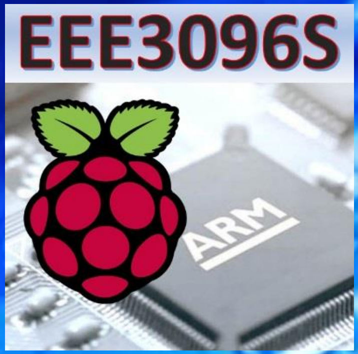


Challenges in embedded systems & Design Lifecycle

Lecture

Embedded Systems II

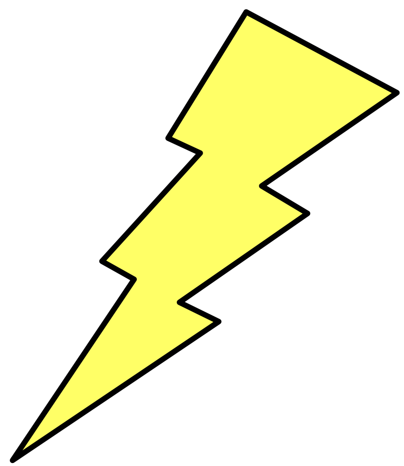
Dr Simon Winberg

Electrical Engineering
University of Cape Town

Outline

- Blitz Quiz – for fun not for marks
- Characteristics of Embedded Systems
- Design Flows
- The 7-phase embedded system design cycle

Quick Quiz



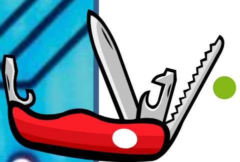
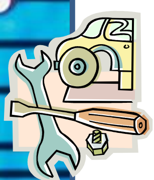
Time almost up!

Time up!

Marking…

Characteristics of Embedded Systems

Characteristic 1 : Dependability

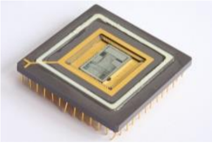
- CPS/ES must be **dependable**, this can be described as metrics of:
  
  - **Reliability $R(t)$ =** probability of system working correctly provided that is was working at time t=0
  
  - **Maintainability $M(d)$ =** probability of system working correctly d time units after error occurred.
  
  - **Availability $A(t)$**: probability of system working at a particular time t
  
  - **Safety**: no harm to be caused by the system
  
  - **Security**: confidential and authentic communication

Characteristic 1 : Dependability

- Even perfectly designed systems can fail if the assumptions about the workload and possible errors turn out to be wrong.
- Making the system dependable must not be an after-thought, it must be considered from the very beginning!

Characteristic 2 : Efficiency

- CPS & ES must be efficient
- Code-size efficient
  (especially for systems on a chip)



