Unit Testing and Software Development Models

2 Dec 2010
CMPT140
Dr. Sean Ho
Trinity Western University

TRINITY WESTERN UNIVERSITY

Outline for today

- Unit testing (continued from last time)
  - doctest: test narratives
  - unittest (PyUnit), test suites
- Software development models
  - Waterfall
  - V
  - Agile (spiral)
    - XP (Extreme Programming)
    - Scrum

doctest: test narratives

See factorialtest.txt

- doctest searches your docstrings for text resembling a test script (e.g., '>>>')
- You can also put your test scripts in a separate file, interspersed amongst your documentation
  - As though the whole file were a docstring
- A test narrative is a document written for humans (e.g., user manual) where test cases are interleaved with the narrative
- Run tests: python -m doctest factorialtest.txt
- Or from code: doctest.testfile("factorialtest.txt")

unittest (PyUnit)

- doctest is quick and easy to use, but limited
- The unittest module provides more flexibility:
  - Organize test cases into suites
    (in separate classes and even separate files)
  - Fixtures: common setup / tear-down
    for all test cases in a suite
- Uses standard methodology from Java (JUnit)
- Test suites are classes (inherit from TestCase)
- Test cases are methods within a suite
  - Prefix method name with “test_”

unittest: test suites

See factorialunit.py

- Usually, put the test suites in separate files from the modules you are testing
- Import unittest and the module you want to test
- Create test suites as subclasses of TestCase:

```
class FactorialTests( unittest.TestCase ):
```

- Fixtures: define setUp() / tearDown() methods to be run before/after each test (separately)
- Test cases: define test_() methods
- Use self.assert() methods to check results:

```
self.assertEqual( factorial(6), 720 )
```

unittest: running tests

- Put the following at the end of the file of tests:

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

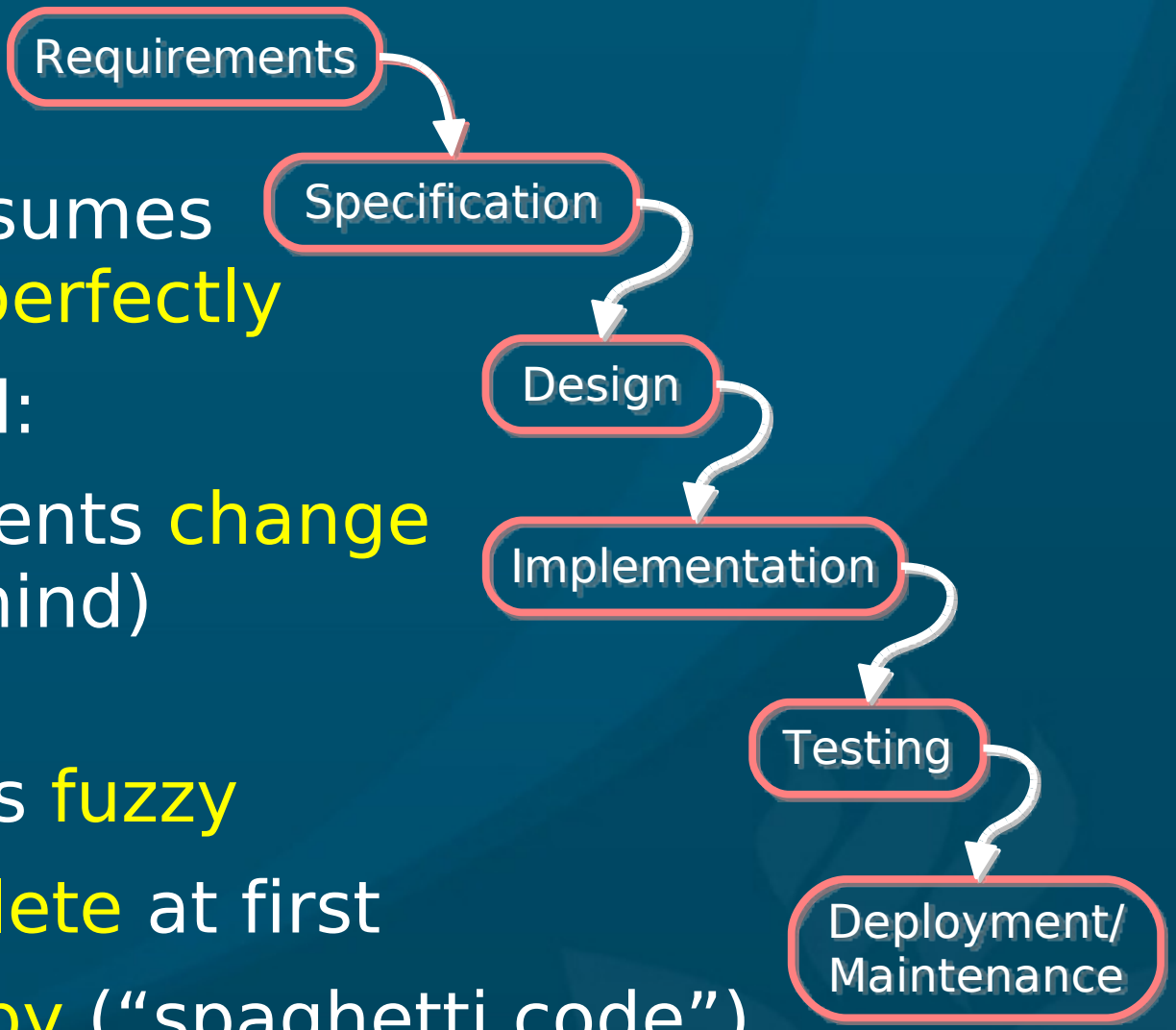
- Run from the command line:

```
python myunittests.py
```

