CSE 504 Assignment 1

Nan-Chen Chen

1. **Debug runtime error related to memory**

   For languages like C/C++, where we can directly access and manipulate memory, it is usually hard to debug runtime errors. For example, there may be one line in a code that accesses an over-range array index, but it may not lead to problem when no other program declares that block of memory. When we move a program that is developed on a machine with plenty of memory to one with smaller memory size, such an error may occur and it may not be trivial for figuring out this is the bug.

   Underlying cause

   (1) Memory operations are not intuitive and requires special attention to use it correctly.

   (2) Such errors are not always easy to reproduce and it may be challenging to surface the problems. Also, memory dump is not always easy to interpret. Therefore, tracing back to what leads to the problem requires great efforts.

   What data/tools to help

   (1) It may be helpful if we can extract the usage of memory in a program and examine if there are potential memory traps.

   (2) If we can collect runtime errors and see if there are patterns for different types of errors, we may have a tool to infer the potential type of runtime errors and guide us to finding errors.

2. **Fast development vs. Better maintenance**

   Sometimes, I write a program for one-time use purpose. In order to finish it shortly, I may not consider the design of the program structures carefully. However, sometimes it turns out that others want to use the program and request adding other features, which may not be easy under current design. On the other hand, sometimes I develop a program with very careful design, but it turns out that lots of the functionalities become not necessary, and it take too long to finish development.

   Underlying cause

   The goal of developing such software is uncertain and it changes after development starts.

   What data/tools to help

   (1) It can be helpful if the requirements are predefined clearly. We can also user studies to understand what parts may be extended in the future and what parts are less important.

   (2) If we can have a tool to take current codes and refactor them to simpler/more sophisticated design, it can be helpful.

3. **Too many scripts to remember parameters and correct workflow**

   When analyzing data, I usually write a set of scripts (in perl or python), and each script take different parameters. If I do not use the set of scripts for a while, it often takes some time to recall

what parameters to give and how to execute them in order.

Underlying cause

Sometime this is due to the scripts are not documented well, but other times it may due to that the scripts are overly complicated.

What data/tools to help

It can be convenient if there is a program that can record the usage of scripts and show the steps in an interface to help recall the steps or to help programmers figure out how to simplify the workflow

4. **Unexpected changes in dependency**

   For languages like python, where lots of packages are available for direct use, sometimes the software will be broken after a dependent package is updated and the updates are not backward-compatible. In some cases, updates are done unintentionally (e.g., we move the codes to another machine and reinstall the development environment, and we may not have versions of packages specified before doing that). Since python programs usually have lots of dependencies, it may not be easy to notice version difference leads to problems.

   Underlying cause

   Things we do not have control get changed and it may take some time to diagnose the errors we get are due to those changes in dependent packages.

   What data/tools to help

   Keep track of dependencies and send alerts when something we currently use in the program is changed in the latest update.

5. **Team development: Hard to be consistent**

   When developing software in a team, since members may have different programming background and habits, splitting tasks but at the same time remain consistency requires lots of communication. For example, everyone may have different naming conventions. Even though it is possible to combine components without consistent variable names/functions name, it may make the software less understandable for future development or sharing. Another example can be different way of using a framework: There are various ways of using Angular.js, and one may like to use curly braces for template ({{ variable }}) but one may like to use ng-bind (</div).

   Underlying cause

   Everyone has different programing experiences, habits, and styles. The resources of learning a library may also impact how a person uses it.

   What data/tools to help

   (1) If we can find inconsistency between codes/programmers, we can use it in code review and to improve consistency.

   (2) When declaring a variable/function, we can have a tool suggesting its name. The suggestions come from existing codes and other people's common styles.