The Embedded System Design Process

Wolf Text - Chapter 1.2

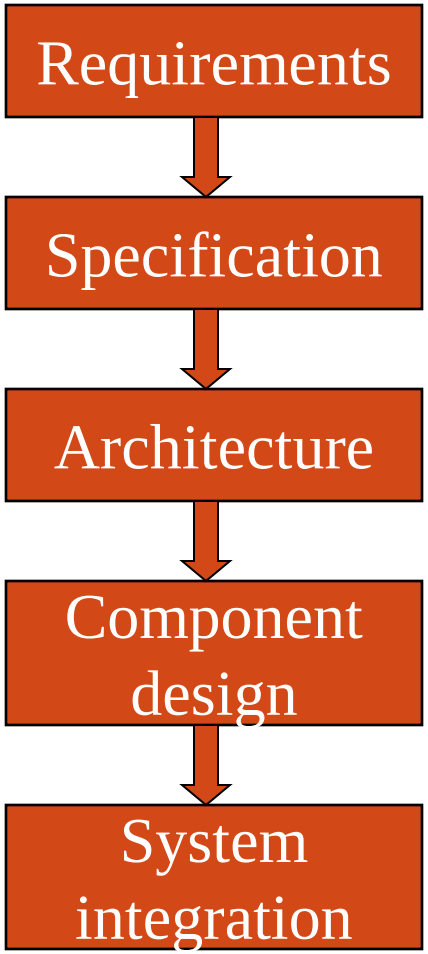
Design methodologies

- A procedure for designing a system.
- Understanding your methodology helps you ensure you didn't skip anything.
- Compilers, software engineering tools, computer-aided design (CAD) tools, etc., can be used to:
  - help automate methodology steps;
  - keep track of the methodology itself.

Design goals

- Performance.
  - Overall speed, deadlines.
- Functionality and user interface.
- Manufacturing cost.
- Power consumption.
- Other requirements (physical size, etc.)

Levels of abstraction



Top-down vs. bottom-up

- Top-down design:
  - start from most abstract description;
  - work to most detailed.
- Bottom-up design:
  - work from small components to big system.

- **Real design uses both techniques.**

Stepwise refinement

- At each level of abstraction, we must:
  - analyze the design to determine characteristics of the current state of the design;
  - refine the design to add detail.

Requirements

- Plain language description of what the user wants and expects to get.
- May be developed in several ways:
  - talking directly to customers;
  - talking to marketing representatives;
  - providing prototypes to users for comment.

Functional vs. non-functional requirements

- Functional requirements:
  - output as a function of input.
- Non-functional requirements:
  - time required to compute output;
  - size, weight, etc.;
  - power consumption (battery-powered?);
  - reliability;
  - low HW costs (CPU, memory) for mass production
  - etc.

