

Preamble

This material is released under a Creative Commons License agreement <http://creativecommons.org/licenses/by/2.0/uk/>. In informal terms we, the authors, give you permission to do most things with the material provided you:

- a) Acknowledge us as the authors of the original material, and retain the acknowledgement in copy of the material, and in any of your material based on ours,
- b) Do not pretend you wrote it,
- c) Do identify any changes you make as your work, not ours.

The material includes opinions of the authors, and may be different from work published by others. We have quoted and referenced work from third parties, and in a few cases have knowingly used or adapted ideas or content from third parties. Where appropriate we ask for permission from the third parties.

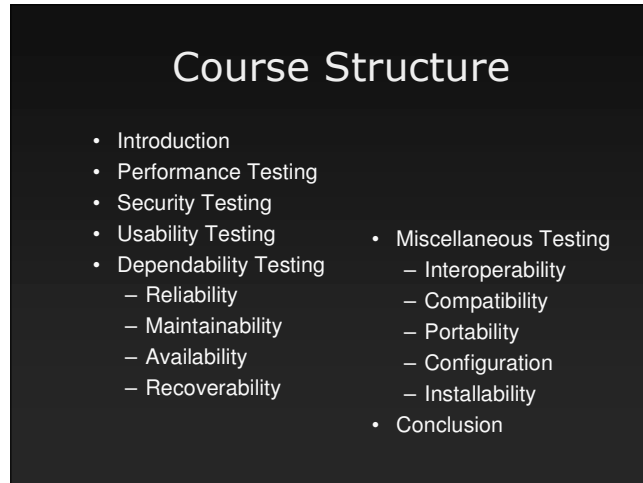
However we make mistakes, as most people do, and may have made mistakes in this work. If you spot something you feel is wrong, or simply something you feel could be improved please let us know via this email address nft.introduction@commercetest.com

We are Stuart Reid and Julian Harty, and are the authors of this material: **An introductory course on non-functional software testing**. If you wish to create derived works you need include the following acknowledgement: at the start and end of the derived works: **This work is based on original material by Stuart Reid and Julian Harty**. We like to know whether our work is being used so please tell us how you're using the content.

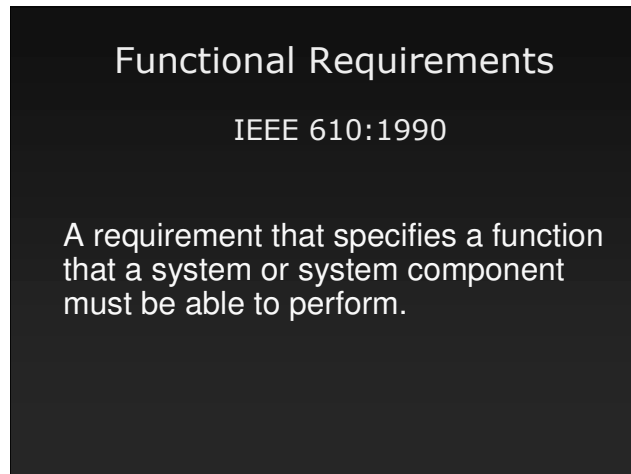
Finally, this work is still in a draft form so you may find inconsistencies, gaps, incomplete content, etc. We hope you will find the content useful anyway and we are very happy to receive suggestions for improvement to the email address mentioned above.

Introduction

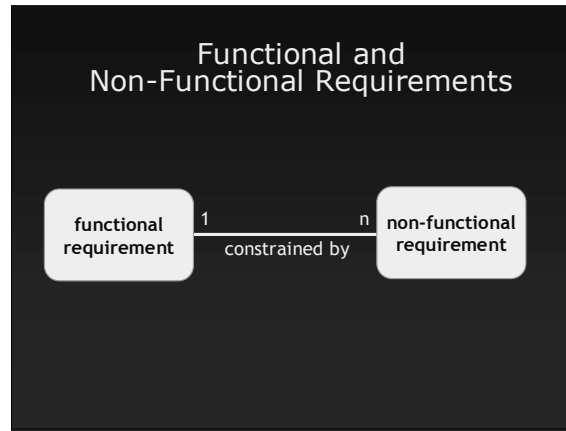
1. This course provides a brief introduction to non-functional testing. By the end of this course you should have an awareness of the importance of both the quality characteristics relevant to software-intensive systems and their evaluation using non-functional testing techniques.



