Privater Masterstudiengang
Software-Qualität

tech technologische universität

technologische universität

Privater Masterstudiengang
Software-Qualität

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Internetzugang: www.techtitute.com/de/informatik/masterstudiengang/masterstudiengang-software-qualitat

Index

01
Präsentation
Seite 4

02
Ziele
Seite 8

03
Kompetenzen
Seite 16

04
Kursleitung
Seite 20

05
Struktur und Inhalt
Seite 28

06
Methodik
Seite 40

07
Qualifizierung
Seite 48

01

Präsentation

Das rasante Wachstum der Branche und die aktuellen Marktanforderungen haben zu einem hohen Maß an technischer Schulden in Softwareprojekten geführt. Denn es ist zwingend notwendig, schnell auf die Anforderungen des Kunden oder des Unternehmens zu reagieren, ohne dass die Qualität des Systems im Detail dabei verliert oder darunter leidet. Hier spiegelt sich die Notwendigkeit wider, die Skalierbarkeit des Projekts während seines gesamten Lebenszyklus zu berücksichtigen, was IT-Kenntnisse erfordert, die von einem *Top-Down*-Ansatz auf Qualität ausgerichtet sind. Dieses Programm entwickelt die Kriterien, Aufgaben und fortgeschrittenen Methoden, um die Relevanz einer Arbeit zu verstehen, die sich an der Notwendigkeit orientiert, eine Qualitätspolitik in *Software Factories* umzusetzen. Das Studium wird vollständig online und mit einer Dauer von 12 Monaten durchgeführt, angepasst an die Methodik der größten digitalen Universität der Welt.

Spezialisieren Sie sich auf Software-Qualität aus technischer und Management-Perspektive; erwerben Sie Ihre Qualifikation in 12 Monaten und machen Sie in Ihrem beruflichen Umfeld einen Unterschied"

Das Konzept der technischen Schuld, das derzeit von zahlreichen Unternehmen und Verwaltungen gegenüber ihren Lieferanten angewandt wird, spiegelt die improvisierte Art und Weise wider, in der Projekte entwickelt wurden. Dies führt zu neuen impliziten Kosten, da ein Projekt aufgrund der Annahme einer schnellen und einfachen Lösung im Gegensatz zu einem skalierbaren Ansatz in der Entwicklung des Projekts wiederholt werden muss.

Seit einigen Jahren werden Projekte sehr schnell entwickelt, mit dem Ziel, sie auf der Grundlage von Preis- und Terminkriterien mit dem Kunden abzuschließen, anstatt einen Qualitätsansatz zu wählen. Diese Entscheidungen fordern nun ihren Tribut von vielen Lieferanten und Kunden.

Dieser private Masterstudiengang befähigt IT-Fachleute, die Kriterien zu analysieren, die der Softwarequalität auf allen Ebenen zugrunde liegen. Kriterien wie Datenbankstandardisierung, Entkopplung zwischen den Komponenten eines Informationssystems, skalierbare Architekturen, Metriken, Dokumentation, sowohl funktional als auch technisch. Zusätzlich zu den Methoden für das Management und die Entwicklung von Projekten und anderen Methoden zur Qualitätssicherung, wie z.B. kollaborative Arbeitstechniken, einschließlich Pair Programming, die es ermöglichen, dass das Wissen im Unternehmen und nicht bei den Menschen verbleibt.

Die überwiegende Mehrheit der Studiengänge dieser Art konzentriert sich auf eine Technologie, eine Sprache oder ein Tool. Dieses Programm ist insofern einzigartig, als es den Fachleuten die Bedeutung der Softwarequalität bewusst macht und die technische Schuld von Projekten mit einem qualitativen statt einem auf Wirtschaftlichkeit und kurze Fristen ausgerichteten Ansatz reduziert; es stattet die Studenten mit Fachwissen aus, so dass die Budgetierung von Projekten gerechtfertigt werden kann.

Um dies zu ermöglichen, hat die TECH Technologische Universität eine Gruppe von Experten auf diesem Gebiet zusammengestellt, die das aktuellste Wissen und die neuesten Erfahrungen weitergeben werden. Über einen modernen virtuellen Campus mit theoretischen und praktischen Inhalten, die in verschiedenen Formaten angeboten werden. Es werden 10 Module angeboten, die in verschiedene Themen und Unterthemen unterteilt sind und es ermöglichen, in 12 Monaten mit der Relearning-Methode zu lernen, die das Einprägen und Lernen auf agile und effiziente Weise erleichtert.

Dieser **Privater Masterstudiengang in Software-Qualität** enthält das vollständigste und aktuellste Programm auf dem Markt. Die hervorstechendsten Merkmale sind:

- Die Entwicklung von Fallstudien, die von Experten für Softwareentwicklung vorgestellt werden
- Der anschauliche, schematische und äußerst praxisnahe Inhalt soll wissenschaftliche und praktische Informationen zu den für die berufliche Praxis wesentlichen Disziplinen vermitteln
- Er enthält praktische Übungen in denen der Selbstbewertungsprozess durchgeführt werden kann um das Lernen zu verbessern
- Ein besonderer Schwerpunkt liegt auf innovativen Methoden
- Theoretischer Unterricht, Fragen an den Experten und individuelle Reflexionsarbeit
- Die Verfügbarkeit des Zugangs zu Inhalten von jedem festen oder tragbaren Gerät mit Internetanschluss

Der private Masterstudiengang in Software-Qualität analysiert die dem Thema zugrunde liegenden Kriterien auf allen Ebenen. Erweitern Sie Ihr Fachwissen. Schreiben Sie sich jetzt ein"

Entwickeln Sie die Kriterien, Aufgaben und fortgeschrittenen Methoden, um die Relevanz qualitätsorientierter Arbeit zu verstehen und Ihrem Unternehmen oder Kunden effektive Lösungen zu bieten"

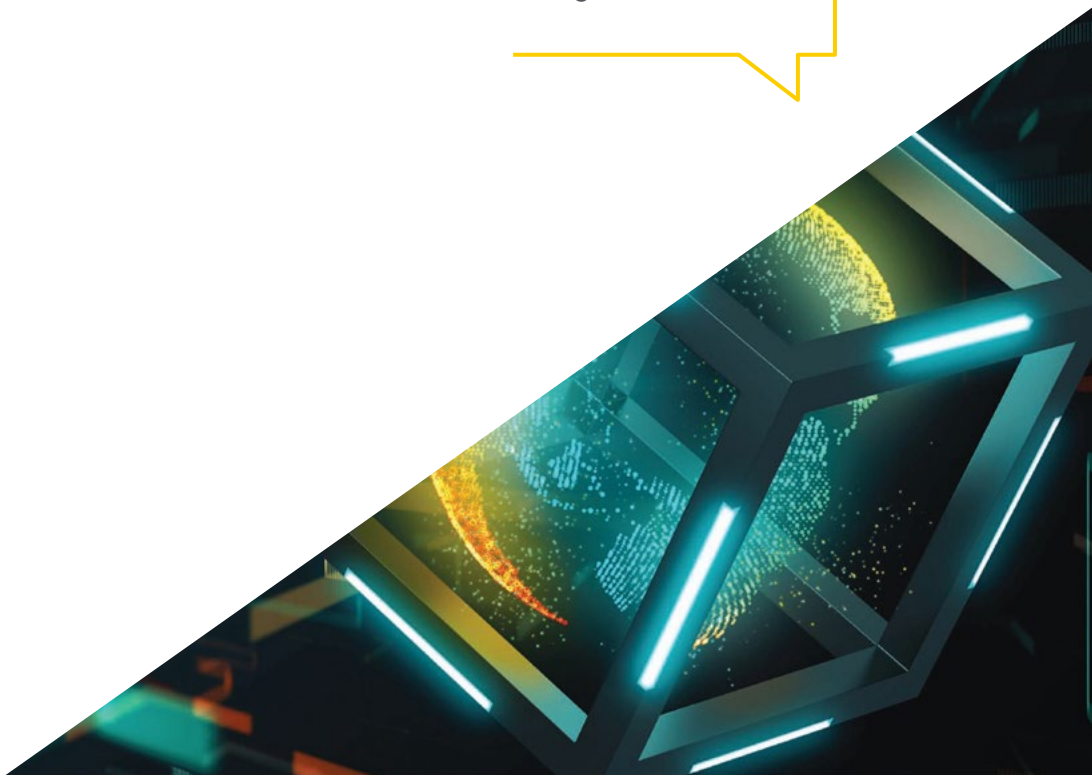
Ein Programm zur Sensibilisierung für die Bedeutung von Software-Qualität und für die Notwendigkeit, Qualitätsrichtlinien in Software Factories einzuführen.

Praxisorientiert und flexibel lernen. Teilen Sie Ihren Alltag mit dieser 100%igen Online-Schulung, die exklusiv von der TECH Technologischen Universität angeboten wird.

Zu den Lehrkräften des Programms gehören Fachleute aus der Branche, die ihre Berufserfahrung in diese Fortbildung einbringen, sowie renommierte Fachleute von Referenzgesellschaften und angesehenen Universitäten.

Die multimedialen Inhalte, die mit den neuesten Bildungstechnologien entwickelt wurden, ermöglichen den Fachleuten ein situiertes und kontextbezogenes Lernen, d. h. eine simulierte Umgebung, die ein immersives Training ermöglicht, das auf reale Situationen ausgerichtet ist.

Das Konzept dieses Studiengangs konzentriert sich auf problemorientiertes Lernen, bei dem die Fachkraft versuchen muss, die verschiedenen Situationen aus der beruflichen Praxis zu lösen, die während des gesamten Studiengangs gestellt werden. Zu diesem Zweck werden sie von einem innovativen interaktiven Videosystem unterstützt, das von renommierten Experten entwickelt wurde.



02

Ziele

Der Private Masterstudiengang in Software-Qualität vermittelt den Studenten eine klare und spezialisierte Vorstellung von der Bedeutung der Qualität in Software-Entwicklungsprozessen. Sowie die fortschrittlichsten Tools zur Implementierung von DevOps-Prozessen und Systemen zur Qualitätssicherung. Kurz gesagt, es wird ein breites und spezialisiertes theoretisches und praktisches Wissen vermittelt, um die Entwicklung von Projekten aus einer modernen und effizienten Perspektive zu verstehen.

