Smarter Product Enablement: Understanding the dynamics for successful delivery

Bola Rotibi, Principal Analyst

November 2009

Over the past decade, advances in software, hardware and networking technologies have enabled more and more technology to be woven into and around physical goods – adding value to those goods even without that technology necessarily being "embedded" in any hard-wired sense. As a result what was once described as the embedded systems market is now both broad and varied ranging from software components and applications embedded in hand held devices and consumer appliances to large complex computational software systems controlling large mechanical constructs such tanks and planes.

This is a market that is now rich in growth opportunities and is exciting in the extent of the commercial reach and in what can be achieved. Given such market expansions and the potential open to a broad range of players, understanding the dynamics for successful delivery becomes paramount and necessary if value is to be both achieved and delivered.

"Smarter Product Enablement" reflects the changing nature of the embedded systems market.

Summary

There is no one common model of software provider

The embedded systems field is vast – covering Consumer, Medical, Aviation, Automotive and Communication electronics and computational systems. The challenges, approaches and priorities in each of these segments can be very different. Conversely for aviation and automotive electronic- and software-based systems, the reliability requirements are of a high order, with development times and test cycles that are longer. Here, the industry is controlled by giants.

The software engineering responsibility in delivery of today's complex systems is in the hands of multiple players – but the arrangement of players, their interaction patterns and their priorities also differ significantly by industry segment. There is no one common model of software provider that holds sway across a homogeneous "embedded systems market".

Look for suppliers who can leverage the support of a wider ecosystem of partners and suppliers

Software usage in products is on the increase: especially to meet raised user expectations, offer more sophisticated features (either user-driven, or to extend hardware system capabilities) and regulatory stipulations. This also increases the potential and reach of product application. As a result it will be vital to engage with suppliers that support broad partner ecosystems and who actively engage in developing cross-domain and cross-industry architectural patterns and blueprints. Such suppliers demonstrate the right attitude in understanding the challenges and demand for cross-domain skills.

Application innovation that goes beyond the physical boundaries

While the potential of software-enabled products which allow and support intelligent applications that engage the user with new interaction experiences is widely recognised, the trick is to demonstrate how "webs" of online, offline and embedded systems can be integrated to deliver innovation and value. To succeed here, you need to adopt an open-minded approach to what can be achieved through a wide ecosystem of technology partners.

There is a danger to limiting the opportunity to drive process change

Despite universal recognition as to the increasing value and importance that software plays in delivering today's consumer and industrial products, there is a significant degree of inertia in fully focusing on how to improve the software engineering process.

Underlying this is a key point: that although the value of software is understood at some level, it's the cost of developing the software that appears to be the key metric used in decision-making. There's a disparity between the cost of development, and the value of what's developed – and this is a problem that needs to be addressed across industry segments.

About this report

In our world today we are surrounded by many products that now have some form of embedded software component or system contained within them to provide smart, intelligent and adaptive functionality. These range from portable handheld devices such as mobile phones and PDAs, watches, household appliances; to more complex installations such as those providing the flight operation systems for planes, missile guidance and tracking systems, the internal driving controls and entertainment systems for cars. The list is endless – and the complexity of the technology can range from low (where functionality is provided by a single microprocessor) to very high (where functionality is provided by large distributed networks of integrated systems and processors).

This report aims to outline a pragmatic framework of considerations and rules of engagement for anyone looking to realise the potential growth opportunities associated within these “Smarter Products”. The framework is based on our analysis of an in-depth qualitative study of embedded systems technology and services providers which was designed to explore the current practices and challenges of organisations operating in the Smarter Products space – specifically regarding software requirements definition, delivery and lifecycle management.

We interviewed organisations from around the world that develop, build and deploy embedded systems or smart software based products. Those that we interviewed spanned most of the key industry segments where systems are deployed or developed and designed to extend the capabilities of physical products: automotive industry, consumer electronics, defence and aerospace, system integrators, design consultancies to name but a few.

Towards Smarter Products market today

The common definition of an embedded system as described by Wikipedia is that of

"A special-purpose computer system designed to perform one or a few dedicated functions, often with real-time computing constraints and is usually embedded as part of a complete device including hardware and mechanical parts".

