How does a tester add value in this new construct?

- A Call to Action!
  - Understand the stack
  - Understand the tools
  - Understand the types of testing
  - Understand the customer
  - Understand where to contribute
- Irrespective of how your manager or team works, go forth!

DAMN IT JIM!
I'M A TESTER, NOT A MIRACLE WORKER
imgflip.com

Automation

- We can’t do DevOps without Automation (test, and other types):

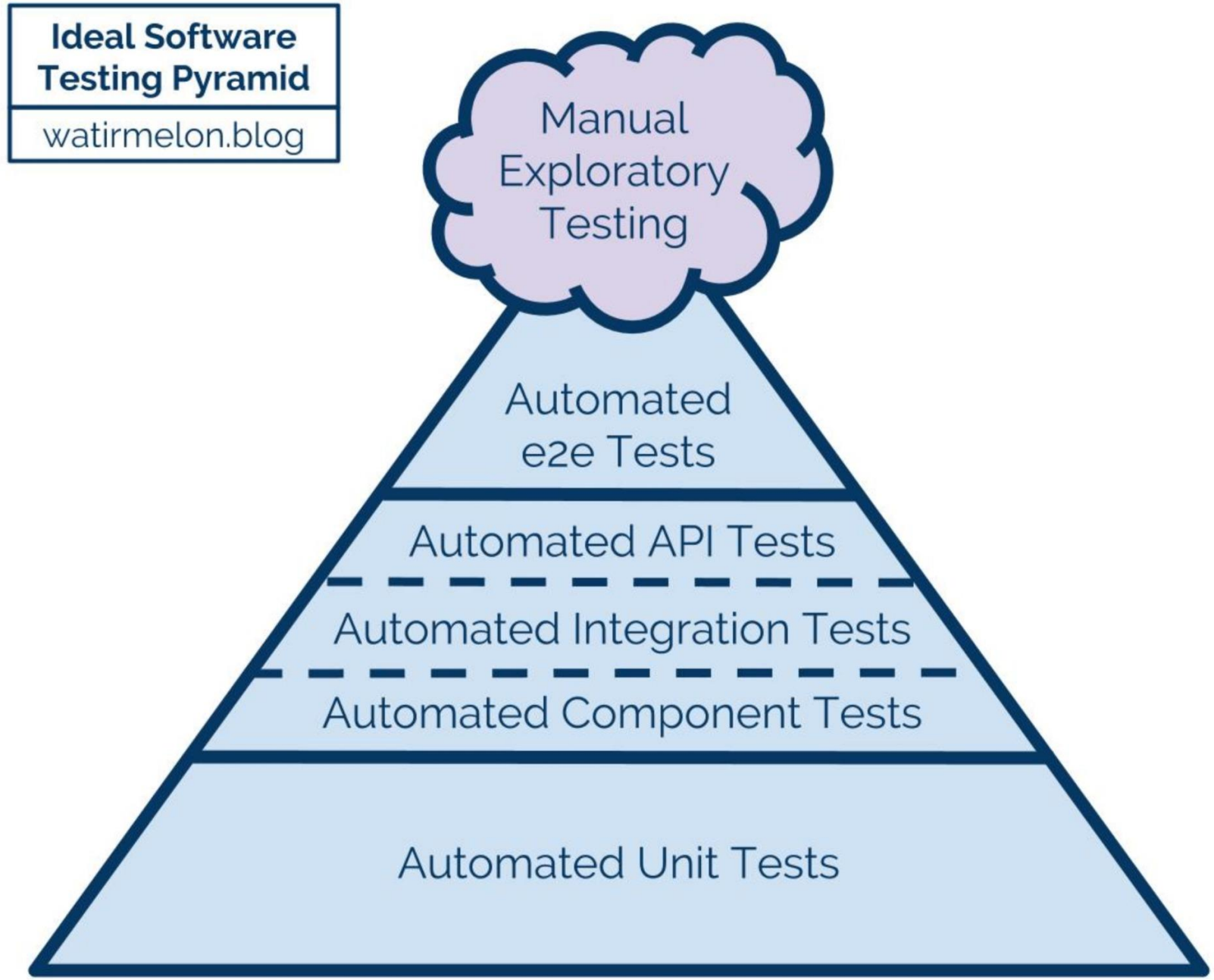
- **Mike D. Kail:** “Automation is a key component”
- **John Arundel:** “Without automation, there is no DevOps”
- **Gregory S. Bledsoe:** “Collaboration and automation are the two foundational elements of DevOps. Without automation, what you are doing isn’t DevOps”
- **Thorsten Heller:** “DevOps requires automation”
- **Eric Vanderburg:** “Automation is critical in DevOps”
- **Quentin Adam:** “ It’s one of our core values ”...

