Best practices in scientific programming
Software Carpentry, Part I

Valentin Hänel
valentin.haenel@bccn-berlin.de

Technische Universität Berlin
Bernstein Center for Computational Neuroscience Berlin

Python Winterschool Warsaw, Feb 2010
Slides based on material by Pietro Berkes

Todays Schedule

Morning

- Valentin
  - Agile Methods
  - Unit Testing
  - Version Control
- Rike
  - Unit Testing Examples
  - Subversion
  - Debugging
  - Profiling

Todays Schedule

Afternoon

Niko

- General Design Principles
- Object Oriented Programming in Python
- Object Oriented Design Principles
- Design Patterns

Motivation

- Many scientists write code regularly but few have formally been trained to do so
- Best practices can make a lot of difference
- Development methodologies are established in the software engineering industry
- We can learn a lot from them to improve our coding skills

Scenarios

- Lone student/scientist

- Small team of scientists, working on a common library
- Speed of development more important than execution speed
- Often need to try out different ideas quickly:
  - rapid prototyping of a proposed algorithm
  - re-use/modify existing code

Outline

What is a Development Methodology

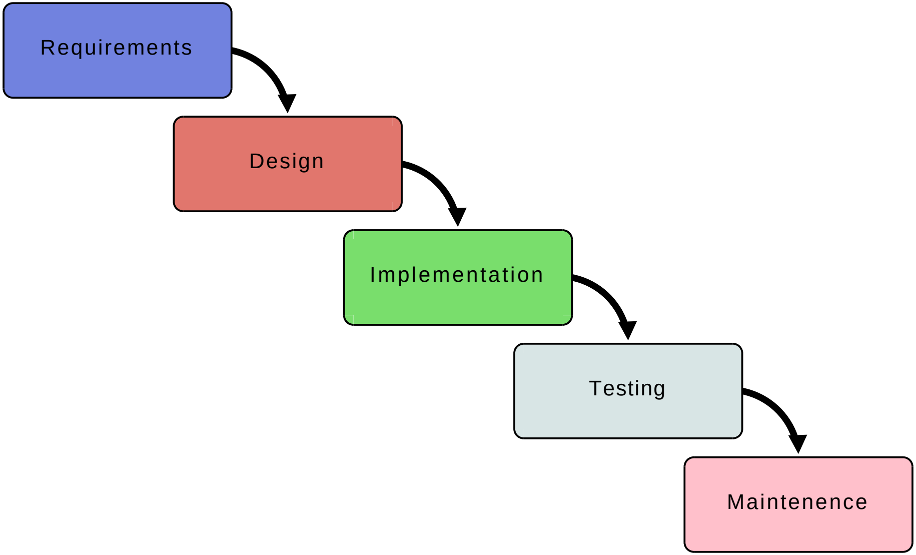
Consist of:

- A philosophy that governs the style and approach towards development
- A set of tools and models to support the particular approach

Help answer the following questions:

- How far ahead should I plan?
- What should I prioritize?
- When do I write tests and documentation?

The Waterfall Model, Royce 1970



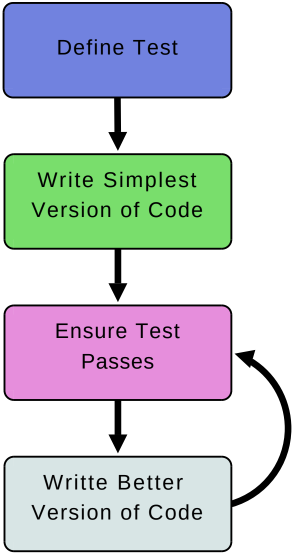
Agile Methods

- Agile methods emerged during the late 90's
- Generic name for set of more specific paradigms
- Set of *best practices*
- Particularly suited for:
    - small teams (less than 10 people)
    - unpredictable or rapidly changing requirements

Prominent Features of Agile methods

- Minimal planning
- Small development iterations
- Rely heavily on testing
- Promote collaboration and teamwork
- Very adaptive

The Basic Agile Workflow



Example

function `my_sum` should return the sum of a list.

