www.trafonetz.de

WEITERBILDUNGSMODUL

SOFTWARE-ARCHITEKTUR

Anwendungsnahe Lernbausteine für Zukunftsthemen
in Ihrem Unternehmen

Transformationsnetzwerk
Nordschwarzwald

Ein Projekt der
REGION NORDSCHWARZWALD
Wirtschaftsförderung

Über das Weiterbildungsmodul

SOFTWARE-ARCHITEKTUR

Die Komplexität moderner IT-Anwendungen erfordert eine gut durchdachte Softwarearchitektur. Dabei ist insbesondere eine saubere Schnittstellengestaltung entscheidend für die reibungslose Integration neuer Softwaremodule. Kleine und mittlere Unternehmen (KMU) benötigen ein Grundverständnis von Softwarearchitektur, um geeignete Technologien für ihre IT-Anforderungen auszuwählen und eine hohe Performance und Sicherheit der Software im Unternehmen sicherzustellen.

Perspektiven und Lerninhalte

Das Weiterbildungsmodul unterstützt produzierende KMU bei der Erhebung der unternehmensspezifischen Anforderungen sowie der Schnittstellengestaltung, z. B. für Änderungskonfigurationen durch neue Produkte oder Erweiterungen der Produktionslinien. Die Lerninhalte umfassen u. a.:

- Anforderungsanalyse für die Softwarearchitektur
- Auswahl geeigneter Technologien für spezifische Anwendungsfälle
- Integration und API-Gestaltung von Softwaremodulen

Über die Weiterbildungsbausteine können produzierende KMU, ihre bestehende Softwarearchitektur hinsichtlich Performance- und Sicherheitsaspekte optimieren. Dazu dienen u. a. die Lerninhalte:

- Optimierung der Softwareleistung
- Sicherheitsanforderungen für die Softwarearchitektur
- Sicherheits- und Performance-Tests

Lernformat

Das Lernen findet selbstgesteuert am Arbeitsplatz statt. Die Mitarbeitenden erschließen sich die Kompetenzen schrittweise anhand von Lernaufgaben.

Voraussetzungen und Zielgruppen

Das Weiterbildungsmodul besteht aus verschiedenen Lernbausteinen und ermöglicht dadurch individuelle Lernpfade. Der Awareness-Lernbaustein bietet einen ersten, niederschwelligen Einstieg für alle Mitarbeitenden. Die weiteren Lernbausteine adressieren abhängig vom individuellen Wissensstand im Bereich Softwarearchitektur Anfänger, Fortgeschrittene oder Experten.

Das Weiterbildungsmodul qualifiziert Mitarbeitende in den folgenden Rollen:

N Nutzer: Alle Mitarbeitenden im Unternehmen, die mit einer komplexen Software arbeiten.

P Planer: Erfahrene Mitarbeitende und Führungskräfte, die Softwareprojekte für das Unternehmen steuern oder die eine Optimierung der bestehenden Softwarearchitektur im Unternehmen planen.

Übersicht und Struktur

Der Lernpfad auf der nächsten Seite gibt einen Überblick über die Kernaufgaben im Bereich Softwarearchitektur für produzierende KMU. Anhand der thematisch zugehörigen Lernbausteine können sie abgleichen, wie das eigene Unternehmen im Bereich Softwarearchitektur aufgestellt ist und mögliche Kompetenzlücken im Unternehmen identifizieren.

Weitere Informationen

Weitere Informationen finden Sie auf unserer Homepage: trafonetz.de/weiterbildungsmodule

Gefördert durch:

Bundesministerium für Wirtschaft und Energie

aufgrund eines Beschlusses des Deutschen Bundestages

Stuttgart, den 30.05.2025

Fraunhofer IAO

Disclaimer:
Die in diesem Lernbaustein verwendeten Personenbezeichnungen beziehen sich immer gleichermaßen auf weibliche, männliche und diverse Personen. Auf eine Doppelnennung und gegenderte Bezeichnungen wird zugunsten einer besseren Lesbarkeit verzichtet.

1.1 Architekturstile in Softwarelösungen

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können verschiedene Architekturstile in Softwarelösungen benennen und diese in unterschiedlichen Arbeitsbereichen identifizieren.
Lernschritte / Vorgehensweise / Inhalte	**Bringen Sie eine Definition von Softwarearchitektur in Erfahrung,** z. B. Strukturierung des Zusammenwirkens von Softwarekomponenten, technische als auch konzeptionelle Sicht auf das System, Basis für die Implementierung und Wartung der Software. **Verschaffen Sie sich einen Überblick über die wichtigsten Aspekte von Softwarearchitektur. Zentral sind zum Beispiel:** • **Komponenten:** Die einzelnen Teile des Systems, die spezifische Funktionen oder Dienste bereitstellen. • **Interaktionen:** Die Art und Weise, wie die Komponenten miteinander kommunizieren und Daten austauschen. • **Muster und Stile:** Die Verwendung von bewährten architektonischen Mustern und Stilen. • **Nicht-funktionale Anforderungen:** Überlegungen zu Aspekten wie Leistung, Sicherheit, Skalierbarkeit, Wartbarkeit und Zuverlässigkeit. • **Technologische Entscheidungen:** Die Auswahl der Technologien, Plattformen und Werkzeuge, die zur Realisierung der Architektur verwendet werden. • **Dokumentation:** Die Aufzeichnung der architektonischen Entscheidungen, Entwurfsdokumente und Richtlinien, die für das Verständnis und die Wartung der Architektur notwendig sind. **Machen Sie sich ein Bild über die wichtigsten Architekturstile:** • **Monolithische Architektur:** Z. B. alle Komponenten in einer einzigen, zusammenhängenden Anwendung integriert. • **Serviceorientierte Architektur (SOA):** Z. B. lose gekoppelte Dienste, die über standardisierte Schnittstellen kommunizieren. • **Microservices-Architektur:** Z. B. Weiterentwicklung der serviceorientierten Architektur, bei der jede Funktionalität in einem eigenen, unabhängigen Modul implementiert ist. • **Event-Driven Architecture (EDA):** Z. B. Systeme, die v.a. über asynchrone Kommunikation auf Ereignisse reagieren und diese verarbeiten. **Ermitteln Sie die Bedeutung von Architekturstilen,** z. B. effizientere Entwicklung und Wartung von Softwarelösungen, größere Anpassungsfähigkeit an sich ändernde Anforderungen. **Ermitteln Sie wahlweise für eine Maschine, eine Produktionslinie oder für die gesamte Produktion Ihres Unternehmens Beispiele für verwendete Architekturstile:** • **Monolithische Architektur:** Z. B. Produktionssoftware, die Maschinensteuerung, Datenerfassung und Reporting in einer einzigen Anwendung vereint. • **Serviceorientierte Architektur (SOA):** Z. B. System, in dem Maschinensteuerung und Datenanalyse als separate Dienste fungieren, die über Schnittstellen verbunden sind. • **Microservices-Architektur:** Z. B. Produktionsumgebung, in der jedes Maschinenmodul als eigenständiger Microservice agiert. • **Event-Driven Architecture (EDA):** Z. B. System, das auf Sensorereignisse reagiert und automatisch Anpassungen an den Maschinen vornimmt. **Bewerten Sie – vorzugsweise im Gespräch mit Kollegen mit IT-Wissen – die Vor- und Nachteile der jeweiligen Architekturstile:** • **Monolithische Architektur:** Z. B. Vorteil: Einfachheit in der Implementierung und Bereitstellung, da alle Teile in einem Paket enthalten sind. Z. B. Nachteil: Schwierigkeiten bei der Skalierung und Anpassung, da Änderungen an einer Komponente die gesamte Anwendung betreffen können. • **Serviceorientierte Architektur (SOA):** Z. B. Vorteil: Hohe Flexibilität und Wiederverwendbarkeit der Dienste, einfache Integration neuer Funktionen. Z. B. Nachteil: Komplexität bei der Verwaltung der Dienste und der Kommunikation zwischen ihnen. • **Microservices-Architektur: Z. B.** Vorteil: Hohe Skalierbarkeit, einfachere Wartung und schnellere Entwicklung neuer Funktionen. Z. B. Nachteil: Erhöhter Aufwand für die Verwaltung der Kommunikation zwischen den Microservices. • **Event-Driven Architecture (EDA):** Z. B. Vorteil: Hohe Reaktivität und Flexibilität, besonders geeignet für sich häufig ändernde Produktionsbedingungen. Z. B. Nachteil: Komplexität der Implementierung und mögliche Schwierigkeiten bei der Fehlersuche. **Bewerten Sie – vorzugsweise im Gespräch mit Kollegen mit IT-Wissen – wie die in Ihrer Produktionsumgebung verwendeten Architekturstile die Flexibilität, Wartbarkeit und Effizienz der Softwarelösungen in Ihrer Produktion beeinflussen.**

