Examiners' commentaries 2015–16

CO3353 Software engineering project management – Zone A

General remarks

Overall performance

The overall standard of answers to this examination paper was average, clustering around the half-mark level. Candidates could choose three questions out of a selection of five – almost everyone chose Questions 3 and 4. The next most popular was Question 2, meaning that the combination of Questions 2, 3 and 4 was very common. Less than half chose Question 1 and only a very few chose Question 5. Percentages answering each question were:

Question 1	Planning and scheduling	42%
Question 2	Risk management	79%
Question 3	Project execution	83%
Question 4	Testing	83%
Question 5	Process improvement	13%

Marks for Questions 2, 3 and 4 were very similar and were quite a bit higher than those for other questions. Marks for Question 1 were rather lower than expected. Questions in order of marks achieved are:

Question 2	Risk management	Highest
Question 3	Project execution	Highest
Question 4	Testing	Highest
Question 5	Process improvement	Next lowest
Question 1	Planning and scheduling	Lowest

Examination questions: general remarks

All questions were on a single independent topic: there was no mixing of different topics within a single question. All questions were broken up into between two and three parts. In some cases, a part was further broken down into subsections. The marks allocated to each part were clearly indicated. The marks for a subsection could be calculated by dividing the mark for that part by the number of distinct points candidates were asked to identify. Marking was carried out strictly in accordance with this scheme. If a part or subsection was not answered, no marks were given for it. Credit was not given for excessive answers to one part at the expense of others. Answering only half the question would attract only half the available marks. No candidates answered more than the required number of questions.

It is important that candidates take note of the following points since not doing so means that questions are not answered as well as they might be.

- Ensure that you fully understand the topic area of the question.
- Ensure that you can answer every part and subsection of the question. Only being able to answer some of the question will not help you achieve a good overall mark.

- Ensure that the level and detail of the answer you give corresponds to the marks allocated to that part. Do not spend too much time and effort on a subsection of the question that, say, is worth only 5 per cent of the overall answer. Similarly, do not merely write cryptic notes or single points for a part of the question that is worth, say, 30 per cent. Try to achieve the balance reflected in the marks indicated.
- Read the question carefully and answer in the way that is requested: wording such as 'describe', 'compare and contrast', 'itemise', 'illustrate', 'explain with diagrams' tells you what sort of answer is expected and what sort of detail you should go into. Make sure you understand what type of answer is expected and do provide diagrams or examples where requested since this is part of the marking scheme for the question.
- Do not spend unnecessary time restating the question, either in your own words or in repeating the question text. This is not required and wastes valuable time.
- Do not spend excessive time answering one question at the expense of others: it is generally better to answer three questions fully than one in great detail and two very briefly.
- Do not spend time providing unnecessary diagrams where this is not explicitly required. If diagrams are called for, they should be clearly labelled and described in full with suitable annotations. Providing as an answer only an unlabelled diagram from memory will not attract good marks.
- Do not repeat details from one part of the question in another part: it is unlikely that this is what the Examiner intended and the focus of your answer in each part should be quite distinct.
- Do try to use tables and lists where appropriate – for example, in a question which asks you to contrast two approaches or itemise the differences between two aspects of a topic.

Comments on specific questions

Question 1

This question, on **Planning and scheduling**, was divided into two parts, based on Chapter 3 (pp.35–44) of the CO3353 subject guide. Part (a) was further subdivided into three parts, each on a specific activity. The question was about understanding the relationships between deliverables (outputs, or products) and the tasks needed to produce them, focusing on the differences between task-based and product-based project planning.

Candidates did best on Parts (ai), (aii) and (b) of this question and rather less well on Part (aiii), where some candidates did not answer the question at all. A good answer to this question would cover the following:

- For Part (ai), which concerned the differences between product-based and task-based approaches to scheduling and the purpose of resource-allocation (i.e. to enable control of budget), resources in a product-based approach are allocated to items in a Product Breakdown Structure (PBS). This focuses on completion of a 'shopping list' of products, whereas a task-based approach uses a Work Breakdown Structure which typically decomposes into a finer-grained set of sub-tasks.
- For Part (aii), the focus was on progress-monitoring and how to keep track of time, effort and quality. The product-based approach to

monitoring is achievement-based (so resources can be reallocated once a checkpoint has been reached), whereas a task-based approach is calendar- or time-driven, making estimates less robust.