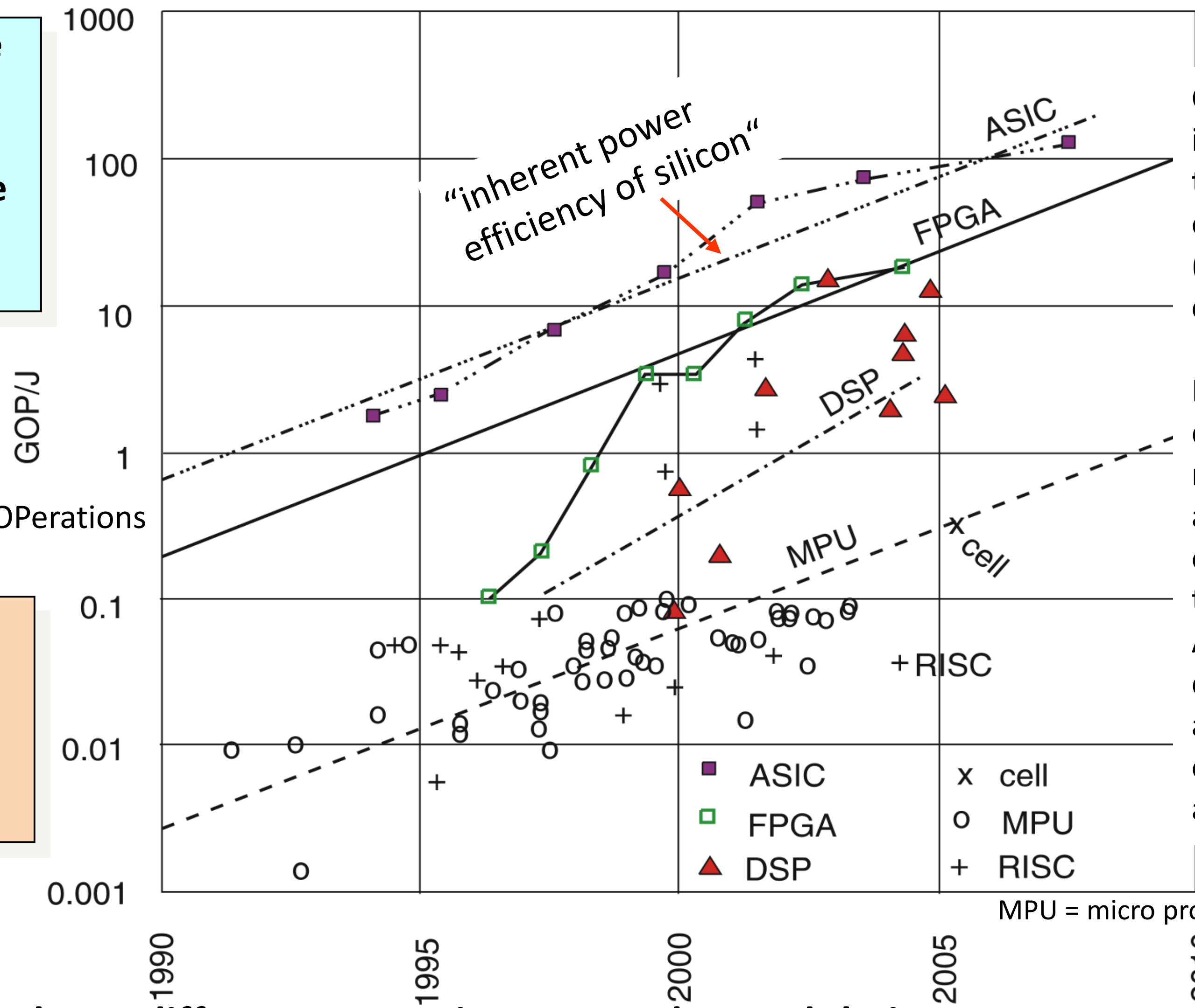
- Run-time efficient

- Weight efficient
- Cost efficient

- Energy efficient

Importance of Energy Efficiency

Efficient software design needed, otherwise, the price for software flexibility cannot be paid.



GOP = Giga OPerations

MPU = micro processors (eg. x86)

Clearly an ASIC is going to give the best efficiency rates (i.e. fully customized)

BUT this diagram does not show anything about engineering time and costs. ASIC development is also the most expensive approach

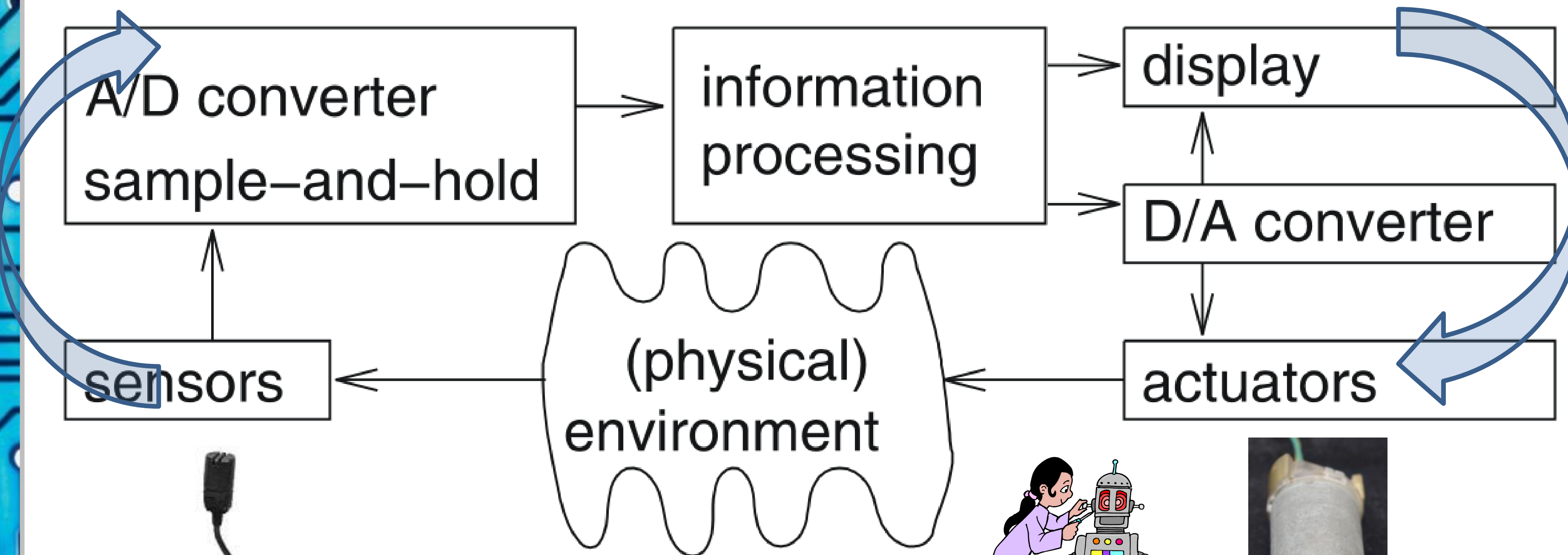
Interesting to see the variety of energy efficiency here, e,g, DSPs as a tradeoff

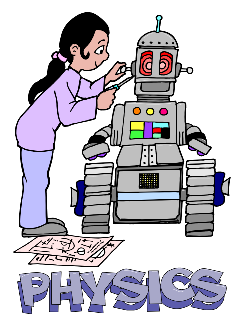
This chart shows different processing approaches and their energy cost for operations per Joules. NB: Logarithmic scale!

CPS & ES Hardware

CPS & ES hardware is frequently used in a loop…

“hardware in a loop” model:



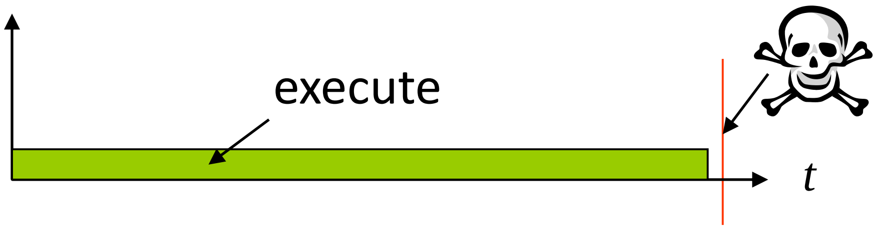


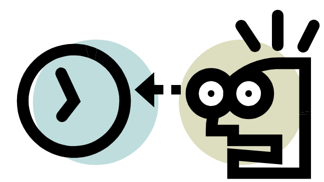
Cyber-physical system (!)

