

Introduction

Embedded Systems / Cyber-Physical System Fundamentals (ES/CPSF)

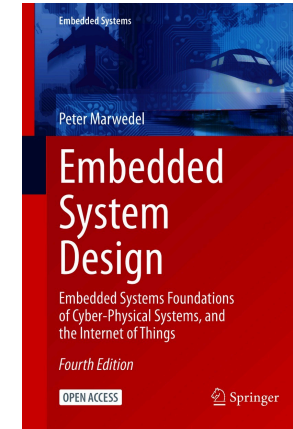
Peter Ulbrich

Course website

Winter Semester 2026/27

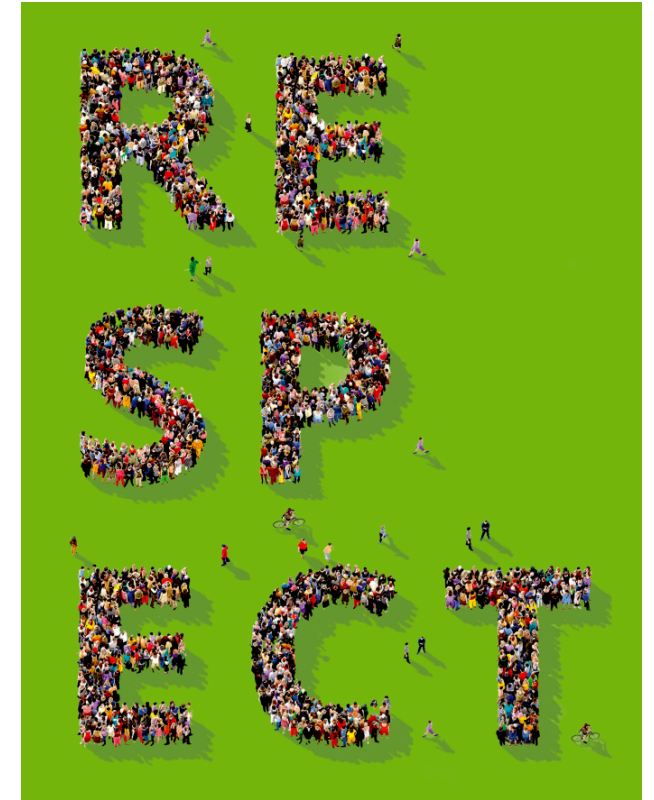
Embedded Systems (ES) / Cyber-Physical System Fundamentals (CPSF)

- Computing inside products, infrastructure, and physical processes
- Chapter 1 of Marwedel: terms, opportunities, challenges, common characteristics, design flows
- Running question: **How do we build trustworthy software-controlled physical systems?**



Code of Conduct

- At the Faculty of Computer Science, we promote a motivating and equal opportunity environment in which we learn, teach, and research together.
- Discrimination and sexual violence are not tolerated, vigorously prosecuted, and sanctioned to the fullest extent of the law.
- Unacceptable behavior includes, but is not limited to, harassing, abusive, discriminatory, intimidating, derogatory, and degrading behavior and language.



© Luise Henkel

Contact Persons

If you experience or observe discrimination or sexual violence, or have questions about the topic, you can contact the following bodies:

- The teacher(s) of your course
- Faculty Diversity Commission (Diversitaetskommission der Fakultaet)
 - Feel free to contact the student representatives directly
 - Or via an [anonymous contact form](#) to all members
- Central advice center for protection against discrimination and sexual violence
- Vice Rector for International Affairs and Diversity

There is a website with the various contact points [here](#).

Organisation and Dates

Lectures

- Tuesday, 10:00-12:00 c.t.
- Thursday, 16:00-18:00 c.t.
- Otto-Hahn-Str. 14, E23
- Lecturer: [Peter Ulbrich](#)

Links and Updates

- [Course website](#)
- [Moodle room](#)
- [LSF entry](#)
- Responsible: [Nils Hoelscher](#)

Exercises and Exam Admission

Theoretical Exercises

- See Moodle and LSF for groups and dates
- Practice material and feedback
- Can be submitted for feedback
- Not required for exam admission

Programming Exercises

- Two programming exercises during the semester
- Binary grading: pass or fail
- Both exercises required for exam admission
- Use them to connect models to running systems

Moodle Room

Moodle provides the operational information during the semester:

Assignments

Exercise sheets and programming tasks

Slides

Lecture material and updates

Schedule

Course updates, rooms, exercise dates

Forum

Questions and clarifications

Chapter 1 Map

1 Terms

2 Opportunities

3 Challenges

4 Traits

5 Curriculum

6 Design Flows

- ▶ Chapter 1 is not a list of definitions. It sets up the design tension for the whole course.

