ANSWERS

A14.2 Development methodologies

Software development, law and ethics · A level · OCR H446 1.2.3, AQA 7517 4.13.1.1, Eduqas A500QS 1.5 · about 45 min

What this lesson is about

Waterfall, spiral, agile, extreme programming and rapid application development: how each works, when to use it, and planning agile sprints.

- Waterfall
- The spiral model
- Agile methodologies
- Extreme programming
- Rapid application development
- Comparing them
- Velocity in practice

Questions

6 marks in all

1. Which methodology is driven by identifying and reducing risk on each loop, often by building prototypes? [1 mark]

- ☐ **A** The spiral model
- ☐ **B** Waterfall
- ☐ **C** Extreme programming
- ☐ **D** Rapid application development

> **Answer:** A. Boehm's spiral model goes round objectives, risk analysis, development and planning, with the biggest risks dealt with first.

2. A hospital commissions software to control drug doses. The requirements are fixed by regulations and must be fully documented. Which methodology suits best? [1 mark]

- ☐ **A** Waterfall
- ☐ **B** Extreme programming
- ☐ **C** Rapid application development
- ☐ **D** Agile with two-week sprints

> **Answer:** A. Stable, fully specified, safety-critical requirements suit waterfall's sign-off at each stage and heavy documentation.

3. Which of these are practices of extreme programming? [1 mark]

Tick every answer that is true.

- ☐ **A** Pair programming
- ☐ **B** Test-driven development
- ☐ **C** Continuous integration
- ☐ **D** Signing off each stage before the next begins
- ☐ **E** Heavy upfront documentation

> **Answer:** A, B, C. XP uses pairing, tests written before code, frequent integration and releases; sign-off and heavy documentation belong to waterfall.

4. What is a drawback of rapid application development? [1 mark]

- ☐ **A** Prototypes built quickly can lead to poorly structured, inefficient code
- ☐ **B** The users never see the system until it is finished
- ☐ **C** It cannot use prototypes
- ☐ **D** Requirements must be fixed before any work starts

> **Answer:** A. RAD's speed and time-boxing trade against code quality, and it needs a lot of the users' time.

5. In agile, what name is given to the number of story points a team completes in one iteration? [1 mark]

> **Answer:** velocity. Velocity is used to plan how many stories fit in each future sprint.

6. Why does waterfall cope badly with changing requirements? [1 mark]

- ☐ **A** A change found late means going back through stages that were already completed and signed off
- ☐ **B** It does not have a testing stage
- ☐ **C** The client is involved in every iteration
- ☐ **D** It uses pair programming

> **Answer:** A. Each stage builds on the signed-off output of the one before, so a late change is expensive to make.

The task: plan the sprints

Plan the delivery robot's sprints. `VELOCITY` is the whole number of story points the team finishes in one sprint. `backlog` is a list of `(story, points)` tuples in priority order: `story` is a string, and `points` is a whole number of 1 or more. 1. Go through the backlog in order. Any story with more points than `VELOCITY` cannot fit in a sprint: print `too big, split it: <story>` and leave it out of the plan. 2. Put the remaining stories into sprints, in order. Add each story to the current sprint while the sprint's total stays at or below `VELOCITY`. When the next story would take the total over `VELOCITY`, that sprint is finished and the story starts a new sprint. 3. Print each sprint as `sprint <n>: <stories> (<total> points)`, numbering from 1, with the stories separated by `, `. 4. Finally print `sprints needed: <n>`. Compare with `VELOCITY` in your code, not the number 8. The robot stays still.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

VELOCITY = 8
backlog = [("drive to a bay", 5), ("stop at the wall", 3), ("beep when blocked", 2), ("map
the room", 13),
           ("report battery", 1), ("follow a line", 8), ("avoid people", 5), ("log each
delivery", 3)]
```

The hint students can ask for: Deal with the stories that are bigger than a whole sprint first. Then keep a list of the stories in the sprint you are filling and its running total; when the next story would take the total past the velocity, print that sprint and start a new one with the story. Remember to print the last sprint after the loop.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

VELOCITY = 8
backlog = [("drive to a bay", 5), ("stop at the wall", 3), ("beep when blocked", 2), ("map
the room", 13),
           ("report battery", 1), ("follow a line", 8), ("avoid people", 5), ("log each
delivery", 3)]

ready = []
for story, points in backlog:
    if points > VELOCITY:
        print("too big, split it:", story)
    else:
        ready.append((story, points))

sprints = []
current, total = [], 0
for story, points in ready:
    if total + points > VELOCITY:
        sprints.append((current, total))
        current, total = [], 0
    current.append(story)
    total += points
if current:
    sprints.append((current, total))
```

```
for n, (stories, total) in enumerate(sprints, start=1):
    print(f"sprint {n}: {', '.join(stories)} ({total} points)")
print("sprints needed:", len(sprints))
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.