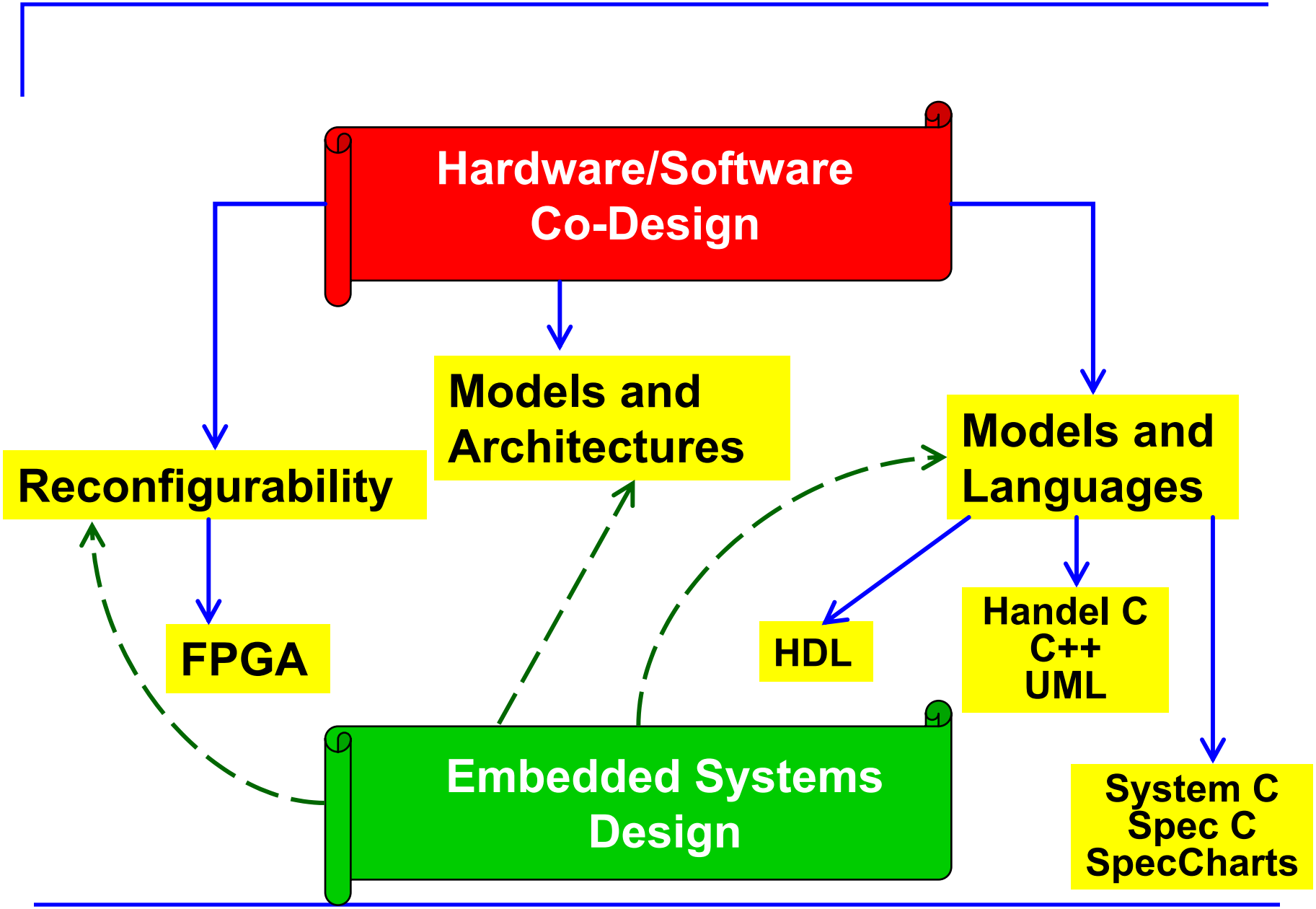
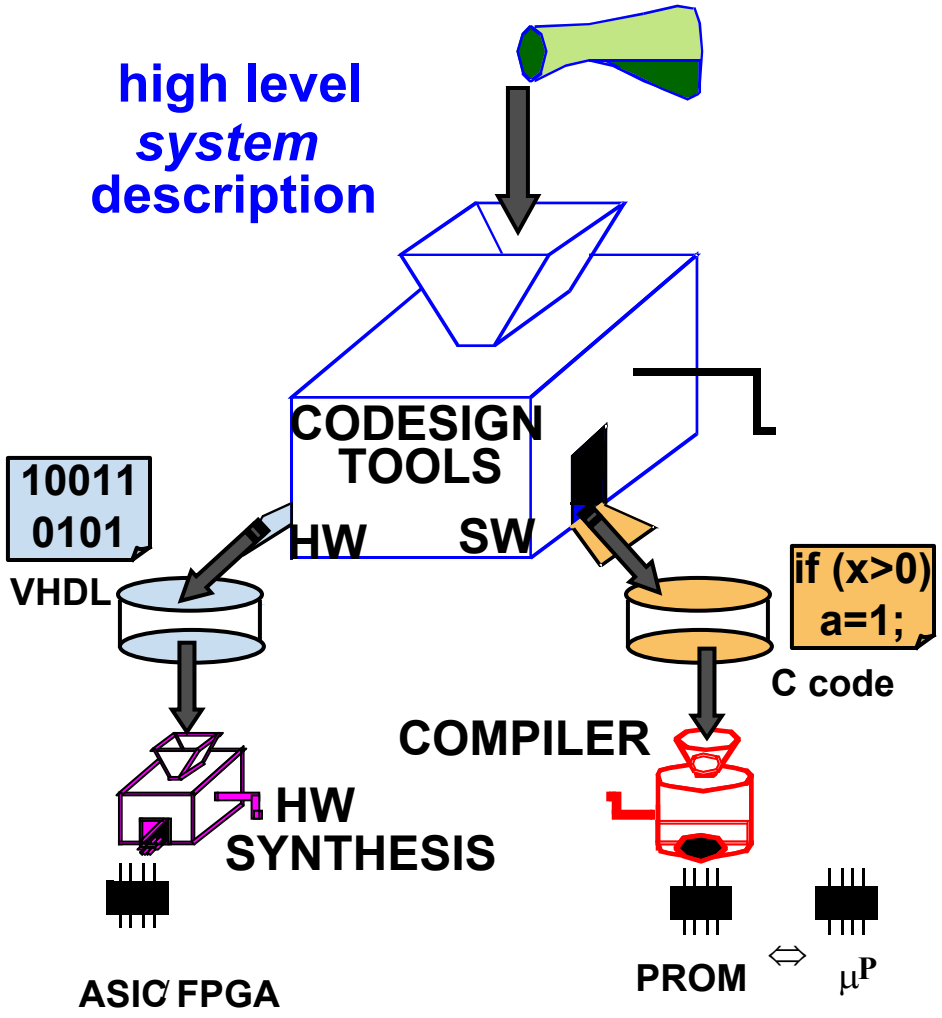

Hardware/Software
Co-Design
Models and
Architectures
Reconfigurability
Models and
Languages
FPGA
HDL
Handel C
C++
UML
Embedded Systems
Design
System C
Spec C
SpecCharts

