can be three types of *User. Manager, Client* and the *Software Developer/Tester*. They can all communicate in the various activities that Ozil offers via the *Video and Telecommunications* interface. *Users* can implement this interface when using *Reports*, when organizing a *Meeting*, or when requesting *Support*.

Tasks, test cases, and defects will all identify with one particular sprint. All activities within this sprint are logged in the server.

There will also be the stand-alone products, which will be projects/classes of their own. This is so that the users do not have to go out of their way to download/purchase other software outside of the Ozil package to be able to work on a project. These sub-products, including text processors and spreadsheet managers, will interface the same video chat interface. This facilitates in making the communication between all members of the team as contained and efficient as possible.
Coliberate – Product Description Summary

Group 4 - Yusuf Azam - Syed Hussain - Vivek Krishnakumar - Arkadiusz Pamula

Introduction

Coliberate is an Agile Project management tool to help teams collaborate in a smart way. This application has key features that differentiate it from other similar software out in the market. The main goal of Coliberate is to have more services in a unified program. Its main features include, but are not limited to manage multiple projects seamlessly. It’s ease of use for both the team and the stakeholder sets it apart and makes it suitable to be used by both small and large businesses. With great visualization options, boring spreadsheets can be transformed into interesting charts and graphs. Coliberate will be completely cloud based, so it can be used on many platforms, and not limited to Mac or PC. It will even have apps for iOS and Android, so users can work on the go and get instant notifications.

Motivation

Scrum is an iterative and incremental agile software development methodology for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal". However there isn’t enough applications in the market that are compact and easy to use. We wanted to develop a software that would let teams collaborate in a smart way to save both time and money.

Goals

Coliberates aims at liberating users from the hassles of collaborating by doing most of the work for them. From Reporting to Billing, this application is a one stop shop for managing all your projects!

Key Features

- **Agile reporting and metrics**: Easy-to-understand progress reports for stakeholders, quality assurance, and percentage completion.

- **Scrum Roles**: The capability to assign individuals with any on the three main roles of the Scrum Team is an essential part of any agile development software and Coliberate makes it easy to assign roles.

- **Progress Report Chart**: “HomeBurn” shows burn down chart for a completed iteration, showing remaining effort and tasks.
- **Communication**: Coliberate automatically notifies stakeholders on the project’s progress. Teams can also set-up notifications so as to make sure that all team members are aware of the latest updates; every change made by a user is automatically shown to their teammates in the “Notifs” panel. Instant message teammates without having to leave Coliberate. With VOIP integration, record calls and tag them to the project.

- **Project assessment**: Identify and remedy project obstacles, evaluate performance and progress with not just spreadsheets but with data visualization. With emphasis on visualizations, Coliberate makes sure that anyone can easily understand just by looking at it.

- **Work Items**: Create new Tasks, make suggestions, ask questions and lot more under work items, add a customizable work item if necessary.

- **Third Party Integration**: Coliberate will have host of third party integration including GitHub, BitBucket, Google calendar, and spreadsheet importing and exporting.

- **Bug-reporting**: With easy to use and manage ticketing section, teams can raise tickets and report bugs as and when the project progresses.

- **Sprints**: Assign teams to sprints and easily add tasks which users from a team can pick up. Coliberate goes one step ahead with the tasks by giving an option to automatically assign tasks if not picked up by any team member before the sprint starts.

- **Timer**: To keep the team on their toes and remind them how near the deadline is, Coliberate provides a nifty little feature called Timey. This feature lets team keep track of remaining time before the deadline.

- **Project Billing**: Coliberate smartly keeps track of every last minute a person in a team has spent working on the project. With integrated billing feature, teams can save time by not having to log their work hours manually.

