NOTES FROM THE LAUNCHPAD

Introduction to Agile

1. Keywords

★ SDLC - Systems Development Life Cycle (or: “Seeing, Doing, Learning, Collaborating”)

★ Agile mindset - set of principles that help people work together. Collaboration. Improving our process as well as the product.

★ Design thinking - method for finding out if we are doing the right thing, putting user as the heart of the problem, putting yourself in the user’s shoes

2. The development cycle

1) Building a brilliant product: Idea -> *magic* -> Shipment
2) There are two different approaches to building a product:

➢ Waterfall - traditional, linear, plan-driven

➢ Agile - evolutionary & flexible, iterative cycles

3. Phases of SDLC

1) Feasibility study

The purpose of this stage is to engage with the client, find out the scope of the problem and if the solutions are valuable enough to move forward. The result will be a ‘go’ / ‘no go’ verdict and many assumptions.

We assess the following:

● Is our proposal possible?

● can we build it in time?

● do we have enough money?

● will there be enough demand?

● is it legal?

2) Requirements analysis

The Product Owner will work with the business/client to discover what the proposed system requires to meet the needs of the business and the users.

<u>The activities:</u>

Gathering information:

● Stakeholder interviews & documenting business processes (assessing business needs).

● User interviews to assess the needs and expectations of the audience.

Analysis:

Are the requirements clear? Aligning on expectations.

Outcomes:

User stories specifications & sometimes a prototype

3) Design
This phase explores the foundations of the system; the technical architecture, data structure, the user journey, and the user interface.
We design the following:
Architectural
● system architecture or structure
● behaviour and different views of the system
Logical
● data flows, inputs and outputs, conducted via modelling, using an abstract model of the system
● ER (entity-relationship) diagrams
Physical
● user interface design
● user experience

4) Development
This phase we get building!
● writing the software
● implementing the design

5) Testing
This phase is carried out in 2 parts.
1. The QA (Quality Assurance) tester makes sure the software works as predicted and designed. They check:
● Software matches the 'acceptance criteria'
● If it is usable, responds correctly to all agreed kinds of inputs
● performance and speed
● can be installed and run in intended environments
2. The design team will run usability testing with users to make sure the software meets the expectations of the user and therefore the stakeholders.

6) Implementation
This phase involves activities that make software available for use.
● In agile development, we release software early and often, this is referred to as continuous deployment
● recently been termed DevOps.

7) Maintenance
This phase is constant, all systems will need maintenance once it's running for the rest of its life. A good example is when you have to update your apps, these are the changes that might have occurred:
● Improving the current system
● Fixing bugs
● Releasing new features
● Running the system to respond to increased demands

4. Waterfall vs Agile

WATERFALL - traditional and plan-driven, step by step, one by one, all about passing your product onto the next stage

- Plan upfront
- Sequential - each step is completed before moving onto the next
- Assuming that nothing will change along the way, we know exactly what people want and how to build it

Strengths

- ➢ Emphasis on documentation, therefore knowledge isn't lost
- ➢ Identifiable milestones
- ➢ Well defined deliverables for each phase
- ➢ Good for getting from A to B, when B is defined and the route is well understood
- ➢ Client has an expectation of costs, timeline & deliverables

Weaknesses

- ➢ It pretty much always fails when it comes to software
- ➢ The final product looks nothing like we expected, because of communication between different phases
- ➢ Assumptions always equal big risks
- ➢ Launch at the end with **big bang reveal** - the solution might turn out wrong (because we can't test software until it's built) and you can't go back and change it
- ➢ By launch, your idea might be out of date - the landscape, audience, tech might have changed
- ➢ Software doesn't always go to plan. Inflexible, hard to go back to a phase without incurring costs & time - hard to respond to a change
- ➢ We can't plan risks that we don't know we don't know

AGILE - iterative cycles

- We discover what people want
  - Is it valuable? (Is it new, is it different?)
  - Is it desirable? (Does anyone want it?)
- We discover how to build it
  - Is it feasible? (Can we even do it?)
  - How fast can we build it? (Cost effective?)
  - Do we have the right skills?
  - Is it usable? (Will people understand it?)
- We accept that things change along the way
  - What risks have we discovered?
  - What new technology is out there? (Will this have an impact on us?)
  - What new platforms are out there? (Do we need to adapt?)
  - What have our competitors just released? (Do we need to pivot?)
- We quickly build and test the simplest iteration of our product to learn what the risks are
- SPRINT: build -> measure -> learn -> build…