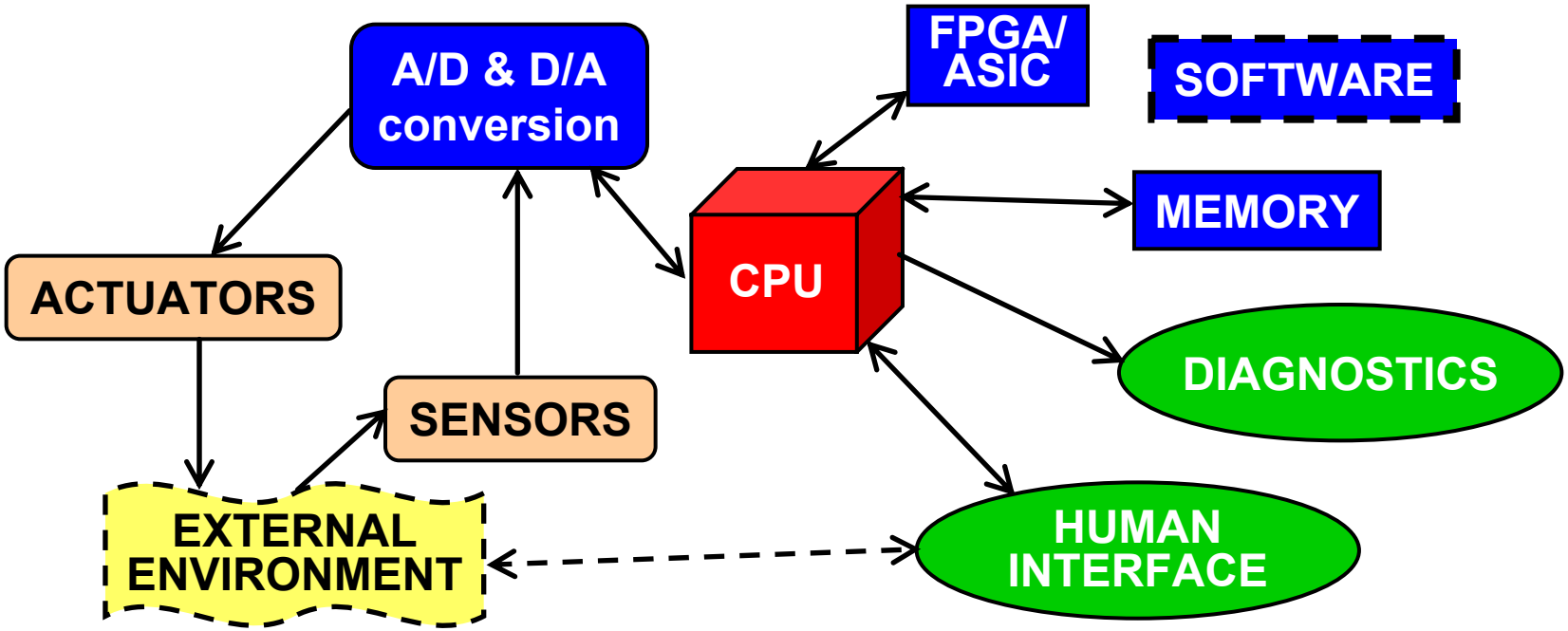
What is Hardware / Software Co-Design?



- Cooperative design of hardware and software components
- Unification of separate hardware and software paths
- Free movement of functionality between hardware and software during design exploration
- Attempt to utilize the "synergy" between hardware and software

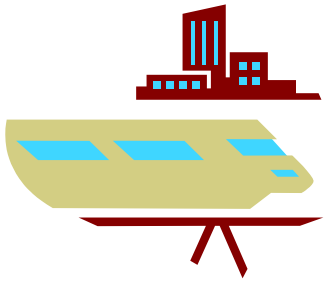
What is an Embedded System?

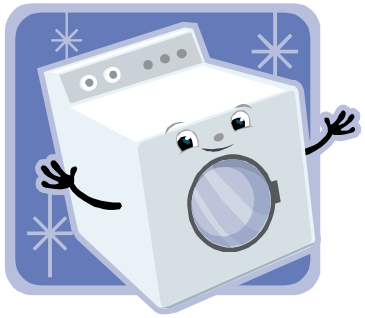
Any item including a programmable device, not intended to be general purpose, with many interface units to measure, control, manipulate the external environment

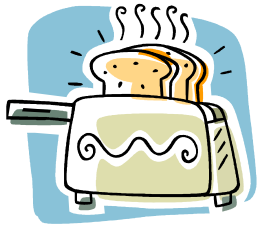


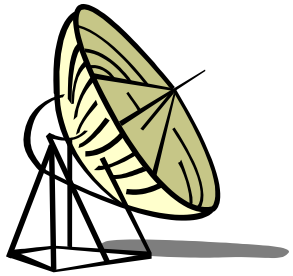
Applications

- Very diverse applications
- Very diverse domains
- What are the common characteristics?



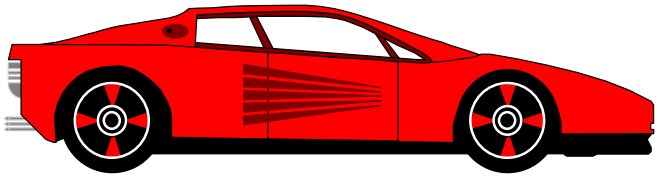






Embedded (Mechatronics) Systems

- Example: car transmission

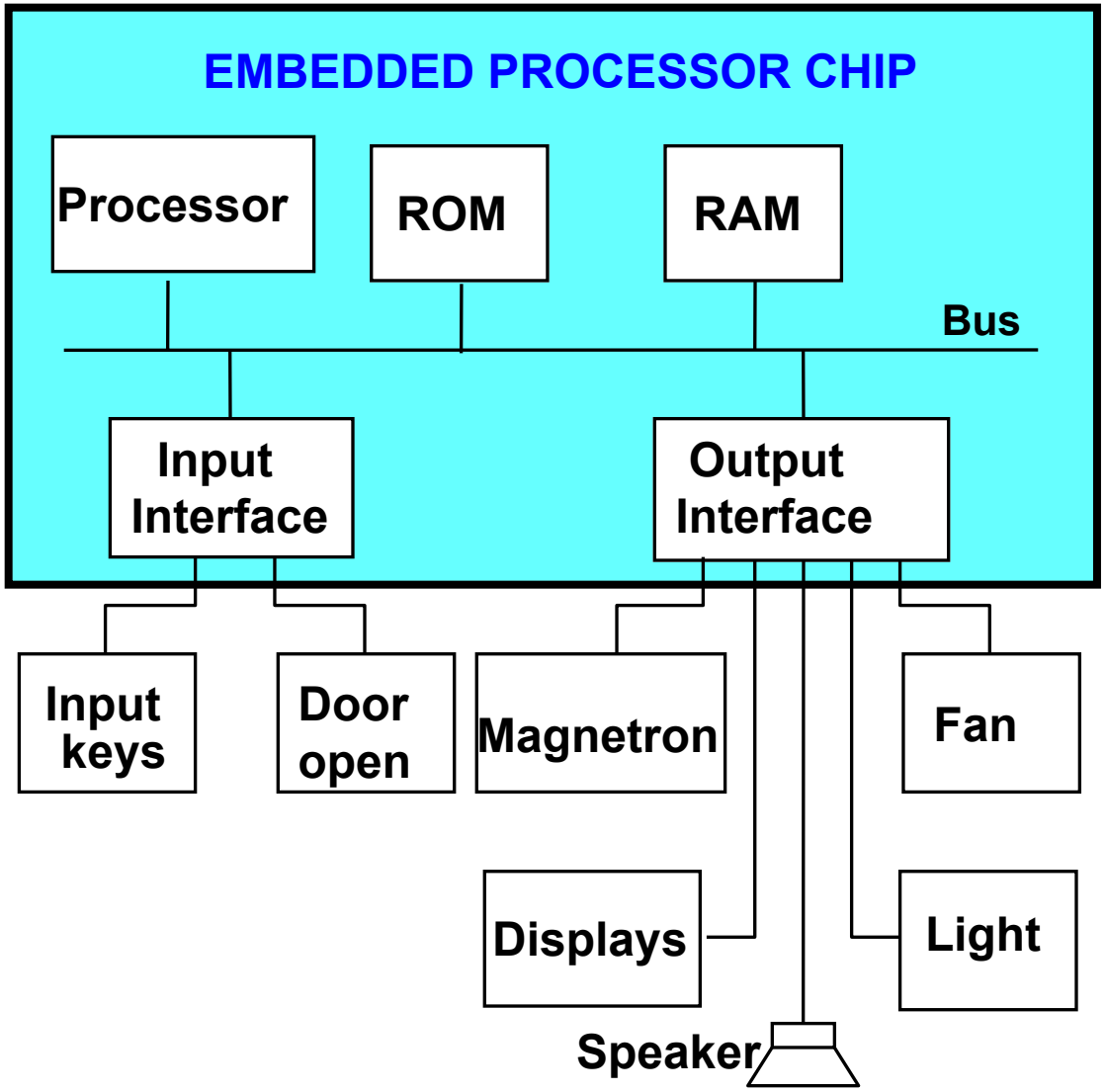


- Microprocessor observing and controlling a car's automatic transmission
  - *observe* engine rpm, manifold pressure
  - *control* shifter, engine throttle
    - real time & fault tolerance requirements in a closed-loop control
    - interface with a larger mechanical system ("embedded")
    - software requirement