Why This Course Exists

Information processing has moved from visible computers into **physical products**, **critical infrastructure**, and **networked environments**.

Embedded

Computation becomes part of a larger product and is often invisible to users.

Physical

Software interacts with time, energy, space, heat, motion, and uncertainty.

Connected

Devices cooperate through communication and are increasingly maintained over long lifetimes.

A Short History of Terms

Ubiquitous Computing

Information processing anytime and anywhere; computers fade into the environment.

Pervasive Computing

The practical penetration of daily life with computing devices and services.

Ambient Intelligence

Smart environments, often focused on homes, buildings, and communication.

- ▶ These visions differ in emphasis. Embedded systems provide the technical foundation beneath all of them.

Embedded Systems

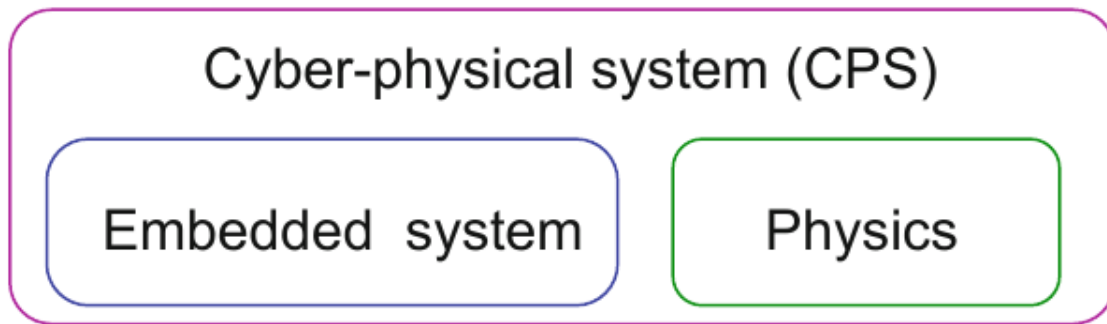
Marwedel: Embedded systems are information processing systems embedded into enclosing products.

- **Product first:** the computer is part of a larger technical system
- **Constraints:** timing, energy, heat, weight, cost, lifetime
- **Dependability:** failures may harm people, assets, or infrastructure
- **Interface:** sensors, actuators, networks, and users shape the software

► **Design consequence:** the computer is not the product; it must fit the product's physical and organizational context.

Cyber-Physical Systems

Lee: Cyber-physical systems are integrations of computation and physical processes.



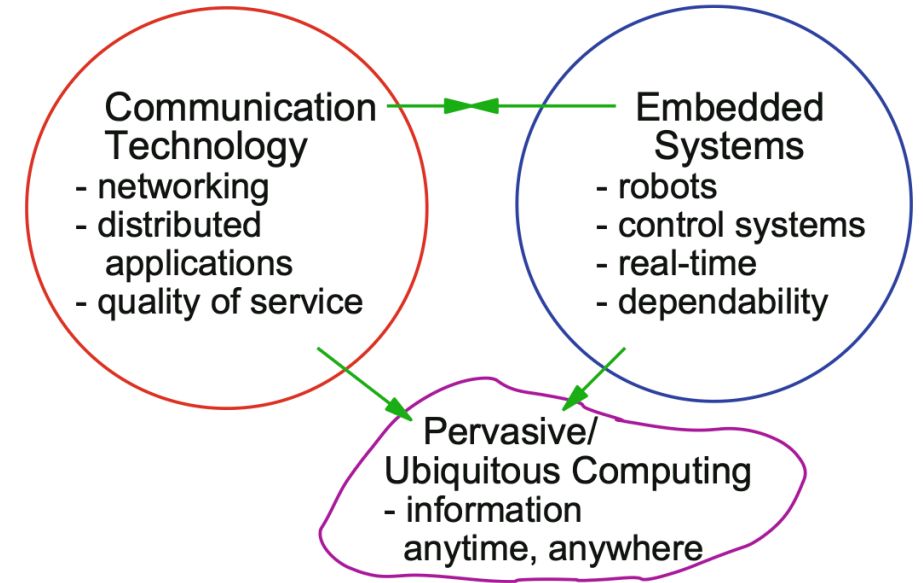
- **Cyber:** computation, models, communication, data
- **Physical:** time, energy, space, motion, heat, uncertainty
- **System:** behavior emerges from the interaction, not from software alone

► Working model for this course: **CPS = embedded system + physical environment.**

Communication Completes the Picture

The Book's View

- CPS definitions often stress physics
- IoT definitions make communication explicit
- Real deployed systems need both
- The boundary between CPS and IoT is often pragmatic



IoT and Industry 4.0

Internet of Things

- **Identity**: devices can be addressed and distinguished
- **Connectivity**: sensors and actuators cooperate via networks
- **Coordination**: local embedded loops become distributed systems

► IoT emphasizes communication and cooperation; CPS emphasizes the physical feedback loop. Real systems often need both.

