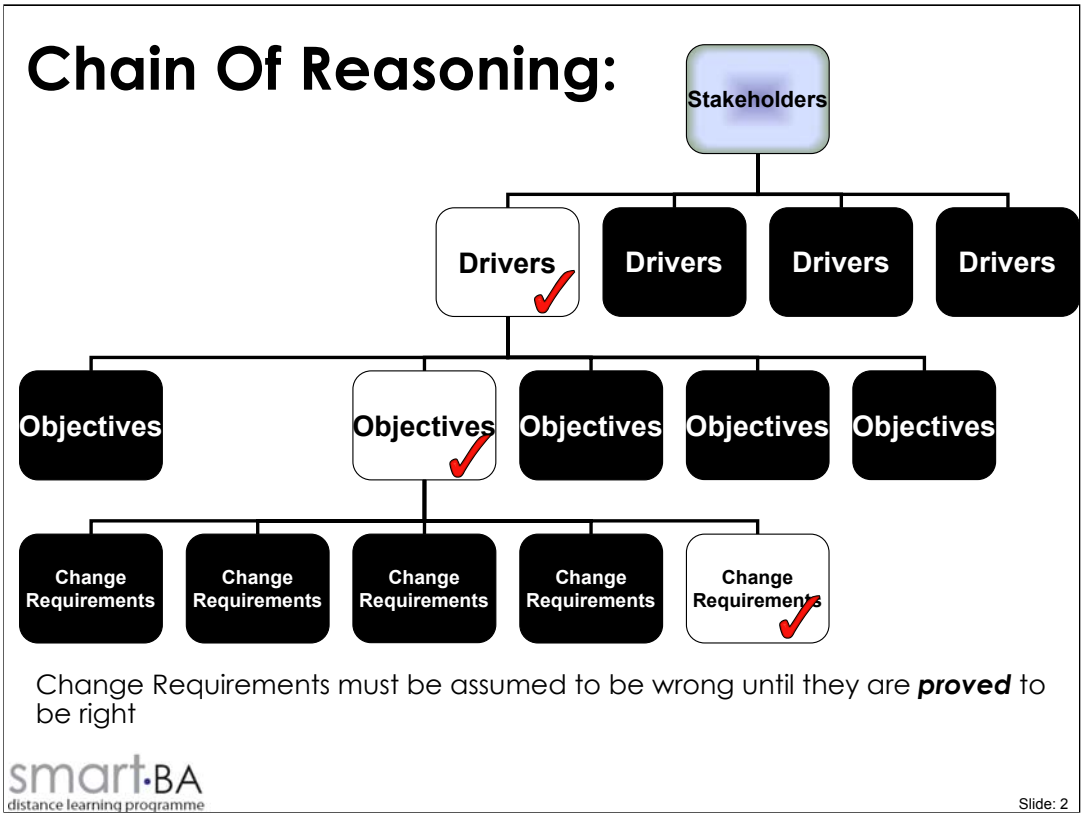
smart·BA Distance Learning Programme

Module 3:
Functional Requirements

smart·BA
distance learning programme

Hello and welcome to the smart BA distance learning programme Module 3 – in this module you are going to analyse the requirements of the solution you are working on.

Before we get in to that, let's just refresh ourselves about where we are in the chain of reasoning.



