Technical Practices through a Kanban Lens

Nayan Hajratwala
nayan@chikli.com
@nhajratw

CHIKLI CONSULTING

Kanban Foundational Principles

- Start with what you do now
- Agree to pursue evolutionary change
- Initially, respect current processes, roles, responsibilities & job titles
- Encourage acts of leadership at every level in your organization

Kanban Core Practices

- Visualize
- Limit WIP
- Manage flow
- Make policies explicit
- Implement feedback loops
- Improve collaboratively, evolve experimentally (using models)

Some Extreme Programming Technical Practices

- Pair Programming
- Continuous Integration
- Simple Design
- Collective Code Ownership
- TDD
- Whole Teams

Some Other Technical Practices

- Acceptance Test Driven Development (ATDD)
- Mob Programming
- Clean Code
- Continuous Deployment
- Feature Toggles
- Refactoring
- Feature Splitting
- Evolutionary Design

Jenkins

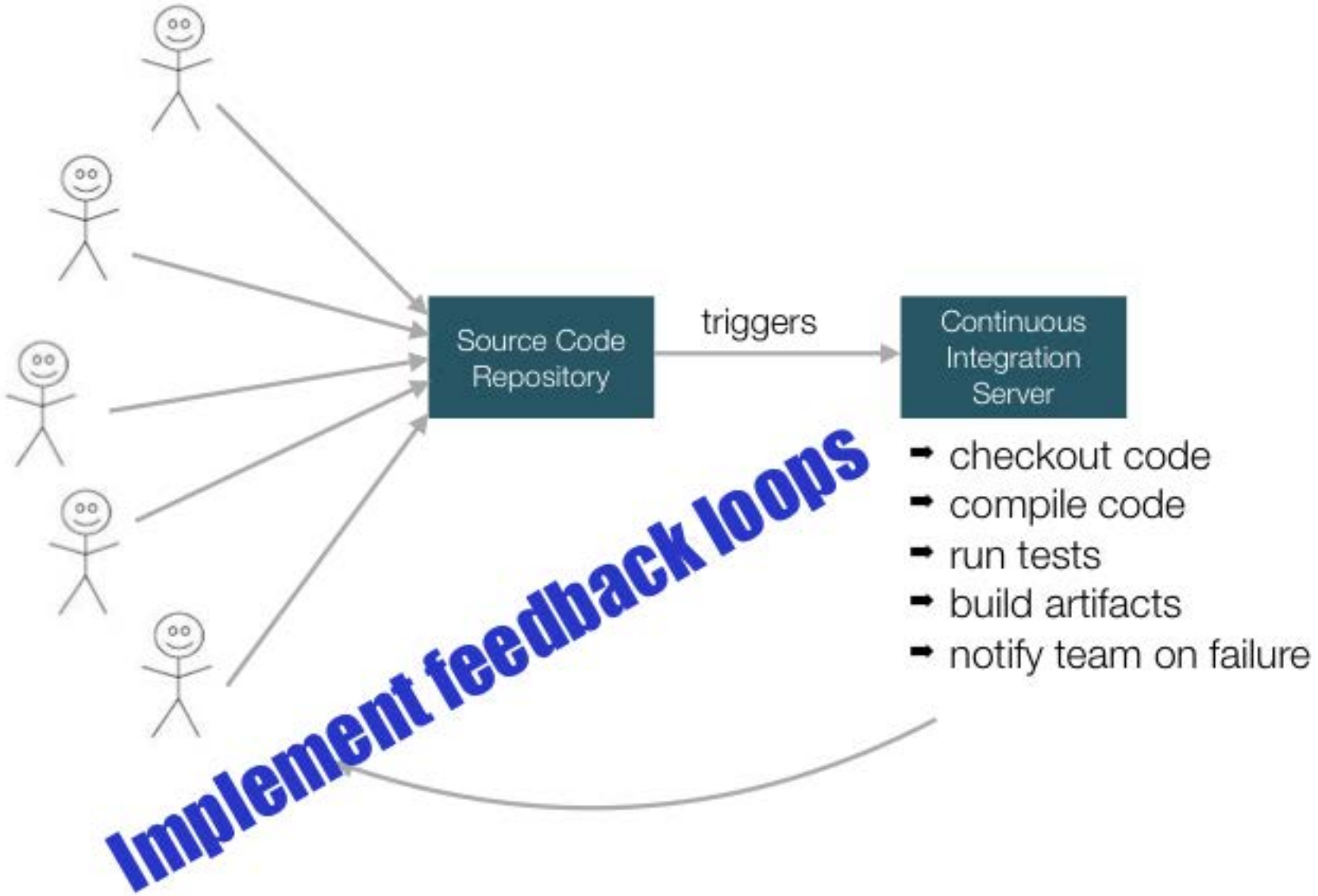
Hudson

Bamboo

go

Continuous Integration

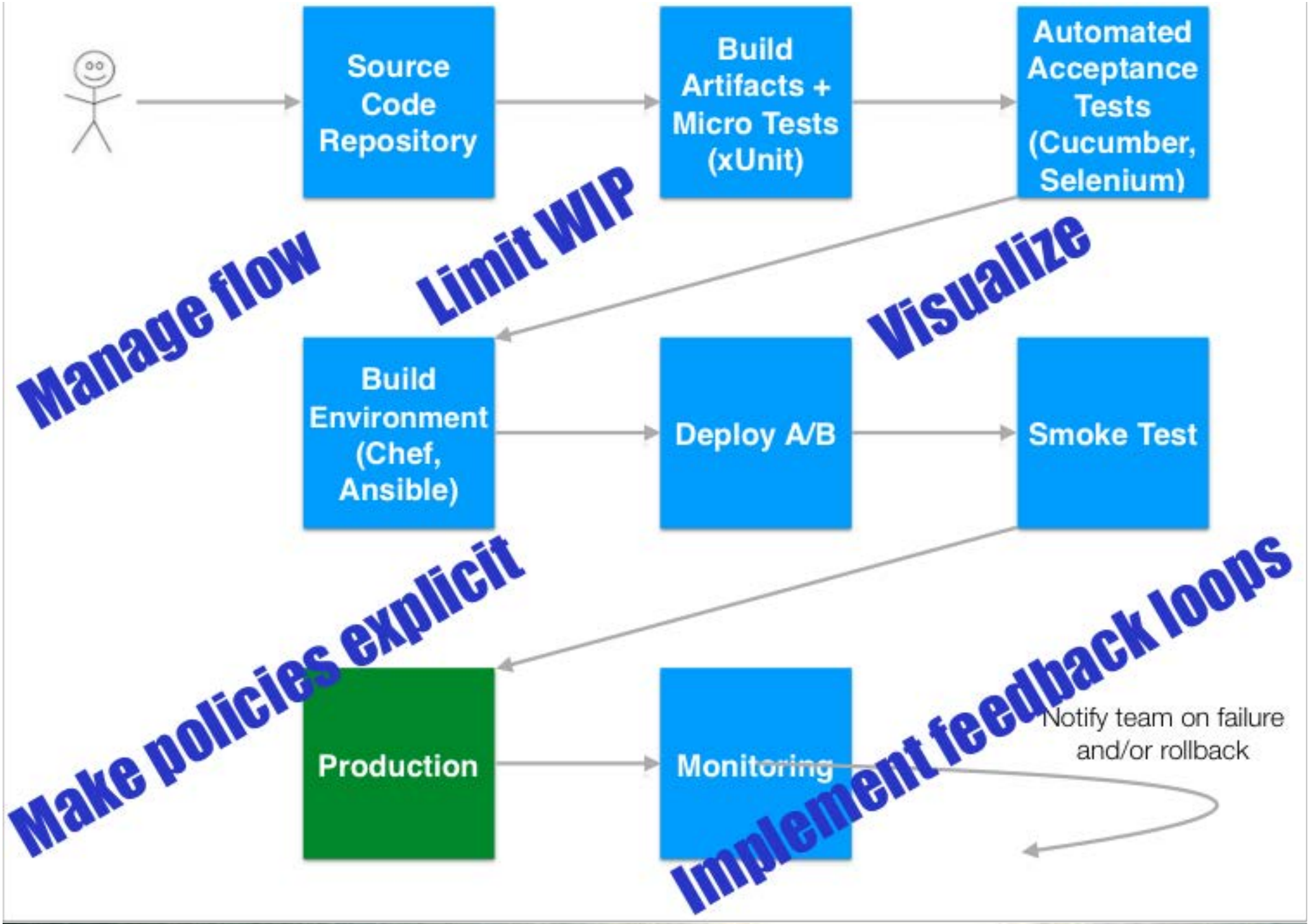
Continuous Integration



Continuous Integration

It compiles!
Let's ship it!

Continuous
Deployment


Source Code Repository
Build Artifacts + Micro Tests (xUnit)
Automated Acceptance Tests (Cucumber, Selenium)
Build Environment (Chef, Ansible)
Deploy A/B
Smoke Test
Production
Monitoring
Notify team on failure and/or rollback
Manage flow
Limit WIP
Visualize
Make policies explicit
Implement feedback loops

Feature Toggles

Careful, these experiments may bite

WARNING These experimental features may change, break, or disappear at any time. We make absolutely no guarantees about what may happen if you turn one of these experiments on, and your browser may even spontaneously combust. Jokes aside, your browser may delete all your data, or your security and privacy could be compromised in unexpected ways. Any experiments you enable will be enabled for all users of this browser. Please proceed with caution. Interested in cool new Chrome features? Try our beta channel at chrome.com/beta.

