Seminare

Stand 10. Mai 2014

www.oop-trainer.de

Ralf Schneeweiß
Gölzstraße 8
72072 Tübingen
Tel: +49 (0)179 2292471
Mail: info@oop-trainer.de

Inhaltsverzeichnis

Kurs: die Programmiersprache ANSI/ISO C

In diesem Seminar wird die Programmiersprache ANSI-C1) anhand von praktischen Beispielen unterrichtet. Dabei wird auf den ANSI bzw. ISO Standard von 1989/1990 zurückgegriffen, der auf praktisch allen Plattformen zur Verfügung steht die C unterstützen. Auch der ANSI/ISO Standard von 1999 wird durchgesprochen und in seinen Unterschieden zum ersten diskutiert. Die Unterschiede zwischen älteren Varianten von C - wie z.B. K&R C oder die frühe UNIX Variante - zu ANSI-C werden auch behandelt. Das günstige Laufzeitverhalten und die einfache Portierung eines C-Compilers auf neue Plattformen rechtfertigen auch heute noch den Einsatz von C. Aus den genannten Gründen wird ANSI-C auch häufig bei der Programmierung von Embedded Systems verwendet.

Die folgenden Themen werden durchgenommen:

Präprozessoranweisungen
Konstanten
Variablen
Das Lokalitätsprinzip
Datentypen
Anweisungen
Kontrullstrukturen
Funktionen
Die prozedurale Programmierung
Funktionsprototypen
Arithmetische Ausdrücke
Die arithmetischen Operatoren
Bitmanipulationen
Bitoperatoren
Zeiger
Strukturen
Aufzählungstypen
Unions
Funktionszeiger
...

Die Standardbibliothek
I/O Operationen
Dateizugriffe
Dynamische Speicherallokation
Speicherklassen in C
Bedeutung von Heap und Stack
Externe und statische Linkbarkeit
Aufbau von C - Projekten
Das Zusammenspiel von Compiler und Linker
Object Files und Bibliotheken
Einfache Makefiles
Hardwarenahe Programmierung
Probleme bei nebenläufiger Programmierung
Entwicklung performanter Software
Codeoptimierung
Footprint und Laufzeitverhalten

Es kann ein beliebiger Standard C oder C++ Compiler eingesetzt werden, der ANSI/ISO kompatibel ist. Als Plattformen für den Kurs empfehlen sich Windows-, Mac OS X-, UNIX- oder Linuxsysteme. Wichtig ist dabei nur, daß die Kursteilnehmer einfache Dateioperationen auf dem eingesetzten System beherrschen. Das Seminar kann auch an bestimmte thematische Anforderungen angepasst werden und mit einem bestimmten, für Sie wichtigen Gesichtspunkt ausgestattet werden. So kann beispielsweise ANSI-C vor dem Hintergrund der Anwendungsentwicklung behandelt werden, oder auch im Zusammenhang mit der Systementwicklung in Embedded Systemen.

1) Im folgenden wird der Einfachheit halber die Bezeichnung ANSI-C verwendet, obwohl korrekterweise auch ISO-C oder ANSI/ISO-C verwendet werden könnte. ISO bezeichnet die internationale Norm ISO/IEC 9899:1990, während ANSI die amerikanische ANSI X3.159-1989 bezeichnet auf der die genannte ISO Norm basiert. Der Gebrauch der Bezeichnung ANSI-C hat sich in der überwiegend englisch-sprachigen Literatur im Umfeld der praktischen Softwareentwicklung ohnehin längst durchgesetzt.

Einführungskurs ANSI/ISO C++

C++ ist eine Programmiersprache die komplexe Softwareentwürfe ermöglicht und mit Laufzeiteffizienz verbindet. Die sprachlichen Strukturen sind einerseits ausdrucksstark andererseits schwer zu erlernen, da die vielen Veränderungen der letzten Jahre vor dem Standardisierungsprozess die Sprache stark gewandelt haben. Waren es bis Anfang der neunziger Jahre noch die objektorientierten Eigenschaften der Sprache, die ihren Kern ausmachten, so sind es inzwischen die Konzepte der generativen Programmierung, die sich gleichberechtigt neben die OO-Sprachkonstrukte stellen. Im Kurs wird auch ein Überblick über diese Entwicklung gegeben und es werden die Hintergründe der genannten Techniken angesprochen.
Dieses Seminar befasst sich vor allem mit den Grundlagen der objektorientierten Programmierung mit ANSI/ISO C++. Dabei ist für diesen C++ Kurs die Kenntnis einer höheren Programmiersprache hilfreich aber nicht zwingend erforderlich.

Themen:

Historische Entwicklung von C++
Grundlegende Syntax
Erste Beispiele
Kennenlernen des Compilers
und des Linkers
Die Anteile der Sprache C in C++
Standarddatentypen
Strukturen
Operatoren
Funktionen
Zeiger und Referenzen
Dynamische Speicherallokation
Klassenkonzept
Datenkapselung
Attribute und Methoden
Instantiierung von Objekten
Konstruktoren und Destruktoren
Funktionsüberladung
Operatorüberladung
Typenkonvertierung
Sichtbarkeitskonzept
Vererbung
Polymorphie
Virtuelle Methoden
Abstrakte Klassen
Mehrfachvererbung
Exception Handling
Templates
Funktionstemplates
Klassentemplates
Die Objektorientierte Programmierung
Vergleich der Sprachen C und C++
Prinzipien der OO-Modellierung
Der Musterbegriff
Sinn und Unsinn von Klassenhierarchien
Diskussion der Abstraktionsebene der
OOP
Einsatzbereiche von C++
Betrachtungen zum Laufzeitverhalten
Sprachstandard und Compilerverhalten

Die Programmiersprache wird anhand praktischer Beispiele erlernt. Dabei kann ein beliebiger C++ Compiler eingesetzt werden. Vorzugsweise werden GNU C++, MS Visual C++ oder Borland C++ verwendet. Der Compiler sollte nicht zu alt sein und die wesentlichen Sprachmerkmale von ANSI/ISO C++ bezüglich Namespaces und Namenskonventionen unterstützen. Auf ausdrücklichen Wunsch kann auch ein älterer C++ Sprachstandard wie z. B. AT&T 2.x oder AT&T 3.x unterrichtet werden. Wichtig ist natürlich der Einsatz einer Plattform, die den Teilnehmern gut bekannt ist - vorzugsweise Windows, Mac OS X oder Linux. Natürlich kann auch jedes beliebige UNIX-System als Entwicklungs- und Übungsplattform für die Schulung eingesetzt werden.

