SOFTWARE ENGINEERING FUNDAMENTALS

FOR AGILE DELIVERY MANAGERS

INTRODUCTION

Administration and Course Materials

- Start and Finish Times
- Fire and Emergency Procedures
- Breaks
- Toilets
- Questions
- Course Materials and Further Reading
- Knowledge Transfer Review

Course Timetable

Day	Sessions
1	• Introduction • Review of Software Engineering Processes • Coding Fundamentals and Tools
2	• Software Testing • Software Deployment

Delegate and Trainer Introductions

- Name
- Current and Recent Projects
- Software Engineering Experience
- Objectives for this Week and Beyond

Course Approach

- Knowledge transfer about Software Engineering tools and techniques
- Significance to Delivery Managers
- Scenarios
- Questions
- Conversations

Agile Delivery Manager Role

- build and maintain motivated teams, making sure there is an iterative plan to work towards
- protect the team and make sure the team collaborates, communicates and focuses on what is most important
- coach team members and others, facilitate continuous improvement and apply the most appropriate agile and lean tools and techniques for their environment
- proactively manage dependencies, overcome obstacles and get the best value against constraints
- understand a broad range of disciplines including: product design, **software engineering, testing**, user research, user experience design and aspects of strategy and policy development

4 key performance indicators (KPIs)

- **cost per transaction** - how much it costs the government each time someone completes the process
- **user satisfaction** - what percentage of users are satisfied with their experience of using the service
- **completion rate** - what percentage of transactions users successfully complete the end to end process
- **digital take-up** - what percentage of users choose the digital service to complete their task over non-digital channels

Case Study:

- Replacing a paper form based process
- E.g. Carer's credit application form
- This needs to be completed by the applicant (ID Service)
- Prerequisites and exclusions (other allowances, benefits including state pension)
- Assessment/certification by health & social care professional
- Approval by DWP
- Carer's credit to be notified to HMRC
- Future changes monitored and notified – maybe a cutoff reassessment and other events such as retirement or change of status of person cared for

Case Study (Example) User Stories

- As a carer I want to get carer's credit so I get a fair pension when I retire
- As a care recipient I want my carer(s) to get credit for the caring they provide to me
- As a health and social care professional I want to assess and certify carers so that they get carer's credit if they are eligible
- As the Service Owner I want people's data to be safe so service users will feel confident to use the service
- As a Customer Service Advisor I want to inform carer's credit service users of the status of their application so they are happy with the service

Energiser Exercise

REVIEW OF SOFTWARE ENGINEERING PRACTICES

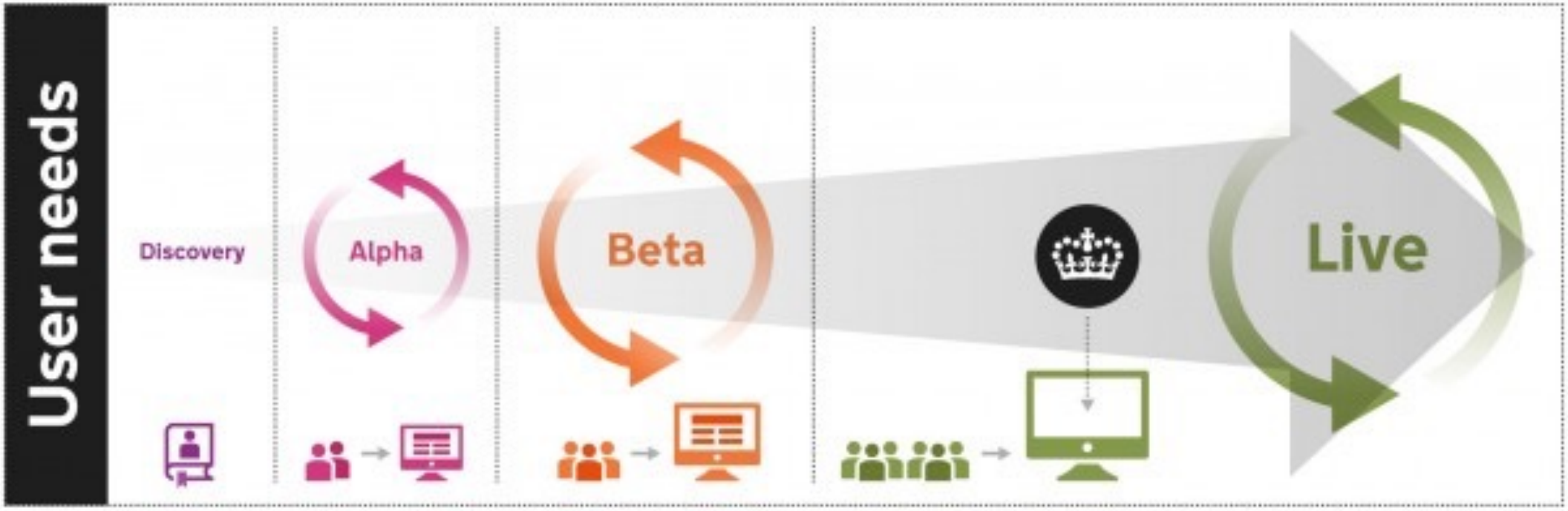
AGENDA – Review of Software Engineering Processes

- Agile Phases
- User Stories and Backlog
- Iterative Design
- TDA Review
- Incremental Delivery
- Scaling Delivery

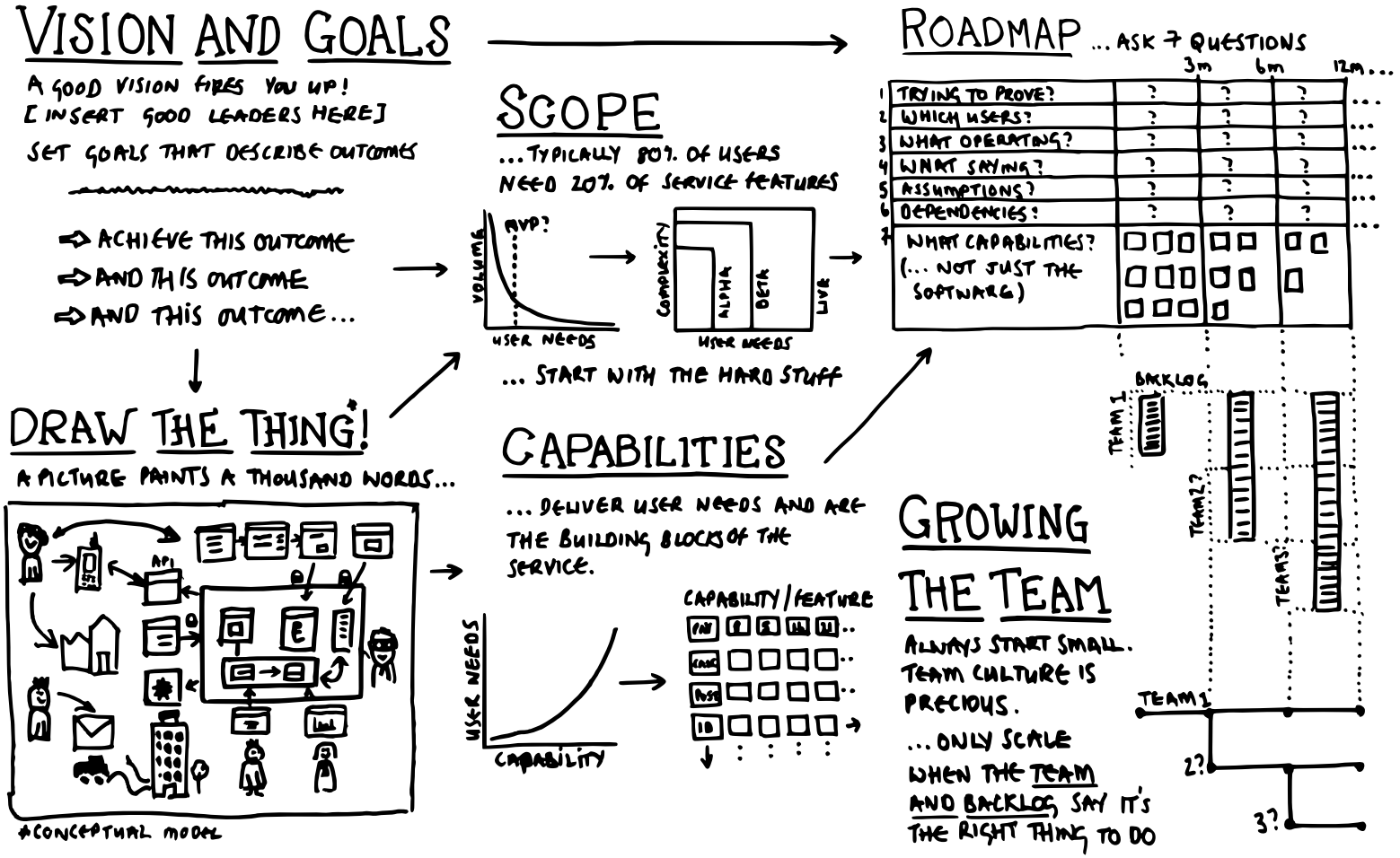
Course Approach – Review of SWE Processes

- Review of SWE Processes
- Significance to Delivery Managers
- Scenarios
- Questions
- Conversations

Agile Service Delivery Phases



Agile Service Delivery



Discovery Phase (4 – 8 Weeks)

Scope
Stakeholder approval
Team Members
Critical Success Factors
List of Related Services

Prioritised User Needs
- (including impairments and assisted digital support)

Prioritised User Stories
Stakeholders

Alpha Phase (8 Weeks)

1. Inception: getting the team together to set out goals
2. Iterations:
   - build prototypes
   - test them
   - learn, change and test again
3. Conclusion: you move on to the beta phase or end the project

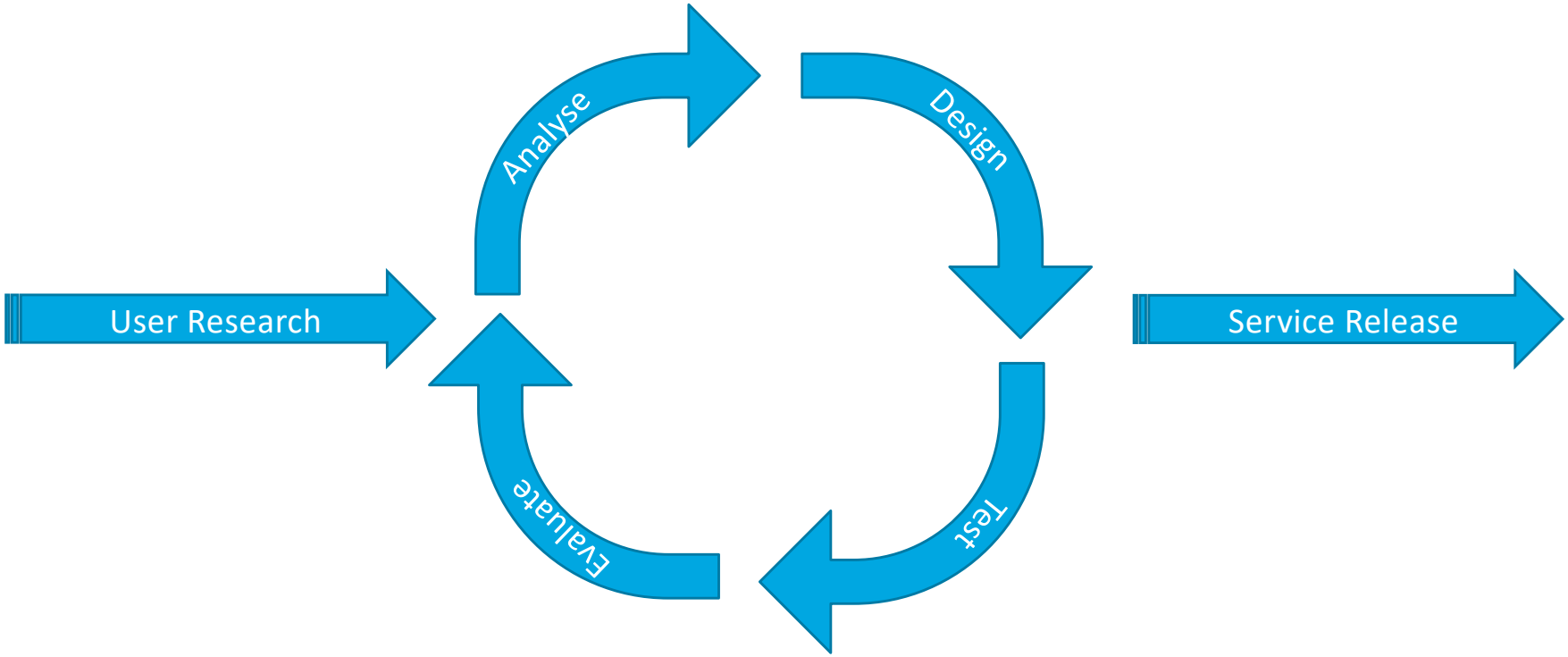
Prototypes

Fidelity

Type	What	Best for
Paper	Sketch out screens Put in front of a user Swap paper screens manually	Very early rough concept validation
Wireframe	Show links between screens Storyboard common scenarios Mock up with PowerPoint (or other tool)	Early validation of the information architecture, or common user journeys within a website or app
Visual design	Produce limited functionality screens Preload data and responses	Getting concrete feedback on proposed finished designs from customers
Code	Match live environment GOV.UK Prototype Kit Full UI navigation functionality Mock up business functionality	These are as close to the real experience that you will ever get allowing user stories to be tested

Speed

Iterative Design



"Great design is the iteration of good design"

M Cobanli

Beta Phase (Few Months)

1. Test with users
2. Solve technical challenges
3. Get service accredited
4. Plan launch
5. Private Beta
6. Public Beta

Live Phase (Many Months/Years)

1. User feedback
2. Analytics
3. Ongoing user research
4. Iterate for continuous improvement

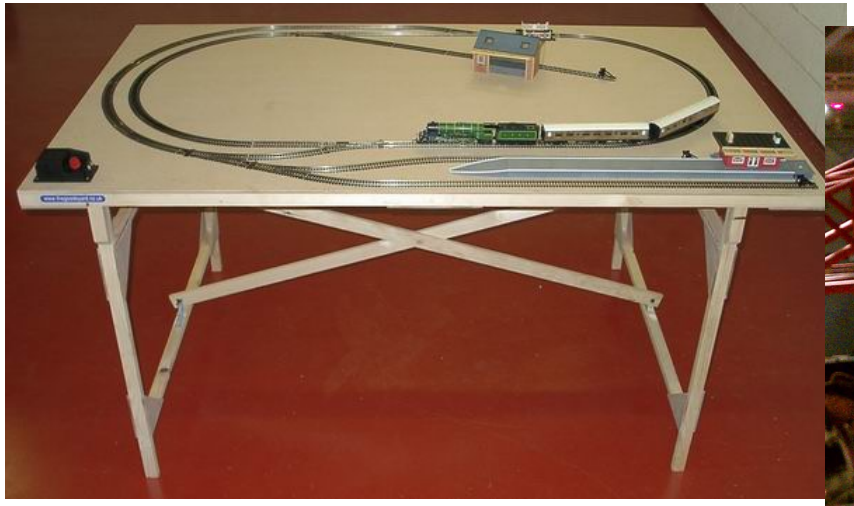
User Stories and Backlog

- Product – Service in focus
- User Story – Unit of delivery
- Epic – End to end User Story
- Theme – Set of related User Stories
- Product Backlog – User Stories awaiting completion
- Story points – estimate of effort
- Velocity – measure of delivery

TDA Review

- Technology choice – Tech Radar
- Open source, reusable code
- Vulnerability/penetration testing
- Performance
- Legacy systems
- Security

Incremental Delivery

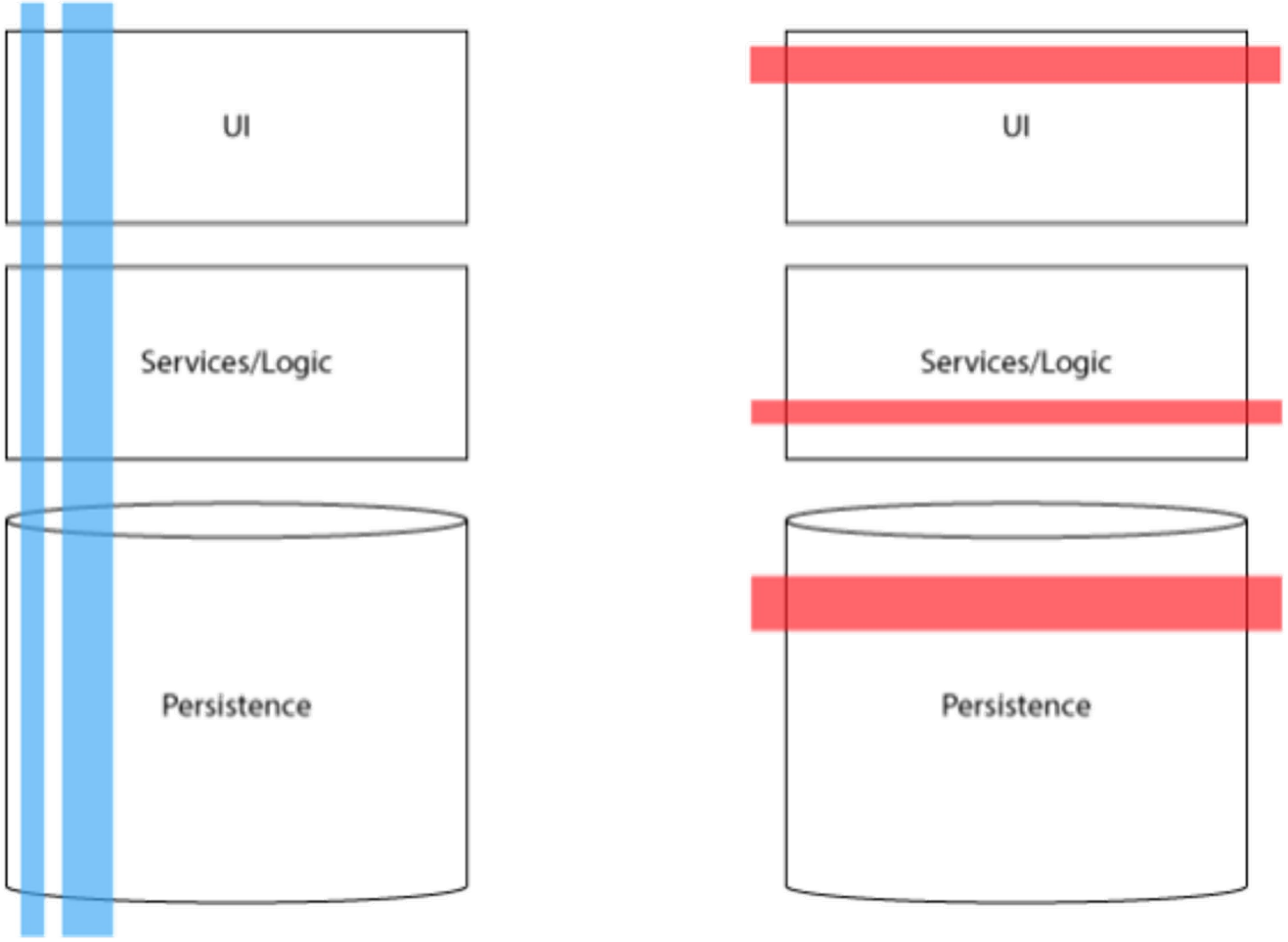


This

to

This

Story Slicing

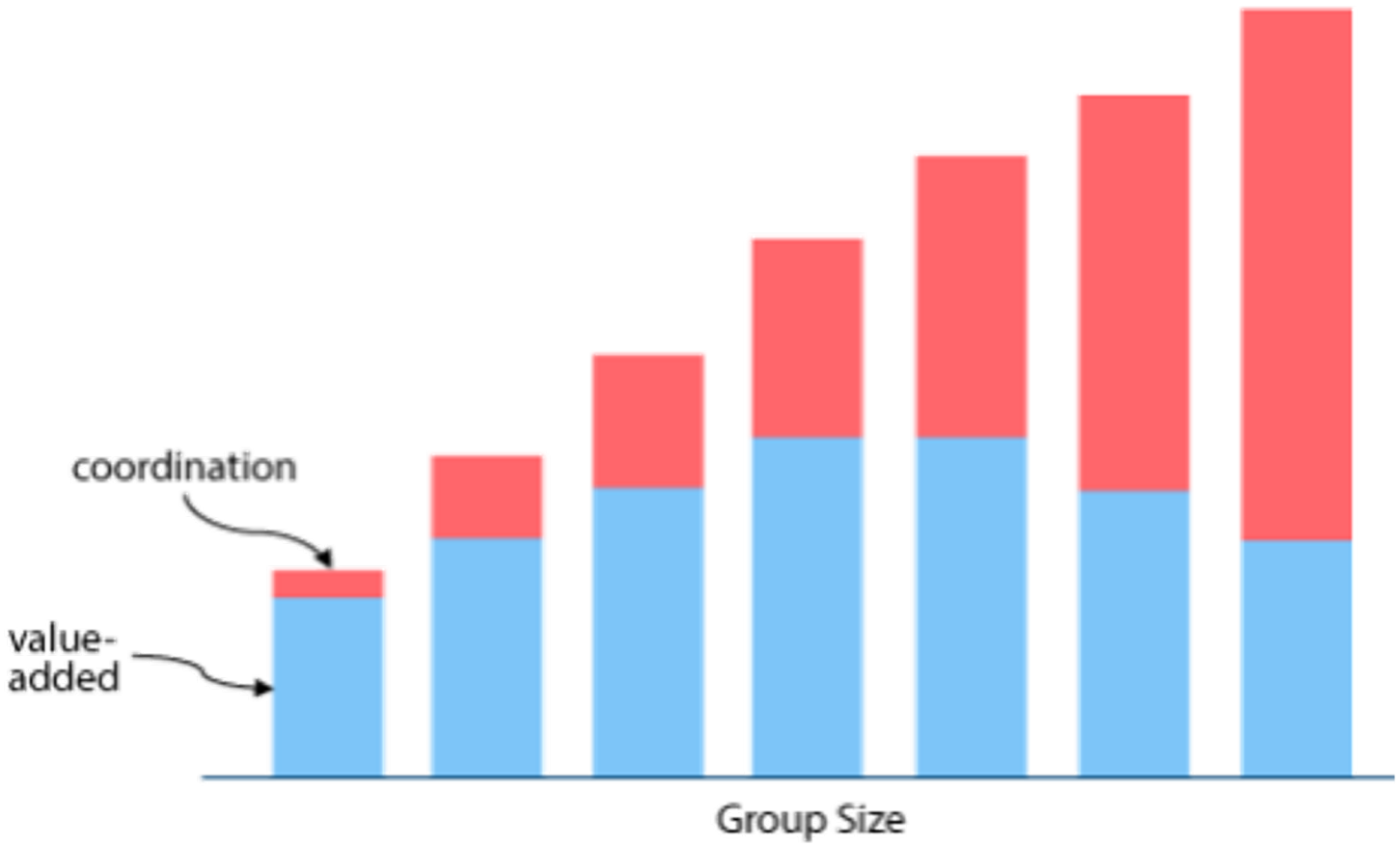


Story Splitting Patterns

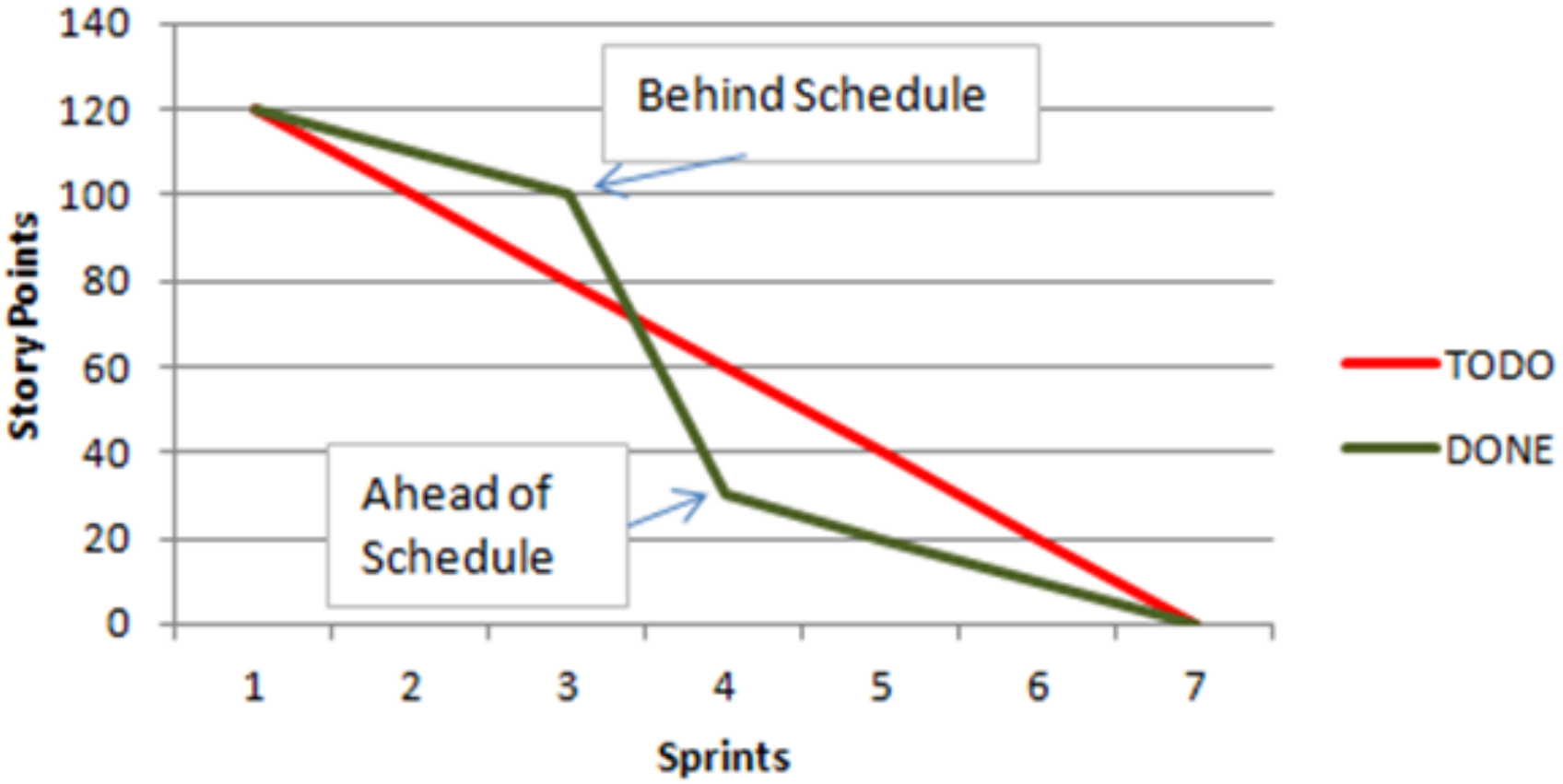
- Starting point: 6-10 stories per sprint (industry average)
- Use Pareto – 80% high value, 20% discard
- Split into equal point value – e.g. an 8pt story into 4 x 2pt stories
- Split by pattern:

Workflow steps	Business Rules	Effort
Simple first	Data driven	UI
Performance	CRUD	Spike

Team Scaling vs. Value Add



Burndown Chart - Velocity



REVIEW– Review of Software Engineering Processes

- Agile Phases
- User Stories and Backlog
- Iterative Design
- TDA Review
- Incremental Delivery
- Scaling Delivery

Further Reading – Review of Software Engineering Processes

www.gov.uk/service-manual/design/making-prototypes

agileforall.com/vertical-slices-and-scale/

agileforall.com/patterns-for-splitting-user-stories/

www.jamiearnold.com/blog/2015/02/17/how-to-scale-agile-service-delivery

www.gov.uk/service-manual/agile-delivery

www.gov.uk/service-manual/agile-delivery/how-the-discovery-phase-works

developer.epa.gov/guide/templates-guides/agile/discovery/

Further Reading – Review of Software Engineering Processes

Videos:

www.youtube.com/watch?v=JMjozqJS44M

www.youtube.com/watch?v=KWGBGTGryFk

www.youtube.com/watch?v=lusOgox4xMI

www.youtube.com/watch?v=DkyMCCnNI3Q

www.ted.com/talks/lang/eng/tom_wujec_build_a_tower.html

CODING FUNFAMENTALS AND TOOLS

AGENDA – Coding Fundamentals and Tools

- How to Turn User Stories into Code
- OO Programming
- Communities of Practice
- Coding Standards + Best Practice
- Estimation
- Code Quality
- Technology Choices and Stacks
- Microservices
- Documentation Standards
- Standards to Support Service Delivery
- Code Management with GIT
- Agile Coding Practices
- Development Environments
- Managing Technical Debt
- Data Storage and Big Data

Course Approach – Coding Fundamentals and Tools

- Knowledge transfer about Coding tools and techniques
- Significance to Delivery Managers
- Scenarios
- Questions
- Conversations

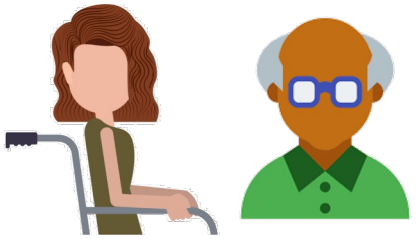
User Stories

- As a Carer I want to apply for Carer's Credit so I get a fair Pension when I retire
- As a Carer I want to be notified of the progress of my Application so I know that it is being processed fairly and speedily
- As a care recipient I want my carer(s) to get credit for the caring they provide to me so they will get a fair Pension
- As a Health and Social Care Professional I want to assess and Certify Looked After People so that their Carer can get Carer's Credit if they are eligible
- As the Service Owner I want people's data to be safe so service users will feel confident to use the service
- As the Service Owner I want to authorise eligible Carer's Credit Applications as quickly as possible so the Applicants can receive their entitlement
- As a Customer Service Advisor I want to inform Applicants for Carer's Credit of the status of their application so they are happy with the service