Strengths

- ➢ Requirements & solutions evolve / adapt

- ➢ One collaborative cross-functional team - no silos
- ➢ Deliver incrementally - early & often - therefore always delivering value
- ➢ Reduces risks & increases quality by continually testing and learning
- ➢ Flexible, allows evolutionary improvements
- ➢ People over process - active ownership from the whole team

Weaknesses

- ➢ Collaboration is hard but worth it
- ➢ You don't have all the answers straight away
- ➢ You have to get used to feeling uncomfortable
- ➢ It's all about having the right mindset

5. Tackling risky assumptions
   - Waterfall: Value is seen at launch - Big Bang reveal
   - Agile: value is delivered incrementally

5. Agile mindset
   - ★ **Deliver value early and often** - software has no benefit until it's in hands of customers
   - ★ **Collaboration as one team** - the best work is accomplished by team members creating together and solving problems together
   - ★ **Testing and learning** - tech is constantly changing. Success belongs to those who learn early, continuously and can adapt to change

The mindset is not a set of complete rules, but a flexible process defined by the team, allowing them to work together effectively.

The Agile Manifesto

Jeff "Cheezy" Morgan - the guy that came up with it along with his team
Date: February 11, 2001

1. Rules:
   1) Individuals and interactions over processes and tools
   2) Working software over comprehensive documentation
   3) Customer collaboration over contract negotiation
   4) Responding to change over following a plan

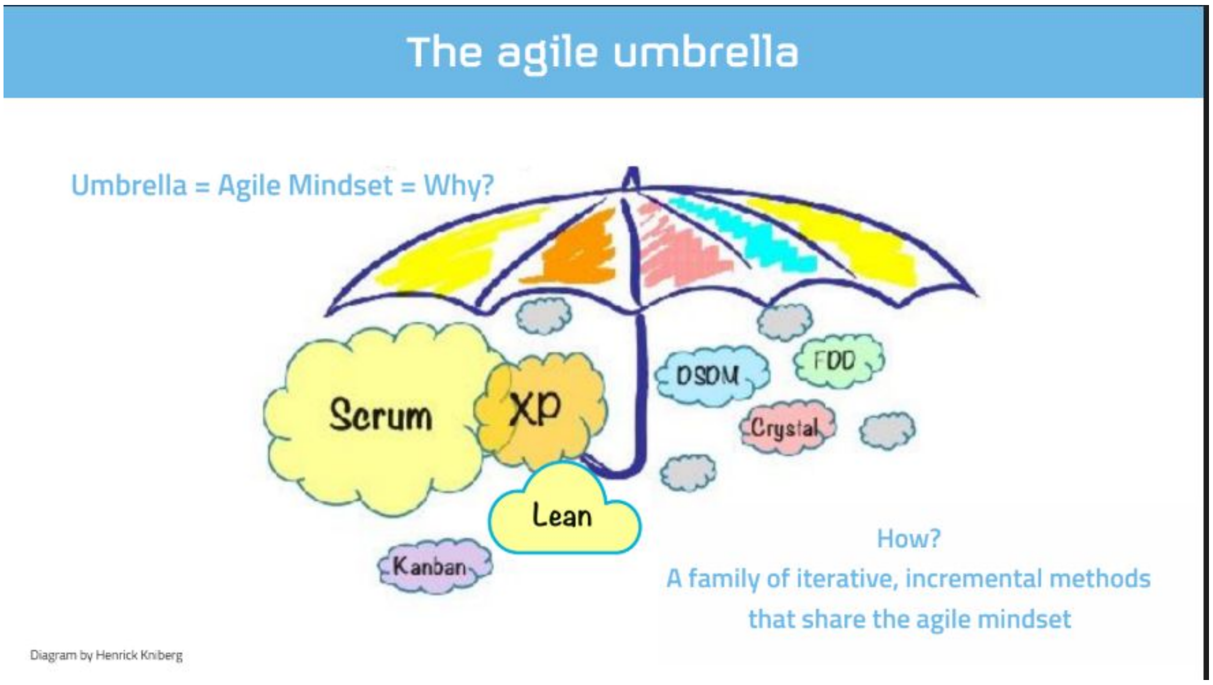
It's all about teamwork and communication, we also need to be aware that our plan might fail. Things on the right (after 'over') are less important than those on the left, but that doesn't mean we don't need them during development. Agile is a mindset, not a set of tools and practices.