Example

Write Simplest
Version of Code

```python
def my_sum(my_list):
    """ Compute sum of list elements. """
    answer = 0
    for item in my_list:
        answer = answer + item
    return answer
```

Example

Ensure Test Passes

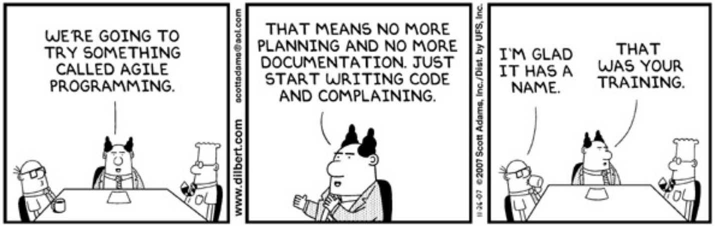
```
>>> my_sum([1,2,3])
6
```

Example

Writte Better
Version of Code

```python
def my_sum(my_list):
    """ Compute sum of list elements. """
    return sum(my_list)
```

Agile methods


WE'RE GOING TO TRY SOMETHING CALLED AGILE PROGRAMMING.
THAT MEANS NO MORE PLANNING AND NO MORE DOCUMENTATION. JUST START WRITING CODE AND COMPLAINING.
I'M GLAD IT HAS A NAME.
THAT WAS YOUR TRAINING.
www.dilbert.com
scottadams@aol.com
11-26-07 © 2007 Scott Adams, Inc./Dist. by UFS, Inc.

Whats Next

- Look at tools to support the agile workflow
- Better testing with **Unit Tests**
- Keeping track of changes and collaborating with **Version Control**
- Additional techniques

Outline

Unit Tests

Definition of a *Unit*

- The smallest testable piece of code
- Example: `my_sum`

- We wish to automate testing of our units
- In python we use the package `unittest`

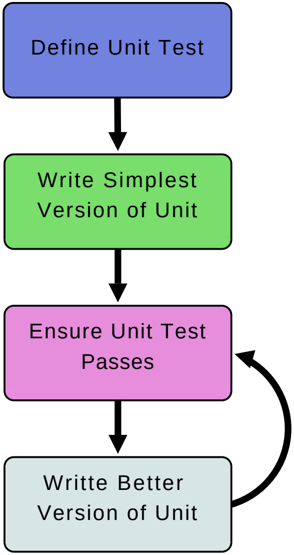
Example

```python
import unittest

def my_sum(my_list):
    """ Compute sum of list elements. """
    return sum(my_list)

class Test(unittest.TestCase):
    def test_my_sum(self):
        self.assertEqual(my_sum([1,2,3]),6)

if __name__ == "__main__":
    unittest.main()
```

