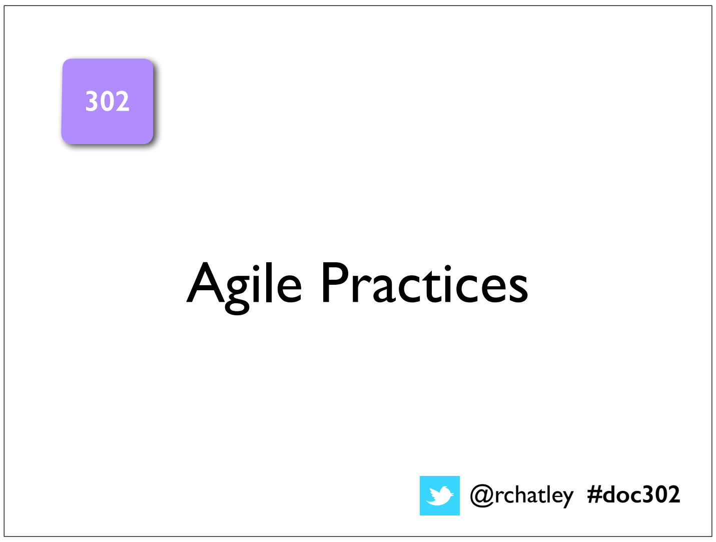


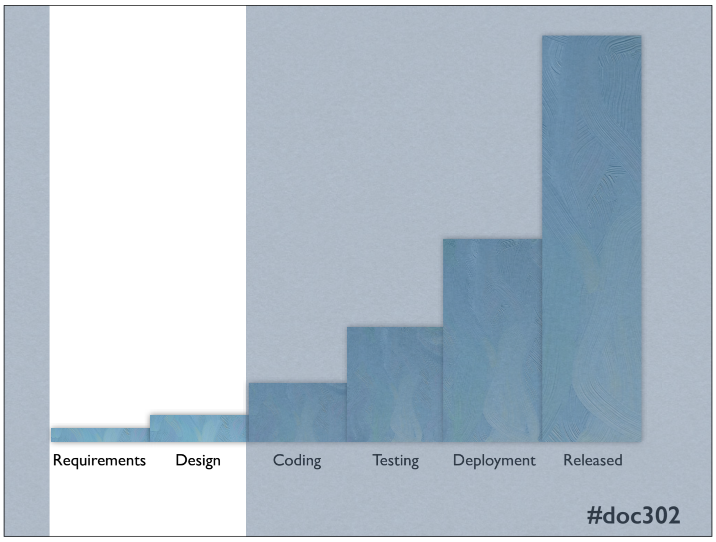
302

Agile Practices

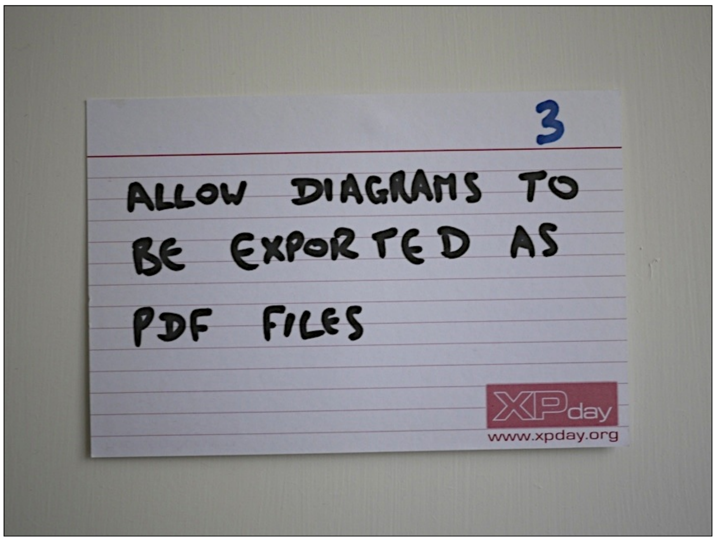
@rchatley #doc302

In this section we look in more detail at some of the individual practices that are commonly used in agile software development. For each of these you can find many more resources online giving more detail about their implementation, and case studies of teams using them.

