System design techniques

- Design methodologies.
- Requirements and specification.

Design methodologies

- Process for creating a system.
- Many systems are complex:
  - large specifications;
  - multiple designers;
  - interface to manufacturing.
- Proper processes improve:
  - quality;
  - cost of design and manufacture.

Product metrics

- Time-to-market:
  - beat competitors to market;
  - meet marketing window (back-to-school).
- Design cost.
- Manufacturing cost.
- Quality.

Mars Climate Observer

- Lost on Mars in September 1999.
- Requirements problem:
  - Requirements did not specify units.
  - Lockheed Martin used English; JPL wanted metric.
  - Not caught by manual inspections.

Design flow

- Design flow: sequence of steps in a design methodology.
- May be partially or fully automated.
  - Use tools to transform, verify design.
- Design flow is one component of methodology. Methodology also includes management organization, etc.

Waterfall model

- Early model for software development:
**Waterfall model steps**

- Requirements: determine basic characteristics.
- Architecture: decompose into basic modules.
- Coding: implement and integrate.
- Testing: exercise and uncover bugs.
- Maintenance: deploy, fix bugs, upgrade.

**Waterfall model critique**

- Only local feedback—may need iterations between coding and requirements, for example.
- Doesn’t integrate top-down and bottom-up design.
- Assumes hardware is given.

**Spiral model**

- System feasibility
- Specification
- Prototype
- Initial system
- Enhanced system
- Requirements
- Test

**Spiral model critique**

- Successive refinement of system.
  - Start with mock-ups, move through simple systems to full-scale systems.
  - Provides bottom-up feedback from previous stages.
  - Working through stages may take too much time.

**Successive refinement model**

- Specify architect
- Design
- Build
- Test
- Initial system

- Specify architect
- Design
- Build
- Test
- Refined system

**Hardware/software design flow**

- Requirements and specification
- Architecture
  - Hardware design
  - Software design
  - Integration
  - Testing
Co-design methodology

- Must architect hardware and software together:
  - provide sufficient resources;
  - avoid software bottlenecks.
- Can build pieces somewhat independently, but integration is major step.
- Also requires bottom-up feedback.

Hierarchical design flow

- Embedded systems must be designed across multiple levels of abstraction:
  - system architecture;
  - hardware and software systems;
  - hardware and software components.
- Often need design flows within design flows.

Hierarchical HW/SW flow

Concurrent engineering

- Large projects use many people from multiple disciplines.
- Work on several tasks at once to reduce design time.
- Feedback between tasks helps improve quality, reduce number of later design problems.

Concurrent engineering techniques

- Cross-functional teams.
- Concurrent product realization.
- Incremental information sharing.
- Integrated product management.
- Supplier involvement.
- Customer focus.

Requirements analysis

- Specification: precise description of what design team should deliver.
- Requirements phase links customers with designers.
Types of requirements

- **Functional**: input/output relationships.
- **Non-functional**:
  - timing;
  - power consumption;
  - manufacturing cost;
  - physical size;
  - time-to-market;
  - reliability.

Good requirements

- Correct.
- Unambiguous.
- Complete.
- Verifiable: is each requirement satisfied in the final system?
- Consistent: requirements do not contradict each other.

Good requirements, cont’d.

- Modifiable: can update requirements easily.
- Traceable:
  - know why each requirement exists;
  - go from source documents to requirements;
  - go from requirement to implementation;
  - back from implementation to requirement.

Setting requirements

- Customer interviews.
- Comparison with competitors.
- Sales feedback.
- Mock-ups, prototypes.
- Next-bench syndrome (HP): design a product for someone like you.

Specifications

- Capture functional and non-functional properties:
  - verify correctness of spec;
  - compare spec to implementation.
- Many specification styles:
  - control-oriented vs. data-oriented;
  - textual vs. graphical.
- UML is one specification/design language.