



Unternehmen für Bildung.

Projektmanagement

Modul: **ITG / E**

Version: 1.0
Oktober 2021

Erstellt von: IT - Abteilung

Modul: **ITG / E - Projektmanagement**

Version, Datum: 1.0, Oktober 2021

Erstellt von: IT - Abteilung

Alle Rechte vorbehalten. Kein Teil des Werkes darf in irgendeiner Form (Druck, Fotokopie oder einem anderen Verfahren) ohne Genehmigung des Herausgebers reproduziert oder unter Verwendung elektronischer Systeme verarbeitet, vervielfältigt oder verbreitet werden.

Die Modulunterlage und alle ihre Bestandteile wurden mit großer Sorgfalt erstellt und geprüft. Dennoch können Fehler nicht vollkommen ausgeschlossen werden. Der Herausgeber und die Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Literaturhinweis

Titel: IT-Berufe Grundstufe

ISBN: 978-3-14-220000-2

Verlag: Westermann Gruppe

Inhalt

TAG 1 - Das Umfeld der Softwareentwicklung analysieren	5
LE 1 - Aufgaben und Kompetenzen von Softwareentwicklern beschreiben.....	6
LE 2 - Software aus Sicht von Softwareentwicklern unterscheiden	7
LE 3 - Neuentwicklung und Anpassung von Software unterscheiden	9
LE 4 - In Softwareprojekten mitarbeiten	10
Lernzielkontrollen Tag 1.....	13
Tag 2 - Grundlagen zur Verwaltung von Daten in IT-Systemen erläutern	17
LE 1 - Informationen und Daten unterscheiden	17
LE 2 - Zahlensysteme der IT beschreiben und umrechnen.....	19
Dezimalsystem (Zehnersystem).....	20
Dualsystem (Zweiersystem).....	20
Hexadezimalsystem (Sechzehnersystem).....	21
Kennzeichnung von Zahlensystemen	22
Umrechnen von Zahlensystemen	23
Lernzielkontrollen Tag 2.....	24
Tag 3 - Darstellungsformen von Daten in IT-Systemen beschreiben	30
LE 1 - Darstellung von Zahlen und numerischen Daten.....	30
Gleitkommazahlen nach IEEE 754/IEC 60559	31
LE 2 - Darstellung von Zeichen und Text	31
LE 3 - Darstellung von grafischen Daten	36
LE 4 - Darstellung von Audiodaten	37
LE 5 - Darstellung von Algorithmen und Programmen	38
Lernzielkontrollen Tag 3.....	39
Tag 4 - Daten hinsichtlich ihrer Art und ihrer Herkunft vergleichen.....	45
LE 1 - Einteilung von Daten	45
LE 2 - Datenherkunft.....	47
Lernzielkontrollen Tag 4.....	49
Tag 5 - Aspekte der Speicherung von Daten präsentieren	51
LE 1 - Verfügbarkeit von Daten, Datenschutz und Datensicherheit	51
LE 2 - Speicheranforderungen.....	52
LE 3 - Speicherlösungen.....	54
Lernzielkontrollen Tag 5.....	56
Tag 6 - Den Prozess der Softwareentwicklung analysieren.....	59
LE 1 - Die Prozessphasen beschreiben	59
LE 2 - Software im Rahmen eines Projektes entwickeln	61
LE 3 - Vorgehensmodelle unterscheiden.....	62
Lernzielkontrollen Tag 6.....	68
Tag 7 - Programmiersprachen und -Werkzeuge unterscheiden	72
LE 1 - Aufbau von Programmiersprachen	72
LE 2 - Historische Entwicklung	73
LE 3 - Einteilung von Programmiersprachen	74
LE 4 - Auswahl einer Programmiersprache	76

