Quality Driven Development

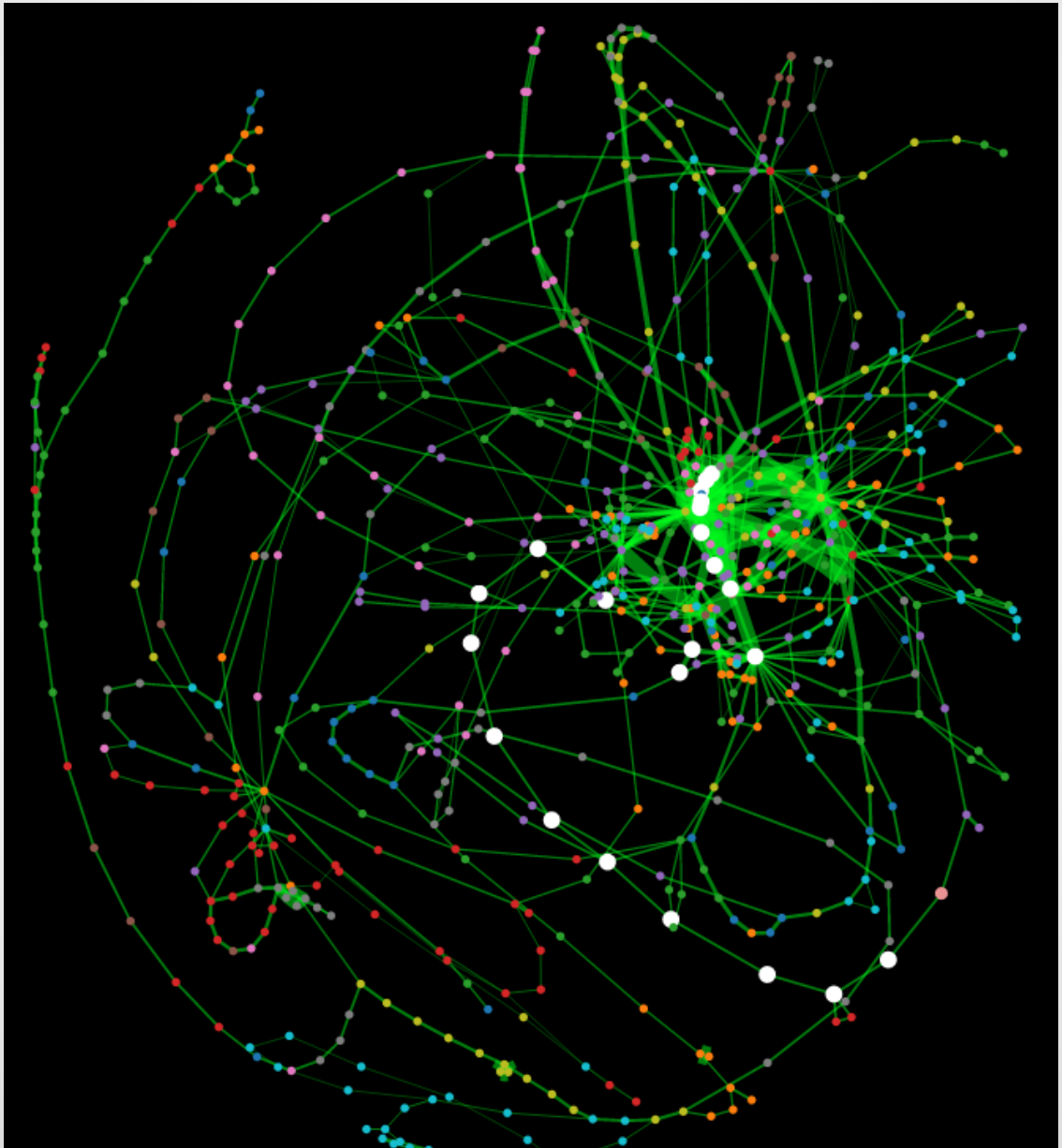
Interacting with BDD/TDD

Hadar Ziv and Vijay Krishna Palepu
Department of Informatics
University of California, Irvine

Brief History (Vijay Krishna Palepu)

- PhD student in software engineering at UCI.
  - Always thought that debugging took too much time; needs to be improved!
  - Spider Lab, UCI.
- Actively researching Software program analysis and visualization.
  - Palepu, Xu, Jones, “Improving Efficiency of Dynamic Analysis with Dynamic Dependence Summaries,” 2013 International Conference on Automated Software Engineering.
  - Palepu, Jones, “Visualizing Constituent Behaviors within Executions,” 2013 International Working Conference on Software Visualization.
- Working on a software visualization project called “The Brain”.
- Worked as a software engineer in a startup for 11 months.
- Computer Engineer with Distinction, University of Pune, India.
  - Bronze Medal, AWES Scholarship, youngest Student Council Member.

The Brain



Brief History (Hadar Ziv)

- PhD, UC Irvine, coined the "Uncertainty Principle in Software Engineering"
  - I was just inspired by Heisenberg and Heisenbugs…
  - About 50 refereed publications, many research collaborations, since…
- Rational-certified trainer in OOAD/UML, RUP, Rose…
- Taught many both in academia and industry
  - Capstone design project for Informatics seniors, game-design seniors starting in 2014
  - 2003 UCI Excellence in Teaching award
  - Fall 2013, 250 students in Introduction to Software Eng…
- Many successful training and consulting engagements
  - TitlePoint by Property Insight (Fidelity National Title)
    - Requirements 2003, introduced 2004, alive and well and well used…
  - Unity for medical-device product-families by St. Jude Medical
    - Requirements 2004-2007, introduced Sept 2007, alive and well…
- Among 28 who signed the "New Deal" for software development, 2013
  - Along with Grady Booch, Philippe Kruchten, Scott Ambler, Walker Royce…

Ziv's Law

Designer signing for your home.

Scrum Log Jeff Sutherland: Origins of Scrum
scrum.jeffsutherland.com/2007/07/origins-of-scrum.html
Jul 5, 2007 - **Ziv's law** - specifications will never be fully understood. Humphrey's law - the user will never know what they want until after the system is in ...

Ten Year Agile Retrospective: How We Can Improve In The Next ...
msdn.microsoft.com/en-us/library/hh350860(v=vs.100).aspx
Few people are aware of **Ziv's Law**, that software development is unpredictable[11]. The failure rate on projects worldwide is over 65%, largely due to lack of ...
You've visited this page 4 times. Last visit: 7/29/13

Some "**Laws**" of Software Development - Simple Talk
https://www.simple-talk.com/.../some-**laws**-of-software-development/
May 17, 2013 - **Ziv's law** states that software development is unpredictable and that specifications and requirements will never be fully understood.

Twitter / ScrumTurkey: **Ziv's Law**: Software Development ...
https://twitter.com/ScrumTurkey/status/324403310868189184
Apr 16, 2013 - **Ziv's Law**: Software Development is Inherently Unpredictable #agile #scrum #lean #kanban. Reply; Retweet Retweeted; Delete; Favorite ...

Agile Is the New Waterfall - Slideshare
www.slideshare.net/nashjain/agile-is-the-new-waterfall
Apr 29, 2009 - Licensed Under Creative Commons by Naresh Jain Thursday, April 30, 2009 30; Agile is Designed to deal with **Ziv's law** - specifications will ...

Its Not Just A Good Idea, Its The **Law** - Techno-Man!
www.techno-man.net/.../In-which-our-process-challenged-marsupial-lear...
May 28, 2013 - In the intro, Jeff talks about some of the accepted laws within the scrum process, specifically Conway, Humphrey and **Ziv's laws. And while** ...

Lessons Learned (from my Brief History)

- When giving a public presentation...
  - Do not give a brief history since time immemorial
  - Do not (re)define and (re)visit well-known terms
  - Do not define new terms
  - Do not show code, or give hands-on assignments
  - Whatever you do, do not mention Waterfall
  - Whatever you do, do not criticize Agile
  - Whatever you do, do not criticize the Government
- Therefore…

Presentation Outline

- Revisit well-known terms and well-known software “history”
  - Software bloat
  - Agile, Testing
  - Mention Waterfall
- Introduce and define new terms
  - TDD, BDD
  - Mention Waterfall
- Show code
- Manage interactive ‘audience participation’
- Constructive criticisms

In the beginning… (Brief History of Software)

- 1969: Software Engineering/The software “crisis”
- 1970: The Waterfall Model (Royce)
- 1975: The Mythical Man-Month (Brooks’ Law)
- 1980: The Spiral Model (Boehm)
- 1986: No Silver Bullet (Brooks)
- 1980s object-oriented design/programming
- 1990s OOAD/UML, Java
- 1999: XP (Kent Beck, et al)
- 2001: The Agile Manifesto, Agile Modeling, ...
- 1997: Software is a Gas (Myhrvold’s First Law)
  - http://www.informationisbeautiful.net/visualizations/million-lines-of-code/

October 1, 2013

- “Spoken to any software developers about HealthCare.gov?
- It seems that every conceivable principle of software development was and is being violated
- It is the ultimate source of compelling illustrations of what not to do
- Whether the issues are requirements, design, testing, deployment, or management, it is replete with anti-patterns
- From the oldest lessons captured in the Mythical Man-Month to the latest best practices in website architecture, HealthCare.gov seemingly has gone the opposite direction”

(Prof. Taylor, UCI Department of Informatics, ISR Newsletter)

Agile Development

“Problem Statement”

- Problem: Accommodating shorter and shorter business cycles
- Long software projects…
  - (one year, two years, or longer…)
  - Exceed budget, blow through schedule, deliver something less than desirable (if at all)
  - Usually end up with a ‘waterfall’ or function-driven process
  - From functional specs…
    - Usually in text format, ‘shall’ statements, or use cases
  - To design to code to test to deployment

Agile Solves the Problem

- As if that wasn't enough…
  - The pace of technological advance and human expectations…
  - Makes it less and less likely that a multi-year project will ever succeed
- Hence
  - Most "agile" projects are 90, 120, or 180 days
  - Iterative and Incremental Development (IID)
  - Longer than 180 days is considered "high risk"
- Developers like IID. Every two weeks they get
  - Closure/satisfaction/sense of doneness
  - Something new to work on
- Managers like it too
- Clients like it too
- "Amazing yet true"... Who said this?!?

