maurer treutner

Embedded Systems · Objektorientierung · SW-Entwicklung · Beratung · Schulung

Schulungskatalog

Übersicht und Themenauswahl

Stand: Februar 2019

Inhaltsverzeichnis

Unser Schulungsangebot

Ziel unseres Schulungsangebots ist es, möglichst viele Aspekte der modernen Software- und Systementwicklung abzudecken. Deshalb richten wir unser Angebot an den Phasen aus, die mechatronische, embedded bzw. reine Softwareprojekte gewöhnlich durchlaufen.

Mit unserem Schulungsangebot wenden wir uns in erster Linie an Unternehmen, die neue Technologien einführen wollen oder bei bereits eingeführten Technologien das vorhandene Wissen ihrer Mitarbeiter gezielt vertiefen möchten. Wir führen unsere Veranstaltungen überwiegend bei unseren Kunden vor Ort durch, wobei die Inhalte konkret auf die individuellen Anforderungen angepasst werden. Auch die Übungseinheiten, die wir in unsere Kurse integrieren, werden auf die individuellen Bedürfnisse und Wünsche unserer Kunden zugeschnitten.

Wenn Sie Interesse an unseren Schulungen haben, setzt sich einer unserer erfahrenen Trainer gerne mit Ihnen in Verbindung und arbeitet auf der Grundlage Ihrer Anforderungen einen Vorschlag für die Agenda Ihrer Schulung aus. Das Training wird mit individuell für Sie zusammengestellten Unterlagen und mit Praxisanwendungen durchgeführt, die auch aktuelle Fragestellungen aus Ihren Projekten berücksichtigen können.

Wir freuen uns auf Ihre Kontaktaufnahme!

Methoden und Entwicklungsprozess

Unser Schulungsangebot zu den Themen Methoden und Entwicklungsprozess befasst sich mit allen wichtigen Aspekten moderner Software- und Systementwicklung.

Auf agilen Entwicklungsmethoden und deren Einsatz im Umfeld der Entwicklung von Embedded- und Echtzeitsystemen liegt dabei unser Schwerpunkt. Wir haben mit diesen Methoden nicht nur im Rahmen unserer Beratungstätigkeit sondern auch in eigenen Entwicklungsprojekten ausgezeichnete Erfahrungen gemacht.

Zu jeder Entwicklung gehören die Teilprozesse Konfigurationsmanagement, Requirements-Management und Change Request-Management. Im Zentrum unserer Schulungen in diesem Bereich steht die Vermittlung entsprechender Methoden und Tools.

Mit Kanban zum Projekterfolg

Kanban ist ein Werkzeug für das Management von Entwicklungsprojekten aus dem Umfeld der Lean-Methoden und versteht sich als Rezept für den Projekterfolg. Wir stellen die wesentlichen Elemente von Kanban vor und zeigen anhand von Beispielen, wie diese umgesetzt werden können. Dieses Training ist so konzipiert, dass es am Beginn der Einführung von Kanban in Ihrer Organisation stehen kann.

Themenauswahl

- Kanban im Überblick
- Change-Management mit Kanban
- Die Wertschöpfungskette in der Systementwicklung
- Qualitätsverbesserung und Kanban
- Visualisieren der Wertschöpfungskette
- Cardwalls implementieren
- Der Lieferrhythmus
- Limit the Work in Progress
- Kennzahlen in Kanban
- Kaizen – die eingebaute Prozessverbesserung
- Prioritäten setzen
- Teamstrukturen in Kanban
- Variabilität vermindern
- Kanban und Scrum
- Kanban für die Entwicklung sicherheitsgerichteter Systeme (IEC 61508)
- Anwendungsbeispiele

Agiles Projektmanagement

In den Schulungen aus diesem Themenfeld werden die wesentlichen Elemente agilen Vorgehens aus Sicht des Projektmanagements vorgestellt. Die Teilnehmer lernen die Besonderheiten der Rollen, der Abläufe und der Kommunikationsformen kennen, die agilen Ansätzen zu eigen sind. Insbesondere zeigen wir Ihnen auf, wie Alternativen zu klassischen, schwergewichtigen Entwicklungsprozessen gefunden und implementiert werden können.
Bei der Themenauswahl finden bewusst auch Aspekte des allgemeinen Projektmanagements Eingang.

Themenauswahl

- Entwicklungsprojekte und Produktionsprojekte
- Phasen und Milestones eines Projektes
- Kernelemente agilen Projektmanagements
- Das 'Agile Manifesto'
- Rollen und Commitments
- Agiles Anforderungsmanagement
- Auftraggeber und Auftragnehmer
- Selbstorganisierende, kooperierende und Cross Functional Teams
- Facilating und die Bedeutung des Mentors
- Kommunikation und Kommunikationsmittel
- Iterationen und Inkremente
- Die Funktion des Timeboxings
- Aufwandsabschätzungen
- Planungsaktivitäten und -werkzeuge
- Steuerungsaktivitäten
- Transparenz, Projektstand, Berichterstattung und Meetings
- Fortlaufendes Qualitätsmanagement
- Multi Project Managment und Matrixorganisation
- Phasenorientierte Managementprozesse und agiles Vorgehen im Einklang
- Einführung von agilem Projektmanagement in bestehende Unternehmensstrukturen

Agile Methoden in der Systementwicklung

Die meisten weit verbreiteten Entwicklungsprozesse für Softwareprodukte bzw. mechatronische Produkte leiten ihre Kernelemente aus denen des klassischen Projektmanagements ab. Entwicklungsprojekte im mechatronischen Umfeld bringen jedoch besondere Umstände und Eigenschaften mit sich, denen mit adäquaten Mitteln begegnet werden muss. Mit dem Etablieren agiler Methoden eröffnen sich den Beteiligten solcher Projekte endlich Möglichkeiten, effizient und pragmatisch Systeme zu entwickeln, ohne sich in starren, praxisfernen Abläufen zu verlieren.
In den Schulungen aus diesem Themenfeld wird der Fokus auf die relevanten Kernelemente agiler Methoden gesetzt und diskutiert, wie man diese in der alltäglichen Praxis umsetzt. Weiterhin werden einige konkrete Methoden im Überblick besprochen.