Industry 4.0

- **Lifecycle**: design, production, operation, and maintenance are linked
- **Flexibility**: factories adapt to variants and changing demand
- **Data**: measurements become operational and engineering feedback

Cyber-Physical Feedback Loop

Process

Sense

Compute

Actuate

Feedback

- Sensors translate physical quantities into data.
 - Software computes under timing and resource constraints.
 - Actuators change the physical process.
 - Communication connects several loops into a distributed system.
- The core lecture pattern: every design decision must respect both computation and the physical feedback loop.

Chapter 1 Map: Opportunities

1 Terms

2 Opportunities

3 Challenges

4 Traits

5 Curriculum

6 Design Flows

Opportunity Landscape

Mobility

Cars, railways, avionics, ships, autonomous and assisted driving.

Production

Factory automation, logistics, product life-cycle integration, Industry 4.0.

Infrastructure

Smart buildings, smart grid, structural monitoring, disaster recovery.

Health

Sensors, therapy support, assistive devices, monitoring, medical equipment.

Science

Physical experiments observed, filtered, triggered, and controlled by IT.

Everyday Systems

Telecommunication, consumer electronics, smart homes, wearables.

Control in Engineered Products

Why the Examples Matter

- Many functions depend on sensing, control, and actuation
- Dependability and timing affect safety directly
- Modern mobility adds autonomy, connectivity, and software updates
- Production systems connect planning, operation, and maintenance



Physical Contexts and Embodiment

Physical Context Matters

- Dams, bridges, mountains, and volcanoes are monitored before failures become visible
- Robots combine mechanics, control, sensing, actuation, and software
- CPS design often starts with the environment, not with the processor



Chapter 1 Map: Challenges

1 Terms

2 Opportunities

3 Challenges

4 Traits

5 Curriculum

6 Design Flows

Why Design Is Hard

Problem

Embedded systems are judged by several objectives at once: **correct values**, **right time**, **low energy**, **safe behavior**, acceptable cost, and manageable evolution.

Design Response

Make constraints explicit early, choose models that expose them, and validate each design step before it becomes expensive to change.

Dependability Comes First

Dependability: The ability to deliver a service that can justifiably be trusted.

Safety

Prevent unacceptable harm caused by system behavior.

Security

Protect the system and its assets against threats.

Reliability

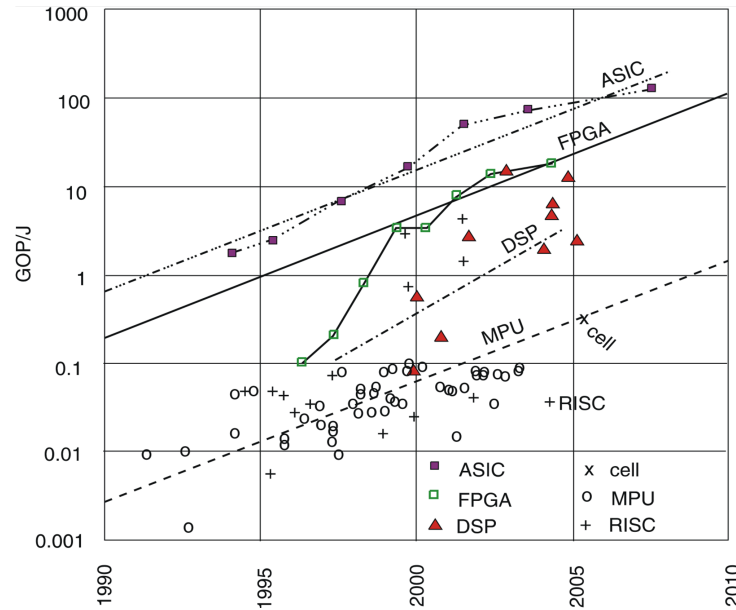
Deliver correct service over a specified period of time.

Availability

Be operational when the service is needed.

- ▶ Dependability is an architectural concern. It cannot be bolted on after implementation.

