4M21 Software Engineering and Design 2022/2023

Solutions

Version: EP/POK1

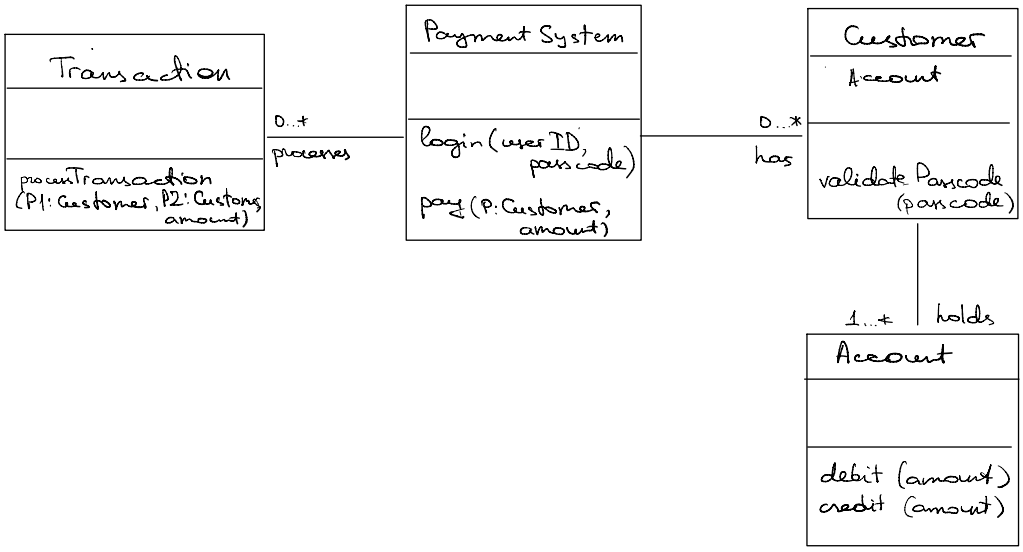
Question 1

a)

Polymorphism is one of the key concepts of object orientation. It separates declaration of functionality from specifics of its implementation. Polymorphism allows us to reduce coupling, increase reusability, and make our code easier to read. A "toy' example could be implementing different characters of a game such as duck, cat, dog yet not having to differentiate between them if we would like an animal to "speak".

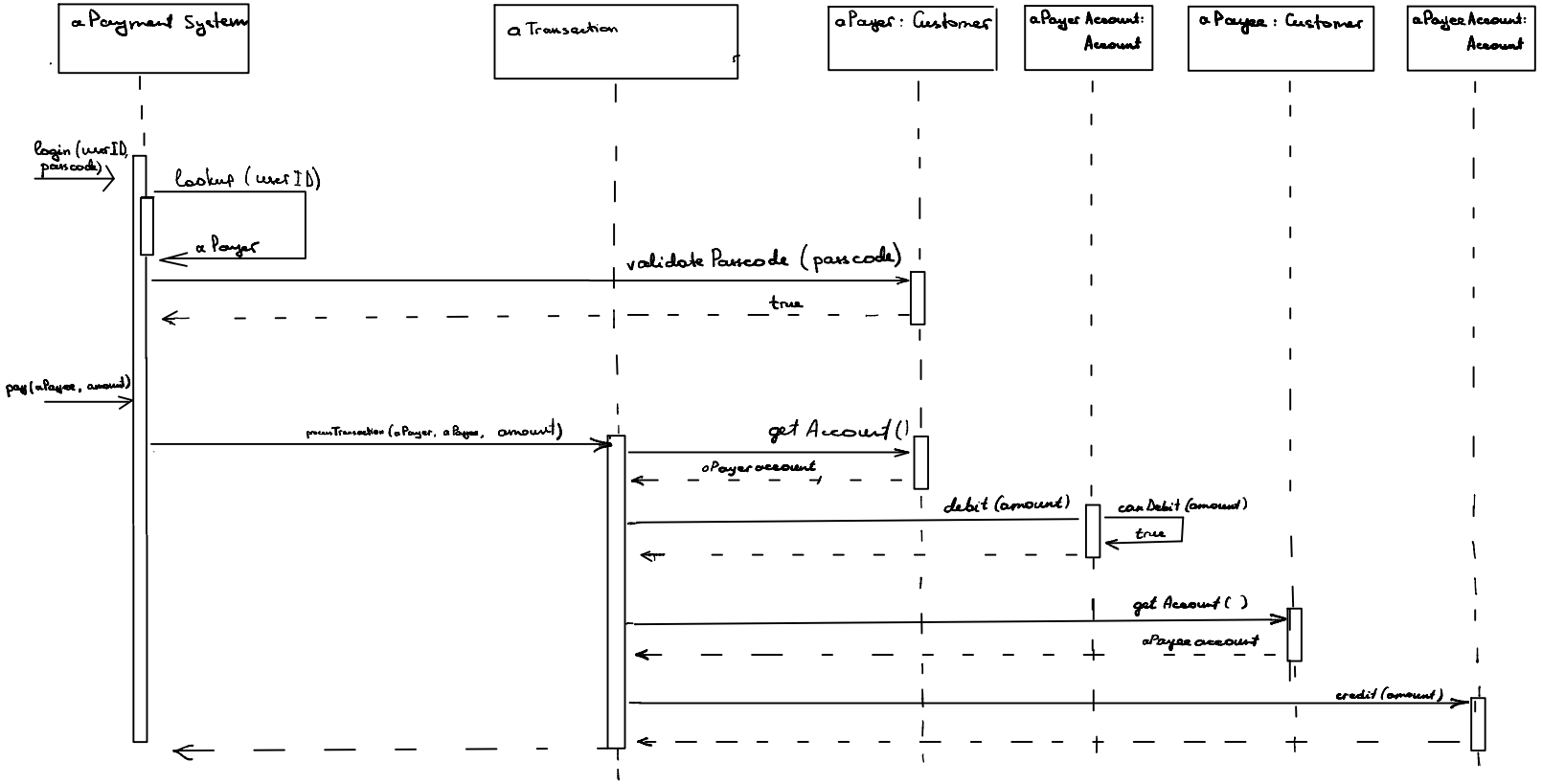
b) (i)

