Characteristics of good requirements

Complete - all functions are identified as you don't want bits of your functionality missing

Unambiguous - requirements should not be vague or fuzzy, as that it going to cause issues when the system is in production

Consistent - if there is a conflict among requirements from different stakeholders, they have to be resolved so that they are consistent

Sufficient - requirements is described in sufficient detail, so that developers can develop the system according to the users requirements

Testable - The acceptance criteria are specified in enough detail so that you can easily test whether the requirements are met and the system is working as the user intended

The correct answer is: Complete → All business functionality identified, Sufficient → You have the required level of detail, Consistent → No conflict amongst the requirements, Testable → Is working as the user intended,

Unambiguous → Nothing too vague or fuzzy

///

Which of the following will you do to help conduct a successful interview:

Prepare a list of questions that you will not divert from, as you want to make sure that all your questions are answered (Incorrect) - as you should be responsive to what the client is saying, and may need to change your questions accordingly

Ask just one broad question at the start of the interview, and let the client speak for the rest of the time as they have all the information (Incorrect) - While asking a broad question is a good way to start and interview, it is a extremely rare client who would be able to give you all the information you need without guiding and prompting. In reality, you will need multiple sessions with the client to gather everything you require, and even then sometimes requirements are missed.

Send the client an interview transcript (word for word), so that they know exactly what was said in the interview (Incorrect) - Doing this is a waste of both the interviewer's and the client's time. The client would much prefer a summary of the key points that you have gathered about the new system, during the interview. You need to provide enough information so that they can confirm that your understanding of what they have told you is accurate.

Research as much information about the company and system in advance so that you can prepare relevant questions (Correct) - If possible you should try and understand as much as you can about the company and the particular system before hand, so that the interview is productive, and so that you can ask specific, targeted information to maximise the information you can receive from the client.

The correct answer is: Research as much information about the company and system in advance so that you can prepare relevant questions

///

What is the Hawthorne Effect?

The Hawthorne effect is a phenomenon whereby workers improve or modify an aspect of their behaviour, or stop working in response to the fact that they are being watched. (Correct)

While it is true that people may not speak up in a group due to being observed by others (Incorrect), and observation is used as a means of data gathering (Incorrect) these are not "The Hawthorne effect".

And, the response of a team member when another team member is given a reward (Incorrect) is a nonsensical response to this question.

The correct answer is: People’s tendency to behave differently when they are being observed

///

Which of the following best describes a UX formative evaluation?

Formative evaluation always takes places during the development process not when the system is already live. So Usability testing on a live site to understand how it compares with its competitors and Quantitative usability testing on the current version of the live site in order to establish a baseline for further quantitative benchmarking are not correct as they are working on the live site. Running a focus group to understand the needs of the target population is not an evaluation as it requires the user to use prototypes to be able to conduct an evaluation. Qualitative usability testing during development to find out which changes need to be made is correct as it is done before the system goes live.

The correct answer is: Qualitative usability testing during development to find out which changes need to be made

///

Questionnaires can be useful in information gathering when users \_\_\_\_\_.​

If a staff member is not well-informed about the business processes (Incorrect) - then they are of very little value to you regardless of the data gathering technique that you choose to use.

The level (low or high) of a staff member (Incorrect) - is of no consequence, what is important when choosing data gathering technique that best suits the type of data you are trying to gather.

Staff being time poor and do not have time for Interviews (Incorrect), does not make a questionnaire more suitable, as they may not have time to do the questionnaire as well, and more important the questionnaire may not be able to capture the in-depth information that you can gather during an interview.

Questionnaires can be useful in information gathering when users are geographically distributed (Correct) because they are easy and cheap to disseminate, follow up, collect and collate the data.

The correct answer is: are widely distributed geographically

///

What is MOST important according to the Agile Manifesto?

The four foundational values of the Agile Manifesto are a guide to the development and delivery of high-quality, working software.

