Spiralling Towards an Elusive Level of Ambition

Patrick Van Hoeserlande

Some articles need time. This is such an article. I would like to tell you that it needed time to mature, but that would be twisting the truth’s arm. It probably developed during the years, but that was certainly not intentional. One of the reasons is that the topic is rather difficult to encompass, certainly if one wants to write about it. By the time I felt confident enough to put the thoughts, and I wrote purposefully not ‘my’, on paper, it felt like evolution had made those obsolete. However, recent developments – the trend towards using several smaller unmanned systems - make me believe that the underpinning ideas may still have value. I let you, as my reader, be the judge of that.

The trade-off between quality and quantity

Acquiring a capability is not only a question about quality but also about quantity. It is more about striking the right balance between these two opposing elements. Quality is subjective, whereas quantity is not. A capability might be assessed as high quality by somebody, whereas another person might think that it is of lower quality. However, it is harder to dispute the number of weapon platforms, although some will want to discuss the number of useable systems (but that is just adding a qualitative element to it). When we count five systems, then there are five. One cannot honestly claim that there are four or six.

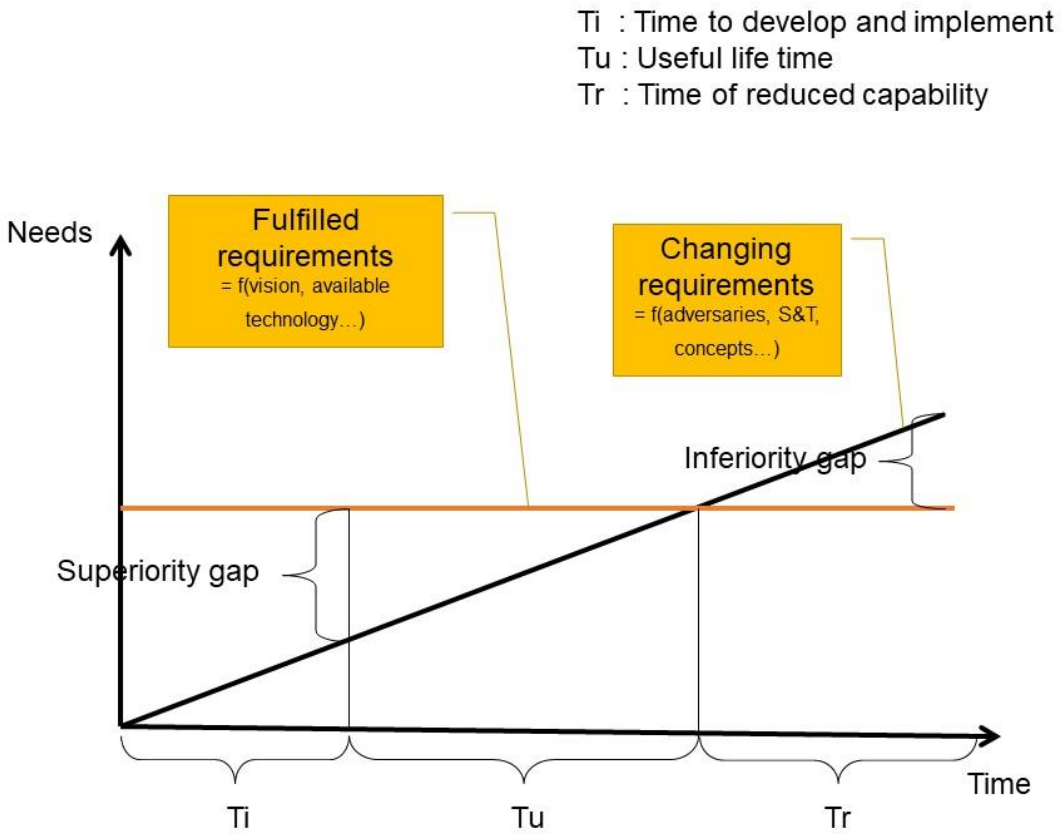
The water gets a bit murky because “quantity has a quality all its own”[1]. This quote should not be interpreted in its original meaning from the paradox of the heap of Eubulides whereby a quantitative change in the number of grains of sand leads to a qualitative change in being a heap or not. It should be considered in the sense that a high quantity of technical “simple” systems (example: drones) become a quantitative complex system by applying swarm logarithms (does Eubulides get a say in this?). Highly complex, expensive systems overloaded by waves of simple, cheap objects.