LE 5 - Frameworks.....	80
LE 6 - Programmierwerkzeuge	81
Lernzielkontrollen Tag 7.....	83
Tag 8 - Den Prozess der Anforderungsspezifikation und des Softwareentwurfes beschreiben	88
LE 1 - Anforderungen an eine Software spezifizieren	88
Funktionale Anforderungen	89
Nicht funktionale Anforderungen	89
LE 2 - Lasten- und Pflichtenheft unterscheiden	90
LE 3 - Den Entwurfsprozess beschreiben	92
Lernzielkontrollen Tag 8.....	92
Tag 9 - Modellierungssprachen unterscheiden	97
LE 1 - Programmablaufplan und Struktogramm	98
LE 2 - Entscheidungstabelle	102
LE 3 - Pseudocode	104
LE 4 - Unified Modeling Language (UML)	105
Lernzielkontrollen Tag 9.....	106
TAG 10 - Vertiefung des Wissens und Modulprüfung	107
LE 1 – Einleitung.....	107
LE 2 - Modulprüfung	107
Lösungen	108
Tag 1	108
Tag 2.....	109
Tag 3.....	111
Tag 4.....	114
Tag 5.....	114
Tag 6.....	116
Tag 7.....	118
Tag 8.....	118
Tag 9.....	120

TAG 1 - Das Umfeld der Softwareentwicklung analysieren

Informationen sind in der heutigen Informationsgesellschaft ein wertvolles Gut. Im Zeitalter der Digitalisierung werden diese Informationen in zunehmendem Maße mithilfe von informationstechnischen Systemen verwaltet und Unternehmen und privaten Personen zur Verfügung gestellt. Dazu müssen zuerst die Informationen mithilfe von Daten digital abgebildet werden. Im Anschluss werden die Daten mit einer entsprechenden Software aufbereitet, verwaltet und können bei Bedarf abgerufen werden. Die Anforderungen an solche Softwarelösungen sind einem ständigen Wandel unterworfen, besonders was die Menge und die Art der zu verwaltenden Daten betrifft. Deshalb sind diese Anwendungen auch regelmäßig an neue Bedingungen anzupassen. Alternativ kann auch eine komplette Neuentwicklung in Betracht gezogen werden. Diese Überlegungen stellen die Ausgangslage für die Inhalte dieses Lernfeldes 5 dar. Es wird der gesamte Anpassungs- oder Entwicklungsprozess einer Software, angefangen mit der Spezifikation der Anforderungen, über die Planung und Programmierung bis hin zum Testen und der Dokumentation, näher betrachtet. Dazu ist es aber im Vorfeld notwendig, sich grundlegende Kenntnisse darüber anzueignen, wie Informationen mithilfe von Daten digital abgebildet werden, um welche Art von Daten es sich dabei handelt und wie verfügbar diese sind. Auch Aspekte des Datenschutzes und der Datensicherheit spielen dabei eine Rolle. Da Daten auf unterschiedliche Art und Weise in informationstechnischen Systemen verwaltet werden können, werden die dafür grundlegenden Konzepte angesprochen. In diesem Zusammenhang wird auf Dateien, Dateiformate, Datenbanksysteme und die Datenbankabfragesprache SQL näher eingegangen.

Sie sollen sich mit dem vielfältigen Arbeitsumfeld der Softwareentwicklung und -anpassung bekannt machen.



In der heutigen Arbeitswelt wird die Produktion immer mehr mit modernster Informations- und Kommunikationstechnik verzahnt. Man spricht in diesem Zusammenhang auch von industrieller Digitalisierung. Aber auch in alle anderen Bereiche des Lebens hält die Digitalisierung mehr und mehr Einzug. Eine große Rolle spielen dabei die Entwicklung moderner Software sowie die Verwaltung und Analyse von Daten. Neben der Neuentwicklung, gewinnt auch die Anpassung vorhandener Software an die Anforderungen der Digitalisierung immer mehr an Bedeutung.

LE 1 - Aufgaben und Kompetenzen von Softwareentwicklern beschreiben

Sie wollen sich darüber informieren, welche Aufgaben Softwareentwickler haben und welche Kompetenzen sie besitzen müssen.