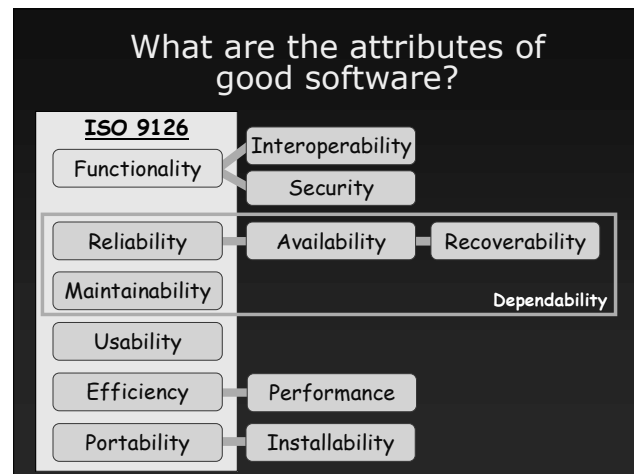
2. The course structure is shown above. It comprises this introduction to the topic, followed by five main sections covering the techniques for the testing of quality characteristics, and finally the conclusion.



3. The official definition above simply confirms that functional requirements specify *what* the system should do. Probably the easiest way to explain 'non-functional requirements' is that they specify all the remaining requirements not covered by the functional requirements. Non-functional requirements specify the system's 'quality characteristics' or 'quality attributes'. Non-functional testing is that testing concerned with the non-functional requirements.

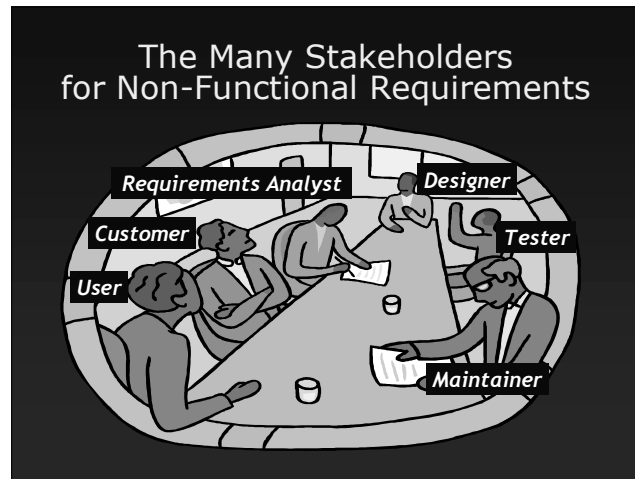


4. Building systems with only functional requirements is relatively easy, but ultimately pointless. No-one wants a slow, unreliable system that costs a fortune to modify. As shown above, the non-functional requirements act as constraints on the functional requirements, with sometimes several different constraints acting on a single functional requirement. For instance, a functional requirement to return some requested information could be constrained both by a requirement for the information to be returned within a certain time (performance) and for the function to be available to users 99% of the time (availability). As we will see later, where there are more than one constraint they often conflict with each other (e.g. it is difficult to perform a function 'as fast as possible' and 'as securely as possible' at the same time).