Test Automation

Good Tests Are….

- Implemented the “right way”:
  - If they’re GUI based: Tolerant to changes
  - Modularized
  - Data Driven
  - De-buggable
  - Version Controlled
  - Stored with the code they’re testing
  - De-coupled from environment
  - Use standard libraries
  - Managed data as part of the test (sets itself up)
  - Value of the test is continually reviewed
  - …

Testing Cloud



Technical Debt

- In our struggle to balance:
  - Build the right thing
  - Build the thing right
  - Build it in time
- Typically, we build tests in the wrong place

Technical Debt

- Tests are **a derivative artifact**
- Not dissimilar to a 'technical debt' - every change made requires more work to maintain tests
- Create items in your backlog to demonstrate the cost and benefit of paying it off
- "Push down the pyramid"

Why pay the debt?

- Moving the tests closer to where the function was developed
- To exercise the function more quickly
- To receive better feedback on change
- Lower maintenance costs

Manual Testing (top of the pyramid)

- Manual testing is **NOT** a bad thing
- “A repeatable test strategy can minimize the chance that you’ll discover new problems. A variable test strategy will at least not avoid finding bugs”
  - James Bach
- The same argument can be made about Automated Tests

Manual Testing

- Actually, there's a lot of manual testing that goes on
- Exploratory, Business Representation, Walkthroughs/Showcases (demos are great)
- Our job is to make this manual testing AS EFFICIENT AS POSSIBLE!
  - Only manually test what needs to be manually tested

Where do we put it?

- With EVERYTHING else - in version control
  - Code
  - Configuration
  - DB Schemas
  - Tests
  - Infrastructure

Version Control (probably GIT)

- There’s going to be:
  - Branches
  - Merges
  - Pull Requests (aka merge requests)
  - Triggers
  - Tags
  - Squashing
  - Rebasing
  - Cloning
  - Forking
  - …
- It’s going to be GREAT!

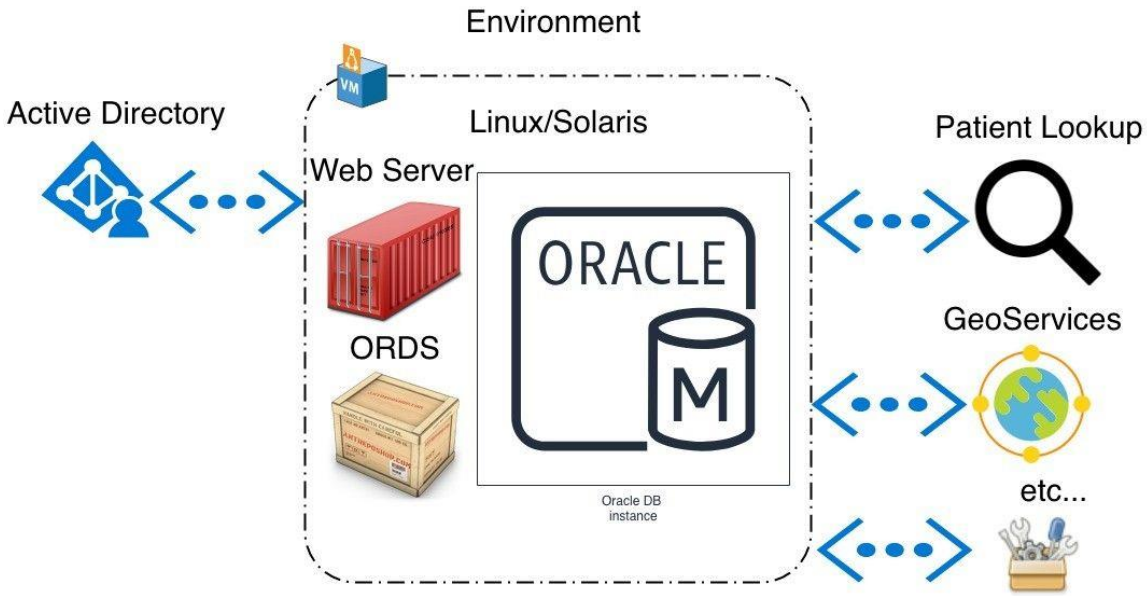
This talk isn’t about Git ☹

- However, why so important?
  - Single source of truth
  - Everyone knows where to look
  - Pull/Merge requests – Governance around peer reviews
  - Testers get a view of code, and developers get a view of tests
  - Full history of everything

