4M21 Software Engineering and Design: 2018/2019

Solutions

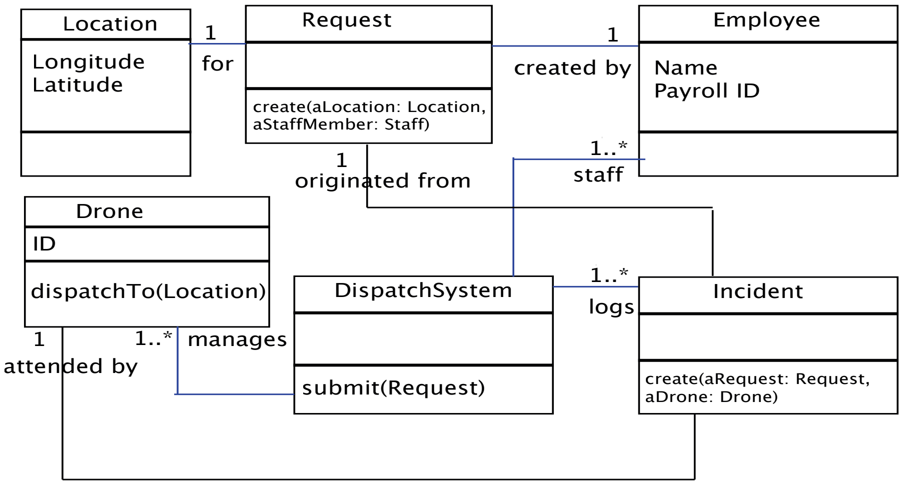
Elena Punskaya

15 January 2019

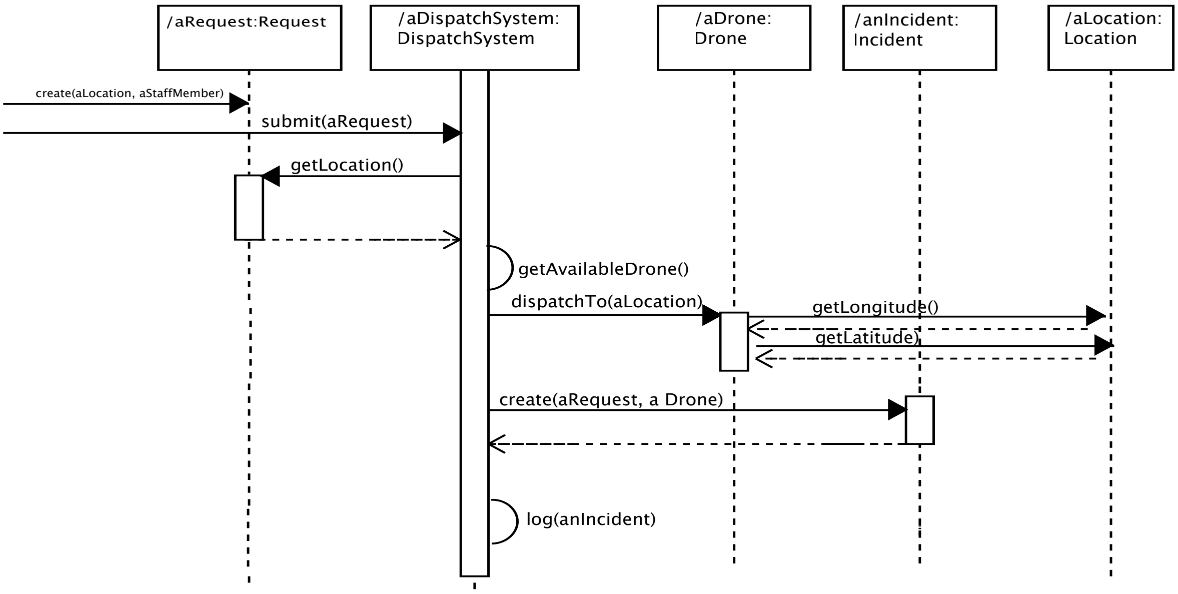
Q1. **(a)** A class represents a key concept within the system. It encapsulates data and behaviour.Classes provide abstraction. A class can be used to create multiple instances, i.e. objects that can contain data and behave according to class definition. Given the same data (state) two independent instances of the same class will behave exactly the same. In production a large number of objects are created, interact with each other, or are destroyed if no longer needed.