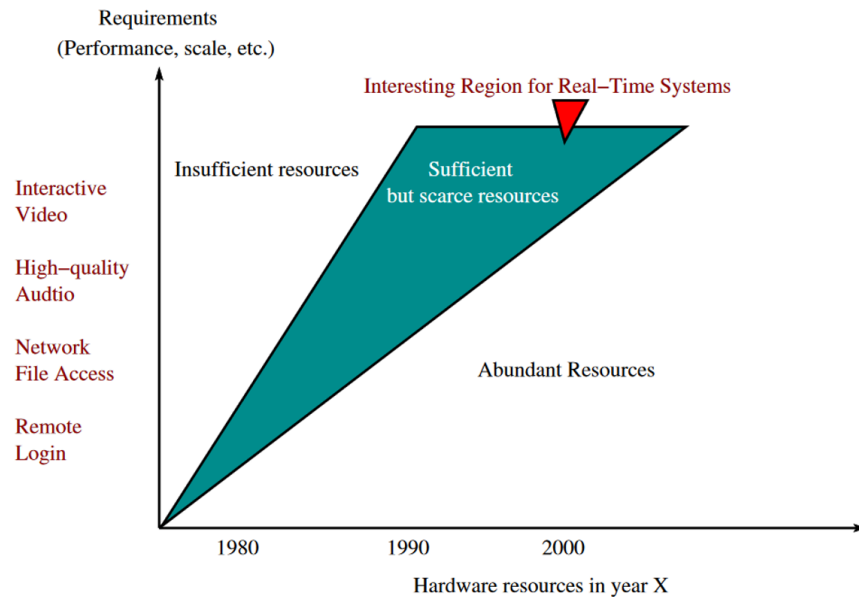
Efficiency Is Multidimensional



Not Just Faster

- Energy per operation shapes architecture choices
- Memory footprint, weight, heat, and unit cost matter
- Flexibility has a price: software on general-purpose processors is not always affordable
- Efficient software design remains necessary even when hardware improves

The Space of Design

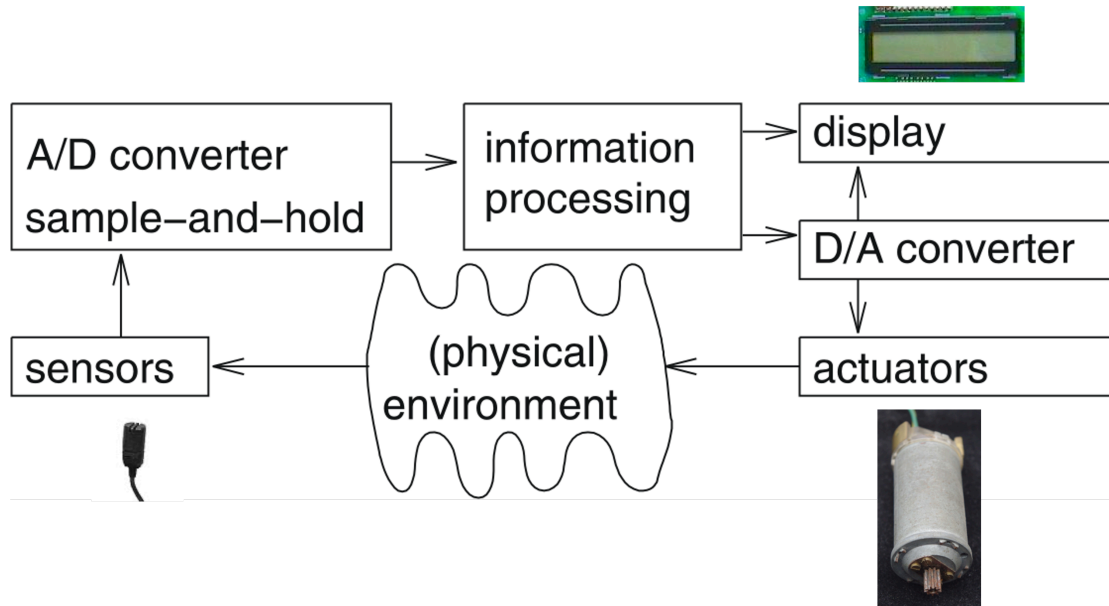


The Interesting Region

- Too few resources: requirements cannot be met
- Too many resources: product becomes too costly, heavy, hot, or energy hungry
- Real-time design often lives in the **sufficient but scarce** region

► The goal is not one maximal metric, but a feasible trade-off.

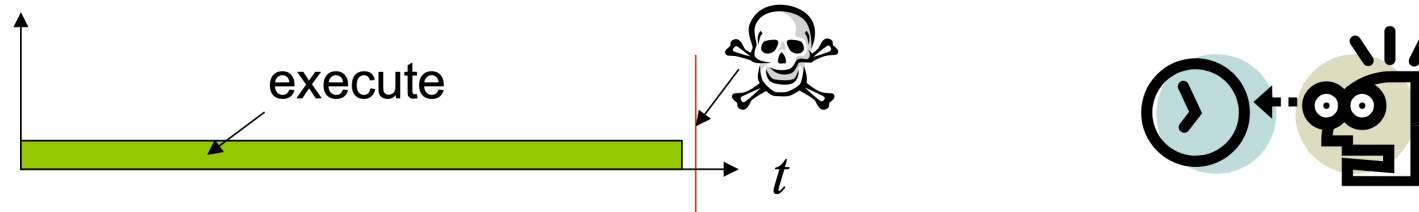
Closed-Loop Embedded Control



A Typical CPS Loop

- Sensors observe physical quantities
- A/D conversion samples and quantizes them
- Software computes a reaction
- D/A conversion and actuators influence the environment
- Displays and communication expose state to humans or other systems