In this article, I will use the quality-quantity interaction in the sense that there is an ‘ideal’ trade-off between the two. On the one side, there is no security in having only one invincible, high-performance platform. On the other side, being the proud possessor of a huge heap of scrap is not assuring either. Somewhere out there, there is

[1] I will not go into the discussion of who came up with this phrase, nor where it came from.

an optimum equilibrium between the two, although I suspect that there will be multiple, optimal combinations. There is not only one.

To make things even more complex, acquiring capabilities is a dynamic given. It takes a long time to go from the initial concept to full implementation. In that period the environment and possibilities might change that much that (part of) the new capability is already outdated. As Donald Rumsfeld has said: "You go to war with the army you have, not the army you might want or wish to have at a later time." You aim at having on the moment of truth the better capabilities in enough quantity. At that moment and not earlier of in some near future.



Ti : Time to develop and implement
Tu : Useful life time
Tr : Time of reduced capability
Needs
Fulfilled requirements
= f(vision, available technology…)
Changing requirements
= f(adversaries, S&T, concepts…)
Inferiority gap
Superiority gap
Ti
Tu
Tr
Time

Figure 1: A simplification of the capability life in a changing environment.

The goal of capability development and transformation is to deliver affordable, economical capabilities that meet evolving requirements within the appropriate timeframe. In a stable environment, it is possible to spend a lot of time to develop and implement the capabilities (Ti) that will satisfy the expectations for a long time (Tu + Tr). It is also possible to take full advantage of the economies of scale by purchasing large quantities of similar equipment.

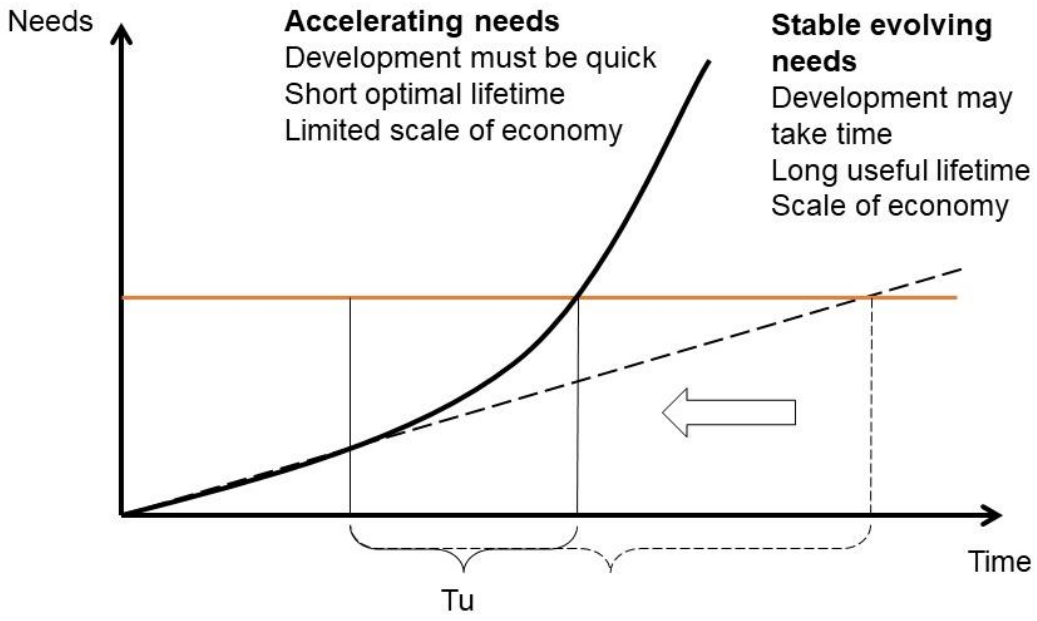


Figure 2: A simplification of the capability life in an accelerating environment.