But… Not so Fast…

- Agile works well in the micro (small team) level
  - Not so clear for large teams, macro-level projects
- Someone smart warned about “No Silver Bullet”…
- Someone wrote “agile” statements before Agile was invented…
  - FDD: Feature Driven Development
  - Others…

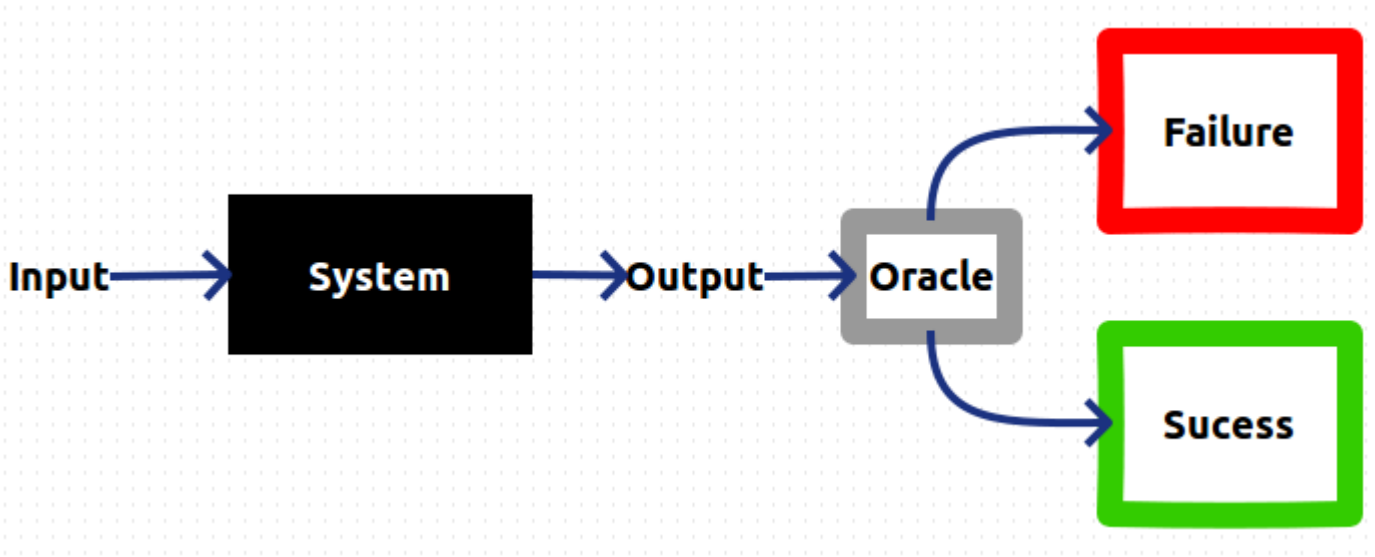
So… What is Agile… after all

- Sure, the manifesto
  - Individuals and interactions over processes and tools, etc.
- Sure, the principles
  - Valuable software, deliver frequently, continuous integration and delivery, motivated individuals working together, etc.
- *Iterative and Incremental Development and Delivery*
- A process framework or “philosophy”
  - Instantiated by specific methods, e.g., SCRUM, Kanban, Lean
  - Iterations, sprints, product backlog, burndown chart, etc.
  - Micro vs. Macro, Scalability issues
- Be Sure to Remember…
  - No Silver Bullet
  - Evolutionary, not Revolutionary

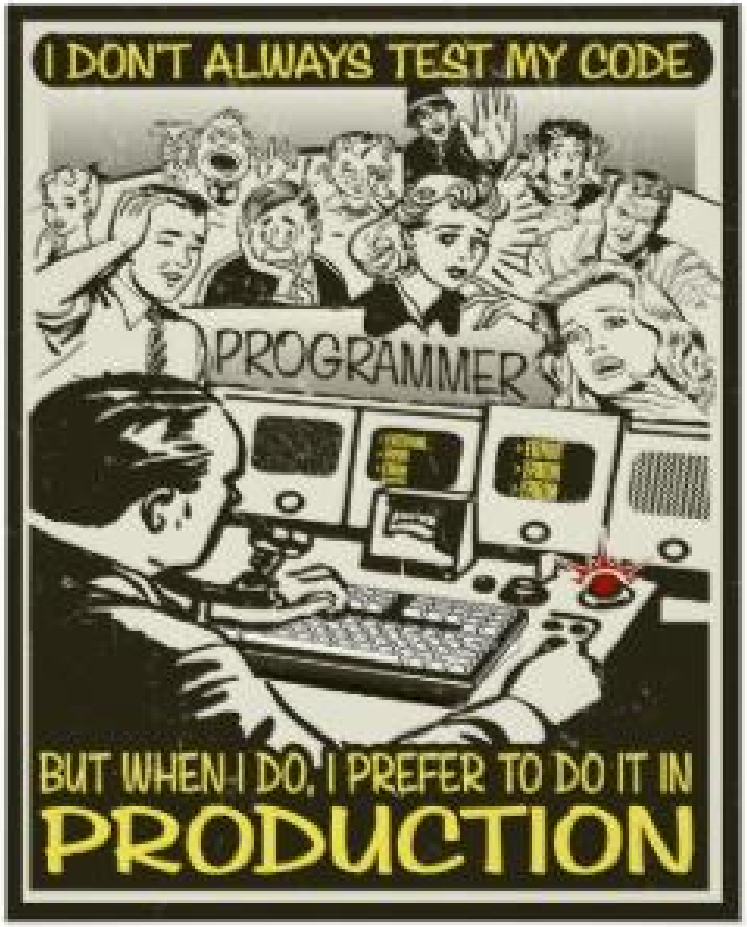
Software Testing

Software Testing In a Nutshell

- Part of Quality Assurance
- Expected Behavior vs. Actual Behavior



“Testing proves the presence of bugs, not absence” - Dijkstra

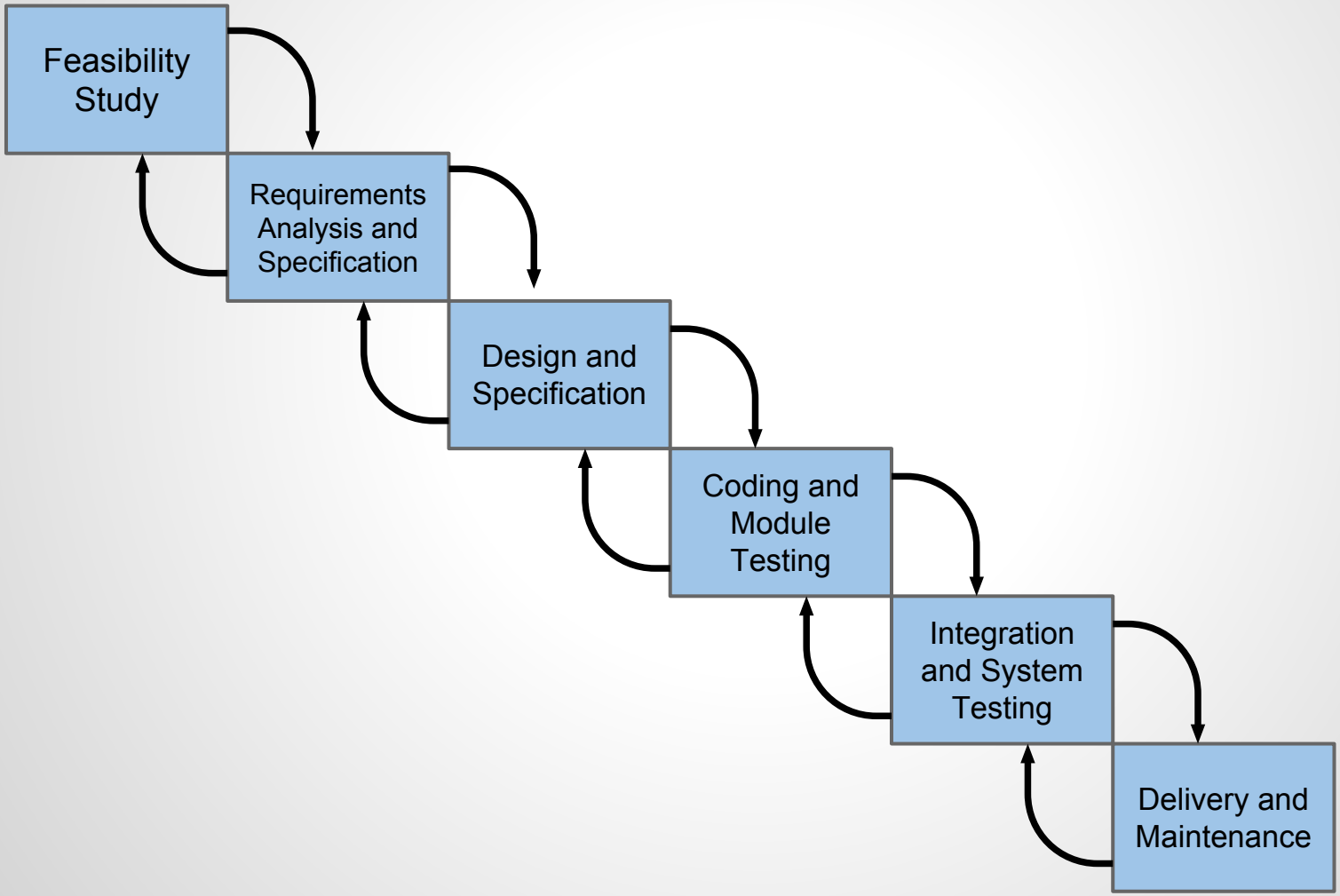


Testing is about confidence.

Testing is about the people.