(b) One of possible solutions is presented below.

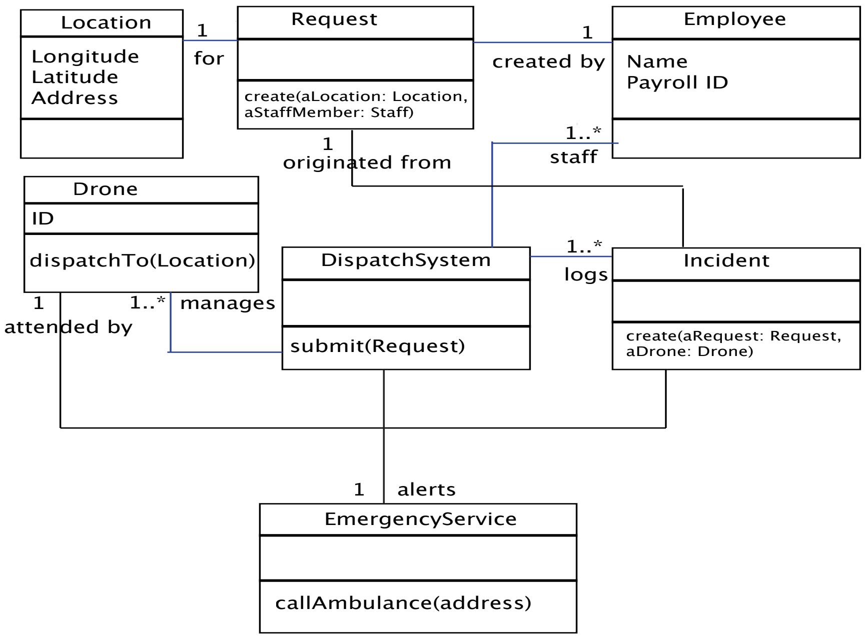
(i) Class diagram

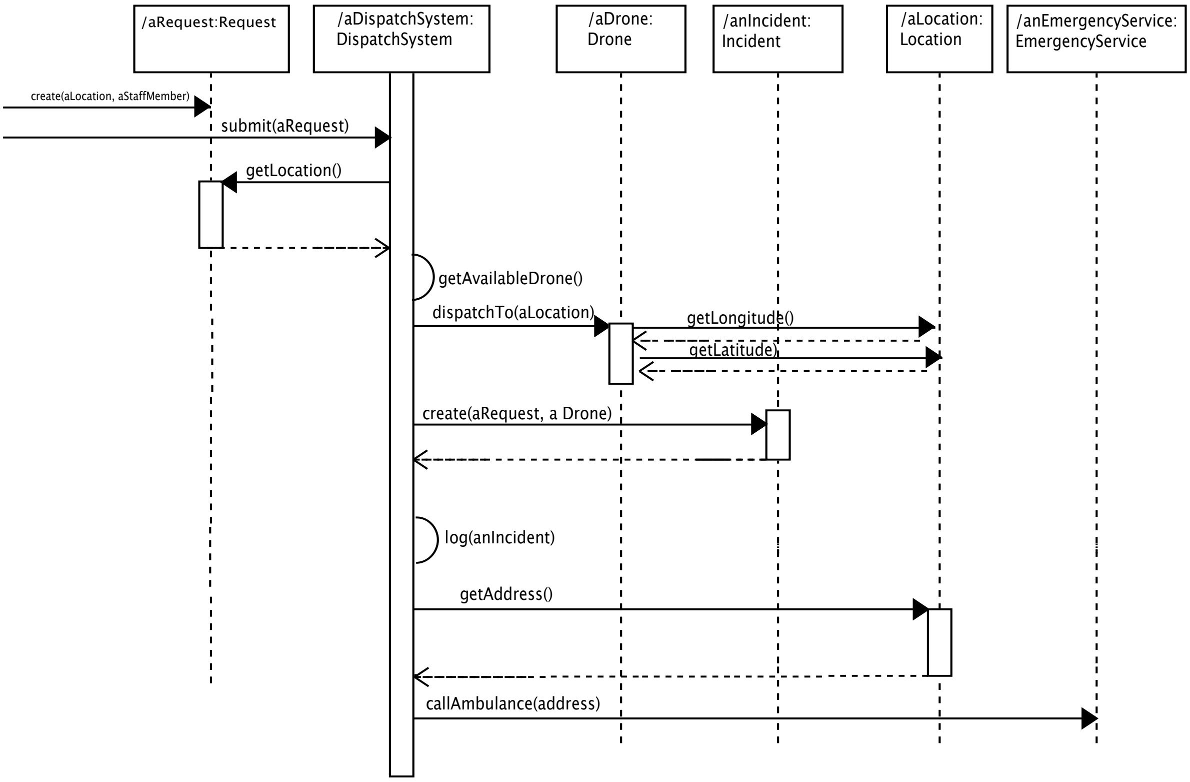


(ii) Sequence diagram



(iii) An example of extension, among others, could be to call ambulance after retrieving a specific address where the incident occurred

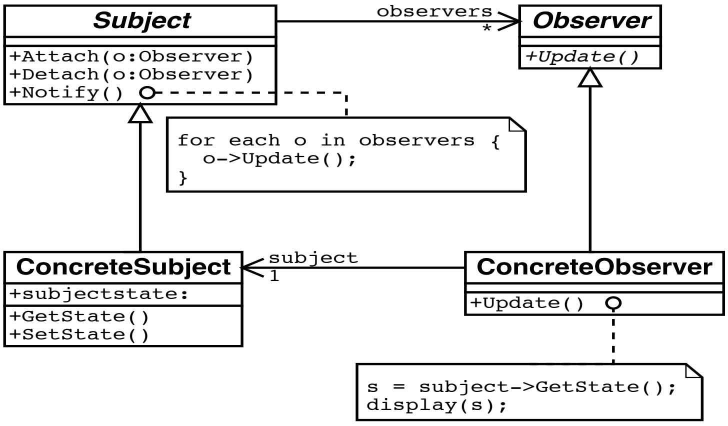




Assessor's comments: The question was designed to test the understanding of the key concepts of the object-oriented design and the ability to apply it in practice. It was a popular question and most students were able to go through an independent design process successfully and communicate the outcome 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.