**Target Users**

Agile methodology is prominent in small businesses and start-ups as it involves teams of small size. Coliberate is mainly aimed at small to large businesses that involves team small teams. This software is also aimed at companies that would not want to invest money on training their employees on how to use this application since it is easy to use and does not require any installation. Even in the perspective of a stakeholder, this application is simple and visually appealing. Does not provide user with too much information nor does it obscure any information.
Nouveau Scrum: Final Development Overview
Group 5: Emanuil Dobrev, Louis Ludkowski, Vismaya Panchasheelan, Patrick Tam

Nouveau Scrum will be a user-friendly website that enables individuals to function effectively as a team, developing and managing products, by using scrum methodologies of software development. Project development happens in various stages, producing a deliverable product increment at the end of each stage. The website provides means to track, collaborate, share, test, and integrate the development of a project.

Project Description
Nouveau Scrum is a website which will provide a superior Scrum platform that will allow teams to maximize their productivity. This product is targeted to software development teams which wish to utilize the agile programming method.

Nouveau Scrum will provide a login system for users with profiles attached to the users’ logins. Profiles within Nouveau Scrum will contain the name, company name, and contact information for its owner, privacy level, and will also contain the chosen role for the owner. There is the ability in Nouveau Scrum for the user to choose their role in a development team, which is between Manager and Developer, and that is set by the user in their profile.

Along with the profiles are the projects that are created to keep track of the development of a product within the site. Each project has a name, a list of users selected by the Manager(s) of the project, a chatroom, and a git repository associated with it. Manager and Developers within the projects have different permissions, for example, only Managers can edit the Features identified with the project (this will be explained thoroughly next section).

Requirements
Notable Functional Requirements:
- The system must provide the user the ability to choose whether they are a Manager or a Developer.
- The system must provide a Manager the ability to add a feature to the project. Features are what describes the actual parts of a product in development within the site.
- The system must provide the user the ability to create stories. Stories are the highest level method for implementing a feature. Any story created is moved to the Sandbox tab. Managers can accept or reject stories, accepting puts stories in the Product Backlog.
- The system must provide the user ability to create tasks for stories and other tasks within the Project Backlog. This provides a more specific view of what is being created.
- The system must provide the Manager the ability to create sprints for a project. Sprints keep track of time in a timeline and to make sure each feature is completed on time.

Notable Non-Functional Requirements
- PostgreSQL will be used for database storage. Note that an md5 hash will be used on private stored information to improve security.
- The product should be available for 23.9 hours each day in order to best ensure that the service is available during critical times of project management and development.
- Persistent data should never be lost in the event in a shutdown of the product.
• Operations using the database should take a maximum of 10 s. to finish.

Design
The following design goals are attributed to Nouveau Scrum: Nouveau Scrum will accommodate as many users as possible. Whenever Nouveau Scrum runs into an erroneous condition, it will notify the user of the condition and continue the operation as normal. The site should also be intuitive, being easy to use for users. The code should be modifiable, easy to understand, and simple to change the code across platforms. Furthermore, there are four subsystems within Nouveau Scrum: Account Management, Project Management, Git Management, and Chatroom Management.

Test Plans
We plan to test every requirement listed above. While testing, data will be at least collected every day at noon, except weekends. If every day is not viable, every three days will suffice. However, if more than 50% of the tests fail, the testing will be halted until further updating on the system. Once when the patching is approved, testing can resume. Whenever new features are added, additional testing shall take place when they are implemented. Nouveau Scrum will be tested on Windows XP and Mac OS X or higher using the latest versions of Google Chrome, Firefox, Safari, and Internet Explorer.

Project Issues
An open issue associated with the product is the firm determination of what to use for automating the testing of the product. A possible solution would be Selenium, a tool used for automating browsers. Selenium can be found here: http://www.seleniumhq.org/

Another example of software that could prove useful to product development would be KompoZer. KompoZer would be used for building the product UI, as well as quickly creating alterations or new version of an existing UI. KompoZer can be found here: http://www.kompozer.net/

Project Planning
The project shall ideally be divided into four phases of development:
• The structure of the UI should be built first. In this first phase, there is no need to implement scripts for UI components, only the existence and appearance of the UI components themselves need appear. There should be one instance of a UI structure for each notably unique page.
• In the second phase, the server-side implementation is done, along with the implementation of sockets to link the serverside and clientside portions of the product as well as the usage of PostgreSQL operations where necessary.
• In the third phase, the client-side implementation is completed through giving proper functionality to UI components. Scripts will be added where necessary.
• The full database in which the product shall use will be prepared for use in the final phase.
A Brief Description