However, in our current times rapidly evolving environment with an ever-increasing speed of change, this approach very quickly results in out-dated capacities too late available. Another approach is required.

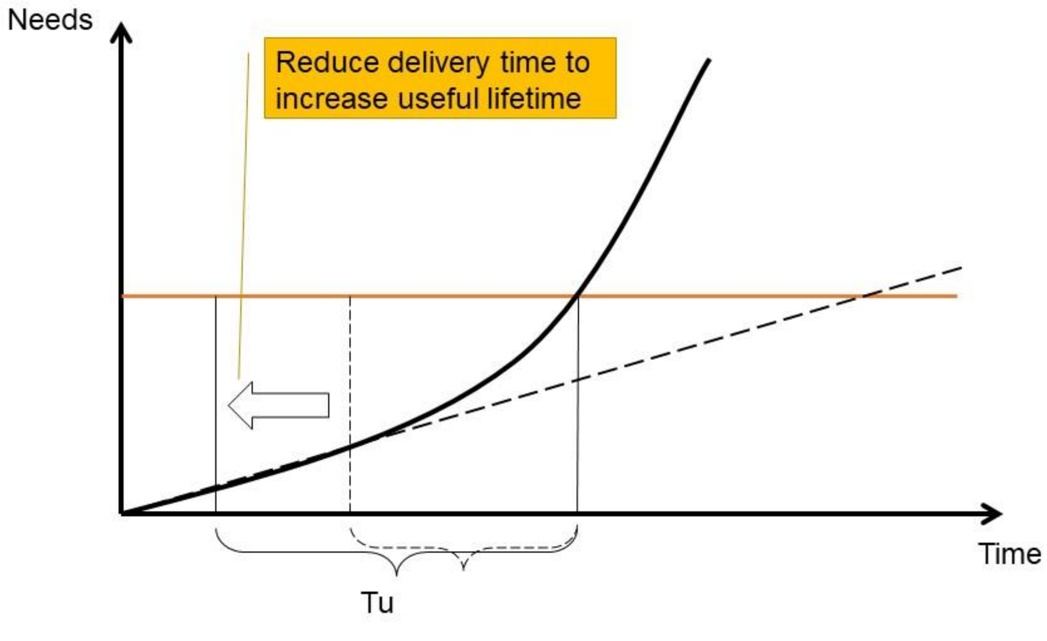


Figure 3: A reduction of the delivery time (Ti) leads to a longer useful lifetime (Tu).

Reducing the development and implementation time (Ti) will directly lead to a longer useful lifetime wherein the capability is superior. Therefore, transformation and agile innovation are important tools. One way to do achieve this is to reduce the requirements for the capability. This might reduce the marginal gain, but it is an important step towards a broader answer to our question on how to approach the problem of delivering the right balance, the trade-off between quantity and quality. How to deliver capabilities to meet the level of ambition?

