



d-wise™

Accelerate to the Speed of Life

ADo1 : Agile-fall

Andy Ratcliffe & Chris Cowling



Agile-Fall

Agile

Agile-Fall

Requirements

Spikes

Flow

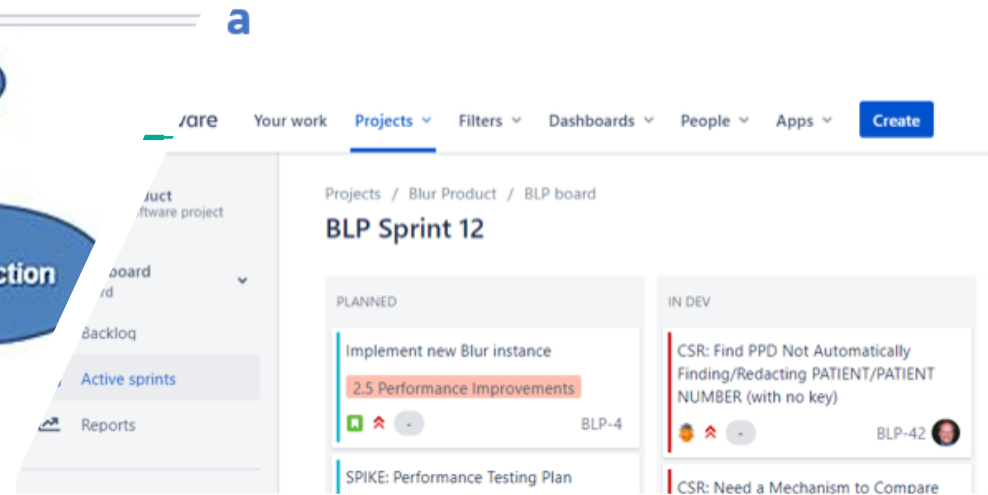
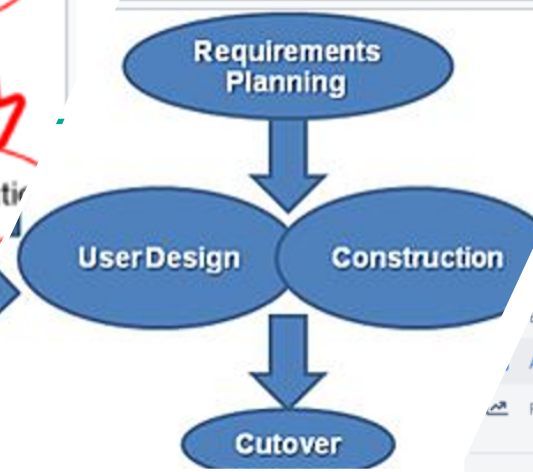
Testing

Development

QA

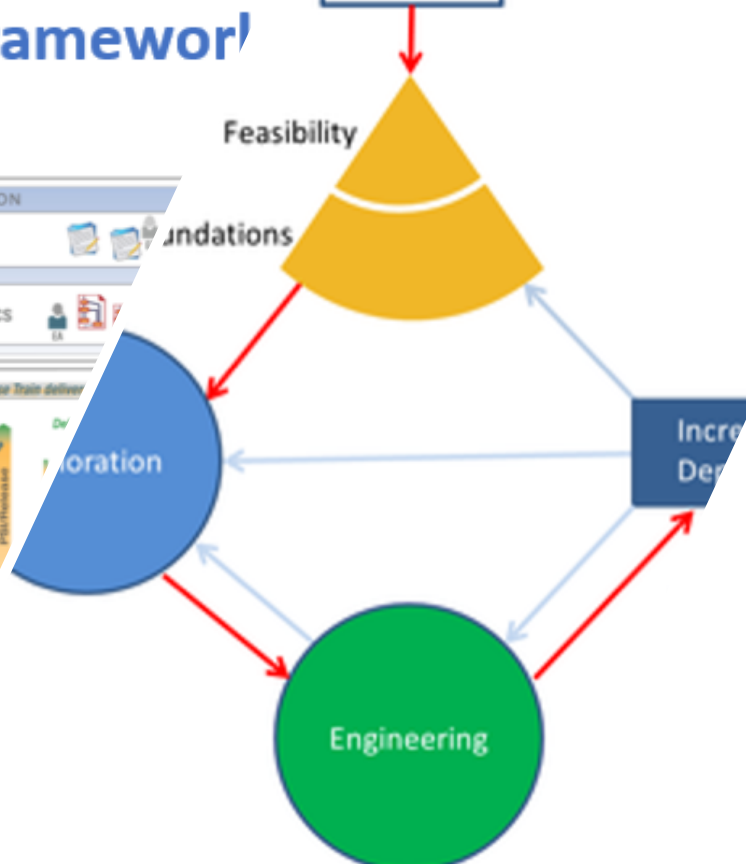
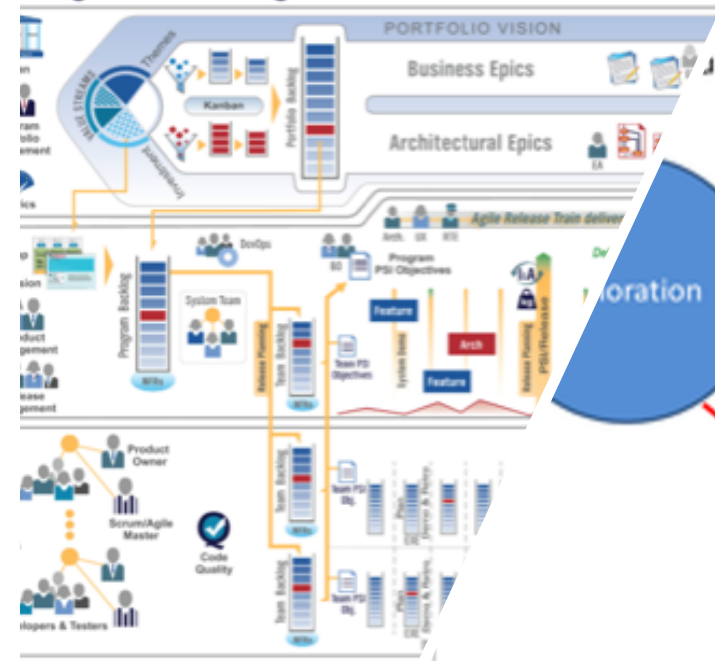
Summary