For Part (aiii), how to manage interdependencies, the purpose is to reduce delays when a team is waiting for some input or resource. The product-based approach typically encapsulates the activities needed to deliver a product (allowing those activities to be delegated within the team responsible for the product), whereas the task-based approach is likely to create multiple interdependencies between different specialist teams. Product-based dependencies arise when one product is needed to build another product. Task-based dependencies occur when a resource is needed to work on two tasks, one after the other.

For Part (b), which asked about tools to document and manage a schedule, two main tools in common use are GANTT charts and MS Project. A GANTT chart shows the order in which products or activities are scheduled in time and can show the dependencies between activities or between products. Note that a PBS does not explain how products decompose over time; that is done using a Product Flow Diagram (PFD), which is a hierarchical representation of the project plan. Software such as MS Project can track progress using three elements (time, task and work or effort) to assist in planning. Simple static GANTT charts can be produced using a graphics package or even a spreadsheet.

Question 2

This question, on Risk management, was divided into two parts, based on Chapter 4 (pp.45–54) of the CO3353 subject guide. The question was about the difference between risks and issues and, in the second part, required a good description of the iterative risk management cycle.

Candidates did well in this question but achieved better marks for Part (a) than for Part (b), where some candidates did not answer the question fully. A good answer to this question would cover the following:

- Part (a) concerned the differences between a project risk and a project issue. A project risk is an external or uncontrollable event (opportunity or threat) that may affect the outcome of the project if it happens. The risk management plan will include ways of reducing the likelihood of it happening and/or reducing the impact. A project issue is an event that has already transpired/materialised. That is, it is now internal and controllable and the project plan must be adjusted to accommodate this exception to the plan. In addition, the risk management plan is monitored and reviewed continually as part of the project management plan.
- Part (b) asked for a full description of the risk management workflow. The steps are 'Identify -> Analyse -> Plan -> Track -> Control'. This is a more complete description than 'Mitigate / Monitor / Manage' and the question actually uses the terms 'identification' and 'control'. The activities in these five iterated steps are:
    1. Identify: Search for and locate risks.
    2. Analyse: Evaluate the impact and likelihood.
    3. Plan: Determine ways of reducing the most important risks.
    4. Track: Monitor the risk indicators and any actions taken against risks.
    5. Control: Correct for deviations from planned risk.

Question 3

This question, on **Project execution**, was divided into five subparts based on Chapter 6 (pp.63–72) of the CO3353 subject guide. The question concerned five areas of inter-related responsibility for a project manager (PM) and the scope of the role itself.

Candidates did best on Part (ai) of this question and achieved very similar marks for all the remaining sections of Part (a).

A good answer to this question would cover the following:

- For Part (ai), on the use of resources, a PM must execute a resource plan (e.g. use of people, facilities) as part of the project plan. That plan requires to be reviewed and updated, depending on the link between resources, cost and time. In addition, there is a need to monitor interdependencies, delays, availability and other scheduling problems.
- For Part (aii), on delegating work to team members, a PM must assign work packages with agreed time, effort and cost parameters, allowing for negotiated tolerances and having a formal reporting process (e.g. checkpoint reviews).
- For Part (aiii), on monitoring progress, a PM must gather data on project cost, schedule, technical and quality progress, identify any deviation from the project plan and update all plans accordingly. There is also a distinction to be made between exception-based reporting and periodic reporting.
- For Part (aiv), on managing risks, a PM must anticipate and respond to risks and potential issues with a risk management plan and a mechanism of how to assess and prioritise risks (likelihood and impact).
- For Part (av), on controlling changes, a PM must respond to and evaluate change requests via a formal process and adapt the project plan, scope and environment where necessary. Note that there are two types of change: 'request for change' (i.e. the client changes mind) and 'off-spec' (which may be a bug).

Question 4

This question, on **Testing**, was divided into three parts, based on Chapter 8 (pp.84–86) of the CO3353 subject guide. The question was about how to relate 'bottom up' testing to the software engineering process and, further, about an appreciation of the differences between 'verification' (compliance with developer's technical specification) and 'validation' (compliance with customer business requirement).