1.1 Architekturstile in Softwarelösungen

Lernmedien	• Materialien, die verschiedene Architekturstile und deren Anwendungsgebiete in der Produktion erläutern, z. B. Übersicht über Architekturstile wie Microservices oder Monolith • Präsentationen, die die Vor- und Nachteile dieser Architekturstile für KMUs darstellen, z. B. Vergleich von Architekturstilen in der Industrie • Dokumente, die einfache Diagramme zur Veranschaulichung von Architekturstilen enthalten, z. B. Diagramm zu Client-Server-Architekturen • Fachliteratur zu den Auswirkungen von Architekturstilen auf die Produktionssoftware, z. B. Studien über Architekturstile und deren Effizienz
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**8 Stunden in 2 Wochen**

1.4 Vorgehen zur Abstimmung mit IT-Dienstleistern

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können ein Vorgehen zur effektiven Abstimmung mit IT-Dienstleistern planen.
Lernschritte / Vorgehensweise / Inhalte	**Definieren Sie in Abstimmung mit (IT-)Kollegen möglichst genau den Bereich, in dem Sie die Unterstützung eines IT-Dienstleisters benötigen, um die Softwarearchitektur Ihres Unternehmens anzupassen. Hierbei ist zu bedenken:** • Integration neuer Softwaremodule • Anpassung der Softwarearchitektur an Kundenanforderungen hinsichtlich Flexibilität, Performance oder Sicherheit • Migration auf Cloud-Technologien • Beratung zu Vorteilen und Kosten einer neuen ERP-Lösung Dazu und zur späteren Feinplanung mit den Beteiligten Ihres Unternehmens einen gemeinsamen Nenner entwickeln, um mit einer Stimme gegenüber dem IT-Dienstleister zu sprechen und Zeit für die (bezahlte) Entscheidungsfindung einzusparen. **Eignen Sie sich mit Hilfe des vorliegenden Weiterbildungsmoduls ein Überblickswissen über die betreffenden Bereiche der Softwarearchitektur an, zu denen Sie einen IT-Dienstleister hinzuziehen möchten, um möglichst auf Augenhöhe mit dem IT-Dienstleister reden zu können.** **Legen Sie möglichst detailliert die Ziele fest, die Sie mit Hilfe der neuen bzw. optimierten Softwarearchitektur für Ihr Unternehmen erreichen wollen, wie:** • **Reduzierung der Bearbeitungszeiten** für relevante Softwareprozesse um 30% innerhalb von 6 Monaten. • **Erhöhung der Systemverfügbarkeit** auf 99,9% innerhalb eines Jahres. • **Reduzierung der Sicherheitsvorfälle** um 50% innerhalb eines Jahres. • **erfolgreiche Integration von neuen Technologien** (z. B. Cloud-Dienste, IoT-Anwendungen) in die bestehende Softwarearchitektur innerhalb von 12 Monaten. • **Verbesserung der Datenverfügbarkeit und -integrität** um 30% durch Implementierung von Backup- und Recovery-Lösungen. Erarbeiten Sie zusammen mit dem IT-Dienstleister eine Strategie, um diese Ziele durch Optimierung der Softwarearchitektur und/oder durch andere Maßnahmen zu erreichen. **Definieren Sie klare Leistungsvereinbarungen (Service Level Agreements, SLAs), die die Erwartungen Ihres Unternehmens an den Leistungsumfang des Dienstleisters festlegen:** • **Bereitstellung von Support** während der Betriebszeiten Ihres Unternehmens. • **Reaktionszeiten auf Support-Anfragen** für kritische und für nicht kritische Probleme. • **Durchführung von Wartungsarbeiten** außerhalb der Betriebszeiten. • **Angebot eines Ticket-Systems** zur Nachverfolgung von Fehlermeldungen und Wartungsanfragen. • **Durchführung von Sicherheitsüberprüfungen** alle 6 Monate. • Bereitstellung und regelmäßige Aktualisierung von **Dokumentationen**. • Durchführung von mindestens 2 **Schulungen** pro Jahr für die Mitarbeiter. **Wenden Sie für die Optimierung der Softwarearchitektur aktives Projekt-management gegenüber dem IT-Dienstleister an**, z. B. Arbeitspakete, Verantwortlichkeiten, Zeitplan, Meilensteine. Die vereinbarten Ziele, Termine und Leistungsvereinbarungen nachverfolgen und ggf. einfordern.
Lernmedien	• Materialien, die Strategien für die effektive Kommunikation mit IT-Dienstleistern erläutern, z. B. Kommunikationsleitfaden • Vorlagen für Vereinbarungen und Verträge mit IT-Dienstleistern, z. B. Mustervertrag für IT-Dienstleistungen • Dokumente, die die wichtigsten Punkte für die Abstimmung und Zusammenarbeit mit externen Dienstleistern darlegen, z. B. interne oder externe Abstimmungsprotokolle • Präsentationen, die erfolgreiche Kooperationen mit IT-Dienstleistern darstellen, z. B. Best Practices für die Zusammenarbeit in der Softwarearchitektur
Empfohlene Vorkenntnisse	**1.3 Konzeption der Anforderungsanalyse für die Softwarearchitektur**
Dauer	**8 Stunden in 2 Wochen**