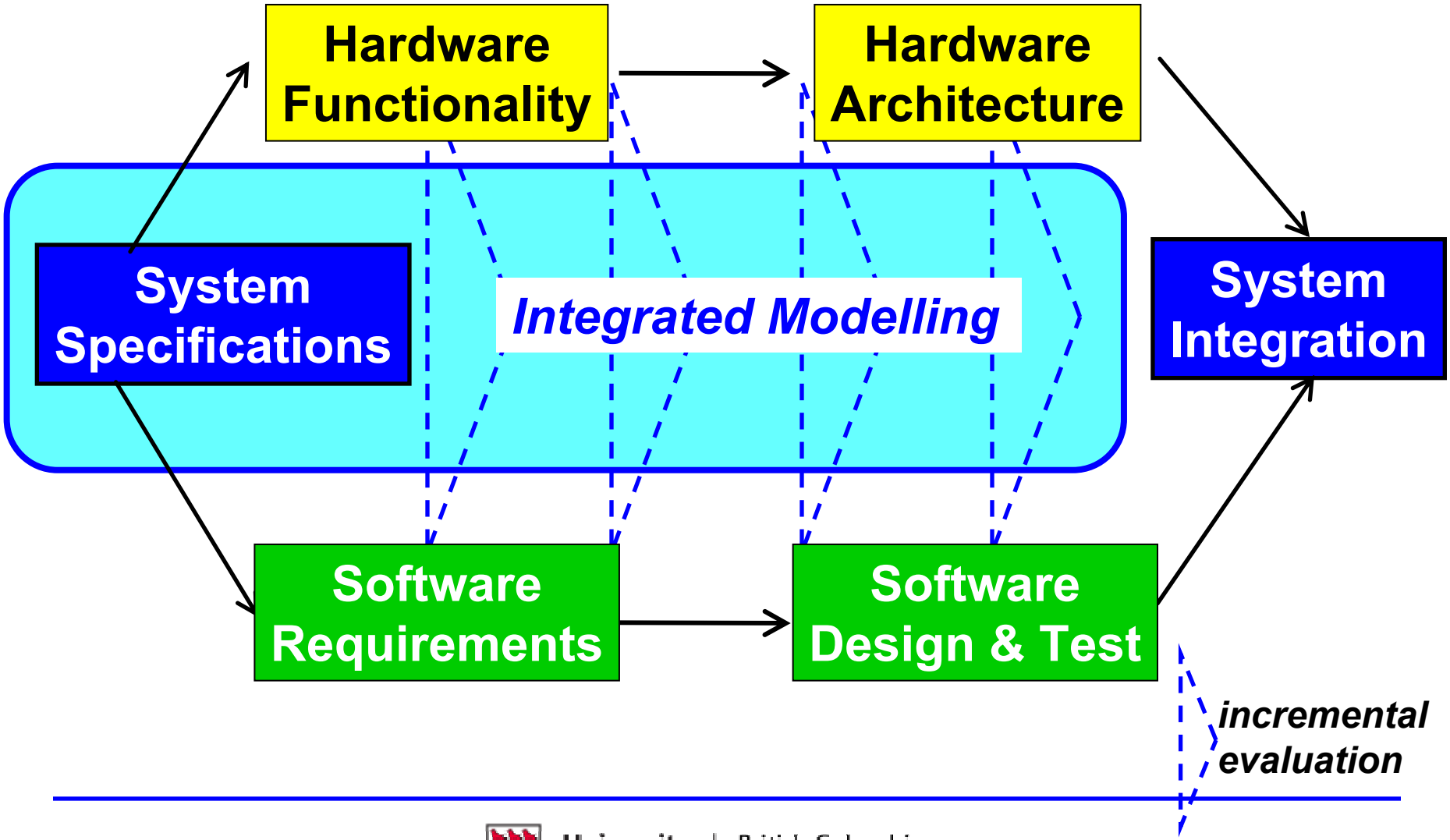
Design issues

Cost	
I/O capability	many options
Size	small 8-bit often acceptable
Power consumption	Portable or not
On-chip memory	small RAM, large ROM
Performance	not critical (except video games)
Software	migrating to high level languages constrained by RAM size
Development tools	critical new wave?
Reliability	portability and remote use
Robustness	

