RENESAS synergy™

Time To Market As A Competitive Advantage

There's no doubt that embedded system development is getting tougher all the time. In the latest UBM Tech Electronics Embedded Market Survey, 61% of total resources was spent on software. Meeting the delivery schedule is harder, too – only 38% of projects finished on or ahead of schedule, a tick downwards from the 42% - 44% of previous years.

"61% of total resources was spent on software. Meeting the delivery schedule is harder, too – only 38% of projects finished on or ahead of schedule, a tick downwards from the 42% - 44% of previous years."

October 2015

Microcontroller Project Trends

What's responsible for these schedule delays? Figure 1 tells the story: the main issues were debugging, test & integration, and code complexity. The top technology challenges were integrating new technology and code complexity (again).

Whatever the cause, missing the schedule increases Time To Market (TTM). And the consequences can be measured in lost revenue, reduced customer satisfaction, or financial penalties.

On the other hand, curing these software headaches - when your competitors can't – gives you a TTM advantage. And that leads to nothing but good things, especially in the competitive IoT market.

Let's take a look at some of these TTM issues and how they might be solved.

TTM Challenges: code complexity & debugging

According to a VDC survey of developers in 2014, the size of the embedded code base is increasing at roughly three times the rate of the number of embedded software developers being hired. When combined with tightening schedules, that's leading to increasing software complexity and use of third party software from both open-source and proprietary providers.

The result can be an uneasy blend of software from diverse sources: a company's own source and object code, externally-obtained binary executables, legacy code that may be out of date, purchased software IP, and miscellaneous other blocks of software, increasing the likelihood of low software quality, questionable reliability, and security holes that can be exploited.

As the code size grows, so do the errors: some studies estimate that every thousand lines of code produced by commercial software developers result in 20 to 30 bugs on average. As the code progresses through the development cycle, defects found become exponentially more expensive to fix - according to a 2007 Forrester report, it's at least 30 times more costly to fix software in the field versus during development.