1.2 Planung der IST-Analyse der Softwarearchitektur in der Produktion

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden sind in der Lage, eine IST-Analyse der Softwarearchitektur in der Produktion zu konzipieren.
Lernschritte / Vorgehensweise / Inhalte	**Formulieren Sie, welche Ziele Sie für Ihr Unternehmen mit der IST-Analyse der Softwarearchitektur in Ihrer Produktion verfolgen**, z. B. Klärung von wiederkehrenden Fehlermeldungen oder Funktionsstörungen in der Produktion, Bewertung der Systemleistung der Softwarearchitektur. **Planen Sie die Auswahl der Beteiligten, die in den Analyseprozess einbezogen werden sollen**, z. B. Produktionsmitarbeiter, IT-Abteilung, Produktionsleiter. **Konzipieren Sie die Methoden, die in der IST-Analyse der Softwarearchitektur verwendet werden sollen. Dies können z. B. sein:** • **Dokumentationsanalyse:** Z. B. technische Spezifikationen, Benutzerhandbücher • **Interviews und Workshops:** Z. B. mit Produktionsmitarbeitern, um deren Erfahrungen und Herausforderungen zu erfassen • **Systemanalyse:** Z. B. Performance-Messungen und Überprüfung der Systeminteraktionen in der bestehenden Softwarearchitektur **Planen Sie die Durchführung der IST-Analyse der Softwarearchitektur in Ihrer Produktion.** Dazu gehört: • **Zieldefinition:** Z. B. Einfluss der aktuellen Softwarearchitektur auf die Systemleistung. • **Datensammlung:** Z. B. relevante Daten über die Nutzung der Software, Systemleistung und Benutzerfeedback erfassen. • **Datenanalyse:** Z. B. Muster und Probleme in den Daten identifizieren. • **IST-Report:** Z. B. Dokumentation der Ergebnisse der IST-Analyse, einschließlich identifizierter Schwächen und Verbesserungspotenziale. **Erstellen Sie vorbereitend eine Liste mit Tools zur Unterstützung der IST-Analyse:** • **Datenanalyse-Tools:** Z. B. Excel oder spezialisierte Software • **Diagrammtools zur Visualisierung der bestehenden Softwarearchitektur:** Z. B. Lucidchart oder Draw.io • **Tools zur Dokumentation der IST-Analyse und der Ergebnisse:** Z. B. Confluence oder Microsoft OneNote **Priorisieren Sie die Ergebnisse der Ist-Analyse zur Softwarearchitektur,** z. B. nach Auswirkungen auf Leistung, Kosten, Machbarkeit und Benutzer. **Planen Sie die Präsentation der Empfehlungen aus der IST-Analyse der Softwarearchitektur,** z. B. vor Produktionsleitung, Geschäftsführung, IT-Abteilung. Darin Ziel, Methodik, Stärken und Schwächen des Status Quos der Softwarearchitektur sowie konkrete Maßnahmen darstellen.
Lernmedien	• Vorlagen zur Durchführung einer IST-Analyse, die spezifische Fragen zur Software-architektur beinhalten • Materialien zur Dokumentation von Ist-Zuständen, die einfach zu handhaben sind, z. B. Dokumentationsleitfaden • Präsentationen, die Methoden zur Datenerhebung für die IST-Analyse vorstellen, z. B. durch Befragung von Nutzern • Fachliteratur Analysen der Softwarearchitektur, z. B. Artikel in IT-Zeitschrift
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**20 Stunden in 4 Wochen**

2.1 Technologien in der Softwarearchitektur

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können ein strukturiertes Vorgehen zur Auswahl geeigneter Technologien planen.
Lernschritte / Vorgehensweise / Inhalte	**Erarbeiten Sie sich einen Überblick über wichtige Technologien der Softwarearchitektur:** • **Programmiersprachen** wie Python, Java und C# • **Datenbanken** wie MySQL und PostgreSQL • **Cloud-Technologien** wie AWS und Microsoft Azure • **API-Technologien** (Application Programming Interfaces) wie Postman oder Swagger • **Middleware**-Lösungen • **IoT-Plattformen** wie Azure IoT Hub und AWS IoT Core • **Datenbankmanagementsysteme** wie MongoDB und Firebase **Bringen Sie im Dialog mit Kollegen und Vorgesetzten in Erfahrung, welche der Technologien mit Bezug zur Softwarearchitektur in der Produktion Ihres eigenen Unternehmens genutzt werden.** Notieren Sie sich, welche der bisher nicht genutzten Technologien für die Softwarearchitektur Ihrer Produktion interessant sein könnten. **Erarbeiten Sie sich ein Grundverständnis davon, wofür Sie die Technologien für die Softwarearchitektur in Ihrer Produktion einsetzen können:** • **Programmiersprachen:** Flexibilität (z. B. Entwicklung von maßgeschneiderten Softwarelösungen für spezifische Produktionsanforderungen), einfache Integration (z. B. viele Bibliotheken und Frameworks unterstützen die Integration von Produktionsmaschinen und Software), Community und Support (z. B. große Entwicklergemeinschaften bieten Unterstützung und Ressourcen). • **Datenbanken:** Datenverwaltung (z. B. effiziente Speicherung, Abfrage und Verwaltung von Produktionsdaten), Transaktionen (z. B. Gewährleistung der Konsistenz von Daten während der Produktion), Skalierbarkeit (z. B. unterstützen das Wachstum des Unternehmens durch die Aufnahme neuer Daten). • **Cloud-Technologien:** Ressourcenzugang (z. B. skalierbare IT-Ressourcen, die leicht an die Bedürfnisse der Produktion angepasst werden können), Kostenersparnis (z. B. reduzieren die Notwendigkeit für Investitionen in physische Hardware), Datenanalyse (z. B. ermöglichen die Analyse großer Datenmengen zur Optimierung der Produktionsprozesse). • **API-Technologien:** Interoperabilität (z. B. erleichtern die Kommunikation zwischen verschiedenen Softwaremodulen und Maschinen), Testen und Dokumentation (z. B. zur Unterstützung der Integration), Flexibilität (z. B. APIs erlauben die Anpassung und Erweiterung der Softwarearchitektur). • **Middleware-Lösungen:** Integration (z. B. erleichtern die Verbindung zwischen verschiedenen Software- und Hardwarekomponenten für eine reibungslose Kommunikation und Datenübertragung), Standardisierung (z. B. sorgen für einheitliche Schnittstellen und Protokolle, die die Integration vereinfachen). • **IoT-Plattformen:** Datenverfügbarkeit (z. B. ermöglichen die Echtzeitüberwachung von Maschinen und Prozessen mittels Daten von IoT-Geräten), Echtzeit-Analysen (z. B. unterstützen die Analyse von Produktionsdaten in Echtzeit für sofortige Anpassungen), Skalierbarkeit (z. B. erleichtern die Integration neuer IoT-Geräte). • **Datenbankmanagementsysteme:** Flexibilität (z. B. NoSQL-Datenbanken mit flexiblen Datenmodellen, die sich leicht an die Änderungen der Produktions-umgebung anpassen lassen), Echtzeit-Daten (z. B. ermöglichen die Echtzeit-Synchronisation von Daten für die Überwachung und Steuerung der Produktion). • **Dokumentieren Sie Ihre Erkenntnisse, um diese für weitere Aktivitäten Ihres Unternehmens im Bereich Softwarearchitektur zur Verfügung zu stellen.**
Lernmedien	• Materialien, die gängige Technologien und deren Anwendungsgebiete in der Softwarearchitektur vorstellen, z. B. Übersicht über Tools wie Datenbanken und Middleware. • Präsentationen, die die Vorzüge verschiedener Technologien für KMUs beleuchten, • z. B. Präsentation zu Technologien der Softwarearchitektur in der Produktion. • Dokumente, die verständliche Erklärungen zu Technologien bieten, z. B. Glossar gängiger Technologien in der Softwarearchitektur. • Fachliteratur über die Implementierung neuer Technologien der Softwarearchitektur in Produktionsumgebungen, z. B. Artikel über Einführung entsprechender Technologien in KMUs.
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**8 Stunden in 2 Wochen**