2. Agile principles
   1) Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
   2) Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3) Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4) Business people and developers must work together daily throughout the project.
5) Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6) The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7) Working software is the primary measure of progress.
8) Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9) Continuous attention to technical excellence and good design enhances agility.
10) Simplicity - the art of maximising the amount of work not done - is essential.
11) The best architectures, requirements, and designs emerge from self-organizing teams.
12) At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Scrum 101 and UCD (User Centered Design)

1. The Agile umbrella


The agile umbrella
Umbrella = Agile Mindset = Why?
Scrum
XP
Lean
Kanban
DSDM
FDD
Crystal
How?
A family of iterative, incremental methods that share the agile mindset
Diagram by Henrick Kniberg

2. Scrum - delivery framework
   a) Scrum vs waterfall
   - Waterfall includes a lengthy planning process, planning is done without entirely understanding the project
   - With scrum process is broken up into smaller pieces
     Plan -> Build -> Test -> Review = SPRINT

Sprint is repeated over and over until product is finished.

a) 3 roles
    i) Product Owner
    ii) ScrumMaster
    iii) Team member
b) 3 artifacts
    i) Product backlog - list of features
    ii) Sprint backlog
    iii) Burndown chart
c) 3 ceremonies
    i) Sprint planning
    ii) Daily Scrum
    iii) Sprint review
d) Scrum workflow

Product backlog -> Sprint planning -> Sprint backlog -> Sprint (1-3 weeks) -> Potentially shippable product -> Sprint review & retrospective

Repeat this workflow for each sprint

3. The cross-functional team

- Product Owner - The PO holds the vision and is the key decision maker who prioritises what business problems the team should solve and in what order, based on the vision and the interests of the customer. **Makes sure that the value is delivered.**
- ScrumMaster - A servant leader who supports the team, facilitates core scrum meetings and encourages continuous improvement. They also unblock things when there is a problem. They don't need to have technical knowledge, but it helps if they do. They support the team. **Makes sure that things flow correctly.**
- Development team - a self-organizing team with all the specialist skills needed to deliver the product. 4-10 people with the following skill set.
    - User experience
        - User Testing / Research
        - Interaction patterns
        - Analysis (biz & user needs)
        - Prototyping
        - Content strategy
        - Interaction patterns
    - Visual design
        - Branding
        - Layout
        - Creative
        - Storytelling
        - Animation
    - Development
        - iOS or web or Android
        - Front -End
        - Back-End
        - QA/Testers

        - Interaction patterns
        - Motion

4. Sprint
A period of time, 2 or 3 weeks in which the team work together on delivering stories from the sprint backlog. The result is a potentially shippable product or MVP: Minimal Viable Product. Enough features to satisfy early customers, and to provide feedback for further development.

5. Recurring team ceremonies
- Planning - The PO & team determine what stories to tackle in the next cycle and how to achieve delivering these.
- Sprint planning session
- Sprint tackling
- Daily standup - 15 mins for the team to discuss the plan for the day and surface any questions or blockers.
- Demo - The team demonstrates completed work and get feedback from a wider stakeholder group.
- Retrospective - The team spends time identifying opportunities and actions for process improvement. What didn't go well, what could we do next

6. User story
User stories are short, simple descriptions of a feature told from the perspective of the person, user or customer of the system. We should come up with user stories by getting information from our users.
Journey of user stories:
- User story workshop - team gets together with PO and they try to write user stories
- Product backlog
- Sprint planning - select stories that we think we can complete in the next sprint.
- Sprint backlog (stories that we try to complete with that specific sprint)

7. User center design - putting yourself in someone else's shoes
Having empathy for the user improves our work and helps us come up with new, meaningful ideas. Therefore reducing risk.
Phases of user center design:
1) Inspiration
2) Ideation
3) Implementation

It's about having empathy for the user, making something that is valuable to them.

8. Personas - creating a reliable and realistic representation of key users
The purpose of personas is to create a reliable and realistic representation of your key audience segments for reference.
Persona profile - a detailed description

Effective personas:
- Represent real people that will use your service / app / product

- Focus on their major needs and expectations
- Give a picture of how users are likely to interact with it e.g. how tech savvy they are
- Help us uncover common features and functionality
- Describe real people with backgrounds, goals, and values
- Represents real people that are going to use our product
- Focuses on needs and expectations
- How users are going to interact with the product

Benefits

- Personas help the team to align & focus decisions by considering a real person
- Helps the PO & the team evaluate new features
- Allows Visual Designers, UX, Copywriters to ensure the content, interactions, interface, behaviors & the labels are appropriate to the audience
- Developers decide which approaches to take based on user behaviours

You aren't the user. Your idea of the user might not be the same as the person next to you.

