Colin
FRAPPER

Software technology
1DV600

# Project plan

## Summary :



02/02/2017

*Note* : For planning my project, I'm gonna use the agile method, to be honest I didn't find so much thing about agile in the lectures (or I don't understand it), so I search on the web and find this website which I found very useful because there is a lot of illustration and that give me a good idea of what i'm doing, I'm gonna use this website to make the plan of the project.
( http://www.agilenutshell.com/ )

# I - Introduction :

*Task : Write a project plan for the project. This project plan should show the way to the complete and finished application, something that you should be able to follow.*

Quick introduction to agile :
*"Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.*
*It works by breaking projects down into little bits of user functionality called user stories, prioritizing them, and then continuously delivering them in short two week cycles called iterations."*

So my task here is to make a project plan, so I can use it to do all the future task who will come with the assignment and to show the way to complete them.

# II - User stories :

This must be the begginninng of the project, sitting down with your customer you make a list of features they would like to see in their software. We call these things user stories and they become the TODO list for your project.

*"User stories are features our customers might one day like to see in their software.*
*User stories are like Agile requirements except that they're not. For one there's no guarantee all these features are going to make it into the final version of the software. Secondly, Agilists know their customers are going to change their mind - and that's OK. Because they weren't really requirements to begin with."*


## Master story list :

- Assignment 1
  - Task 1
    - Reading the subtasks and understand them.
    - Subtask A :
      - Create an object that represents book.
      - Create a function who output a list of books.

    - Subtask B :
      - Convert object into a JSON Object (use the subtask A).
      - Show them in the terminal.

    - Subtask C
      - Print the JSON Object in the web browser.

  - Task 2
    - Reading the task and understand it.
    - Make a vision document

- Assignment 2
  - TODO

- Assignment 3
  - TODO

- Assignment 4
  - TODO

# III – Estimation :

I will use the tool toggl, to be more precise on the time I spent on the assignment(s)/task(s). I didn't do it for this first assignment because I didn't know this tool and then I saw a message on slack who makes me discover this tool.


# Master story list :

- Assignment 1 (3days, 1hour, 5 minutes)
  - Task 1 (1hour, 5 minutes)
    - Reading the task and understand them. (15 minutes)
    - Subtask A : (30 minutes)
      - Create an object that represents book. (10 minutes)
      - Create a function who output a list of books. (20 minutes)

    - Subtask B : (10 minutes)
      - Convert object into a JSON Object (use the subtask A). (8 minutes)
      - Show them in the terminal. (2 minutes)

    - Subtask C (10 minutes)
      - Print the JSON Object in the web browser. (10 minutes)

  - Task 2 (3 days)
    - Reading the task and understand her. (1 day)
    - Make a vision document (2 days)