This definition is fine, but it's clear that over the past decade, advances in software, hardware and networking technologies have enabled more and more technology to be woven into and around physical goods – adding value to those goods – without that technology necessarily being "embedded" in any hard-wired sense. Furthermore, although the platforms and requirements of embedded systems differ in many ways from those associated with desktop PCs, laptops or general-purpose enterprise servers, even the operating systems that run at the heart of today's embedded systems have evolved away from yesterday's "locked down", fixed function environments. The lines are blurring between what was once typically defined as an embedded system and the computational and programmability of desktop, laptop and server computers.

A broader market definition and scope

This broadening of scope and deepening of software related functionality is a trend towards something we call "Smarter Products". As a consequence of the changes noted above, the supply chain through which Smarter Products are designed, integrated, tested, sold, delivered and serviced is a lot more complicated than the traditional embedded systems market. In relation to this, the "Smarter Products enablement" (SPE, for the purposes of this report) market is the market for software components, assets, methods and professional services that help all those involved in augmenting the value of physical goods (such as vehicles, entertainment and communication devices, industrial plant and machinery and defence systems) with software elements.

Fiat's Eco:Drive – an example of the scope of today's "Smarter Product enablement" market

A good example of the extent and commercial reach of this market today can be seen in Fiat's Eco:Drive solution, in which embedded software systems are opening up new avenues of engagement and interaction as well as opportunities and potential for new revenue streams. Eco:Drive is actually a combination of:

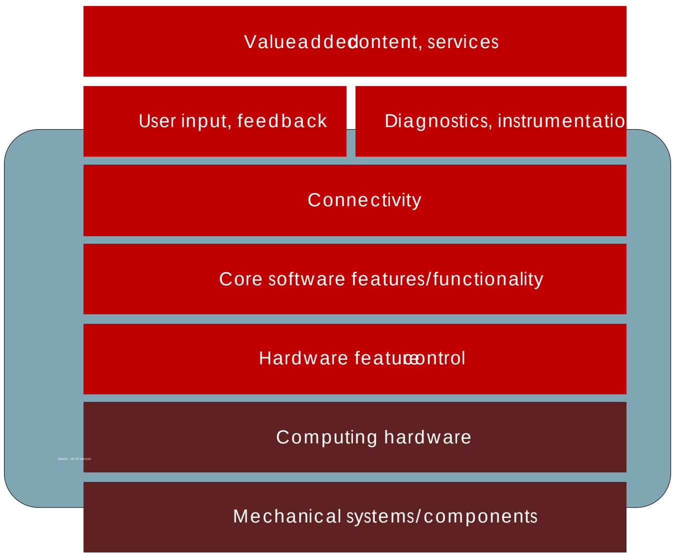
- A software application residing within an onboard computer system that provides an abstraction layer from the core software that might operate the engine, the gearbox and the fuel injection system, collecting and storing measurement data which tracks, amongst other things, the efficiency (or lack of it in many cases) and effectiveness of drivers in minimising their CO2 output when driving the car.
- An online software-based service, developed by AKQA, which presents the relevant data gathered by the embedded onboard computer system that allows drivers to benchmark and improve their ability to reduce their carbon footprint when they drive. The online application focuses on engaging the driver into driving with a more "green" conscience and offers further potential for enriching the interaction

between the customer and the car dealer/manufacturer.

Four SPE software delivery contracting models

Today, the SPE market contains a wide variety of providers with myriad interconnections and interactions of varying complexities. There are multiple layers of software engineering and delivery involved in the assembly and delivery of today's Smarter Products, and different parties experience different challenges, depending on where they sit in the overall delivery chain – see figure 1.

Figure 1: A multi-level market model: different concerns, different pressures


Valueaddedcontent, services
User input, feedback
Diagnostics, instrumentatio
Connectivity
Core software features/functionality
Hardware featurcontrol
Computing hardware
Mechanical systems/components

In today's SPE market, complete systems can require the integration of everything from mechanical components to remote, value-added software-based content and services. Different levels of concern are may be distributed to specialised providers, and different providers tend to have their own priorities and challenges.