Sie können auf alle Inhalte zugreifen, wann immer Sie wollen. Von Ihrem Computer oder Ihrem Lieblingsgerät aus. Sie können sie auch für Ihre nächste Anfrage herunterladen"

Allgemeine Ziele

- Entwicklung von Kriterien, Aufgaben und fortgeschrittenen Methoden, um die Bedeutung qualitätsorientierter Arbeit zu verstehen
- Analyse der wichtigsten Faktoren für die Qualität eines Softwareprojekts
- Entwicklung der relevanten regulatorischen Aspekte
- Implementierung von DevOps und Systemprozessen zur Qualitätssicherung
- Reduzierung der technischen Schulden von Projekten mit einem Qualitätsansatz anstelle eines Ansatzes, der auf Wirtschaftlichkeit und kurzen Fristen basiert
- Vermittlung des Know-hows, um die Qualität eines Softwareprojekts messen und quantifizieren zu können
- Die wirtschaftlichen Vorschläge von Projekten auf der Grundlage von Qualität verteidigen

Spezifische Ziele

Modul 1. Software-Qualität. TRL-Entwicklungsstufen

- Die Elemente, die die Softwarequalität ausmachen, klar und prägnant entwickeln
- Die Modelle und Standards je nach System, Produkt und Softwareprozess anwenden
- Vertiefung der angewandten ISO-Qualitätsnormen sowohl im Allgemeinen als auch in spezifischen Bereichen
- Die Standards je nach Bereich der Umgebung anwenden (lokal, national, international)
- Prüfung der TRL-Reifegrade und deren Anpassung an die verschiedenen Teile des Softwareprojekts, die behandelt werden sollen
- Erwerb der Abstraktionsfähigkeit, um ein oder mehrere Kriterien von Elementen und Ebenen der Softwarequalität anzuwenden
- Unterscheidung der Anwendungsfälle der Standards und Reifegrade in einem simulierten Realfallprojekt

Modul 2. Software-Projektentwicklung. Funktionelle und technische Dokumentation

- Bestimmung des Einflusses des Projektmanagements auf die Qualität
- Entwicklung der verschiedenen Phasen eines Projekts
- Unterscheidung der Qualitätskonzepte für funktionale und technische Dokumentation
- Analyse der Phase der Anforderungserfassung, der Analysephase, des Teammanagements und der Konstruktionsphase
- Die verschiedenen Software-Projektmanagement-Methoden einführen
- Kriterien erstellen, um zu entscheiden, welche Methode je nach Art des Projekts am besten geeignet ist

Modul 3. Software *Testing*. Testautomatisierung

- Die Unterschiede zwischen Produktqualität, Prozessqualität und Nutzungsqualität ermitteln
- Die ISO/IEC 15504-Norm kennen
- Die Details von CMMI ermitteln
- Die Schlüssel zu kontinuierlicher Integration, Repositories und deren Auswirkungen auf ein Software-Entwicklungsteam verstehen
- Die Bedeutung der Einbeziehung von Repositories für Softwareprojekte feststellen Erfahren, wie man sie mit TFS erstellt
- Die Bedeutung der Skalierbarkeit von Software bei der Konzeption und Entwicklung von Informationssystemen erfassen

Modul 4. Software-Projektmanagement-Methoden. *Waterfall*-Methoden versus agile Methoden

- Bestimmung, woraus die Waterfall-Methode besteht
- Vertiefung in die Scrum-Methodik
- Die Unterschiede zwischen Waterfall und Scrum herausarbeiten
- Die Unterschiede zwischen der Waterfall und Scrum -Methode und wie der Kunde sie sieht
- Untersuchung des Panel Kanban
- Dasselbe Projekt mit Waterfall undScrum angehen
- Ein Hybridprojekt einrichten

Modul 5. TDD (*Test Driven Developement*). Testgetriebener Softwareentwurf

- Kennenlernen der praktischen Anwendung von TDD und seiner Möglichkeiten für das Testen eines Softwareprojekts in der Zukunft
- Vervollständigung der vorgeschlagenen realen Simulationsfälle, um dieses TDD-Konzept kontinuierlich zu erlernen
- In den Simulationsfällen zu analysieren, inwieweit die Tests vom konstruktiven Standpunkt aus erfolgreich sein oder fehlschlagen können
- Bestimmen Sie die Alternativen zu TDD und führen Sie eine vergleichende Analyse zwischen ihnen durch

Modul 6. DevOps. Software-Qualitätsmanagement

- Analyse der Unzulänglichkeiten eines traditionellen Prozesses
- Bewertung möglicher Lösungen und Auswahl der am besten geeigneten Lösung
- Geschäftsanforderungen und ihre Auswirkungen auf die Implementierung verstehen
- Die Kosten der durchzuführenden Verbesserungen abschätzen
- Entwicklung eines entwicklungsfähigen Software-Lebenszyklus, angepasst an die tatsächlichen Bedürfnisse
- Mögliche Fehler vorhersehen und bereits im Entwurfsprozess vermeiden
- Die Verwendung der verschiedenen Implementierungsmodelle rechtfertigen

Modul 7. DevOps und kontinuierliche Integration. Fortgeschrittene praktische Lösungen in der Softwareentwicklung

- Die Phasen des Softwareentwicklungs- und -auslieferungszyklus identifizieren, die an bestimmte Fälle angepasst sind
- Entwurf eines Softwareentwicklungsprozesses mit kontinuierlicher Integration
- Entwicklung und Implementierung von kontinuierlicher Integration und Bereitstellung auf der Grundlage eines vorherigen Entwurfs
- Automatische Qualitätskontrollpunkte für jede Softwarelieferung einrichten
- Aufrechterhaltung eines automatisierten und robusten Softwareentwicklungsprozesses
- Anpassung zukünftiger Anforderungen an den Prozess der kontinuierlichen Integration und Bereitstellung
- Analyse und Vorhersage von Sicherheitsschwachstellen während und nach der Auslieferung der Software

Modul 8. Datenbank-Design (DB). Standardisierung und Leistung. Software-Qualität

- Die Verwendung des Entity-Relationship-Modells für den vorläufigen Entwurf einer Datenbank beurteilen
- Eine Entity, ein Attribut, einen Schlüssel usw. anwenden, um die beste Datenintegrität zu gewährleisten
- Bewertung der Abhängigkeiten, Formen und Regeln der Standardisierung von Datenbanken
- Spezialisierung auf den Betrieb eines OLAP-Data-Warehouse-Systems, Entwicklung und Verwendung von Fakten- und Dimensionstabellen
- Bestimmung der wichtigsten Faktoren für die Datenbankleistung
- Durchführung von vorgeschlagenen realen Simulationsfällen zum kontinuierlichen Lernen von Datenbankdesign, Normalisierung und Leistung
- In der Simulation die Optionen festlegen, die bei der Erstellung der Datenbank vom konstruktiven Standpunkt aus zu lösen sind

Modul 9. Entwurf skalierbarer Architekturen. Architektur im Software-Lebenszyklus

- Das Konzept der Software-Architektur und ihrer Merkmale entwickeln
- Die verschiedenen Arten der Skalierbarkeit in der Softwarearchitektur bestimmen
- Analyse der verschiedenen Stufen, die bei der Web-Skalierbarkeit auftreten können
- Erwerb von Fachwissen über das Konzept, die Phasen und Modelle des Software-Lebenszyklus
- Die Auswirkungen einer Architektur auf den Software-Lebenszyklus mit ihren Vorteilen, Einschränkungen und unterstützenden Tools bestimmen
- Vervollständigung der vorgeschlagenen realen Simulationsfälle, um die Architektur und den Software-Lebenszyklus kontinuierlich zu erlernen
- In der Simulation beurteilen, inwieweit das Architekturdesign durchführbar oder überflüssig ist

Modul 10. ISO/IEC 9126 Qualitätskriterien. Metriken zur Software-Qualität

- Entwicklung des Konzepts der Qualitätskriterien und der relevanten Aspekte
- Prüfung der Norm ISO/IEC 9126, Hauptaspekte und Indikatoren
- Analyse der verschiedenen Metriken für ein Softwareprojekt, um die vereinbarten Bewertungen zu erfüllen
- Untersuchung der internen und externen Attribute, die bei der Qualität eines Softwareprojekts zu berücksichtigen sind
- Unterscheidung der Metriken nach der Art der Programmierung (strukturiert, objektorientiert, schichtweise usw.)
- Abschluss von realen Simulationen, als kontinuierliches Lernen der Qualitätsmessung
- In den Simulationsfällen sehen, inwieweit es machbar oder unnötig ist, d.h. vom konstruktiven Standpunkt der Autoren aus gesehen

“Unterstreichen Sie Ihr berufliches Profil mit dieser exklusiven Fortbildung. Erlangen Sie Ihre Qualifikation in 12 Monaten und auf praktische Weise mit der Methodik, die Ihnen nur die TECH Technologische Universität bieten kann“

03

Kompetenzen

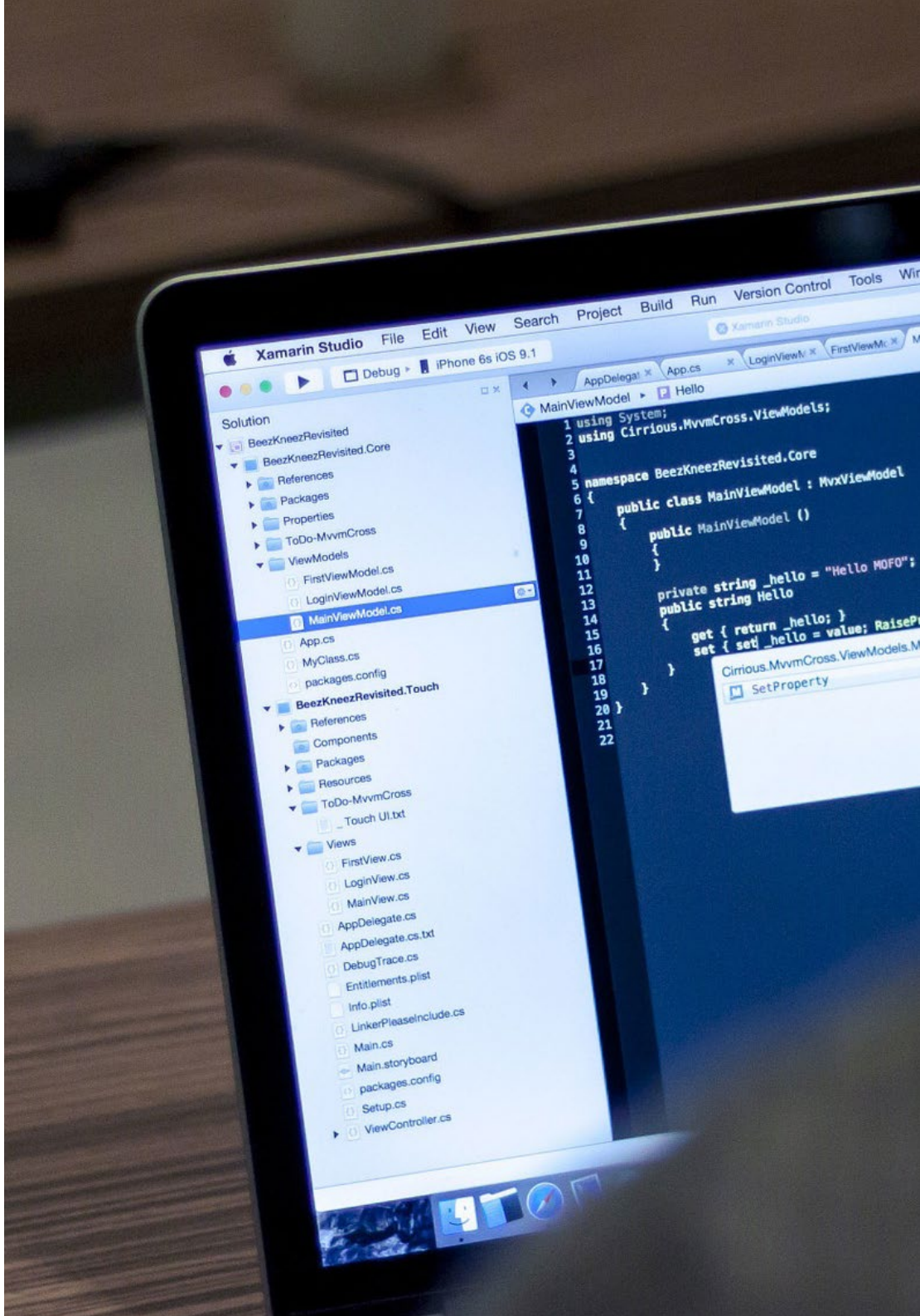
Die Studenten dieses Privaten Masterstudiengangs in Software-Qualität werden das Thema sowohl aus technischer als auch aus Managementperspektive beherrschen. Sie werden in der Lage sein, den Ansatz für ein Projekt sowie dessen Ausführung zu entwickeln und eine nachhaltige, effektive und qualitativ hochwertige Architektur in den Softwareprojekten zu entwickeln, die ihnen vorgelegt werden. Zu diesem Zweck haben die Dozenten ihre gesamte persönliche Erfahrung in die Ausarbeitung einer Vielzahl praktischer Fälle gesteckt, die der Kontextualisierung und Weiterentwicklung angesichts dieser "technischen Schuld" dienen, die es nicht mehr geben wird.