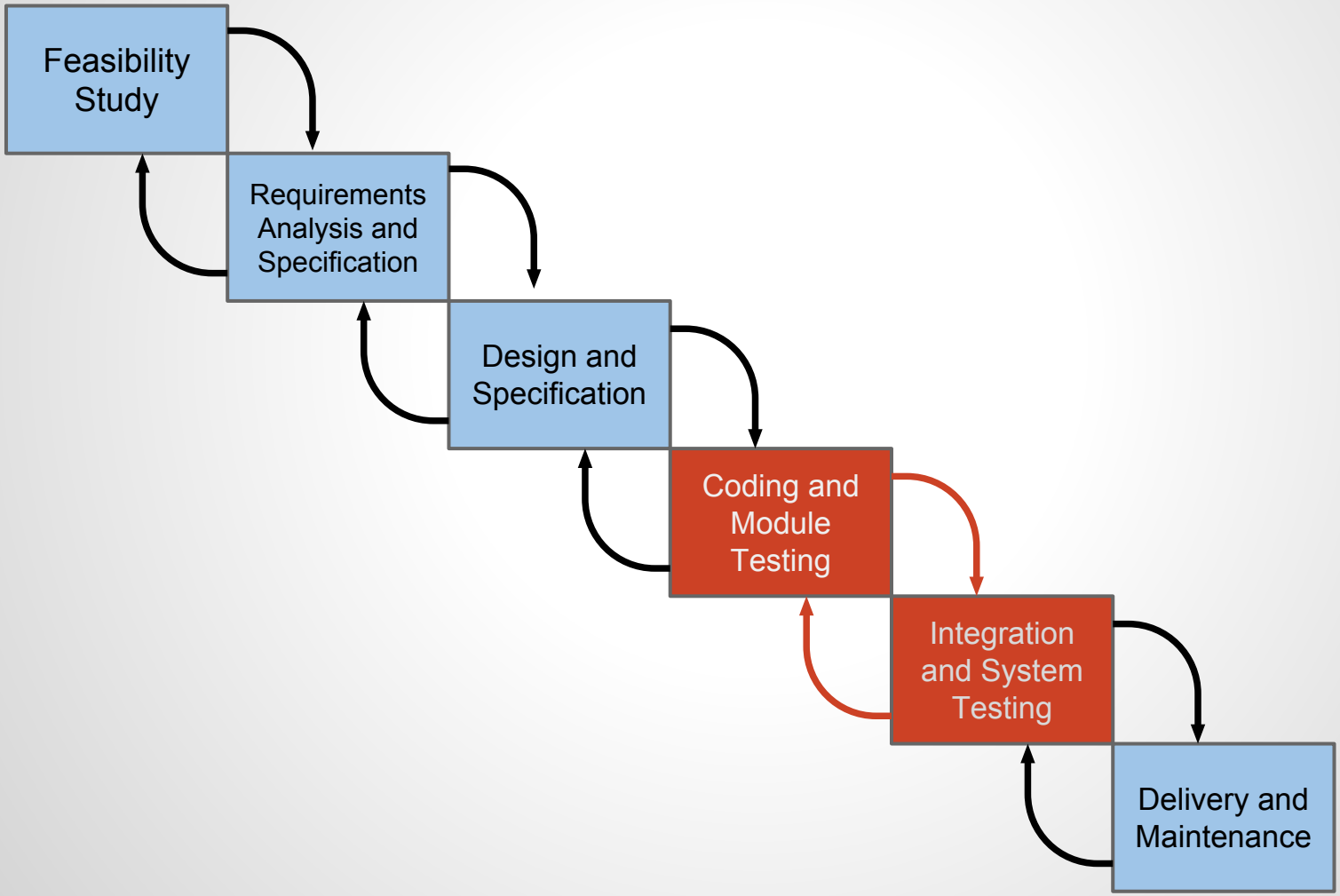
D-tour: Process Models

In the beginning …



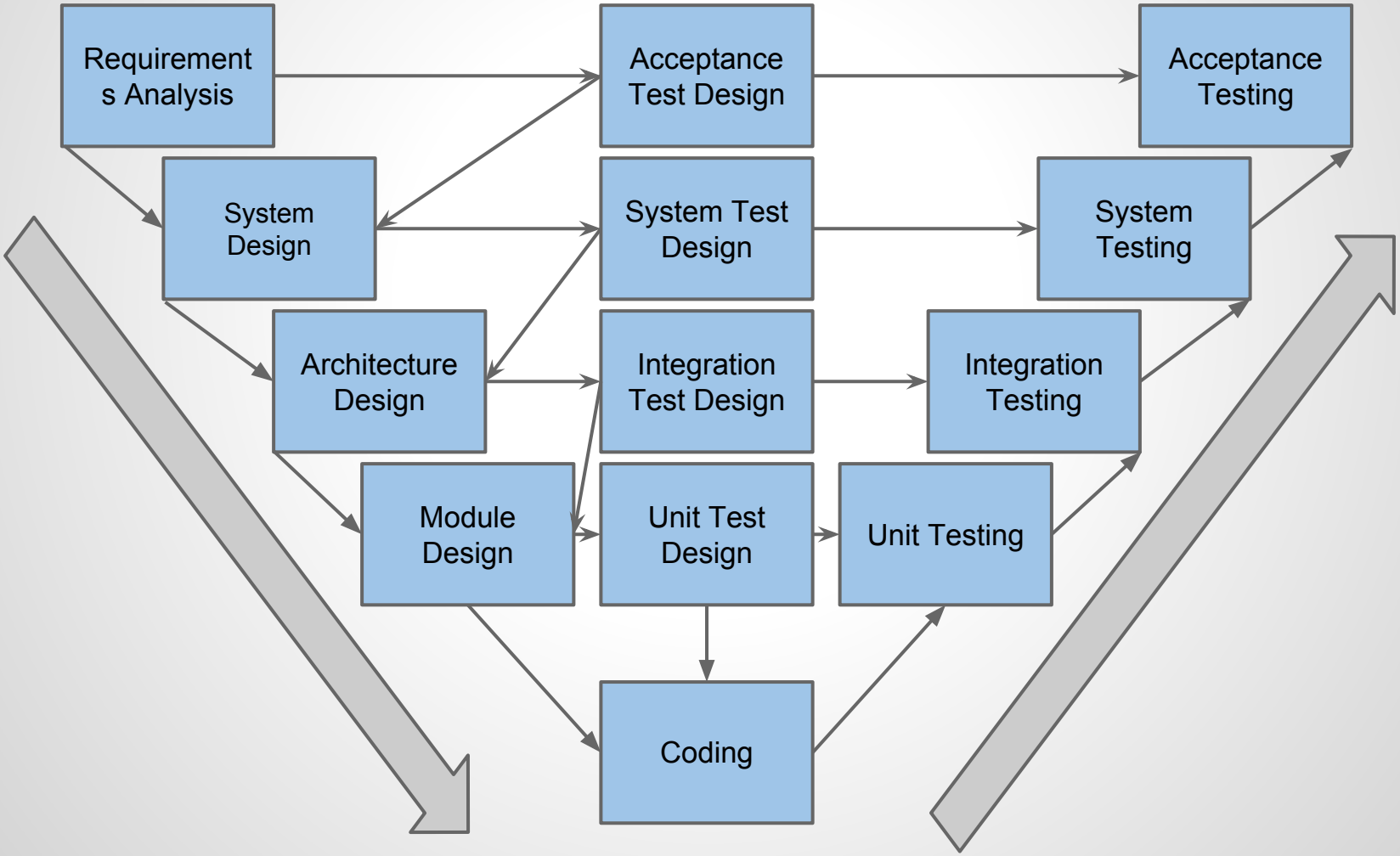
The **Waterfall Model** of Software Development (Royce 1970)

In the beginning …

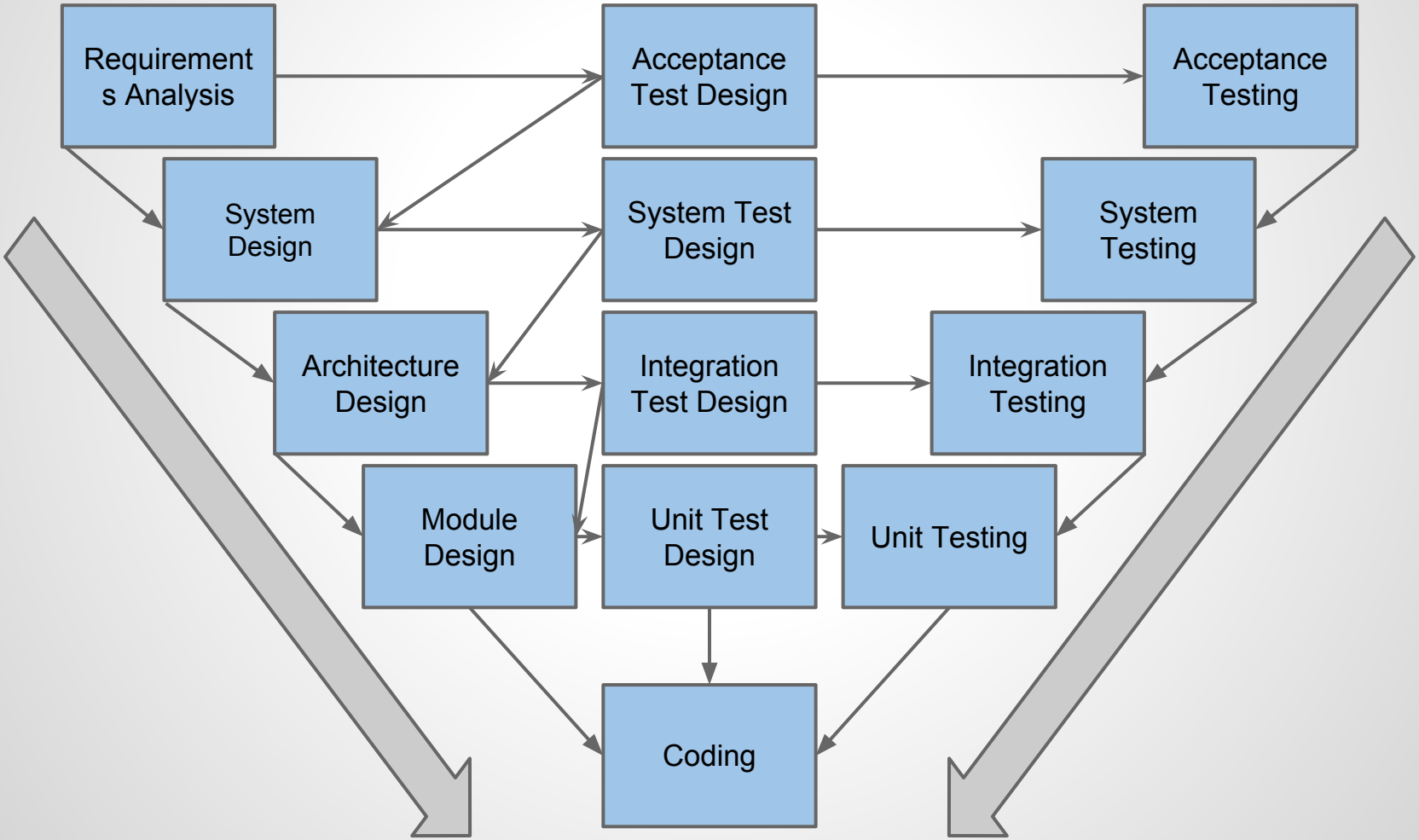


The **Waterfall Model** of Software Development (Royce 1970)