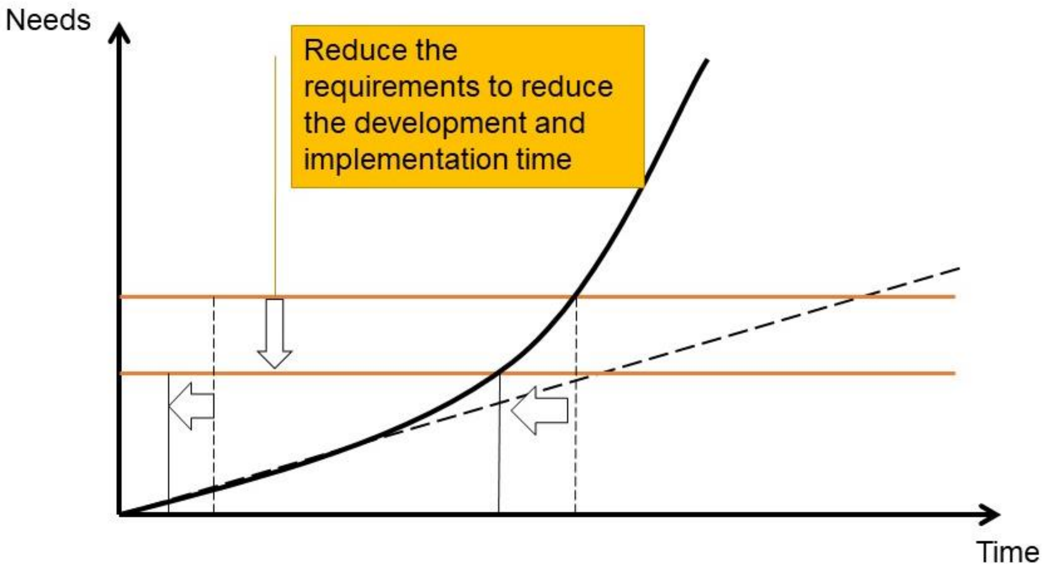

Needs
Reduce the requirements to reduce the development and implementation time
Time

Figure 4: A reduction of the requirements leads to a longer useful lifetime (Tu).

But before we can do that, we need to discuss spiral development.

Spiral Development

Spiral development is "a process for developing and transformation of capabilities within which the end-state requirements are not known at program initiation but are refined through continuous user feedback, experimentation, and risk management so that each increment provides the user the best possible capability".

A characteristic of spiral development is that the users are constantly involved from the start of the process and are thus co-developers. Resources are made available to conduct early operational experiments. This makes it possible to test very quickly and at low cost whether the concept has a chance of satisfying effectively. This results in "greater space to learn" and "less impact of mistakes". After each iteration, the concept and requirements are revised in the light of lessons learned and changing circumstances. The goal of "spiral development" is to obtain a product that evolves with the needs of the end-user.

Although spiral development 'pur sang' aims at delivering a final capability, we can easily adapt it to deliver intentionally an ever-evolving stream of capability packages. The capability to match the LoA needs to be divided into layers. The number of layers is a function of several variables like the size of the main platform, the speed of technological evolution, the maturity of the capability, the number of systems… If we elaborate on Figure 4 by adding our adapted spiral development approach, we understand why reducing the requirements may prove opportune, even if that means a reduced marginal gain of Tu.

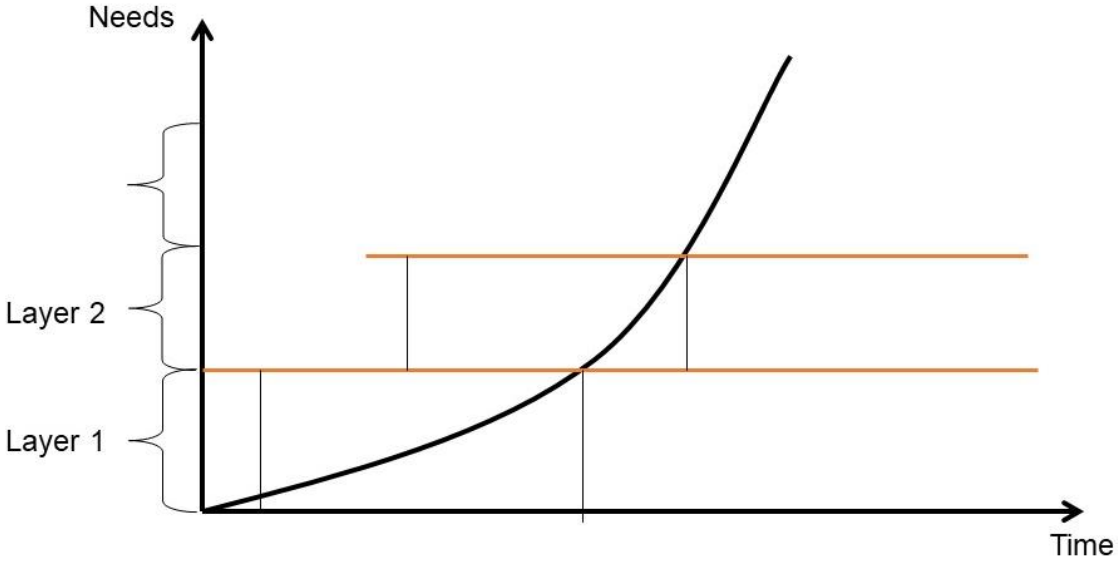


Figure 5: The application of spiral development on the capability life cycle.

