

SHORT INTRODUCTIONS

- Who are you?
- What do you do?
- What is your main learning objective?

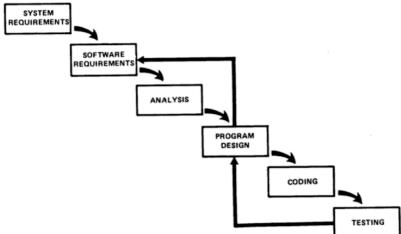
GROUND RULES

- Break times
- Restroom
- Start on time
- Phones off or on silent
- Laptops down
- No side conversations / one conversation at a time
- Criticize ideas, not people
- No hierarchy
- Respect

QUESTIONS?

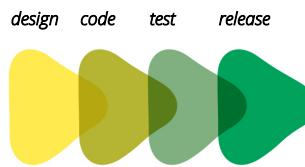
ThoughtWorks®

THE "WATERFALL" DIAGRAM

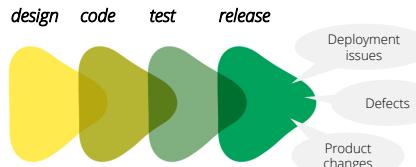


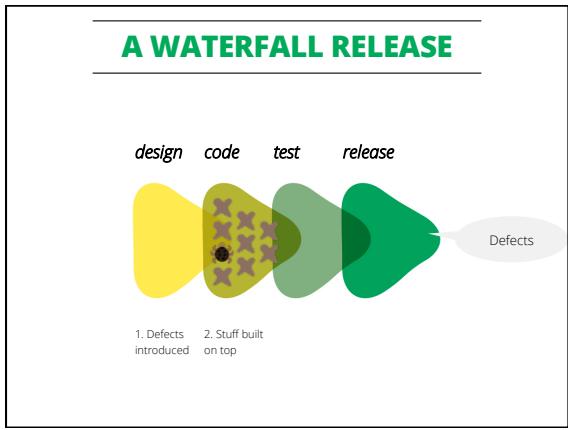
[1] Royce, Winston. "Managing the Development of Large Software Systems", Proceedings of IEEE WESCON26 (August); 1-9. 1970.
[2] Bell, Thomas F., and T. A. Thayer. "Software requirements: Are they really a problem?", Proceedings of the 2nd international conference on Software engineering, IEEE Computer Society Press, 1976.

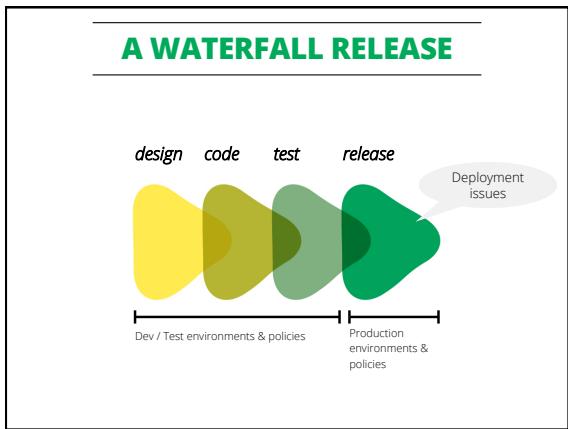
A WATERFALL RELEASE

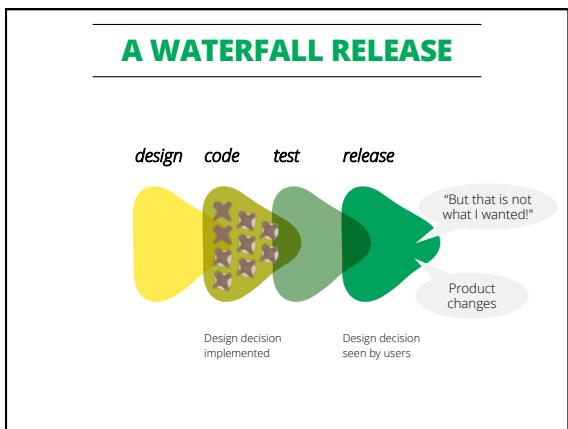


A WATERFALL RELEASE

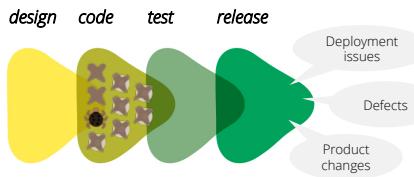








A WATERFALL RELEASE



ThoughtWorks®

AGILE SOFTWARE DEVELOPMENT

WHAT IS AGILE?

*A way of building software that is
all about planning for, and
assuming change*

MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

TWELVE PRINCIPLES OF AGILE SOFTWARE

1. Priority 1: Satisfy customer via early, continuous, valuable delivery
2. Welcome changing requirements, even late in development
3. Deliver working software frequently
4. Business people and developers must work together
5. Build projects around motivated individuals.
6. Face-to-face conversation is most effective way to communicate.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development.
9. Continuous attention to technical excellence and good design
10. Simplicity (art of maximizing the amount of work not done)
11. Emergent design and requirements from self-organizing teams
12. Regular reflection on how to become more effective

IT IS NOT EITHER-OR

- Scrum
- XP (eXtreme Programming)
- Lean-kanban
- Hybrid (e.g., "Scrumban")

ThoughtWorks®

WHAT AGILE IS NOT

AGILE IS NOT...

- A "silver bullet"
- One size fits all
- Go through the motions and mechanics, and then "we are agile"
- Ad-hoc with no plan

