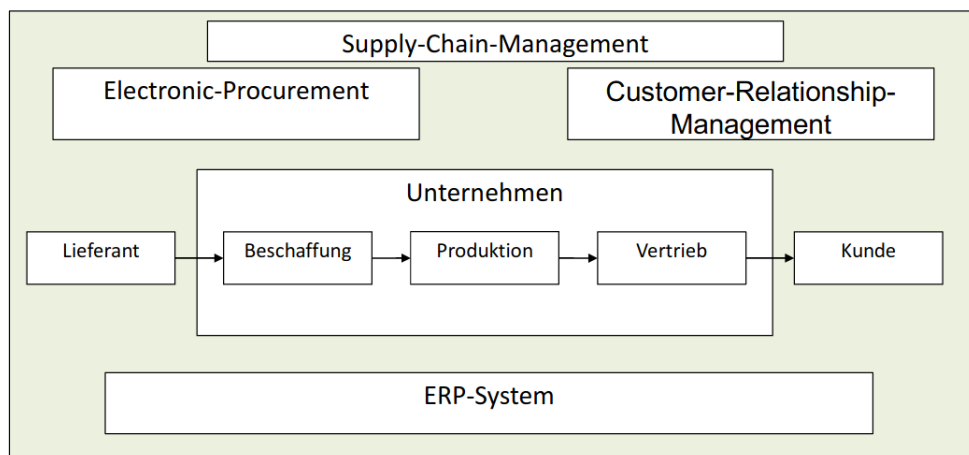


3 Management von Informationssystemen

3.1 Integrationsorientierte Informationssysteme






- **Digitaler Zwilling:**
Digitale Darstellung eines realen Objekts oder Systems (materiell oder immateriell)
- **Integrationsansätze**
 - **Datenintegration:**
Datenbestände von mehreren Informationssystemen werden zentral gespeichert (nicht mehrfach)
 - **Funktionsintegration:**
Mehrere Funktionen werden in einem Informationssystem gebündelt
 - **Prozess- oder Vorgangsintegration:**
In einem Prozess aufeinander folgende Funktionalitäten sind über ein Informationssystem nahtlos miteinander verbunden (Schnittstellen)
- **SAP:** Global führender Anbieter von ERP-Systemen
Beispiel: Verarbeitung eines Kundenauftrags
 1. Kundenauftrag wird erfasst
 2. Automatisches Ausführen von: Bestellung der Rohmaterialien, Erzeugung von Fertigungsaufträgen, Übermittlung an die Finanzplanung
 3. Rollen & Rechte verteilen
- **ERP- (Enterprise-Ressource-Planning) Systeme:**
Integrierte betriebswirtschaftliche Softwarelösungen, die eine Vielzahl Geschäftsprozesse eines Unternehmens abdecken
 - Hohe Datenintegration: Zentrale Datenbank
 - Hohe Funktions- und Prozessintegration: Schnittstellen

Informationssysteme in der Praxis: Enterprise Resource Planning (ERP)



3.2 Auswahl von Informationssystemen

- Systembereitstellung – Goldene Regeln:

	1. Software ist nie fertig.	<ul style="list-style-type: none"> ▪ Laufende Aktualisierungen ▪ iterative Verbesserungsprozesse ▪ Anpassung an Kundenbedürfnisse
	2. Software ist eine Teamleistung, niemand kann alles machen.	<ul style="list-style-type: none"> ▪ Komplexität erfordert Aufgabenteilung (Projekt) ▪ Heterogene Expertise (Domäne und Entwicklung) ▪ Verwendung von Vorgehensmodellen
	3. Großartiges entsteht durch Tausende kleiner Verbesserungen.	<ul style="list-style-type: none"> ▪ Inkrementelle Verbesserungen ▪ Modulare Architektur
	4. Software läuft nicht von selbst.	<ul style="list-style-type: none"> ▪ Ständige Überwachung ▪ Veränderung und Aktualisierung
	5. Komplexe Systeme benötigen DevOps, um gut zu laufen.	<ul style="list-style-type: none"> ▪ Kontinuierliche Veränderungen ▪ Kontinuierliches Testen ▪ Kontinuierliches Verbessern

- Softwareindustrie:

- Direkte und Indirekte Netzeffekte:
Der Nutzen eines Programms für einen einzelnen Kunden steigt häufig mit der Gesamtzahl der Nutzer.
- Keine Vervielfältigungskosten:
Hohe initiale Entwicklungskosten, anschließend jedoch nahezu kostenfreie Vervielfältigungsmöglichkeiten (Fixkostendegression)
- Kein Wertverlust durch Gebrauch
- Make or buy?