Real-time constraints

CPS must meet **real-time constraints**

- A real-time system must react to stimuli from the controlled object (or the operator) within the time interval **dictated** by the environment.





- **“A real-time constraint is called hard, if not meeting that constraint could result in a catastrophe“** [Kopetz, 1997].
- All other time-constraints are called **soft**.
- A guaranteed system response has to be explained without statistical arguments [Kopetz, 1997].

Real-Time Systems & CPS

- *Are*
  CPS, ES and Real-Time Systems synonymous?
  - For some embedded systems, real-time behavior is less important (smart phones)
  - For CPS, real-time behavior is essential, hence RTS $\cong$ CPS
  - CPS models also include a model of the physical system

Reactive & hybrid systems

– Typically, CPS are **reactive systems**:
“A reactive system is one which is in continual interaction with is environment and executes at a pace determined by that environment“
[Bergé, 1995]

Behavior depends on **input and current state**.
Therefore…
☞ automata models are appropriate;
✘ modeling just computable functions inappropriate

– **Hybrid systems =**
analog + digital parts


Dedicated systems

- These are
  - **Dedicated** towards a certain **application**. Knowledge about behavior at design time can be used to minimize resources and to maximize robustness
  - **Dedicated user interface** (no mouse, keyboard and screen)
  - Situation is slowly changing here: systems become less dedicated

Security

- Defending against
  - Cyber crime
    - Annual U.S. Cybercrime Costs Estimated at $100 Billion! [Wall Street Journal, 22.7.2013]
  - Cyber attacks
  - Cyber terrorism
  - Cyber war (Cyber-Pearl-Harbor [Spiegel Online, 13.5.2013])
- Connectivity increases threats
  - Entire production chains can be affected
  - Local islands provide some encapsulation, but contradict idea of global connectedness

Dynamics

All about:

Frequent change of the environment

Supporting a dynamic system can involve significantly more design effort

Deciding if it is an embedded system

- Not every CPS & ES has all of the above characteristics.
- **Deciding if it is an ES**:
  Information processing systems having most of the above characteristics are generally called embedded systems.
  (even thought the embedded computer system might not be small and low power)

Characteristics lead to corresponding challenges

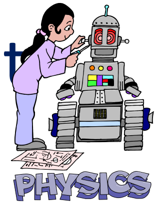
(some of the main challenges…)

– Dependability

– Efficiency

- In particular: Energy efficiency

– Hardware properties, physical environment



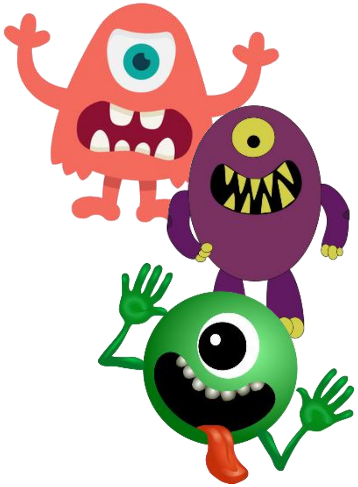
– Meeting real time requirements

– ….

Challenges & the 'ilities'

1. Safety
2. Security
3. Confidentiality
4. Reliability
5. Repairability
6. Availability

Sir Ility of the embedded realm



The Monster challenges of ES

Resource Aware Embedded Systems

- There are further challenges, in terms of mobile ES and IoT applications, such as
  - Energy / Power* use
    - "Power is considered as the most important constraint in embedded systems" [1]
  - Run-time (wasted processor cycles should be avoided)
  - Code size (especially true for systems on a chip (SoCs))
  - Weight*
  - Cost (especially for high-volume)
  - Size* (physical dimensions of the embedded computer)

"What is your SWAP?"… *i.e. your Size, Weight And Power*

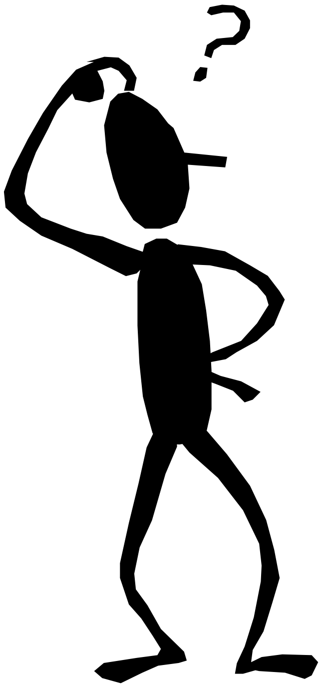
[1] Eggermont, L.: Embedded systems roadmap. STW. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.6407&rep=rep1&type=pdf (2002)

Challenges for implementation in hardware

- Early embedded systems were frequently implemented in hardware (custom boards)
- Mask cost for specialized application specific integrated circuits (ASICs) becomes very expensive (M$ range, technology-dependent)
- ASICS / custom circuits lack flexibility (e.g. to adjust for changing standards)
- The trend is towards more implementation in software (or possibly FPGAs, to explore later)

