



University of Colombo, Sri Lanka

University of Colombo School of Computing



**DEGREE OF BACHELOR OF INFORMATION TECHNOLOGY
(EXTERNAL)**

Academic Year 2024— 2nd Year Examination — Semester 4

IT4406 — Agile Software Development

Part 2 - Structured Question Paper

(ONE HOUR)

To be completed by the candidate

Index Number

--	--	--	--	--	--	--

Important Instructions

- The duration of the paper is **ONE HOUR**.
- The medium of instructions and questions is English.
- This paper has **2** questions and **8** pages.
- Answer both questions. Both questions carry **equal** marks.
- **Write your answers in English** using the space provided **in this question paper**.
- Do not tear off any part of this answer book.
- Under no circumstances may this book (or any part of this book), used or unused, be removed from the Examination Hall by a candidate.
- Questions appear on both sides of the paper. If a page is not printed, please inform the supervisor immediately.
- All kinds of electronic devices, including calculators, are **not allowed**.
- *All rights reserved.*

**To be completed by
the examiners**

1	
2	

Index Number

--	--	--	--	--	--	--

1. A company is a few months into adopting Scrum and is facing difficulties. The Product Owner is a busy senior manager who attends few of the team's events, leaves the Product Backlog in no specific order, and changes priorities without explanation. All effort estimates are produced by a single senior developer. Items pulled into a Sprint are vague and have no acceptance criteria.

(a). The Product Owner's behavior is contributing to the team's problems.

(i) Explain **three** (3) principal responsibilities of the Product Owner in Scrum (9 marks)

(ii) State **two** (2) characteristics or skills an effective Product Owner must have, and explain why **each** matters for the team described above (6 marks).

[15 marks]

(i) **Three principal responsibilities (3 marks each):**

- Owns and orders (prioritises) the Product Backlog so the most valuable work is done first; the backlog being left unordered is a direct failure of this duty.
- Defines and communicates the product vision and decides what the product will contain (scope and priority), so the team and stakeholders share one direction.
- Defines acceptance criteria for items and accepts or rejects completed work; manages product economics to maximise value/ROI.

(ii) **Two characteristics, with relevance (3 marks each: 1 characteristic, 2 explanation):**

- *Available and engaged.* The team needs timely answers and decisions; a Product Owner who attends few events (as here) blocks the team and lets vague items slip through.
- *Empowered and decisive.* An effective Product Owner must have the authority to set priorities and make product decisions stick, otherwise priorities clash and the team cannot rely on the plan.

(Also accepted: domain/business knowledge, good communicator/collaborator etc.)

Marks: (i) $3 \times 3 = 9$; (ii) $3 \times 2 = 6$.

Index Number

--	--	--	--	--	--	--

(b). Currently, a single senior developer produces all estimates.

(i) Describe a collaborative estimation technique that the team could use instead. Include the **specific steps** involved in applying the technique to this project (*12 marks*).

(ii) Explain **two** (2) reasons why estimating collaboratively is beneficial for this project (*6 marks*).

[18 marks]

(i) Marks given for any valid collaborative technique correctly described with steps. Example answer: *Planning Poker*.

- Each estimator has a deck of cards with a limited scale of values (often a modified Fibonacci sequence such as 1, 2, 3, 5, 8, 13, 20).
- A Product Backlog Item is presented and the team discusses it briefly to clarify scope.
- Each estimator privately selects a card and all reveal simultaneously, so no one is anchored by hearing others first.
- If estimates differ widely, the highest and lowest estimators explain their reasoning, the team discusses, and re-votes; this repeats until the estimates converge.

(ii) Two reasons collaborative estimation is better:

- It draws on the whole team's knowledge: the people who will actually do the work all contribute, surfacing hidden complexity and missing information through discussion.
- It reduces anchoring/authority bias (e.g. simultaneous reveal) and builds shared understanding and team buy-in, typically giving more reliable estimates than one person's guess.

Marks: (i) technique named and described 12; (ii) 2 benefits: $2 \times 3 = 6$.