NiceScrum is a computer-aided software engineering (CASE) tool used to aid agile software development by providing a meaningful way to coordinate the project and monitor progress for a group of people. The software is especially useful for teams which aim at utilizing agile programming methodologies in which requirements and solutions evolve and adapt, promoting continuous and rapid development. The objectives of the product include increasing efficiency as a team, better communication among the team members, providing a direction to the team’s vision and a common platform to increase productivity of the team as a whole. The software will aim to be a one stop solution for software engineers whose aim is to collaborate ideas and develop a product for a client as a team.

Our product has features to make sure the customers feel on track in terms of their progress and also in the right direction. The product aims at also two other important features for a team–making communication effective and have a direction in their progress. The team will then be able to share ideas on what they think will be a good solution. The shared ideas can be looked upon as a team. Tasks can be aligned with code sprints that have set deadlines. Each task can be viewed from different viewpoints like the team member responsible for it, the deadlines or the priority. Our product includes a good chat system where the team can form chat groups within themselves to make sure they stay connected always. The product will also have a quick meeting notification setup. This feature will enable team mates who require a quick team meeting to address any issue or progress with the development to notify the needed colleagues in a short notice.

Requirements

Functional Requirements:

- The application must provide the project manager (user) an option to add/edit members to the team at any point.
- The application must allow the project manager to take decisions on the project implementation plans.
- The application must provide all users the option to view reports as and when needed by the user.
- The application must provide a quick meeting option to allow all users to organize a quick meeting in a short notice.
- The application must provide all the users with an option to draw UML diagrams.
- The application must provide all the users an option to submit their ideas for the project.
- The application must provide all users with an option to view the project plan that is scheduled and followed by the team.
- The application must allow the DBA to manage team members’ data.
- The application must provide all the users with an option to upload their code for the tasks.

Non-Functional Requirements:

Performance: Users should be able to login and see the front page of the CASE tool within 3 seconds of hitting the login button. Any saves or updates made by users should be seen by others within 5 seconds. Transitioning between pages such as the sandbox to the task page
should not take more than 3 seconds. Reports should be created and viewable within 10 seconds of requesting it. Logout and exiting should be secure and be within 3 seconds of hitting logout.

**Accuracy**: Each users should be seeing identical front pages. This means each member should have similar diagrams, charts, logs, etc. when they first log into the CASE tool.

**Dependability**: In case of sudden shutdown, all data that has been changed should be saved and include a previous version for the user to decide to keep or discard on next login.

**Maintainability and supportability**: The original development team must leave behind detailed notes about the product, along with any original source code, notes, bugs, and their associated fixes. All of this documentation must be shared with new employees and that are added to the original development team. They must be trained enough to provide superior support to the end user.

**Security**: The system should maintain end-to-end encryption and must be free of potential exploits. It should have a secure network to reduce the possibility of counterfeit data.

**Usability**: The application will provide the user with a user manual, describing how to use the product and helping the user resolve any issues they may encounter while using the application.

**Look and Feel**: The user can choose between varieties of different color themes. These theme colors will persist the entire time they are using this product until they change it.

**Legal and Standard**: Licenses are owned for all software and tools used to develop the product. The name and logo are trademarks. All information given by users will be protected.

**Test Plans and Testing Schedule**

Testing the accuracy and effectiveness of NICESCRUM during each stage of development will ensure that NICESCRUM will satisfy the functional and nonfunctional requirements in turn becoming a successful fully operational software at launch. The testing process will be systematic and regular which will allow us to find bugs and loopholes in the software as they arise. Not only will the tests check for the functionality of the program, it will also check how different tasks and stories interact with each other and if any optimization is required as each task or story is completed. For each particular test period, there are a selection of test cases that the software is put through some including but not limited to checking if the software works in different web browsers and operating system, checking if the agile development can communicate between each group member instantaneously as well as checking that each group member can log into NICESCRUM.

