CSC440/540 Team Project

- Written by Samuel Cho (chos5@nku.edu)
- Version history
  - ver 1.00 (revision with rubrics) - [2017/10/31]
  - ver 0.91 (1st revision) - [2017/10/30]
  - ver 0.90 (draft) - [2017/10/22]

Introduction of CSC440/540 Team Project

Software engineering is about building software systems to solve real-world problems. Building a software system is hard mainly because of the complexities inherent and accidentally introduced in building the system. In this team project, students learn how to build a software system–Knowledge Management System using wiki–in a team. Specifically, they learn how to apply the ideas and theories to build a high-quality software system that solves a real-world problem by managing the complexities. Students also learn how to make various types of software documents to support efficient software development and maintenance. The professor works as a coordinator of your team project. Your team, as a whole, is responsible for the software product you build.

Project Goals

1. Application of Theories: **apply** software engineering ideas/theories/philosophies learned in the 1st eight weeks of this course to a real-world software development process.
2. Teamwork: learn and experience how to define and solve problems (mainly software development related issues) in a **team**.
3. Software Development: learn and experience real-world **software development** tools, design methods, and processes.
   1. build a professional level software product (Knowledge Management System) with software tools including Python, PyCharm, Flask, Wiki, UML, VCS, UnitTest and with software documentation tools including Markdown, GitHub, readthedocs.org system.
   2. use the developed software product and documentation for their own software portfolio.

Project Stages

Why a team project in stages?

This team project is designed to mimic a real-world situation where (1) you work for a software company (2) as a software engineer. I assume you are hired as a junior-level software engineer, but you aim to be the leader of the team: i.e., a manager (or higher) or a senior-level software engineer (or higher) in the long run. Team members take on the role of team leader, in turns, to coordinate the team effort.

As a junior software engineer, you are typically given a short period of time to familiarize yourself with software development tools, and a company's software products once hired. The D stage and C stage mimic each of the situation. The D stage is for learning the tools including Python, PyCharm, and Flask. The C stage is for learning how the current product, the existing Wiki, is structured and designed using the tools learned in stage D. Once C stage is completed, you will extend the features of the current Wiki in the B stage. A stage is to take what you learned and developed a new product, Knowledge Management System.

You also can think that each stage represents a unique iteration. In other words, I expect four iterations to build a new software product from an existing one.

Expected deliverables and sharing project information

The deliverables for your team are provided at the end of each stage later in this document. I encourage everyone to cooperate for learning Python, Flask, and Wiki in C and D stages to jump into B stage as early as possible. You can share any study material or documentation during your study with any other students except for the specified deliverables.

In A and B stages, I prohibit from sharing any project related information among teams because (1) you compete to accomplish better features and services than other teams and (2) your results will be compared and graded against other teams' output. I value more in creativity and systematic approach than in programming skills and pretty UIs.

Class Wiki (Wiki440)

We have the wiki system for CSC440/540 (Wiki440) deployed with the existing Wiki. You will use Wiki440 for (1) testing the wiki features and (2) communication among team members and the coordinator (the professor).

1. The coordinator will use Wiki440 to teach Python, Flask, and Wiki topics that students need to know and understand for this team project.
2. The team will use Wiki440 to record its project progress, link their project documents and artifacts.

Stage Goals - from D to A+

D stage - learning the tools

In this stage, you learn Python/PyCharm and Flask.

1. The coordinator (the professor) provides D stage study materials (Flask examples and tutorials) and gives outside class lectures.
2. Students attend the lectures (at least one of the team members should attend the outside class lectures) and share the information with other team members if necessary.
3. It is important for you to understand how you learn to program effectively. Students can organize their learning experience to add to your "SE book".

C stage - learning the existing wiki system

In this stage, you understand the structure of Wiki system built with Python, Flask, and RESTful API.

1. Students use PyCharm to reverse engineer the existing wiki system.
2. Students understand the software architecture and design by building a new Knowledge Management System from the existing wiki system.
3. Students understand RESTful APIs, Web request and response architecture, HTTP protocols, and how Python's OOP paradigm can reduce complexities in developing software.

B stage - contribution

In this stage, you extend the existing Wiki system by adding features.

There are unlimited possibilities, but I suggest only some of possible extensions from MVC (Model, View, Controller) perspectives. I expect your team can come up with better ideas.

Model

1. Use SQL or MySQL for database system instead of markdown.
2. Support other formats than markdown including asciidoc, LaTeX, or any possible data format.

View

1. Better UI.
2. Javascript - e.g., better online editor
3. Calendar or other facility features.

Controller

1. Plugin system
2. Subitem structure, i.e., wiki/service is a subitem of a wiki
3. Security - login system (the only skeleton is implemented for current wiki system)
4. Version control system, users can revert to old Wiki pages if necessary.

A stage - application

Upon completing B stage, you have (1) a database and (2) Python code to control and manage the database. This means that you don't need to use the wiki to process the database. In this stage, you can build Knowledge Management System (KMS) that processes the database for your needs. For example, you can build a command line Python script to collect all the definitions stored in the database. Ultimately, you can build your "SE book" automatically (1) by putting all the information in the database and (2) select only the needed information and typesetting the output to the form you would prefer. You may think of other KMS applications using the database and Python code.

A+ stage - beyond

The team activity at this stage includes deployment and continuous integration.

However, due to time limitation, I don't expect all of the team reach this stage. Any team that reaches this stage and beyond will be rewarded with bonus points.

Deliverables

I request deliverables to make sure the three project goals–Application of Theories, Teamwork, and Software Development–are achieved.

