1.2 Types of Software and Development Methodologies: 1-hour Pack

Total Marks

OCR H446 · Calculators not allowed · about 60 minutes

Name: _____ Date: ____ / ____ / _____

Answer ALL questions. Show all your working.

1 Agile methodologies are an alternative to traditional models such as waterfall.

(a) State three principles that underpin agile approaches to software development, as set out in the agile manifesto. **(3)**

(b) Explain two ways in which agile development handles changing requirements differently from the waterfall model. **(4)**

(c) A government department is planning a large IT project with a fixed contract value and a fixed delivery date set in law. Discuss whether an agile methodology would be appropriate for this project. **(6)**

(Total for Question 1 is 13 marks)

© Revision Library | revisionlibrary.co.uk

2 The spiral model combines iterative development with a strong focus on risk management.

(a) Describe the four phases carried out in each loop (cycle) of the spiral model. **(4)**

(b) Explain why the spiral model is well suited to large, high-risk projects. **(3)**

(c) Explain one disadvantage of using the spiral model. **(2)**

(Total for Question 2 is 9 marks)

---

3 Extreme Programming (XP) is an agile methodology that emphasises technical practices.

(a) State three practices associated with Extreme Programming. **(3)**

(b) Explain one benefit of pair programming. **(3)**

(c) Explain one benefit of using continuous integration in a development team. **(3)**

(Total for Question 3 is 9 marks)

© Revision Library | revisionlibrary.co.uk

4 A start-up company is developing a mobile app for online banking. The app must be launched within 4 months to meet a marketing deadline, but the founders expect features to change as user feedback comes in after each release. Security and regulatory compliance are critical.

(a) Recommend and justify a suitable software development methodology for this project. **(8)**

(Total for Question 4 is 8 marks)

© Revision Library | revisionlibrary.co.uk

5 A small independent bookshop wants a new till (point-of-sale) and stock management system. The owner has a very limited budget and standard requirements similar to many other small shops, but wants some control over any future changes to the software.

(a) Discuss which combination of software type (bespoke vs off-the-shelf) and licensing model (open source vs closed source) would best meet the bookshop's needs, and justify your recommendation. **(8)**

(Total for Question 5 is 8 marks)

---

6 Evaluate the extent to which agile methodologies have made the waterfall model obsolete for modern software development.

(Total for Question 6 is 9 marks)

© Revision Library | revisionlibrary.co.uk

Mark scheme · 1.2 Types of Software and Development Methodologies: 1-hour Pack

Question 1

(a) One correct agile principle, from: individuals and interactions valued over processes and tools; working software valued over comprehensive documentation; customer collaboration valued over contract negotiation; responding to change valued over following a fixed plan; iterative/incremental delivery in short cycles (sprints); continuous customer/user feedback	**B1**
(a) A second, different correct agile principle from the same list	**B1**
(a) A third, different correct agile principle from the same list. Maximum 3 marks; do not credit the same principle twice	**B1**

(a) Answer: e.g. working software is valued over comprehensive documentation; customer collaboration is valued over contract negotiation; responding to change is valued over following a fixed plan.

(b) Point 1: agile develops software in short iterations/sprints, each delivering a working increment (oe)	**B1**
(b) Development of point 1: so requirements can be reviewed and adjusted between sprints without restarting the whole project, unlike waterfall (oe, ft)	**B1**
(b) Point 2: agile involves continuous/frequent feedback from the customer/user throughout development (oe)	**B1**
(b) Development of point 2: so changes can be incorporated as they arise, rather than (as in waterfall) only being possible before design begins (oe, ft)	**B1**

(b) Answer: Agile develops software in short sprints, each producing a working increment, so requirements can be reviewed and changed between sprints without restarting the project; agile also involves continuous customer feedback throughout, so changes can be incorporated as they arise, unlike waterfall where requirements are effectively fixed once the design stage begins.

(c) **L3** A well-balanced discussion that applies agile's characteristics directly to the fixed-price, fixed-deadline, legally constrained context, weighing genuine advantages and disadvantages, and reaching a reasoned, justified conclusion.	**5-6**
(c) **L2** Some balanced discussion of agile's suitability, but with limited application to the fixed contract/legal deadline context; conclusion may be weak or generic.	**3-4**
(c) **L1** Basic or one-sided statements about agile with little or no application to the scenario.	**1-2**
(c) **L0** No creditable content.	**0**

(c) Indicative content:

- Agile's flexible/evolving scope is hard to reconcile with a fixed-price contract, since the final requirements and total cost cannot be fully fixed upfront, which could lead to disputes over what has been delivered.
- A fixed legal deadline could still be met by fixing the time and cost (time-boxing sprints) and allowing scope/features to flex, delivering the highest priority features first (a 'fixed time, fixed cost, variable scope' approach).
- Agile's frequent releases and feedback can reduce the risk of building the wrong system, which is valuable in a large project.
- However, a project needing full upfront agreement, sign-off and heavy documentation for legal/audit purposes may be better suited to waterfall or a hybrid approach, since agile produces less upfront documentation.
- A reasoned conclusion should weigh the risk of scope/cost disputes against agile's ability to deliver a prioritised, working system by a fixed date.