As standardisation and computing power of available hardware has increased, the push towards reuse of common components to drive down costs the use of standard software systems and components over proprietary developed ones has grown. Both these actions have lowered the barriers to entry for many suppliers, allowing them to focus on providing differentiation through advanced software-based functionality and services. This has resulted in diffusion of responsibility for software delivery: responsibility tends to migrate to specialist players with the skills, expertise and resources to develop the required software components and systems. In our research, we've found four software delivery contracting models that are commonly pursued.

The use of external partners in helping to determine the role that software should play in a product depends on the sensitivity of the product delivery process.

External prototyping, integrated development

In a number of cases the initial design and prototyping is carried out by a specialist research design institute to explore and prove a particular technique or function, before being handed over to the client company as a reference architecture or template for further refinement and “industrialisation”.

Distributed software component supply chain

Other models see multiple third parties either providing specific software components or fully functioning software based products in their own right (e.g. radios, software controlled instrumentation etc.) that are then integrated into larger systems, devices and appliances. Whilst responsibility for the engineering process of individual components resides firmly with the third party provider organisations, overall quality control remains the responsibility of the company assembling the myriad components and products into a final delivered product (e.g. car, plane, army tank).

The ease with which many organisations which own a particular product brand seek the aid of external suppliers to provide integrated hardware/software components is perhaps surprising. However the pressure to drive down costs – particularly in the development of consumer products where profit margins can be tight but the expectation for sophisticated feature functionality can be high – there is a drive towards the use of standardised hardware and software systems.

External development by accredited specialist third-party

In the cases where there were strong safety critical regulatory rules and policies to adhere to, the software engineering process is often carried out by well known, well-established and accredited third party suppliers owing to the expense, skills, experiences and timescales in achieving accreditation.

This is particularly common in industries such as Telecoms, Defence, Aerospace, manufacturing, automotive, medical equipment, consumer electronics.

Vertically-integrated software creation

There are still organisations that maintain ownership of the entire engineering and delivery process of a product i.e. software, electrical/electronic and mechanical. They do so for reasons of high security and safety critical regulations (often government backed) that require a tightly integrated relationship between the software, electronic and mechanical engineering processes. However, many of these organisations are now turning to external resources and skill centres once again for reasons of costs and the reuse of common or standard components. But in doing so they will for the most part look to accredited providers and will often not make known or available the entire product schematics or architecture.

A market that echoes the needs and challenges of the mainstream IT industry...

In many respects those organisations charged with the software engineering process for systems and embedded software face similar challenges and issues to those faced by IT organisations responsible for delivering, operating and managing generic "business applications". Both communities have challenges with a variety of common issues such as:

- Accurate cost estimation and reporting.
- Requirements elicitation, validation and management.
- Effective communication and collaboration and across silos of working practices.
- Cross platform testing and validation of heterogeneous systems and environments.

...But with variations

However, there are significant variations and deviations in strategy, behaviour and process execution with respect to the mainstream IT industry. The world of SPE is different in its constraints and assessment of risks – especially legal, quality, security and safety risks. Organisations developing, managing and maintaining SPE components often have better-defined formal processes that are for the most part, strictly adhered to; and furthermore those processes tend to be significantly more automated. More effort is spent on documentation.

This is not surprising, given the clear understanding that exists in such organisations of the relationship between good products and revenue earned (or between bad products and revenue lost). In addition, strict adherence to such processes is a necessary requirement for development of software systems and components that will be used in highly-regulated and/or safety-critical products.

Their processes tend to be significantly automated with strong support for build and validation steps. More effort is spent on documentation (a regular dislike of developers) because it will be called upon in the event of a product failure – where the consequences can be costly and high-profile.

IT organisations have similar processes but with the exception of ISVs (who have good insight into the value of and the revenue earned from the software they develop), they are not always as strictly followed or automated as they are in the SPE market. At the same time, mainstream IT organisations are favoured with a lot more tools that specifically focus on ease-of-use through support for higher level declarative frameworks and languages.