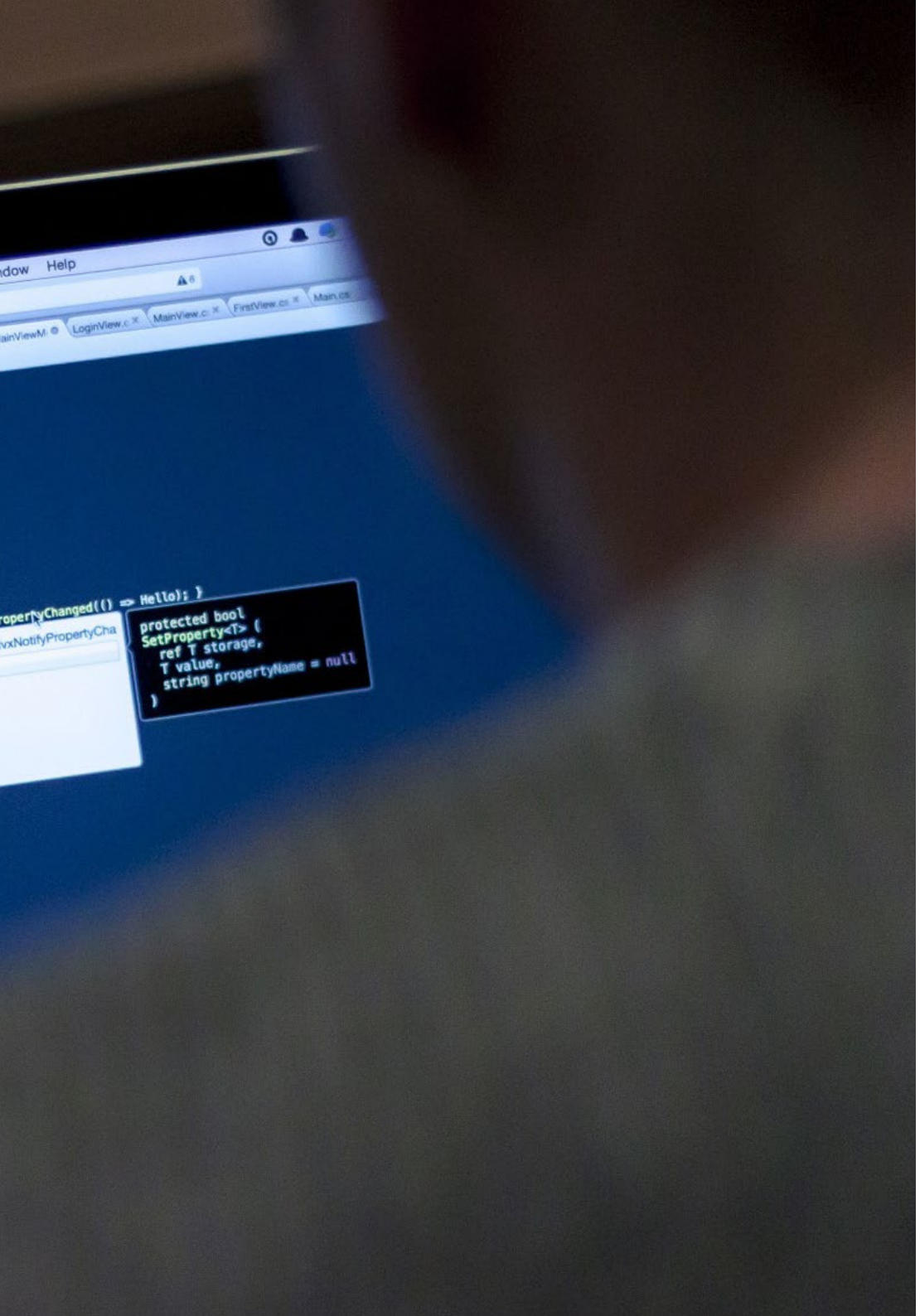
Ein qualitätsorientierter IT-Experte ist für Software-Beratungsunternehmen und große Unternehmen von wachsendem Wert. Schreiben Sie sich jetzt für diesen privaten Masterstudiengang in Software-Qualität ein"

Allgemeine Kompetenzen

- Die technischen Schulden von Projekten mit einem Qualitätsansatz statt mit einem Ansatz, der auf Wirtschaftlichkeit und kurzen Fristen basiert, zu reduzieren
- Messung und Quantifizierung der Qualität eines Softwareprojekts
- TDD richtig durchführen, um die Qualitätsstandards für Software zu erhöhen
- Qualitätsorientierte Projektbudgetierung begründen
- Entwicklung von Qualitätsstandards, Modellen und Normen
- Prüfung verschiedener Bewertungen des Technologiereifegrads
- Reduzierung des Risikos und Gewährleistung der Pflege und Kontrolle nachfolgender Versionen
- Die Phasen beherrschen, in die ein Projekt aufgeteilt ist

Verbessern Sie Ihre Fähigkeiten und entdecken Sie die unendlichen Möglichkeiten für berufliches Wachstum, die Ihnen diese neue Erfahrung eröffnet“





Spezifische Kompetenzen

- Ein Software-System nach dem Grad des Fortschritts im Projektprozess bewerten
- Diese Punkte der Zuverlässigkeit, Metriken und Sicherheit in Softwareprojekten richtig und strategisch angehen
- Auf den Prozess der Entscheidungen über die im Projekt zu verwendende Methodik eingehen
- Beherrschung der wesentlichen normativen Aspekte für die Erstellung von Software
- Automatisches *Testing* entwickeln
- Aufbau einer angemessenen Kommunikation mit dem Kunden, um zu verstehen, wie er/sie das Projekt gemäß der angewandten Methodik wahrnimmt
- Ausarbeitung der Liste der Testanforderungen
- Durchführung von Abstraktion, Aufteilung in einheitlichere Tests und Eliminierung dessen, was für die gute Durchführung der Tests des durchzuführenden Softwareprojekts nicht relevant ist
- Aktualisierung der Liste der Testanforderungen auf angemessene und korrekte Weise
- Anpassung der DevOps-Kultur an die Geschäftsanforderungen
- Entwicklung der neuesten Praktiken und Tools für die kontinuierliche Integration und Bereitstellung
- Umstrukturierung und Bewältigung der Datenverwaltung und-koordination

04

Kursleitung

Fachkundige Dozenten mit einem umfangreichen Studienplan im Bereich IT-Lösungen und Softwareentwicklung und -forschung leiten diesen privaten Masterstudiengang, um den zukünftigen Studenten die notwendigen Werkzeuge und Kenntnisse zu vermitteln, die sich auf die Qualität in Softwareentwicklungsprozessen und die fortschrittlichsten Werkzeuge zur Implementierung von DevOps-Prozessen und -Systemen zur Qualitätssicherung konzentrieren. Dieses Team von Fachleuten wird den Studenten jederzeit zur Seite stehen, um die Ziele aus der Ferne zu erreichen, da es sich um ein reines Online-Programm handelt, das der avantgardistischsten Methodik von TECH folgt.

Spezialisierte Dozenten sind bestrebt, Ihnen die besten Inhalte zu vermitteln und Ihren Studienprozess zu einer lebendigen und dynamischen Erfahrung zu machen. Um Ihre Zweifel zu klären und Sie auf Ihrem Weg zu begleiten"

Internationaler Gastdirektor

Daniel St. John blickt auf eine mehr als 30-jährige berufliche Erfahrung im Technologiesektor zurück und ist ein angesehener **Computeringenieur**, der sich auf **Softwarequalität** spezialisiert hat. In diesem Bereich hat er sich aufgrund seines pragmatischen Ansatzes, der auf kontinuierlicher Verbesserung und Innovation beruht, als echter Experte etabliert.

Im Laufe seiner Karriere hat er an internationalen Einrichtungen wie **General Electric Healthcare** in Illinois mitgewirkt. So konzentrierte sich seine Arbeit auf die Optimierung der **digitalen Infrastrukturen** von Organisationen mit dem Ziel, die **Nutzererfahrung** deutlich zu verbessern. Auf diese Weise konnten zahlreiche Patienten eine individuellere und flexiblere Betreuung erhalten, die einen schnelleren Zugang zu klinischen Ergebnissen und zur Gesundheitsüberwachung ermöglicht. Gleichzeitig hat er technologische Lösungen implementiert, die es Fachkräften ermöglicht haben, auf der Grundlage großer Datenmengen besser informierte **strategische Entscheidungen zu treffen.**

Er hat diese Arbeit zudem mit der Entwicklung von innovativen technologischen Projekten kombiniert, um die Effizienz der operativen Prozesse der Institutionen zu maximieren. In dieser Hinsicht hat er die **digitale Transformation** zahlreicher Unternehmen in verschiedenen Branchen geleitet. So hat er neue Tools wie **künstliche Intelligenz**, ***Big Data*** oder ***Machine Learning*** eingeführt, um komplexe tägliche Aufgaben zu automatisieren. Dadurch waren diese Unternehmen imstande, sich umgehend an Marktentwicklungen anzupassen und ihre langfristige Zukunftsfähigkeit zu sichern.

Es ist erwähnenswert, dass Daniel St. John als Redner an mehreren wissenschaftlichen Konferenzen auf der ganzen Welt teilgenommen hat. Auf diese Weise hat er sein umfangreiches Wissen in Bereichen wie der Einführung **agiler Methoden, Anwendungstests** zur Gewährleistung der Zuverlässigkeit von Systemen oder der Implementierung innovativer ***Blockchain***-Techniken, die den Schutz vertraulicher Daten garantieren, weitergegeben.