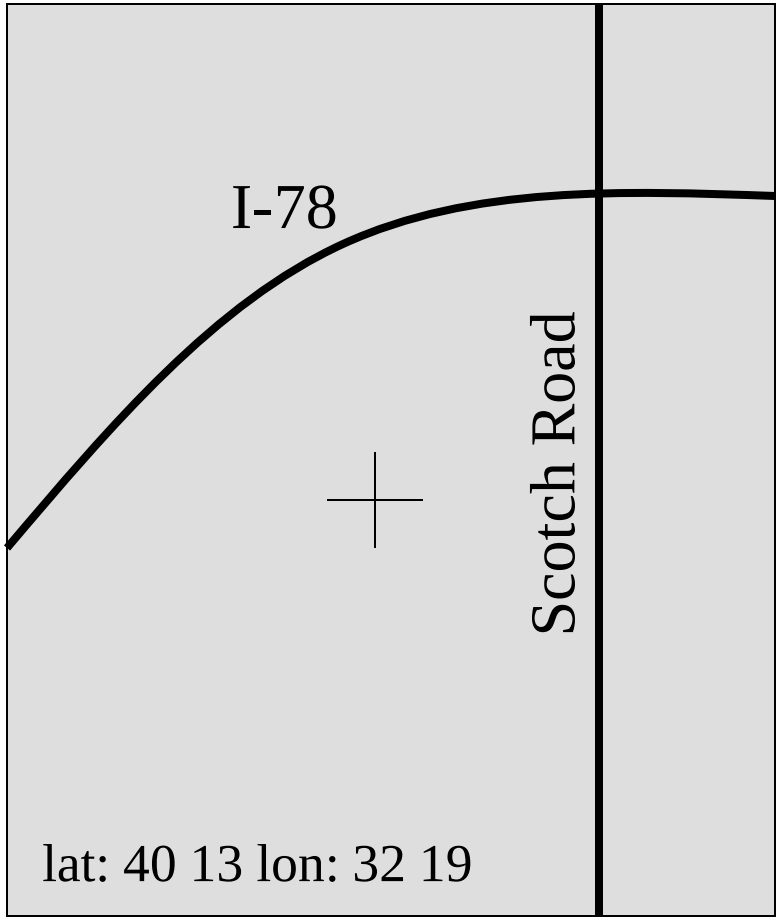
Sample requirements form

Use form to assist “interviewing” the customer.

name
purpose
inputs
outputs
functions
performance
manufacturing cost
power
physical size/weight

Example: GPS moving map

- Moving map obtains position from GPS, paints map from local database.



GPS moving map requirements

- **Functionality:** For automotive use. Show major roads and landmarks.
- **User interface:** At least 400 x 600 pixel screen. Three buttons max. Pop-up menu.
- **Performance:** Map should scroll smoothly. No more than 1 sec power-up. Lock onto GPS within 15 seconds.
- **Cost:** $500 street price.
- **Physical size/weight:** Should fit in dashboard.
- **Power consumption:** Current draw comparable to CD player.

GPS moving map requirements form

name	GPS moving map
purpose	consumer-grade moving map for driving
inputs	power button, two control buttons
outputs	back-lit LCD 400 X 600
functions	5-receiver GPS; three resolutions; displays current lat/lon
performance	updates screen within 0.25 sec of movement
manufacturing cost	$100 cost-of-goods-sold
power	100 mW
physical size/weight	no more than 2” X 6”, 12 oz.

Specification

- A more precise description of the system:
  - "What will the system do?" (functions, data, etc.)
  - should not imply a particular architecture;
  - provides input to the architecture design process.
- May include functional and non-functional elements.
- May be "executable" or may be in mathematical form for proofs.
- Often developed with tools, such as UML

