As of 21.04.2026

Design Patterns Training (not only) for Embedded Systems - Live Online Training

Objectives

This training shows you under which conditions classic design patterns can be used efficiently even in embedded systems with limited resources.

A selection of the most suitable patterns is presented based on examples typical of embedded systems.

Most of the patterns are developed step by step in complete code examples before being presented in their final abstracted UML representation.

Moreover, as part of this process, solutions are discussed that are often applied but had better be avoided ("Anti Patterns").

This approach facilitates a well-grounded assessment of the memory and/or runtime cost in relation to the benefit of using a design pattern.

This course also highlights new debugging possibilities resulting from the application of design patterns, considerably reducing the development time and improving code quality which is crucial especially in embedded systems.

Participants

C++ software developers, software architects

Requirements

Good knowledge of the programming language C++.

Live-Online-Training

* Price per attendee, in Euro plus VAT

Training code: LE-DP

Face-To-Face - English

Duration
4.5 days

Live Online - German

Date	Duration
09.11. – 13.11.2026	5 days

Face-To-Face - German

Date	Duration
06.07. – 10.07.2026	4.5 days

MICROCONSULT

As of 21.04.2026

Design Patterns Training (not only) for Embedded Systems - Live Online Training

Content

Introduction

- History
- What are design patterns?
- GoF design patterns
- Typical problems in embedded systems
- Design patterns in embedded systems

Creational Patterns

- Example: Motor control application
- Flexible design based on interfaces
- Practical exercise: Measuring the memory space and runtime costs of an interface
- Comparison of static and dynamic polymorphism
- Example: Position tracking for a goods transport system
- Reusing the position tracking system for airplanes
- Position tracking based on the design pattern Abstract Factory
- Factory generation using the design pattern Singleton

Structural Patterns

- Example: Motor control application
- Alternative design based on the design pattern Adapter
- Workshop exercise: Debugging a counter application
- Solutions based on the design pattern Decorator
- Example: Multithread application
- Identifying the problems of typical solutions
- Flexible solution based on the design pattern Proxy
- Protection proxy, virtual proxy, remote proxy
- Smart reference / smart pointer

Behavioral Patterns

- Example: Handling timer events
- Flexible solution based on the design pattern Observer
- Practical exercise: Using the observer pattern in the elevator control
- Pitfalls in interface design or implementation
- "Horizontal" and "vertical" interfaces
- Event handling based on the design pattern Command
- Practical exercise: Using the command pattern in the elevator control
- Example: Traditional implementation of a state machine in C
- Object-oriented solution based on the design pattern State
- Practical exercise: Using the state pattern in the elevator control
- Example: User-defined memory management
- Partitions and their management using partition managers
- Flexible memory management based on the design pattern Strategy
- Practical exercise: Using the strategy pattern in the elevator control
- Example: Strategies with a common basic structure
- Implementation based on the design pattern Template Method

More Patterns

- Design pattern Factory Method
- Entwurfsmuster Prototype
- Design pattern Facade
- Design pattern Composite
- Design pattern Memento
- Design pattern Chain of Responsibility
- Design pattern Flyweight
- Design pattern Iterator
- Design pattern Mediator

Practical Exercises in the Design Patterns Training

- The exercises are performed with IAR Embedded Workbench and the design tool Enterprise Architect

You will acquire the following practical knowledge:

- How to perform storage space and runtime measurements
- How to use design patterns to enhance software quality
- How to use patterns for debug purposes
- How to develop projects further by using design patterns