Challenges for implementing in software

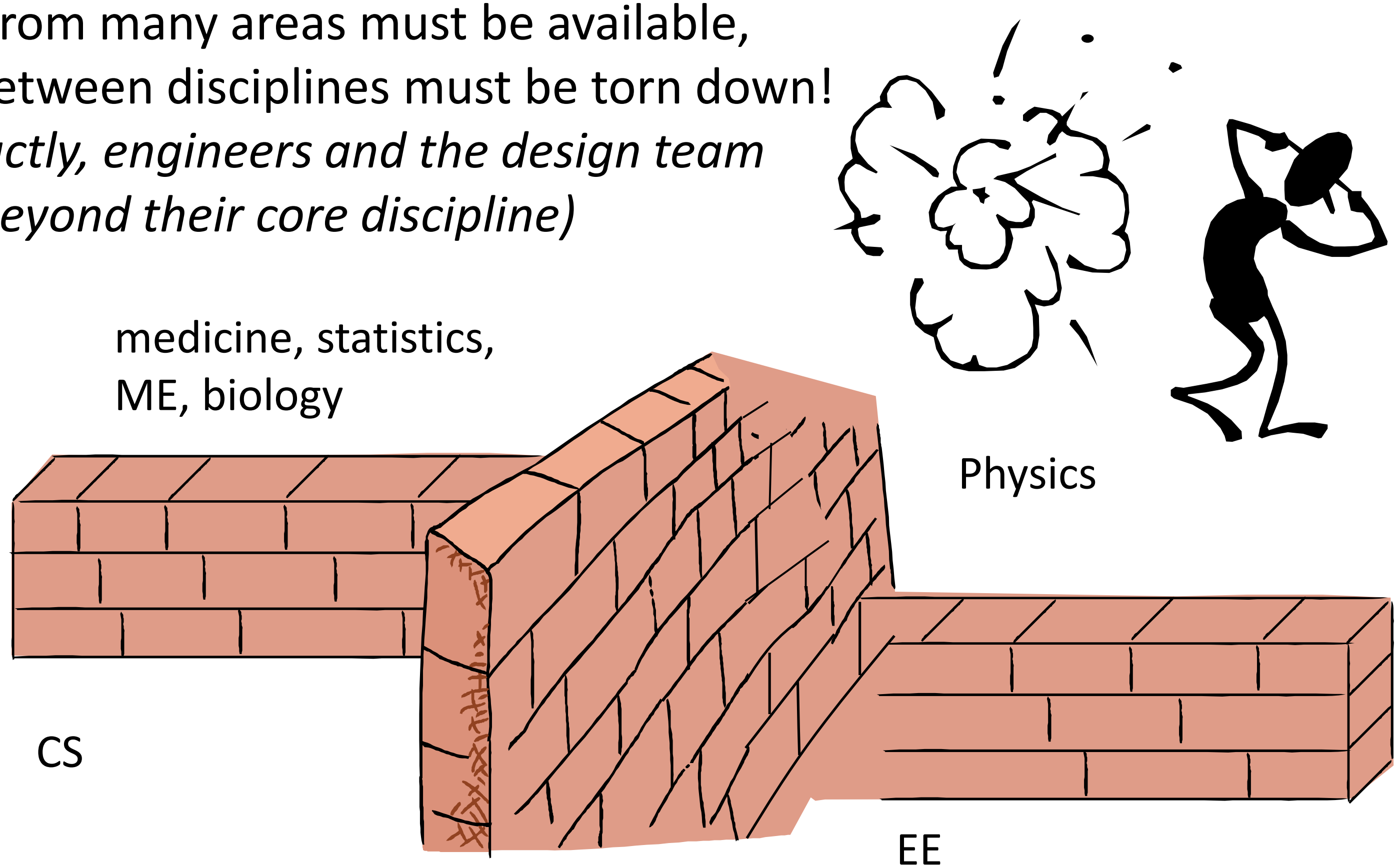
If CPS/ES will be implemented mostly in software, then why don‘t we just (re)use what software engineers have come up with?



Embedded Systems Engineers

It is not sufficient to consider CPS/ES as a special case of software engineering…

Knowledge from many areas must be available, the ‘walls’ between disciplines must be torn down!
(or more exactly, engineers and the design team need to go beyond their core discipline)

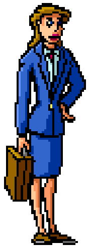


Real Nature of Embedded Engineering

Developing Domain Knowledge

Challenges for CPS/ES Software

- Dynamic environments
- Capture the required behaviour!
- Validate specifications
- Efficient translation of specifications into implementations!
- How can we check that we meet real-time constraints?
- How do we validate embedded real-time software? (large volumes of data, testing may be safety-critical)



Software complexity is a challenge

Software in a TV set

- Source 1*:

Year	Size
1965	0
1979	1 kB
1990	64 kB
2000	2 MB

- Source 2°: 10x per 6-7 years

Year	Size
1986	10 KB
1992	100 kB
1998	1 MB
2008	15 MB

☞ Exponential increase in software complexity

... > 70% of the development cost for complex systems such as automotive electronics and communication systems are due to software development
[A. Sangiovanni-Vincentelli, 1999]