Agile



Model of the systems develop.

Fe : Scaled Agile Framework



Agile principles are not new – there is a plethora of agile methodologies.

Validation and Agile have not been good housemates up till now.

What does being agile mean?

Agile is about addressing risk..

1. Risk of failing to turn time/money into customer value
 - Build the wrong thing – not what the customer needs
 - OR
 - Build the thing wrong – doesn't work or works very badly
2. Human shortfalls are at the root cause of failure
 - Planning failures (including things happening that we did not anticipate)
 - Cognitive biases in decision making
 - Poor communication
 - Slips or lapses in execution
 - Missing skills and competencies
 - and so on
3. **Agile compensates for human fallibility with an integrated cross functional team approach**

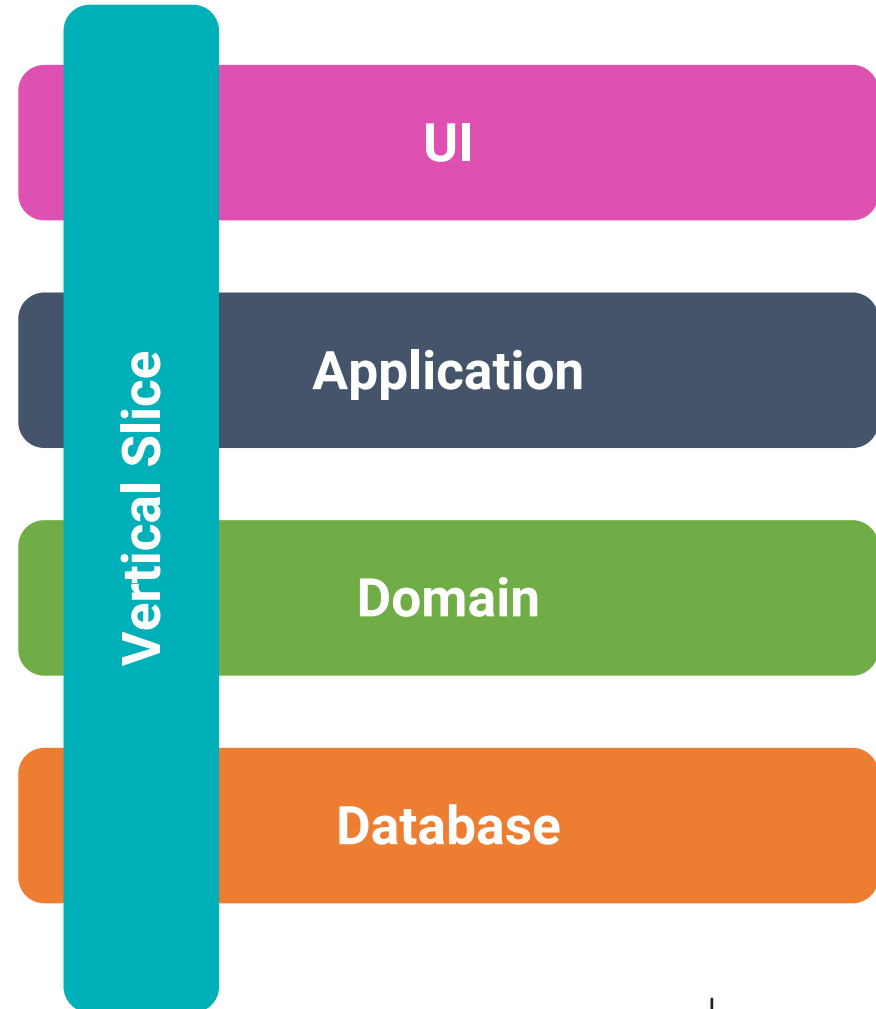
There is no failure in agile

Breaking deliverables up into small slices of functionality which deliver value to the customer ensures that the end product will meet the customer need