"Contract" between customer & architects

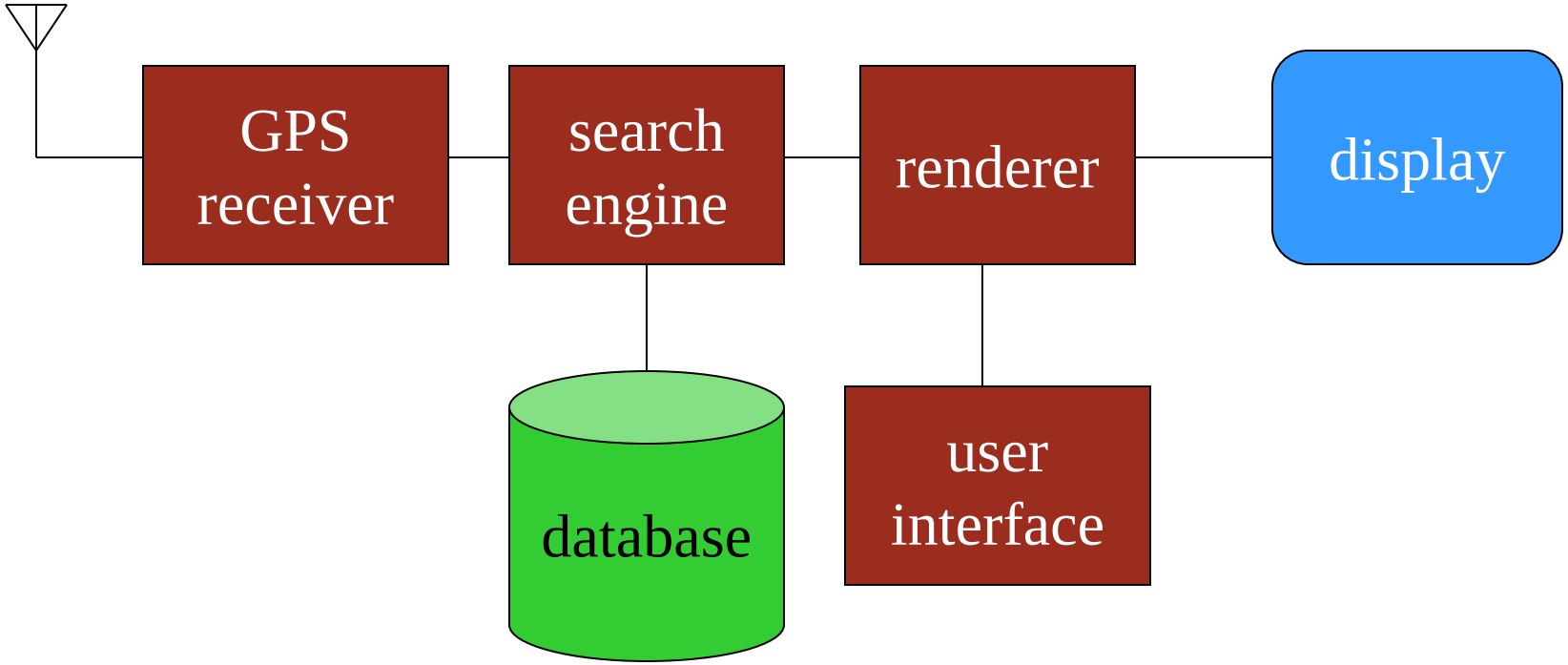
GPS moving map specification

- Should include:
  - what is received from GPS (format, rate, …);
  - map data;
  - user interface;
  - operations required to satisfy user requests;
  - background operations needed to keep the system running.

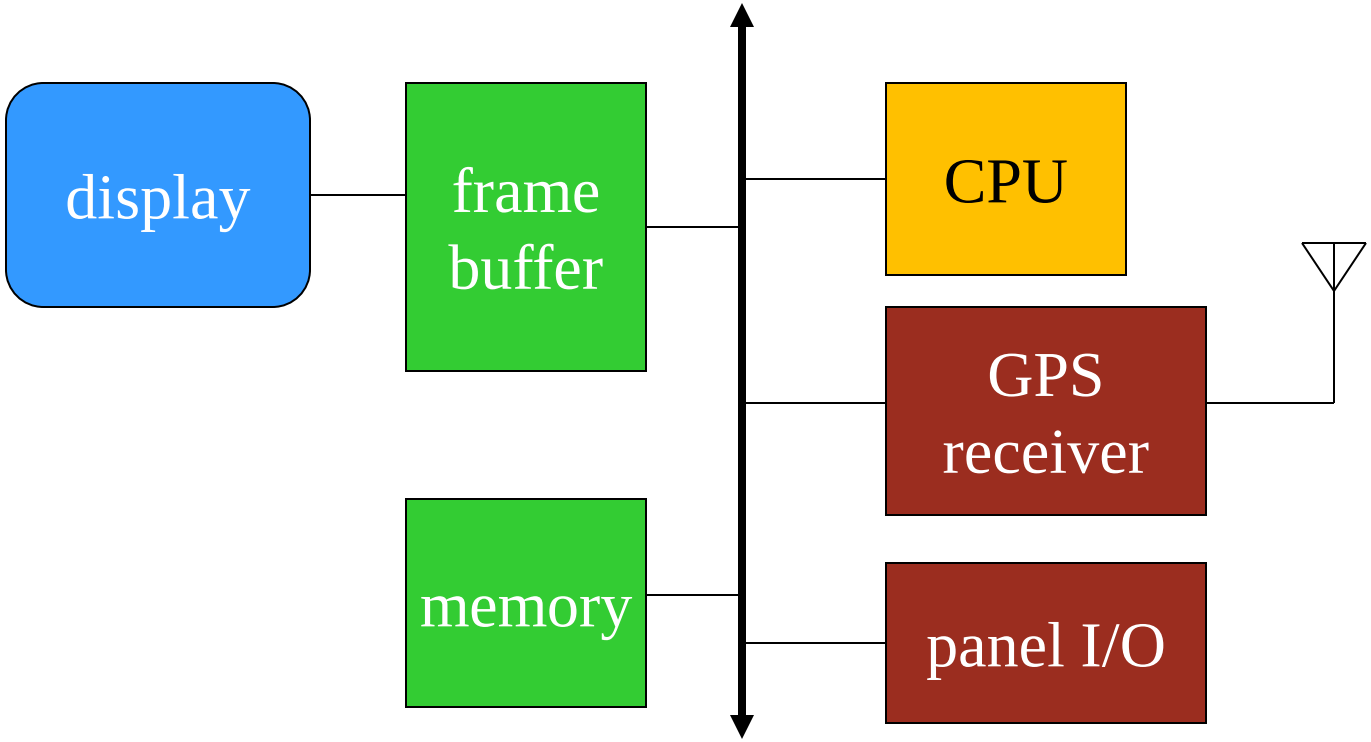
Architecture design

- What major components go to satisfying the specification?
- Hardware components:
  - CPUs, peripherals, etc.
- Software components:
  - major programs and their operations.
  - major data structures
- Evaluate hardware vs. software tradeoffs
- Must take into account functional and non-functional specifications.