Aufbaukurs ANSI/ISO C++

Aufbauend auf einem Grundverständnis von C++ wird in diesem Kurs in die weiterführenden Konzepte von ANSI/ISO C++ eingeführt, die in älteren Standards nicht oder nur unvollkommen realisiert waren. Dabei werden insbesondere die Techniken der Standardbibliothek und der generativen Programmierung betrachtet. Die Templateprogrammierung als wesentliche Erweiterung von ANSI/ISO C++ gegenüber älteren C++-Dialekten führte zu einem Paradigmenwechsel in der Architektur der Standardbibliothek. Sie lässt diesen Paradigmenwechsel auch für eigene Architekturen zu und eröffnet ganz neue Anwendungsbereiche und Möglichkeiten, die Sprache in Projekten einzusetzen. Es werden der "alte" und der "neue" Teil von C++ voneinander abgegrenzt und zueinander in Beziehung gebracht. Außerdem werden Chancen und Konsequenzen des Einsatzes verschiedener moderner Techniken an praktischen Beispielen demonstriert.

Themen:

Überblick über die Standardbibliothek
Namespaces und die using Direktive
Der ANSI-C++ Standard
Streamklassen (Methoden, Operationen und Manipulatoren)
Strings
Der Copy-Konstruktor
Virtuelle Destruktoren
Reimplementieren
Statische Elemente
Statische Methoden
Verschachtelte Klassen
Vererbung unterschiedlicher Sichtbarkeit
Virtuelle Vererbung
Scopes
V-Tables und RTTI
Programmierung mit Templates
Generische Programmierung
Die Standard Template Library (STL)
Container im Allgemeinen
Listen
Vektoren
Sets
Multisets
Iteratoren
Algorithmen
Funktionsobjekte
Idiome in C++
Der Einsatz von Exception Handling
Wichtige C++-Literatur

Die Programmiersprache wird anhand praktischer Beispiele erlernt. Dabei kann ein beliebiger C++ Compiler eingesetzt werden. Vorzugsweise werden GNU C++, MS Visual C++ oder Borland C++ verwendet. Sehr wichtig für diesen Kurs ist ein Compiler, der die wesentlichen Sprachmerkmale von ANSI/ISO C++ bezüglich Namespaces und Namenskonventionen unterstützen. Die Entwicklungsplattform - vorzugsweise Windows, Mac OS X, Linux oder UNIX - sollte den Entwicklern gut bekannt sein.
Neben den oben genannten C++-Themen werden auch aktuelle Publikationen zu ANSI/ISO C++ besprochen.

ANSI/ISO C++ Umsteigerkurs für C-Programmierer

In diesem C++ Kurs wird eine Einführung in die objektorientierte Programmierung gegeben und der prozeduralen Programmierung mit C gegenübergestellt. Die Gemeinsamkeiten und die Unterschiede der Programmiersprachen C und C++ sind ein Hauptgegenstand des Kurses. Da C++ auf dem C-Speichermodell basiert und der Einsatz des Compilers und Linkers im wesentlichen gleich sind, wird auf dem vorhandenen Vorwissen von C Entwicklern aufgebaut, um sich den Sprachstrukturen von ANSI/ISO C++ zuzuwenden.
ANSI/ISO C++ ist eine Programmiersprache die komplexe Softwareentwürfe ermöglicht und mit der Laufzeiteffizienz von C verbindet. Die sprachlichen Strukturen sind einerseits ausdrucksstark andererseits schwer zu erlernen, da die vielen Veränderungen der letzten Jahre vor dem Standardisierungsprozess die Sprache stark gewandelt haben. Waren es bis Anfang der neunziger Jahre noch die objektorientierten Eigenschaften der Sprache, die ihren Kern ausmachten, so sind es inzwischen die Konzepte der generativen Programmierung, die sich gleichberechtigt neben die OO-Sprachkonstrukte stellen. Im Kurs wird auch ein Überblick über diese Entwicklung gegeben und es werden die Hintergründe der genannten Techniken angesprochen.
Dieses Seminar befasst sich vor allem mit den Grundlagen der objektorientierten Programmierung mit ANSI/ISO C++. Dabei ist für den Kurs die Kenntnis der Programmiersprache C erforderlich.

Historische Entwicklung von C++
Grundlegende Syntax im Vergleich zu C
Erste Beispiele
Die Anteile der Sprache C in C++
Funktionsüberladung
Operatorüberladung
Typenkonvertierung
Zeiger und Referenzen
Datenkapselung
Klassenkonzept
Attribute und Methoden
Sichtbarkeitskonzept
Operatoren
Instantiierung von Objekten
Konstruktoren und Destruktoren
Dynamische Speicherallokation
malloc() und free() versus new und delete
Vererbung
Virtuelle Methoden
Abstrakte Klassen
Polymorphie
Mehrfachvererbung
Exception Handling
Templates
Funktionstemplates
Klassentemplates
Die Objektorientierte Programmierung
Vergleich der Sprachen C und C++
Prinzipien der OO-Modellierung
Der Musterbegriff
Sinn und Unsinn von Klassenhierarchien
Diskussion der Abstraktionsebene der OOP
Einsatzbereiche von C++
Betrachtungen zum Laufzeitverhalten
Sprachstandard und Compilerverhalten

Die Programmiersprache wird anhand praktischer Beispiele erlernt. Dabei kann ein beliebiger C++ Compiler eingesetzt werden. Vorzugsweise werden GNU C++, MS Visual C++ oder Borland C++ verwendet. Der Compiler sollte nicht zu alt sein und die wesentlichen Sprachmerkmale von ANSI/ISO C++ bezüglich Namespaces und Namenskonventionen unterstützen. Wichtig ist natürlich der Einsatz einer Plattform, die den Teilnehmern gut bekannt ist - vorzugsweise Windows, Mac OS X oder Linux. Natürlich kann auch jedes beliebige UNIX-System als Entwicklungs- und Übungsplattform eingesetzt werden.

Der neue Standard ISO C++11

Aufbauend auf einem Grundverständnis von C++ wird in diesem Kurs der neue Standard ISO/IEC 14882:2011 (vereinfacht ISO C++11 – vormals auch als C++0x bezeichnet) dieser Programmiersprache vorgestellt. Dabei werden die Änderungen von C++11 gegenüber dem alten Standard ISO C++98 und dem korrigierten Standard ISO C++03 besprochen. Soweit dies schon möglich ist und eine ensprechende Compilerunterstützung bereits existiert, werden Beispiele des Standards anhand lauffähiger Codes demonstriert.