**Design and Conclusion**

The aim of this project is to develop an interesting and useful CASE tool to aid agile development process. The design started off with system design followed by an analysis of the current software architecture. The user interface design has been proposed by devoting time to user experience. Object designs and class interface guidelines were also mentioned for possible classes.

Thus, NiceScrum is a CASE tool that aims to help software developer working on the agile development process. The development was combined with detailed analysis of the problem statement, requirements, design and the testing criteria. The application we believe is a one stop solution for the whole process. Along with the important features, the concentration on user design and experience will make the application a standout application among the competitors in the market.
The Agile Development Tools Suite is a direct response to the shortcomings of current agile development solutions on the market, particularly IceScrum, the agile development platform used throughout the duration of the CS 440 Software Engineering course coding project assignment. It was through the use of IceScrum, and seeing its shortcomings, that inspiration for the functionality of our product took shape.

Agile Development Tools Suite is designed to take the positive aspects already established in collaborative software development tools currently on the market and accentuate them with our ideas to create a more fully enclosed software solution to address as many of the needs of a development team as possible. Throughout the functional requirements phases, we identified numerous areas of need to be addressed, chief amongst them, the ability to visualize the development process to make the development process, particularly one as active as agile development, more easy to understand. With this in mind we introduce our innovative Sand Castle feature, which takes the features of our implemented Github functionality and expands upon it by relating Githubs key aspects of branches to our presentation method, defined as follows:

**The Sand Castle:** The main container within the Sandbox which holds the currently accepted user stories. User stories may be directly added or merged from a Sand Tower.

**The Sand Tower:** Similar to working branches from Git. They are dynamically created containers which hold suggested user stories.

**Sand Bricks:** Individual user stories and tasks that can be added to the Sand Castle or a Sand Tower.

As a short review of the design patterns that will be used, We determined that the Sand Castle will likely follow a Singleton Design pattern. We only want one instance to be created. This instance can then create multiple instances of Sand Towers. Our Sand Castle will also follow the Memento pattern since we want to be able to redo/undo the addition of user stories.

Our product also introduces the concept of accommodating and encouraging the community feel common to agile development studios, which are usually smaller operations and inclined to be closer knit than larger software development studios. Instead of having a stand-alone product that is a separate entity on its own, we want to integrate features outside of the common development tools available today to bring developers together and create not only a sense of community, but also a knowledge base from which users can collaborate and discuss their work and insights.

We achieve this through a variety of features, which include:

**Instant Messaging and Forums:** Through integration of an instant messaging system, users are able to collaborate and exchange ideas on their project as needed, with the ability to link
pieces of code or sand bricks inside of chat windows for instant access to problems or inquiries from within the team. Forums allow the knowledge base to build over time, and also integrate community managers to help mold the community.

**Web Camera integration**: This will allow daily standups to be facilitated from within the software platform, allowing for a less bulky transition from collaboration to work.

**Integrated LinkedIn profiles**: This feature will allow additional metrics to be gathered from project managers to help select the best people for the task at hand.

Our product will include major improvements over existing Agile development software that can be implemented through several key features. Some of these features include:

**An easy and intuitive GUI**: Existing products, such as iceScrum, are unnecessarily difficult to use and require users to spend a significant amount of time before they can properly use and be productive. We intend to fix that through a better designed and easier to use GUI that users can learn quickly and intuitively.

**Git implementation**: Rather than requiring a third party program to manage a project’s files through a system such as Git, our product will feature Git implementation directly. This enables users to manage a project’s files more easily using a single program.

**Progress and Productivity Measurements**: Measurements of progress include means of measuring progress such as Burndown charts and timelines related to current tasks. Productivity measurements can include things such as keeping track of what each developer is working on and what each developer has contributed to so far.

**Offline Mode**: It is not always possible for users to be online but they may still want to work on and contribute to a project. This is why we have included the ability for the project to be worked on while offline. When the user connects to the internet, the project will be updated to include their changes.