Running the Example

```
% python example-test2.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

The Basic Agile Workflow - Reloaded



Goals

- check code works
- check design works
- catch regression

Benefits

- Easier to test the whole, if the units work
- Can modify parts, and be sure the rest still works
- Provide examples of how to use code

How to Test ?

- Test with simple cases, using hard coded solutions
  - `my_sum([1,2,3]) == 6`
- Test special or boundary cases
  - `my_sum([]) == 0`
- Test that meaningful error messages are raised upon corrupt input
  - `my_sum(['1', 'a'])`
  - $\rightarrow$ `TypeError: unsupported operand type(s) for +: 'int' and 'str'`

What Makes a Good Test?

- independent (of each other, and of user input)
- repeatable (i.e. deterministic)
- self-contained

Stuff Thats Harder to Test

Probabilistic code

- Use toy examples as validation
- Consider fixing the seed for your pseudo random number generator

Hardware

- use mock up software that behaves like the hardware should

Plots

- (any creative ideas welcome)

Test Suits

- All unit tests are collected into a test suite
- Execute the entire test suite with a single command
- Can be used to provide reports and statistics

Refactoring

This is what its called when you write a *better* version of your code.

- Re-organisation of your code without changing its function:
  - remove duplicates by creating functions and methods
  - increase modularity by breaking large code blocks into units
  - rename and restructure code to increase readability and reveal intention
- Always refactor one step at a time, and use the unit tests to check code still works
- Learn how to use automatic refactoring tools to make your life easier

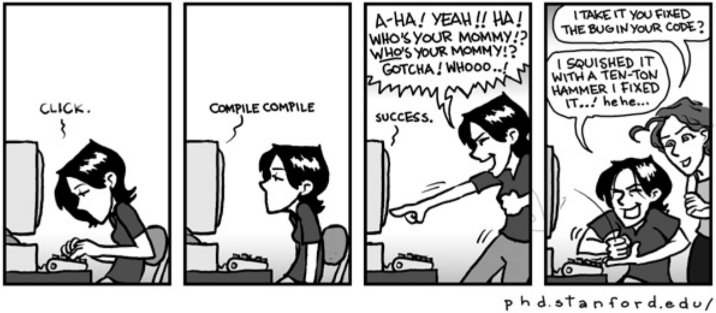
Dealing with Bugs

- Isolate the bug (using a debugger)
- Write a unit test to expose the bug
- Fix the code, and ensure the test passes
- Use the test to catch the bug should it reappear

Debugger

A program to run your code one step at a time, and giving you the ability to inspect its current state.

Dealing with Bugs


CLICK.
COMPILE COMPILE
SUCCESS.
A-HA! YEAH!! HA! WHO'S YOUR MOMMY!? WHO'S YOUR MOMMY!? GOTCHA! WHOOO..!
I TAKE IT YOU FIXED THE BUG IN YOUR CODE?
I SQUISHED IT WITH A TEN-TON HAMMER I FIXED IT..! hehe...
phd.stanford.edu/

Introducing New Features

- Split feature into units
- Use the agile workflow
- Tests drive the development
- Keep the iterations small

Some Last Thoughts

- Tests increase the confidence that your code works correctly, not only for yourself but also for your reviewers
- Tests are the only way to trust your code
- It might take you a while to get used to the idea, but it will pay off quite rapidly

- Questions?

Outline

What is Version Control?

Problem 1

"Help my code worked yesterday, but I can't recall what I changed!"