- Neue Schlüsselwörter für den Umgang mit Typen und Deklarationen:
  - auto
  - alignas
  - alignof
  - decltype
- Alignment
- Das Schlüsselwort nullptr
- Neue Typen
- Stark typisierte enums
- Nicht eingeschränkte Unions
- Nachfolgende Definition von Rückgabetypen
- Unicode Unterstützung
- Unicode Typen
- Prefixes
- Strings
- Die neuen Möglichkeiten der using-Direktive
- Template aliases
- Namespace Assoziation
- Neue Initialisierungssyntax
- Initialisierungssequenzen:
  - Der Container std::initializer_list<T>
  - Konstruktoren mit Wertelisten
- Der delegierende Konstruktor
- Kopierbarkeit und Bewegbarkeit von Objekten
- R-Value Referenz
- Konstante Ausdrücke
- Das neue Schlüsselwort constexpr
- Lambdas
- Referenzierung des Lambda Aufrufkontextes
- Kopieren des Lambda Aufrufkontextes
- Type Traits
- static_assert
- Die Thread-Unterstützung der Standard-bibliothek
- Die Basismechanismen
- Mutexe
- Atomare Operationen
- Atomare Typen
- Bedingungs-Variablen (condition variables)
- Lokaler Threadspeicher
- Threading mit Lambdas (closures)
- Futures
- Smart Pointer
  - unique_ptr
  - shared_ptr
  - weak_ptr
- forward_list
- Reguläre Ausdrücke
- Variadic Templates

Das Seminar führt theoretisch und praktisch in den neuen Standard ein. Nach den theoretischen Abschnitten werden praktische Übungen eingelegt, sodaß die Anwendbarkeit der neuen Sprach- und Bibliotheksbestandteile vom Teilnehmer erfahren werden kann.

Der neue Standard wird jedoch noch von keinem Compiler voll unterstützt, weshalb nicht jedes Thema mit praktischen Beispielen demonstriert werden kann. Die meisten neuen Sprachmerkmale werden durch den neuesten GNU C++-Compiler geboten. Aber

auch die aktuelle Version des Visual C++ Compilers von Microsoft setzt den Standard bereits zu einem großen Teil um. Insofern ist es naheliegend einen der genannten Compiler für das Seminar zu wählen. Es kann allerdings auch ein beliebiger anderer C++ Compiler zum Einsatz gebracht werden, wenn es gewünscht wird.

Einführung in die C++ Standard Template Library

Dieser Kurs wendet sich an diejenigen, die den Umgang mit der Standard Template Library - STL - lernen möchten. Dabei ist ein gutes Grundverständnis der Programmiersprache ANSI/ISO C++ Voraussetzung.
Die STL ist eine flexible und mächtige Bibliothek allgemeiner Container-implementierungen, containerspezifischer Operationen und Algorithmen. Sogar Elemente funktionaler Programmierung sind in ihr enthalten. Dabei besitzt die STL eine erweiterbare Struktur die es erlaubt die in ihr enthaltenen Schichten zu erweitern oder Fremdcode zu adaptieren. Die Aspekttrennung ihrer Implementierung hat weit über die STL hinaus Anwendung gefunden und ist heute eines der Basisprinzipien moderner Bibliotheks- und Framework-implementierungen. Insofern ist dieses Seminar nicht nur für Teilnehmer interessant, die die STL direkt einsetzen wollen. Es ist auch für jene sinnvoll, die diese Prinzipien verstehen und in eigenen Bibliotheken zum Einsatz bringen wollen.

In diesem Kurs wird die STL mit praktischen Beispielen vorgestellt. Neben der Praxis wird die Theorie vorgestellt, die von der traditionellen objektorientierten Programmierung abweicht. Neben der Schichtstruktur der STL stehen dabei die eingesetzten Designprinzipien der generischen Programmierung im Vordergrund. Diese Prinzipien sind sowohl für die Anwendung der bestehenden Elemente der STL wichtig, wie auch für den Entwurf eigener Bibliothekselemente, die mit der ihr zusammenarbeiten sollen.
Natürlich wird auch grundsätzlich auf die Templateprogrammierung in C++ eingegangen, da diese eine unverzichtbare Voraussetzung für den Einsatz der STL ist. Weitere wichtige Aspekte sind das Laufzeitverhalten und der Footprint bei der Verwendung von STL Elementen. Für die Anwendung der STL in speziellen Umgebungen wie zum Beispiel in der Controller-Programmierung ist außerdem das Allokationsverhalten von zentraler Bedeutung. Auch diese Themen werden ausführlich besprochen.

Seminarinhalte:

Abgrenzung: STL und C++ Standardbibliothek
Generische Programmierung
Der Aspekt-orientierte Ansatz
Funktionale Programmierung
Die Templateprogrammierung in C++
Überblick über die STL
Container
Adapter
Iteratoren
Algorithmen
Listen
Vektoren
Queues und Dequeues
Stacks
Sets und Multisets
Maps und Multimaps
Das Laufzeitverhalten der Container
Das Allokationsverhalten der Container
Footprint
Compilerunterstützung
Integration der STL in die Standard C++ Bibliothek
Die Änderungen des neuen C++11 Standards
Iteratordesign
Konstante und gegenläufige Iteratoren
Forward, Bidirectional und Random-Access Iteratoren
Stream Iteratoren
Algorithmen
Modifizierende und nicht-modifizierende Algorithmen
Funktions- und Methodenzeiger
Parameterbindung
Funktionsobjekte

Generische Funktoren
Designprinzipien in der STL
Containerdesign
Eigene Iteratoren
Design von Algorithmen
Die Verwendung von typename
Die Bedeutung von typedef
Die Aspekttrennung im Containerdesign
Allokatoren
Die Aspekttrennung im Design von Algorithmen
Codebloat und Codeoptimierung durch Templates
Die Anwendung von Typetraits

Da in dem Kurs praktische Beispiele durchgeführt werden sollen, ist ein C++ Compiler nötig, der annähernd ANSI/ISO konform ist. Typischerweise wird ein GNU C++ Compiler, clang oder Visual C++ eingesetzt. Die Entwicklungsplattform - vorzugsweise Windows, Mac OS X, Linux oder UNIX - sollte den Kursteilnehmern gut bekannt sein.

