AGILE SYSTEMS ENGINEERING

Technical Article

Introduction

It is hard to say whether increasing complexity is the cause or the effect of man's effort to cope with his expanding environment. In either case a central feature of the trend has been the development of large and very complex systems which tie together modern society. […]

"The growth of these systems has increased the need not only for overall planning, but also for long-range development of the systems. This need has induced increased interest in the methods by which efficient planning and design can be accomplished in complex situations where no one scientific discipline can account for all the factors." – Arthur D. Hall, A Methodology for Systems Engineering (1962).

Many industries have now reached the threshold of complexity past which traditional methods of project management are no longer up to the task, but aerospace and defence was the first, and systems engineering was the result. Since it first started to emerge in the 1940s, systems engineering has approached complexity by emphasising the structural elements of a system as a whole as the primary generator of its behaviour. It gives engineers tools to analyse and describe the emergent, holistic properties of a complex system over and above the mechanical details of individual components.

Throughout that period, systems engineering has continued to evolve, refining and developing the processes it recommends for the governance of engineering and project management. The Tactical Data Links (TDL) industry has been at the vanguard of many of these innovations, pioneering system-of-systems methodologies and model-based verification. In even more recent times, processes like interoperable Systems Management and Requirements Transformation (iSMART) and SyntheSys' own System Process for Interoperability Requirements and Implementation Testing (SPIRIT) approaches have further advanced the relevance of systems engineering techniques to the development of ever more complex systems.

It is precisely these sorts of innovations which now enable systems engineering to take a more open approach to the future, learning lessons from other industries which have reached the same complexity threshold that motivated the development of systems engineering, but reacted to that complexity in very different ways. Agile software development emerged in the early 2000s in precisely this way. The 1990s were, of course, a time when the proliferation and complexity of software applications grew at an unprecedented rate. Using traditional project management approaches, developing these systems could take years, and in this fast-changing world, by the time the software had been delivered, business needs had often already moved on. Because software development had crossed this threshold of technical complexity, project management needed to better respond to the more complex and ever-changing business needs that it was supposed to be serving.

Pure Agile and pure Systems Engineering (SE) were developed in very different contexts, and although decisionmakers have tried to apply Agile in SE's traditional domain in the past, including in the TDL industry, pure Agile has proved challenging to adapt. Agile was designed for pure software products, where updates are frequent and inexpensive, integration costs are low, there is no or little reliance on specialised hardware, and fabricators are the same people as the designers. In other words, there is little risk associated with getting it wrong the first time, and because the product value is primarily derived from the cumulative benefits of discrete features rather than the emergent properties of the whole, partially working implementations of an idea will often take customers a lot further than they would in TDLs or any other traditional SE domain.

But nonetheless, Agile methods have a proven track record of delivering considerable benefits to projects, even outside of software engineering. These benefits go beyond speed and cost, though of course these are key motivations for applying the methodology, but reach as far as providing better scope control and adapting more readily to requirements change. In their respective traditional industries, Agile methodologies have actually outperformed SE in research with respect to Return On Investment (ROI): 7:1 in the case of systems engineering activity[1], and 11:1 in the case of Agile project management[2].

But, of course, those benefits cannot be expected to straightforwardly translate out of their original intended context. Nonetheless, cutting edge projects have shown it is possible to access the best of both worlds, with a hybrid methodology that learns the most important lessons of both approaches. If the TDL industry and other traditional SE domains could implement a hybrid Agile Systems Engineering (ASE), it could be the next major step forward in our ability to drive value.

Agile in a System-of-Systems Context

The TDL industry has led the way in recognising that Systems-of-Systems (SoS) require a different analysis toolkit and different development methodologies to more integrated and standalone systems. The SoS should be an open system enabled by open interfaces in the sense that constituent systems should conform to defined interface standards for their inputs and outputs, allowing the SoS to treat them as much as possible as a black box. Yet, because of changes in the political environment and strategic context, many individual SoS exist for far shorter timescales, and all will certainly be expected to change configuration frequently and rapidly, using existing assets that are autonomous both in terms of function and life cycle.

