Australian National University

Effective Embedded Systems

ENGN8537

Embedded Systems

Overview

- Measures of Success
- Architecture Business Cycle
- Prototyping and Testing
- Maintenance and Support

Measures of Success

We all hope that our Embedded Systems are going to be successful, but what makes an Embedded System design and implementation successful?

Measures of Success

We all hope that our Embedded Systems are going to be successful, but what makes an Embedded System design and implementation successful?

Embedded Systems are designed and implemented to a specification to meet technical outcomes, but the success of the Embedded System though is defined with respect to business outcomes.

The design of an Embedded System is to design a technical solution to a business problem

Measures of Success

This implies that there are two equally important tasks in system creation:

1. Producing a good technical solution to the specified problem
2. Ensuring that the technical solution actually solves the business problem

Measures of Success

As Engineers, the student would be forgiven for believing that their duty lies entirely in the creation of a technical solution and that any shortfall with respect to the business goals will be due to poor specification from the Business Analyst, Client Liaison etc.

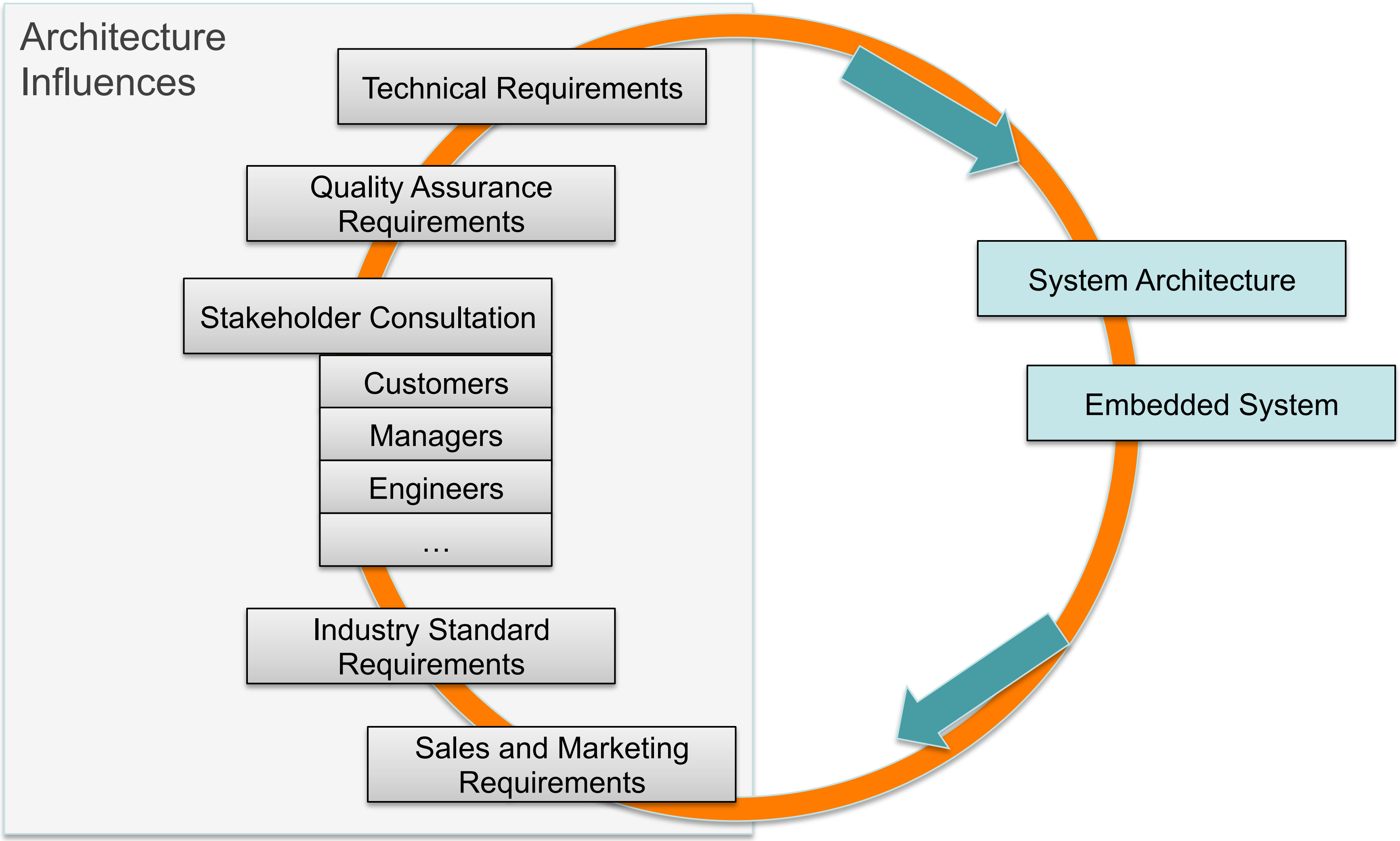
This is arguably true in theory, however in a small organisation the specification is likely to be drafted by the Engineer themselves. In larger organisations where a Business Analyst can and does draw the specification, it is rarely technically complete and the Engineer must be able to make choices that satisfy business goals