C++ Entwicklung im Embedded System

Der Kurs setzt ein Grundverständnis von C++ voraus und geht auf die Besonderheiten der Sprache für die Embedded Softwareentwicklung ein. C++ wird immer mehr aus den klassischen Bereichen der Softwareentwicklung verdrängt. In der Entwicklung von Anwendungssoftware und Serverdiensten haben Java & Co. die Sprache C++ inzwischen weit überrundet. Auch die Technologieführerschaft in der Objektorientierten Softwareentwicklung hat inzwischen Java inne. Auf der anderen Seite lassen sich aus diversen Gründen diese neuen Sprachen in vielen Bereichen der Embedded Entwicklung nicht einsetzen. Eine wesentliche Ursache ist das Fehlen eines deterministischen Laufzeitverhaltens. Weitere Gründe sind der enorme Resourcenverbrauch und das ungünstige Hochstartverhalten von Java-Applikationen.
Den Determinismus teilt C++ mit C - der klassischen Sprache der Systementwicklung und Embedded Programmierung. Im Resourcenbedarf und im Aufstartverhalten lassen sich C++-Programme ebenso effizient wie C-Programme gestalten.
Der Entwicklung leistungsfähiger Hardware und flexibler Compiler - allen voran GCC - ist es zu danken, dass C++ immer mehr in die Domäne von C einbricht. C++ verschwindet also nicht vom Markt, sondern erlebt in der eingebetteten Programmierung eine Renaissance. Der Einsatz von C++ in der Industrie steht heute auf einer breiteren technologischen Grundlage als noch vor zehn Jahren, als C++ hauptsächlich zur Erstellung von GUI-Oberflächen verwendet wurde. Dieser Veränderung des Einsatzes von C++ muss der Softwareentwickler Rechnung tragen, wenn er erfolgreich sein möchte. In diesem Seminar werden ganz wesentlich Bereiche des Einsatzes von C++ in der Embedded Softwareentwicklung behandelt. Insbesondere werden die Fragen des Laufzeitverhaltens, des Footprints und der Code-Coverage von C++-Code gestellt.

Historische Entwicklung von C++
Grundlegende Syntax
Erste Beispiele
Kennenlernen der Tools
Der C++-Compiler
Der Linker
Standarddatentypen
Strukturen
Operatoren
Funktionen
Zeiger und Referenzen
Funktionsüberladung
Operatorüberladung
Typenkonvertierung
Klassenkonzept
Datenkapselung
Attribute und Methoden
Sichtbarkeitskonzept
Instantiierung von Objekten
Konstruktoren und Destruktoren
Dynamische Speicherallokation
Vererbung
Polymorphie
Virtuelle Methoden
Abstrakte Klassen
Mehrfachvererbung
Exception Handling
Templates
Funktionstemplates
Klassentemplates
Speicherklassen in C++
Bedeutung von Heap und Stack
Externe und statische Linkbarkeit
Aufbau von C++ Projekten
Das Zusammenspiel von Compiler und Linker
Object Files und Bibliotheken
Einfache Makefiles
Hardwarenahe Programmierung
Probleme bei nebenläufiger Programmierung
Entwicklung performanter Software
Codeoptimierung
Footprint und Laufzeitverhalten
Sprachstandards
ANSI/ISO C++
EC++

Es werden die folgenden Fragestellungen behandelt:

Wie verhält sich C++ im Vergleich zu C im Embedded System?
Welche Entscheidungen fällt der ANSI-C++ Standard für die Embedded Entwicklung?
Welche grundsätzlichen Fehler kann man beim Einsatz von C++ machen?
Welche Lösungen bietet der EC++ Standard?
Welche Auswirkungen haben unterschiedliche Speicherverwaltungsstrategien für Applikationsdaten auf das Gesamtsystem?
Welche praktischen Auswirkungen haben physikalische und virtuelle Speicherverwaltung auf das Design von Software?
Was bedeutet Speicherfragmentierung und wie beherrscht man Probleme damit?
Welche Auswirkungen haben der Einsatz von Polymorphie auf Laufzeit und Größe des Programmcodes?
Wie können Templates eingesetzt werden?
Welche Laufzeitaspekte sind bei der Verwendung der Standard Template Library zu beachten?
Wie kann Exception Handling im Embedded System eingesetzt werden?
Welche Auswirkungen hat Exception Handling auf Laufzeit, Footprint und Codestruktur?
Welche Rahmenbedingungen gelten für die Entwicklung nebenläufiger Systeme?
Was bedeutet Tracing und Logging für das zeitliche Verhalten?

Es kann ein beliebiger ANSI/ISO-C++ Compiler eingesetzt werden - vorzugsweise GCC C++. Je nach Bedarf können Übungen direkt auf einem Embedded System ausgeführt werden. Dafür gibt es unterschiedliche Lösungen von der echten Hardware bis zur Simulation. Wenn ein spezielles System zum Einsatz gebracht werden soll, müssen die Rahmenbedingungen und die Machbarkeit vor dem Kurs noch abgesprochen werden. Sehr gerne setze ich QNX und Embedded Linux ein. Der Kurs kann auch mit Beispielen durchgeführt werden, die man auf PCs unter Windows, Linux oder Mac OS X zum Laufen bringt.

Die Regelkataloge MISRA-C:2004 und MISRA-C:2012

In diesem MISRA1)-C Kurs wird die Entwicklung sicherheitskritischer Software mit der Programmiersprache ANSI/ISO-C in den Varianten C89/90 und C99 im Embedded System im Automobilumfeld betrachtet. Dabei werden die Fehlermöglichkeiten in einem Softwareprojekt analysiert und diskutiert. Die typischen Fehler bei der Implementierung mit standard C werden kategorisiert und mit den MISRA-Regeln in Zusammenhang gebracht. Dabei wird neben der Arbeit an den Regeln der Aufbau eines der MISRA Regeldokumente von 2004 oder 2012 durchgesprochen. Die Regeln selbst werden ausführlich behandelt und die Anforderungen an die Dokumentation diskutiert, die nötig ist, um die Regelkonformität nachzuweisen.
Ziel des Seminars ist die Prinzipien und die Regeln kennenzulernen, die MISRA zur Fehlervermeidung in C-Projekten zur Verfügung stellt. Neben diesem technischen Thema wird auch das Thema der Einführung von Codierrichtlinien in ein Softwareprojekt diskutiert.

Der Fehler in der Software.
Der Fehler in der Codierung (Implementierung).
Überlegungen zur Entwicklung im Embedded System.
Sicherheitsrelevante und sicherheitskritische Software.
Spezielle Fehlerquellen in C.
Undefiniertes Verhalten von C.
Implementierungsabhängiges Verhalten von C.
Verbreitete Fehler in C.
Lesbarkeit und Eindeutigkeit von C-Code.
Der MISRA-C Regelkatalog.
Geforderte und empfohlene Regeln.
Statische Codeanalyse.
Programmierrichtlinien und Coding Styles.
Der Entwicklungsprozess.
Verbesserte Beschreibung der Regeln gegenüber älteren Standards.
Vereinfachung der Regelbeschreibung.
Klare Klassifizierung von Regeln und Richtlinien.
Automatische Überprüfbarkeit.
Regel-Scope.
Regeln zu ISO C99.