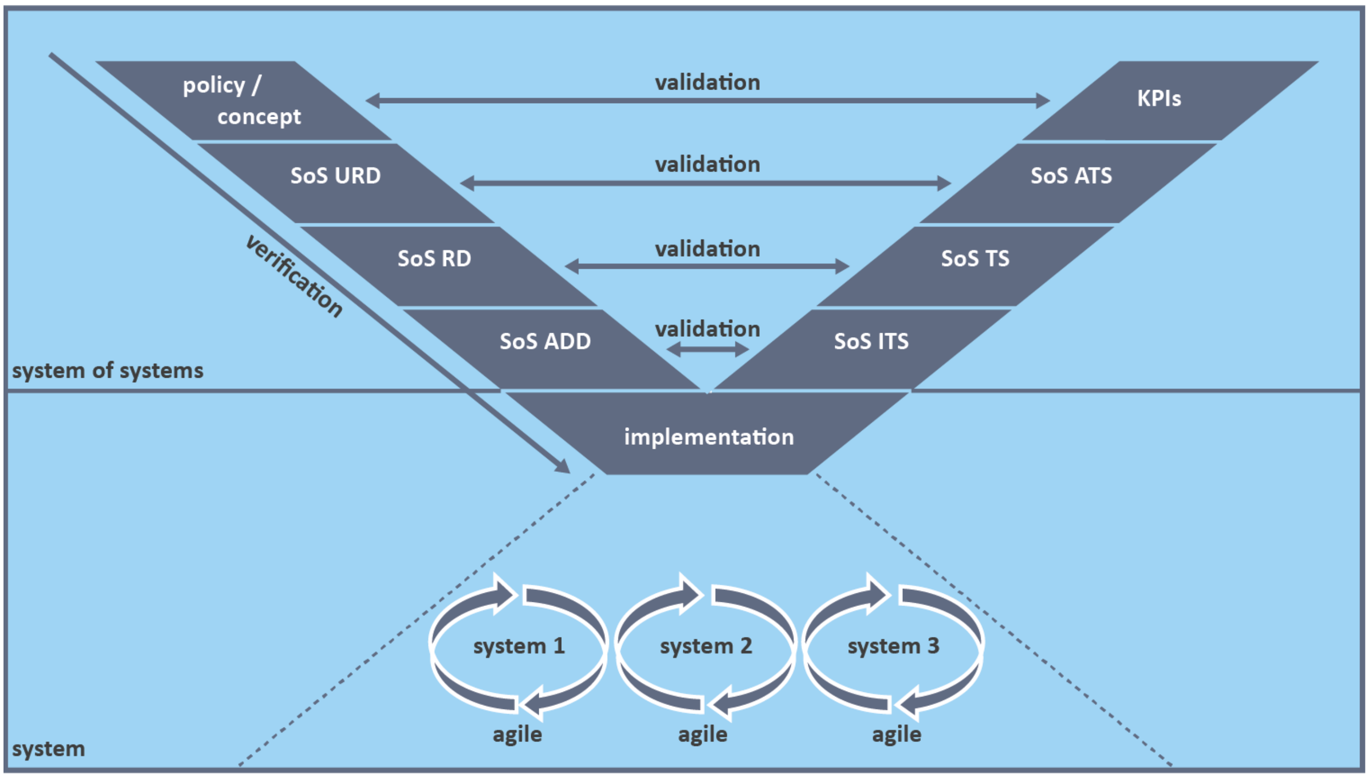
In the TDL domain at least, a consensus has emerged that the crucial factor in successful SoS engineering is a strong focus on interface standards, or to use the TDL-specific term, interoperability assurance. The SoS should be an open system enabled by open interfaces in the sense that constituent systems should conform to defined interface standards for their inputs and outputs, allowing the SoS to treat them as much as possible as a black box. As long as all components of the TDL SoS have a common approach to information exchange, the resulting networks can be assembled as needed quickly and securely, and the long-term evolution of the SoS standards can proceed without being constrained by the mechanical details of individual constituent systems with autonomous functions and very different life cycles. Systems engineering best practice in this context strongly separates the engineering activity associated with defining the interface standards for the SoS and the design and mechanical details of constituent systems, which should be treated as a black box. In the context of a traditional systems engineering V-model, the top half of the V and the bottom half of the V are in an SoS context essentially independently managed and governed, passing only standards and validation results between them.

The great virtue of systems engineering methodology is its ability to address the holistic aspects of a system independently of the sum of the parts, and to analyse the ways the structure of the system generates its behaviour beyond the mechanical details of individual components. It is precisely this whole-system view that Agile lacks, essentially because it isn't needed in its traditional domain, where product value is primarily derived from the cumulative benefits of discrete features.

[1] Eric Honour. Systems engineering return on investment, PhD diss, 2013. University of South Australia. https://www.hcode.com/seroi/documents/SE-ROI%20Thesis-distrib.pdf

[2] David Rico, Hasan Sayani and Saya Sone. The business value of agile software methods: Maximizing ROI with just in-time processes and documentation, 2009. FL: J. Ross Publishing. https://www.semanticscholar.org/paper/The-Business-Value-of-Agile-Software-Methods%3A-ROI-Rico-Sayani/e1f8a6a88a92b3c6f5cebb5ba0e50320f0e27115

And likewise, Agile methodologies exceed the capabilities of pure systems engineering in cost, speed and change-readiness terms where individual components can largely be treated in isolation, conforming to clearly specified requirements and where integration with the larger system doesn't have to be a major concern of the development process. So in other words, we have from SoS engineering the thought that interfaces and component systems should be engineered independently, and we have in SE and Agile two different engineering management approaches best suited to managing one of those two parts of the engineering project.