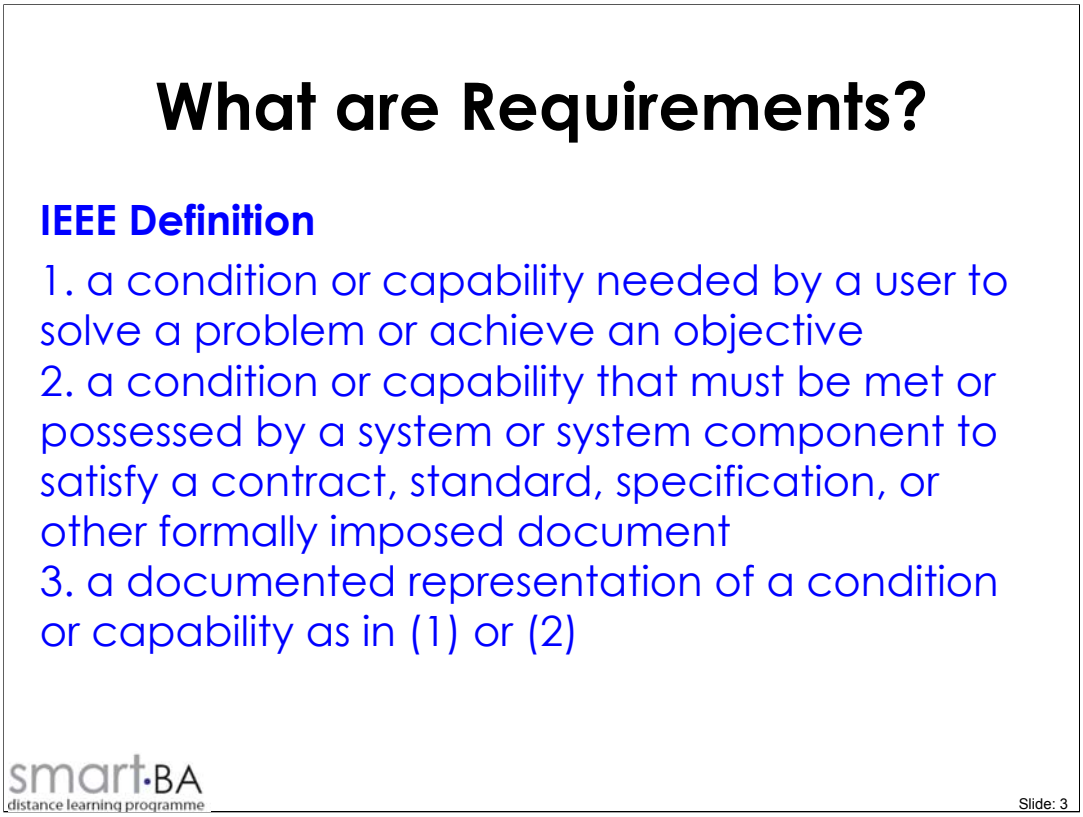
What is the project is trying to achieve? It is trying to achieve the objectives which prove that the drivers have been addressed …

The next set of questions to be answered is how will it achieve the objectives? What will need to be there in order for the objectives to be realised?

There are many types and levels of requirements and these are the subject of the next 5 modules.

What you need to bear in mind for this module is that we will be answering the question – “what changes will your project make that will deliver the objectives?”.

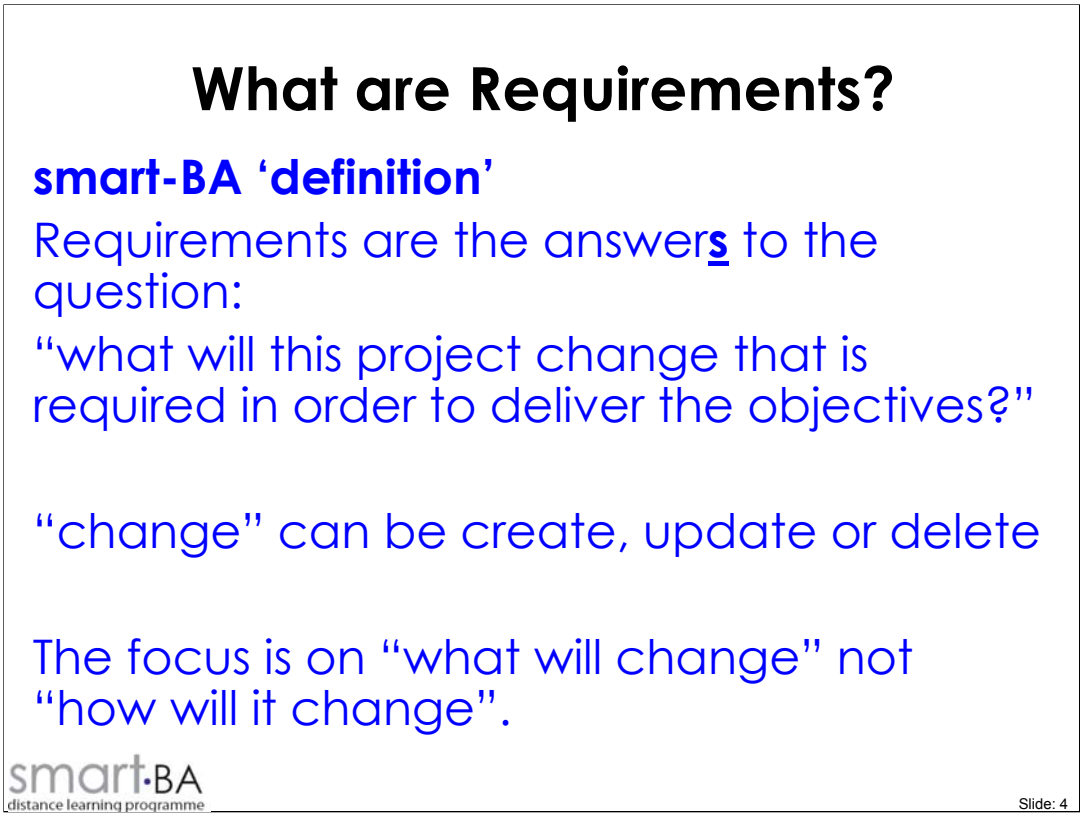
And remember: as there an infinite number of requirements that don’t support achieving project objectives and only a small set that do, so requirements must always be assumed to wrong - in the sense they don’t help achieve objectives – until they can be proved to be right.



