Assumptions - Assume you have to assume!

There is a general view (assumption?) that assumptions are bad and lead to risk and are to be avoided. There is the saying "assumptions make an ass out of 'u' and me". It is true that assumptions (like short cuts) can lead to big trouble, long delays and expensive rework. But it is not making assumptions that is the problem – it is either not realising the assumption has been made and not doing anything to correct (or validate) your assumptions.

When developing a solution of any kind, making assumptions is an essential step towards finding a solution. An assumption lets you progress forward and explore the problem or the solution further this enables some progress and may make you realise that the assumption was simply irrelevant as you need to think in a different direction you've gone in a wrong direction. *[This paragraph on the necessity and positivity of assumptions needs expanding / elaborating]*

So, what is the problem with assumptions?

There are four problems that can occur when engineering a solution by making assumptions:

1) Not recognising that what you have stated as a truth is in fact an assumption. Therefore, you proceed with confidence that the assumption is correct / true until facts come and show you that it is wrong and you have prepared no mitigation. Usually this is late in the program, when making change is difficult.
2) Having made an assumption and recognised it's an assumption you go forward and forget that it's an assumption, so it goes from a being a "known -unknown "to an "unknown – unknown" because you have forgotten (or rather it becomes a "forgotten-known". Usually when you remember it is an assumption you have got too far forward without reviewing and exploring the assumption, and adjusting to the "truth" is expensive and difficult.
3) Even if you note and remember the assumption is an assumption, you have to do something about it. It might be the "something" is to take a risk and "assume" your assumption is correct – but that needs to be a clear decision, not the more problematic implicit taking of risk. In an engineering plan or activity an assumption (whilst often necessary, often referred to as "to be confirmed" or "tbc") is only acceptable if there is a plan to address the assumption:
    a. Determine when the assumption being "correct" is important – what "important" decision rely on validating the assumption. Note that this might be a progressive issue – you may need successively "better" assumptions at a series of points. This gives a focus for point b), which is the planning
    b. Having determined when you need to know you need to plan an activity to address the assumption – what is to be done to remove the assumption and replace with "fact". If you don't then you will reach the point when the

assumption matters and you need to take a decision based on the assumption. At least in this case knowing there is an assumption you are taking a conscious risk, and can handle it as part of your risk mitigation.

If you don't do anything about an assumption then when you realise the "truth" rather than the assumption it may be expensive or time consuming to make the change, or it may not even be possible to make the change – because you are likely to only recognise the incorrectness of an assumption in the later parts of the solution development lifecycle, or in actual production or service – when rather than solution change the technical capability of the solution may have to be compromised.

4) Confusing change with assumptions. Something can be correct and then circumstances change and what was true has to change. There is a separate issue around "uncertainty" and the probability of change – but many of the mitigation around that are similar to those needed to handle assumptions. [There will be later discussions on uncertainty and change – including looking at the probability and effect of change in something]. The issue with change is that you don't have a point where you "need to know" – the change can happen whenever – but you do need to be ready to adapt to change and be resilient to change, just like as an assumption is adjusted to "closer to the truth"

We need to address why engineering teams struggle with assumptions – I think it is partly due to the mindset (positive go-getting) of engineering and program leads, and partly to do with the level of effort dealing with every assumption [This will be addressed further in personality / mindset discussions in "People Make Systems"], The reasons for difficulties include (in my opinion):

1) Not being comfortable with uncertainty. I contend a fundamental element of taking a systems approach is a comfort with uncertainty (see summary below). However there is a lot in the personality of people, especially engineering and program leaders in a pressurised product development or delivery environment, that is very uncomfortable and intolerant of uncertainty. If there is any doubt about this consider how acceptable the answer "I don't know" is in most technical reviews! As I will repeat often, the first step to certainty is realising you are uncertain. As an aside – realise your stakeholders don't necessarily know everything and their statements of needs (or requirements) to you will contain assumptions – both in terms of what they want and what they think you can do.
2) Assuming you have understood. People are different and have different perspectives on things – so when they explain what is wanted you may assume you have understood, when in fact you may not really have grasped everything they said as you filtered information through your (natural) biases. So don't assume you understand – that is why validation is a continuous activity in

solution development – you need early requirements validation and even more preliminary concept validation – show your solution and that will reveal misunderstandings and unknown assumptions.

