**Industry In The Classroom**

**Industry In The Classroom**

**Virtual Engagement Lesson Guide**

**Software Development**

Version 1.2

## June 2021

## 

## Introduction

The web is filled with educational resources that can be used to introduce learners to software development. But it can be difficult to know which ones will work well for young learners, and which are suitable to be delivered virtually in classrooms.

This lesson plan is an example of a high quality activity suitable for highschool age students.

This activity introduces Behaviour Driven Development, a modern agile work practice followed by many tech companies, and puts it into practice with a representative example.

In this activity, pupils step into the shoes of a testing team who have been tasked with creating a test plan for a cinema chain’s new booking system. This will get learners to think about the importance of testing, and how it fits into modern software development processes.

The lesson is designed to be delivered within a single school period. This activity may be used to support the teaching and delivery of learning outcomes for the National 5 Computing Science qualification.

*Daniel Devine*

*Digital Skills Education*

*June 2021*

## How to use this lesson plan

* Read through the lesson plan alongside the [sample slides](https://docs.google.com/presentation/d/16inGIwZEYsxR_mFRH1PtVdcXUTU0Gppj4BcsehBHq4U/edit?usp=sharing), this suggests timings and talking points.
* Adapt the lesson plan and [sample slides](https://docs.google.com/presentation/d/16inGIwZEYsxR_mFRH1PtVdcXUTU0Gppj4BcsehBHq4U/edit?usp=sharing). Where possible use these notes but you should also add in your own knowledge and experience.
* Practice running through the activity using the lesson plan

Remember, for advice on organising and running this virtual engagement session you should refer to the [Virtual Engagement Best Practice Guide](https://www.ourskillsforce.co.uk/invest-in-young-people/tech-industry-in-the-classroom).

## Factsheet

**Topics:** Software Engineering, Software Testing, Agile Development

**Target Audience:** S1-S4 (ages 11-16) secondary school students.

**Target Volunteers:** Anyone working in technology with a familiarity with the basics of how  
 software is produced. Background knowledge in software testing is  
 useful but not required.

**Skills Level:**   Suitable for Beginners.

## **Need to know:** This activity introduced Behaviour Driven Development, a popular agile software development methodology in a fun and interactive way.

**Preparatory work:** 1)Read the [Virtual Engagement Best Practice Guide](https://www.ourskillsforce.co.uk/invest-in-young-people/tech-industry-in-the-classroom)

2) Follow the activity by yourself

3) Customise your session

**Required equipment:** Projector screen for slides

**Websites required:** n/a

**Hints for on the day:** Impart your own experience, if you’ve encountered something similar,   
 say so! Share your anecdotes.