2.2 Vorgehen zur Auswahl von Technologien für die Softwarearchitektur

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können ein strukturiertes Vorgehen zur Auswahl geeigneter Technologien planen.
Lernschritte / Vorgehensweise / Inhalte	**Erstellen Sie eine Liste mit den Rahmenbedingungen Ihres Unternehmens für die Auswahl von Technologien, die für die Softwarearchitektur in der Produktion benötigt werden, wie:** • **Verfügbare finanzielle und personelle Ressourcen des Unternehmens:** Z. B. Auswahl kosteneffizienter und einfach zu implementierender Technologien erforderlich. • **Produktionsprozesse:** Z. B. Just-in-Time-Produktion erfordert Technologien, die eine ständige Kommunikation zwischen Produktionsstätten, Lieferanten und Logistikpartnern sicherstellen. • **Marktanpassungsfähigkeit:** Z. B. flexible Technologien, die leicht angepasst oder aktualisiert werden können. • **Qualitätsanforderungen:** Z. B. hohe Qualitätsstandards insbesondere in der Automobilzulieferindustrie erfordern geeignete Technologien zur Qualitätssicherung. • **Integration bestehender Systeme:** Z. B. Kompatibilität vorhandener IT-Infrastruktur und Softwarelösungen mit neuen Technologien. • **Datensicherheit und Compliance:** Z. B. Technologien müssen den gesetzlichen Anforderungen und Branchenstandards entsprechen. • **Mitarbeiterqualifikation:** Z. B. der Wissensstand der Mitarbeitenden beeinflusst die Auswahl von Technologien, die mehr oder weniger intuitiv zu bedienen sind bzw. Schulungen erfordern. **Planen Sie die wichtigsten Elemente des Vorgehens zur Auswahl von Technologien für die Softwarearchitektur, wie:** • **Zielsetzung festlegen**: Z. B. Anpassung der Softwarearchitektur für die Just-In-Time-Produktion • **Beteiligte definieren:** Z. B. Produktionsleitung, IT-Abteilung, Geschäftsführung • **benötigte Ressourcen benennen:** Z. B. Personalkapazität • **Vorgehen klären:** Z. B. Ist-Analyse, Anforderungsanalyse, Auswahl von Bewertungskriterien und deren Gewichtung, Technologiebewertung, Entscheidung über Auswahl, Terminplanung, Dokumentation Ggf. die Ergebnisse der Lernbausteine *1.2 Planung der IST-Analyse der Softwarearchitektur in der Produktion* und *1.3 Konzeption der Anforderungsanalyse für die Softwarearchitektur* einbeziehen. **Informieren Sie sich, mit welchen Technologien die Softwarearchitektur flexibler gestaltet werden kann:** • **Für schnelle Änderungskonfiguration bei Produktänderungen/-neuanläufen:** Technologien, die modulare Architekturen unterstützen (z. B. Microservices oder Containerisierung). • **Für einfache Verwaltung von Softwareversionen:** Technologien zur Integration von Versionierungssystemen, sodass verschiedene Produktvarianten parallel betrieben werden können. • **Für Anpassung der Softwarearchitektur bei Produktionslinienerweiterungen:** Schnittstellentechnologien, die eine einfache Integration neuer Komponenten ermöglichen (z. B. APIs, Application Programming Interfaces). • **Für vereinfachte Integration neuer Komponenten:** Middleware-Lösungen, die als Vermittler zwischen verschiedenen Softwarelösungen fungieren. **Konzipieren Sie nach Eingrenzung der Technologieauswahl Kriterien für eine Produkt-Bewertung,** z. B. Kosten, Funktionalität, Benutzerfreundlichkeit, Support/Wartung und wie diese Kriterien gewichtet werden sollen. **Planen Sie ggf. zur Evaluierung der Technologieauswahl für die Software-architektur einen ersten Prototyp.** Mit begrenztem Invest die Funktionalität und Passung der neuen Technologie mit der bestehenden Produktionsumgebung testen. **Planen Sie die Dokumentation der Kriterien, der Entscheidung und der Begründung der Technologieauswahl.** Diese Informationen an die Beteiligten kommunizieren.
Lernmedien	• Materialien, die einen strukturierten Prozess zur Auswahl von Technologien in der Softwarearchitektur erläutern, z. B. Auswahlleitfaden • Vorlagen zur Bewertung und zum Vergleich von Technologien, die auf die Bedürfnisse von KMUs zugeschnitten sind, z. B. Bewertungsmatrix • Präsentationen, die Entscheidungskriterien für die Auswahl von Technologien in der Softwarearchitektur darstellen, z. B. Kriterienkatalog • Dokumente, die Beispiele für erfolgreiche Technologieauswahlprozesse für Softwarearchitektur in der Industrie zeigen, z. B. Fallstudien
Empfohlene Vorkenntnisse	**2.1 Technologien in der Softwarearchitektur**
Dauer	**20 Stunden in 4 Wochen**

3.1 Application Programming Interface (API) – Gestaltung in der Softwarearchitektur

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden sind in der Lage, die API-Gestaltung im Kontext der Softwarearchitektur zu planen.
Lernschritte / Vorgehensweise / Inhalte	**Recherchieren Sie eine Definition von API (Application Programming Interface),** z. B. Schnittstellen, die definieren, wie Maschinen, Steuerungen und Softwarelösungen Daten austauschen und miteinander interagieren. **Bringen Sie in Erfahrung, welche Typen von APIs in Ihrem Unternehmen genutzt werden, dies können z.B. sein:** • **RESTful APIs:** Am häufigsten verwendet, nutzt HTTP-Protokolle und überträgt Daten im JSON- oder XML-Format. • **SOAP APIs:** Älterer Standard, verwendet XML und legt strenge Verträge für die API-Kommunikation fest. • **GraphQL:** Moderne API-Sprache, ermöglicht es Clients genau die Daten abzufragen, die sie benötigen. **Identifizieren Sie in Abstimmung mit (IT-)Kollegen Regeln für Ihr Unternehmen, wie APIs gestaltet werden sollen,** wie klare Dokumentation für die Nutzung durch Entwickler, einheitliche Namenskonventionen für Endpunkte (z. B. /getMachineStatus) und Sicherheitsmechanismen für Authentifizierung und Autorisierung (z. B. durch OAuth 2.0). **Konzipieren Sie die Überwachung und Wartung von APIs in Ihrem Unternehmen,** hinsichtlich Leistungsüberwachung und zur Identifikation von Engpässen und hinsichtlich regelmäßiger Wartung und Updates zur Vermeidung von Sicherheitsrisiken. **Beziehen sie in die Gestaltungskonzepte der APIs ein, inwieweit diese die Änderungskonfiguration bei Produktänderungen/-neuanläufen unterstützen,** in Hinblick auf Modularität (z. B. APIs aktualisieren spezifische Funktionen von Maschinenmodulen unabhängig voneinander, was die Anpassung an neue Produktanforderungen erleichtert) und auf Flexibilität (z. B. durch APIs können neue Funktionen schnell implementiert werden, ohne bestehende Systeme grundlegend zu verändern). **Legen Sie API-Vorgaben für die Anpassung der Softwarearchitektur bei Produktionslinienerweiterungen fest,** wie für die Schnittstellenanpassung (die bestehenden APIs müssen z. B. so gestaltet werden, dass sie die neue Hardware unterstützen und dass neue Endpunkte hinzugefügt werden können, um die Daten von neuen Maschinen zu integrieren) und die Skalierbarkeit (die API sollte so konzipiert sein, dass sie z. B. mit dem Wachstum des Unternehmens und der Erweiterung der Produktionslinie skalierbar bleibt). **Planen Sie die Auswahl und Verwendung von Tools und Technologien für die API-Gestaltung:** • **API-Management-Tools:** Z. B. Postman, Swagger zur Dokumentation und zum Testen von APIs • **Entwicklungsumgebungen:** Z. B. Node.js, Python Flask, Java Spring Boot für die Implementierung von APIs • **Monitoring-Lösungen:** Z. B. Grafana, Prometheus zur Überwachung der API-Performance und zur Identifikation von Engpässen
Lernmedien	• Materialien, die Grundlagen der API-Gestaltung und deren Bedeutung erläutern, z. B. Einführung in RESTful APIs • Präsentationen, die die besten Praktiken für die Gestaltung von APIs im Kontext der Softwarearchitektur erläutern, z. B. Best Practices für API-Design • Vorlagen für API-Dokumentationen, die einfach genutzt werden können • Dokumente, die häufige Fehler bei der API-Gestaltung auflisten
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**10 Stunden in 4 Wochen**