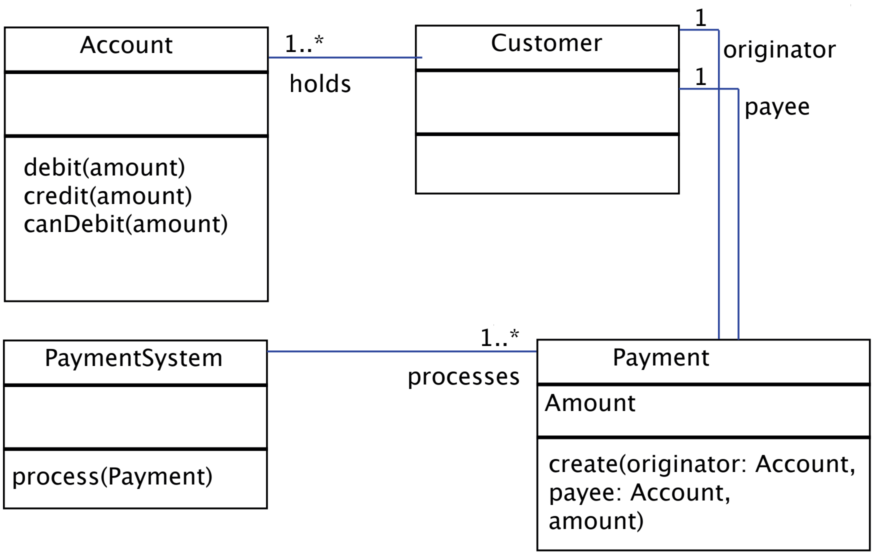
Q2. (a) We often find the same architectural structures occurring repeatedly (with subtle variations) created in response to commonly recurring problems. These solutions are identified and recorded as design patterns. Design patterns are described using standard format, including motivation, solution options, optimal solution, code example, design pattern, disadvantages.

One of the examples is an Observer pattern, which allows multiple object to maintain a consistent view on the state of the object of interest. For example, Twitter followers can be observers.

