maurer treutner

Trainingskatalog

Übersicht und Themenauswahl

Stand: 2023

Inhaltsverzeichnis

Unsere Trainings

Um möglichst viele Aspekte der modernen Software- und Systementwicklung abzudecken, richten wir unser Trainingsangebot an den Phasen aus, die mechatronische, Embedded bzw. reine Softwareprojekte gewöhnlich durchlaufen.

Wir wenden uns in erster Linie an Unternehmen, die neue Technologien einführen wollen oder bei bereits eingeführten Technologien das vorhandene Wissen ihrer Mitarbeiter gezielt vertiefen möchten.

Sie können unsere Trainings buchen als

- Online-Veranstaltung über alle gängigen Videokonferenzsysteme – ganz nach den zeitlichen Bedürfnissen Ihres Teams
- mehrtägige Veranstaltung in Ihrem Haus

Das Training wird individuell für Sie zusammengestellt und mit Praxisanwendungen durchgeführt, die auch aktuelle Fragestellungen aus Ihren Projekten berücksichtigen können.

Wir freuen uns auf Ihre Kontaktaufnahme!

Methoden und Entwicklungsprozess

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

Auf agilen Methoden und deren Einsatz bei 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
- Card Walls 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 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 Management 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
- 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

Requirementsmanagement

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.

Beim Thema Requirementsmanagement 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 Requirementsmanagements genau auf das Projekt und dessen Rand-bedingungen 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
- Requirementsengineering und -management als Teil des Vorgehensmodells
- Requirementstracing und -tracking
- Strukturierungsgrad von Anforderungsdokumenten und Beziehung zu anderen Systemmodellen
- Qualitätssicherung, Verifikation und Test
- Requirementsmanagement in agilen Methoden
- Einfluss von verbindlichen Normen auf das Requirementsmanagement
- Tools für das Requirementsmanagement

Systemmodellierung und Architektur

Die Software- und Systemarchitektur hat maßgeblichen Einfluss auf den Erfolg von Soft- und Hardwareprodukten. 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 ist ein allgemein anerkannter Standard in der Softwareentwicklung. 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 ausdrucksstarke Sprache für die Systemmodellierung 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 2 Grundlagen

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

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 2

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
- Requirementsmodellierung 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

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

Softwareentwicklung und Implementierung

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

Unser Trainingsprogramm dreht sich um die Sprachen C und C++. 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.

Embedded Systeme mit ANSI C programmieren

Diese 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
- Applikations-Layer
- Implementierungsstrategien für Zustandsautomaten
- Programmflussüberwachung
- Speichertests
- Interrupt-Programmierung
- Hardwareanbindung und Design von Treibern
- Unittests für ANSI C

C++ Grundlagen

C++ ist eine der am meisten verbreiteten objektorientierten Programmiersprachen. C++ wird sowohl in der Systemprogrammierung als auch in der Anwendungsprogrammierung eingesetzt. Die Möglichkeit, effiziente, maschinennahe Programmierung mit den mächtigen Sprachmitteln objekt-orientierter und generischer Programmierung zu kombinieren, macht C++ für viele Anwendungsbereiche interessant.

Unsere Themenauswahl zu C++ Grundlagen richtet sich an Entwickler, die bisher keine oder nur geringe Erfahrung mit C++ sammeln konnten. Meist haben Entwickler bereits einen recht fundierten C-Hintergrund, bevor sie nach C++ wechseln, oder haben bereits mit Java oder C# Erfahrungen gesammelt.

Alle wichtigen Elemente des aktuellen Standards C++20 werden vorgestellt.

Themenauswahl

- C++ im Überblick: Historie, Versionen
- Objektorientierte Programmierung
- Variablen, Zeiger, Referenzen
- Einfache Objekte
- Klassen
- Operationen
- Vererbung
- Überladen von Operatoren
- Struktur von Programmen
- Namespaces
- Dynamische Speicherverwaltung und Smart Pointers
- Fehlerbehandlung (Exceptions)
- Parametrisierte Klassen und Funktionen (Templates)
- Die Standardbibliothek
- Laufzeit- und Speicherverhalten von C++
- Programmiertechniken für Embedded Systems

C++ Grundlagen für Embedded Entwickler

Gerade in der Entwicklung von Embedded Systems nimmt C++ als Programmiersprache eine Schlüsselstellung ein. Für viele Microcontroller stehen neben C-Compilern heute auch betriebsbewährte C++ Compiler zur Verfügung. Mit steigender Komplexität der Systeme lohnt sich der Umstieg auf C++.