Themenauswahl

- Die Erfolgsfaktoren eines Entwicklungsprojekts
- Das Entwicklungsteam im Zentrum des Geschehens
- Kommunikation und Kooperation
- Sinnvolle Artefakte
- Anforderungen und Issues
- Lernen und Adaption
- Zuständigkeiten sinnvoll zuordnen
- Planung des Projekts und der Iterationen
- Schätzen und Messen
- Milestones und Releases
- Release Iterations
- Continuous Integration
- Qualität der Modelle und des Codes
- Testen in agilen Projekten
- Change Requests und wie mit ihnen umgegangen wird
- Kanban
- Scrum
- Extreme Programming (XP)
- Feature Driven Development

Requirements-Management

Die Analyse des zu konstruierenden Systems begleitet in modernen Projekten durchgängig alle weiteren Aktivitäten bis zur Produkteinführung. Die dabei ermittelten Anforderungen, ihre Strukturen und Beziehungen zu Entwurfsdokumenten und dem Produkt selbst erreichen schnell eine Komplexität, die zielgerichtetes und systematisches Vorgehen notwendig macht. Untersuchungen zeigen, dass sich mangelnder Projekterfolg bzw. Produktqualität in vielen Fällen auf nicht optimales Management der Anforderungen zurückführen lässt.
In unseren Schulungen zum Thema Requirements-Management stellen wir verschiedene Möglichkeiten und Best Practices vor, die den optimalen Umgang mit diesem Thema sicherstellen. Weiterhin ist es uns wichtig zu zeigen, dass der Prozess des Requirements-Management genau auf das Projekt und dessen Randbedingungen zugeschnitten sein muss, um praktikabel und effektiv zu bleiben.

Themenauswahl

- Anforderungen als treibende Kraft eines Projekts
- Anforderungskategorien
- Anforderungsermittlung
- Anforderungen, Stakeholder und Kommunikation
- Dokumentation von Anforderungen
- Requirements-Engineering und -Management als Teil des Vorgehensmodells
- Requirements-Tracing und -Tracking
- Strukturierungsgrad von Anforderungsdokumenten und Beziehung zu anderen Systemmodellen
- Qualitätssicherung, Verifikation und Test
- Requirements-Management in agilen Methoden
- Einfluss von verbindlichen Normen auf das Requirements-Management
- Tools für das Requirements-Management

Software- und Systemarchitektur

Die Software- und Systemarchitektur hat maßgeblichen Einfluss auf den Erfolg von Soft- und Hardware-produkten. Eine gute Architektur ist für die Qualität eines Produktes von zentraler Bedeutung. Sie kann nicht nur die Produkteinführungszeit verkürzen, sondern durch bessere Wartbarkeit und Erweiterbarkeit auch die Lebenszeit eines Produkts erhöhen. Entwickeln Sie mittel- bis langfristige Produkte innerhalb einer Domäne oder bewegen sich Ihre Bemühungen innerhalb eines Produktportfolios? Dann ist eine wohldurchdachte Architektur unerlässlich!

UML hat sich in den letzten Jahren zu einem allgemein anerkannten Standard in der Software-modellierung entwickelt. Die aktuelle Version der UML bietet eine in sich konsistente Basis, die für alle Einsatzfelder der Softwaremodellierung - von der Dokumentation bis hin zur modellgetriebenen Softwareentwicklung - verwendet werden kann.

Mit SysML steht mittlerweile für die Systemmodellierung eine entsprechende Sprache zur Verfügung, die auch für die softwarefernen Bereiche der Systementwicklung einen Mehrwert bringt. Die gegenüber UML wesentlich einfachere Notation sowie spezielle Diagramme erleichtern es Einsteigern, diese Modellierungssprache schnell produktiv einsetzen zu können.

Speziell für die Neueinführung von UML bzw. SysML in Unternehmen bieten wir integrierte Pakete an, die neben der Durchführung von Schulungsmaßnahmen auch die Beratung bei der Auswahl von Modellierungstools sowie die Begleitung Ihrer Teams während der Einführungsphase umfassen.

Wir führen unsere Kurse je nach Ihren Bedürfnissen mit oder ohne Modellierungswerkzeug durch.

UML Grundlagen

Die hier zusammengefassten Themen vermitteln Anfängern sowie Umsteigern von früheren UML-Versionen einen Überblick über die aktuelle UML Version 2.5

Themenauswahl

- Grundbegriffe der Objektorientierung
- Objektorientierte Analyse und Design
- Geschichte der UML
- Grundlagen der Softwaremodellierung
- Profile und Metamodellierung
- Klassen- und Paketdiagramm
- Objektdiagramm
- Kompositionsstrukturdiagramm
- Komponentendiagramm
- Deployment-Diagramm
- Use Case-Diagramm
- Aktivitätsdiagramm
- Zustandsautomat
- Sequenzdiagramm
- Kommunikationsdiagramm
- Timing-Diagramm
- Interaktionsübersichtsdiagramm

Softwaremodellierung mit UML

Hier ist die Anwendung der UML im Fokus, weshalb sich diese Schulung insbesondere an Entwickler und Systemarchitekten richtet. Wir kombinieren diese Themen auf Wunsch auch mit der Einführung in ein UML-Tool.

Themenauswahl

- Anwendungsbereiche der UML
- Modellierung im Entwicklungsprozess
- Requirements-Modellierung mit UML
- OO-Analyse mit der UML
- Softwarearchitektur
- Logische und physikalische Architektur
- Architektur- und Design Patterns
- Modellierung von Schnittstellen
- Persistenz
- Verhaltensmodellierung mit Zustandsautomaten
- Protokollzustandsautomaten
- Anforderungsänderungen in der Modellierung
- Modellierung für webbasierte Anwendungen
- Anbindung relationaler Datenbanken
- Anbindung graphischer Benutzeroberflächen
- Anbindung von Mensch-Maschine Schnittstellen
- Anbindung eines Embedded Frameworks
- Anbindung logischer und physikalischer Geräte

Systemmodellierung mit SysML

Die OMG Systems Modeling Language (SysML) wurde geschaffen, um eine auf UML beruhende, standardisierte Modellierungssprache für das Systemengineering zur Verfügung zu stellen. SysML bietet eine Grundlage, um Anforderungen zu modellieren, Systeme zu analysieren und zu designen. Die gemeinsame Notation erleichtert es allen Projektbeteiligten miteinander zu kommunizieren. Dadurch, dass SysML auf UML beruht ist bei der Softwaremodellierung ein nahtloser Übergang in UML-Modelle möglich.
Unsere Themenauswahl deckt alle Elemente der SysML sowie deren Einsatz ab.