ThoughtWorks*

WHAT AGILE IS

▪ Incremental

Build software gradually
Show progress

da Vinci as an incremental painter



▪ Incremental

Build software gradually

Show progress

▪ Iterative

Multiple iterations and releases

da Vinci as an iterative painter

▪ Incremental

Build software gradually

Show progress

▪ Iterative

Multiple iterations and releases

▪ Adaptive

Incorporate feedback and changes to goals

Lessons learned

*da Vinci as an incremental
iterative
adaptive painter*



■ **Incremental**

Build software gradually
Show progress

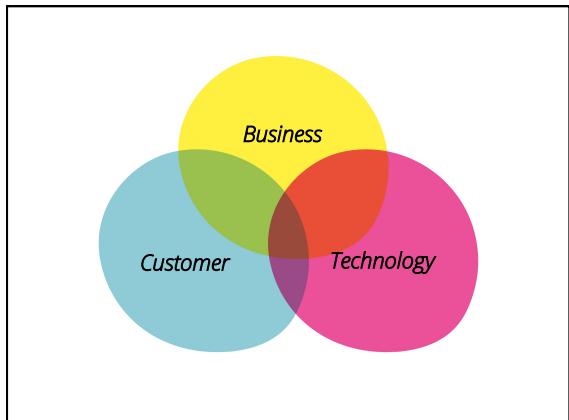
■ **Iterative**

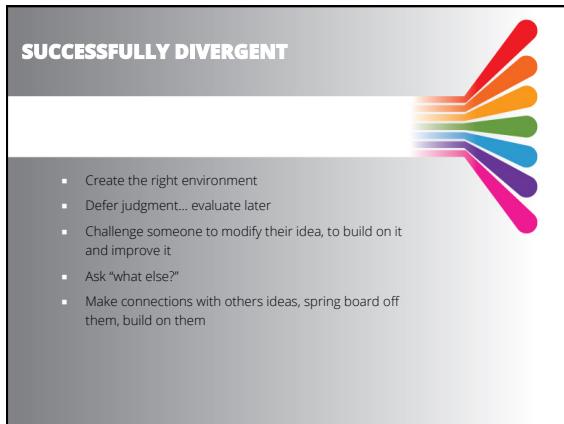
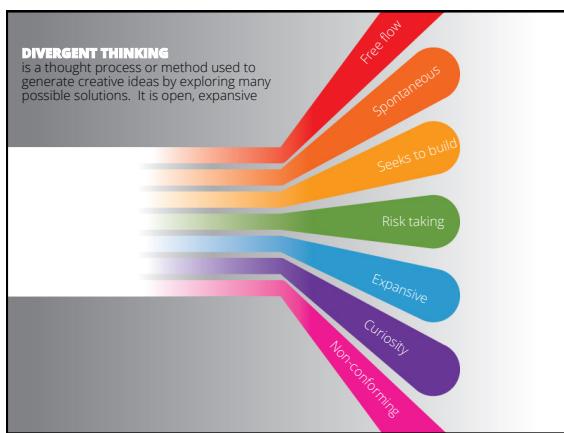
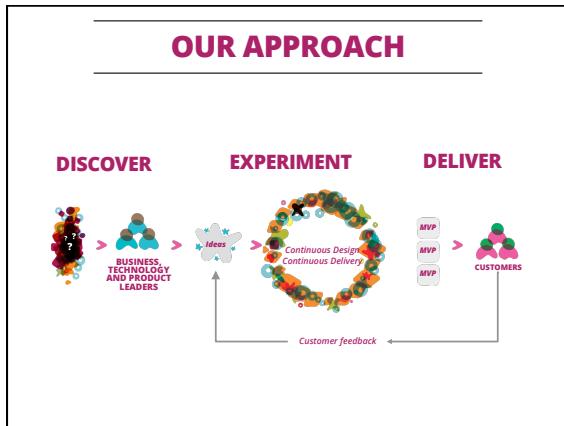
Multiple iterations and releases

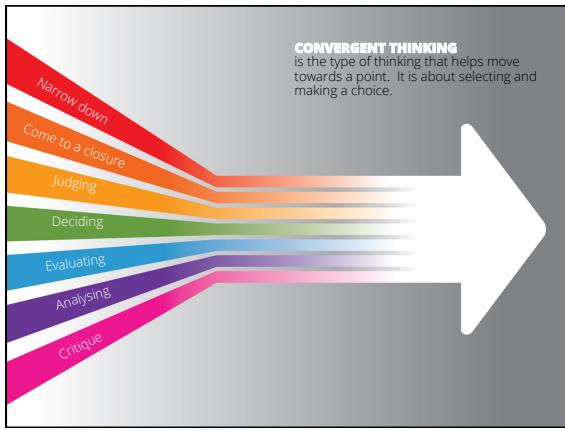
■ **Adaptive**

Incorporate feedback and changes to goals
Lessons learned

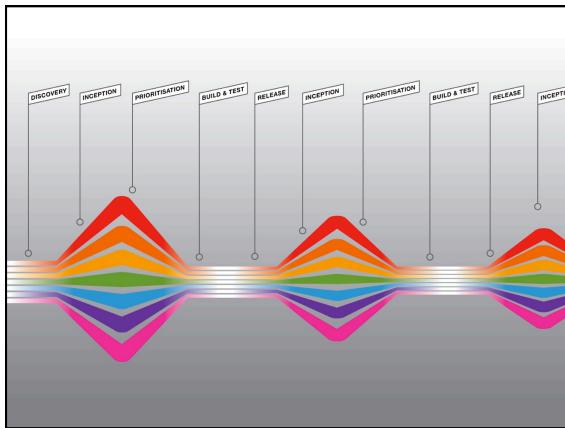
■ **Collaborative**





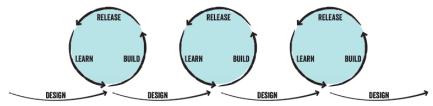




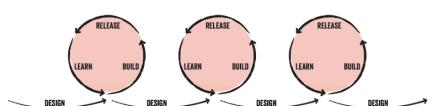


DESIGN THINKING AND AGILE

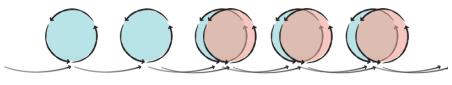
Design Thinking is the process of continually designing and learning from people (consumers, customers, patients, staff)



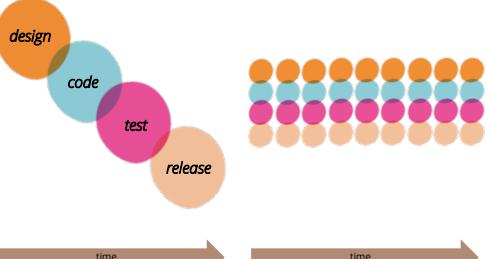
Agile Software Development is a process of rapid software design and build cycles which means constant releases to customers.



