

1. Unit Overview and Foundations

Previous Learning (What You Already Know)

- Prototypes: You should have seen prototypes and created a prototype for an interface (for example, with buttons and messages).
- Data Testing: You should have tested block-based programs using different data types (normal data accepted, invalid data rejected).
- Evaluation: You should have evaluated programs by identifying positive elements and considering improvements.
- Project Plans: You should have followed a project plan when developing a project on your own or with others.
- Debugging or Error Correction: You should have corrected errors in block-based or text-based programs.

Learning Objectives

Identify the positive and negative elements of a software prototype.

Explain why project plans are used in software development projects.

Identify the key features of a test plan.

Use a test plan to test an algorithm.

Understand the different ways errors can happen in a program.

Test or debug programs to find errors.

Key Words Glossary

Bug: An error in a program.

Debug / Debugging: The process of finding this bug and correcting it.

Evaluate: Looking at the prototype, comparing it against its original purpose, and seeing how good it is by identifying positive points and where it could be improved.

Interface: Lets you interact with the computer; what you see on screen (words, images, buttons) or use to input (controller buttons, opening windows).

Logic error: An error that occurs when the syntax is correct and the code runs, but does not do what it was supposed to do and gives an incorrect output.

Project plan: Plans used to decide what you are going to do and when you are going to do it, created before starting work on a project.

Prototype: An early version of an end product or system, possibly for part of the system and not fully working.

Software development: The process where you design, create, and test a program.

Systematically: Working through a process one step at a time from start to finish.

Test plan: A formal way of recording your testing by listing tests, inputs, and expected outputs before testing starts.

2. Core Concepts and Classroom Discussion

Part A: Software Prototypes & Evaluation

A prototype is an early, preliminary version of a software system or interface. It is created during software development so designers and users can examine features before full programming is completed.

The Software Development Cycle follows a continuous feedback loop:

Cycle: 1. Create Prototype → 2. Evaluate Prototype → 3. Change Prototype (add new or modifies features) → 4. Working Product

This process is repeated until a final working product is created that everyone is happy with.

Evaluation Checklist: Six Key Questions for Evaluating Any Prototype:

1. Does it meet requirements? Compare against original requirements and purpose.
2. Is the design appropriate? Are words, colours, and layout easy to read and follow?
3. Is the input method suitable? Does the user clearly know what and how to input?
4. Is the output suitable? Are text and images clear, suitable, and understandable?
5. Positive features: What works well and what do you like?
6. Areas for improvement: What needs to be changed or added to make it better?

Part B: Project Plans (Think Tank)

Contextualizing Scenario (Why do we need a Project Plan?)

A company hires a large team (artists, designers, programmers, testers, evaluators) to build a complex 3D virtual world game. If work starts immediately with no plan, team members duplicate code, work out of order, or produce components that fail to connect when combined.

Why We Plan: Key reasons why project plans are essential in software development.

- Divide project between team members: Ensures people always have suitable tasks to do and stay busy.
- Identify deadlines: Helps meet the final date and allows problems to be spotted early so others can help.
- Agreed requirements: Ensures everyone works towards the exact same end product so parts combine properly.
- Clarity of responsibilities: Stops people from duplicating tasks or leaving tasks unfinished.
- Reduce development risks: Gives team members a constant reference guide whenever they are unsure what to do.
- Task sequencing / ordering: Ensures team members are not blocked waiting for prerequisite tasks to finish.

Part C: Test Plans and Test Data

Programs are tested to make sure they work properly and cannot break. A test plan is written before the program is run. If any test fails, you must make corrections and repeat ALL tests from the beginning, because a fix can break an earlier working test.

Test Data Categories: Three Types of Test Data:

- Normal Data: Typical, expected data that should be accepted and give a correct output (e.g. inputs 1 and 2 produce 3).
- Invalid Data: Data the program cannot process and must not accept; it should not give a result (e.g. entering "house" instead of a number).
- Unusual Data: Data that is not expected or usual, but is acceptable to use (e.g. negative numbers like -3 or decimals like 2.5).