Not just ‘Test’ Automation

- Infrastructure Automation and Verification (testing)
  - Provisioning
  - Configuring
  - Orchestrating
  - Checking

It’s all about the STACK



Infrastructure as Code (IaC) is cool ☺

- Docker
- Kubernetes
- OpenShift
- Puppet
- Chef
- Terraform
- Ansible
- SaltStack
- Service Virtulisation
- …

Capture the recipe

- We address the problem of "Environmental drift”
- Environments are available – no downtime
- Aim for production to be same as every other environment
- Enables parallel testing (tests don’t pollute each other’s environment)
- We approach a situation where we have true, **ephemeral** environments

- OPS are here to help!

Test the Infrastructure

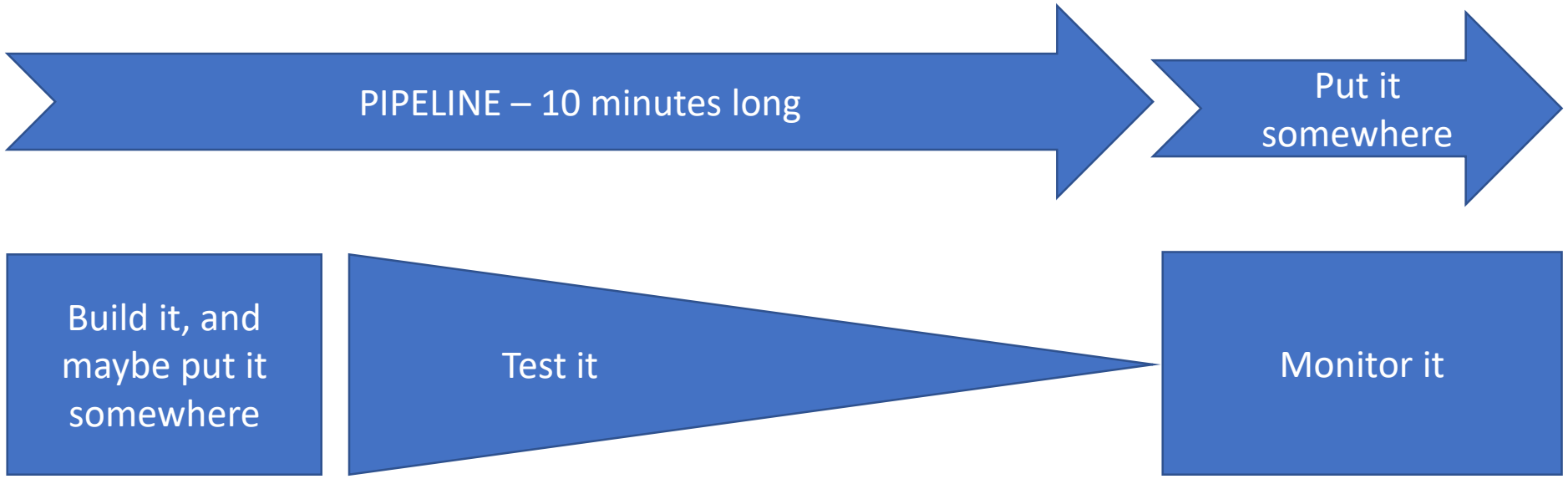
- How do we know that our ephemeral environment is ‘orchestrated’ properly?

