UNIVERSITY of WASHINGTON

DATA 515A

Software Engineering for Data Scientists
Working in Teams

David Beck[1,2], Joseph Hellerstein[1,3], Jake VanderPlas[1,4]

Jay Garlapati[3]

[1]eScience Institute

[2]Chemical Engineering

[3]Computer Science Engineering

[4]Astronomy

The University of Washington

April 27, 2017

UNIVERSITY of WASHINGTON
eScience Institute

Agenda

- Software licenses
- Software development overview
- Project updates
- Team process with in-class exercises:
  - Code reviews (15 min)
  - Technology review (25 min)
  - Status standup (10 min)

Software Licenses

Overview of Software Licenses*

A software license is a legal instrument (usually by way of contract law, with or without printed material) governing the use or redistribution of software. Under United States copyright law all software is copyright protected, in source code as also object code form. The only exception is software in the public domain. A typical software license grants the licensee, typically an end-user, permission to use one or more copies of software in ways where such a use would otherwise potentially constitute copyright infringement of the software owner's exclusive rights under copyright law.

*Mark Webbink

Software Licenses*

Rights granted	Public domain	Non-protective FOSS license (e.g. BSD license)	Protective FOSS license (e.g. GPL)
Copyright retained	No	Yes	Yes
Right to perform	Yes	Yes	Yes
Right to display	Yes	Yes	Yes
Right to copy	Yes	Yes	Yes
Right to modify	Yes	Yes	Yes
Right to distribute	Yes	Yes, under same license	Yes, under same license
Right to sublicense	Yes	Yes	No
Example software	SQLite, ImageJ	Apache Webserver, ToyBox	Linux kernel, GIMP

FOSS = Free and Open Source Software

*Mark Webbink

Beck, Hellerstein & VanderPlas, 2016

Software Licenses*

Rights granted	Freeware/Shareware/ Freemium	Proprietary license	Trade secret
Copyright retained	Yes	Yes	Yes
Right to perform	Yes	Yes	No
Right to display	Yes	Yes	No
Right to copy	Often	No	No
Right to modify	No	No	No
Right to distribute	Often	No	No
Right to sublicense	No	No	No
Example software	Irfanview, Winamp	Windows, Half-Life 2	Server-side World of Warcraft

The default (no license): No rights are granted.

