On the Importance of Software Architecture for Embedded Systems

Claude Bélisle, NordiaSoft CEO
Kevin Richardson, US DoD Joint Tactical Network Centre

Abstract

Software has become the largest component of electronic systems today demanding more development resources than hardware. Software Defined Systems (SDS) provide the flexibility to be reconfigured to adapt to changing environment and requirements. However, as the underlying electronic becomes more complex with the rapid evolution of processors and heterogeneous nature of embedded systems, developing software has also become a challenge.

Conventional approaches whereby software applications are tightly coupled with the processing hardware is no longer acceptable. Any modification to the hardware (product evolution, obsolescence, third party technology insertion) requires significant adaptation which are prone to errors and time consuming. A paradigm shift is required in the development of software for Heterogeneous Embedded and Distributed Systems (HEDS). Military systems must be much more responsive to adapt to the constant and rapid evolution of opponent's systems.

In this article, we review how the complexity of those Software Defined Systems (SDS) impact the development of the software applications affecting both the time-to-market and cost. We vow for a paradigm shift in the software development process that will enable greater design flexibility and speedup the introduction of innovations, extend the lifetime of products and reduce the overall lifecycle cost. We examine how using open standard architectures principles, as a means to streamline development activities, can be very beneficial and how the Software Communications Architecture (SCA) stands out as a perfect solution for SDS.

Introduction

The proliferation of software in electronic systems over the past 15 years has been dramatic, enabling generic hardware devices to be specialized to execute multiple defined tasks. The most well-known example of this is our smartphone which can be turned into a social news exchange media (email, Facebook, Twitter and others), a web browser, a television, a navigation system, a gaming machine, a wallet, and even, a real telephone for those still using that feature. All those capabilities are enabled by the installation of software that makes use of the available hardware peripherals. As of mid-2017, there were close to 6 million 'Apps' in the Google, Apple and Windows stores [1]. Software has become the largest 'component' of electronic systems.

While the smartphone industry has been extremely successful in the development of software defined units, those devices are designed around single processors, single operating systems and a single data bus. In many cases though, embedded systems embody much more design complexity as for example in military and public safety communications systems, radar, electronic warfare, cybernetic systems, robotics, vehicle, and avionics. They incorporate multiple processors (GPP, GPU, DPS, FPGA, SoC), multiple operating systems (e.g. black and red sides in secure military radios), and multiple data transfer busses (serial, Ethernet, PCI, AXIe, etc.) over different boards, chassis or even systems. The diversity of the physical architecture complicates the development of software applications for such platforms, impacting on time-to-market and cost.

And yet, with the rapid advance of technology, it is imperative to reduce the development cycle. For military systems, flexibility and versatility are major requirements to enable rapid insertion of new features and new capabilities in order to keep an edge over the opponent. The term 'opponent' in the military context can be translated to 'competitor' in the industry one. As suggested by Megginson reflecting on Darwin's theory, 'It is not the strongest species that survives, nor the most intelligent one, but the one most responsive to change'.

In this article, we review how the complexity of those Software Defined Systems (SDS) impact the development of the software applications affecting both the time-to-market and cost. We vow for a paradigm shift in the software development process that will enable greater design flexibility and speedup the introduction of innovations, extend the lifetime of products and reduce the overall lifecycle cost. We examine how using open standard architectures principles, as a means to streamline development activities, can be very beneficial and how the Software Communications Architecture (SCA) stands out as a perfect solution for SDS.

Heterogeneous Embedded Distributed Systems (HEDS)

Those complex systems, exhibiting multiple processor types, operating systems, and data buses, where signal processing is performed in multiple program spaces, can be defined as Heterogeneous Embedded Distributed Systems (HEDS).

A recent survey produced by EETimes and Embedded.com [2], showed that 65% of the projects in the embedded industry are heterogeneous. When these products are upgraded, they include new or different processors or OS 40% of the time; new or different connectivity capability in 20% of the time; and require modifications because of discontinued software or hardware 17% of the time.

This heterogeneity raises significant challenges for the efficient development of high performing software applications. Conventional software implementation approaches, whereby applications are tightly coupled to the specific hardware configuration and rewritten or adapted when the operating environment changes (e.g. newer processor generation, new data bus) is no longer acceptable. The time and cost devoted to this task, not to mention the potential for software bloating and error introduction, is too high for today`s requirements.

Most requirements compel program managers, on both the manufacturing and acquisition sides, to minimize product development and sustainment costs. To attain those objectives, products must simplify software and hardware upgrades, minimize system integration activities, allow for product extensibility and maximize product quality. The design approaches that performed so well for homogeneous systems (such as smart phones and many consumer-grade electronics) are no longer compatible with this new reality. There is a need for a paradigm shift in the development of software.

Software Development Paradigm Shift

Successful development within this new environment needs to abstract, or decouple, the software from the platform environment. We are all familiar with the Applications Programming Interfaces (APIs) that standardize the data exchange with peripherals. However, what is less known is the importance of the overall software architecture on development time and cost. Such architecture must promote the development of software as components providing 'black-box' functionalities that can be reused between projects. Reuse differs from recycle because it implies that software is used 'as is', i.e. it does not require modification[1].

Reuse introduces the concepts of partitioning and code assembly. An application is created by assembling software modules that implement specific functionalities, similar to an electronic board assembled from chips. The partitioning of specific functionality within self-contained components facilitates software reuse because it eliminates dependencies between collections of software. In addition, the focused nature of the black-box code, reduces the potential of software bugs and code bloating that are often introduced by recycled code.