- Agile philosophy is “**Fail quickly, fail cheaply, learn and move on**”
- Delivering small – means failing small so we can learn and move on
- Failure always brings enlightenment of what not to do
- A continual feedback loop with the customer is key to delivering optimized quality software

Slicing

- Smaller slices are easier to understand
- Estimates are more accurate
- Focuses in on the “must have” functionality
- Faster feedback loop
- More frequent small wins
- Delivery of true value to the business from each slice



Agile-Fall

Agile-fall is more than Wagile

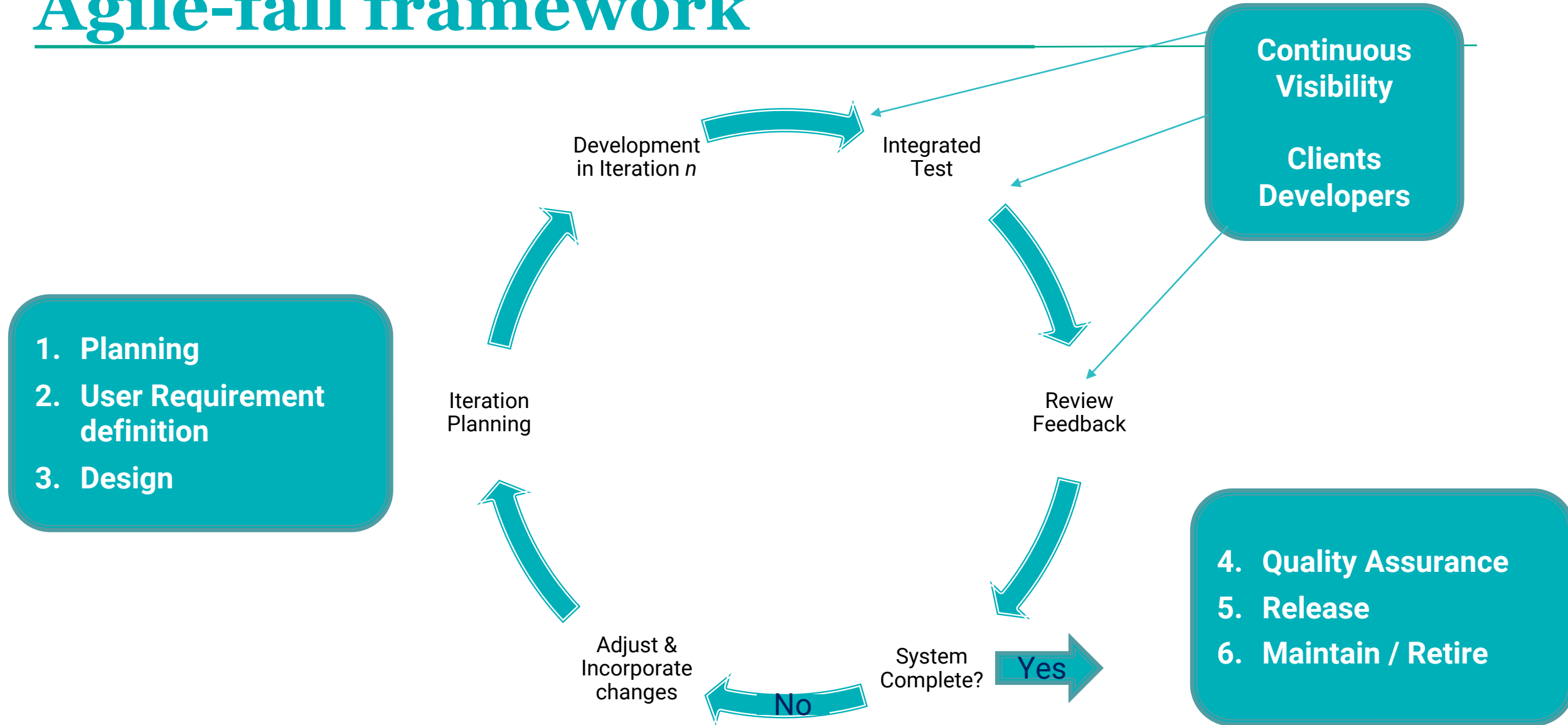
- “Wagile software development is a group of software development methodologies that result from slipping from agile back into waterfall, doing a lot of short waterfalls and thinking it is agile, Waterfall model masquerading as Agile software development, etc” www.techwell.com
- Agile-fall embraces Computer System Validation requirements within an Agile flow.
- Why did we create this way of working?
 - Waterfall alone gives you what you can describe not necessarily what your users need
 - Agile enables User Experience design-led development
 - **GxP requirements** are realised using Agile workflow and iterations
 - Created and maintained at an **appropriate regularity**
 - Agile-fall gives you what your users need.

