How to Master Software Development and Design Skills

In the information era, computers have accounted for a fundamental portion of workplace and academic study. A variety of fantastic software has also been developed by different companies or individuals since then. However, plenty of software projects have failed in different software development stages for several reasons, such as overall cost, poor design, bad quality, useless product, etc. Considering its importance, this paper will offer suggestions to improve your existing projects and future career in software development. These suggestions include: version control, software development model and Pair programming technique.

Version Control:

Version control is a software development tool, which keeps a record of the changes made over time and allows you to support multiple releases of your software. Today, it is delinquent for a creative and cautious individual to operate a project without a backup plan. Considering data is ephemeral, it can easily be lost in several events, such as source code changes or catastrophic disk crashes. Consequently, it is wiser to sustain a live archive of all activities. By observing its important role, version control systems may be the most efficient way to customize the functional habits and project objectives for software developers. Specially, as one of the most popular version control system, Git is a highly strict command-line tool (in the version control system with free and open distributed source) which may help you to handle every project with speed and efficiency.

Table1 list some basic Git commands to get you going with Git.

Command	Usage
Configurations (Configure user information for all local repositories)	
git config --global user.name "[name]"	Sets the name you want attached to your commit transactions
git config --global user. Email "[email address]"	Sets the email you want attached to your commit transactions
Create repositories (Start a new repository or obtain one from an existing URL)	
git init [project-name]	Creates a new local repository with the specified name
git clone [url]	Downloads a project and its entire version history
Make changes (Review edits and craft a commit transaction)	
git status	List the status of your files that you have changed or created
git add [file]	Snapshots the file in preparation for versioning
git push <branch name>	To share the commits you've done with others, you need to push your changes to the remote repository
git pull	merge changes on the remote server to your working directory
git commit -m "[descriptive message]"	Record changes to the repository
Group changes (Name a series of commits and combine completed efforts)	
git branch [branch-name]	Creates a new branch

git checkout [branch-name]	Switches to the specified branch and updates the working directory
git merge [branch]	Combines the specified branch's history into the current branch
git push --tags origin	Push all tags to remote repository

(Table – 1: Basic Git commands refer to GIT CHEAT SHEE)

For more details, please go to *https://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository* to get to more Git commands and workflows, including some great examples.

Configurations

Before starting work on a new project, a set of git config commands should be executed. They can specify Git configuration settings. One of the first things that need to be done is set up your name and email address:

```
$ git config --global user.name "Andy"
$ git config --global user.email Andy@example.com
```

These two commands can configure the author name and email address to be used with your commits.

Create repositories

Basically, you can get a Git project using two main approaches.
The **first** way is to take an existing project or directory from your own machine and import it into Git. First of all, create the project's directory and go to this directory and type **git init.** Then, this directory will become a Git directory.

```
% mkdir repo
% cd repo/
% Initialized empty Git repository in ~/repo/.git/
```

The **second** way is to clone an existing Git repository from another server. In order to clone a repository, **git clone [url]** needs to be executed. For example, if you want to clone an existing project in git server and its url address is *https://github.umn.edu/umn-csci3081-F15/repo-group-Tiger*, Simply do this:

```
% git clone https://github.umn.edu/umn-csci3081-F15/repo-group-Tiger
```

That creates a directory named "repo-group-Tiger", initializes a .git directory inside it, and pulls down all the data for that repository.

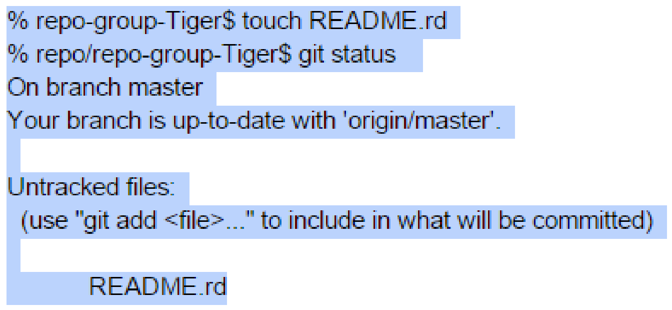
Make changes

The **git status** command is a main tool to determine which files are in which state. In Git, there are two states for each file: tracked or untracked. Tracked files are files that can be unmodified, modified, or staged. Untracked files are everything else – any files in your working directory that were not in your last snapshot. (Git—distributed, 2015) If a user runs the **git status** command directly after a clone, they should see something like figure-1.

```
% git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        repo-group-Tiger/

nothing added to commit but untracked files present (use "git add" to track)
```

(Figure 1 git status command directly after a clone)