To effectively deliver your Group Projects, you will likely want to adopt some of these practices - you will want to read and research further to choose something that is appropriate for your project.



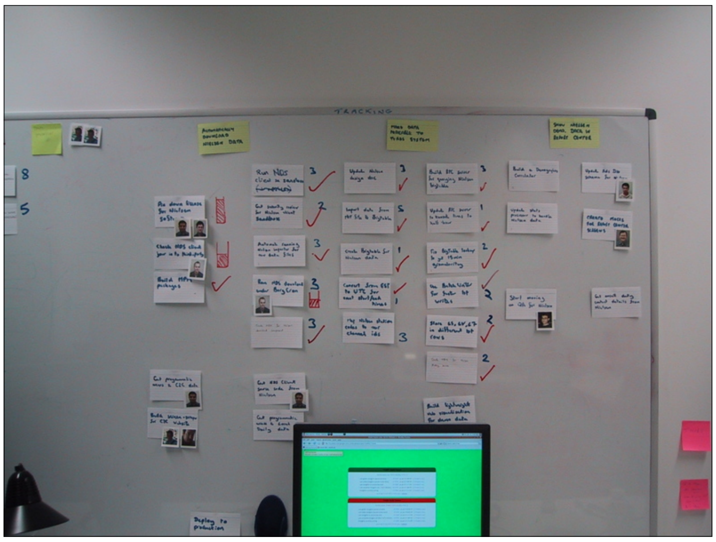
Firstly, let's look more concretely at some of the techniques and practices we might use in the early part of the iteration to specify features and to plan our work.



Agile teams do not typically write long specification documents detailing everything that will be present in the system they are building. Instead they often write each requirement in the form a of User Story. A very brief description, just enough to remind us of discussions we have had with customers about the feature. To avoid these stories evolving into long specification documents, they are often written on index cards. A story card is a promise of a conversation. It does not have all of the detail we'll eventually need - but it has enough to serve as a discussion point.

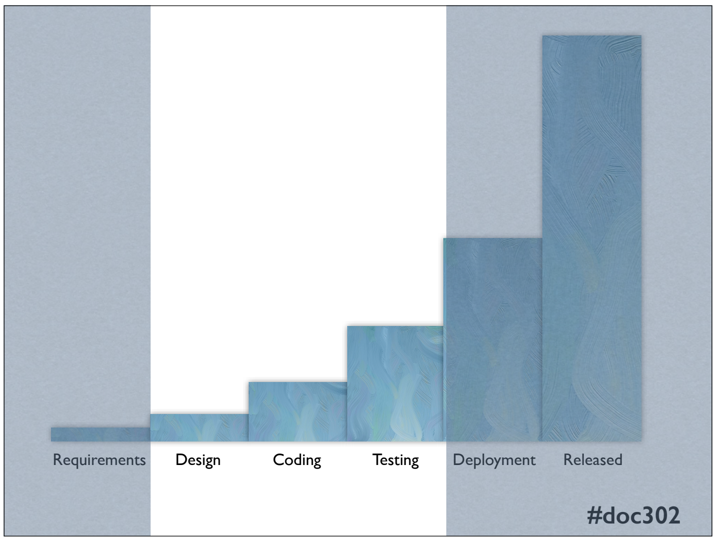
In XP and Scrum, at the beginning of an iteration, the team holds a planning meeting with the customer, to work out what features they can build (completely) within the next iteration. The customer can prioritise which features they want next. The team discusses the requirements with customer, and can estimate how much work each feature will be.

In Kanban, there are still customer discussions, but planning tends to happen "just in time", rather than on a regular schedule. This can be more efficient, but it can also be more difficult to organise.