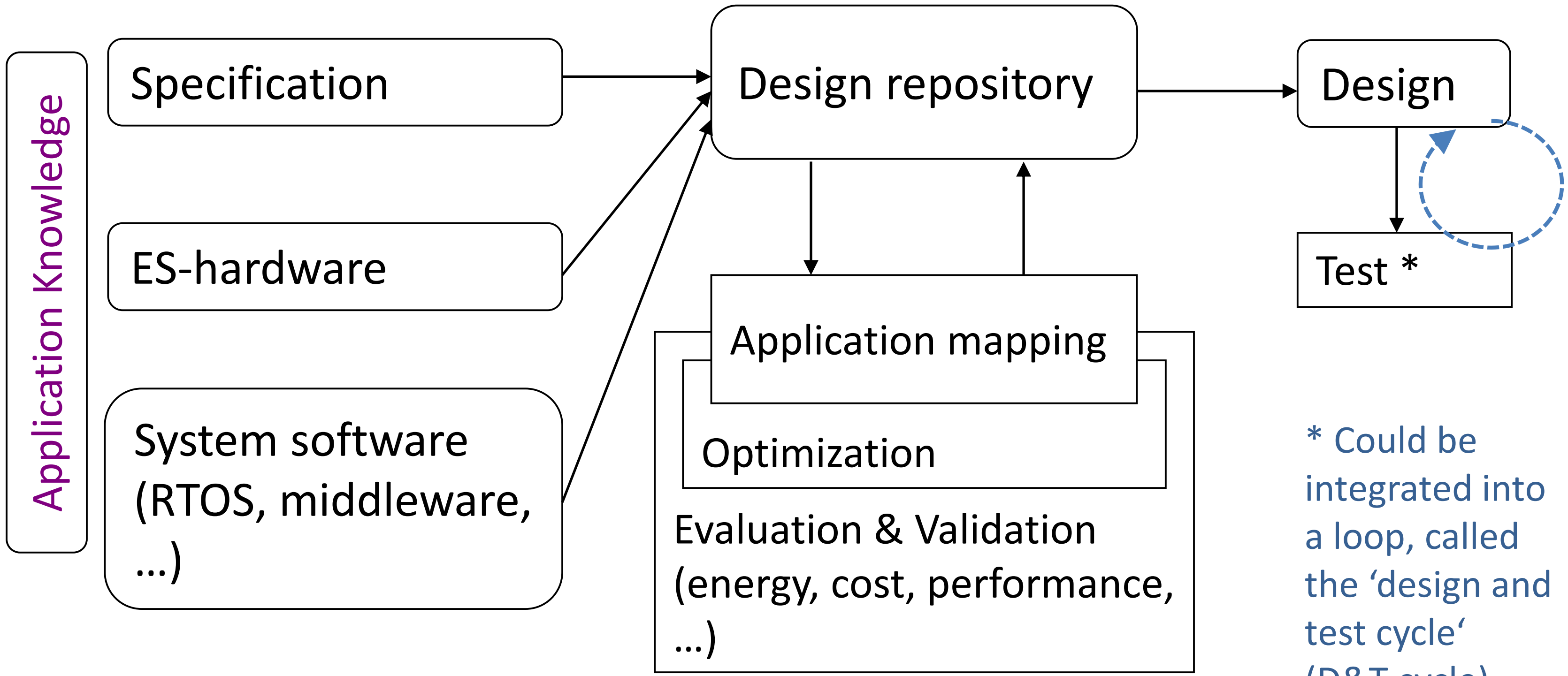
* Rob van Ommering, COPA Tutorial, as cited by: Gerrit Müller: Opportunities and challenges in embedded systems, *Eindhoven Embedded Systems Institute*, 2004

° R. Kommeren, P. Parviainen: Philips experiences in global distributed software development, *Empir Software Eng*. (2007) 12:647-660

Design flows
&
The Embedded System Lifecycle (part 1)