Hr. St. John, Daniel

- Direktor für Softwaretechnik bei General Electric Healthcare in Wisconsin, USA
- Leiter der Abteilung Softwaretechnik bei Siemens Healthineers, Illinois
- Direktor für Softwaretechnik bei Natus Medical Incorporated, Illinois
- Leitender Software-Ingenieur bei WMS Gaming in Chicago
- Leitender Software-Ingenieur bei Siemens Medical Solutions, Illinois
- Masterstudiengang in Datenstrategie und -analyse von der Lake Forest Graduate School of Management
- Hochschulabschluss in Informatik von der Universität von Wisconsin-Parkside
- Mitglied des Beratungsausschusses des Instituts für Technologie in Illinois
- Zertifizierungen in: Python für Datenwissenschaft, Künstliche Intelligenz und Entwicklung, SAFe SCRUM und Projektmanagement

Dank TECH werden Sie mit den besten Fachleuten der Welt lernen können"

Leitung

Hr. Molina Molina, Jerónimo

- IA Engineer & Software Architect NASSAT - Internet Satélite en Movimiento
- Consultor Sr. bei Hexa Ingenieros Einführung in die künstliche Intelligenz (ML und CV)
- Experte für auf künstlicher Intelligenz basierende Lösungen in den Bereichen Computer Vision, ML/DL und NLP Derzeit untersucht er die Möglichkeiten der Anwendung von Transformers und Reinforcement Learning in einem persönlichen Forschungsprojekt
- Universitätsexperte für Unternehmensgründung und -entwicklung Bancaixa–FUNDEUN Alicante
- Computer-Ingenieur Universität von Alicante
- Masterstudiengang in Künstliche Intelligenz Katholische Universität von Avila
- MBA-Executive Forum Europäischer Business Campus

Professoren

Hr. Pi Morell, Oriol

- Product Owner von Hosting und E-Mail CDMON
- Funktionsanalytiker und Software-Ingenieur in verschiedenen Organisationen wie Fihoca, Atmira, CapGemini
- Dozent für verschiedene Kurse wie BPM in CapGemini, ORACLE Forms CapGemini, Business Processes Atmira
- Hochschulabschluss in technischem Ingenieurwesen in Computer Management von der Autonomen Universität Madrid
- Masterstudiengang in Künstlicher Intelligenz
- Masterstudiengang in Business Management und Verwaltung MBA
- Masterstudiengang in Information Systems Management Lehrerfahrung
- Nachdiplomstudium in Design Pattern Offene Universität von Katalonien

Hr. Tenrero Morán, Marcos

- DevOps Ingenieur-Allot Communications
- Application Lifecycle Management & DevOps–Meta4 Spain Cegid
- QA Automation Engineer-Meta4 Spain Cegid
- Hochschulabschluss in Computertechnik an der Universität Rey Juan Carlos
- Professionelle Anwendungsentwicklung für Android-Universidad Galileo (Guatemala)
- Entwicklung von Cloud-Diensten (nodeJs, JavaScript, HTML5)-UPM
- Kontinuierliche Integration mit Jenkins-Meta4 Cegid
- Webentwicklung mit Angular-CLI (4), Ionic und nodeJS Meta4 - Universität Rey Juan Carlos

Dr. Peralta Martín-Palomino, Arturo

- CEO und CTO bei Prometeus Global Solutions
- CTO bei Korporate Technologies
- CTO bei AI Shephers GmbH
- Promotion in technischer Informatik an der Universität von Castilla la Mancha
- Promotion in Wirtschaftswissenschaften, Unternehmen und Finanzen an der Universität Camilo José Cela Außerordentlicher Promotionspreis
- Doktor der Psychologie an der Universität von Castilla la Mancha
- Masterstudiengang in fortgeschrittenen Informationstechnologien von der Universität von Castilla la Mancha
- Masterstudiengang MBA+E (Master in Business Administration and Organisational Engineering) an der Universität von Castilla la Mancha
- Außerordentlicher Professor, der an der Universität von Castilla la Mancha Bachelor- und Masterstudiengänge in Computertechnik unterrichtet
- Professor für den Masterstudiengang in Big Data und Datenwissenschaft an der Internationalen Universität von Valencia
- Professor für den Masterstudiengang in Industrie 4.0 und den Masterstudiengang in Industriedesign und Produktentwicklung
- Mitglied der SMILe-Forschungsgruppe der Universität von Castilla la Mancha

Fr. Martínez Cerrato, Yésica

- Technikerin für elektronische Sicherheitsprodukte bei Securitas Segurdad Spanien
- Business Intelligence Analyst bei Ricopia Technologies (Alcalá de Henares) Abschluss in elektronischer Kommunikationstechnik an der Polytechnischen Hochschule, Universität Alcalá
- Verantwortlich für die Schulung neuer Mitarbeiter in Vertriebsmanagement-Software (CRM, ERP, INTRANET), Produkte und Verfahren bei Ricopia Technologies (Alcalá de Henares)
- Verantwortlich für die Schulung neuer Stipendiaten, die in die Computer-Klassenzimmer integriert werden an der Universität von Alcalá
- Projektmanagerin im Bereich Großkundenintegration bei Correos y Telégrafos (Madrid)
- Computertechnikerin - Verantwortlich für die Computer-Klassenzimmer OTEC, Universität von Alcalá (Alcalá de Henares)
- Lehrerin für Computerkurse bei der Vereinigung ASALUMA (Alcalá de Henares)
- Stipendium für die Ausbildung zum Computertechniker in OTEC, Universität Alcala (Alcalá de Henares)

Unser Lehrkörper wird Ihnen sein ganzes Wissen zur Verfügung stellen, damit Sie auf dem neuesten Stand der Dinge sind“

05

Struktur und Inhalt

Die Struktur und der Inhalt dieses privaten Masterstudiengangs wurden entwickelt, um die wichtigsten Themen für die Entwicklung von Qualitätssoftware abzudecken. Bestehend aus 10 Lehrmodulen, die von der Entwicklung von Softwareprojekten, funktionaler und technischer Dokumentation, *Test Driven Developement* und den verschiedenen Methoden bis hin zur Implementierung fortgeschrittener praktischer Lösungen mit DevOps und kontinuierlicher Integration reichen und alle auf das Erreichen von Softwarequalität ausgerichtet sind. Die umfangreichen multimedialen Inhalte, die von fachkundigen Dozenten sorgfältig ausgewählt wurden, sind eine große Hilfe bei der Erleichterung des Studiums und dienen als Referenzmaterial für die Zukunft.

Die praktischen Fälle, die auf der Realität beruhen, werden dazu dienen, die Theorie, die während des Programms erlernt wird, zu verstärken und zu kontextualisieren"

Modul 1. Software-Qualität. TRL-Entwicklungsstufen

1.1. Elemente, die die Softwarequalität beeinflussen (I). Die technische Schuld
- 1.1.1. Die technische Schuld. Ursachen und Folgen
- 1.1.2. Software-Qualität. Allgemeine Grundsätze
- 1.1.3. Software mit und ohne Qualitätsprinzipien
  - 1.1.3.1. Konsequenzen
  - 1.1.3.2. Die Notwendigkeit der Anwendung von Qualitätsprinzipien bei Software
- 1.1.4. Software-Qualität. Typologie
- 1.1.5. Qualitätssoftware. Besondere Features

1.2. Elemente, die die Softwarequalität beeinflussen (II). Zugehörige Kosten
- 1.2.1. Software-Qualität. Beeinflussende Elemente
- 1.2.2. Software-Qualität. Missverständnisse
- 1.2.3. Software-Qualität. Zugehörige Kosten

1.3. Software-Qualitätsmodelle (I). Wissensmanagement
- 1.3.1. Allgemeine Qualitätsmodelle
  - 1.3.1.1. Total Quality Management
  - 1.3.1.2. Europäisches Modell für Business Excellence (EFQM)
  - 1.3.1.3. Sechs-Sigma-Modell
- 1.3.2. Wissensmanagement-Modelle
  - 1.3.2.1. Dyba Modell
  - 1.3.2.2. SEKS-Modell
- 1.3.3. Erlebnisfabrik und QIP-Paradigma
- 1.3.4. Modelle mit hoher Nutzungsqualität (25010)

1.4. Software-Qualitätsmodelle (III). Qualität bei Daten, Prozessen und SEI-Modellen
- 1.4.1. Modell der Datenqualität
- 1.4.2. Modellierung von Softwareprozessen
- 1.4.3. *Software & Systems Process Engineering Metamodel Specification* (SPEM)
- 1.4.4. SEI-Modelle
  - 1.4.4.1. CMMI
  - 1.4.4.2. SCAMPI
  - 1.4.4.3. IDEAL

1.5. ISO-Software-Qualitätsstandards (I). Analyse der Standards
- 1.5.1. ISO 9000-Norm
  - 1.5.1.1. ISO 9000-Norm
  - 1.5.1.2. ISO-Familie von Qualitätsstandards (9000)
- 1.5.2. Andere ISO-Normen zum Thema Qualität
- 1.5.3. Normen zur Qualitätsmodellierung (ISO 2501)
- 1.5.4. Normen zur Qualitätsmessung (ISO 2502n)

1.6. ISO-Software-Qualitätsstandards (II). Anforderungen und Bewertung
- 1.6.1. Normen für Qualitätsanforderungen (2503n)
- 1.6.2. Standards zur Qualitätsbewertung (2504n)
- 1.6.3. ISO/IEC 24744:2007

1.7. TRL-Entwicklungsstufen (I). Stufen 1 bis 4
- 1.7.1. TRL-Stufen
- 1.7.2. Stufe 1: Grundlegende Prinzipien
- 1.7.3. Stufe 2: Konzept und/oder Anwendung
- 1.7.4. Stufe 3: kritische analytische Funktion
- 1.7.5. Stufe 4: Komponentenvalidierung in einer Laborumgebung

1.8. TRL-Entwicklungsstufen (II). Stufen 5 bis 9
- 1.8.1. Stufe 5: Komponentenvalidierung in der entsprechenden Umgebung
- 1.8.2. Stufe 6: System/Subsystem-Modell
- 1.8.3. Stufe 7: Demonstration in realer Umgebung
- 1.8.4. Stufe 8: Vollständiges und zertifiziertes System
- 1.8.5. Stufe 9: Erfolg in realer Umgebung

1.9. TRL-Entwicklungsstufen. Verwendungen
- 1.9.1. Beispiel für ein Unternehmen mit Laborumgebung
- 1.9.2. Beispiel für ein FuEuI-Unternehmen
- 1.9.3. Beispiel für ein industrielles FuEuI-Unternehmen
- 1.9.4. Beispiel für ein Joint-Venture-Unternehmen zwischen Labor und Technik

1.10. Software-Qualität. Wichtige Details
- 1.10.1. Methodische Details
- 1.10.2. Technische Details
- 1.10.3. Details zum Software-Projektmanagement
  - 1.10.3.1. Qualität der IT-Systeme
  - 1.10.3.2. Qualität von Softwareprodukten
  - 1.10.3.3. Qualität der Softwareprozesse

Modul 2. Software-Projektentwicklung. Funktionelle und technische Dokumentation

2.1. Projektmanagement
- 2.1.1. Projektmanagement für Softwarequalität
- 2.1.2. Projektmanagement. Vorteile
- 2.1.3. Projektmanagement. Typologie

2.2. Methodik des Projektmanagements
- 2.2.1. Methodik des Projektmanagements
- 2.2.2. Projekt-Methoden. Typologie
- 2.2.3. Methodik des Projektmanagements. Anwendung

2.3. Phase der Bedarfsermittlung
- 2.3.1. Identifizierung der Projektanforderungen
- 2.3.2. Verwaltung von Projekttreffen
- 2.3.3. Vorzulegende Dokumentation

2.4. Modell
- 2.4.1. Anfangsphase
- 2.4.2. Analysephase
- 2.4.3. Bauphase
- 2.4.4. Testphase
- 2.4.5. Lieferung

2.5. Zu verwendendes Datenmodell
- 2.5.1. Festlegung des neuen Datenmodells
- 2.5.2. Identifizierung des Datenmigrationsplans
- 2.5.3. Datensatz

2.6. Auswirkungen auf andere Projekte
- 2.6.1. Auswirkungen eines Projekts. Beispiele
- 2.6.2. Risiken im Projekt
- 2.6.3. Risikomanagement

2.7. "*Must*" des Projekts
- 2.7.1. *Must* des Projekts
- 2.7.2. Die Identifizierung des *Must* des Projekts
- 2.7.3. Identifizierung der Ausführungspunkte für die Lieferung eines Projekts

2.8. Das Konstruktionsteam des Projekts
- 2.8.1. Rollen, die je nach Projekt zu spielen sind
- 2.8.2. Kontakt mit der Personalabteilung für die Rekrutierung
- 2.8.3. Leistungen und Projektzeitplan

2.9. Technische Aspekte eines Softwareprojekts
- 2.9.1. Projekt Architekt. Technische Aspekte
- 2.9.2. Technische Leiter
- 2.9.3. Aufbau des Softwareprojekts
- 2.9.4. Bewertung der Codequalität, SonarQube