Timelines, planning & tracking

Iterations		Phase 1																														
		Iteration 1								Iteration 2								Iteration 3														
Sprint	0	Sprint 1			Sprint 2			Sprint 3				Sprint 4			Sprint 5			Sprint 6				Sprint 7			Sprint 8			Sprint 9				
Sprint Week	-1	1	2	3	1	2	3	1	2	3	4	1	2	3	1	2	3	1	2	3	4	1	2	3	1	2	3	1	2	3	4	
Prototype Acceptance Plan	v1										v2										v3											
Prototype Status Report					v1									v2										v3								
Prototype Review Report											v1										v2										V3	
Validation deliverables											d1										d2									v3		

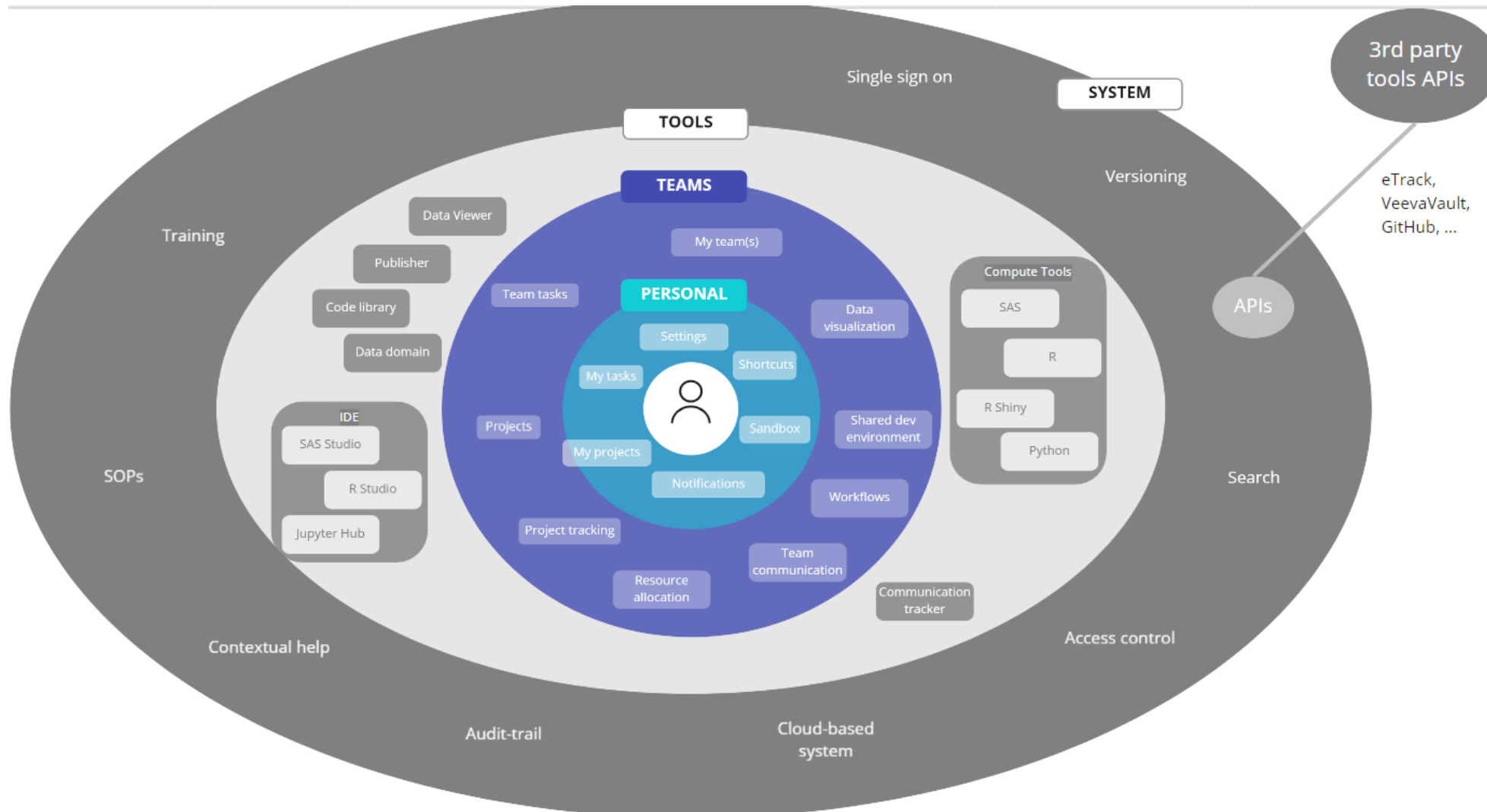
Roadmap		Iteration 1	Iteration 2	Iteration 3
EpicGroup		Iteration 1	Iteration 2	Iteration 3
Access_data_files		User stories : As a, I want to ...	User stories : As a, I want to ...	User stories : As a, I want to ...
Printing		User stories : As a, I want to ...	User stories : As a, I want to ...	User stories : As a, I want to ...