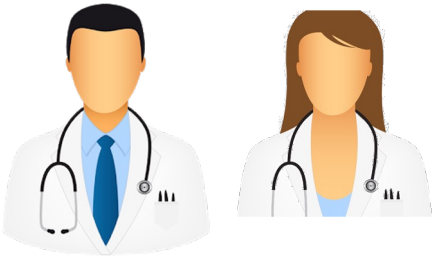
Object Orientated (Design &) Programming

Carer's Credit Application

Review Meeting



Looked After Person



Health Professional

Care Certificate

Phone Contact

Approver

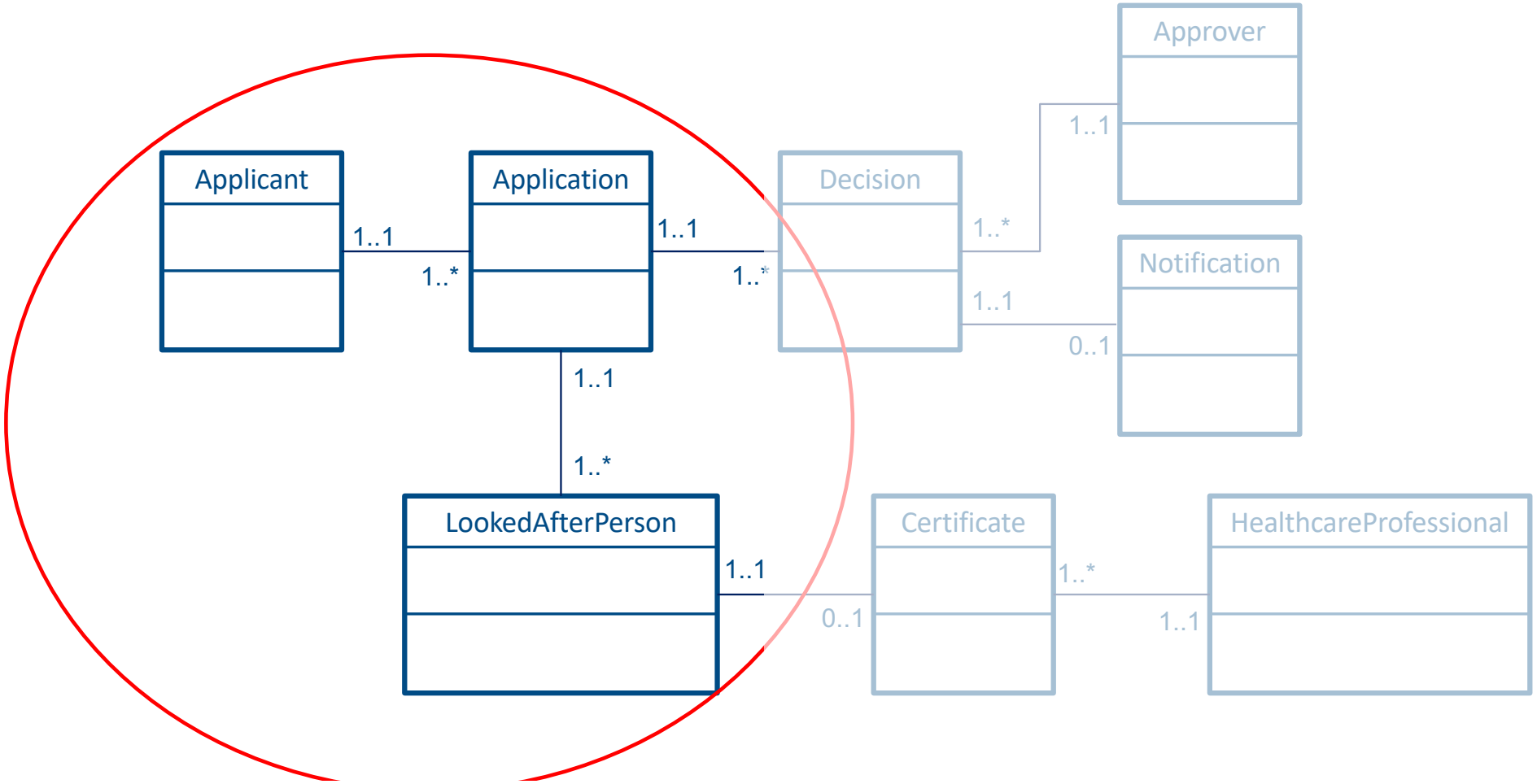
Applicant/Carer

Decision

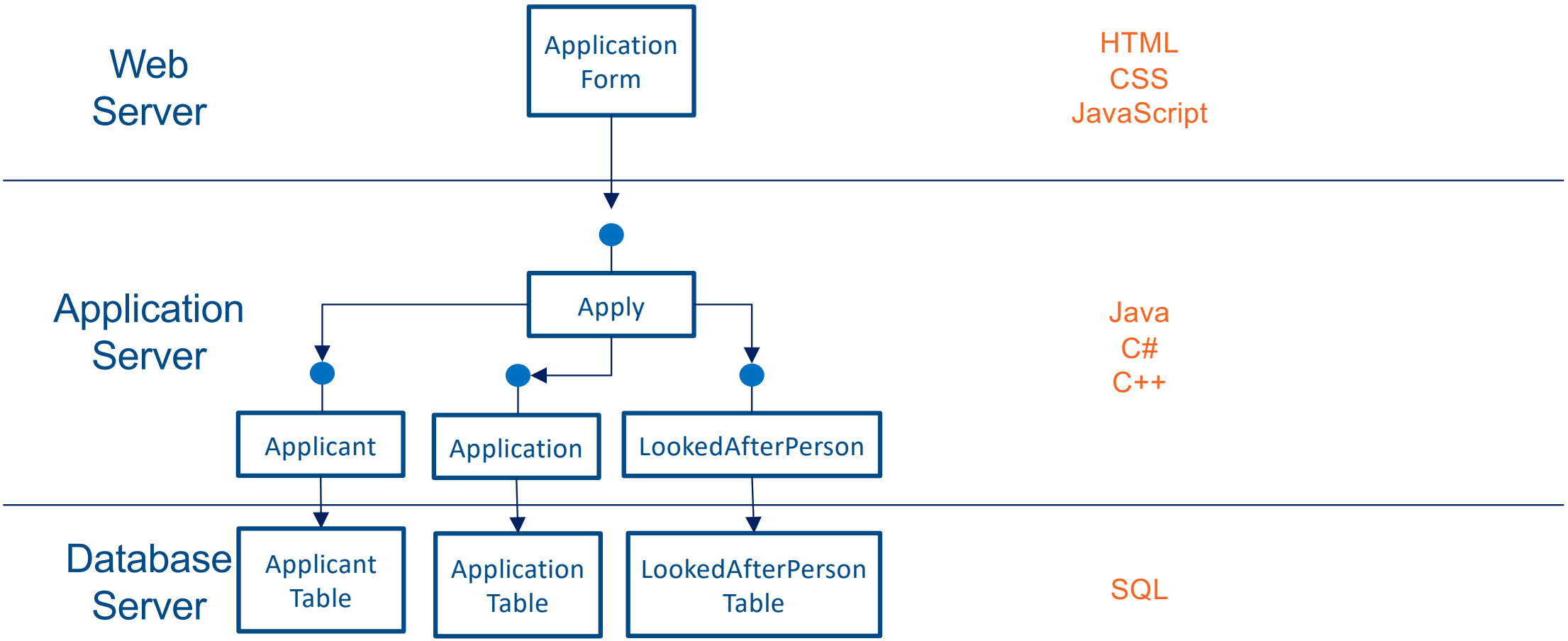
Authorisation

Notification

Domain Class Diagram



Multi-Tier Web Application



Apply Class

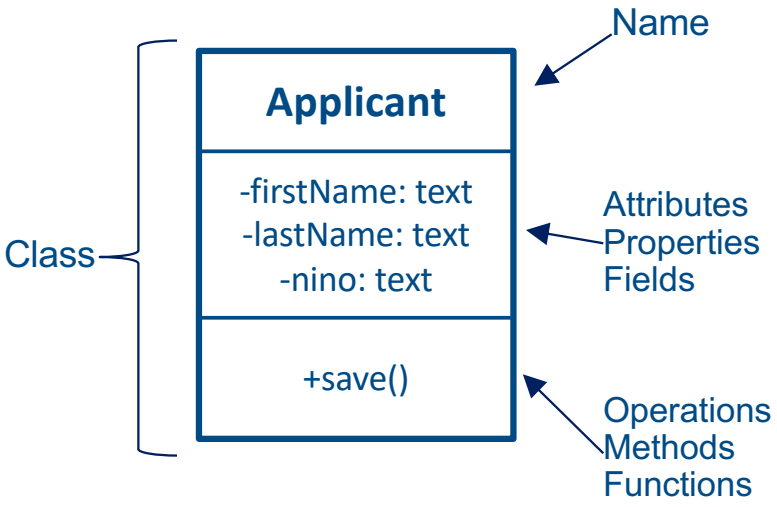
```
public class Apply{

   public void doPost(HttpServletRequest requestData,
      HttpServletResponse resp) throws ServletException, IOException {

      applicant = createNewApplicant(requestData);

      applicant.save();
   }
   private createNewApplicant(){
      // code to create applicant from form data
   }
}
```

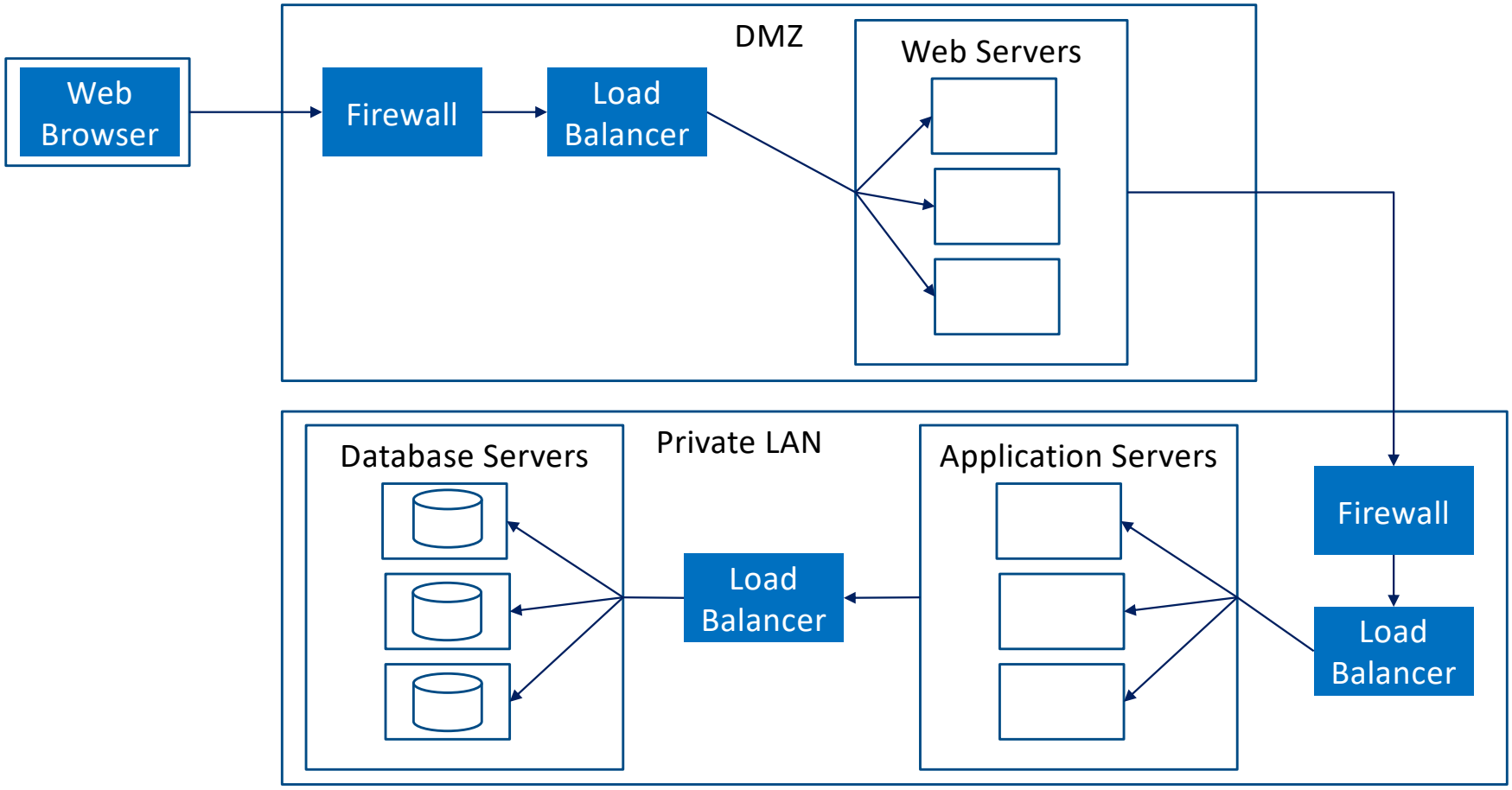
Applicant Class




```
public class Applicant{
    private String firstName;
    private String lastName;
    private String nino;

    public void save {
        // code to connect to database
        // code to save data to Applicant table
    }
}
```

Server Access and Security

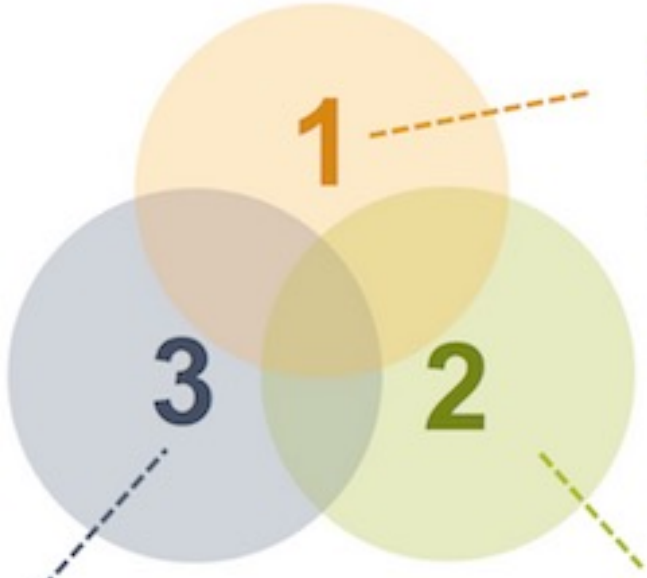


Communities of Practice

Engineering and Quality Assurance practice is composed of the following roles:

- Infrastructure Engineer
- Dev Ops Engineer
- Software Engineer
- Test Engineer
- Technical Lead
- QA Tester

Communities of Practice: Traits



Domain

What we care about

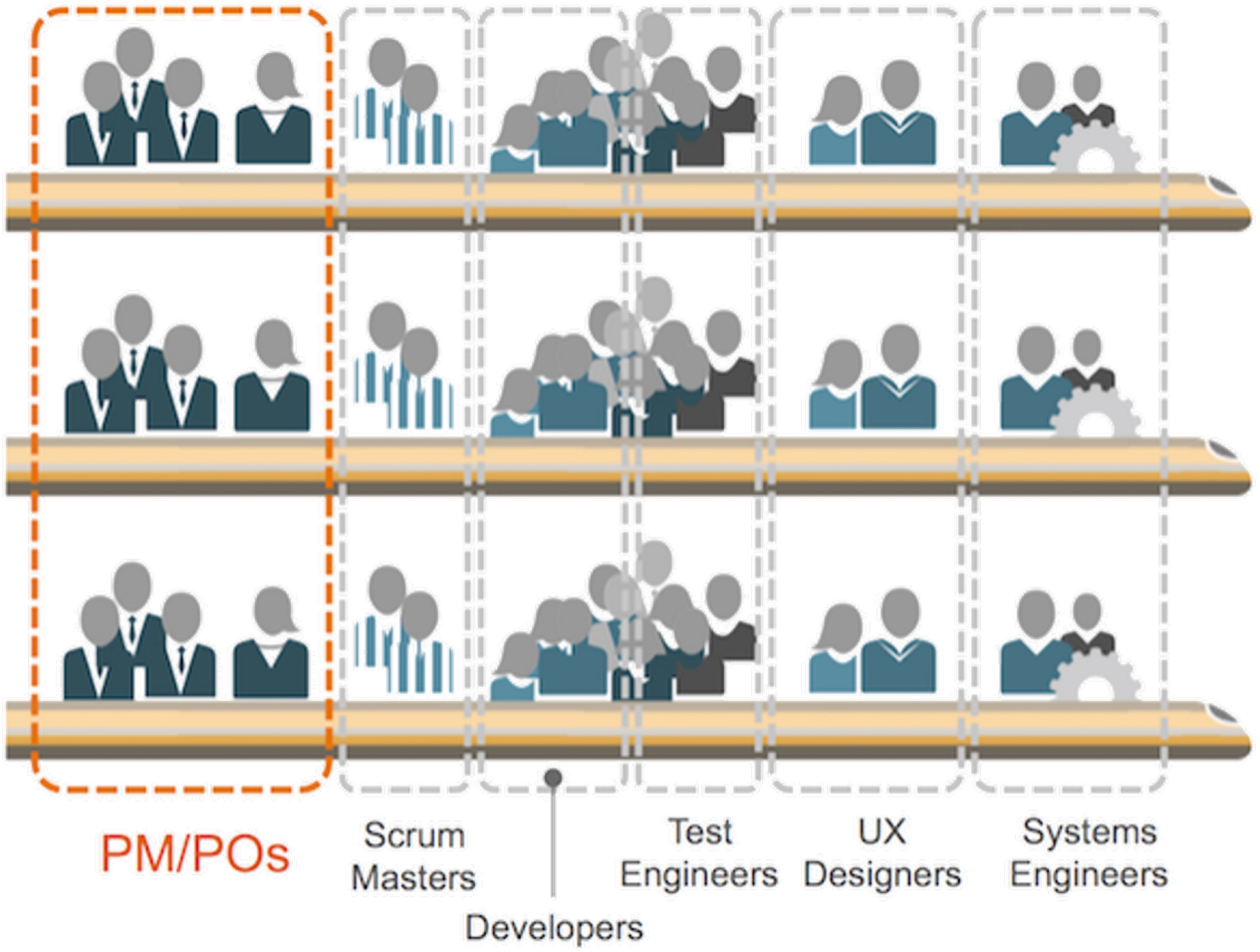
Community

Who cares about it

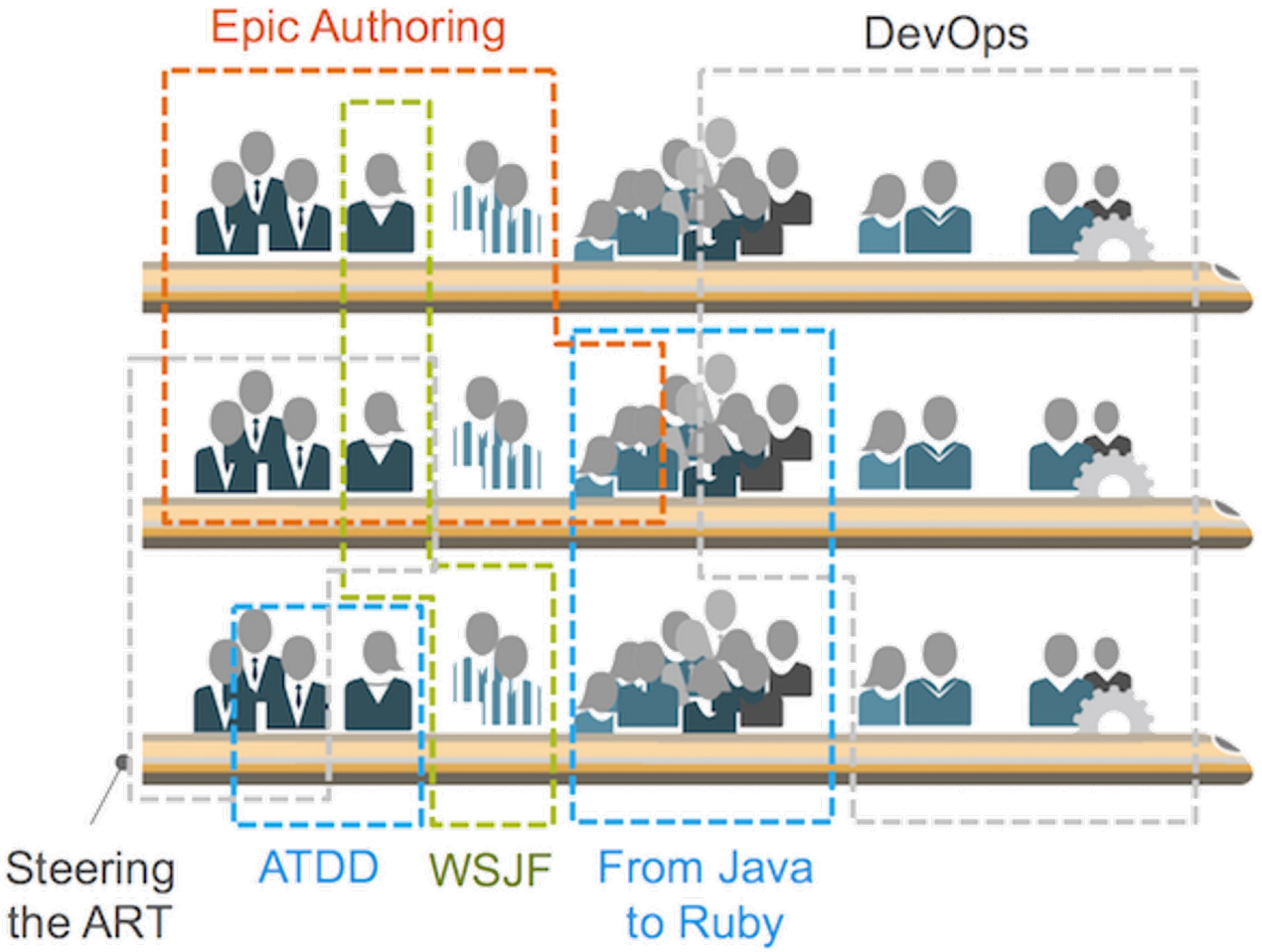
Practice

What we do together about it

Communities of Practice: Role-Based



Communities of Practice: Topic-Based



Communities of Practice: Participation

- Core team
- Active
- Occasional
- Peripheral
- Transactional

Code Smells

Bloater	Long Method Large Class Primitive Obsession	Long Parameter List Data Clumps
OO Abusers	Switch Statements Temporary Field Refused Bequest	Alternative Classes with Different Interfaces
Change Preventers	Divergent Change Shotgun Surgery	Parallel Inheritance Hierarchies
Dispensables	Comments Duplicate Code Lazy Class	Data Class Dead Code Speculative Generality
Couplers	Feature Envy Inappropriate Intimacy Message Chains	Middle Man Incomplete Library Class

Design Patterns

- Creational
  - For class instantiation using inheritance and object creation using delegation.
- Structural
  - For class and object composition to reuse code components.
- Behavioural
  - For managing communication and messaging between objects.
- Anti-patterns
  - Similar to code smells

Coding Principles

- DRY (Don’t repeat yourself!)
- KISS (Keep it simple stupid!)
- YAGNI (You ain’t gonna need it!)
- Simplicity:
  1. Code + tests must communicate everything (and be understood) so no need for comments
  2. No duplicate code
  3. Fewest possible classes/modules
  4. Minimum data (attributes, arguments)
  5. Fewest possible methods/operations
  6. Shortest possible methods/operations

SOLID Principles

- **Single Responsibility**
  A class should have one and only one reason to change, meaning that a class should have only one job
- **Open Closed**
  Objects or entities should be open for extension, but closed for modification
- **Liskov Substitution**
  Every subclass/derived class should be substitutable for their base/parent class
- **Interface Segregation**
  A client should never be forced to implement an interface that it doesn't use or clients shouldn't be forced to depend on methods they do not use
- **Dependency Inversion**
  Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module, but they should depend on abstractions

Estimation


I'LL NEED A PROJECT PLAN TO JUSTIFY THE RESOURCES WE NEED TO CHANGE OUR SOFTWARE.
I CAN MAKE THOSE SOFTWARE CHANGES IN TEN SECONDS.
DONE.
GOOD WORK.
NOW ALL WE NEED IS THAT PLAN.
www.dilbert.com scottadams@aol.com
© 1996 United Feature Syndicate, Inc.
EXTREME PROGRAMMING
I CAN'T GIVE YOU ALL OF THESE FEATURES IN THE FIRST VERSION.
AND EACH FEATURE NEEDS TO HAVE WHAT WE CALL A "USER STORY."
OKAY, HERE'S A STORY: YOU GIVE ME ALL OF MY FEATURES OR I'LL RUIN YOUR LIFE.
www.dilbert.com scottadams@aol.com
© 2002 United Feature Syndicate, Inc.