Experiments Reset all to default

Override software rendering list Mac, Windows, Linux, Chrome OS, Android
Overrides the built-in software rendering list and enables GPU-acceleration on unsupported system con… …anore-gpu-blacklist
Enable

Experimental canvas features Mac, Windows, Linux, Chrome OS, Android
Enables the use of experimental canvas features which are still in development. #enable… …ntal-canvas-features
Enable

Accelerated 2D canvas Mac, Windows, Linux, Chrome OS, Android
Enables the use of the GPU to perform 2d canvas rendering instead … …ftware rendering. #disable-accelerated-2d-canvas
Disable

Display list 2D canvas Mac, Windows, Linux, Chro…
Enables the use of display lists to record 2D canv… …s. This allows 2D canvas rasterization to be performed on separate thread. #enable-display-list-2d-canvas
Default

Composited render layer borders Mac, …ows, Linux, Chrome OS, Android
Renders a border around composited Render Layers to help debug and study layer compositing. #composited-layer-borders
Enable

WebRTC Stun origin header Mac, Windows, Linux, Chrome OS, Android
When enabled, Stun messages generated by WebRTC will contain the Origin header. #enable-webrtc-stun
Enable

Native Client Mac, Windows, Linux, Chrome OS, Android
Support Native Client for all web applications, even those that were not installed from the Chrome Web Store. #enable-nacl
Enable

Native Client GDB-based debugging Mac, Windows, Linux, Chrome OS
Enable GDB debug stub. This will stop a Native Client application on startup and wait for nacl-gdb (from the NaCl SDK) to attach to it. #enable-nacl-debug
Enable

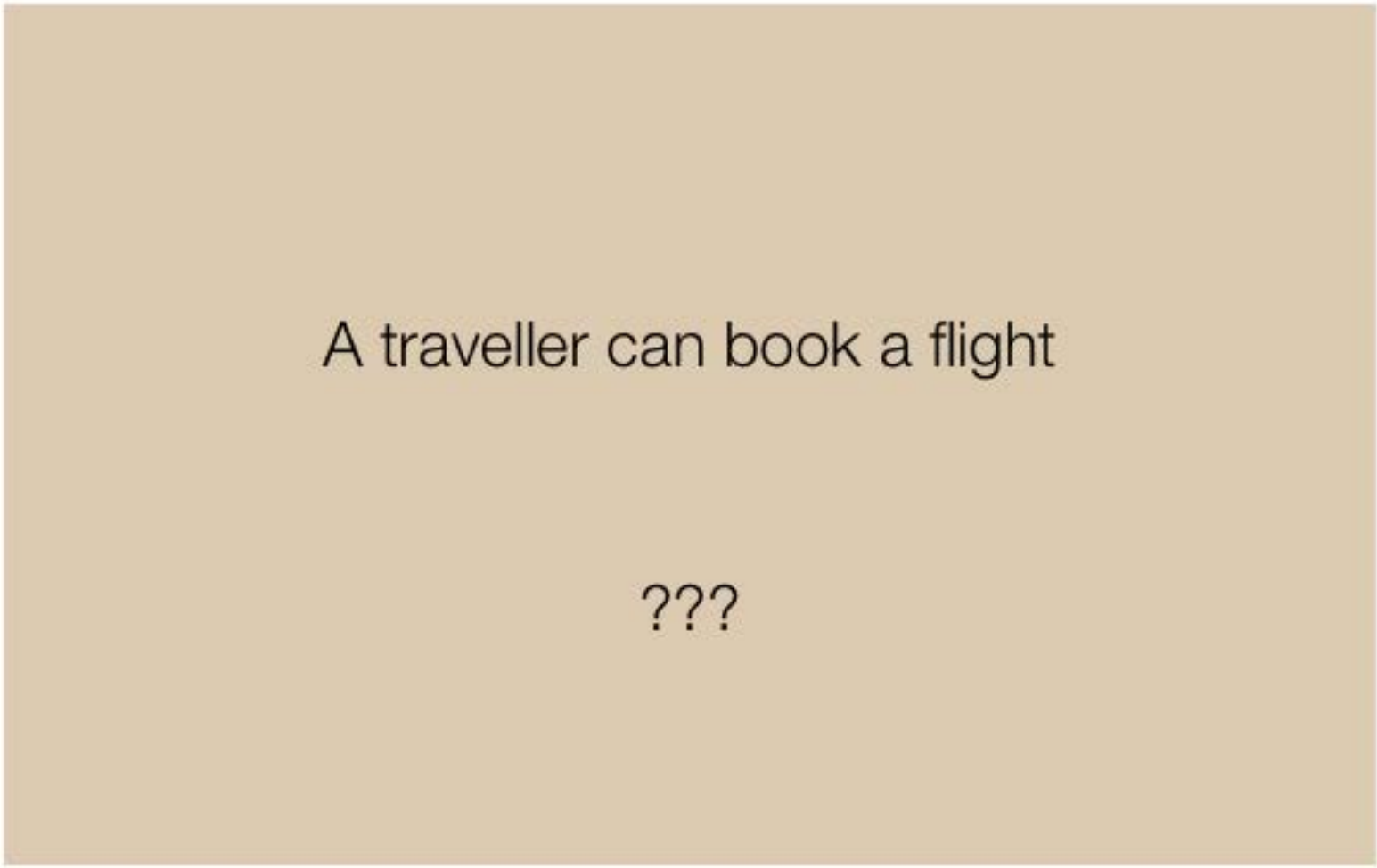
Restrict Native Client GDB-based debugging by pattern Mac, Windows, Linux, Chrome OS
Restricts Native Client application GDB-based debugging by URL of manifest file. Native Client GDB-based debugging must be enabled for this option to work. #nacl-debug-mask

Implement feedback loops

Manage flow

Feature Splitting

Feature Splitting



Feature Splitting

A traveller can book a one-way flight

???

Feature Splitting

A traveller can book a one-way flight
that starts at DTW

???

Feature Splitting

A traveller can book a one-way flight
that starts in DTW and arrives in SAN

???

Feature Splitting

A traveller can book a one-way flight
that starts in DTW at 10:00 am and
arrives in SAN at 1:25 pm