Es kann ein beliebiger C Compiler eingesetzt werden, der kompatibel zum Standard ANSI C89 bzw. ISO C90 ist. Für MISRA-C:2012 muss der Compiler den C99 Standard unterstützen, was die meisten heute üblichen Compiler auch tun. Als Plattformen empfehlen sich Windows-, Mac OS X, UNIX-, Linux- oder QNX-Systeme. Wichtig ist dabei nur, dass die Kursteilnehmer die Programmiersprache C und einfache Dateioperationen auf dem eingesetzten System beherrschen. Es werden einige Regeln anhand praktischer Programmierbeispiele durchgespielt.

Der MISRA-C++ Regelkatalog vom 05.06.2008

In diesem MISRA1)-C++ Kurs wird die Entwicklung sicherheitskritischer Software mit ANSI/ISO-C++ im Embedded System im Automobilumfeld betrachtet. Dabei werden die Fehlermöglichkeiten in einem Softwareprojekt analysiert und diskutiert. Die typischen Fehler bei der Implementierung mit ANSI/ISO-C++ werden kategorisiert und mit den MISRA-Regeln in Zusammenhang gebracht. Dabei wird die Struktur des Aufbaus des MISRA Regeldokuments von 2008 durchgesprochen. Die Regeln selbst werden ausführlich behandelt. Daneben werden die Dokumentations- und Prozess-anforderungen besprochen, die zur Erreichung der Regelkonformität nötig sind. Ziel des Seminars ist die Prinzipien und die Regeln kennenzulernen, die MISRA zur Fehlervermeidung in C++-Projekten zur Verfügung stellt. Neben diesem technischen Thema wird auch das Thema der Einführung von Codierrichtlinien in ein Softwareprojekt diskutiert.

Die folgenden Themen werden durchgenommen:

Der Fehler in der Software.
Der Fehler in der Codierung (Implementierung).
Überlegungen zur Entwicklung im Embedded System.
Sicherheitsrelevante und sicherheitskritische Software.
Typische Fehlerquellen in C++.
Spezielle Fehlerquellen in C++.
Undefiniertes Verhalten von C++.
Implementierungsabhängiges Verhalten von C++.
Lesbarkeit und Eindeutigkeit von C++-Code.
Die sichere Anwendung objektorientierter Konzepte in C++.
Der MISRA-C++ Regelkatalog.
Geforderte und empfohlene Regeln.
Statische Codeanalyse.
Programmierrichtlinien und Coding Styles.
Der Entwicklungsprozess.

Es kann ein beliebiger C++ Compiler eingesetzt werden, der ANSI/ISO kompatibel nach zum Standard von 1998 oder dem korrigierten Standard von 2003 ist. Als Plattformen empfehlen sich Windows-, Mac OS X-, UNIX-, Linux- oder QNX-Systeme. Wichtig ist dabei nur, dass die Kursteilnehmer die Programmiersprache C++ und einfache Dateioperationen auf dem eingesetzten System beherrschen. Es werden einige Regeln anhand praktischer Programmierbeispiele durchgespielt.

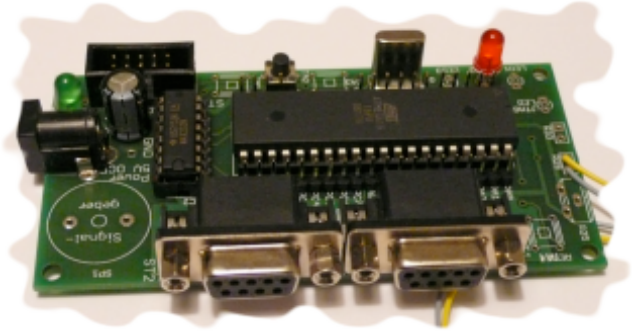
Programmierung des AVR ATmega16 auf dem Entwicklungsboard

Kursinhalte:

Die Programmierung des AVR ATmega16 wird durch Beispiele und Übungen an einem Mikrocontroller-Entwicklungssystem mit der Sprache C vorgestellt. Es wird eine Einführung in die verwendete IDE, den Compile-Download-Test-Zyklus und die verschiedenen Module des Mikrocontrollers und deren Programmierung gegeben.

Die folgenden Themen werden durchgenommen:

C-Programme auf dem Entwicklungsboard
Eine kleine Hello-World Applikationen
Testen der IDE und des Simulators
I/O-Ports – LED blinken lassen
Das EEPROM – eine kleine Bibliothek
Der Timer-Interrupt und Anwendungen
Der externe Interrupt
Entprellung von Tasten
Die serielle Schnittstelle
Kommunikation mit dem Host-Rechner
Der Analog-Digital-Converter
Das Beispiel einer Temperaturmessung



Als Plattformen für die Entwicklung wird Windows, Mac OS X oder Linux verwendet. Wichtig ist dabei nur, dass die Kursteilnehmer die Programmiersprache C und einfache Dateioperationen auf dem eingesetzten Entwicklungssystem beherrschen. Das Seminar kann auch an bestimmte thematische Anforderungen angepasst werden und mit einem bestimmten, für Sie wichtigen Gesichtspunkt ausgestattet werden.

Einführungskurs in die Java-Programmierung

Dieser Kurs richtet sich an Neueinsteiger in die Sprache Java. Es ist jedoch sinnvoll, wenn die Teilnehmer schon etwas Erfahrung in einer beliebigen anderen Programmiersprache mitbringen. Im Kurs werden dann alle Grundkonzepte von Java ausführlich anhand praktischer Beispiele eingeübt. Dabei stehen Syntax und häufig verwendete Javaklassen im Vordergrund. Es werden Applikationen und Applets geschrieben, einfache grafische Benutzeroberflächen und Kommandozeilenapplikationen.