Change drivers and challenges for SPE providers

Change drivers

In our research work interviewing SPE providers from across multiple industry sectors and pursuing a variety of different contracting models, we've found remarkable consistency in what providers tell us about the changes they're currently wrestling with. The chief technology and market changes affecting SPE providers are:

- **More software**. Products and systems are becoming more dependent on software for delivery of innovative features, and also for delivery of compelling user experiences for customers. Software also helps providers deliver feature variations across product lines without changing underlying product hardware (so reducing cost), increase product lifetimes, reduce support costs (enabling feature fixes to be applied to products without lengthy and costly returns processes), and deliver products and systems that can be monitored and managed remotely. The roles that software can play have also been enlarged due to the maturity and availability of low-cost, high-powered yet open embeddable computing platforms.
- **Increased focus on user experience, interaction and design**. Customers have high expectations and expect software based products to integrate and work connected and intuitively, particularly if being used in a work capacity. This drives the need for interfaces to become simpler with a greater focus on usability, interaction design and automation.
- **Development timescales driven down by the consumer market**. The more that consumer products become embedded with software components and systems, the shorter the lifecycles of the embedded applications become – with the need for more fluid updates to meet ever-changing expectations and requirements. As timescales for embedded development become shorter, many organisations see the need for greater interaction, communication and collaboration between software, hardware and electronics engineers.
- **Increased component standardisation**. Providers are moving steadily towards standardisation – for both software and hardware. Proprietary development no longer delivers an advantage – particularly in industry sectors like consumer electronics and appliances. This move has been largely precipitated by (as we mention above) the abstraction of programmable software controllers from hardware circuitry. But other factors prevail too: namely the need to minimise costs (through ready component reuse) and the need to bring products to market as quickly as possible.
- **Greater support for "open source" technology**. In our research, we see interest and support for technology delivered using Open Source models – in both platforms and tooling – across industry sectors. This is especially noteworthy given the complex, detailed and in some cases highly sensitive requirements and development environments that many in industry are trying to work with. Working with open-source technology is widely acknowledged across sectors as a means of achieving cost savings, particularly for acquiring development tools to help deliver functionality (e.g. unit testing) during key phases of the software engineering process. Many

organisations we’ve interviewed have experienced significant cuts in tooling budgets and are looking to use free products to help reduce their costs.

- **A drive towards model-driven development**. Prototyping is becoming prohibitively expensive. What's more the logistics can be complicated: it's not always possible to have prototype hardware available against which to test prototype software components, for example. There may also be widespread distribution of third party involvement in development, making it even tougher to bring components and skills together. As a result many organisations are looking to do more model-driven development and validation. The value of this isn't just about aiding prototyping: organisations we've interviewed see it as speeding up the development process (by identifying potential issues as early as possible); facilitating design and development communication across various engineering and development disciplines; ensuring project requirements, goals and outcomes are consistent across product development and delivery teams; and cementing the validation process as early as possible.
- **Improved software engineering processes**. There are three key areas within the software engineering process where we see significant evolution:
  - Firstly, in quality management (even in those sectors that already pay strong attention to quality, e.g. automotive, aerospace, defence and medical technology). Many organisations are looking to use defect analysis and architectural patterns to understand potential architectural mismatches between components.
  - Secondly, in formal verification. As SPE providers have begun to utilise many more third party software tools and components than previously, so there is a greater requirement for formal verification and certification. This is driving engineering teams to adopt validation and testing process that come much earlier in the development lifecycle (i.e. at the requirements definition and modelling stages).
  - Thirdly, in agile development. Increasingly, agile development processes are seen as playing an important role in speeding up software delivery – and also speeding up end-to-end product delivery. The need for a formal design phase will not go away; however, agile processes have their place and more patterns of appropriate use will evolve over time.

Challenges

In the context of the technology and market changes that we see, in our research work we've also found a high degree of consistency across sectors regarding the software development and delivery challenges that SPE providers report. The most commonly-cited challenges are:

- **Access to configurable tools**. Development processes need to evolve to effectively address the software engineering requirements involved in delivering smarter software-based products. Many of those providers we've interviewed feel that it's unrealistic to expect their organisations to retrain all developers; as a result they want tools that allow for the flexible configuration processes and workflows. Tools that require users to instantly change their way of working are not going to be successful.
- **Testing and validation complexity**. The increasing role of software within smarter products is resulting in the need for more testing at multiple stages of product delivery

process. However, writing test cases for many is too time-consuming. Moreover, in complex systems that rely on many assembled software and hardware components, testing and validation are highly challenging activities. The key issue stems from how testers characterise all the different contexts of use and all the possible types of interactions that can occur, and anticipate all the different risk points.

