Nick Forleo

**Chapter 6**

6.1 What can be some pitfalls to consider when ranking requirements?

Some pitfalls the requirements engineer should consider when ranking requirements are ranking too many requirements too high (or at least the at the same level) and not updating the requirements rankings once they have been set. If too many requirements are ranked at the same level, then the purpose of ranking becomes lost. If the requirements engineer must make a choice between requirements for the current iteration of the system and too many requirements are ranked as high, then the rankings are irrelevant and there was no point to that process in the first place. Keeping rankings static throughout the development life cycle is bad because as time goes on and the system is developed, requirements may change. As these requirements change, it is necessary to update the rankings so they are as accurate as possible. Rules and regulations may also change which might make a requirement obsolete, hence why it is important to do rankings as often as necessary.

6.2 Describe two different ways to identify ambiguity in an SRS.

One way to identify ambiguity in an SRS is to look for weak phrases. Weak phrases are wording that can have multiple different interpretations, which can lead to incorrect implementation of requirements. Some of these phrases from the chapter include “as a minimum” and “as appropriate”. Another method to identify ambiguity is to look for imprecise terms. These are terms that describe “measurable quantities,” but cannot be verifiably measured. For example, “many times” would be imprecise and an example of ambiguity. To reduce the ambiguity, you would specify the exact amount such as “five times”

6.3 Which of the IEEE Standard qualities for individual requirements seem most important? Can you rank these?

I think that the most important IEEE Standard qualities are singularity, ambiguity, completeness, and consistency. I consider these to be the most important because following these standards will still provide the customer with a minimum viable product. If I were to rank these standards from most important to least important, it would be this: ambiguity, completeness, consistency, and singularity. Ambiguity is the most important because the requirement must be understood by all stakeholders to mean the same thing. This will ensure that the requirement is implemented correctly. I put completeness second because if a requirement is unambiguous, but incomplete, it can be implemented correctly with the ability to account for completeness later. For example, say the requirement reads as “The system shall store [TBD] GB of user data,” the requirement can still be implemented and updated as soon as the information to complete it is received. Consistency is third because a requirement that fulfills the first two standards, but is written differently the others can still be implemented. Finally, I ranked singularity last because although it does increase the complexity of the requirement, with a little bit more effort it will still be interpreted the same. It is on the list because meeting this standard will vastly improve readability of the SRS.

6.6 Should implementation risk be discussed with customers?

Yes, I absolutely believe that implementation risk should be discussed with customers. It is part of keeping a transparent dialog, managing expectations, and building trust. A transparent dialog will provide the customer with all of the information that you have, thus allowing them to have a more clear understanding of the direction of the project. Managing expectations with implementation risk will keep the customer happy. By discussing implementation risk with the customer, the requirements engineer is building trust. The customer will know that you thought about all possible directions. This conveys to the customer that you are just as concerned with the success of the system as they are. All of this means the customer can make informed decisions and understand the intricacies of building the system.

6.7 What are the advantages and risks of having requirements engineering conducted (or assisted) by an outside firm or consultants?

Some of the advantages of having the requirements engineering process conducted by an outside entity is that they are able to look at the system as an impartial third party and that they may bring expertise to the process that an internal team may not have. However, the customer runs the risk of building dependency on the external team as well as ballooning costs.

It can be beneficial to the customer to have an unbiased opinion on the system or the domain when writing requirements. This is because they may be able to provide a perspective that does not face internal culture or political pressures. An external team may also make suggestions that are unaffected by “domain blindness”. By this I mean, when an individual or group works in an environment for so long, it may be difficult to think in terms outside of that domain. As an example, albeit on a much smaller scale, I was working on a system of virtual machines that used a package called dnsmasq to assign IP addresses to other machines. I was running into problems when standing up a new cluster of machines that I could not figure out. I happened to mention the problem to a friend and they suggested trying another package called dhcpd that had very similar functionality - and it worked. I was working in the dnsmasq domain for so long I “forgot” to consider alternatives. This is the same type of thinking that an external team could bring.

An outside firm or consultant can also bring the expertise of requirements engineering to the customer. This can mean more accurate, more comprehensive, and higher quality requirements than an internal team can write. They also would be able to create the SRS much faster, thus saving time and money (the latter depending on the value of the contract!).