User stories

1. User stories are short, simple descriptions of a feature told from the perspective of the person, user or customer of the system.

Template:

As a <type of user>
I want <goal/objective>
So that <benefit/value>

They don't offer a solution to the problem.

2. Purpose
Encourages the team to discuss features of the problem -> 'What does it actually mean, what could the features be'

3. Focus on a goal, not a solution
What does it actually need to do instead of how features are going to look like

4. Epics
Sometimes a story is too big to estimate or complete in a sprint or the details are sketchy. This is an epic. Think of it as a headline or a chapter heading. So we have to break it down into smaller stories.

4. Acceptance criteria
Acceptance criteria allows you to describe when a story is complete.

Template:

Given <some precondition>
When <I do some action>
Then <I expect some result>

Benefits:

- Allows the team to gain a deeper, shared understanding of the story/feature from a users perspective
- Allows the PO to answer what they need in order for this feature to provide value
- Allows developers and testers to derive tests
- Helps the team know when to stop adding functionality to a story

Where do they come from? -> Conversation: Acceptance criteria should be developed as a joint effort between the development team and the product owner.
How? -> By asking questions

5. What makes a good user story?
INVESTment
Independent (of all others)
Negotiable (can be changed or removed without impact to everything else?)
Valuable (delivers value to user/stakeholders)
Estimable (to a good approximation of it's complexity)
Small (can be completed in a sprint)
Testable (can this story be tested and verified)

UCD + Personas = Empathy for our users
Personas + User stories = Product Backlog
Asking questions + discussions = Alignment / Agreement
Communication + Collaboration = Effective Team
Prototyping + Iterating = Improved quality + Reduces risks

Sprinting

1. The journey of a story
Product backlog -> sprint planning -> sprint backlog
Sprint planning: Stories need enough information for the team to estimate how complex it is using story points

2. **Story points** - a relative measure for how big we think a piece of work is. They represent the effort required to deliver a story, in relation to other stories.
Sizing is a method used to help predict how long a piece of work will take to complete.
Rather than estimating using time (e.g. hours, days), we instead use story points which size pieces of work relative to each other.
It's hard to estimate, but easier to compare

3. **Velocity** - a measure of how fast a team is going, the more you know about a team's velocity the better you can predict how long a project will take. Number of story points completed in a fixed time period (e.g. sprint).

4. Fibonacci
The larger the story is, the more uncertainty there is around it and the less accurate the estimate will be. The Fibonacci sequence helps teams to recognise this uncertainty, deliberately creating a lack of precision.