5. There are many quality attributes used as non-functional requirements for software systems. The ISO 9126 international standard lists six quality attributes for these systems as shown above (and these are then broken down further sub-attributes). It can be seen that this standard includes functionality as one of the six, and includes security and interoperability as sub-attributes of functionality. This classification is not widely-used and most people consider all quality attributes except pure functionality ('what' the software should do) as being potential non-functional requirements. We will therefore include security testing and interoperability testing as part of this course.
6. ISO 9126 uses the more general term 'efficiency' to include the industry standard term of performance, and also includes installability within the area of portability. The diagram groups the attributes of reliability, availability, maintainability and recoverability under the

umbrella term of dependability (as per the IEC standard), and these are covered in this course under that same grouping.



7. Potentially many different stakeholders have an interest in getting the non-functional requirements right. As can be seen from the diagram customers and users are shown separately. This is because for many large systems the people buying the system are completely different from those who are going to use it. The UK government procurement of a hospital booking system for the NHS is an example of where ignoring the users (in this case the GPs) has meant that the GPs are now having to be paid to encourage them to use a system to which they had little or no input before it was built.



8. There are many examples of projects underestimating the importance of non-functional requirements, and Victoria's Secret is typical, if perhaps higher profile than most of the others. Victoria's Secret is a web-based lingerie retailer, and one of their main market channels is through a webcast of their fashion show, in which scantily-clad supermodels take part. Its 1998 webcast was advertised during the Super Bowl and attracted an enormous number of visitors (presumably the football wasn't too interesting that year). However, with over 1 million hits in a very small timeframe the website could not cope and most 'viewers' were frustrated (due to lost connections and extremely slow response times) and had to return to watching the Super Bowl. For the following year's show Victoria's Secret invested \$9 million to ensure that the relatively simple *functional* requirement of providing a webcast was

adequately supported on the non-functional side so that website visitors were not turned away due to unacceptable quality characteristics such as poor response times.

Qualitative Specifications

- Don't ask for a system that is:
 - “Easy to use”
 - “Faster than a cheetah on speed”
 - “As reliable as possible”

9. A project's problems with non-functional requirements often start very early in the life cycle because in many cases they are not considered at all or only at a very superficial level. It can thus be a major achievement just to persuade projects that non-functional requirements are worthy of consideration, however many projects stumble at the next hurdle by specifying these requirements in such a manner that they are practically useless. Examples of this are shown above, where qualitative requirements are stated rather than quantitative ones.
10. The difficulty with purely qualitative requirements is that they are neither measurable nor testable – one person's “easy to use” may be another's worst nightmare. Ideally all non-functional requirements need to be specified in a way that makes them understandable by all of the stakeholders. One way of achieving this is by using what are known as ‘SMART’ requirements.

S-M-A-R-T

- **S**pecific
- **M**easurable
- **A**ceptable
- **R**ealisable
- **T**raceable

11. The ‘SMART’ acronym is used in a number of different situations to mean slightly different things. We are using it here specifically to describe a way of comprehensively specifying software requirements (it works for both functional and non-functional). By ‘specific’, we want requirements that are both precise and thorough, while ‘measurable’ should encourage the use of quantitative testable requirements that have been assigned an actual value to both

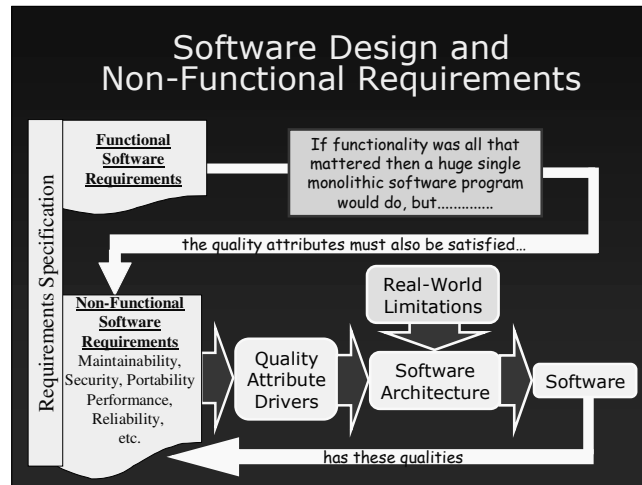
achieve and test against. ‘Acceptable’ requirements should satisfy the needs of the customers and users. The main point of ‘realisable’ requirements is that they should be realistic and achievable given the constraints of the project, while ‘traceable’ requirements should allow traceability both back to the user requirements and forwards to the subsequent implementation and tests.