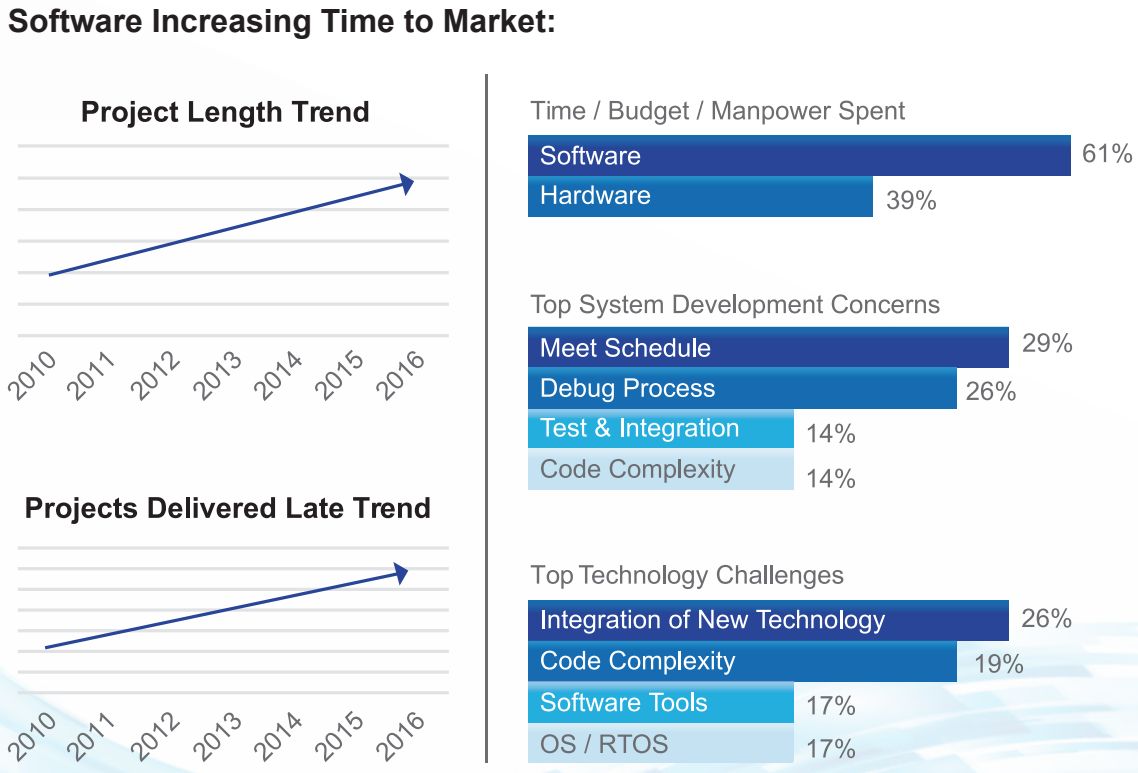


Figure 1: Software is a major contributor to increasing TTM (Source: 2014 UBM Tech Embedded Market Study)

According to a VDC survey of developers in 2014, the size of the embedded code base is increasing at roughly three times the rate of the number of embedded software developers being hired

How to reduce TTM errors

How can you reduce the incidence of errors? Research by Barry Boehm and his team at USC gives some guidelines, including:

- Statistically, about 80% of the defects come from about 20% of the modules (yes, that 80/20 rule again!), so identifying the characteristics of error-prone modules is time well spent.

- Peer reviews are an invaluable tool, catching up to 60% of defects.

- Disciplined personal practices can reduce defect introduction rates by up to 75 percent. One example is **Cleanroom Software Engineering**, originally developed Harlan Mills at IBM, which includes software development using formal methods, and implementation under SQL techniques.

- All other things being equal, it costs 50 percent more per source instruction to develop high-dependability software products than to develop low-dependability software products. However, the investment is more than worth it if the project involves significant operations and maintenance costs.

How about tools – apart from the ever-popular debugger, of course? Static program analysis using an automated tool can find errors by examining the source code - without actually executing the program. The analysis can take place at different levels: for example, within a specific program or between programs. Analysis of binary executables is much more difficult, but specialized tools are available that will check for such security vulnerabilities as buffer overruns and command injections.

TTM Challenges: integration of new technology

One of the issues with any new technology is that it tends to be more complicated that whatever came before. Security is a good example. Many systems that are now connected to the cloud were never designed with security in mind: an HP study found 70 percent of IoT devices to be vulnerable to security attacks, including webcams and home security alarms, with 80 percent of such devices lacking passwords of sufficient complexity.

An HP study found 70 percent of IoT devices to be vulnerable to security attacks

Given the current situation, highlighted by some well-publicized scare stories, security is a top priority for IoT-enabled embedded systems: according to researcher Markets & Markets, the IoT security market is expected to grow from USD $6.89 billion in 2015 to USD $28.90 billion by 2020, at a Compound Annual Growth Rate (CAGR) of 33.2% from 2015 to 2020.
For the embedded systems designer, IoT security must be tackled on multiple levels:

- **Protecting Communication:** requires encryption and authentication for devices to know whether or not they can trust a remote system.
- **Protecting Devices:** requires both code signing, to be sure all code is authorized to run, and run-time protection, to be sure malicious attacks don't overwrite code after it's loaded.
- **Managing Devices:** requires the capability to remotely update code to fix vulnerabilities discovered after the device has shipped.
- **System-Level Protection:** an IoT analytics capability that detects and flags network anomalies that might be malicious.

