Software Development Models

1. Software Life Cycle

Three main stages of a computer program:

1) *Development*
2) *Use*
3) *Maintainance*

As the names indicate, the development phase includes all the planning, writing, testing, and debugging of a computer program. When you are fairly certain that a program is usable, then you release it for use.

Any time a program is being used, it is quite probable that the users will find problems with the original release of a program. Even if they do not find major problems, they will almost certainly be able to suggest improvements. These reports are then used to improve the quality of the original program. These improvements are the maintenance of the original software.

Finally, most programs become too cumbersome to maintain at some point. It would be more beneficial for a company to start from scratch and write a new program for a task than to continue maintaining the old program. At this point, the program becomes retired. (Of course, if it really is good, like Michael Jordan, then it'll continue to come back several times from retirement.)

The duration of this life cycle is dependant on a number of things: purpose of the program, the extensibility of the original design, and the changes in the demands of the user.

One key note: many beginning programmers don't realize just how costly (time wise) maintenance can be. In fact, nearly three times as much time is used to maintain a program than is used to originally develop it.

Why should we think about this now???

Because how well we initially design and develop our programs could potentially drastically reduce the maintenance time of those programs.

If we have the foresight to identify future uses or changes that a customer may need and create our design to make those changes easier to implement, then we will save maintenance time. Hopefully we will save more time than the extra planning time that is needed.

The goal of software development is NOT to minimize the development time, but to minimize the overall effort necessary to develop, use and maintain a program over a period of time.

A couple analogies:

1) Need to plan to build a house
2) Brush your teeth! (Yes, I have taken bribes from local dentists!!!)

2. Development Process Model

Build-and-Fix Model:

1) Write your program
2) Run it until you find a problem, if you don't for a while, you are done, release your program.
3) Fix that problem
4) Go to 2

The problem here is that the testing of the program is not planned and not systematic. Maybe the user will catch some errors, but it's far more likely that some will fall through the cracks, without more careful testing.

In the 70s, some proposed the *Waterfall Model*:

1) Establish Requirements
2) Create Design
3) Implement Code
4) Test system

Once again, even if we plan how to test our program beforehand, this system does not allow us to go back to a previous step. Even with a great design, it's possible that we can't predict issues that will only present themselves during steps 3 or 4. It these cases it would be nice to go back to steps 1 and 2.

Using these two models, we can create the Iterative Process of program development.