DESIGN THINKING AND AGILE



WATERFALL VS. AGILE APPROACH



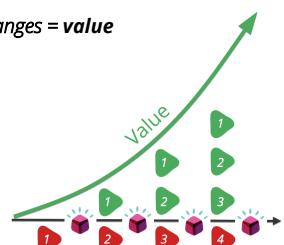
WATERFALL VS. AGILE APPROACH

unreleased changes = risk



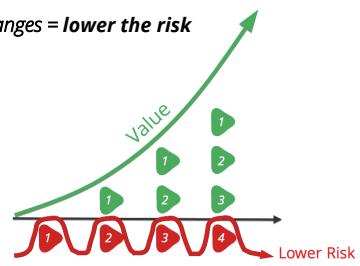
WATERFALL VS. AGILE APPROACH

released changes = value

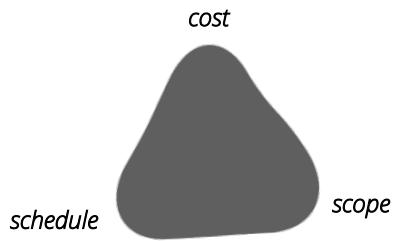


WATERFALL VS. AGILE APPROACH

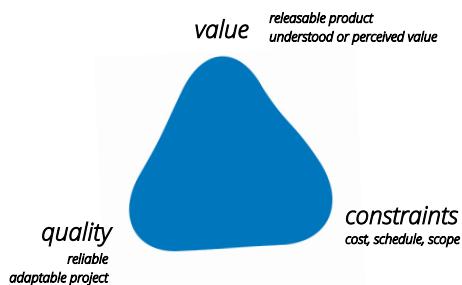
released changes = lower the risk



THE TRADITIONAL TRIANGLE



THE AGILE TRIANGLE

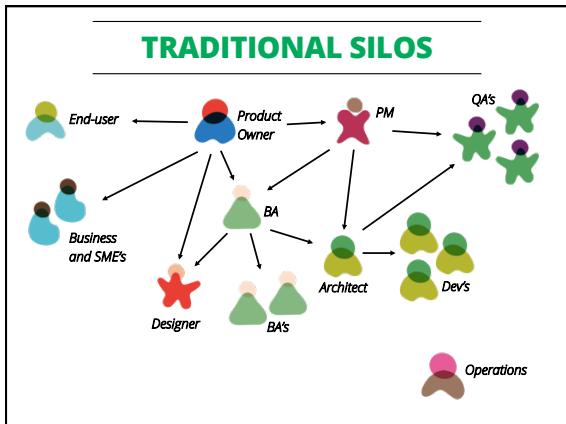


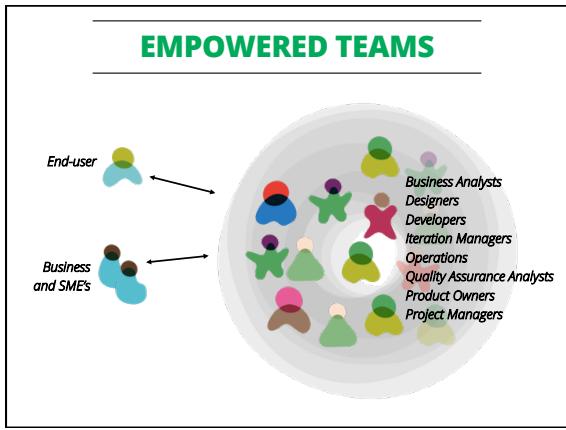
Both management and technical practices are important.

To be truly agile, the software itself needs to be agile.



- Representation across roles
- No specialized teams
- Typically co-located









USER STORIES

- A tool for **iterative** development
- Represents a **unit** of work that should be developed
- Help **track** that piece of functionality's lifecycle
- Placeholder for a **conversation**

WHY SHOULD I USE THEM?

- It is a piece of customer-visible functionality written in common language
Universally understood
- Ensure you only build things for a reason
Prevents waste
- Simple and flexible
Minimum overhead
- Proven way of gathering requirements on agile projects
Effective teams

ThoughtWorks®

As a **<role>**
I want to **<business goal>**
So that **<value>**

INVEST IN STORIES

- I** Independent *No overlap – order is ok!*
- N** Negotiable *No contract. Details can change.*
- V** Valuable *Incremental benefit to something.*
- E** Estimable *Relative size to other stories.*
- S** Small *Shouldn't be bigger than an iteration.*
- T** Testable *Should be able to tell when it is done.*

INVEST IN STORIES

- V** Valuable

Someone is benefiting from what we are building

Why are we building it?



INVEST IN STORIES

- T** Testable

Now we know what to do and know when we will be done



INVEST IN STORIES

S Small

Just enough to get
feedback and avoid
waste



INVEST IN STORIES

I Independent

Should be able to realize the
value of each single story.
Distance from the airport
can be in another story.



INVEST IN STORIES

N Negotiable



INVEST IN STORIES

E Estimable



WHAT IS IN A USER STORY

- "As a ... I want ... So that ..."
- Acceptance Criteria
- Prototype
- User Interface design
- Other text/images/content to provide context

ACCEPTANCE CRITERIA

- Tell you the story is finished Acceptance tests tell you the system is working
- A story can have one or multiple AC's
- It informs the criteria to the tests that will be implemented or executed
- Common format:
Give <context>
When <event>
Then <outcome>

ACCEPTANCE CRITERIA

As an Internet Banking customer I want to see the list of my accounts so that I can choose to see more details of a particular account

Alternate path

Alternate path

Bad path

Given the customer has one transaction account and one credit account
When they have completed logging in
Then the screen should show the names and numbers of the two accounts sorted in account number order

Given the customer has just one transaction account
When they have completed logging in
Then the screen should show the name and number of the account

Given the customer has no accounts
When they have completed logging in
Then the screen should show a message stating that no accounts are available

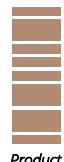
Given the customer has more than 20 accounts
When they have completed logging in
Then the screen should show the first 20 accounts (in account number order) only

Given the customer has some accounts
And they have completed logging in
When the system cannot retrieve the account details
Then the screen should show an error message with associated code and details to contact for support

WHAT IS NOT IN A USER STORY

- Unit and Integration tests
- Functional/Acceptance tests
- Documentation
- Specifications