- **Architecture management**. As overall product/service offerings become more multi-tiered, architecture, design and testing strategies need to be considered from multiple perspectives and viewpoints, and all these need to be consistently mapped. Creating these viewpoints and perspectives and keeping them consistent requires sophisticated skill-sets (for example based around human interaction design) that have traditionally not existed in embedded systems development teams. What's more, the need for greater integration between different engineering disciplines and delivery processes (electronic, mechanical, software etc.) for smarter product delivery requires a skilled and experienced systems engineer/architect role that can understand and balance the challenges, requirements and dynamics of the different environments in play. These people are hard to come by.
- **Requirements definition and management**. This is a particularly strong challenge for a number of reasons. The first is the separation of non-critical and critical functionality; along with the separation of functional and non-functional requirements such that all stakeholders are clearly aware of all key constraints and goals. Identifying the right requirements and applying the right level of importance and focus to them is a major driver for reducing costs that most of those we've interviewed highlight as a significant challenge to manage.
- **Managing product and component variation**. This is a particular issue for component suppliers, who – as their customers seek to drive more and more reuse and standardisation – are having to deliver and support increasing numbers of product and feature variations. The issue of having to cater for all the different variations that a customer might request has forced some organisations to pull out of certain types of software development and engineering functions.
- **Source-code management (SCM) and change impact analysis**. Obtaining good visibility into the software engineering process and understanding the impact of changes becomes harder, the more control and responsibility for delivering software is outsourced to external parties. Some of the providers we've interviewed feel that too many SCM tools still focus on delivering file level version control rather than delivering control at the level of objects or other domain-relevant abstractions.
- **Uncertainty about implications of open-source use**. Many providers we've interviewed are concerned about licensing issues related to the use of open source technology and tools, especially when they're used in conjunction with commercial tooling. Interviewees frequently express uncertainty about whether using open source technology raises other issues – particularly with regard to keeping proprietary IP that has been added to a community-developed platform or technology separate from that which needs to stay in the open-source domain.
- **Tracking standards**. In large, complex, multi-component systems the number of technology standards that may come into play may be very large, and the dependencies, conflicts and other interplays between standards across multiple domains can be very complicated to track. Interviewees often cite a need for a set of established standards, architectural blueprints and practices that can drive interoperability and compatibility across many platforms and different parts of the supply chains that can be involved in complicated contracting models.
- **Managing legacy tools and processes**. Some companies are finding that they need to maintain their systems for longer timescales (e.g. 15 years) – resulting in the need to

maintain development environments that remain “live” for very long periods. This can have an impact on SCM practice; but more importantly, it raises the requirement for shared repositories that are easily accessible by multiple systems (as individual tools may come and go throughout the entire lifecycle of a product).

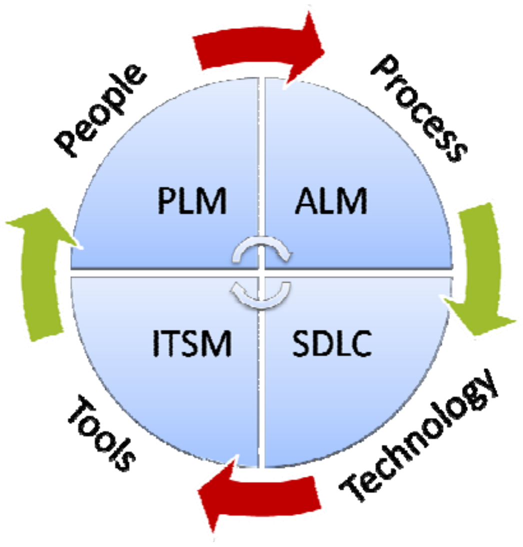
- **Informal use of tools.** Tools like Microsoft Word and Excel are commonly used to store information used in key phases of the development and delivery lifecycle (e.g. requirements capture and management). The use of such applications makes it harder to employ automation, maintain the integrity and consistency of the requirements process (without wrapping around extra authorisation steps) and make harder to take advantage of the wider benefits of a more integrated tooling platform.