One of the possible solutions is presented below.



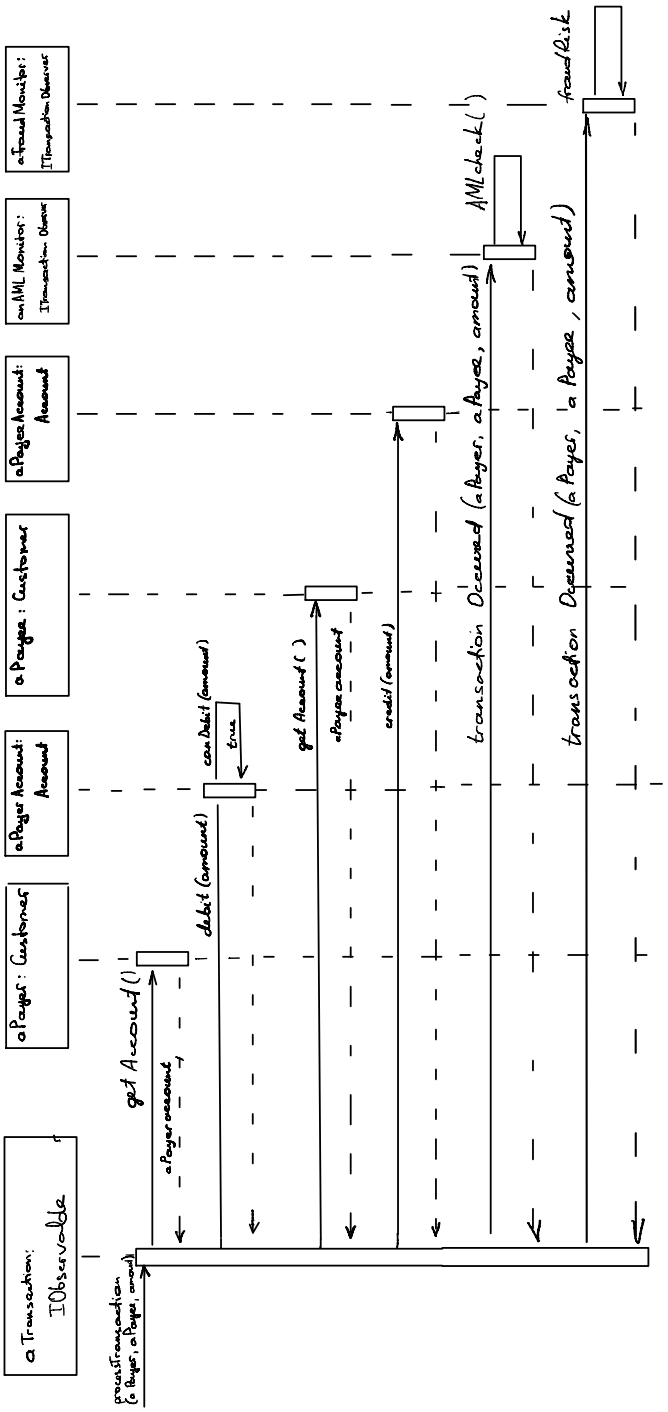
b) (ii)