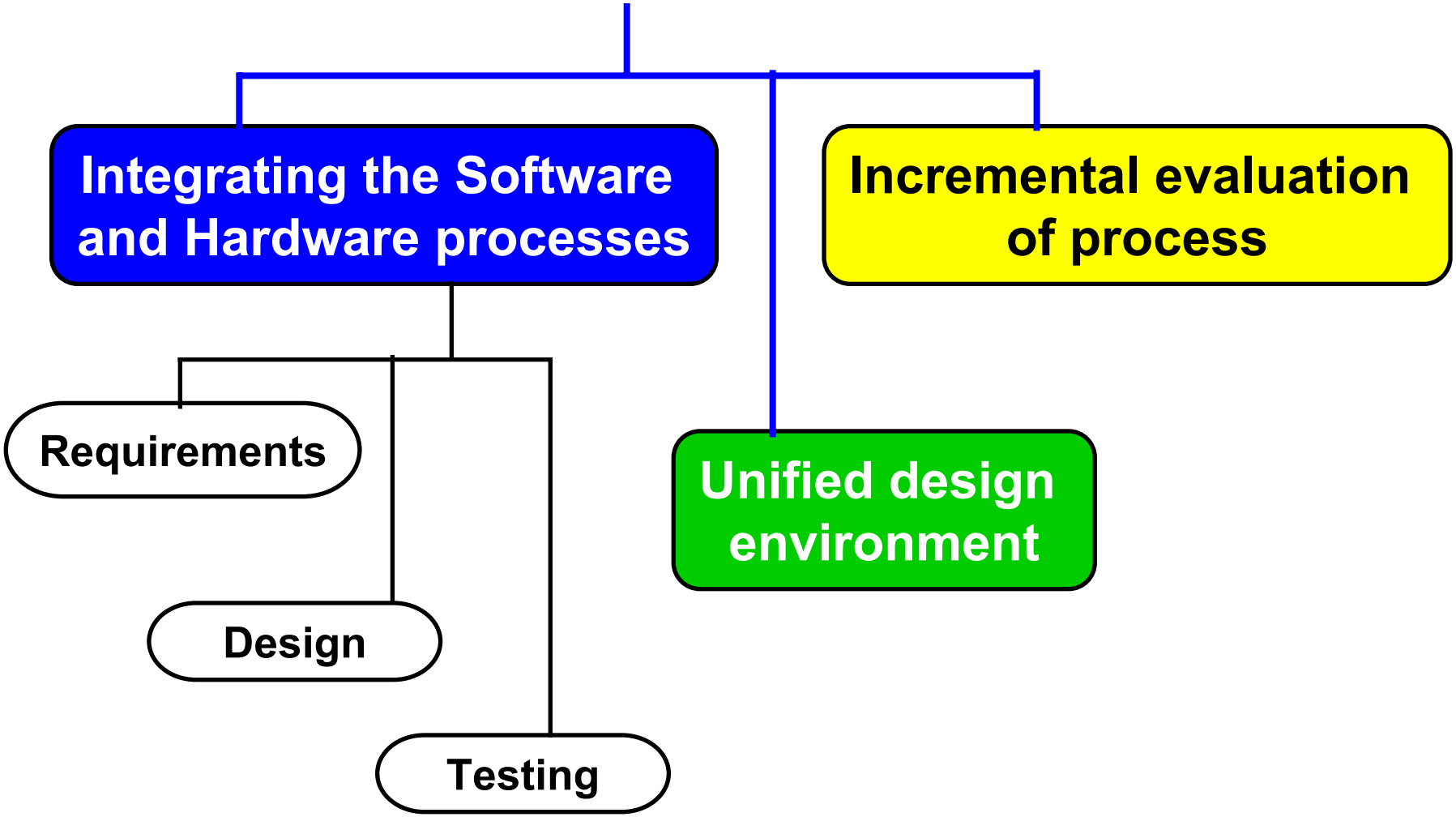
A block diagram of a microwave oven



System Hardware/Software Co-Design



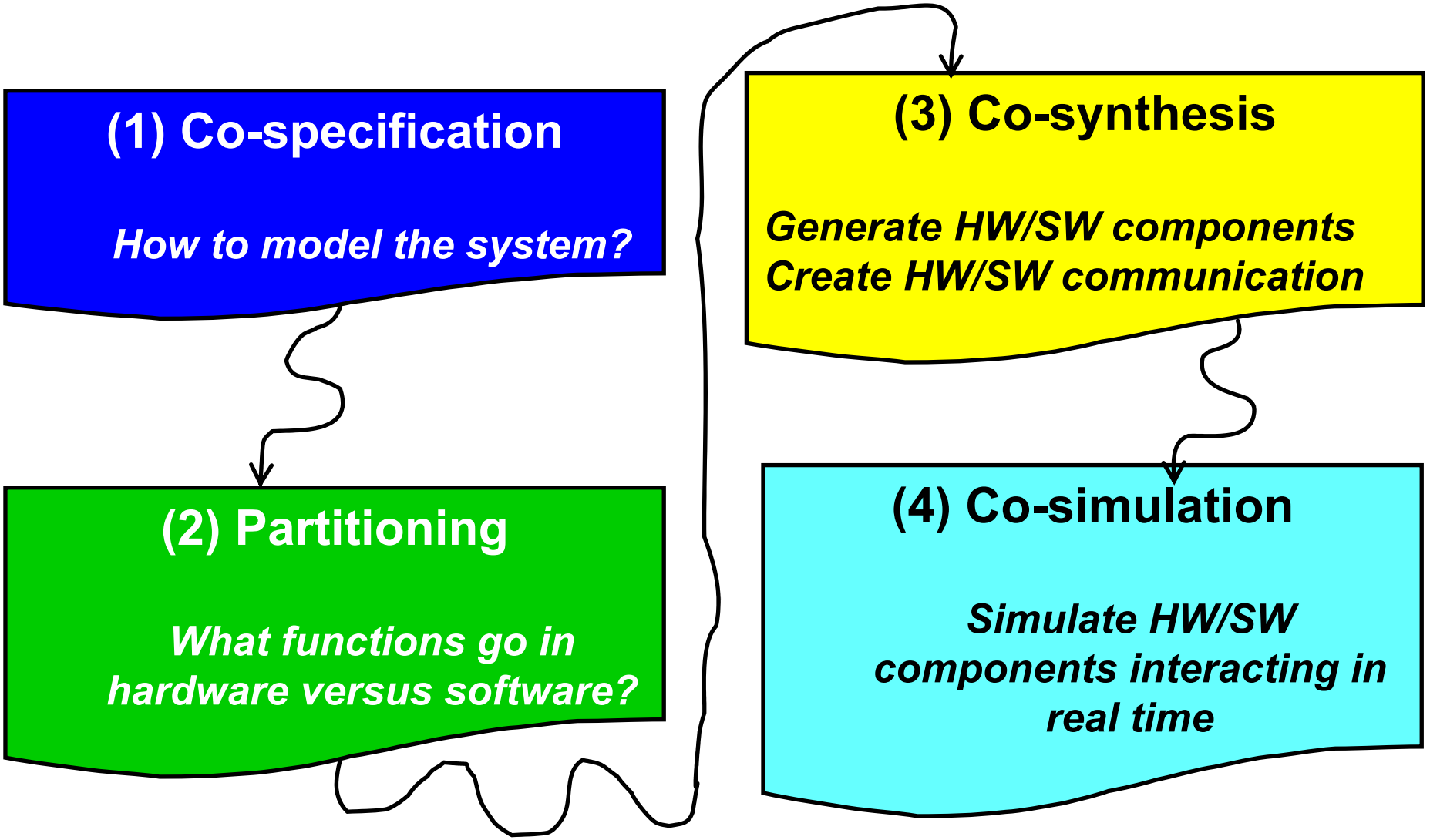
What are the Goals of Co-Design?



What are the Challenges in Co-Design?

- HW/SW split decided early, on ad hoc basis
- Ability to explore HW/SW trade-offs is restricted
- Different cultures hamper integration
- Cost increases and schedule over-runs

What are the Co-Design Research Areas?



Modelling

- **What is the best methodology for specifying HW/SW system?**
  - **a high-level functional / algorithmic spec?**
  - **HW-type language (VHDL, HW-C) or SW-type (C, C++)?**
  - **formal spec language? (provably correct)**
  - **same/different methods for HW vs. SW specs? (avoid too early binding)**

- **Internal Representation (Co-modelling)**

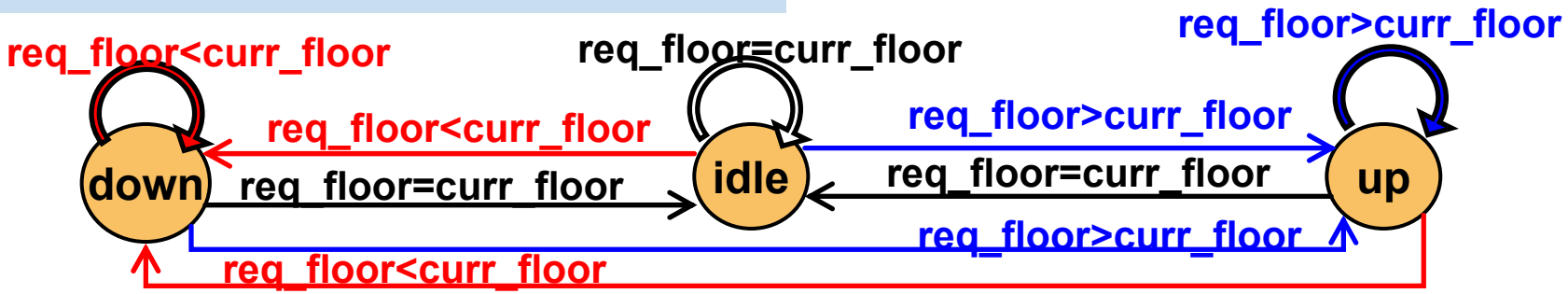
Elevator Controller

"If the elevator is stationary and the floor requested is equal to the current floor, then the elevator remains idle.

If the elevator is stationary and the floor requested is less than the current floor, then lower the elevator to the requested floor.

If the elevator is stationary and the floor requested is greater than the current floor, then raise the elevator to the requested floor"

```
loop
    if (req_floor = curr_floor) then
        direction := idle;
    elseif (req_floor < curr_floor) then
        direction := down;
    elseif (req_floor > curr_floor) then
        direction := up;
    end if;
end loop;
```



Partitioning

- **How to divide specified functions between HW & SW?**
  - based on functional spec + constraints (timing, cost, size, power)
  - primarily *manual* so far; want to automate
  - generic specs, use heuristics or interactive guidance (with quick evaluation) to divide
- **Approaches**
  - start with all HW, extract functions to move to SW, stop when performance constraints are violated
  - start with all SW, extract performance critical portions
- **Criteria**
  - For system tasks:
    - Does task interact closely with the O.S? ==> S/W
    - Does task interact closely with external signals? ==> H/W
  - For remaining tasks:
    - cost and speed.

- ***Efficiency of a system is directly related to how the functionality in H/W or S/W is allocated***

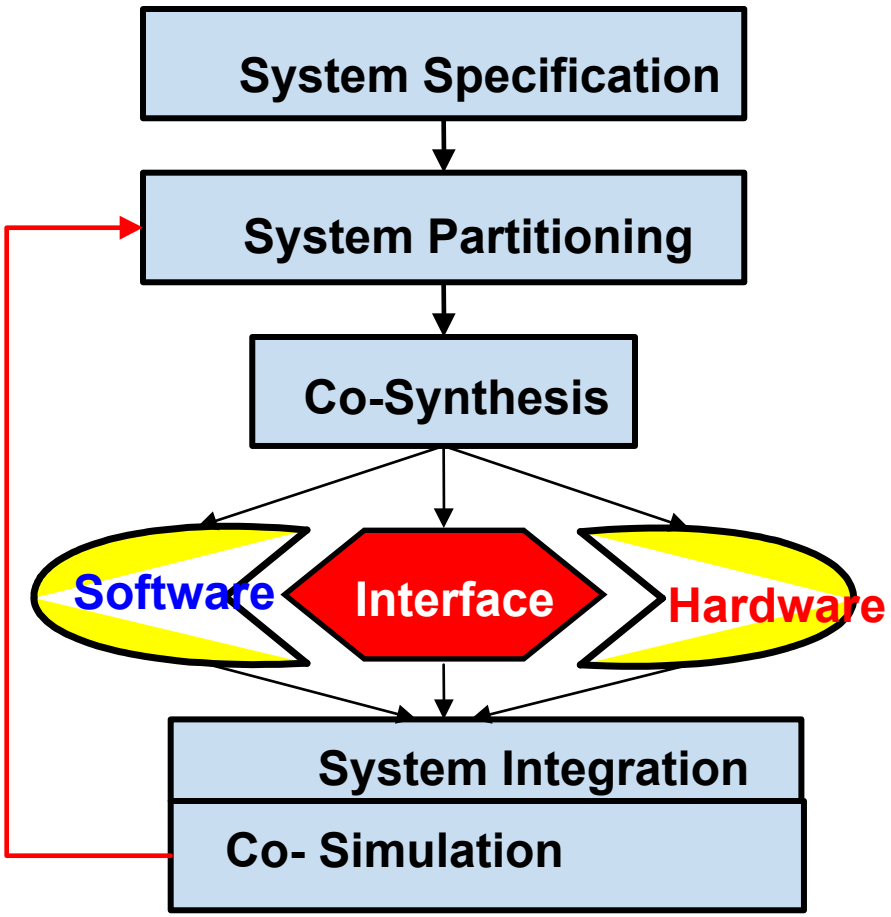
Co-synthesis and Co-simulation

- **Generating the HW & SW components**
  - HW synthesis built on existing CAD tools
  - generate HW / SW interprocess communication
  - schedule SW processes to meet timing constraints (static vs. dynamic)