STORY DETAILS AT THE RIGHT TIME



Product backlog



Release backlog



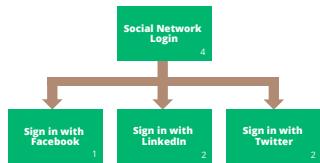
Iteration backlog



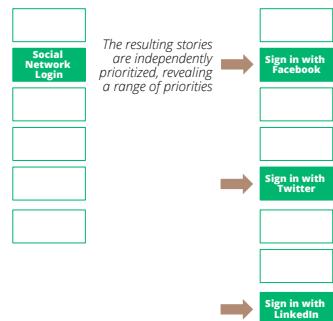
STORY BREAKDOWN

- Identify opportunities to split individual stories into one or more smaller independent stories, in order to:
 - Make stories an appropriate size for delivery
eg. *small enough that they are easily understandable, estimable and will give a good indication of progress*
 - Identify differences in priority
to ensure only the highest priority work is completed first

SPLITTING BY FEATURE



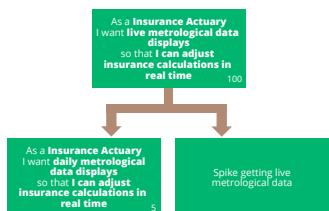
SPLITTING BY PRIORITY



SPLITTING BY VALUE

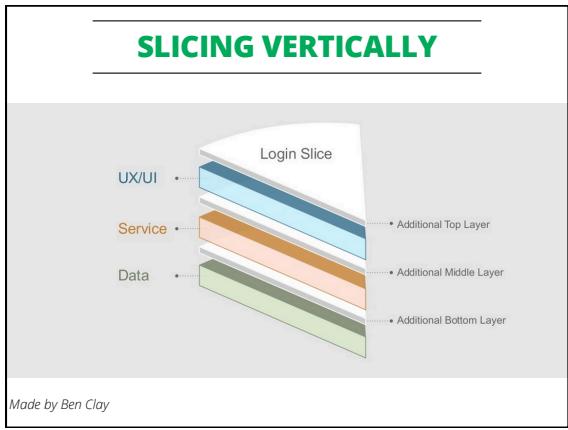


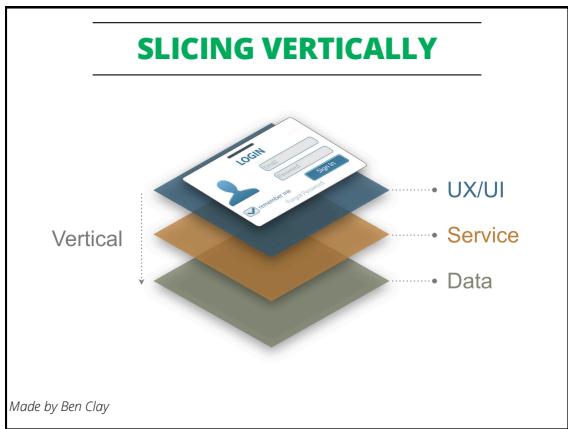
SPLITTING BY RISK



SPLITTING DEPENDENCIES









PERSONAS

- Concrete characterization of a user group
- Archetype / representation / profile
- Description of end users based on research
- Reference document
- It can be used a synthesis of different profiles

HOW TO DEVELOP PERSONAS

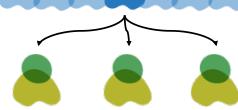
1. Conduct research
2. Identify and cluster the behavioral variables
3. Identify trends
4. Create a persona for each of the trends
5. Create scenarios and tell stories
6. Create personas documentation

HOW TO DEVELOP PERSONAS

Identify roles and goals



Distribute behaviorally distinct traits



Assign attributes and create the personas based on research



ROLES VS. PERSONAS

- Personas can easily become a reflection of roles within a system, usually a result of not digging deep enough to identify the different behavior patterns within each role
 - Personas should be able to identify different behaviors of roles
 - Danger of role-based personas is that there is often a temptation to create them based existing team knowledge of the system

QUESTIONS?

ThoughtWorks®



WHY ESTIMATE?

- Provides a measure of progress
- Helps with prioritization
- Estimation discussions are very helpful for defining an approach to delivering an individual story
- BUT:
 - Time consuming for large numbers of stories
 - Not a measure of time

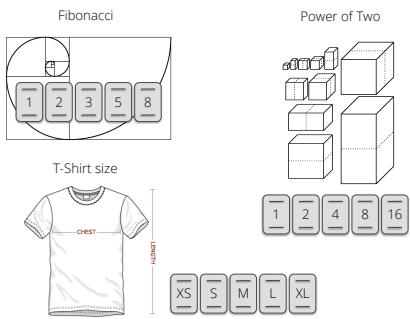
WHAT TO ESTIMATE

- Don't try and estimate all the stories in a project
- Have a prioritized backlog and estimate highest priority first
- Ensure there are estimated stories for the next iteration
- Stories that are likely to be difficult

WHAT TO ESTIMATE

- A common view of what needs to be complete for a story to be counted as done and to include its story points in velocity
- Commonly includes: functional code completed, tests passed, acceptance test passed, and deployed to production environment
- Definition should be discussed and agreed as part of inception
- Should define a situation where the team will not have to work on the story again

HOW TO ESTIMATE



GOOD PRACTICES ON ESTIMATION

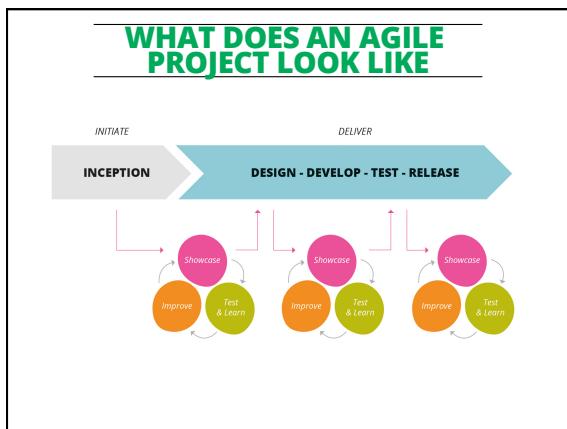
- A story should be deliverable from start to finish within an iteration. If not split the story
- The smallest story should represent a single story point
- Those involved in delivery should be part of estimation
- Participants should discuss the story until they all agree on an estimate
- There should be sufficient understanding of how the story will be delivered
- Have spikes when technical uncertainty is too high
 - Solve hard technical problems through time-boxed and real experiments without tampering with production code

QUESTIONS?

ThoughtWorks®



WHAT DOES AN AGILE PROJECT LOOK LIKE?



WHAT IS AN INCEPTION?