2.10. Projektleistungen
- 2.10.1. Funktionsanalyse
- 2.10.2. Datenmodell
- 2.10.3. Zustandsdiagramm
- 2.10.4. Technische Dokumentation

Modul 3. *Software Testing.* Testautomatisierung

3.1. Software-Qualitätsmodelle
- 3.1.1. Produktqualität
- 3.1.2. Prozessqualität
- 3.1.3. Qualität der Nutzung

3.2. Prozessqualität
- 3.2.1. Prozessqualität
- 3.2.2. Reifegradmodelle
- 3.2.3. ISO 15504-Norm
  - 3.2.3.1. Verwendungszwecke
  - 3.2.3.2. Kontext
  - 3.2.3.3. Etappen

3.3. ISO/IEC 15504-Norm
- 3.3.1. Prozess-Kategorien
- 3.3.2. Entwicklungsprozess. Beispiel
- 3.3.3. Profil Fragment
- 3.3.4. Etappen

3.4. CMMI (*Capability Maturity Model Integration*)
- 3.4.1. CMMI. Integration des Capability Maturity Model
- 3.4.2. Modelle und Bereiche. Typologie
- 3.4.3. Prozessbereiche
- 3.4.4. Kapazitätsstufen
- 3.4.5. Prozessmanagement
- 3.4.6. Projektleitung

3.5. Verwaltung von Änderungen und Repositorys
- 3.5.1. Software Change Management
  - 3.5.1.1. Konfigurationselement. Kontinuierliche Integration
  - 3.5.1.2. Zeilen
  - 3.5.1.3. Flussdiagramme
  - 3.5.1.4. *Branches*
- 3.5.2. Repository
  - 3.5.2.1. Versionskontrolle
  - 3.5.2.2. Arbeitsteam und Nutzung des Repository
  - 3.5.2.3. Kontinuierliche Integration in das Repository

3.6. *Team Foundation Server* (TFS)
- 3.6.1. Installation und Konfiguration
- 3.6.2. Ein Team-Projekt erstellen
- 3.6.3. Hinzufügen von Inhalten zur Versionskontrolle
- 3.6.4. *TFS on Cloud*

3.7. *Testing*
- 3.7.1. Motivation für Tests
- 3.7.2. Verifikationsprüfung
- 3.7.3. Beta-Tests
- 3.7.4. Implementierung und Wartung

3.8. Belastungstests
- 3.8.1. *Load testing*
- 3.8.2. *LoadView*-Tests
- 3.8.3. Testen mit K6 *Cloud*
- 3.8.4. Testen mit *Loader*

3.9. Unit-, Stress- und Dauertests
- 3.9.1. Motivation für Unit-Tests
- 3.9.2. *Unit Testing* Tools
- 3.9.3. Motivation für Stresstests
- 3.9.4. Testen mit *StressTesting*
- 3.9.5. Motivation für Stresstests
- 3.9.6. Testen mit *LoadRunner*

3.10. Skalierbarkeit. Skalierbares Software-Design
- 3.10.1. Skalierbarkeit und Software-Architektur
- 3.10.2. Unabhängigkeit zwischen den Ebenen
- 3.10.3. Kopplung zwischen Schichten. Architektur-Muster

Modul 4. Software-Projektmanagement-Methoden. *Waterfall*-Methoden versus agile Methoden

4.1. *Waterfall*-Methode
- 4.1.1. *Waterfall*-Methode
- 4.1.2. *Waterfall*-Methode. Einfluss auf die Softwarequalität
- 4.1.3. *Waterfall*-Methode. Beispiele

4.2. Methodik *Agile*
- 4.2.1. Methodik *Agile*
- 4.2.2. Methodik *Agile*. Einfluss auf die Softwarequalität
- 4.2.3. Methodik *Agile*. Beispiele

4.3. Scrum-Methodik
- 4.3.1. Scrum-Methodik
- 4.3.2. Scrum Manifest
- 4.3.3. Scrum Implementierung

4.4. Kanban-Panel
- 4.4.1. Kanban-Methode
- 4.4.2. Kanban-Panel
- 4.4.3. Kanban-Panel. Beispiel einer Anwendung

4.5. Projektmanagement in *Waterfall*
- 4.5.1. Phasen eines Projekts
- 4.5.2. Projektvision in *Waterfall*
- 4.5.3. Zu berücksichtigende Leistungen

4.6. Projektmanagement in Scrum
- 4.6.1. Phasen eines Projekts Scrum
- 4.6.2. Projektvision in Scrum
- 4.6.3. Zu berücksichtigende Leistungen

4.7. *Waterfall* vs. Scrum Vergleich
- 4.7.1. Ansatz des Pilotprojekts
- 4.7.2. Projekt nach dem Prinzip *Waterfall*. Beispiel
- 4.7.3. Projekt nach dem Prinzip Scrum. Beispiel

4.8. Kundenvision
- 4.8.1. Dokumente in *Waterfall*
- 4.8.2. Dokumente in Scrum
- 4.8.3. Vergleich

4.9. Kanban Struktur
- 4.9.1. Anwenderberichte
- 4.9.2. *Backlog*
- 4.9.3. Kanban-Analyse

4.10. Hybride Projekte
- 4.10.1. Projekt Konstruktion
- 4.10.2. Projektleitung
- 4.10.3. Zu berücksichtigende Leistungen

Modul 5. TDD (*Test Driven Developement*). Testgetriebener Softwareentwurf

5.1. TDD. *Test Driven Development*
- 5.1.1. TDD. *Test Driven Development*
- 5.1.2. TDD. Einfluss von TDD auf die Qualität
- 5.1.3. Testgesteuertes Design und Entwicklung. Beispiele

5.2. TDD-Zyklus
- 5.2.1. Auswahl einer Anforderung
- 5.2.2. Testen. Typologien
  - 5.2.2.1. Einheitstests
  - 5.2.2.2. Integrationstests
  - 5.2.2.3. *End To End*-Tests
- 5.2.3. Prüfung des Tests. Misserfolge
- 5.2.4. Erstellung der Implementierung
- 5.2.5. Ausführung von automatisierten Tests
- 5.2.6. Eliminierung von Doppelarbeit
- 5.2.7. Aktualisierung der Liste der Anforderungen
- 5.2.8. Wiederholung des TDD-Zyklus
- 5.2.9. TDD-Zyklus. Theoretisches und praktisches Beispiel

5.3. TDD-Implementierungsstrategien
- 5.3.1. Mock-Implementierung
- 5.3.2. Dreieckige Implementierung
- 5.3.3. Offensichtliche Implementierung

5.4. TDD. Nutzung. Vorteile und Nachteile
- 5.4.1. Vorteile der Nutzung
- 5.4.2. Beschränkungen der Nutzung
- 5.4.3. Qualitätsbilanz in der Implementierung

5.5. TDD. Bewährte Verfahren
- 5.5.1. TDD-Regeln
- 5.5.2. Regel 1: Durchführung eines früheren Tests, falls dieser fehlschlägt, bevor in der Produktion programmiert wird
- 5.5.3. Regel 2: Nicht mehr als einen Einheitstest schreiben
- 5.5.4. Regel 3: Nicht mehr Code schreiben als nötig
- 5.5.5. Zu vermeidende Fehler und Anti-Patterns bei TDD

5.6. Simulation eines realen Projekts zur Anwendung von TDD (I)
- 5.6.1. Allgemeine Beschreibung des Projekts (Unternehmen A)
- 5.6.2. Implementierung von TDD
- 5.6.3. Vorgeschlagene Übungen
- 5.6.4. Übungen. *Feedback*

5.7. Simulation eines realen Projekts zur Anwendung von TDD (II)
- 5.7.1. Allgemeine Beschreibung des Projekts (Unternehmen B)
- 5.7.2. Anwendung von TDD
- 5.7.3. Vorgeschlagene Übungen
- 5.7.4. Übungen. *Feedback*

5.8. Simulation eines realen Projekts zur Anwendung von TDD (III)
- 5.8.1. Allgemeine Beschreibung des Projekts (Unternehmen C)
- 5.8.2. Anwendung von TDD
- 5.8.3. Vorgeschlagene Übungen
- 5.8.4. Übungen. *Feedback*

5.9. Alternativen zu TDD. *Test Driven Development*
- 5.9.1. TCR (*Test Commit Revert*)
- 5.9.2. BDD (*Behavior Driven Development*)
- 5.9.3. ATDD (*Acceptance Test Driven Development*)
- 5.9.4. TDD. Theoretischer Vergleich

5.10. TDD TCR, BDD und ATDD. Praktischer Vergleich
- 5.10.1. Problemstellung
- 5.10.2. Lösen mit TCR
- 5.10.3. Lösen mit BDD
- 5.10.4. Lösen mit ATDD

Modul 6. DevOps. Software-Qualitätsmanagement

6.1. DevOps. Software-Qualitätsmanagement
- 6.1.1. DevOps
- 6.1.2. DevOps und Softwarequalität
- 6.1.3. DevOps. Vorteile der DevOps-Kultur

6.2. DevOps. Beziehung zu Agile
- 6.2.1. Beschleunigte Lieferung
- 6.2.2. Qualität
- 6.2.3. Kostensenkung

6.3. Implementierung von DevOps
- 6.3.1. Identifizierung des Problems
- 6.3.2. Implementierung in einem Unternehmen
- 6.3.3. Metriken zur Implementierung

6.4. Software-Lieferzyklus
- 6.4.1. Design-Methoden
- 6.4.2. Abkommen
- 6.4.3. Roadmap

6.5. Entwicklung von fehlerfreiem Code
- 6.5.1. Wartbarer Code
- 6.5.2. Entwicklungsmuster
- 6.5.3. Code-*Testing*
- 6.5.4. Software-Entwicklung auf Code-Ebene. Bewährte Verfahren

6.6. Automatisierung
- 6.6.1. Automatisierung. Arten von Tests
- 6.6.2. Kosten für Automatisierung und Wartung
- 6.6.3. Automatisierung. Fehler abmildern

6.7. Einsätze
- 6.7.1. Zielbewertung
- 6.7.2. Entwurf eines automatischen und angepassten Prozesses
- 6.7.3. Feedback und Reaktionsfähigkeit

6.8. Management von Zwischenfällen
- 6.8.1. Bereitschaft für Zwischenfälle
- 6.8.2. Analyse und Lösung von Vorfällen
- 6.8.3. Künftige Fehler vermeiden

6.9. Automatisierung des Einsatzes
- 6.9.1. Vorbereitungen für automatisierte Einsätze
- 6.9.2. Automatische Bewertung des Prozesszustands
- 6.9.3. Metriken und Rollback-Fähigkeit

6.10. Bewährte Verfahren. Entwicklung von DevOps
- 6.10.1. DevOps Leitfaden für bewährte Verfahren
- 6.10.2. DevOps. Methodik für das Team
- 6.10.3. Nischen meiden

Modul 7. DevOps und kontinuierliche Integration. Fortgeschrittene praktische Lösungen in der Softwareentwicklung

7.1. Ablauf der Softwarelieferung
- 7.1.1. Identifizierung von Akteuren und Artefakten
- 7.1.2. Entwurf des Softwareentwicklungsprozesses
- 7.1.3. Ablauf der Softwareentwicklung. Anforderungen zwischen den Phasen

7.2. Prozessautomatisierung
- 7.2.1. Kontinuierliche Integration
- 7.2.2. Kontinuierliche Bereitstellung
- 7.2.3. Konfiguration von Umgebungen und Verwaltung von Geheimnissen