Agile-fall framework



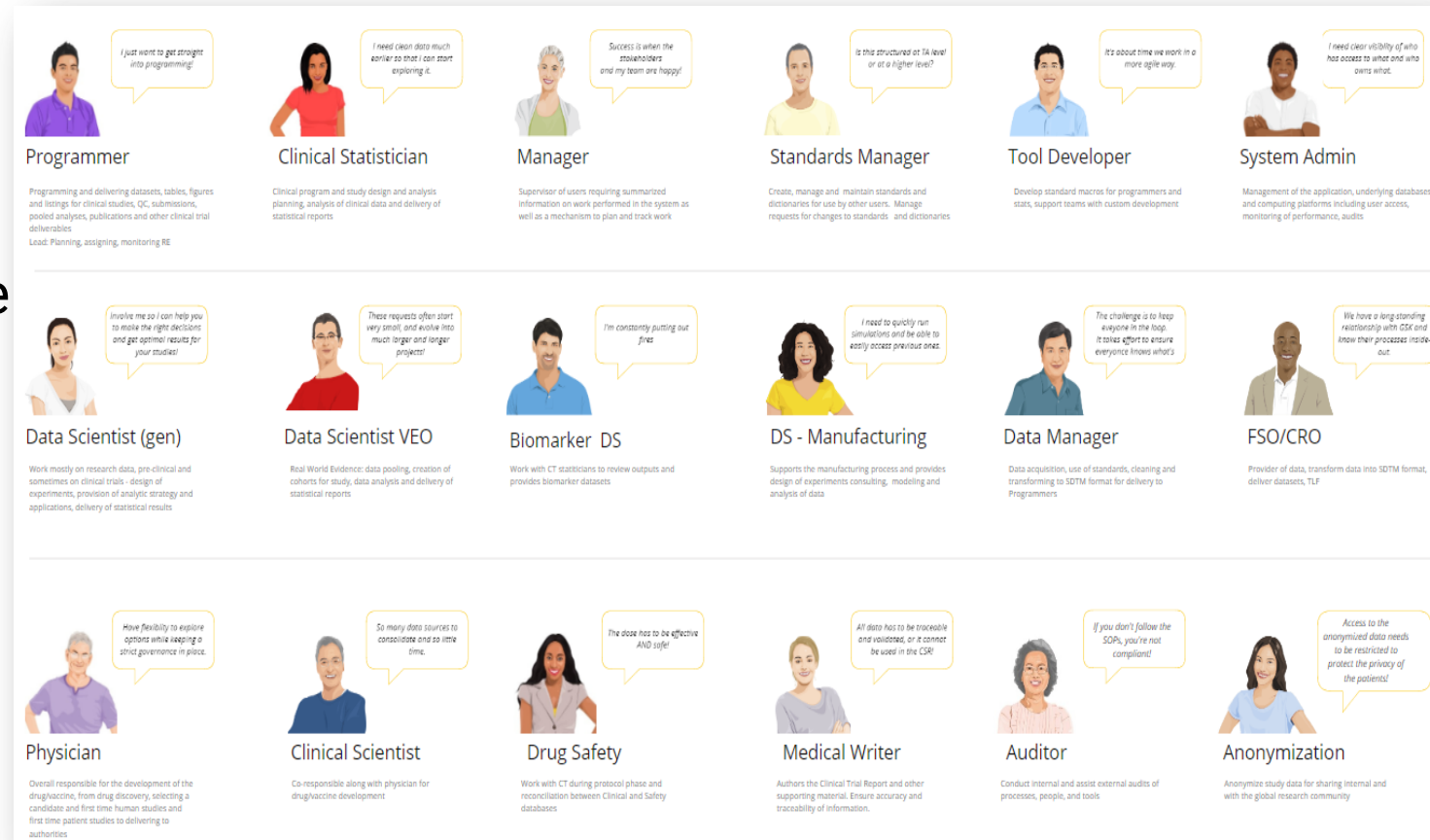
Requirements

Holistic view of application and concepts



UX led design and requirements definition

- Agile-fall starts with User Experience (UX) design led requirements.
- Identifies personas - what is important for them.
- Captures core functionality requirements
- Enables the creation of a clickable wireframe
- Confirmation of true business requirements
- Demonstrate preferred ways of working
- Identify the critical success factors
- Equals - Business Requirements for validation.





*I just want to get
straight into programming*

Programming and delivering datasets, tables, figures and listings for clinical studies, QC, submissions, pooled analyses, publications and other clinical trial deliverables.