Themen:

Historische Entwicklung von Java
Grundlegende Syntax
Erste Beispiele
Kennenlernen des Compilers
und der VM
Standarddatentypen
Strings
Operatoren
Klassenkonzept
Methoden
Referenzen
Objektinstanziierung mit new
Garbagecollection
Datenkapselung
Attribute und Methoden
Konstruktoren
Vererbung
Die Methode finalize()
Interfaces
Polymorphie
Funktionsüberladung
Statische Methoden und Elemente
Typenkonvertierung
Sichtbarkeitsregeln
Exception Handling
Arrays
Pakete
Das Paket java.util
Abstract Windowing Toolkit
GUI Programmierung
Das Listenerkonzept

Die Programmiersprache Java wird anhand praktischer Beispiele erlernt. Dabei wird das aktuelle JDK von Oracle und eine beliebige Entwicklungsumgebung eingesetzt. Das eingesetzte Betriebssystem spielt keine Rolle, da Java plattformunabhängig ist. Die Teilnehmer sollten allerdings einfache Dateioperationen auf dem eingesetzten System beherrschen. Üblicherweise wird Windows, Mac OS X, Linux oder UNIX als Entwickungs-plattform verwendet.

Aufbaukurs Java-Programmierung

Dieser Kurs richtet sich an Personen mit Grundkenntnissen in Java, die ihre Kenntnisse vertiefen wollen. Insbesondere die Qualitäten von Java, die mit der Typinformation zur Laufzeit zusammenhängen und ganz neue Vorgehensweisen im Vergleich zu C++ zulassen stehen im Zentrum des Kurses. Der Kurs wird anhand praktischer Beispiele durchgeführt, wobei häufig verwendete Pakete aus der Javabibliothek Verwendung finden.

Themen:

Verschachtelte Klassen
Statische Verschachtelung
Anonyme Klassen
Die Klasse Objekt
Die Klasse Class
Laden von Klassen zur Laufzeit
Auslesen von Typinformationen
Die Methoden clone() und equals()
Der statische Kontruktor
Das Streamkonzept von Java
Programmierung von Applets
Die Swing-Bibliothek
GUI Programmierung mit Swing
Herstellen einer Netzverbindung
Auslesen einer URL
Java Reflection
Remote Method Invocation (RMI)
Das Native Code Interface (NCI)
Objektserialisierung
Das JAR Utility
Die JDBC Datenbankschnittstelle

Es wird das aktuelle JDK von Sun verwendet und eine beliebige Entwicklungsumgebung wie z.B. NetBeans oder Eclipse. Das Betriebssystem ist frei wählbar. Normalerweise werden Windows, Linux, UNIX oder Mac OS X eingesetzt.

Java-Umsteigerkurs für C++ Programmierer

Dieser Kurs richtet sich an diejenigen Neueinsteiger in die Sprache Java, die schon gründliche Erfahrungen mit C++ haben. Ausgehend von diesem Hintergundwissen werden im Kurs alle Grundkonzepte von Java ausführlich anhand praktischer Beispiele eingeübt. Dabei steht die Syntax, die Anwendung und der strukturelle Vergleich zwischen C++ und Java im Vordergrund. Es werden Applikationen und Appletts geschrieben, einfache grafische Benutzeroberflächen und Kommandozeilenapplikationen.

Historische Entwicklung von Java
Grundlegende Syntax
Erste Beispiele
Kennenlernen des Compilers
und der VM
Standarddatentypen
Strings
Operatoren
Klassenkonzept
Methoden
Referenzen
Objektinstantiierung mit new
Garbagecollection
Datenkapselung
Attribute und Methoden
Konstruktoren
Vererbung
Die Methode finalize()
Interfaces
Polymorphie
Funktionsüberladung
Statische Methoden und Elemente
Typenkonvertierung
Sichtbarkeitsregeln
Exception Handling
Die Methoden clone() und equals()
Der statische Kontruktor
Arrays
Pakete
Das Paket java.util
Abstract Windowing Toolkit
GUI Programmierung
Das Listenerkonzept
Java Reflection
Remote Method Invocation (RMI)
Das Native Code Interface (NCI)
Objektserialisierung
Das JAR Utility

Die Programmiersprache wird anhand praktischer Beispiele erlernt. Dabei wird das aktuelle JDK von Oracle und eine beliebige Entwicklungsumgebung eingesetzt. Das eingesetzte Betriebssystem spielt keine große Rolle. Die Teilnehmer sollten allerdings einfache Dateioperationen auf dem System beherrschen. Üblicherweise wird Windows, Mac OS X, Linux oder UNIX als Entwicklerplattform verwendet.

Seminar zu Entwurfsmuster in der objektorientierten Softwareentwicklung

Was sind Entwurfsmuster1)? Diese scheinbar theoretische Frage steht im Mittelpunkt des Seminars. Die Frage wird beantwortet, indem alle Entwurfsmuster der GoF2) durchdiskutiert und einige implementiert werden. Der Teilnehmer dieses Kurses kann also erwarten, dass er in die Lage versetzt wird, die Frage nach dem Wesen der Entwurfsmuster nicht nur theoretisch zu beantworten, sondern sich auch einer Mustersprache zu bedienen kann.
Für die Entwicklung einer grundlegenden Mustersprache werden alle Muster des Buches von Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides: "Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software" durchgesprochen und teilweise implementiert.
Zur Implementierung der Muster kann C++ oder auch Java eingesetzt werden.

Es werden die Muster der GoF besprochen:

Abstrakte Fabrik	Befehl
Erbauer	Beobachter
Fabrikmethode	Besucher
Prototyp	Interpreter
Singleton	Iterator
Adapter	Memento
Brücke	Schablonenmethode
Dekorierer	Strategie
Fassade	Vermittler
Fliegengewicht	Zustand
Kompositum	Zuständigkeitskette
Proxy	

Im Seminar werden viele Muster durch praktische Übungen realisiert. Dabei kann der Kurs mit C++ oder Java durchgeführt werden. Auf Wunsch können auch beide Programmiersprachen eingesetzt werden. Bei der Wahl der Entwicklungsplattform ist darauf zu achten, dass die Teilnehmer des Seminars damit schon gearbeitet haben. Vorzugsweise wird Windows, Mac OS X, Linux oder UNIX verwendet.
Es kann jeder übliche C++ und Java Compiler eingesetzt werden. Auch die Wahl der IDE ist frei.

1) Entwurfsmuster = engl. Design Pattern

2) GoF: Gemeint ist die "Gang of Four", Erich Gamma, Richard Helm, Ralph Johnson und John Vlissides, die Autoren des Buches: Entwurfsmuster. Elemente wiederverwendbarer objektorientierter Software. 1995.

