CSCI 578 – Course Project

Deadline: by 11:59:59 p.m. on Sunday, May 6, 2018

Submission Information: Please submit on D2L (details near the end of this document).

Please note that this assignment will require us to green-light your project. You will have to submit an overview of what you are planning to do by April 12.

15-minute project demos will take place on Monday, May 7, between 10am and 4pm. Further details of the location and a sign-up sheet will be provided.

What this Assignment is About

This is the team project you waited for – you get to select your team and your project, and try to wow us with its architecture and implementation.

Team Member Selection

You are expected to form a team. A team may have 3 or 4 members. An exception may be granted for DEN students on a case-by-case basis, although they are also strongly encouraged to join a team. (There are many tools available to enable effective remote collaboration.) Any DEN student who wishes to undertake the project alone should consult the instructors first.

Your team will need to have a good balance of the following skills:

1. Programmer: Writes the code for a system.
2. Architect: Determines the architecture of a system.
3. User/Manager: Comes up with the requirements for a system and evaluates whether the implemented system fulfills the requirements.

Do not mistake this balance of skills for some sort of strict division of labor. As we spoke in class, software engineers and their teams are expected to be flexible and share many responsibilities, often simultaneously.

Your teams will need to be finalized by the time the project overview has to be submitted (see “Deliverables”).
SELECT YOUR PROJECT

You must select one of the following three projects. The instructors have no specific preference for any one of the projects. The first project may be the safest choice because it continues the homework-assignment arc. However, it may also be a risky choice because the expectations of what constitutes an effective solution may be higher, and there may be more competition among the teams. You may want to take such factors into account.

PROJECT 1 - VISUALIZATION

An image says more than a thousand words: In this assignment, you'll be visualizing architectures based on the output of an architecture recovery method of your choice. From the previous assignments, you should have a good idea and understanding of what the output of architecture recovery looks like and what information it contains. You have also seen the visualizations coming from two recovery methods and know their strengths and weaknesses. It is now time for you adapt an existing visualization tool or create your own.

SELECT YOUR VISUALIZATION

Select a visualization. This could be anything you can find, as long as it is:

- Freely available
- Currently supported (i.e., the last release (non-beta) version has come out in 2017 or later).

Examples include:

- Any visualization achievable with Graphviz
- Any D3.js visualization

A list with many examples of free visualization tools can be found here: https://www.springboard.com/blog/31-free-data-visualization-tools/

CREATE OR SELECT YOUR DATA

Select the kind of data you would like to visualize. This has to be data that is

- Produced by one of the three recovery methods you have used,
- Has a meaningful relation to the architecture. (This means the data is part of the principal output and not just a byproduct, such as a log file.)

Optionally, you can use the source code of the system and the output of code summarization or metrics tools such as the Universal Code Count (UCC) in addition to the data mentioned above.
GUIDELINES

Your visualization should

- Emphasize qualitative changes between system versions over quantitative ones. As an example, you would do this by making it easy to see that the character of the system has changed (e.g., through the introduction of a new concern) as opposed to simply listing the numbers of entities in tables.
- Show the architecture or aspects thereof meaningfully,
- Allow scaling, and
- Be producible in a reasonable amount of time (10 minutes max per visualization, once the system’s architecture has been recovered).

It is recommended (but not strictly required) that you

- Make use of Interactive elements to zoom in or pop out elements, and
- Use an output format that can be rendered in different sizes.

SELECT A PROGRAM LANGUAGE AND WRITE YOUR TOOL

You will have to write a tool that will accept one or more files from the output of an architecture recovery run. It will then need to either

- Produce a visualization by itself,

or

- Produce output that can be processed by an existing visualization application, and
- Cause that application to produce a visualization based on that output (e.g. through scripting, submission to a website).

Note that your tool may have to consist of two or more components (e.g. a compiled component and a script that runs the first component and submits the converted data to the external visualization).

The final product after running your tool needs to be either

- A file in a common bitmap or vector graphics format (JPG, PNG, SVG etc.),
- A PDF, or
- A web page.