2 days

Make policies explicit

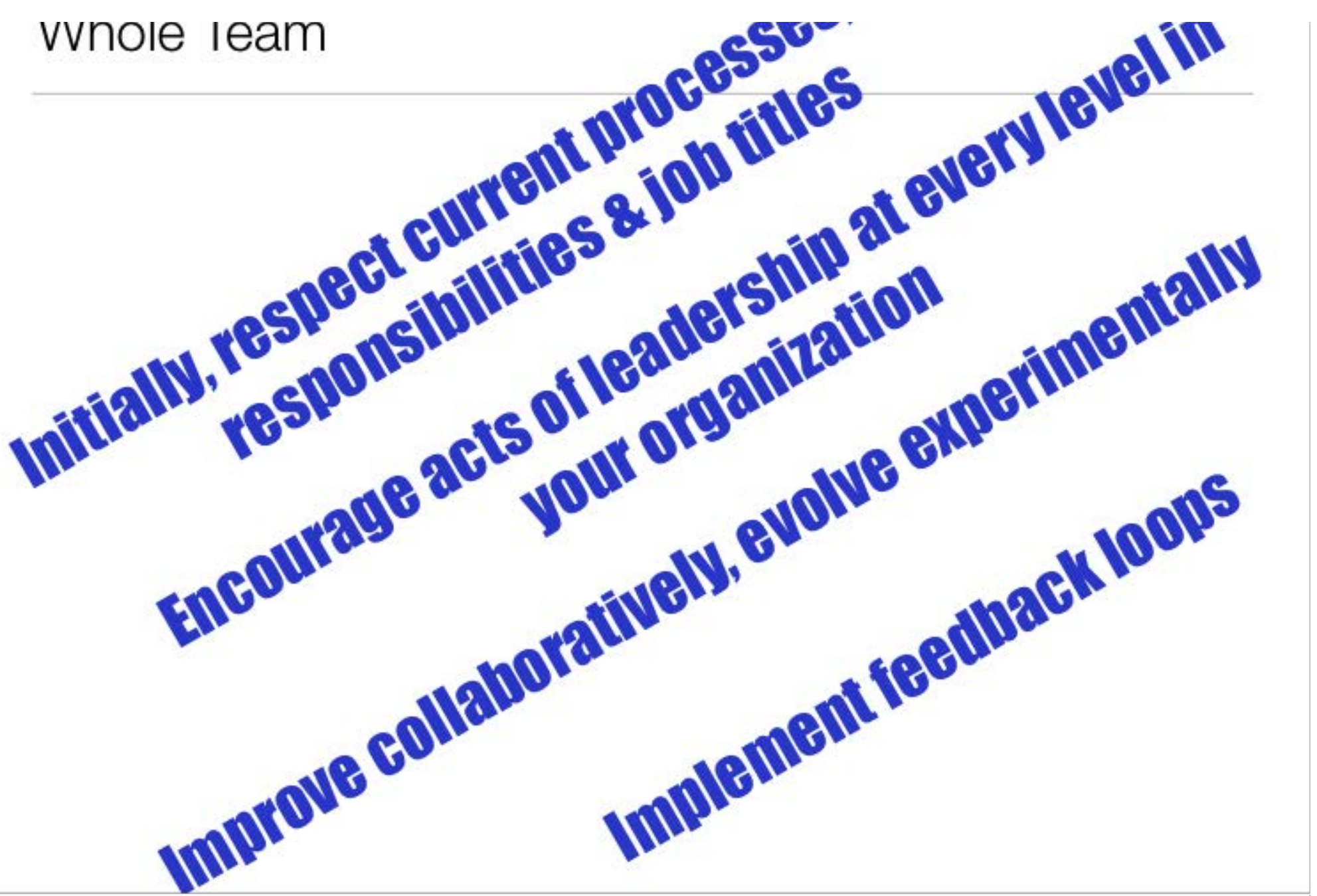
Feature Splitting

Whole Team

Whole Team

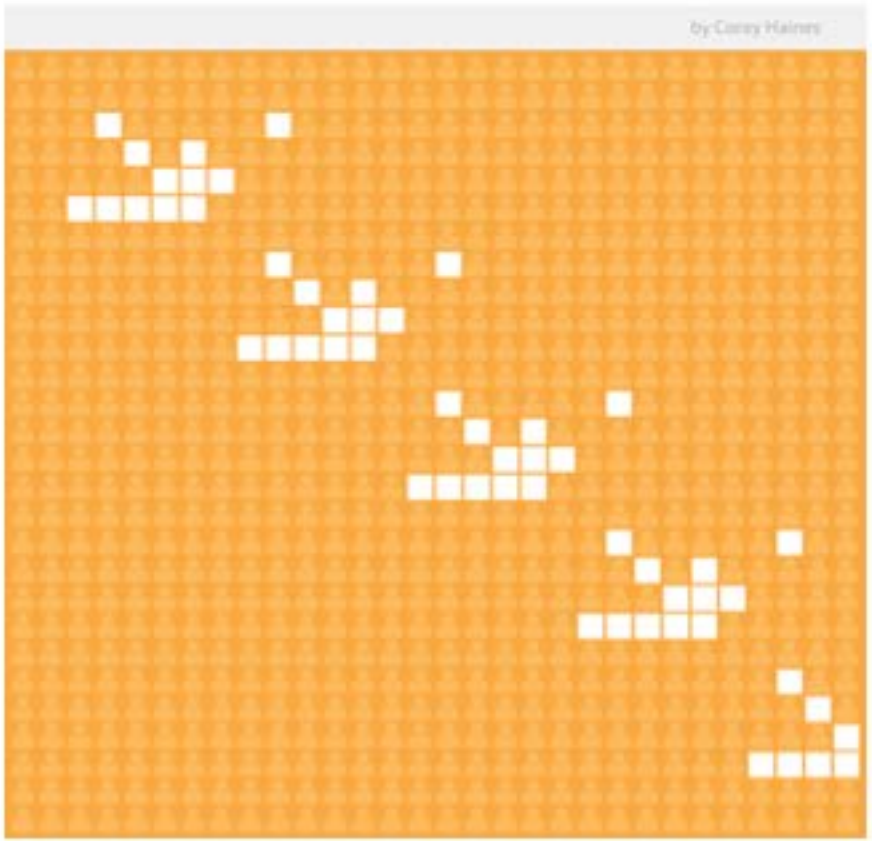
ces, roles,

Whole Team



UNDERSTANDING THE

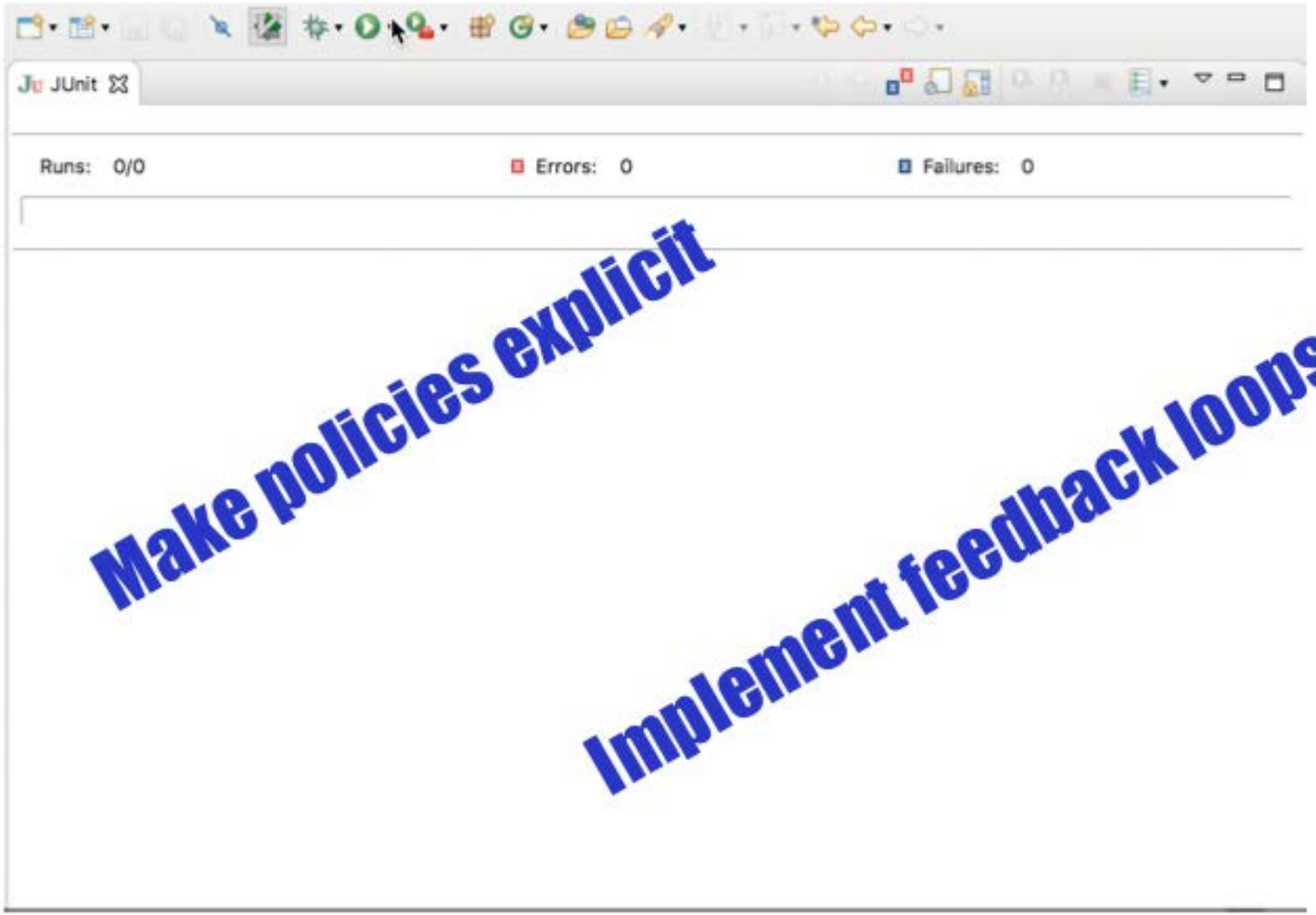
4 RULES OF SIMPLE DESIGN



Simple Design

1. - Passes All the Tests

1 - Passes All the Tests



2 - Don't Repeat Yourself (DRY)

```
def create_regular_user(id)
  user = User.new
  user.id = id
end
```

```
def create_admin_user(id)
  user = User.new
  user.id = id
  user.admin = true
end
```

2 - Don't Repeat Yourself (DRY)

```
def create_regular_user(id)
  user = User.new
  user.id = id
  user.password_expiration = '2017-12-31'
end



def create_admin_user(id)
  user = User.new
  user.id = id
  user.admin = true
end
```

2 - Don't Repeat Yourself (DRY)

```
def create_regular_user(id)
  user = User.new
  user.id = id
end
```

```
def create_admin_user(id)
  user = User.new
  user.id = id
  user.admin = true
end
```

2 - Don’t Repeat Yourself (DRY)

```
def create_regular_user(id)
  user = User.new
  user.id = id
end
```

```
def create_admin_user(id)
  user = create_regular_user(id)
  user.admin = true
end
```

2 - Don’t Repeat Yourself (DRY)

```
def create_regular_user(id)
  user = User.new
  user.id = id
  user.password_expiration = '2017-12-31'
end
```

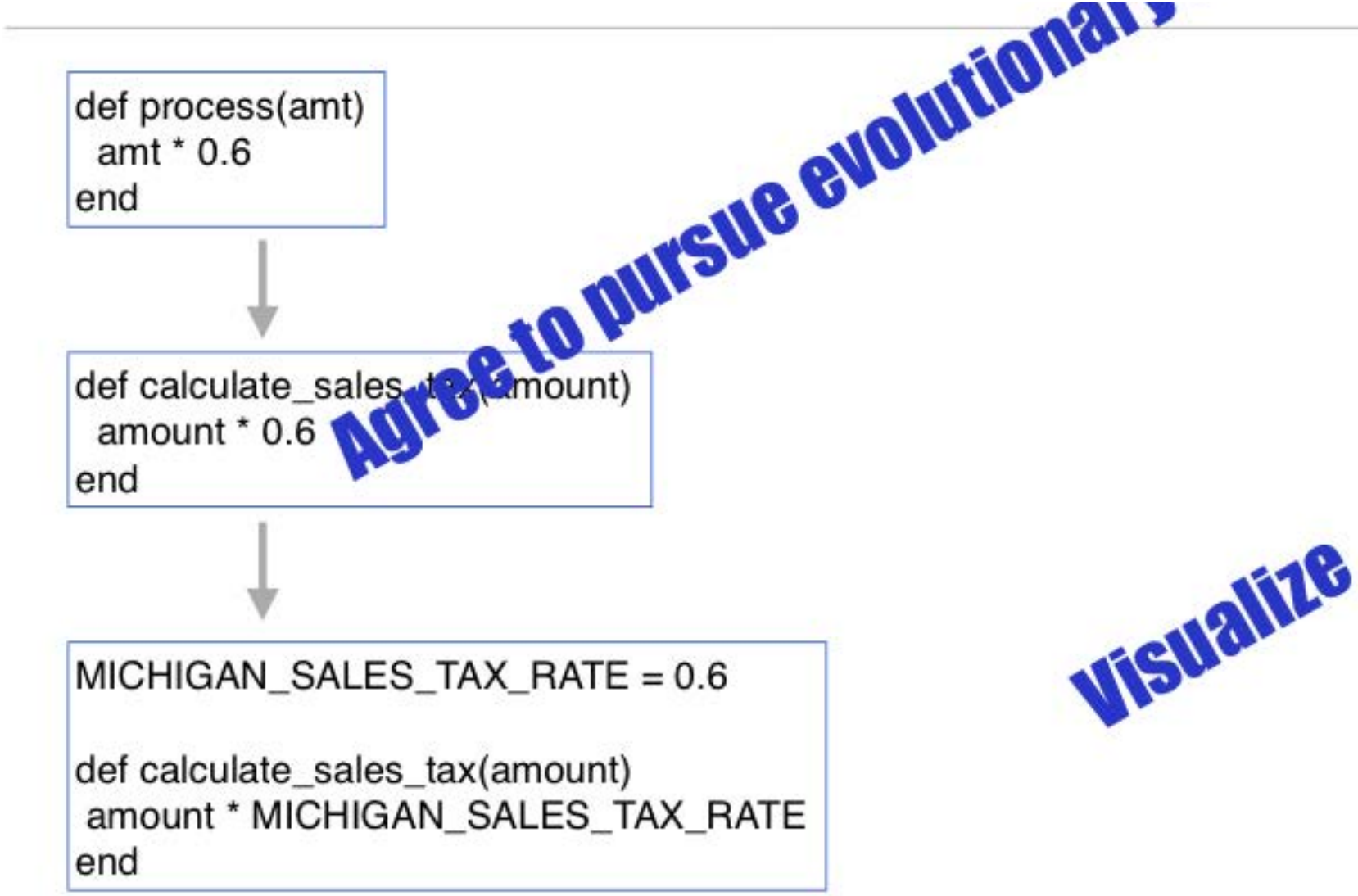
Limit WIP

```
def create_admin_user(id)
  user = create_regular_user(id)
  user.admin = true
end
```