3.2 Planung der Integration von Softwaremodulen

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können die Integration von Modulen einer Softwarelösung konzipieren.
Lernschritte / Vorgehensweise / Inhalte	**Ermitteln Sie eine Definition von Softwareintegration,** z. B. Prozess, in dem verschiedene Softwaremodule miteinander verbunden werden, um einen reibungslosen Austausch von Informationen und Funktionen zu ermöglichen. **Identifizieren Sie in Ihrer Produktion Softwaremodule, deren Elemente bisher oder zukünftig integriert wurden/werden,** z. B. Maschinensteuerungssoftware, ERP/ Enterprise Resource Planning-Systeme, Qualitätssicherungssysteme, Bestellmanage-mentsysteme, Produktionsplanungssysteme. **Verschaffen Sie sich einen Überblick über Anlässe, die zur Integration von Softwaremodulen führen:** • **Anlässe bei Maschinensteuerungssystemen:** Z. B. regelmäßige Wartungen, Produktionswechsel, Notfälle • **Anlässe bei ERP-Systemen:** Z. B. Produktionswechsel, Lieferantenwechsel, technologische Veränderungen • **Anlässe bei Qualitätssicherungssystemen:** Z. B. Qualitätsprüfungen, Notfälle, gesetzliche Änderungen • **Anlässe bei Bestellmanagementsystemen:** Z. B. Lieferantenwechsel, unregelmäßige Änderungen in der Beschaffung • **Anlässe bei Produktionsplanungssystemen:** Z. B. technologische Veränderungen, saisonale Produktwechsel **Verschaffen Sie sich einen Überblick über die Herausforderungen, die bei der Integration von Modulen einer Softwarelösung auftreten können:** • **Technische Herausforderungen:** Kompatibilität (z. B. verschiedene Softwaremodule können unterschiedliche Technologien, Programmiersprachen oder Datenformate verwenden, was zu Kompatibilitätsproblemen führt), Schnittstellenentwicklung (z. B. das Erstellen geeigneter Schnittstellen (APIs) zwischen den Softwaremodulen erfordert spezifische technische Kenntnisse und kann zeitaufwendig sein, Fehler in der Schnittstellendefinition können zu Datenverlust oder -inkonsistenzen führen), Datenmigration (z. B. bei der Integration müssen oft bestehende Daten in unterschiedlichen Formaten oder in unvollständiger Form in das neue System übertragen werden), Leistung und Ressourcen (z. B. die Integration neuer Softwaremodule kann die Systemleistung beeinträchtigen). • **Sicherheits- und Compliance-Herausforderungen:** Datensicherheit (z. B. bei der Integration neuer Softwaremodule müssen Sicherheitsaspekte berücksichtigt werden, insbesondere bei der Übertragung sensibler Daten, Compliance-Anforderungen (z. B. Unternehmen müssen sicherstellen, dass die neuen Softwarelösungen den gesetzlichen Anforderungen und internen Richtlinien entsprechen). • **Organisatorische Herausforderungen:** Prozessanpassungen (z. B. die bestehenden Arbeitsprozesse müssen häufig angepasst werden, um die neuen Softwarelösungen zu integrieren. Dies kann zu Widerstand bei den Mitarbeitenden führen, die an den alten Prozessen festhalten wollen), Schulungen (z. B. Mitarbeitende müssen möglicherweise geschult werden, um die neuen Systeme effektiv nutzen zu können, unzureichende Schulung kann zu Fehlern und Ineffizienzen im Betrieb führen), Kommunikation (z. B. eine unzureichende Kommunikation zwischen den verschiedenen Abteilungen kann dazu führen, dass wichtige Informationen verloren gehen oder Missverständnisse entstehen. Alle relevanten Akteure müssen über den Integrationsprozess informiert werden). **Planen Sie die Schritte der Integration von Softwaremodulen:** • **Bedarfsanalyse durchführen:** Z. B. bestimmen, welche Softwaremodule integriert werden müssen und warum. • **Schnittstellen definieren:** Z. B. Schnittstellen festlegen, über die die Softwaremodule miteinander kommunizieren. • **Technologie auswählen:** Z. B. passende Technologien und Tools wie APIs und Middleware auswählen, die die Integration unterstützen. • **Testplanung erstellen:** Z. B. Testplan erstellen, um die Funktionsfähigkeit der Integration zu prüfen. • **Implementierung durchführen:** Z. B. Integration gemäß den vorher festgelegten Schritten durchführen. • **Schulung der Mitarbeiter durchführen:** Z. B. sicherstellen, dass alle betroffenen Mitarbeiter die neuen Systeme verstehen und nutzen können. **Dokumentieren Sie Ihre Planung der Integration von Softwaremodulen in geeigneter Weise,** so dass Integrationsziele, betroffene Softwaremodule, Beteiligte und Integrationsschritte schnell festgelegt und ausgeführt werden können.

3.2 Planung der Integration von Softwaremodulen

Lernmedien	• Materialien, die Schritte zur erfolgreichen Integration von Softwaremodulen darstellen, z. B. Integrationsleitfaden • Vorlagen zur Planung und Dokumentation der Integrationsschritte • Präsentationen, die verschiedene Integrationsstrategien vorstellen • Dokumente, die häufige Herausforderungen bei der Softwareintegration aufzeigen, z. B. Checkliste zu Integrationsproblemen
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**20 Stunden in 2 Monaten**