So what is a requirement?

The Institute of Electricians and Electrical Engineers (IEEE) have a definition that has been widely accepted…

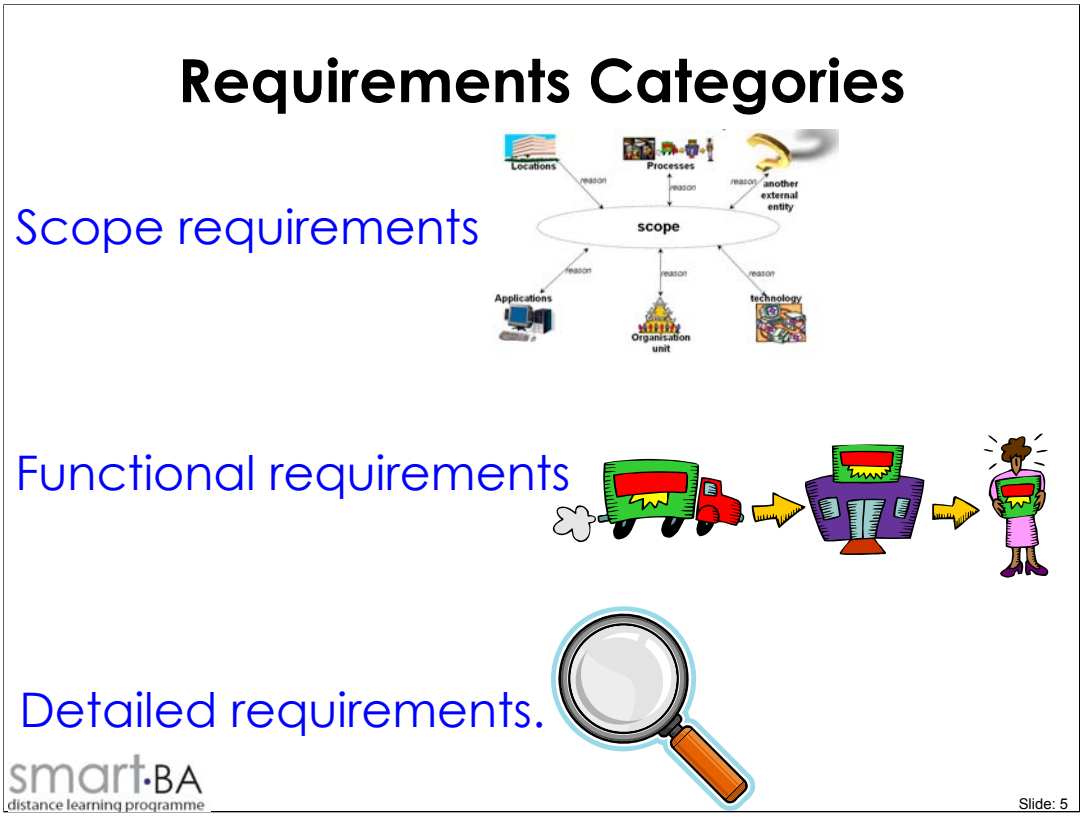
1. a condition or capability needed by a user to solve a problem or achieve an objective
2. a condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document
3. a documented representation of a condition or capability as in (1) or (2)



We would suggest a more pragmatic definition that will help you as you try to identify requirements in your project.

So, requirements are precise statements that answer a particular question and do no more than that. Should be easy…