Candidates again did well in this question but achieved better marks for Part (a) than for Parts (b) and (c), where some candidates did not answer the question at all. A good answer to this question would cover the following:

- Parts (a) and (b) were on the four main forms of software testing, as below, with examples of the specific issues that have to be addressed being identified:
    1. Unit testing: Verify the behaviour of the functions of individual classes or methods for the application by testing them with a complete range of possible inputs. Those classes or methods have to be tested before they can be assembled into components of the system.

2. Component testing: Verify the behaviour of fully-tested units when they interact with each other. What is being tested is the interaction between the elements in a component, not the behaviour of the component as part of the system. In essence, component testing assesses compliance with the technical design, not the system requirement specification.
3. System testing: Verify the behaviour of the complete system by evaluating whether the requirement specification has been achieved. This should include identifying functional and nonfunctional requirements and the need for a method of evaluating nonfunctional requirements.
4. Acceptance testing: Validate the compliance of the complete, tested, system from the customer's perspective. By this stage, the system has already been fully tested technically and the customer is responsible for the validation.

- Part (c) was about the scope and content of a typical test plan:
  1. Unit testing requires that each unit should be exhaustively tested by invoking each function or method that it provides using the range of values of each parameter or attribute those functions require (including invalid and missing values) and the function results for each state change that those parameters cause.
  2. Component testing potentially involves a very large number of discrete combinations which can be simplified using techniques such as boundary case testing, equivalence partitioning and orthogonal array testing.
  3. System testing can often focus on the original use cases, scenarios or user stories, since each will identify one or more functional requirements. Executing tests based on the entire use case model should exercise each requirement at least once.
  4. Acceptance testing by the customer should follow a plan that was established at the time of drawing up a contract, since that is the only mechanism that a supplier and customer have of determining when the work is finished.

Question 5

This question, on Process improvement, was divided into two parts, based on Chapter 9 (pp.94–96) of the CO3353 subject guide. The question concerned process improvement as an integral part of the engineering lifecycle, and not an occasional or optional event.

Candidates did poorly in this question, but with better marks by far for Part (a) than for Part (b).

A good answer to this question would cover the following:

- Part (a) required a description of the Plan Do Check Act (PDCA) process improvement methodology. PDCA is cyclic and continual. For each process to be considered for improvement, which will have a predicted value to the business, a trial is conducted and the change adopted (typically by building and empowering a team to deliver the change) if the evaluation is positive:
  1. PLAN: Identify ways of improving existing process.
  2. DO: Select a single project for trial, perhaps using a pre-existing template to trial change on a small scale.
  3. CHECK: Has the change worked? Is the process stable?

4. ACT: Gain benefit from change by rolling the new process out across more projects.

- Part (b) asked how PDCA could be used to analyse and mitigate a problem such as reduction of incomplete or inconsistent requirements where the proposed mitigation is to introduce a validation methodology. The process under investigation is requirements capture and the problem being addressed is inconsistency of requirements:
  1. PLAN: This should include the explanation of how the decision to try a new validation procedure was made, including details of what viable solutions could be found, what criteria were used to select the chosen solution, how the approach was selected and how it will be evaluated.
  2. DO: A suitable project is nominated for the trial, one which has not yet started its requirement capture. Ideally, this should be a project whose manager has had positive past experience of PDCA.
  3. CHECK: Identify what was the outcome of the evaluation. If the result was not positive, then will the organisation select another approach and repeat the assessment?
  4. ACT: A change team is appointed and the new methodology implemented in other development teams. Actions should include documenting and disseminating the outcome of the trial.