4.1 Performance-Optimierung von Softwarearchitektur

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können die Performance-Optimierung in der Softwarearchitektur planen.
Lernschritte / Vorgehensweise / Inhalte	**Ermitteln Sie eine Definition der Performance-Optimierung,** z. B. die Fähigkeit eines Systems, Aufgaben schnell und effizient hinsichtlich Reaktionszeiten, Verarbeitungszeiten und Ressourcennutzung zu erledigen. **Bringen Sie die Vorteile der Performance-Optimierung von Softwarearchitektur in Erfahrung,** z. B. effizientere Zusammenarbeit der Produktionsmaschinen und schnellere Erreichung von Produktionszielen. **Sammeln Sie unterschiedliche Parameter zur Performance-Messung und gleichen Sie ab, welche dieser Parameter in Ihrer Produktion verwendet werden:** • **Latenz:** Die Zeit, die benötigt wird, um eine Anfrage zu bearbeiten. • **Durchsatz:** Die Anzahl der erfolgreich verarbeiteten Anfragen innerhalb eines bestimmten Zeitrahmens. • **Fehlerrate:** Prozentsatz der fehlgeschlagenen Transaktionen oder Anfragen im Vergleich zur Gesamtzahl der Anfragen. • **Skalierbarkeit:** Fähigkeit eines Systems, seine Leistung proportional zur Erhöhung der Ressourcen wie zusätzlicher Server zu verbessern. • **Ressourcenauslastung:** Engpässe von Hardware-Ressourcen wie CPU, RAM oder Netzwerk, die die Performance der Softwarearchitektur beeinträchtigen können. **Erarbeiten Sie sich einen Überblick über Techniken zur Performance-Optimierung von Softwarearchitektur:** • **Caching:** Häufig abgerufene Daten in einem schnellen Zugriffsspeicher (Cache) speichern, um die Anzahl der Berechnungen oder Datenbankabfragen zu reduzieren wie In-Memory Caching (Verwendung von Speicher wie Redis oder Memcached, um Daten temporär zu speichern) oder HTTP Caching (Browser- oder Proxy-Caching zur Reduzierung von Serveranfragen). • **Lastverteilung (Load Balancing):** Den eingehenden Datenverkehr auf mehrere Server oder Dienste verteilen, um Überlastung zu vermeiden und die Verfügbarkeit zu erhöhen, wie Round Robin (Anfragen werden gleichmäßig auf alle Server verteilt) oder Least Connections (Anfragen werden an den Server mit den wenigsten aktiven Verbindungen gesendet). • **Asynchrone Verarbeitung:** Aufgaben, die nicht sofortige Ergebnisse erfordern, von der Hauptanwendung trennen, um die Benutzeroberfläche reaktionsfähig zu halten, wie Message Queues (Verwendung von Warteschlangen wie RabbitMQ oder Kafka für die asynchrone Verarbeitung von Aufgaben) und Event-Driven Architecture (Reagieren auf Ereignisse, um Prozesse zu steuern und zu optimieren). • **Lasttests und Monitoring:** Performance-Probleme identifizieren und beheben, bevor sie den Benutzern auffallen, wie Lasttests (Simulation von Benutzerlasten, um die Systemleistung zu testen) und Echtzeit-Überwachung (Verwendung von Monitoring-Tools wie Prometheus oder Grafana, um die Systemleistung kontinuierlich zu überwachen). • **Verwendung von Content Delivery Networks (CDNs):** Bandbreitennutzung reduzieren und Ladegeschwindigkeit verbessern, wie durch Datenkompression (Reduzierung der Größe von Daten, die über das Netzwerk gesendet werden) oder Lazy Loading (Laden von Inhalten nur, wenn sie benötigt werden, anstatt alles auf einmal). **Planen Sie das Konzept einer Fallstudie, in der Strategien zur Performance-Optimierung von Softwarearchitektur in Ihrer Produktion entwickelt werden,** z. B. mit der Beschreibung von Problem, Zielsetzung, Methodik, Datenanalysen, Ergebnissen und Empfehlungen. Dafür die in diesem Lernbaustein aufgeführten Informationen nutzen.
Lernmedien	• Materialien, die Techniken zur Performance-Optimierung erläutern • Präsentationen, die Methoden zur Messung und Verbesserung der Softwareperformance vorstellen, z. B. Anwendung von Parametern für die Optimierung von Softwarearchitektur in der Produktion • Fallstudien, die konkrete Beispiele für Performance-Optimierungen bieten • Vorlagen zur Durchführung von Performance-Tests, z. B. Testplan-Vorlage
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**10 Stunden in 4 Wochen**

4.2 Sicherheitsanforderungen für die Softwarearchitektur

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können die Sicherheitsanforderungen für die Softwarearchitektur konzipieren.
Lernschritte / Vorgehensweise / Inhalte	**Erarbeiten Sie sich eine Beschreibung, was Sicherheit in der Softwarearchitektur bedeutet,** z. B. die Fähigkeit einer Software, vor unbefugtem Zugriff, Datenverlust und anderen Bedrohungen geschützt zu sein. **Fassen Sie zusammen, worin für Ihr Unternehmen die Bedeutung von Sicherheit in der Softwarearchitektur liegt,** z. B. eine sichere Softwarearchitektur schützt nicht nur die Daten, sondern trägt auch zur Zuverlässigkeit der Produktionsprozesse bei. **Stimmen Sie – bevorzugt mit IT-Kollegen – ab, was die wichtigsten Sicherheits-anforderungen für die Softwarearchitektur in Ihrer Produktion sind:** • **Zugriffskontrolle:** Sicherstellen, dass nur autorisierte Benutzer Zugriff auf bestimmte Funktionen und Daten haben. • **Datenverschlüsselung:** Schutz sensibler Informationen durch Verschlüsselung, sowohl bei der Speicherung als auch während der Übertragung. • **Authentifizierung und Autorisierung:** Überprüfen der Identität von Benutzern und Festlegen von Berechtigungen für den Zugriff auf Ressourcen. • **Sicherheitsprotokolle:** Implementierung von Protokollen zur Erkennung und Reaktion auf Sicherheitsvorfälle. **Ermitteln Sie die wichtigsten Risiken unsicherer Softwarearchitektur für Ihr Unternehmen:** • **Datenverlust:** Unzureichende Sicherheitsmaßnahmen können zu Datenverlust führen, was erhebliche Auswirkungen auf die Produktionsabläufe haben kann. • **Unbefugter Zugriff:** Schwächen in der Architektur können dazu führen, dass unbefugte Dritte auf kritische Informationen zugreifen. • **Betriebsunterbrechungen:** Sicherheitsvorfälle können zu Ausfallzeiten führen, die die Produktion beeinträchtigen. **Identifizieren Sie notwendige Maßnahmen zur Risikominderung für die Softwarearchitektur Ihrer Produktion:** • **Regelmäßige Sicherheitsüberprüfungen:** Durchführung von Audits und Tests, um Sicherheitslücken zu identifizieren und zu schließen. • **Schulung der Mitarbeiter:** Sensibilisierung der Mitarbeiter für Sicherheitsrichtlinien und Best Practices. • **Notfallpläne:** Entwicklung von Plänen zur Reaktion auf Sicherheitsvorfälle, um den Betrieb schnellstmöglich wiederherzustellen.
Lernmedien	• Materialien, die Sicherheitsanforderungen und Standards für Softwarearchitekturen erläutern, z. B. Sicherheitsleitfaden für Softwarearchitektur in KMUs • Präsentationen, die häufige Sicherheitsrisiken in der Softwarearchitektur aufzeigen • Dokumente, die einfache Checklisten zu Sicherheitsanforderungen bereitstellen • Fachliteratur über IT-Sicherheit in der Softwarearchitektur, z. B. Artikel über Strategien für Datensicherheit
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**10 Stunden in 4 Wochen**