Concerning project issues, we have determined that the implementation of Agile Development Tools Suite will alter the workflow of the development team using it. Proper time must be allotted for the developers to adapt to the new workflow. Other products exist that could be potential solutions (such as IceScrum) but we feel that Agile Development Tools Suite expands upon our competitors and features such as the Sand Castle help to set our product apart from the rest.

In conclusion, we see our Agile Development Tools Suite as both a necessary product due to the problems and faults within existing Agile development programs and also as a significant improvement upon these programs. We think our product will not only meet the demands of Agile software developers and project managers but also exceed them and provide them with a valuable tool and resource that will be used for many years.
‘Cauldron’ Project Final Summary
Oliver Panasewicz, Joe Kallarackel, Mac Carter, Yancarlo Maldonado

There are many agile-development tools on the market right now, but many of them are clunky and don’t have many features. In particular, iceScrum offers a scrumming environment for agile development, but it’s lack of documentation and clunky design leave a lot to be desired. Our solution to this problem is a scrum based agile development tool titled Cauldron.

With Cauldron, we aspire to create an environment where developers can seamlessly collaborate on software development projects. Our project aims to provide users with easy communication and an intuitive interface that generates UML diagrams on the fly from code. And allows users to generate UML diagrams that create code. Cauldron will provide all the features of a standard agile development application along with other desirable features such as messaging and repository integration.

Cauldron is composed from three main parts called the Den, the Rack, and the Cauldron itself. The Den is where the user stores all code and header files. This can be done by uploading, or by linking your favorite repository. The Rack is where users collaborating on a project specify the requirements for the project. We believe that best programming takes place when all developers have a clear understanding of project requirements at all times. With the requirements stored within the application, the user can pull up this information at any time to help stay on track. The Cauldron is the high-level view that the user gets of their project. Inside the Cauldron, users can view project progress, as well as users diagrams.

The Den exits to store and maintain the integrity of code, it is to be integrated with a 3rd party versioning system such as git. The user can literally go and view files in the ‘den’ folder with a file browser inside the Caldron project folder.

The requirement Rack is intuitive for adding requirements to the project. The idea is to have the Rack insert comments into the right file at the head of the document as comments. Once a developer has completed the task, they can simply adjust the comment in code. Once they have saved their code and look in Cauldron, the changes have already been picked up on by the program and it shows the requirement as done. That being said there is also a backup of every requirement just incase a user deletes a requirement inside the code instead of inside the Cauldron program.

Cauldron’s most innovative feature is its ability to construct UML diagrams for the user on the fly. This is done by appending descriptive code stubs to project header files once they are included in the done. These code stubs contain information on requirements as well as class relationships in order to accurately draw UML diagrams and display project progress information. This allows anyone familiar with UML design
capable of laying out the foundation code for their program. Not only that but Member function subs and member variables with access control can be added. Functionality for creating design patterns automatically is a design goal that is a non-functional requirement. But this can be made as an addon to Cauldron, inspiring more users and Software architects to adopt this software. The ability to import existing projects into the den and see their structure is a major advantage of Cauldron.

The Cauldron has many requirements of its own. Without going into too much detail, we require that Cauldron is friendly with all standard storage system formats. This is so that our Den can operate as designed. We also require that our application seamlessly interfaces with mainstream repository services, particularly git. In addition, we require that Cauldron can successfully append information stubs to code, as well as accurately parse the stubs and interpret them to generate UML diagrams that accurately represent the user’s project. Since this is the main identifying feature of our application, this is arguably our most important requirement that must be fulfilled.

In order to ensure fulfillment of all requirements, it is important that we thoughtfully designate a testing process for all requirements and follow through on them. For the needs of testing for Cauldron, we will be using a standard unit-testing platform such as gTest or JUnit for the back end, while the front end will be testing using Selenium. Since our application while primarily be running in-browser, we determined that Selenium would be the best option for testing our application’s front-end as Selenium allows from browser manipulation and element targeting while also being compatible with standard unit-testing assertions.