Themenauswahl

- Geschichte der SysML
- Requirements-Diagramm
- Grundkonzepte der SysML
- Einführung in die Systemmodellierung
- Profile und Metamodellierung
- Use Case-Diagramm
- Blockdefinitionsdiagramm
- Internes Blockdiagramm
- Parametric-Diagramm
- Paketdiagramm
- Aktivitätsdiagramm
- Sequenzdiagramm
- Zustandsautomat
- Unterschiede und Gemeinsamkeiten zwischen SysML und UML
- Marktübersicht SysML-Tools
- Echtzeitmodellierung

Software- und Systemarchitektur

Die Themen in diesem Abschnitt vermitteln Entwurfskonzepte und Strategien für die Entwicklung flexibler, wiederverwendbarer Soft- und Hardwarearchitekturen. Natürlich stehen auch für diese Themenblöcke praxisnahe und durchgängige Beispiele zur Verfügung.
Im Rahmen unserer Schulungen werden die Diagrammtypen aus UML und SysML, die zur Beschreibung von Architekturen relevant sind, umfassend vorgestellt.

Themenauswahl

- Prinzipien der Software- und Systemarchitektur
- Wechselwirkung zwischen Hard- und Software
- Physikalische Verteilung
- Modulare Softwaresysteme
- Konzepte der Objektorientierung und Architektur
- Architektur im Entwicklungs- und Modellierungsprozess
- Die Rolle des Architekten
- UML und SysML zur Beschreibung von Architekturen
- Schichtenmodelle
- Entwurfskonzepte für Softwareschichten
- Komponentenmodelle
- Entwurfskonzepte für Komponenten
- Einsatz von Architektur- und Design-Patterns
- Anbindung von Subsystemen und Geräten
- Architektonische Berücksichtigung der Nebenläufigkeit
- Architekturen für Kommunikationsanwendungen
- Frameworks und ihr Einsatz
- HW-Plattformen
- Standardhardware vs. Speziallösungen
- Sicherheitsanforderungen
- Der Einfluss der Mechanik

Modellgetriebe Softwareentwicklung

Unter modellgetriebener Softwareentwicklung versteht man ein Vorgehen, bei dem der Quellcode (oder zumindest Teile davon) aus einem fachlichen Modell über entsprechende Transformationen automatisch erstellt wird. Wir stellen in unseren Schulungen zu diesem Thema Konzepte, Tools und Best Practices vor, die in den letzten Jahren entstanden sind.

Themenauswahl

- Grundbegriffe der modellgetriebenen Softwareentwicklung (MDSD)
- Die MDA (Model Driven Architecture) der OMG
- Architekturzentriertes MDSD
- Metamodellierung
- Modelltransformationen
- Zielarchitekturen für modellgetriebene Entwicklung
- Codegenerierung
- UML Tools mit MDSD- und MDA-Unterstützung
- Domain Specific Languages
- Eclipse Modeling Framework
- Konfigurationsmanagement für modellgetriebene Softwareentwicklung
- Modellgetriebe Entwicklung für Embedded Systems
- Strategien zur Umsetzung von modellgetriebener Softwareentwicklung

Versionsmanagement mit Git

Git ist eine sehr leistungsstarke Versionsverwaltung, die sich durch hohe Performance und Flexibilität auszeichnet. Um Git erfolgreich einsetzen zu können, sollten Sie dessen grundlegende Konzepte und Mechanismen zu verstanden haben. Außerdem ist es notwendig, einen Workflow für die Arbeit mit Git zu definieren.

Unser Training stellt einen idealen Einstieg in die Arbeit mit Git dar. Anhand des Aufbaus von Git-Repositories erklären wir die Arbeitsweise von Git und diskutieren unterschiedliche Workflows und deren Einsatzmöglichkeiten für Ihr Team.

In der Schulung arbeiten wir im wesentlichen mit der Git-Kommandozeile, da wir zunächst ein grundlegendes Verständnis der Arbeitsweise dieses Tools benötigen. Die Arbeit mit einer grafischen Benutzeroberfläche fällt dann erfahrungsgemäß wesentlich leichter und es werden wenige Fehler gemacht.

Themenauswahl

- Git Grundbegriffe
- Aufbau von Repositories
- Die wichtigsten Git-Kommandos
- Tags und Branches
- Merge und Rebase
- Der einfache Git-Workflow
- Remote-Repositories
- Remote-Workflows
- Pull-Requests
- Git-Flow und seine Verwandten
- Nützliches und Wissenswertes
- Grafische Git-Tools
- Git-Backends

Softwareentwicklung und Implementierung

Mit unseren Schulungen aus dem Bereich *Softwareentwicklung und Implementierung* werden Sie in die Lage versetzt, die Programmiersprachen, die gängigen Bibliotheken und die jeweiligen Implementierungsumgebungen zu beherrschen.

Wir bieten Schulungen zu den Programmiersprachen C und C++ an, dabei können wir besonders auf deren Einsatz bei der Entwicklung von Embedded Systemen eingehen. Unsere Philosophie bzgl. der Inhalte folgt unseren eigenen langjährigen Erfahrungen als Softwareentwickler: beide Aspekte müssen durchdrungen werden – das wirkliche Verständnis der Konstrukte als auch die Tipps und Tricks im alltäglichen Umgang mit der Programmiersprache.

Modernes C++: Einführung

C++ ist eine der verbreitesten Programmiersprachen und wird sowohl in der Systemprogrammierung als auch in der Anwendungsprogrammierung eingesetzt. Gerade in der Entwicklung von Embedded Systems nimmt C++ als Programmiersprache eine Schlüsselstellung ein. Unter *modernem C++* versteht man die ab C++11 eingeführten Änderungen und Erweiterungen des C++-Standard, die es ermöglichen noch effektiveren und noch robusteren Code zu schreiben. Inzwischen stehen für alle geeigneten Microcontroller Compiler zur Verfügung, die modernes C++ unterstützen.