Rules of engagement: practical guidelines for Smarter Product Enablement

All SPE market players require tools and techniques to help them deliver their products and components efficiently, effectively and to the required levels of quality. But the requirements of the SPE market players need to be considered less in terms of the delivery processes related to individual components; and more in the context of the processes that govern the delivery of a set of integrated software components and systems that together form part of a fully functioning product – be that a car, handheld device, industrial machine, consumer appliance or the avionic control system of a plane.

As with most areas of the software industry what is need is a connected model for engagement that focuses on people, processes, tools and technology. Already in the SPE market there are lifecycle models in play that address the product delivery and the software engineering process along with the subsequent management and maintenance of both product and software once deployed onto the market on put into operations. The lifecycle models of Product Lifecycle Management (PLM), Application Lifecycle Management (ALM), Software Development Lifecycle (SDLC) and IT Service Management (ITSM) each influences the other and share concepts such as the central role of process governance, workflow management and automation.. So when used, the lifecycle and management frameworks should interconnect and align if the most is to be gained and leveraged (see figure 2).

Figure 2: Lifecycle Alignment and Management



In It is important that the focus areas of people, processes, tools and technology should underpin each of these key lifecycle models when considering their practice. Moreover, neither of these key focus areas works in isolation of each other. They too intertwine and influence in conjunction e.g. tools can be used to support and address people and *process concerns.*

So what does this mean, exactly? It means that there are a number of things that SPE market players need to look for in order to help them to find the right tools and processes that will improve their software engineering delivery processes.

You need to apply strategies and environments that address the dynamics of different stakeholder processes (i.e. software engineering, electrical/electronic and product delivery) in a way that exposes and aligns the main goals/SLAs/requirements for each group to achieve better management control and so that better handover policies can be defined. Add to this the need to ensure that there is greater coordination and collaboration between people and teams and between the underlying systems.

People

Managing people and sourcing the right skills is a significant challenge facing the employment market across all industries. However, both must be addressed with careful management and dedicated support and the commitment to put in place the right tools and processes.

At the heart of the people focus is the need for better communication and collaboration between stakeholders, and better definition and management of requirements – especially ones that impact across the different lifecycle domains and engineering teams.

Collaboration and communication

Communication and collaboration between hardware, mechanical and software engineering teams can present a significant hurdle - hampering the overall product delivery process and resulting in too much redundant communication and rework between teams.

Investing in establishing collaborative integration practices, particularly at the start of the requirements capture and definition process, allow you to be better able at apportioning the right requirements to the different delivery teams (internal and external) and achieving a smoother overall process execution. Robust handover policies (automated where possible) developed collaboratively are also significant to the success of the product delivery process.

Strategies for collaboratively sharing code repositories within the organisation and with external partners are paramount for those organisations with widely distributed teams, especially those who rely heavily on third party providers for significant areas of the product delivery process or who are themselves third party providers (i.e. consultancies and system integrators with mature processes).

With the advent of web-based collaboration technologies, social computing and networks, it's quick and easy to take advantage of relatively sophisticated solutions for promoting collaboration (e.g. wikis, blogs, tagged data sites and online discussion forums and meeting services). The result of raising the experience and engagement of collaborative processes can lead to wider interest and support and better integration and co-ordination across multiple teams. In addition, we've found that smaller sized development and delivery teams are more capable of managing and fostering strong collaboration and communication links and processes.

Implementing sustainable collaboration amongst working teams is a constant pain point for many organisations. To achieve this requires a focus on people, processes and tooling supported by appropriate technology. On the tooling front what is required is a collaboration and integration platform that can help organisations join up their different tools. The goal should be to enable a more integrated software engineering approach that can span IT development processes, and facilitate shared access to common/reusable code or common development services.