1. Individuals and interactions over processes and tools  - Understanding the requirement of the clients is essential to developing great products. Individual and interactions help create customer-focussed products.  Developing good products requires effective team-work and interactions, irrespective of the processes and tools used.Communication is an example of the difference between valuing individuals versus process. In the case of individuals, communication is flexible and happens when a need arises. In the case of process, communication is scheduled.
2. Working Software Over Comprehensive Documentation - The amount of time spent on documenting user and technical requirements in tradition development often led to project delays and unmet client expectations. Agile focuses on providing enough documentation to get the job done, without getting bogged down in excessive detail.
3. Customer Collaboration Over Contract Negotiation - This value describes a customer who is engaged and collaborative throughout the development process, making it far easier to meet their requirements. With traditional development, customers negotiate the requirements for the product, often in great detail, prior to any work starting, with contracted delivery points creating a wall between the customers and developers.
4. Responding to Change Over Following a Plan - Traditionally change was considered a cost to be avoided.  The focus was on following a detailed plan with a defined set of features to deliver. Agile considers change as a valuable agent and responds and adapts to it with the iterative model.

SO - People and communication would be the most important from the options provided.

The correct answer is: People and communication

///

Agile Manifesto says to value responding to change over following a plan. Which of the following statements illustrates this?

The correct answer is: Changes are accepted at any time during the development effort depending on the business value of the change, the Product Owner's acceptance, and the ability of the Dev Team to respond in a timeframe acceptable to the Product Owner.

///

A dental surgery would like to move from their manual appointment system to an online system. A vendor has said they have an appointment system that they have sold to many dental clinics. Which of the following is NOT a benefit of considering a ready-made vendor solution:

Considering a ready-made vendor solution can:

* help provide new ideas that you had not thought about (incorrect)
* be state of the art rather than just automating your current system (incorrect)
* often be cheaper and less risky as you know exactly what you are getting (incorrect)

However, it is very rare that a vendor solution can meet most of your specified business requirements (correct).  This means that you will either have to pay for the modification to meet your requirements, or you will have to change your business practices to suit the system, which may not be ideal.

The correct answer is: It meets most of your specified business requirements

///  
When testing interface designs with users, who should you use?

Typical users is the correct answer as you want to test the system with who is most likely to use the system.  The others would provide one type of user so just using them would not cover the spectrum of typical users of the system.

///

Which of the following statements is true?

* When a domain model class diagram is modified to become a design class diagram it contains operations as well as the attributes required by the system. (correct answer).
* The domain model class diagram does not contains operations as well as the attributes required by the system - just the attributes.
* The design class diagram is an extended version of the domain model class diagram so it does not contain just the attributes required by the system - but details required by developer such as the methods.
* The design class diagram is NOT an important tool to show the users how the system will actually work to meet their needs.  It would be too technical to show users and would not be a good tool to use to help understand their needs.

The correct answer is: The design class diagram contains operations as well as the attributes required by the system

///

An OPT sequence diagram frame:

* The OPT frame stands for optional and is a portion of an interaction path that may be done. (correct answer)
* The ALT frame for alternative shows alternative paths where one of the options will be done
* The LOOP frame represents a repeating loop of functionality
* There is no particular frame used to show parallel functionality.

The correct answer is: is a portion of an interaction path that may be done

///

System testing represents an integration test of:

System testing represents an integration test of an entire software system.

The following are incorrect because:

* an entire method would usually be one module which would be testing using unit testing
* an entire class would usually have a number of methods which would require integration testing after they had gone through individual unit testing for each method
* multiple unrelated classes would not really be part of a system - a system would be made up of a number of related classes

The correct answer is: an entire software system

///

Which three measures are needed to establish how usable a product is?

User Effectiveness, Efficiency and Satisfaction is the correct answers.

Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use.

* Effectiveness: accuracy and completeness with which users achieve specified goals.
* Efficiency: resources expended in relation to the ‘effectiveness’ with which users achieve goals.
* Satisfaction: the comfort and acceptability of the work system to its users and other people affected by its use.

The correct answer is: User Effectiveness, Efficiency and Satisfaction

///

What is a user persona?

What is a user persona?