To ensure that our back end is properly tested, we will be a test-driven development strategy so that we test as we go, as well as conducting unit and functional tests at the end of our development process. The unit tests will be on a standard platform, and as such, will follow standard unit-testing practice, which involves instantiating the environment, running a function, and asserting its output. Test driven development should limit the amount of bugs present at the end of the application’s development

The persistence of data management is something to be thought of in a new way, with users able to access the den directly and change requirements in code and the program, a reasonably structured paradigm for simultaneous, if possible we seek to come up with a new way of keeping data persistent in both the running program and the code in the den in which the changes are saved.

The project is meant to take place over a year, and the budget was $2,012,000. Figuring 4 software architects and 12 software engineers and four months of marketing.

Fun Fact:
The naming scheme of this project was inspired by the Lion(Den) the Witch(Cauldron) and the Wardrobe(Requirement Rack).
Cloudgile - Final Summary

Group 9 - Kevin Nguyen, Tianniu Lei, Erik Reyes, Naga Jyothi Sai Pavani Velagapudi

Project Overview

Currently there are many commercial agile tools that are available both for free and paid to use. However the current tools available have limited functionalities and have a steep learning curve. To resolve those issues, we propose Cloudgile.

Cloudgile is a cloud-based platform where developers, managers, and other stakeholders can easily collaborate on projects in an agile manner. Unlike our competitors, Cloudgile offers many features that are not found in today’s commercial products while maintaining its ease of usability. Our mission is to remove the frustration of current tools as well as improving productivity. We believe that agile should be agile with no strings attached. Features that we are building into our platform is virtual IDE, Git integration, chat system, machine learning, and a built in search engine. Although we are adding more features into the tool, we are keeping the user interface design as simple as possible.

Requirements

For some of our functional requirements, our platform must be able to perform all the features mentioned in the previous section.

For non-functional requirements, since our platform is running from a web browser, our users do not have to save our application onto their hard drive disk. However, we require our users to have a stable internet connection. The platform’s response time between the user and the system should be two seconds or less. Not only should our application be swift, but it should be able to handle one million users simultaneously. Our website shall have an uptime of 99.5% and an uptime of 99.9% on our servers. The only time the server/website will be offline is when we are performing daily maintenance, server upgrades, or website updates. For security requirements, we provide administrative powers to project managers only because we want to prevent abusing features from users who are not part of the project. We want Cloudgile to be picked up as easy as possible, therefore we are using symbols, words, and other icons that is easily recognizable by the user community. Not only is the user interface simple, but we are also offering colorblind mode for users who are partially sighted.
Design

The tool was designed with a client-server pattern in mind. The client will ask the server to do something and the server responds. While behind the curtain we will use a MVC pattern to handle most of the user’s commands. We will also incorporate event driven styles in our code base. This is simply because it is the easiest way to handle all of the user’s movements across our platform. The entire system is broken into their respective MVC fields. Most of the features are implemented as the controller of the system. This is so everything that changes the website is grouped together. The model will be handled by the server as the one that handles the user’s commands. The controller is none other than the user manipulating the website.

Test Plans

Cloudgile’s testing phase is scheduled to be weekly. If at any point of our features that we are testing failed more than 60% of the time, we will stop testing, update, and patch the application. The software and hardware requirements for testing is the device must have an operating system of at least Windows XP, Mac OS X, or Linux and they must have a stable internet connection.

Open Issues

The application is developed for machines that can run any modern operating systems. But with the increasing demands of tablets, it is likely that there will be another development of the app in the Android or iOS store. As a result, the platform will be created from scratch because the web application is not optimized for mobile devices.
Co-Op Swift – Summary
Group 10 - Suvadeep Ghosh, Guillermo Martinez, Anthony Zuluaga, Akshay Raj

Introduction to the software

Agile software development is one of the most popular methodologies used in industries today. Its rapid popularity resulted in a huge number of tools designed to aid this process. This gave the companies an option to choose tools suited to their specific needs. While some may need a sophisticated combination of complex tools, others may be able to work with only limited features, resulting in an uneven coding standards in the industry. Another problem that has plagued the industry is the effort required in managing the software development process. Most free software has limited features, and paid-versions are either too expensive or are not able to cope with today’s development needs, and fail to provide a smooth integrated platform, which makes agile software development effortless to manage. To address these issues, we present our tool, Co-Op Swift, which targets industries that focus on agile software development.