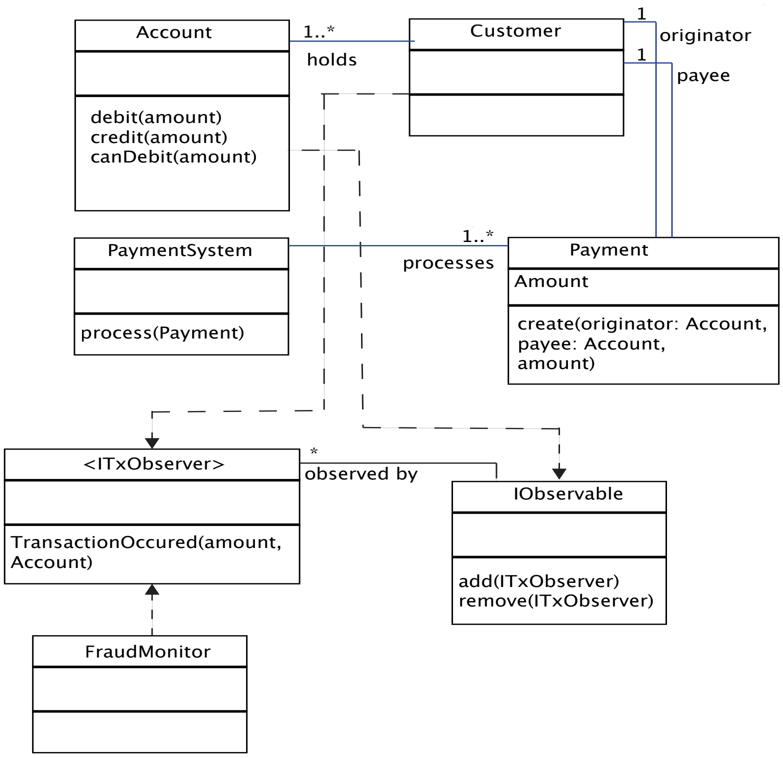


Disadvantages: can lead to a large amount of computation overhead if not safe-guarded, in particular, of a rare of notifications is high and the reaction to those updates is a heavy-load operation.

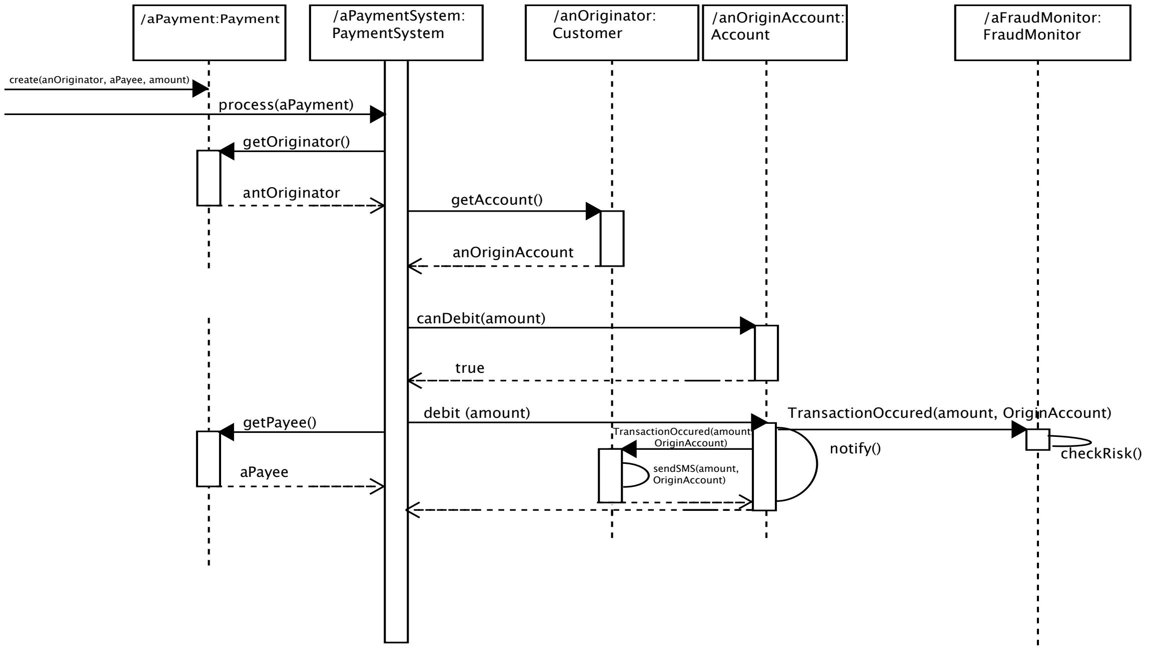
(b) One of possible solutions is presented below