Eigenentwickelte Software („Make“)	Fremdentwickelte Software („Buy“)
<ul style="list-style-type: none"> • Nahezu vollständige Abdeckung unternehmensspezifischer Anforderungen • vollständige Integration in die Gesamtheit bereits implementierter Anwendungen • Kosten für Anpassung und Einführung entfallen weitestgehend 	<ul style="list-style-type: none"> • Eliminierung der Entwicklungszeiten durch rasche Produktverfügbarkeit • Reduzierung der Einführungs- und Übergangszeit im Vergleich zu Individual-Software • Gewährleistung der Weiterentwicklung durch den Anbieter • Unabhängigkeit der Programmentwicklung von der Verfügbarkeit der IT-Ressourcen

– **Kostenvergleichsrechnung:**

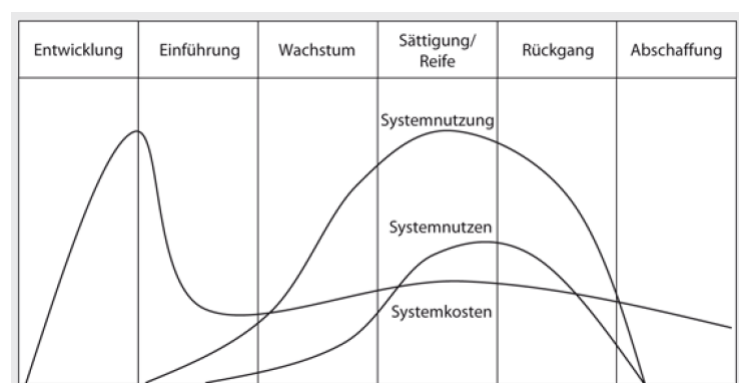
	Trad. Standard- software	Open-Source-Software	Cloud-basierte Informationssysteme
Lizenzkosten	Ja	Nein	Nein
Schulungskosten	Ja	Ja	Ja
Kosten der Infrastruktur	Ja	Ja	Nein
Einführungs- und Customizingkosten	Ja	Ja	Nein
Entwicklungskosten	Nein	Weiterentwicklung	Nein
Nutzungsentgelte	Nein	Nein	Ja
Wartungskosten	Ja	Ja	Nein

– **Nutzenkategorien von Informationssystemen:**

	monetär bewertbar	nicht monetär bewertbar
quantifizierbarer Nutzen	<ul style="list-style-type: none"> • Verkürzung von Bearbeitungszeiten • Abbau von Überstunden • Materialeinsparung • Personalreduzierung 	<ul style="list-style-type: none"> • Schnellere Angebotsbearbeitung • Weniger Terminüberschreitungen • Höherer Servicegrad • Weniger Kundenreklamationen
nicht quantifizierbarer Nutzen		<ul style="list-style-type: none"> • Erhöhung der Datenaktualität • Verbesserte Informationen • Gesteigertes Unternehmensimage • Erweiterte Märkte und Geschäftsfelder

• **Anwendungslebenszyklus:**

1. Entwicklung
2. Einführung
3. Wachstum
4. Sättigung / Reife
5. Rückgang
6. Abschaffung

**3.3 Erstellung von Individualsoftware**• **Planung eines Softwareentwicklungsprozesses:**

1. Anforderungsanalyse und Erstellung einer Spezifikation
2. Design

3. Entwicklung
4. Test und Integration
5. Auslieferung des Produkts
6. Wartung und Support

- **Strukturgetriebene Softwareentwicklung: Spiralmodell**

Wiederholender Durchlauf von Entwicklungsphasen in Iterationen von jeweils 4 Schritten mit kontinuierlicher Bereitstellung von Prototypen.