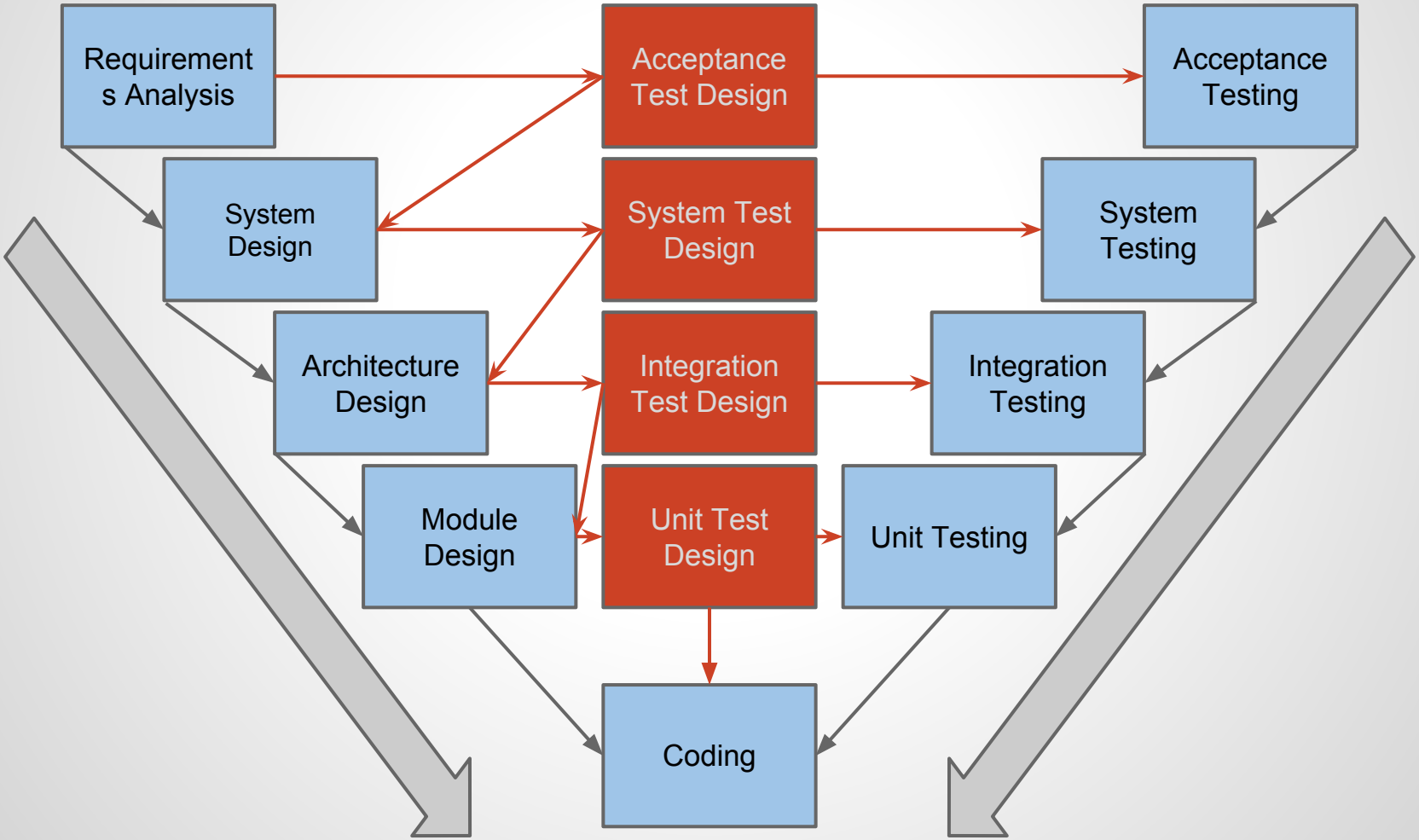
V-Model


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

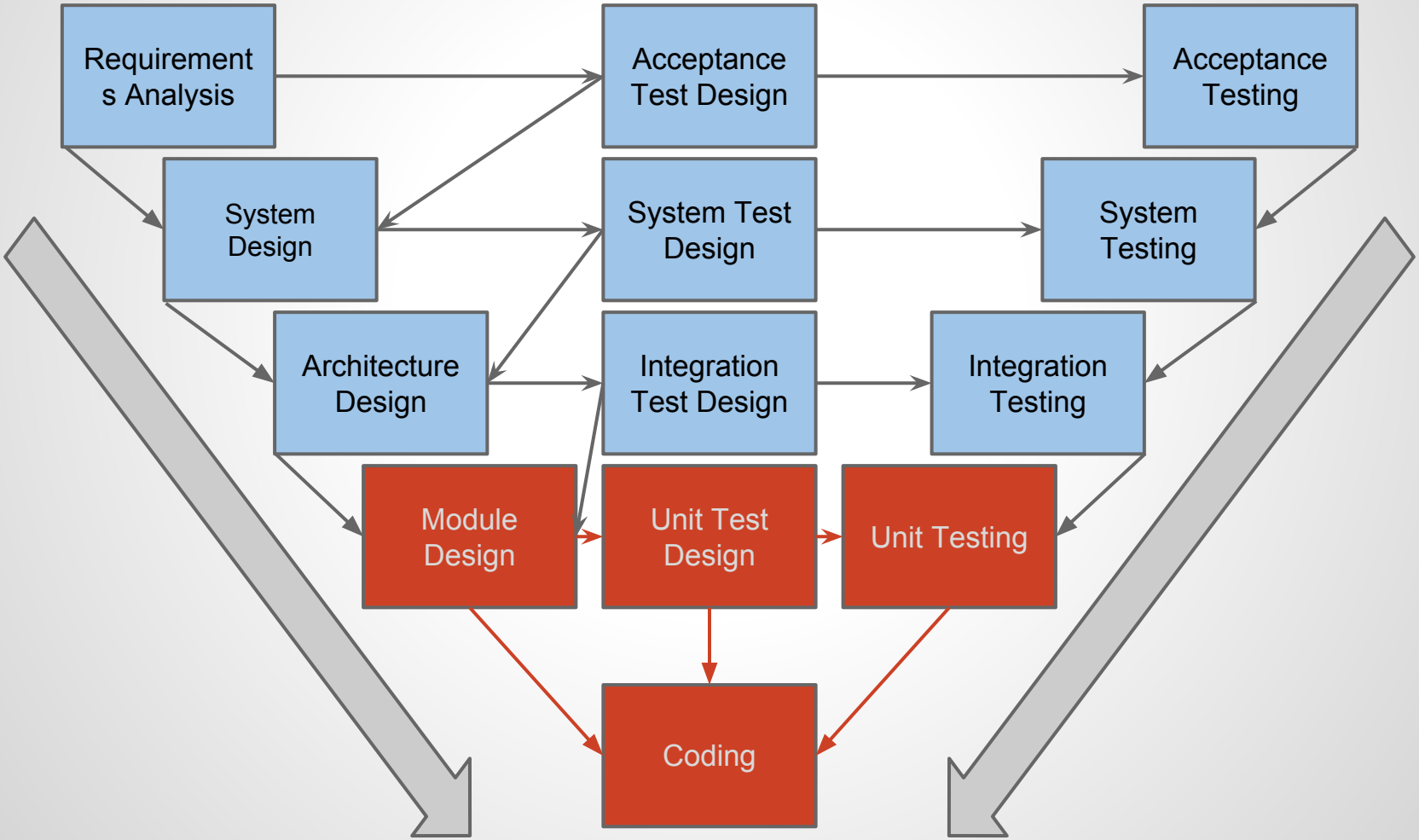
V-Model


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

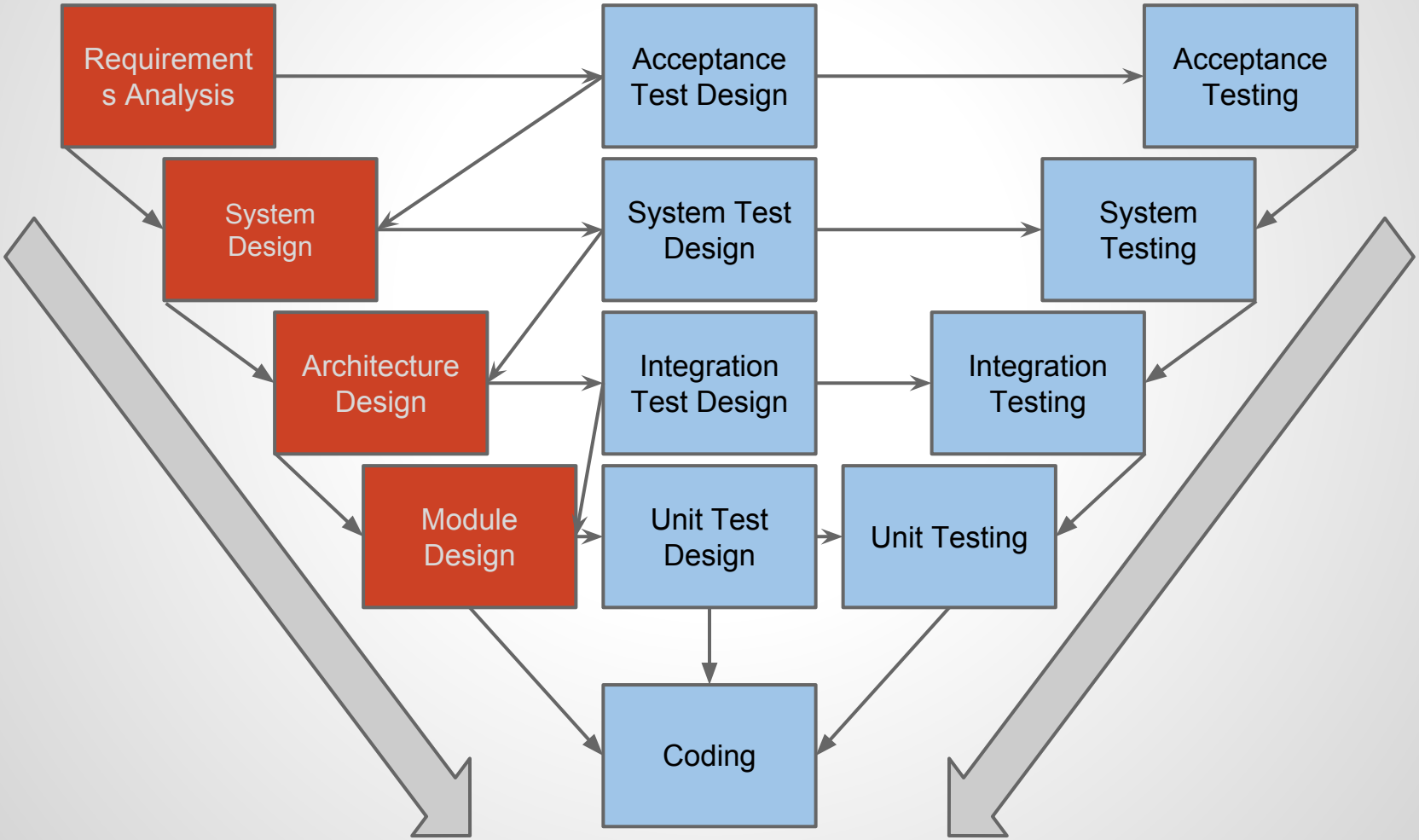
V-Model


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

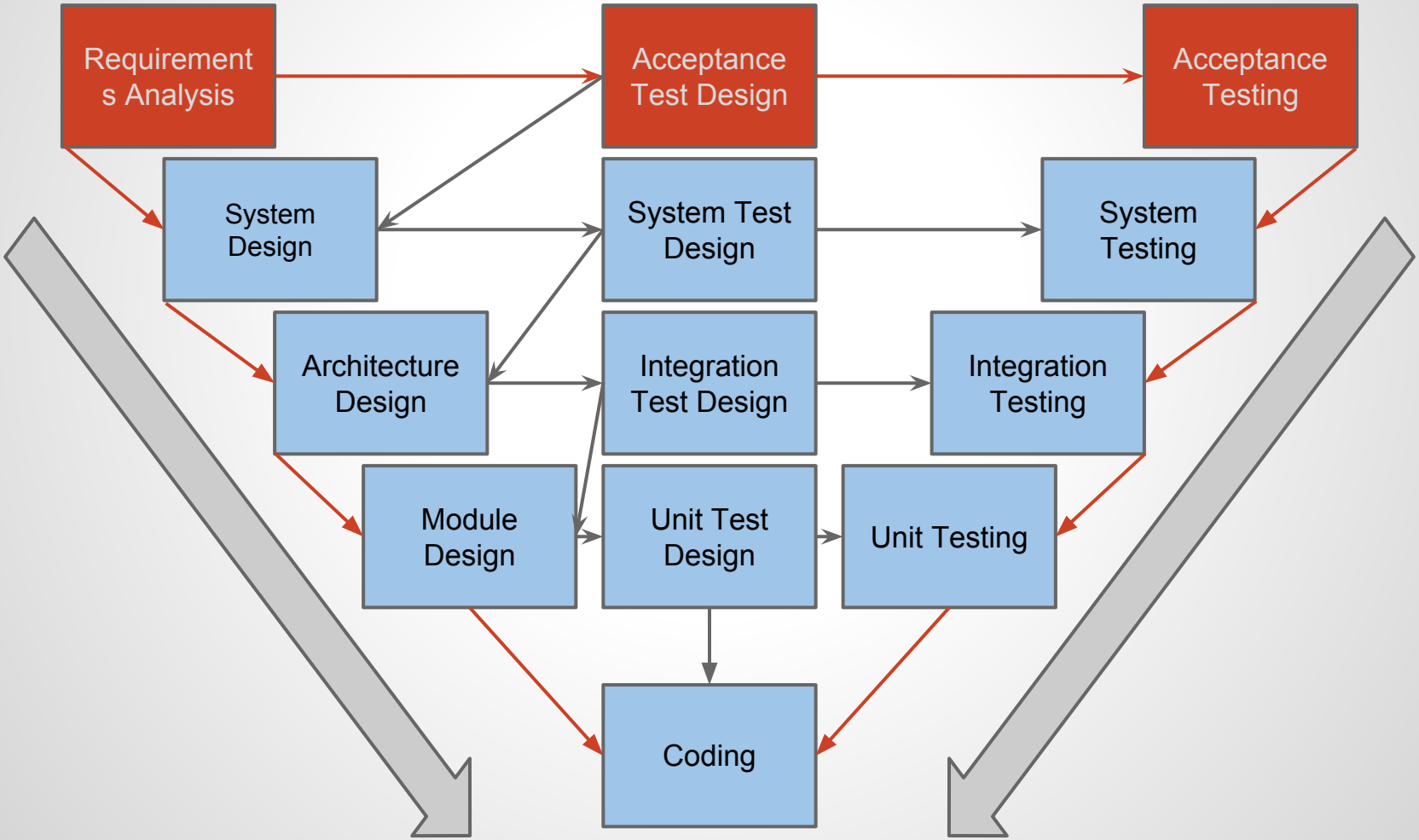
TDD


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

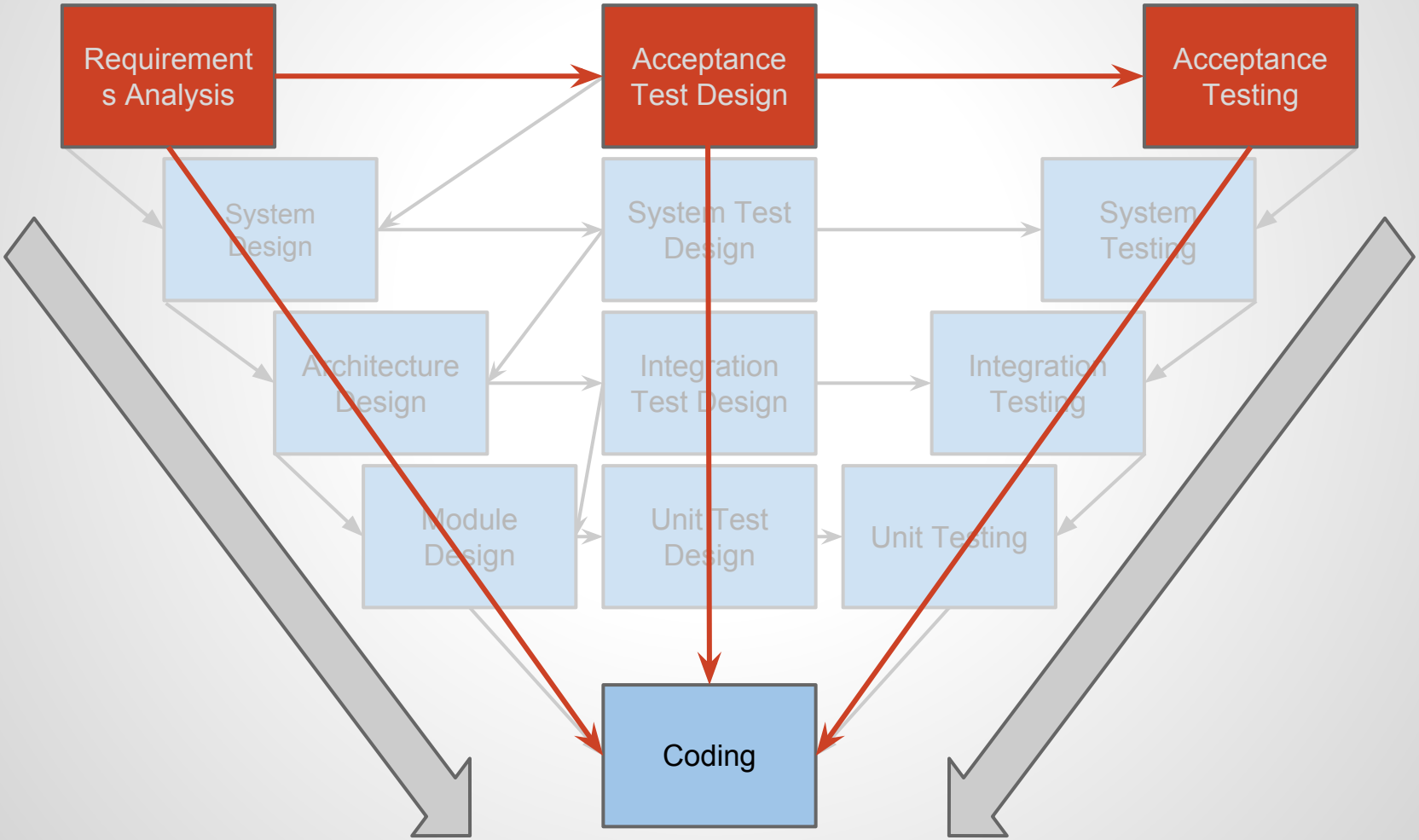
Behavior Modeling



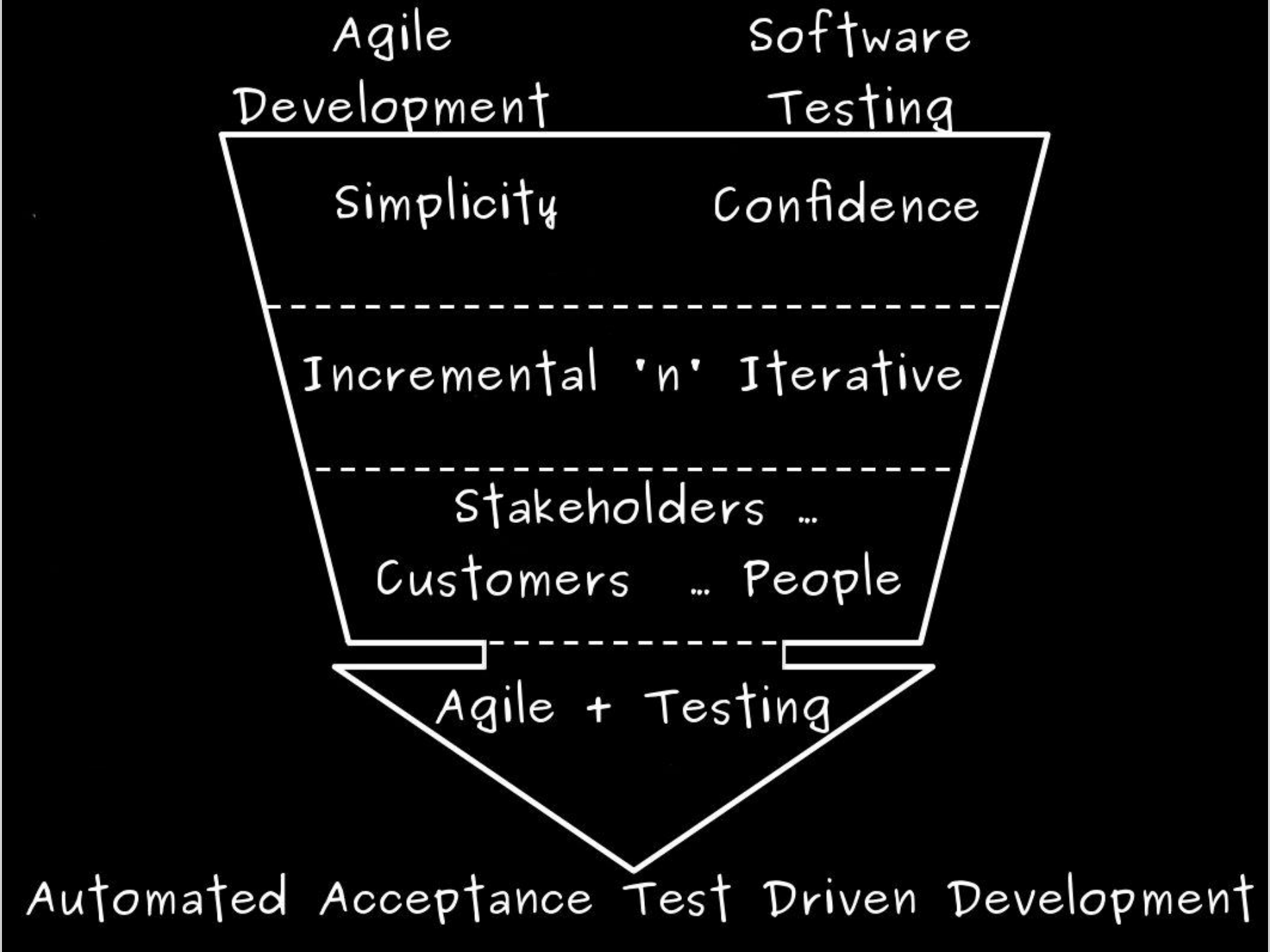
BDD

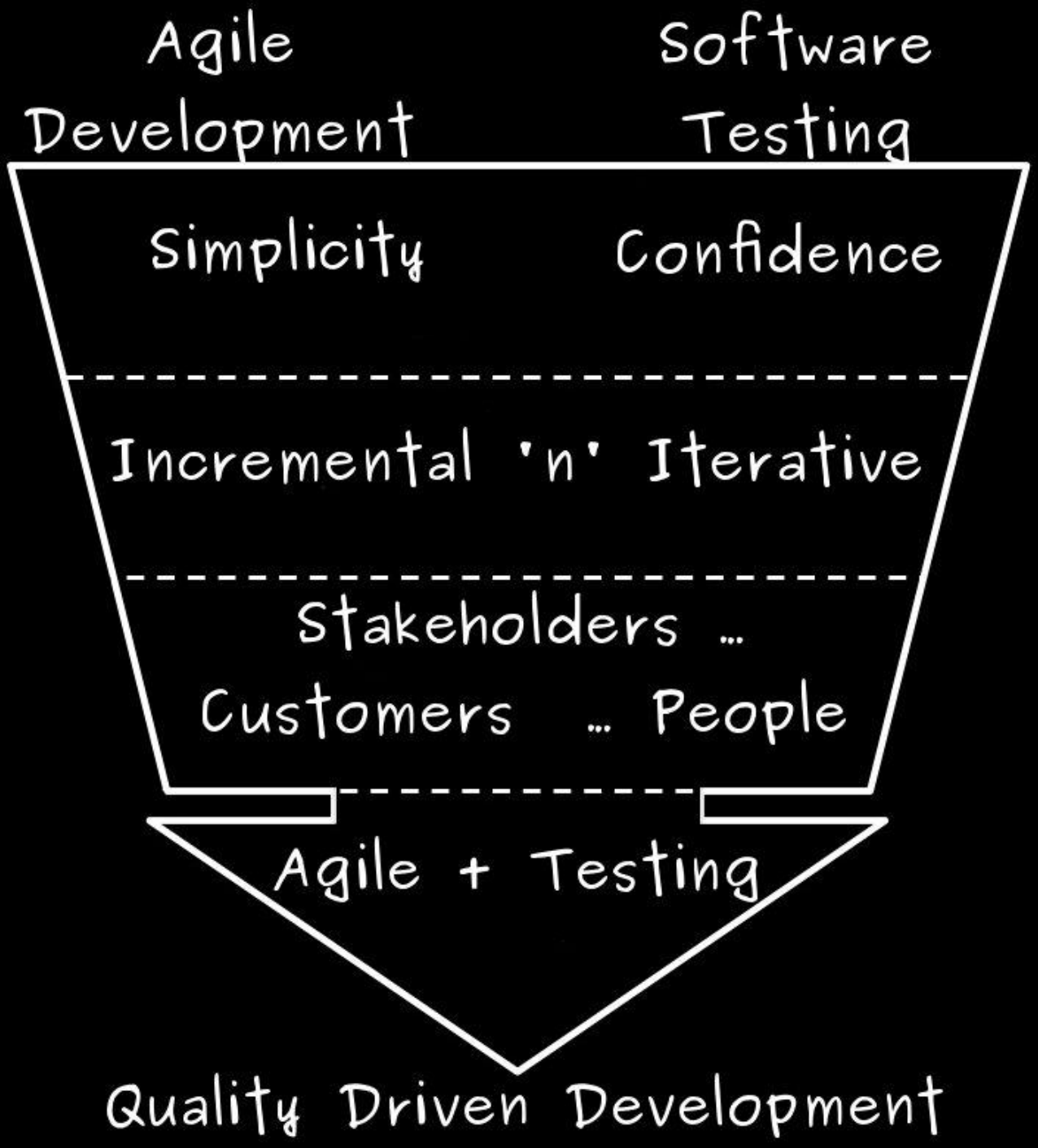

Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

BDD


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

Agile + Testing


Agile Development
Software Testing
Simplicity
Confidence
Incremental 'n' Iterative
Stakeholders …
Customers … People
Agile + Testing
Automated Acceptance Test Driven Development


Agile Development
Software Testing
Simplicity
Confidence
Incremental 'n' Iterative
Stakeholders …
Customers … People
Agile + Testing
Quality Driven Development

Behavior Driven Development (BDD)