Common to stages

1. Application of Theories
    1. As an individual, you write any of the SE rules you could identify or apply to the software development in the current stage. You can find possible examples in Wiki440 wiki page.
    2. The team leader starts collecting and then updates any rules that each member identified or applied at stage D/C.
2. Teamwork
    1. A current team leader must measure the time spent when more than two students meet to work on a project.

        - **why:** Time is the most important resources as a group effort. You must record (1) who attended the meeting and when and where the meeting took place, (2) what was decided–including expecting schedule and manpower, (3) how the previous decisions were executed.

3. Schedule and Agile retrospective documents
    1. At the beginning of each stage, your team will create a schedule using Gantt chart. A current team leader should keep a change of schedule, and a list of expected deliverables together with a checklist if the deliverables are delivered or not.
        - **why:** Keeping your schedule on track is one of the most important aspects in a team project.
    2. After each stage, a team leader should assemble a meeting to retrospect the previous progress and write a report. You can find possible teamplate in Wiki440 wiki page
        - **why:** Agile retrospective is the critical part of Kaizen (small progress in each iteration), and also a part of an important Agile process.

Deliverables at each stage

1. D stage deliverables
    - Identify SE rules you can apply when you learn Python, Flask, and Jinja2. Justify your rationale.
    - Each team member adds whatever they learned from Python/Flask to his/her 'SE book.'
        - **why:** I think your identification of SE rules from reading other's code can be an importatnt part of your "SE book".
2. C stage deliverables
    - Update SE rules you can identify and apply when you reverse-engineer the existing wiki.

    - Analyze of the existing wiki structure. You can use UML tool such as StarUML or PyCharm.
        - **why:** I believe reverse engineering is one of the best ways to understand the existing system.
    - Python unit test code extended from existing Wiki test.
        - **why:** I believe making tests is the only way to ensure safety to check introduced changes do not affect an existing system. Also, tests are a good way to teach functionalities of each object.
    - The initial GitHub repository: The first team leader should own the first version of your team's Wiki. Other team member clones the team's wiki and contribute.
        - **warning:** Your team's source should not be shared with other teams' members. Remember, you are graded by your contribution

to the source repository.
    - **why:** Most modern software development uses Git for version control system. You should familiarize yourself with Git and BitBucket. If your GitHub account allows private repository, you can use GitHub for your team project.
    - **hint:** You can use BitBucket that provides the private repository, but you also can request GitHub educational discount to use a private repository.
3. B, A, and A+ stage deliverables
    1. Agile requirements documents using User Stories.
    2. Design documents with UML diagrams.
    3. GitHub code structure.
    4. Agile retrospective documents.

Schedule

The professor works as a coordinator of your team project. Your team, as a whole, is responsible for the software product you build.

1. Each team should set up a todo list and a deadline of each stage.
2. A team leader's main job is to monitor the progress of each member's accomplishment. The team leader should report the coordinator if anything different from the schedule occurs.

Grading

This project is worth 400 points for undergraduate students, and 300 points for graduate students. These rubrics apply to both undergraduate and graduate students, but for graduate students, points are scaled to 300 points in total. Refer to the rubric documents.

Grading from the instructor (300 points)

Each stage has the deadline your team set. You can change your team's deadline, but the current team leader should report the change of deadline and write a report what caused the delay before the deadline. The schedule related report should be a part of deliverables.

- A team leader submit the required all the documentation before a deadline (100%).
- A team leader missed the deadline without any change notification to the notification (90%)
- A team leader doesn't send any documentation after a deadline. A coordinator asks the documentation to get it (50%)

- A team leader doesn't send any documentation after all. (0%)

All of the team members get up to 100 points per stage.

1. D stage (50 points) - grading rubrics
2. C stage (50 points) - grading rubrics
3. B stage (100 points) - grading rubrics
    - Each team should implement at least five features (20 points per each implementation up to 100 points).
        - At least one of the feature is not listed in my examples.
    - Each feature should have requirement document, design document, and implementation/unittest document.
4. A stage (100 points) - grading rubrics
    - Each team should implement at least two features (50 points per each implementation up to 100 points).
    - Compared to the B stage features, A stage features should contain a "problem description" document that contains the problem defintion, how they solve the problem, and how users can benefit from this application.

Evaluation results will be relative: when team A submits better outputs than those of team B, the instructor will give better grades to team A than team B. The final grading will be adjusted: when the best team scores 310, the team's score will be adjusted to 400, and all of the other teams' score is re-adjusted. For example, the team that earned 200 (64.5%) will earn 258.

Grading from peer reviews (80 points)

All of you participate in the grading of other team's work. Your grade has four parts. You can give only one team a full 20 points, and you can consider giving 0 points if you think the team's output is below par.

1. Project documentation quality. (up to 20 points)
2. Project code quality. (up to 20 points)
3. Project features and implementations ideas. (up to 20 points)
4. Presentation skills and quality. (up to 20 points)

I collect all the points and average them to rescale so that the highest points will be 80 points. All the team members get the same points.

Grading from peer reviews inside your team (20 points)

You must evaluate your team members performance. Your evaluation results will keep secret.

1. You give others' contribution from 0 to 10 points.
2. You give others' understanding of the code base from 0 to 10 points.

A coordinator collects all the points and averages them to grade.

To be fair, a coordinator may check each team's GitHub repository to see the activities to make sure of each member's contribution. To avoid this issue, a team member can/may explain why his/her contribution in GitHub is not active in the reports.