Embedded Systems II

Hypothetical design flow

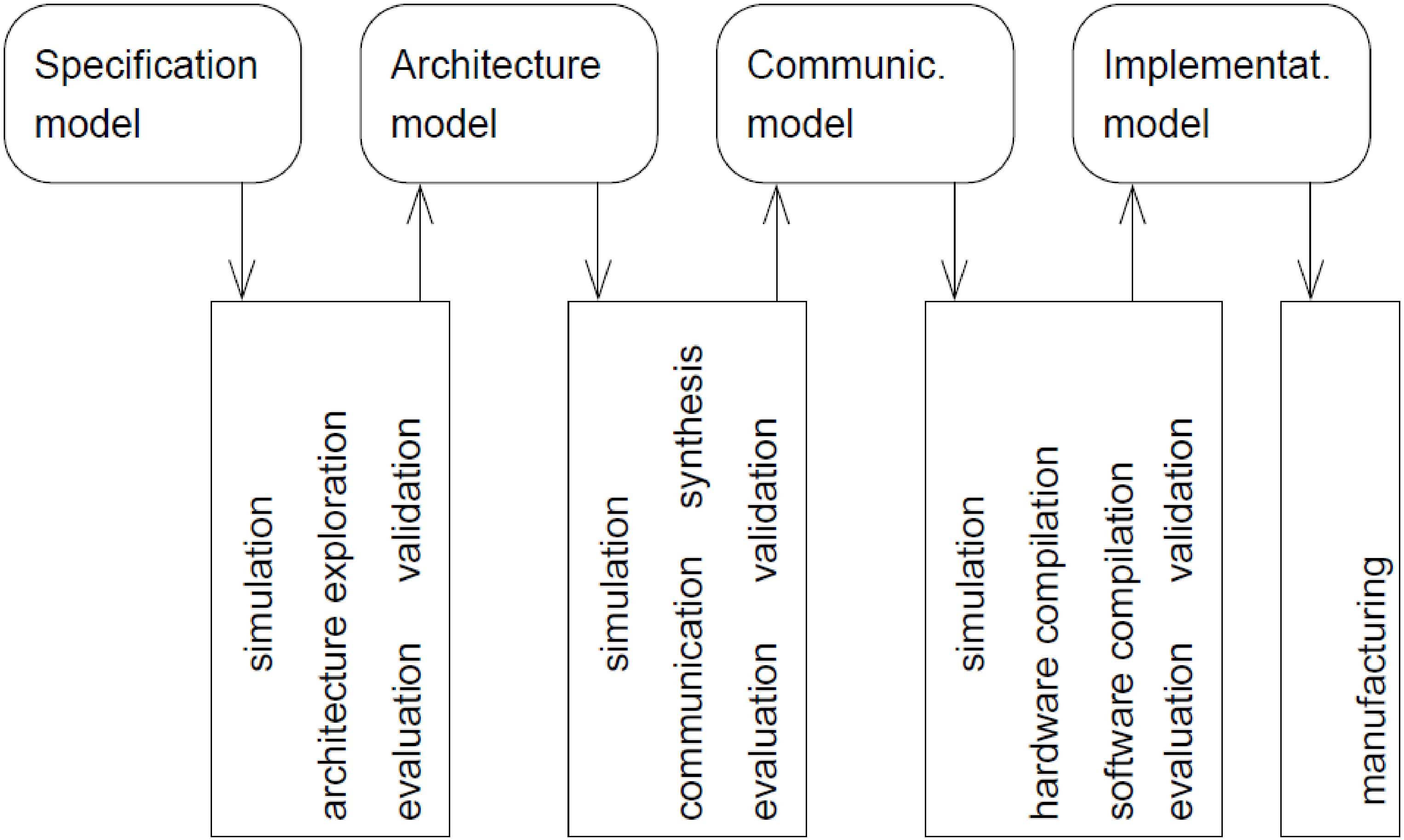


* Could be integrated into a loop, called the ‘design and test cycle‘ (D&T cycle)

This is the generic design cycle: tool chains may cause the number and type of the iterations to differ

Iterative design (1): - After unrolling D&T loop

- E.g. using SpecC and its tools

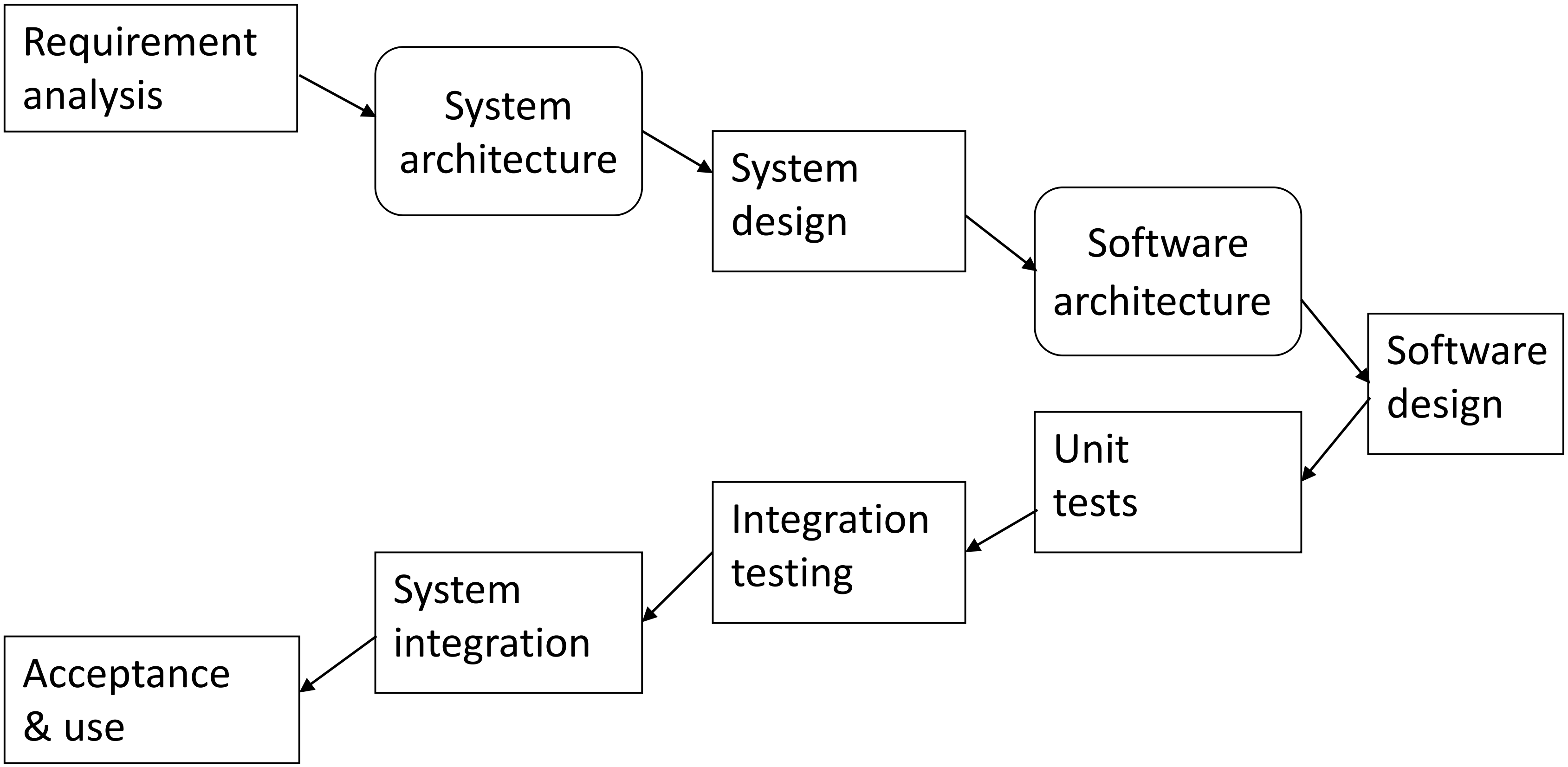


SpecC is a System Description Language (**SDL**), or System-level Design Language (SLDL), and is an extension of the ANSI C programming language.