Estimating Units

Estimating Process: Planning Poker

Accurate Estimates

1. Estimate in terms of ideal engineering days (story points), not calendar time
2. Use velocity to determine how many story points the team can finish in an iteration
3. Use iteration slack to smooth over surprises and deliver on time every iteration
4. Use risk management to adjust for risks to the overall release plan

Code Quality Metrics

Test Coverage:

- Functional
- Statement
- Branch

Complexity:

- Cyclomatic
- Essential
- Integration
- Cyclomatic density

Quality:

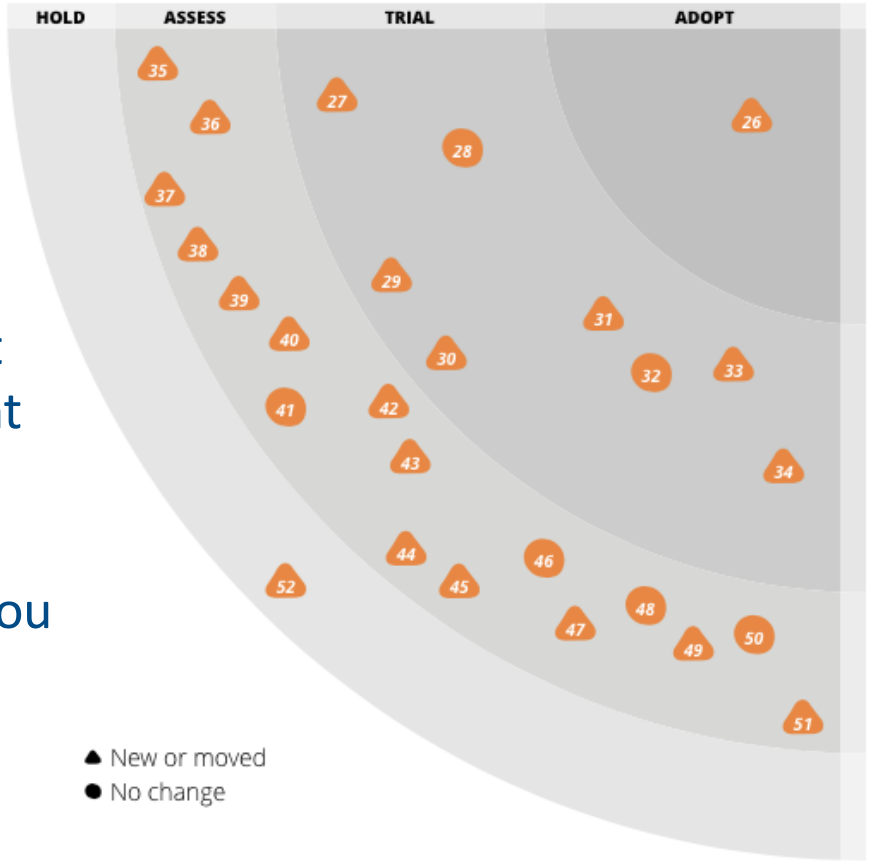
- Average Percentage of Faults Detected (APFD)
- Fault Severity
- Production Incidents
- Quality Over Release Life Cycles

Technology Choices and Stacks: Techradar

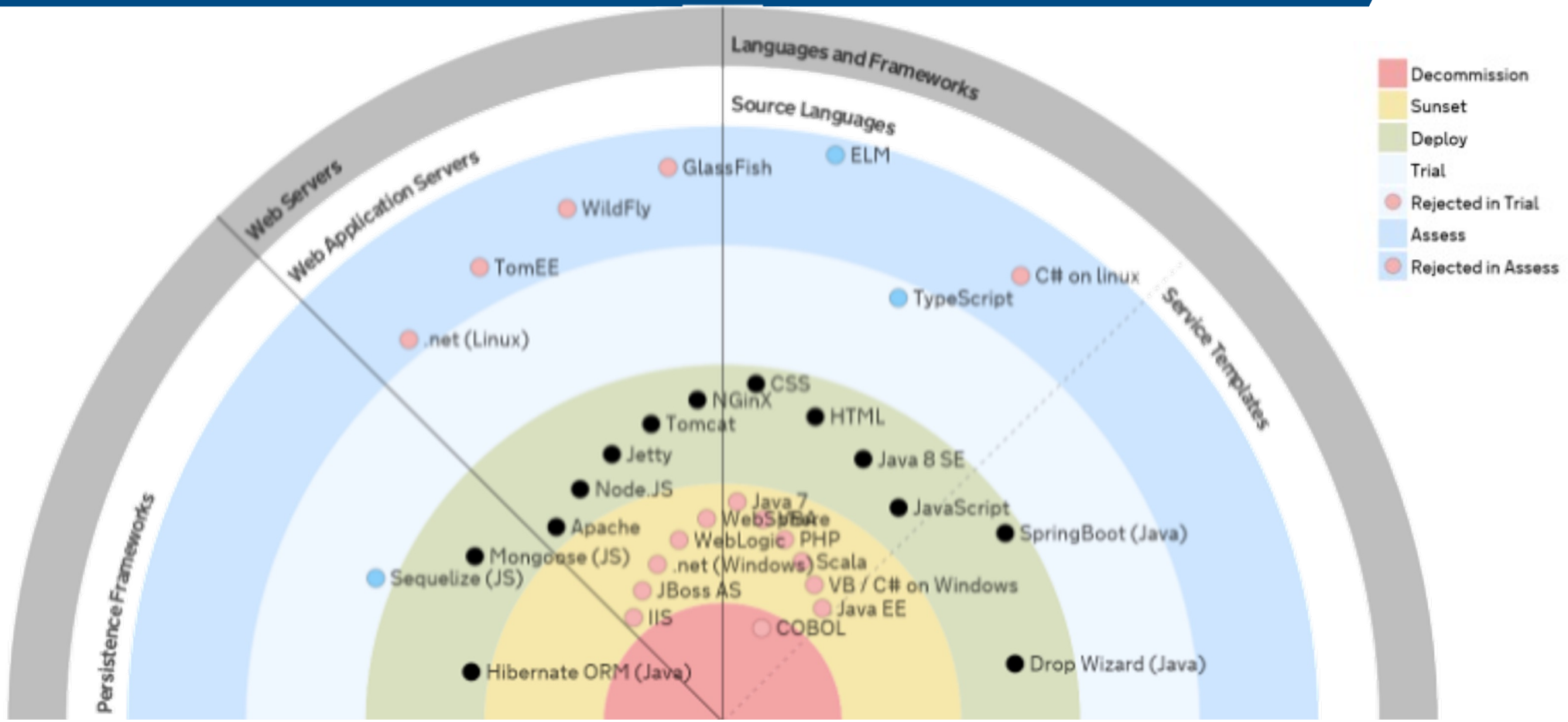
Discuss across all organisational levels
Review entire technology portfolio

This enables you to:

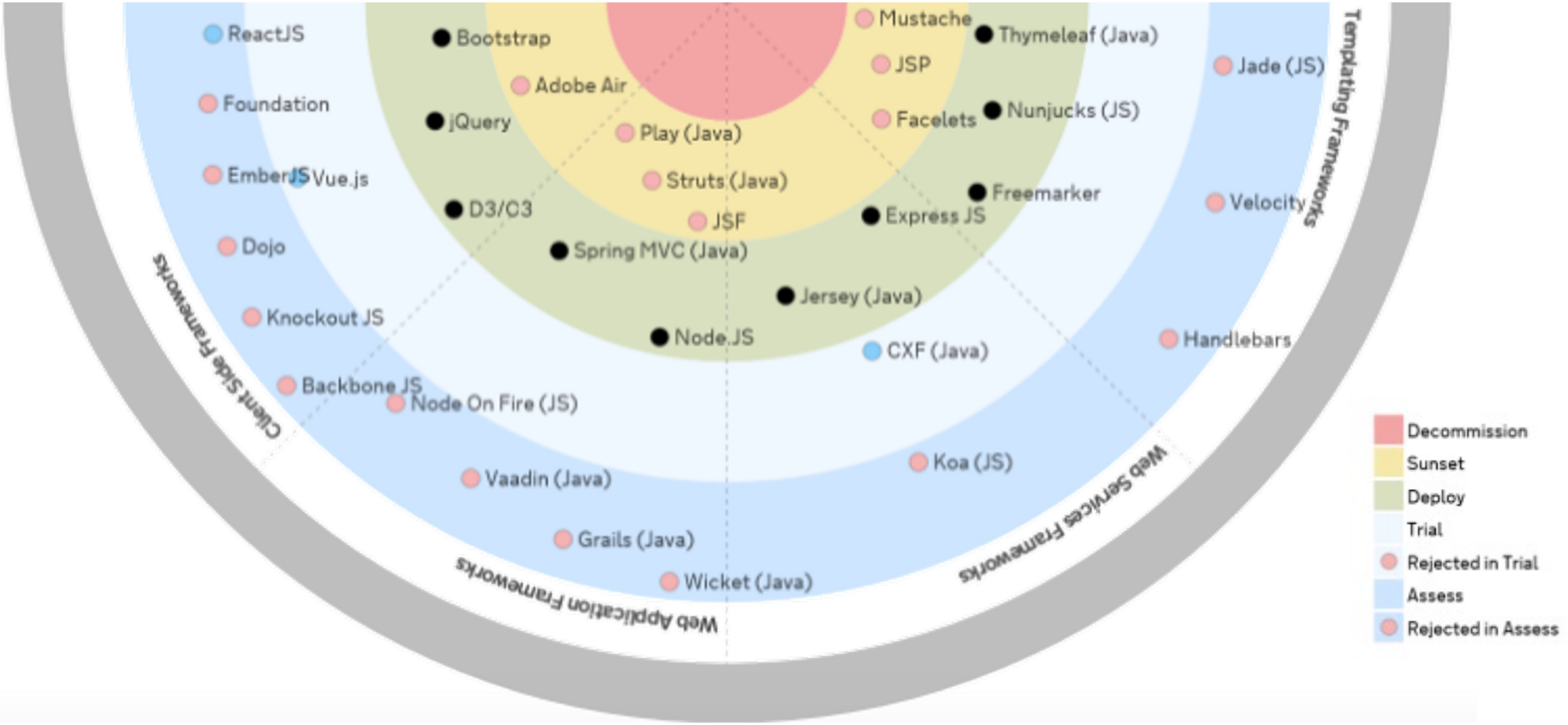
- Objectively assess what's working, and what isn't
- Pollinate innovation across teams and experiment accordingly
- Balance the risk in your technology portfolio
- Work out what kind of technology organisation you want to be
- Set a path for future success



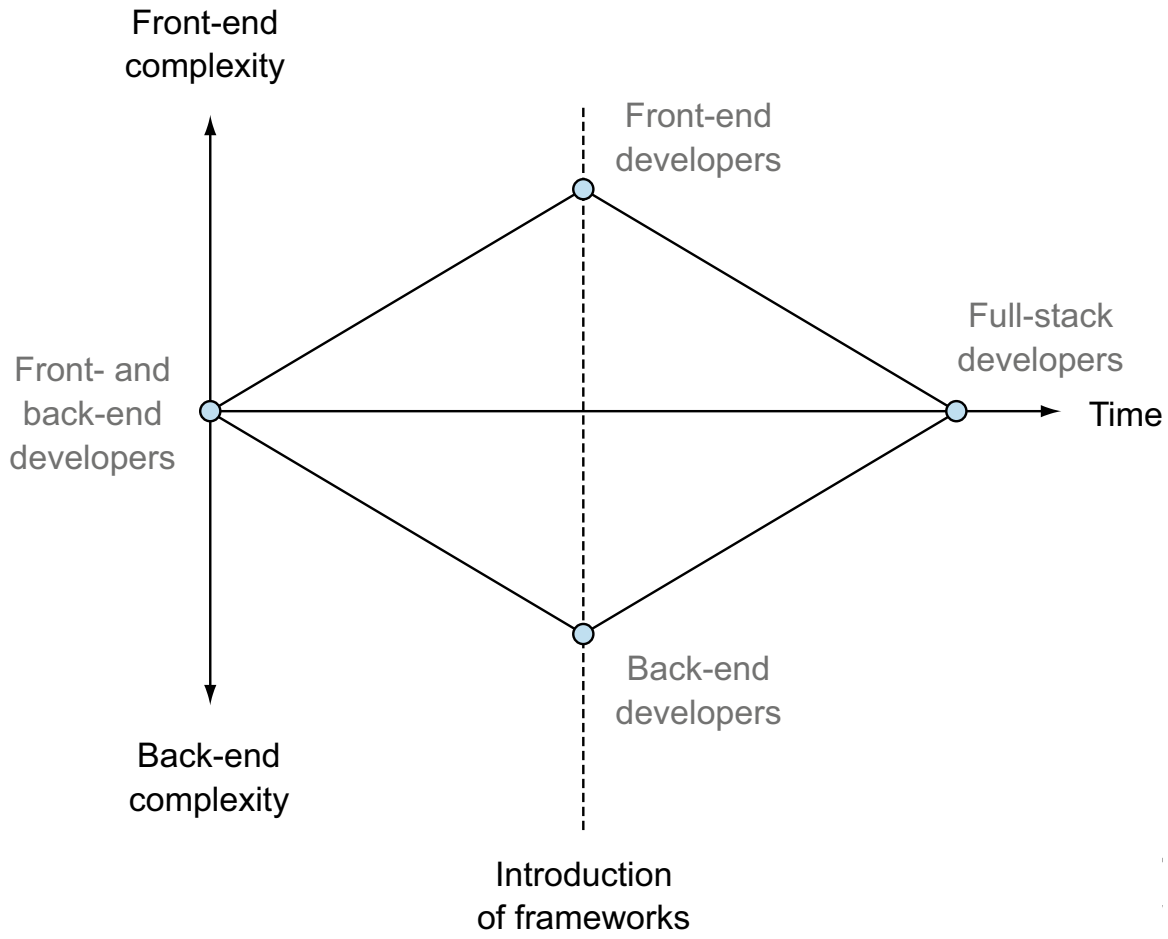
Tech Radar 1


Languages and Frameworks
Source Languages
Web Servers
Web Application Servers
Persistence Frameworks
Service Templates
ELM
GlassFish
WildFly
TomEE
.net (Linux)
C# on linux
TypeScript
CSS
NGinX
Tomcat
HTML
Jetty
Java 8 SE
Node.JS
Java 7
JavaScript
Apache
PHP
SpringBoot (Java)
WebLogic
Mongoose (JS)
.net (Windows)
Scala
Sequelize (JS)
JBoss AS
VB / C# on Windows
Java EE
IIS
COBOL
Hibernate ORM (Java)
Drop Wizard (Java)
Decommission
Sunset
Deploy
Trial
Rejected in Trial
Assess
Rejected in Assess

Tech Radar 2


ReactJS
Bootstrap
Mustache
Thymeleaf (Java)
JSP
Jade (JS)
Adobe Air
Foundation
jQuery
Play (Java)
Facelets
Nunjucks (JS)
EmberJS
Vue.js
Struts (Java)
Freemarker
Velocity
D3/C3
JSF
Express JS
Dojo
Spring MVC (Java)
Jersey (Java)
Knockout JS
Node.JS
CXF (Java)
Handlebars
Backbone JS
Node On Fire (JS)
Koa (JS)
Vaadin (Java)
Grails (Java)
Wicket (Java)
Templating Frameworks
Web Services Frameworks
Web Application Frameworks
Client Side Frameworks
Decommission
Sunset
Deploy
Trial
Rejected in Trial
Assess
Rejected in Assess

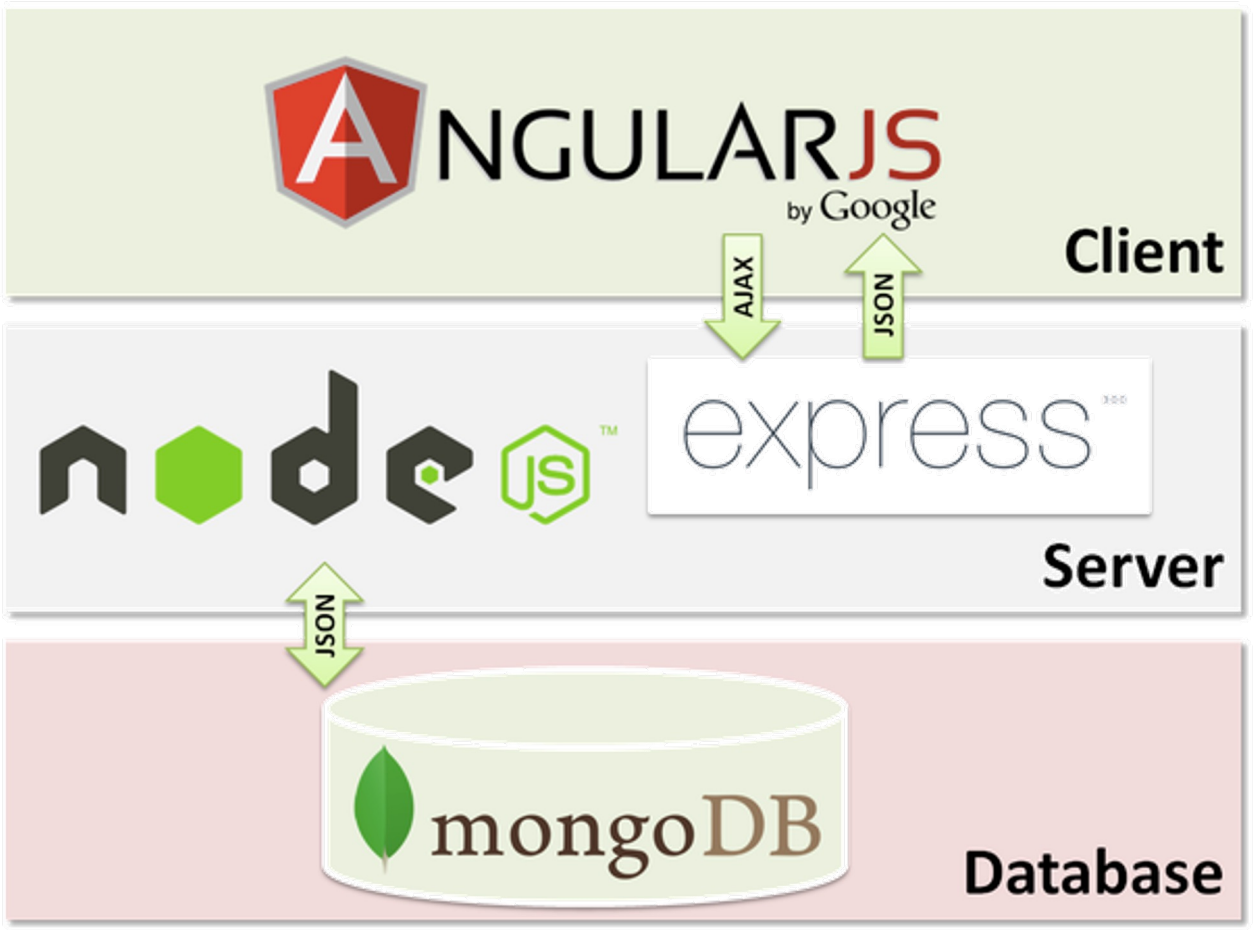
Full Stack Development



Full Stack Tools

- **MEAN** (MongoDB, Express, AngularJS, Node.js)
- MongoDB - Database
- Express – Web Application Framework
- AngularJS – UI/UX Framework
- Node.js – Application Platform
- **LAMP** (Linux, Apache, MySQL, PHP)
- Replace Node with Go
- HTTP and REST

MEAN Stack



Benefits:

- Based on JavaScript
- Code understood by all developers
- Single team, whole solution
- Aligns with vertical story slicing

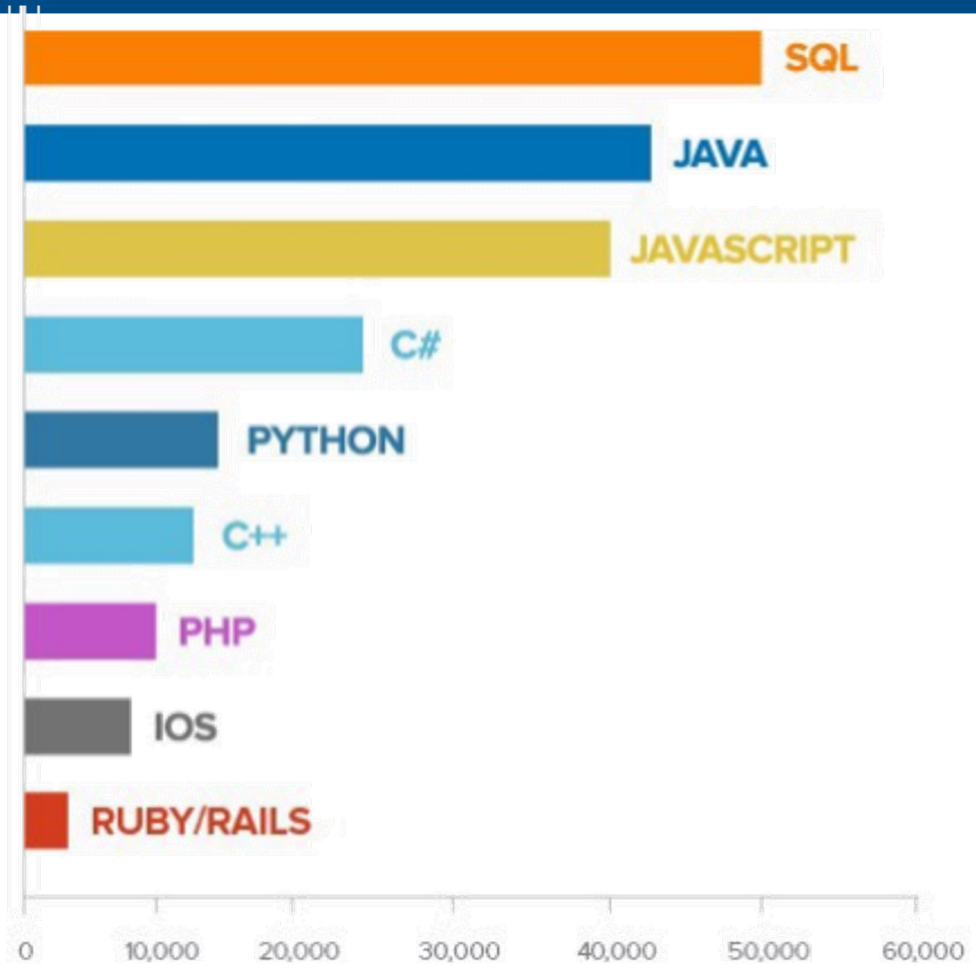
Note:

- AngularJS not used
  - JQuery
  - VueJS in Trial
  - React in Assess

Why Use Full-Stack Development?

- Flexible team members
- Visibility of all layers
- Fewer developers required
- Easy skills upgrade
- Faster delivery
- Faster problem resolution
- Lower technical debt
- Tied to stack technology

Programming Languages



Programming Languages

- SQL – Access (query) and manipulate databases
- Java – Multi-platform back end coding
- JavaScript – Scripting to manipulate pre-built code – Full Stack
- C# – Microsoft .NET platform otherwise Java-like
- Python – Widely used in Data Science
- C++ – Powerful (dangerous?) platform specific
- PHP – Web scripting
- IOS – for Apple apps (language is actually Objective-C)
- Ruby on Rails – Web development framework a bit like node.js

Open Source

Using Open Source:

- solving common problems with readily available open source technology
- more time and resource for customised solutions to solve the rare or unique problems
- lower implementation and running costs

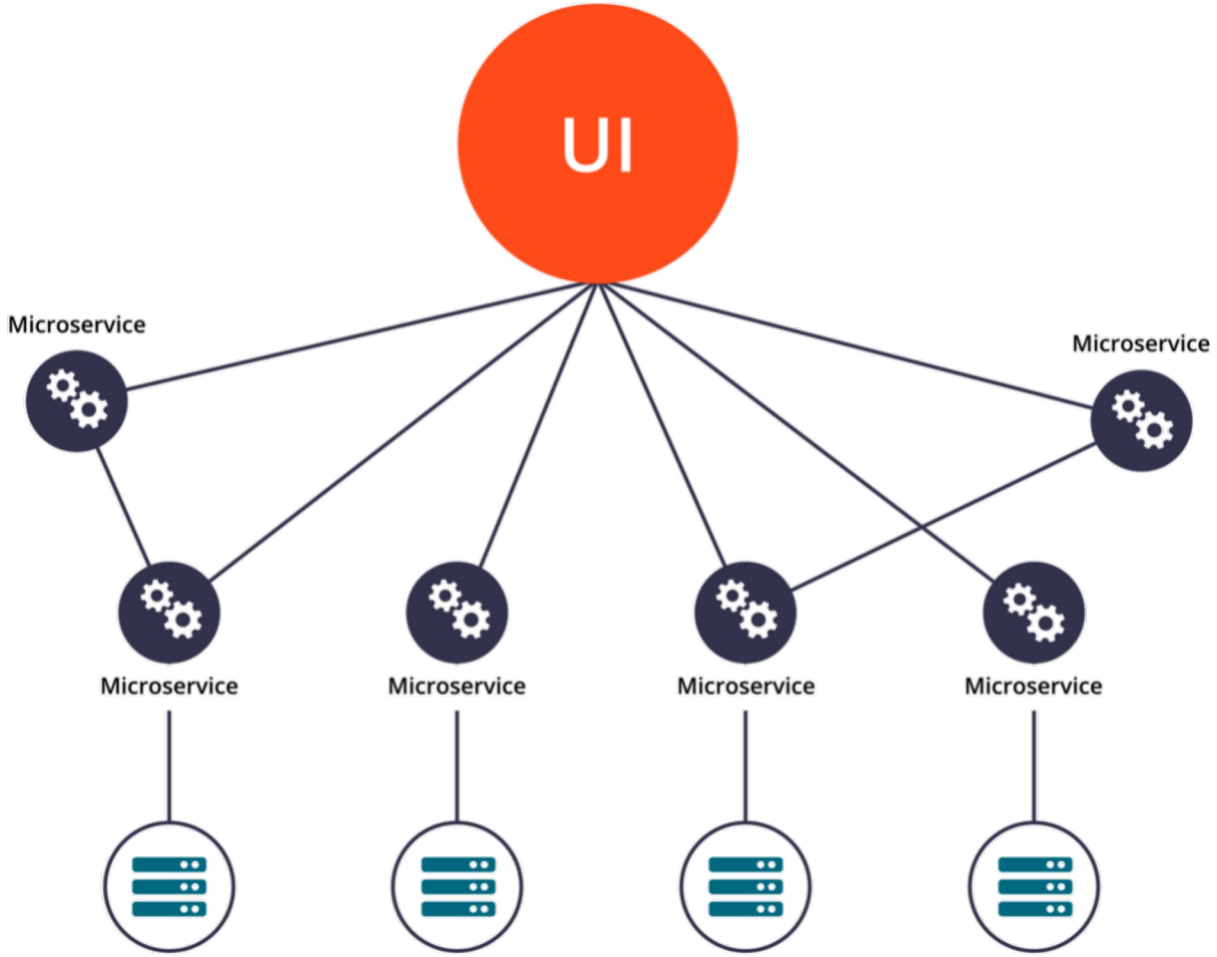
Publishing Your Source Code Encourages:

- clearer documentation - easier to maintain the code
- cleaner and well-structured
- clarity around data that needs to remain protected and how that's achieved
- suggestions from others about how the code can be improved or where security can be improved

Why not Publish Under OGL

- Embargoed Government Policy
- Algorithm to Avoid Fraud
- Keys and Credentials should not be in code

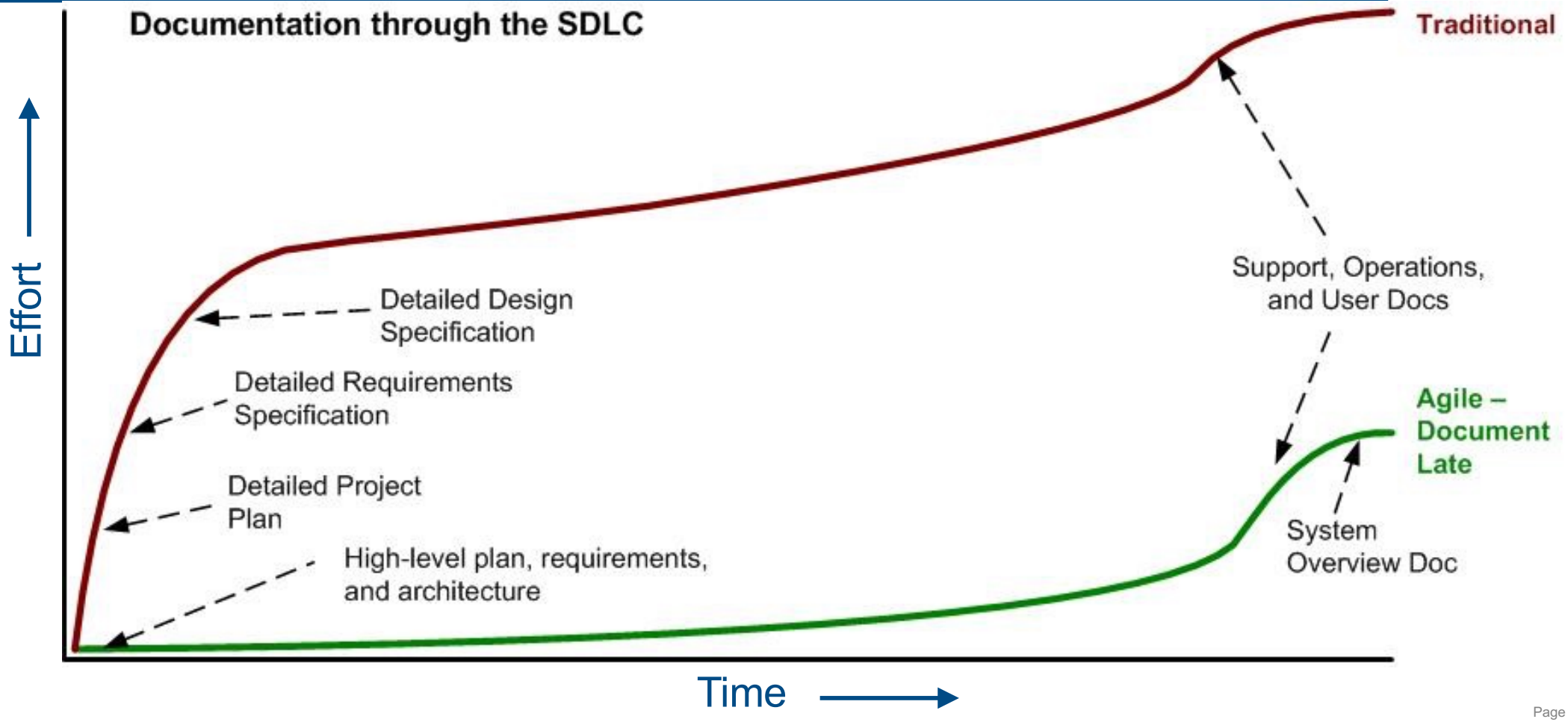
Microservices



Why Use Microservices?