Measures of Success

Embedded Systems are so diverse in functionality as to make any survey of functional requirements outside the scope of this course.

Functional requirements are, however, only a small part of the overall system specification; others come out of the Architecture Business Cycle.

Architecture Business Cycle



Measures of Success

The ABC starts with requirements from sales and marketing; they have identified a new business opportunity and require a piece of technology to exploit it. Their requirements are added to industry standard requirements (such as the RoHS directive discussed in previous lectures) and the complete set of requirements are discussed by all stakeholders.

Measures of Success

Once the project requirements are stated, they can be re-cast as technical requirements. This is the point at which the business problem is stated as a technical problem; it’s the most critical to get right.

The technical requirements lead to the creation of a system which is then passed back to sales and marketing for evaluation.

Measures of Success

The ABC might be seen as competing with the System Engineering principles introduced through the rest of the degree, however this is not the case. The ABC is a minimal abstraction of the common elements of many other Systems Engineering processes, it's the necessary and sufficient subset for our discussions.

If a more detailed and formal process is required to implement the assessment of 'Sales and Marketing Requirements', draw out a 'System Architecture' etc., refer to a Systems text such as Blanchard and Fabricky.

Measures of Success

Most of the steps of the ABC have obvious consequences when skipped, however a few warrant closer investigation:

Don’t examine industry requirements separately from the sales and marketing ones. Engineers may assume that the client is sufficiently well versed in the field to identify environmental targets, required interoperability, required manufacturing process etc. This is rarely the case, the client and Engineer should both identify (potentially) relevant standards and bring their research together during the stakeholder meeting.

Measures of Success

Most of the steps of the ABC have obvious consequences when skipped, however a few warrant closer investigation:

Focus on getting a good technical solution, assuming that such a solution will be of high quality. Quality goals should be an input to the technical design process, not a byproduct. Quality will almost certainly have to be traded off against other variables and the value of the trade will be set by business outcomes.

Measures of Success

Most of the steps of the ABC have obvious consequences when skipped, however a few warrant closer investigation:

Asking stakeholders to commit to positions during a single stakeholder meeting. An Engineer may not know whether something is technically possible off the top of their heads. Similarly, a marketer may not have all their client research to hand and a customer may have an impulse reaction to a concept quite different from if they think about it for a while. One of the most dangerous things in design is committing to something too early.

Measures of Success

General feature outputs from an ABC may be obvious things to be traded off against each other, such as

Cost

Always minimised within the other requirements

Schedule, Time to Market

Always minimised within the other requirements

Risk Profile

Always minimised within the other requirements

Measures of Success

Some more interesting ones are

Capability

Should all features be implemented immediately or will some be held back to motivate a client upgrade later? Should the upgrade functionality be by unit replacement or firmware/software upgrade? Should multiple versions be offered with different capabilities, and what technical mechanism should control this?