- Outputs results and any failures
- Can also run from IDLE, but it will try to SystemExit after tests are finished

Top-down development

- WADES!
- Waterfall model: assumes every step is done perfectly
- But in the real world:
  - Written requirements change (client changes mind)
  - Apprehension of requirements is fuzzy
  - Design is incomplete at first
  - Execution is sloppy ("spaghetti code")
  - Scrutinization results in endless debugging!



Software development process

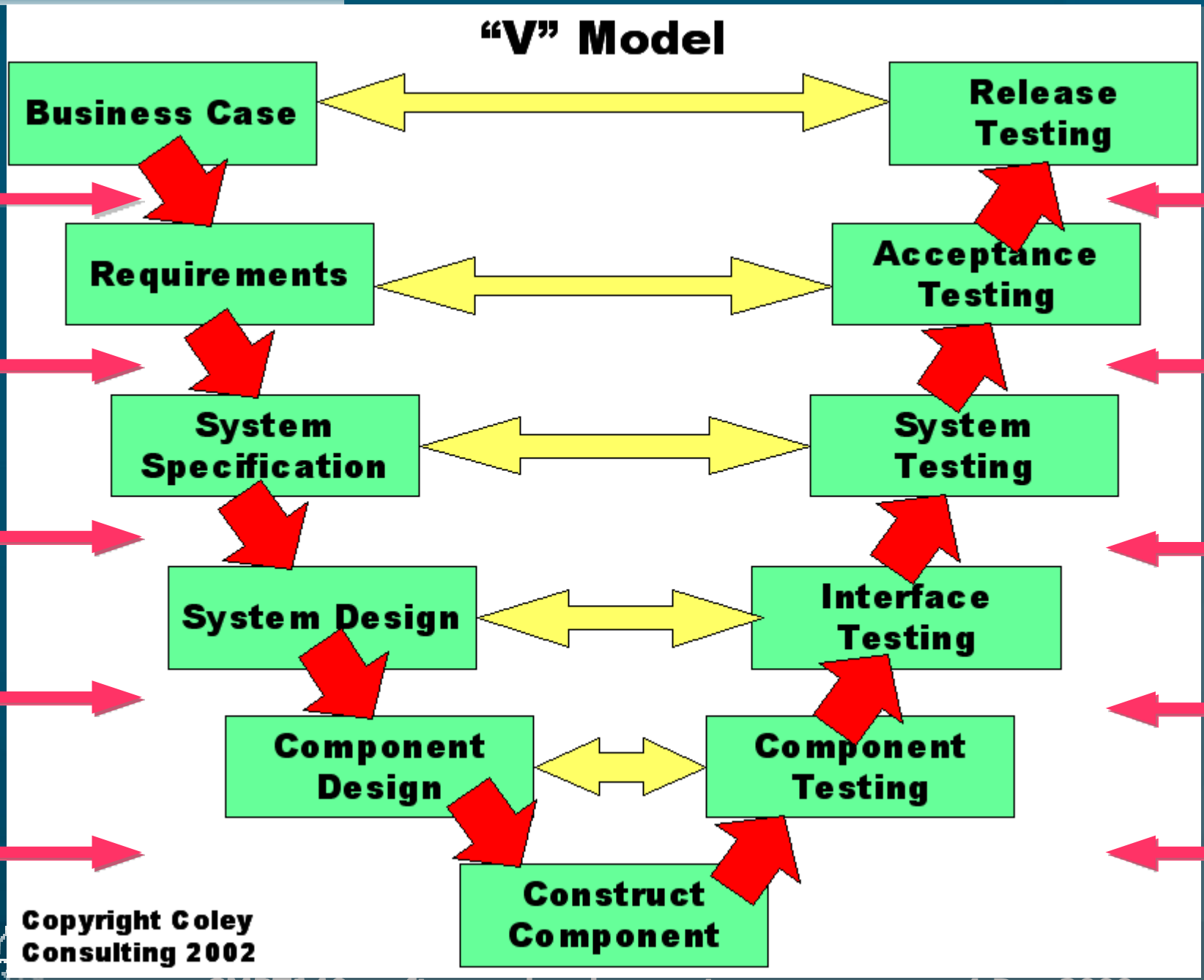
- Lots of people have tried to design better ways to develop that reflect the real world:
  - Development process: how you do the work
- No silver bullet: different projects, different people may require different processes
- Be flexible: your future employer may demand that you use a particular process
  - Or might not have any process in mind: then it's up to you to structure your time!
  - Get results; make the client happy!