- Independent deployment and maintenance
- Fault isolation
- Different languages and technologies
- Fits in with CI/CD
- Single Responsibility Principle - maps to business capability
- Scalability and reusability
- Good with containers, such as Docker.
- Good with cloud
- Simplified security monitoring
- Parallel development/delivery

Documentation Standards

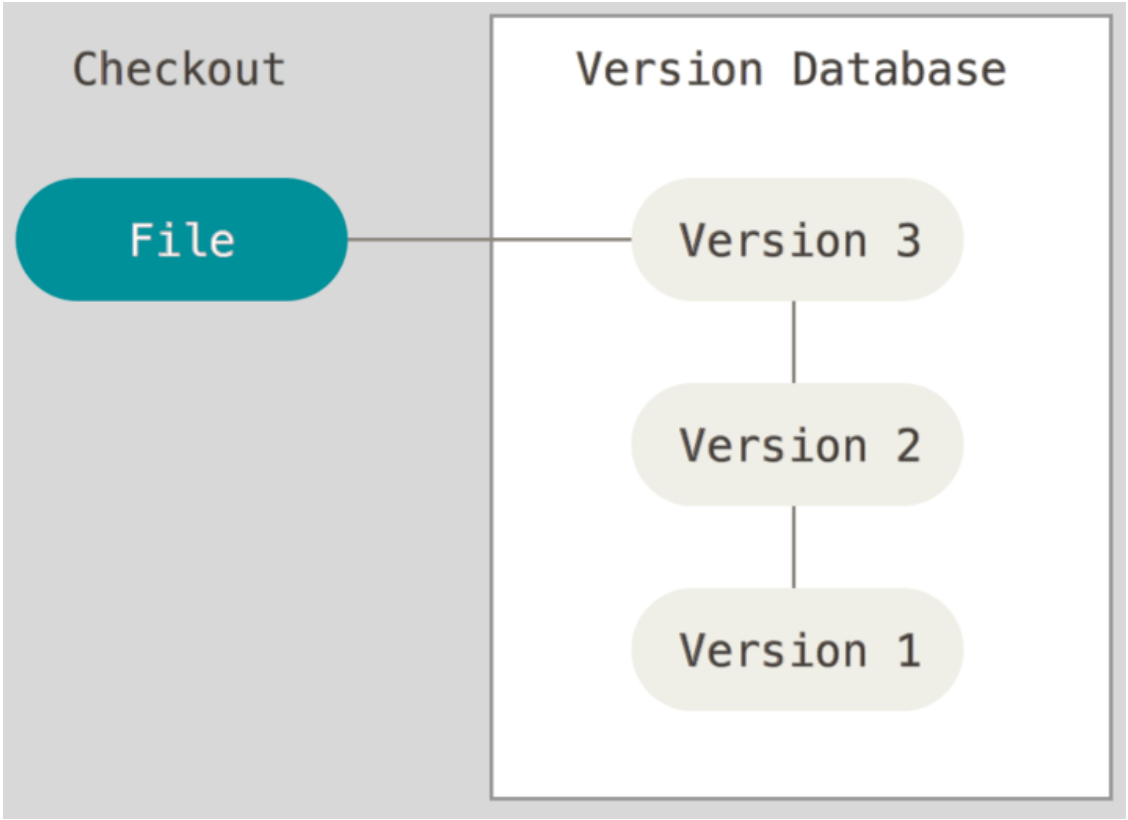


Service Delivery Documentation

- Service Epic
- Component Epic
- Vision
- Objectives & Outcomes (including KPIs)
- Business Case
- Business Model Canvas
- Architectural Impacts
- Service Stakeholders
- Value Proposition

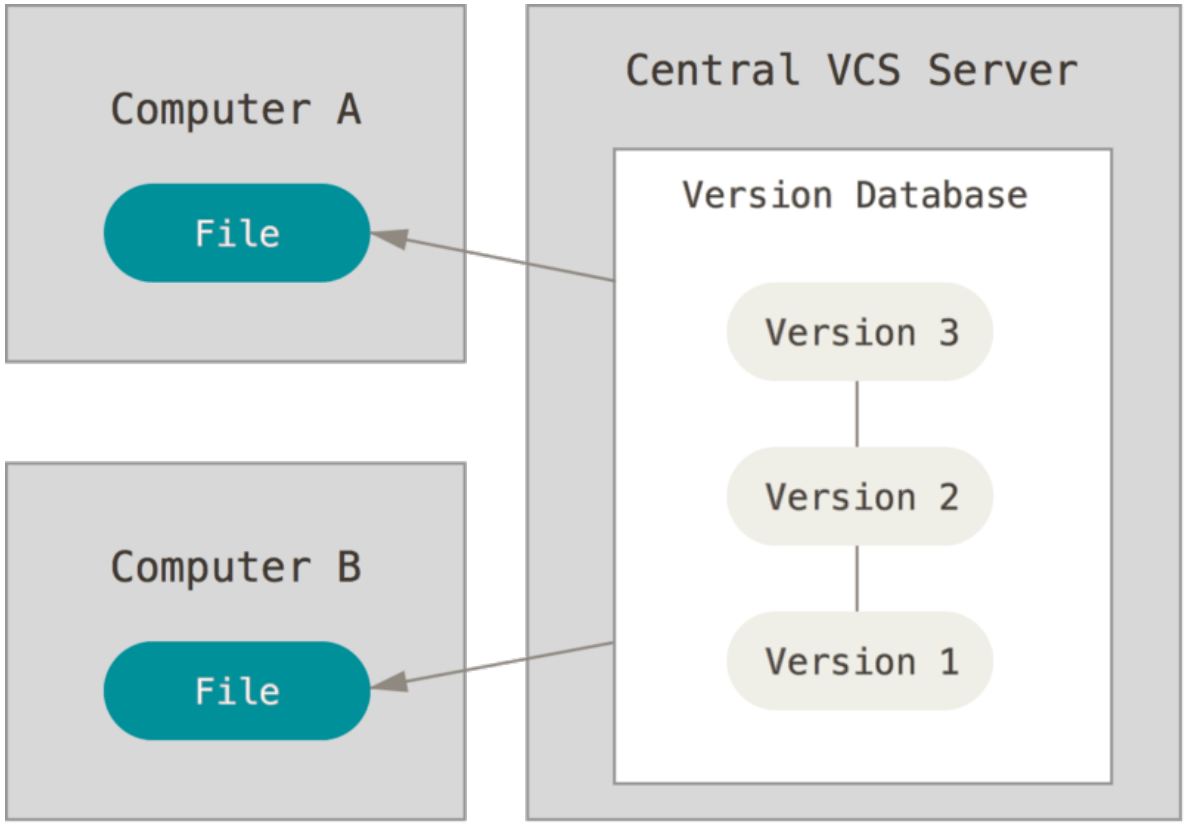
Code Management: Local Version Control

Local Version Control – e.g. RCS



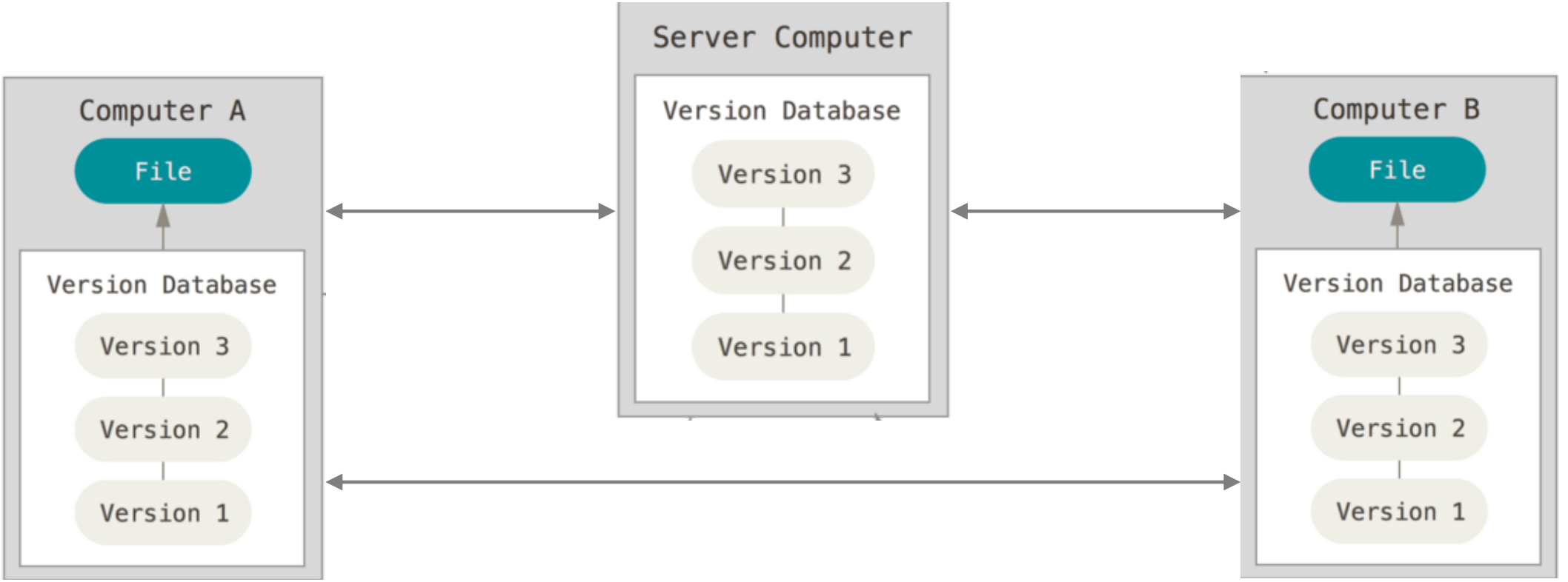
Code Management: Central Version Control

Central Version Control – e.g. CVS, Subversion and Perforce

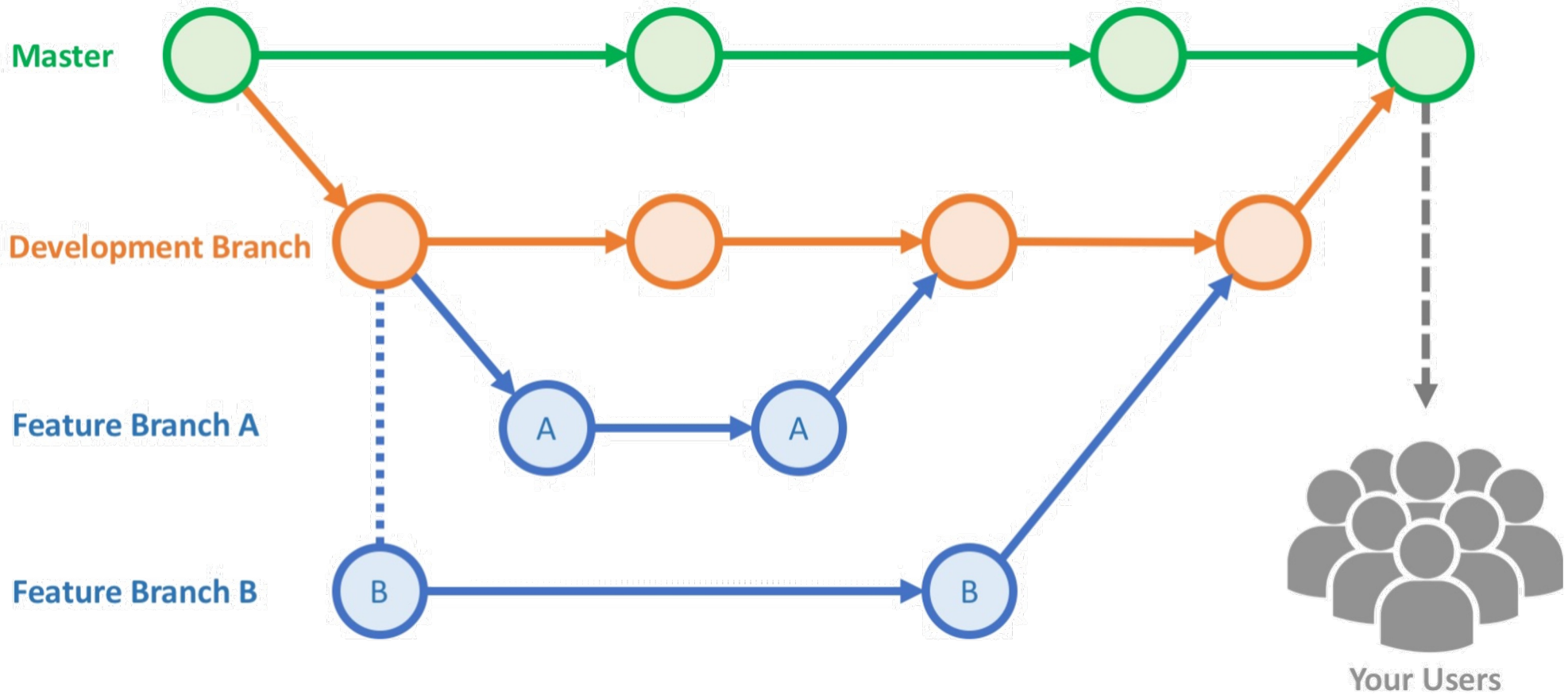


Code Management: Distributed Version Control

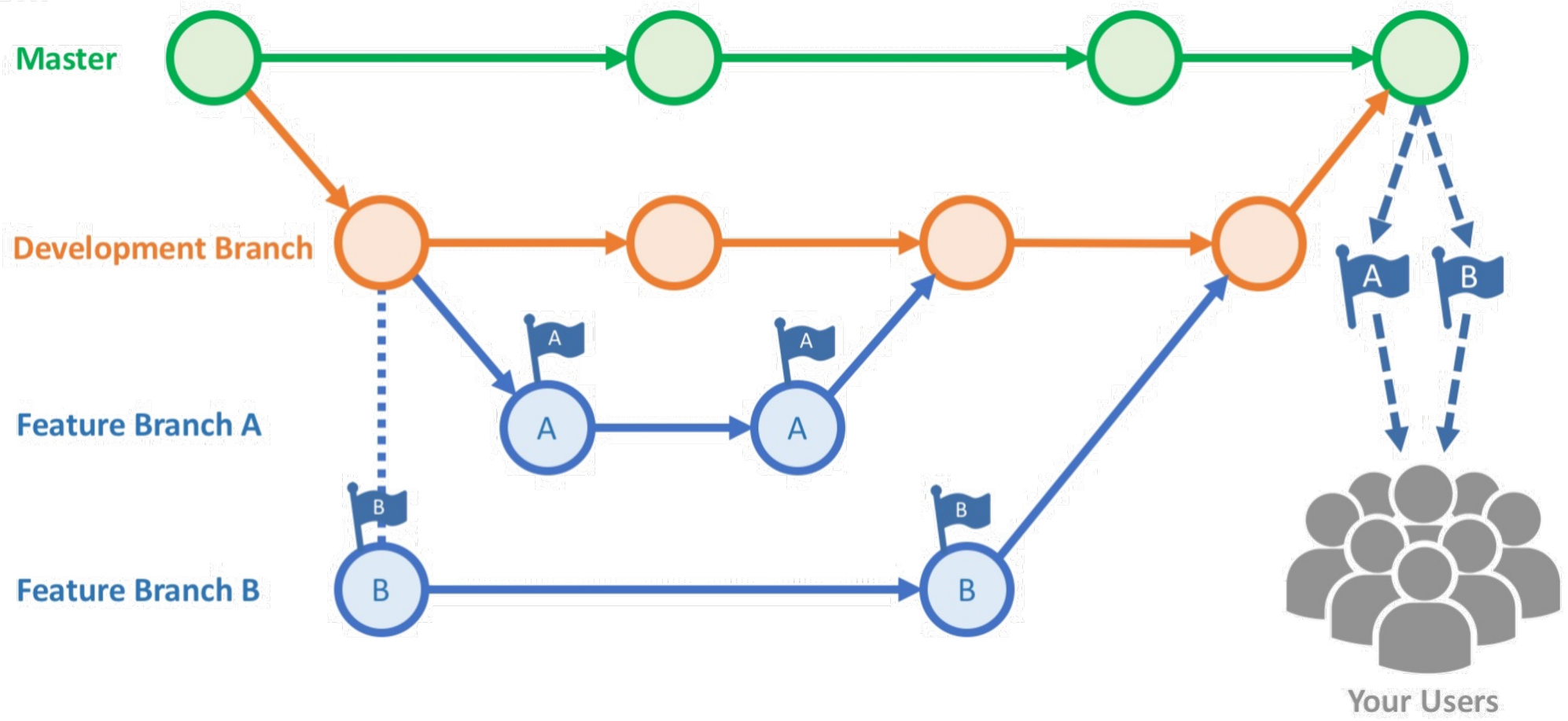
Distributed Version Control – GIT, Mercurial, Bazaar and Darcs



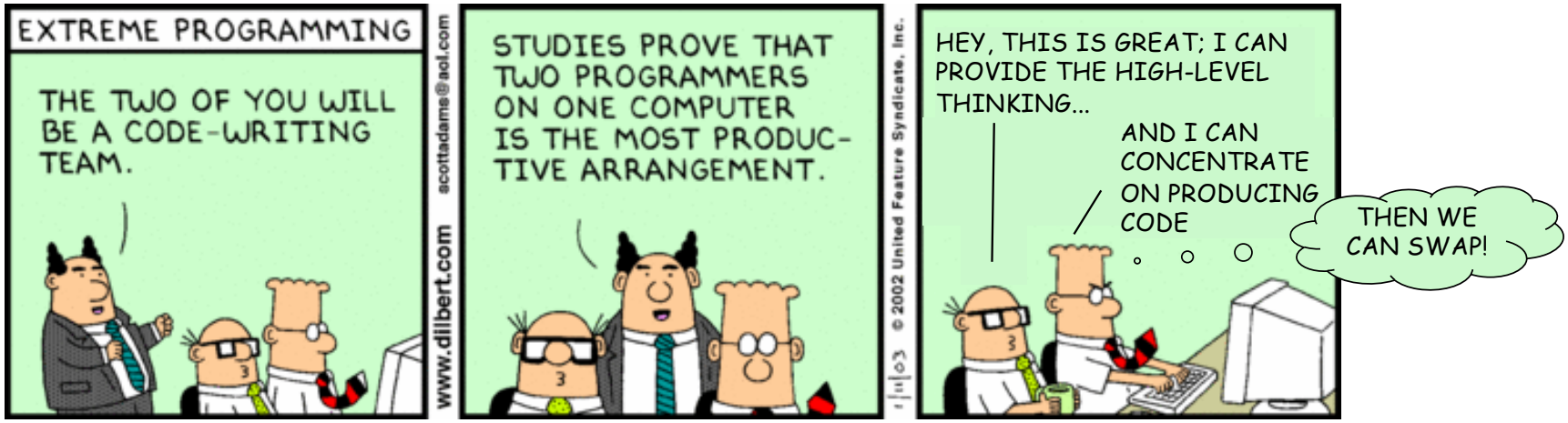
Feature Branching Without Flags/Toggles



Feature Branching Using Flags/Toggles

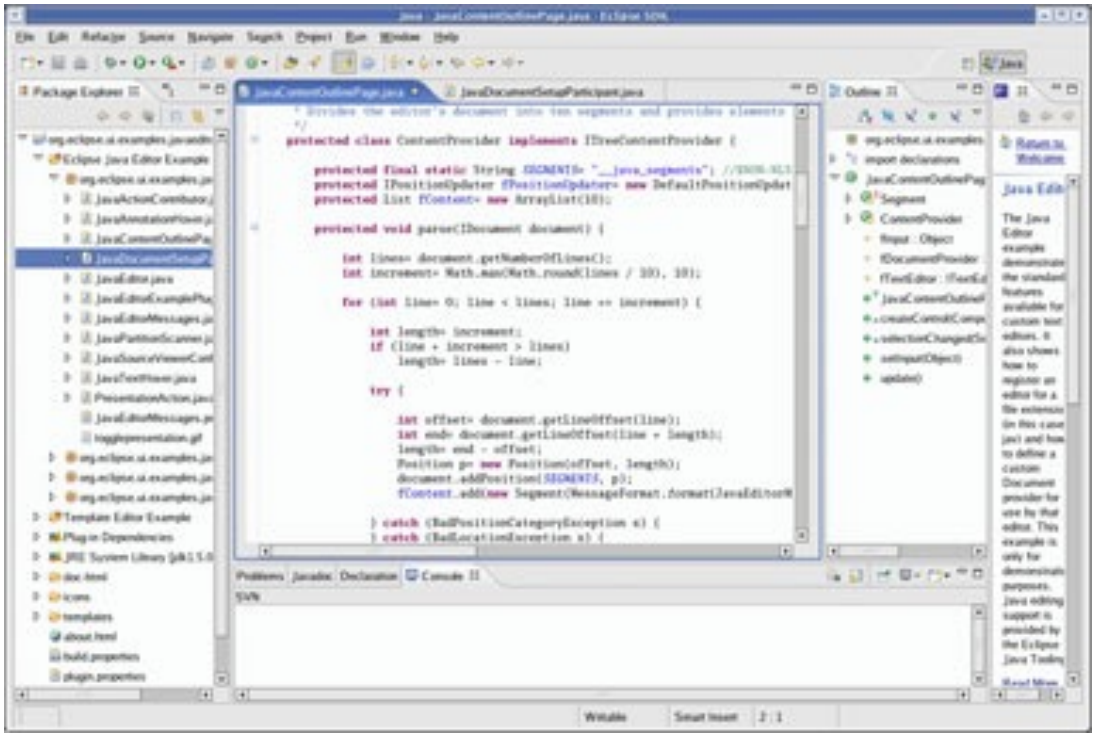


Agile Coding Practices – Pair Programming



- Pass the keyboard regularly
- Allow the driver to finish a task before commenting or questionning
- Don’t use your phone or do anything unrelated to the task
- Don’t touch the screen

Development Environments

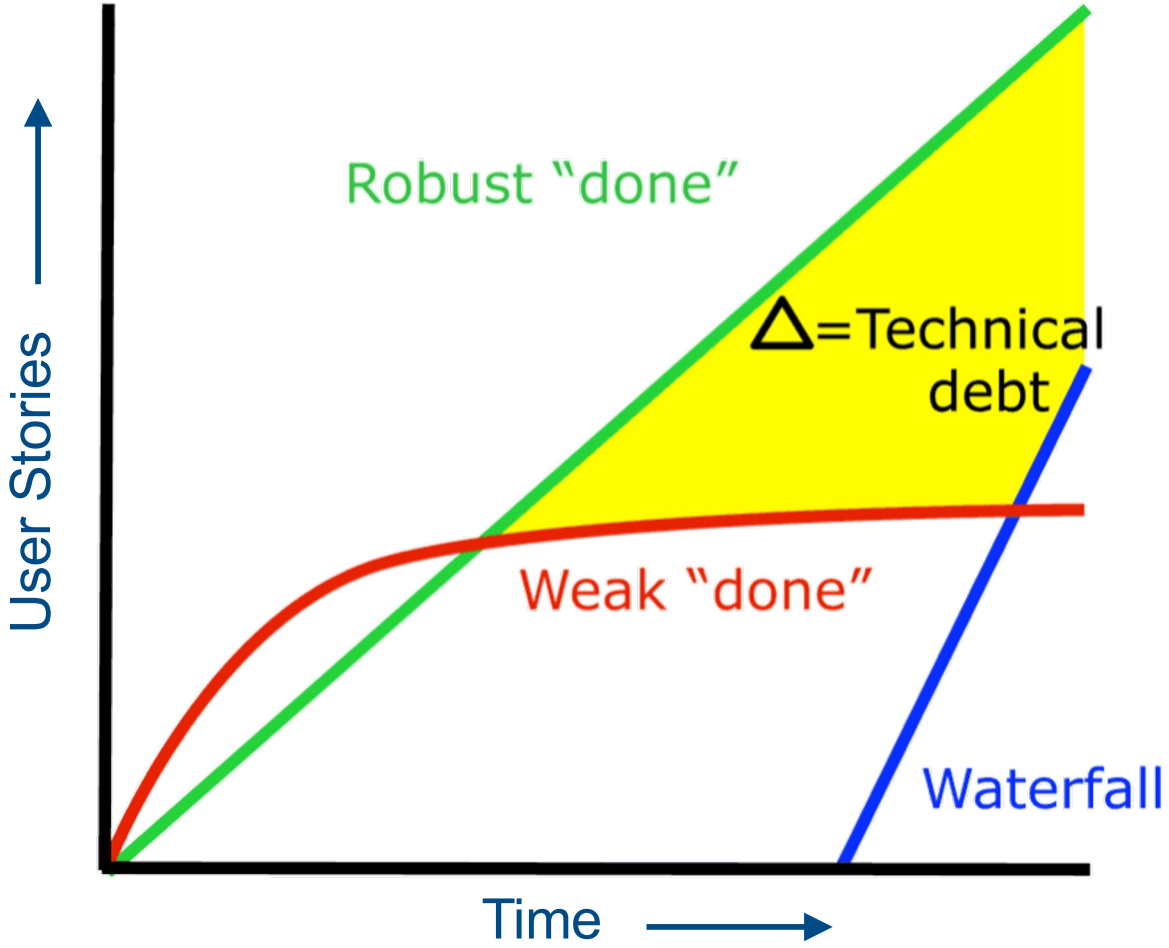


Integrated Development Environment (IDE)

Components

- Code editor:
  - layout
  - colour code
  - type-ahead
  - auto code
  - refactor
- Compiler:
  - Reports errors
  - shows running code
- Debugger:
  - Steps through code
- Build automation tools:
  - Package
  - Deploy

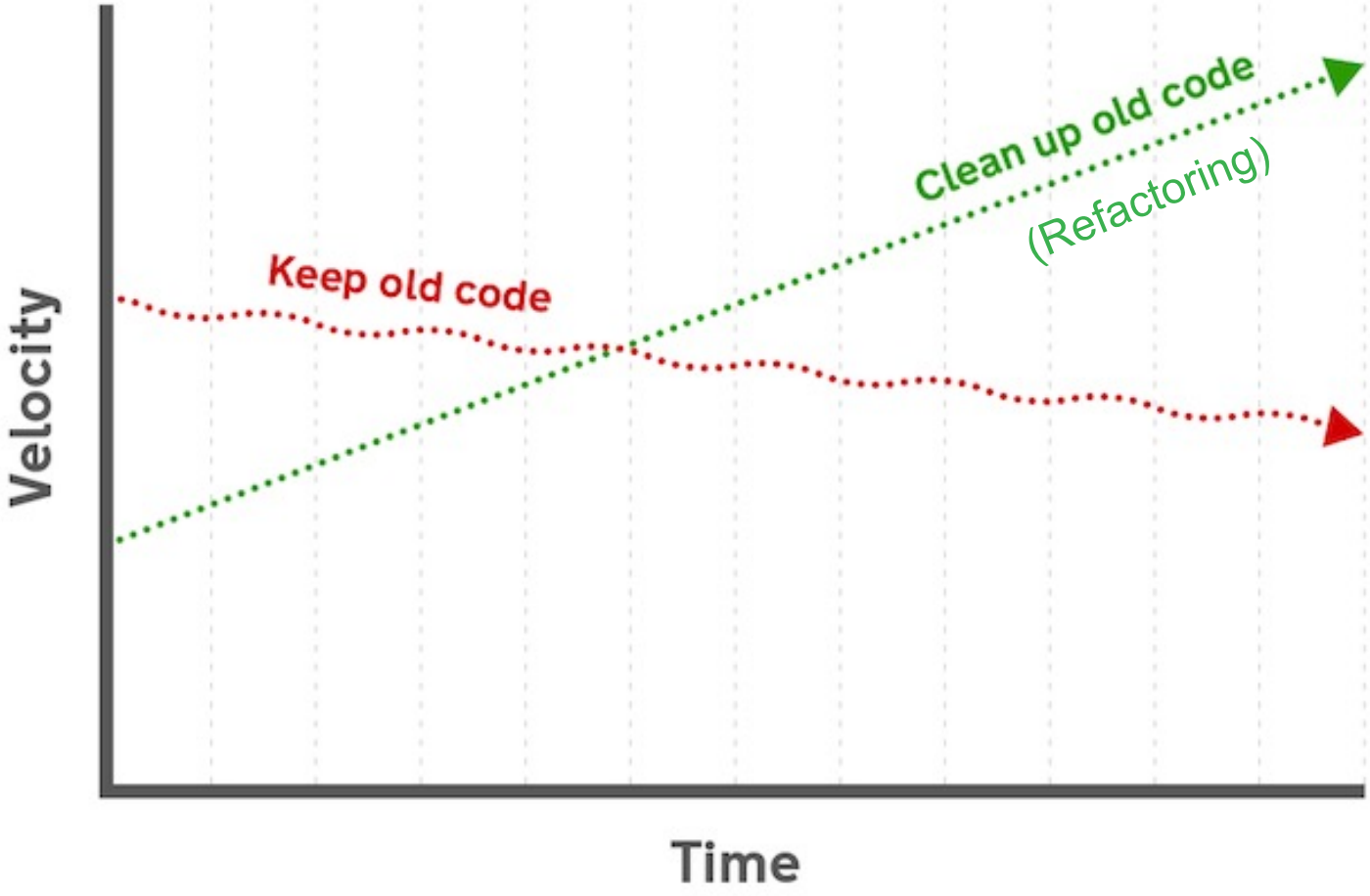
Technical Debt



Categories of debt:

- Reliability
- Security
- Performance Efficiency
- Maintainability

Paying Down Technical Debt



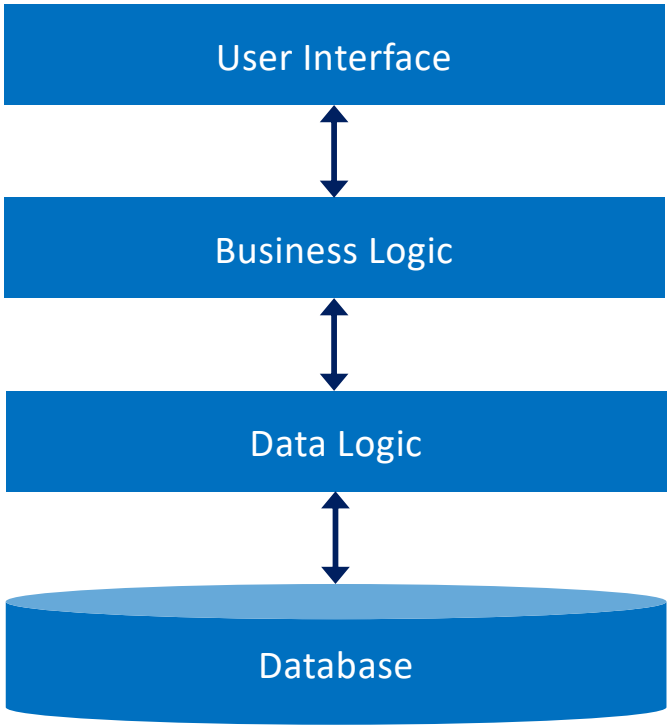
Technical Debt affects:

- Reliability
- Security
- Performance Efficiency
- Maintainability

Why is Big Data an Issue?

