The Solid Principles of OO & Agile Design

Dependency Management

- What is dependency management?
- What bearing does DM have on software?
- What is the result of poor DM?
- What is the advantage of good DM?

- What is OO?
    - All languages went from 'goto' to OO
    - Why did OO win?
        - More maintainable code?
        - Code is easier to change due to modularity/dependency management
        - Objects require polymorphic interface
        - Program control opposes flow of dependencies
            - Pointers to functions(callback)
    - Copy Program
        - Draw diagram
        - Copy → Read Keyboard → WritePrinter

```
void copy()
{
   int c;
   while ((c = rdkbd()) != EOF)
      wrtprt(c);
}
```

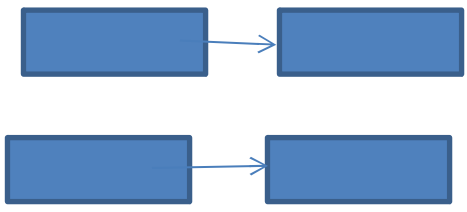
- Abstracted =

```
void copy()
{
   int c;
   while ((c = getchar()) != EOF)
      putchar(c);
}
```

- Getchar/putchar don't mention devices so no rot occurs on update
-

What is OO?

- Open/Closed Principle
    - A principle which state that we should add new functionality by adding new code, not by editing old code.
    - Defines a lot of the value of OO programming
    - Abstraction is the key
- Open for extension, but no modification needed to original.
    - Add polymorphic extensions
    - Now can update
- Abstraction is Key
    - Client/Server relationships are 'open'



    - Changes to servers cause changes to clients
    - Abstract servers ‘close’ client to changes in implementation
- If derivative needs less then base class, it isn’t a derivative
    - Should be its own classs
- “IS A” relationship
-