Unsere Themenauswahl richtet sich an Entwickler, die bisher keine oder wenig Erfahrung mit C++ haben, aber bereits Sprachen wie C, Java oder C# beherrschen. Wir können dabei auch intensiv auf die speziellen Aufgabenstellungen bei der Entwicklung von Embedded Systems in C++ eingehen. Besonderes Augenmerk liegt bei unseren C++-Schulungen darauf, den Teilnehmern ein Verständnis für das Laufzeit- und Speicherverhalten von C++-Anwendungen zu vermitteln. Die Schulung basiert auf dem aktuellen Sprachstandard C++17, kann auf Wunsch aber auch auf die älteren Standards C++14 oder C++11 eingehen.

Themenauswahl

- Objektorientierung mit C++
- Compiler und Linker
- C++-Kontrollstrukturen
- Variablen, Zeiger, Referenzen
- Einfache Objekte
- Klassen und Vererbung
- Überladen von Operatoren
- Namespaces und Aufbau von Programmen
- Fehlerbehandlung (Exceptions)
- Templates und deren Verwendung
- Automatische Typableitung mit auto und decltype
- RValues und die Move-Semantik
- Smart Pointer und Speicherverwaltung
- Lambdas und ihr Einsatz
- Multithreading
- Container und Iteratoren
- Algorithmen
- Ein- und Ausgabe mit Streams
- Type Traits
- Ausblick auf C++2a

Modernes C++: Update

C++ ist durch den neuen Standard C++11/14/17 umfassend erweitert worden, teilweise um sehr mächtige Sprachelemente. Alle gängigen C++-Compiler unterstützen mittlerweile den Standard. Gerade bei der Entwicklung von Embedded Systems lohnt sich der Umstieg auf modernes C++, da es weniger Fehlerquellen als früherer Standards enthält und außerdem erlaubt, performanteren und robusteren Code zu schreiben.

Diese Schulung richtet sich an erfahrene Entwickler, die mit den älteren C++-Standards (C++98 bzw. C++03) vertraut sind und auf modernes C++ umsteigen möchten. Zu allen Elementen wird mit Hilfe von Beispielcode Sinn und praktischer Einsatz diskutiert.

Die Schulung basiert auf dem aktuellen Sprachstandard C++17, kann auf Wunsch aber auch auf die älteren Standards C++14 oder C++11 eingehen.

Themenauswahl

- Templates und Typableitung
- Automatische Typableitung mit auto und decltype
- Neue und erweiterte Sprachelemente
- Smart Pointer
- RValues und die Move-Semantik
- Lambdas und ihr Einsatz
- Multithreading
- Zeit und Datum
- Neue Container
- Hash-basierte Container
- Type Traits
- Tupel und ihre Anwendung
- Weitere Neuerungen in der Standardbibliothek

Modernes C++: Tipps und Tricks

Diese Schulung behandelt gängige Sprachkonstrukte in C++ in größerer Tiefe und zeigt deren gegenseitige Abhängigkeiten, Gefahren und Möglichkeiten in der Praxis auf. Sauberer Klassenaufbau, sicheres und effektives Memory-Management und ein Überblick über die C++ Standardbibliothek helfen, schwer auffindbare Laufzeitfehler zu vermeiden.

Diskussionen über den richtigen Umgang mit Templates, Run Time Type Information, Smart Pointern und Exception Handling führen zum Beherrschen von Programmiertechniken, die Effizienz und Stabilität der Software steigern. Mit dieser Schulung vermitteln wir Ihnen Themen, die nach unserer Erfahrung schwer zugänglich sind und häufig nicht richtig verstanden werden.

Themenauswahl

- Konstruktoren und Destruktoren
- Spezielle Klassenfunktionen (Zuweisungsoperator, Move-Konstruktor etc.)
- Virtuelle Elementfunktionen
- Speicherverwaltung
- Smart Pointer (shared_ptr, unique_ptr etc.)
- Kapselung und Schnittstellen
- Werte- und Referenzsemantik
- Namespaces
- Vererbung
- ISO-Casts
- Exception Handling
- Exception Safety
- Templates und automatische Typableitung
- Perfect Forwarding

Modernes C++: Elemente für performante Embedded Systeme

Performanter Code hat in vielen Anwendungen höchste Priorität. Tatsächlich hält C++ viele Elemente bereit, um genau das zu erreichen, ohne faule Kompromisse bzgl. „sauberer" Objektorientierung machen zu müssen. In einigen Teams werden jedoch genau diese Elemente nicht oder ungeschickt verwendet und damit das Potential mit C++ nicht ausgeschöpft. Performance-Tuning kann auf vielen Ebenen der SW-Entwicklung angesetzt werden. Dabei spielen der Einsatz geeigneter Bibliotheken, der professionellen Anwendung moderner Bausteine fürs Multithreading u.v.m. eine Rolle.

In Embedded Systemen spielen effizientes (oft echtzeitfähiges), deterministisches Management des Hauptspeichers eine große Rolle. Statemachines und Multithreading sind in den meisten Systemen heute kaum wegzudenken. In der individuell zugeschnittenen Schulung zeigen wir, wie Sie die für ihre Systeme zentralen Implementierungsaspekte mit modernen, robusten C++ Elementen abdecken ohne Effizienzeinbußen befürchten zu müssen. Selbstverständlich nehmen wir auch Bezug auf die in Frage kommenden Elemente des aktuellen C++17-Standards. Außerdem werden Regeln vorgestellt, wie man objektorientierte Entwürfe explizit in die Implementierung überführt. Je nach Ausrichtung werden in dieser Schulung vor allem sprachkernnahe Aspekte rund ums C++ Objektmodell besprochen oder zusätzlich weitere Aspekte, die von diversen Bibliotheken unterstützt werden.