Converservly, depending on the size and complexity of the system or customer itself, hiring an outside firm can be incredibly expensive. If the system constantly needs to be reworked often, it can blow through the project’s budget for building and maintaining requirements quite quickly.

Finally, if the firm is “too good” at what they do, it can be too difficult or complex to transition the project back to an internal team for future development and maintenance. If the process doesn’t include open and transparent dialog, then it can result in a lack of knowledge on how to properly utilize the requirements document.

**Chapter 7**

7.1 Are customers more likely to feel confident if formal methods are explained to them and then used?

Yes, customers are generally more likely to feel confident in, and understand the importance, of formal methods if they are explained to them before they are used. People often fear what they don’t understand. However, if the customer can understand the method, then they can understand the value. This is especially true if the system contains any sort of high fidelity requirements. If the formal method results in some form of mathematical expression and the customer can understand it, then it will be more precise and less verbose than its natural language counterpart.

7.6 Determine if the following requirements for a wet-well control system, such as the one given   
 in Appendix B, are consistent:

7.6.1 If the pump is on, then the valve shall be open.

7.6.2 If the valve is closed, then the water level is ≤ 10 m.

7.6.3 If the water level is >10 m, then the pump is on.

You can use a truth table to verify this using P (pump on), V (valve open), and W (water level > 10m)

| **P** | **V** | **W** | **!P** | **!V** | **!W** | **P -> V**  **7.6.1** | **!V -> !W**  **7.6.2** | **W -> P**  **7.6.3** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T | T | T | F | F | F | T | T | T |
| T | T | F | F | F | T | T | T | T |
| T | F | T | F | T | F | F | F | T |
| T | F | F | F | T | T | F | T | T |
| F | T | T | T | F | F | T | T | F |
| F | T | F | T | F | T | T | T | T |
| F | F | T | T | T | F | T | F | F |
| F | F | F | T | T | T | T | T | T |

Using this truth table, we can see that there are multiple ways to meet each requirement, therefore these requirements are consistent

7.7 Consider the following set of requirements for an insulin pump system:

7.7.1 If the insulin dose button is pressed, then the insulin dose is administered.

7.7.2 If the insulin dose is administered, then the dose indicator light is off.

7.7.3 If the insulin dose button is pressed and the dose indicator light is on, then the  
 insulin dose is administered.

7.7.4 If the insulin dose is administered, then the dose indicator light is on.

Determine using a truth table if these requirements are consistent or not.

Using B (button pressed), A (administered), L (light on)

| **B** | **A** | **L** | **!B** | **!A** | **!L** | **B&&L** | **B->A** | **A->!L** | **B&&L->A** | **A->L** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| T | T | T | F | F | F | T | T | F | T | T |
| T | T | F | F | F | T | F | T | T | T | F |
| T | F | T | F | T | F | T | F | T | F | T |
| T | F | F | F | T | T | F | F | T | T | T |
| F | T | T | T | F | F | F | T | F | T | T |
| F | T | F | T | F | T | F | T | T | T | F |
| F | F | T | T | T | F | F | T | T | T | T |
| F | F | F | T | T | T | F | T | T | T | T |

Based on this truth table, there are two rows for where it is True for the requirement propositions, but they are only true when the necessary conditions are false so it is not consistent. Also, requirements 7.7.2 and 7.7.4 are in direct conflict. Admittedly, this is confusing because the book says “...determine if any one of these rows has all “T”s in the columns corresponding to propositions… meaning each of the requirements is satisfied. If we can find such a row, then the requirements are consistent.” We can find such a row in the table above, but logically it does not make sense.

7.9 Why is it easier to show that a set of requirements is inconsistent rather than prove that they are consistent?

Showing that a set of requirements is inconsistent is easier than proving that they are consistent because all you have to do is find one time where they are not versus having to prove that every instance is true. In the example above, the requirements have a clear inconsistency, the light cannot be both on and off at the same time, therefore the two requirements are inconsistent. This is the same theory that applies to proof in math. To prove an assertion is false, you only need one example that is. For example, I make the assertion that the square of any integer is even. To prove this wrong, I can find an example that disproves this assertion. If our integer is 3, the square is 9, therefore the statement is false. However, if I wanted to prove this statement true, I would have to find an equation that is true for all possible inputs. This is why it is easier to prove inconsistency over consistency.