LESSON 72

Iterative Design

40 minutes | Software Engineering and Testing | Outcome 1.19 | Outcome 1.20 | Outcome 1.21 | Outcome 1.23 | Outcome 1.5 | Outcome 2.21 | Outcome 2.22

WHAT THIS LESSON DOES

In this lesson, you'll explore the process of iterative design through a step-by-step approach. Learn how to develop software in small, repeating cycles, from planning and implementing features to testing and refining based on feedback, enhancing your project incrementally.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.19	identify features of both staged and iterative design and development processes
1.20	collaborate and assign roles and responsibilities within a team to tackle a computing task
1.21	identify alternative perspectives, considering different disciplines, stakeholders and end users
1.23	reflect and communicate on the design and development process
1.5	evaluate alternative solutions to computational problems
2.21	critically reflect on and identify limitations in completed code and suggest possible improvements
2.22	explain the different stages in software testing

WHAT STUDENTS SHOULD BE ABLE TO DO

- Understand the core principles and cycles of iterative design in software development.
- Recognise the advantages and disadvantages of using an iterative approach.
- Apply the steps of planning, designing, implementing, testing, and refining in a project context.
- Evaluate feedback to make informed improvements in each iteration.
- Develop adaptability and problem-solving skills through reflective practice.

BEFORE THE LESSON

- This is a discussion and theory lesson with no code to run. Nothing needs installing. Have a real example of an iteration ready to talk through, ideally a feature of the Coursework Project or an Applied Learning Task students recognise.
- For the reflection, have students ready to write in a notebook or an open document. Decide in advance whether they keep it, as this thinking is useful evidence when documenting their own project's development.

Iterative Design

HOW THE LESSON RUNS

Introduction
Understanding Iterative Design
Planning a small part
Designing and implementing
Testing and reviewing
Refining
Reflection Activity
Wrap Up

TEACHING MOVES

- 2. Understanding Iterative Design. Name the four phases as a loop, not a line. Stress that each iteration produces a working version, however small. Link this explicitly to how the Coursework Project is expected to develop over time rather than in one final build.
- 3. Planning a small part. Push the word small. Ask each student to name one feature they could finish and test in a single sitting. Reject anything too big to test in one cycle and get them to slice it smaller.
- 4. Designing and implementing. Separate designing from coding. Ask students to sketch or describe the feature before they picture writing any code. Emphasise that the aim is a functional increment you can actually test, not a finished product.
- 5. Testing and reviewing. Make the point that a screen that looks right is not proof it works. Ask what specific test would tell them the feature is correct, and who could give useful feedback beyond themselves.
- 6. Refining. Connect refining back to specific feedback from the previous phase. Warn against changing things nobody asked about. Show that refining closes one iteration and feeds the planning of the next.
- 7. Reflection Activity. Circulate and check each student's iteration is genuinely one small cycle, not the whole project relabelled. Ask two or three to describe their planned iteration aloud before they commit it to paper.
- 8. Wrap Up. Draw the contrast with staged, upfront planning directly. Ask when a rigid plan would actually be safer, so students see iterative design as a choice with trade-offs, not always the right answer.

WHAT TO WATCH FOR

- Students treat an iteration as the whole project done once, so their planned iteration is far too large to design, test and refine in a single cycle. Insist an iteration is something they could realistically finish and test quickly. Have them keep slicing a feature until one honest cycle fits inside a short time.
- Students believe iterative design is simply the newer, better method and staged, upfront planning is outdated. Ask for a situation where requirements are fixed and cannot change, such as safety or legal constraints. Show that iterative design suits dynamic projects, not every project.
- Students think a feature that runs and looks right has passed testing. Separate looking correct from being correct. Ask what result a test should produce and how they would know if the feature failed silently.

DIFFERENTIATION

- Emerging: Give the four phase names on the board and ask the student to match each to a one line description in their own words before attempting the reflection. Pair them with a peer to describe a single feature aloud, so planning happens in conversation before writing.
- Developing: Ask them to explain why feedback from someone other than themselves improves the refine phase, not just to list the steps.
- Proficient: Have them contrast iterative design with a fully staged approach and justify when each is appropriate, which is the kind of evaluation expected at Higher Level. Ask them to draft how they would document one real iteration of their own Coursework Project, including what evidence they would keep of each phase.