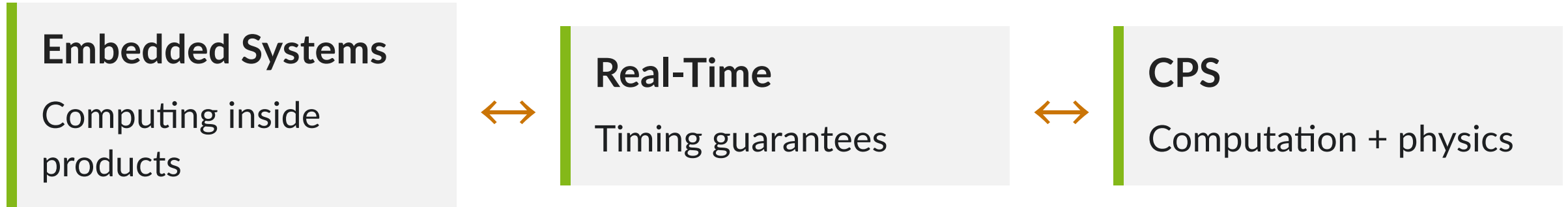
Real-Time Constraints



What Makes It Real-Time?

- A guaranteed response must be argued without statistical averages
 - The environment dictates the time interval
 - In hard real-time systems, a missed deadline can be catastrophic
 - Soft deadlines degrade service quality rather than causing system failure
- **Correct value + wrong time = wrong behavior.**

Typical Misconceptions



- ▶ The concepts influence each other, but they are not causal chains and not synonyms.
 - Not every embedded system is a real-time system.
 - Not every CPS is hard real-time.
 - Faster processors can still violate timeliness through caches, interrupts, contention, or energy management.

Reactive and Hybrid Systems

Reactive

The system stays in continual interaction with its environment. Its behavior is not a single input-output function, but a sequence of reactions.

Hybrid

Analog physical behavior and discrete digital computation meet. Sampling, quantization, clocks, and delays become part of the design.

- ▶ Automata, state machines, timed models, and control-oriented abstractions are natural tools because the environment is active.

Dynamics and Complexity

Modern CPS complexity is less about one large program and more about **software-defined products** that keep changing after deployment.

100M+

lines of code are common in
high-end vehicles today

OTA

updates turn products into
long-lived software
platforms

Safety + security

must be validated together
across variants

- This connects directly to the later pressure points: edge compute, connected products, digital twins, and sustainability.

Not Just Software Engineering

Embedded/CPS engineering breaks the walls between disciplines.

Computer Science

Models, algorithms, software, compilers, scheduling, verification.

Electrical Engineering

Circuits, signal processing, power, sensors, communication.

Physical Domain

Mechanics, medicine, biology, energy systems, production, transportation.

- ▶ The central skill is translating domain constraints into computational models and implementation constraints.

Chapter 1 Map: Characteristics

1 Terms

2 Opportunities

3 Challenges

4 Traits

5 Curriculum

6 Design Flows

Common Characteristics

Dedicated Function

Applications are often known at design time.

Physical Interfaces

Sensors and actuators introduce units, noise, latency, and calibration.

Resource Constraints

Energy, size, memory, heat, and cost shape the architecture.

Real-Time Relevance

The environment can impose deadlines.

Dependability Needs

Failures can affect people, assets, or infrastructure.

Invisible Interfaces

Users may not see the computer, but they feel its behavior.

Embedded vs. PC/Server Design

Aspect	Embedded / CPS	PC- or Server-like
Architecture	often heterogeneous and compact	mostly homogeneous
Applications	frequently known at design time	usually open-ended
Optimization	energy, size, timing, heat, cost, reliability	average performance often dominates
Timing	frequently real-time relevant	usually best effort
Safety	may be safety-critical	usually not
Validation	must include platform and environment assumptions	often software stack centered

Current CPS Design Pressures

Edge AI and TinyML

More sensing and inference move close to the physical process, often under tight energy and memory constraints.

Digital Twins

Engineering practice increasingly links deployed systems with executable models and operational data.

Connected Products

Long-lived devices now need updates, secure boot, supply-chain awareness, and lifecycle management.

Sustainability

Energy, repairability, resource use, and lifetime are becoming explicit design objectives.