--	--	--	--	--	--	--

- (c). Items pulled into a Sprint are often vague and lack acceptance criteria.
- (i) Explain what a *Definition of Ready* (DoR) is and where it fits in this project (6 marks).
 - (ii) Give **two** (2) example DoR criteria suitable for this team (4 marks).
 - (iii) Explain how a DoR would help with the problems described, and identify **one** (1) risk or limitation of relying on a DoR, with examples (7 marks).

[17 marks]

(i) **What a DoR is, and where it fits (6 marks).**

- An agreed set of entry criteria that a Product Backlog Item must satisfy before the team will bring it into a Sprint during Sprint Planning.
- It sits at the boundary of backlog refinement and Sprint Planning: it is effectively the exit gate of refinement and the entry gate to a Sprint, ensuring only well-understood, actionable items are started.
- It is a shared agreement owned by the whole team (not imposed by one person), so everyone applies the same bar.

(ii) **Two example criteria (2 marks each).** For example: the item is clearly described and understood by the team; it has acceptance criteria defined; it has been estimated/sized; it is small enough to fit in one Sprint; its dependencies are identified or resolved. (*Any two suitable criteria.*)

(iii) **How it helps the team (4 marks), and one risk/limitation (3 marks).**

- It directly blocks the team's core problem: vague items with no acceptance criteria fail the DoR, so they cannot enter a Sprint until refined.
- The team therefore commits only to items it genuinely understands, reducing mid-Sprint surprises, rework, and failed Sprint commitments.
- It forces the disorganised backlog to be refined and clarified upstream, improving predictability and protecting the Sprint Goal.

Risk/limitation (any one valid answer, with example):

- The DoR can become a rigid stage-gate that reintroduces mini-waterfall behaviour — e.g. the team refuses to start any item that is not “perfectly” specified, delaying valuable work and discouraging collaborative refinement during the Sprint.
- Alternatively, it may be applied mechanically as a checklist — e.g. ticking “acceptance criteria exist” without genuine shared understanding, so vague items still slip through in substance.

Marks: (i) 6 (definition 3, placement as refinement-exit/Sprint-entry gate 2, team-owned agreement 1); (ii) $2 \times 2 = 4$; (iii) 7 (how it helps: connects DoR to vague/criteria-less items 2, fewer surprises/rework/-better predictability 2; one valid risk/limitation 2, with a suitable example 1).

--	--	--	--	--	--	--

2. An e-commerce company is building a new *digital-payments product* in a market with high uncertainty. The requirements for this product are expected to evolve as the market and customer needs become clearer. However, the company’s leadership wants this development aligned with the organization’s other product initiatives and wants the development team to produce a release plan while still delivering value early.

- (a). Given the high uncertainty, explain how **each** of the following Agile/Lean principles helps this development team, with an example for each: (i) Embracing variability and uncertainty; (ii) Validated learning; (iii) Limiting Work-In-Process (*4 marks each*).

[12 marks]

- (i) *Embracing variability and uncertainty.* Agile treats variability and uncertainty as inherent and potentially beneficial, to be reduced through iteration and feedback rather than eliminated by a fixed up-front specification; it distinguishes market/requirements (“end”) uncertainty from technical (“means”) uncertainty and keeps options open. *Example:* rather than fully specifying all payment features now, the team builds a thin slice (e.g. one payment method), learns from real usage and regulatory feedback, and adapts the next features.
- (ii) *Validated learning.* Important assumptions are tested quickly with real evidence (build–measure–learn); each increment is an experiment whose feedback validates or invalidates an assumption before further investment, reducing the waste of building unwanted features. *Example:* release a minimal payment flow to a subset of users, measure completion/adoption, and use the data to decide whether to expand or pivot.
- (iii) *Limiting WIP.* Capping partially-done work reduces the carrying cost of inventory, shortens feedback loops, lowers the cost of change, and exposes bottlenecks; finishing before starting new work improves flow and delivers value sooner. *Example:* the team drives a small number of payment features to “done/releasable” instead of starting many at once, so value ships earlier and defects surface sooner.