- **Volume**
  - 40 Zettabytes (Trillion GB) by 2020
  - 7+ Billion Phones
  - IoT/IoE exponential growth
- **Velocity**
  - 100+ Million Events/second as standard
  - Would have been a DDOS attack 3 years ago
  - More events being captured each day
- **Variety**
  - Data Domains – healthcare, finance
  - Type – text, image, video
- **Veracity** (or Value?)
  - Level of uncertainty
  - Data-driven operations
  - Trust underpins business and government

Relational Database Storage



Attributes

Branch

Relation

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Cardinality

Degree

Primary key

Foreign key

Staff

Relation

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

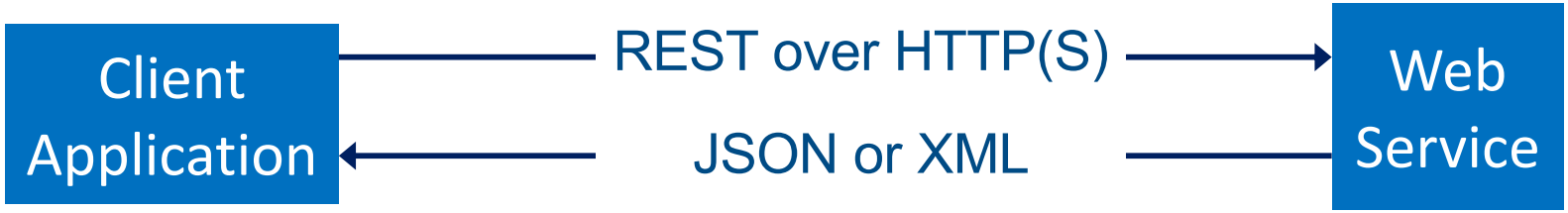
RDBMS Functions

- Storage
- Retrieval
- Integrity
- Security
- Management

SQL – Structured Query Language

- Set up (CREATE)
- Modify (ALTER)
- Manipulate (INSERT, UPDATE, DELETE)
- Query (SELECT)
- Security (GRANT, REVOKE)
- Transactions (START TRANS, COMMIT, ROLLBACK)

Data Transmission

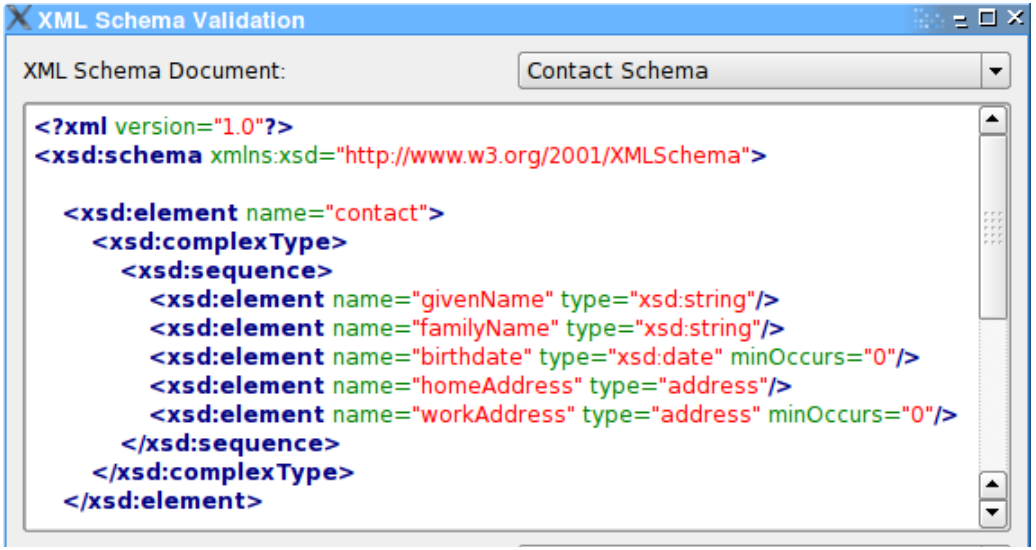



```
<empinfo>
   <employees>
      <employee>
         <name>James Kirk</name>
         <age>40></age>
      </employee>
      <employee>
         <name>Jean-Luc Picard</name>
         <age>45</age>
      </employee>
      <employee>
         <name>Wesley Crusher</name>
         <age>27</age>
      </employee>
   </employees>
</empinfo>
```

```
{  "empinfo" :
      {
            "employees" :  [
            {
                  "name" : "James Kirk",
                  "age" : 40,
            },
            {
                  "name" : "Jean-Luc Picard",
                  "age" : 45,
            },
            {
                  "name" : "Wesley Crusher",
                  "age" : 27,
            }
                        ]
      }
}
```

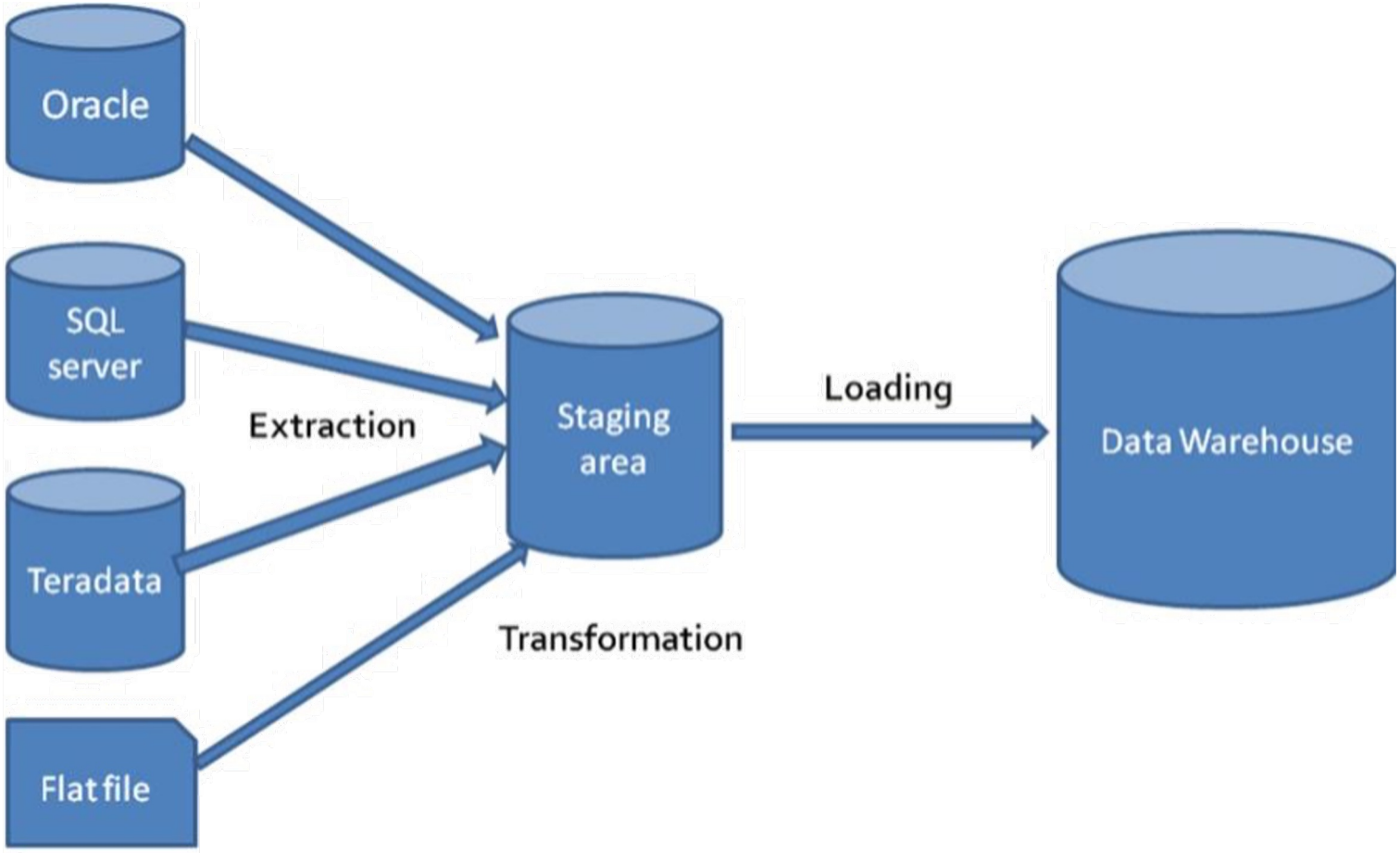
XML Instance Defined by Schema

Schema (*.xsd)



Instance (*.xml)

ETL – Extract, Transform, Load



Benefits of NoSQL (e.g. MongoDB)

Data models:

- No predefined schema required – can be added later
- Much easier to update as data and requirements emerge

Data structure:

- Ok with unstructured data (e.g., texts, social media posts, video, email)

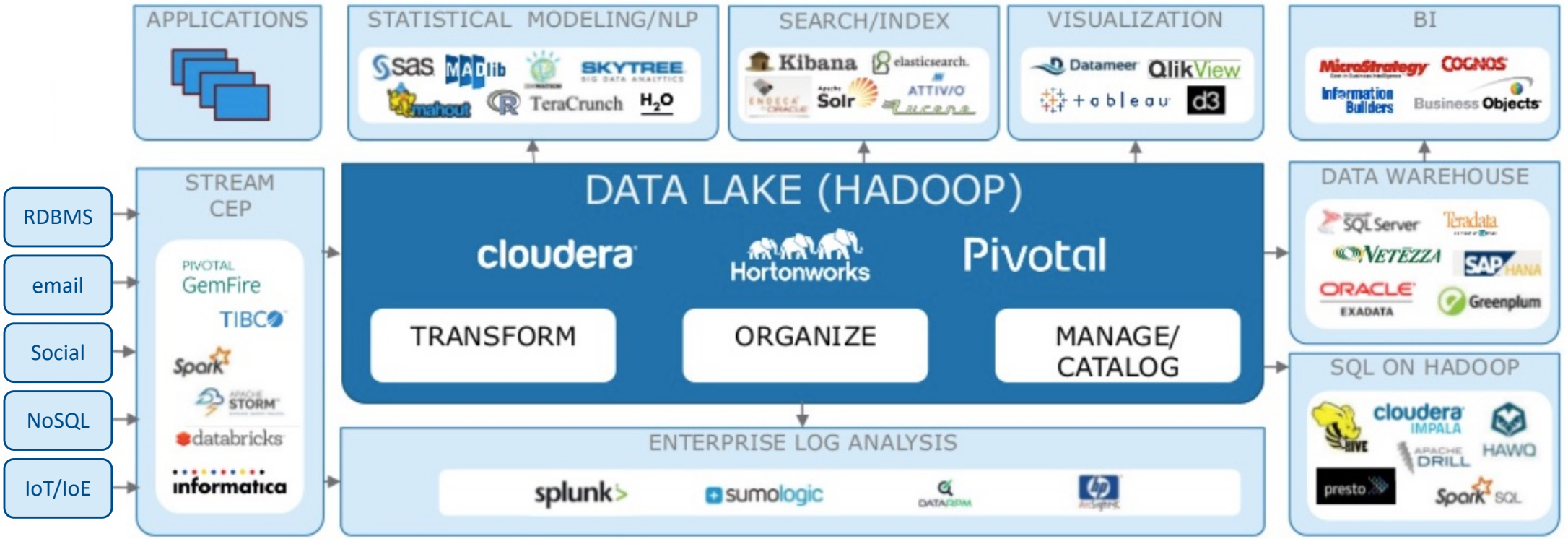
Scaling:

- Cheaper to scale

Development model:

- Open source
- No licence fee

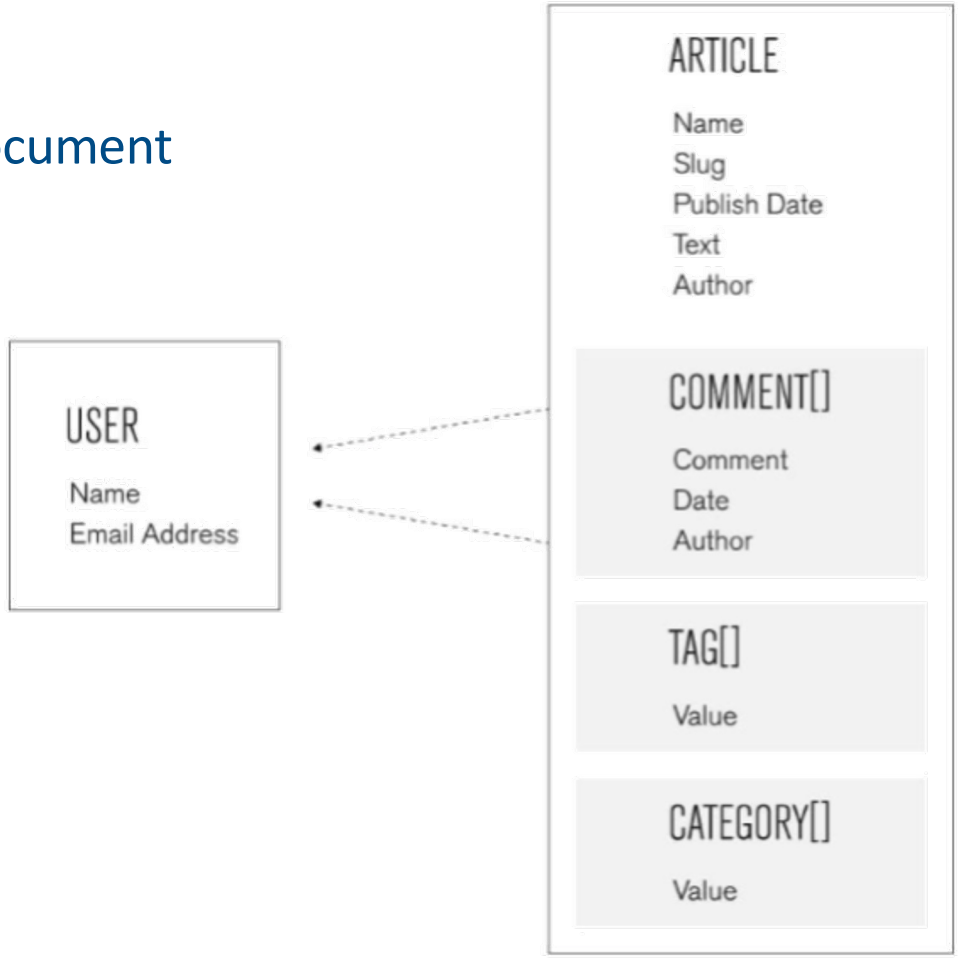
Data Lake


APPLICATIONS
STATISTICAL MODELING/NLP
SEARCH/INDEX
VISUALIZATION
BI
DATA LAKE (HADOOP)
cloudera
Hortonworks
Pivotal
TRANSFORM
ORGANIZE
MANAGE/ CATALOG
STREAM CEP
RDBMS
email
Social
NoSQL
IoT/IoE
DATA WAREHOUSE
SQL ON HADOOP
ENTERPRISE LOG ANALYSIS

NoSQL - Document

e.g. MongoDB:

- All data for a given record in a single document
- Stored as BSON (Binary JSON)
- Design changes dynamically
- Structure (and governance) evolves



REVIEW– Coding Fundamentals and Tools

- Communities of Practice
- Coding Standards + Best Practice
- Estimation
- Code Quality
- Technology Choices and Stacks
- Microservices
- Documentation Standards
- Standards to Support Service Delivery
- Code Management with GIT
- Agile Coding Practices
- Development Environments
- Managing Technical Debt
- Data Storage and Big Data

Further Reading – Coding Fundamentals and Tools

dwpdigital.blog.gov.uk/2016/09/30/building-a-software-engineering-community/

www.scaledagileframework.com/communities-of-practice/

sourcemaking.com/refactoring/smells

sourcemaking.com/design_patterns

refactoring.guru/design-patterns

sourcemaking.com/antipatterns/software-development-antipatterns

scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

airbrake.io/blog/metrics/code-quality-metrics-management

martinfowler.com/bliki/TechnicalDebt.html

www.omg.org/spec/ATDM/1.0/Beta2/PDF

Further Reading – Coding Fundamentals and Tools

refactoring.guru/refactoring

www.mongodb.com/collateral/mongodb-architecture-guide

git-scm.com/book/en/v2/Getting-Started-About-Version-Control

git-scm.com/book/en/v2/Git-Branching-Branches-in-a-Nutshell

www.agilealliance.org/glossary/pairing

www.thoughtworks.com/insights/blog/effective-navigation-in-pair-programming

www.extremeprogramming.org

blueprint.itsbeta.net/radar.html#popup

https://www.weblineindia.com/blog/what-is-full-stack-development-how-it-benefits-in-the-virtual-world/

Further Reading – Coding Fundamentals and Tools

Videos:

www.youtube.com/watch?v=pqeJFYwnkjE

www.youtube.com/watch?v=y4yg7aT4NgM

www.youtube.com/watch?v=v-MFXnF1vXE

www.youtube.com/watch?v=sPhpf_Ll98g

vimeo.com/221024846

SOFTWARE TESTING

AGENDA – Software Testing

- UI Testing
- Accessibility testing
- Test-Driven Development
- Unit Test Frameworks
- Mocking Frameworks
- Behaviour-Driven Development
- Test Automation
- Integration Testing
- Security testing – prior to ITHC
- Release Testing – This needs clarification
- Performance Testing

Course Approach – Software Testing

- Knowledge transfer about Software Testing tools and techniques
- Significance to Delivery Managers
- Scenarios
- Questions
- Conversations

UI Testing

- Manual/Prototype
- Automated:
  - Functional:
    - Selenium
    - Gherkin
    - Cucumber
  - Performance:
    - JMeter
    - Gatlin
    - Artillery
    - Autocannon

Accessibility Testing

Web Content Accessibility Guidelines (WCAG 2.0) – Level AA required

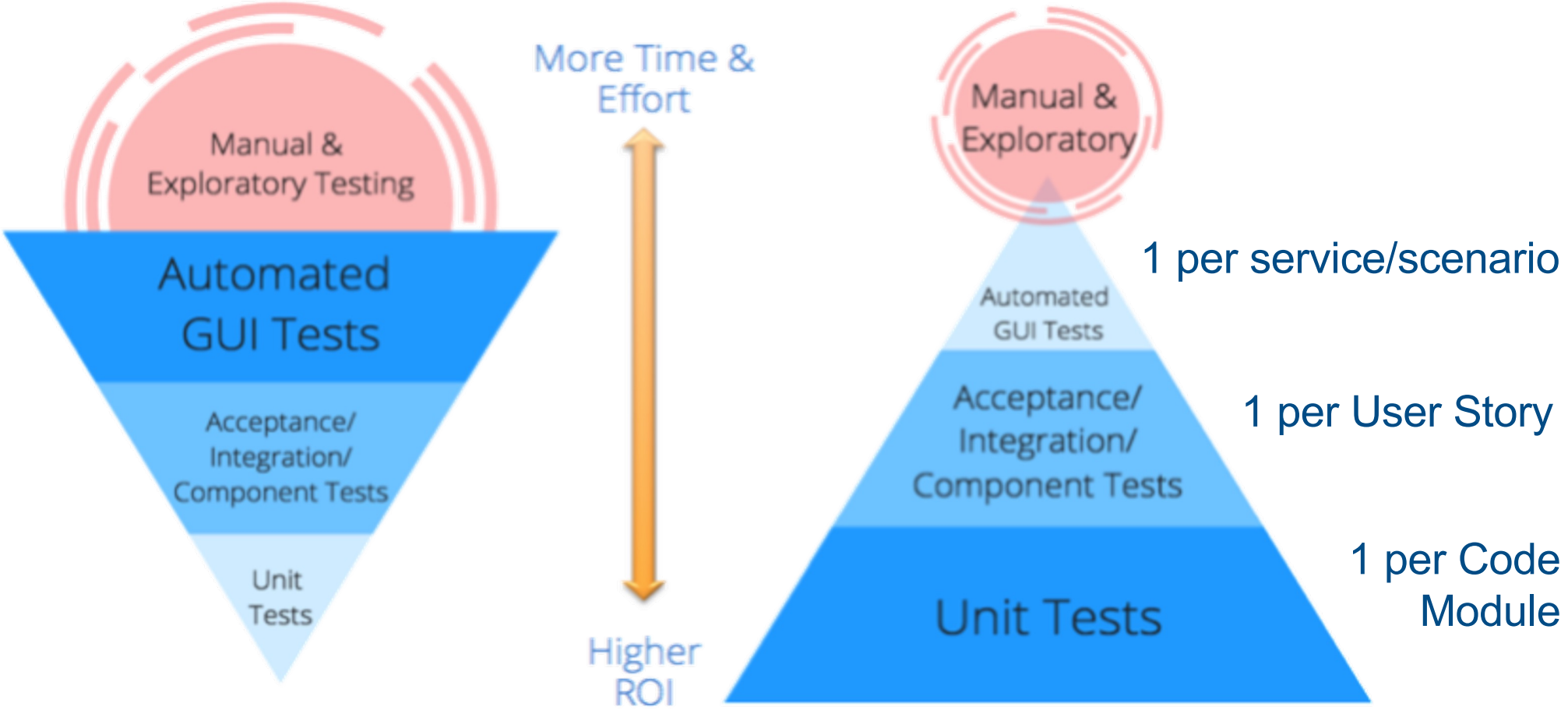
How digital services can be accessible to everyone, including users with impairments:

- sight - like blind, partially sighted or colour blind people
- hearing - like people who are deaf or hard of hearing
- mobility - like those who find it difficult to use a mouse or keyboard
- thinking and understanding - like people with dyslexia, autism or learning difficulties

Work on assistive technologies:

- JAWS, ZoomText, Dragon NaturallySpeaking, NVDA, VoiceOver

Cone vs. Pyramid



Test Tools

UI/GUI

- Selenium
- Sahir, Wati, Abbot, Frankenstein

Acceptance

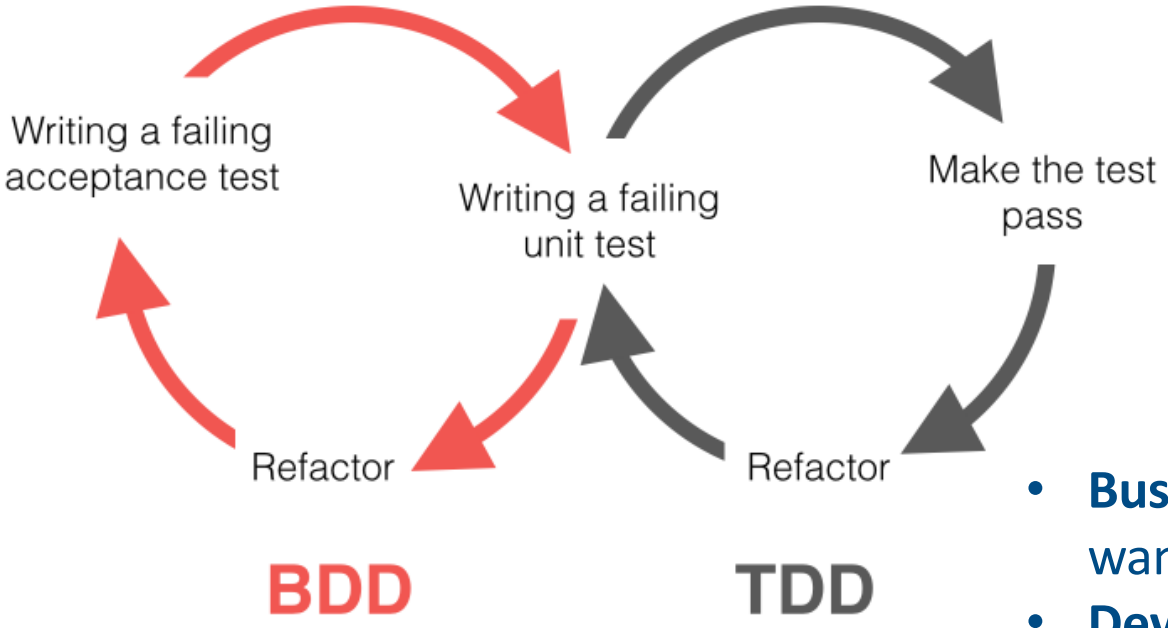
- Cucumber
- FitNesse
- RSpec, Jbehave

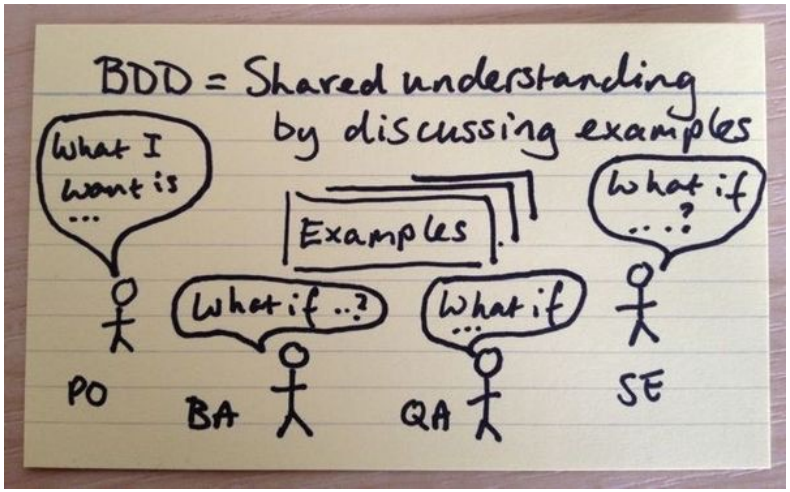
Unit

- xUnit
- TestNG

Test Driven Development

Behaviour Driven Development (BDD)





- **Business person** specifies behaviours they want
- **Developer** asks questions and notes additional behaviours
- **Tester** bases acceptance tests on behaviours

Behaviour Driven Development
Cucumber and Gherkin

```
Feature: Refund item

  Scenario: Jeff returns a faulty microwave
    Given Jeff has bought a microwave for £100
    And he has a receipt
    When he returns the microwave
    Then Jeff should be refunded £100
```

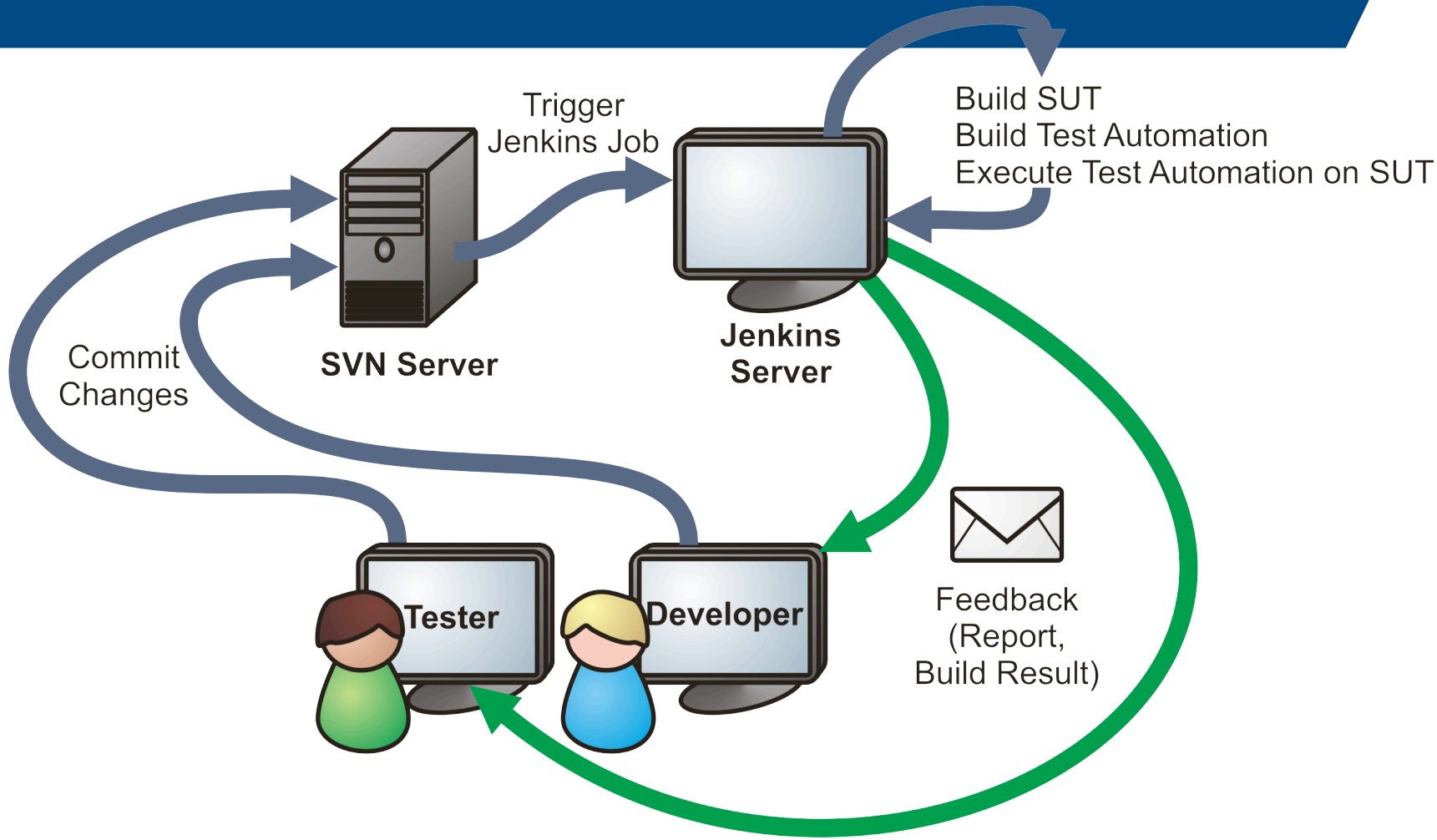
Unit Testing - Definitions

- Unit: Smallest piece of code
  - Class/Component
  - Method/Operation
- Test: Code that executes the unit of code being tested
- Test Runner: System that runs tests and records results
- Assertion: Statement that must be true for the test to pass
- System Under Test (SUT): Code being tested
- Depended On Collaborator (DOC): Code called from the SUT