- Assignment 2
  - TODO

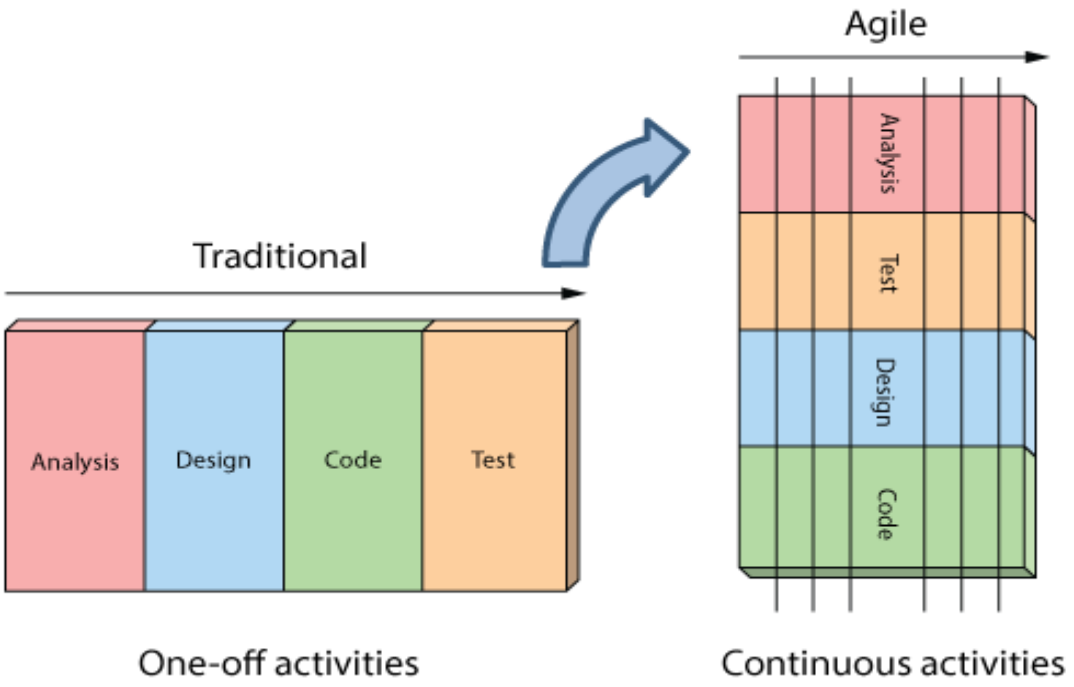
- Assignment 3
  - TODO

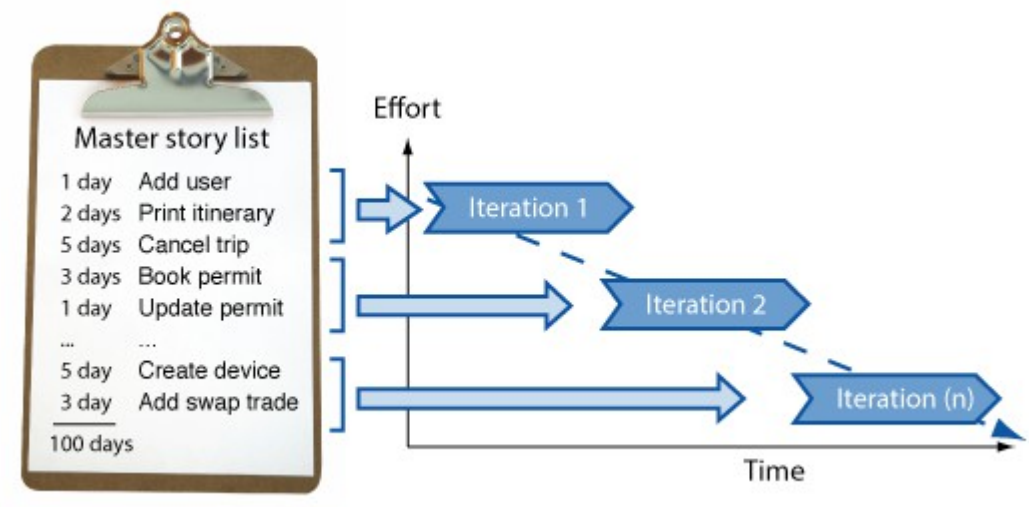
- Assignment 4
  - TODO

*Note* : It was very hard to know how long a task will take when you never done this kind of task before.

# IV - Iteration Planning :

Everything happens during an iteration. Analysis, design, coding, testing.



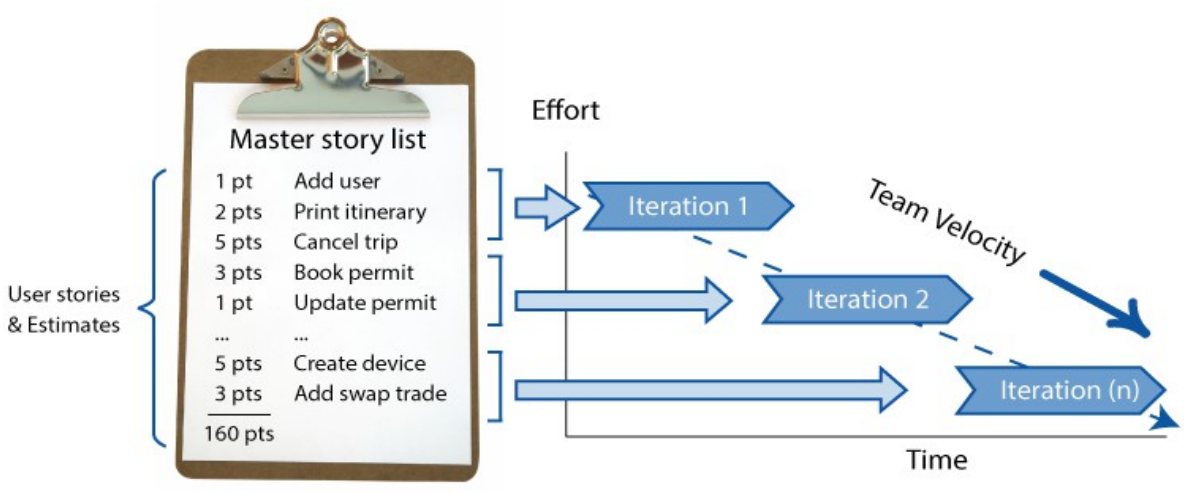
Explication : In our project, iterations are like every task in our todo list, each assignment will build on the previous.