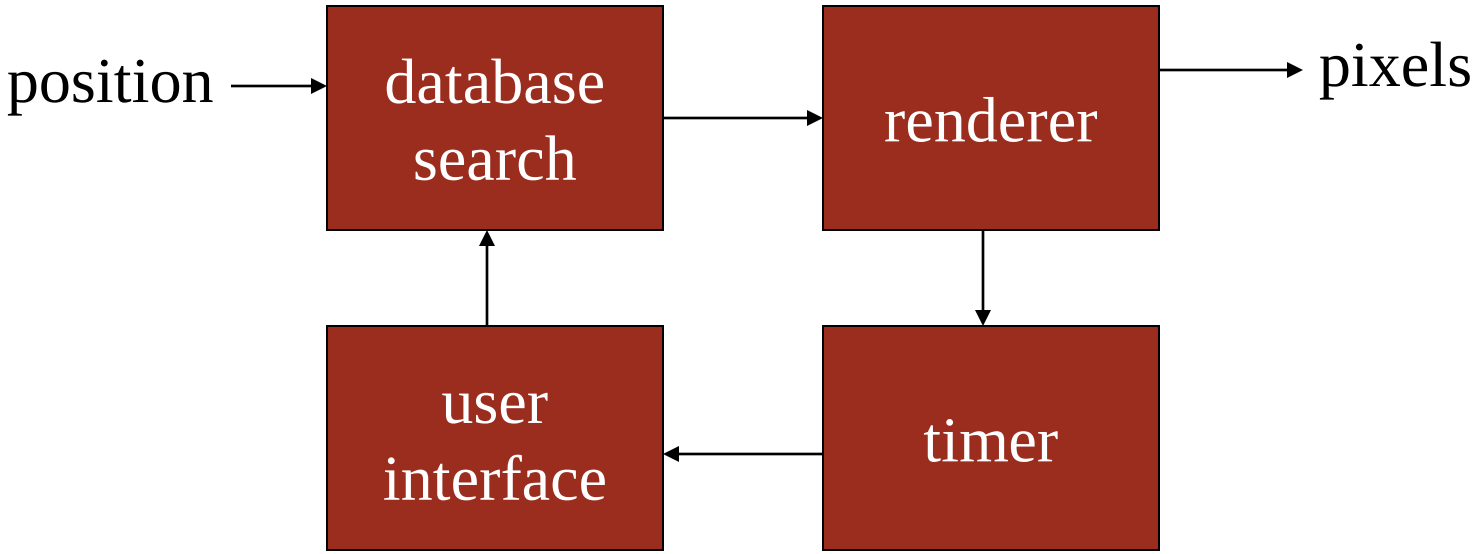
GPS moving map block diagram



GPS moving map hardware architecture



GPS moving map software architecture



Designing hardware and software components

- Must spend time architecting the system before you start coding or designing circuits.
- Some components are ready-made, some can be modified from existing designs, others must be designed from scratch.

System integration

- Put together the components.
  - Many bugs appear only at this stage.
  - Interfaces must be well designed
- Have a plan for integrating components to uncover bugs quickly, test as much functionality as early as possible.
  - Test to each specification

Challenges, etc.

- Does it really work?
  - Is the specification correct?
  - Does the implementation meet the spec?
  - How do we test for real-time characteristics?
  - How do we test on real data?
- How do we work on the system?
  - Observability, controllability?
  - What is our development platform?

Summary

- Embedded computers are all around us.
- Chip designers are now system designers.
  - Must deal with hardware and software.
- Today's applications are complex.
  - Reference implementations must be optimized, extended.
- Platforms present challenges for:
  - Hardware designers---characterization, optimization.
  - Software designers---performance/power evaluation, debugging.
- Design methodologies help us manage the design process.