S-M-A-R-T and Stakeholders					
Stakeholder	S	M	A	R	T
	Specific	Measurable	Acceptable	Realisable	Traceable
Customer	✓	✗	✓	✗	✓
User	✗	✗	✓	✓	✗
Req'ts Analyst	✓	✓	✓	✓	✓
Developer	✓	✗	✗	✓	✗
Tester	✓	✓	✗	✗	✗
Maintainer	✓	✗	✗	✗	✓

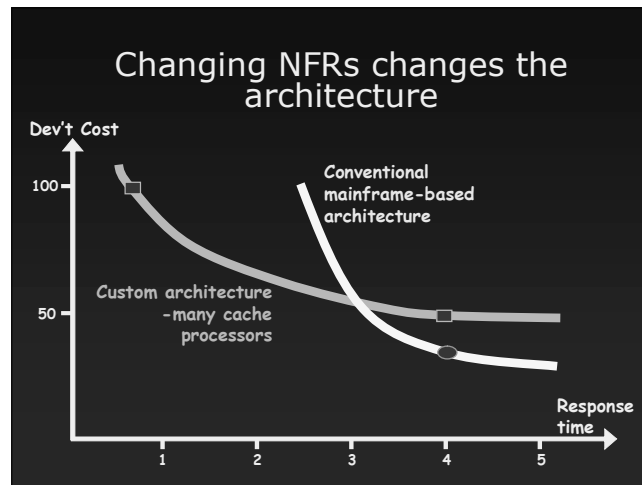
12. As can be seen above, the ‘SMART’ attributes are not necessarily all relevant to all stakeholders (although the above ‘black and white’ classification is not true for all situations). For instance, the tester wants precise and thorough descriptions of requirements to know what to test, and also wants these requirements to have a value assigned to them so that when they are testing it is clear whether, or not, the requirement has been achieved. Whether the requirements are acceptable to the customer, realisable by the developers and traceable to the original user requirements all tend to be secondary consideration for the tester.

Example NFR Specification	
ID	SR17.1
DESCRIPTION	The time required for the ATM to respond to a balance request
USER REQ	UR35.7
ATTRIBUTE	Performance
MEASURE	The time to respond in seconds
BASIC REQ	Max. ave. time to respond under typical system load = 10 secs Absolute max. time to respond = 20 secs
Ideal Req't	N/A
Current value	Current ATM ave. balance response time is 17 seconds
TECHNIQUE	Performance testing (see www.testingstandards.co.uk)
Test Req't	Test link to remote database required Performance and load testing tools required
Operational Req't	Manual test. Initially after 3 months, then every 6 months
CONFLICTS	The checking to ensure the link to remote database is secure (see SR23.1B)
PRIORITY	7 (on range 1-10 for this project)

13. Provided above is an example of the specification of a non-functional requirement (in this case it is a performance requirement) that follows the ‘SMART’ criteria. By using a template for the specification of non-functional requirements, a consistent level of definition can be more easily achieved across a project.