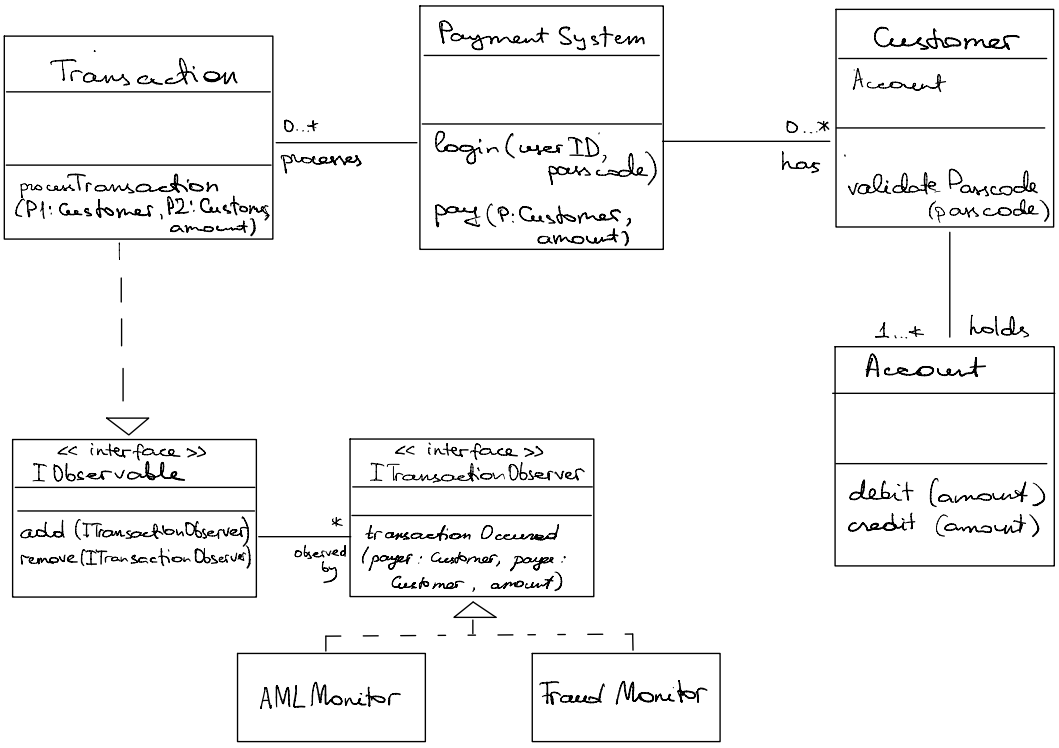
One of the possible solutions is presented below.


aPayment System
aTransaction
aPayer : Customer
aPayer Account: Account
aPayee : Customer
aPayee Account: Account
login (userID, passcode)
lookup (userID)
aPayer
validatePasscode (passcode)
true
pay (aPayee, amount)
processTransaction (aPayer, aPayee, amount)
getAccount()
aPayer account
debit (amount)
canDebit (amount)
true
getAccount()
aPayee account
credit (amount)

b) (iii)

One of the possible solutions is presented below.





Assessors' comments: The question was designed to test understanding of the key concepts of the object-oriented design such as polymorphism and inheritance; the ability to interpret the requirements and apply the main object-oriented design concepts in practice (in particular, principles of decoupling and abstraction), communicating the design through class and sequence diagrams; and the ability to extend the design by identifying a common design problem and figuring out a reliable solution to it in a form of an appropriate design pattern.