## 

## Lesson Plan

|  |  |
| --- | --- |
| **Time** | **Software Development (Testing) Virtual Engagement**  **Volunteer:**  **Teacher:**  **Sample Slides**:  <https://docs.google.com/presentation/d/16inGIwZEYsxR_mFRH1PtVdcXUTU0Gppj4BcsehBHq4U/edit?usp=sharing> |
| **Hook** | |
| 2 | *Make sure to include:*   * *Your name and how you’d like to be addressed* * *Where you are calling from* * *What your role is (a brief description of what you do)* * *Who you work for and what the company does*   “Hello, I’m Andrew McLeish. Today you can call me Andy. I’m joining you today from our office in Dundee. I am a software engineer and I work at Dundee Developer Collective. We’re a company that produces specialised software for other companies to use.”  “Today you’re going to learn to think like software testers, and find out how software is really made.”  “You’re also going to get the chance to ask me any questions about my role as a software engineer.” |
| **Engage** | |
| 5 | *Give a short, 2-3 slide, presentation on the area and what you do. Think about what context the learners might need for the activity - in this case linking it to the need to work digital evidence.*  *As part you this section you could:*   * *Talk about a task you do at work* * *Ask the pupils a question* * *Discuss a relevant recent news story*   *See slides 4 and 5 for inspiration.* |
| **Explore** | |
|  | **How is software developed?**  Slide: What is software engineering?  Slide: Teams that all work together  *Cover the process of creating software and the various teams in chronological order that come together to build software (very brief):*   * *Architecture, Business analysis, Design, Development, Testing, Release*   *Highlighting these various areas and how they have to work together shows the large number of people who have to work together in order to bring an application to life.*  Discussion - “When does the testing team first get involved?”  We’re going to be talking about testing today, so here’s a question for us to think about.  Traditionally after the software has been developed or created, a test plan is written and then followed to test that the code does what the developer expected.  You might have come across this in your computing science lessons, ‘normal’ ‘extreme’ and ‘exceptional’ etc.  Here’s the traditional way of doing things. You have these teams, and they do their job, then pass the project onto the next team to do their bit.  First planning talks about the business side of things with the client who is the person or company who needs the software.  Then, they tell the analysis team what the problem is that needs solved. They investigate it, and let the design team know what the client needs, and about the problem.  The design team then goes away and designs a solution, how should the software go about solving the issue. They then tell the programmers in the development team what they need.  The development team follows the design then tests it works. And it passes!  But actually many companies like us involve testers right from the start!  We’ll show you why that’s a good way of working in this session.  Slide: Why do we test?  There’s a few reasons why we test software:  To find bugs  To check if it’s reliable  And whether it does what it’s supposed to do  A big problem in software development is that you end up building the wrong thing. The app might look great, and not have any bugs. But it might not actually do the thing the customer needs it to do! Sometimes things get left out, or misunderstood. We don’t want to end up building the wrong thing!  How can we help prevent this? We can come up with a shared understanding of what we’re trying to build. We group together the most important people - the business analysts (representing the customer), the software developers (who make the app), and the testing team. Together, we come up with a shared understanding of what the app needs to be able to do.  **What is BDD?**  Slide: What is Behaviour Driven Development  There’s a technique, a way of working, that lots of software companies like us use to stop us making the wrong thing.  It’s called Behaviour Driven Development.  **Slide**: Three Amigos  At the start of the project we have something called a Three Amigos meeting. The three amigos are business analysts, the programmers, and the testers.  Together we decide what we’re trying to build.  It’s really important that all three teams understand what it is we’re making together. Not everyone is technical, so it’s got to be clear and written in English.  At the meeting we create user stories which are a type of test plan. These are small documents that explain all of the things that a user needs to be able to do when using the finished app.  Today you’re all going to be an Amigo, let’s show you an example of a user story...  Slide: As a… thirsty person  Well here’s our technical document! You’ll notice it’s just written in English and is in first person too. Maybe this isn’t what you expected a technical testing document to look like.  Here’s the problem we’re trying to solve.  As a thirsty person,  in order to make a cup of tea,  I want to boil some water  What type of kitchen appliance do you think this user story is for? (a. A kettle)  This explains:   1. Who the feature is for, thirsty people 2. Why is the feature relevant, they want to make tea 3. Why do they want the feature, to boil water   So now we know why people need a kettle.  Even if someone on our team had never heard of tea or kettles before, they’d have been able to read that story and understand that people need kettles to make tea. In order to make tea you need to be able to boil water.  But that story doesn’t describe how the kettle actually boils the water!  How will we know we’ve made the right thing? How do we want it to behave?  Let’s think about what we’d want to test, what the kettle needs to be able to do.  So...what’s the main feature of a kettle. Boiling water!  Slide: Everyday example  The scenario is Turning the kettle on to boil water  GIVEN I want to boil the water  WHEN I press the switch down  THEN the heating element turns on  We’ve got a structure to follow:  **Given** – the starting point (we want to boil water)  **When** – the action the user takes (we press the switch)  **Then** – What happens next (heating element goes on)  I think you can agree that if the kettle behaves in this way, then it’s doing its job - that’s what we want to happen. It’s letting the user boil water.  But this doesn’t tell the full story though, what happens after the heating element is turned on? When does a kettle turn itself off? It doesn’t keep boiling forever!  How else might we expect the kettle to behave?  *ASK!*  We need another test case to answer when it turns off.  Slide: Everyday example (cont’d)  GIVEN the heating element is turned on  WHEN the water temperature reaches 100c  THEN the heating element will turn off  Again, I think you can agree that if the kettle behaves in this way, then it’s doing its job - that’s what we want to happen. The kettle will turn off once it’s finished - so you know when it’s done.  Slide: Considering if things go wrong...  GIVEN the heating element is turned on  WHEN the heating element overheats  THEN the heating element will turn off  This time, there are lots of different ways that you could tell if the kettle had no water. Maybe you check the weight, maybe it has some other kind of sensor. This is the type of thing that the engineer, or programmer would be able to use their expertise and advise the best way to do it. That’s why they are one of the three amigos.  Look at the name we’ve given the test case. It described clearly what the feature being tested is.  With these three test cases, it’s really clear how the kettle needs to work. Everyone involved, whether they are designers, engineers, the people who test, or the people who have to market and sell the kettle all have the same understanding of the kettle, and it’s main features.  And that’s Behaviour Driven Development. |