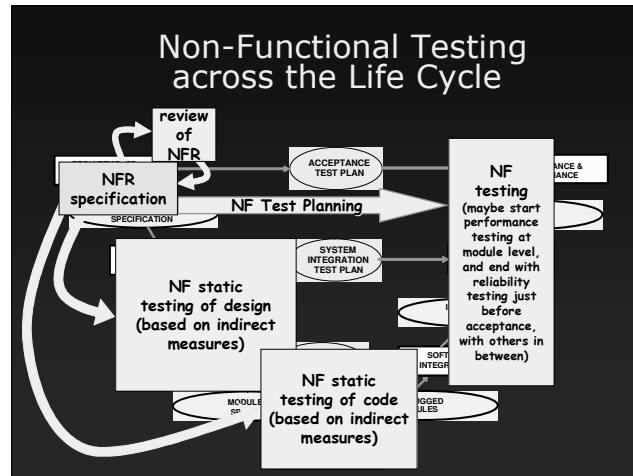


14. Although specified during the requirements analysis phase of the software life cycle, it is during the next (design) phase that the non-functional requirements are used to drive the software development. The major 'target' of these requirements is the software architecture, which defines the overall structure of the software. Examples of architectural choices made by software designers (or architects) could include the level of redundancy to build into a system to achieve the required level of reliability, or the choice of language to support the required maintainability.

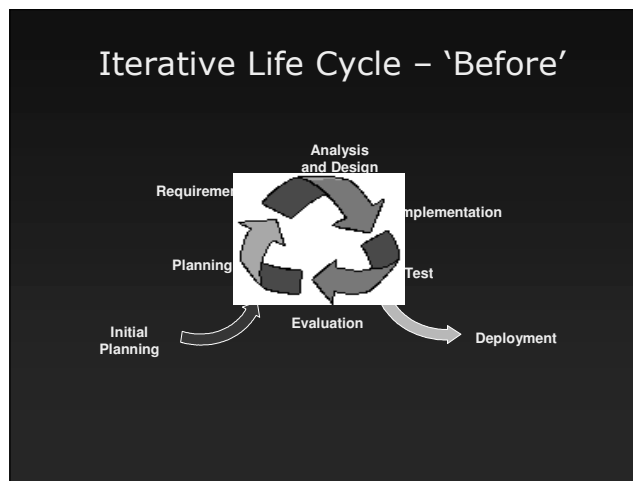


15. In the example above, from Barry Boehm's paper on 'Unifying Software Engineering and Systems Engineering', the original performance requirement for a query and analysis system was for a response time of no more than 1 second. To achieve this exacting requirement the developers found that they would have to create a bespoke architecture, at a commensurate cost. This led to further analysis of the requirement, which led them identify that a 4 second response time would satisfy users 90% of the time. Although achievable with the bespoke architecture, it was also determined that a conventional architecture could satisfy this requirement cheaper, and, presumably with better potential maintainability.

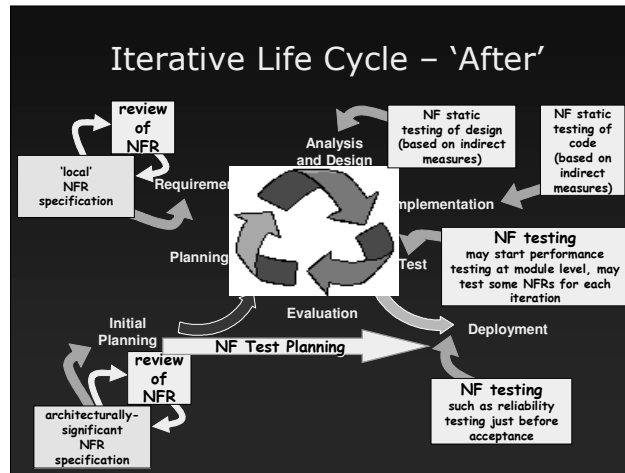
enough to act as baselines before the next activity begins. Second, the close relationship between development activities down the left and test and integration activities up the right can be clearly seen.



19. We can superimpose the non-functional testing activities on the V-model, as shown above. Initially testing supports the review of the specification of non-functional requirements (NFR), and an extra output of this stage is the test planning required for later dynamic non-functional testing. Static testing of design attributes, such as the coupling and cohesion to indirectly suggest future maintainability, may be performed once designs become available. Similarly, static testing of code, such as to detect the use of disallowed coding constructs such as 'goto's' could also point to potential difficulties meeting maintainability requirements. Once programs becomes available, then 'traditional' dynamic non-functional testing can start, perhaps even at the individual module level for some attributes, but more likely at the system test level.