A popular question that candidates were able to answer reasonably well. Most students were able to go through an independent design process successfully, identifying the key concepts and communicating the outcome clearly using the standard notation. Not everyone was careful when working with both class and sequence diagrams as often sequence diagram did not correspond to the class diagram. Many students demonstrated their understanding of polymorphism and were successful in identifying the observer pattern as the appropriate one. However, not everyone was able to apply the observer pattern correctly, there were some inconsistencies and inaccuracies in UML notation, and some candidates struggled with the concept of polymorphism. Not everyone read the question carefully, and, as a result, some functionality was omitted / alternative functionality was provided.

Question 2

a)

Analysis Discovering and describing those aspects of a software development about which there is no choice.

Design Creating a definition of how the project goals are going to be achieved.

Implementation The process of writing code.

Building Creating a complete version of the software.

Testing is about making sure that small independent parts of code work correctly (unit tests), that all code parts work together (integration tests), that the functionality meets requirements (acceptance), that any new code doesn?t break the old (regression).

Deployment Actual release of the software into the end user environment.

Maintenance Supporting the software during its lifetime.

b) (i)

Among others: lives are potentially lost; reputational damage, credibility and trust loss, potential breach of contract leading to legal issues, competitive advantage loss, financial losses (including due to loss of customers, delays and overheads, time, legal issues etc.).

b) (ii)

A computer-aided London Ambulance Services (LAS) dispatch system failure 1995. Some of the major problems at different stages of the project can be mentioned such as:

Analyses and Design Major flaws in the process of gathering requirements and writing specifications/designs; key users such as ambulance operators, dispatchers etc. were not consulted; the scope was too large and no plan to iterate or introduce features one step at a time; attempt to redesign the whole process rather than develop a software that fitted into the process.

Implementation and Building No experience, no solid team; the company had no experience with real-time applications; the software development process was flawed with no firm foundation; unreasonable timescales (completed in 11 months); no solid project management, no solid team (in fact two people primarily in change were "a manager expecting to become redundant and a contractor who was a temporary addition to the organisation").

Testing No quality assurance; no validation or verification testing (seems to be working as users wanted but is this what they needed) no testing in real world conditions, errors / recovery not handled (didn't function well when given invalid or incomplete data regarding the position or statuses of ambulances; when ambulance crew incorrectly pushed wrong buttons and then tried to correct mistakes the system didn't accept the fix), no performance testing, stress testing or testing under load (in including memory leak in small portion of code), no usability tests (quirks and issue in the UI such as terminal screens had black spots preventing ambulance operators from getting all the information, attempts by staff to remedy mistakes were not accepted, staff trained 10 months in advance.

Deployment There were 81 known issues when the system went live, no load tests run and no provisions for a backup system; re-use some of the hardware that was already purchased for the previously closed project as part of the attempt to save money instead of purchasing hardware more suitable for the new system.

Lessons learnt should correspond to the problems given and can include the need for consultation of all stakeholders at the design stage, verification and validation testing, a stepwise approach to a new system introduction, clarification of the essential system goals and good-to-have features, solid project management, extensive testing including integration testing, under real-world conditions resource exhaustion/error/recovery, performance/stress testing and usability testing, risk analyses and backup, training.

b) (iii)

One may argue that not much changed as we still experience large software failures today. However, we now have well-established and often automated testing processes and procedures, including usability testing - we definitely know how to test better. Project management tools have been improved and we have techniques and tools that allow us to manage the projects and people better and communicate better. However, it is clearly the case that the intrinsic complexity of building large software system is unavoidable and the requirements will always remain changing and even conflicting, and therefore deciding precisely what to build is still a major challenge. It is though widely understood and accepted now that expert advice might be required from specialists (security, medical, law enforcement experts) and users have to be consulted at early stages. The idea of incremental delivery is also well established these days.

The core of this system is critical so it's important to get the system requirements right and dedicate significant amount of time analysing not just the functional requirements but also non-functional requirements such as usability, safety and security requirements. A formal waterfall model might therefore be appropriate since it allows early clarification of the main system goals and different types of requirements. However, the process may become more iterative once the main requirements are flashed out. It would be beneficial to separate core requirements from "good to have" features that can be introduced later incrementally, and to deploy the system in stages.

Effective formal project management tools have to be employed, thorough testing strategy must be designed, including Integration testing, verification and validation, resource exhaustion/errors/recovery; performance testing, stress testing and testing under load; usability testing and non-functional testing. Risk management and backup/fall back scenarios should be planned in advance, should things go wrong.

Assessors' comments: A reasonably straightforward question on software engineering methodologies and their application. Those who did attempt the question answered most parts well.