Lead role : Planning, assigning, monitoring
RE

Programmer

Usability testing



What :

- User is given clickable wireframes
- UX designer observes the user's behaviour and listen for feedback

Why :

- Identify problems in the design of the product or service
- Uncover opportunities to improve
- Learn about the target user's behaviour and preferences

Including UI wireframes in Jira tickets

As a user I can see a "flag" of the program status of my programs



Description

Normal text ▾ | **B** *I* ... | **A** ▾ | | + ▾

This story is Done when:

The SCE / UI shows a visible indication of the program status

Comment:

e.g. there is a column called 'Status' that shows the status value

✓ ASSIGN ✓ VERIFY ✓ LOCK <--VIEW PROGRAMMING PLAN IN PROGRESS-->

☐	ID ↑	Planned Items	Title	Type	Level
☐	01	 tsfae12a.rtf	Adverse Events lead...	TLF	● Low ▾
☐	02	 tsfae12b.rtf	Adverse Events lead...	TLF	● Low ▾
☐	03	 tsfae12b.rtf	Adverse Events lead...	TLF	● Low ▾
☐	04	 aesumm.sas	Macro to support ...	Macro	N/A ▾
☐	05	 gefos02_cco1.lst	Adverse events ana..	Dataset	● Low ▾
☐	06	 teftpp10.rtf	Synthesis demogra..	TLF	● Low ▾

The icon in the UI changes, depending on the status. See lifecycle diagram for different icons

Usability testing makes sure what is described is what is really needed before coding begins, it also fleshes out the features that need to be described

Spikes

Proof of Concept (PoC)

- When to use PoC

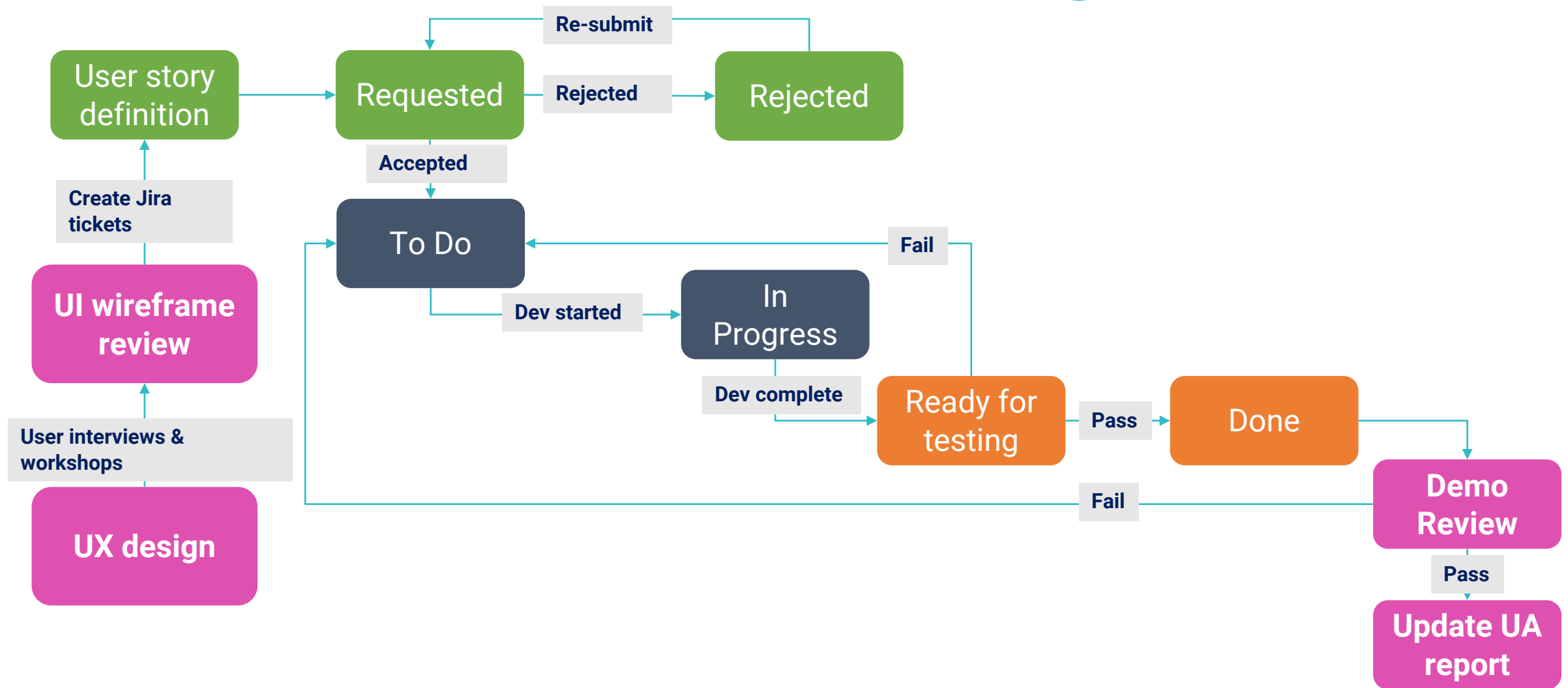
- Selecting between different approaches
- Considering using a novel approach
- Uncertain if the idea will work as needed
- Turning an idea into reality
- Surface potential technical or logistical issues

