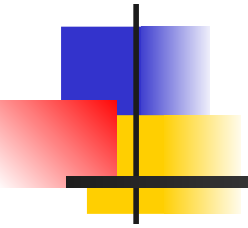

Why Object-oriented Programming?

Why not code everything in C?

James Brucker

Problems of Software Development

1. Software is *Complex*.

- Microsoft Office had 30,000,000 lines of code in 2006. Mostly C++. https://blogs.msdn.microsoft.com/macmojo/2006/11/02/its-all-in-the-numbers/
- Apache Web server (httpd) has 1,832,007 lines of code. Mostly C.

https://www.openhub.net/p/apache

Problems of Software Development

2. Software *changes*. *Lots of change*

- ✓ requirements change
- ✓ our understanding changes
- ✓ technology changes

...and it happens *during* the project

Apache httpd:

2,805 Commits in last 12 months

28 Contributors

Problems of Software Development

3. Modeling real-world problems is *hard*.

Modeling mismatch:

Behavior of real things does not match behavior of software components.

Problems of Software Development

4. Software tends to have a lot of *defects*.

   This is partially a result of *complexity*, *change*, *and modeling mismatch*.

   It also is related to our development process

Problems of Software Development

5. Software is expensive to develop and maintain.

   Too much *change*, *complexity*, and *mismatch*.

   Too many *defects.*

   Lack of standard components.

   Too much "custom development".

What We Want

- ✓ **Reduce complexity**... divide into simple parts
- ✓ **Hide details** inside the parts
- ✓ **Easy to change or replace**.
- ✓ **Model software closely to real-world objects**
- ✓ Improve **testability**... to reduce bugs.
- ✓ **Reuse code**... to reduce cost.
- ✓ **Reuse entire applications**... just "plugin" custom features.

OO reduces complexity

Divide a program into simple classes

- each class has a single responsibility

Encapsulate complexity

- each class has its own *responsibilities* and *data*
- class has a *well-defined* interface
- *hide* implementation and data

Easy to Change

Classes are loosely coupled

- class depends only on behavior of other classes
- *don't depend on how behavior is implemented*
- *depend on only a few classes*

as a result...

- *localize the effect of change*

Modeling is Simpler

Objects resemble "real world" things:

- they have behavior
- they have knowledge or state
- they relate to other objects (things)

Let us think about the problem instead of the code

Help to *simplify the problem* by encapsulating details.

Code Reuse

- classes with few dependencies are reusable in other applications
- *polymorphism* lets us substitute one class for another one
- *inheritance* lets us build new classes that reuse code from old classes
- combining these 3, we can reuse (almost) entire applications. Called *frameworks.*

Easier to Test

- we can test each class by itself (unit testing)
- we can test a group of classes together (component testing, integration testing)

Intrinsic Complexity of Software

"There is no single development, in either technology or in management technique, that by itself promises even one order of magnitude improvement in productivity, reliability, or simplicity."

"*No Silver Bullet*" by Frederick Brooks. *Computer*, 1987

What does he think of O-O Programming? ...

Brooks on O-O

"***Many students of the art hold out more hope for object-oriented programming than for any of the other technical fads of the day.***

I am among them."

"*No Silver Bullet*" by Frederick Brooks. *Computer*, 1987

Read the article to learn <u>why</u> Brooks believes in O-O programming.

Miller's Law and Complexity

At any one time, a person can concentrate on at most 7 ± 2 chunks (units of information)

- ⇨ Need to ***limit complexity***.
- ⇨ How to limit complexity?
  - hide it!
  - modular design
  - only use a module's *public interface*
  - limit dependencies between modules

Procedural vs O-O Paradigm

http://www.youtube.com/watch?v=D8jZ0l_GwXQ