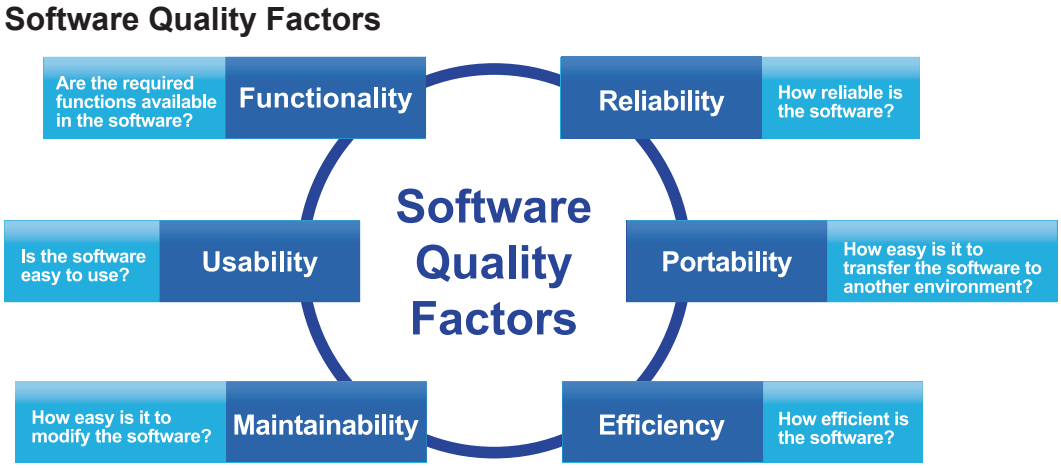


Figure 2: Software quality factors (source: Agile Software Solutions)

Integrating security into an embedded design is therefore a complex job: software security controls need to be introduced at the OS level, take advantage of any microcontroller hardware security features, and extend up through the device stack.

How can you ease the pain? Look for an integrated set of modules which includes security features such as a TLS1.2/SSL library, IPsec via IPv6, a cryptographic library and secure vault. Make sure you select a robust RTOS as well as microcontroller with hardware features such secure key storage, a random number generator, and an AES encryption engine.

"Adding manpower to a late software project makes it later."

– Brooks' Law, Carnegie Mellon University

TTM Challenges: knowledge transfer & training

Another problem inherent in software complexity is a human one: how to make sure that the whole team is familiar with all of the new code – its advantages and pitfalls, etc. Transferring knowledge between team members is critical so that past work - and past mistakes - are not repeated. This is made doubly difficult in a large corporation, where team members may be spread across the globe.

Bringing new team members up to speed isn't an easy task, especially if the project is already under time pressure. A famous observation known as Brooks' Law makes the bold claim that "adding manpower to a late software project makes it later", because the time required for new programmers to learn about the project and the increased communication overhead will consume an ever increasing quantity of the calendar time available; eventually the effect becomes negative and every extra person delays the project further.

To reduce issues related to knowledge transfer, it's important to have consistent coding standards, excellent documentation, and a software framework with state-of-the-art help tools such as context-sensitive smart manuals.

TTM Challenges: software quality

One way to view complexity is as a measure of the interactions of various elements of the software. According to an Intel study presented at the QA&Test 2014 conference, software complexity is a direct indicator of software quality and costs: if the complexity for any code is high, the quality of that code will be lower and it will cost more to manage it. Among the issues are:

- Higher risk of defects
- Difficult to add new functionality
- Difficult to understand/maintain the code
- Difficult to validate

In a fast-paced environment where there's continual pressure to add new features, over time the code becomes an unwieldy aggregation of old and new modules that is increasingly difficult to maintain. In this case some code refactoring be may required.

How do you know when your code is a good candidate for refactoring? One indicator is the detection of some symptom in the source code that may indicate a violation of a fundamental design principle – the imaginatively-named code smell. Examples include duplicated code, overly complicated design patterns, or a class that makes extensive use of the methods of another class.

One way to keep software quality from slowly degrading over time is to start with a software platform which has a cohesive underlying architecture and a set of rigorously-tested modules that meet software best practices such as SDLC protocols and applicable quality standards. Examples of these may include:

MISRA C: a set of programming guidelines developed by the Motor Industry Software Reliability Association (MISRA). Originally aimed at automotive applications, MISRA-C is now widely recognized as a leading guideline for C programming in the development of a broad range of safety-critical applications.

IEC-61508, IEC-62304 and ISO 26262: standards governing the functional safety of electrical, electronic, and programmable electronic safety-related medical devices, process control systems, industrial machinery, automobiles and railway control systems.