- **H/W \ S/W Integration**
  - SW and HW operate as separate processes and communicate through suitable IPC (inter-process communication) mechanisms (e.g. Unix sockets)

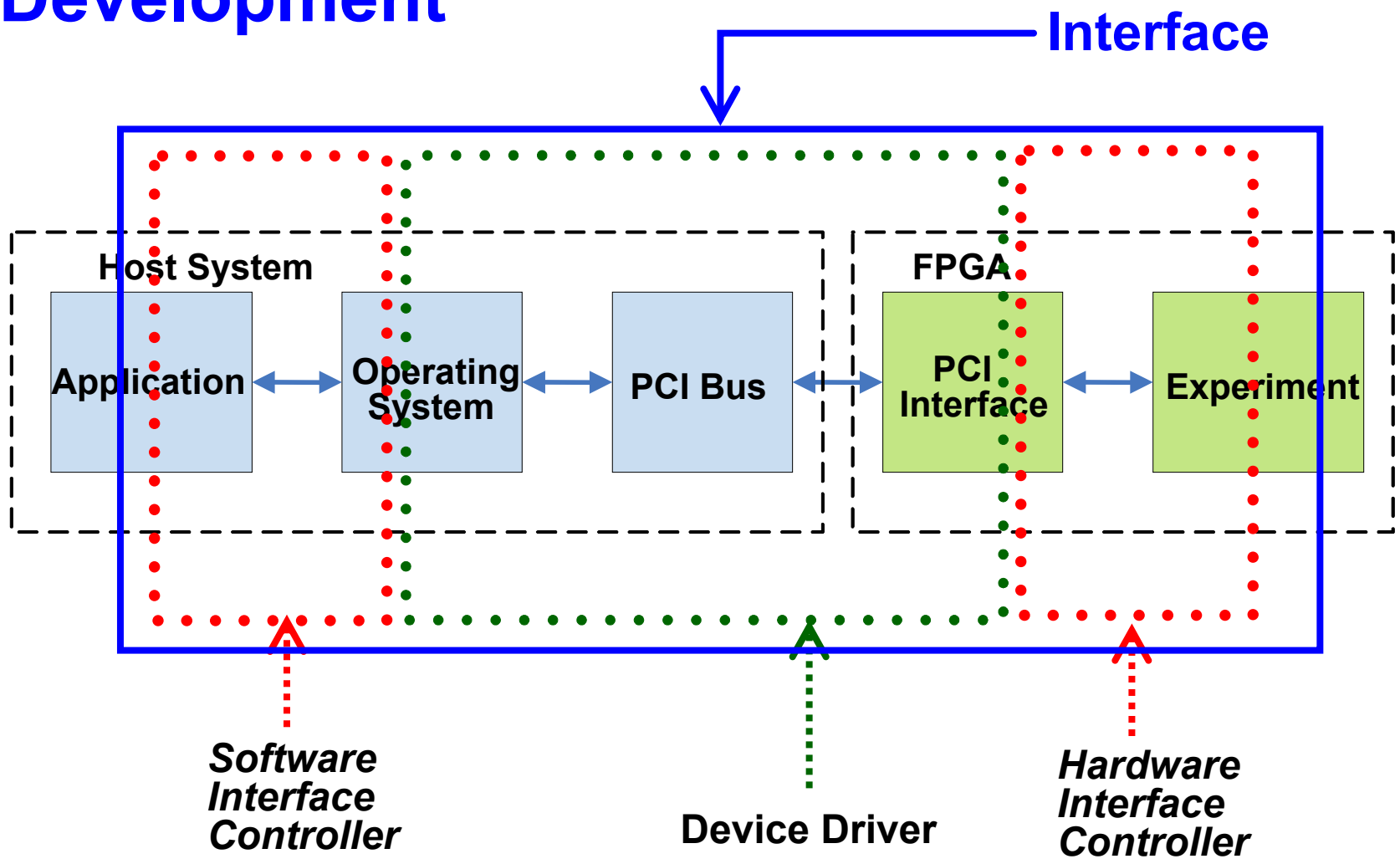
- **Co-simulation: evaluating the synthesized design**
  - simulate SW & HW components interacting in real time
  - for design / constraint verification, performance estimation
  - for feedback to partitioning step

Hardware/Software Co-Design Process

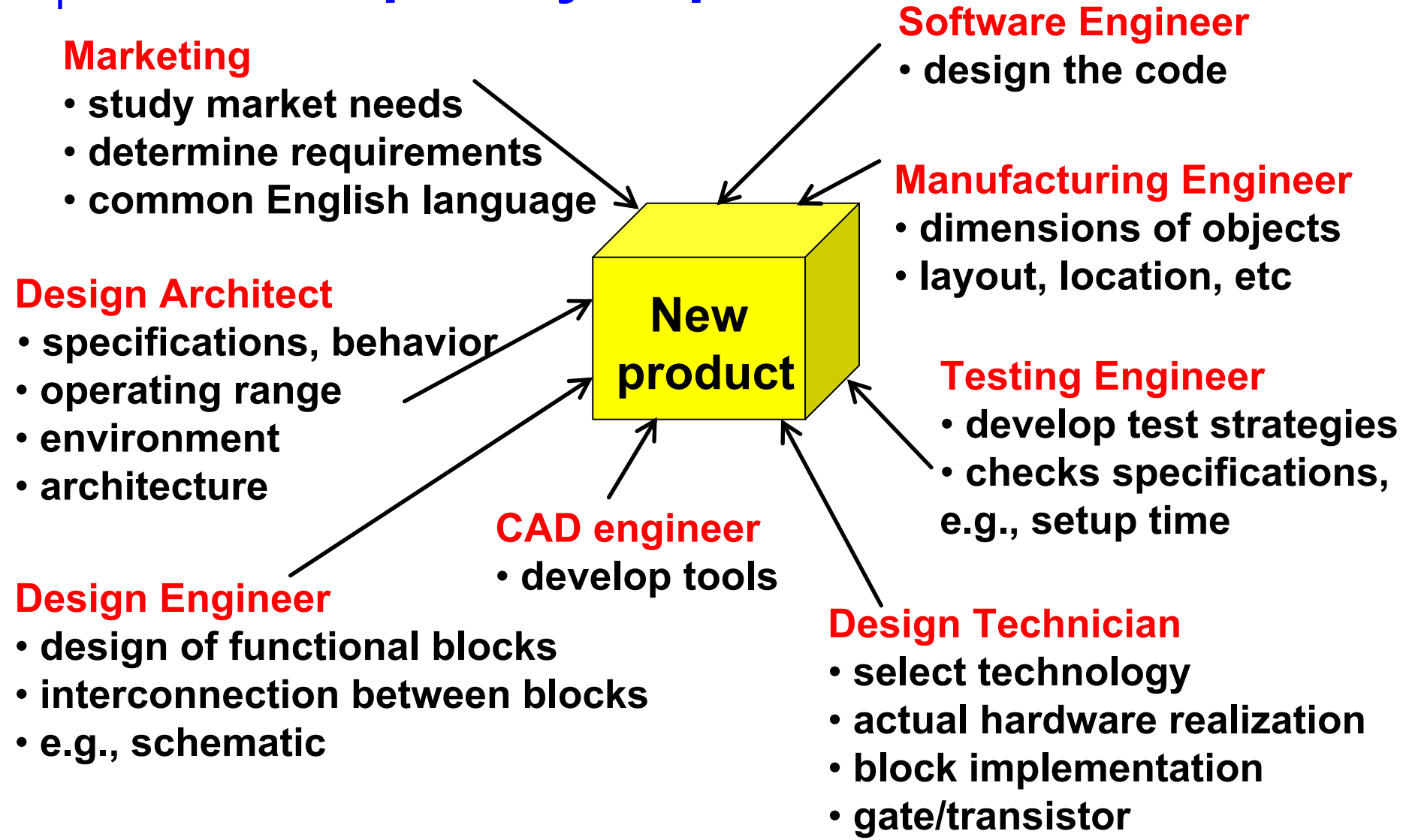


- **Feedback from co-simulation to partitioning**
- **Interface design**
  As partition changes so must the interface between hardware and software
- **Implicit in the process is a unified system representation that can move to a hardware, software, and interface representation**

What is the hardest task? Interface Development



Multidisciplinary Topic



Some Observations on Embedded Systems

- Embedded system's behavior is defined by its interaction with its environment
- CPU is only a specialized part of the system
- The functionality provided is specific to the application
- One does not design an embedded computer ➔ one designs an embedded system
- Often there is dedicated software / OS (customizable)
- Often it replaces electromechanical component
- Possibly no keyboard
- Interdisciplinary design area

System Types

❖ *Data-dominated*

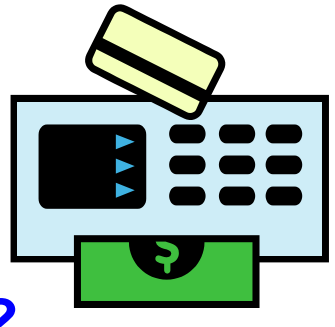
- ➢ Transform input stream into output stream
- ➢ Heavy digital signal processing
- ➢ Often modeled using dataflow diagrams

➔ Cell Phone

❖ *Control-dominated*

- ➢ Reactive real-time stimulus/response
- ➢ Often modeled with state transition diagrams

➔Automatic Teller Machine (ATM)



But how do we describe these systems?