Agree to pursue evolutionary change

3 - Reveal Intent

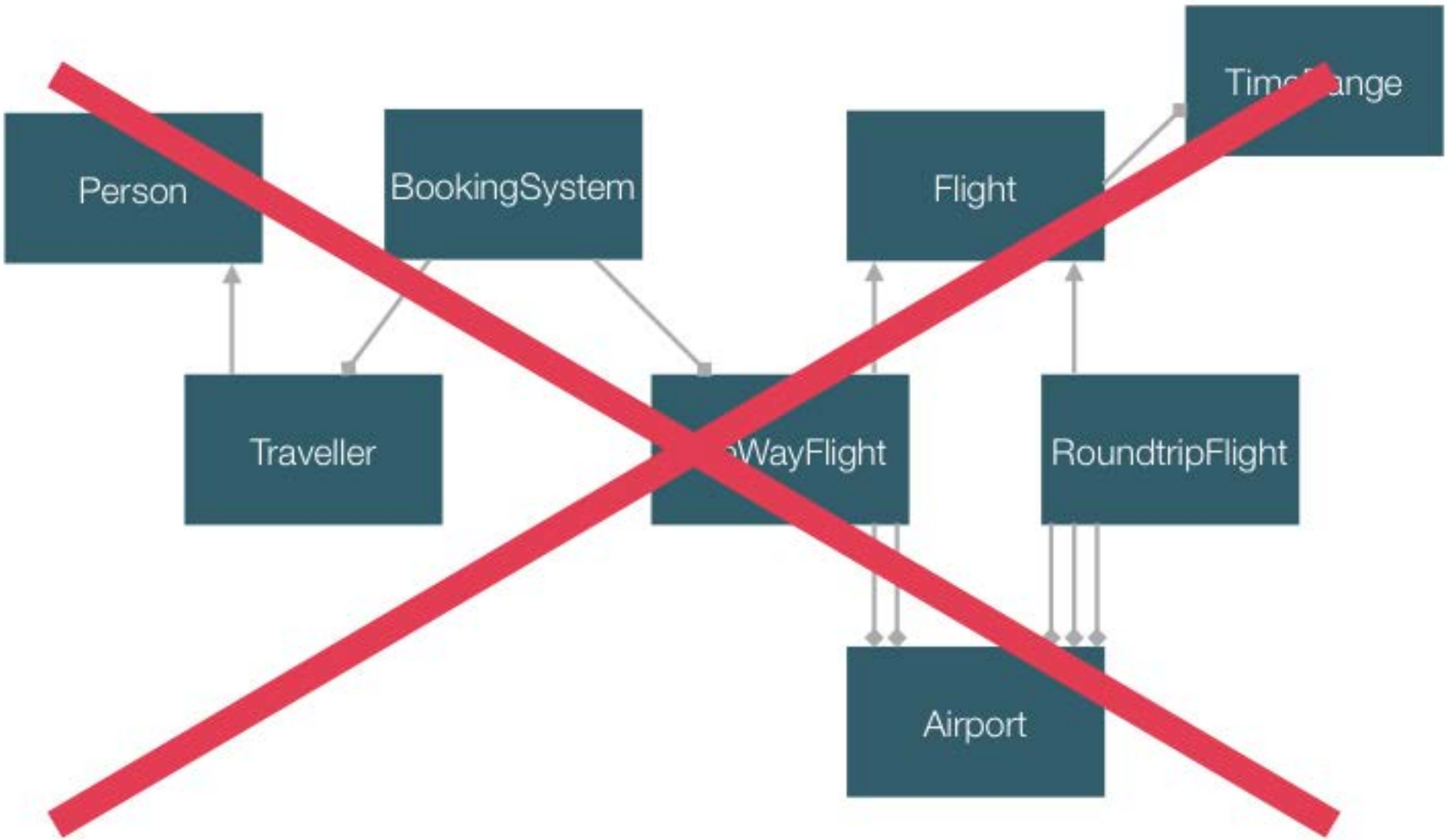
ary change



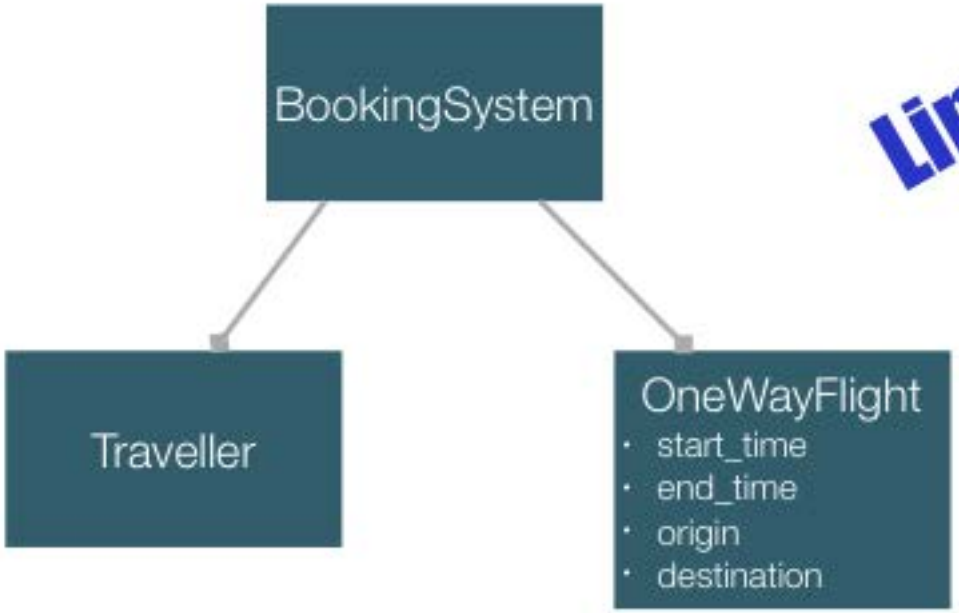
4 - Minimize Moving Parts

A traveller can book a one-way flight
that starts in DTW at 10:00 am and
arrives in SAN at 1:25 pm