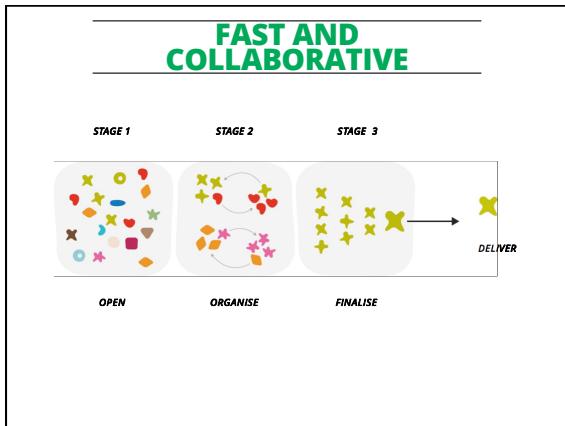
HOW INCEPTIONS FEEL

- Highly collaborative and inclusive
- Time-boxed and rapid, focused on doing 'just enough'
- Feedback-driven and highly adaptive
- Is not upfront analysis
- Highly visual and workshop oriented to help evolve a vision for the project.

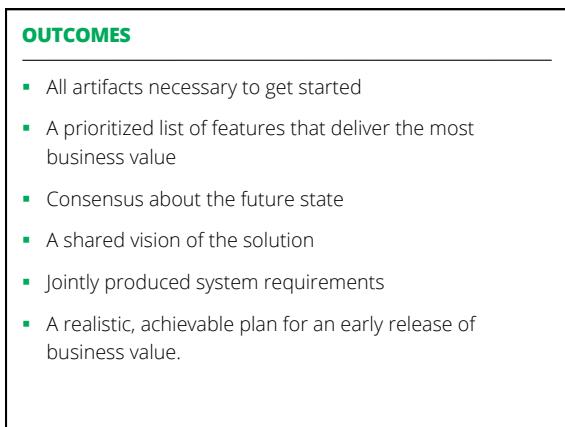


INCEPTION OBJECTIVES

Build a shared understanding of project vision and goals	Evaluate high-level scope and core processes
Evaluate key risks, issues and constraints in delivering to the end goal	Establish technical and testing approach









CREATE A SHARED VISION

UNDERSTAND BUSINESS VISION

(Note to reviewers - a new, crafted business model canvas will replace this one.)



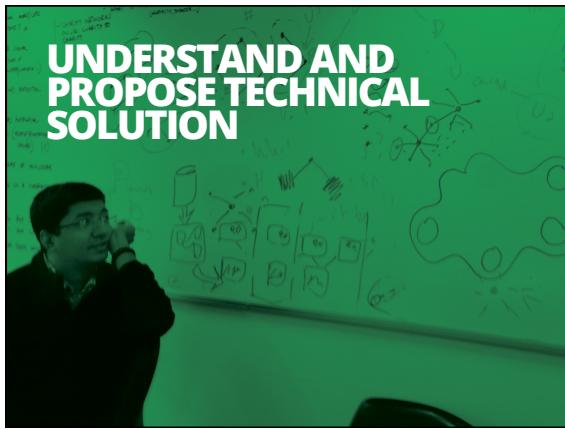
USER EXPERIENCE DESIGN

- Discovery
 - User Personas
- Concept
 - User Journey
 - Information Architecture
 - Collaborative Design
 - Wireframes
- Visual Design

A photograph of a whiteboard featuring a large hand-drawn cloud with the text "WE ❤️ JHB" inside. Numerous colorful sticky notes are pinned around the cloud, some with handwritten text like "Globe Campus", "JHB", and "Johannesburg".

COLLABORATIVE DESIGN APPROACH

A wireframe diagram titled "BA/DESIGN/BUILD/QA". It includes sections for "This is Responsive", "WE NEED TO CONSIDER ALL 3 PLATFORMS", and three categories: "① SMARTPHONE", "② TABLET", and "③ DESKTOP". Each category shows a mobile device icon and a corresponding wireframe layout. A legend at the bottom left indicates "MOBILE UI" and "LAYOUT CONTENT".



UNDERSTAND AND PROPOSE TECHNICAL SOLUTION

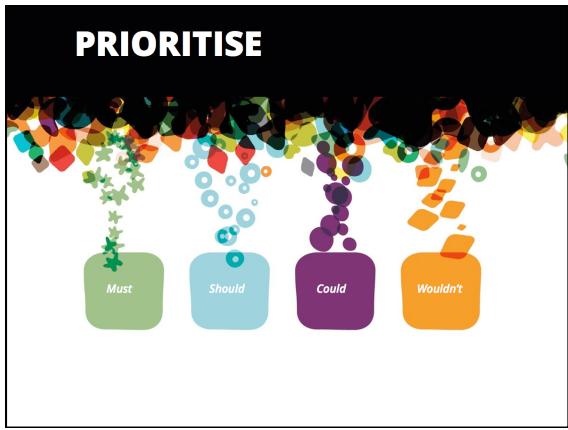
TECHNICAL APPROACH

- Technical visioning
- Architecture
- Proposed testing approach
- Addressing Non Functional Requirements (NFRs)

A collection of icons used to represent different technical concepts and tools.