V development model

- Design from Top-Down:
  - What does the client want? (requirements)
  - What will our system do? (specification)
  - How will we do it? (design)
  - What components will we need?
- Test from Bottom-Up:
  - Does each component work as it should?
  - Do the components integrate correctly?
  - Does it do what we promised it would?
  - Is the client happy?

V model (Coley)

gate



Agile development

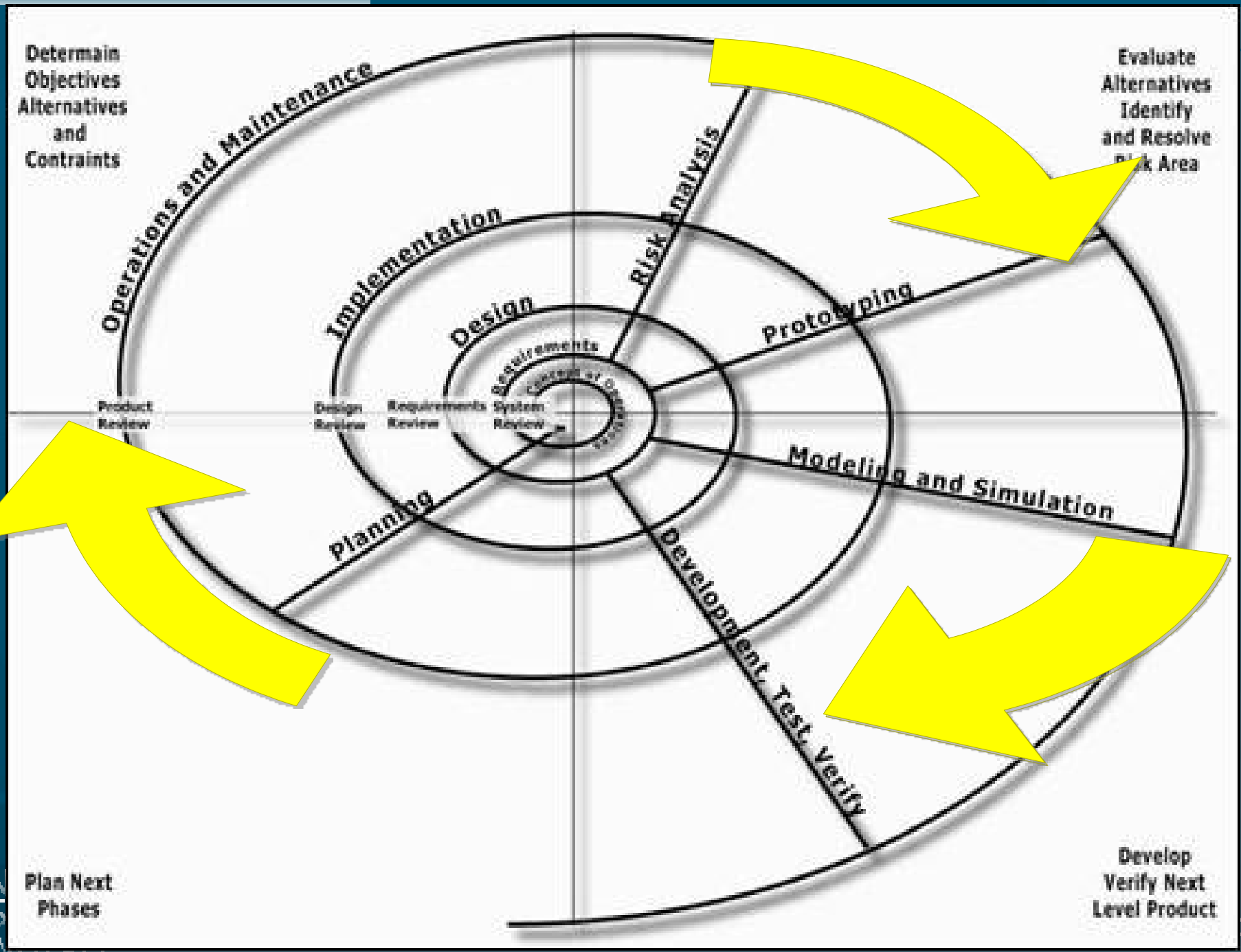
- Waterfall and V are very rigid, slow
- Agile refers to a broad class of methods:
  - Get results quickly and adapt to change
- "Agile Manifesto" philosophy:

Individuals and Interactions	Not: Software and Tools
Working Software	Not: Comprehensive Documentation
Customer Collaboration	Not: Contract Negotiation
Responding to Change	Not: Following a Plan

Agile: spiral model

- Agile methods follow a spiral process, like an iterative waterfall:
  - Repeat parts of WADES, refining as you go
- Get a prototype out early
- Get feedback early
  - Don't waste time developing what the client doesn't want
- Client: “I'll know it when I see it”
  - Developer: “Is this what you want?”
- Anticipate several cycles/refinements!

Agile: highway construc. (FHA)



Agile and the Toyota Way

- Agile does not mean total anarchy!
  - Clear goals, communication with client