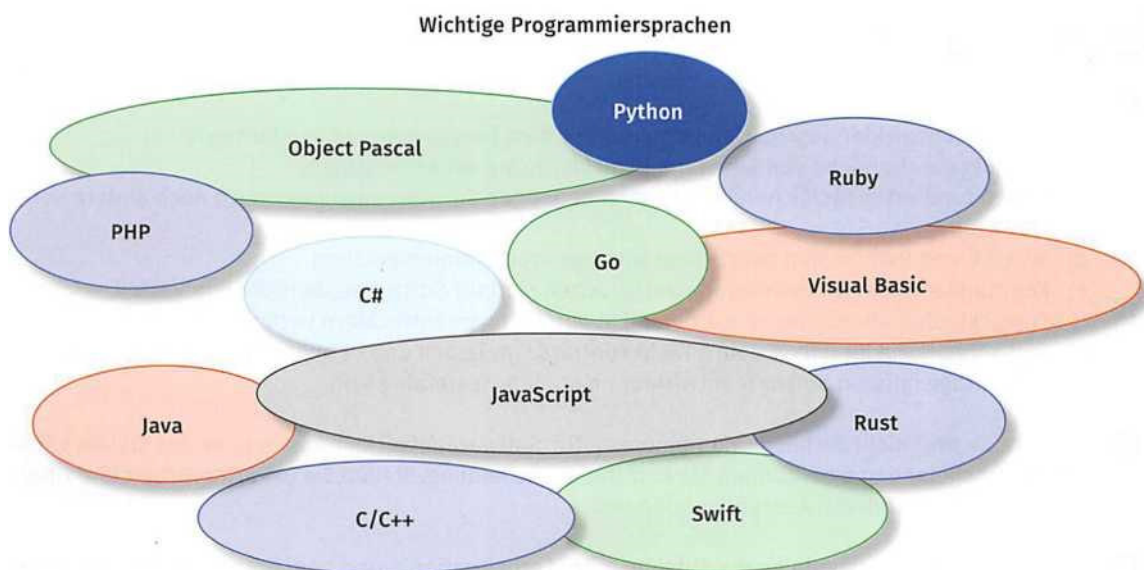
Grundsätzlich kann man sagen, dass Softwareentwickler an der Entwicklung oder Anpassung von Software oder Softwarebausteinen mitwirken. Dieses ist aber eine sehr allgemeine Aussage. In der Praxis sind die Aufgabenfelder sehr vielfältig.

Beispiele für Softwareentwickler

- Frontend- und Backend-Entwickler
- Mobile-Entwickler
- Spiele-Entwickler
- System-Entwickler
- CRM-, ERP- und CMS-Entwickler
- Datenbankentwickler
- Datenanalyst
- Test-Ingenieur
- Algorithmen-Ingenieur
- Embedded-System-Entwickler
- Web-Entwickler
- Betriebssystem-Entwickler
- DevOps-Entwickler
- Full-Stack-Entwickler
- Compiler-Ingenieur

Compiler-Ingenieur Je nachdem, an welcher Art Software die Entwickler mitarbeiten und welche Rolle sie dabei ausführen, kann man sie in unterschiedlichste Kategorien einordnen. Dabei können sich die benötigten Kenntnisse und Fähigkeiten erheblich unterscheiden. So befassen sich Datenanalysten sowohl mit der Auswertung von Daten als auch mit Entwicklungsaufgaben im Umfeld von Big Data. Andere passen CRM-, ERP- oder CMS-Systeme (z.B. SAP, WordPress) an die Wünsche des Kunden an. DevOps-Entwickler optimieren wiederum Prozesse im Bereich Systemadministration und Softwareentwicklung. Sie sind sozusagen Entwickler und Administrator in einem.

An diesen Beispielen sieht man deutlich, dass der Begriff "Softwareentwicklung" nur als Oberbegriff für eine Vielzahl von Aufgaben gesehen werden kann. Je nach Aufgabe nutzen Softwareentwickler eine passende Programmiersprache, um diese umzusetzen. Da die Programmiersprachen vom Konzept her aber oft sehr ähnlich sind, sind gute Softwareentwickler in der Lage, sich schnell in weitere Sprachen einzuarbeiten. Angefangen haben die meisten Entwickler aber mit relativ einfachen Sprachen, z. B. Python, und haben dann nach und nach ihre Programmierfähigkeiten und ihr Wissen über Konzepte von Programmiersprachen erweitert.