Please note that not all programming languages are acceptable for your tool. Your code and visualization must not be platform-specific and must be able to run within your Ubuntu VM.

EXCLUSIONS

You MAY NOT implement a sundial diagram or a derivative. “Been there, done that.” Try to think outside the box.
PROJECT 2 - CONNECTORS

As you know, none of our three recovery methods (ACDC, ARC and RELAX) is currently able to detect software connectors. Your task for this project is to address this. In order to keep the project’s scope reasonable, you only need to do so for Java-based systems.

You will need to do the following:

● Describe and implement a method to detect software connectors in Java programs.
● Describe how to integrate your tool and its results with one or more of our the three recovery methods and its existing visualization(s).

This project affords more flexibility than the previous one in that you can propose different foci and scopes. It is also more of a “green field” project in that there are currently no facilities for connector recovery so you may achieve quite a bit even by picking off “low hanging fruit”.

EXCLUSIONS

Your solution MUST NOT rely on topic modeling or text classification.

PROJECT 3 - NON-FUNCTIONAL PROPERTIES

This project is like project 2, but for non-functional properties. You may take any of the properties covered in Chapters 12 and 13, or other NFPs.

This project will give you the greatest amount of leeway and creative freedom.
Before you get started on a project, it will have to be green-lit by us. For this, we will need a description from you that lets us determine whether

1. The project is achievable for your team before the deadline, and
2. Whether it allows you to come up with a new architecture or enhancements to an existing one.

The description should be no shorter than one page and no longer than two pages, with 1-inch margins, of single-column, single-spaced text, using 12-point Times New Roman font. Any figures and tables must fit within the two pages. If we cannot agree on a project within the time frame, we will assign a project to your team.

Selecting a project is not easy. It will require research, thought, and discussion within the team. Project proposals that are carefully thought-out, creative, and ambitious, will receive higher grades than projects that were largely or completely specified by us.

Acceptable Programming Languages

- Java
- Python
- C
- C++
- Javascript
DELIVERABLES

OVERVIEW SUBMISSION

You will need to submit a PDF with a 1-2 page overview of your project on DEN before Thursday, April 12 at 23:59:59. This will, in essence, serve as a project proposal and should let us determine the scope of your project, along with your team’s composition. We will then get back to you by the 13th and either

- Give your project the go-ahead without changes,
- Give your project the go-ahead along with recommended changes, or
- Assign a different project to you.

No late submissions of project overviews will be accepted. While there is a relatively small percentage of overall project grade that is set aside for the overview, as detailed below, a failure to submit an overview will result in a grade of zero for the project. Please plan accordingly.

MAIN SUBMISSION

Please submit the following on D2L:

CODE

A ZIP file containing

- The source code of your project,
- A compiled version of your project that is runnable on at least one of
  - The CS 578 Ubuntu VM,
  - Windows 10,
  - macOS High Sierra,
  - iOS 11.x,
  - Android 8.x
- All necessary external tools and libraries in source or compiled form (unless they already came preinstalled on the Ubuntu VM you have received), and
- A short file named README_<Team Name>_<<Project Name>>.txt with instructions on how to compile your project (or stating that no compilation is necessary if that is the case)

WRITE-UP

A description of your solution and how you implemented it. This needs to be a PDF. We will not read beyond 2500 words.
Your project will be graded by the following criteria:

- 5%: Your project proposal
- 10%: Creativity, ambitiousness, and scope of your project.
- 5%: Your ability to meet your major goals as stated in your project proposal.
- 35%: The quality and aptness of your solution to the problem you are tackling.
- 25%: Your resulting system that implements your solution.
- 10%: The quality of your project demo.

Extra credit: The two teams whose projects impress us the most will get extra credit. The team in first place will get 30% extra credit, while the second place will get 15%. Please note that the extra credit is completely up to our discretion and no regrading requests for it will be entertained.

Late Submissions:
- No late submissions of the final projects will be accepted!