Test Plan Columns: The 5 Columns of a Formal Test Plan

Column 1: Test number | Column 2: Description | Column 3: Input | Column 4: Expected output | Column 5: Actual output.

Columns 1 to 4 are completed BEFORE testing starts. Column 5 is completed DURING testing. If expected matches actual, the test passes; if different, the test fails.

Test Plan for a Program That Adds Two Numbers

Test Data Type	Description	Example Input(s)	Expected Output	Result
Normal	Typical values entered	5, 3	8	Pass
Boundary	Lowest valid values	0, 0	0	Pass
Boundary	Highest valid values	100, 100	200	Pass
Unusual / Invalid	Letter entered instead of a number	A, 5	Error message shown	Pass
Unusual / Invalid	Blank input	(blank), 5	Error message shown	Pass

Part D: Syntax Errors vs. 2. Semantic Error or Logic Errors and Debugging

Syntax is the grammar and structure of the language (exact keywords, brackets, spelling). A syntax error breaks these rules (e.g., missing bracket in `print("Hi"` or using `:=` instead of `=`). The program will not run fully, it fails to start or crashes with an error message.

Semantic error is a logical error that occurs when the syntax is correct and the program runs from start to end without crashing, but gives an incorrect output or result because it did not do what it was intended to do.

3. Short Answer Questions & Answers

Q1: What is a prototype?

Answer: A prototype is an early version of an end product or system. It might only be for part of the system and might not be fully working, such as an early design for a user interface.

Q2: What is an interface?

Answer: An interface is what allows a user to interact with the computer. It includes what appears on the screen such as words, images, and buttons as well as input devices like game controllers.

Q3: When evaluating a prototype, what should you compare it against?

Answer: When evaluating a prototype, you must compare it against its original purpose and requirements. This allows you to identify what it does well and what areas need to be improved.

Q4: What is a project plan and when is it created?

Answer: A project plan is a schedule used to decide what you are going to do and when you are going to do it. It is created before anyone starts working on the project.

Q5: Why is it important to set deadlines in a project plan?

Answer: Deadlines ensure the final completion date is met. They allow the team to identify problems along the way so that other members can help out to avoid delays.

Q6: What is the difference between normal test data and invalid test data?

Answer: Normal test data is typical, expected data that should be accepted and give a correct output. Invalid test data is data the program cannot process, which should be rejected without producing a normal result.

Q7: What is invalid or unusual test data?

Answer: Invalid or unusual test data is data that is not expected or typical, but is completely valid and OK to use. Examples include negative numbers or decimals when entering numeric amounts.

Q8: Which columns of a test plan are completed before testing starts?

Answer: The first four columns Test Number, Description, Input, and Expected Output must be completed before testing starts. Only the Actual Output column is filled while testing.

Q9: What is a syntax error and how does the computer react?

Answer: A syntax error is when code breaks the rules and structure of the programming language. The computer will not run the code fully, it will either fail to start or crash with an error message.

Q10: What is a semantic or logical error and how can you detect it?

Answer: A semantic or logical error is when the code runs completely from start to finish without crashing, but gives an incorrect output. It can only be detected by testing the program and comparing actual outputs with expected outputs.

4. Second-Order Thinking Skills Questions

Question 1: A program calculates the perimeter of a classroom by asking for length and width. Classify each of the following test inputs as Normal, Invalid, or Unusual, and give a reason: (a) 15, (b) "long", (c) -4.5.

Answer:

(a) 15 is Normal data because it is a typical expected positive number.

(b) "long" is Invalid data because it is text and cannot be calculated as a number.

(c) -4.5 is Unusual data because negative decimal lengths are unusual in real rooms, but numeric decimals are valid numeric inputs.

Question 2: A prototype interface for a game start screen displays only a blank white box with no buttons, text, or instructions. Evaluate this prototype using the criteria:

(1) Does it meet requirements?

(2) Suggest two specific improvements.

Answer:

(1) Does it meet requirements? ➔ It fails the requirements because the user cannot tell what to input or how to interact with it.

(2) Suggest two specific improvements: ➔ Add clear title text and instructions, and add labeled interactive buttons such as 'Start New Game', 'Load Game', and 'Quit'.