*"The beauty of working this way, is every couple weeks the customer gets something of great value (working software), but it's also a great way to track progress (measuring the rate at which the team can turn user stories into production ready working software)."*

It's very important to keep the customers aware of the development of the project, so there is some confidence between you and your client.

## V – Planning :

*"The speed at which we turn user stories into working software is called the team velocity. It's what we use for measuring our team's productivity and for setting expectations about delivery dates in the future.*
*The engine for getting things done is the agile iteration - one to two week sprints of work where we turn user stories into working, production-ready software."*



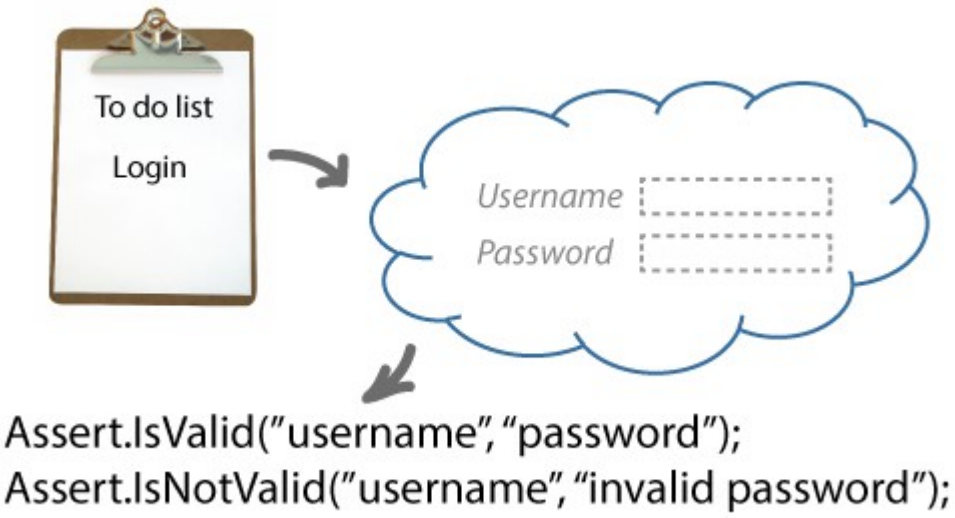

How you evaluate the task in term of points, well there is a picture to help us to do that :

*“To give us a rough idea about delivery dates, we take the total effort for the project, divide it by our estimated team velocity, and calculate how many iterations we think we’ll require to deliver our project. This becomes our project plan.*

*iterations = total effort / estimated team velocity”*

# VI - Unit testing :

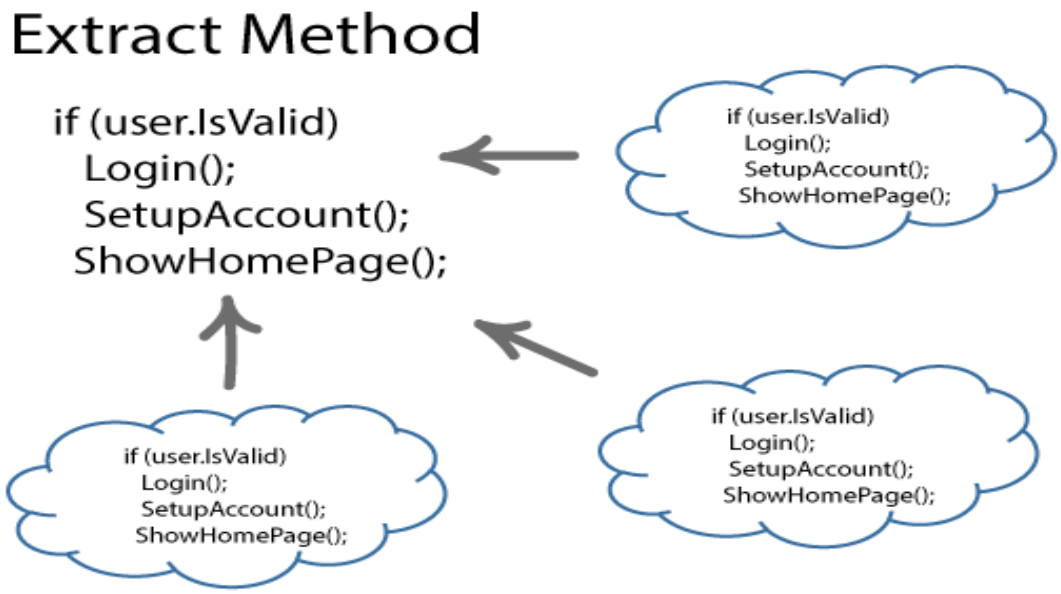
*“Unit tests are snippets of test code developers write to prove to themselves that what they are developing actually works. Think of them as codified requirements.”*




This part is exactly what you must do when you finished a task on the TODO list or Master story list, it means that every times a task is done you must test it.