3) You need to document your current understanding in your information management system. There are four barriers to this
    a. Generally information management is not always valued (seen as an overhead – there is the agile software idea of preferring working code over documented solution). But it is an investment and the costs of not managing your information are huge. [big separate section on information / configuration management, and the changing attitude to change] But we need to recognise that the value of capturing and documenting the preliminary data and rationale for solutions is not always self-evident.
    b. If there is good information management in place there can be a strong reluctance to put something in knowing it will change. Again, this is a wrong-headed application of "lean" – you need to know what you currently know (and don't know).
    c. Linked to the discomfort with uncertainty, documented (I know there are issues with document-based engineering, but some form of reporting is needed). I had an occasion where an initial requirements document for a sub-system was rejected by the technical manager "until all the tbc's" were removed". He said that an "incomplete" technical document was unacceptable. By proposing this he was in danger of turning recognised assumptions being forgotten as they wouldn't be captured. But the discomfort and reluctance to document "ignorance" is real and has to be understood.
    d. Once an assumption is resolve and either verified or correction some see capturing and storing this (replacing the old assumption with more up to date information) as waste ("We know it know"). A case was when the mechanical loads team re-ran their analysis of the loads once the more detailed sub-system geometry was available – so the assumed loads from preliminary analysis were improved / corrected). All the sub-systems confirmed that their solution was still sound with the new loads. When asked, none of the sub-system teams had updated their requirements with the new loads – they saw no point as they had verified their solution against them and the "designs were OK". This left a set of now known to be incorrect loads in the requirements documents – and assumption that when a new physical change came the engineers would remember the change in the loads (remember – the only certainty is change!)

Having accepted that making assumption is both inevitable and beneficial, what has to be done to handle the assumptions and not having them turn into later problems?

What can we pragmatically do to allow progress through assumptions, but not get caught out later? Here are some guides

1) Don't "assume nothing" – rather doubt everything, or assume everything is an assumption
2) Note the assumption – as tbc – and note rationale / confidence in assumption (it might be a reasonable estimate, or a blind guess – be honest!)
3) Explore the potential range of variation if possible (just as you need to if change expected) and try to make the solution robust to the change. Or at least know the "inflection" point, where the change in the assumed value becomes significant (drives a change in concept, severely hinders achieving a key attribute or function). This then allows informed conversation about the impact of the "correct value". (Note – this is equally relevant when considering the probability of change, rather than correction of assumption – knowing what change doesn't really matter is important in change management considerations
4) Put in place plan to review assumption – when does it need to be validated (or made "better", and how is that to be done. Otherwise, like the "Risk Admiration Society" you will have wasted your time recognising the assumption as you have done nothing about it!
5) Keep your information database up to date, and explore the knock-on impacts as assumptions are made. You could say don't "assume you know what is affected by the assumption you made".
6) As will all change, once the assumption is update / made better you may have to change things as a result. If you assume one set of operating conditions don't refuse to make changes to the program of work when the true conditions become known. This is why it is important to have removed / confirmed critical assumptions before critical decisions (especially ones that "baseline" the solution" and take you into the stage where change is getting more difficult and expensive that earlier in the development). However fast things are adjusting, you need a current version of what is known – it may expose new uncertainties, unknowns and make some assumptions more explicit or more important.

The point about "unknown" rather than "assumed" applies to the above. As well as tbc's in knowledge there can also be "tbd's" – but the first step with a To be determined" unknown is to make a reasonable assumption.

To my mind, consideration of the necessity of making assumptions confirms, if any confirmation where needed, that Systems Engineering is not a waterfall, top down approach. To progress you need to make assumptions, and to prevent those assumptions become risks you need to iterate around the development as the assumptions become clearer

Conclusions –

A) Assumption are not inherently bad – the dangers are
1) Not recognising the assumption, or forgetting the assumption has been made
2) Not doing anything to validate the assumption
3) Not adjusting plan / solution / work once an assumption has been improved

Without assumptions we'd be stuck in waiting for certainty – to be pragmatic reasonable assumptions must be knowingly made. You need to ensure at each phase in program that the understanding and the assumptions are "suitable to proceed". That does make the planning challenge being clear what "suitable" is at any stage, and realising that it changes which each subsequent stage / decision in the solution development

B) Alignment with my "5 hows" of Systems Engineering – the fundamentals that allow the potential value of Systems Engineering to be achieved
   1) *Be comfortable with uncertainty* – so don't see assumptions as evil, but necessary. But recognise them
   2) *Use Systems Thinking as a framework for uncertainty* – assess the impact of the assumption – what are consequences of truth being different?
   3) *Work on common integrated data* – this includes recording assumptions, and sharing the fact you have made assumptions with the whole team (so back to being comfortable with uncertainty).
   4) *Integrate on progressive iterations* – iteration and change are inevitable – so make sure you update and trace through changes / revisions as assumptions are matured.
   5) *Make (suitable) decisions in a timely way* – know when you have to make decisions and work to get suitable confidence in assumptions – or recognise the risk being taken making decision with assumptions in place – and record them o

Richard Beasley, 19th June 2025

(Draft input to part of overview technical introduction (Section 2) of "Pragmatic Practice of (Systems) Engineering".