TABLE OF CONTENTS

* 1. Architectural style………………………...........…………………….Page 2
  2. Development cycle……………………...........……………………...Page 3
  3. Use Case Diagram……………………………………………………Page 4
  4. System design…………………………………………………………Page 5
  5. Product requirements…………………………………………………Page 6

2.4 Use Case Diagram……………………………………………………Page 7

Bibliography…………………………………………………………………Page 8

QUESTION 1

1.1 The architectural style used is called the MVC (Model View Controller)

* The Model according to Matemera (2017:42) he states that this is in the bottom layer, which handles all core logic of the system being created. This layer interacts with the data source. Systems using the MVC have a controller which passes information to the model in order to store and retrieve data.
* The Controller is where the business logic is defined. The user is able to send a response from the view, the response is then used to process the internal work of the request function, also it processes the response back to the user.
* The View is known as the presentation layer. This is what the end user will interact with the final product. A system can have multiple views that are different, ranging from command line output rendered HTML. The view does not consist of business logic, in most designs. The interface is an area of interaction.

The factors that affect the system architecture are as follows Matemera (2017:40):

* Existing systems

If there are systems already present in the organisation, then the architecture of a new system has to conform to that software.

* Enterprise architectures

Enterprise architectures provide a way of modelling the enterprise and aspects of the way it conducts business and concepts on how information systems are intended to support business.

* Technical reference architectures

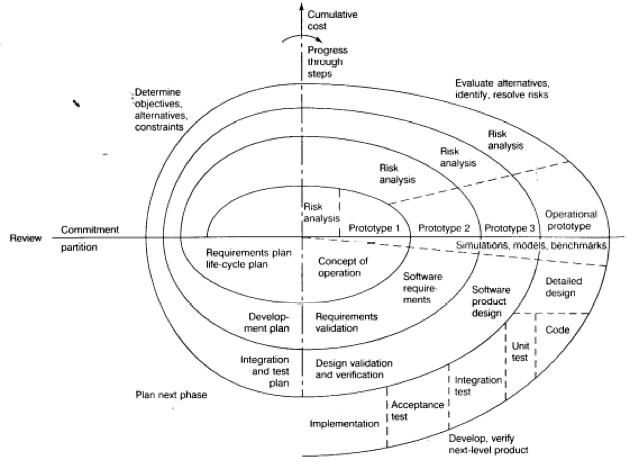
They focus on the technology that is used within the enterprise, the standards of the technology to be used and guidance on how to apply that technology.

1.2 The spiral model

According to Sommerville, I. 2014 this model focuses on addressing risks incrementally, in order of priority. This is why it will be the best approach to use in the given scenario. Boehm's spiral model distinguishes the following rounds:

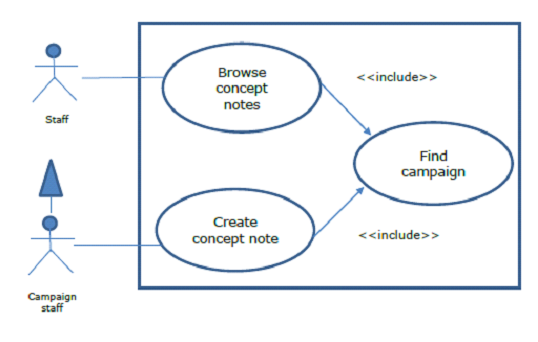
* Concept of Operation,
* Software Requirements,
* Software Product Design,
* Detailed Design,
* Code,
* Unit Test,
* Integration and
* Test,
* Acceptance Test,
* Implementation.

Each round is consisting of four phases. The first three rounds are shown in a polar coordinate system. The polar coordinates r = (l, a) of a point indicate the resource spent in the project and the type of activity, e.g. the creation of a concept note.



* During the first phase (upper left quadrant), developers explore the options to check whether or not the work be sub-contracted to smaller agencies, whether the agency is busy or whether we are able to open another ad agency that deals specifically with certain types of advertising, and many others. Define constraints these may be cost, time or quality constraints. Also identifying objectives.
* During the second phase (upper right quadrant), developers manage the solutions involved in the design of a system that meets the criteria specified above, and specifies a prototype to be built in the next phase. The design is used as input into the prototype that will be built by the developers.
* During the third phase (lower right quadrant), developers realise and validate a prototype or the part of the system associated with the risks addressed in this round. The developers here compare the prototype to the specifications given and see if the prototype does exactly as specified.
* The fourth phase (lower left quadrant) focuses on the planning of the next round based on the results produced in the current round. We check whether the current system works as required and if changes need to be implemented to further enhance the functionality of the system they currently have as a prototype.
* The last phase of the round is usually conducted as a review involving the project participants, including developers (company developing the system), clients and users (the users of the system from Agate which include the general staff and the directors). This review covers the artifacts developed during the previous and current rounds and the plans for the next round. Sommerville, I. 2014.

