Name: Gustavo Hammerschmidt.

CS 441 Software Engineering

Assignment 1

1. Read the paper “No Silver Bullet: Essence and Accidents of Software Engineering” and answer the following two questions.

(1) Fred Books identified four essential difficulties of software systems. Based on your own experience and understanding of software development, which of these difficulties do you think is most challenging, and which difficulty is least challenging? Please justify your answer. (20 Points)

For Frederick P. Brooks, the four essential difficulties of software systems are complexity, conformity, changeability, and invisibility. I would say that the most challenging of them is conformity, because, many times, a software is designed to serve one specific purpose, but the environments which the software is used don’t operate well with it. That is, as I believe, the most complicated part of software design: when designers have to create and update frequently, as the user’s need evolves, interfaces to communicate with the software; or when they need the software to conform with machines or operating systems that it was not previously designed to work with, once these adapter-software components increase the difficulties in the software maintenance and need to be updated as the environment changes. The Least challenging difficult would be, in my opinion, the invisibility of the software, because there are ways to represent software that help many developers in exposing main concepts about the software; I believe that this difficulty in finding a good software representation tool is a matter of time. Currently, programmers can develop good software using the tools we have available today. Though it is a problem that can make the development process a lot more error-prone, programmers can still manage to deal with software representation problems than they would with complicated complexity and conformity problems, that appear in the development and testing phases and have to be solved afterwards.

(2) Pick one of the software methods or tools that you have used or you have learned about, and specifically explain whether or not you think this method or tool is a “promising attack” on any of the essential difficulties discussed in the paper. If yes, which specific difficulty does it address, and how? (20 Points) Grading Criteria: justification (concrete, consistent), writing (clear, no spelling errors).

I’ve used the UML(Unified Modelling Language) in a class about software architecture. To the extent that I have applied it, which was a pharmacy software, I think it is a excelent simple-way to represent basic features in the software and how they interact with each other to represent advanced features of the software; I personally like the components, required and provided interfaces idea that every part, considered to be a component, in the software is replaceable. Considering my experience with UML and what I’ve read in Brooks’ No Silver Bullet paper, I say that the UML has the potential to attack conformity and changeability difficulties of the software, as well as it can handle some designers difficulties on the software invisibility problem. But I do agree that it is not the best approach to manage all the requirements of some software systems: I was at a conference with the SUS(brazilian public health-care program) software department leader and she said that, most of the times when a problem with the system occured, they would first solve the problem and then update the software UML diagrams; with that they could focus on the problem itself instead of focusing on abstracting the errors first. To her, making diagrams in order to solve the invisible problems of the software was a waste of time and would only disturb the team’s productivity.

2. Discuss the main differences between prototyping and incremental development. (20 Points)

Brooks says, in his paper, that prototyping development opposes from the incremental one by trying to capture the final expectations of the client with a simulation of what the final software should be, by making adjustments on it as the client specifies it; there may be various prototypes of the same software in a hope to find a version that best fits with what the client requested. So the prototyping development would be a “runnable blueprint” of the basic features and interfaces of the desired software that, once it is set to be the prototype version that best fulfills the client’s needs, it is built upon the “blueprint” or the selected prototype version. The Incremental development aims on the contrary aspect of the prototyping development. In fact, what this approach does is to create a initial runnable-software that solves a little or nothing of the client’s requirements; and, from that initial version, the developers start to solve aspect by aspect of the desired software. What differs most from the incremental to the prototyping development is that the incremental grows a software to fit the client’s expectations and the prototyping builds a software from a prototype that the client likes better. The incremental developers build a software version and specify it, evolve it to the client’s needs; the prototyping developers refine the needs of the client and, upon that, they build the software from the initial to the final stage of the development.

3. Propose a software application that you would like a group of students in our class to build as their class projects. Your description of the application must be specific and concrete. If your application is chosen by the instructor, you will receive 1 extra credit (on top of your overall grade) at the end of the semester. Of course, you can still work on your own application if it is chosen by the instructor. (40 Points)

I propose to build an online petition application that can group regional petitions in a list, where users would be able to read and vote or give support to petitions. The application would basically request the user to log in the application’s website and the user would then be able to create a petition, as well as define details of it – such as: description, name, expiration date of the petition, etc. --, and would also be able to visualize other’s petitions and sign or upvote it. These petitions would have a thread behaviour: they would be clustered into the region to which they apply(for example, a petition to change some city feature would be grouped with other petitions of this same city), they would “die” if they exceed the final date and they would be ranked conforming to the relevance given by the users. The purpose of the application is to promote visibility and credibility to petitions, once they exceed the expiration date defined by the petition’s user-owner, sharing features would be available to all users to share in social networks and promote it to government agencies.