# CSCI 578 – ASSIGNMENT 3

Deadline: by 9 a.m. on Monday, April 10, 2017

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

## WHAT THIS ASSIGNMENT IS ABOUT

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.

## BEFORE YOU GET STARTED

You will need the virtual machine from the previous assignments. If you don't have it anymore, you can download it here.

Regardless of which version of the virtual machine you have, you will need to run the upgrade script available here. **You will only be able to perform the steps necessary for this assignment after running this script!**

## SELECT YOUR VISUALIZATION

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

- Freely available
- Currently supported

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 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).

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 which 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.

These languages will be acceptable:

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

If you wish to implement your system in a language that is not included above, please contact the instructor and TA.

## DELIVERABLES

Please submit the following files on D2L:

- A PDF of a write-up (at least 350 words) in which you
    - Describe the input and output of your tool,
    - Include at least two examples of the visualizations the tool produces,
    - Describe how the visualizations relate to the architecture of a system,
    - Describe how they scale to different system sizes, and
    - How they can give a good overview of a system.
- A zip file containing
    - The source code of your tool,
    - A compiled version of your tool that is runnable on the CS 578 Ubuntu VM,
    - Any external tools and libraries in compiled form, and
    - A short file named README_<First Name>_<Last Name>.txt with instructions on how to compile your tool (or stating that no compilation is necessary)

On extraction, the zip file needs to produce a self-contained directory that contains everything required to compile and run your tool. Note that any compilation systems or scripts that rely on downloads (such as Maven) are NOT acceptable.

## GRADING

1. 50% for the visualization,
2. 20% for the write-up, and
3. 30% for the tool.

Your visualization will be graded on how well it

- Relates to system architectures,
- Scales to system sizes,
- Allows rendering to different sizes,
- Incorporates interactive components, and
- Gives a good overview of a system.

Your write-up will be graded on clarity and helpfulness.

Your tool will be graded on code quality (maintainability, readability) and performance.

## LATE SUBMISSIONS:

- Up to 48 hours after the deadline: 1% off for each hour late
- More than 48 hours after the deadline: No submissions accepted anymore