* A user persona should be **a fictional individual used to represent the demographics, goals, values and pain points of a large proportion of a product’s user group** - correct answer.  The primary persona should represent your largest group of users as your system should primarily cater for them.
* It should not be the **description of a single important user’s behaviour and personality**regardless of how important they are as it should represent the majority of your users.
* It should not be about **the behaviours, characteristics and goals which the UX designer hopes from a group of users.**What the designers hope for is irrelevant - the persona should be based on the representative behaviours, characteristics and goals of the users gathered via suitable research methods.
* It should not be based on the **collection of behaviours, characteristics and goals which belong to a small group of your actual users,**as you would then have way too many personas to use and it would be impossible to design and interface to meet all the competing needs of small groups of users.

The correct answer is: A fictional individual used to represent the demographics, goals, values and pain points of a large proportion of a product’s user group

///

When prototyping it is important to:

When prototyping it is important to set clear expectations with the user as to why you are prototyping so that they do not think it is the actual solution - this is especially true of high fidelity prototypes.  When building prototypes - reuse as many elements as possible from previous prototyping exercises or from the software library, it is a waste of time building all elements from scratch.  It is important to understand what the technology is capable of and not just build things during prototyping that are easy to build as it is a mock up, but can never be built in reality.  This will just build false expectations for the user, and cause a great deal of frustration.   The aim of prototyping is to try and understand what the user requires and show them a design that reasonably reflects what they can expect. Developing a perfect, very realistic prototype so that they know exactly what they will be getting with the final system is a waste of time and money without any added benefits.

The correct answer is: Set clear expectations as to why you are prototyping so that the user does not think it is the actual solution

///

Which of the following statements is applicable to an Actor in a Use Case Diagram?

An actor specifies a role played by a user or an external system that interacts with the system being depicted.  An actor is always shown outside the system boundary, and can interact with one or more use cases.  An actor in this context has nothing to do with the design of the system being built.

The correct answer is: An actor specifies a role played by a user or an external system that interacts with the system being depicted

///

You are close to the delivery date for a project but cannot guarantee that there are no errors in the system. Based on the further information provided in each statement, which of the statements provides a valid option:

It is okay to deploy the system as it is currently, as the risks associated with failure of the system are not high, so even if something does go wrong it will not be catastrophic and the issues can be fixed as part of maintenance.

The following are incorrect because:

* It is a high risk system, but you will use the direct cutover method to deploy the system on time to minimise the risk - using the direct cutover method is in fact high risk - if anything goes wrong there is no back up like there is with the parallel method
* It is never acceptable to deploy a system with errors, so you keep testing until you can guarantee that the system is 100% error free - as the cost of finding all errors is very high, most systems are deployed with the knowledge that there are some errors but they will not result in massive issues.  If the type of system is such that errors will result in catastrophe then the system should be 100% error free.
* Since the delivery date is close, you deploy the system as you must deliver the system on time to meet the requirements of the contract - the contract delivery date should not drive this decision, as if it is a high risk system forcing delivery could result in significant issues for the company

The correct answer is: You deploy the system as it is currently, as the risks associated with failure of the system are not high

///

In interface design, it is difficult to input correct data when:

* A generic message that is used for all errors is displayed when input is the correct answer - as it is very difficult to input correct data if the error message is general and not specific to the error that has occurred, and does not give a clear indication of what is wrong and how to fix it. You are then doomed to most likely re-entering incorrect data.
* The format of the input is provided in the input field - is helpful to input correct data as it tells you exactly what is expected of your data input - so having the format in light font in the input field or as part of the label is very helpful in preventing errors.
* A solution message is provided when an error occurs - is helpful as it tells you how to solve your error so that you do not repeat entry of the incorrect input
* A drop down list of possible inputs are available for the input field - if it suits the particular entry field it is very useful to provide a drop down list of the valid options, as it makes it much easier to input a valid option.  Thought has to be given to the ordering of the options in the drop down list to maximise ease of use.

The correct answer is: A generic message that is used for all errors is displayed when input is incorrect

///

What is addressed in the daily standup or daily scrum?

Why didn't I get my work done yesterday – what were the issues?

What am I planning to work on today? What have I done since the last daily stand up?

///

The Initiation Phase is the first phase of Systems Development. It is where you assess the feasibility - it is where you decide whether it is even worth going ahead with the project.