To explain the approach a bit more in detail, let us use three layers. We define these layers related to the LoA by the minimum quantity of capabilities need to satisfy combinations of scenarios with a similar likelihood:

- 1st layer: routine (most likely);
- 2nd layer: less likely;
- 3rd layer: unlikely.

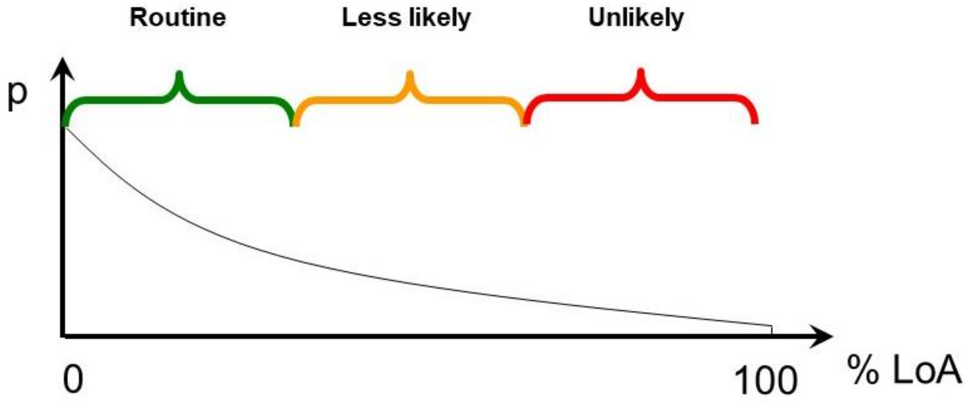


Figure 6: Layers as a function of the likelihood of executing parts of the LoA.

While we spiral towards the evolving requirements, we add the approach of layered acquisition. In doing so, we provide the latest spiral outcome, i.e., the most modern capability at high readiness to the most likely combination of scenarios. We foresee the 'older' core capability for the less likely layer.

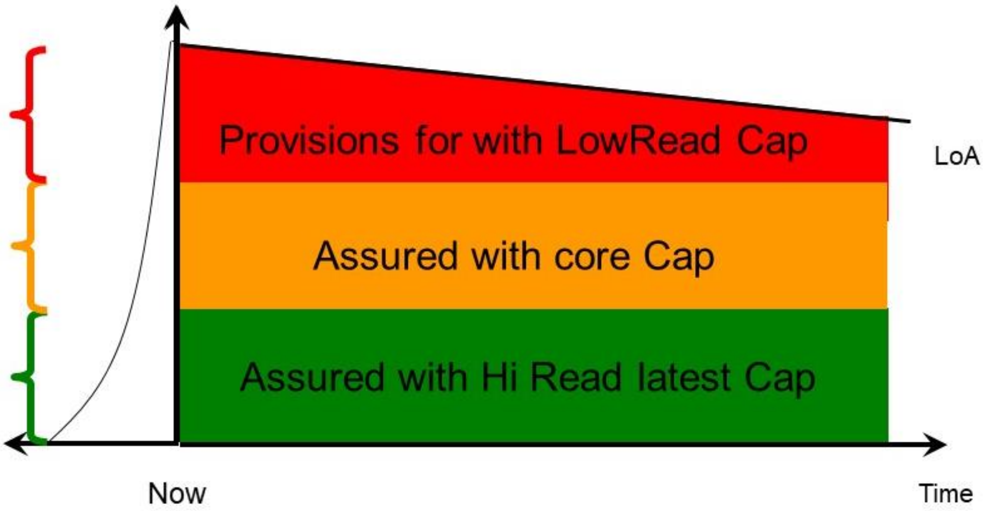

Provisions for with LowRead Cap
Assured with core Cap
Assured with Hi Read latest Cap
LoA
Now
Time

Figure 7: Layered acquisition linked to the likelihood of executing parts of the LoA.