Let's look at some examples…



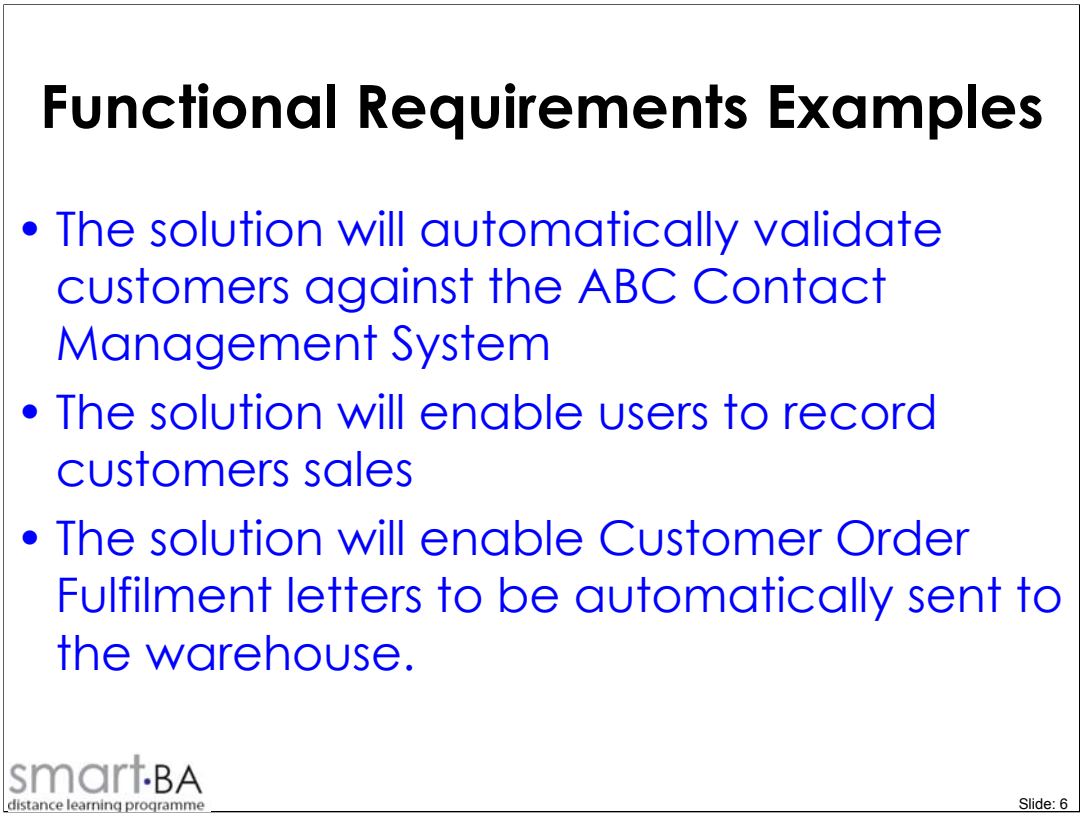
There are lots of different ways of categorising requirements – such as those required by the BCS ISEB Diploma in Business Analysis categorisation.

However, what do we **need** in order to be able analyse all of a solution's requirements? We need to be able to analyse and specify

1. Scope and Context of the solution (and we did this in module 3) – this tells us who and what is impacted by the solution, and where they are and whether they are actually being changed by the solution or just interacting with it

2. what the solution needs to be able to do – functional requirements – (and that is the subject of this module)

3. What rules must be incorporated in to the solution terms of
    1. Process flows (module 5)
    2. Process logic (module 7)
    3. Process performance (module 7)
    4. Process access (module 7)
    5. Data relationships (module 6)
    6. Data attributes (module 8)
    7. Data performance (module 8)
    8. Data history (module 8)