///

reason for prototyping?

Prototyping is used to help understand the users requirements and assess the usability of a design.  It is useful to sort out any disagreement of the vision of the project as a prototype can show how an idea works (or not as the case may be).  It also reduces maintenance of the system as the user has a reasonably clear idea of what to expect before the system goes live, so you would not expect many changes of the live system.

Prototyping is not about spending time and money getting things exactly right.  It should be the minimum amount of time and money to ensure that the users understand what they are getting.  While developing a high fidelity prototype is useful, it is better for training and should not be pursued if it is going to be very time consuming and expensive.

The correct answer is: Spending the time and money developing a high fidelity prototype and getting it exactly right, so that the user knows exactly what they are getting

///

User Stories?

User stories are the Agile form of requirements documentation. They are the business requirements of the new system that are broken down into smaller, more manageable forms. and they must be detailed enough for a Scrum team to develop

Users do not need any training to use User Stories as they are written in a simple form that anyone can understand and use.

The false answer is: Users have to be trained to document their requirements properly using user stories

///

Agile value

The four foundational values of the Agile Manifesto are a guide to the development and delivery of high-quality, working software.

1. **Individuals and interactions over processes and tools  -**Understanding the requirement of the clients is essential to developing great products. Individual and interactions help create customer-focussed products.  Developing good products requires effective team-work and interactions, irrespective of the processes and tools used.Communication is an example of the difference between valuing individuals versus process. In the case of individuals, communication is flexible and happens when a need arises. In the case of process, communication is scheduled.
2. **Working Software Over Comprehensive Documentation -**The amount of time spent on documenting user and technical requirements in tradition development often led to project delays and unmet client expectations. Agile focuses on providing enough documentation to get the job done, without getting bogged down in excessive detail.
3. **Customer Collaboration Over Contract Negotiation -**This value describes a customer who is engaged and collaborative throughout the development process, making it far easier to meet their requirements. With traditional development, customers negotiate the requirements for the product, often in great detail, prior to any work starting, with contracted delivery points creating a wall between the customers and developers.
4. **Responding to Change Over Following a Plan -**Traditionally change was considered a cost to be avoided.  The focus was on following a detailed plan with a defined set of features to deliver. Agile considers change as a valuable agent and responds and adapts to it with the iterative model.

The correct answer is: Working software → over comprehensive documentation, Individuals and interactions → over processes and tool,

Customer collaborations → over contract negotiation, Responding to change → over following a plan

///

Tight coupling

Coupling is a qualitative measure of how closely classes are linked which is depicted by how much the classes are interconnected by the amount of data passed between the classes. You can see this by the number of navigation arrows on a design class diagram and the number of parameters passed by the methods of the class.

If you have low or loose coupling the system make the system easier to understand and maintain and as a result of minimal data passing, the ripple effect is minimal.

So conversely, if you have tight coupling which is not good, it shows that the class is related to a lot of other classes which would make it difficult to understand  (the correct answer).   It does not make a system easy to maintain or allow modules to be easily reused, and it is the result of low (not high) cohesion.

The correct answer is: makes a system difficult to understand

///

What is the purpose of Requirements Gathering?

To negotiate on the price the user will pay for the new system (Incorrect):  This is nothing to do with requirements gathering.  This process would be part of the initiation phase, where you decide whether you want to proceed with the project.

To let the user know what the new system you are going to develop will do for them (Incorrect) and To find out what the user wants the new system to do (Correct): Requirements gathering is about listening to the user to find out what their needs are, it is not about making decisions for them about what they need, and just informing them of your decision.  However, it is important to discuss their needs and add value based on your experience with the issues and opportunities they are facing, so it's not just about listening - it is about having a conversation.

To interview each and every person in the organisation so you do not miss out on any vital information (Incorrect):  While in a very small organisation this may be practical, in most instances you will choose individuals who represent a particular role to interview.  For example, if you needed information about a booking function, you would not need to interview all the booking consultants, just one or two would be fine.

The correct answer is: To find out what the user wants the new system to do

///

A sequence diagram:

A sequence diagram is a type of interaction diagram which depicts the objects involved in a scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. It shows object interactions arranged in time sequence.  So all the options is the correct answer.