7.3. Deklarative Pipelines
- 7.3.1. Unterschiede zwischen traditionellen, codeähnlichen und deklarativen Pipelines
- 7.3.2. Deklarative Pipelines
- 7.3.3. Deklarative Pipelines in Jenkins
- 7.3.4. Vergleich der Anbieter von kontinuierlicher Integration

7.4. Qualitätsprüfpunkte und erweitertes Feedback
- 7.4.1. Qualitätsprüfpunkte
- 7.4.2. Qualitätsstandards mit Qualitätsprüfpunkten. Wartung
- 7.4.3. Geschäftsanforderungen für Integrationsanfragen

7.5. Verwaltung von Artefakten
- 7.5.1. Artefakte und Lebenszyklus
- 7.5.2. Systeme zur Aufbewahrung und Verwaltung von Artefakten
- 7.5.3. Sicherheit bei der Verwaltung von Artefakten

7.6. Kontinuierliche Bereitstellung
- 7.6.1. Kontinuierliche Bereitstellung in Containern
- 7.6.2. Kontinuierliche Bereitstellung mit PaaS
- 7.6.3. Kontinuierliche Bereitstellung von mobilen Anwendungen

7.7. Verbesserung der Pipeline-Laufzeit: statische Analyse und *Git Hooks*
- 7.7.1. Statische Analyse
- 7.7.2. Code-Stilregeln
- 7.7.3. *Git Hooks* und Einheit*tests*
- 7.7.4. Die Auswirkungen der Infrastruktur

7.8. Container-Schwachstellen
- 7.8.1. Container-Schwachstellen
- 7.8.2. Scannen von Bildern
- 7.8.3. Regelmäßige Berichte und Warnmeldungen

Modul 8. Datenbank-Design (DB). Standardisierung und Leistung. Software-Qualität

8.1. Entwurf von Datenbanken
- 8.1.1. Datenbanken. Typologie
- 8.1.2. Derzeit verwendete Datenbanken
  - 8.1.2.1. Relational
  - 8.1.2.2. Schlüssel-Wert
  - 8.1.2.3. Netzwerkbasiert
- 8.1.3. Datenqualität

8.2. Entwurf eines Entity-Relationship-Modells (I)
- 8.2.1. Entity-Relationship-Modell. Qualität und Dokumentation
- 8.2.2. Einheiten
  - 8.2.2.1. Starke Einheit
  - 8.2.2.2. Schwache Einheit
- 8.2.3. Attribute
- 8.2.4. Beziehungsset
  - 8.2.4.1. 1 zu 1
  - 8.2.4.2. 1 zu vielen
  - 8.2.4.3. Viele zu 1
  - 8.2.4.4. Viele zu viele
- 8.2.5. Schlüssel
  - 8.2.5.1. Primärschlüssel
  - 8.2.5.2. Fremdschlüssel
  - 8.2.5.3. Schwacher Primärschlüssel der Einheit
- 8.2.6. Beschränkungen
- 8.2.7. Kardinalität
- 8.2.8. Vererbung
- 8.2.9. Aggregation

8.3. Entity-Relationship-Modells (II). Tools
- 8.3.1. Entity-Relationship-Modell. Tools
- 8.3.2. Entity-Relationship-Modell. Praktisches Beispiel
- 8.3.3. Durchführbares Entity-Relationship-Modell
  - 8.3.3.1. Visuelles Beispiel
  - 8.3.3.2. Beispiel in tabellarischer Darstellung

8.4. Standardisierung von Datenbanken (DB) (I). Überlegungen zur Softwarequalität
- 8.4.1. DB Standardisierung und Qualität
- 8.4.2. Abhängigkeit
  - 8.4.2.1. Funktionsabhängigkeit
  - 8.4.2.2. Eigenschaften der Funktionsabhängigkeit
  - 8.4.2.3. Abgeleitete Eigenschaften
- 8.4.3. Schlüssel

8.5. Standardisierung von Datenbanken (DB) (II). Normalformen und Codd-Regeln
- 8.5.1. Normale Formen
  - 8.5.1.1. Erste Normalform (1NF)
  - 8.5.1.2. Zweite Normalform (2NF)
  - 8.5.1.3. Dritte Normalform (3NF)
  - 8.5.1.4. Boyce-Codd-Normalform (BCNF)
  - 8.5.1.5. Vierte Normalform (4NF)
  - 8.5.1.6. Fünfte Normalform (5NF)
- 8.5.2. Codd's Regeln
  - 8.5.2.1. Regel 1: Information
  - 8.5.2.2. Regel 2: Garantierter Zugang
  - 8.5.2.3. Regel 3: Systematische Behandlung von Nullwerten
  - 8.5.2.4. Regel 4: Beschreibung der Datenbank
  - 8.5.2.5. Regel 5: Integrale Untersprache
  - 8.5.2.6. Regel 6: Ansicht aktualisieren
  - 8.5.2.7. Regel 7: Einfügen und Aktualisieren
  - 8.5.2.8. Regel 8: Körperliche Unabhängigkeit
  - 8.5.2.9. Regel 9: Logische Unabhängigkeit
  - 8.5.2.10. Regel 10: Unabhängigkeit der Integrität
    - 8.5.2.10.1. Integritätsregeln
  - 8.5.2.11. Regel 11: Verteilung
  - 8.5.2.12. Regel 12: Nicht-Subversion
- 8.5.3. Praktisches Beispiel

8.6. Datenlager / OLAP-System
- 8.6.1. Data Warehouse
- 8.6.2. Faktentabelle
- 8.6.3. Tabelle der Abmessungen
- 8.6.4. Erstellung des OLAP-Systems. Tools

8.7. Leistung der Datenbank (DB)
- 8.7.1. Index-Optimierung
- 8.7.2. Optimierung von Abfragen
- 8.7.3. Tabelle Partitionierung

8.8. Simulation des realen Projekts für DB-Design (I)
- 8.8.1. Projektübersicht (Unternehmen A)
- 8.8.2. Anwendung von Datenbankdesign
- 8.8.3. Vorgeschlagene Übungen
- 8.8.4. Vorgeschlagene Übungen. *Feedback*

8.9. Simulation des realen Projekts für DB-Design (II)
- 8.9.1. Allgemeine Beschreibung des Projekts (Unternehmen B)
- 8.9.2. Anwendung von Datenbankdesign
- 8.9.3. Vorgeschlagene Übungen
- 8.9.4. Vorgeschlagene Übungen. *Feedback*

8.10. Relevanz der DB-Optimierung für die Softwarequalität
- 8.10.1. Design-Optimierung
- 8.10.2. Optimierung des Abfragecodes
- 8.10.3. Optimierung von gespeichertem Prozedur-Code
- 8.10.4. Der Einfluss von *Triggers* auf die Softwarequalität. Empfehlungen für die Verwendung

Modul 9. Entwurf skalierbarer Architekturen. Architektur im Software-Lebenszyklus

9.1. Entwurf skalierbarer Architekturen (I)
- 9.1.1. Skalierbare Architekturen
- 9.1.2. Grundsätze einer skalierbaren Architektur
  - 9.1.2.1. Zuverlässig
  - 9.1.2.2. Skalierbar
  - 9.1.2.3. Wartbar
- 9.1.3. Arten der Skalierbarkeit
  - 9.1.3.1. Vertikal
  - 9.1.3.2. Horizontal
  - 9.1.3.3. Kombiniert

9.2. Architekturen DDD (*Domain-Driven Design*)
- 9.2.1. DDD-Modell. Domain-Ausrichtung
- 9.2.2. Schichten, Aufteilung der Verantwortung und Entwurfsmuster
- 9.2.3. Entkopplung als Grundlage für Qualität

9.3. Entwurf skalierbarer Architekturen (II). Vorteile, Einschränkungen und Designstrategien
- 9.3.1. Skalierbare Architektur. Vorteile
- 9.3.2. Skalierbare Architektur. Beschränkungen
- 9.3.3. Strategien für die Entwicklung skalierbarer Architekturen (Beschreibende Tabelle)

9.4. Lebenszyklus der Software (I). Etappen
- 9.4.1. Lebenszyklus der Software
  - 9.4.1.1. Planungsphase
  - 9.4.1.2. Analysephase
  - 9.4.1.3. Entwurfsphase
  - 9.4.1.4. Phase der Umsetzung
  - 9.4.1.5. Testphase
  - 9.4.1.6. Phase der Installation/Einrichtung
  - 9.4.1.7. Phase der Nutzung und Pflege

9.5. Software-Lebenszyklus-Modelle
- 9.5.1. Wasserfall-Modell
- 9.5.2. Wiederholtes Modell
- 9.5.3. Spiralförmiges Modell
- 9.5.4. *Big Bang* Modell

9.6. Lebenszyklus der Software (II). Automatisierung
- 9.6.1. Lebenszyklus der Softwareentwicklung. Lösungen
  - 9.6.1.1. Kontinuierliche Integration und Entwicklung (CI/CD)
  - 9.6.1.2. Agile Methodologien
  - 9.6.1.3. DevOps / Produktionsbetrieb
- 9.6.2. Zukünftige Trends
- 9.6.3. Praktische Beispiele

9.7. Software-Architektur im Software-Lebenszyklus
- 9.7.1. Vorteile
- 9.7.2. Beschränkungen
- 9.7.3. Tools

9.8. Simulation eines realen Projekts zum Entwurf einer Software-Architektur (I)
- 9.8.1. Projektübersicht (Unternehmen A)
- 9.8.2. Anwendung Softwarearchitektur-Design
- 9.8.3. Vorgeschlagene Übungen
- 9.8.4. Vorgeschlagene Übungen. *Feedback*

9.9. Simulation eines realen Projekts zum Entwurf einer Software-Architektur (II)
- 9.9.1. Allgemeine Beschreibung des Projekts (Unternehmen B)
- 9.9.2. Anwendung Softwarearchitektur-Design
- 9.9.3. Vorgeschlagene Übungen
- 9.9.4. Vorgeschlagene Übungen. *Feedback*

9.10. Simulation eines realen Projekts zum Entwurf einer Software-Architektur (III)
- 9.10.1. Allgemeine Beschreibung des Projekts (Unternehmen C)
- 9.10.2. Anwendung Softwarearchitektur-Design
- 9.10.3. Vorgeschlagene Übungen
- 9.10.4. Vorgeschlagene Übungen. *Feedback*

Modul 10. ISO, IEC 9126 Qualitätskriterien. Metriken zur Software-Qualität

10.1. Qualitätskriterien. ISO/IEC 9126 Norm
- 10.1.1. Qualitätskriterien
- 10.1.2. Software-Qualität. Rechtfertigung. ISO/IEC 9126 Norm
- 10.1.3. Messung der Softwarequalität als Schlüsselindikator

10.2. Qualitätskriterien für Software. Eigenschaften
- 10.2.1. Verlässlichkeit
- 10.2.2. Funktionsweise
- 10.2.3. Effizienz
- 10.2.4. Benutzerfreundlichkeit
- 10.2.5. Instandhaltbarkeit
- 10.2.6. Tragbarkeit
- 10.2.7. Sicherheit

10.3. ISO, IEC 9126 (I). Präsentation
- 10.3.1. Beschreibung von ISO, IEC 9126
- 10.3.2. Funktionsweise
- 10.3.3. Verlässlichkeit
- 10.3.4. Benutzerfreundlichkeit
- 10.3.5. Instandhaltbarkeit
- 10.3.6. Übertragbarkeit
- 10.3.7. Qualität im Einsatz
- 10.3.8. Metriken zur Softwarequalität
- 10.3.9. Qualitätsmetriken in ISO 9126

10.4. ISO, IEC 9126 (II). McCall und Boehm Modelle
- 10.4.1. McCall Modell: Qualitätsfaktoren
- 10.4.2. Böhm Modell
- 10.4.3. Mittleres Niveau. Eigenschaften