Combining Figure 4 and Figure 7, results in a chessboard-like approach to capability life cycle management. In our case, the initial first layer capabilities move after the introduction of a next spiral capability to the second layer, the second spiral to the third layer, the third is phased out.

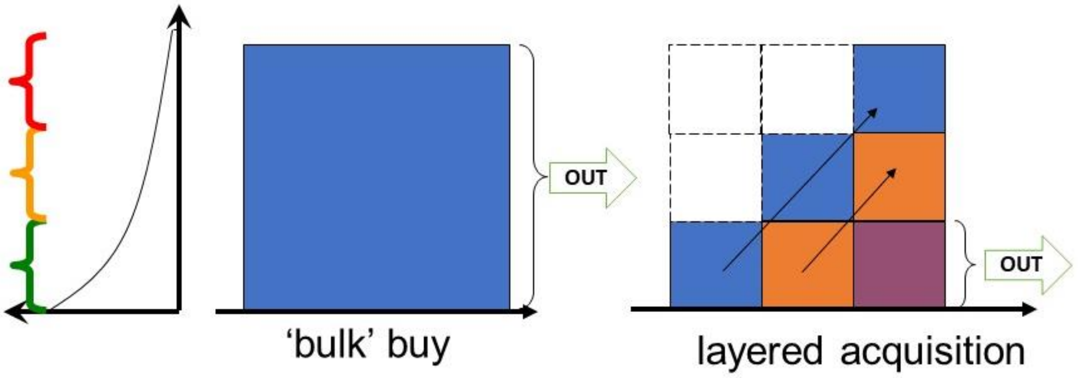


Figure 8: The combination of layered acquisition and spiral development compared to a bulk approach.

This can be done in several ways depending on your orientation towards material or capability. Or maybe you rotate the layers so that you send the latest capabilities to current operations? Just make sure that you send capabilities with every DOTMLPFI equally implemented and not just new equipment.

An important nuance is that layer 2 capabilities do not mean that these are less effective. Having the latest equipment, newest training methods… sometimes do not compete against a well-oiled interaction of 'older' line of development pushing the overall envelope beyond the initial boundaries.

The final step to complete our approach is to add a changing LoA – here illustrated by a declining LoA – and ever shorter development and implementation cycles. This is illustrated in Figure 9.

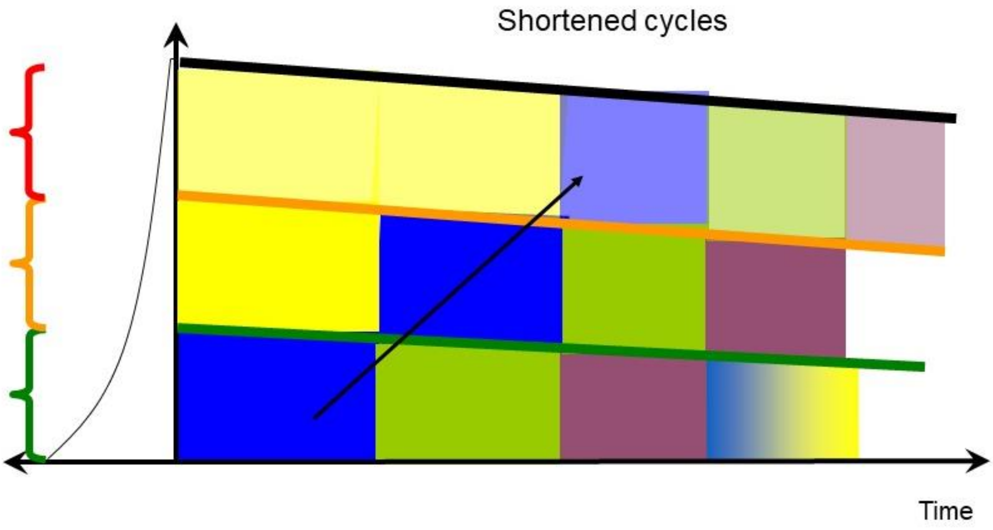


Figure 9: The complete approach to balance quantity with quality.