The correct answer is: All the options

///

message types

The right description for each of the following message types is as follows:

* A Synchronous message - is used when the sender waits for the receiver to process the message and return before carrying on with another message
* An Asynchronous message - is used when the message caller does not wait for the receiver to process the message and return before sending other messages to other objects within the system
* A Return message - is used to indicate that the message receiver is done processing the message and is returning control over to the message caller

///

Sprint Review Meeting- Demo of the Increment

Sprint Retrospective- The team discusses the improvements that can be applied for the upcoming sprints

///

The later in the development life cycle a fault is discovered, the more expensive it is to fix. Why?

The later in the development life cycle a fault is discovered, the more expensive it is to fix, because the fault gets built into each stage of the development life cycle as it proceeds, and when you discover the fault you have to go right back and redo each stage incurring all the costs associated with the redevelopment at each stage.

///

The main focus of acceptance testing is:

Acceptance testing is all about ensuring that the system meets the functional and non-functional requirements specified by the users.

///

best describes how the documenting of requirements using User Stories connects to Agile values:

With the following Agile values:

* The focus is on individuals and interactions rather than processes and tools
* The user is allowed to change their mind throughout the process rather than following a plan
* Rather than focusing on collaboration at contracted times, it encourages participation throughout the process

we could use any other means of documenting the requirements (not just User Stories) and still connect to these Agile values.

However, with the following Agile value:

* It makes sure that working software is produced rather than extensive documentation

it is the use of User Stories to document the requirements, where the focus is on producing short, simple documentation - just enough for the developers to know what they need to code, rather than extensive documentation.

///

how the documenting of requirements using User Stories connects to Agile values

With the following Agile values:

* The focus is on individuals and interactions rather than processes and tools
* The user is allowed to change their mind throughout the process rather than following a plan
* Rather than focusing on collaboration at contracted times, it encourages participation throughout the process

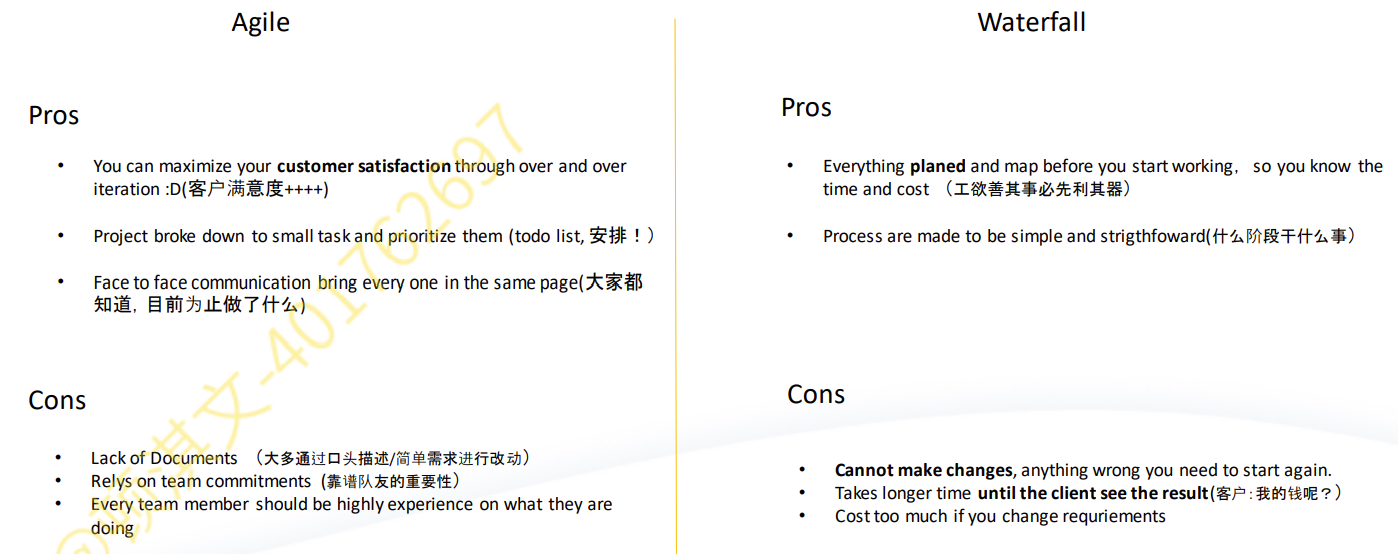
we could use any other means of documenting the requirements (not just User Stories) and still connect to these Agile values.