Iterative design (2): - After unrolling D&T loop

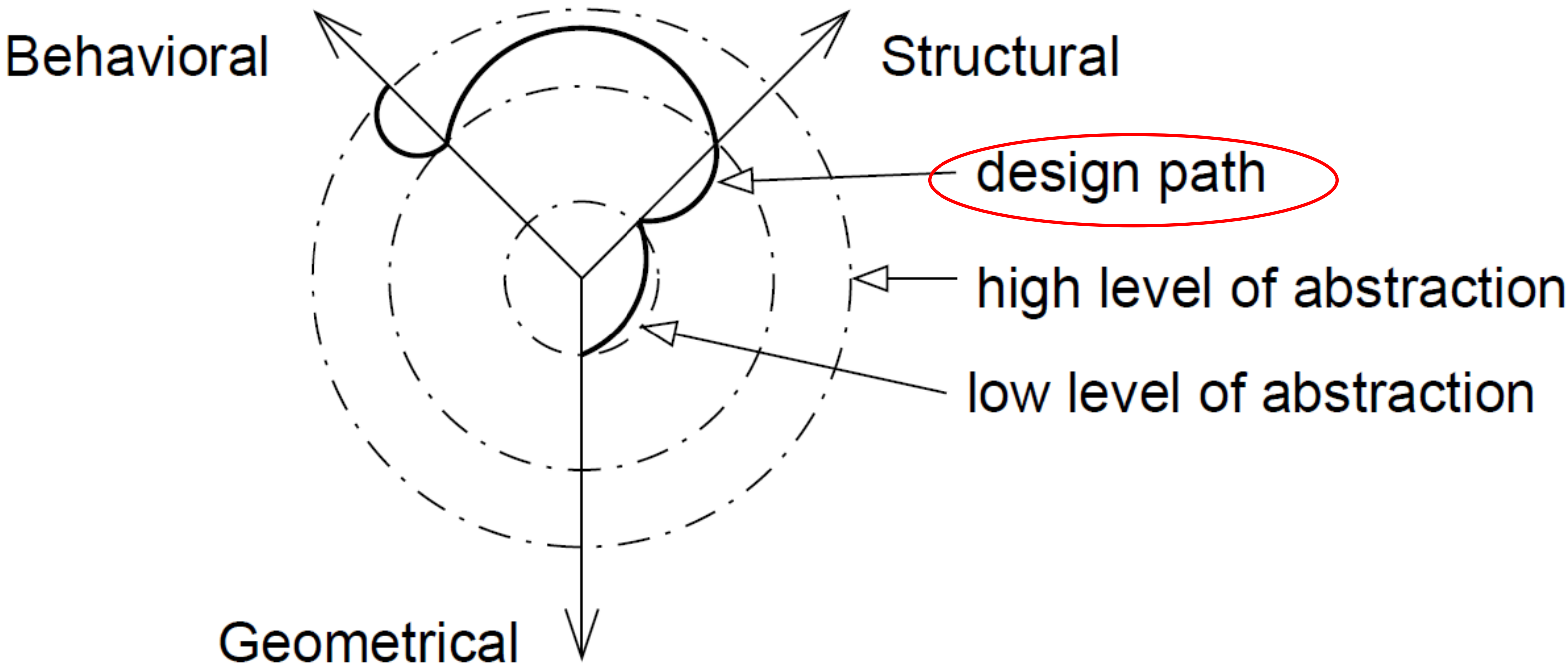
This is the main general high-level model to remember

Example: V-model



This model you will do in much detail in the design course

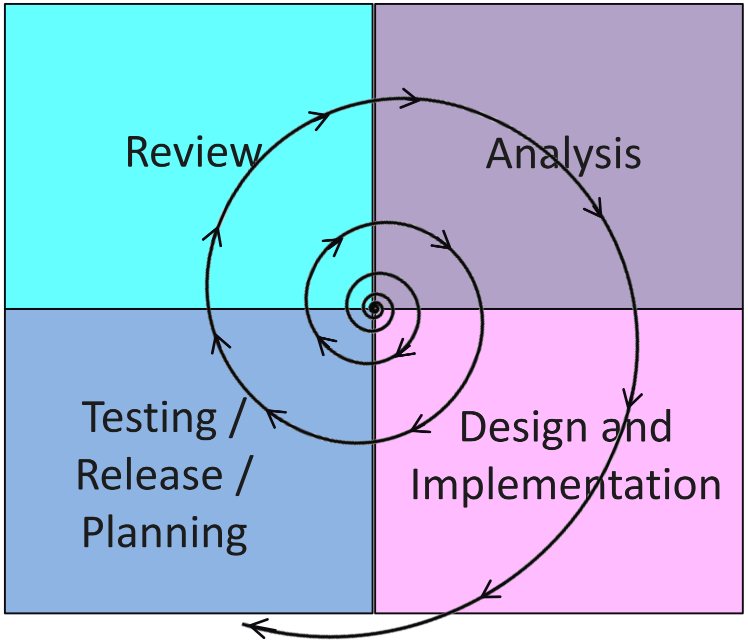
Iterative design (3): - Gajski's Y-chart



Spiral Model: the natural way

As per general development projects, Embedded Systems can be seen as following the Spiral Model (Bloehm 1988) with phases of…

Starting small (i.e. start from centre of spiral and expand out) with little risk. Adding features and mitigating risk with each additional iteration.