4 - Minimize Moving Parts



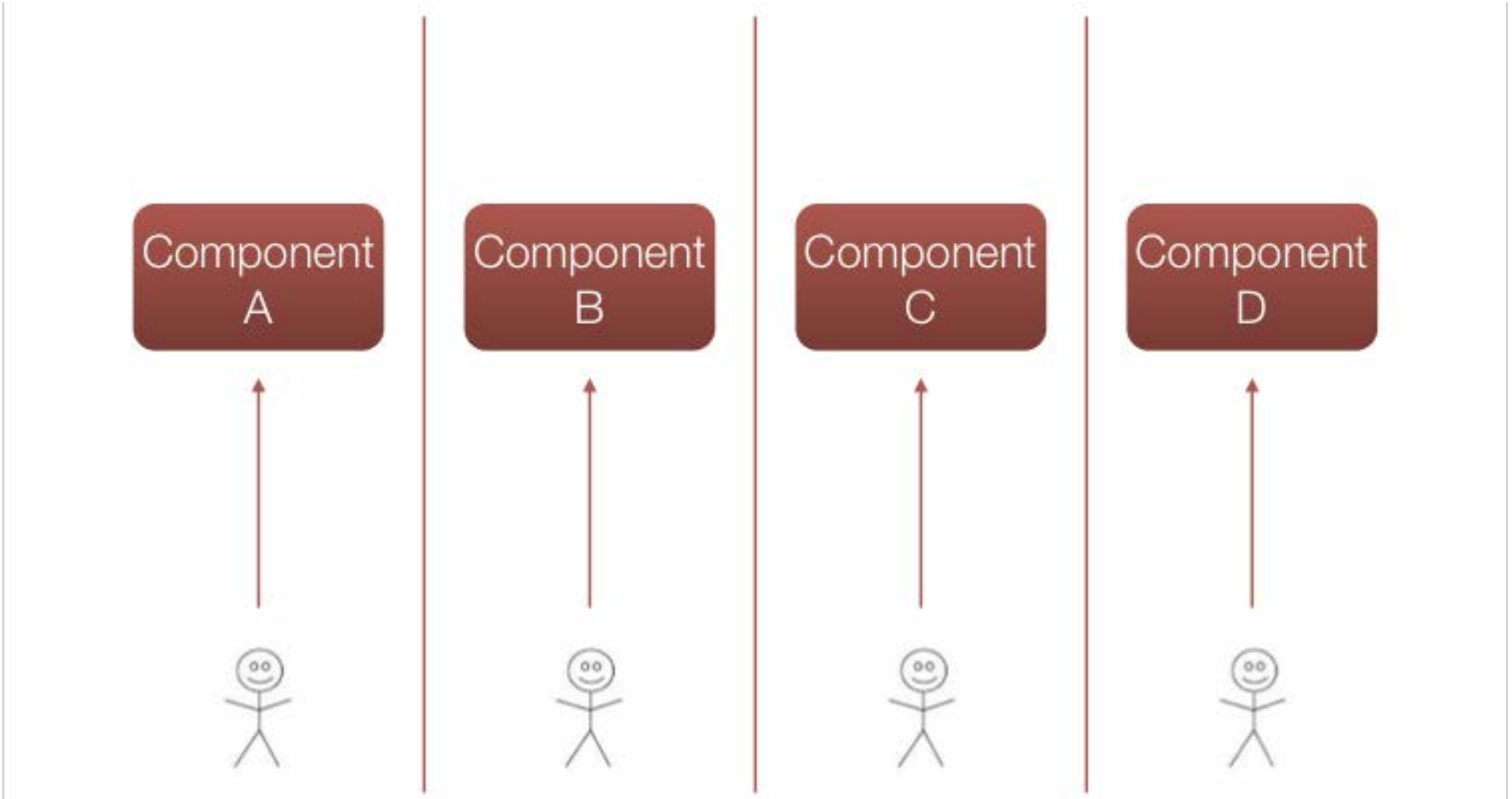
4 - Minimize Moving Parts


BookingSystem
Traveller
OneWayFlight
start_time
end_time
origin
destination

Limit WIP

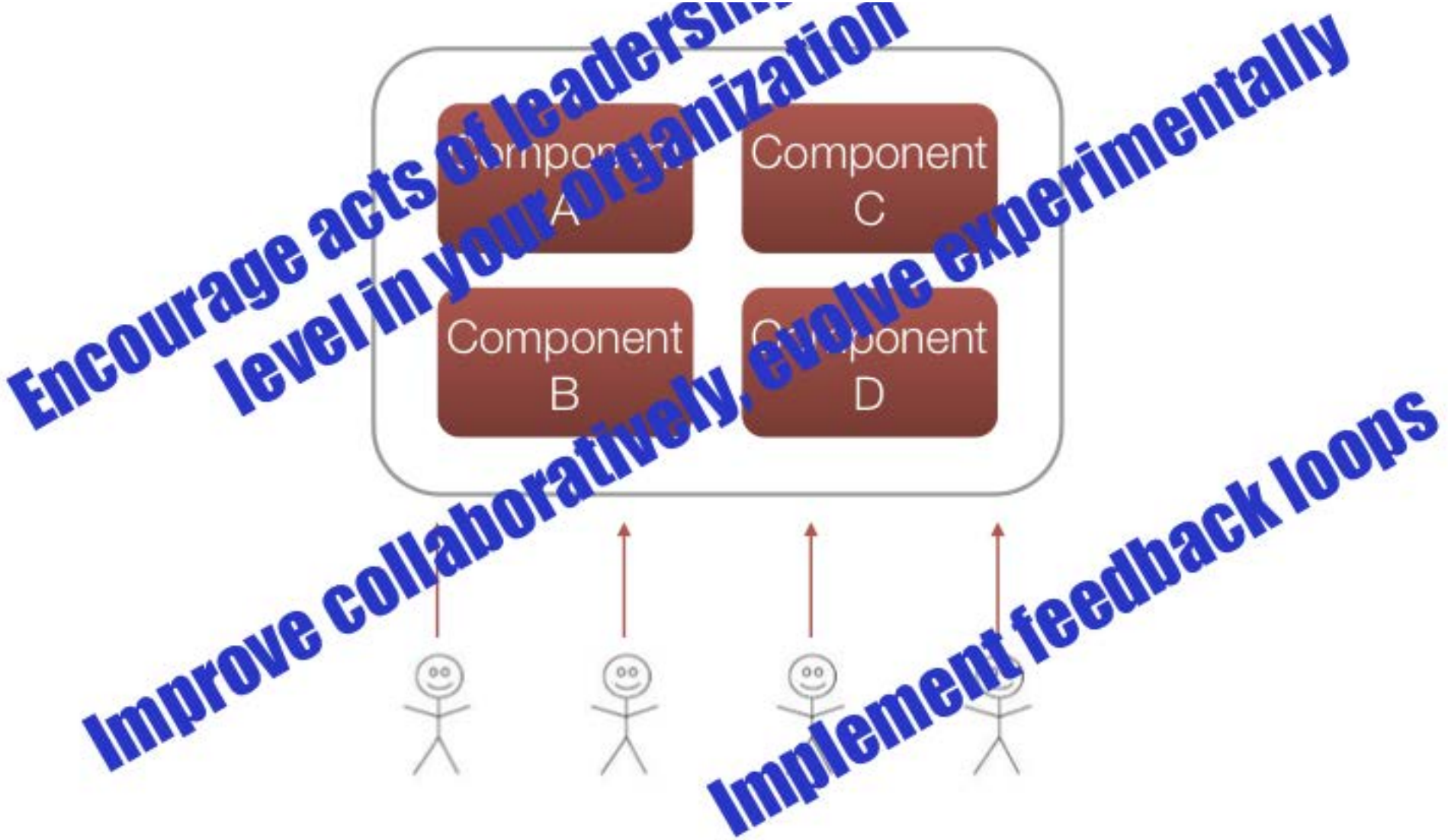
Collective Code Ownership

Collective Code Ownership

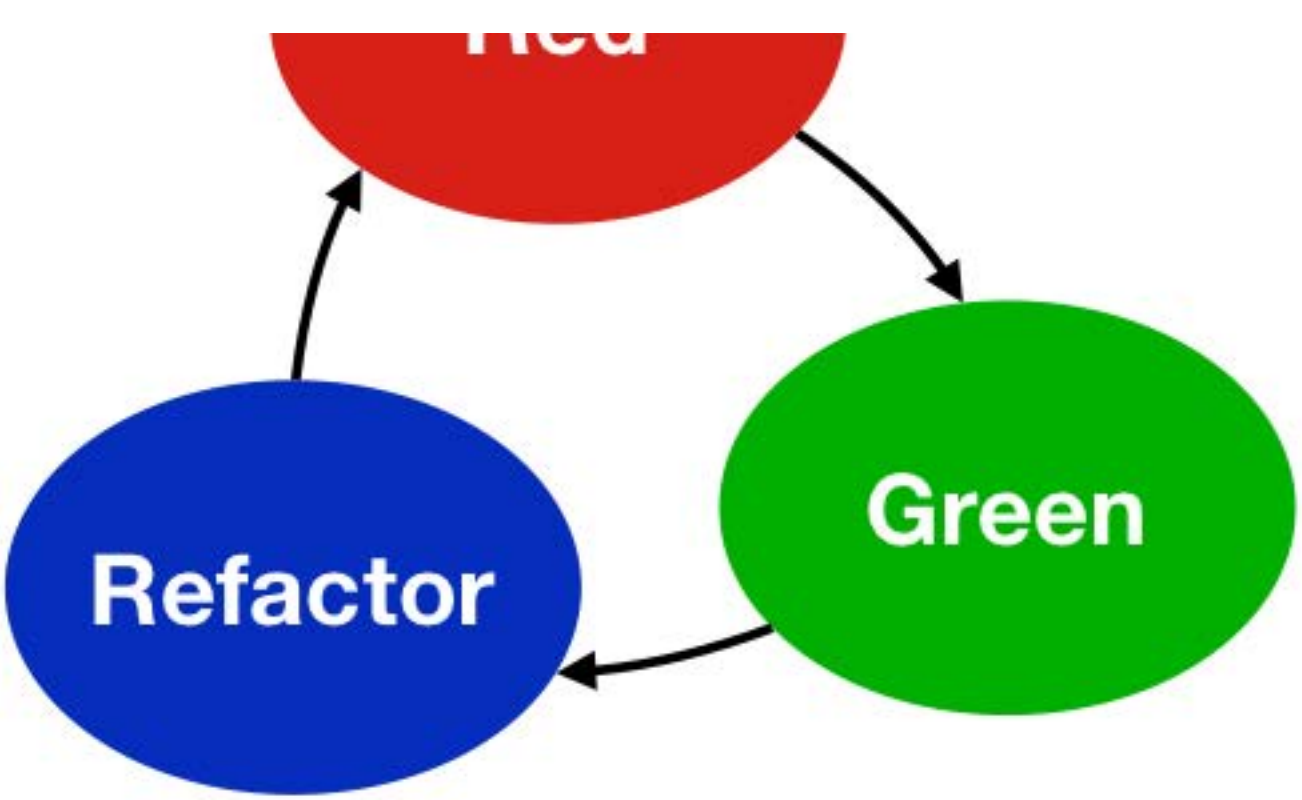


Collective Code Ownership

ship at every


Component A
Component C
Component B
Component D
Encourage acts of leadership at every level in your organization
Improve collaboratively, evolve experimentally
Implement feedback loops

Red



Test Driven Development

Test Driven Development (TDD)

```
describe "tic tac toe"
  it "starts with an empty board"
    ttt = TicTacToe.new
    ttt.board.should ==
      " - - -
        - - -
        - - - "
  end
end
```

Visualize

Limit WIP

Test Driven Development (TDD)

```
describe "tic tac toe"
  it "starts with an empty board"
    ttt = TicTacToe.new
    ttt.board.should ==
      " - - -
        - - -
        - - - "
  end
end
```

```
class TicTacToe

  def initialize
    @board = [
      ['-', '-', '-'],
      ['-', '-', '-'],
      ['-', '-', '-']
    ]
  end
end
```

Implement feedback loops

Test Driven Development (TDD)

```
describe "tic tac toe"
  it "starts with an empty board"
    ttt = TicTacToe.new
    ttt.board.should ==
      " - - -
        - - -
        - - - "
  end
end
```

```
class TicTacToe
  BLANK = '-'

  def initialize
    @board = [
      [BLANK, BLANK, BLANK],
      [BLANK, BLANK, BLANK],
      [BLANK, BLANK, BLANK]
    ]
  end
end
```

Improve collaboratively, evolve experimentally (using models)

Test Driven Development (TDD)

```
describe "tic tac toe"
  it "starts with an empty board"
    ttt = TicTacToe.new
    ttt.board.should ==
      " - - -
        - - -
        - - - "
  end

  it "plays an X first" do
      ttt = TicTacToe.new
      ttt.play(0,0)
      ttt.board.should ==
        " X - -
          - - -
          - - - "
    end
end
```

```
class TicTacToe

  BLANK = '-'

  def initialize
    @board = [
      [BLANK, BLANK, BLANK],
      [BLANK, BLANK, BLANK],
      [BLANK, BLANK, BLANK]
    ]
  end
end
```

Test Driven Development (TDD)

```
describe "tic tac toe"
  it "starts with an empty board"
    ttt = TicTacToe.new
    ttt.board.should ==
      " - - -
        - - -
        - - - "
  end

  it "plays an X first" do
      ttt = TicTacToe.new
      ttt.play(0,0)
      ttt.board.should ==
        " X - -
          - - -
          - - - "
    end
end
```