Neben den Kenntnissen in Programmiersprachen sind auch noch andere Fachkenntnisse erforderlich. Softwareentwickler müssen sich im Debuggen und Testen von Programmen genauso gut auskennen wie mit Vorgehensmodellen in der Programmentwicklung oder der Erstellung von Entwickler- und Benutzerdokumentationen. Wenn man ein echter Computerenthusiast ist, kann man sich Fachkenntnisse in Programmierung und Softwareentwicklung natürlich auch in der Freizeit aneignen. Und viele haben auf diesem Weg zur Softwareentwicklung gefunden. Doch in der Regel verlangen Unternehmen eine entsprechende Ausbildung.

Weitere Kompetenzen von Softwareentwicklern betreffen in zunehmendem Maße auch die sogenannten Soft Skills, wie z.B. persönliche, soziale und methodische Kompetenzen. Oft hat man es mit komplexen Fragestellungen zu tun, die man mit Geduld und Ausdauer lösen muss. Weiterhin muss ein Softwareentwickler heutzutage zunehmend teamfähig und im Umgang mit Kunden bewandert sein.

Aufgaben und Kompetenzen von Softwareentwicklern	
Aufgaben	• Analyse von Prozessen, Architekturen und Anforderungen in der IT • Entwicklung von Front- und Backend-Lösungen • Programmierung mit einer oder mehreren Programmiersprachen • Kundenberatung und -betreuung • Planen, Durchführen und Auswerten von Tests • Pflege, Wartung und Dokumentation von Software und Prozessen
Wichtige Kompetenzen	• Technisches und mathematisches Verständnis • Analytisches Denkvermögen • Geduld und Spaß an komplexen Fragestellungen • Sehr gute Englischkenntnisse in Wort und Schrift • Interesse an innovativen Entwicklungen • Denken über den Tellerrand hinaus • Abstraktionsvermögen • Teamfähigkeit und Organisationstalent • Soziale Kompetenzen im Umgang mit Kunden • Fachkenntnisse u.a. in Programmiersprachen, Softwarearchitektur und Systemdesign, Algorithmen und Datenstrukturen, Entwicklungsplattformen, Frameworks, Stacks und Tools, Datenbanken und SQL, Versionsverwaltung, Build- und Deployment-Systeme, Testen und Debugging, Entwicklungsmethoden, Modellierungswerkzeuge

LE 2 - Software aus Sicht von Softwareentwicklern unterscheiden

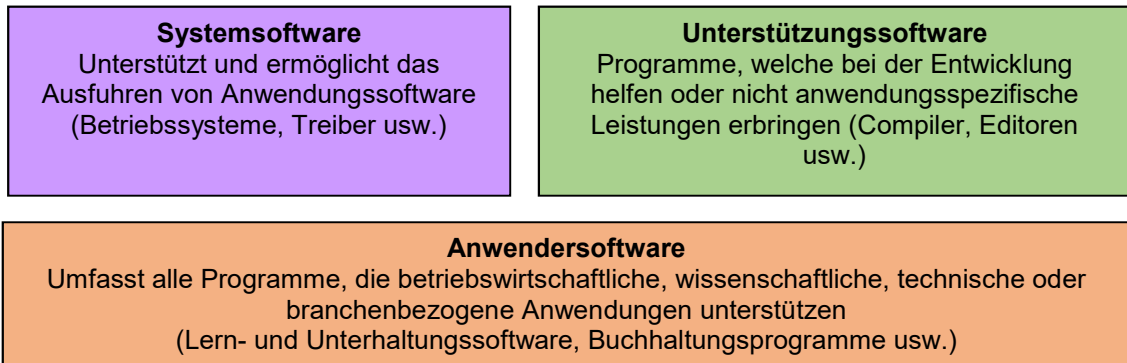
Der Begriff "Software" ist bis heute nicht einheitlich definiert. Das Wort als solches stammt aus dem Englischen und ist ein Kunstwort, welches als Gegenstück zum Wort „Hardware“ geschaffen wurde. Im Alltagsgebrauch wird der Begriff "Software" oft als Synonym für „Computerprogramme“ verwendet. Diese Betrachtungsweise schränkt den Begriff allerdings zu stark auf Programme ein. Eine Erweiterung des Softwarebegriffes stellt die Definition des Institute of Electrical and Electronics Engineers dar, der IEEE Standard 610.12.