Dan North's "Introducing BDD" - http://dannorth.net/introducing-bdd/

Idea:

- Model Requirements as User Stories.
  - User Story
    - **As a** [X]
    - **I want** [Y]
    - **So that** [Z]

- Define Acceptance Criteria/Tests as Scenarios.
  - Scenario
    - **Given** some initial context
    - **When** an event occurs
    - **Then** ensure some outcomes.

- Derive Code for Test cases using the formats for Scenarios.
- Derive Code for Classes using the Scenarios.

Summary:

- Define Behavior (Requirements)
- Define+Derive Tests for Behavior
- Implement Functionality for Behavior
- Test Functionality against Behavior (automated test cases)
- Iterate

Example

Grammar Rule: Spell out numbers in written English.

User Story: Translate Numbers from Numerals to Words in English

As a Newspaper Editor,
I want to edit newspaper articles to translate numerals into actual words,
so that I can have the satisfaction of following an esoteric rule of English Grammar.

Scenario: Number is 1
Given number is 1
When translated to words,
Then translation is “one”.

Scenario: Number is 2
Given number is 2
When translated to words,
Then translation is “two”.

Scenario: Number is 9
Given number is 9
When translated to words,
Then translation is “nine”.

Scenario: Number is 10
Given number is 10
When translated to words,
Then translation is “ten”.

Scenario: Number between 13 and 19
Given number is less than equal to 13
and number greater than equal to 19
When translated to words,
Then translation ends with “teen”
and translation has one word.

Scenario: Two digit number starts with 2
Given number has two digits
and number starts with 2
When translated to words,
Then translation starts with “twenty”.

Scenario: Two digit number starts with 9
Given number has two digits
and number starts with 9
When translated to words,
Then translation starts with “ninety”.

Scenario: **Number is positive integer**
Given number is greater than 0
And number is not a fraction or decimal.
When translated to words,
Then translation should not use “minus”

Scenario: **Number is negative integer**
Given number is less than 0
and number is not a fraction or decimal.
When translated to words,
Then translation should start w/ “minus”