Marks: $4 \times 3 = 12$. Per principle: 2 for a correct explanation of the principle, 2 for a relevant, concrete payments-product example.

Index Number

--	--	--	--	--	--	--

- (b). Leadership wants the payments product scheduled appropriately among the organization's other initiatives. Describe **two** (2) scheduling strategies used in portfolio planning (one **in-flow** strategy and one **outflow** strategy) with a brief example of how **each** helps manage the flow of initiatives and improve time-to-market (*6 marks each*). **[12 marks]**

- **Inflow strategy** (controls how/when new initiatives enter the portfolio). Governs admission so the system is not overloaded — e.g. apply an economic filter so only initiatives above a value threshold enter, admit work on a regular cadence, or start new work only when capacity frees up (portfolio WIP limit). *Effect*: prevents over-commitment and keeps queues short, so initiatives that are started flow through faster, improving time-to-market. *Example*: the payments product is admitted only when an in-flight initiative completes and frees capacity.
- **Outflow strategy** (controls how initiatives complete and leave). Governs getting work finished and out — e.g. “finish what you start” (favour completing in-flight initiatives over starting new ones) and favour small, frequent releases. *Effect*: reduces portfolio WIP and accelerates ROI/time-to-market. *Example*: ship the payments product in small releases so value is delivered and the initiative exits the portfolio sooner.

Marks: $6 \times 2 = 12$. Per strategy: 3 for a correct description that matches its category (inflow = entry, outflow = completion/exit), 3 for a sensible example linked to flow/time-to-market. Any valid strategy that correctly fits the function is accepted; require one inflow and one outflow (capped if both are the same type).

--	--	--	--	--	--	--

(c). The payments product's first release has a committed scope: a set of regulatory features that must all be present.

(i) Outline **four** (4) key steps in performing fixed-scope release planning (8 marks).

(ii) Explain how a fixed-scope approach forecasts the likely release date, and the role velocity plays in that forecast (8 marks).

[16 marks]

(i) **Four key steps (2 marks each; any four valid steps reflecting the fixed-scope logic is accepted):**

- Groom and estimate the full committed scope — ensure all required PBIs (including the regulatory features) are in the backlog, refined, and sized (e.g. story points).
- Establish the team's velocity as a range (from history or a forecast for a new team).
- Compute the number of sprints needed = total estimated size ÷ velocity per sprint.
- Convert sprints to a calendar date (sprints → sprint length, plus a buffer) and re-forecast each sprint as actual velocity emerges; communicate the forecast.

(Other acceptable steps: refine MRFs, identify dependencies/risks, calculate cost etc.)

(ii) **Forecasting the date and the role of velocity (8 marks):**

- With scope fixed, the unknown is the date: estimate total work, divide by velocity to get the number of sprints, then convert to a date.
- Velocity converts scope into time; using a velocity range yields an earliest–latest date range rather than a single point, and the forecast narrows as real velocity data accrues. *(Contrast: fixed-date planning fixes the date and forecasts scope — the inverse.)*

Marks: (i) $2 \times 4 = 8$; (ii) 8 (4 for the mechanism ; 4 for the role of velocity.)

Index Number

--	--	--	--	--	--	--

- (d). Given the committed regulatory scope described above, state how the three release constraints (scope, date, and cost/budget) interact in release planning. Explain using the payments product as an example.

[10 marks]

- The three release constraints are interdependent: in any release plan, fixing some forces the others to flex, and all three cannot be locked rigidly at once.
- For the payments product, scope is the fixed constraint, since the regulatory features must all be present. The flexibility must therefore come from date and/or cost, for example by adding a sprint to the timeline or adding people (cost) to deliver the committed scope.
- Agile planning usually keeps scope as the variable that flexes (delivering the highest-value features first), but here scope is committed, so the team flexes date and cost instead and uses velocity to forecast the resulting release date.

Marks (10): 5 for correctly explaining the interaction (the three constraints are interdependent; fixing some forces others to flex; cannot fix all three). 5 for a correct, concrete payments-product example (scope fixed by the regulatory features ↔ date and/or cost must flex, with a suitable illustration).