Softwaredefinition nach IEEE Standard 610.12

Software umfasst alle Programme, vorgeschriebenen Abläufe, Dokumentation und Daten, die zum Betrieb eines Rechnersystems erforderlich sind.

Das ist eine sehr allgemeine, aber für die weiteren Ausführungen nützliche Betrachtungsweise. Hierbei werden neben den eigentlichen Programmen auch Dokumentationen und Daten als Software betrachtet.

Software lässt sich auf verschiedene Art und Weise kategorisieren. Man kann Software z. B. aufgabenbezogen, nach dem Grad der Individualität oder nach dem Nutzungsrecht einteilen. Die nachfolgende Unterscheidung ist aufgabenbezogen und in der ISO/IEC 2382 festgelegt.



Die allermeisten Softwareentwickler werden Anwendersoftware erstellen bzw. anpassen. Das Spektrum reicht dabei vom Erstellen einfacher Webseiten über Spiele bis hin zu kompletten Datenbankanwendungen. Bereiche, für welche diese Anwendersoftware erstellt wird, sind beispielsweise die Automobilindustrie, der medizinische- und Krankenhausbereich, der Finanz- und Bankensektor oder der E-Commerce- Bereich. Für Softwareentwickler ist die Art der Software bzw. die Branche insofern von Bedeutung, weil für die Erstellung von Software auch grundlegendes Fachwissen der entsprechenden Branche notwendig ist. So ist z.B. die Anpassung einer Software, welche für die Überwachung der Vitalfunktionen von Patienten zuständig ist, ohne medizinische Kenntnisse kaum vorstellbar.

Neben der Einteilung der Software nach Aufgabenbereichen bzw. Branchen kann man Software auch nach dem Grad der Individualität einteilen. Hier werden Individual- und Standardsoftware unterschieden. Diese Unterscheidung ist für Softwareentwickler ebenfalls von großer Bedeutung, weil sich daraus Konsequenzen für die Entwicklung bzw. Anpassung von Software ergeben. So können z.B. Schnittstellen nötig sein, um die Standardsoftware mit anderer vorhandener Software zu verbinden. Solche Schnittstellen sind entweder schon vorhanden und müssen noch konfiguriert werden oder sie werden separat entwickelt. Individualsoftware wird dagegen ganz auf die Belange des Kunden abgestimmt. Dieses erfordert in der Regel andere Vorgehensmodelle als bei der Erweiterung von Standardsoftware. Oft entwickelt sich auch aus einer Individualsoftware, durch Ausbau und Erweiterung des Funktionsangebotes, eine am Markt angebotene Standardsoftware.

Einteilung von Software nach dem Grad der Individualisierung	
Standardsoftware	Standardsoftware ist frei am Markt zum Kauf angebotene Software, welche standardisierbare Aufgabenstellungen abdeckt und bei verschiedenen Anwendern in mehr oder weniger der gleichen Form zum Einsatz kommt. Sie entspricht im Gegensatz zur Individualsoftware nur zu einem gewissen Grad den Anforderungen eines konkreten Nutzers und wird mittels Customizing oder Erweiterungsprogrammierung in der Regel an die im Unternehmen vorliegenden Strukturen und Prozesse angepasst.
Individualsoftware	Individualsoftware wird speziell für einen bestimmten Geschäftsprozess entwickelt und auf die Bedürfnisse des Kunden zugeschnitten. Sie lässt sich leicht an die individuellen Unternehmensprozesse anpassen und bleibt auch weiterhin flexibel, selbst wenn sich die Anforderungen ändern. Eine Anbindung zu weiterer im Unternehmen eingesetzter Software ist in der Regel ohne großen Aufwand realisierbar.