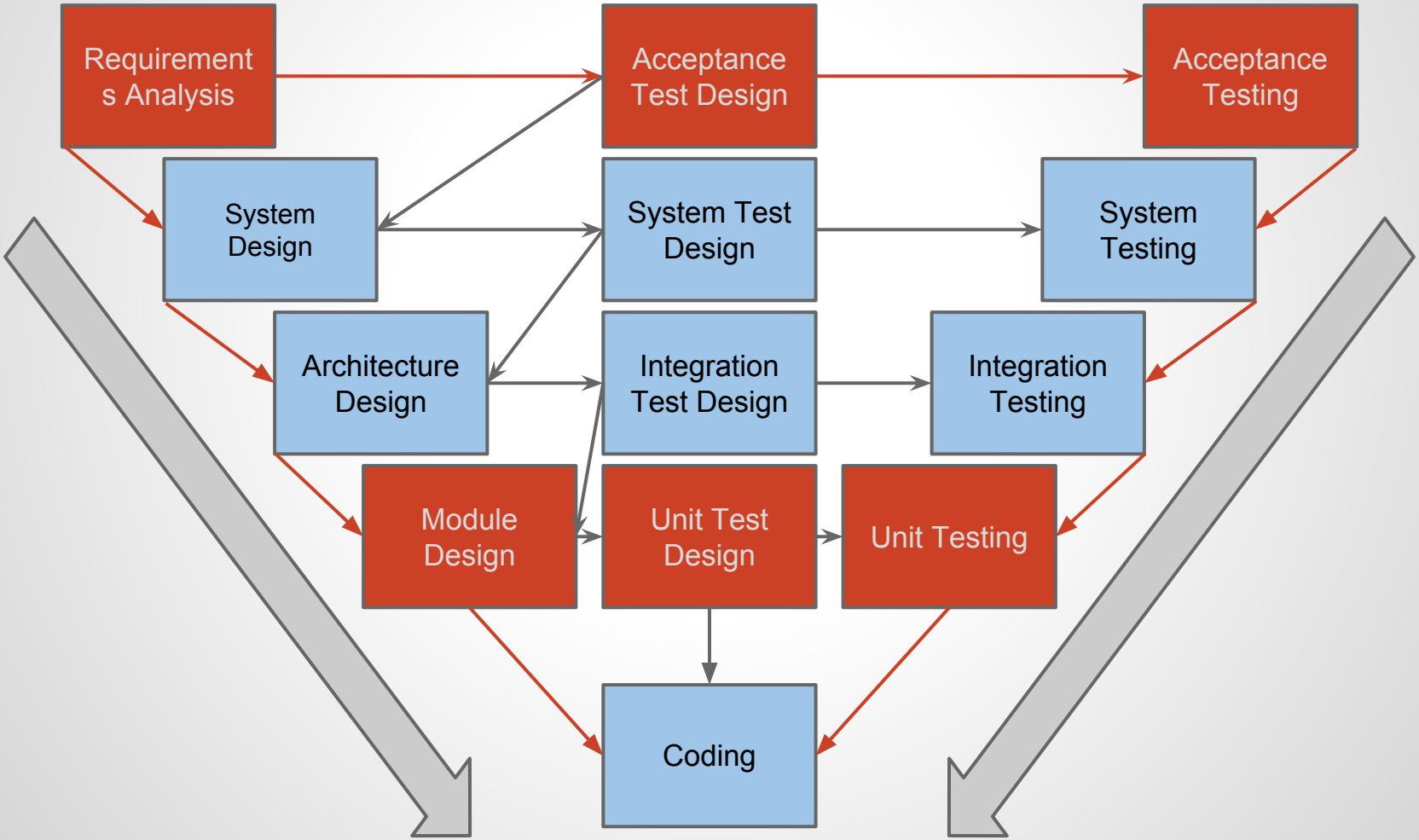
Scenario: **Positive integer ends with 0**
Given number end with 0
and number is not a fraction or decimal.
and number is positive
When translated to words,
Then translation should contain only one word

Scenario: Number is greater than one billion
Given number greater than one billion
When converted to words,
Then do not translate to words,
And notify of such occurrence
And record article, page# and line.

Scenario: Number is a decimal
Given number is a decimal
When translated to words,
Then do not translate to words,
And notify of such occurrence
And record article, page# and line.

Scenario: Number is decimal or large
Given number greater than one billion
Or number is a decimal
When converted to words,
Then do not translate to words,
And notify of such occurrence
And record article, page# and line.

TDD/BDD


Requirements Analysis
Acceptance Test Design
Acceptance Testing
System Design
System Test Design
System Testing
Architecture Design
Integration Test Design
Integration Testing
Module Design
Unit Test Design
Unit Testing
Coding

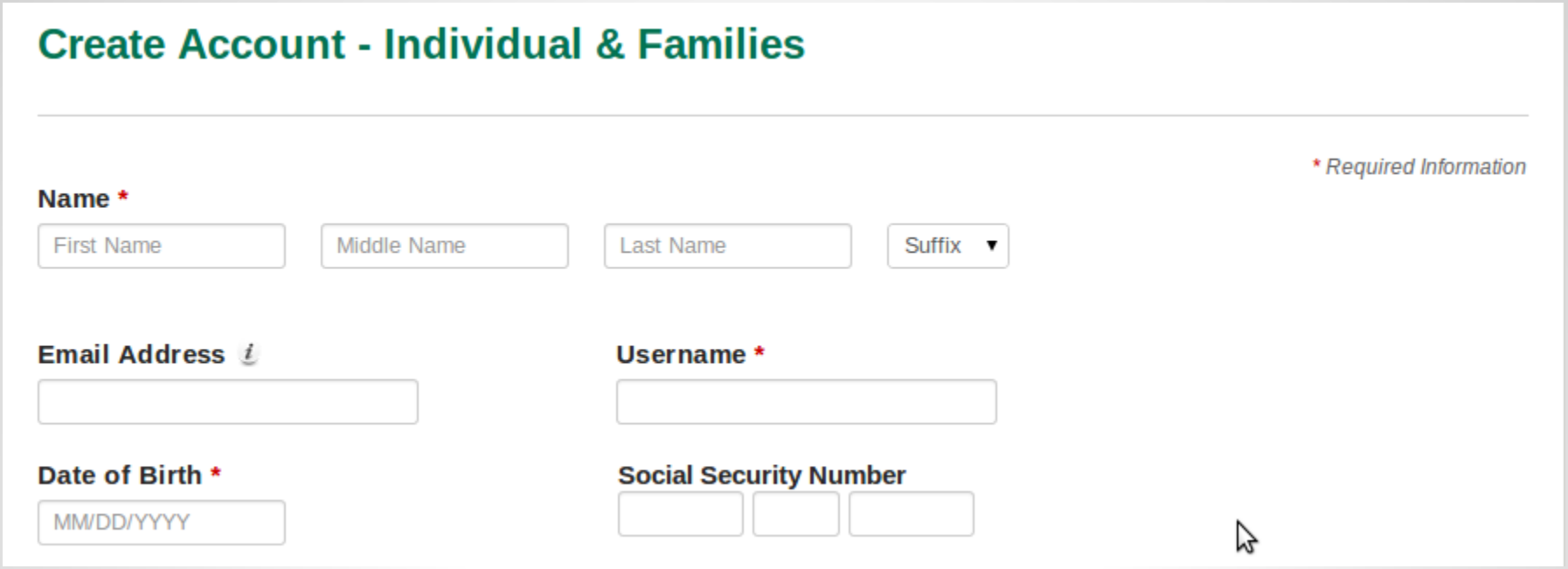
TDD/BDD

- BDD came from TDD.
  - Requirement Modelling vs. Module Modelling.
  - “Behaviour” is a more useful word than “test” - D. North
  - BDD lends structure and method to tests/testing.
- TDD revived and encouraged testing again!
- But, TDD was still focused on the tester and the code.
- BDD takes the focus away from tester and code;
- BDD puts focus on client and product behavior.

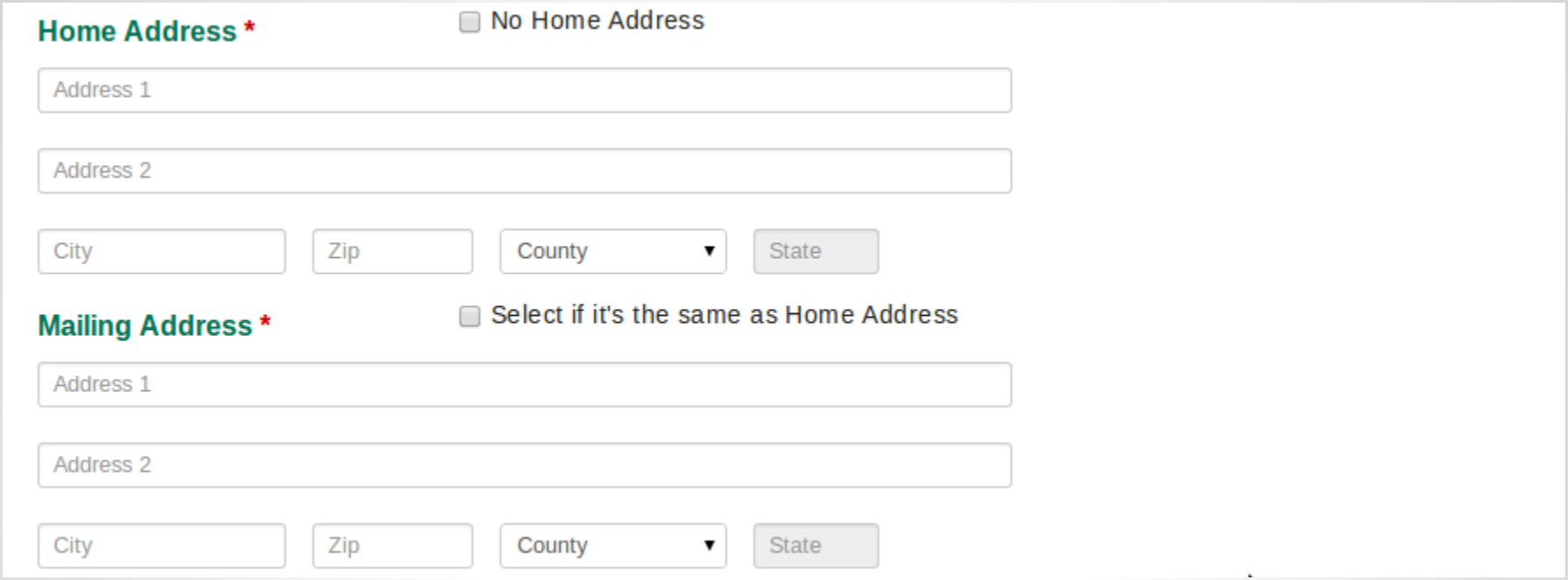
Hands On Assignment

User Story: Sign up online for Health Insurance.

As the head of my family,
I want to create an account on the health insurance website,
So that I can purchase health insurance for my family and myself.



Scenario: User forgets to enter Last Name
Given “First Name” field is entered
and “Last Name” field is empty
When user starts filling out “Email Address” field
Then highlight that “Last name” is required.