Einführung in das Konzept und die Elemente der Unified Modelling Language UML

Die moderne Softwareentwicklung hat viel mit konzeptioneller Arbeit und wenig mit dem Entwurf mathematisch-numerischer Algorithmen zu tun. In der Gesellschaft hält sich das Bild vom Informatiker als Zwilligsbruder des Mathematikers zwar hartnäckig. In der Realität jedoch hat diese Vorstellung nur wenig Berechtigung. Softwareentwicklung heute bedeutet in den seltensten Fällen die Beschäftigung mit Themen der Mathematik. Die Hauptaufgabe des Entwicklers ist in den allermeisten Fällen die Nutzbarmachung und Strukturierung vorhandenen Wissens. Dementsprechend müssen die Methoden der Softwareentwicklung heute Hilfestellung bei der Strukturierung von Wissen und bei der Kommunikation von Wissensinhalten dienen. Die Objektorientierte Software-entwicklung ist eine solche Methode, die sich an sprachlichen Strukturen anlehnt, um die genannten Leistungen für die technische Realisierung von Software zu bringen1).
Um Inhalte objektorientierter Lösungen zu beschreiben kann man neben der Textform auch grafische Darstellungen wählen. Damit diese grafischen Darstellungen eindeutig interpretierbar sind wurde durch die Object Management Group - kurz OMG - ein Standard geschaffen, der Symbole und Semantik einer grafischen Repräsentation für objektorientierte Software festschreibt: die Unified Modelling Language. Die UML ist heute ein wichtiges Hilfsmittel für die objektorientierte Softwareentwicklung und aus der Praxis kaum mehr wegzudenken. Deshalb verfolgt dieser Kurs das Ziel einer umfassenden Einführung für den Einsatz der UML in der modernen Softwareentwicklung. Neben der reinen Softwaresicht befasst sich der Kurs aber auch mit der Prozessicht auf ein Softwareprojekt.

Es werden alle Diagrammarten der UML besprochen und in den Kontext der objektorientierten Entwicklung gebracht. Dafür werden Beispiele herangezogen, die gemeinsam geübt und diskutiert werden. Als methodisches Gerüst des Kurses dient der Software-Lifecycle. Die verschiedenen Phasen des SW-Lebenszyklus werden mit UML-Diagrammen vorbereitet und dokumentiert. Einen Schwerpunkt hat der Kurs in der Unterstützung der Analyse- und Designdisziplin. Dabei wird auch gezeigt, wie die UML in den Prozess der Softwareentwicklung eingebunden wird.

Die Inhalte des Kurses:

Use-Cases
Aktivitätsdiagramme
Sequenzdiagramme
Kollaborationsdiagramme
Analyseklassendiagramme
Entwurfsklassendiagramme
Einführung in die Entwurfsmuster
Zustandsdiagramme
Moduldiagramme
Installationsdiagramme

Die Diskussion der Diagrammtypen findet vor dem Hintergrund einer prozessorientierten Softwareentwicklung statt. Es werden praktische Übungen anhand von wirklichkeitsnahen Problemstellungen durchgeführt. Dabei kann auch ein beliebiges UML-Tool eingesetzt werden.

Seminar zu den Rahmenbedingungen und Entwurfsprinzipien zur erfolgreichen Entwicklung eines Frameworks

In diesem Kurs werden sowohl die technischen Aspekte wie auch die spezifischen Vorgehensmodelle zur Framework-entwicklung behandelt. Von den Anforderung an ein Framework über die verschiedenen Techniken bis hin zu Qualitäts-aspekten wird die Sonderstellung der Entwicklung von Basistechnologien dargestellt.

Kursinhalte:

Begriffsdefiniton Framework
Begriffsdefiniton Bibliothek
Begriffsdefiniton Softwarearchitektur
Begriffsdefiniton Softwaredesign
Die Anforderungen an ein Framework
Explizite Anforderungen
Implizite Anforderungen
Funktionale Anforderungen
Nichtfunktionale Anforderungen
Das Qualitätsziel
Der Einsatzkontext eines Frameworks
Der Stellenwert eines „eleganten“ Designs
Besprechen einiger verbreiteter Frameworks und Komponententechnologien:
MFC
COM/DCOM
CORBA
Java Frameworks
.NET
Schlüsseltechnologien in verbreiteten Frameworks:
Systemabstraktion
RTTI
Komponententechnologie
Musterorientierte Entwicklung
Der Softwareentwicklungsprozess:
in dem ein Framework eingesetzt werden soll
für das Framework selbst
Besondere Anforderungen an den Entwicklungsprozess eines Frameworks
Anforderungen an das Frameworkteam
Das Thema Akzeptanz
Erhöhung und Erhalt von Akzeptanz
Der menschliche Faktor
Support und Maintanance

Der Kurs hat eine große theoretische Komponente die insbesondere den Entwicklungsprozess betrifft. Die Behandlung der Theorie verläuft Diskurs-orientiert.
Die technologische Komponente wird mit praktischen Beispielen untermauert. Dabei lassen sich zwar nicht alle technischen Aspekte an selbst-erarbeiteten praktischen Beispielen demonstrieren. Die meisten werden aber anhand von Beispiele in C++, C# oder Java im Kurs behandelt.

Für die Durchführung des Kurses verwendet man am besten Rechner unter Windows, Mac OS X oder Linux mit den üblichen Entwicklungs-werkzeugen in Java und C++.

Seminar: Standardprobleme und deren Lösungen in der Programmierung parallel ablaufender Threads

In der modernen Softwareentwicklung auf Desktop- und Serversystemen sowie in Embedded Software ist die Nutzung der Möglichkeit der Programmierung parallel ablaufender Threads längst weit verbreitet. Für den Entwickler ist die Beherrschung der dafür notwendigen Techniken zur Pflicht geworden, denn es gibt häufig architektonische Vorentscheidungen, die die Anwendung konkurrierender Ablaufstränge notwendig machen und den Entwickler zwingen, sich mit der Technik und den daraus entstehenden Problemen zu beschäftigen.

Der Kurs gibt einen fundierten Einstieg in die Techniken, die für den erfolgreichen Einsatz paraleller Threads beherrscht werden müssen. Dazu werden praktische Übungen auf Betriebssystemen wie UNIX, Linux, Mac OS X oder Windows® durchgeführt. Dabei lernen die Teilnehmer die Standardprobleme in der Multithreadingprogrammierung und deren Lösung kennen. Für die Durchführung des Kurses wird auf Anschaulichkeit Wert gelegt. Es werden Probleme diskutiert und typische Fehlersituationen aufgezeigt. Alle Fehlersituationen werden in Beispielen von den Teilnehmern simuliert, Lösungsstrategien praktisch erarbeitet und erprobt.
Neben dem vorgegebenen Stoff gibt der Kurs genügend Raum zur Diskussion eigener Projektbeispiele und Problemsituationen.