20. As with the linear life cycle, we can similarly show that the activities associated with non-functional testing can be applied across all of an iterative life cycle model (shown above).



21. With the iterative life cycle, the same non-functional testing activities apply as before, but, as can be seen above, they are applied both for every iteration, as well as being applied at a higher level to the initial planning and final deployment activities.
22. When specifying non-functional requirements we should consider *all* the different quality characteristics, and then specify those that are relevant to our system. We find, however, that in many cases the design that helps achieve one attribute also militates against another attribute. If both have been specified we have a problem – the two requirements conflict. Conversely, we also find that occasionally some characteristics can complement each other, where the design necessary to achieve one characteristic also helps achieve another.
23. As an example of conflicting requirements, consider a safety-critical software application that many people occasionally rely on, a flight control system for a passenger aircraft. As passengers we want these systems to be highly reliable, while the manufacturers also want them to use as few computing resources (processor power, memory, etc.) as possible to keep their hardware costs as low as possible. The only way we know to make flight control systems as safe as they are is to build large amounts of redundancy into the software. This redundancy will mean that critical functions will be duplicated, triplicated, etc., and outputs from each individually-developed function compared to ensure no single point of failure causes a catastrophe. The difficulty with all this redundancy is that the extra processing consumes extra computing resources. Thus, the two requirements conflict. In fact, a non-functional requirement of high reliability generally conflicts with several other quality characteristics, such as performance and maintainability, because of the extra complexity required by the fault tolerance built in to the system.
24. If we consider commercial systems, then a typical conflict arises between the performance of a system and the cost of the system. Users nearly always want systems that respond faster, while customers (who may differ from the users) nearly always want a system that costs as little as possible.

An Example Quality Profile

	Less ← IMPORTANCE → More			
	D	C	B	A
Functionality			✓	
Reliability			✓	
Usability				✓
Efficiency		✓		
Maintainability	✓			
Portability	✓			

25. When specifying the non-functional requirements for a system, we must therefore strike a balance between the different relevant quality characteristics. One approach to this suggested in Erik van Veenendaal's paper on 'Measuring Software Quality' is to create a 'Quality Profile' for the system, as shown above. With this approach each of the quality characteristics of the system are rated in terms of their importance to the project, creating a 'profile' that indicates which characteristics should be given priority over others. In the example above, the quality characteristics are taken directly from ISO 9126, and rated on a four point scale, but obviously other sets of characteristics and scales of importance can be used dependent on the situation.

**Summary:
Introduction to
Non-Functional Testing**

- What is a Non-Functional Requirement
- The Stakeholders
- Specification of Non-Functional Requirements
- Quality Attributes as Design Drivers
- Testing throughout the Life Cycle
- Conflicting Non-Functional Requirements
- Prioritisation of Non-Functional Requirements

26. This introduction to the topic of non-functional testing has provided a brief explanation of the importance and specification of non-functional requirements. It has also presented the idea that non-functional testers should be involved from the start of the project when these quality attributes are initially specified.
1. The remainder of the course comprises sections covering the testing of specific quality characteristics of software-intensive systems.

Self-Assessment Questions

- Q1. Which one of the following statements is correct?
- (a) Functional requirements are constrained by non-functional requirements.
 - (b) Non-functional requirements are a subset of functional requirements.
 - (c) Functional requirements are also known as quality characteristics.
 - (d) Functional requirements are also known as quality attributes.
- Q2. Which quality attributes does ISO 9126 include as part of functionality?
- (a) Interoperability and security.
 - (b) Availability and reliability.
 - (c) Interoperability and availability.
 - (d) Security and reliability.
- Q3. Which of the following quality attributes is not normally included as part of dependability?
- (a) Security.
 - (b) Maintainability.
 - (c) Recoverability
 - (d) Availability.
 - (e) Reliability.
- Q4. Which one of the following stakeholders would normally have the least interest in the non-functional requirements?
- (a) Programmer.
 - (b) Requirements analyst.
 - (c) Customer.
 - (d) Tester.
 - (e) User.
 - (f) Maintainer.
- Q5. Which one of the following excerpts from statements of non-functional requirements is the least useful?
- (a) "...acceptable to the users..."
 - (b) "...response times of less than 1 second...."
 - (c) "...reliability of 0.93...."
 - (d) "...portable to Windows XP Pro with 3 days' effort..."