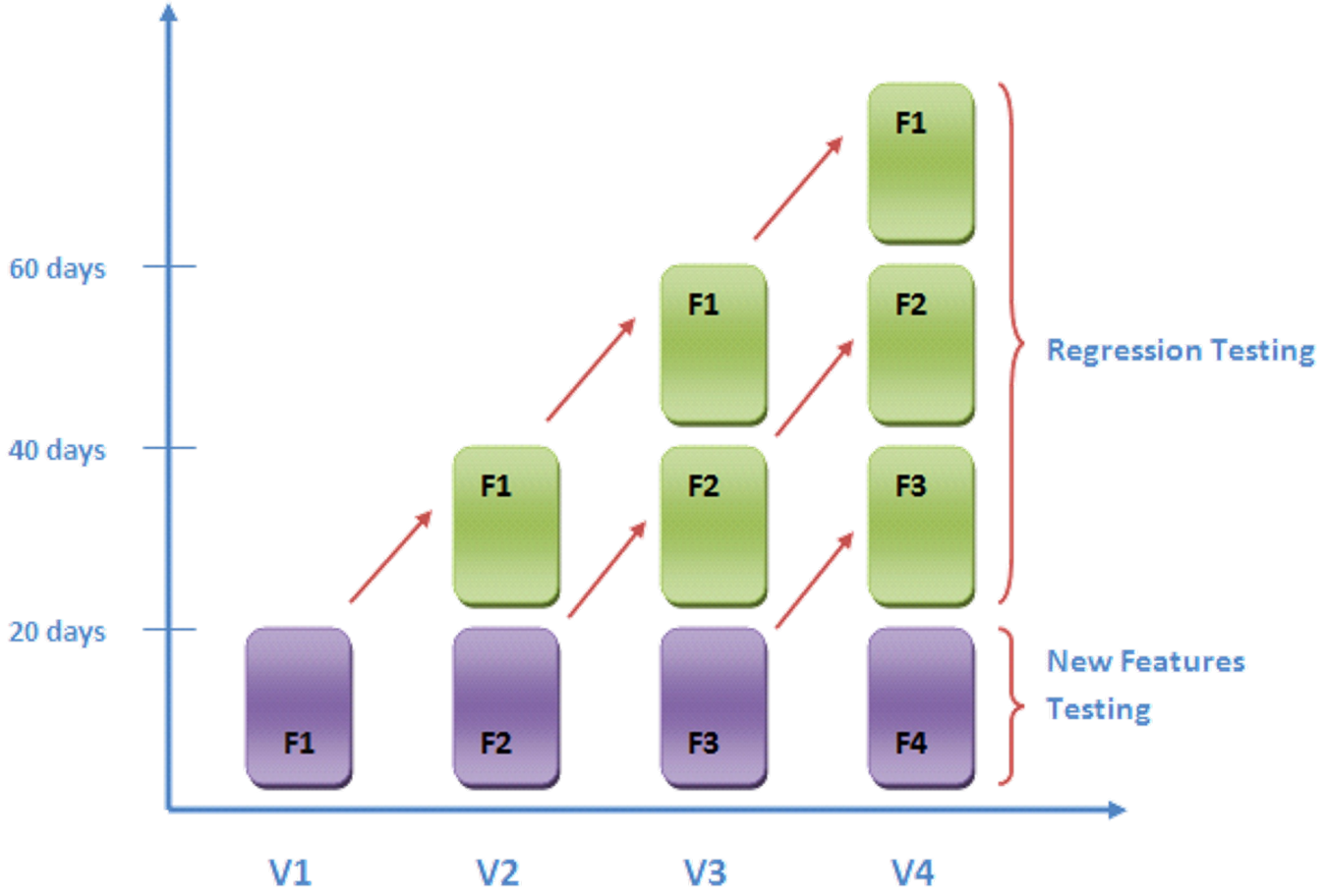
Rationale for Unit Testing

- Software deployed in a modular way
- Important to be sure that each module:
  - Meets specification
  - Is reliable
  - Has no bugs
- Works well with OO software development
- Fits into V-model and Agile development lifecycle
- Isolates the (potential) problem

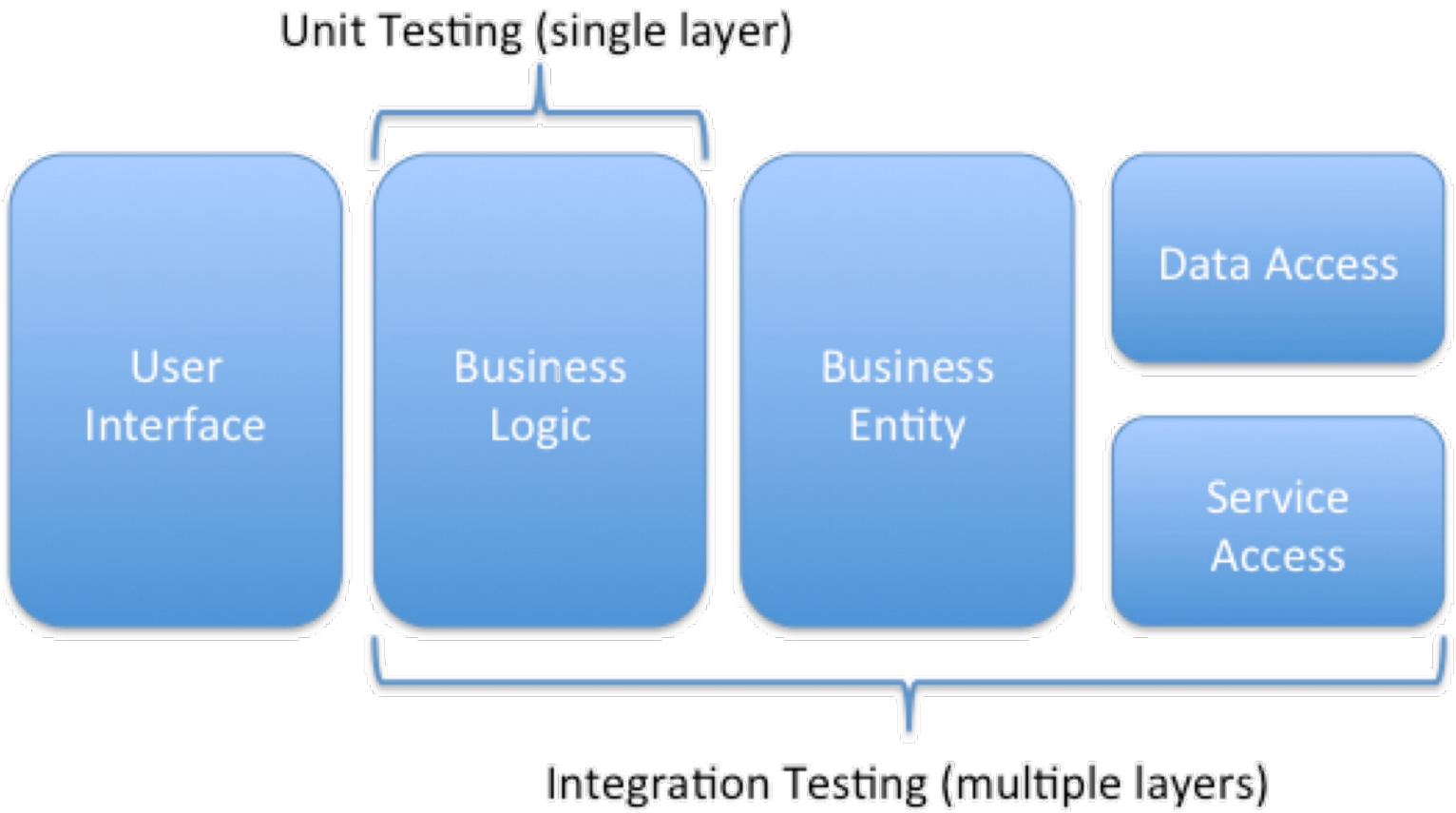
Test Automation with Jenkins (Continuous Integration)



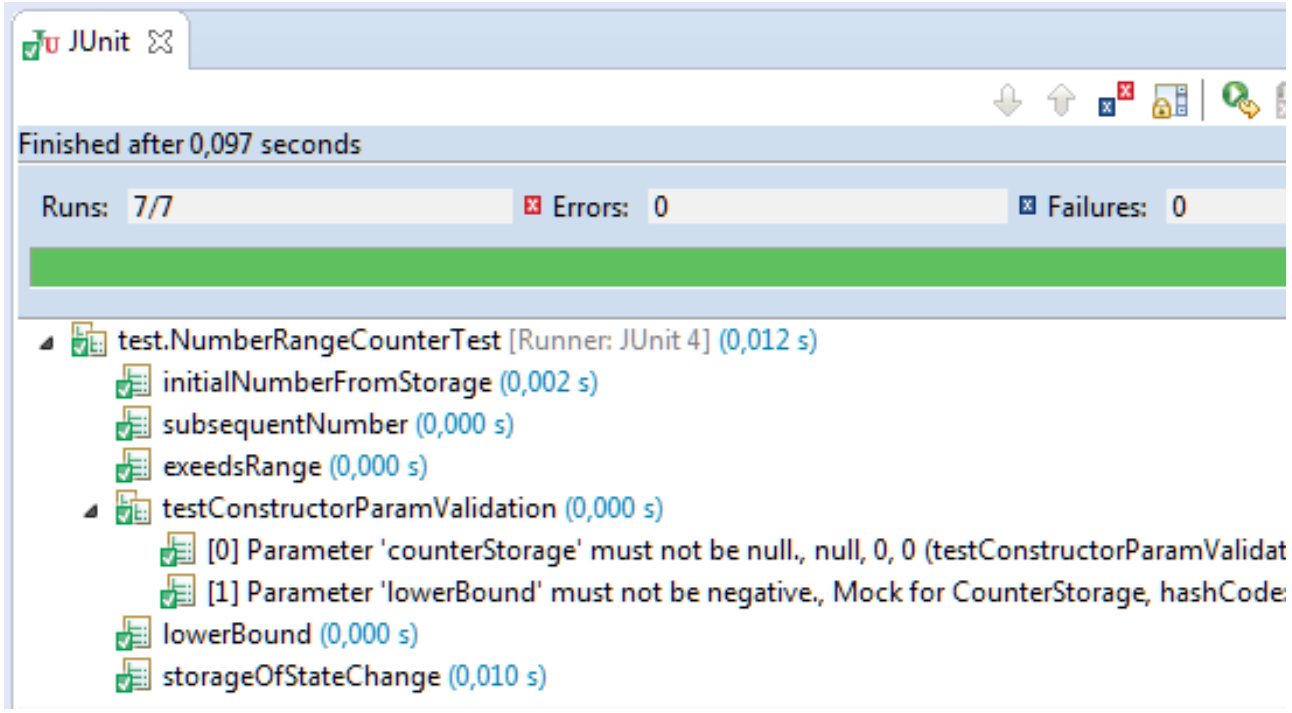
Regression Testing



Unit vs. Integration Testing



JUnit Framework for Java



Test Coverage – e.g. Cobertura

Coverage Report - All Packages

Package	# Classes	Line Coverage		Branch Coverage		Complexity
All Packages	221	84%	2970/3513	81%	859/1060	1.727
junit.extensions	6	82%	52/63	87%	7/8	1.25
junit.framework	17	76%	399/525	90%	139/154	1.605
junit.runner	3	49%	77/155	41%	23/56	2.225
junit.textui	2	76%	99/130	76%	23/30	1.686
org.junit	14	85%	196/230	75%	68/90	1.655
org.junit.experimental	2	91%	21/23	83%	5/6	1.5
org.junit.experimental.categories	5	100%	67/67	100%	44/44	3.357
org.junit.experimental.max	8	85%	92/108	86%	26/30	1.969
org.junit.experimental.results	6	92%	37/40	87%	7/8	1.222
org.junit.experimental.runners	1	100%	2/2	N/A	N/A	1
org.junit.experimental.theories	14	96%	119/123	88%	37/42	1.674
org.junit.experimental.theories.internal	5	88%	98/111	92%	39/42	2.29
org.junit.experimental.theories.suppliers	2	100%	7/7	100%	2/2	2
org.junit.internal	11	94%	149/157	94%	53/56	1.947
org.junit.internal.builders	8	98%	57/58	92%	13/14	2
org.junit.internal.matchers	4	75%	40/53	0%	0/18	1.391
org.junit.internal.requests	3	96%	27/28	100%	2/2	1.429
org.junit.internal.runners	18	73%	306/415	63%	82/130	2.155
org.junit.internal.runners.model	3	100%	26/26	100%	4/4	1.5
org.junit.internal.runners.rules	1	100%	35/35	100%	20/20	2.111
org.junit.internal.runners.statements	7	97%	92/94	100%	14/14	2
org.junit.matchers	1	9%	1/11	N/A	N/A	1

Defined by:

- Statements
- Branches
- Paths
- Conditions

Unit Test Quality

Consistent

- Always return the same result with no changes to code

Atomic

- Pass or fail – no partial failures
- No dependencies on other tests

Single Responsibility

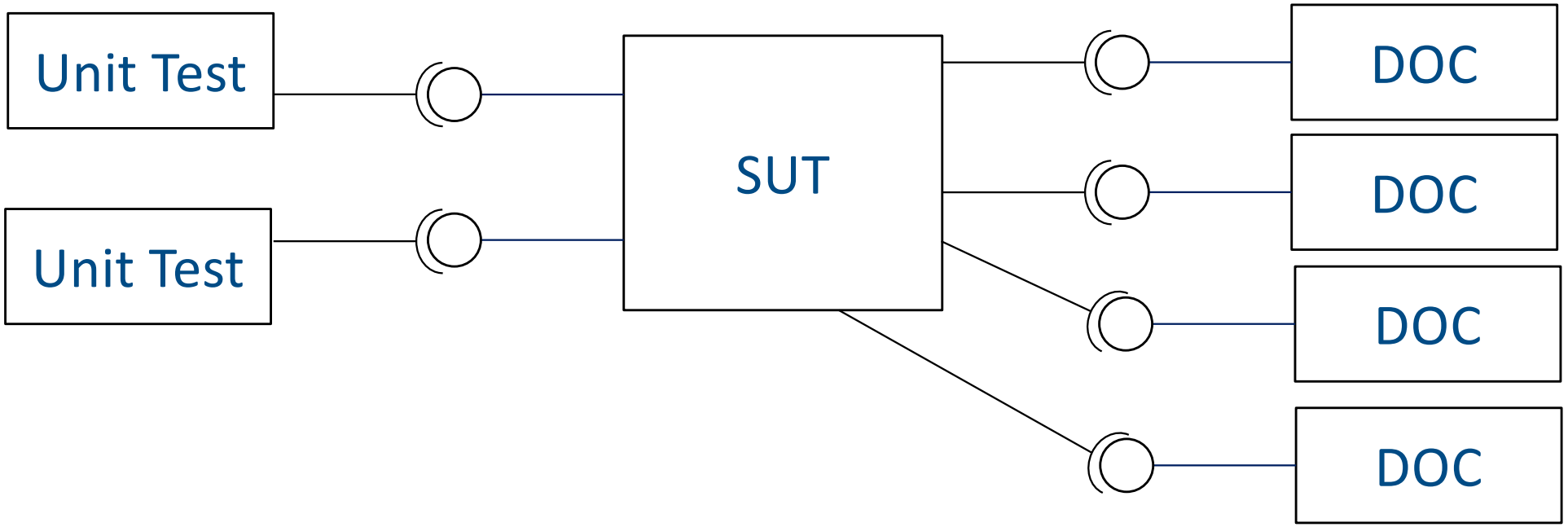
- One test, one assertion

Self-descriptive and easy to follow

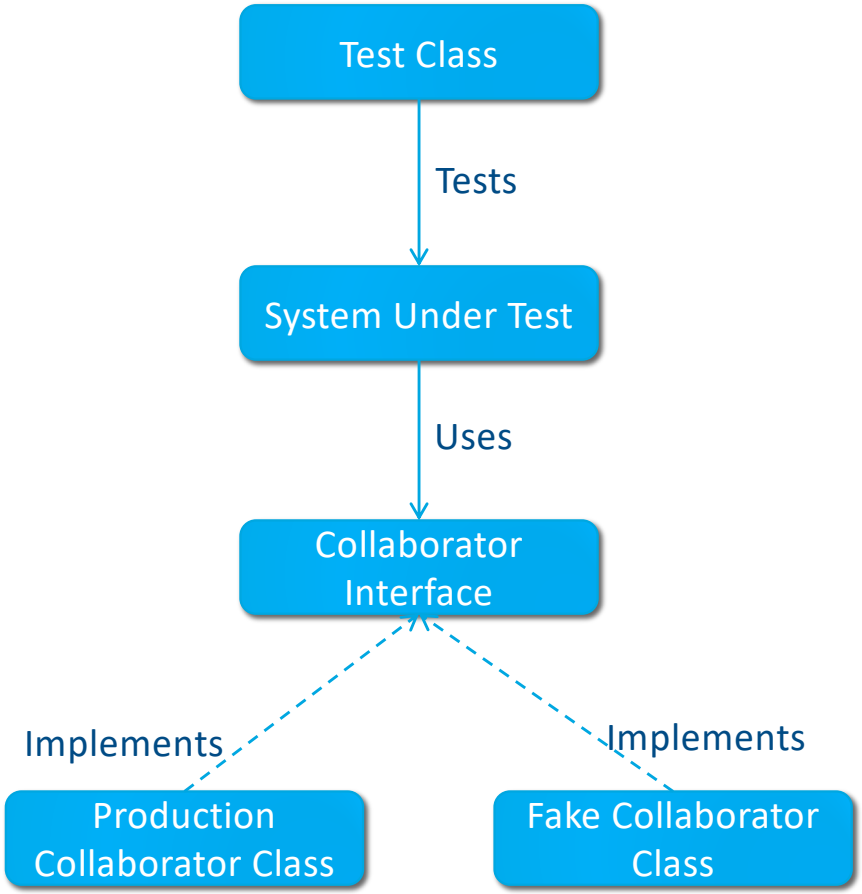
- No logic or loops
- No exception handling unless testing for one specific exception
- Name reflects assertion being tested
- Variables/methods clearly named
- Informative assertion messages

Mocking

What is being tested?



Test Doubles/Fakes



Automated Testing

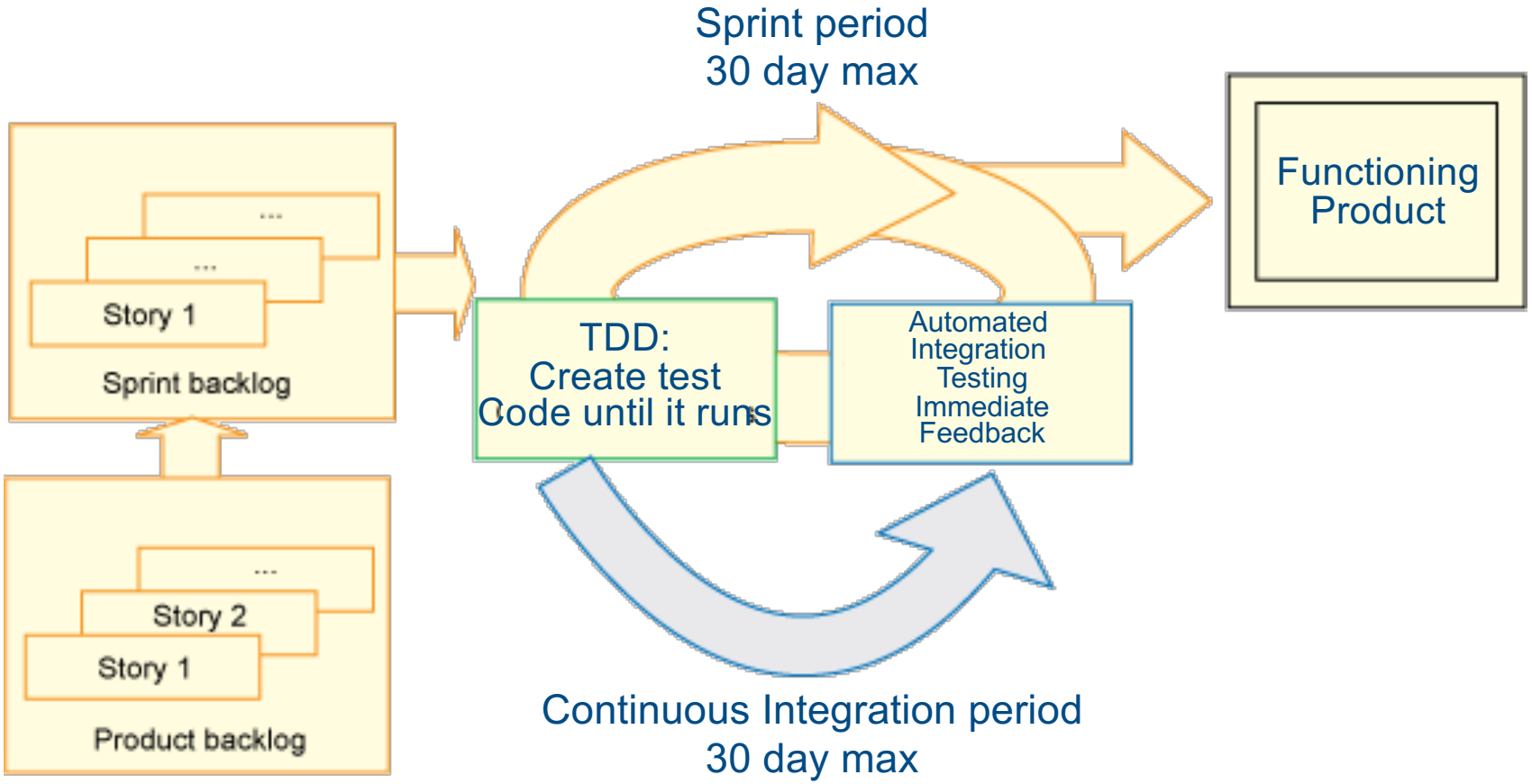
Manual:

- Compile Test Project
- Open in Test Runner
- Select and Run Tests

Automated:

- Build Server Compiles and Runs Tests Based on Scripts
- Any Test Failure Halts Build
- In CD, the Build will not be Delivered

Integration Testing



Architecture Risk Assessment

Threat modeling identifies:

- Design violates security design patterns.
- System omits security controls.
- Security controls suffer from misconfiguration, weakness, or misuse.

Architecture risk analysis uses:

- Attack resistance analysis
- Underlying framework analysis
- Ambiguity analysis

A security architecture survey (SAS) evaluates:

- Compliance with industry best practices

Penetration Testing

External:

- Access server
- Gain ID credentials
- Disrupt

Internal:

- Gain extra authorisation
- Bypass auditing and detection
- Bypass access control

Network:

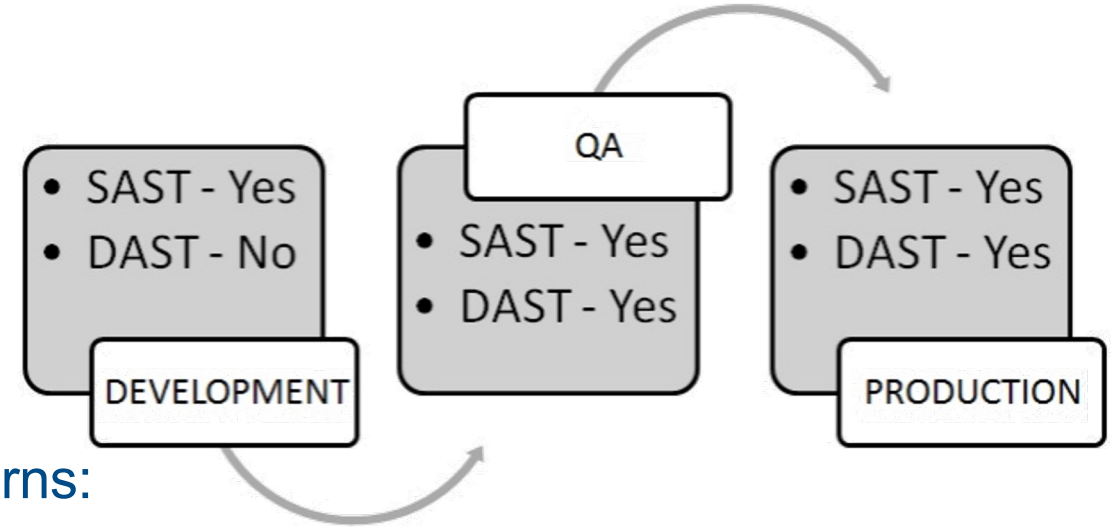
- Move between nodes
- Move between networks
- Access key data

Implications:

- Fail ITHC
- Require rework
- Delay delivery
- Worst case: vulnerability found in Live

Static Testing - SAST

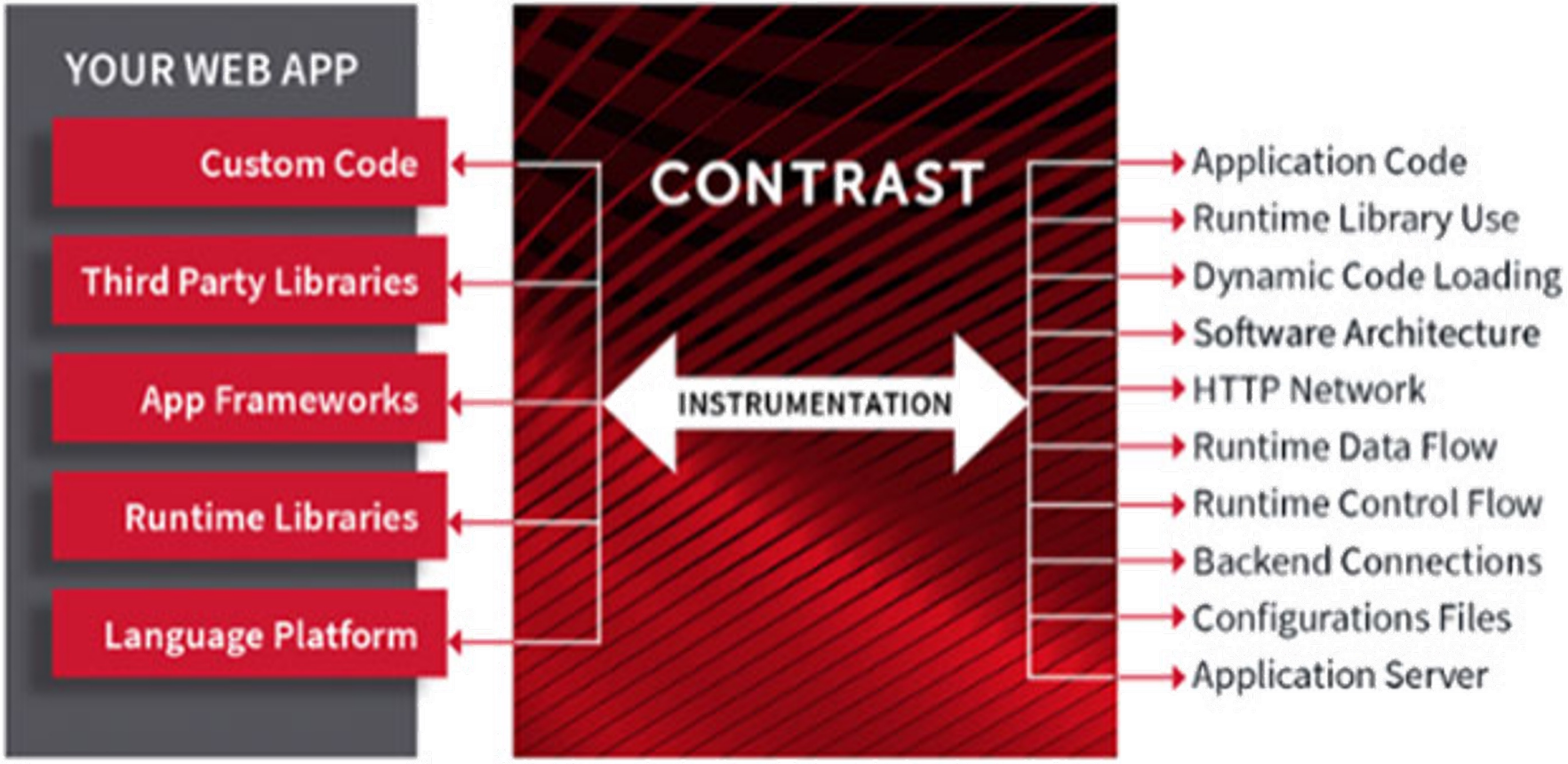
- Starts in single developer coding
- Gives immediate feedback
- White Box
- Source Code Analysis
- Based on industry standard patterns:
  - OWASP Top-10
  - SANS Top 25 Software Errors
- Identifies Technical Debt



Fuzzing

- A “poet” creates the malformed inputs or test cases
- A “courier” delivers test cases to the target software
- An “oracle” detects target failures
- Result:
  - Exception
  - Failure
  - Internal details exposed
  - Secure data returned
  - Crash
  - Denial of Service

Security Testing DAST



Performance Testing

Load Test (or Workload Test)

- Designed to simulate the current “peak day in production”

Stress Test

- Understanding how a system reacts to extreme load conditions—like 200% of current production peak, for example

Capacity Test

- Sees if the footprint and number of servers are right-sized

Endurance or Soak Test

- Runs for an extended time to see if anything fails

REVIEW– Software Testing

- UI Testing
- Accessibility testing
- Test-Driven Development
- Unit Test Frameworks
- Mocking Frameworks
- Behaviour-Driven Development
- Test Automation
- Integration Testing
- Security testing
- Performance Testing

Further Reading – Software Testing

www.checkmarx.com/resources/

www.synopsys.com/software-integrity/security-testing/fuzz-testing.html

medium.com/agile-vision/starting-with-bdd-for-collaborative-development-in-agile-environments-5fb034078b3c