*Mark Webbink

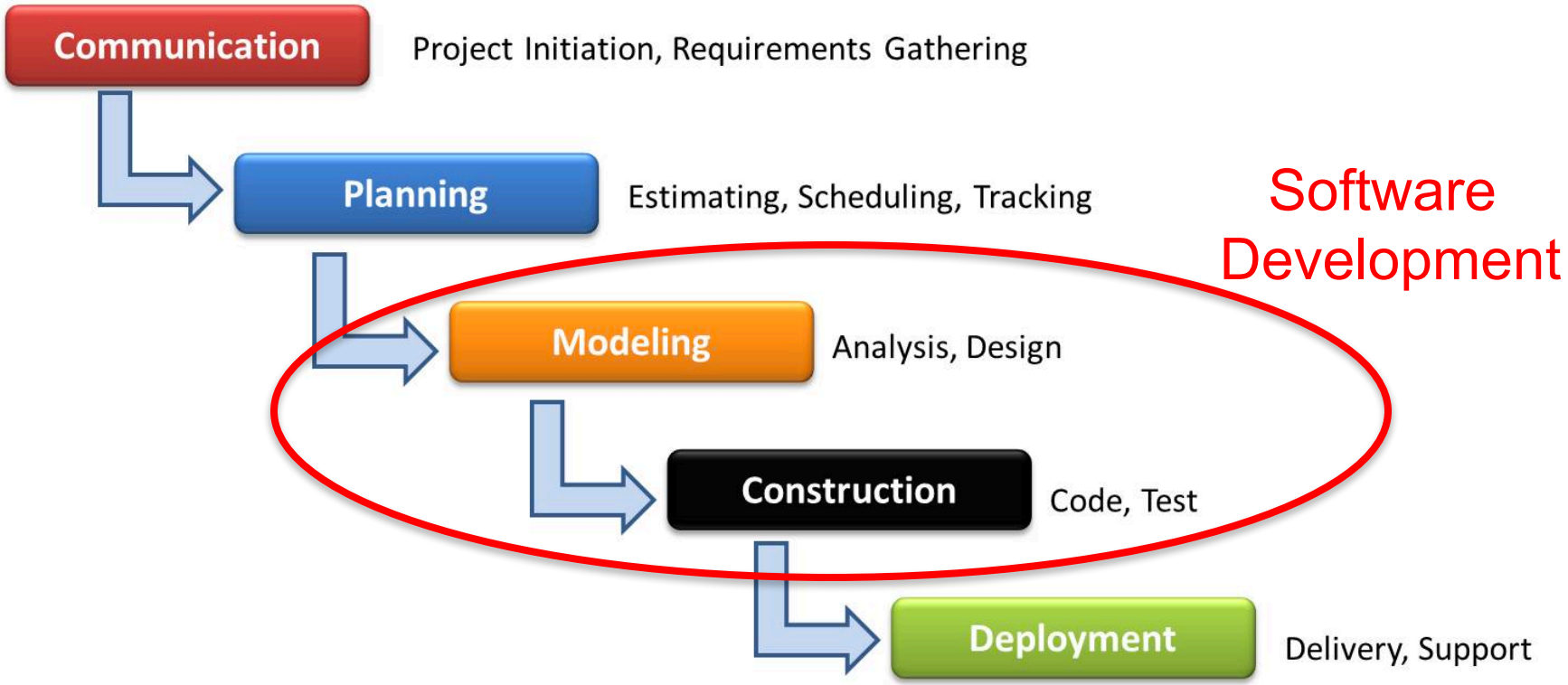
Beck, Hellerstein & VanderPlas, 2016

Software Development Overview

Software Development Phases

Beck, Hellerstein & VanderPlas, 2016

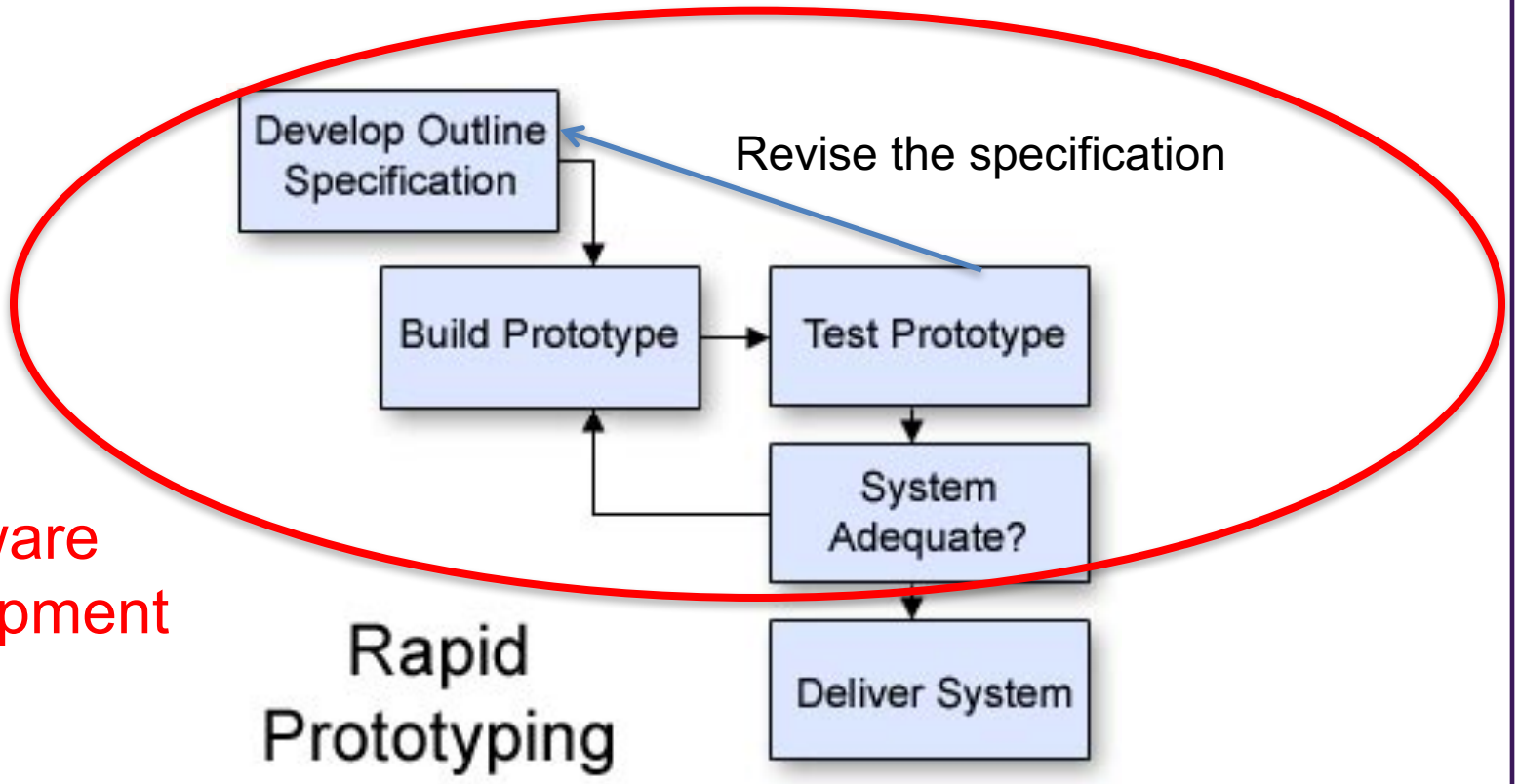
Waterfall Process Model



Why does this work poorly?