- Q6. Which one of the following is not a part of the 'SMART' acronym for specifying software requirements?
- (a) Attainable.
 - (b) Measurable.
 - (c) Specific.
 - (d) Traceable.
 - (e) Realisable.
- Q7. Which one of the following is not a part of the 'SMART' acronym for specifying software requirements?
- (a) Time-bounded.
 - (b) Measurable.
 - (c) Specific.
 - (d) Acceptable.
 - (e) Realisable.
- Q8. Which one of the following parts of the 'SMART' acronym for specifying software requirements is least likely to be of interest to the tester?
- (a) Realisable.
 - (b) Specific.
 - (c) Measurable.
- Q9. Which one of the following is the least likely benefit of the early involvement of testers with the specification of non-functional requirements?
- (a) Better tester remuneration.
 - (b) Ensuring testable requirements.
 - (c) Lending expertise to reviews.
 - (d) Early non-functional test planning.
- Q10. Which one of the following statements is correct?
- (a) Non-functional testing can be applied across the life cycle.
 - (b) Different non-functional testing activities are required for linear and iterative life cycles.
 - (c) The 'V-model' was a German WWII weapon of terror.
 - (d) The 'V-model' is an iterative life cycle model.

Q11. Which one of the following quality attributes is most likely to conflict with a requirement for high reliability?

- (a) Fast response times.
- (b) High availability.
- (c) Good usability.
- (d) Ease of installation.

Q12. Which one of the following is most likely to be used to cope with conflicts between non-functional requirements?

- (a) Prioritisation of quality characteristics.
- (b) Cost of implementation.
- (c) Keeping the customer happy.
- (d) Selection based on ease of testing.

Self-Assessment Answers

Note that the correct answer for all questions is currently set to (a).

Q1. Which one of the following statements is correct?

(a) Functional requirements are constrained by non-functional requirements.

Good.

(b) Non-functional requirements are a subset of functional requirements.

No – non-functional requirements are normally considered as all those requirements not covered as functional requirements.

(c) Functional requirements are also known as quality characteristics.

No – ‘quality characteristics’ is another term used to describe non-functional requirements.

(d) Functional requirements are also known as quality attributes.

No – ‘quality attributes’ is another term used to describe non-functional requirements.

Q2. Which quality attributes does ISO 9126 include as part of functionality?

(a) Interoperability and security.

Good.

(b) Availability and reliability.

No.

(c) Interoperability and availability.

No.

(d) Security and reliability.

No.

Q3. Which of the following quality attributes is not normally included as part of dependability?

(a) Security.

Good.

(b) Maintainability.

No – this is probably the most important attribute within dependability according to the IEC definition.

(c) Recoverability

No – this attribute contributes to availability as the quicker a system recovers from a failure the less time it is unavailable.

(d) Availability.

No – dependability is based on whether users can rely on the system, so availability is one of the key attributes.

(e) Reliability.

No - dependability is based on whether users can rely on the system, so reliability is one of the key attributes.

Q4. Which one of the following stakeholders would normally have the least interest in the non-functional requirements?

(a) Programmer.

Good.

(b) Requirements analyst.

No – the requirements analyst is the author of the requirements specification, a key part of which will be the non-functional requirements.

(c) Customer.

No – the customer must assure themselves that those non-functional requirements they feel are relevant to the system have been specified and are later implemented in the system.