1.3 Use Case Diagram



From the scenario we can create two descriptions:

Description A, Create concept note :

* A member of staff working on a campaign can create a concept note,
* the concept not can records ideas, concepts and themes that will be used in an advertising campaign.
* The note is in text form.
* Each note has a title.
* The person who created the note, the date and time is also recorded.

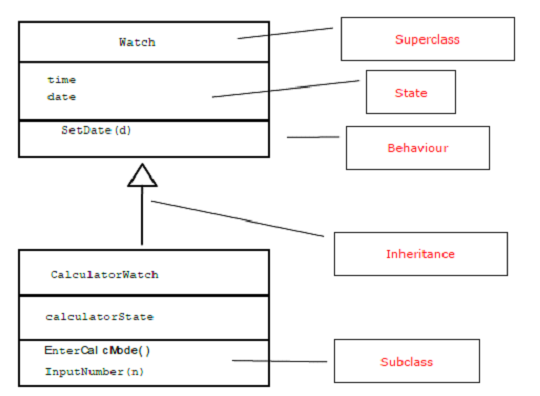
Description B, Browse concept note :

* Any member of staff may view concept notes for a campaign.
* The campaign must be selected first.
* The titles of all notes associated with that campaign will be displayed.
* The user will be able to select a note and view the text on the screen.
* Having viewed one note, others can be selected and viewed.

1.4 a)

* Abstract data type: implementation is hidden from the rest of the system.
* Class: An abstraction in the context of object-oriented languages. A class encapsulates state and behavior, e.g. watch.
* Subclass: This can be defined in terms of other classes using inheritance, e.g. CalculatorWatch.

b) DIAGRAM



QUESTION 2

2.1

A communication system incorporates capabilities to do the following: Avison, D.E. & Fitzgerald, G. 2002.

1. The need to be able to display the appropriate GRID to the corresponding user

2. The need to be able to send messages to the users of the system

To solve 1, we need to exploit some form of abstraction

* based on generalisation to that which we used to create a general framework.
* By defining an abstract drawGrid operation within the Grid class, we provide the simplest solution.
* When implementing communication between systems, this would have an appropriate method defined in a specialisation, which uses the desired package.
* A more elaborate class would be a GridImage class to which update and display requests could be sent.
* A number of specialisations hereof could be provided to suit different output possibilities. This might seem redundant, but would allow a single implementation, which could decide what form of object, and therefore interaction,
* to create when starting up.

2.2

Implementation of a game on a single machine

* Deployment is very easy to do, if the game is implemented on one computer.
* Messages can be passed on method invocations available in Java or in any other application design language.
* Java has a thread library which is used to simulate asynchronous behaviour.
* Since everything will be run over a single processor, there will not be any genuine transparency.

Implementation over a network

* Choose the elements which are to be deployed.
* Using these elements, then determine correspondence of associations to remote message passing. UML provides deployment diagrams to help with the associations.
* When a link between communicating objects is shown as an association, this is an abstraction from the physical realisation of that communication.
* Physical links are shown in a deployment diagram between the nodes and the logical links between components.
* Combining the two should be done upon matching of dependencies between the components.
* A component may match some high level object capable of partial autonomy.
* The associations between the objects form the dependencies with the physical links defining the properties of the association during instantiation.
* The process of examination of the physical properties of nodes and links brings up the estimation of the performance of the deployed application.
* Bottlenecks are identified and users are warned of the problems from such.

2.3 a) According to Ojo, A. & Estevez, E. 2015:

* Computer

Fast numerical calculations and word processing.

* Aeroplane

Fast transport from a place to the destination.

* Food processor

Cut vegetables, mix, and whip ingredients.

* Online banking

Perform banking transaction from a distance at any time of day.

* Life insurance

To provide for family after death.