Die Kursinhalte in Stichworten:

Gründe für die Parallelisierung von Abläufen
Historisches zu Prozessen und Threads
Hardware
Betriebssysteme
Prozesse und Threads
POSIX, Pthreads
Threads unter MS Windows®
Architekturen der Nebenläufigkeit
Threadzustände
Echtzeitaspekte
Parallelität in der Hardware
Schedulingmechanismen
Prioritätsbasiertes Scheduling
FIFO- oder Warteschlangen-Scheduling
Zeitscheibenverfahren
Round Robin Verfahren
Runtime Credit Verfahren
Datenzugriff aus mehreren Threads
Synchronisierung von Datenzugriffen
Condvars
Events
Critical Sections - Spin-Lock Verfahren
Mutexe
Semaphore
Der neue C++ Standard C+11
Typische Situationen des Datenaustauschs zwischen Threads
Typische Fehler
Aushungern (Starvation)
Deadlocks
Race Conditions
Prioritätsinversion
Lösungsansätze für die typischen Fehler
Back-Lock Strategie
Monitorkonzept
Priority-Boost
Endliche Statusmaschine
Threadlokaler Speicher
Run Once Initialization
Diskussion von Nebenläufigkeitsmodellen
Pipeline
Workgroup Modell
Manager/Worker (Client/Server)
Fork on Idle
Threading und Objektorientierung
Diskussion neuer Bibliotheken und Techniken
Open MP
Boost

Es kann ein beliebiger C oder C++ Compiler eingesetzt werden. Als Plattformen empfehlen sich Windows®-, Mac OS X-, UNIX- oder Linuxsysteme. Wichtig ist dabei, dass die Kursteilnehmer das eingesetzte System kennen. Idealerweise arbeitet man mit mehreren unterschiedlichen Systemen, um Ähnlichkeiten und Unterschiede in modernen Betriebssystemen demonstrieren zu können. Dabei können auch Beispiele in Java zur Ergänzung der Technologiepalette herangezogen werden.
Das Seminar kann an spezielle thematische Anforderungen angepasst werden und auf einen bestimmten, für Sie wichtigen Gesichtspunkt, ausgerichtet werden.

Einführungskurs in die C#-Programmierung

Dieser Kurs richtet sich an Neueinsteiger in die Sprache C#. Es ist jedoch sinnvoll, wenn die Teilnehmer schon etwas Erfahrung in einer beliebigen anderen Programmiersprache mitbringen. Im Kurs werden dann die Grundkonzepte von C# und des .NET-Frameworks ausführlich anhand praktischer Beispiele eingeübt. Dabei stehen die Syntax und die häufig verwendeten Frameworkelemente im Vordergrund. Es werden Applikationen mit grafischen Benutzeroberflächen geschrieben und Kommandozeilenapplikationen.

Themen:

Historische Entwicklung von C#
Die Einbettung von C# in das .NET Framework
CIL (MSIL) und CTS
Grundlegende Syntax
Erste Beispiele
Kennenlernen des Compilers
Referenzen
Objektinstanziierung mit new
Garbagecollection
Standarddatentypen
Strings
Operatoren
Klassenkonzept
Methoden
Eigenschaften (properties)
Namensräume
Arrays
Collections
Datenkapselung
Sichtbarkeitsregeln
Attribute und Methoden
Statische Methoden und Elemente
Konstruktoren
Der Destruktor
Vererbung
Überschreiben von Methoden
Interfaces
Polymorphie
Funktionsüberladung
Abstrakte Methoden
Versiegelte Methoden
Funktionsüberschreibung mit new
Typenkonvertierung
Typenabfrage
Referenztypen und Wertetypen
Boxing, Unboxing und Autoboxing
nullable types
Exception Handling
Delegates
Multidelegates
Events
Partielle Klassendefinitionen
Windows Forms
GUI Programmierung
Das Eventkonzept
Reflection

Die Programmiersprache C# wird anhand praktischer Beispiele erlernt. Dabei wird das aktuelle Visual C# von Microsoft unter Windows eingesetzt.

Seminar zu Prozessen in der Softwareentwicklung

In diesem Seminar werden Standardvorgehensmodelle und Prozesse der Softwareentwicklung behandelt. Im historischen Rückblick werden verschiedene Ansätze, Erfolgskonzepte und Irrtümer diskutiert. Dabei werden dem Teilnehmer begriffliche Systeme vermittelt, die zur Beschreibung von Entwicklungsprozessen tauglich sind. Über die traditionellen Prozessmodelle hinaus werden aktuelle Veränderungen bei den Software-entwicklungs-prozessen durchgesprochen.

Neben den rein theoretischen Modellen setzt sich das Seminar auch mit der Frage auseinander, wie bestehende Prozesse beschrieben und weiterentwickelt werden können.
Welche Ziele verfolgt man mit der Beschreibung und Entwicklung von Prozessen? Mit welchen mentalen und sozialen Widerständen muss gerechnet werden, wenn man Entwicklungsprozesse in einem Unternehmen oder in einem Projekt verbessern möchte? Was bedeutet Qualität? Was haben die Prozesse im Unternehmen mit dem geforderten Qualitätsbegriff zu tun? Und was bedeutet es, wenn der Auftraggeber eines Softwareprojektes auf die Transparenz und die Prüfbarkeit von Entwicklungsprozessen drängt?

Themen:

Prinzipielle Vorgehensweisen bei der Softwareentwicklung
Wasserfallmodell
Spiralmodell
Prototypisches Vorgehendmodell
Iterative Modelle
Analytische Modelle
Agile Modelle
Rahmenmodelle
Einige ausgewählte Beispiele
V-Modell
eXtreme Programming
Scrum
SPICE
Qualität und Prozesse
Zielsetzung von Prozessen
Prozesseinführung
Prozessverbesserung
Qualitätssicherung
QS am Produkt
QS am Prozess
Das Thema Anforderungen
Anforderungen an das Produkt
Anforderungen an den Prozess
Der Kunde und das Projekt
Machtbeziehungen
Kommunikationsbeziehungen
Aufgabenverteilungen
Die Projektmitarbeiter
Kultureller Hintergrund und „Realität“
Ethik und Qualität
Frustration
Rollen
Gegenspieler

Das Seminar wird in Form von Vorträgen, Diskussionen und kleinen sozialen Experimenten durchgeführt. Erfahrungen in der Softwareentwicklung werden zwar nicht vorausgesetzt, sind aber von Vorteil. Auch Projektleitungserfahrungen stellen eine günstige Ausgangsbasis für diesen Kurs dar.