Examiners' commentaries 2015–16

CO3353 Software engineering project management – Zone B

General remarks

Overall performance

The overall standard of answers to this examination paper was disappointing. Candidates could choose three questions out of a selection of five – almost everyone chose Question 3, the next most popular being Question 1, followed by Question 4. Very few chose Question 5. Percentages answering each question were:

Question 1	Milestones and quality gates	85%
Question 2	Plan-based and agile approaches	65%
Question 3	Risk mitigation	96%
Question 4	Design and re-use	40%
Question 5	Cleanroom methodology	13%

Marks for Question 3 were higher than for all other questions. Marks for Question 5 were especially low and clustered around quite a low figure under the half-mark for the remaining questions. Questions in order of marks achieved are:

Question 3	Risk mitigation	Highest
Question 4	Design and re-use	Next highest
Question 2	Plan-based and agile approaches	Next lowest
Question 1	Milestones and quality gates	Next lowest
Question 5	Cleanroom methodology	Lowest

Examination questions: general remarks

All questions were on a single independent topic: there was no mixing of different topics within a single question. All questions were broken up into between two and three parts. In some cases, a part was further broken down into subsections. The marks allocated to each part were clearly indicated. The marks for a subsection could be calculated by dividing the mark for that part by the number of distinct points candidates were asked to identify. Marking was carried out strictly in accordance with this scheme. If a part or subsection was not answered, no marks were given for it. Credit was not given for excessive answers to one part at the expense of others. Answering only half the question would attract only half the available marks.

It is important that candidates take note of the following points since not doing so means that questions are not answered as well as they might be.

- Ensure that you fully understand the topic area of the question.
- Ensure that you can answer every part and subsection of the question. Only being able to answer some of the question will not help you achieve a good overall mark.
- Ensure that the level and detail of the answer you give corresponds to the marks allocated to that part. Do not spend too much time and effort on

a subsection of the question that, say, is worth only 5 per cent of the overall answer. Similarly, do not merely write cryptic notes or single points for a part of the question that is worth, say, 30 per cent. Try to achieve the balance reflected in the marks indicated.

- Read the question carefully and answer in the way that is requested: wording such as 'describe', 'compare and contrast', 'itemise', 'illustrate', 'explain with diagrams' tells you what sort of answer is expected and what sort of detail you should go into. Make sure you understand what type of answer is expected and do provide diagrams or examples where requested since this is part of the marking scheme for the question.
- Do not spend unnecessary time restating the question, either in your own words or in repeating the question text. This is not required and wastes valuable time.
- Do not spend excessive time answering one question at the expense of others: it is generally better to answer three questions fully than one in great detail and two very briefly.
- Do not spend time providing unnecessary diagrams where this is not explicitly required. If diagrams are called for, they should be clearly labelled and described. Providing as an answer only an unlabelled diagram from memory will not attract good marks.
- Do not repeat details from one part of the question in another part: it is unlikely that this is what the Examiner intended and the focus of your answer in each part should be quite distinct.
- Do try to use tables and lists where appropriate – for example, in a question which asks you to contrast two approaches or itemise the differences between two aspects of a topic.

Comments on specific questions

Question 1

This question, on **Milestones and quality gates**, was divided into two parts, based on Chapter 1 (pp.18–20) of the CO3353 subject guide. The question was about understanding the Unified Process (UP) and recognising the purpose of milestones or checkpoints. Part (a) asked for a definition of the milestones associated with each phase of the UP. Part (b) required an explanation of the transition mechanism between phases of a project, together with suitable examples.

Candidates did best on Part (a) and poorly on Part (b) of this question. Examples for Part (b) were often missing or poorly constructed and explained.

A good answer to this question would cover the following:

- For Part (a), the milestones are:
  1. LCO, Lifecycle Objectives – system requirements known.
  2. LCA, Lifecycle Architecture – functional and nonfunctional requirements known.
  3. IOC, Initial Operating Capacity – system ready for customer testing and sign-off.
  4. PR, Product Release – system signed off and ready for rollout.
- For Part (b), the transitions (of which there are only three) are:
  1. Sign-off of system requirements permitting start of software

requirements gathering. This encompasses defining the scope of the system requirements document and explaining how system requirements are used in defining detailed software requirements (e.g. FURPS).

2. Sign-off of analysis of requirements permitting start of system design. This encompasses defining the role of architectural design, and explaining how such architectural design is decomposed (e.g. 4+1 views).
3. Sign-off of system testing and acceptance testing permitting operationalisation. This encompasses defining the scope of unit, component and system testing and the role of the test plan, and explaining the role of acceptance testing and the difference between validation and verification.