IEEE-730-2014: a set of requirements for initiating, planning, controlling, and executing the Software Quality Assurance processes of a software development or maintenance project.

Lessons from other fields

The problems confronting embedded software developers – the increase in code complexity, integration of new technologies, etc. - have faced by software professionals in other fields, such as automotive electronics and aerospace, for many years. These systems can have millions of lines of code, all of which must interact and communicate to enable the product to run smoothly. In some cases, a single minor defect in one line of code can affect the operation of the entire system, and can delay getting the product to market, add significantly to development costs, lead to product recalls, or even cause fatalities.

A modern passenger jet such as a Boeing 787, for example, has around 6.5 million lines of code, compared to 400,000 lines in the older Boeing 747. This is dwarfed, though, by the modern high-end automobile, which contains up to 50 embedded systems incorporating about 100 million lines of code, a huge increase compared to the early days – in 1981, GM was getting by with a "mere" 50,000 lines.

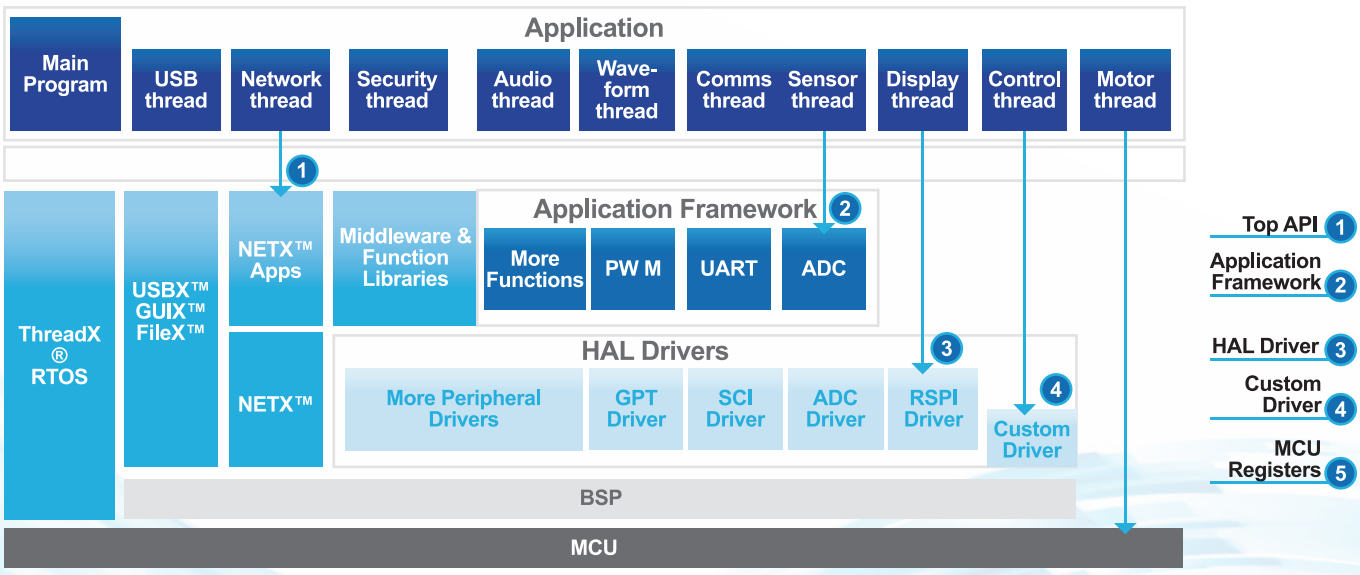


Figure 3: An embedded system development platform (source: Renesas)