Wir bieten hier eine Themenauswahl, die speziell auf die Bedürfnisse im Umfeld der Entwicklung von Embedded Systems in C++ ausgerichtet ist. Mit Beispielen und Übungen, die sich auf typische Anwendungsfälle aus dem Embedded Bereich beziehen, gehen wir auf die für dieses Umfeld typischen Fragestellungen ein.

In der Schulung behandeln wir die modernen Sprachstandards C++14, 17 und 20.

Themenauswahl

- C++ im Überblick: Historie, Versionen
- Objektorientierte Programmierung
- Variablen, Zeiger, Referenzen
- Einfache Objekte
- Klassen und Vererbung
- Überladen von Operatoren
- Namespaces und Aufbau von Programmen
- Dynamische Speicherverwaltung und Smart Pointers im Embedded Systems
- Fehlerbehandlung (Exceptions)
- Parametrisierte Klassen und Funktionen (Templates)
- Die Standardbibliothek
- Kombination von C- und C++ Code und Bibliotheken
- Laufzeit- und Speicherverhalten von C++
- Programmiertechniken für Embedded Systems

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
- Effizienz und Laufzeitverhalten
- Einführung in die STL und Boost

Effiziente C++ Entwicklung: Elemente für hochperformante 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++ performante Anwendungen zu erstellen nicht ausgeschöpft. Performance-Tuning kann auf vielen Ebenen der SW-Entwicklung angesetzt werden. Dabei spielt der Einsatz geeigneter Bibliotheken, der professionellen Anwendung moderner Bausteine fürs Multi-Threading u.v.m. eine Rolle.

Diese Schulung richtet sich an Entwickler, die hauptsächlich im Embedded Systems Umfeld arbeiten.

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 C++11 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
- Allocators
- 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

Einführung in die Multithreading-Programmierung mit C++

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 im Entwickleralltag nicht einfach, oft wird die Komplexität der Aufgaben unterschätzt.

Teilnehmervorausetzungen: Gute C++ Kenntnisse und Praxiserfahrung. Grundkenntnisse im Themenfeld Multi-Threading sind wünschenswert.

Themenauswahl

- Die Begriffe Nebenläufigkeit, Parallelität, Multiprocessing, Multithreading
- Quasi-Parallelität und echte Parallelität, Multi-Core-Plattformen
- Vorteile von Multithreaded-Anwendungen
- Herausforderungen bei der Konstruktion von Multithreaded-Anwendungen
- Wichtige Grundlagen und Begriffe
- Überblick über gängige Mechanismen im Multithreading-Umfeld
- Multithreading in C++20
- Überblick und Gegenüberstellung: Bekannte Bibliotheken für die Multithreading-Programmierung im C++ Umfeld, insbesondere Boost, Qt und POCO
- Einfache Synchronisations-Primitive, z.B. Mutex und Condition Variable
- 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
- Shut down und Cancellation
- Verschiedene Ausprägungen von Task Schedulern
- Event-Handling im Multithreading-Umfeld
- Asynchronous Completion Notification
- Future Pattern

- Active Object Pattern
- Generierung von Elementen für das Active Object Pattern mit C++ Bordmitteln
- Überblick: Weitere Synchronisationsmittel
- Berücksichtigung von Multi-Threading in der Architekturarbeit
- Welche Building Blocks sind in welchen Szenarien sinnvoll: Ausbildung, neues Team, Architekturarbeit
- Ausblick: Spezielle Techniken und Bibliotheken für die effektive Nutzung von Multi-Core-Architekturen

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 Qualitätsmanagement eingerichtet, in dem wir uns vor allem den Aspekten 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-Testtools 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. Uns ist es wichtig zu zeigen, wie man den Ansatz der testgetriebenen Entwicklung in das gesamte Vorgehensmodell (Requirementsmanagement, Entwicklung usw.) harmonisch einbettet. Das Testkonzept, die Tools und die Methoden müssen sich am übergeordneten Qualitätsmanagement orientieren. Wir bieten Schulungen mit verschiedenen Unit Testtools 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 Testframeworks
- 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
- JUnit
- CppUnit
- NUnit
- XMLUnit

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
- Requirementsmanagement, 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