(c) Answer: Model answer: agile could be made to work within a fixed contract by time-boxing sprints and letting scope flex (delivering the highest-priority features first to meet the legal deadline), but its evolving scope sits awkwardly with a fixed-price contract and heavy documentation/audit requirements, so a hybrid approach (e.g. fixed requirements and budget agreed upfront, delivered via agile sprints) may be the most balanced recommendation.

© Revision Library | revisionlibrary.co.uk

Question 2

(a) 1. Determine objectives, alternatives and constraints for that iteration (oe) **B1**

(a) 2. Identify and evaluate/resolve risks (risk analysis), often using prototyping (oe) **B1**

(a) 3. Develop and test/verify the next level (increment) of the product for that iteration (oe) **B1**

(a) 4. Review the iteration and plan the next loop (oe) **B1**

(a) Answer: Each loop: (1) determine objectives, alternatives and constraints; (2) identify and analyse/resolve risks, often via prototyping; (3) develop and test the next increment of the product; (4) review the results and plan the next loop.

(b) Each loop/iteration includes an explicit risk analysis phase, often using prototypes (oe) **B1**

(b) This allows risks to be identified and addressed early, before large amounts of time/money have been committed (oe, ft) **B1**

(b) Reducing the chance of costly failure later in a large, unfamiliar or technically complex project (oe, ft) **B1**

(b) Answer: Because every loop of the spiral includes a dedicated risk-analysis phase, often supported by prototyping, risks are identified and dealt with early, before large amounts of time and money have been committed, reducing the chance of costly failure later in a large or technically complex project.

(c) Requires specialist risk-assessment expertise/skills to be carried out properly (oe) **B1**

(c) It can be more expensive and time-consuming than simpler models due to the repeated risk analysis and prototyping, which may not suit smaller or low-risk projects (oe; any distinct valid disadvantage acceptable) **B1**

(c) Answer: The spiral model requires specialist risk-assessment expertise and repeated prototyping/analysis in every loop, making it more expensive and time-consuming than simpler models, which is not justified for smaller or low-risk projects.

Question 3

(a) One correct XP practice, from: pair programming; test-driven development (writing tests before code); continuous integration; refactoring; collective code ownership; small/frequent releases; on-site customer; simple design; coding standards **B1**

(a) A second, different correct XP practice from the same list **B1**

(a) A third, different correct XP practice from the same list. Maximum 3 marks; do not credit the same practice twice **B1**

(a) Answer: e.g. pair programming, test-driven development, continuous integration.

(b) Description: two developers work together at one workstation, one writing code (driver) while the other reviews it in real time (navigator) (oe) **B1**

(b) Errors/bugs are spotted and corrected earlier, as code is reviewed as it is written (oe) **B1**

(b) Knowledge of the codebase is shared between developers, reducing dependency on a single person and aiding training (oe; any one further distinct benefit acceptable) **B1**

(b) Answer: In pair programming, two developers share one workstation, with one writing code (driver) and the other reviewing it as it is written (navigator). Errors are caught earlier because the code is reviewed continuously, and knowledge of the codebase is shared between developers, reducing reliance on any one individual.

(c) Description: code changes are merged into a shared repository and automatically built/tested frequently, e.g. several times a day (oe) **B1**

(c) Integration problems/conflicts between different developers' code are identified quickly, while they are small and easy to fix (oe) **B1**

(c) This reduces the risk of large, difficult-to-diagnose integration problems occurring near the end of the project (oe) **B1**

(c) Answer: Continuous integration merges developers' code into a shared repository and automatically builds/tests it frequently. This means integration problems between different developers' code are found quickly while they are small, reducing the risk of large, difficult integration problems appearing near the end of the project.

© Revision Library | revisionlibrary.co.uk

Question 4

(a) **L4** A clear, justified recommendation that explicitly applies methodology characteristics to multiple scenario constraints (tight deadline, evolving features, security/regulatory needs), with balanced consideration of at least one alternative and a coherent, well-structured argument. **7-8**

(a) **L3** A reasonable recommendation with justification linked to most scenario factors, but with less developed evaluation of alternatives. **5-6**

(a) **L2** Some relevant points made with partial application to the scenario; justification is limited or one-sided. **3-4**

(a) **L1** Basic or generic statements about a methodology with little or no application to the scenario. **1-2**

(a) **L0** No creditable content. **0**

(a) Indicative content:

- An agile approach (e.g. Scrum) is likely most appropriate because features are expected to change based on user feedback, and agile's iterative sprints allow the app to be adapted after each release.
- Short sprints could deliver a minimum viable product within the 4 month deadline, with regular releases allowing early and continuous testing.
- However, security and regulatory compliance requirements are often non-negotiable/fixed, so these may need to be gathered and specified up front, and/or validated through a formal, dedicated review stage, suggesting a hybrid approach (agile development combined with a fixed security/compliance specification and review process) rather than pure agile.
- Waterfall is less suitable since it does not easily accommodate the expected changes to features, and risks missing the fixed 4 month deadline if all requirements must be fully fixed upfront.
- RAD could also be considered given the tight deadline and reliance on prototyping, provided users are available to give regular feedback; however it may not provide the rigorous documentation needed for regulatory compliance.
- A well-argued answer might recommend Scrum with an embedded, non-negotiable security/compliance review at each sprint, or a hybrid model combining agile delivery with waterfall-style compliance sign-off.

(a) Answer: Model answer: recommend an agile (e.g. Scrum) approach for the evolving features and tight 4 month deadline, combined with a fixed, non-negotiable security/regulatory specification validated at each sprint (a hybrid model), since pure waterfall would not meet the deadline or accommodate expected changes, and pure agile alone risks under-specifying compliance requirements.

Question 5

(a) **L4** A clear, well-justified recommendation combining a software type and licensing model, with detailed application to the bookshop's budget, standard requirements and desire for future control, and balanced consideration of trade-offs. **7-8**

(a) **L3** A reasonable recommendation linked to most scenario factors, with some consideration of trade-offs, but less fully developed. **5-6**

(a) **L2** Some relevant points with partial application to the scenario; justification limited or one-sided. **3-4**

(a) **L1** Basic or generic statements about software types/licensing with little or no application to the scenario. **1-2**

(a) **L0** No creditable content. **0**

(a) Indicative content:

- Off-the-shelf is likely appropriate given the limited budget and standard/typical requirements shared with other small shops, avoiding the high cost of bespoke development.
- Open source licensing could reduce ongoing licence costs and, since the source code is available, could give the owner (or a hired developer) some ability to make or commission future changes without being tied to a single vendor, addressing the desire for control.
- However, open source solutions may need technical expertise to adapt or maintain, which could add cost the owner did not expect.
- Closed source, off-the-shelf point-of-sale packages are common, cheaper up front than bespoke, well tested and supported, but the owner would have no direct control over future changes and would depend on the vendor's roadmap and licence terms.
- A balanced answer might recommend an open source, off-the-shelf EPOS (electronic point of sale) package as a compromise: low cost, standard functionality already built in, and some flexibility for future changes, while noting the trade-off in guaranteed technical support.

(a) Answer: Model answer: recommend an open source, off-the-shelf point-of-sale package, since off-the-shelf suits the limited budget and standard requirements, while the open source licence gives the owner some ability to adapt the software in future without being locked to one vendor; the main trade-off is reduced guaranteed technical support compared with a closed-source commercial product.

© Revision Library | revisionlibrary.co.uk

Question 6

Level 3 A sustained, well-evidenced evaluation considering multiple project contexts (e.g. safety-critical/regulated systems vs consumer apps), with a balanced argument on both sides and a clear, well-supported overall judgement. Technical vocabulary used accurately throughout.	**7-9**
Level 2 Some evaluation with valid points on both sides, applied to at least one context; an overall judgement is present but less developed or not fully consistent.	**4-6**
Level 1 Mostly one-sided or a generic list of features of agile and/or waterfall with little evaluative comment; a weak or absent judgement.	**1-3**
Level 0 No creditable content.	**0**

Indicative content:

- Case for agile making waterfall obsolete: agile's flexibility better matches fast-changing markets and evolving user expectations; iterative delivery gets working software to users sooner and incorporates feedback, reducing the risk of building the wrong product; agile fits well with modern continuous deployment/DevOps practices.
- Case against: waterfall's clear documentation, fixed scope/cost/schedule and formal sign-off stages remain valuable, or required, for safety-critical, highly regulated, or contractually fixed-price projects (e.g. medical devices, government/defence systems, embedded firmware) where requirements are well understood in advance and cannot change late in development.
- Some organisations lack the culture or skills for agile, or need extensive audit trails that waterfall's documentation-heavy stages naturally provide.
- Hybrid models (e.g. spiral, or agile combined with staged formal reviews) are often used in practice instead of either pure model, showing that the choice is not simply binary.
- Overall judgement: agile has become dominant in many sectors (e.g. web/app/consumer software) but waterfall, or waterfall-like staged approaches, persists and remains appropriate for certain project types, so it is not fully obsolete; the correct choice depends on the nature of the project rather than one model being universally superior.

© Revision Library | revisionlibrary.co.uk