- How

- Describe what is needed
- Define success
- Map pain points to solutions and get feedback
- Prototype your solution and test
- Create an MVP (Minimum Viable Product)
- Confirm the roadmap steps for full product

Flow

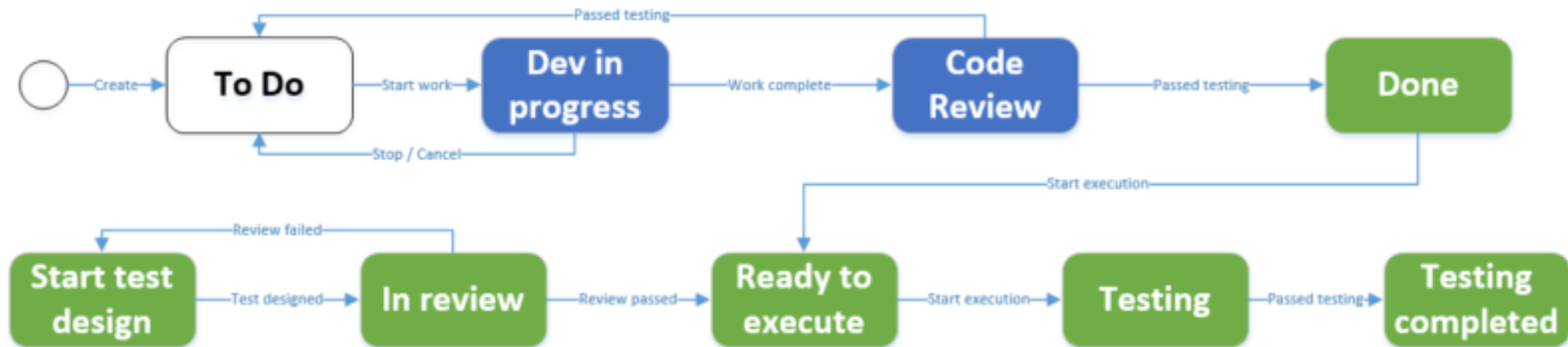
End to end workflow – using Jira



Jira Objects and states

All statuses can transition to "To Do"

All statuses can transition to "Done"



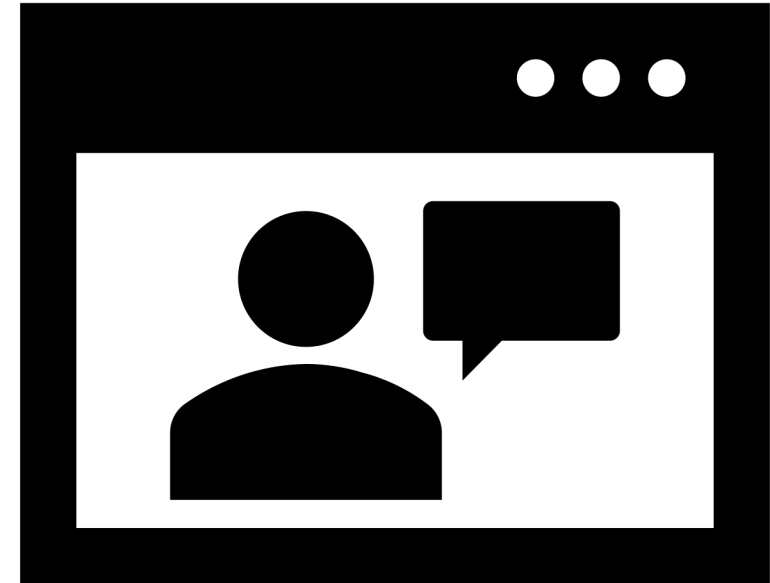
Testing

Examples of testing artefacts

- PoC outcomes
- Business process assets
- Demo scripts
- UI designs
- Annotated wireframes
- Mapping Jira Stories to demo script steps
- Mapping Jira Stories to Functional Tests - x-ray for Jira

Testing review and acceptance

- Jira ticket has the definition of done
- Testers record their test execution
- Reviewers can see errors
- Jira ticket updated with fix
- Fix implemented
- Tester re-tests and confirms that issue is fixed



Testing review and acceptance

test-sstar.d-wise.com/workspace/compounds/test_compound/studies/test_study/dbrs/test_dbr/reportingEfforts/test_re/plan