Device Lifetime

Should the devices have a target lifetime that isn't simply the maximum within the other bounds? For example, should a poor quality battery be used to motivate the client upgrading in the future?

Measures of Success

Capability and Device Lifetime are interesting in that they are technical parameters that may be primarily or directly driven by business requirements.

Other requirements in this category might be portability (especially w.r.t software), availability, conformance to standards etc.

Prototyping and Testing

The ABC defines a circular pattern, implying that there must be different stages of prototyping and testing.

It’s rare that a specification is complete and correct at the beginning of a project; the marketplace may change, a feature may be modified once it’s seen in action, new influences may be thought of and integrated. The specification is often a living document, although of course its life should be strictly controlled.

Prototyping and Testing

The key to keeping the specification evolving in a sensible way is to control the prototyping and testing of the system.

Prototyping is the process of making preliminary systems, or parts of systems, in order to evaluate particular options for implementation. Minimising the number of prototype cycles is often seen as an effective way to minimise cost, however it is often a false economy as the end product may fall short of requirements.

Prototyping and Testing

It's important to distinguish between two different types of prototypes; functional and technical.

Functional prototypes are created during early iterations and don't share much in the way of technology with a final system. They are designed to allow rapid evaluation of design choices, everything from user interface design to data quality from particular sensors.

Prototyping and Testing

A common functional prototype for Embedded Systems is to use an off the shelf hardware design and simulated peripherals to allow software development to commence.

Examples might be:

The use of an off the shelf consumer tablet as a development system for a touchscreen medical device, even though the final system will be unlikely to share the same hardware, or even library sets.

The recording and comparison of data from supplied evaluation kits when assessing GPS module performance, even if the final system will be much more tightly integrated.

Prototyping and Testing

A good functional prototype is as high fidelity as can reasonably be expected, but no more. It should be cheap, quick and easy to debug.

Focus must be kept on its core purpose, the phrase “while we’re at it, let’s just add X as well” doesn’t often lead to good prototypes as unforeseen interactions between untested parts may end up taking time away from the actual purpose for which the prototype was created. For the same reason, it should not attempt to fit inside final enclosures or exactly represent final choices of components unrelated to the purpose of the prototype.

Prototyping and Testing

Once functional prototypes determine the best components and interfaces, technical prototypes can be employed to start testing interactions.

These interactions may be between different subsystems or between components and enclosures.

Prototyping and Testing

Prototypes exist in order to be tested. Like interfaces, there's often no need to re-invent the wheel when you can standardise:

Mil-STD-180G	Dynamic (vibration and shock) and Environmental tests for electronic equipment
Mil-STD-202	Test methods for electronic components
Mil-STD-750	Test methods for semiconductors
IPC-610	Acceptability of Electronic Assemblies

and many more

Prototyping and Testing

It’s important that testing be done to a top-level specification. The requirement from the customer may be “quality better than competitor X” or “lifetime more than Y under normal use”.

By what metric do you measure that quality, and under what conditions? What conditions qualify as “normal use”? These questions are usually driven by the Engineer but must be justified in writing and agreed to by the client.

Prototyping and Testing

Example: GNSS Modules

From the data sheets

	uBlox MAX7	Fastrax IT600	Skytraq Venus
Power consumption	69mW	170mW	60mW
Accuracy	2.5m RMS	3m RMS	2.5m RMS
Time to First Fix (cold)	29s	35s	29s
Sensitivity	-162 dBm	-161 dBm	-165 dBm
Update Rate (max)	10Hz	10Hz	20Hz
Constellation Compatibility	GPS, GLONASS	GPS, GLONASS, BAIDU, GALILEO	GPS

Prototyping and Testing

Example: GNSS I

From the data she

Skytraq module has the lowest power consumption, best sensitivity, highest update rate and equal best accuracy.