A spiral* model overview of development

Main phases of development (usually starts with requirements; the subsequent iterations start with a requirements review and deciding what next to do.)

* B. W. Boehm, "A spiral model of software development and enhancement," *Computer, vol. 21, pp. 61-72, 1988.*

Spiral model (Boehm, 1988)

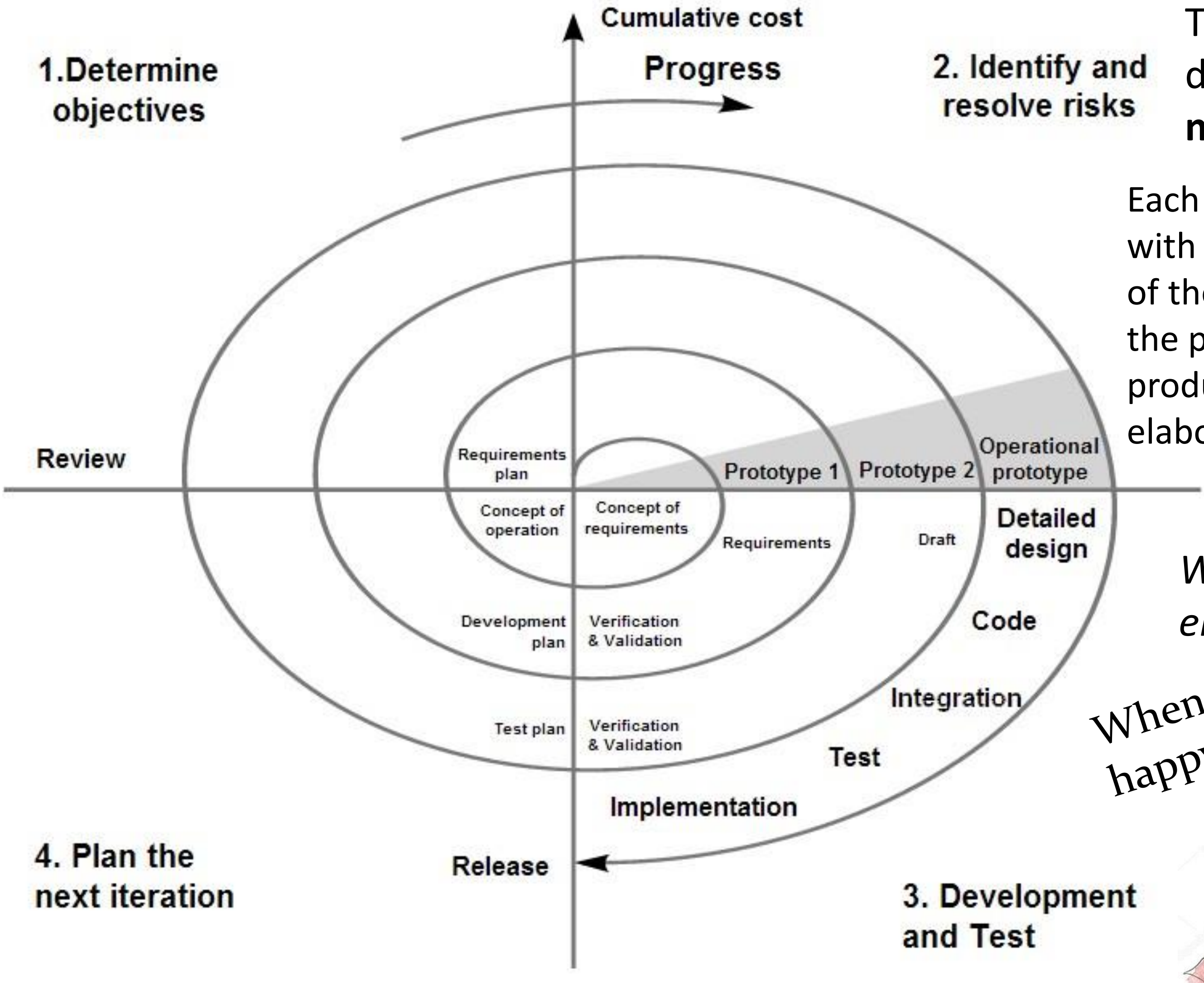


Image from Wikipedia open commons

This is the more detailed **classic model**.

Each cycle begins with identification of the <u>objective</u> of the portion of the product to be elaborated

*When does it end?! **…***

When you are happy enough?

More focused:

The 7-Stage Embedded System Design Cycle

This is the main ES specific design cycle to remember

This is the main ES specific design cycle to remember

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***

Prof. Arnold Berger's (University of Washington) Model

End of Lecture

The Next Episode…

Lecture L03

Specifications and modelling, Requirements, models of computation, Customer Tour, observer pattern, early design phases.

References and Acknowledgements

This presentation partly based on slides from the textbook's companion website. Used with permission by the author Prof. Peter Marwedel

Textbook:
Embedded System Design:
Embedded Systems Foundations of Cyber-Physical Systems
By Peter Marwedel, TU Dortmund