In the context of HEDS, a component based development (CBD) approach makes it much simpler to deploy the software (components) on the most appropriate processor, selected for processing speed, power consumption, or memory availability. This also enables parallel processing, a capability that is becoming quite important as we are reaching the limit of Moore's law; that is we can no longer count on transistor density doubling or increased clock speed to process more data. The ability to process data in parallel over multiple processors (not in multiple threads on the same processor which does not lead to parallelism) must be harvested. Loading individual modules in different cores of a multi-core processor or into multiple processors (GPP, GPU, DSP, FPGA) is quite natural when the application was developed using a component based approach.

Software Architecture

Software architecture is of utmost importance when designing such complex systems. One must think not only about what functional requirements need to be implemented, but how they should be implemented to achieve the program's requirements over its complete system life cycle.

The software that performs the actual data processing, the 'business logic', is most often only a small fraction of the overall software needed to run the application. The remainder of the software, the 'infrastructure code', includes logic to load, configure, start/stop the application, exchange data over the bus between sensors or processors, manage memory and file systems, and so on. The infrastructure code is extremely important to ensure proper execution of the application within the platform but has little to do with the fundamental purpose of the application, which is the signal processing. In most cases today, the infrastructure code is embedded in the application code, thus making it very difficult to adapt to different environments. For every change in the environment, the infrastructure code within the application must be changed. The more complex the system is, the more painful and error prone that adaptation can be.

[1] Recycling extracts existing code segments (objects, functions) and then adapts them within another project, i.e. copy, paste, and adapt

There is currently a push, led by the US DoD and various other military organizations worldwide [2,3] towards the definition of Open System Architectures (OSA) that would address the challenges evoked above. OSA defines a set of rules and behavior that are to be followed when building systems and applications.

When followed, OSA brings many advantages over closed, proprietary architectures:

a. Includes industry best practices process: Leading to improved overall system performance.
b. Development investment can be targeted towards business logic, relying on the OSA 'recipe' for the infrastructure: Improving application features
c. Facilitates interoperability of code between platforms, promoting software reuse: Reducing development time and cost.
d. Facilitates third party technology insertion: Reducing innovation cycle
e. Eco-system of tools and turn-key development environment available: Reducing time to market, development cost and improving overall performance.

The OSA principles result in an ecosystem that allows PMs and manufacturers to manage costs and risks throughout the development lifecycle.

Software Communications Architecture (SCA)

While there is a push towards open software architectures, none have achieved the level of completeness of the SCA for the design of HEDS. The origin of the SCA dates back 20 years. SCA version 2.2.2 was published in 2006 and have been deployed worldwide in over 400,000 radios [5]. The development of the SCA standard was initiated by the US DoD for its radio replacement program and is now evolved in collaboration with the Wireless Innovation Forum (WInnF), an international consortium of industry, academia and government labs promoting the development of software defined systems and innovative technologies for wireless systems.

The current version of the SCA specification, 4.1, was released in 2015. This version was developed to enhance the flexibility, performance and security of SCA products; leveraging the advances in processor and operating system capabilities. Version 4.1 allows application developers to make better use of multi-core processors and SoC, thus facilitating parallel processing capabilities and improved performance. System boot time has been significantly reduced with novel component registration approaches. System security was reinforced by refining several interfaces in accordance with the least privilege pattern to prevent consumers (expected and unexpected) from retrieving information about other system components. Lastly, lightweight profiles and a platform independent representation of the specification were developed that provide SCA developers with more freedom to align implementations with their technology stack of choice and mission focused requirements.

SCA Eco-System

As SCA has matured, a complete ecosystem of commercial off the shelf (COTS) products has been created to simplify and accelerate system and application development. Complete infrastructure stacks, including the SCA core framework, operating system, and inter-process communications mechanism, can be purchased and adapted to one’s platform. Software tools are available which enable the development of SCA-compliant devices or applications, automatically generating the infrastructure code based on graphical models of the platforms and applications. Fully integrated turnkey systems that can emulate the company’s product are also available. With those, the developer can now focus on the business logic of the application, rather than expending resources, time and cost, developing and maintaining infrastructure code.

Conclusion

OSA’s hold the promise of supporting the development of cost efficient, technically superior products. Through their use they can address the concerns and challenges of organizations that build and buy HEDS. OSA’s benefit from the accumulated knowledge of the organizations and participants that contributed to the content of the specification and can provide a means to mitigate cost and performance risks across product development and operations.

The SCA, with almost 20 years of existence, evolution, extensive battlefield deployments and worldwide adoption, provide an optimal representation of an OSA for HEDS. SCA is specified at a level that maximizes portability and interoperability yet leaves the door open for product developers to construct innovative solutions. The SCA model allows for decoupled development across hardware, software and services and supports an outcome that enables the production of secure, reliable, cost-efficient HEDS.

References

[1] https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
[2] www.embedded.com/electronics-blogs/embedded-market-surveys/4458724/2017-Embedded-Market-Survey
[3] http://www.acqnotes.com/Attachments/Open%20System%20Architecture%20(OSA)%20Contract%20Guidebook%20for%20Program%20Managers%20June%2013.pdf
[4] http://www.openarchitecturesummit.com/
[5] http://www.wirelessinnovation.org/assets/Collateral_and_Supporting_Docs/sca%20sell%20sheet%20march%202015.pdf