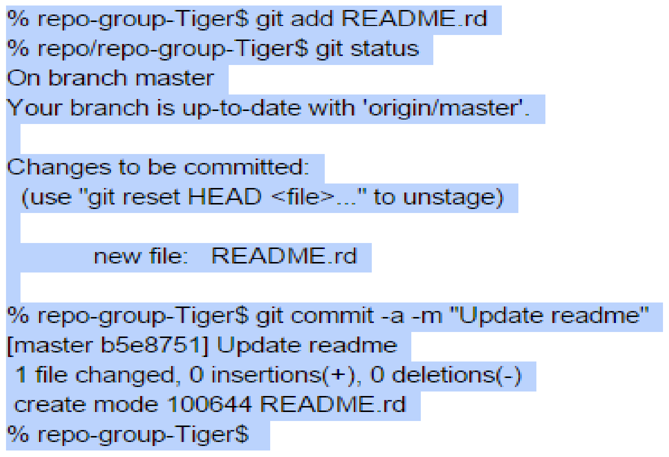
This means you have a clean working directory and there is no modified files. However, if you add a new file to your project, say a simple README file, and run **git status** again as figure 2 shows.



```
% repo-group-Tiger$ touch README.rd
% repo/repo-group-Tiger$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        README.rd
```

(Figure 2: git status after creating README)

This figure shows a new README file is under untracked files category, meaning Git sees this file which you didn't have in the previous snapshot (commit). In order to begin tracking a new file, **git add [file]** command must be executed. Now README file was staged, so you're ready to commit this change. Finally, use **git commit [file]** to record changes to the repository. Also, each commit command must have a message explaining the changes it contains. This can help users identify what they modified for each stage. The figure 3 shows how to use git add and git commit.



```
% repo-group-Tiger$ git add README.rd
% repo/repo-group-Tiger$ git status
On branch master
Your branch is up-to-date with 'origin/master'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   README.rd

% repo-group-Tiger$ git commit -a -m "Update readme"
[master b5e8751] Update readme
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.rd
% repo-group-Tiger$
```

(Figure 3: The usage of git add and git commit)

After you set your local repository followed by the previous descriptions, you can distribute projects in synchronizing the changes via **git push** and **git pull**. Git push can push your changes to the remote repository, and git pull can merge changes on the remote server to your own working directory.

Group changes

Almost every version control system has branching support. Git's branching and merging feature is essential for a team to develop separate aspects of their projects. To create a branch on your local machine, simply use the command git **branch [branch-name**], and use git checkout [branch-name] to switch to this branch. After you complete your work on that branch, use **git merge** to combine the specified branch's history into the current branch. If two branches have changed the same part of the same file, and user tries to merge those branches together, git will has trouble understanding which change should be used, so it asks you to help out.

Git also has the ability to tag specific points to mark different release points. This functionality is also useful for group changes. To make your tag, type the following command: **git tag -a [tag name] -m "[description of your tag]"**. Then, you need to push this tag up to the central repository. To do this, issue the following command: **git push origin –tags.** You can also use **git checkout [tag-name]** to check the specific tag.

Pros for git

There are lots of advantage for Git. Git is fast because nearly all operations are performed locally. It gives a huge speed advantage on centralized systems that constantly communicate with a server somewhere. Also, Git is an open source tool and it's really convenient to learn. Once you understand how each git command works, you can simply open your machine's terminal and work with your team members. If you have some trouble with Git, simply go to site *https://git-scm.com/documentation* and there are fully written documentations that can help you solve your issues simply and fast.

Cons for git

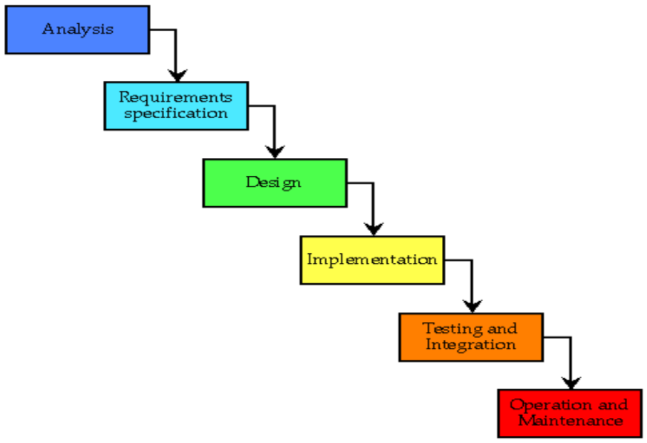
As every coin has two sides, Git is not an exception. One main issue is that Git doesn't allow multiple users to modify the code base at the same time. This is the most common conflict. For example, two programmer are working in the same file. The first one initially made a change on a particular line in a file while the other also makes some changes on the exact same line of that file, then a merge conflict occurred. Git has trouble understanding which person's code causes error message and programmers will get frustrated if they keep getting the same conflict message. The best way to solve this issue is simply waiting for your partner's signal and then pull the changes that your partner has made before pushing your own changes. Finally, you can modify the code and push it into the remote repository.