Themenauswahl

Best Practices zur Unterscheidung: Run-Time-Stack- oder Heap-Objekte einsetzen

Effiziente Übergabe von Objekten als Parameter und Rückgabe

Temporäre Objekte und Return-Value-Optimierung

Der Move Constructor und sein Einsatz zur Performanzsteigerung

Effizientes Management dynamisch allozierter Objekte mit deterministischem Speicher- und Laufzeitverhalten

Beziehungen zwischen Objekten und wie bildet man sie optimal in C++ ab

Smart Pointer und deren Speicher- und Laufzeitverhalten

Welche Smart Pointer eignen sich in Embedded Systems

Object Pooling

Performante Container-Klassen der STL und Boost und deren Laufzeitverhalten

Allokatoren

Überblick: Moderne C++-Building Blocks für die Multithreading-Programmierung

Coroutinen vs. Multithreading

Lock-free Algorithmen und Container

Performance-Tuning in Multithreaded-Anwendungen

Leichtgewichtige Komponentenmodelle

Implementierung von State Machines mit Boost und anderen bekannten Bibliotheken

Eventing: Callbacks und Ereignissysteme

Modernes C++: Multithreading (Teil 1)

Multithreading-Anwendungen eröffnen Möglichkeiten, die bei Singlethreaded-Anwendungen nicht zur Verfügung stehen. Die effektive Ausnutzung von Multi-Core-Architekturen ist eines der aktuell häufig diskutierten Beispiele. Allerdings sind die Anwendung von Design Patterns für nebenläufige Anwendungen, die Auswahl passender Bibliotheken und der professionelle Umgang mit Synchronisationsmitteln u.ä. im Entwickleralltag nicht einfach, oft wird die Komplexität der Aufgaben unterschätzt.

Möchte man Multi-Threading-Technologien mit modernen Prinzipien der SW-Architektur verbinden, kommen weitere anspruchsvolle Fragestellungen hinzu. Das sehr herausfordernde und komplexe Thema soll durch die Diskussion robuster und bewährter Building Blocks, insbesondere bekannter Library-Elemente und Design Patterns praxisgerecht vereinfacht werden.

Diese Schulung vermittelt zunächst das nötige Basiswissen rund um die Themen Thread-Erzeugung und einfache Synchronisationsmittel und geht dann auf die spezifischen Schwierigkeiten bei der Multithreading-Programmierung ein. Anschließend werden praxistaugliche Lösungsansätze erarbeitet, in denen höhere Mechanismen, wie das Active Object Pattern bevorzugt werden. Das Training wird durch viele Demonstrationen mit lauffähigem Code unterstützt, dabei kommen neben der C++-Standardbibliothek auch Teile der Boost-Bibliothek zum Einsatz.

Teilnehmervorausetzungen: Gute C++ Kenntnisse und Praxiserfahrung.

Themenauswahl

- Nebenläufigkeit, Parallelität, Multi-Processing, Multithreading
- Quasi-Parallelität und echte Parallelität, Multi-Core-Plattformen
- Vorteile von Multithreaded-Anwendungen
- Herausforderungen bei der Konstruktion von Multi-Threaded-Anwendungen
- Wichtige Grundlagen und Begriffe
- Taskbasierte Programmierung
- Multithreading in modernem C++
- C++-Multithreading und native Threading-Mechanismen
- C++ Synchronisations-Mechanismen
- Immutable Objects
- Monitor Object Pattern
- Strategized Locking Pattern
- Asynchroner Funktionsaufruf
- Einfache, zustandsorientierte aktive Objekte
- Prinzip der synchronisierten Produzenten/Konsumenten
- Synchronisierte Container

- Einfache Thread Pools und Message Scheduler
- Shutdown und Cancellation
- Verschiedene Ausprägungen von Task Schedulern
- Event Handling im Multithreading-Umfeld
- Asynchronous Completion Notification
- Future und Promise
- Active Object Pattern und seine Umsetzung
- Nebenläufigkeit in der Architektur
- Ausblick: Spezielle Bibliotheken für die effektive Nutzung von Multi-Core-Architekturen

Modernes C++: Multithreading (Teil 2)

Ausgehend von dem Basiswissen um die Besonderheiten des Multithreading und der grundlegenden Higher Level Building Blocks wird diskutiert, wie man Performance-Potentiale mit einfachen Mitteln ausschöpfen kann. Robustheit und Lesbarkeit sollen dabei nicht kompromittiert werden. Auf der technischen Ebene kommen auch stärker plattformabhängige Elemente zum Einsatz. Es wird großen Wert darauf gelegt, die Randbedingungen und Seiteneffekte der Bausteine zu diskutieren, um im Projekt die Risiken im Vorfeld besser einschätzen zu können.

Diese Schulung richtet sich an alle C++ Entwickler mit guten Grundkenntnissen der Mulitthreading-Programmierung und ist darüber hinaus eine ideale Ergänzung zu unserer Schulung Modernes C++: Multithreading (Teil 1)

Themenauswahl

- Multithreading: Vorteile, Nachteile und Mythen
- Vier Abstraktions- und Orientierungsebenen
- Die wichtigsten Elemente des Betriebssystems und des Prozessors in Bezug auf Performance und Robustheit
- Ein einfaches Modell einer Multi-Prozessor-Plattform
- Scheduling, Context Switch, Contention, Cache, Cache Lines, False Sharing
- C++ Memory Model
- Schwache und starke Thread Safety
- C++-Sprachkern und Standardbibliothek: Welche Elemente sind unter welchen Randbedingungen Thread-safe
- Vergleich wichtiger Mutex-Varianten
- Atomic Variables
- Thread-safe Container und Lock-free Container
- Tipps zur Reduzierung des Synchronisationsaufwands
- Ausgewählte Best Practices für das Design von Multithreading-Anwendungen
- Best Practices zum Schnittstellenentwurf
- Shared Data und Exclusive Usage
- Snapshots
- Optimistisches und pessimistisches Locken
- Higher Level Building Blocks und Performance
- Wie konfiguriert man Task Scheduler abhängig von Plattform und Timing Constraints
- Klassisches Pipelining und das Half Sync/Half Async Pattern
- Modernes Pipelining und das Leader-Followers Pattern

- Pipelinig mit der Intel TBB mit CPU- und I/O-intensiven Filtern
- Alternativen zum Multithreading

Modernes C++: Performante und robuste Anwendungen

Performanter Code hat in vielen Anwendungen hohe Priorität. Tatsächlich hält C++ viele Elemente bereit, um genau das zu erreichen. In einigen Teams werden jedoch diese Elemente nicht oder ungeschickt verwendet und damit das Potential mit C++ performante Anwendungen zu erstellen nicht ausgeschöpft.

Performance-Tuning kann auf vielen Ebenen der SW-Entwicklung angesetzt werden. Dabei spielen der Einsatz geeigneter Bibliotheken, die professionelle Anwendung moderner Bausteine u.v.m. eine Rolle. Man muss nur selten Kompromisse bzgl. modernem objektorientierten Entwurf und Robustheit machen, um eine gute Performance zu erreichen. Oft ergänzen sich die Maßnahmen für gute Performance, Robustheit und Lesbarkeit sogar gegenseitig. Doch wie erreicht man nun hohe Ablaufgeschwindigkeit und wenig Speicherverbrauch konkret? Und welchen Einfluss hat modernes C++ auf Performance, Robustheit und Umsetzung objektorientierter Entwürfe?