  - Rapid development with frequent feedback
  - No room for procrastination!
- Agile influenced the "Toyota Way", lean process:
  - Have a long-term philosophy or goal
  - The right process
  - Invest in people
  - Continuously solve root problems

Agile: Extreme Programming

- Extreme programming (XP) was coined by Kent Beck in 1999 while making Chrysler's payroll sys.
- Many spirals, with varying scope and frequency
- Code: if there are two competing solutions, implement both!
  See which works better.
- Test: it's the only way to be sure it works
- Values: Communication, Simplicity, Feedback, Courage, and Respect

Agile: Scrum

- Most prevalent form of spiral dev. now is Scrum
- "Pig" Roles (committed):
  - Team (5-9 ppl): design, impl., test, comm., etc.
  - ScrumMaster: protect, keep Team on-task
  - Product Owner: "voice" of the client, writes use-cases ("stories", requirements), gives feedback on results to Team
- "Chicken" Roles (involved):
  - Client, stakeholders: business need, marketing, artistic vision, design studio, etc.

Scrum process

- Prioritized features go in backlog
- Divide backlog into sprints (1-4 wks)
- Sprint planning meetings
  - Choose features to tackle
- Daily stand-up scrum meetings
  - Time-boxed to 15min
  - Only "pigs" may speak
- Sprint review meetings
  - Get client feedback
  - Team feedback on process