Software Development Model:

Like building the house, building the software is an incremental process which consists of various stages: planning, execution and inspection. Using some software development models can help programmer to achieve this goal. Many of them are described in software engineering. They have some distinct processes or methods in the development process of each project in accordance with project requirements and goals. In order to develop and maintain software systems, a variety of highly connected activities is required to use. Managing these activities needs various models. These models include waterfall model, iterative development, formal transformation model, spiral model, and build-and-fix model. Choosing the right model for each software product or application is very essential for company or individual to get success. In this paper, I will mainly focus on waterfall model and iterative development model discussed in our class.

Different Stages of Waterfall Model:

The waterfall model (**Figure 4**) is a sequential design process. It has distinct goals for each phase in the process of development. A good example is to imagine a waterfall on the cliff of a mountain. Once the water has flowed over a vertical drop in the course of a stream or river, it has begun its journey and it can no longer turn back again. This is very close to waterfall development model. Once a phase of development is completed, the development must proceeds to the next phase and there is no turning back. Waterfall model is still the most widely used deliverable based model. The different phases for waterfall development model is consist of: requirements analysis, design, implementation, testing integration, and maintenance. In the requirements and analysis phase, you need to find all possible requirements of software system and write them in a requirement specification document. In the design phase, you need to specify hardware and system requirements and define overall system architecture. During the implementation phase, user need to divide their programs into several small pieces called units. Each one of units is developed and tested for its functionality. In the testing and integration phase, user need to compare their unit test which they did in the implementation phase and compare program's functionality to their product requirements. In the last phase, there will be some issues which exist in client environment. User need to fix these issues in the last stage. Also, user need to enhance the product functionalities and update some better versions of program.



(Figure 4 Waterfall Model** Jon McCormack & Damian Conway, n.d. Web. 15 Dec. 2015. <http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/07.13.SWEng1/html/text.html* ***)

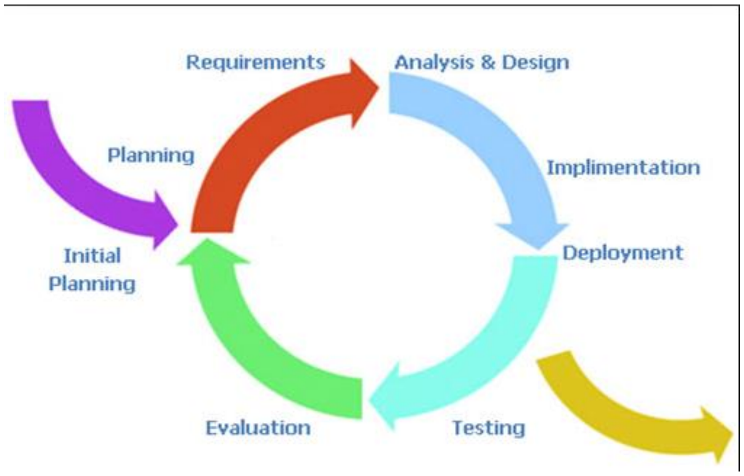
Pros and Cons for Waterfall Model:
Table 2 specifies the advantage and disadvantage for Waterfall model.

Waterfall model	
Advantage	Disadvantage
Simple and easy to understand and use	Not a good development model for long, complex, and objected oriented program
Easy to manage due to the constraint of the model. Each phase has specific deliverables and a review process.	No working software is designed until the last phase
Works well for projects where the requirements are very well known	It is difficult to measure progress within stages.
Phases are completed one at a time	There is a lot of risk and uncertainty with this development model.

(Table 2: Advantage and Disadvantage for Waterfall model)

Iterative Development Model:

The second software development model is **iterative development model (Figure 5)**. Although it has very similar stages as waterfall model, iterative model breaks the program down by functionality. In iterative development, feature code is designed, developed and tested in repeated cycles. With the application of iterations, additional characteristics can be added to the existing program, and the process of developing and testing. Your program has full functionality and is ready for costumer to use. There is a great example for iterative model. In CSCI 3081, there is a semester long project which is based on iterative development model. This project is to build a language translator which can map a small domain-specific language for climate analysis down to C or C++ for execution. In the first iteration, the requirement is to create a scanner which reads data from our input files. The second iteration requires us to extract our data from reading file and put them in a form so that we can parse all of them. In the third iteration, the data is added into parser which was created in the second iteration to build an abstract syntax tree data structure. In the final iteration, we will extend our abstract syntax tree classes to translate a given program written in the Climate Data Analysis Language to C++. From this specific example, it's clear that we kept adding new functionality into our existing program and repeated the process of developing and testing until our program finally achieve our initially requirement. The purpose of working iteratively is to allow more flexibility for changes.