- Write a test that proves the infrastructure is good – against the recipe
  - https://serverspec.org/
  - Execute your test as part of the pipeline

The Pipeline…

- Continuous Integration:
  - Everyone commits every day
  - Confident (and demonstrable) coverage of tests
- Continuous Delivery:
  - We *can* go to production, but we choose not to, the business might want intervene
- Continuous Deployment:
  - Every change goes to production

A “Pipeline”


PIPELINE – 10 minutes long
Put it somewhere
Build it, and maybe put it somewhere
Test it
Monitor it

The Pipeline…

- 10 Minute Build:

- “The 10-minute build practice is an extreme programming practice where the code base is designed by the developer to be built automatically. The code base is also designed to test run in ten minutes or less. It is from the amount of time required for the code base to finish running all tests that the 10-minute build derives its name.”

https://explainagile.com/agile/xp-extreme-programming/practices/10-minute-build/

Tools: A shared and interoperable ecosystem

- Wiki
- Ticket system (force ranked backlog)
- Source Control
- CI Pipeline
- Binary Storage
- Infrastructure

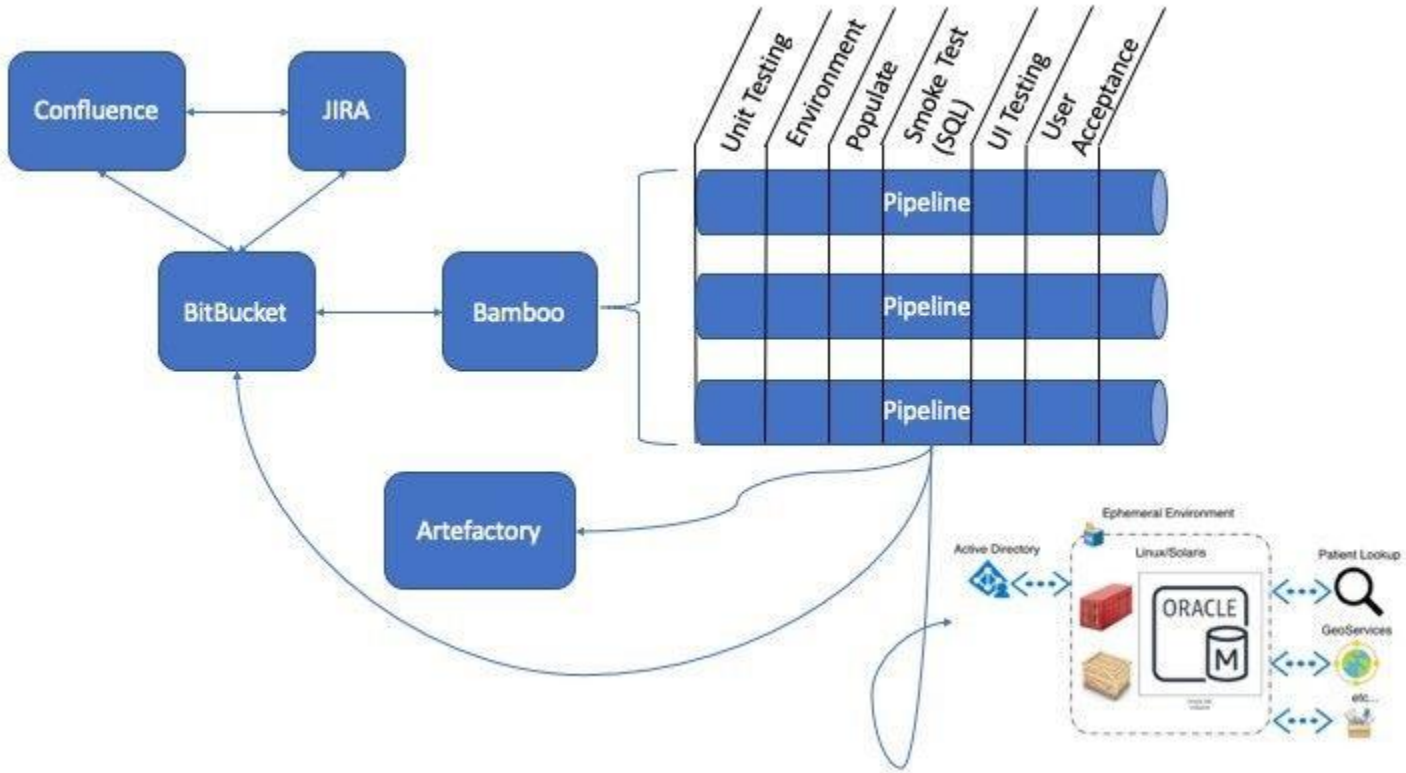
Methods…

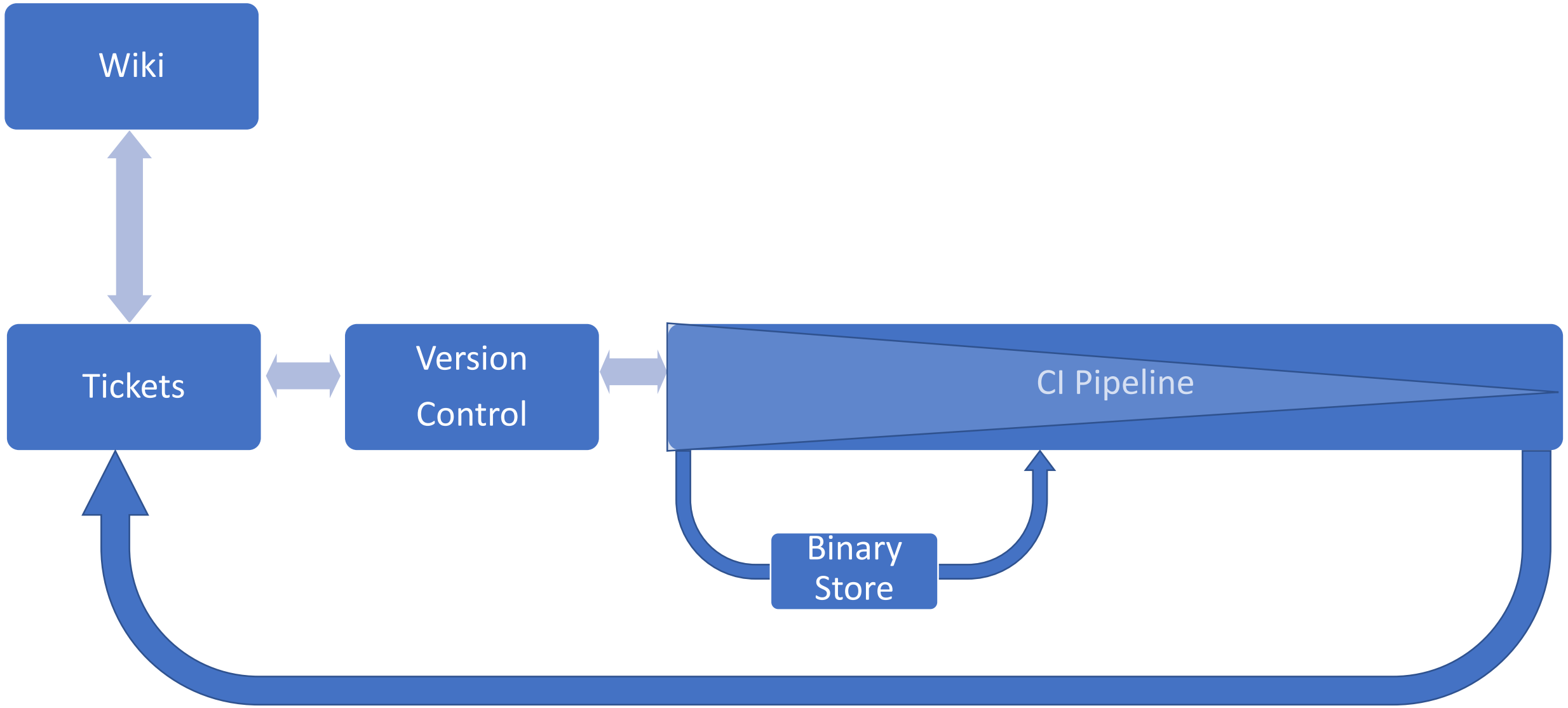
- Iterate
- Definition of Done
- Definition of Ready
- Single (awesome) Product Owner
- Visibility
- Communication
- Measurement

Measurement

- Metrics are VITAL for improving things – we do more with DevOps
- Build them into your process
- Monitor production too
- Learn and change, but not on a whim

How does this work?