10.5. Metriken zur Softwarequalität (I). Elemente
- 10.5.1. Messung
- 10.5.2. Metrik
- 10.5.3. Indikator
  - 10.5.3.1. Arten von Indikatoren
- 10.5.4. Maßnahmen und Modelle
- 10.5.5. Umfang der Software Metriken
- 10.5.6. Klassifizierung von Softwaremetriken

10.6. Messung der Softwarequalität (II). Praxis der Messung
- 10.6.1. Metrische Datenerfassung
- 10.6.2. Messung der internen Produkteigenschaften
- 10.6.3. Messung von externen Produktattributen
- 10.6.4. Messung der Ressourcen
- 10.6.5. Metriken für objektorientierte Systeme

10.7. Entwerfen eines einzigen Software-Qualitätsindikators
- 10.7.1. Einzelner Indikator als Global Scorer
- 10.7.2. Entwicklung, Rechtfertigung und Anwendung von Indikatoren
- 10.7.3. Beispiel für eine Anwendung. Notwendigkeit, die Details zu kennen

10.8. Simulation eines realen Projekts zur Qualitätsmessung (I)
- 10.8.1. Projektübersicht (Unternehmen A)
- 10.8.2. Anwendung der Qualitätsmessung
- 10.8.3. Vorgeschlagene Übungen
- 10.8.4. Vorgeschlagene Übungen. *Feedback*

10.9. Simulation eines realen Projekts zur Qualitätsmessung (II)
- 10.9.1. Allgemeine Beschreibung des Projekts (Unternehmen B)
- 10.9.2. Anwendung der Qualitätsmessung
- 10.9.3. Vorgeschlagene Übungen
- 10.9.4. Vorgeschlagene Übungen. *Feedback*

10.10. Simulation eines realen Projekts zur Qualitätsmessung (III)
- 10.10.1. Allgemeine Beschreibung des Projekts (Unternehmen C)
- 10.10.2. Anwendung der Qualitätsmessung
- 10.10.3. Vorgeschlagene Übungen
- 10.10.4. Vorgeschlagene Übungen. *Feedback*

06

Methodik

Dieses Fortbildungsprogramm bietet eine andere Art des Lernens. Unsere Methodik wird durch eine zyklische Lernmethode entwickelt: ***das Relearning***.

Dieses Lehrsystem wird z. B. an den renommiertesten medizinischen Fakultäten der Welt angewandt und wird von wichtigen Publikationen wie dem ***New England Journal of Medicine*** als eines der effektivsten angesehen.

Entdecken Sie Relearning, ein System, das das herkömmliche lineare Lernen aufgibt und Sie durch zyklische Lehrsysteme führt: eine Art des Lernens, die sich als äußerst effektiv erwiesen hat, insbesondere in Fächern, die Auswendiglernen erfordern"

Fallstudie zur Kontextualisierung aller Inhalte

Unser Programm bietet eine revolutionäre Methode zur Entwicklung von Fähigkeiten und Kenntnissen. Unser Ziel ist es, Kompetenzen in einem sich wandelnden, wettbewerbsorientierten und sehr anspruchsvollen Umfeld zu stärken.

Mit TECH werden Sie eine Art des Lernens erleben, die die Grundlagen der traditionellen Universitäten in der ganzen Welt verschiebt"

Sie werden Zugang zu einem Lernsystem haben, das auf Wiederholung basiert, mit natürlichem und progressivem Unterricht während des gesamten Lehrplans.

Die Studenten lernen durch gemeinschaftliche Aktivitäten und reale Fälle die Lösung komplexer Situationen in realen Geschäftsumgebungen.

Eine innovative und andersartige Lernmethode

Dieses TECH-Programm ist ein von Grund auf neu entwickeltes, intensives Lehrprogramm, das die anspruchsvollsten Herausforderungen und Entscheidungen in diesem Bereich sowohl auf nationaler als auch auf internationaler Ebene vorsieht. Dank dieser Methodik wird das persönliche und berufliche Wachstum gefördert und ein entscheidender Schritt in Richtung Erfolg gemacht. Die Fallmethode, die Technik, die diesem Inhalt zugrunde liegt, gewährleistet, dass die aktuellste wirtschaftliche, soziale und berufliche Realität berücksichtigt wird.

> *Unser Programm bereitet Sie darauf vor, sich neuen Herausforderungen in einem unsicheren Umfeld zu stellen und in Ihrer Karriere erfolgreich zu sein"*

Die Fallmethode ist das am weitesten verbreitete Lernsystem an den besten Informatikschulen der Welt, seit es sie gibt. Die Fallmethode wurde 1912 entwickelt, damit die Jurastudenten das Recht nicht nur anhand theoretischer Inhalte erlernen, sondern ihnen reale, komplexe Situationen vorlegen, damit sie fundierte Entscheidungen treffen und Werturteile darüber fällen können, wie diese zu lösen sind. Sie wurde 1924 als Standardlehrmethode in Harvard eingeführt.

Was sollte eine Fachkraft in einer bestimmten Situation tun? Mit dieser Frage konfrontieren wir Sie in der Fallmethode, einer handlungsorientierten Lernmethode. Während des gesamten Kurses werden die Studierenden mit mehreren realen Fällen konfrontiert. Sie müssen Ihr gesamtes Wissen integrieren, recherchieren, argumentieren und Ihre Ideen und Entscheidungen verteidigen.

Relearning Methodik

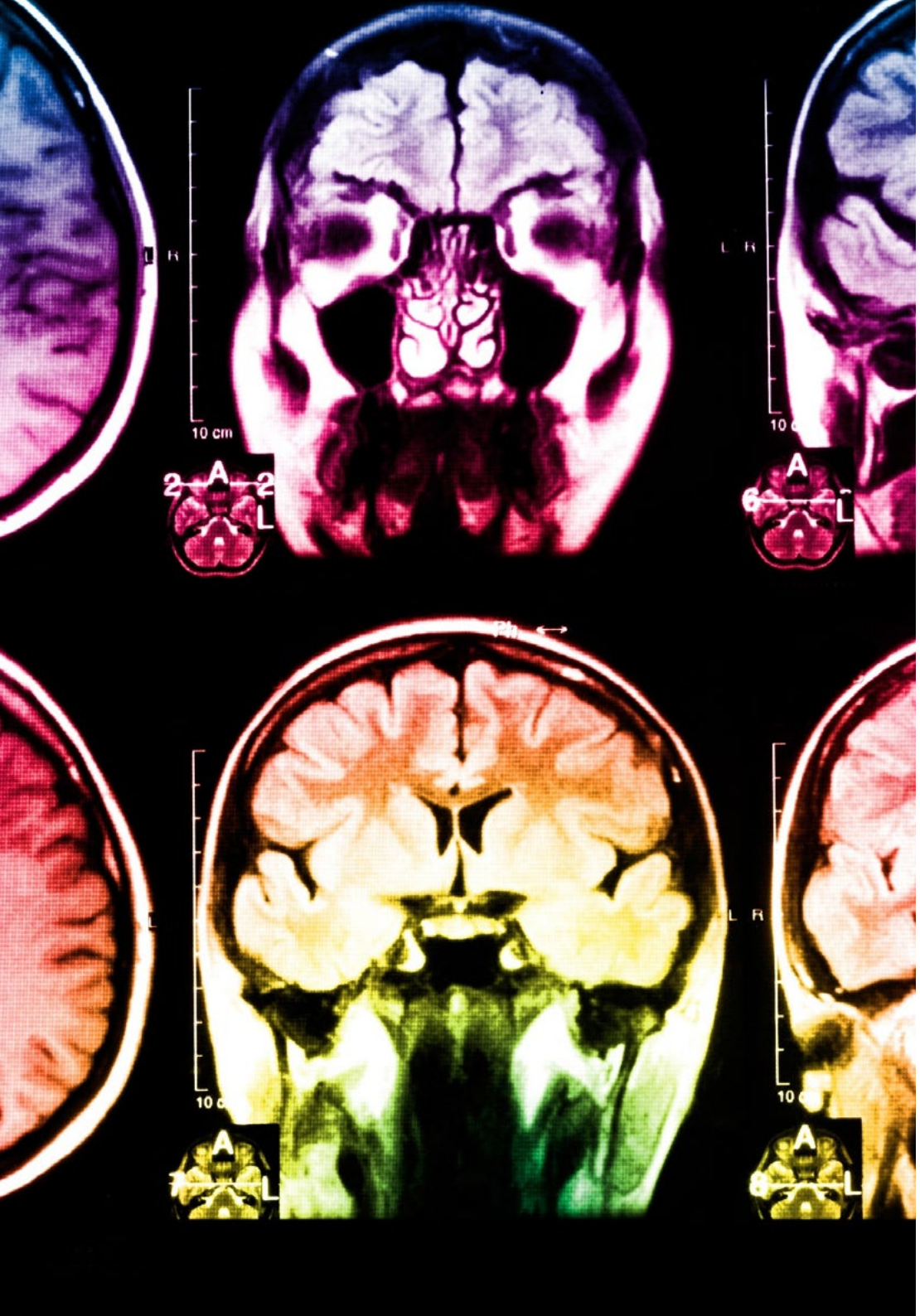
TECH kombiniert die Methodik der Fallstudien effektiv mit einem 100%igen Online-Lernsystem, das auf Wiederholung basiert und in jeder Lektion verschiedene didaktische Elemente kombiniert.

Wir ergänzen die Fallstudie mit der besten 100%igen Online-Lehrmethode: Relearning.

Im Jahr 2019 erzielten wir die besten Lernergebnisse aller spanischsprachigen Online-Universitäten der Welt.

Bei TECH lernen Sie mit einer hochmodernen Methodik, die darauf ausgerichtet ist, die Führungskräfte der Zukunft auszubilden. Diese Methode, die an der Spitze der weltweiten Pädagogik steht, wird Relearning genannt.

Unsere Universität ist die einzige in der spanischsprachigen Welt, die für die Anwendung dieser erfolgreichen Methode zugelassen ist. Im Jahr 2019 ist es uns gelungen, die Gesamtzufriedenheit unserer Studenten (Qualität der Lehre, Qualität der Materialien, Kursstruktur, Ziele...) in Bezug auf die Indikatoren der besten Online-Universität in Spanisch zu verbessern.



In unserem Programm ist das Lernen kein linearer Prozess, sondern erfolgt in einer Spirale (lernen, verlernen, vergessen und neu lernen). Daher wird jedes dieser Elemente konzentrisch kombiniert. Mit dieser Methode wurden mehr als 650.000 Hochschulabsolventen mit beispiellosem Erfolg in so unterschiedlichen Bereichen wie Biochemie, Genetik, Chirurgie, internationales Recht, Managementfähigkeiten, Sportwissenschaft, Philosophie, Recht, Ingenieurwesen, Journalismus, Geschichte, Finanzmärkte und -Instrumente ausgebildet. Dies alles in einem sehr anspruchsvollen Umfeld mit einer Studentenschaft mit hohem sozioökonomischem Profil und einem Durchschnittsalter von 43,5 Jahren.

Das Relearning ermöglicht es Ihnen, mit weniger Aufwand und mehr Leistung zu lernen, sich mehr auf Ihr Fachgebiet einzulassen, einen kritischen Geist zu entwickeln, Argumente zu verteidigen und Meinungen zu kontrastieren: eine direkte Gleichung zum Erfolg.