Videos:

blog.codeship.com/cucumber-bdd/

SOFTWARE DEPLOYMENT

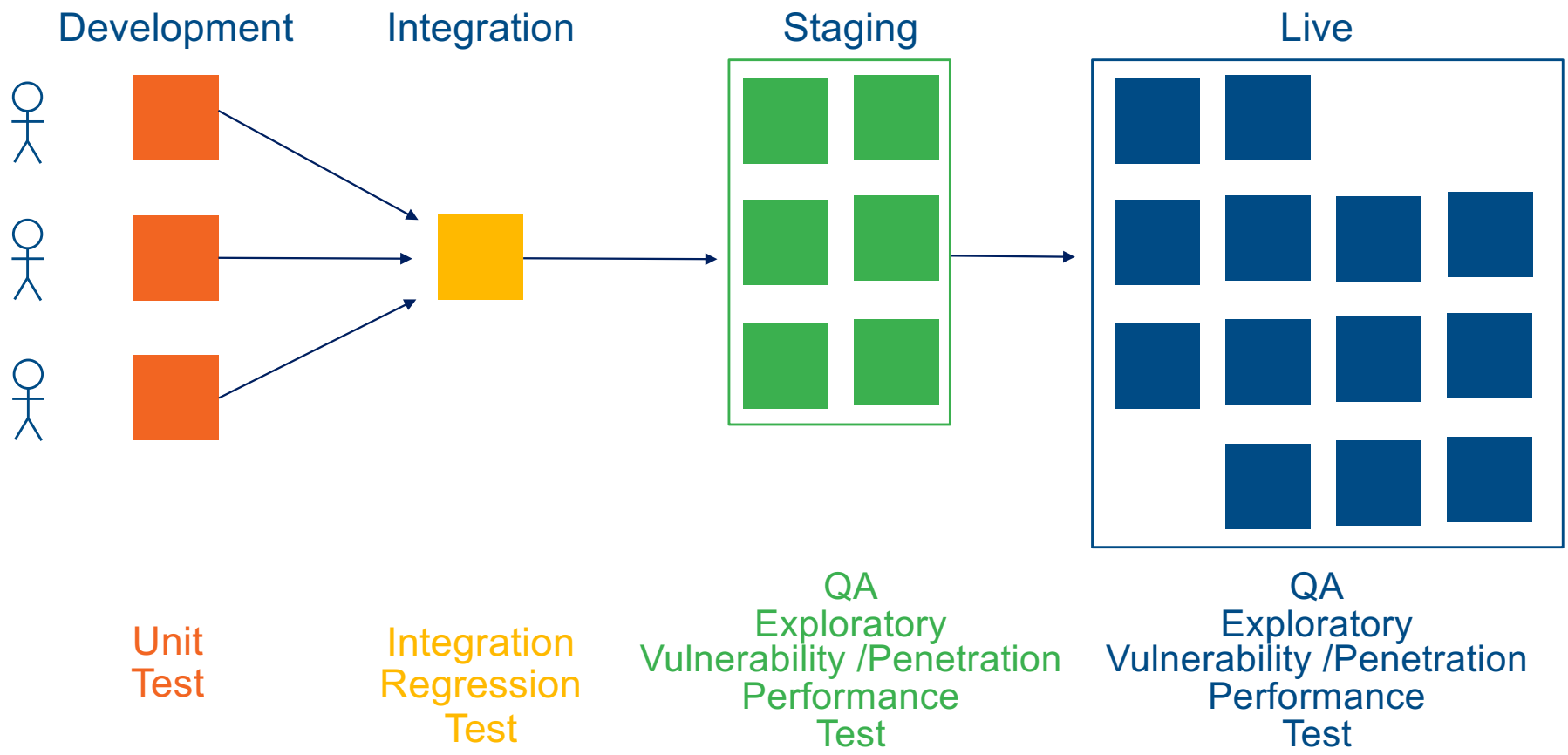
AGENDA – Software Deployment

- Development, Testing and Production Environments
- Continuous Integration
- Cloud and Hosted Deployment
- Implementation Strategies
- Continuous Deployment
- Service logging and Health monitoring
- Operational Metrics
- Live operating model

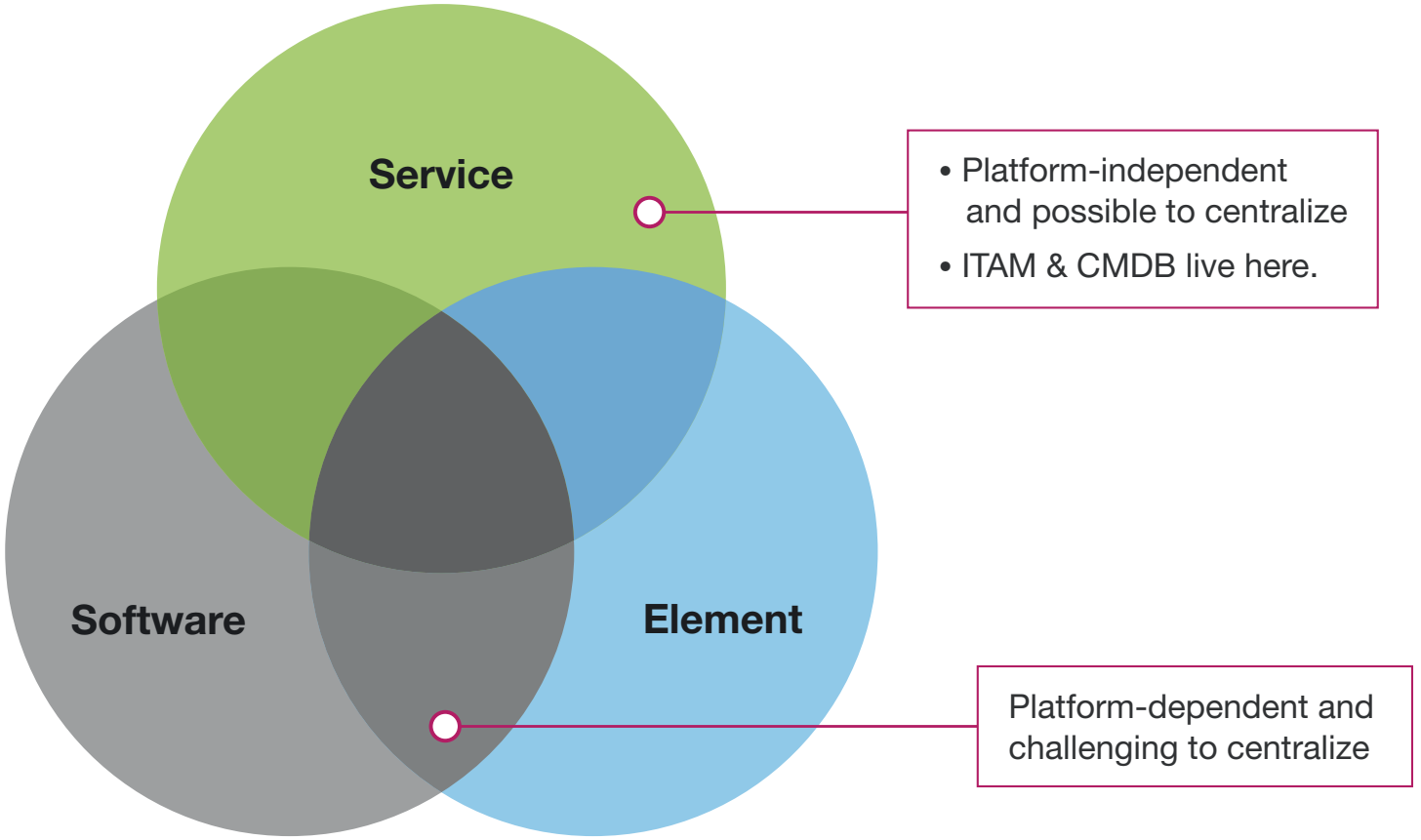
Course Approach – Software Deployment

- Knowledge transfer about Software Deployment tools and techniques
- Significance to Delivery Managers
- Scenarios
- Questions
- Conversations

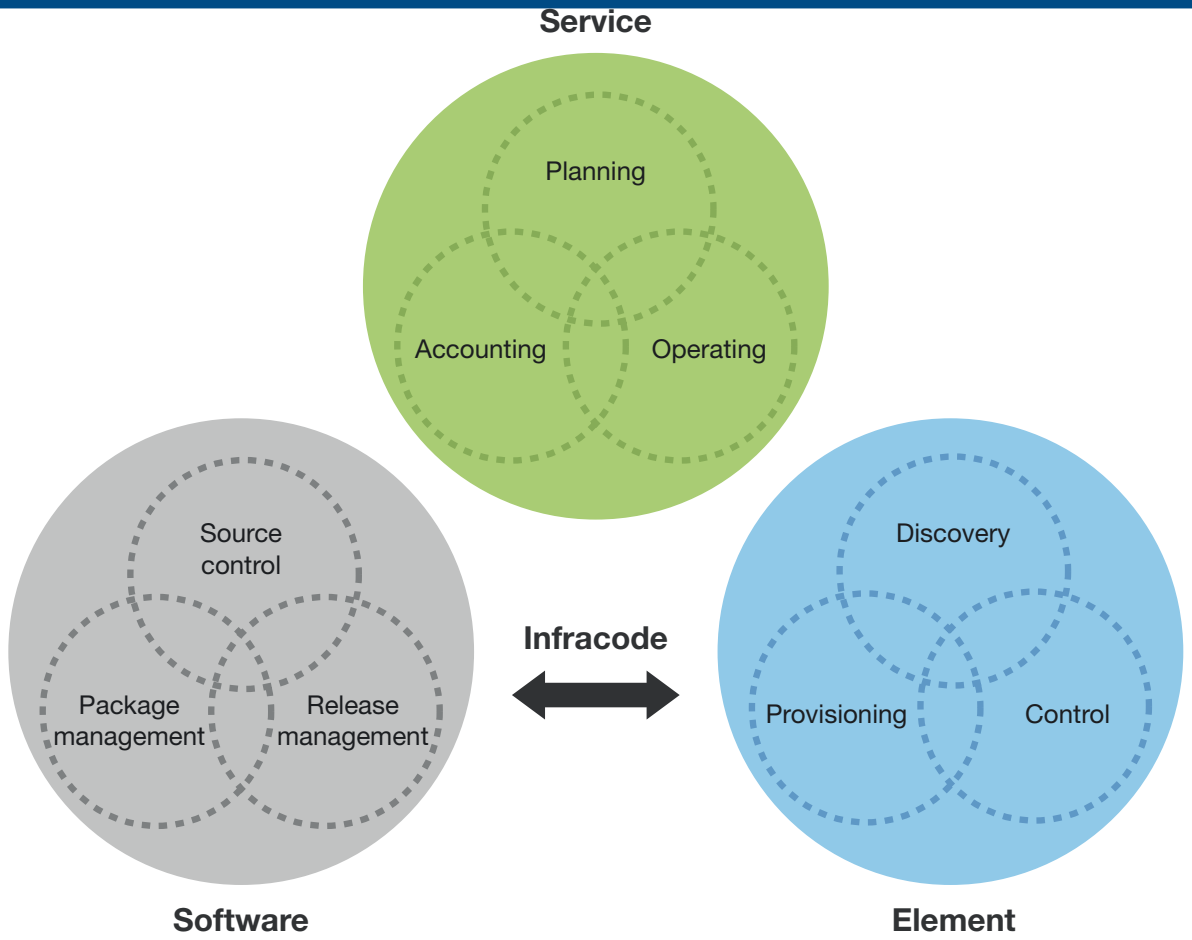
Development, Testing and Production Environments



IaC: Asset and Configuration Domains



IaC: Domain Processes and Tools

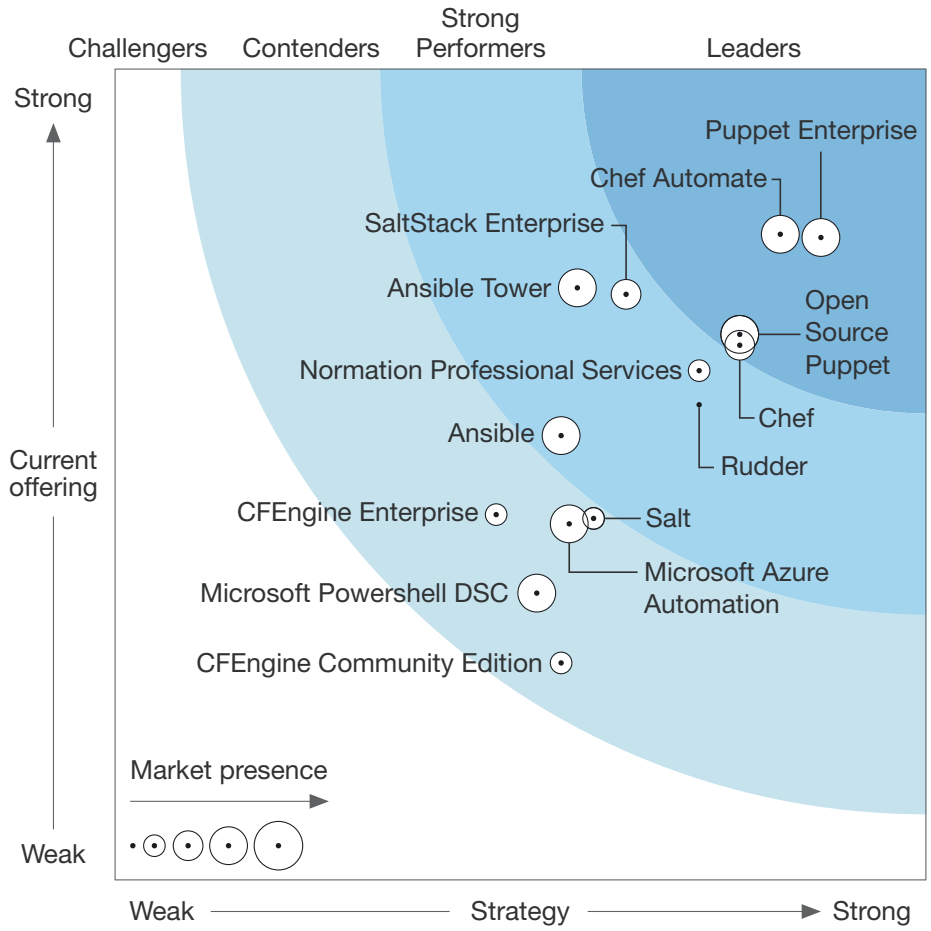


Configuration Management

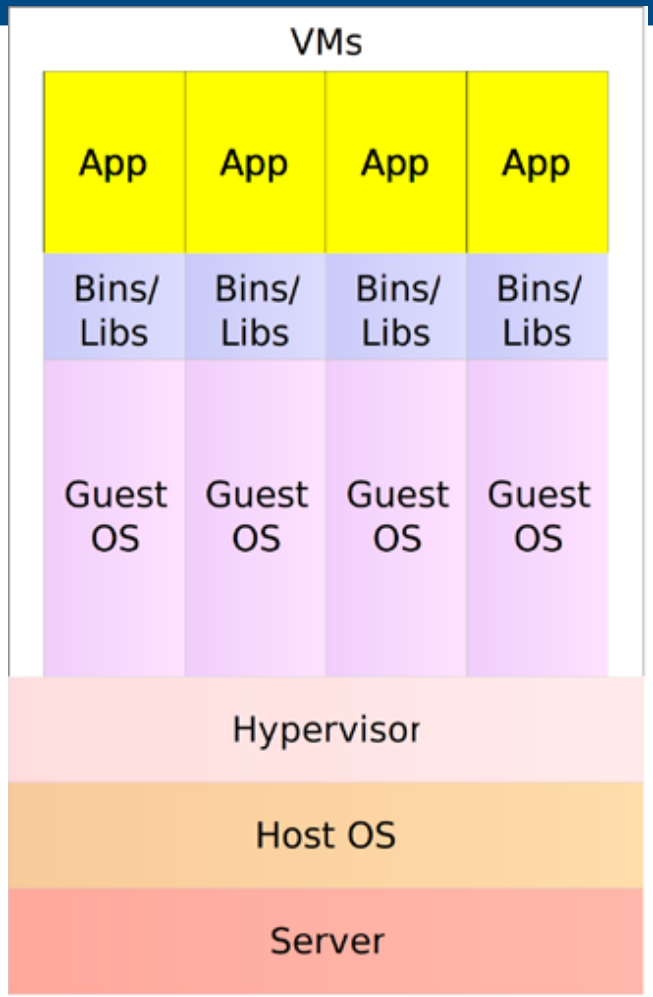
Configuration management tools:

- help to document and maintain configurations and dependencies
- are available in open source:
  - Ansible
  - CFEngine
  - Chef
  - Puppet
- IaC promotes:
  - testing
  - reuse
  - documentation

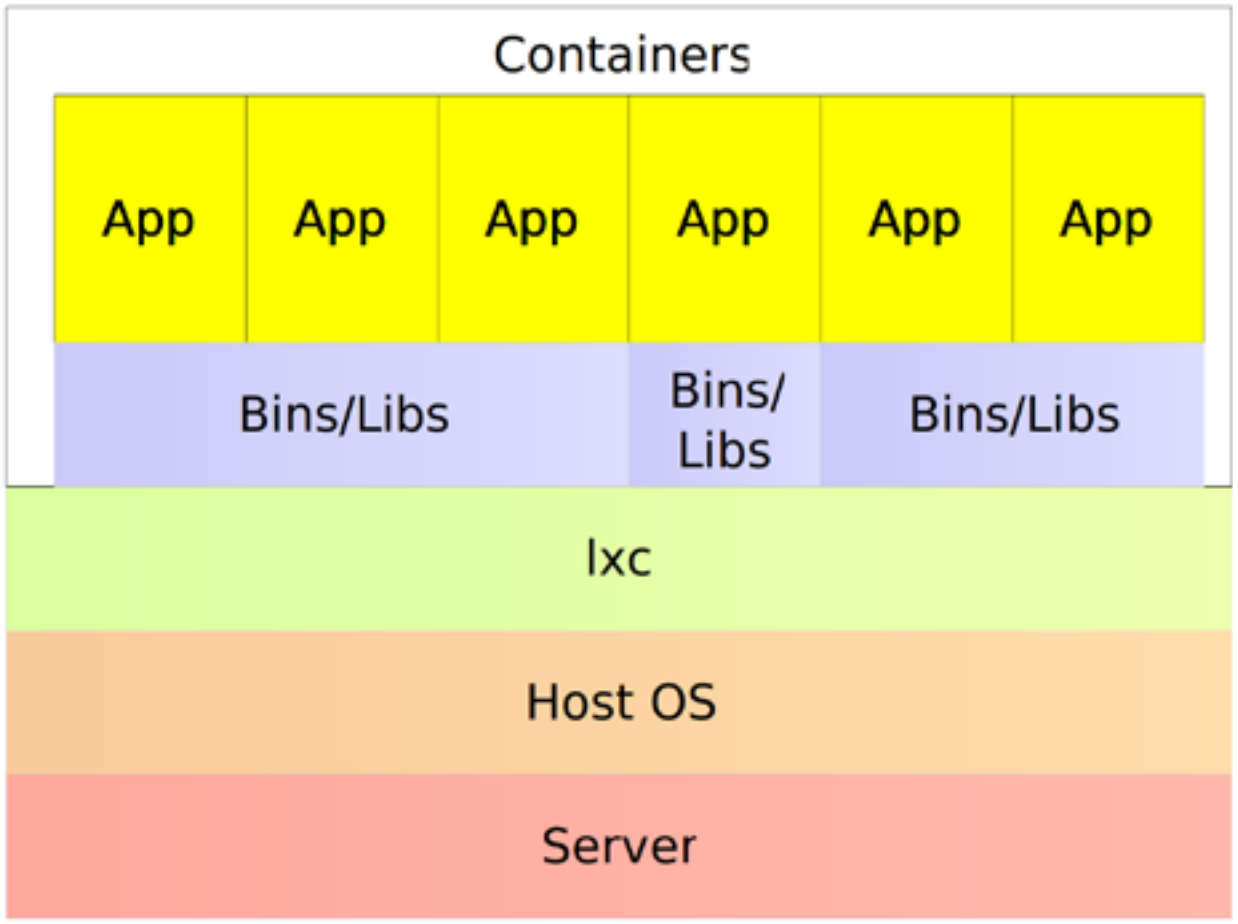
IaC Tools


Challengers
Contenders
Strong Performers
Leaders
Strong
Current offering
Weak
Puppet Enterprise
Chef Automate
SaltStack Enterprise
Ansible Tower
Open Source Puppet
Normation Professional Services
Chef
Ansible
Rudder
CFEngine Enterprise
Salt
Microsoft Azure Automation
Microsoft Powershell DSC
CFEngine Community Edition
Market presence
Weak
Strategy
Strong

Virtual Servers



Containers – e.g. Docker



Continuous Integration

From the Government Service Design Manual (Web operations):

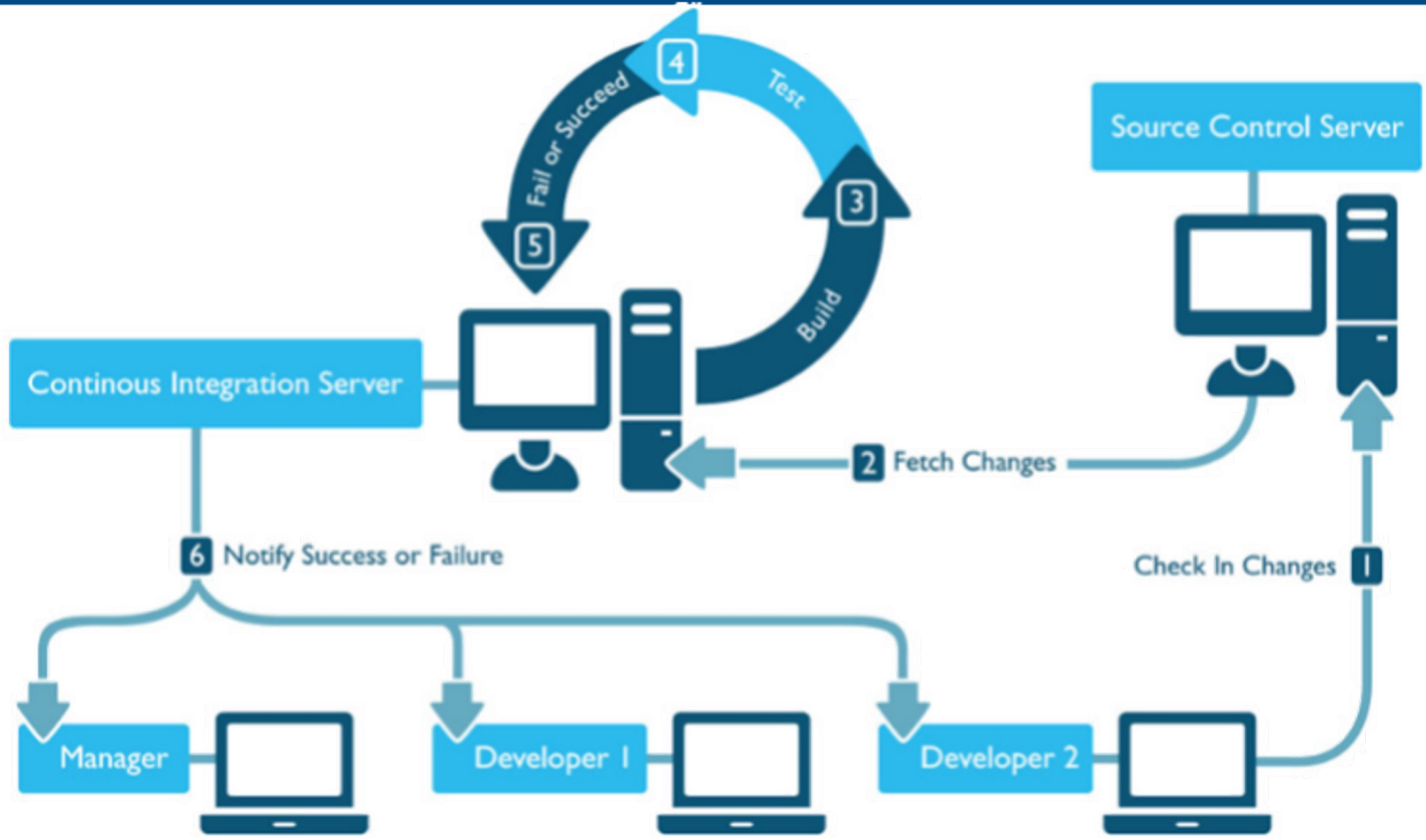
As a developer working on the service

So we can ensure a high level of quality in the code

And so we can minimise the time needed for regression testing

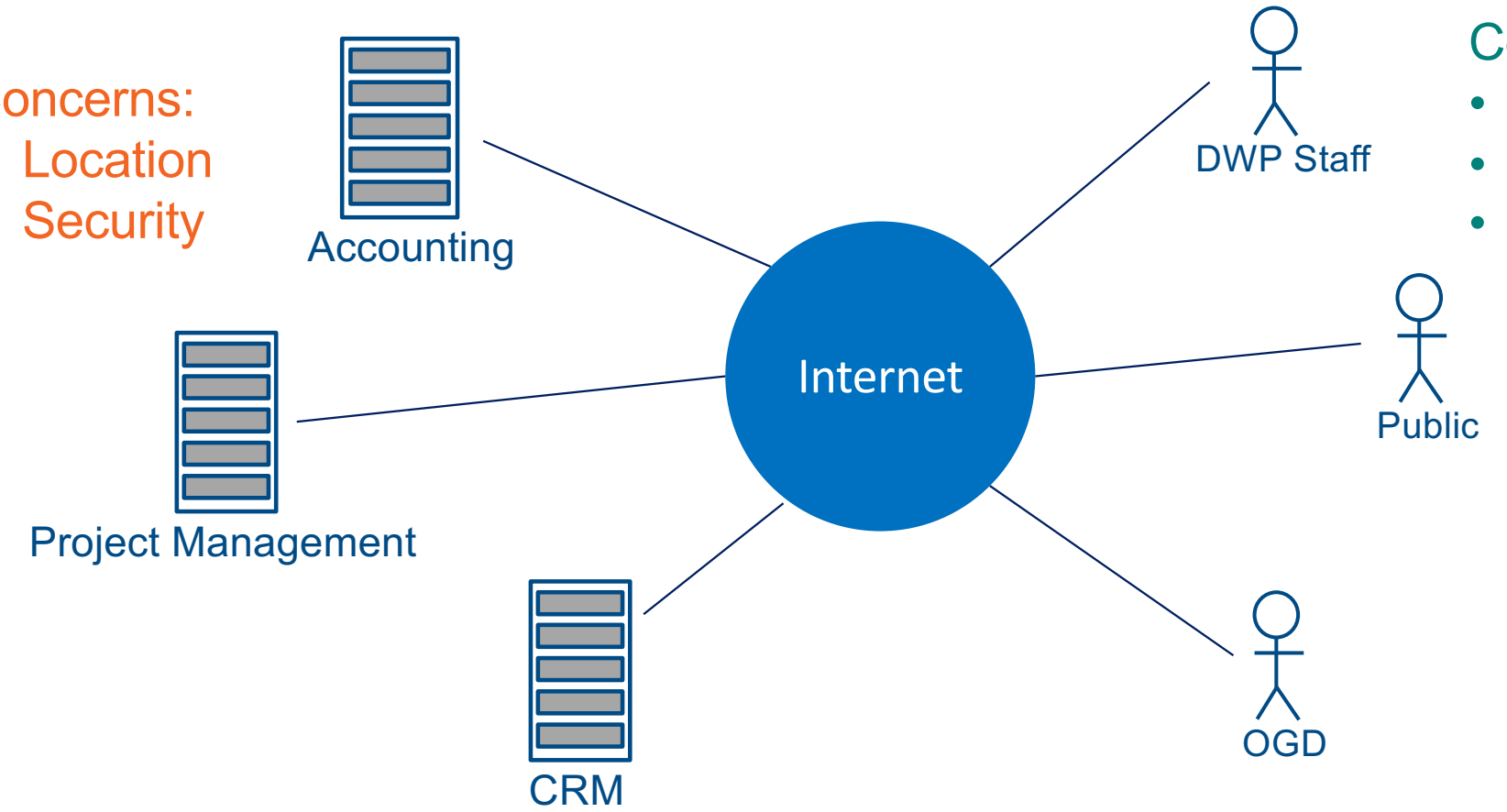
I want a Continuous Integration environment which automatically runs tests against every commit