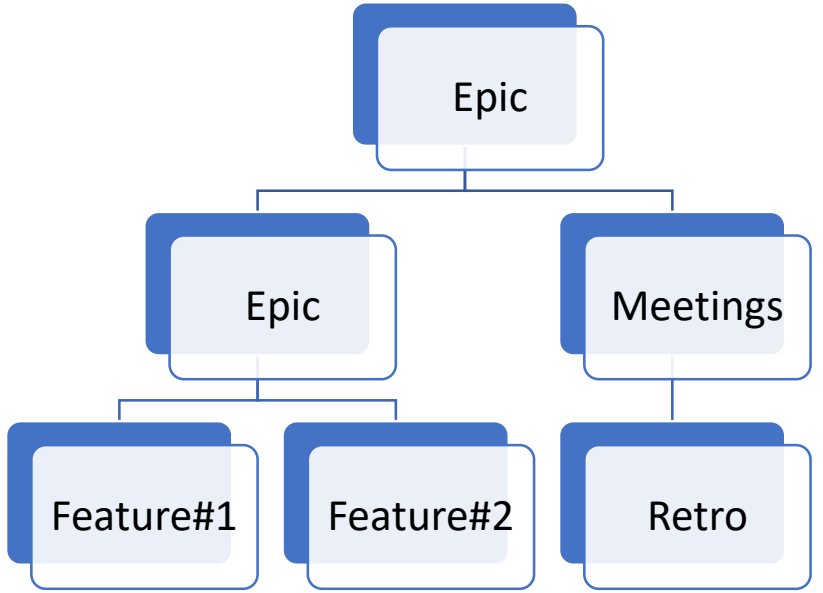


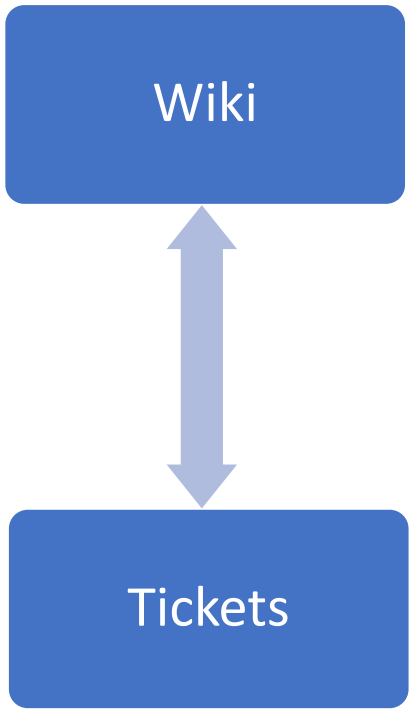

Wiki
Tickets
Version
Control
CI Pipeline
Binary
Store

Wiki

Wiki:

- English words
- Collaboration
- Versioning
- Transparency
- Templates



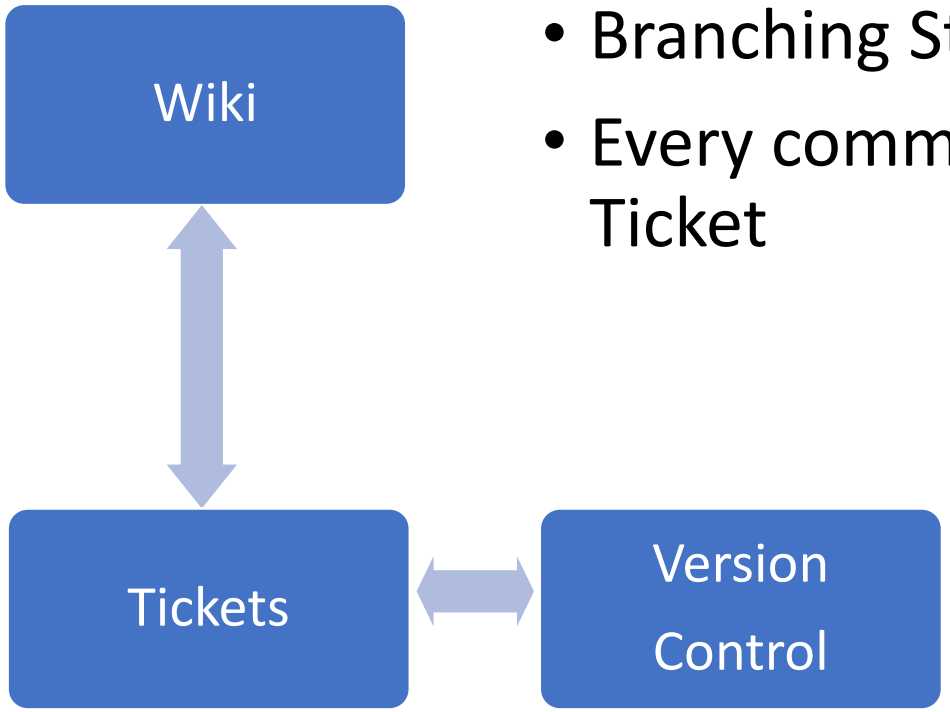


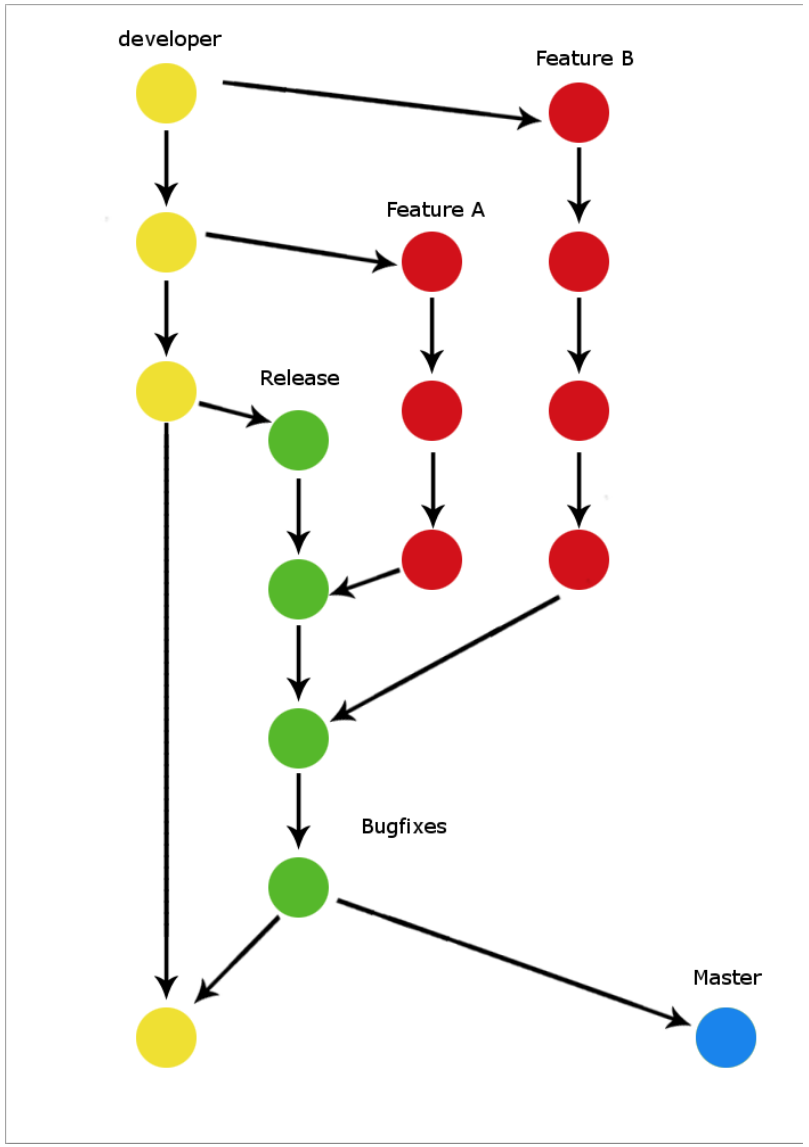
Tickets:

- Scrum/Kanban/etc…
- Captures data
- Visible/Radiator
- Definition of Done
- Integrated
- Kicks off feature branches

Version Control:

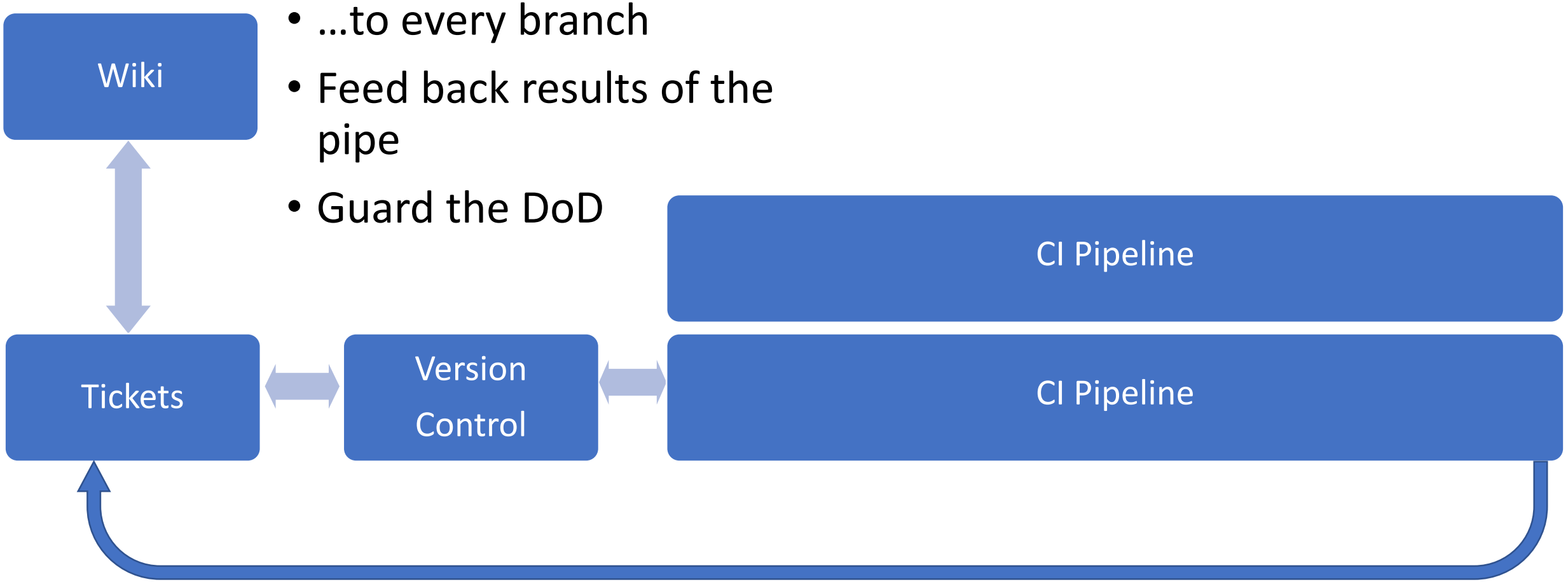
- Ticket == Branch
- Branching Strategy
- Every commit diff back to Ticket





Pipeline:

- Triggered on every commit
- …to every branch
- Feed back results of the pipe
- Guard the DoD



And finally…

Bug Fix Bingo

Rules of the Game

1. Bingo squares are marked off when a developer makes the matching statement during bux fix sessions.
2. Testers must call "Bingo" immediately upon completing a line of 5 squares either horizontally, vertically or diagonally.
3. Statements that arise as a result of a bug that later becomes "deferred", "as designed", or "not to be fixed" should be reclassified as not marked.
4. Bugs that are not reported in an incident report can not be used.
5. Statements should also be recorded against the bug in the defect tracking system for later confirmation.
6. Any tester that marks off all 25 statements should be awarded 2 weeks stress leave immediately.
7. Any developer found using all 25 statements should be seconded into the test group for a period of no less than 6 months for re-education.

1. "It works on my machine."	2. "Where were you when the program blew up?"	3. "Why do you want to do it that way?"	4. "You can't use that version on your system."	5. "Even though it doesn't work, how does it feel?"
6. "Did you check for a virus on your system?"	7. "Somebody must have changed my code."	8. "It works, but it hasn't been tested."	9. "THIS can't be the source of THAT."	10. "I can't test everything!"
11. "It's just some unlucky coincidence."	12. "You must have the wrong version."	13. "I haven't touched that module in weeks!"	14. "There is something funky in your data."	15. "What did you type in wrong to get it to crash?"
16. "It must be a hardware problem."	17. "How is that possible?"	18. "It worked yesterday."	19. "It's never done that before."	20. "That's weird..."
21. "That's scheduled to be fixed in the next release"	22. "Yes, we knew that would happen"	23. "Maybe we just don't support that platform"	24. "Its a feature, We just haven't updated the specs"	25. "Surely nobody is going to use the program like that"

Read what people are saying about Bug Fix Bingo !!!

"Thanks Bug Fix Bingo, You have made defect review meetings fun again !!!"

"I used to fall alseep when meeting with developers, now with BFB, I anticipate every word"

http://www.kjross.com.au/bingo

Questions?