Good collaboration begets process improvement

Investing in building strong collaborative practices and tools to support team collaboration can help to support much more sophisticated *post mortem* work and process improvement activities that can lead to more effective outcomes. Strong collaboration services expose more people to the successes and failures of a particular implementation, provide access to those involved, and help educate others in understanding the key criteria for project success or failure.

Systems integrators and consultants demonstrate particularly noteworthy support in this field, especially in expending more effort in determining which architectures worked well once deployed and which processes delivered the right outcomes. These organisations are on the whole spending more time putting in place automated processes for capturing lessons learnt; key metrics to ensure future delivery improvements; and better cost estimation. Others wanted the same level of control, reliability and repeatability are well advised to follow their lead.

Processes

Development and delivery

In general there are good levels of maturity in the development processes of most SPE players. We believe that on the whole, most manage the software development and engineering process relatively well and have reasonable quality assurance processes with central assessment teams promoting quality across the whole company.

That said, not all development processes are able to display "process resilience" in the face of constant changes in requirements. Tackling this can require small connected teams with less formal processes and more fluid development environments. Consumer electronics and appliance providers dealing with rapidly changing client requirements often follow this model.

As experience and engagement in interactions delivered on the back of software and advances in technology (screen, communication etc.) becomes more important, additional steps in the software engineering process or standard software development lifecycle take centre stage:

- Upfront prototyping (to establish initial design and ensure the right feature/function could be achieved). This is generally limited to areas that require strong focus and important decisions that need careful thought processes.
- Simulation. This is a particular consideration for those who did not always have access

to the hardware platform owing to the difference in delivery time leads, cost of providing the platform or the sensitivity of the product being built.

- Requirements validation. The use of requirements validation techniques is especially important for organisations delivering services or who are trying to prove the capability of a hardware platform.

Process automation is important, but it's important not to overdo it – the need for human intervention to review and authorise process changes is still vital from a governance perspective. You should seriously consider automation for functions such as the testing and build phases which require consistency, repeatability and load and can be programmatically executed, though.

Agile

We see general caution as to the appropriateness and relevance of agile development processes within the SPE market. This is because of the fear of risks in managing parallel working across software and hardware development processes. The required hardware platform may not always be ready at the same time as a software component. There may also be issues that arise with the hardware engineering process that must be resolved by software workarounds, thereby negating some of the benefits of parallel development.

Some perceive the practices underpinning agile in the software world to be more loosely and informally set. The active push away from formal design drives the perception that agile processes are too lightweight and unsuitable for use in the building of complex and often highly regulated systems and products. Here the step by step signed off progression and formality of the Waterfall methodology that caters for establishing the necessary upfront designs needed by such software engineering teams has taken root. This has been down to the complex dependencies and interaction of other software components and technologies (e.g. mechanical/electrical/electronic) that rely on a concrete design to be in place. Some in the SPE market worry that without the necessary upfront design, an agile process may create “drift” away from the strategic intent and overall design of a system.

However, there are already plenty of examples of complex systems being developed with a mix of agile development practices integrated within a wider formal methods framework. The success of such mixes is down to high levels of interaction, communication and collaboration with all participating roles and stakeholders at the start of the planning discussions to understand and locate the key dependencies and interoperability points. From here complex design and integrations are identified and put into early iterations to ensure that they are able to address the feature requirements of dependant processes and applications.

Although it goes without saying that detailed planning is still very much required, agile development can prove beneficial within complex systems with multiple interdependent components. The focus on commitment and delivering working code at the end of each iteration forces attention on addressing high risk items early on in a product release cycle. The quicker delivery of risky code allows dependent products to get quick access allowing them to improve their overall quality. The agile process helps promote prioritisation of key or known weaknesses and areas of challenge.

Tools and support services

Tools are critical in supporting the development and delivery processes. However process and having the right processes in play (manual or automated) should still outweigh tools as a top-line consideration.

Given the broad scope of industries where SPE principles apply, the tooling arsenal required can be varied and wide ranging, comprising in-house developments, commercial off-the-shelf platforms, packaged application workbenches, and open source tools. The widespread deployment of the Eclipse development workbench and IDE among many of the industry tool suppliers opens up the way for a common, unified and connected tooling platform however.