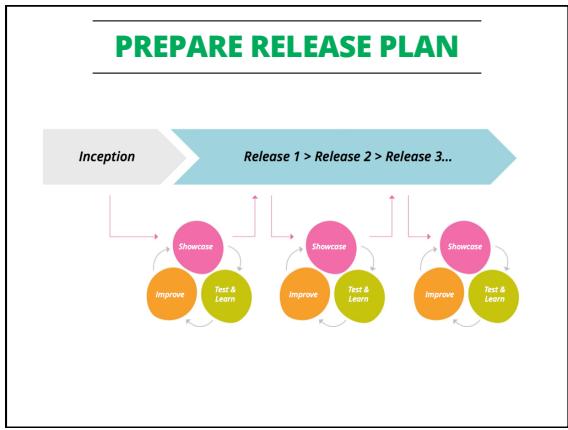


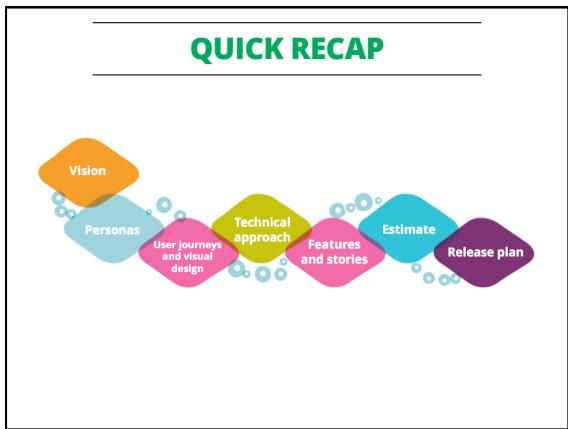
IDENTIFY & DISCUSS FEATURES AND STORIES

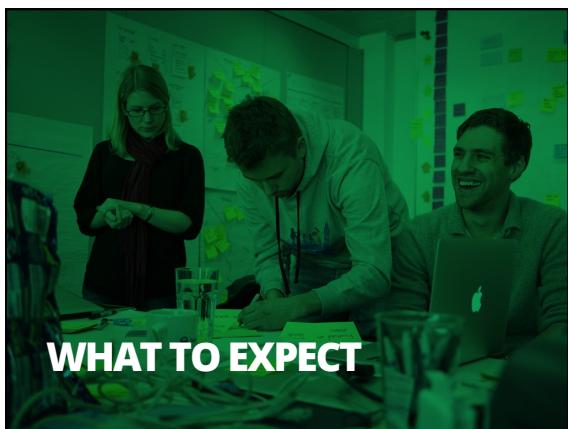










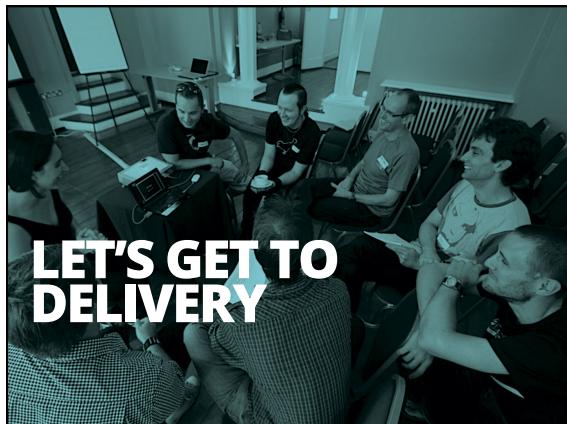


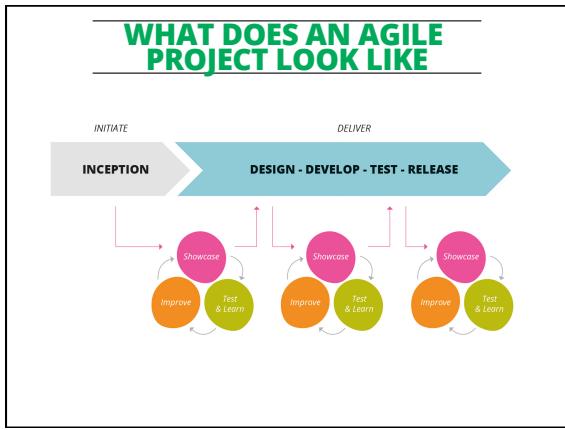
INCEPTION DELIVERABLES

- Business Vision and Project Objectives
- Prioritised High Level Requirements (Stories & NFRs)
- Technical Approach
- Visual Design Mock-ups
- Release Plan
- RAIDs & Communication Plan
- Next Steps

QUESTIONS?

ThoughtWorks®





ThoughtWorks®

As a **customer**
I want to **be able to sign in using my Facebook account**
so that **I don't need to remember *another* username and password**

INITIAL REQUIREMENTS

- project inception
- unfinished from previous projects
- ad-hoc conversations
- analysis activities
- non-functional reqs

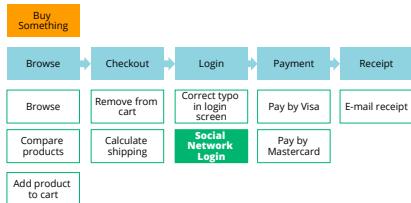
Initial project requirements come from a wide variety of sources. Perhaps the most common source is inception workshops that involve the core delivery team and stakeholders

Requirements will often vary greatly in level of details ... from loosely-defined **epics** through to **stories**, down to very well-defined **bugs**

PRODUCT BACKLOG

- These initial requirements form our Product Backlog
- At the simplest, just a prioritized/ranked list of requirements
- The Product Backlog is owned and maintained by the Product Owners with assistance of Analysts over the lifespan of the project
- Product Backlogs evolve over the course of the project (they are not fixed after being defined up front!)

STORY MAP



NON-FUNCTIONAL REQUIREMENTS

some examples:

Auditability	Authentication	Authorization
Availability	Help	Localization
Performance	Portability	Regulatory
Scalability	Security	Usability

NFR: WHAT TO DO WITH THEM?

- Discover which ones matter
- New stories (a mind map may help)
- Acceptance criteria
 - in some stories
 - in all stories
- Constraints
 - "Constraint cards"
- Review the list of 'ilities' in a facilitated session

ANALYSIS AND UX ELABORATION

- Analysts, UXers and Product Owners progressively elaborate Stories for development - high priority first!
- Stories are picked up for elaboration far enough ahead of when they are expected to be implemented to ensure detail is ready, but not so far in advance that it's hard to change course
- Elaboration activities include:
 - Story breakdown
 - Finishing User Stories, Acceptance Criteria and Narratives
 - Prioritization
 - Estimation

ANALYSIS AND UX ELABORATION

Sign in with Facebook

Narrative
As a customer I want to be able to sign in using my Facebook account so that I don't need to remember another* username and password

Acceptance Criteria

- Should capture Facebook name, email, country and birthday
- Should be available anywhere in site

ESTIMATION AND PRIORITIZATION

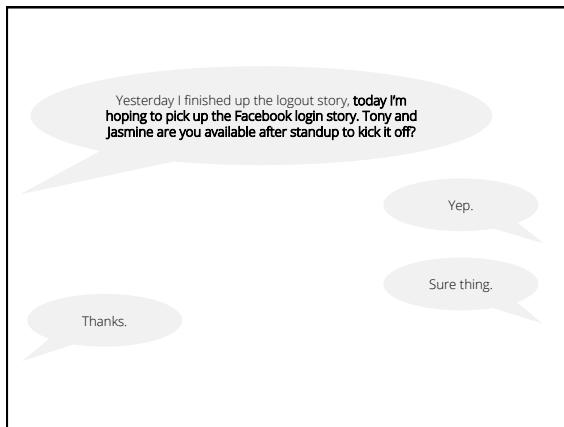
- Periodically (every couple of iterations) prioritization should be revisited with Product Owners to see if any priorities have shifted
- Similarly if elaboration identifies substantial changes in scope or potential effort, estimates should be revisited with the Core Team
- This activity of revisiting the backlog for estimation, prioritization and add/remove stories based on recent learning is sometimes known as "backlog grooming"