Problem 2

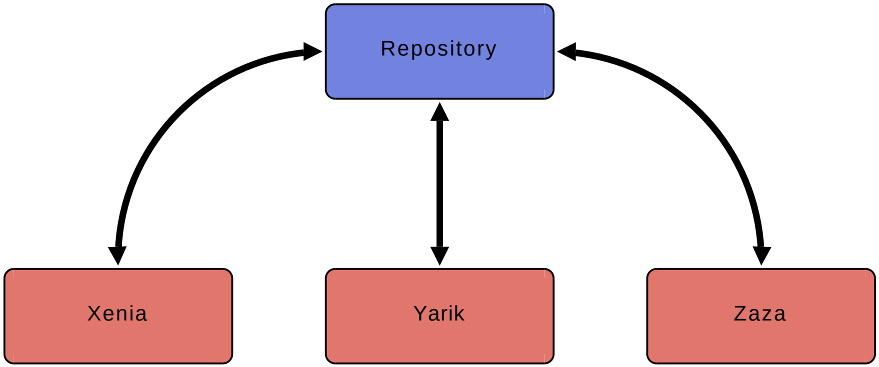
"We would like to work together, but we don't know how!"

- Version control is a method to track changes in source code
- Concurrent editing is possible via merging

Features

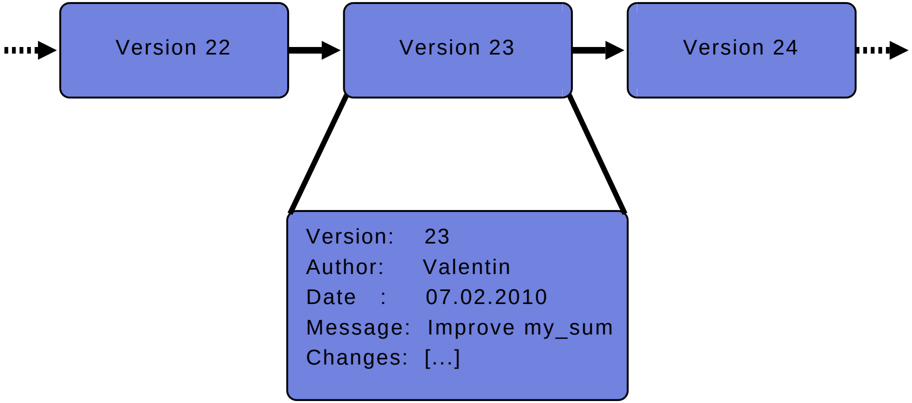
- Revert to previous versions
- Document developer effort
  - Who changed what, when and why?
- Easy collaboration across the globe

Where the Versions are Stored?



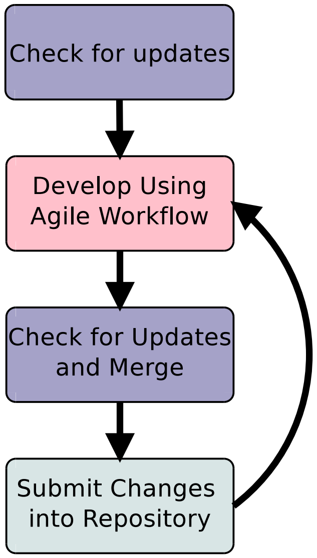
- **repository** is located on a server
- Developers must connect to this server

Contents of the Repository



Version: 23
Author: Valentin
Date : 07.02.2010
Message: Improve my_sum
Changes: [...]

Basic Version Control Workflow


Check for updates
Develop Using Agile Workflow
Check for Updates and Merge
Submit Changes into Repository

What Will We Use ?

- Many different systems available
- We will use the de-facto standard:

SUBVERSION

Some Last Thoughts

- Use version control for anything thats text
    - Code
    - Thesis
    - Letters
- We will be using **centralised** version control, note there exists also **decentralised** version control
- Again, it might take a while to get used to the idea, but it will pay off rapidly.
- Questions

Outline

Pair Programming

- Two developers, one computer
- Two roles: driver and navigator
- Driver sits at keyboard
- Navigator observes and instructs
- Switch roles every so often

Optimization for Speed

- Readable code is usually better than fast code
- Only optimize if its absolutely necessary
- Only optimize your bottlenecks
- ...and identify these using a profiler, for example **cprofile**

Profiler

A tool to measure and provide statistics on the execution time of code.

Prototyping

- If you are unsure how to implement something, write a prototype
- Hack together a proof of concept quickly
- No tests, no documentation
- Use this to explore the feasability of your idea
- When you are ready, scrap the prototype and start with the unit tests

Coding Style

- Give your variables meaningful names
- Adhere to coding conventions
- OR use a consistent style
- Use automated tools to ensure adherence: **pylint**

Documentation

- Minimum requirement: at least a docstring
- For a library document arguments and return objects
- Use tools to automatically generated website from code: **pydoc**

Results

- Every scientific result (especially if important) should be independently reproduced at least internally before publication. (German Research Council 1999)
- Increasing pressure to make the source used in publications available
- With unit tested code you need not be embarrassed to publish your code
- Using version control allows you to share and collaborate easily

The Last Slide

- Open source tools used to make this presentation:
  - wiki2beamer
  - LaTeXbeamer
  - dia

Questions ?