Make policies explicit

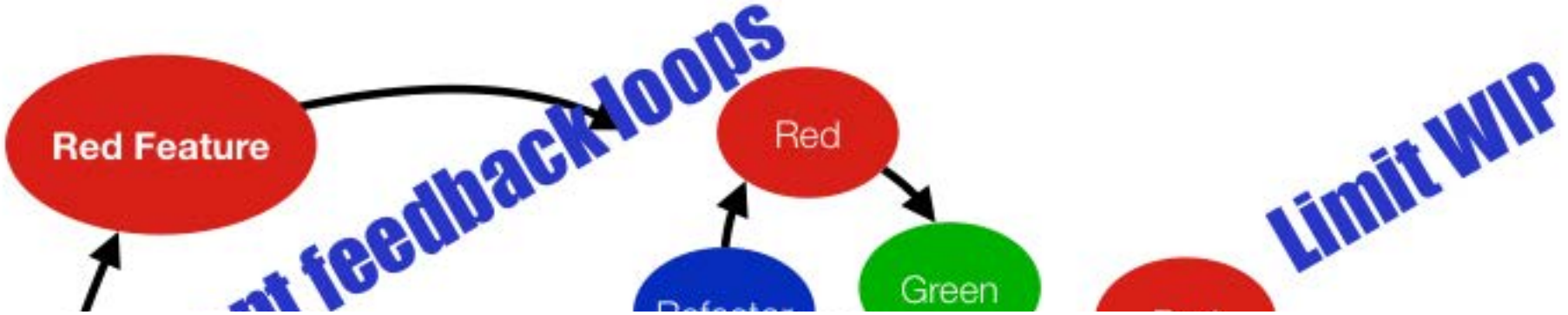
Manage flow

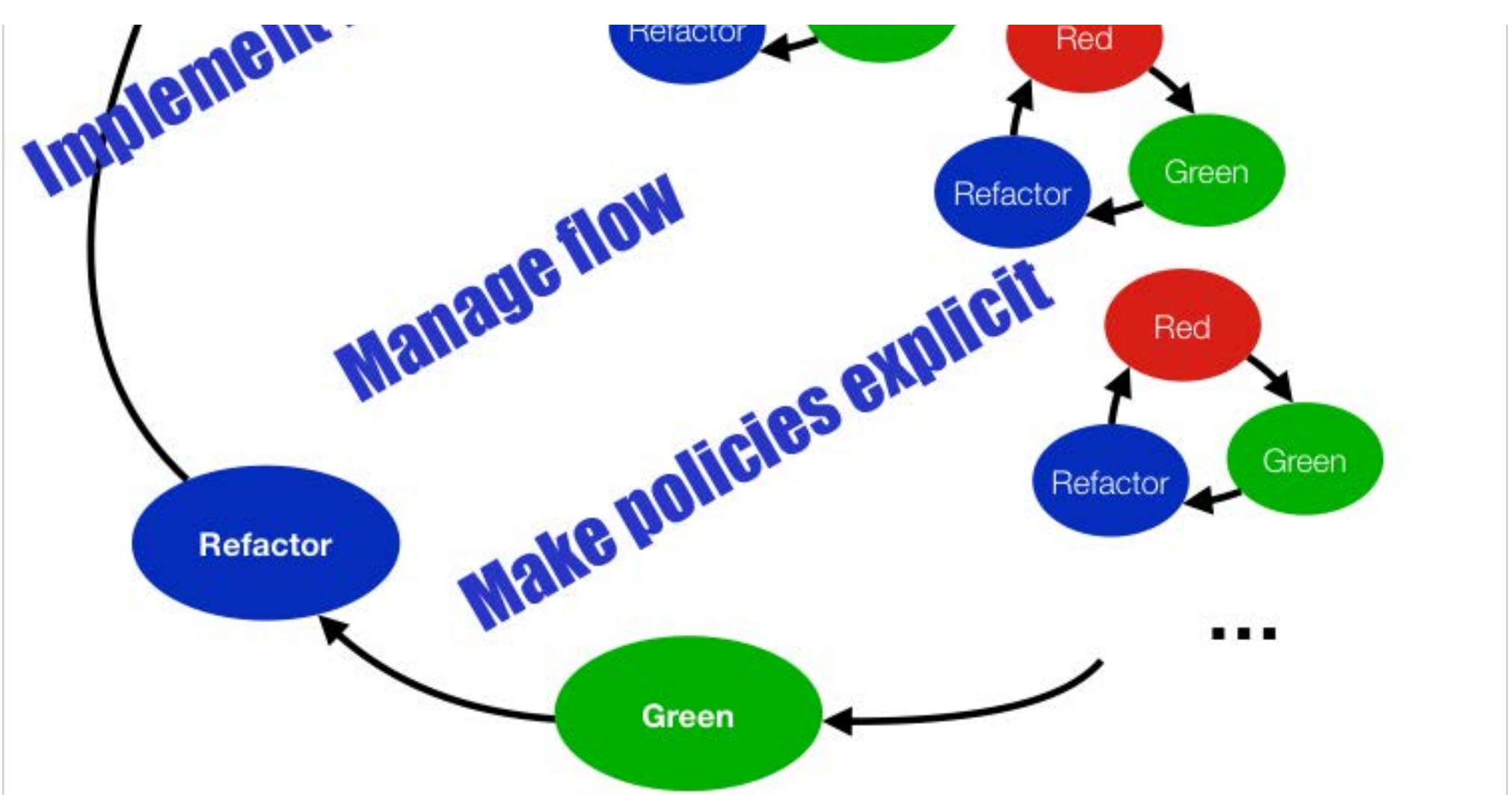
```
class TicTacToe
  BLANK = '-'

  def initialize
    @board = [
      [BLANK, BLANK, BLANK],
      [BLANK, BLANK, BLANK],
      [BLANK, BLANK, BLANK]
    ]
  end

  def play(row, column)
    @board[row][column] = 'X'
  end
end
```

Acceptance Test Driven Development





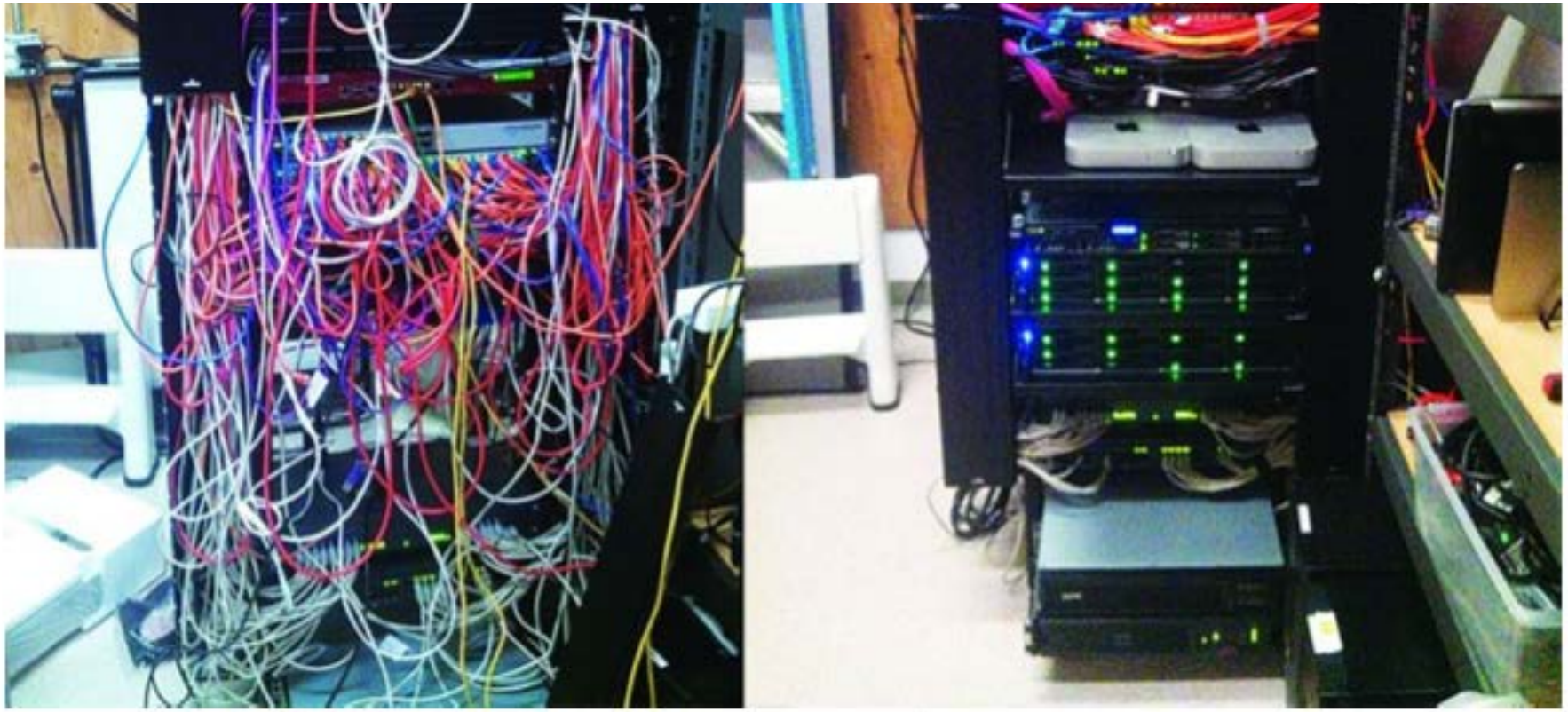
Acceptance Test Driven Development (ATDD)

Visualize

Given Toni Traveller is booking a flight
When she completes payment
Then she receives an email confirmation

Improve collaboratively, evolve experimen
(using models)

Initially, respect current processes, roles,
responsibilities & job titles



Refactoring

Refactoring (extract method)

```
def calculate_gross_domestic_product(approach)
  if approach == “income”
    national_income
      + indirect_business_taxes
      + depreciation
      + net_foreign_factor_income
  else
    consumer_spending
      + business_investment
      + government_spending
      + (exports - imports)
  end
end
```

Refactoring (extract method)

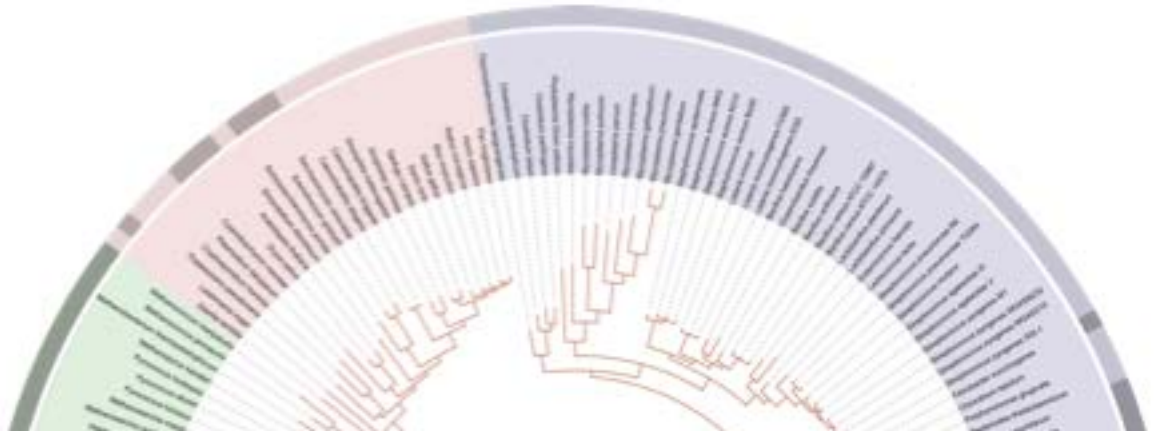
```
def calculate_gross_domestic_product(approach)
```

