Making Automotive Embedded Systems More Agile

Even before the Covid crisis, the next few years looked like a period of significant change for the automotive industry. Pressure to innovate has perhaps never been stronger in its entire history than it is right now. Decarbonisation, autonomous vehicles and Brexit are changing regulatory demands, driving up product complexity, and encouraging the industry to invest in digital manufacturing and supply chain integration.

The process for developing automotive systems, especially those with both hardware and software components, is getting exponentially more complicated, and as vehicles get smarter and smarter, that complexity is only going to become more and more difficult to manage.

When industries turn 'smart', when that threshold of technical complexity is passed, it usually requires of engineering to radically rethink its workflow and practices. When it happened to software engineering in the 1990s, Agile was the result. The automotive industry is getting to a similar position now, and the time has come for it to explore options for how it can adapt to this changing environment.

Pure Agile has allowed automotive engineering to manage complexity more effectively when it comes to the software components of their products, but it has rarely been applied to pure hardware or to embedded systems – and for good reason. Agile was designed for products with frequent update potential, low integration costs, little dependence on specialised hardware, and with minimal regulatory constraints.

As a result, automotive embedded systems are most often still developed hardware first, software second, rather than in parallel. In a faster-paced and innovation-driven environment, it may be time for automotive suppliers to look to more sophisticated and specialised engineering management methodologies, designed to handle complexity better, faster and more efficiently.

Software engineering wasn't the first industry to reach a complexity threshold which required them to throw out the engineering management playbook: that honour goes to aerospace and defence, which began overhauling its methods in World War II, and has continued to develop these engineering management techniques – which came to be known as Systems Engineering (SE) – through the space programme and into the most complex military systems of today.

SE is about drawing on the science of finding patterns in organised complexity, and the analysis of the emergent properties of a whole rather than the specific behaviour of individual components. The critical shift in understanding that SE brings to the table is that it is the structure of a system that generates its behaviour, more than the mechanical details.

In the last decade, systems engineers have explored how their techniques can be combined with insights from Agile, and the results have spoken for themselves. Agile systems engineering techniques have proven particularly successful in projects with both software and hardware components, research and development demands, and strict time and budget constraints.

In other words, they are proving extremely well suited to meeting the emerging demands of the changing automotive sector.

As the systems developed by the automotive industry grow ever smarter and more complex, many automotive manufacturers have turned to SE as a way to ensure they keep enhancing the value of their products in a more technically demanding market. SE is about drawing on the science of finding patterns in organised complexity, and the analysis of the emergent properties as a whole rather than the specific behaviour of individual components. Thinking in this way has produced a robust and scientific approach to requirements management and verification, a greater focus on the full life cycle of a product, and novel modelling techniques for complex emergent behaviour.

Almost all SE activities are the sort of thing that any engineering team will be doing as a matter of course: collecting and managing customer requirements; designing a product and modelling its behaviour; managing implementation workflows, and providing the basis for verification and validation tests. SE adds value to those processes by introducing techniques for analysis and information-gathering that close the gap between the scientific and engineering methods, introducing a single source of truth that persists throughout the product life cycle and beyond it to new variants, and introducing ways of managing the stakeholder relationship to give you assurance right from the start that you are building the right product in the right way.

SE techniques are particularly well suited to projects where the customer value is primarily derived from the emergent properties of the whole system, as compared to pure Agile methodologies, which are best suited to applications where the value comes from the cumulative value of essentially discrete features. Being Agile allows the project to develop those features with speed, efficiency and change-readiness.

By using a hybrid approach involving both Agile and SE techniques, engineering management can access those advantages of agility while maintaining focus on the behaviour of the whole system: getting the best of both worlds. In an embedded systems context, the most concrete advantage of Agile SE (ASE) is the potential to develop hardware and software in parallel rather than consecutively, leading not just to greater efficiency but also greater freedom to innovate. More sophisticated customer needs can be met faster and more precisely.

Collaborative Hybrid Agile Systems Engineering (CHASE)

ASE is about formally separating the engineering of the structure of a system from the details of how individual components are designed and work.

The primary value of using SE techniques in an engineering process is the ability to identify, through carefully designed requirements and modelling techniques, the complex interactions within a system and between the system and its environment. SE techniques will identify patterns and trends in how the system changes over time, the impact of time delays in the system's operation, the circular nature of complex cause-and-effect relationships, and where unintended consequences are likely to emerge.

The philosopher of science Karl Popper famously said that for a statement to be considered scientific, it must be falsifiable: you have to be able to tell the difference between a world in which the statement is true and a world in which its false. Similarly, systems engineers work towards requirements by which it is possible to tell the difference between a system that achieves them and one that doesn't. The requirements that result are – among other benefits – clear, verifiable, functional, minimal and consistent.

The process begins by identifying what the client wants in terms of a problem that they need to solve, or an opportunity that they want to pursue. Without yet looking to specific solutions, the first step is to develop the 'operational concept': what users want the system to do. The context and environment for the system – its basic inputs and outputs – should be understood as clearly as possible while the system as a whole is still being treated as a black box.