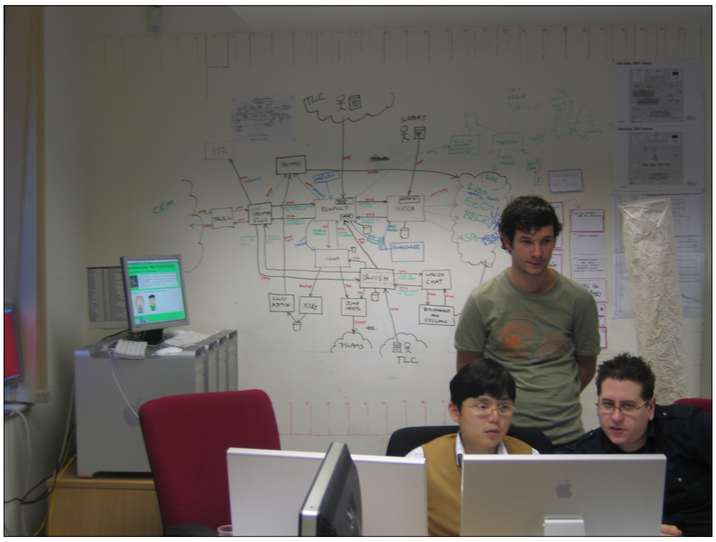


Once the plan for the iteration is set, the team typically uses a noticeboard or a wall to put the plan up in plain sight. They may add other charts or diagrams related to the project. This makes the information visible to everyone and is often called an Information Radiator.

Some teams use digital versions of these dashboards, either instead or as well as the low-tech versions.

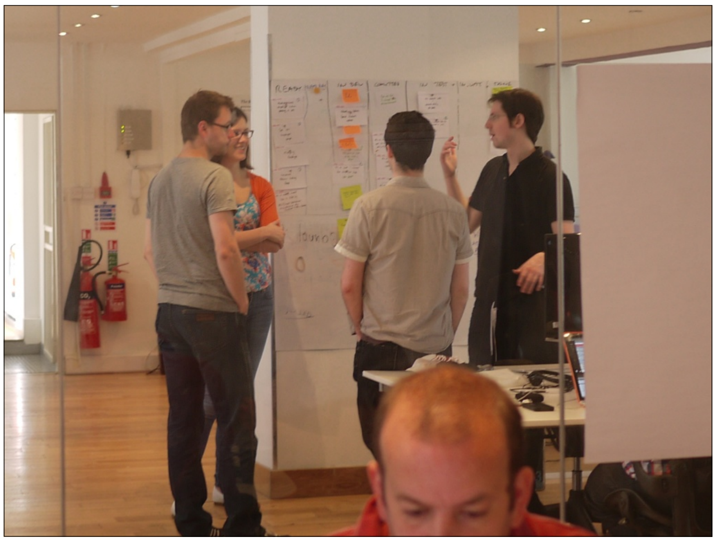


Now let's look at some more technical practices, concerned with the development of features during the iteration.

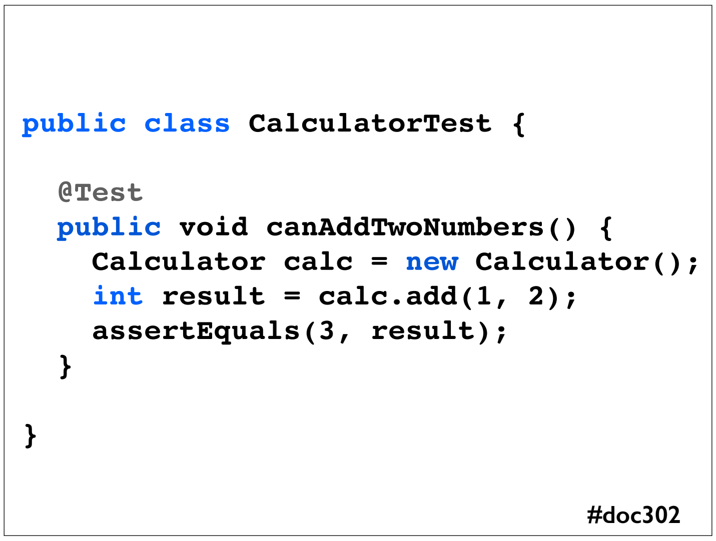


We see that one of the artefacts that the team has in this picture is a broad-brush architecture diagram. As it is up on a whiteboard people can easily talk about it, and update it when the design changes. This is better than keeping it in a computerised document, which is less likely to be kept up to date.