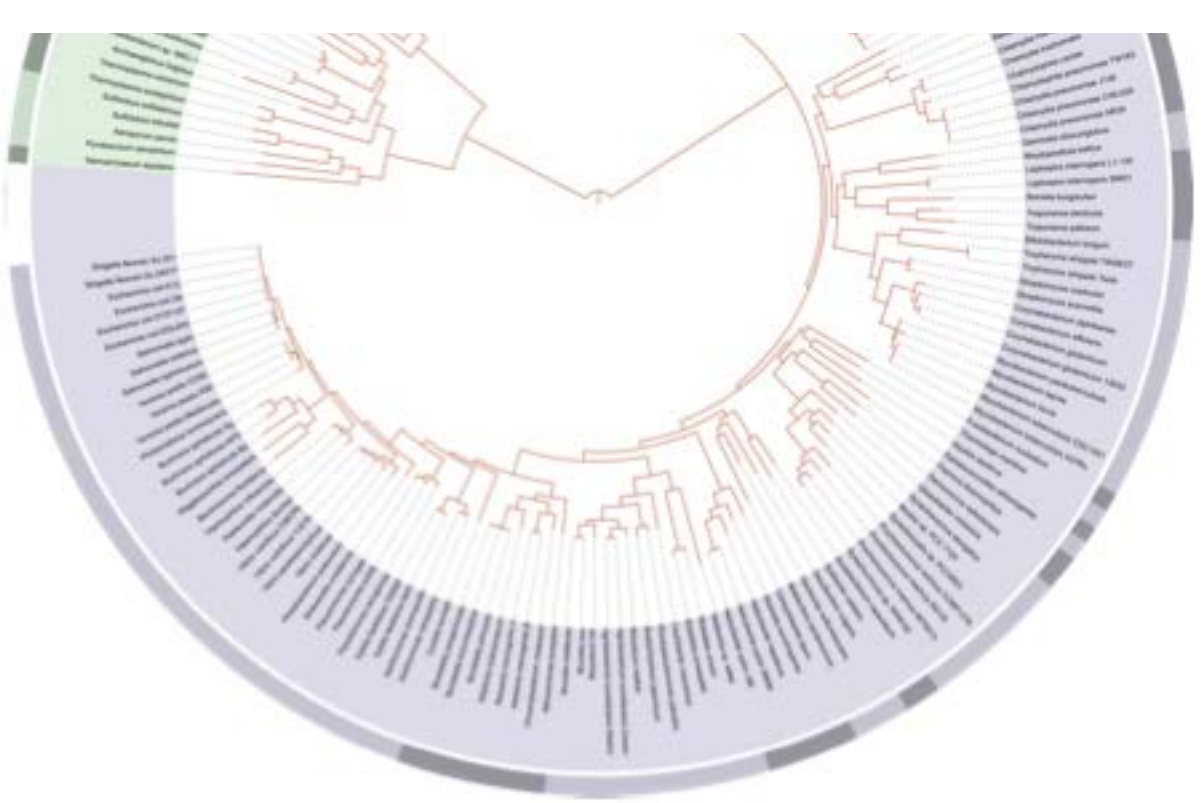
```
approach == "income"
  ? gdp_income_approach
  : gdp_spending_approach
end
```

Agree to pursue evolutionary change

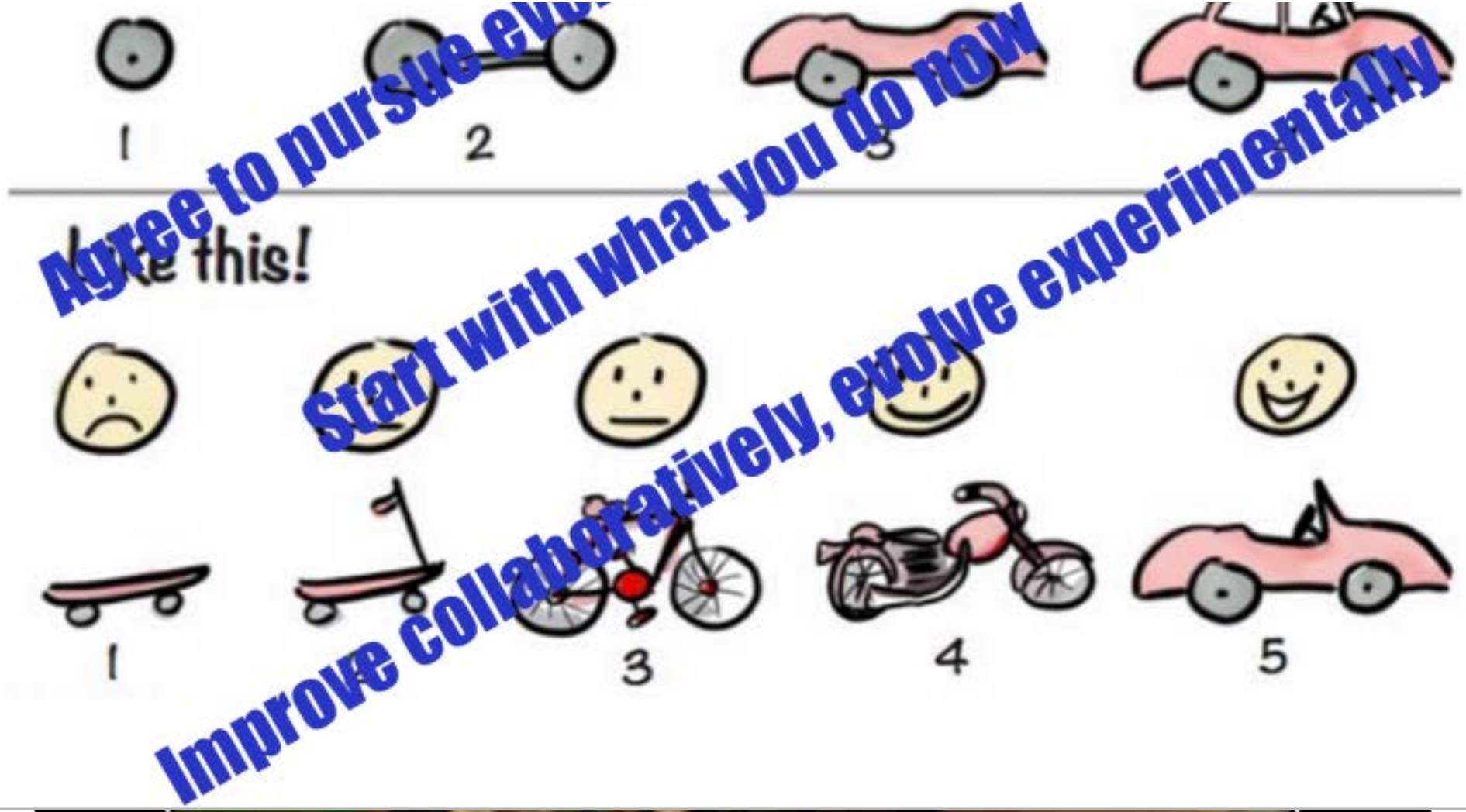
```
def gdp_income_approach
  national_income
    + indirect_business_taxes
    + depreciation
    + net_foreign_factor_income
end
```

```
def gdp_spending_approach
    consumer_spending
      + business_investment
      + government_spending
      + (exports - imports)
end
```