LE 3 - Neuentwicklung und Anpassung von Software unterscheiden

Neben der Neuentwicklung spielt heutzutage die Verbesserung und Anpassung von bestehenden Softwaresystemen an neue Anforderungen eine immer größere Rolle. Dabei muss man zwischen der Anpassung von Individualsoftware und Standardsoftware unterscheiden. Bei Individualsoftware erfolgt in der Regel eine Anpassung ohne große Probleme und man kann dabei sehr flexibel sein. Die Veränderungen können vom Entwickler oder bei entsprechenden Kenntnissen auch vom Kunden selbst vorgenommen werden. Dabei wird im Allgemeinen der Quellcode verändert und umgeschrieben. Die Anpassung von Standardsoftware dagegen erfolgt in der Regel durch Konfigurieren und Parametrisieren. In diesem Zusammenhang spricht man auch von **Customizing**. Wenn das Customizing der Software nicht ausreicht, um die Anforderungen des Kunden abzubilden, kann auf eine sog. Erweiterungsprogrammierung zurückgegriffen werden. Grundsätzlich sind die Möglichkeiten des Anpassens von Standardsoftware über Customizing und Erweiterungsprogrammierung aber relativ beschränkt und führen manchmal nicht zu den gewünschten Ergebnissen. In diesem Fall bleibt noch die komplette Neuentwicklung in Form einer Individualsoftware. Dabei kann am besten auf die Belange des Kunden eingegangen werden.

Softwareentwicklung

Unter Softwareentwicklung versteht man die Neuentwicklung oder die Verbesserung und Anpassung von Softwaresystemen und die damit verbundenen Prozesse.

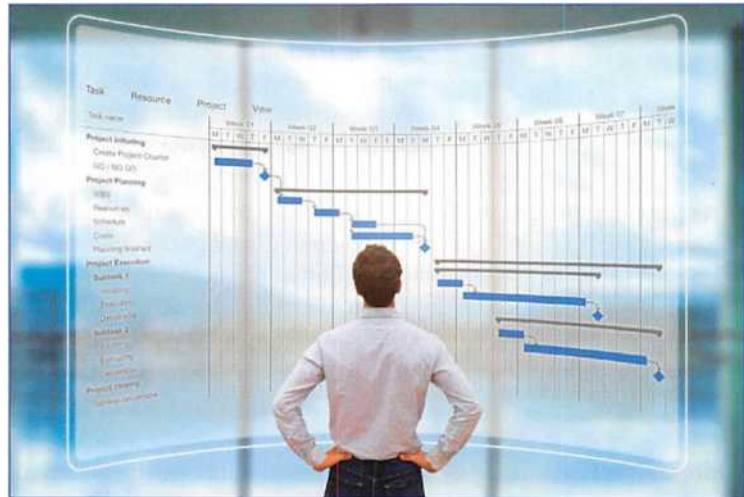
Arten der Softwareanpassung	
Customizing	Customizing bedeutet das Anpassen von Standardsoftware an die individuellen Wünsche des Kunden. Dabei wird der Quellcode nicht verändert. Es werden folgende Vorgehensweisen unterschieden: • Konfiguration: Hier wird durch die Auswahl und Konfiguration von entsprechenden Modulen die Software angepasst. • Parametrisierung: Hier wird durch das Setzen von Parametern die Software auf den benötigten Funktionsumfang angepasst. Dabei werden Teile der Standardsoftware aktiviert bzw. deaktiviert.
Erweiterungsprogrammierung	Erweiterungsprogrammierung wird zur Anpassung einer Standardsoftware genutzt, wenn die Möglichkeiten vom Customizing nicht ausreichen. Dabei wird die Software um individuell entwickelte Programmteile erweitert, welche über definierte Schnittstellen mit dem Hauptprogramm verbunden werden.
Neuentwicklung	Reichen die Möglichkeiten von Customizing und Erweiterungsprogrammierung nicht aus, wird die Software komplett neu als Individualsoftware entwickelt.
Datenmigration	Bei der Datenmigration wird ein vorhandenes System zur Verwaltung von Daten durch ein neues ersetzt. Dabei müssen die Daten vom Altsystem auf das neue System übertragen werden.