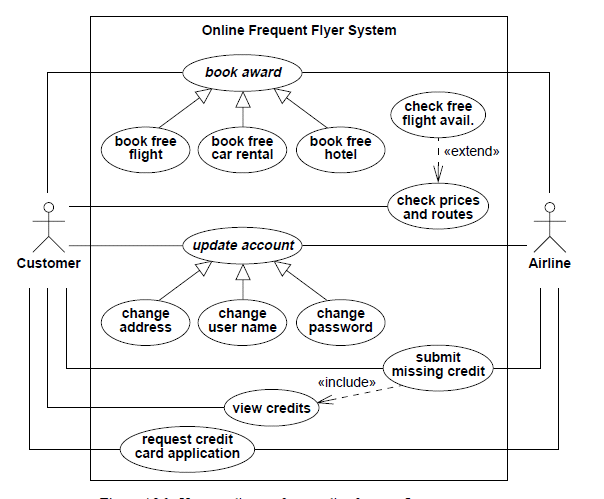
* Party planning

To ensure a party goes well and guests are fed and entertained properly.

b)

* Messages are instructions and information sent to objects in the expectation that the recipient objects will carry out certain actions.
* Messages are the means by which objects interact with each other and with the outside world.
* What an object does (or operations) is defined by the class; how the operations are carried out is decided by their implementation (or methods).
* A message may consist of a simple command, or it may carry additional information that qualifies the command.
* A message may also elicit a reply that is returned from the receiver of the message to its sender.

2.4 USE CASE DIAGRAM



BIBLIOGRAPHY

* Avison, D.E. & Fitzgerald, G. 2002. Information systems development methodologies, techniques, and tools. 4th edition. England: McGraw-Hill Education.
* Bennett, S. & McRobb, S. 2010. Object-oriented systems analysis and design. 4th edition. Berkshire: McGraw-Hill Higher Education.
* Best-Practice Software Engineering. 2013. Model view controller pattern. [Online] Available at: http://best-practicesoftwareengineering.ifs.tuwien.ac.at/patterns/mvc.html [Accessed: 24 October 2014].
* Booch, G. 1994. Object-oriented analysis and design with applications. 3rd edition. California: Benjamin-Cummings Publishing Company.
* Dash, S.K. 2014. How UX fits in different models of SDLC? [Online] 8 July 2014. Available at: http://samirshomepage.wordpress.com/2014/07/08/1797/ [Accessed: 24 October 2014].
* Khorev, A.B. 2014. COMP6700: Lectures Block 2. [Online] Available at: cs.anu.edu.au/student/comp6700/lectures/block-2/block-2 [Accessed: 24 October 2014].
* Ojo, A. & Estevez, E. 2015. Obeject-oriented analysis and design with UML. [Online] Available at: https://www.cpe.ku.ac.th/~jan/214574/review.pdf [Accessed: 16 November 2015].
* OpenLoop Technologies. 2010. Introduction to UML. [Online] Available at: http://www.openloop.com/softwareEngineering/uml/overview/ [Accessed: 24 October 2014].
* Schatten, A.; Biffl, S.; Demolsky, M.; Gostischa-Franta, E.; 謘treicher, T. & Winkler, D. 2013. Best-Practice Software Engineering. Model view controller pattern. [Online] Available at: http://best-practicesoftwareengineering.ifs.tuwien.ac.at/patterns/mvc.html [Accessed: 24 October 2014].
* Sommerville, I. 2014. Software engineering. 9th edition. Schweiz: Pearson Education Limited.
* Weisfeld, M.A. 2008. The object-oriented thought process. 3rd edition. Upper Saddle River, NJ: Addison-Wesley Professional.
* [Online] Available at: http://www.oxfordjournals.org/en/ Oxford University Press (OUP) publishes the journals and delivers this research to the widest possible audience.
* [Online] Available at: http://www.provenmodels.com/page/about A web-based library of management models aimed at graduates, executives and management consultants.
* [Online] Available at: http://www.jurn.org/#gsc.tab=0&gsc.q=commerce&gsc.sort= JURN is a unique search-engine primarily dedicated to indexing free and pen-access e-journals in the arts and humanities. It is aull text finder that harnesses all the power of Google to search across quality open access content
* [Online] Available at: http://www.netmba.com/ Articles in the subject fields related to Business Administration, including both elementary and advanced topics as well as frameworks and theories involved in solving challenging problems.