Typical Characteristics of Embedded Systems - *Functionalities*

- **State Transitions**
  - Intrinsically state-based
  - Constant transitions from one mode/state to another
- **Behavioral Hierarchy**
  - Functionality is more easily conceptualized as behavior
- **Behavioral Completion**
  - Finite activities may not terminate solely by external events but continue until behavior completion
  - May need to initiate a new activity at time of completion
- **Concurrency**
  - Both Task-level and Statement-level concurrency
- **Exceptions**
  - Certain events require an instantaneous response
  - Interrupt handling is crucial
- **Complex algorithms**
  - Usually with complex programming constructs
- **Complex Users Interfaces**
  - Multiple menus & options (e.g. navigational maps for GPS)

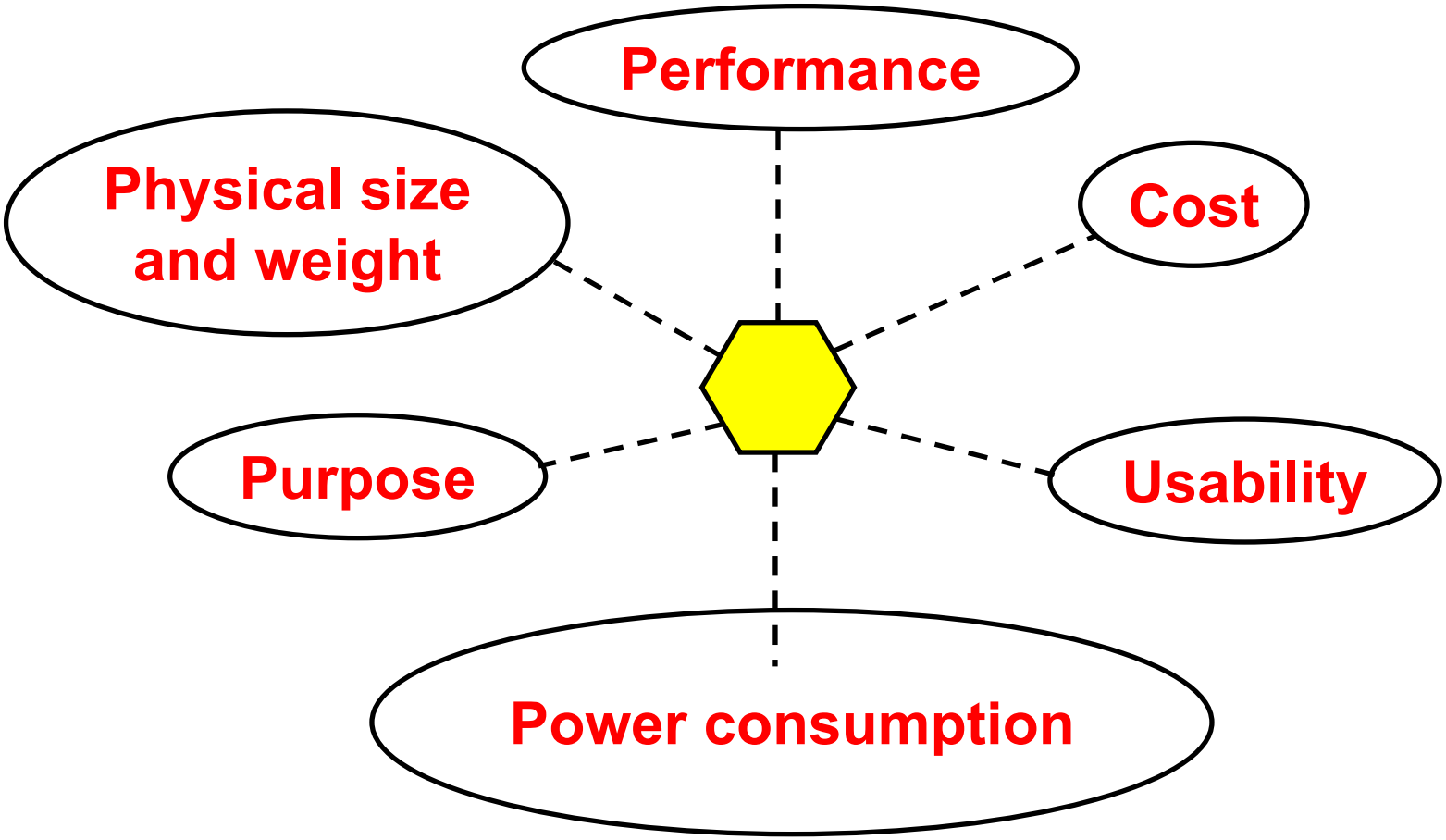
Typical Characteristics of Embedded Systems – *Timing and Costs*

- **Real Time**
  - Correctness of computation depends on the time it is delivered
  - Failure to meet deadlines might create safety issues or simply unhappy customers
  - Mostly reactive computation – it executes as a response to external events
- **Multirate**
  - Several real time activities at the same time (e.g. audio and video portions in multimedia)
- **Manufacturing Costs**
  - Crucial for marketing
  - Microprocessor, flash memory, I/O, sensors
- ***Power***
  - ***Battery life!***
  - ***Heat dissipation!***
  - ***Peaks of use!***

Typical Embedded Systems Functional Requirements

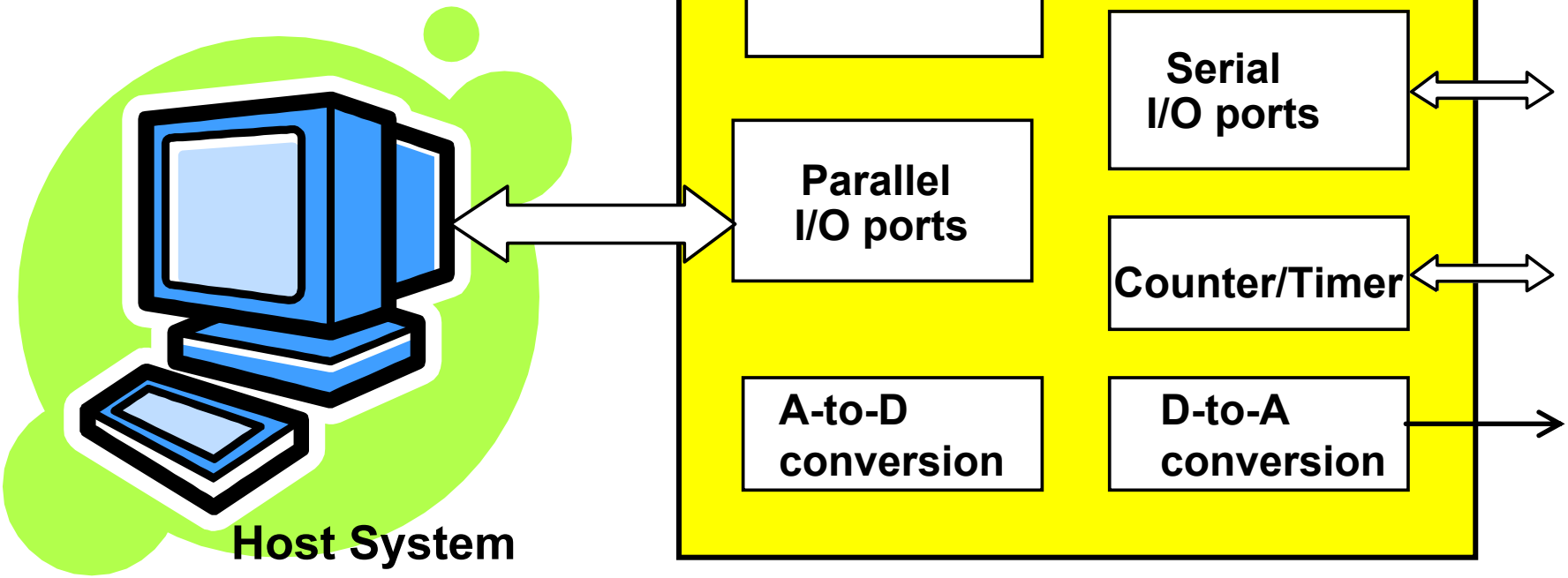
- **Control laws** (e.g. PID control, fuzzy control)
- **Sequencing control** (e.g. FSMs, mode changes, switching between control laws)
- **Signal processing** (e.g. voice, video)
- **Fault response** (e.g. detection, reconfiguration)
- **Application-specific user interface device** (e.g. buttons, bells, lights)

