Marc Brooker

EEE3074W Lecture 05

Embedded Systems Design Lifecycle

6 March 2007

7 Phases of the Design Cycle

Design your own 7 stage embedded systems design cycle.

7 Phases of the Design Cycle

1. Specification
2. Partitioning into HW and SW components
3. Iteration and Implementation
4. Design of SW and HW done independently
5. Integration of SW and HW components
6. Acceptance Testing and Release
7. Maintenance and Upgrade

Remember: SPIDIAM

The Tool Based View

1. Processor Selection
Based on experience
Features, availability of simulators, vendor support, etc

2. Design Phase
Software design team and hardware design team work separately

3. Software/Hardware Integration

4. Maintenance and Upgrade

Specification: The Customer Research Tour

- Four People – Including Engineers and Marketing/Sales people
- Prepare for the visit – plan interview questions
- Each member has a specific role
- 1 Leader
- 1 Technical Writer/Note Taker
- Rest of the team offer technical input or just observe
- Followed by a debriefing

Specification: After the Tour

- Translate notes into a set of formal product requirements
- Requirements intended to guide the team in the development phase
- Further tours might be useful (or required) to fine-tune requirements
- Incorrect requirements can be extremely costly

Specification: The Team's Vision

The entire team needs to share a common vision of the product.

You **do not** want:

- Manager thinking they are making a pogo stick
- Marketing selling the customer an automated beer maker
- Engineers developing a 500kW supercar

In the end the client will be dissatisfied – company will get a reputation for incompetence

Specification: Tools

Development tool requirement should be part of product specification.

This prevents setting unrealistic goals for the tools.

The textbook (Berger) recommends:

- Begin meeting with a list of **musts** and **wants**
- All stakeholders must agree the list is valid
- Project must not continue until **musts** and **wants** are resolved.

Hardware/Software Partitioning

The goal of this phase is to decide:

- What is to be done in hardware
- What is to be done in software

Design team must understand the importance of the decisions to be made.

- Algorithms can be generalized into hardware and software components

The Hardware/Software Tradeoff

Hardware

- Higher development cost
- Extremely expensive to fix errors (eg new ASIC and PCB fabrication runs)
- But can be much faster than software for some tasks

Software

- Potentially lower development cost
- Easier to fix errors (and refactor if necessary)
- Costly to fine-tune performance

Partitioning is Optimization

- Lowest possible cost
- Best possible performance
- Novelty (important for consumer products)
- Market Competitive (better than competitors)
- Proprietary (can be patented)

Need to find the Pareto optimal solution to the partitioning problem

Can be extremely complex

Iteration and Implementation

This phase includes early design work and refining of the hardware/software partition as more information becomes available.

Hardware Designers

- Use simulation tools to model system performance
- Possibly build early prototypes

Software Designers

- Running benchmarks on single board computers
- Using processor simulators to evaluate performance

Detailed HW and SW Design

1. Collaborate
2. Review (and refine the requirements)
3. Plan (and revise planning docs)
4. Design (and revise design docs)
5. Implement (and revise implementation docs)
6. Test and Demonstrate (and revise testing docs)

Can The Requirements Change?

This depends on company policy. There are several useful rules:

- Don't over specify
- Clients appreciate flexibility (mostly because they weren't sure what they wanted originally)
- The requirements tend to change as the design process continues
- Clients need to be "**in the loop**" (the old "clean front room and busy back room" approach is fading out)

Can The Requirements Change?

However:

- The further down the design and implementation process the more complex/costly a change in requirements is going to be
- Refactoring can be extremely expensive
- Many projects (and companies) have failed because requirements changed at the wrong time

Most Importantly: A balance is needed between flexibility and inflexibility.

Hardware/Software Integration

Many bugs and problems are revealed at this stage

This is mostly because of two things:

- Incorrect specifications (can be prevented)
- The complex and chaotic behaviour of realtime systems

It is extremely hard to model this accurately with a simulator, because:

- Simulators run much slower
- Simulators are not identical to reality

Product Testing and Release

- Testing is extremely important and can be extremely stringent
  - Especially for life critical systems (medical, avionics, etc)
- Testing criteria can include:
  - Reliability
  - Fault tolerance
  - Power consumption
  - Performance

Who Does Testing?

Pre-release testing:

- Should **not** be done by the design team
- Should be done by a dedicated test team
- Experience shows designers do not test effectively

Who Does Testing?

Compliance Engineering (CE):

- Extremely important
- Can result in product liability if not correctly handled
- Must be done by expert CE engineers
- Outsourced by many companies

Maintenance and Upgrade

Most embedded systems designers maintain and upgrade existing products.

Maintenance and upgrade not usually done by the original design team.

Process relies on:

- Skill and experience of maintenance engineers
- Documentation of old product (extremely important)
- Product itself
- Sometimes even reverse engineering is required

In the Next Exciting Lecture

It's time to get your hands dirty and design a real embedded systems product.

The Project