Themenauswahl

- Was ist Performance und was ist Robustheit?
- Vorgehensweise in der Projektpraxis: Performance und Robustheit sichern
- Modellebene und technische Ebene
- Präzise Klassenmodelle helfen Performance und Robustheit steigern
- Prinzip der Begrenzung des Geltungsbereichs und der Lebenszeit
- Schwache und starke Kapselung
- Whole-Part-Beziehungen
- Holder und DDD-Aggregate
- Value-, Entity- und finale Entity-Objekte
- Shared vs. Exclusive Ownership
- Shared vs. Exclusive Usage
- Transfer Value vs. Share Value
- Effektive Umsetzung präziser Klassenmodelle in C++
- RValue References und Move Semantics
- Schnittstellen gestalten, die Move Semantics berücksichtigen und damit Performance und Robustheit steigern
- Laufzeitverhalten, Speicherbedarf und Robustheit ausgewählter Elemente des Sprachkerns und der Standardbibliothek
- Return Value Optimierung vs. Move Semantics
- Container und Datenstrukturen in modernem C++
- Constraints des Modells versteh<ulen und den richtigen Container wählen

- Robuste Container und Iteratoren
- Ranges und Bulks als Parameter und Rückgaben
- Performance steigern auf technischer Ebene
- Container und Smart Pointer in modernem C++
- Performance, Robustheit und dynamische Speicherverwaltung
- Memory- und Object-Pooling
- Allocators
- Robustes Überladen von new und delete
- Dynamische Arrays auf dem Stack, Variable Length Arrays in C++17

Modernes C++: Standardbibliothek

Die C++ Standardbibliothek ist seit der Standardisierung Bestandteil der Programmiersprache C++. Sie bietet weitreichende Lösungen für Standardproblemstellungen in der C++ Programmierung.

Unser Schulungsangebot richtet sich an C++-Entwickler, die mit den Grundlagen von modernem C++ bekannt sind und deren Ziele es ist, einen tieferen Einblick in die Standardbibliothek zu erhalten um diese effektiver zu verwenden. Wir zeigen den richtigen Umgang mit den verschiedenen Teilen der Standardbibliothek und gehen intensiv auf das Laufzeitverhalten verschiedener Konstrukte ein.

Unsere Schulung beruht auf dem aktuellen Standard C++17, kann sich auf Wunsch aber auch an dessen Vorgängern C++14 oder C++11 orientieren.

Themenauswahl:

- Container und Containeradapter
- Assoziative Container
- Iteratoren und Iteratoradapter
- Speicherverwaltung
- Allokatoren und Speicherverwaltung mit Containern
- Algorithmen
- Strings und reguläre Ausdrücke
- Ein- und Ausgabe mit Streams
- Strings
- Zeit und Datum
- Lokalisierung
- Numerische Bibliotheken
- Bitsets und Valarrays
- Threads (Überblick)
- Ausblick auf C++2a

ANSI C für Embedded Systems

Die Entwicklung der Programmiersprache C begann schon in den 70er-Jahren. Seit 1990 liegt mit ANSI C (C89) ein ISO/IEC Standard vor, für den es Compiler auf praktisch jedem Microcontroller gibt. Gerade für die Entwicklung von Embedded Systems auf kleineren Microcontrollern ist C bis heute die wichtigste Programmiersprache. Aber auch die grundlegenden Programme aller Unix-Systeme und viele Betriebssystemkerne sind in C programmiert. Deshalb sind fundierte C-Kenntnisse in vielen Bereichen der Softwareentwicklung nach wie vor von großer Bedeutung.
Unser Schulungsangebot zum Thema ANSI C vermittelt umfassende Kenntnisse der strukturierten Programmierung insbesondere für die Entwicklung von Embedded Systems. Daneben vermitteln wir Ihnen Themen, in denen wir die Herangehensweisen und Techniken für Standardaufgaben aus dem Bereich der Embedded Programmierung behandeln.
Wir können uns je nach Vereinbarung in unserem Kurs sowohl auf den aktuellen Stanard C18 bzw. C11 als auch auf die älteren, immer noch weit verbreiteten Standards C99 oder C89 beziehen. Ebenso können wir auf Wunsch auch auf spezielle C-Compiler eingehen.

Themenauswahl

- Einführung und Übersicht
- Variablen, Zeiger und Felder
- Kontrollstrukturen
- Funktionen
- Benutzerdefinierte Datentypen
- Die Werkzeuge von ANSI-C
- Zeichenketten in C
- Ein- und Ausgabe und Dateizugriff
- Funktionen der Standardbibliothek
- Präprozessor
- Strukturierte Programmierung
- Programmiertechniken für Embedded Systems
- Grundbegriffe der Objektorientierung
- Objektorientierte Programmierung in C
- C11 im Vergleich zu C89 und C99

Embedded Systeme mit ANSI C programmieren

Die hier zusammengefassten Themen richten sich an Teilnehmer, die Erfahrung mit ANSI C haben und ihre Kenntnisse in der Programmierung auf Embedded Controllern vertiefen möchten. Es geht hier also weniger um die Sprache an sich als um Programmiertechniken, wie man sie auf modernen Mikrocontrollern benötigt.

Themenauswahl

- Hardwareabstraktion
- Embedded Betriebssysteme
- Eventgetriebene Systeme
- Zeitgetriebene Systeme
- Applikationslayer
- Implementierungsstrategien für Zustandsautomaten
- Programmflussüberwachung
- Speichertests
- Interruptprogrammierung
- Hardwareanbindung und Design von Treibern
- Unittests für ANSI C

Technologien und Libraries

In diesem Bereich sind Kurse zusammengefasst, die sich mit speziellen Bibliotheken oder Technologien befassen, die in modernen Entwicklungsprojekten benötigt werden.

Embedded Internet

Internet-Technologien spielen in Embedded Systems eine wachsende Rolle. Dabei kann es sich zum Beispiel um die Anbindung eines Systems an die lokale Infrastruktur mit Hilfe einer Ethernetschnittstelle handeln, die Konfiguration über Webseiten oder das Versenden von Status E-Mails.