A few candidates described what can go wrong with the drones rather than discussed the consequences of a potential failure of the system. Most candidates successfully identified London Ambulance Services (LAS) dispatch system failure 1992 as the most relevant example, and discussed a variety of lessons learnt following the software development lifecycle and identifying what went wrong at each stage. Some candidates did not read the last part of the question carefully and did not assess the extent to which the lessons learnt in (b)(ii) are still relevant today. Furthermore, some candidates did not provide direct answers to the questions, making a variety of generic statements instead. A few identified critical nature of the application and consequently suggested a formal waterfall software development model in the last part of the question, yet backtracked almost immediately and concentrated on agile.

Question 3

a)

The system boundary encompasses the user, the data, the spreadsheet application, the forms, and the programming by example software agent at a bare minimum. Since the purpose of a system boundary is to use it for system mapping and subsequent risk analysis, the boundary also encompasses the rules and policies for ensuring the data is accurately translated into the forms.

b)

The type of automation is *action automation* as the programming by example system performs the automated action of transferring data from a spreadsheet to a form.

The appropriate level of automation is level 4 according to the types and levels of automation framework: the computer makes a decision to perform automatic execution if the human approves. (It is sufficient to state the level of automation qualitatively, the precise quantitative level need not be stated). This automation level is appropriate as the user will need to be sure the software agent is capable of correctly automating the task before automation commences.

c)

The primary evaluation criteria for this task are (1) mental workload; (2) complacency; and (3) skill degradation. Note that situational awareness, which in general is a primary evaluation criterion for automation, is not applicable for this specific task.

The secondary criteria are (1) automation reliability; and (2) costs of decision and action outcomes.

d)

A mixed-initiative interface is a user interface involving a software agent that lets both the user and the software agent take initiative for interaction and automation using the principle of direct manipulation.

Several principles are applicable as long as they are properly described, motivated, valid, and relevant. Here are three examples drawn from the Horvitz principles for mixed-initiative interfaces:

Develop significant value-added automation Only automate if a direct manipulation approach is clearly inferior.

Employ dialog to resolve key uncertainties If the software agent is unsure how to automate it should request user confirmation.

Allow efficient direct invocation and termination Provide mechanisms for the user to both trigger automation and terminate it.

e)

The primary risk here is incorrect automation, that is, the software agent fills out the forms incorrectly. Thus a *fault tree* is appropriate as it allows a comprehensive analysis of the factors and events that can lead up to this overall system critical error.

Other risk assessment methods are of course possible. As long as the motivation for the risk assessment method is accurate the answer will be acceptable.

Assessors' comments: This was a popular question. The question asked candidates to reason about an automation task in terms of its system boundary, type and appropriate level of automation, and primary and secondary evaluation criteria. It then asked candidates to critique the automation solution from the perspective of a mixed initiative interface and propose a method for assessing the key risks in the system. In general, candidates were able to propose a well-defined system boundary, state the type and appropriate level of automation, and explain the primary and secondary evaluation criteria. Candidates struggled to fully explain what is meant by a mixed-initiative interface but were mostly able to articulate three relevant mixed initiative principles for the task. Most candidates could propose a risk assessment method, but most methods were poorly motivated. A few candidates struggled to arrive at a type and level of automation appropriate for the automation solution. Several candidates failed to explain the primary and secondary evaluation criteria. Overall, this question was answered well by candidates.

Question 4

a)

We need the following operators, names and designations can obviously be different:

$FI(n)$ Find Icon among n preceding icons; $FI(n) = In$, where I is the time it takes the user to decide if an icon is the intended target.

SI Select Icon

TP Touch Phone

Old Method: $FI(n) + SI + FI(m) + SI$, where n are the number of preceding icons on the main page and m are the number of preceding icons in the folder.

New Method: TP.

The new method is more efficient if $TP < FI(n) + SI + FI(m) + SI$.

The limitations of this analysis is that it (1) assumes error-free behaviour from both the user and the system (for example, misrecognitions for the TP operator are not possible); (2) assumes optimal user behaviour; and (3) operator estimations are just heuristics and do not accurately reflect the precise time incurred, nor do they they have a notion of uncertainty (variance).