Continuous Integration



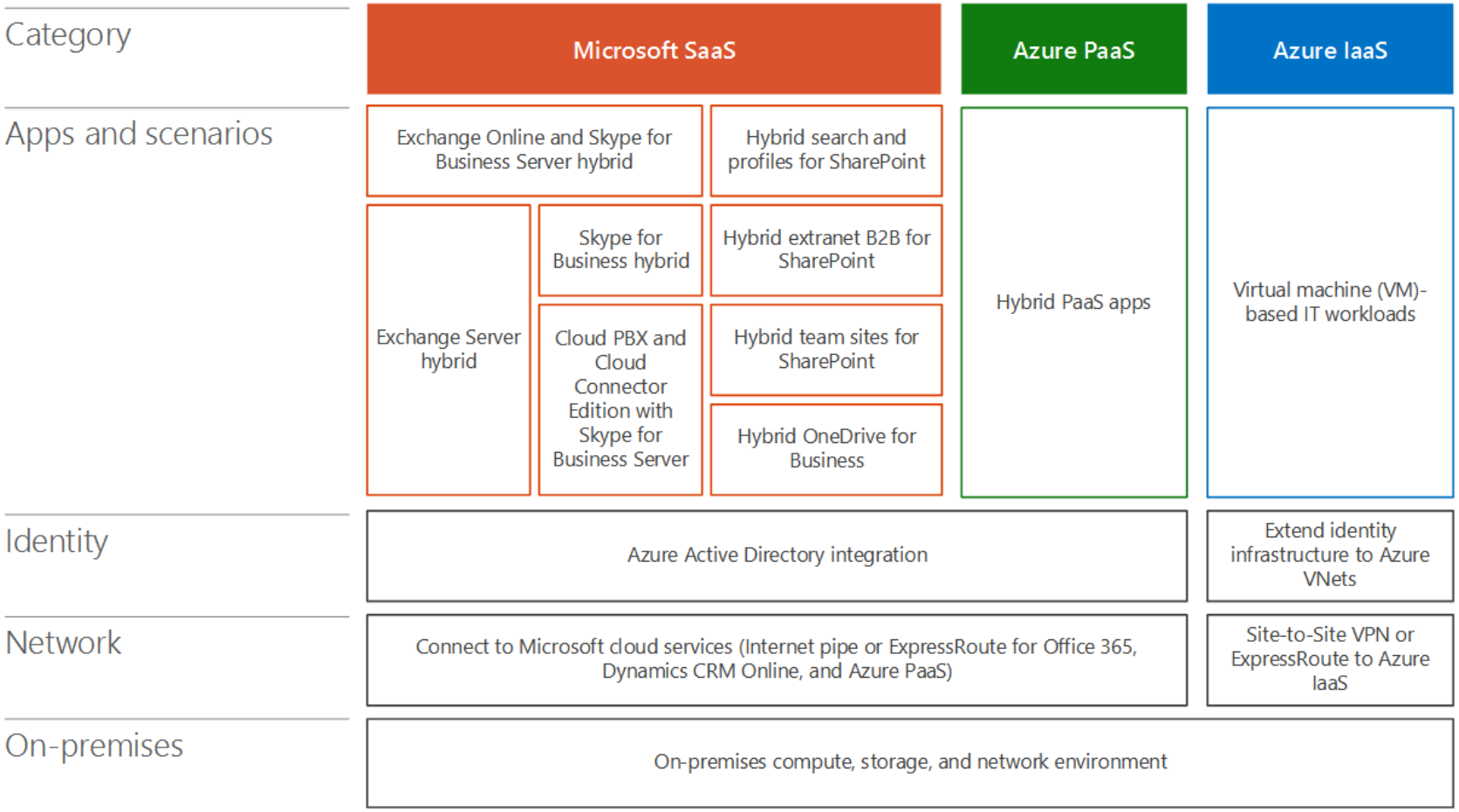
Cloud and Hosted Deployment - SaaS

Concerns:
- Location
- Security

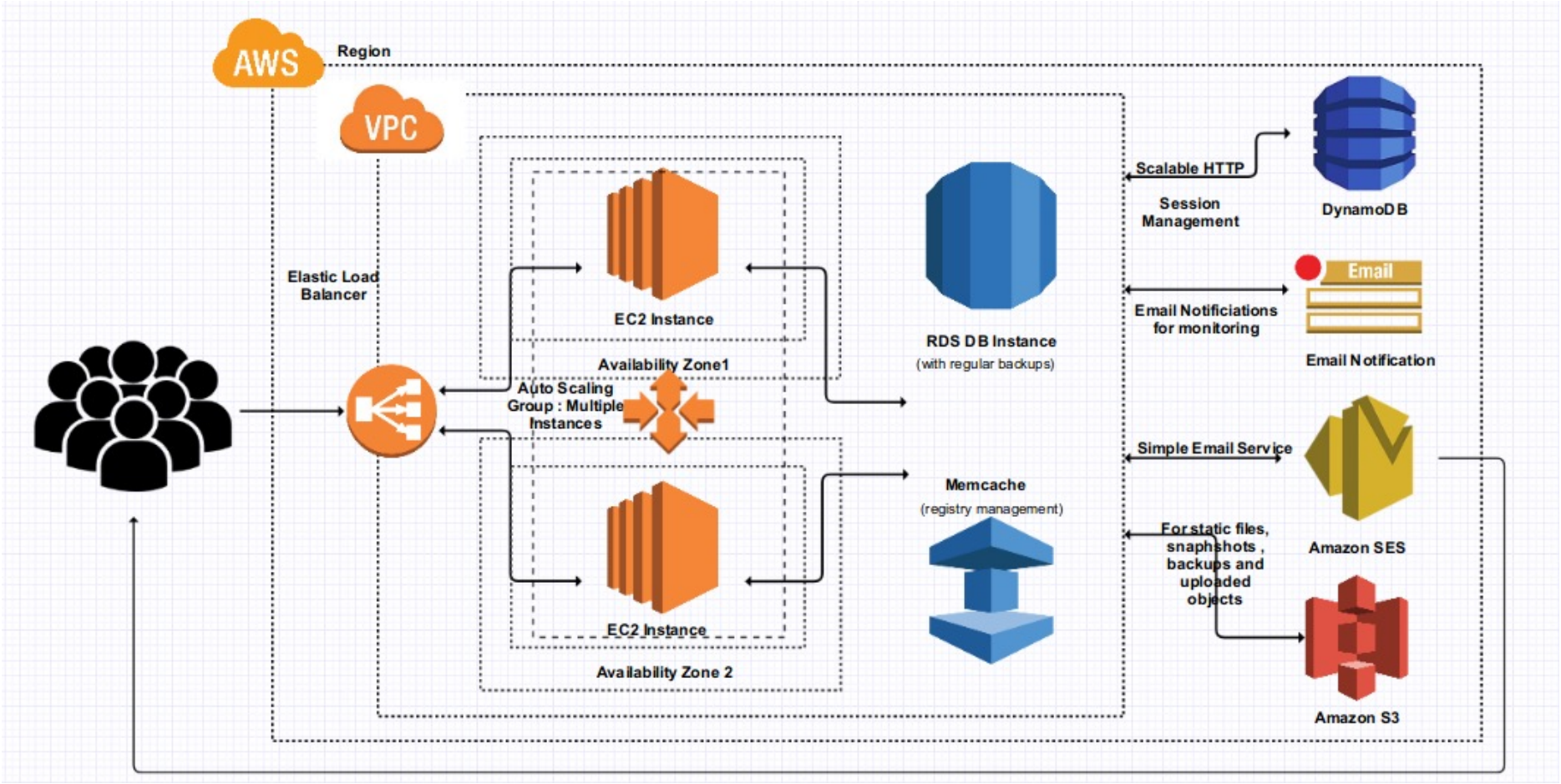


Concerns:
- Access Control
- Accessibility
- Encryption

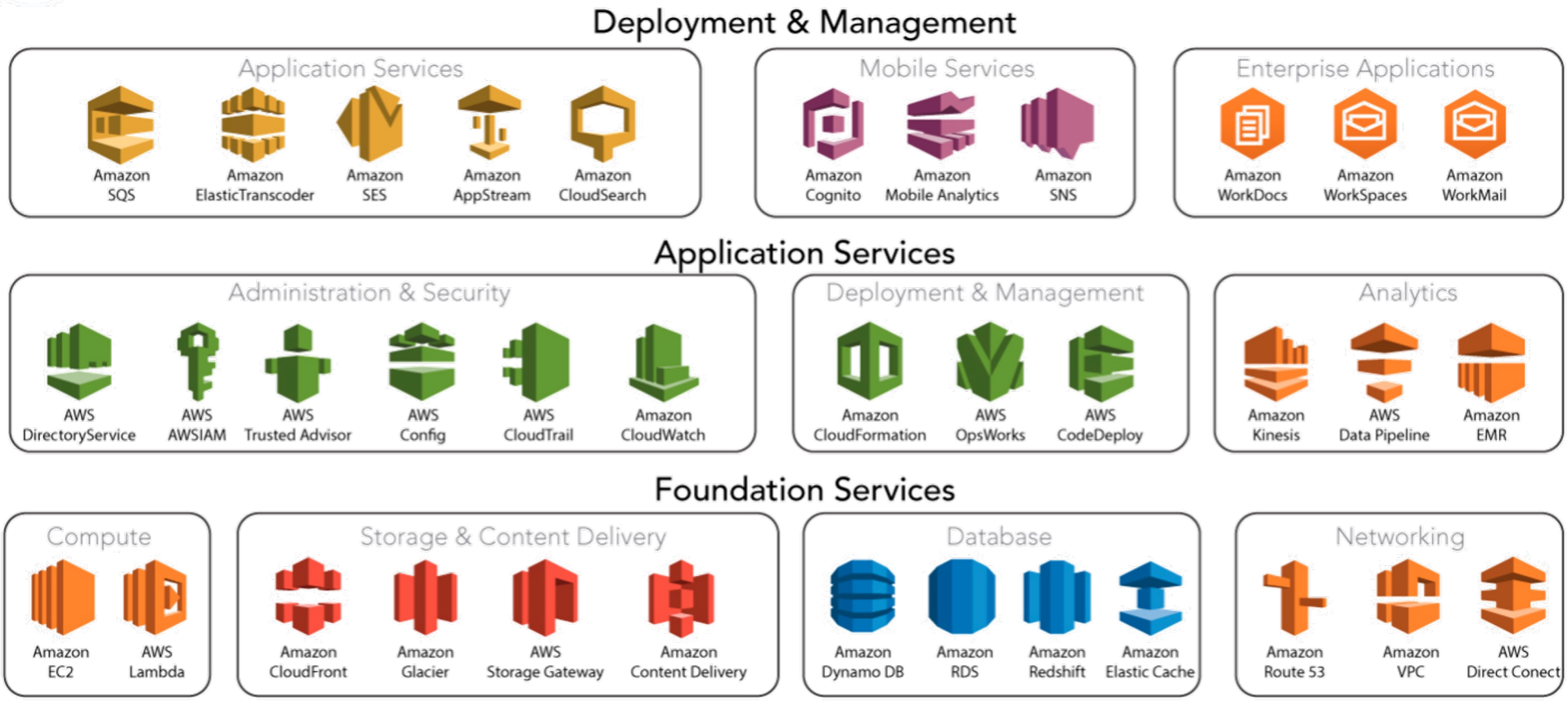
Microsoft Cloud



AWS


AWS
Region
VPC
Elastic Load Balancer
EC2 Instance
Availability Zone1
Auto Scaling Group : Multiple Instances
EC2 Instance
Availability Zone 2
RDS DB Instance
(with regular backups)
Memcache
(registry management)
Scalable HTTP
Session Management
DynamoDB
Email
Email Notificiations for monitoring
Email Notification
Simple Email Service
Amazon SES
For static files, snapshots , backups and uploaded objects
Amazon S3

AWS Services



Cloud and Hosted Deployment – PaaS

Platform as a Service:

- Client (DWP) provided with a running software environment
- Web application may be developed and installed
- Platform configurable
- Restricted access to underlying OS and server components
- Less work and skills required
- Probably higher cost
- Less flexibility and control
- Back-end support redirects to cloud provider

Cloud and Hosted Deployment – IaaS

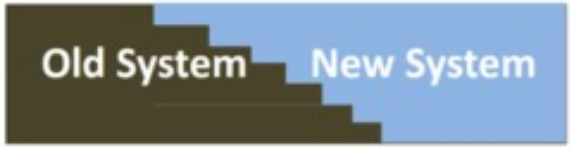
Infrastructure as a Service:

- Client (DWP) provided with a set of virtualised hardware resources
- Software environment must then be installed and configured
- Web application may then be developed and installed
- Platform and infrastructure configurable
- Access to underlying OS and server components
- More work required
- More flexibility available
- May be lower cost
- More support resource required

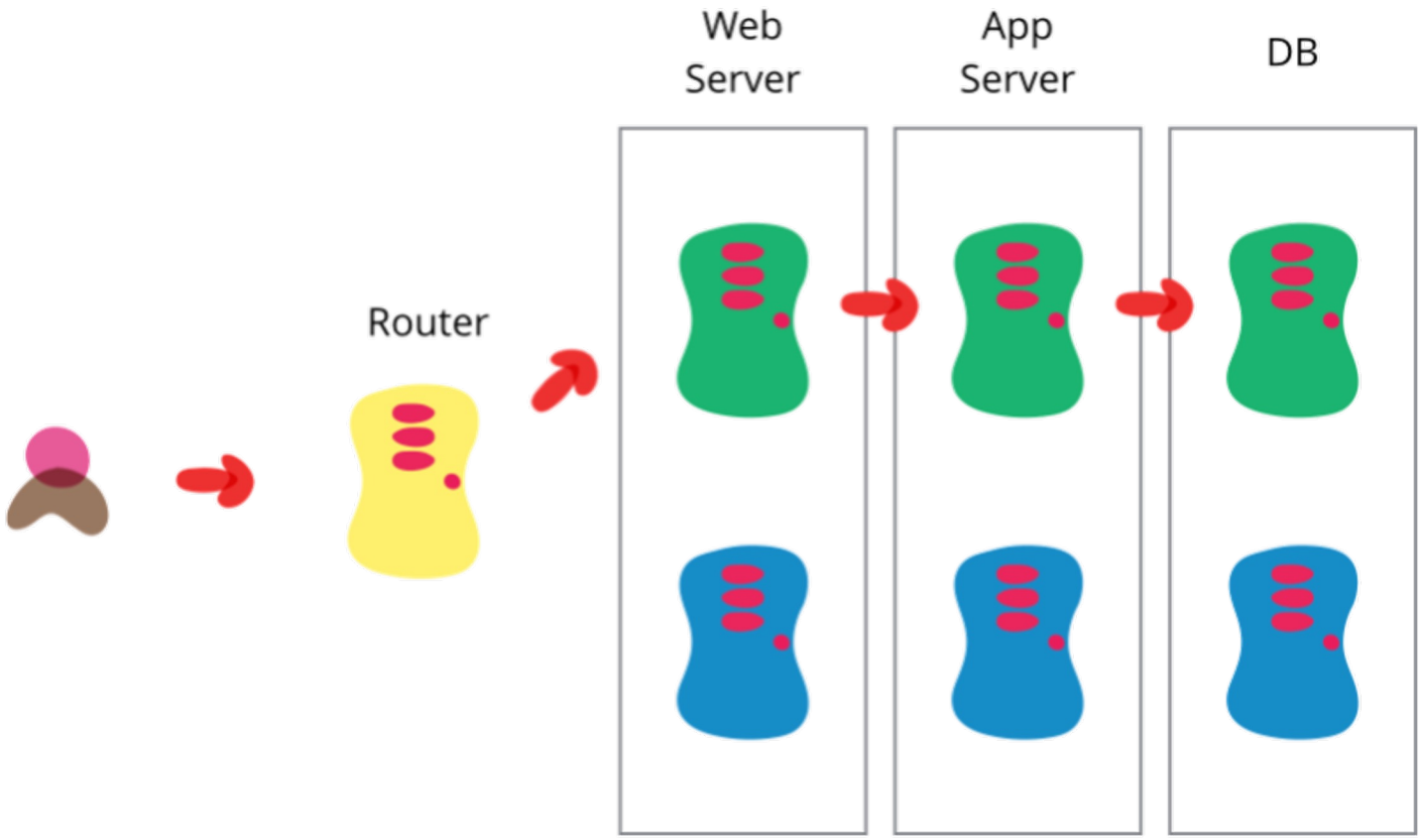
Cloud Providers

- UKCloud: Public cloud for the exclusive use of UK Public Sector organisations
- Crown Hosting: Joint venture between the Cabinet Office and Ark Data Centres
- G-Cloud: Marketplace for Cloud services
- AWS: Commercial Cloud Service Provider
- Others: Azure, Google, IBM etc.

Implementation Strategies

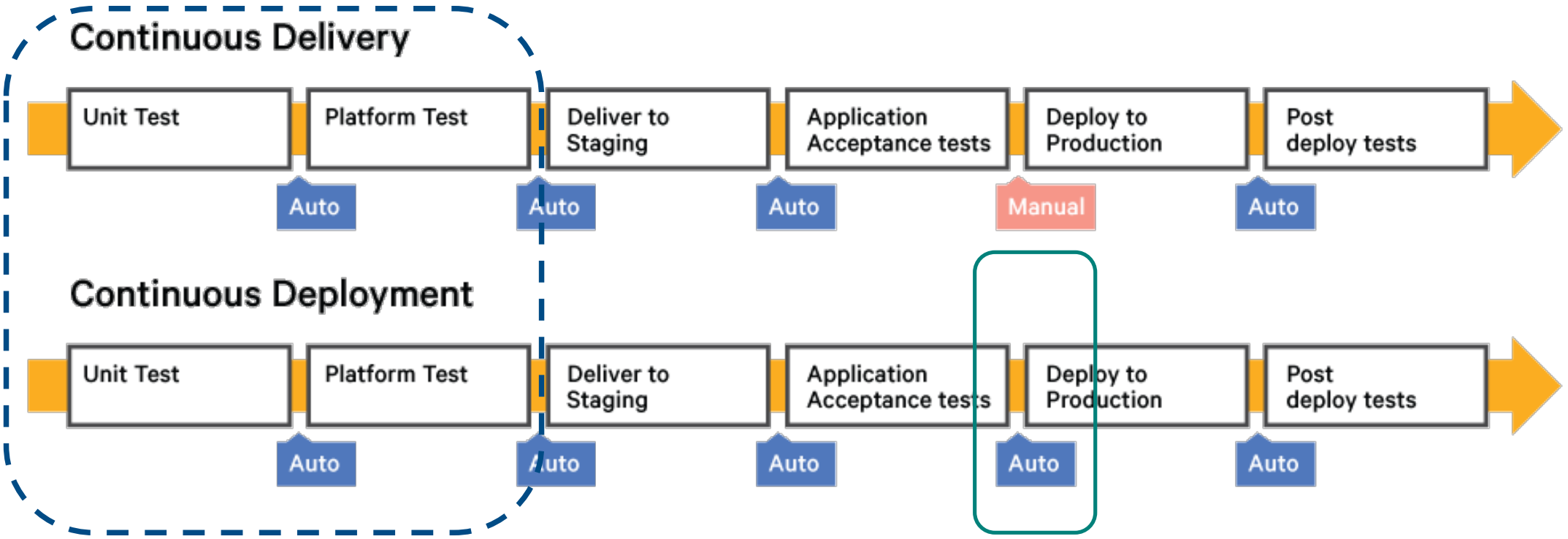
Parallel Transition		*Both old and new systems are operated simultaneously until Steering Committee agrees to covert.*
Pilot Transition		*One department, business unit test new system first.*
Phased Transition		*Only parts of the new system or only a few departments, sites are migrated at a time.*
Plunge (Big Bang) Transition		*Direct abandonment of old system and transition to new system.*

Blue/Green Deployment



- Small % = Canary Testing
- Rollback to Blue is:
  - Simple
  - Quick
  - Low risk
- Requires separate resources
- May be optimised differently

Continuous Deployment



Continuous Integration

Software released with no manual gateway

Health monitoring

Application metrics
As a developer working on the service
So that I can gain visibility of how my application is running in production
And so we can find and fix problems with it quickly
I want a simple way of instrumenting my application to feed metrics to the metrics system

System metrics
As a web operations engineer working on the service
So that we can identify and fix problems with the system, ideally before they occur
I want to set up collection of low level system metrics like load, disk, network io, etc.

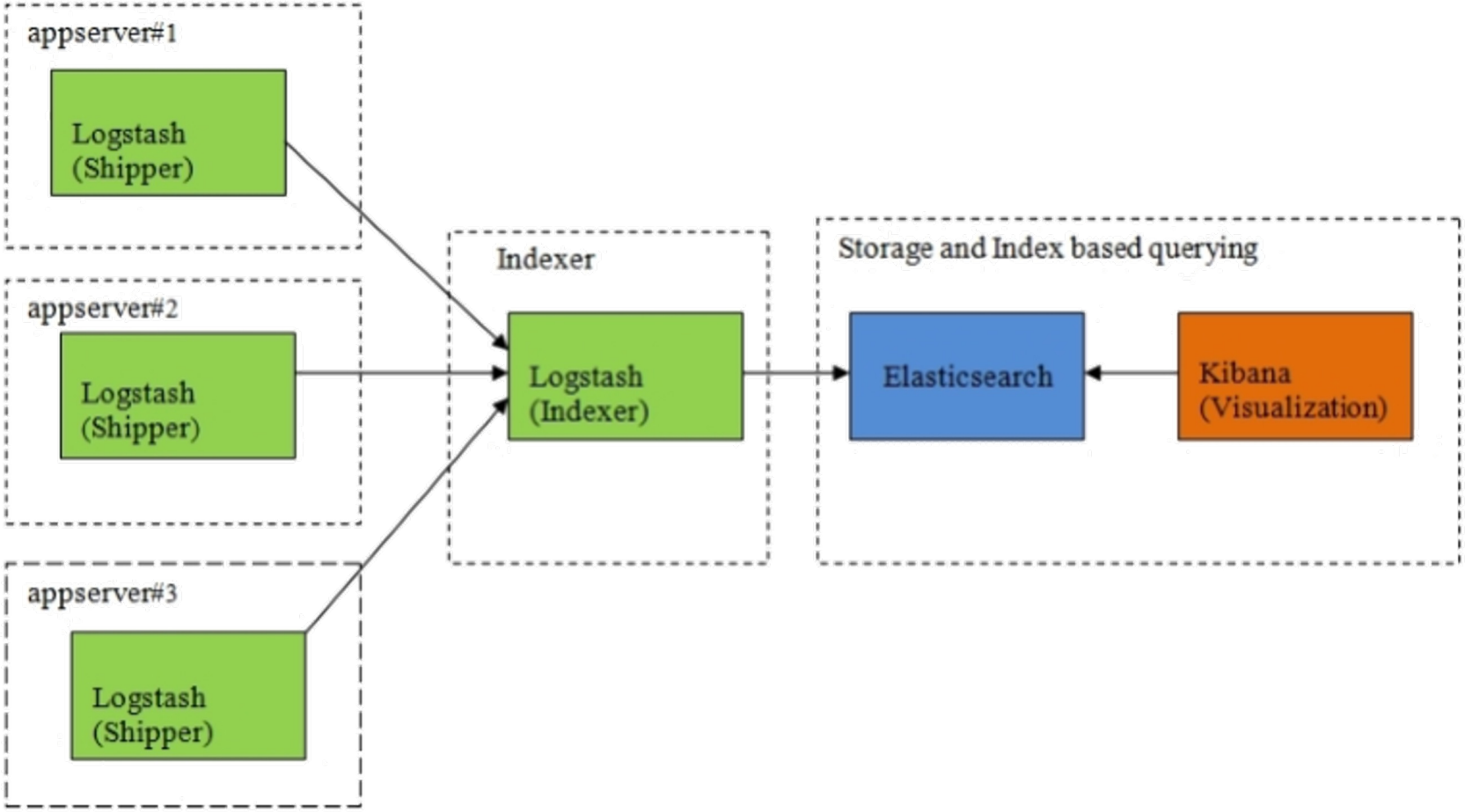
Service Logging

Log collection
As a web operations engineer working on the service
So that I can easily see everything that is happening in specific applications
I want to collect all the logs from applications running on the same host in one place

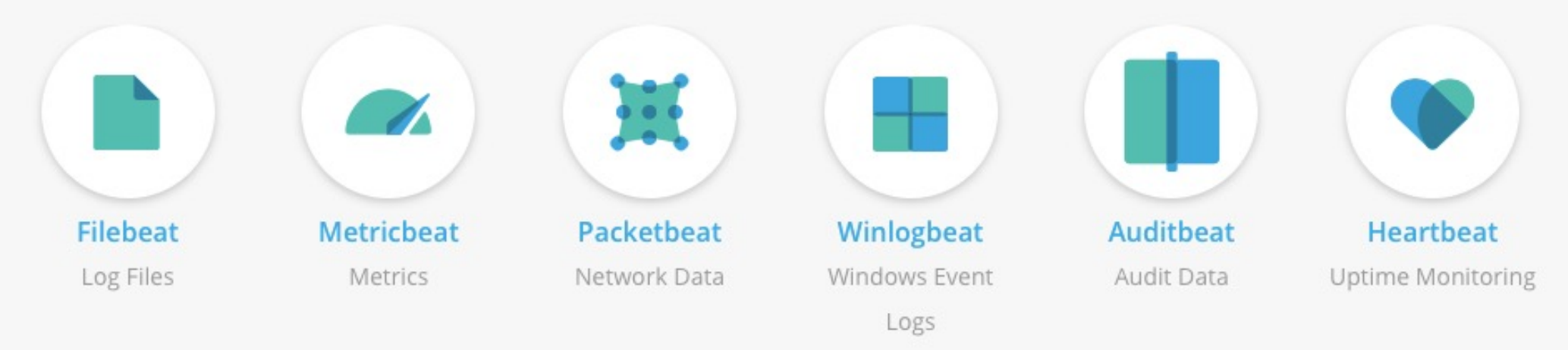
Log viewing
As a web operations engineer working on the service
So that I can see what is happening across the infrastructure
I want a mechanism for viewing and searching logs in as near real time as possible

As a developer working on the service
So that I can extract information from logs to aid with improving the service
I want a mechanism to run queries across the aggregated logs

ELK Stack


appserver#1
Logstash
(Shipper)
appserver#2
Logstash
(Shipper)
appserver#3
Logstash
(Shipper)
Indexer
Logstash
(Indexer)
Storage and Index based querying
Elasticsearch
Kibana
(Visualization)

Beats Services



Measuring Success (reminder)

Cost per transaction

The average cost to government of each transaction.

User satisfaction

Satisfaction is calculated by asking people to rate a service.

Completion rate

The percentage of people who successfully complete a government service.

Digital take-up

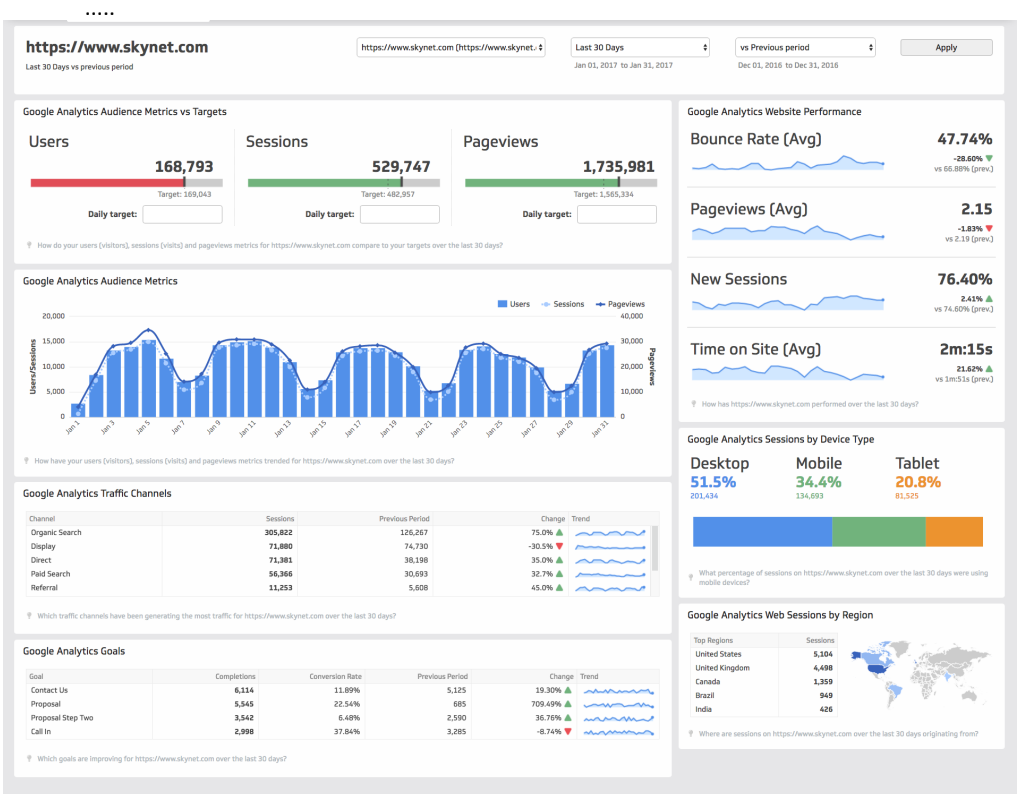
The percentage of people using government services online compared to other methods (eg phone or post).

Operational Metrics

Metric collection service (from government service design manual)

As a web operations engineer working on the service
So that we can collect large numbers of time series metrics from the running service
I want to install and configure a metric collection system

Google Analytics



Acquisition KPIs	Behavior KPIs	Conversions KPIs
Visitors	Bounce Rate	Goal Conversion Rate
Percentage of New Visitors	Pages per Visit	Goal Completions
Number of New Visitors	Average Visit Duration	Goal Value

Activities that Continue in Live

You should:

- continue doing user research
- monitor the status of your service
- maintain uptime and availability
- perform vulnerability and penetration testing
- test your service’s performance
- test new features to make sure they’re accessible
- maintain quality assurance

Live Operating Model

Customer interface	Business relationship management	Business and desktop application portfolio strategy	Business intelligence and analytics	End-user support	End-user computing and devices (for example, laptops, peripherals)	
Applications and management	Application design and architecture	Application development	Testing and QA	**IT planning, service delivery, and governance**		
	Application support	Release, configuration, and change management	Enterprise application integration and middleware	Security and risk management	IT service management	Financial management
Data management	Data governance and master data management	Storage and backup services	Database services	Innovation management	Enterprise architecture	Program and project management (PMO)
Infrastructure management	Data center and network operations	Server administration and management	Data network	Vendor and contract management	Capacity management	Training
	IT asset management		Telecommunications infrastructure			

REVIEW– Software Deployment

- Development, Testing and Production Environments
- Continuous Integration
- Cloud and Hosted Deployment
- Implementation Strategies
- Continuous Deployment
- Service logging and Health monitoring
- Operational Metrics
- Live operating model

Further Reading – Software Deployment

blog.launchdarkly.com/feature-branching-using-feature-flags/

devops.com/feature-branching-vs-feature-flags-whats-right-tool-job/

martinfowler.com/articles/feature-toggles.html

martinfowler.com/articles/continuousIntegration.html

www.ansible.com

cfengine.com

www.chef.io/automate/

puppet.com

martinfowler.com/bliki/BlueGreenDeployment.html

www.atlassian.com/agile/project-management/metrics

Further Reading – Software Deployment

https://github.com/alphagov/government-service-design-manual/blob/master/service-manual/operations/web-operations-stories.md

Videos:

www.youtube.com/watch?v=gxm1C92XhCQ

Course Review

- Summary of Key Topics
- Final Q & A Session
- Further Training (GDS Academy?)

Knowledge Transfer Review