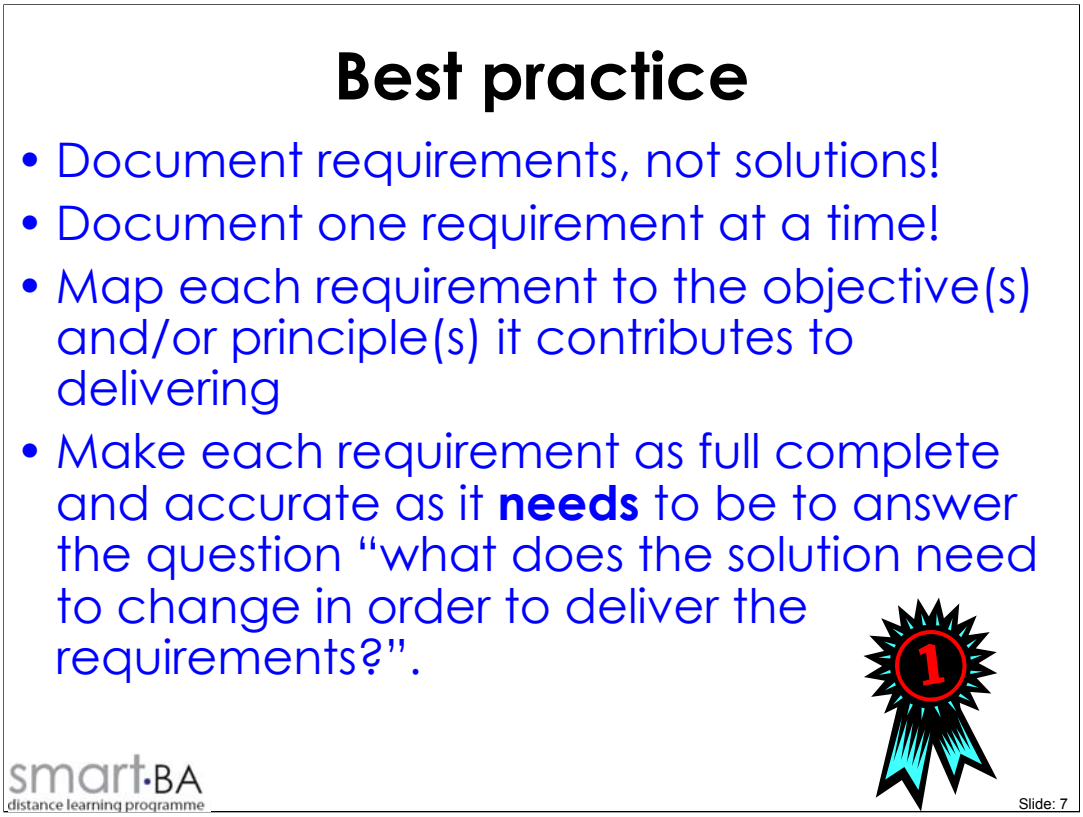


Do you remember the example we were working on in Module 3? It was all about supplying a solution that allowed users to record sales – they used data from the ABC Contact Management System and order fulfilment messages were sent to the warehouse…

These are some functional requirements for that project.

Note that they are not long, complex statements. They are short and as simple as possible.

Think of them as instructions that you are using to instruct stakeholders, users, designers, testers, trainers and anyone else who needs to know on what the solution will be capable of doing.



Here are some characteristics of good functional requirements:

1. There are no elements of the technical or otherwise solution in there **unless it scopes the requirement**. For example, "The solution will enable Customer Order Fulfilment letters to be automatically sent to the warehouse". Letters are a solution. If it is true that the solution must produce *letters* in order for the warehouse to be able to work then this should be recorded as a constraint (that letters must be used, not emails etc) and this requirement is correct. If it is not true that letters have to be used then the requirement should be reworded to take the solution out – for example "The solution will enable Customer Order Fulfilment *messages* to be automatically sent to the warehouse". Changing the word "letters" to "messages" allows the solutions designers to think about different ways of getting the message to the warehouse.
2. There is only 1 functional requirement per statement – if you see joining words like 'and', 'or', 'but', 'however' etc or there are full-stops then it is an **indicator (not a rule)** that there is more than one requirement being documented. You cannot hope to track, prioritise or identify who generated the requirement if one requirements statement has more than one requirement in it. For example, "the solution will record and report on sales". Either record and report are the same thing (in which case use just one word) or they are different and should be in 2 requirements: "record sales" and "report on sales". Note that "record sales" seems pretty fundamental to the project but we don't know at this stage if "report on sales" is required in order to deliver the objectives. Having more than one requirement in one requirement statement means we can not assess them individually.
3. They only answer the question "what will the solution change in order to achieve the project objectives?" The fact a requirement contributes towards achieving one or more objectives and/or principles can be proved by mapping the functional requirements to the objectives and/or principles that it contributes towards achieving. For example it is highly likely that the requirement "the solution will enable users to record sales" will contribute to at least one of the objectives of the "record sales" solution!
4. They are as full, definitive, accurate and complete as possible but only in order to answer the question we are analysing when analysing functional requirements – namely "what does the solution need to change in order to deliver the requirements?". If you know some information that clarifies or scopes the requirements then state it. If you do not know it for a fact as defined by your killer stakeholders then do not define it.

Examples of poor functional requirements

1. Be able to use diary functionality
2. Be able to flag premium customers
3. Be able to track and report on sales
4. Increase accuracy of sales information
5. Allow authorised users of team-leader and above to cancel sales orders
6. Prompt the owner of the sales order to notify the customer of cancelled sales orders.

smart·BA distance learning programme

Slide: 8

Let's look at what may seem like some reasonably good requirements but aren't.

1. There are words like 'diary' and 'workflow' and workqueue' and 'process' and 'maintain' that can mean lots of different things all summed up in to one word. Does 'diary' in this context mean a history of things that **have** happened or a schedule of things that **will** need to happen or **both**? Or something else for that matter. Good requirements state a single capability that is needed as unambiguously as possible.

2. Sometimes solutions creep in without being noticed. **Why** does the requirement originator want a flag to identify premium customers? They probably expect that they can do whatever they actually want to do if they have this flag. The flag is a solution enabling them to do something else. Find out what they would accomplish with the flag and you might find out what the functional requirement actually is.

3. There are two functional requirements in this requirement: be able to track sales and to be able to report on sales. Why split them out? Because if they are not split out how can each one

-Be mapped to the objectives and/or principles they support to **prove** they are required and so be prioritised

-Be tracked to who originated it in case it needs to change

-Be tested that it has been delivered

-…and so on

4. What functionality or capability is being specified here? None! This is an objective (if accuracy can be measured) or principle (if it can not).

5. There is a functional requirement here: be able "to cancel sales orders". All functional requirements will have rules about who can use the functionality and they are specified in the process access rules we will cover in module 7. Functional requirements are about specifying what can be done, not by who or under what rules or conditions.