Typical Embedded Systems Non-functional Requirements

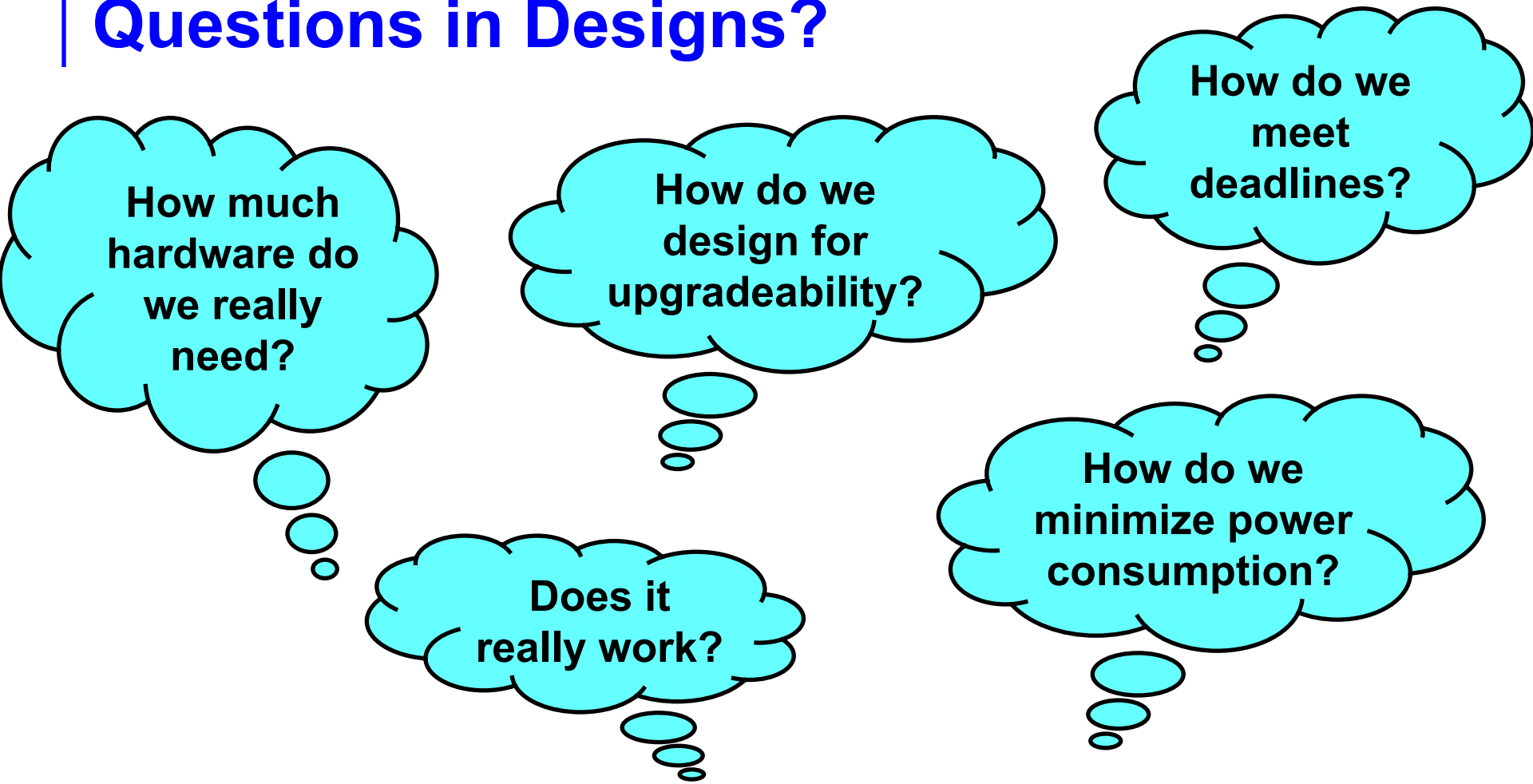


Development and Debugging

- ✓cross-compilers
- ✓testbench hardware
- ✓testbench software



Questions in Designs?