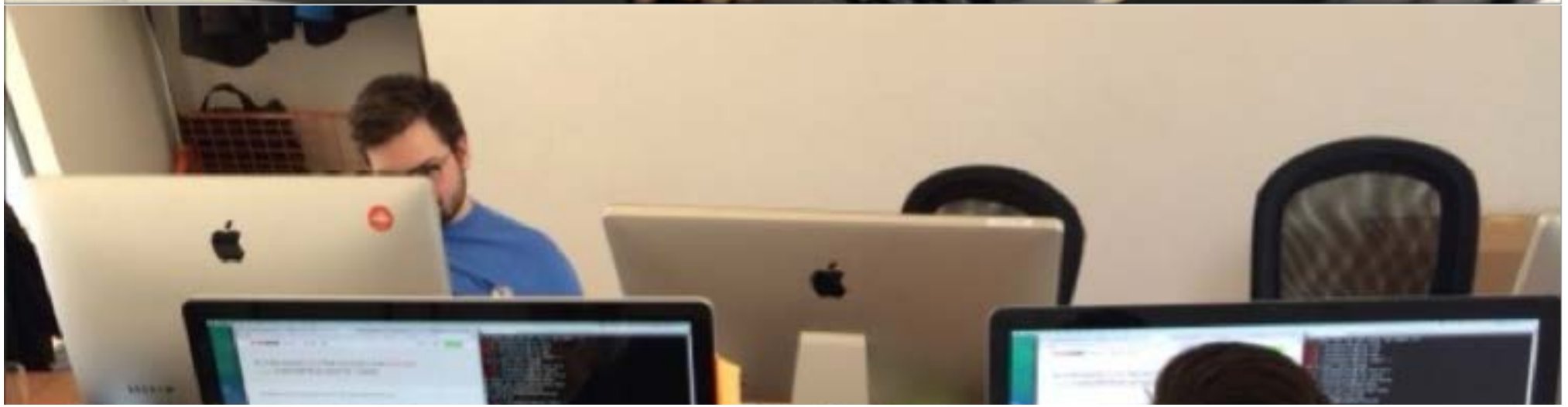


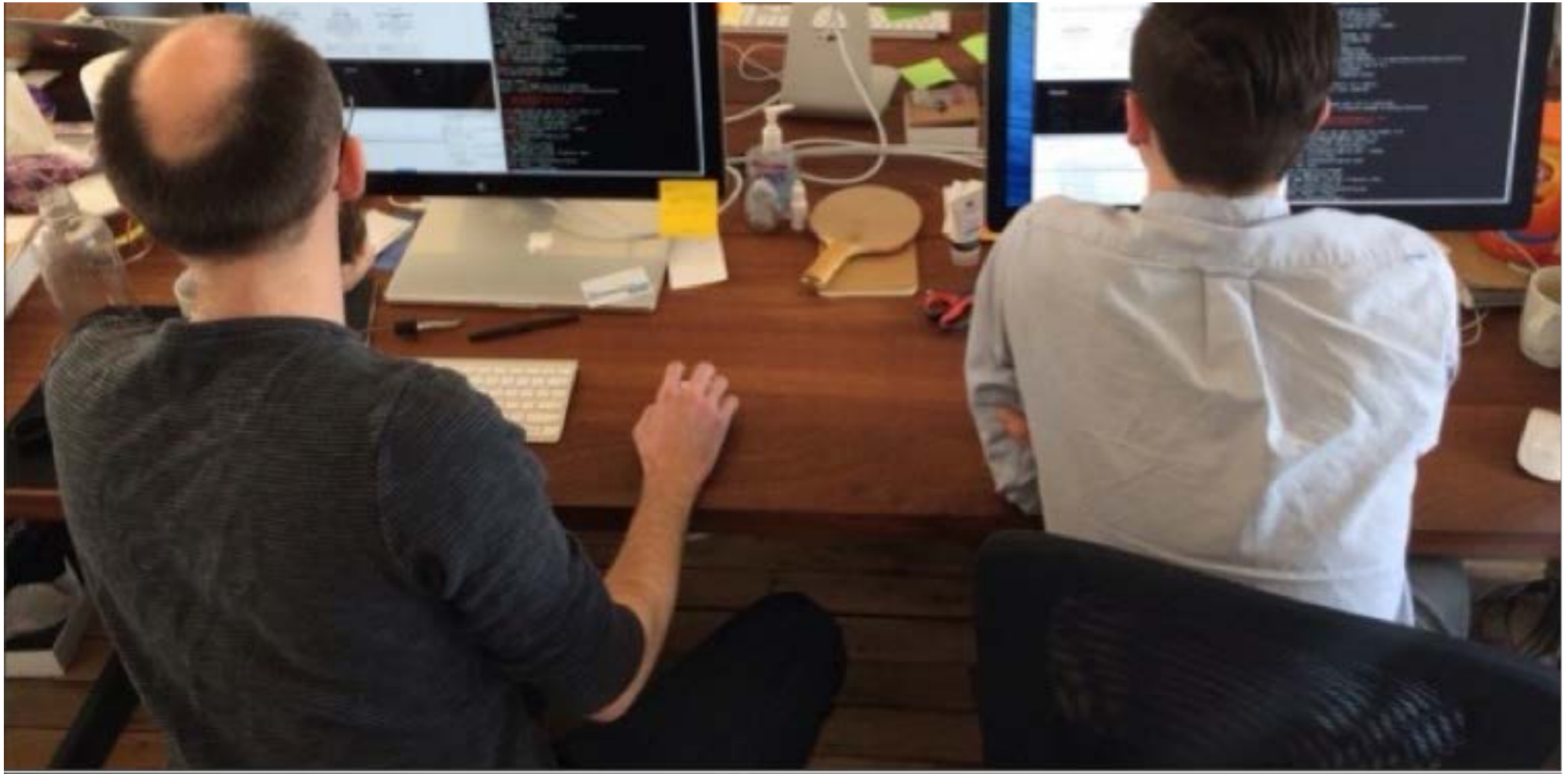


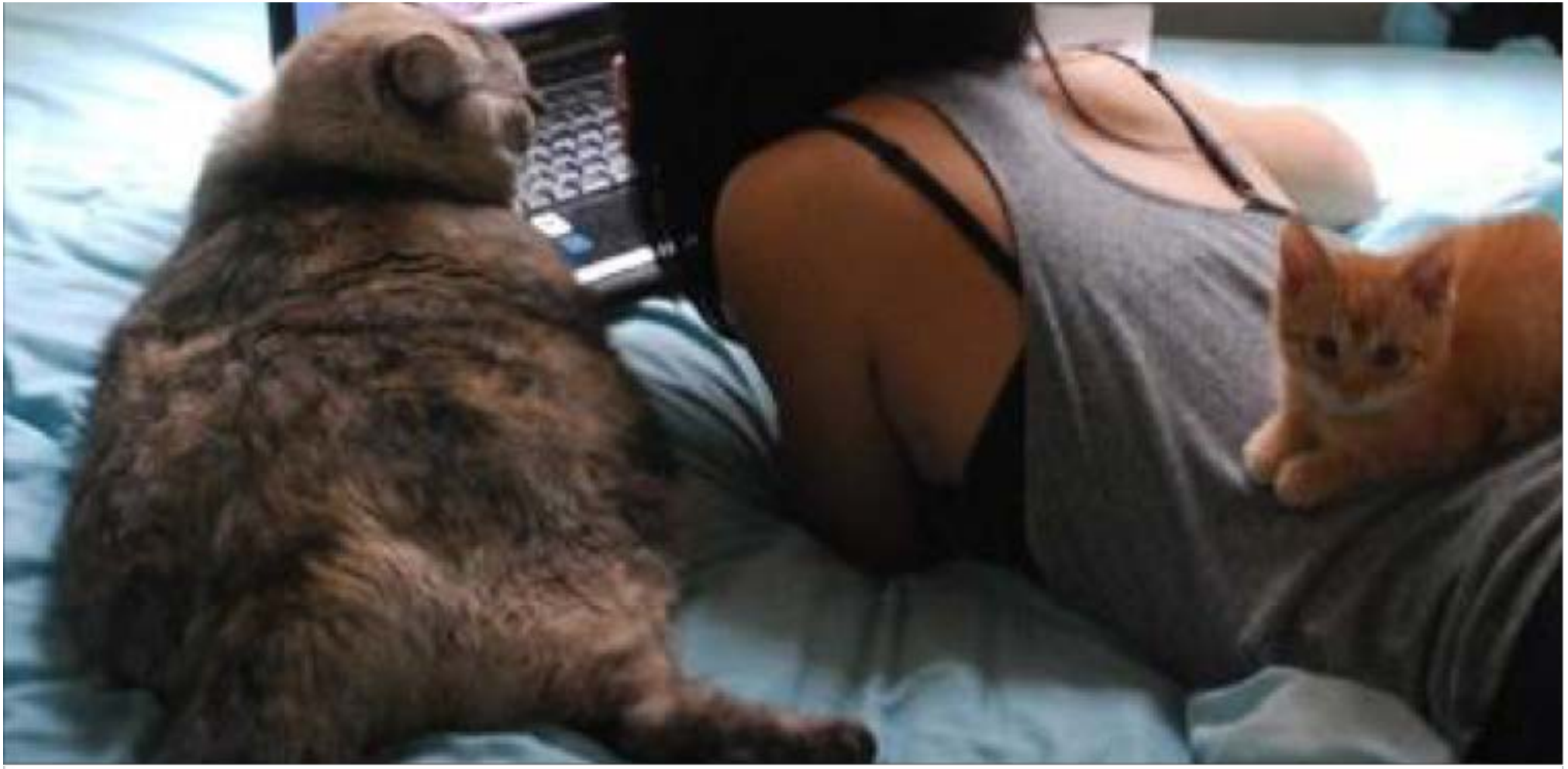
Evolutionary Design


1
2
3
this!
1
3
4
5
Agree to pursue evo
Start with what you do now
Improve Collaboratively, evolve experimentally

Pair Programming







Pair Programming

Limit

Encourage acts of leadership at every le
your organization

Implement feedback loops

Mob Programming

A Whole Team Approach

Mob Programming

Mob Programming

Some very unscientific data

- 4 - Encourage acts of leadership at every level in your organization
- 2 - Agree to pursue evolutionary change

- 2 - Initially, respect current processes, roles, responsibilities & job titles
- 1 - Start with what you do now

- 10 - Implement feedback loops
- 8 - Limit WIP
- 8 - Improve collaboratively, evolve experimentally (using models)
- 5 - Visualize
- 5 - Manage flow
- 5 - Make policies explicit

Photo Credits

- https://www.flickr.com/photos/oldben/2322196242
- http://nelsonwells.net/wp-content/uploads/2012/03/pair_programming.jpg
- https://camo.githubusercontent.com/ef1defc6a8e99ec339c3de381ec7650757ea4b4c/687474703a2f2f637331302e6f72672f737031352f7265736f7572636573732f696d616765732f706169727270726f6772616d6d696e672e6a7067

- https://s3.amazonaws.com/insights-images-prod/pairprogramming_f0d3ae7ef121e981e150bfcae4ecb995.jpg
- http://blog.zuehlke.com/wp-content/uploads/2013/04/iStock_000006143865Medium-755x521.jpg
- http://images.wookmark.com/32334_bedroom-cat-cats-computer-cute-favim.com-128320.jpg
- https://www.spkaa.com/wp-content/uploads/2015/10/It_Compiles_Ship_It.png
- https://leanpub.com/4rulesofsimpledesign
- https://s3.amazonaws.com/wapopartners.com/wweek-wp/wp-content/uploads/2016/03/29165302/1three3.png
- http://idahostockimages.photoshelter.com/image/I0000JYOmipH_L7Q
- http://i.dailymail.co.uk/i/pix/2012/08/27/article-2194334-14778A33000005DC-112_634x435.jpg
- http://blog.crisp.se/2016/01/25/henrikkniberg/making-sense-of-mvp
- https://upload.wikimedia.org/wikipedia/commons/thumb/1/11/Tree_of_life_SVG.svg/2000px-Tree_of_life_SVG.svg.png
- http://cdn8.themanual.com/wp-content/uploads/2014/12/how_to_chop_wood.jpg

Technical Practices through a Kanban Lens

Nayan Hajratwala
nayan@chikli.com
@nhajratw
CHIKLI CONSULTING

Kanban Foundational Principles

- Start with what you do now
- Agree to pursue evolutionary change

- Initially, respect current processes, roles, responsibilities & job titles
- Encourage acts of leadership at every level in your organization

Kanban Core Practices

- Visualize
- Limit WIP

- Manage flow
- Make policies explicit
- Implement feedback loops
- Improve collaboratively, evolve experimentally (using models)

Some Extreme Programming Technical Practices

- Pair Programming
- Continuous Integration

- Simple Design
- Collective Code Ownership
- TDD
- Whole Teams

Some Other Technical Practices

- Acceptance Test Driven Development (ATDD)
- Mob Programming

- Clean Code
- Continuous Deployment
- Feature Toggles
- Refactoring
- Feature Splitting
- Evolutionary Design