SoS=System-of-Systems, URD=User Requirements Document, RD=Requirements Document, ADD=Architectural Design Document, ITS=Integrated Test Specification, TS=Test Specification, ATS=Acceptance Test Specification, KPI=Key Performance Indicator

Driven by the open system architecture that enables both of these methodologies to work together, hybrid ASE techniques have unlocked some impressive capabilities, demonstrated well by a paradigm and extensively studied ASE project: the Johns Hopkins Applied Physics Laboratory's Multi-Mission Bus Demonstration project[3]. This project was a successful attempt to produce a military space satellite to the size - and weight-restricted 'CubeSat' specification, which would allow the satellite to be launched more cheaply through 'ridesharing' with other payloads.

The team had very strict time and budget constraints, and the project required extensive development of new technologies and system components. As such, it was clear that traditional project management wasn't going to be apt to the challenge.

A pure Agile methodology wasn't going to work, either, because of the extreme constraints on how the individual components of the satellite had to fit and work together. As such, the project did some initial SE-like activity at the outset to define the plug compatibility standards, the interface with the spacecraft bus itself, and the external form, fit and function of the individual subsystems[4].

[3] INCOSE. 2015. Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0

[4] Philip Huang, et al. Agile hardware and software systems engineering for critical military space applications, 2012. Proc. of SPIE Vol. 8385. https://www.spiedigitallibrary.org/conference-proceedings-of-spie

From there, Scrum-like teams were assigned to work on the individual component subsystems, and were empowered to make incremental improvements to the design within the constraints assigned by the overall system architecture. These constraints could be modified if necessary, but only through a more SE-like top-down committee involving all of the subsystem Scrum masters and the program manager. The project was a success, and the satellite, as well as by now a number of successor projects, are working in orbit.

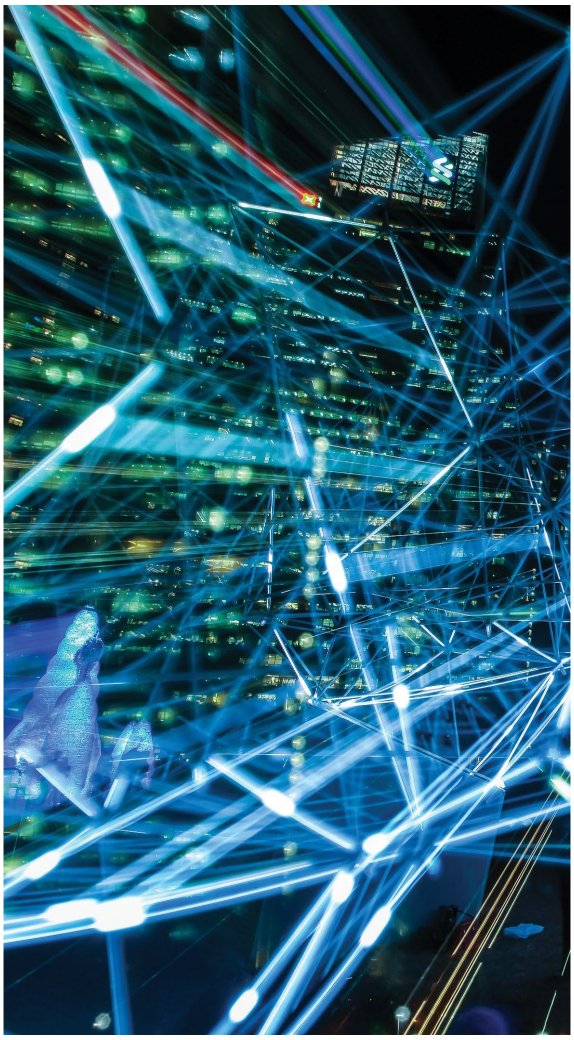
Collaborative Hybrid Agile Systems Engineering (CHASE) Process and Tools

As the Johns Hopkins project shows, Agile systems engineering methods can be successfully applied to complex, integrated systems as well as in an SoS context, though of course the methodology draws heavily on techniques developed for SoS.

Hybrid Agile systems engineering needs to be collaborative. Both domains need to learn from one another, work together well and to an extent learn from one another's techniques: SE teams managing interfaces will also likely need to understand and promulgate project plans that use Agile methods of dependency tracking, prioritisation and workflow management, and Agile teams developing components will have to learn to conform to SE interface standards and likely adapt to requirements specified in ways that more closely resemble SE good practice. Nonetheless, SE absolutely has to own the programme governance, the high-level requirements and the interface standards to which individual projects and components must conform.

Collaborative Hybrid Agile Systems Engineering (CHASE) is our approach to bringing these two engineering management philosophies together and delivering the value benefits of both.

Doing CHASE well requires excellence in both Agile and SE processes, a unique tool configuration, and skills among team members to allow these techniques to work together effectively. But done right, we believe it could be a major step forward in the TDL industry and beyond.



SyntheSys DEFENCE