Rapid Prototyping

- Why?
  - Cannot specify all requirements in advance

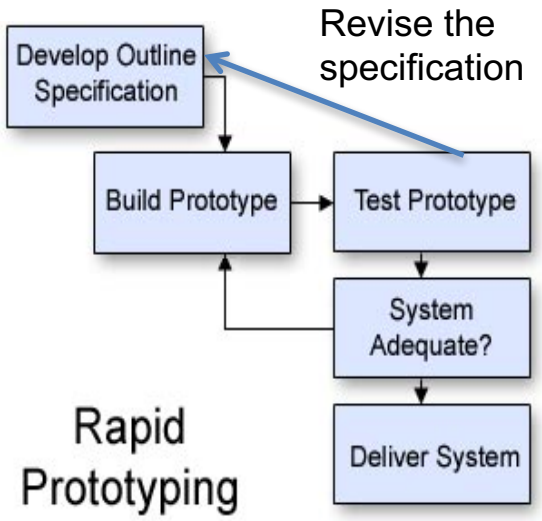


Beck, Hellerstein & VanderPlas, 2016

Team Process

Beck, Hellerstein & VanderPlas, 2016

Team Activities



- Reqs gathering (functional spec.)
- Design
  - Technology assessments
  - Write specifications
  - Review specification
- Implementation
  - Code
  - Code review
- Bug prioritization and resolution
- Standups (status update)

Beck, Hellerstein & VanderPlas, 2016

Projects

Project Updates

- What is your data?
  - You should have 2 datasets in hand!
- Who are your users?
  - General public? Scientists? Analysts?
- What questions are users trying to answer?
- What are the use cases (user-system interactions) to answer their questions?
- What issues are there ("known unknowns") with building your system?

Code Review

Beck, Hellerstein & VanderPlas, 2016

Code Review Template

- Why code review?
  - Improve code quality and find bugs

- Background
  - Describe what the application does
  - Describe the role of the code being reviewed
- Comment on
  - Choice of variable and function names
  - Readability of the code
  - How improve reuse and efficiency
  - How use existing python packages

Beck, Hellerstein & VanderPlas, 2016

*INTERCEPT

- Sci
  for
  –

```
1  import scipy.stats as ss
2  SLOPE, INTERCEPT, _, _, _ = ss.linregress(INV_S, INV_V)
3  SLOPE = np.round(SLOPE, 3)
4  INTERCEPT = np.round(INTERCEPT, 3)
5
6
7
8
9
10
```

Submit Cancel

row	S							
1	0.01	0.11	100.0	9.09	4.358	0.047	0.229	0.011
2	0.05	0.19	20.0	5.26				
3	0.12	0.21	8.33333333333	4.76				
4	0.2	0.22	5.0	4.55				
5	0.5	0.21	2.0	4.76				
6	1.0	0.24	1.0	4.17				

`makeEvaluationScriptProgram()`

- Generates python code that evaluates the formulas associated with each column in the spreadsheet
- No assumption about order of evaluation of the columns

```
class ProgramGenerator(object):
  """
  Evaluates and otherwise processes formulas in a table.
  Returns a program as a string.
  The following programs are created.
    a) Formula execution program. For table evaluation,
       this is a script. For table export, this is a function.
    b) Unittest program for table export.
  Formula execution programs are structured into sections. For
  table evaluation this consists of:
    1. Prologue statements: executed first and only once.
       Note that the APIFormulas object is created by the
       script runner.
    2. Variable assignment statements that assign column values
       to variables used in the script.
    3. Formula evaluation blocks, one for each formula.
    4. Checking for termination of the formula evaluation loop
  Exported functions have the following structure:
    1. Prologue statements. This includes the creation of the
       Plugin API object.
    2. Function header (def statement)
    3. Variable assignment statements (but not for the
       function inputs).
    4. Checking for termination of the formula evaluation loop
    5. Return statement
  In addition, a test program is created that calls the exported function
  and verifies that the function produces the output columns in
  the table when it is called with the input columns.
  Code generated for these blocks often makes use of the
  api object. For table evaluation, this has the class APIFormulas.
  For exported functions, this has the class APIPlugin.
  """
```

```
33
34   def makeEvaluationScriptProgram(self, create_API_object=False):
35     """
36     Creates a python script that evaluates the table formulas
37     when there is a change to the scisheet
38     :param bool create_API_object: True means that code will be generated
39                                    that creates the API object.
40     :return str program: Program as a string
41     """
42     sa = StatementAccumulator()
43     # TODO: This won't work with nested columns
44     statement = '''# Evaluation of the table %s.
45
46     ''' % self._table.getName(is_global_name=False)
47     sa.add(statement)
48     sa.add(self._makeAPIInitializationStatements(
49         create_API_object=create_API_object))
50     sa.add("try:")
51     sa.indent(1)
52     sa.add(self._makePrologue())
53     sa.add(self._makeFormulaEvaluationStatements())
54     statement = "if %s.controller.getException() is None:"  \
55         % API_OBJECT
56     sa.add(statement)
57     sa.indent(1)
58     sa.add(self._makeEpilogue())
59     sa.indent(-2)
60     # Handle exceptions in the Prologue and Epilogue
61     statement = """
62 except Exception as exc:
63   %s.controller.exceptionForBlock(exc)""" % API_OBJECT
64     sa.add(statement)
65     sa.add(self._makeClosingException(is_absolute_linenumber=False))
66     return sa.get()
```