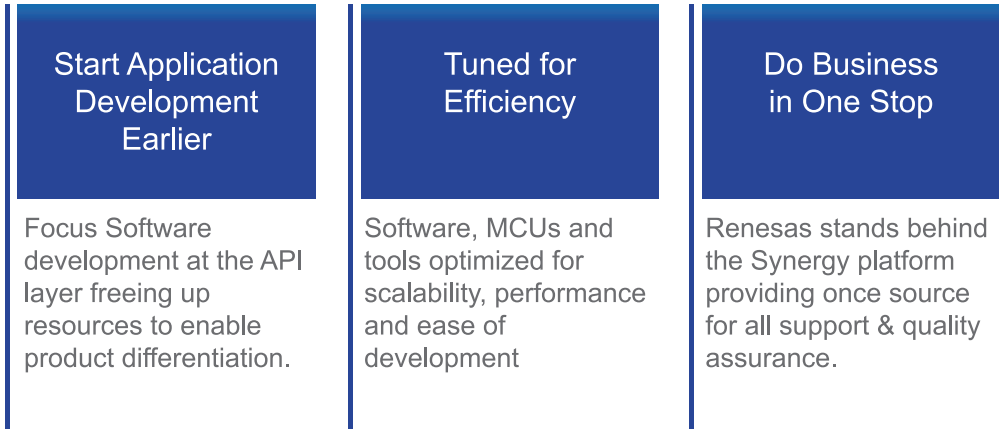


Figure 4: Synergy features to reduce TTM

Large, complex projects such as these demand a much different approach to development than the simple ground-up method where one-of-a-kind coding, patchy documentation, limited code reuse, and ad-hoc quality control has too often been the order of the day. For large projects – which increasingly include embedded IoT systems - it's advantageous to move to a platform-based development system as illustrated in figure 3. Platform-based development is a paradigm where the common elements that form the basis of almost all embedded IoT systems – the RTOS, security, device drivers, communications protocols - are identified, developed once, and used many times. Those components provide the base upon which to develop the application code – i.e., your value-added, differentiating features.

Projects developed using the features and common modules included with the platform have a much shorter TTM; a whole series of projects can be developed with maximum code re-use. At the base level of the platform, the microcontroller hardware is scalable to maximize code reuse at both ends of the price/performance spectrum.

How Synergy Helps You Reduce Time To Market

The Renesas Synergy Software Platform (SSP) includes numerous features to help reduce your TTM; it runs on a scalable family of microcontrollers based around the ARM Cortex-M4 core and incorporates a full-featured RTOS, a powerful IDE, and a rich library of standard modules including memory, connectivity, analog, timing, system and power management, security and encryption, safety and human machine interface functions.

You can find out more information at the Synergy site, but some features are particularly noteworthy in reducing the TTM of IoT embedded systems, with their unique requirements:

Synergy kits

The SSP includes kits and design examples designed to shrink the design cycle and shorten TTM. These pre-integrated hardware and software projects give developers at all stages of the design process the capability to quickly develop and test the capabilities of a design and help them begin writing application code as soon as possible.

There are three types of Synergy kits for general development around each of the Synergy MCU series devices, and two types of Synergy design examples to guide developers as they explore how to implement designs for specific end-products or use specific technologies with the SSP. Several levels of kits, from specific Application Kits illustrating particular IoT solutions to full-on Development Kits, are available based on a developer's particular requirements.

Security features

Synergy MCUs include significant new security capabilities in hardware, where they're less susceptible to attack. The MCUs integrate new security-related functions: when each MCU is manufactured, for example, it's assigned a unique 128-bit number which can be used to generate a unique key for each device.

The high and mid range devices feature hardware accelerators for both symmetric and asymmetric cryptography (including

RSA2048, DLP, ECDLP, and DSA 2048) as well as HASH (SHA1, SHA224 and SHA256). Each Series S7 and S5 Cortex-M4 MCU also features a true random number generator, an accelerator for asymmetric key generation and key secure storage. Series S3 MCUs feature similar security capabilities with symmetric key generation; S1 Series devices feature a true random number generator and basic encryption functions such as AES 128 and 256.

Software Quality

To ensure consistently-high standards, Renesas is responsible for software functionality and quality. The SSP incorporates an industry-standard design process and tools, including project management, configuration management, coding standards and analysis, test and quality assurance, and continuous integration.