6. This looks similar to the previous example but it is worse than that – it is wrong! The requirement assumes that that the owner of the sales order CAN be notified – suppose they are ill, on holiday, left the company? This **rule** that specifies notifying an owner belongs with all the other rules about owner availability in the detailed requirements process logic you will detail in module 7.

Common mistakes

- Designing the solution
- Unjustified requirements
- Putting in unjustified extra information
- Not putting sufficient detail in
- Protecting requirements – ego fuelled analysis!

smart·BA distance learning programme

Slide: 9

Here are the most common errors we see with requirements specifications

1. Specifying solutions not requirements. For example saying how communication will be done with a customer, not just that the solution needs to communicate with them and let the designers do their job: design the best solution to the requirement. The result is a high risk of developing things that do not need to be developed because there were better ways of doing it but the designers were not able to propose them given the 'requirements'!

2. Requirements that are not mapped to an objective or principle. These may be justified on the grounds of 'future-proofing' or 'good practice' but the reality is (assuming that your objectives and principles don't have these items) that the solution will not be assessed on this at all by anyone (if your analysis of objectives is correct). The result is that anyone can start proposing lots more requirements justified on the grounds of future proofing and each requirement will need processes developing, development, testing, training and so on – all this effort will be expended on requirements that do not contribute in any way whatsoever to success as defined by your killer stakeholders.

3. Adding in rules or examples or scoping statements or objectives – anything in fact except the requirement. For example "the solution will increase sales by getting the person who took the sale or their line manager to flag it as important"! The result will be a huge number of 'requirements' that will be impossible to prove are correct.

4. Leaving out known information that is required to fully, accurately and definitively state the requirement. For example omitting that only sales for electrical goods are to be recorded. Do not think that this will come out later on – it may or it may not. If you know it now, and it is proveably correct and it is justifiably part of the requirement then document it as such. Failure to do so will result in re-doing the analysis that led to the conclusion in the first place or – worse – forgetting it all together!

5. Some business analysts will start to defend the requirements they have documented when the requirement is challenged instead of trying to find out the truth of the matter. Business Analysts do not own requirements, they just document them. Properly! The result of ego-fuelled analysis is solutions which the BA thinks are great, the users find unusable and the killer stakeholders reject!

Functional Requirement Prioritisation

- **M**ust have requirement
- o
- **S**hould have requirement
- **C**ould have requirement
- o
- **W**ish list requirement

smart·BA
distance learning programme

Slide: 10

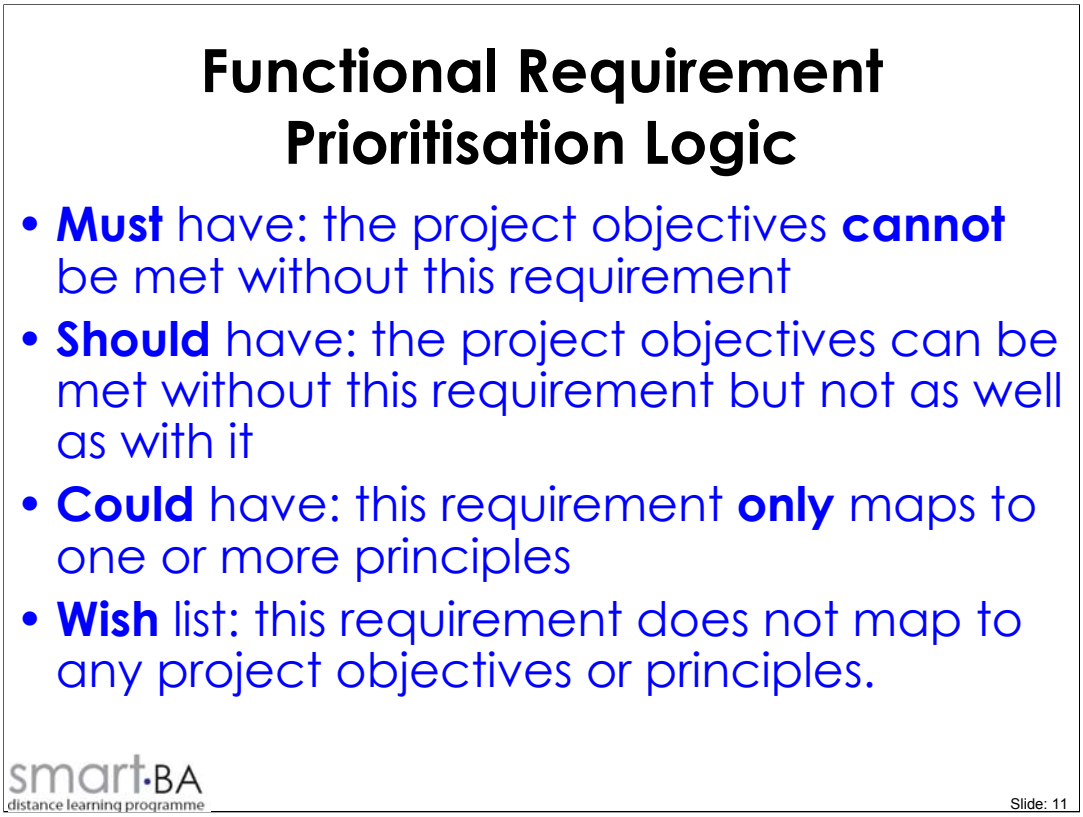
Sometimes, a project cannot implement all the requirements for a whole variety of reasons. Once you have established that some requirements will have to be dropped, the question arises of which requirements can be dropped safely and still ensure that the project will achieve the objectives?