Chapter 1 Map: Curriculum

1 Terms

2 Opportunities

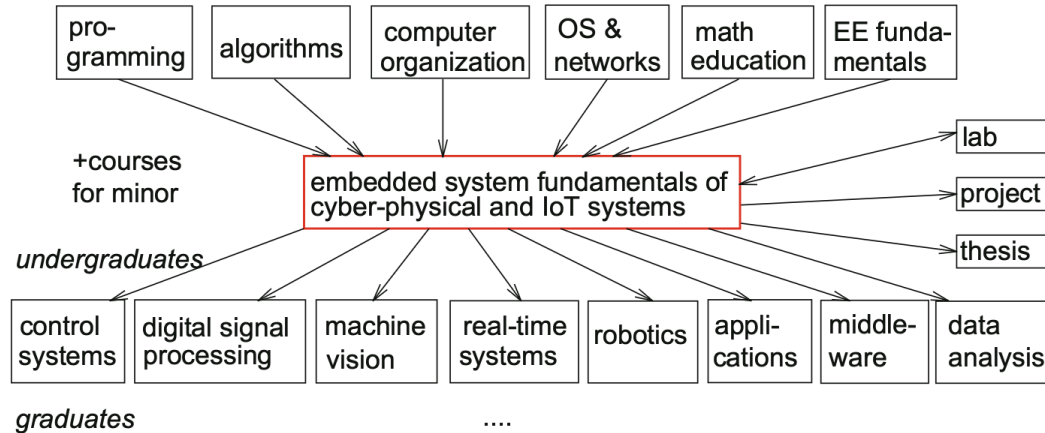
3 Challenges

4 Traits

5 Curriculum

6 Design Flows

Where This Course Fits



Bridge Course

- Uses programming, algorithms, computer organization, operating systems, networks, and mathematics
- Prepares for real-time systems, robotics, DSP, machine vision, middleware, security, and data analysis
- Connects software techniques with physical and resource constraints

Recurring Design Questions

Measure

Compute

Time

Resources

Evidence

- **Measure:** sensors define what the system can know about the world
 - **Compute:** models and algorithms turn observations into decisions
 - **Time:** deadlines decide whether a correct value is still useful
 - **Constrain:** energy, memory, cost, heat, and size limit the solution space
 - **Justify:** tests, analysis, and verification support trust in the deployed system
- The course returns to these questions whenever it introduces models, hardware/software mapping, timing, energy, and validation.

Chapter 1 Map: Design Flows

1 Terms

2 Opportunities

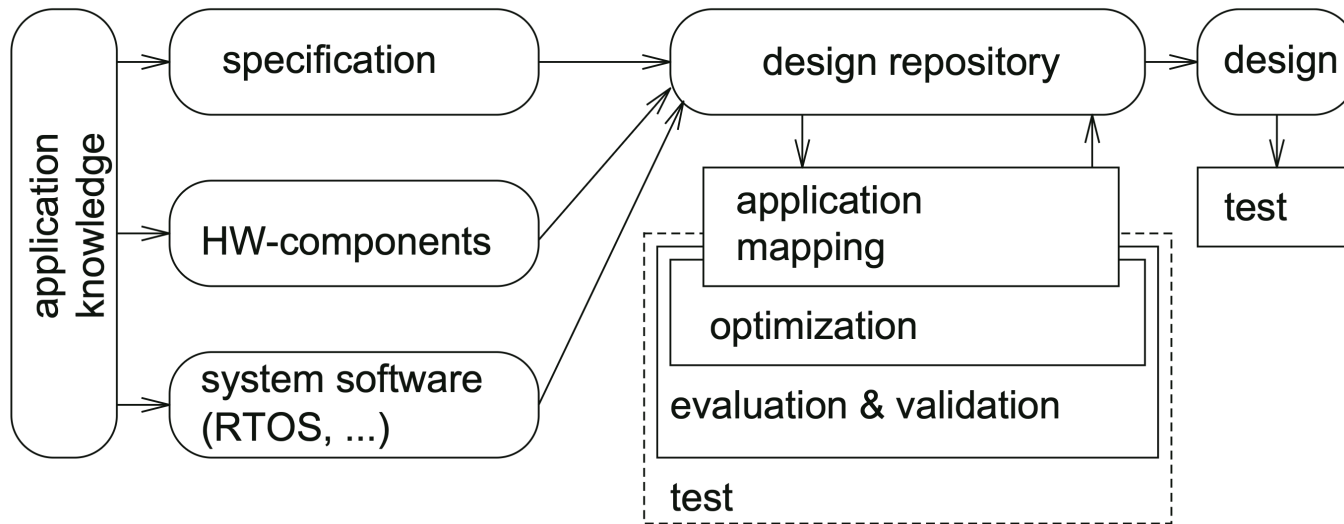
3 Challenges

4 Traits

5 Curriculum

6 Design Flows

Generic Design Flow



What the Flow Says

- Capture application knowledge as a specification
- Reuse hardware and system software components
- Keep design information in a repository
- Iterate mapping, optimization, evaluation, validation, and testing