The documentation validating the software package specification and quality (including all test data) is available for customer inspection. SSP software from third-party vendors must also meet the same quality standards under the Synergy Qualified Software Add-On (QSA) program.

The quality program has four components:

Software Best Practices:

- Renesas SDLC guideline document
- Requirements & Traceability
- Coding Standards
- Design Descriptions
- Code Reviews and Unit Test Development
- Continuous Integration and Integration Reports
- Release Process & Management

Software Data Sheet:

- Published and maintained on Renesas.com website
- Specs and performance metrics tested and documented
- Includes benchmarks, code size, context switch times, latencies, execution times, cyclical testing, fault tolerance and more.
- Basis of SSP

Software Quality Standards:

- MISRA C:2012 – Guidelines for the Use of the C Language in Critical Systems
- ISO/IEC/IEEE 12207 – Software life cycle processes
- CERT 2nd Edition – C Programming Language Secure Coding Standard
- Testing artifacts available for process certification - TUV & UL

Software Quality Assurance:

- Renesas SQA document – Software Quality Assurance Plan
- Requirements traceability throughout development
- Documented processes
- SQA metrics & process artifacts available to customers
- Test plans, test suites, reports

Conclusion

The days of low performance, simple MCU-based embedded designs are long gone. As an embedded system designer, you have to deliver higher performance, increasingly complex cloud-connected solutions in a fraction of the time.

To accomplish that task and meet shrinking development schedules, you need an integrated platform solution that includes all of the standard embedded system and IoT-specific building blocks, so you can concentrate on developing value-added, differentiated application code.

The Renesas Synergy Platform is a prime example of such a solution, offering you a powerful way to cut TTM and gain a competitive advantage in the marketplace.

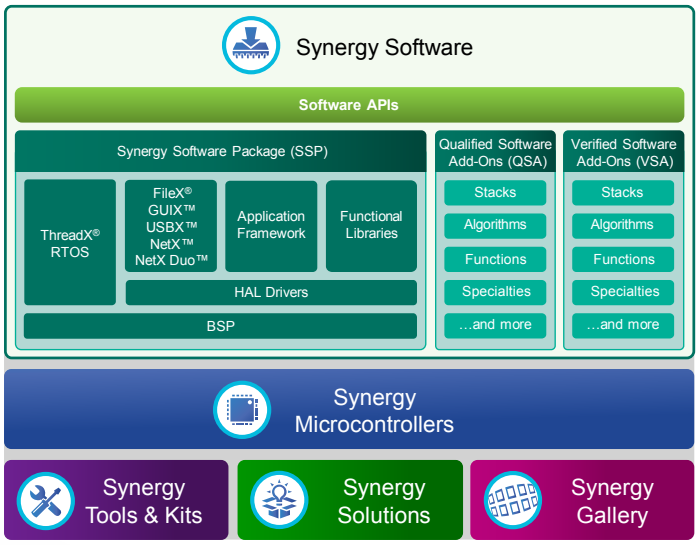
Introducing the Renesas Synergy™ Platform

Today, companies large and small race to capitalize on the rapidly growing IoT embedded system markets. Design engineers face many challenges – acquiring and mastering new technologies, developing code for low-level system infrastructure, performing integration and test, meeting aggressive schedueles – all while facing intense cost and resources pressures. A solid embedded software platform is the answer to these challenges by freeing resources to develop differentiated products instead of creating and maintaining the fundamental, yet essential system structure underneath.

Renesas introduces such a platform that is truly complete, fully tested and qualified, and systematically maintained and supported so you can start your application software development immediately at the API level without worry.

What makes the Renesas Synergy™ Platform unique?

Unlike other embedded development environments, all the Renesas Synergy™ Platform elements were designed from the ground up as a single platform. This provides unprecedented scalability and compatibility, allowing developers unparalleled code reuse. The platform will continue to grow, adding new technologies and features over time to keep your products on the cutting edge without new investments. To learn more, please visit: www.renesassynergy.com



RENESAS synergy™

Accelerate. Innovate. Differentiate.