Beck, Hellerstein & VanderPlas, 2016

Team Exercise: 15 minutes

- Team Breakout (10 min)
  - Review `makeEvaluationScriptProgram()` based on code review template
- Breakout reports (5 min)

Technology Review

Technology Review Template

- Why technology reviews?
  - Determine if use a package

- Background
  - Requirements that indicate a need for the proposed package
- Discuss
  - How the package works
  - Appeal of using the package
  - Drawbacks of using the package

Example of A Technology Review

Antimony Package for Kinetics Modeling

- Background
  - Need kinetics models to explore certain what-if questions in chemical systems.

Using Antimony

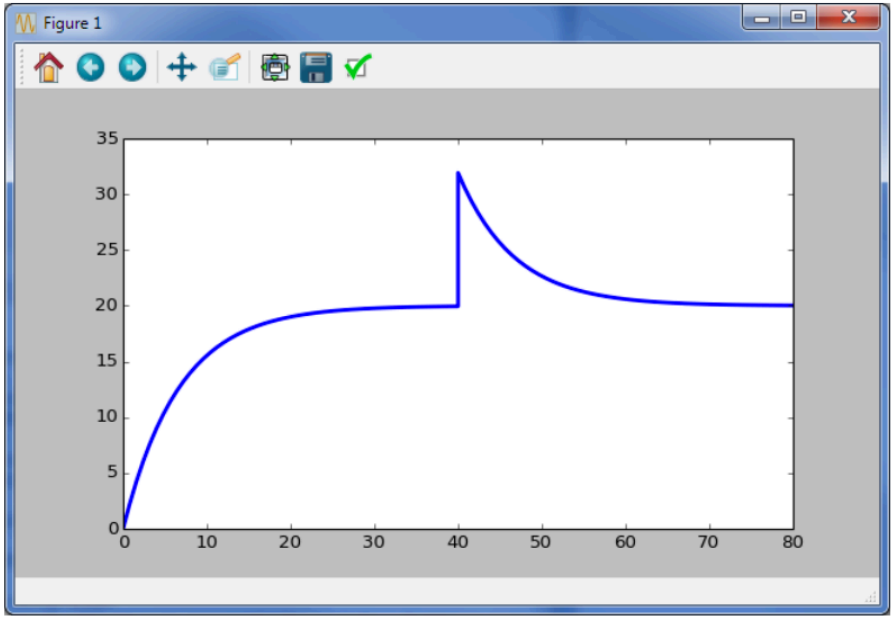
```
import numpy # Required for vstack
import tellurium as te

rr = te.loada ('''
     $Xo -> S1;  k1*Xo;
      S1 -> $X1; k2*S1;

     Xo = 10; k1 = 0.3; k2 = 0.15;
''')

m1 = rr.simulate (0, 40, 50)
```

Kinetics model is a python string



Perturbation of *S1*

Assessment of Antimony

- Appeal
  - Readable kinetics models
  - Can use python with Antimony
  - Exports to and imports from SBML (systems biology modeling language)
- Drawbacks
  - Poor support for the package
  - Scaling may be a problem

Team Exercise: 25 minutes

- Team breakouts (20 min)
  - Identify a technology that may use in project
    - Pick within 5 minutes
  - Complete the technology review template
- Breakout reports (5 min)

Standup Template

- Why standups?
  - Communicate status and actions within and between teams
- Should be presented in 1-2 minutes

- Progress this period
  - How it compares with the plan
  - If behind plan, how compensate to make plan end date
- Deliverables for next period
- Challenges to making next deliverables such as:
  - Technology uncertainties and blockers
  - Team issues

Team Exercise: 10 minutes

- Team breakouts (5 min)
  - Draft a standup report
- Breakout reports (5 min)