1. **Analyse:**
Definition von Rahmenbedingungen, Zielen, Anforderungen und Lösungsalternativen, Freigabe zur Umsetzung
2. **Evaluierung:**
Evaluierung der umgesetzten Lösungsalternativen. Darauf basierend Erkennung von Risiken und Erarbeitung adäquater Strategien zur Vermeidung der Risiken.
3. **Realisierung:**
Definition und anschließende Realisierung des Vorgehens, basierend auf den identifizierten Risiken.
4. **Planung:**
Review der vorangegangenen Schritte und Planung der nächsten Iteration

- **Prinzipien agiler Softwareentwicklung:**

- Transparenz und Geschwindigkeit der Entwicklung erhöhen
Reaktion auf Änderungen > Verfolgung eines festgelegten Plans
- Fehler minimieren
Funktionierende Software > Umfangreiche Dokumentation
- Kommunikation und Interaktion!
Kooperation mit Projektbetroffenen > Vertragsverhandlungen
Individuen und Interaktionen > Prozesse und Tools

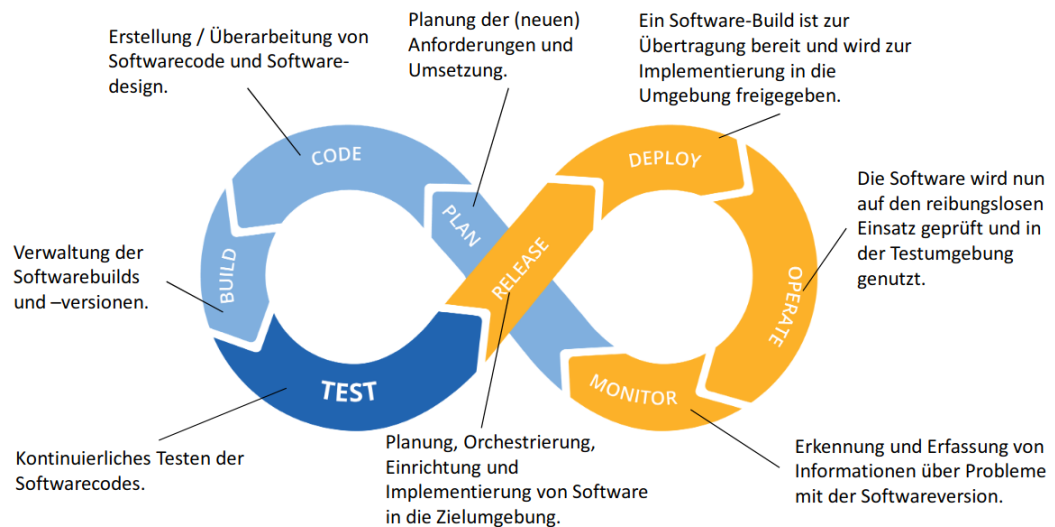
- **SCRUM:**

- Modell der agilen Softwareentwicklung
- Transparenz, Überprüfung und Anpassung
- Grober, zeitlicher Rahmen wird definiert und dann angepasst
→ Sprint Planning
- Teams sind selbstorganisiert
→ Scrum Master, Product Owner, Team
→ Daily SCRUM Meetings

- **DevOps:**

- Development + Operations
- **Ziel:** In sich verändernden Umgebungen mit schlanken und flexiblen Software-Entwicklungsprozessen schnell zu reagieren

– DevOps zur Integration von Entwicklung und Betrieb:



– Limitationen und Herausforderungen von DevOps:

- * Flexibilität
- * Automatisierung
- * Lean-Prinzipien → System optimieren
- * Alignment-Herausforderung → Überwachung der wichtigsten Indikatoren
- * Kultur- und Wissensaustausch

• *Magisches Dreieck* des Projektmanagements

Erfolgsfaktor 1: Qualität & Funktionalität

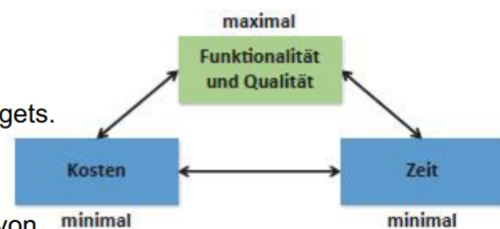
Präzise Definition der Funktionalität und Qualität sowie deren permanente Überprüfung.

Erfolgsfaktor 2: Kosten

Kostenkontrolle innerhalb des zugewiesenen Budgets.