1.3 Konzeption der Anforderungsanalyse für die Softwarearchitektur

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können eine fundierte Anforderungsanalyse für die Softwarearchitektur planen.
Lernschritte / Vorgehensweise / Inhalte	**Benennen Sie die Ziele, die Ihr Unternehmen mit der Anforderungsanalyse für die Softwarearchitektur verfolgt.** Diese Ziele (z. B. Effizienz und Leistung der Produktionsmaschinen verbessern, Kosten senken, Missverständnisse im Rahmen der gemeinsamen Durchführung der Anforderungsanalyse vermeiden) aktiv an die Beteiligten der Anforderungsanalyse kommunizieren. **Stellen Sie sicher, dass die erforderlichen Ressourcen für die Anforderungsanalyse zur Verfügung stehen,** z. B. eigene Kapazität und die der Interviewpartner, technische Mittel, Zugang zu Software und Datenauswertungen. **Planen Sie die Methoden für die Durchführung der Anforderungsanalyse,** z. B. Interviews mit Beteiligten, Workshops zur Erfassung der Anforderungen, strukturierte Fragebögen zur Sammlung quantitativer Daten. **Prüfen Sie die Ergebnisse der IST-Analyse der Softwarearchitektur (*Lernbaustein 1.2 Planung der IST-Analyse der Softwarearchitektur in der Produktion*) auf Hinweise zu Anforderungen an die Softwarearchitektur.** Ggf. enthaltene Probleme und Begrenzungen der bestehenden Softwarearchitektur in Anforderungen für die Optimierung oder Erneuerung der Softwarearchitektur überführen. **Informieren Sie sich zum Vergleich auch über gelungene/ weniger gelungene Lösungen der Softwarearchitektur in der Produktion von anderen Unternehmen, die mit Ihrem Unternehmen vergleichbar sind.** Diese Lösungen beispielsweise aus Fachzeitschriften, dem Internet oder von bekannten Unternehmen beziehen. **Erstellen Sie eine Vorlage für die präzise Formulierung von Anforderungen,** z. B. eindeutige Beschreibung, detaillierte Beschreibung, betroffene Nutzer und Entscheidungsträger, Kriterien für die Erfüllung, Priorität der Anforderung, Abhängigkeiten zu anderen Softwarelösungen. **Planen Sie die Priorisierung der Anforderungen, die sich aus der Anforderungsanalyse ergeben.** Dazu passende Methode auswählen wie MoSCoW-Methode, Nutzwertanalyse, Einfaches Ranking, 100-Punkte-Methode. Unternehmensziele und Ressourcen bei der Priorisierung berücksichtigen. **Konzipieren Sie die Ergebnispräsentation der Anforderungsanalyse,** z. B. größte Optimierungspotenziale, wichtige Kennzahlen und wie diese verbessert werden können, Diagramme und Tabellen, Zitate aus den Interviews, Zusammenfassung und Gesamtbotschaft der Präsentation. Ggf. Anforderungen in Form von User Stories abbilden, wenn z. B. Nutzereingaben an Bedienoberflächen Teil der Anforderungsanalyse sind. **Definieren Sie KPIs (Key Performance Indicators), um später den Fortschritt bei der Optimierung der Softwarearchitektur zu messen:** • Systemverfügbarkeit/ Ausfallzeiten von Maschinen • Fehlerraten • Reaktionszeit der Software auf Benutzeranfragen oder Systemereignisse • Wartungsaufwand der Softwarearchitektur • Entwicklungszeit für neue Maschinenfunktionen • Benutzerzufriedenheit • Auslastung von Servern, Datenbanken und anderen Ressourcen • höhere Testabdeckung zur Sicherstellung der Softwarequalität • Effizienzgewinn durch Senkung der Betriebskosten pro Transaktion • Integrationserfolg von neuen Software- und Maschinenkomponenten **Planen Sie den regelmäßigen Abgleich der Ergebnisse der Anforderungsanalyse mit den später erreichten Verbesserungen der Softwarearchitektur.** Dazu Verbleibe über die Optimierungsziele der Softwarearchitektur im Verlauf der Maßnahmen regelmäßig mit dem Status Quo spiegeln und verbleibende Nachsteuerungsbedarfe kommunizieren.
Lernmedien	• Materialien, die den Prozess der Anforderungsanalyse einfach darstellen, z. B. Leitfaden zur Anforderungsaufnahme • Vorlagen für Anforderungsdokumente, die in der Praxis verwendet werden können • Präsentationen, die Best Practices zur Erhebung von Anforderungen zeigen • Dokumente, die häufige Herausforderungen bei der Anforderungsanalyse aufzeigen, z. B. Checkliste typischer Anforderungen
Empfohlene Vorkenntnisse	**1.2 Planung der IST-Analyse der Softwarearchitektur in der Produktion**
Dauer	**20 Stunden in 4 Wochen**

3.3 Vorgehen zum Einsatz von Middleware in der Softwareintegration

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können ein Vorgehen zur Nutzung von Middleware für die Integration von Softwaremodulen planen.
Lernschritte / Vorgehensweise / Inhalte	**Ermitteln Sie eine Definition von Middleware und welche Funktion sie in der Softwarearchitektur besitzt,** z. B. Software, die als Brücke zwischen Betriebssystemen und Anwendungen, die Kommunikation und den Datenaustausch zwischen verschiedenen Softwaremodulen ermöglicht und unterschiedliche Systeme und Anwendungen miteinander verbindet. **Recherchieren Sie verschiedene Arten von Middleware:** • **Datenbank-Middleware:** Ermöglicht den Zugriff auf Datenbanken und deren Verwaltung. • **Message-Oriented Middleware/MOM:** Kümmert sich um die Übertragung von Nachrichten zwischen Anwendungen, z. B. RabbitMQ oder Apache Kafka. • **Remote Procedure Call/RPC:** Erlaubt es, Funktionen auf einem entfernten Server auszuführen, als wären sie lokal. **Ermitteln Sie – vorzugsweise mit IT-Kollegen – welche Arten von Middleware in Ihrer Produktion eingesetzt werden und welche verbindende Funktion sie für die jeweilige Softwarearchitektur besitzen.** **Verschaffen Sie sich einen Überblick über die Vorteile von Middleware in der Softwarearchitektur hinsichtlich:** • **Flexibilität:** Z. B. Middleware ermöglicht die Integration unterschiedlicher Systeme, was die Flexibilität erhöht. • **Skalierbarkeit:** Z. B. durch den Einsatz von Middleware können Systeme einfacher erweitert oder angepasst werden. • **Echtzeitkommunikation:** Z. B. Middleware kann Echtzeitkommunikation zwischen Softwaremodulen ermöglichen, was für Produktionsprozesse entscheidend ist. • **Wiederverwendbarkeit:** Z. B. Middleware fördert die Wiederverwendbarkeit von Softwaremodulen, da sie nicht für jede Integration neu entwickelt werden müssen. **Entwickeln Sie Szenarien für den Einsatz von Middleware in Ihrer Produktion:** • **Unterstützte Änderungskonfiguration von Maschinenmodulen bei Produktänderungen/-neuanläufen:** Z. B. über Middleware die Datenflüsse zwischen den Maschinensteuerungen und den Produktionsplanungssystemen anpassen, ohne dass eine vollständige Neuprogrammierung erforderlich ist. • **Unterstützte Erweiterung der Produktionslinie und Integration neuer Maschinenkomponenten:** Z. B. Middleware kann sicherstellen, dass neue Maschinen nahtlos in die bestehende Produktionsumgebung integriert werden. **Ermitteln Sie eine/mehrere Optionen für den Einsatz von Middleware-Anwendungen in Ihrer Produktion die Herausforderungen, die sich bei der Implementierung von Middleware für Ihr Unternehmen stellen** wie Komplexität, benötigtes Fachwissen, Kosten für Implementierung und Betrieb von Soft- und Hardware, regelmäßige Wartung und Aktualisierung. **Konzipieren Sie das Vorgehen zum Einsatz von Middleware in Ihrer Produktion,** z. B. in Form eines Leitfadens: • **Bedarfsanalyse:** Systeme identifizieren (z. B. bestimmen, welche Softwareanwendungen wie Maschinensteuerungen, ERP-Systeme, Qualitätsmanagementsysteme) integriert werden sollen), Anforderungen erfassen (z. B. welche Kommunikations- oder Datenübertragungsbedarfe bestehen, müssen Echtzeitdaten übertragen werden, gibt es spezielle Datensicherheits-anforderungen?). • **Auswahl der Middleware:** Verschiedene Middleware-Lösungen und deren Funktionalitäten recherchieren (z. B. RabbitMQ, Apache Kafka, Microsoft Azure Service Bus), Kriterien festlegen (z. B. Flexibilität, Skalierbarkeit, Kosten, Benutzerfreundlichkeit, Integrationsfähigkeit), Varianten vergleichen (z. B. verschiedene Middleware-Lösungen in einer Testumgebung testen, um deren Eignung zu überprüfen). • **Architekturdesign:** Schnittstellen festlegen (z. B. definieren, wie die Middleware mit den verschiedenen Softwareanwendungen kommunizieren soll und welche APIs oder Protokolle verwendet werden), Datenflussdiagramme erstellen (z. B. den Datenfluss zwischen den verschiedenen Systemen und der Middleware darstellen). • **Implementierung:** Middleware gemäß den definierten Anforderungen installieren und konfigurieren, Anwendungen integrieren (z. B. die notwendigen Schnittstellen entwickeln, um die Softwareanwendungen mit der Middleware zu verbinden), Integration testen (z. B. Tests durchführen, um sicherzustellen, dass die Middleware korrekt funktioniert und die Daten wie gewünscht überträgt). • **Monitoring und Wartung:** Monitoring-Tools verwenden, um die Leistung der Middleware und die Integrität der Datenübertragungen zu überwachen. Regelmäßige Wartungsarbeiten einplanen, um sicherzustellen, dass die Middleware stets auf dem neuesten Stand und fehlerfrei ist.

