Embedded Software - the Revolution Continues

I'll admit to having been a software engineer in the late 1970s and early 1980s. I remember coding for 4,8 and 16 bit processors respectively to implement a sound generator for a children's toy, to manage an office telephone system, and to translate nightmare characteristics of some 'raw' graphics hardware into a programmable interface. The 'real-time' nature of the software linked these projects together. At the time, business justification for these sorts of projects was either easy or impossible. This was because in every case the still new'ish microprocessor technology looked as if it would deliver results in completely unexplored parts of the price / performance / functionality spectrum. You either believed, or you kept well clear!

Thirty years on, and the embedded software revolution has transformed design engineering in automotive, aerospace and both high tech and consumer electronics sectors. A 2005 NATO report estimated there would be 30 billion lines of code in a future US submarine. Between 10 and 25 percent of job adverts for software engineers mention embedded software. Many, probably more than half of electronics engineers working at chip level have to write or configure code to be embedded in their systems.

And now, traditional industry sectors are demanding the capabilities of embedded software. Touch screens, network connectivity, cameras, microphone/speaker systems, GPS, motion sensors and so on are all easy-to-order catalog items, often with software stacks that simplify integration. Product concepts previously regarded as exotic are now candidates for introduction next year. A wristwatch that monitors your blood pressure and uses your cell phone to transmit results to your doctor? It's been done.

From industrial machines to consumer goods, design teams know that embedded software will provide a growing part of the value and innovation in their products. This creates demand for software engineers, training, development tools and, critically, the capability to partition problems across technologies and still be able to review, test and integrate results.

So where can an engineering manager look for the tools to do the job? There's a lot of choice in this $2.6B market (*), with some 'turbulence' as both technology and provider boundaries change. For example, this year, PLM vendor PTC acquired embedded software tools provider MKS. Other global PLM vendors such as Siemens and Dassault Systemes offer systems engineering tools which integrate with their design systems, and cover multiple technologies. Dassault bought *Geensoft* and extended its requirements and automotive software capabilities. EDA companies such as Cadence, Mentor and Synopsys offer tools such as system-level design, virtual prototyping and requirements management in which embedded software is a usual component of, say, system-on-a-chip designs. IBM Rational offers system and software development tools with the option of a management environment that can monitor progress and performance indicators across an extended team. Business systems providers such as SAP and Oracle have relevant design and manufacturing applications covering parts libraries, bill of materials, version and status handling, data access management, and workflows. Oracle also offers technical development tools for embedded systems including Java tools specific to TV, smartcard and general embedded use. National Instruments offers the LabView system, a way of creating embedded software (especially for test systems) directly from diagrams. IBM's 'Rhapsody' also builds code from diagrams, integrating with requirements and test handling.

This is a long yet still very partial list. Microcontroller manufacturers usually offer an Eclipse-based free development environment. Tools usually interface to Mathworks Simulink multi-domain simulation software. And there is a growing market of software components for embedded systems.

The hands-on detail of each specific user environment is key to the choice of tools. But if I was getting into embedded software development for the first time, there are two 'big-picture' questions I would want to ask any potential tools provider:

"What about security?" Products with embedded software often have connectivity to the outside world. It's not just PCs that can get hacked when they are connected.

"How will I test and prove that the finished software matches the requirements?" Traditional engineering and especially electronics engineers and systems engineers are more likely to guide than be guided by software engineers in this area.

Just don't ignore embedded software.

(*): see http://www.cambashi.com/embedded-software-development-tools [1]

By Peter Thorne, Managing Director, Cambashi (www.cambashi.com)

Source URL (retrieved on *11/24/2014 - 4:49am*):
http://www.wirelessdesignmag.com/blogs/2011/09/embedded-software-revolution-continues?qt-blogs=0

Links:
[1] http://www.cambashi.com/embedded-software-development-tools