# VII – Refactoring :

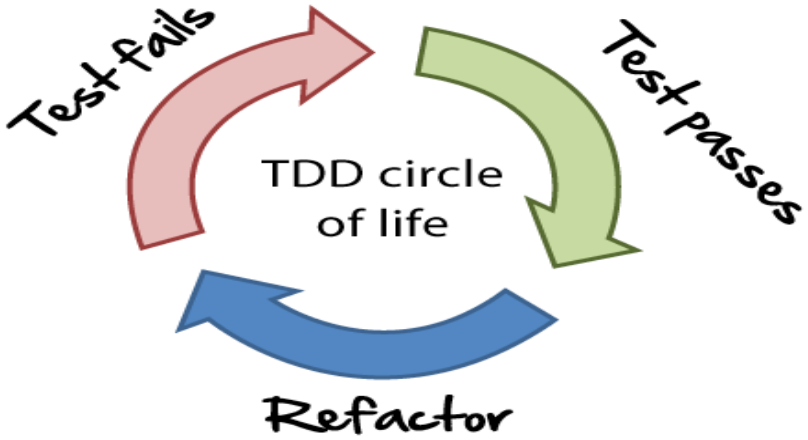
*“As we add more functionality to the system we need a way of maintaining our design and keeping our house in order. In Agile we call this refactoring.*
*For example, instead of copying and pasting code every every time we need some functionality, it's much easier to maintain if we extract that code into a one place and call it from where ever we need it.”*




In our project, this example can be compared with the creation of a book, we know that we gonna use this (and others) functions multiple times, so instead of writing it again, we must extract the method.

## VIII - Test driven development :

*"Test Driven Development is about writing the test first before adding new functionality to the system. Agile developers work in this circle of life when adding new code. Write the test first. Make it pass. Then refactor."*




It will permit to :

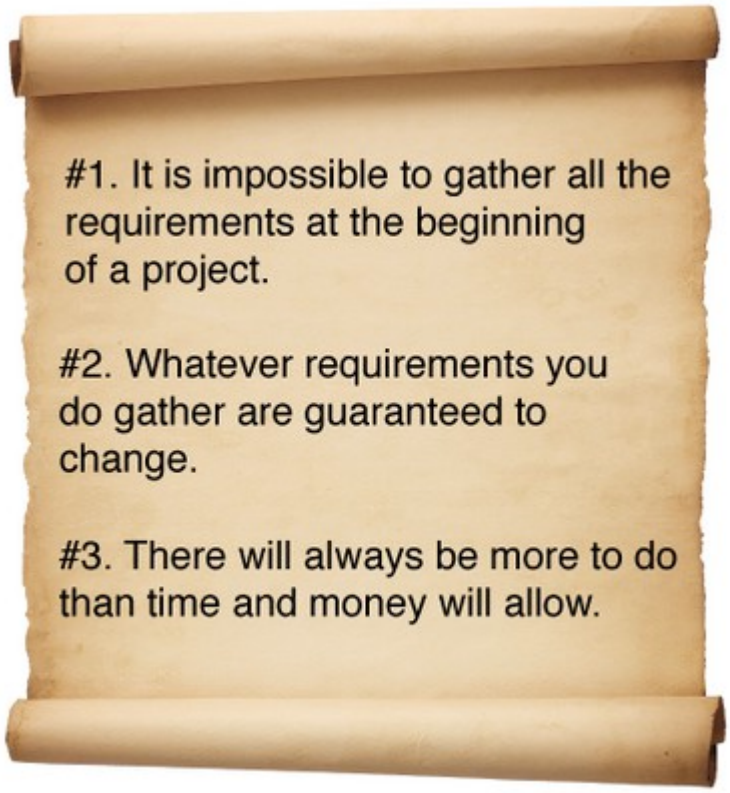
- Defines success up front.
- Helps break our design down into little pieces, and
- Leaves us with a nice suite of unit tests proving our stuff works.

In our project, we know that we must implement multiple function like addBook, deleteBook, etc. So first we gonna make some test about it, and ask myself what must be the result, what do I need to implement this function, do I need a function already written ? It will be really helpful to have a big picture of what I'm gonna need to make the requirement/task succeed.

# IX – Conclusion :

The resume of my project plan :

1. Define the requirements with the customers (here with myself).
2. Make an estimation of each task.
3. Write the test first before adding new functionality to the system (Test driven development).
4. Iterate each task (make the analysis, unit testing, design, coding).
5. Check the planning after each task if we are late or in advance and show the result of the test to the customers to keep them aware of the advancement of the project. (in this project to myself).




I really like those rules, this is exactly what will happen in the “real” life.