Question 2

This question, on **Plan-based and agile approaches**, was divided into three parts, based on Chapter 3 (pp.36–37) of the CO3353 subject guide. The question was testing knowledge and understanding of different types of project and how they can benefit from different development paradigms, such as the two identified in the question. Part (a) concerned the application of agile principles and Part (b) looked for eight attributes of a project that need to be considered before deciding whether to adopt a plan-based or an agile approach. Part (c) expanded on this by asking for an explanation of how each of these attributes affects that decision.

Candidates did best on Parts (a) and (b) and very poorly on Part (c) of this question, which was sometimes just not answered.

A good answer to this question would cover the following:

- For Part (a), applying agile principles to large projects allows the combination of a high-level plan-based approach with each phase being run on an agile basis. Important points to be made are to explain the iterative (e.g. incremental) approach and that iteration only occurs within phases controlled by milestones, not across phase boundaries.
- For Part (b) there are a number of potential responses and credit was given to viable answers. For example, solution-specific constraints (such as complexity of the problem, size of the project team, rigidity of external constraints and need for extended lifetime support) suggest a plan-based approach. Cultural factors (such as the likelihood of rapid feedback and the ability to work flexibly and informally and the use of collaborative development tools by experienced team members) suggest that agile is more appropriate.
- For Part (c) a detailed explanation of the attributes given in Part (b) was expected. Again, for example:
    1. Plan-based would be preferred depending on the complexity of the problem, size of the project team, rigidity of external constraints and need for extended lifetime support.
    2. Agile would be preferred if rapid feedback, flexible working, collaboration through use of tools and the experience (maturity) of the team are all considered as important (and are shared) by the customer and the development team.

Question 3

This question, on **Risk mitigation**, was divided into two parts, based on Chapter 4 (pp.45–53) of the CO3353 subject guide. The question was about risk mitigation strategies and required an understanding of the risk probability/impact matrix. Part (a) asked for a description of how evaluating the probability and impact of a risk helps in selecting a suitable mitigation strategy. The question then asked for examples of four specific different forms of mitigation based on different risks. Part (b) required an explanation of the difference between a known risk and a predictable risk.

Candidates did best on certain sections of Part (a) (on Technology risk and on Estimation risk) and on Part (b) as a whole.

A good answer to this question would cover the following:

- For Part (a), once major risks have been identified, they can be analysed for probability and impact and treated accordingly:
    1. A risk that is unlikely to happen and would cause minimal harm can be monitored in case the classification changes (but not ignored).
    2. A risk that is highly likely and potentially damaging should be avoided or given to a subcontractor who is more likely to control the risk.
    3. Other risks that are likely to have a high impact should be given contingency plans that are only put into operation if the risk transpires.
    4. Other risks that are highly likely to occur should be mitigated throughout the project to reduce the likelihood.
- For specific risks mentioned, some examples may be as in the table below.
- For Part (b) a known risk is a project-specific type of risk associated with a factor such as the technology being used or the timescale involved. Predictable risks are not project-specific and must be reassessed for each project as they are generic. A known risk has a known cause (typically identified in project documentation) but will affect different projects in different ways. There may be a checklist or risk breakdown structure in place.

Risk type	Risk example	Probability/ Impact	Mitigation strategy
Technology	Lack of experience of customer platform: the customer has good IT support function and there is a choice of components.	Probability of an issue is therefore LOW but the impact may be MEDIUM/ HIGH.	Set up joint working with experts from the customer organisation to share the risk and put appropriate contingency plans in place.
Human	Failure to recruit staff: the company has not experienced this before but is large enough to reallocate staff from other teams.	Probability LOW, impact LOW.	Monitor for change in the analysis.

Organisational	Project seen as a distraction from 'core business'.	A relatively HIGH probability of resulting in staff feeling that they are being treated as 'second-class' members of the team but the impact will be LOW unless the problem escalates.	Probability can be reduced by being more open with information rather than letting rumours spread.
Budgeting	Neither the customer nor the supplier is confident about the accuracy of projections for the scalability of some aspect of the project.	The probability of a problem is HIGH, and so is the potential impact.	This is a type of risk that should be avoided. Either obtain a more accurate assessment from an expert or find an alternative way of meeting the customer's needs.