Die von uns angebotene Themenauswahl umfasst alle wichtigen Aspekte, die üblicherweise benötigt werden, wenn es darum geht, Internet-Technologien in Embedded Systems zu integrieren.

Themenauswahl

- Das OSI-Modell
- DHCP
- DNS
- Das Routing
- Single-, Broad- und Multicast
- Das Transmission Control Protocol (TCP)
- Das User Datagram Protocol (UDP)
- IPv4 vs. IPv6
- Security auf Vermittlungs- und Transportschicht
- Masquerading
- HTTP
- HTTPS
- SMTP und ESMTP
- Virtuelle Netze und VPN
- Wireless LAN
- Microcontroller für Ethernetanschluss
- Stecker für Embedded Internet

C++ Programmierung mit Boost

Boost ist eine freie und portable C++ Bibliothek und stellt eine Sammlung von Unterbibliotheken für unterschiedliche Aufgaben zur Verfügung. Ein Review-Prozess durch die Boost Community stellt sicher, dass alle Unterbibliotheken einem hohen Qualitätsstandard entsprechen und ihre Lizenzbedingungen dem Anspruch einer Nutzung auch solchen Softwareprojekten genügen, die keiner Open-Source-Lizenz unterliegen. Der 2006 verabschiedete Technical Report 1 bezwiehungsweise C++11/14/17 erweitern die C++ Standardbibliothek um Teile aus Boost.
Unser Schulungsangebot rund um Boost umfasst die einzelnen Teilbibliotheken. Wir stellen daraus Schulungen nach Ihren individuellen Bedürfnissen zusammen.

Themenauswahl

- Ergänzung fehlender Elemente der Standardbibliothek
- Bearbeitung von Strings
- Pattern Matching
- Parsing
- Spezielle Container
- Multi-Threading
- Coroutines und Fibers
- Statemachines
- Networking
- Verschiedenes

POCO: Einführung in das Application Framework

POCO (pocoproject.org) ist ein Open Source Projekt, mit dessen Elementen sehr elegant typische Implementierungsaspekte einer Anwendung umgesetzt werden können. Struktur und Footprint des Frameworks passen zu Embedded Systems. Aus unserer Sicht gehört POCO zu den besten Application Frameworks im C++ Umfeld und wir haben damit in vielen Embedded Projekten positive Erfahrung gesammelt.
In dieser Schulung werden die Kernelemente von POCO anhand von Beispielen diskutiert und weiterführende Tipps gegeben.

Themenauswahl

- Überblick über POCO und seine durchgängigen Konzepte
- Speicherverwaltung mit POCO, Smart Pointers und Pooling
- Ereignissystem
- Loggingsystem
- Shared Libraries und dynamisches Laden
- Networking
- Multi-Threading
- Datenbank-Zugriff

Einführung in die GUI-Programmierung mit Qt 5

Soweit es die GUI-Entwicklung im C++ Umfeld betrifft, kann man ohne weiteres behaupten: Qt ist Defacto-Standard. Qt bietet zahlreiche Tools und Bibliothekselemente zur GUI-Gestaltung.
In dieser Schulung geht es darum, eine fundierte Grundlage zu schaffen und auch hinter die Kulissen zu schauen.

Themenauswahl

- Grundlagen der Qt-Klassenbibliothek
- Widgets, Objekthierarchien, Speicherverwaltung und Q
- Object
- Ereignisbehandlung: Signal/Slot-Mechanismus
- Qt-Tools: Designer, Assistant, uic, moc, Class Wizard
- Dialoge erstellen
- Layout-Manager
- Wiederverwendung von GUI-Elementen (Look & Feel)
- Hauptfenster gestalten: Menüs, Toolbars, Shortcuts
- Ereignisbehandlung: Primitive Events
- Direkte Überprüfung während der Eingabe: Validatoren
- Mehrsprachige Anwendungen mit Qt entwickeln
- Anbindung der GUI an den Anwendungskern mit Qts Model/View-System

Qt 5 als Application Framework

Viele C++ Experten ordnen die weitverbreitete C++ Bibliothek Qt hauptsächlich als GUI-Framework für PC-Anwendungen ein. Tatsächlich handelt es sich um ein ausgewachsenes System, das Bausteine für die Implementierung nahezu aller zentralen Aspekte moderner Anwendungen bereit stellt. Es ist gut modularisiert und effizient genug um auch in kleineren (Embedded) Systemen Anwendung zu finden. In der Schulung werden auf Ihre Bedürfnisse maßgeschneiderte Implementierungsaspekte jenseits des GUI-Teils diskutiert und aufgezeigt, wie sie mit Qt umgesetzt werden können.

Themenauswahl

- Das Qt-Objektmodell
- Speicherverwaltung in Qt
- Leichtgewichtige Komponentenmodelle mit Qt
- Signal/Slot-Ereignissystem
- Properties
- Statemachines
- Networking
- Multi-Threading
- Parallelität mit Multi-Threading und Event-Loop
- Datenbankzugriff

Reaktionsschnelle, parallele GUI-Anwendungen mit Qt

Diese Schulung richtet sich an alle C++ Programmierer, die nach einem einfachen Architekturschema für GUI-Programme mit Multi-Threading-Anteilen suchen. Der Schwerpunkt liegt weniger auf allgemeine Architekturkonzepte, hier geht es um „Kochrezepte", die direkt in der Praxis umsetzbar sind! Anhand einer kleinen Beispielanwendung mit C++/Qt, die Schritt für Schritt erweitert wird, werden für typische Problemfälle aus dem Bereich "Multi-Threading-Anwendung mit GUI-Anbindung" Lösungen diskutiert. Ziel ist es, ein allgemeines Schema im Rahmen einer klassischen Drei-Schichten-Architektur für Multi-Threaded-Anwendungen mit GUI zu skizzieren, das für ein weites Feld von Applikationen tragfähig ist.