However, with the following Agile value:

* It makes sure that working software is produced rather than extensive documentation

it is the use of User Stories to document the requirements, where the focus is on producing short, simple documentation - just enough for the developers to know what they need to code, rather than extensive documentation.

System Development Approaches – Agile Vs Waterfall



///

Select the best name for a Use Case from the following options:

A Use Case should be named using a Verb followed by a Noun so the correct answer is 'Enrol Student'. None of the other options follow this structure.

///

Acceptance criteria are an important part of a user story because:

Acceptance criteria are not only an important part, but an essential part of a user story.  It ensures that there is clarity, no miscommunication or confusion, and it clearly gives the development team a template of what they have to deliver to the user.  So, all choices are correct.

///

Usually 2-6 months after the system is deployed and is live, a post implementation review is conducted to understand what went right/wrong during development and to learn for the future (correct).

///

* Entity - objects represent the data required for the business functionality of the system
* Boundary - objects that interface with the system’s actors
* Control - objects manage the execution of the messages to the system objects
* Data access - objects separate access to the database

///

Which of the following is NOT correct when gathering business documents and procedure descriptions when investigating requirements within an organisation

Gathering business documents and procedure descriptions when investigating requirements within an organisation does help:

* obtain a preliminary understanding of processes (incorrect)
* identify business rules, discrepancies, and redundancies (incorrect)
* provide context and guide interviews (incorrect)

However, while it may provide a complete and accurate representation of the business data and processes, there is no guarantee of that, as often business documents and procedures are out-of-date and can provide a misleading picture of the business. (correct)

///

Which of the following statements about 'cohesion' is true?

* Cohesion measures the unity of purpose of the methods within a class and the class itself - how strongly methods in a class are related to each other (correct answer).
* It has nothing to do with how classes interact with each other - that is coupling.
* While high cohesion does lead to low coupling - it makes the system easy (rather than very difficult) to maintain.
* Low cohesion increases the extent to which objects are dependent and can causes ripple-through effects, reducing the ability to reuse parts of the system

///

A good quality user story does NOT have the following characteristic:

A way of assessing the quality of a user story is by using the INVEST acronym:

* Independent – Stories should not be dependent on other stories.
* Negotiable – Stories should allow space for discussion and negotiation
* Valuable – Stories should clearly illustrate value to the customer.
* Estimable – Stories should provide just enough information so they can be estimated. It is not important to know the exact way that a particular problem will be solved. It must only be understood enough to provide a high-level estimate.
* Small – Stories should strive to be granular enough in scope that they may be completed in as little time as possible, from a few weeks to a few days.
* Testable – Stories need to be understood well enough so that tests can be defined for them. An effective way to ensure testability is to define user acceptance criteria for all user stories.

So

It can be developed separately from other stories - is Independent

It can be discussed and evolve as required - can be Negotiated

It is of a size that can be ‘done’ in a sprint - is Small

It specifies the technologies need to develop the story - NO this is not a characteristic of good quality user story as it is focussing on the technology rather than WHAT the user wants.