Note that $FI(n)$ is linear as a function of the number of icons. It does *not* follow the Hick-Hyman law as searching for the icon is a closed-loop task. For Hick-Hyman to apply, the task must be explicitly considered an open-loop task and this would also involve modelling the entropy of the icon distribution (e.g. the individual probabilities of a user accessing each specific icon). While such an analysis is possible to carry out, this is no longer a KLM-GOMS analysis.

An alternative solution is to assume that the user can either skilfully select the icon open-loop in both instances, or to treat the identification of an icon as a holistic mental operation. In this case the operator $FI(n)$ is changed from a function to a constant FI.

Estimations of operator durations are approximate as long as they are not unreasonable. As an example, a mental operation, such as the constant operator FI is usually assigned an estimated duration of 1.20 s. Selection, SI, is assigned 1.10 s, while a click, such as TP, is assigned 0.20 s. For $FI(n) = In$, I can be assigned a time of 0.20–0.40s, and n and m can be estimated based on the approximate number of preceding icons, such as 4–8.

b)

User control and freedom An important aspect of this heuristic is bestowing the system with an undo functionality so that the user can revert the system state if the user is accidentally finding themselves in the wrong system state due to an incorrect action or misrecogition. The mitigation strategy is to enable undo of functions triggered by this new function access method.

Error prevention Errors should be prevented. There are fundamentally two types of errors inherent in this method. First, human error, which in this case means incorrect triggering of a function. This can be mitigated by ensuring the user knows which functions can activate, and when (see below, recognition rather than recall, for a mitigation strategy). Second, misrecognitions due to classification errors in the machine learning algorithm. This can be mitigated by ensuring the algorithm is trained on data appropriate for the target audience within the environmental contexts where this method will be realistically used, that is, when people are walking around outside, in the tube, etc., not just in a lab setting. This is typically addressed by ensuring the verification cross-reference matrix specifies accurate verification environments for the machine learning algorithm.

Recognition rather than recall This heuristics states that, overall, there is tendency to prefer allowing users to visually recognise how to do things rather than asking them to recall how

to do things from memory. A mitigation strategy is to provide discreet visual feedback at the sides of the phone, indicating when the new function access method can be used, and which function will be activate. This feedback strategy can optionally be made adaptive and gradually remove visual feedback for functions that the user frequently invokes correctly without requesting an immediate undo. That particular extension strategy is called a training wheels approach.

c) (i)

There is no risk of an asymmetrical skill-transfer effect as such an effect is induced by one condition effectively training the participant to perform better in the next condition than vice versa. For such an effect to occur the task must have some inherent complexity that enables asymmetric learning to occur. The task here is a too simple visual-motor task for this to happen.

c) (ii)

The internal validity is the credibility of the causal relationship between the measured time and accuracy as a sole consequence of manipulation of the independent variable, in this case the specific function access method. The internal validity is high if confounding variables are minimised, which they are. The external validity relates to how well the experimental findings generalise to a use-context that is credible for validation. The external validity is questionable for the heuristics identified in (b) for the following reasons:

User control and freedom Undo and other error recovery strategies are not part of the experimental design. This can lead to misleading results as the actions necessary to recover from errors are frequently more costly than an improvement in average access time to functions.

Error prevention Realistic environmental use contexts are not covered by the experimental design. This can lead to misleading results that overestimate the accuracy of the new function access method due to the controlled setting of the experiment artificially reducing noise.

Recognition rather than recall The experimental design does not cover this aspect as the experimental design implicitly assumes expert behaviour—that is, that participants already know how to access each desired function. This can lead to misleading results on walk-up use (immediate efficacy) as users cannot be expected to know which functions are triggered without explicit guidance.

Assessors' comments: This was a less popular question which provided candidates with two alternative solutions for an interaction technique and asked them to carry out a KLM-GOMS analysis, reason about how well the two interaction techniques adhered to three prominent heuristics, and finally explain whether a proposed experiment would be robust to inform the design of the interaction techniques. Candidates were generally able to carry out a KLM-GOMS analysis but many analyses were flawed in that they either did not clearly define the operators or omitted critical operators. The discussion around the limitations of KLM-GOMS frequently omitted key considerations. Candidates struggled to interpret the heuristic ?user control and freedom? correctly, frequently omitting the principle of reversal. Overall, candidates were able to reason around the internal and external validity of the experiment, including whether there was a risk of an asymmetrical skill-transfer effect. However, many candidates failed to relate the discussion on the validity of the experiment back to the three heuristics, as asked in the question.