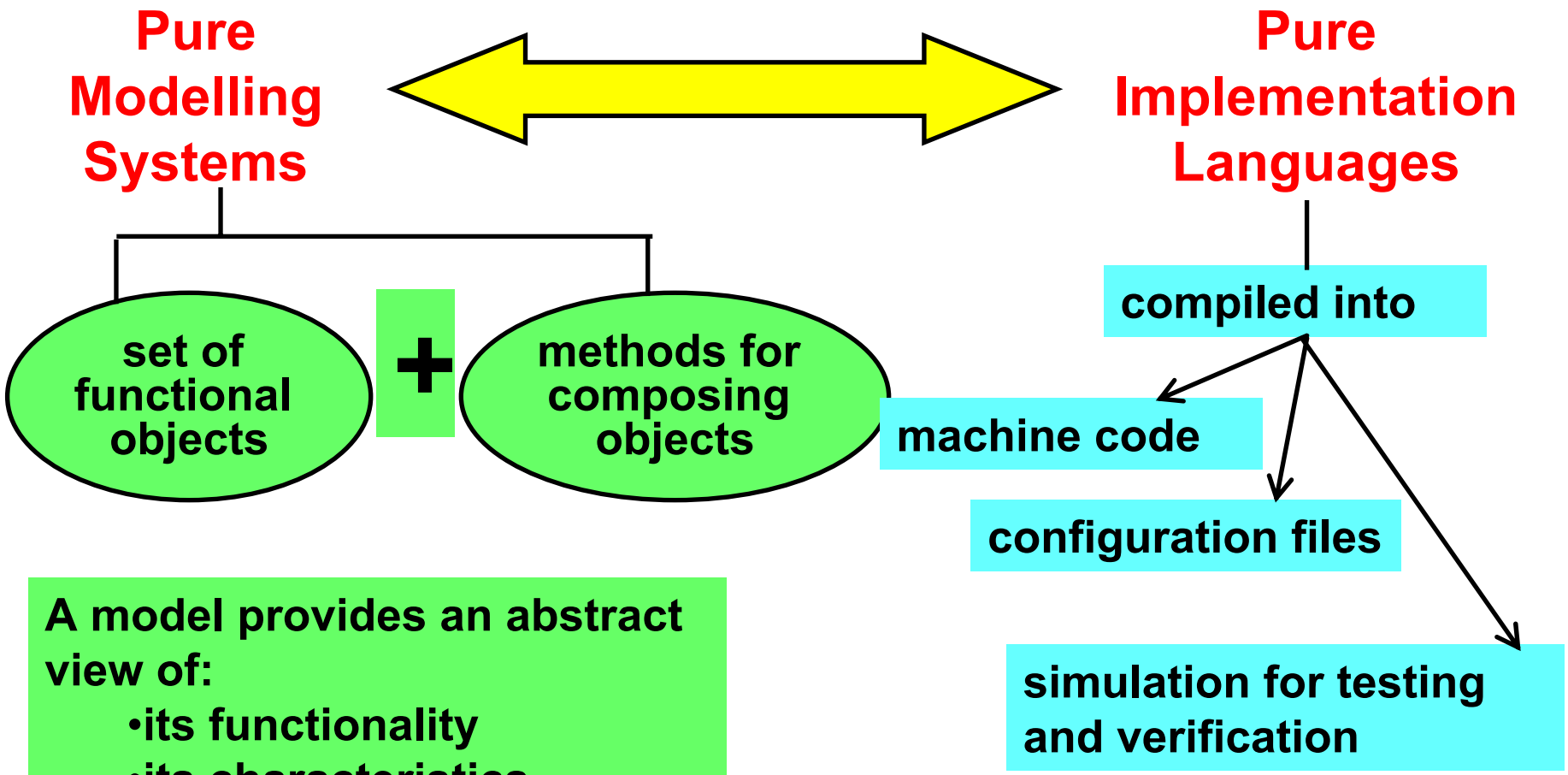


Limited observability and controllability

Complex testing

Restricted development environment

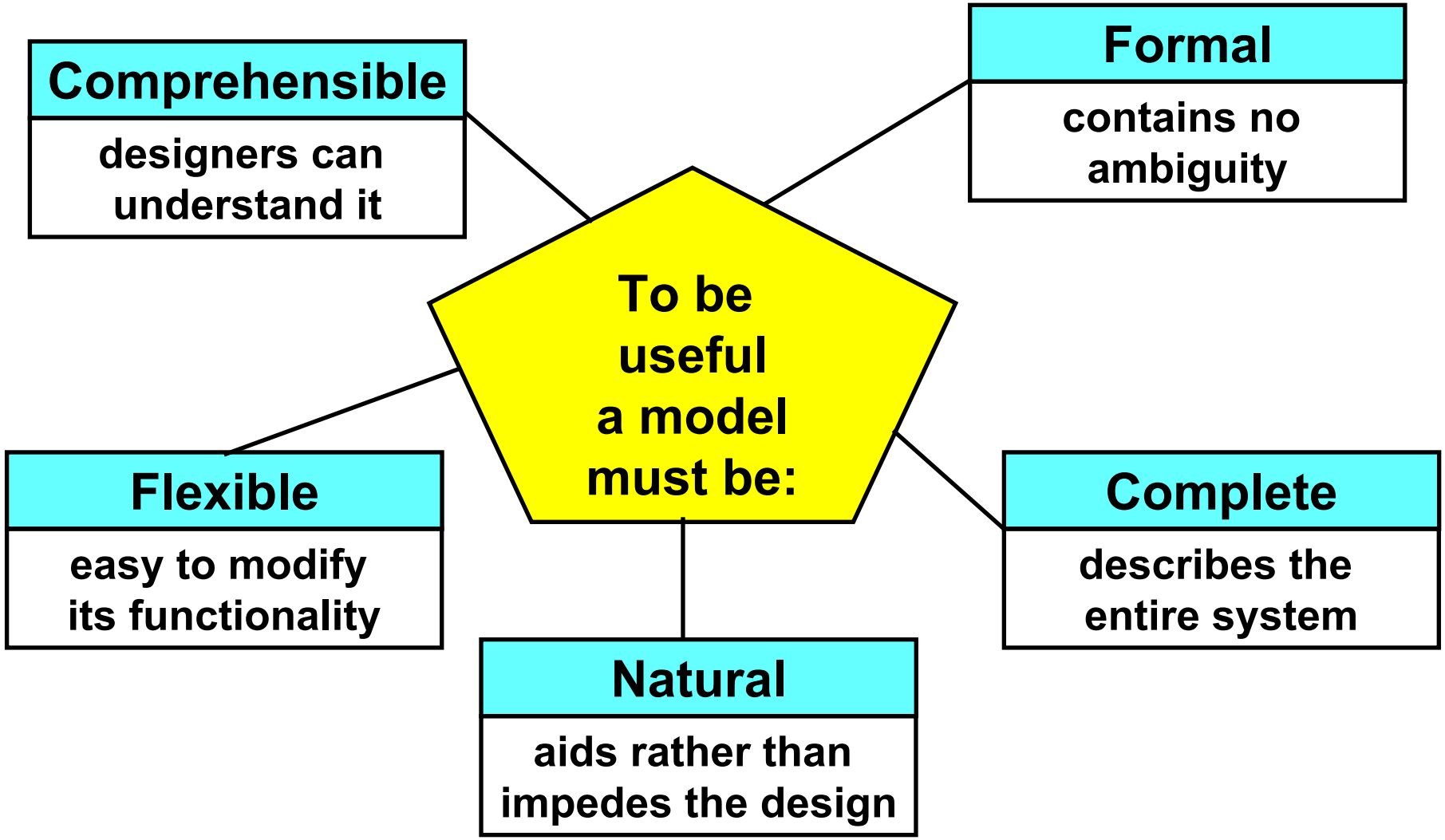
Models & Languages



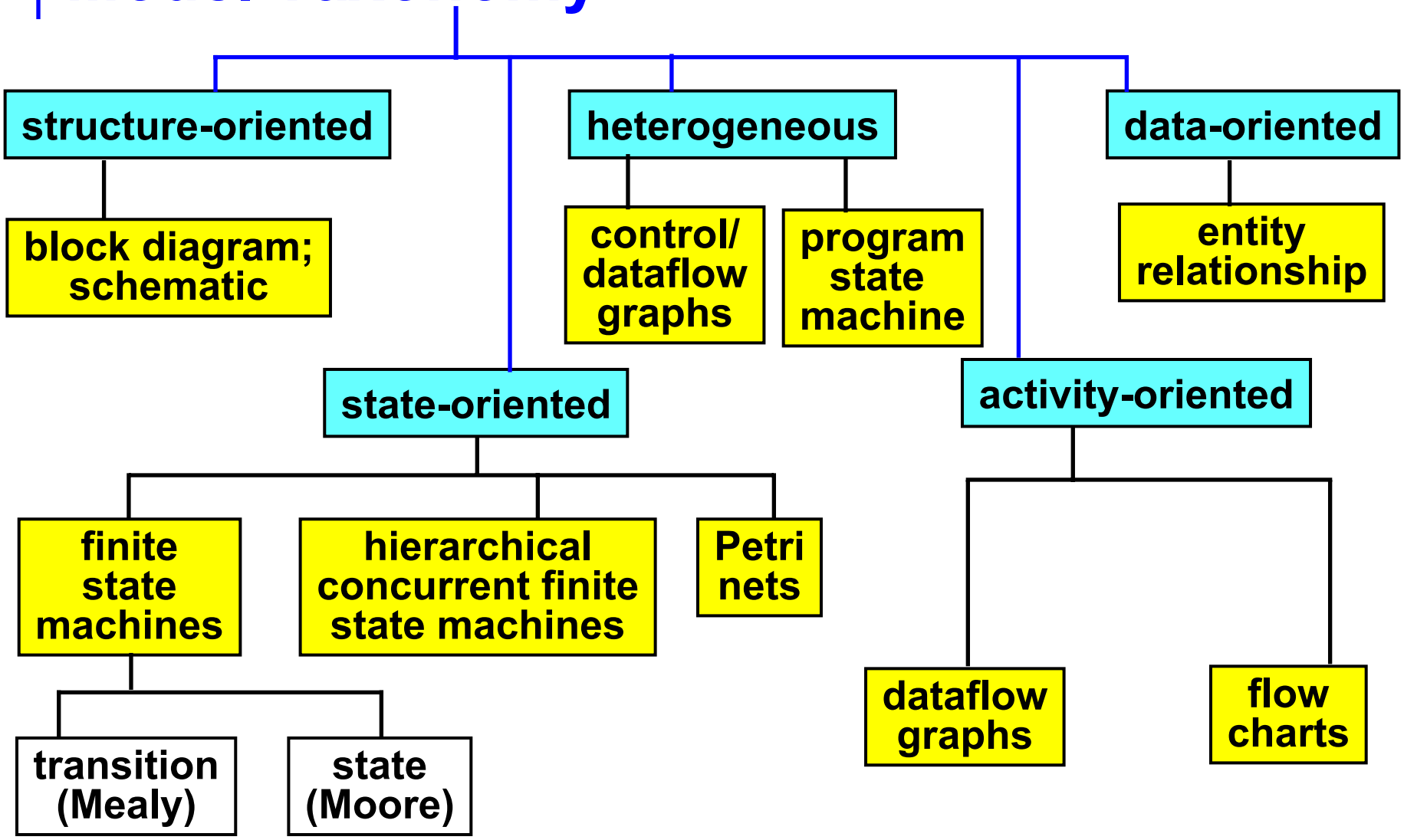
A model provides an abstract view of:

- its functionality
- its characteristics
- how the system works

Characteristics of Models



Model Taxonomy



Acknowledgements

- *"Computers as Components"*

  by W. Wolf, Morgan Kaufmann Pub.

- *"Specification and Design of Embedded Systems"*

  by D. Gajski, F. Vahid, S. Narayan, J. Gong, Prentice Hall.

- *"Embedded System Design: A Unified Hardware/Software Introduction"*

  by F. Vahid and T. Givargis, John Wiley & Sons.

- *"Embedded System Design Issues (the Rest of the Story)"*

  by P. Koopman, CMU