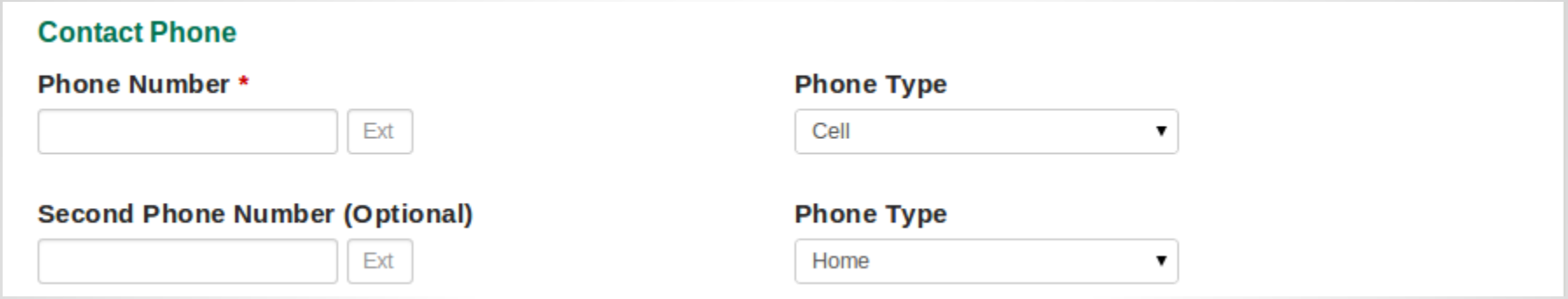


Scenario: City and Zip code match

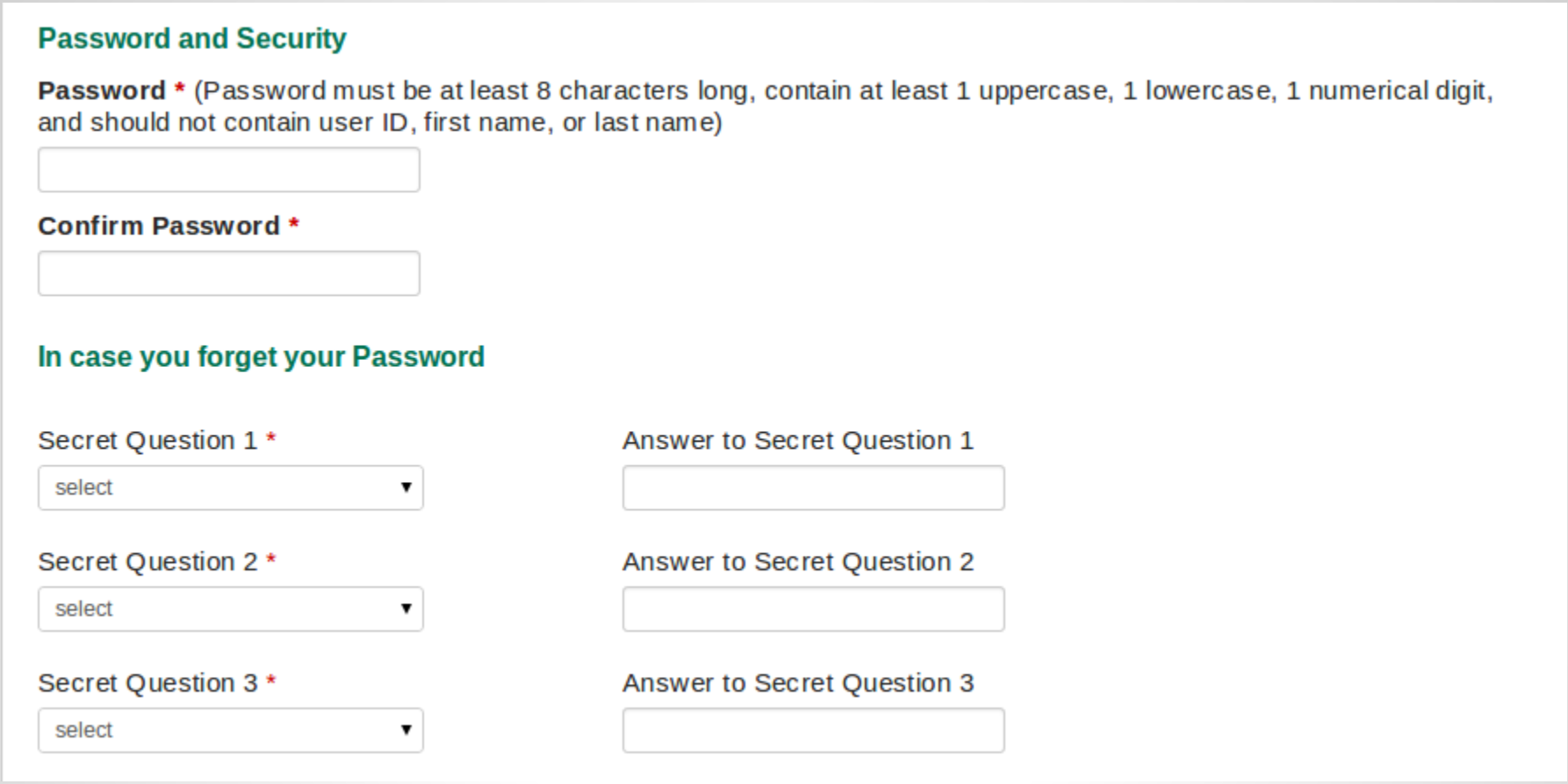
Given “City” field is filled in “home address” section,

When “Zip” field is filled in “home address” section,

Then “Zip” field contents should be valid w.r.t “City” field contents.



Scenario: International country code is recognized as valid.
Given “Phone Number” field is filled in “Contact Phone” section,
and “Phone Number” field contains the international country code
When user moves away to a different field
Then the contents of the “Phone Number” field is marked as valid
and no error message is displayed.

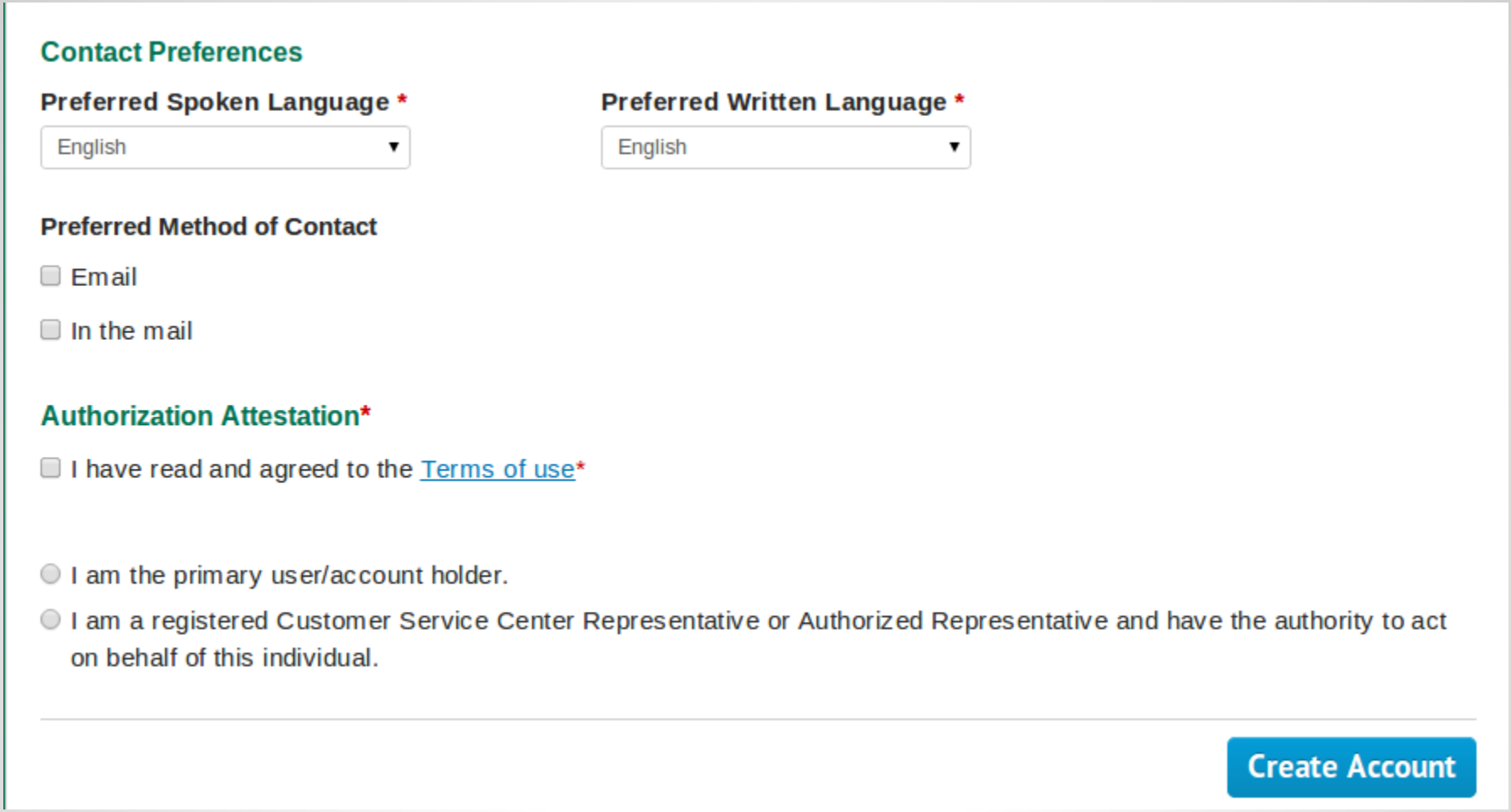


Scenario: Passwords should never be in plain text

Given “Password” field is filled

When user moves away from “Password” field,

Then contents of the “Password” field should not be displayed in plain text.



Scenario: Confirmation page after clicking Create Account

Given user is done filling out the form

When user clicks the “Create Account” button,

Then a confirmation page showing all the entered information should be displayed.

In Summary: Agile

- Agile ideas/principles/methods have been around longer than Agile
  - Many are simply sound SE principles and lessons learned
  - Agile is No Silver Bullet
- XP/Agile started out at the micro (team) level
  - Still excel there
- Waterfall still has value at the macro level
  - Water-SCRUM-Fall is not necessarily bad
  - Agile at Scale, Scalable Agile Framework (SAFe)

In Summary: Testing

- Testing is **part of** a larger **quality assurance strategy**.
- Testing is (should be) a **constant activity** throughout the software lifecycle.
- Testing is about **Confidence**.
  - Assurance != Ensurance
  - Remember Dijkstra
- Many tools for **Automation**.
  - Junit, TestNG, JBehave, Cucumber, etc.
- Testing is about the **People**.
  - Client, Developer/Tester

In Summary: Agile + Testing

"Agile + Testing" leads to:

- **Automation**.
- Focus on **Acceptance criteria and tests**.
  - by extension **Requirements**, i.e. **Behaviors**
- Focus on the **People**.
  - **Developers/Testers** and **Clients**.