Nach den neuesten wissenschaftlichen Erkenntnissen der Neurowissenschaften wissen wir nicht nur, wie wir Informationen, Ideen, Bilder und Erinnerungen organisieren, sondern auch, dass der Ort und der Kontext, in dem wir etwas gelernt haben, von grundlegender Bedeutung dafür sind, dass wir uns daran erinnern und es im Hippocampus speichern können, um es in unserem Langzeitgedächtnis zu behalten.

Auf diese Weise sind die verschiedenen Elemente unseres Programms im Rahmen des so genannten neurokognitiven kontextabhängigen E-Learnings mit dem Kontext verbunden, in dem der Teilnehmer seine berufliche Praxis entwickelt.

Dieses Programm bietet die besten Lehrmaterialien, die sorgfältig für Fachleute aufbereitet sind:

Studienmaterial

Alle didaktischen Inhalte werden von den Fachleuten, die den Kurs unterrichten werden, speziell für den Kurs erstellt, so dass die didaktische Entwicklung wirklich spezifisch und konkret ist.

Diese Inhalte werden dann auf das audiovisuelle Format angewendet, um die TECH-Online-Arbeitsmethode zu schaffen. Und das alles mit den neuesten Techniken, die dem Studenten qualitativ hochwertige Stücke aus jedem einzelnen Material zur Verfügung stellen.

Meisterklassen

Die Nützlichkeit der Expertenbeobachtung ist wissenschaftlich belegt.

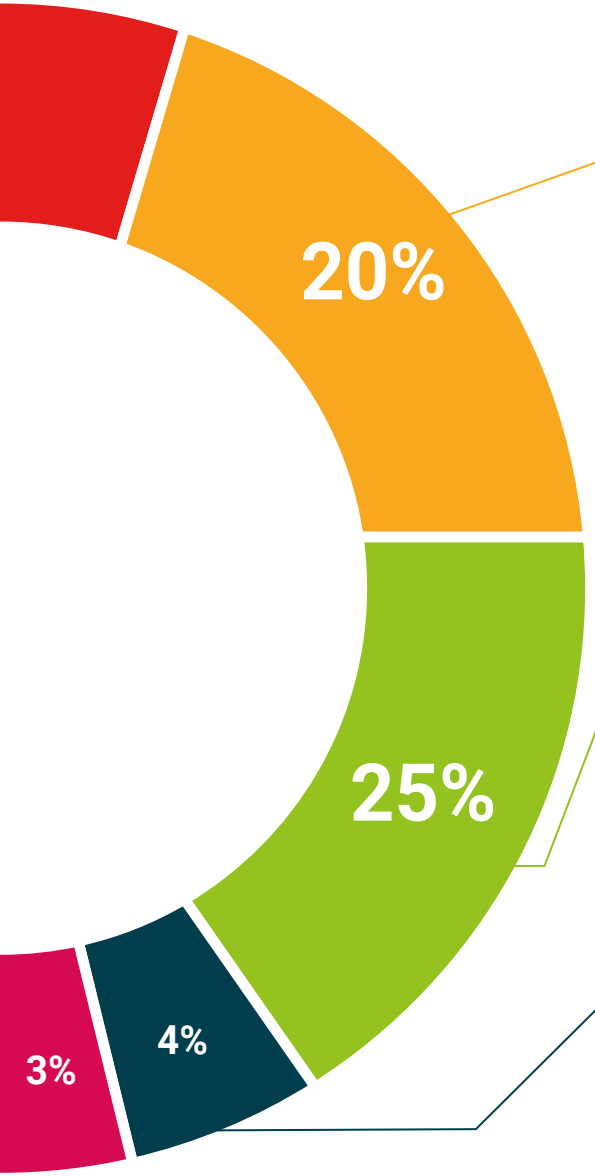
Das sogenannte Learning from an Expert baut Wissen und Gedächtnis auf und schafft Vertrauen für zukünftige schwierige Entscheidungen.

Fertigkeiten und Kompetenzen Praktiken

Sie werden Aktivitäten durchführen, um spezifische Kompetenzen und Fertigkeiten in jedem Fachbereich zu entwickeln. Praktiken und Dynamiken zum Erwerb und zur Entwicklung der Fähigkeiten und Fertigkeiten, die ein Spezialist im Rahmen der Globalisierung, in der wir leben, entwickeln muss.

Weitere Lektüren

Aktuelle Artikel, Konsensdokumente und internationale Leitfäden, u.a. In der virtuellen Bibliothek von TECH haben die Studenten Zugang zu allem, was sie für ihre Ausbildung benötigen.



Fallstudien

Sie werden eine Auswahl der besten Fallstudien vervollständigen, die speziell für diese Qualifizierung ausgewählt wurden. Die Fälle werden von den besten Spezialisten der internationalen Szene präsentiert, analysiert und betreut.

Interaktive Zusammenfassungen

Das TECH-Team präsentiert die Inhalte auf attraktive und dynamische Weise in multimedialen Pillen, die Audios, Videos, Bilder, Diagramme und konzeptionelle Karten enthalten, um das Wissen zu vertiefen.

Dieses einzigartige Bildungssystem für die Präsentation multimedialer Inhalte wurde von Microsoft als "europäische Erfolgsgeschichte" ausgezeichnet.

Prüfung und Nachprüfung

Die Kenntnisse der Studenten werden während des gesamten Programms regelmäßig durch Bewertungs- und Selbsteinschätzungsaktivitäten und -übungen beurteilt und neu bewertet, so dass die Studenten überprüfen können, wie sie ihre Ziele erreichen.

07

Qualifizierung

Der Privater Masterstudiengang in Software-Qualität garantiert neben der strengsten und aktuellsten Ausbildung auch den Zugang zu einem von der TECH Technologischen Universität ausgestellten Diplom.

Schließen Sie dieses Programm erfolgreich ab und erhalten Sie Ihren Universitätsabschluss ohne lästige Reisen oder Formalitäten"

Dieser **Privater Masterstudiengang in Software-Qualität** enthält das vollständigste und aktuellste Programm auf dem Markt.

Sobald der Student die Prüfungen bestanden hat, erhält er/sie per Post* mit Empfangsbestätigung das entsprechende Diplom, ausgestellt von der **TECH Technologischen Universität**.

Das von **TECH Technologische Universität** ausgestellte Diplom drückt die erworbene Qualifikation aus und entspricht den Anforderungen, die in der Regel von Stellenbörsen, Auswahlprüfungen und Berufsbildungsausschüssen verlangt werden.

Titel: **Privater Masterstudiengang in Software-Qualität**

Anzahl der offiziellen Arbeitsstunden: **1.500 Std.**

tech technologische universität

Verleiht dieses

DIPLOM

an

Herr/Frau _____, mit Ausweis-Nr. _____

Für den erfolgreichen Abschluss und die Akkreditierung des Programms

PRIVATER MASTERSTUDIENGANG

in

Software-Qualität

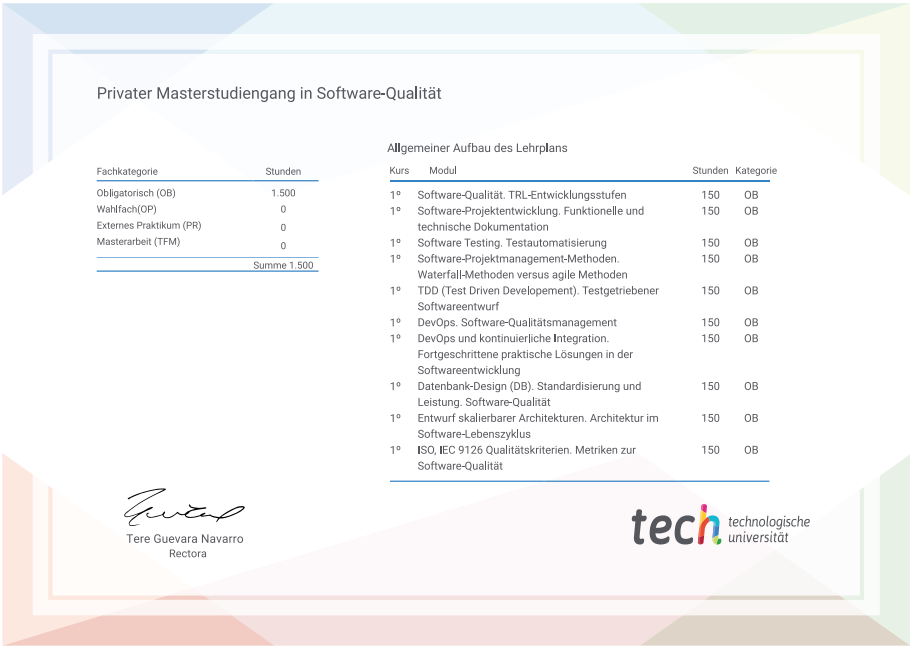
Es handelt sich um einen von dieser Universität verliehenen Abschluss, mit einer Dauer von 1.500 Stunden, mit Anfangsdatum tt/mm/jjjj und Enddatum tt/mm/jjjj.

TECH ist eine private Hochschuleinrichtung, die seit dem 28. Juni 2018 vom Ministerium für öffentliche Bildung anerkannt ist.

Zum 17. Juni 2020

Tere Guevara Navarro
Rectora

Diese Qualifikation muss immer mit einem Hochschulabschluss einhergehen, der von der für die Berufsausübung zuständigen Behörde des jeweiligen Landes ausgestellt wurde. einzigartiger Code TECH: AFWOR235 techtitute.com/titel



Privater Masterstudiengang in Software-Qualität

Fachkategorie	Stunden
Obligatorisch (OB)	1.500
Wahlfach(OP)	0
Externes Praktikum (PR)	0
Masterarbeit (TFM)	0
	Summe 1.500

Allgemeiner Aufbau des Lehrplans

Kurs	Modul	Stunden	Kategorie
1º	Software-Qualität. TRL-Entwicklungsstufen	150	OB
1º	Software-Projektentwicklung. Funktionelle und technische Dokumentation	150	OB
1º	Software Testing. Testautomatisierung	150	OB
1º	Software-Projektmanagement-Methoden. Waterfall-Methoden versus agile Methoden	150	OB
1º	TDD (Test Driven Developement). Testgetriebener Softwareentwurf	150	OB
1º	DevOps. Software-Qualitätsmanagement	150	OB
1º	DevOps und kontinuierliche Integration. Fortgeschrittene praktische Lösungen in der Softwareentwicklung	150	OB
1º	Datenbank-Design (DB). Standardisierung und Leistung. Software-Qualität	150	OB
1º	Entwurf skalierbarer Architekturen. Architektur im Software-Lebenszyklus	150	OB
1º	ISO, IEC 9126 Qualitätskriterien. Metriken zur Software-Qualität	150	OB

Tere Guevara Navarro
Rectora

tech technologische universität

*Haager Apostille. Für den Fall, dass der Student die Haager Apostille für sein Papierdiplom beantragt, wird TECH EDUCATION die notwendigen Vorkehrungen treffen, um diese gegen eine zusätzliche Gebühr zu beschaffen.

zukunft
gesundheit vertrauen menschen
erziehung information tutoren
garantie akkreditierung unterricht
institutionen technologie lernen
gemeinschaft verpflichtung
persönliche betreuung innovation
wissen gegenwart qualität
online-Ausbildung
entwicklung institutionen
virtuelles Klassenzimmer studieren

tech technologische universität

Privater Masterstudiengang
Software-Qualität

- Modalität: **online**
- Dauer: **12 Monate**
- Qualifizierung: **TECH Technologische Universität**
- Aufwand: **16 Std./Woche**
- Zeitplan: **in Ihrem eigenen Tempo**
- Prüfungen: **online**

Privater Masterstudiengang
Software-Qualität

tech technologische universität