A technique for managing this you may have come across is known as MoSCoW prioritisation for requirements. This allows you to juggle which requirements to keep and which to drop.

It is a 'must have', 'should have', 'could have', 'wish list' categorisation of requirements. "Must have" requirements are kept, "wish list" requirements can be dropped and those in-between debated.

Traditionally this prioritisation has been done by asking the killer stakeholders and users how they would categorise their requirements. In other words, the prioritisation is done **by** them.

However, there is some logic you can use to ***tell*** the killer stakeholders and users what the priority of each and every requirement is ***if*** what they have told you so far has been true. In other words, the prioritisation is **calculated** by **you!**



Must have: the project objectives **cannot** be met without this requirement. This is a straight forward decision: can the project achieve success without this requirement? Yes or no. These requirements tend to be few and very obvious. For example, in the "record sales" solution if the functional requirement to be able to record sales is dropped, the project can in no way succeed.

Should have: the project objectives can be met without this requirement but not as well as with it. Most functional requirements will fall in to this category. They contribute to objectives but without them the objectives can still be achieved. The priority of a functional requirement in this category can change. Suppose that a project is being forced to drop requirements. There comes a point where dropping a "should have" requirement means the project cannot achieve its objectives and will fail. In other words that requirement **became** a "must have" requirement as a result of other "should have" requirements being dropped.

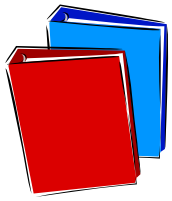
Could have: this requirement **only** maps to one or more principles. So the project can still achieve success – can still achieve its objectives – without this requirement. The principle(s) the requirement contributes to will be never be assessed so this requirement can 'safely' be dropped as only unassessed principles will be affected.

Wish list: this requirement does not map to any project objectives or principles. There is no hard or soft benefit to implementing this requirement – it is only being done as favour or is one someone's wish list. This requirement can be dropped with absolutely no negative impact on the project or solution.

So – not only can you prioritise the requirements – only the "should have" can be debated: the "must have" must be kept, the "wish list" and "could have" requirements can safely be dropped if the project needs to.

Functional Requirement – where do they come from?

- Stakeholders
  - Interviews
  - Workshops
  - 'Casual' communications
- Constraining standards and procedures
  - Documents
  - Interviews
  - workshops
- Business Analysts!
  - All the time
  - Any way that is needed



smart·BA distance learning programme

Slide: 12

You get requirements from anywhere and everywhere in order to do the best job you can in ensuring that the objectives are delivered. Here are a couple of common sources…

Stakeholders – all types – through interviews and workshops and other unstructured communications such as emails and phone calls. Stakeholders will not differentiate functional requirements from objectives from detailed rules – for example a stakeholder might tell you that it's a nightmare trying to record sales for premium customers who are in bad debt with company as the debt has to be cleared before placing the order and too many sales like that are getting through! In that statement was the requirement to record sales, a rule specifying when that that sale can be placed and an objective to reduce the number of sales to customers in bad debt! Your job is to isolate and document the different information they impart. And then prove each piece of information to be correct.

Quite often there will be laws of the land, and internal to your organisation standards and procedures that have to be adhered to with varying consequences of failure to do so. You need to find that information any way you can but they must be documented somewhere. However, that documentation may be large and/or highly technical so interviews and workshops with experts might be called for.

And of course you – you will be thinking all the time of candidate requirements through observation and thinking and from documenting other requirements. For example, you might start to identify a requirement to be able to put on hold sales for premium customers in bad debt!

Assignment Task 1:
Document the solution functional requirements

- Use the functional requirements templates in the supporting documentation pack – add to documentation from module 2
- Map which objectives and/or principles they contribute to
- Prioritise them
- Challenge them yourself
- Get them challenged by killer stakeholders
- Get them signed off.

smart·BA
distance learning programme

Slide: 13

Your first task in this module then is analyse the functional requirements fro your project's solution.