We also see people pair-programming. This is a key XP practice, and makes programming into a conversation. Two people work on each feature, so that more than one person knows how each part of the system works. It also helps to keep good habits, and to improve the design of the system.



XP (and Scrum) teams typically hold short daily meetings to plan what they are going to do that day, which stories to tackle, who will pair together, and to share anything they have learned the previous day. These meetings are typically held standing up to help keep them focussed and short, hence they are known as standups.



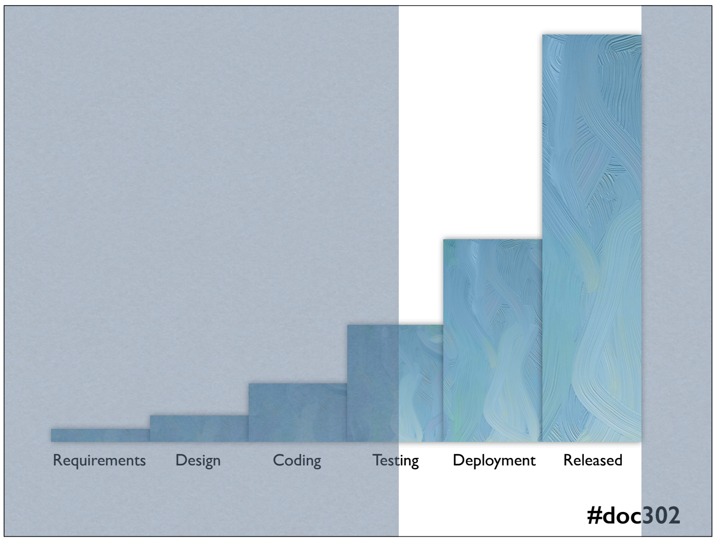
```
public class CalculatorTest {

  @Test
  public void canAddTwoNumbers() {
    Calculator calc = new Calculator();
    int result = calc.add(1, 2);
    assertEquals(3, result);
  }

}
```

#doc302

To allow us to release new versions regularly, without introducing bugs and regressions, automated testing is a key practice in agile development. Unit tests help us to ensure the quality of classes and methods inside our codebase. A good test suite give the developer confidence to change parts of the system, even if they are unfamiliar with the code, as they know that if they break something then a test will (hopefully) fail and let them know straight away.



Let's look at the remaining phases, what happens at the end of the iteration.

This is sometimes known as the "last mile". Often actually getting software released can take a long time, even though it is "code complete". It is also often a stressful time. We want to automate as much of this as possible, and make it a quick, easy, reliable, repeatable process.

Push-Button Release

We want to release our software to production (or as close to production as possible) every iteration. If the user cannot use a new feature, it is not "done". Releasing is often a stressful time, when things go wrong, especially when deadlines are involved. The best way to relieve this pressure is to automate as much of the deployment procedure as possible, and to practise it often.

Showcase

At the end of the iteration, we demo the features we've built to the customer in a showcase. This is a great time to get feedback to feed into the next planning meeting. When people see features working, they often have new ideas about how they could work better. That's ok, this is the best way to learn how to improve our product. We can feed these ideas in to the next iteration and make the product better.

Retrospective

As well as reflecting on the features that we have built, the end of the iteration is a good time to reflect on how the team is working. Is the process working? Are there things we could do better? A quick retrospective meeting is a good way to review these issues. Work out something to try and improve during the next iteration. Retrospectives are key to continuous improvement.

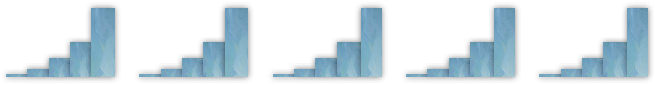
There is lots of material online giving ideas for different styles of retrospective., e.g. http://retrospectivewiki.org/

Iterative Approach

Start with the simplest possible version of the system

Once you've learned that it is worth it, improve it

Iterative is not the same as incremental



#doc302

By picking an agile method, applying the practices, and reflecting on your progress through retrospectives, you can iteratively improve the way that your team works to make it more efficient and more effective.

You can also continuously improve your product by releasing new versions and eliciting feedback.