Design as a Learning Loop

Need

Candidate

Evidence

Decision

Repository

Step

Meaning

Need

required function, constraint, or product decision

Candidate

mapping to hardware, software, communication, and timing

Evidence

performance, energy, timing, safety, and cost checks

Decision

keep, refine, or reject the candidate

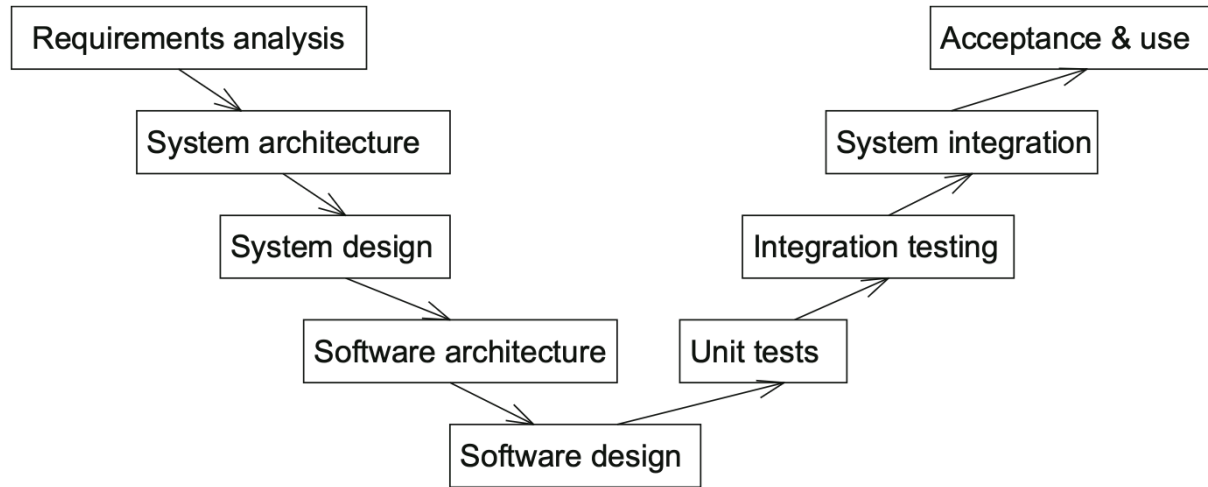
Repository

reusable components, assumptions, models, measurements, and test results

► Feedback matters: every iteration updates both the design and the repository.

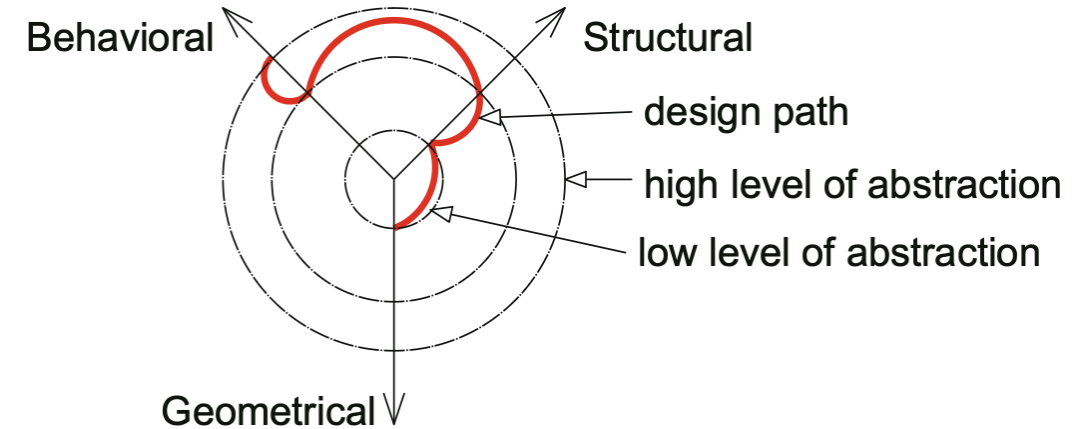
Design Flow Diagrams

V-Model



- Reads development as phases with matching validation activities
- Links early decisions to later tests

