Object-Oriented Programming

Vertical Development

Michael L. Collard, Ph.D.

Department of Computer Science, The University of Akron

Workflow

- When working on a project, you can work *horizontally* or *vertically*

- On most projects, you should work *vertically*

Terminology

Layer	f()	g()	h()
Function Call	`if (f())`	`if (g())`	`if (h())`
Function Declaration	`bool f();`	`bool g();`	`bool h();`
Function Definition	`bool f() {}`	`bool g() {}`	`bool h() {}`

Horizontal Workflow

1. Write the *function declarations* for multiple functions in *xml_parser.hpp*

2. Commit to the repository

3. Write the *function definitions* for those functions in *xml_parser.cpp*

4. Commit to the repository

5. Replace the code in *srcfacts.cpp* to call these new functions

6. Commit to the repository

Horizontal Commits

```
...
...
...
git commit -am "Add function declarations for
↪ f(), g(), and h() to xml_parser include file"
...
...
...
git commit -am "Add function definitions for
↪  f(), g(), and h() to xml_parser implementation file"
...
...
...
git commit -am "Add code in srcFacts to call xml_parser
↪ functions f(), g(), and h()"
...
...
...
git commit -am "Remove code replaced by calls to
↪ xml_parser functions f(), g(), and h()"
```

- Must be done in this order
- Must know the complete list of functions to move before the start

Vertical Workflow

1. Write a *function declaration* for a single function in *xml_parser.hpp*

2. Write the *function definition* for that function in *xml_parser.cpp*

3. Call the new function in *srcfacts.cpp*

4. Remove the old code in *srcfacts.cpp*

5. Commit to the repository

Vertical Commits

```
# Add function declaration for f()
# Add function definition for  f()
# Add code in srcFacts to call f()"
# Remove code replaced by call to f()"
git commit -am "Extract function f()
↪ from srcFacts to xml_parser"

# Add function declaration for g()
# Add function definition for  g()
# Add code in srcFacts to call g()"
# Remove code replaced by call to g()"
git commit -am "Extract function g()
↪ from srcFacts to xml_parser"

# Add function declaration for h()
# Add function definition for  h()
# Add code in srcFacts to call h()"
# Remove code replaced by call to h()"
git commit -am "Extract function h()
↪ from srcFacts to xml_parser"
```

- Complete one function at a time
- Can be in any order

Horizontal Workflow Disadvantages

```
...
...
...
git commit -am "Add function declarations for
↪ f(), g(), and h() to xml_parser include file"
...
...
...
git commit -am "Add function definitions for
↪  f(), g(), and h() to xml_parser implementation file"
...
...
...
git commit -am "Add code in srcFacts to call xml_parser
↪ functions f(), g(), and h()"
...
...
...
git commit -am "Remove code replaced by calls to
↪ xml_parser functions f(), g(), and h()"
```

- The *cohesion* of the commit is at a low level, closer to the programming language features
- There are no results (i.e., improvements in the file *srcfacts.cpp*) until we are all done
- We do not learn as we go and often have to rewrite what we have done
- Can only report progress at the very end
- This is especially true when the functions grouped together have different parameter lists, implementations, etc.

Vertical Workflow Advantages

```
# Add function declaration for f()
# Add function definition for  f()
# Add code in srcFacts to call f()"
# Remove code replaced by call to f()"
git commit -am "Extract function f()
↪ from srcFacts to xml_parser"

# Add function declaration for g()
# Add function definition for  g()
# Add code in srcFacts to call g()"
# Remove code replaced by call to g()"
git commit -am "Extract function g()
↪ from srcFacts to xml_parser"

# Add function declaration for h()
# Add function definition for  h()
# Add code in srcFacts to call h()"
# Remove code replaced by call to h()"
git commit -am "Extract function h()
↪ from srcFacts to xml_parser"
```

- The *cohesion* of the commit is at a high level, closer to the functionality
- See results almost immediately. Each commit results in an improvement to the design of *srcfacts.cpp*
- Learn as we go. As you start extracting functions, you will see whether it will work as you go along
- Can report progress better. Each commit reflects an improvement in *srcfacts.cpp* that you can show
- Can be performed in any order, which can change anytime
- Scale
what
chan

Similar Terms

Term	Definition	Characteristics	Similarity
iterative	Work in *iterations*	Functionality/Improvement	vertical
incremental	Work in *increments*	Often in *layers*	horizontal

When to work horizontally

```
...
...
...
git commit -am "Add function declarations for
↪ f(), g(), and h() to xml_parser include file"
...
...
...
git commit -am "Add function definitions for
↪  f(), g(), and h() to xml_parser implementation file"
...
...
...
git commit -am "Add code in srcFacts to call xml_parser
↪ functions f(), g(), and h()"
...
...
...
git commit -am "Remove code replaced by calls to
↪ xml_parser functions f(), g(), and h()"
```

- Very experienced with the procedure
- Lots of practice
- It is much more efficient to do all of something instead of as a separate step
- Completion time of all horizontal steps is short and within a working session
- None of these apply to the programming in this class

```
# Add function declaration for f()
# Add function definition for  f()
# Add code in srcFacts to call f()"
# Remove code replaced by call to f()"
git commit -am "Extract function f()
↪ from srcFacts to xml_parser"

# Add function declaration for g()
# Add function definition for  g()
# Add code in srcFacts to call g()"
# Remove code replaced by call to g()"
git commit -am "Extract function g()
↪ from srcFacts to xml_parser"

# Add function declaration for h()
# Add function definition for  h()
# Add code in srcFacts to call h()"
# Remove code replaced by call to h()"
git commit -am "Extract function h()
↪ from srcFacts to xml_parser"
```

Summation

- Work vertically

- Get one thing done on *srcfacts* that shows improvement to *srcfacts* as fast as possible

- Quit working horizontally