***(Figure 5: Iterative Development Model** Voltreach. N.p., n.d. Web. 15 Dec. 2015. <http://www.voltreach.com/uploadedimages/iterative-model.jpg>.)*

Pros and Cons for Iterative Development Model:

Table 3 specifies the advantage and disadvantage for Iterative Development Model.

Iterative Development Model	
Advantage	Disadvantage
To build and improve the product step by step. Avoid the downward flow of the	Each phase of an iteration is rigid with no overlaps

defects.	
Able to get the reliable user feedback.	It will cost more time on system architecture or design issues may arise because not all requirements are gathered up at the beginning for the entire lifecycle
Spend less time on documenting and more time on designing.	Need to spend more time on debugging for each stage.

(Table 3: advantage and disadvantage for Iterative Development Model)

Pair Programming:

The effectiveness of software creation depends on factors such as creation pattern and people organization. Pair programming constitutes a kind of group work where two creators join and partner on one computer (Gevaert 2007). One programmer is identified as the driver, and his work is to control the typing function as well as execute the program. The second programmer is defined as the observer and has the role of analyzing the work to recognize any defects and offer suggestions. The two programmers tend to work interchangeably, where they switch roles throughout the process. There is a lot of speculation about whether pair programming is a good software development process or not. Because pair programming may not be fit for every development project or scenario, here are some advantages and disadvantages on pair programming.

Pros for Pair Programming

1. Quality of Design and Implantation

One of the most beneficial of pair programming is that the pairing partner will catch coding mistakes, and the overall quality of your program will become better since there always be another person to go through design and implantation stage one more time. Simple mistakes such as syntax errors or repeated variable names can be easily caught and fixed quickly. This can decrease debug time in the later testing stage and also prevent small irritating bugs.

2. Combining knowledge

The purpose of class CSCI 3081 is not only enhance your software design and development skills, it will also help you to build social networking skills. Computer science is a vast field and most of high technology is fast-paced. It's hard for any single student or programmer to have a comprehensive knowledge of what's been going on in class or in industry, so having a group of two and working together means that they can share knowledge each other, and it also leads to extra creativity, which results in improved ways of solving issues.

3. Good training ground for large software projects

As we all know, very little software nowadays is written by one person, team work for software is getting more important. A team ranging from small to big is very necessary for all software and technology company. Pair programming teaches us lots of soft skills such as tolerance, respect, understanding, and responsibility that you'll need when you enter to industry in the future.

4. Improve communication skills

Pair programming also provides benefits on improving our communication skills. Regardless of positions in an organization, improving good communication skills is a key component of professional success. Once you become a senior programmer in your company, you will spend more time communicating with your supervisor or boss than actually programming. Therefore, good communications skills, like written and oral, play vital roles in the software industry and pair programming offers us a chance to improve it.

Cons for Pair Programming

1. Skill disparity

This is the most common problem on pair programming. If two partners are totally different in skill levels, a programmer who has higher skill sets will probably be doing all the work or constantly tutoring the other. This is fine if you are trying to introduce computer programming to some people who are new in this field, but if your goal is trying to build a program together, then it can defeat the entire purpose of pair programming.

2. Conflict

Pair programming demands a certain level of accordance between partners. There will be a lot of conflicts that appear when two people are working together. For example, there may be time conflicts between their schedules, some people would like to choose the task or program functionality that he/she is familiar with.

Best solution for Pair Programming

The best solution is to give freedom of choice to developers. Pair programming is not a necessary step of the software development process, but it is important to understand its benefits. If the task is simple enough, coding alone is a better choice. However, when challenges or issues come, pair programming usually offers better results.

Conclusion:

We also explore other software design and development skills in this class. How many programming languages you have already known is not a good indicator for becoming a good software developer and engineers, because version control, software development model and pair programming technique can potentially help you to have a good career in the future. I hope what I have listed throughout this paper can motivate you to take CSCI 3081 in the coming semester. If you want to learn more about software engineers, this class is for you.

References

GIT CHEAT SHEET. Digital image. Training.github. N.p., n.d. Web. 15 Dec. 2015. <https://training.github.com/kit/downloads/github-git-cheat-sheet.pdf>.

Git--distributed. Digital image. Git-scm.com. N.p., n.d. Web. 15 Dec. 2015.<https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the Repository>.

Topic 13: The Software Development Process. Digital image. Csse.monash. Jon McCormack & Damian Conway, n.d. Web. 15 Dec. 2015. <http://www.csse.monash.edu.au/~jonmc/CSE2305/Topics/07.13.SWEng1/html/text.html>.

Arctern. Digital image. Voltreach. N.p., n.d. Web. 15 Dec. 2015. <http://www.voltreach.com/uploadedimages/iterative-model.jpg>.

Gevaert, H. (2007). Pair programming unearthed. Ottawa, Library and Archives Canada = Bibliothèque et Archives Canada.