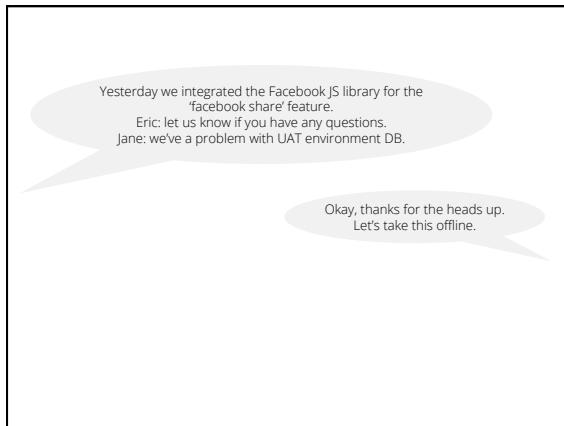
ITERATION PLANNING MEETING

- Agenda
 - Confirm priorities
 - Calculate expected team capacity
 - Overview of high priority stories
 - Development team indicates which stories they can commit to that iteration

STAND-UP MEETING

- Team meets for 15 minutes daily to provide a status update among team members:
 - What did I accomplish yesterday?
 - What will I do today?
 - What obstacles are impeding my progress?
- Goal is to highlight potential challenges and identify ownership to coordinate efforts to resolve difficult or time-consuming issues
- It is not a goal to resolve those issues in this 15 minute meeting







STORY KICK-OFF

- Product Owner or Analyst introduces the rest of the team to the story and answers any questions that come up at this point (may tweak story details)
- Developers describe how they intend to approach the solution
- Testers describe how they intend to approach the testing
- Designers talk about how the UX should be like
- Meeting ends when everybody is comfortable (5-30min)

DEVELOPMENT

- Developers work alongside UX, Analyst, Product Owners and Testers as required. Here conversations fill in any remaining details and any options (such as different solutions) are evaluated
- Testers may also prepare test cases and annotate on the story
 - Test cases are not acceptance criteria!
 - Test cases will include things not covered in acceptance criteria
- Development continues until developers are confident the acceptance criteria are met, and an environment can be provided for testing
- Do regular desk checks with the rest of the team

HANOVER TO TEST

- Developer hands over story to Tester
 - If available, analyst, designers and product owners might be included
- Demonstrates functionality against the acceptance criteria in the test environment (if can't be demonstrated... it isn't ready!) (5-15 min)

TESTING

- Tester finalizes and executes test cases for the story. Will seek additional clarification from Product Owners, Analysts, Designers or Developers
- Tester will do further exploratory testing to test the edge cases of the solution
- Any issues identified that would prevent the story from meeting acceptance criteria should be properly handled

STORY ACCEPTANCE

- Analyst or Tester demonstrates the solution to Product Owners to answer two questions:
- Does this solution meet the acceptance criteria (as agreed)... are we done on this story?
- Based on now seeing the working software, does this inspire other changes to this feature?
All new ideas → new user stories (unless the delivery team agrees) Product Owners prioritize new stories in existing backlog
- Demonstration often involves walkthrough of Given...When... Then acceptance criteria, as well as some exploratory use by Product Owners
- It should not be the first time the Product Owners see the story implemented

SHOWCASE

- Product Owners and Team demonstrate the stories that have been accepted as 'done' during the iteration to the full set of users and stakeholders
ie. everyone with any stake in the project
- Feedback is gathered
All changes, ideas → new user stories (product owners prioritize)

RELEASE

- The earlier the product under development reaches the hands of the end-users the better – it provides an opportunity to measure and respond to real usage of the system while there is still time to adjust

RETROSPECTIVE

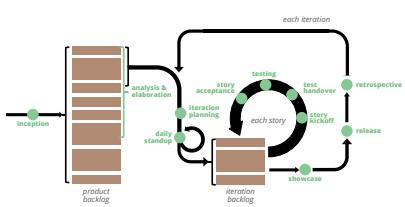
- Team brainstorms on sticky-notes:
 - What went **well** with the delivery process?
 - What went **not so well** with the delivery process?
 - What **ideas** do we have about things we can do differently?
 - What still **puzzles** us about the delivery process?
- Similar thoughts are combined into groups (sticky-notes placed together)
- Team votes on the groups to identify which they want to discuss most
- Team discusses top-voted groups, identifies and formulates actions: **who** will do **what** by **when**

RETROSPECTIVE PRIME DIRECTIVE

*Regardless of what we discover,
we understand and truly believe
that everyone did the best job
they could, given what they knew
at the time, their skills and
abilities, the resources available,
and the situation at hand.*

(Norm Kerth)

WRAPPING UP THE ITERATION

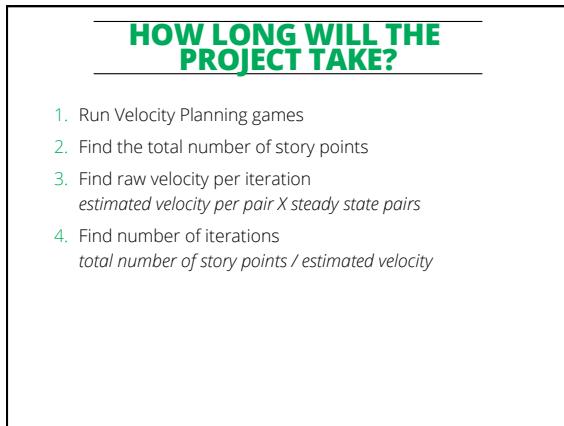
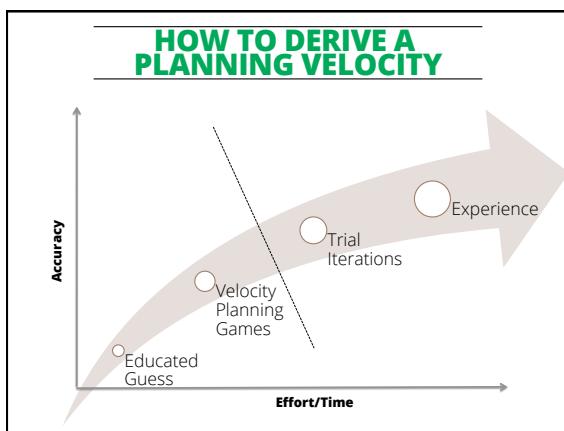