Use the templates that accompany this module to document the products of your analysis. Copy the relevant worksheets to the analysis deliverables spreadsheet from module 2 to keep all the analysis together in one spreadsheet.

Make sure you map them to all the objectives and principles that they contribute to.

Challenge them and every part of them until you are happy they are correct.

When you are confident that they are right, get them challenged by all the killer stakeholders that are entitled to determine the objectives, principles, scope and functional requirements of the solution.

When they are happy with them, get them to sign them off.

…and finally

- Dependencies
  - What processes/factors/groups/people are you dependent on and for what?
- Constraints
  - What constrains your activities? In what way? How much?
- Issues
  - What factors have arisen that are impeding your work which are outside of your control? How do they impede you?
- Assumptions
  - What assumptions have you made – and why (to work around an issue?)
- Risks
  - What might go wrong (maybe as a result of the issues you identify), how likely is it they will occur and what will be the impact?

smart·BA distance learning programme

Slide: 14

*As always! There is one more set of information we need to analyse and this is a set of elements that can arise at any time but varies in subject matter by what you are analysing. So for this module we are interested in these elements as they relate to requirements **only**.*

Assignment Task 2:
Document the dependencies, constraints, issues, assumptions and risks you discover

- Use the relevant templates in the supporting documentation pack for this module to document them
- Analyse the material impact they have on the project and/or solution
- Challenge them
- Escalate them as required.

smart·BA
distance learning programme

Slide: 15

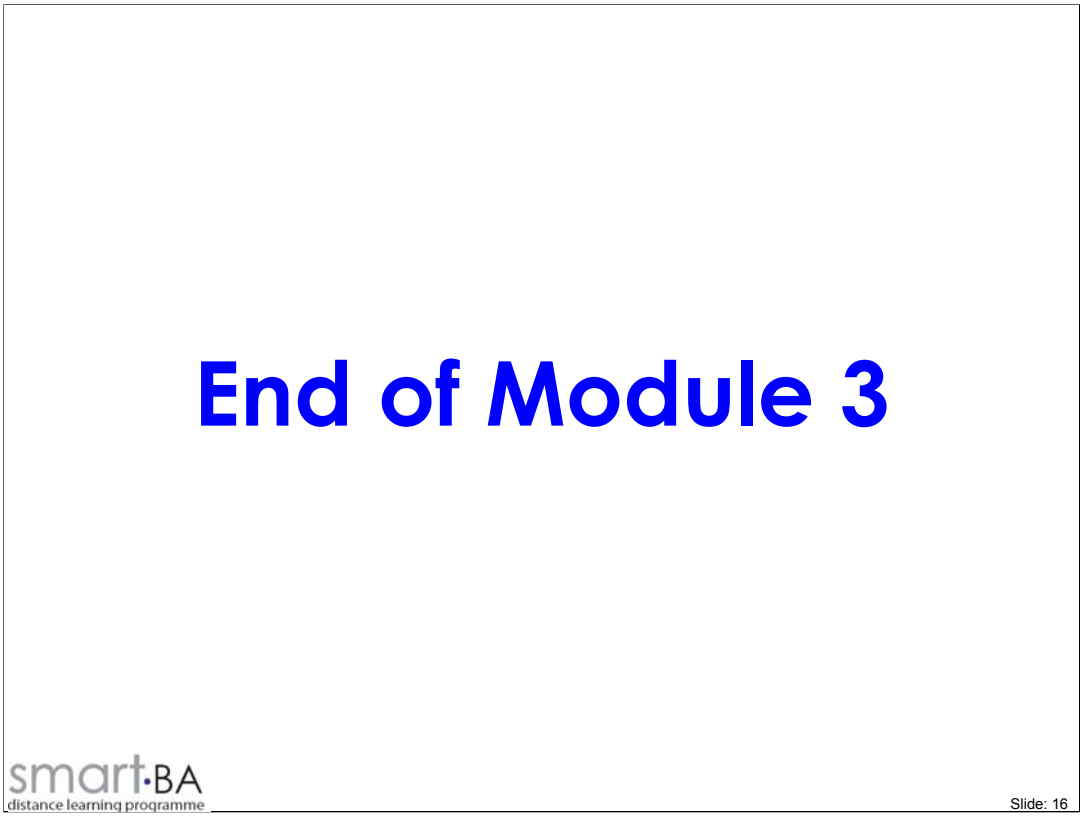
Your final task in this module then is analyse the project's dependencies, constraints, issues, assumptions and risks.

Use the templates in the material that accompanies this module to document them.

Don't document them for the sake of documenting them: analyse the material impact they have on the project.

Challenge them and every part of them until you are happy they are correct.

When you are confident that they are right, escalate any that need escalating (particularly risks, issues and assumptions) – probably to the Project Manager.



That is the end of the introduction to Module 3. I look forward to receiving your analysis deliverables!