Question 4

This question, on **Design for re-use**, was divided into two parts, based on Chapter 7 (pp.77–78) of the CO3353 subject guide. The question concerned reusability of software and design solutions. Part (a) asked for a description of the benefits of software re-use at four given levels. Part (b) asked for examples of three specific types of cost associated with re-use. The costs associated with evaluating, acquiring and implementing solutions based on each level must be compared to enable the most cost-effective approach, on the basis of the benefits identified in Part (a), to be selected.

Candidates did best sections of Part (a) (on language-specific object libraries and component-based frameworks) and very poorly on Part (b) of this question, where examples were not always supplied.

A good answer to this question would encompass the following:

- For Part (a), and the given levels of software re-use:
    1. Patterns: A problem can often be categorised by genre or pattern (based on the knowledge that specific types of problem have specific types of solution), with high-level descriptions based on past experience of the types of classes required and the ways in which they are organised and interact. Patterns make it easier to define the methods and properties of those classes.
    2. Libraries: These encapsulate functionality for a specific development environment/language and hide the detail of how classes/objects are built, reducing the need for development of specific software for implementing components of the design solution. They demonstrate portability across environments/languages/application types, and reduce the development effort and provision of standard functional building blocks at the object level.
    3. Frameworks: Component-based frameworks support the use of 'black-box' components as part of a design solution. They demonstrate a tested compatibility of components and maintainability and aid speed of development.

4. COTS: Application frameworks and other kinds of off-the-shelf systems (such as ERP) permit rapid development of solutions to standard problems by reconfiguring rather than rebuilding components so the technique is configurable instead of programmable. COTS is designed for standard problems, and for maintainability and access to a large community of users.

- For Part (b), the costs associated with re-using tools or components are: (i) the cost of evaluating them, (ii) the cost of acquiring them and (iii) the cost of configuring and maintaining them.

Question 5

This question, on **Cleanroom methodology**, was divided into two parts, based on Chapter 8 (pp.88–89) of the CO3353 subject guide. The question was about the Cleanroom testing methodology. Part (a) required an explanation of the three types of model used and Part (b) required identification of seven features which assure that the executable programme is consistent with the system specification. Cleanroom is focused on formal verification of products that come from the top-down development lifecycle – functional, architectural and then detailed design specifications – rather than on bottom-up testing (unit, component, system) and this focus was what was required for an answer to this question.

Candidates did very poorly on all parts of this question. It was also the lowest marked of all questions.

A good answer to this question would cover the following:

- For Part (a), as below:
  1. A 'black box' functional specification defines the transition rules required by the increment to produce specific behaviour in response to a sequence of stimuli (external events). This is decomposed until a black sub-box defines a single class of behaviour. This behaviour is either defined as a mathematical function or specified in natural language.
  2. A 'state box' architectural specification defines the black box behaviour in terms of the state-transitions that result from the external events and the data and services that are required to achieve each state-transition. Thus, it encapsulates (completely) the black box it is describing.
  3. A 'clear box' technical design specification defines the procedures or methods that are required to implement the services needed by the state box. Each procedure can be decomposed into clear sub-boxes – ultimately, if necessary, until a sub-box defines one programming construct that is logically indivisible (that is, it contains no conditional statements).
- For Part (b), from the definitions given for Part (a) and given that the black box functional specification defines the transitions in the state of the system that are the responses to external events, the software solution can be considered complete if, for example:
  1. A 'black box' functional specification defines the transition rules required by the increment to produce specific behaviour in response to a sequence of stimuli (external events).
  2. Each sub-box within the black box defines a single class of behaviour (responses to external events).

3. All classes of behaviour have been defined within a black box and verified.
4. Each class of behaviour is formally and completely defined using mathematical functions or natural language.
5. The behaviour of a black sub-box is wholly encapsulated within a corresponding state box.
6. The services required to achieve each black box state-transition have been defined in that state box and verified.
7. The procedures or methods required to implement each state box service have been defined in the clear box technical specifications and verified.
8. Clear box technical specifications are decomposed into logically unconditional programming constructs.