Workspace > test_compound > test_study > test_dbr > test_re

test_re

PLAN & REVIEW PROMOTE & DEMOTE TEST SAS STUDIO TEST FILE COPY

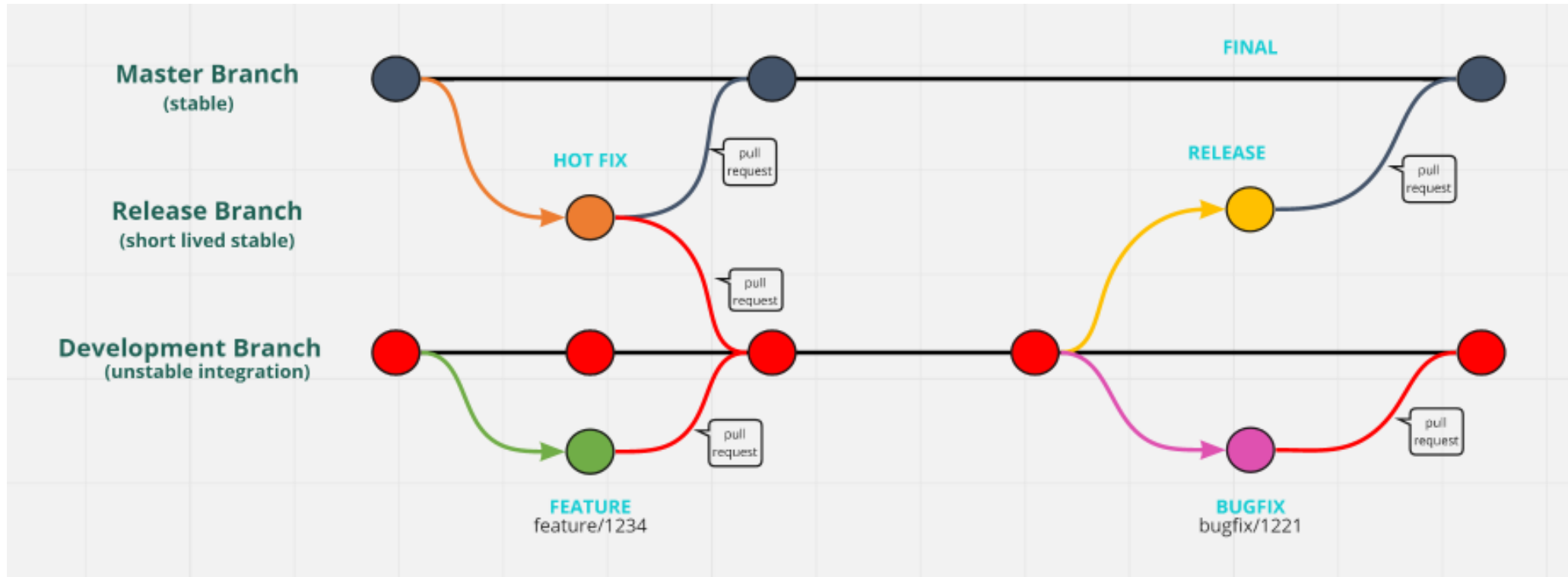
✓ ASSIGN ✓ VERIFY ✓ LOCK ↑ UPLOAD PLAN

☐	Id	Planned Item	Title	Type	Level	Code	Programmer	Reviewer	Date last run	Status
	5	aeprog2.sas (1.0.0)	aeprog2.sas output	Program	● Low	aeprog2.sas				Not ready
	6	demog.sas (1.0.0)	demog.sas output	Program	● Low	demog.sas				Not ready
	7	demog2.sas (1.0.0)	demog2.sas output	Program	● Low	demog2.sas				Not ready
	8	pops.sas (1.0.0)	pops.sas output	Program	● Low	pops.sas				Not ready
	9	ulti....sas (1.0.0)	ultimate_program.sa...	Program	● Low	ultimate_program.sas				Not ready
	2	aelist.sas (1.0.0)	aelist.sas output	Program	● Low	aelist.sas			Mon Oct 14 2019	Not ready
☒	1	adae.sas (1.0.0)	adae.sas output	Program	● Low	adae.sas			Mon Oct 14 2019	Production
	3	aelist2.sas (1.0.0)	aelist2.sas output	Program	● Low	aelist2.sas			Mon Oct 14 2019	NO_INDEPENDENT_...
	4	aeprog1.sas (1.0.0)	aeprog1.sas output	Program	● Low	aeprog1.sas				NO_INDEPENDENT_...

Development

CICD Branching

- Weekly deployment
- Incorporating automated testing



QA

CSV documentation

- Definition of done for each cycle of delivery (Iteration)
- Success criteria
 - *“For this iteration the user will be able to*
 - ...
 - ...”
 - *“In this demo the user will be able to see*
 - ...
 - ...”
- Deliver validation documents throughout the project
- End of iteration deliver code which can be executed
- Deliver testing assets which client can use to confirm the requirements are met.

Summary

In Conclusion

- Agile principles, with a Scrum-like framework, have great value
 - Delivering working software, regularly
- Agile-Fall enhances Agile/Scrum, making them suited for our regulated industry
 - Delivering validateable working software, regularly
- Gathering feedback early and often
- Deploying when appropriate

Thank you for watching