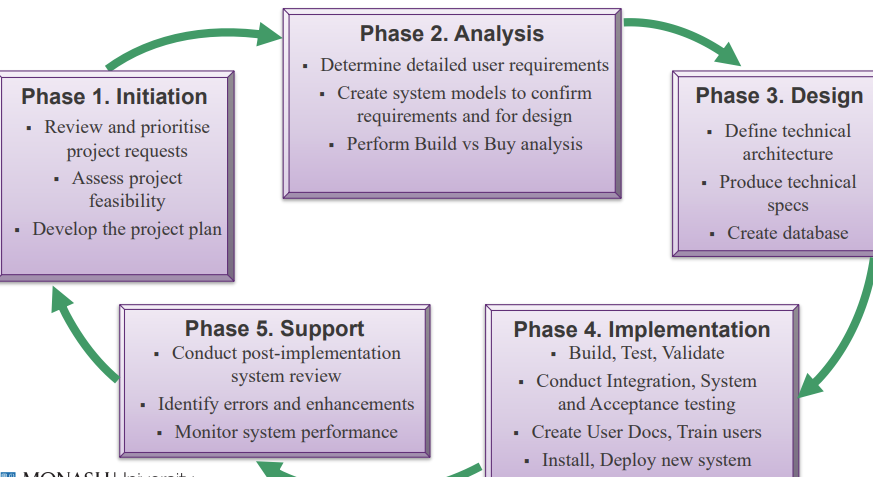
///

Which of the following statements best describes a Use Case?

 A Use Case represents behaviour which when performed is of value to one or more actors.  It is not only associated with one actor, and it has nothing to do with the performance of the system, or the Business Analyst making a case.

///

Enhancement, upgrades and bug fixing are done during the \_\_\_\_\_\_\_\_\_ phase in the SDLC.



///

Which interface design guideline do ALL these images represent?

Ben Shneiderman’s 8 Golden Rules

///

Your need to make a decision about deploying a medical system which could cause significant patient issues if there are problems with the system. The system has to be deployed at a number of hospital locations. Which of the following deployment options would you recommend?

*Decision 1:  Multiple locations?  Yes.  Since there are multiple hospital locations and as there is significant risk to patients if the system has problems then****it is a good idea to do a pilot installation****to minimise the risk, and use the results to iron out any problems with the installation.  Once you are confident that the system is working well then it could be rolled out to all other hospital locations.*

*Decision 2: Multiple functions? We have not been given information about the functionality, so we are unable to make a decision as to whether 'phased' is a valid option.  The first thing you would need to know is whether there are discrete bits of functionality that could be rolled out while the IT developers are still building the system.  If there is, a phased approach to installation is possible.  This is typically what happens with Agile development.  The only issue here is that any new functionality delivered has to be integrated with old functionality of the current systems, and sometimes the additional work involved in this is too time consuming, so they wait to roll out all the new functionality at the same time.*

*Decision 3:  Installation alternatives*

***PARALLEL vs DIRECT installation***

* ***Cost:****if there are cost constraints certain choices are not viable - no information given about this though generally speaking, while parallel is the safer option it is expensive.*
* ***System criticality:****if system failure is catastrophic for the organisation, then the safest approach should be selected regardless of cost.  As there could be significant patient issues if there are problems with the system then parallel would be better, as you could go back to the old system if there are problems to minimise the risk*
* ***Disruption to business:****what level of disruption to the company and IS operations is acceptable. We have not been given any information to make a decision.  However patient issues would cause disruption to the business, so parallel conversion would be advisable.*
* ***User computer experience:****the more experience the users have, the less necessary it is to delay changeover &****User resistance:****need to consider what the users are best able to cope with.  No information provided about this.*
* ***System complexity:****the more complex the system, the greater the chance of flaws ... a safer approach is better.  No information provided about this.*

***In conclusion, the recommendation would be:****Pilot and parallel approach at one hospital location to iron out all problems, before implementing at all other hospital locations using the parallel approach to minimise the risk of patient problems.  The following are incorrect because:*

* *Pilot and Phased - we do not have information to make a decision about phased*
* *Direct and Parallel - it has to be either direct OR parallel*
* *Pilot and Direct - this is too risky an option when patient issues can result if things go wrong*

///

Summative usability testing can best be described as:

Quantitative usability testing on the current version of the live site in order to establish a baseline for further quantitative benchmarking, is correct, as Summative testing takes place post implementation and can be quantitative testing of thing such as task times, completion rates and satisfaction. Finding and fixing usability problems with the user during the prototyping stage and Assessing an application for how usable it is at any time during development are incorrect as the evaluations are taking place during development rather than after the system has gone live. The developers using the live system in a lab with one way mirrors to conduct usability tests is not correct as it should be the users using the system during usability testing.

///