(c) **(i)** Updated Class Diagram



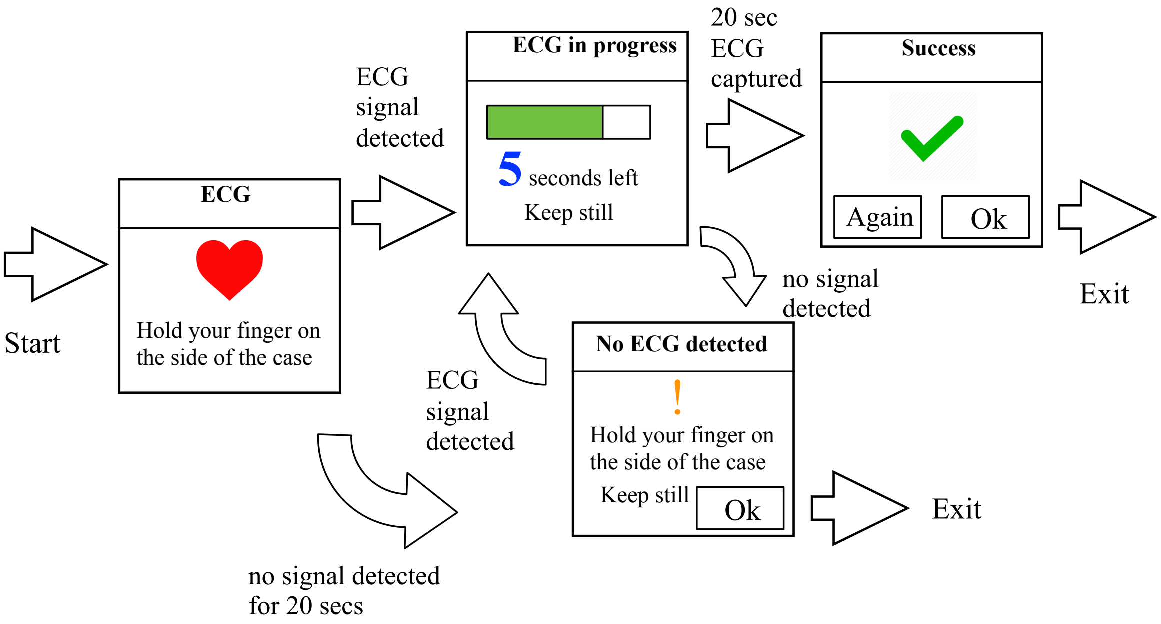
(ii). Updated Sequence Diagram


/aPayment:Payment
/aPaymentSystem: PaymentSystem
/anOriginator: Customer
/anOriginAccount: Account
/aFraudMonitor: FraudMonitor
create(anOriginator, aPayee, amount)
process(aPayment)
getOriginator()
antOriginator
getAccount()
anOriginAccount
canDebit(amount)
true
debit (amount)
TransactionOccured(amount, OriginAccount)
notify()
checkRisk()
TransactionOccured(amount OriginAccount)
sendSMS(amount, OriginAccount)
getPayee()
aPayee

Assessor's comments: This question was on understanding sequence diagrams, extending the design for software growth and the use of design patterns in the design process. This was the most popular question which was completed reasonably well, however, many students found it challenging to derive the class diagrams from the sequence diagram and, in particular, struggled with associations. Most students were successful with application of the observer pattern in the last part of the question although rarely were able to apply it to full extend.

Q3 (a) Metaphor is a set of UI visuals, actions, tasks, framework that the user is already familiar with to establish user's conceptual model. Using metaphors helps the user to align their conceptual model with the way that the program actually works. Ebook reader approach and media player UI could be used as examples.

(b) **(i)** One of possible solutions is presented below



(ii) The usability test should describe

Method

- test setup – how use cases are presented to the users

- approach – qualitative interviews by user experienced specialists

- session time: e.g. 30 mins

Target users sample

- age, gender, level of experience with technology

Scenarios – specific actions users are asked to perform

Feedback

- qualitative interviews (how the users felt about the device)

- subjective ranking in pre-defined categories (easy to understand, innovative, etc.)