3.3 Vorgehen zum Einsatz von Middleware in der Softwareintegration

Lernmedien	• Materialien, die die Rolle und den Einsatz von Middleware in der Softwarearchitektur erklären • Präsentationen, die Anwendungsbeispiele für Middleware im Produktionsumfeld darstellen • Vorlagen zur Planung des Middleware-Einsatzes, die auf KMUs zugeschnitten sind • Dokumente, die die Vor- und Nachteile der Middleware-Nutzung auflisten, z. B. Übersicht der Middleware-Optionen
Empfohlene Vorkenntnisse	**Awareness Softwarearchitektur**
Dauer	**10 Stunden in 4 Wochen**

4.3 Konzeption von Sicherheits- und Performance-Tests

Niveaustufe 1 2 3

Rolle	**Planer**
Lernziel	Die Lernenden können Sicherheits- und Performance-Tests der Softwarearchitektur planen.
Lernschritte / Vorgehensweise / Inhalte	**Erstellen Sie eine Liste von Sicherheitsanforderungen der Softwarearchitektur, die in Ihrem Unternehmen getestet werden sollen,** z. B. Authentifizierung, Datenverschlüsselung, Datenvalidierung, Verfügbarkeit oder Zugriffsrechte für verschiedene Benutzerrollen. **Wählen Sie Testmethoden aus, die in Ihrem Unternehmen zur Prüfung der Sicherheit in der Softwarearchitektur Ihres Unternehmens eingesetzt werden sollen:** • **Penetrationstests:** Angriffe auf ein System simulieren, um Schwachstellen zu identifizieren, sowohl mit Hilfe automatisierter als auch manueller Testmethoden (z. B. mit Tools wie Burp Suite, Metasploit). • **Sicherheitsüberprüfungen (Code Reviews):** Manuell überprüfen (z. B. per SQL-Injection, XSS) oder automatisiert testen (z. B. Einsatz von Static Application Security Testing (SAST) Tools wie SonarQube, Checkmarx), die den Code auf Schwachstellen scannen. • **Sicherheitsarchitektur-Überprüfung:** Architekturdesigns analysieren und die Implementierung überprüfen, z. B. mit Architektur-Diagrammen oder durch Risikoanalyse, um potenzielle Bedrohungen zu bewerten (z. B. STRIDE-Methode). • **Sicherheitsanforderungstests:** Die wirksame Implementierung von Sicherheitsmaßnahmen überprüfen, z. B. durch die Erstellung und Durchführung von Testfällen. • **Schwachstellenscans:** Schwachstellen automatisiert identifizieren, wie veraltete Software sowie Fehlkonfigurationen und unsichere Einstellungen erkennen. • **Fuzz Testing:** Zufällige Daten in die Software einspeisen, um Schwachstellen zu entdecken, die durch unerwartete Eingaben verursacht werden (z. B. mit Tools wie AFL – American Fuzzy Lop). **Erstellen Sie eine Liste von Performanceanforderungen der Softwarearchitektur, die in Ihrem Unternehmen getestet werden sollen,** z. B. Reaktionszeit, Durchsatz, Skalierbarkeit, Verfügbarkeit, Ressourcennutzung, Benutzerlast, Latenz, Wartbarkeit und Erweiterbarkeit. **Wählen Sie Testmethoden aus, die zur Prüfung der Performance in der Softwarearchitektur Ihres Unternehmens eingesetzt werden sollen:** • **Lasttests:** Lasttests simulieren eine spezifische Anzahl von Benutzern oder Transaktionen, um zu prüfen, wie gut die Software unter normaler und überdurchschnittlicher Last funktioniert. • **Stresstests:** Prüfen, wie das System unter extremen Bedingungen reagiert und welche Schwachstellen bei Überlastung auftreten können. • **Ausdauertests (Endurance Testing):** Prüfen, ob das System auch über längere Zeiträume stabil bleibt und keine Leistungseinbußen oder Speicherlecks auftreten. Schleichende Probleme identifizieren, die sich über die Zeit ergeben können. • **Leistungsprofilierung (Performance Profiling):** Analysewerkzeuge (z. B. JProfiler, YourKit oder VisualVM) einsetzen. Ergebnisse wie CPU- und Speicherverbrauch untersuchen, um Engpässe zu finden sind. Code oder Architektur basierend auf den Profiling-Ergebnissen anpassen. • **Kapazitätstests (Capacity Testing):** Die maximale Anzahl von Benutzern oder Transaktionen ermitteln, die ein System verarbeiten kann, bevor die Leistung beeinträchtigt wird. • **Benchmarking:** Die Leistung der Software mit definierten Standards oder anderen ähnlichen Systemen vergleichen. • **Anwendungs- und Datenbanktests:** Engpässe in der Kommunikation zwischen Anwendung und Datenbank identifizieren. SQL-Datenbankabfragen testen und optimieren. Leistung von API-Aufrufen und deren Reaktionszeiten testen. Tools wie SQL Profiler oder APM (Application Performance Monitoring) einsetzen. **Planen Sie das Vorgehen für die durchzuführenden Sicherheits- und Performance-Tests für die Softwarearchitekturen Ihres Unternehmens:** • **Ziele des Tests definieren:** Z. B. funktionale und nicht-funktionale Anforderungen festlegen. • **Testumfang festlegen:** Z. B. welche Teile des Systems und welche Szenarien getestet werden sollen. • **Testmethoden auswählen:** Z. B. Testverfahren kombinieren. • **Testumgebung einrichten:** Z. B. isolierte Testumgebung bereitstellen, um ungewollte Auswirkungen auf die übrigen Systeme zu verhindern. • **Testfälle und Szenarien entwickeln:** Z. B. normale Last, Spitzenlast und Dauerbelastung. • **Risiko des Tests bewerten und minimieren:** Z. B. Destabilisierung des Produktionssystems durch Sicherheitstests verhindern. • **Ergebnisse dokumentieren und berichten:** Z. B. Testpläne, Testfälle, Ergebnisse und Beobachtungen festhalten und für die Nachverfolgung und zukünftige Referenzen bereitstellen.

4.3 Konzeption von Sicherheits- und Performance-Tests

Lernmedien	• Materialien, die Best Practices für die Planung von Sicherheits- und Performance-Tests erläutern, z. B. erprobter Testleitfaden • Vorlagen zur Dokumentation von Testergebnissen, z. B. Testprotokoll-Vorlage • Präsentationen, die die Durchführung von Tests anschaulich erklären, z. B. Präsentation zu Testmethoden • Dokumente, die häufige Herausforderungen bei Sicherheitstests auflisten, z. B. Checkliste für Sicherheitstests
Empfohlene Vorkenntnisse	**4.1 Performance-Optimierung von Softwarearchitektur, 4.2 Sicherheitsanforderungen für die Softwarearchitektur**
Dauer	**20 Stunden in 4 Wochen**