(d) Tester.

No – The tester will be aware that much of system testing is concerned with non-functional requirements and that getting them right ‘up front’ will save time and effort overall.

(e) User.

No – The user will be most affected if the non-functional requirements are wrong, and will often have different views from the customer.

(f) Maintainer.

No – the maintainer will find their task far harder if non-functional requirements are ignored as this will lead to assumptions by the designers and the likelihood that specific requirements in this area are left implicit rather than explicitly stated.

Q5. Which one of the following excerpts from statements of non-functional requirements is the least useful?

(a) "...acceptable to the users..."

Good – this is purely qualitative (no value provided), so very little use.

(b) "...response times of less than 1 second...."

No – this is quantitative (an actual value has been provided), so could well be useful.

(c) "...reliability of 0.93...."

No – this is quantitative (an actual value has been provided), so could well be useful.

(d) "...portable to Windows XP Pro with 3 days' effort..."

No – this is quantitative (an actual value has been provided), so could well be useful.

Q6. Which one of the following is not a part of the 'SMART' acronym for specifying software requirements?

(a) Attainable.

Good – the actual quality starting with 'A' is Acceptable.

(b) Measurable.

No – measurable is a necessary characteristic as it makes the requirements testable.

(c) Specific.

No – we want requirements that are precise and thorough.

(d) Traceable.

No – we want to be able to trace our requirements back to the user requirements and forward to the implementation and tests.

(e) Realisable.

No – we want realistic, achievable requirements.

Q7. Which one of the following is not a part of the 'SMART' acronym for specifying software requirements?

(a) Time-bounded.

Good – when used for personalised objective setting the 'T' often represents time-bounded, but this is not appropriate for software requirements.

(b) Measurable.

No – measurable is a necessary characteristic as it makes the requirements testable.

(c) Specific.

No – we want requirements that are precise and thorough.

(d) Acceptable.

No – we want requirements that satisfy the needs of the user and customer.

(e) Realisable.

No – we want realistic, achievable requirements.

Q8. Which one of the following parts of the ‘SMART’ acronym for specifying software requirements is least likely to be of interest to the tester?

(a) Realisable.

Good – the tester does not need to get involved in the underlying technology unless necessary.

(b) Specific.

No - requirements that are precise and thorough provided a solid basis for testing.

(c) Measurable.

No – measurable requirements are more likely to be testable.

Q9. Which one of the following is the least likely benefit of the early involvement of testers with the specification of non-functional requirements?

(a) Better tester remuneration.

Good – a nice idea but unlikely.

(b) Ensuring testable requirements.

No – this is a positive benefit as untestable requirements cannot be checked.

(c) Lending expertise to reviews.

No – this is a positive benefit as better reviews will mean less faults.

(d) Early non-functional test planning.

No - this is a positive benefit a otherwise planning for non-functional testing can be left too late.

Q10. Which one of the following statements is correct?

(a) Non-functional testing can be applied across the life cycle.

Good.

(b) Different non-functional testing activities are required for linear and iterative life cycles.

No – the same activities are required whichever life cycle is used, but in the iterative life cycle they are applied more times to smaller components.

(c) The ‘V-model’ was a German WWII weapon of terror.

No – doodle bugs, etc. were not models.

(d) The ‘V-model’ is an iterative life cycle model.

No - The ‘V-model’ is a linear life cycle model.

Q11. Which one of the following quality attributes is most likely to conflict with a requirement for high reliability?

(a) Fast response times.

Good – the necessary redundancy required for high reliability will inevitably slow the system down.

(b) High availability.

No – many of the same design and programming approaches for achieving high availability are the same as those to achieve high reliability.

(c) Good usability.

No – the extra effort put into high reliability systems is also often reflected in the high quality of the user interfaces.

(d) Ease of installation.

No – the extra effort put into high reliability systems is also often reflected in the high quality of the installability.

-- End of document --