Erfolgsfaktor 2: Zeit

Vermeidung von Verzögerungen und Festlegung von Projekt-Schritten („Milestones“)



3.4 Beschaffung von Standardsoftware

• Vorgehen zur Softwareauswahl:

1. Ist-Analyse
2. Definition der Anforderung
3. Marktanalyse
4. Vergleich der Angebote
5. Vertragsverhandlung

- **Kriterien für die Softwareauswahl:**

	Aktuelle Kriterien	Strategische Kriterien
Produktbezogene Kriterien	<ul style="list-style-type: none"> • Erfüllung funktionaler Anforderungen • Erfüllung technischer Anforderungen • Preis / Lizenzmodell 	<ul style="list-style-type: none"> • Modernität der Technologie • Flexibilität des Systems • Produktstrategie
Anbieterbezogene Kriterien	<ul style="list-style-type: none"> • Branchenerfahrung • Qualität / Ruf • Reaktionsgeschwindigkeit • Supportangebot • Seriosität 	<ul style="list-style-type: none"> • Zukunftssicherheit des Anbieters • Marktstellung des Anbieters

- **Proprietäre vs. Open Source Software:**

Traditionelle Informationssysteme	Open Source Informationssysteme
Entwicklung durch Softwareunternehmen	Entwicklung durch Programmierer verschiedener Organisationen und Freiwillige
Quellcode verbleibt im Softwareunternehmen	Quellcode der Software ist öffentlich zugänglich
Verbesserungen und Fehlerbehebungen langwieriger	Zügige Verbesserungen und Fehlerbehebungen möglich
Lizenzmodelle: Organisationen erwerben Lizenz zum Betrieb der Software	Keine Lizenzkosten
Spezifische Kundenanpassung nicht möglich	Möglichkeit der Kundenanpassung gegeben

- **IT-Outsourcing: Vor-und Nachteile**

Vorteile	Nachteile
<ul style="list-style-type: none"> • Kosteneinsparungen • Flexibilität • Schnellere Umsetzungen • Keine Investitionsausgaben • Freies Personal zur Fokussierung auf Kernkompetenzen 	<ul style="list-style-type: none"> • Verlust der Management-Kontrolle • Abhängigkeit vom Dienstleister • Sicherheitsrisiken und geringe Kontrolle über Datengut • Hoher Kommunikationsaufwand • Verlust des Know-Hows

- **Cloud Computing:**

Dynamische Bereitstellung von IT-Ressourcen über das Internet zur schnelleren Innovation und für flexiblere Ressourcen / Skaleneffekte

- **Infrastructure-as-a-Service (IaaS):**

Umfasst alle IT-Leistungen der Basisinfrastruktur z.B.Rechnerkapazitäten, Netzwerke und Speicherplatz.

- **Platform-as-a-Service (PaaS):**
IT-Leistungen, mit denen sich Anwendungssoftware und -komponenten entwickeln und integrieren lassen.
- **Software-as-a-Service (SaaS):**
Anwendungen und Dienste, die über Cloud Dienstebereitgestellt werden.

3.5 QUIZFRAGEN

- ERP-Systeme sind modular aufgebaut.
- Das Ziel der Prozess- oder Vorgangsintegration ist ursprünglich voneinander isolierte Prozesse aneinander anzugleichen oder auch zu verknüpfen.
- Vorteile von Standardsoftware (im Vergleich zu eigenentwickelter) sind die Gewährleistung der Programmwartung und -weiterentwicklung durch den Anbieter und der Profit vom *Know-How*, das von vielen Anwendern in der Software abgebildet ist.
- Bei Cloud Software werden Nutzungsentgelte verrechnet, aber es entstehen keine Wartungskosten für das nutzende Unternehmen.
- Die Zusammenarbeit verschiedener Entwickler birgt ein immenses Innovationspotential bei Open Source Software.
- Bei Open Source Software kann der Quellcode von jedermann eingesehen, verändert, manipuliert und ausgebaut werden. Dabei gibt es weder Garantien noch einen klassischen Support.
- Wenn Unternehmen auf ein Höchstmaß an technischer und organisatorischer Integrität bestehen, sollten Sie die Software eigenständig entwickeln.
- Bei eigenentwickelter Software ist die Integration der Software unkompliziert, da die Software an die Prozesse angepasst wird.
- Agile Vorgehensmodelle haben eine gute Einsetzbarkeit bei unklaren Zielen und sich ändernden Anforderungen, erhöhten Kommunikations- und Abstimmungsaufwand, hohe Flexibilität und verringerte Komplexität der Projektverwaltung.
- SCRUM basiert auf der Grundannahme, dass eine detaillierte Planung zu Beginn wenig Sinn ergibt, da Projekte schlichtweg zu komplex sind.
- Cloud Computing unterscheidet sich vom IT-Outsourcing, indem lediglich einzelne Anwendungen ausgelagert werden, der Kern der IT aber im Unternehmen verbleibt.