Key features

Co-Op Swift is a state-of-the-art multi-platform (Linux, Windows and OSX) open-source tool designed to aid agile software development. The tool combines existing features from popular tools and adds several new features, which makes it a perfect candidate for both small and large size agile software development. The key features of the tool are listed below:

• The tool allows software developers to create projects, add developers and clients to the projects, create release and sprints, record user stories, create tasks from stories, assign/take up tasks, assign effort to the tasks, and categorize tasks under to-do, under progress, testing, done, reviewed.
• The tool will allow integration of Web-Ex, allowing sessions to be recorded. This will aid developers to revisit requirement gathering, in case confusion arises while development. It will also help others who might have missed the meeting.
• The tool provides Instant Messaging and Video chat feature which allows the developers to chat with the internal team as well as the client. This will encourage e-mail free environment, and will speed-up work.
• The tool keeps a track of due dates for all tasks. As the deadline approaches, it triggers email notifications to the owner of the tasks. This ensures that a timeline is never missed in cases where tasks are too many to be tracked manually.
• The tool records the time automatically for any task. The tool has a start button on the tasks which starts recording time spent by developers on that task. It also has a pause and stop button.
• The tool allows integration of communication tools like Skype, Facebook, and Hangouts. This feature can be controlled by a project owner who can choose not to enable it for privacy reason. However, some prefer using it in order to understand their client better, and add a human touch to the process.
• The User Interface allows the tasks to be seen in the form of a tree, with the tasks having no dependencies at the bottom, and tasks dependent on other tasks above it, giving it an
inverted tree-like structure. Development would start from the bottom and go all the way up to the root. Completed tasks are marked by green, in-progress indicated by yellow, and tasks in the to-do list represented by green. If a task has missed a deadline, it is marked red. This view helps in understanding where the overall project stand, and what needs more attention.

• Focusing on the quality of code, the tool introduces a feature called peer-review. Once the developer has completed a task, the send button on the task sends a request to another developer who performs a sanity check on the code. This design helps eliminate most bugs in the development phase.

• The tool keeps a track of the effort of the tasks assigned to each developers, and the tasks in the product backlog. It then recommends tasks to be allocated to developers based on prediction of their future availability. This saves a lot of time for the project owner, who can focus on other aspects of the project.

• The tool supports GIT integration, and has a built-in Integrated Development Environment which allows collaborative development.

The design of the product being developed will include constraints that are mandated towards this specific product. The product should be developed in a memory-safe object oriented language.

Target Audience
Co-Op Swift targets mainly two types of users, i.e., software developers and users involved in project management. The developers can keep a track of their project timeline, software components development schedule, new requirements, task analysis and intra-team communication whereas the project managers would ensure that the challenges faced by the team are addressed on priority, and that there is no hurdle in the development process. They also monitor project deadlines, developer bandwidth management and communication with clients via this tool.

Software versions
Co-Op Swift will come in 2 flavors i.e., a professional version which is heavy, and needs higher resources to operate due to all the features, and a student version, containing limited but essential features for those working on smaller projects suitable to work on a typical computer configuration. It is highly recommended that the students using this software have positive attitude and desire to work as a team. However, any member can get overburdened with tasks by claiming multiple stories. Students are expected to coordinate and cooperate to balance the task load. The student version will not have automatic assigning of tasks, as the software does not expect it to be huge, and hence, an unnecessary overhead.

The primary users of Co-Op Swift will be software developers, who will use this tool to keep track of the ongoing activities involved in the software development life cycle. The majority of feedback for this software will be received from this group, and the tool will be modified based on their suggestions.