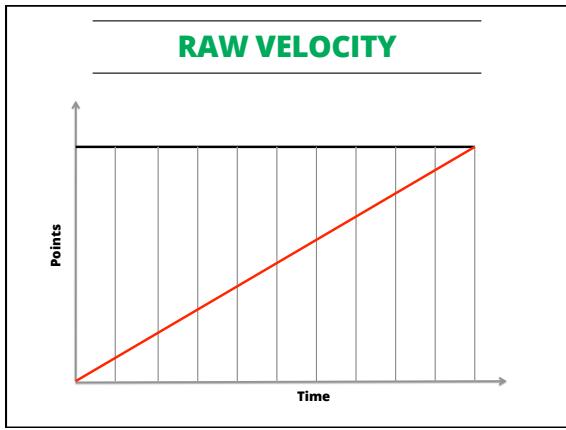
QUESTIONS?

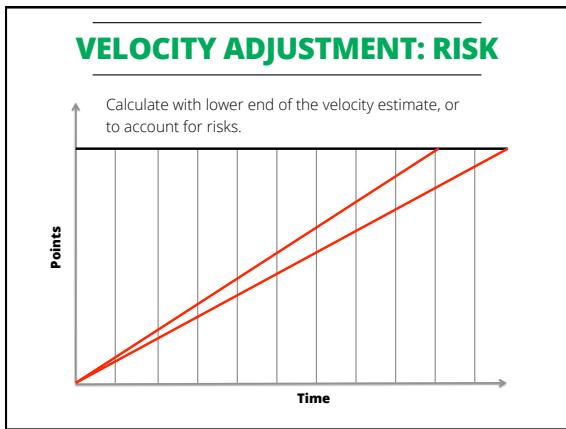
ThoughtWorks®

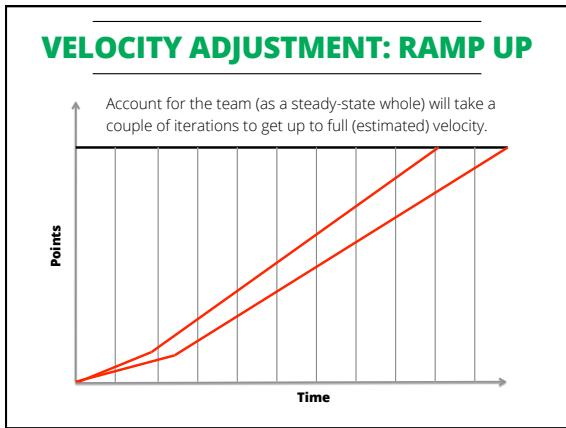
ThoughtWorks®

PLANNING AND TRACKING



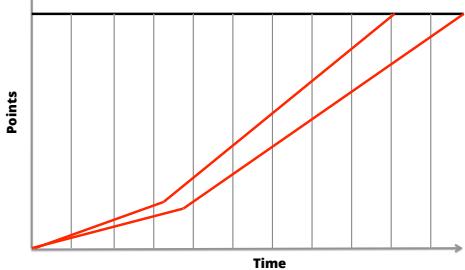




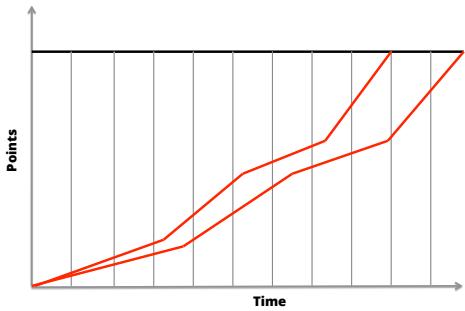


VELOCITY ADJUSTMENT: ONBOARDING

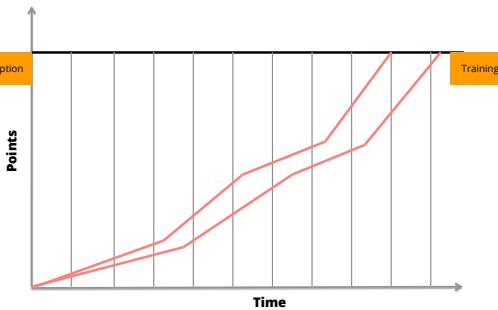
Depending on size of team, not all developers may start on the first day. May need to adjust further to match velocity to onboarding.

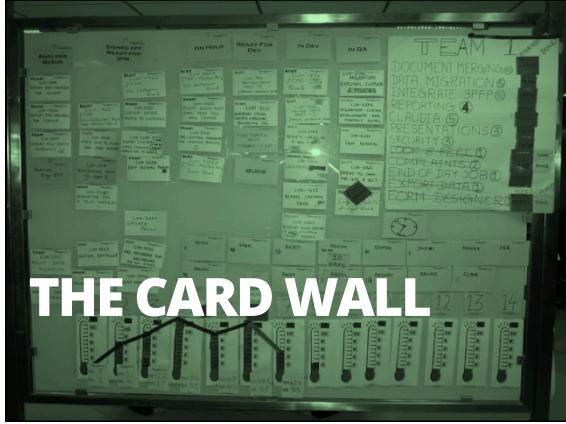


VELOCITY ADJUSTMENT: TIME-OFF



NON-DEVELOPMENT TIME





QUESTIONS?

ThoughtWorks®

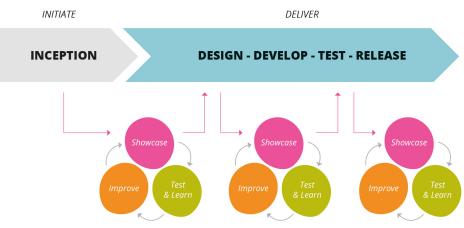
ThoughtWorks®

WRAPPING UP ON THE PROJECT LIFE

IT STARTS WITH THE INCEPTION...



...AND ITERATES THROUGHOUT THE PROJECT



QUESTIONS?

ThoughtWorks®