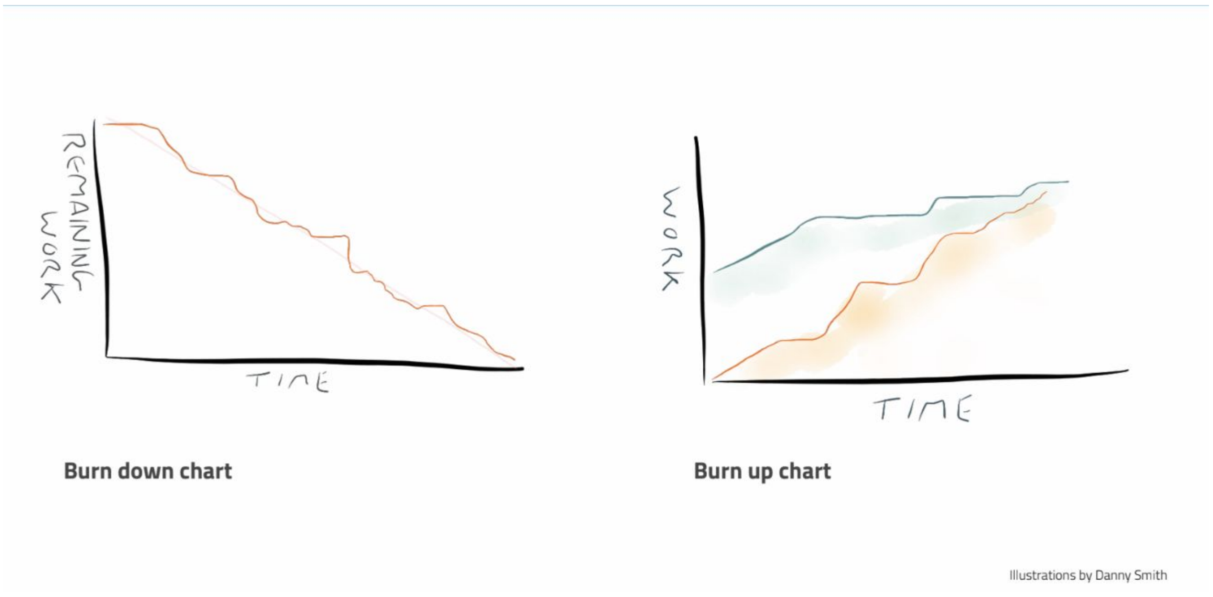
1 2 3 5 8 13 20 40 …

These numbers are often used to assign story points.

5. Things to note

a) **Who should estimate?**
   The development team doing the work. Each member they may have different perspective. This promotes discussion.
b) **Size can change**
   The size of a story reflects our understanding at a point in time, that can change as we discover more.
c) **Estimations are team specific**
   Points are relative but each team will have their own baseline.
d) **If an estimation is large?**
   The story is too big (Epic) and needs to be broken into smaller stories.

6. Burndown charts



Design thinking: empathy

1. What is design thinking?
A method for solving problems. Ensures we build the right thing. Reduces risk by validating ideas early with users.

● Having empathy for people
● Being comfortable with getting things wrong
● Exploring many paths
● Testing our solutions
● Having an iterative approach
● Teamwork

2. Eliminating the unknowns
Discovery - what people want/need, how to build it, will people understand it

3. What does it look like?
There is a lot of terminology out there. All companies will have a different version of it but they tend to follow a similar process.
1. GDS (call it Service manual)
Alpha, Beta, Launch
2. IDEO (call it Human Centred Design)
Inspiration, Ideation, Implementation
3. IBM
Observe, reflect, make

4. The common themes

a) Inspiration and empathy - Understand the problem space through research, interviews, observations, surveys, focus groups
b) Exploration - Taking what we have learnt and come up with lots of ideas! We then define a hypothesis to test.
c) Prototype - Refine an idea. Quickly design & make a prototype e.g. paper prototype, tools such as marvel or jump into code.
d) Test - Test prototype with real people to get real
feedback to validate your idea and learn what users want and how they want it.
e) Learn and iterate - We now know what to build. Define next steps and continuously iterative & improve the solution.

5. What our users want and need?
Needs - Something essential that must be fulfilled, in order to solve a problem
Wants - Something non-essential you wish to have (or think you wish to have)

Design thinking: exploration

Fall in love with the problem, not the solution
Basically, be creative

Amazing ideas come from understanding a specific problem.
Technology is the medium / platform / delivery mechanism for amazing ideas - it's not the solution.
The best ideas are usually the most simple. Often they 'hijack' something existing.

Design thinking: prototyping

1. Ideation
   - Amazing ideas come from understanding a specific problem
   - Technology is just a medium for ideas (focus on your idea, not technology to achieve this)
   - The best ideas are usually the simplest

2. Prototyping (=working software + customer collaboration)
   a) Why build prototypes?
      - We have to find out if we are building the right thing, is our idea something
      that people will actually want and maybe buy?
      - So we build prototypes to get our ideas into people's hands as early as possible.
      - It's a great way to see which ideas work best and how we might improve them.
   b) How to build them?
      There are many ways to make a prototype, it depends on your skills & how much time you have.
      It doesn't really matter as long as it allows you to quickly convey your ideas.
      Common tools:
      ● Paper prototypes (print outs or
      sketches of artefacts)
      ● A click through website/app using
      tools such as Marvel, Invision.
      ● Code (Framer or HTML)
      ● Video
      ● Role-play

Design thinking: testing

1. Pivot or perservere?
   a) Pivot - We couldn't prove our hypothesis so we create a new one to test
      Are we making enough progress to believe that our original hypothesis is correct, or do we need to make a major change and head in a new direction?
   b) Perservere - We've learnt that there is potential in our idea and we carry on…

Design thinking: retrospective

1. Why?
The best way to learn and improve is through reflection, and retrospectives are a great format for encouraging reflection and uncovering ways to improve as a team.

2. 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.