	uBlox MAX7	Fastrax IT600	Skytraq Venus
Power consumption	69mW	170mW	60mW
Accuracy	2.5m RMS	3m RMS	2.5m RMS
Time to First Fix (cold)	29s	35s	29s
Sensitivity	-162 dBm	-161 dBm	-165 dBm
Update Rate	10Hz	10Hz	20Hz
Compatibility	GPS, GLONASS	GPS, GLONASS, BAIDU, GALILEO	GPS

Prototyping and Testing

Example: GNSS Modules

In Testing

	uBlox MAX7	Fastrax IT600	Skytraq Venus
Power consumption	Not significant w.r.t overall system consumption		
Accuracy	2-5m	3-7m	10-15m
Time to First Fix (cold)	~30s	~30s	30-60s
Sensitivity	Irrelevant! Accuracy and TTFF are all that matter		
Update Rate	5Hz is all that's required		
Compatibility	GPS constellation the only one actually used		

Note: Figures loosely based on actual test data, modified for the sake of illustration

Prototyping and Testing

Example: GNSS Mo

In Testing

The benefits of the Skytraq module on paper either failed to materialise during testing, or were irrelevant to the actual use case at hand

	uB [illegible]	[illegible]	Skytraq Venus
Power consumption	Not significant w.r.t overall system consumption		
Accuracy	2-5m	3-7m	10-15m
Time to First Fix (cold)	~30s	~30s	30-60s
Sensitivity	Irrelevant! Accuracy and TTFF are all that matter		
Update Rate	5Hz is all that's required		
Compatibility	GPS conste		

uBlox module performed best against the metrics actually relevant to the final product

Note: Figures loosely based on actual test data, modified for the sake of illustration

Prototyping and Testing

Example: GPS Modules

The values given in data sheets are usually theoretical, or at least “best case”. Testing the GNSS modules in situ with actual update rate settings, antenna choice and limiting the constellation to just GPS changed the performance of the modules significantly. When tested against the specified product goals, the front-runner on paper was the worst performing in the application.

Prototyping and Testing

The uBlox module had the best performance against the specification, but almost wasn't selected

.. how about Skytraq, we might want 20Hz in the future! Consider the likelihood and time frame against the degraded performance in the short term

.. how about Fastrax, it'd be great to say that our stuff works with GALILEO and BAIDU! The lifespan of the product is such that it's expected to be obsolete before those systems are operational

Stick to the specification. If 20Hz or BAIDU are genuinely good things, change the specification and test again from first principles

Maintenance and Support

System maintenance and support come last in the product lifecycle, but must still be actively considered during the development phase.

What will the warranty period be and how does that change component choices?

How maintainable must the system be, will it be upgraded in the future or is it disposable?

What's the expected system lifetime?

Maintenance and Support

Comparing two modules, Module A with a mean time between failure of 50,000 hours and Module B with MTBF 10,000 hours. The product is expected to have a lifetime of 8,000 hours.

Module B may be expected to fail towards the end of the product lifetime. Will the client be put off the brand because their units start to fail right around the time they're looking for new ones? Or will the failures encourage them to upgrade sooner? The MTBF is an average so Module B might fail more under warranty, does this impact costing strategy?

Maintenance and Support

Development questions such as these only have their answer in the maintenance and support strategy for the system.

This, like everything else, must be specified and tested against as part of the core development procedure.

Maintenance and Support

The inclusion of maintenance strategies in design specifications seems intuitive, however it's often overlooked. Sometimes accidentally as the actual field failure of a product seems so far away from development. Sometimes intentionally as the client wishes to get something to the market, what happens when it gets there is seen as incidental.

Of course, what good is getting your foot in the door if the marketplace is going to react badly and cut it off?

Maintenance and Support

Summary

- Successful Embedded Systems perform well with respect to business goals, not technical ones
- The synchronisation of technical goals with business ones may be the most important part of the design process
- Systems Engineering principles apply, but if all else fails, remember your ABCs
- Prototypes and testing must be directly relevant to the specification, nothing else matters (if it did, it would be in the specification!)