CS 458 - Homework 2

Deadlines

Problems 2 onward are due by 11:59 pm on Friday, September 16, 2016

(Problem 1 was completed during the Week 3 Lab.)

How to submit

Submit your files for Problem 2 onward for this homework using `~st10/458submit` on **nrs-projects**, with a homework number of `2`

(Problem 1 was completed during the Week 3 lab.)

Instructions for using the tool ~st10/458submit

- If they are not already on **nrs-projects**, transfer your files to be submitted to a directory on **nrs-projects**.
- Once all of your files to be submitted are in a directory on **nrs-projects**, then use `ssh` (or Putty in a campus Windows lab) to connect to `nrs-projects.humboldt.edu` .
- use `cd` to change to the directory containing the files to be submitted -- for example,

  ```
  cd 458hw02
  ```

- type the command:

  ```
  ~st10/458submit
  ```

  ...and give the number of the homework being submitted (or whatever number you have been asked to do for lab-related files) when asked, and answer that, `y`, you do want to submit all of the files with of-interest-to-458 suffixes in the current directory. (Note that I don't mind if a few extraneous files get submitted as well -- I'd rather receive too many files that too few, and typing in all of the file names for each assignment is just too error-prone...)
- you are expected to carefully check the list of files that the tool believes have been submitted, and make sure all of the files you hoped to submit were indeed submitted! (The most common error is to try to run `~st10/458submit` while in a different directory than where your files are...)

Purpose

To get started working within your CS 458 project team (during the Week 3 Lab), to make some use of HSU's subscription to the ACM Digital Library, and to think and write about the different software development process models discussed in Jalote, Chapter 2.

Important notes

- Note that some of your submissions for this assignment may be posted to the course Moodle site.

Problem 1

Your team's file from the Week 3 Lab, submitted during that lab using `~st10/458submit` with a number of **2**, is "counting" as Problem 1 of this homework.

(That is, you will be graded on working with your team during that lab, and whether your team successfully completed this file, meeting the stated specifications, as part of your team meeting during that lab.)

Problem 2

A CS graduate -- as well as a software engineer -- ought to be able to use a resource such as the Association for Computing Machinery (ACM) Digital Library to look up articles on computing topics. For this problem, you are required to use the HSU Library's subscription to the ACM Digital Library to do some additional reading relating to software development process models. (This is both to give you experience doing so, and to help justify its continued availability here on-campus.)

Here is one of probably-several ways to reach the HSU Library's link to the ACM Digital Library -- you may find a more straightforward way. But this did work when I tried it using the Chrome browser right before posting this homework handout:

- Go to `http://library.humboldt.edu` .
- In the "Search and Locate" drop-down menu (in the top-ish left), click on "Articles & Databases".
- Scroll down to the "Databases by Subject" section, and click on the "Computer Science" link in the left-hand column.
- You should now be at the "Computer Science Research Guide".
- In the center, you should see a section called "Databases: Highly Recommended" -- click the first thing in that section, a link to the "ACM Digital Library".
  - If you are doing this on-campus, clicking this should lead straight to the opening page of the ACM Digital Library. If you are doing this off-campus, you should be asked to enter your HSU username and password, and then you should be taken to its opening page.
- You will see a variety of links here, in addition to a Search textfield and button on the top right.

As noted on the ACM Digital Library's home page, this includes the "Full text of every article ever published by ACM and bibliographic citations from major publishers in computing." And these articles include those from ACM journals, magazines, conference proceedings, and special interest groups as well as special collections that include links to complete e-books.

Sometimes I will ask you to search for articles on some topic -- but in this case I want you to look up and read a specific article: an opinion piece in the February 2004 issue of ACM Queue Magazine that is a pertinent companion piece to Jalote Chapter 2:

"'The Demise of the Waterfall Model is Imminent' and Other Urban Myths", by Phillip A. Laplante and Colin J. Neill

Because part of the purpose of this problem is to show that you have used the HSU Library's subscription to the ACM Digital Library -- in addition to also reading this article -- you are required to to the following:

- **FIRST**: take a screen shot of the ACM Digital Library "Search Results" page showing the link to this article.
  - (This can be a screen-shot of that window, or a cell-phone photo of that window, etc. Make sure it can be saved as a `.jpg`, `.gif`, `.png`, or `.pdf`, whichever screen-shot means you use -- and name it `hw2-acm-dl-img` followed by the appropriate suffix.)
- **SECOND**: follow the link to your preferred choice, HTML or PDF, for the full text version of this article, and read it. Then answer the following questions in your file `458hw2.txt`.

2 part a

In the cited survey, what was the dominant software development process model the practitioners claimed to have used?

2 part b

Which myth that they discuss did you find either the most interesting or the most surprising (your choice)? Why did you find that myth to be most interesting/most surprising?

2 part c

Give at least one of their recommendations from this article, and also say why you happened to choose it.