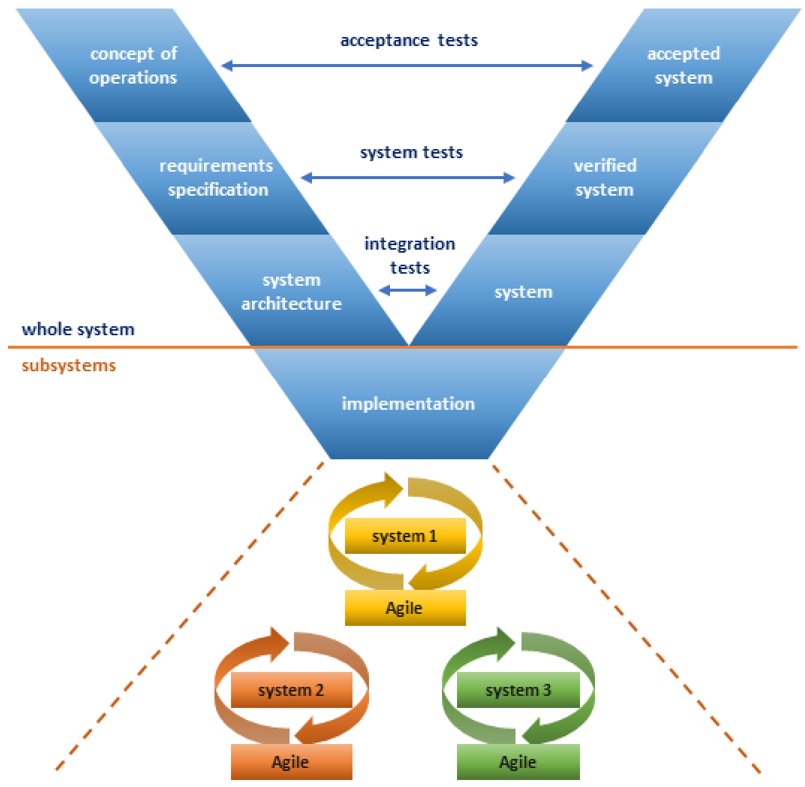
From there, systems engineers continue with a 'top down' approach to requirements and modelling, creating similar ways of understanding the product at different levels of specificity. As requirements get clarified and detailed, the model progresses down equivalent layers of complexity, at each stage fundamentally treating subsystems and individual system elements as black boxes that transform inputs into outputs.

A formal SE model is built entirely out of black boxes, taking inputs from users and the environment and outputting stakeholder needs. The model is not concerned with how individual components work, but rather with the structure of a system as a whole: the inputs, outputs and interactions of subsystems.

As such, the benefits of a systems engineering model are not just confined to presenting a clear, coherent architecture to designers, testers and operators: it also allows the behaviour of the system as a whole to be anticipated prior to engineering the specific details of individual components.

The reason why Agile techniques don't often adapt well to complexity, or the end-to-end development of embedded systems is precisely because these holistic aspects of engineering can't readily be addressed with only Agile methodology. By building an SE model prior to proceeding with Agile development – CHASE methodology in a nutshell – it becomes possible to understand how individual subsystems are expected to interact before their detailed workings are understood, and generate interface standards which can be passed to scrum teams along with their requirements.

Interface standards can be anything from the form, fit and function of specific hardware elements, to data schemas and network protocols: anything about the rest of the system which constrains or requires an individual subsystem to be designed in a particular way. Once those standards are in place, Agile teams can freely develop components to meet requirements with their accustomed speed, efficiency and flexibility.



Case study: Johns Hopkins Applied Physics Laboratory's Multi-Mission Bus Demonstration[1]

Driven by the open system architecture that enables SE and Agile methodologies to work together, demonstrated well by a paradigm and extensively studied ASE project: the Johns Hopkins Applied Physics Laboratory's Multi-Mission Bus Demonstration project. 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.

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.

CHASE Process and Tools

Hybrid Agile SE 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.

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.

SE tools are designed to enable an end-to-end system for managing the entire engineering life cycle, from requirements management through complete systems design, modelling and testing. The best in class tools can integrate information for and from all teams and stakeholders with a role in the product life cycle into a single, adaptable source of truth, improving traceability, communication and change-readiness.

As an IBM® Gold Business Partner, we provide application knowledge, experience, and flexible delivery mechanisms to provide our customers with a deep understanding of the software solution that is right for them, to ensure they can maximise effectiveness. We have the expertise and can offer flexible licensing options across the full suite of IBM® Engineering Lifecycle Management tools, including DOORS® and DOORS® Next, Design Rhapsody®, Engineering Workflow Management, Engineering Test Management and Engineering Lifecycle Optimization.

Gold
Business
Partner
IBM.

[1] Philip Huang, et al. Proc. of SPIE Vol. 8385. [Agile hardware and software systems engineering for critical military space applications (spiedigitallibrary.org)]