| 2 | **Introduce the activity:**  Slide: AYE-Macs logo  It’s now your turn to be one of the three amigos, representing the testing team.  We’ve been employed by a cinema, the *AYE-Macs,* who are creating a new app which will allow customers to book film tickets on their phone before they get to the cinema.  Slide: Quote  Here’s the requirements from the business analyst.  “Our customers want to browse films, and buy tickets for a specific showing. The solution is a new app.”  We are the testing team, the senior test engineer has already started producing user stories, but the scenarios aren’t finished yet.  You need to help us complete the scenarios by filling in the blanks.  Slide: Given-When-Then  Let’s look at the first scenario the senior tester has written.  Remember, the point of each scenario is to give us a test case, it’s a way of checking the app is behaving the way we’d expect it to.  What would be the most sensible option here?  Slide: Solution  Giving the user a ticket with a seat number was the right thing to say here. That’s what you’d expect to get when you book a ticket.  In an ideal world, whenever a customer goes to book a ticket, they get one. That’s what we call the happy path, the user gets what they want and everything works perfectly. They’re happy.  In testing, we also have to think about what else could happen. What does the ‘unhappy path’ look like?  With this scenario, are there any reasons that could lead to the user not getting given a ticket? There can be lots of unhappy paths.  *(Some examples might be. Don’t try and explain all these!)*   * *We don’t have the film they want to see* * *They don’t have a time slot that suits them* * *They aren’t any tickets left* * *They don’t have enough money to buy a ticket* * *There isn’t a cinema near enough to them*   Slide: Two scenarios  The business analyst says that film showings regularly sell out and we don’t want to oversell tickets. It would be bad for the business.  What the lead tester has done, has created a new scenario for when the showing is sold out. This is an unhappy path, it’s a path where the user doesn’t get what they want.  We need to clearly describe the way the app should behave if the showing is sold out and we can’t sell any more tickets.  Slide: The showing is sold out for a film I want to see  Read the test case, and have a think about what you would choose for this scenario.  *Discuss.*  B is the best option. It would work from a technical perspective, if we can’t sell a ticket just tell the customer that it’s sold out.  The business analysts might have additional ideas about what else should happen. That’s why it’s important to involve everyone when making these test cases.  What else could the app tell the user if the showing is sold out? If you were buying tickets and there was none left, what would be helpful? What would a human say?  *Examples: ‘times of other showings with seats available’, ‘other films you might like’, ‘join a waiting list for more tickets’.* |
| 10 | **Worksheet**  [**https://docs.google.com/document/d/1Z0EKTPhd\_wkx3SL64vRoK5EGlSIeDnZ0bYTIeGYxffc/edit?usp=sharing**](https://docs.google.com/document/d/1Z0EKTPhd_wkx3SL64vRoK5EGlSIeDnZ0bYTIeGYxffc/edit?usp=sharing)  Now it’s your turn to have a go completing the scenarios for some other features of the app.  Think about how you think the app should behave in these scenarios.  What would be technically possible? What would be a good experience for the customer? What might be good for the business?  When there’s a blank space to fill in, there’s lots of potential answers. Say what you’d want to happen if this was your app.  *Give students a few minutes to work through the sheet before discussing answers. You may wish to step through the questions together depending on the class and level of interaction.*   1. *Asked to log into my account* 2. *Example: All films tagged with Star Wars sorted by relevance* 3. *Example: Removing tickets from my basket, Ticket is removed* 4. *Example: I type in the promotional code HALFOFF into the box, the price of each ticket changes to £4* |
| 1 | **Recap**  Slide: There’s more to testing than just finding bugs  Hopefully you now agree with me that there's a lot more to testing than just sitting alone finding bugs. The test team is really important and it’s a key link between other technical and non-technical teams.  Really well done with the worksheet and your answers today. If you enjoyed developing the scenarios, and thinking like a customer then you might have the right skills and attitude to be a software tester. |
| **Involve** | |
| 10 | Q&A with the class. The teacher will relay questions to you.  **Common Questions and sample answers:**  “*How much do you get paid?”*  *“People working in Software testing on average get paid around £26,000 a year when they start. With 3-5 years experience, you can earn £35,000 to £50,000 a year. This is about the same as other types of software jobs.*  *“What and where should I study after school?”*  *“There are different options, most universities have degrees in software engineering and computer science, but there are also modern and graduate apprenticeships. I went to Abertay University and studied computer science, then changed job roles after a few years at my previous company.”*  *“What school subjects should I pick?”*  *“People in software engineering have a wide range of backgrounds. Computing is probably the most important subject. Maths and Physics are useful too.”*  Link what you say to everyday life, make it relatable to the learners.  Share *your* experience, you can give so much more than generic career information. |
| **Sustain** | |
| 2 | * Visit the Digital World website  (<https://www.digitalworld.net/>) * Is your company planning any outreach? * Does your company offer apprenticeships or other early careers? |

# **References**

**Authors:**

Daniel Devine

Craig Steele

[digitalskillseducation.com](http://digitalskillseducation.com)

**Thanks to:**

Eva Gilburt  
Anita Squires  
Greig Baird  
Laura Casci  
Shona McAlpine  
Stirling High School  
BJSS  
Debbie McCutcheon

## 