Problem 3

Consider the six software development process models discussed in Jalote Chapter 2:

- Waterfall Model
- Prototyping
- Iterative development (Jalote describes two variants, iterative enhancement and iterative delivery. For the purposes of this homework assignment, we'll limit this to the iterative **enhancement** variant.)
- Rational Unified Process (RUP)
- Timeboxing model
- Extreme Programming (Jalote grouped these as Extreme Programming and Agile Processing, but for the purposes of this homework we'll limit this to the more specific example agile model of Extreme Programming.)

In your file `458hw2.txt`, for **EACH** of these six software development process models:

- List at least three significant features/characteristics of that model.
- List at least one potential advantage of that model. (Note that this can be one from the chapter OR it could be something that you feel is an advantage, based on your reading and understanding about that model.)
- List at least one potential disadvantage of, or potential concern about, that model. (Note that this can

be one from the chapter OR it could be something that you feel is an disadvantage or that you feel COULD be a concern, based on your reading and understanding about that model.)

Problem 4

The following is adapted from the Self-Assessment Exercise 3 on pp. 35-36 of the Jalote course text.

Consider again the six software development process models discussed in Jalote Chapter 2 (and again, using the iterative **enhancement** model variant for iterative development, and using Extreme Programming for the category of agile models).

Also consider the following projects:

a) A simple data processing project.

b) A data entry system for office staff who have never used computers before. The user interface and user-friendliness are extremely important.

c) A spreadsheet system that has some basic features and many other desirable features that use these basic features.

d) A web-based system for a new business where requirements are changing fast and where an in-house development team is available for all aspects of the project.

e) A web-site for an on-line store which has a long list of desired features it wants to add, and it wants a new release with new features to be done very frequently.

Now, for an actual project, one could, of course, determine/develop a process that was some combination of different "classic" software development process models. But as an exercise for thinking about the classic models, I want you to think about how well these different models, as described, might work for these different projects.

In your file `458hw2.txt`, for **EACH** of these 5 projects a) through e):

- **PART 1**: Determine which of the above process models 1-6 -- as described in Chapter 2 -- might be the best "fit" for use with that project. **Name** the process model that you chose, and **describe why** you believe it is best choice for this project from amongst this set of process models.
  - If you feel you have to make assumptions about the project to determine the best process model "fit", then **include** those assumptions along with your model selection.
- **PART 2**: Then, for **EACH** of the remaining 5 process models that you **did not** choose, decide on one that you think would be a particularly poor choice, if not the worst choice, for that particular project, with the caveat that you should NOT choose the same process model as "worst" for all 5 projects, AND give at least one reason WHY you think that process model is a particularly poor fit for that particular project.
- (So, note that you'll have FIVE sets of answers, one for each example project a) through e),

  and EACH project's answer-set includes:
  - the name of the model you chose for that project,
  - a description of why you chose it,
  - the name of the model you think would be a poor or worst fit, and

- why you think it would be a poor or worst fit for that project.)

Problem 5

Consider the Timeboxing model, and answer the following questions in your file `458hw2.txt`.

Assume a project decides on a time box of 8 weeks, and in this case decides to have four stages (Requirements, Design, Build, and Test). So, Requirements has an iteration for 2 weeks, then passes it on to Design for 2 weeks, then passes it on to Build for 2 weeks, and then passes it on to Test for 2 weeks. (But, as discussed, after Requirements passes iteration 1 on to Design, Requirements works on iteration 2 for 2 weeks while Design is working on iteration 1. And after 2 weeks, Design passes iteration 1 to Build, while Requirements passes iteration 2 on to Design, and then Requirements starts on iteration 3. And so on...)

Assume in EACH of the following questions that "time 0", the start time, is when Requirements starts work on iteration 1 -- also assume that all goes well, and each stage completes its work on-schedule. (That is, consider these to be "best-case scenario" questions.)

5 part a

After how many weeks will the customer receive the completed first iteration?

5 part b

After how many weeks (from "time 0") will the customer receive the completed second iteration?

5 part c

After how many weeks (from "time 0") will the customer receive the completed third iteration?

5 part d

Assume there are 8 iterations. After how many weeks (from "time 0") will the customer receive the completed eighth iteration?

5 part e

NOW assume that instead of 4 teams of developers -- a Requirements team, a Design team, a Build team, and a Test team -- you only have 1 team, who handle requirements, then design, then build, then test, in sequence. Assume this single team is to develop the "same" 8 iterations/versions, and also spend 2 weeks on the requirements, 2 weeks on the design, 2 weeks on the build, and 2 weeks on the test, for each iteration. (This could be perhaps/arguably be viewed as an unusually-consistent iterative-delivery approach?)

Assume that "time 0", the start time, is when the team starts the requirements for iteration 1. After how many weeks will the customer receive the completed first iteration?

After how many weeks will the customer receive the completed eighth iteration?

(**NOT** TO ANSWER, but to **THINK** about:

- Note that, comparing the two scenarios, if each team happened to consist of the same number of people, the time-boxing approach takes four times as many developers to achieve the reduced delivery time (the way these imaginary scenarios happened to be set up).
- Consider also: if the customer, using the first iteration, discovers that what they "really" want is different, which scenario can more gracefully handle such changes in requirements?)

Submit your resulting file `458hw2.txt` along with Problem 2's `hw2-acm-dl-img.*` screenshot image file.