- observations made by user experience specialist during the scenarios (for example, tried to put a finger on the screen rarther than side of the device)

(iii) Some of the examples could be:

elderly people potentially might find it difficult to read text in small font; given the small size of the screen it might be a good idea to display clear pictures rather than provide text descriptions elderly people might find it difficult to press small buttons; a clear simplified UI, large buttons etc.

elderly people might not be familiar with/ might find difficult some of the gestures such as swiping (if used)

elderly people might be hesitant to try / might be overwhelmed with too much information - contextual help/pictures as tips at each stage

Assessor's comments: This was a User Interface design question that was answered well by most of the candidates. A popular question that candidates were able to complete without any major challenges, however, some candidates did not read the questions carefully and some omitted some functionality/features. Some also lacked the methodological approach to the usability study design and provided only generic descriptions.

Q4. (a) Agile methodologies: Extreme Programming, Test Driven Development, Feature Driven Development, Agile Modelling, Agile Unified Process, Dynamic System Development, Scrum among others.

Agile techniques allow to incorporate a more iterative process to software development and therefore allow to adapt and grow software, and incorporate changes. They do not necessarily allow to clarify system goals early in the process, and it might be difficult to charge customers for changes. If all requirements are not defined well in advance testing and delivery according to specification might be not suitable for critical products. Some of the more traditional project management tools might not work well with it although agile management tools are available.

(b)

(i) Due to the nature of the application the core functions of the system are critical and have to work reliably from the first moment it goes live

- the system is used during laboratory experiments and the cost of the core functionality not working is high - if the system fails the experiment has to be repeated and often limited resources (including time) are wasted

- the system is used to support evidence and make conclusions regarding genetic cause of human disease and to identify and test new drugs to treat them - the system needs to be thought of reliable to make sure that these conclusions are not challenged

- there is stringent regulatory framework for medical software/devices, therefore the system might need to go through both quality assurance and performance certification, requiring thorough formal documentation

It might be beneficial to use a formal waterfall model to develop the core functionality of the system.

At the next stage it might be beneficial to develop a number of supporting features such as visualisation or reporting tools deemed not critical - these additional functionality could potentially be developed using an iterative process and a more agile approach.(

(ii) For the development of the core functionality, waterfall model follow a sequential process flowing steadily down the phases:

Requirements and Analyses: the environment and processes in which software will be used need to be analysed to establish operational parameters and required interfaces. Product requirements documentation is produced in compliance with relevant regulations.

Design: software architecture is defined to meet functional specifications. The design is documented in relation to requirements being addressed.

Implementation: code is produced and tested to perform functions according to specifications.

Testing: the test specification is developed and the product is tested in simulated environments or in a real lab environment alongside existing experiment tools for all required parameters, including capturing and interpreting data, failover and error handling scenarios etc. Test results are captured and where necessary reviewed by qualified medical professionals.

Operations: requirements and guidelines are established for installation, migration and maintenance of the product

This project will typically require a Quality Management System in place and will be utilising Gantt chart for project management and Source Code Repository for code development.

(iii)

- Not well analysed / misunderstood system requirements - expert advice from the specialists and potential users of the system is required when "one is deciding what to build", detailed use cases and scenarios should be created in the requirement analysis phase and reviewed during workshops with lab staff

- Some of the requirements might not be described exactly and might be misinterpreted when translated into software - testing strategy might need to be designed in consultation with the specialists

- Some of the requirements might not be identified at the initial stage - introduce a more iterative approach at the later stages, a change request system should be available to allow changes in the existing specifications

- Performance might not meet specification - test strategy including integration, validation and verification, resource exhaustion, error recovery, performance and stress testing as well as non functional tests, use of real fish should be encouraged during the implementation and testing, this could be further supplemented by using data captured by existing systems

- The users might find it difficult to use the system / might not use the system as intended - usability tests

- Not completed on time /on budget - formal project management tools employed

Assessor's comments: A reasonably straightforward, however less popular, question on software engineering methodologies and their application. Those who did attempt the question answered most parts well. Some candidates did not provide direct answers to the questions and only provided generic statements.