Support for standards remains a consistent and constant requirement to ensure portability, integration and interoperability. But so is the requirement for robust QA, static analysis and code coverage tools and Source Code Configuration Management repositories linked into a wider asset and artefact repository framework and build process and management tools.

In terms of sourcing the right tooling platform you should consider vendors that can provide:

- A unified environment that supports collaboration across multiple disciplines but that enables the sharing and governance of common goals, policy, requirements and outcomes.
- Role- and function-specific views and perspectives from tools.
- Industry specific implementation patterns and practices.
- A cross-platform environment that provides an integration and interoperability framework for tools, processes and data from multiple disciplines with out-of-the-box support for key best of breed products such as SCM, QA and Portfolio Management tools.
- Support and management of open source technology and tools.
- Tools that help improve key quality gates: relating to requirements elicitation, process validation and process improvement.
- Delivery of ongoing improvements in key development technologies, e.g. modelling and simulation support.

Employ the right strategies in the right way – making way for greater intelligence, analytics and internal assessment

Before employing strategies to address many of the focus areas and topics mentioned above, SPE players need to look internally and employ a wider assessment of their capabilities.

One of the strange challenges in getting organisations to improve their delivery processes is that process improvement appears to often be considered a lower priority than quality improvement, even though the two are obviously symbiotically linked. It appears that management teams only really start to address process improvement when the pain of staying with an existing process or tool far exceeds the challenges of making a change. Process change is seen as being high-risk.

Carrying out internal assessments can help clearly link issues of process and quality. By exposing areas of weakness and their impact on the execution process and the goals and outcomes a SPE provider is looking to achieve, such assessment strategies can make it easy to apportion revenue gain or lost as well as identify risks that could potentially be avoided with appropriate processes and tooling and process automation.

There is also a need to improve cost estimation, especially when it comes to understanding the impact that rectifying software bugs and resolving issues might have on the overall product delivery process and revenue potential. Cost estimation is made more reliable by having more visible insight into the processes in play and improved communication and collaboration processes with key stakeholders. Assessment tooling and processes that help identify the processes that are being employed well and the holes and gaps that need to be plugged to enable an organisation to deliver effectively against its own targets, clearly come into their own and pave the way for more realistic and achievable ROI.

Do more to address software variation management

There is increasing need to manage and support software and hardware variation more effectively to promote reuse and faster delivery of new product configurations. The use of software malleability and adaptability to support new features and provide greater intelligence and contextual interaction will require greater focus on the management of software variation. The management of software variation is a challenge, but managing both software and hardware variations is considered by most to be complex and challenging and in need of robust tooling and process support that bridge combined hardware and software variation requirements.

Source the right supplier support

Those suppliers who recognise and actively address the fact that no (or relatively few) organisations work in a homogenous tools or technology environment will be important to businesses facing multidimensional software development and engineering challenges.

Poor understanding of the architectural and governance issues impacting the different systems that are used within a product translates to a need of system engineers that are cognisant of hardware, electrical and software issues. However, there is a skills challenge with on-boarding "system architects" that have a more rounded view of the various environments and processes in play – not just software delivery, but product delivery too. It's important to focus on suppliers that provision for best practice "content" within tools to advance knowledge levels across skills and technology silos, and also to help foster a common language so that personnel with different domain skills can work together.

Moreover, suppliers that support broad partner ecosystem strategies and who actively engage in developing cross-domain and cross-industry architectural patterns and blueprints demonstrate the right attitude in understanding the challenges and demand for the cross domain skills shortage. More specifically you should look to those suppliers which can address convergence between product management lifecycle and software engineering processes and which are focused in tackling variation management.

Modelling and model driven development, with more emphasis on validation at the model level, should be a key focus area going forward with the value of abstract model-based software design, development, delivery and operation. The supply community needs to go further and provide greater prominence to modelling and the industry vertical or domain specific variations such as those of SysML, DoDAF, MoDAF.

Suppliers that support Domain Specific Modelling Languages help to provide an abstraction layer that architects and software engineers can use to describe common functionality and environments that can then be used to identify and manage variations. This will provide better insight into specifying the use of standard or off the shelf components and the differentiation points that add value.

RAL14021-USEN-00