

Views (sometimes heretic) on Embedded System Design

Presentation Technology Leadership Day
2005-10-11

Andreas Stucki
Solcept AG

Contents

Background

System design & its problems

Ideas for improvement = lessons learned

Background

Contract R&D

Also applicable in a general R&D context

Each example/ problem we have experienced/ seen ourselves

What is a System?

Embedded system

Electronics (software & hardware)

Typically containing 1..several processors/ controllers

Mechanics

Other technologies or sub-technologies

Optics, sensors, RF,...

What is the Goal of System Design?

Good system design

Leads to:

Simple development of subsystems/ modules

Delivery of the functions the user wants

Maintainability

Extendability

Note: system design is not learned at university

What is the Reality?

A lot of systems:

- are difficult to develop & integrate

- have interfaces which are unclear

- have poor maintainability & extendability

... maybe this can never be

remedied completely, but at least we can try

Start at the Beginning Especially with specifications

Develop use cases

On usage

And also on the whole life cycle! (like installation, repair)

Only then search for a technological solution

E.g. the processor type and OS is defined before the requirements are clear, this leads to overpowered, overpriced and undercooled solutions

Write the Testlist First ... or together with the specifications

The two documents will reveal problems in each other quite efficiently

Interfaces

Missing use cases

Design for test

etc.

Freeze Specifications Late

As late as possible

Changes are part of the process:
“agile” approach

This is nice for software, but what about hardware/ mechanics!

Agile up to manufacturing (also for prototypes)

Needs adapted process/ organization

Agile Manifesto: “www.agilemanifesto.org”

Software is King ... and hardware adapts

Develop hardware according software (e.g. OS)
used

Which hardware is supported by the software?

Use existing concepts from e.g. evaluation boards

E.g. missing Ethernet port and non-supported Flash for a Linux
system add to development cost and time

Hardware works Always

... or at least this is what customers expect

The only differentiator is software

Bad for “hardies”:

as soon as a controller is inside, software gets most important

E.g. your mobile phone: no one marvels about the high-tech PHY layer or cool protocol stack when the menu is difficult to use

Is Product Cost Really King? Is your production lot really *that* big?

Saving money on the purchased parts/ manufacturing can severley backfire on

Development

Time/ time to market

Reliability

E.g. 10'000 pcs. production, an IC which is 1 CHF cheaper: can the change be done within 10 days of engineering?

System Partitioning is Key

For now and the future

Intelligent, but sometimes counterintuitive partitioning can save money

Keeps interfaces simple

Allows redesign of isolated parts for technology-tracking and market adaptation

E.g. a USB service interface for a high reliability processor: a separate, cheap processor saves money on the regression-testing and qualification of the whole system (and retains extendability)

The Biggest Problems are the Small Ones

Beware of them showing up too late

System designs fail on small problems

Problem sources are always “secondary”

keep all side-effects in mind: good use-cases can help!

Examples:

supply/ cooling

updates/ downloads

servicing

mechanics (size, weight, stability,...)

EMC

Do everything completely different!

Every rule must be bent sometimes:
Be pragmatic!