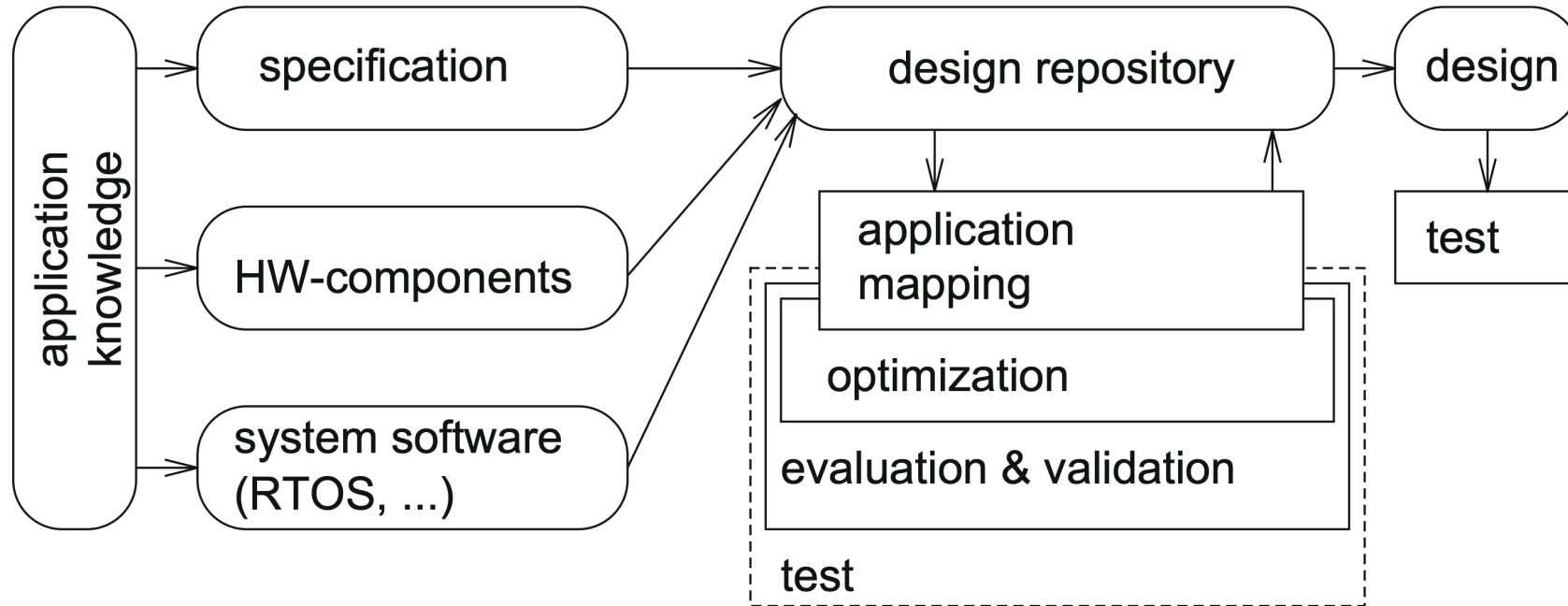
Y-Chart



- Reads design as movement between behavior, structure, and geometry
- Shows refinement from abstract models to concrete implementations

► Use both views: the V-model links design steps to evidence. The Y-chart links system views across abstraction levels.

Generic Development Paradigm



The same generic idea appears here: start from a specification and reusable components, then iterate mapping, optimization, evaluation, validation, and testing.

Structure of the Course

Foundations

Terminology, constraints, hardware/software interfaces.

Models

State, time, concurrency, communication, and physical interaction.

Implementation

Platforms, code generation, scheduling, memory, and energy.

Analysis

Timing, reliability, safety, security, and validation.

Systems View

Integration into CPS and IoT contexts.

Engineering Judgment

Choosing trade-offs that match the application domain.

Summary

What You Should Keep

- ES: information processing inside enclosing products
- CPS: computation integrated with physical processes
- IoT: communication and cooperation among identifiable devices
- The course studies the embedded foundations beneath all three

Why It Is Hard

- Dependability, timing, energy, cost, and complexity interact
- The physical environment is part of the specification
- Design must iterate between models, implementation, evaluation, and validation

Self Check

Explain

Why is “fast software” not the same as “real-time software”?

Compare

How do ES, CPS, and IoT differ in emphasis?

Apply

Pick one everyday product and identify its sensors, actuators, timing constraints, and dependability concerns.

References

Reading in Marwedel

Fourth edition (2021)

- ES, CPS, IoT; history of terms → Chapter 1.1
- Application domains and opportunities → Chapter 1.2
- Timing, dependability, efficiency → Chapter 1.3
- Common characteristics; ES vs. PCs → Chapter 1.4
- Curriculum and prerequisites → Chapter 1.5
- Design flows and iteration → Chapter 1.6
- Course and book structure → Chapter 1.7

Bibliography

1. P. Marwedel. *Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, and the Internet of Things*. 4th ed., Springer, 2021. [Open-access book](#).
2. E. A. Lee. *Computing Foundations and Practice for Cyber-Physical Systems: A Preliminary Report*. Technical Report UCB/EECS-2007-72, UC Berkeley, 2007. [Report](#).