Themenauswahl

- Top-Level Komponenten und deren Responsibilities in einer Drei-Schichten-Architektur
- Soll der GUI-Teil einer Anwendung Multi-Threaded oder Single-Threaded sein?
- Wie bringt man ereignisbasierte Systemteile mit Multi-Threaded-Teilen zusammen
- An welchen Stellen in der Architektur baut man auf synchrone, an welchen auf asynchrone Kommunikation?
- Wie werden nicht-modale Dialoge typischerweise eingebunden?
- Wie geht man mit langlaufenden Diensten bzw. Dauerläufern um?
- An welchen Stellen in der Architektur und mit welchen Mitteln entkoppelt man Threads?
- Wie wird die Reaktivität des gesamten Systems erhalten?
- Wie kann man den Shutdown des Gesamtsystems realisieren?

Die Bibliotheken Boost, ACE, POCO und Qt im direkten Vergleich

Ausgehend von zentralen Implementierungsaspekten, wie Networking, Multi-Threading, Eventing, Speicher- und Objektverwaltung werden weit verbreitete C++ Frameworks und Bibliotheken gegenüber gestellt. Dabei diskutieren wir intensiv, inwiefern sie sich für die effiziente Anwendungsentwicklung eignen.

Themenauswahl

- Implementierungsaspekte typischer Anwendungen
- Vordefinierte Datentypen
- Error und Exception Handling
- Logging und Debugging
- Objektmodell und Memory Management
- Zugriff auf systemnahe Dienste
- Callbacks und Ereignissystem
- Komponentenmodell, Shared Libraries
- Konfiguration
- Multi-Threading: Primitives und Higher Level Building Blocks
- XML
- Netzwerkprogrammierung
- Datenbankprogrammierung
- GUI

Test und Qualitätsmanagement

Hohe Qualität eines Produkts und der Entwicklung selbst setzt vielfältige Maßnahmen an den richtigen Stellen im gesamten Vorgehen voraus.
Daher spielt die Fragestellung der Qualität in allen Phasen der Entwicklung und dementsprechend in all unseren Themenbereichen eine wichtige Rolle.
Wir haben darüber hinaus einen eigenen Themenbereich Test- und Qualitätsmangement eingerichtet, in dem wir uns vor allem dem Aspekt des System- und Softwaretests widmen.

Testgetriebene Entwicklung

Unter einer testgetriebenen Entwicklung versteht man ein Vorgehen, bei dem parallel zum eigentlichen Programmcode Tests für diesen Code entwickelt werden. Dabei kommen Unit-Test-Tools zum Einsatz, die automatisierte Modultests durchführen und ursprünglich auf das SUnit-Konzept in Smalltalk zurückgehen. Inzwischen gibt es neben dem bekannten JUnit für Java für fast alle Programmierumgebungen entsprechende xUnit-Tools.

Unsere Kurse rund um die testgetriebene Entwicklung führen sowohl in die Konzepte als auch in gängige Tools ein. WIr möchten zeigen, wie man den Ansatz der testgetriebenen Entwicklung in das gesamte Vorgehensmodell (Requirements-Management, Entwicklung usw.) harmonisch einbettet. Das Testkonzept, die Tools und die Methoden müssen sich am übergeordneten Qualitätsmanagement orientieren. Wir bieten Schulungen mit verschiedenen Unit-Test-Tools und unterschiedlichen Schwerpunkten an.

Testgetriebe Entwicklung wird meistenteils auf der Ebene des Modultests angesetzt. Zum besseren Verständnis und zur Abgrenzung von Testverfahren auf anderen Ebenen lohnt es sich, ausgewählte Themen aus dem Themenfeld Testmethoden in der Systementwicklung mit in die Schulung zu integrieren.

Themenauswahl

- Idee und Motivation der testgetriebenen Entwicklung
- xUnit-Test-Frameworks
- Unit-Test-Frameworks für C++
- Einsatz der Frameworks im Entwickler- bzw. Testteam
- Vorgehen beim testgetriebenen Entwickeln
- Testcode erstellen
- Geforderte Testabdeckungen einhalten
- Refactoring von Testcode
- Verifikation von Testresultaten
- Testorganisation
- Probleme beim Einsatz testgetriebener Entwicklung

Testmethoden in der Systementwicklung

Strukturiertes und systematisches Testen wird von allen Beteiligten eines Softwareprojekts und in der Systementwicklung allgemein als unerlässlicher Bestandteil anerkannt. In der Praxis ergibt sich leider ein gemischtes Bild. Oft wird unvollständig und unsystematisch getestet. Gelegentlich wird zwar umfangreich getestet, der Schwerpunkt im Testkonzept jedoch falsch gewählt und damit der Entwicklungsprozess behindert.

Wir bieten verschiedene Schulungen in diesem Bereich an, wobei die variierenden Rollen (Projektleitung, Entwicklungsteam, Testteam) berücksichtigt werden. Besonderen Wert legen wir dabei auf praxisnahes Vorgehen.

Themenauswahl

- Grundlagen des strukturierten und systematischen Prüfens und Testens
- Bedeutung und Rolle des Testens im Systementwicklungsprozess
- Requirements-Management, Softwarequalität und Systemtest
- Testkonzept und Teststrategie
- Testplan und Testfall
- Testabdeckung
- Entwicklungsteam, Testteam und Testumgebung
- Automatische Tests und Regressionstests
- Das V-Modell und seine Teststufen
- Statischer und dynamischer Test
- Testen im agilen Umfeld
- Besonderheiten beim Test objektorientierter Software
- Testen von Embedded Systems und mechatronischer Systeme
- Testwerkzeuge

Testautomatisierung für Embedded Systeme

Der Testaufwand nimmt bei der Entwicklung von Embedded Systemen durch die steigende Komplexität und Vernetzung der Systeme einen immer größeren Stellenwert ein. Angesichts der zunehmend kürzeren Lebenszyklen elektronischer Produkte ist ein hoher Automatisierungsgrad beim Test sehr wichtig.

Unsere Themenvorschläge in diesem Bereich beinhalten neben Standardverfahren auch HIL-Produkte (HIL = Hardware in the Loop) und -Tools, die bei der Testautomatisierung eingesetzt werden können.

Themenauswahl

- Blackbox-Testverfahren in der Systementwicklung
- Whitebox-Testverfahren in der Systementwicklung
- Graybox-Tests in der Systementwicklung
- Model in the Loop
- Software in the Loop
- Processor in the Loop
- Hardware in the Loop
- HIL-Produkte
- Testwerkzeuge
- Testautomatisierung mit Matlab-Simulink
- Testautomatisierung mit dSpace
- Testautomatisierung mit LabVIEW