Die Anpassung einer Software kann verschiedene Bereiche umfassen. Solche wären z.B. länderspezifische Anpassungen wie Sprache und Währung oder Anpassung der Oberfläche an unternehmensspezifische Vorgaben. Weiterhin spielen auch der Umfang und die Komplexität der notwendigen Anpassung eine Rolle. Kleinere Anpassungen können von IT-Mitarbeitern des Unternehmens selber vorgenommen werden, wenn diese über die entsprechenden Kenntnisse verfügen. Größere und komplexere Anpassungen werden meistens vom Softwarehersteller, vom Softwareanbieter oder auch von externen Systemhäusern direkt beim Kunden vor Ort vorgenommen.

LE 4 - In Softwareprojekten mitarbeiten

Die Entwicklung oder Anpassung von Software erfolgt in den meisten Fällen als Softwareprojekt. Dieses kann auch als Teilprojekt im Rahmen eines größeren Projektes geschehen. Zu Beginn der Ausbildung wird man an Softwareprojekten teilnehmen, welche schon geplant wurden, und dort mehr oder weniger komplexe Aufgabenstellungen realisieren. Später wird man als Softwareentwickler an der Projektplanung und -organisation mitwirken oder diese sogar selbst vornehmen.

Die Arbeit in Projekten unterscheidet sich stark von täglicher Routinearbeit. Routinearbeit ist normalerweise dauerhaft strukturiert, wird ständig wiederholt und von den Mitarbeitern ohne große Probleme beherrscht. Dadurch sind Routineaufgaben auch gut automatisierbar und können teilweise durch künstliche Intelligenz (KI) ersetzt werden. Projekte dagegen sind einmalige Vorhaben, welche einen definierten Anfang und Abschluss besitzen und durch eine zeitliche Befristung und eine relative Neuartigkeit gekennzeichnet sind. Das bedeutet, die Aufgabenstellung wiederholt sich nicht und es kann bei der Projektabwicklung nur teilweise auf eine bereits vorhandene Planung zurückgegriffen werden.



Projektdefinition nach DIN 69901

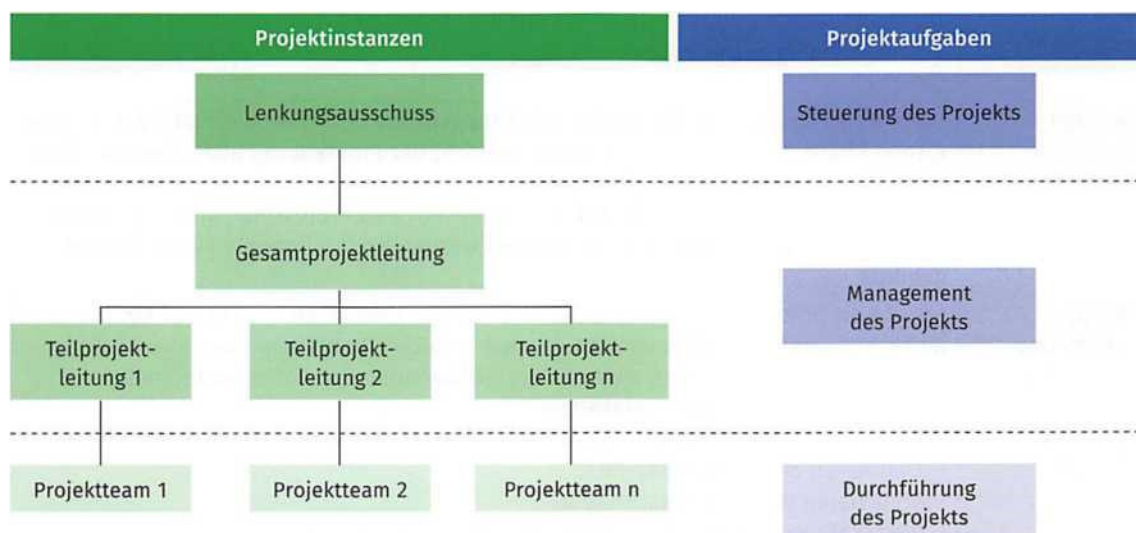
Als Projekt wird ein Vorhaben bezeichnet, welches im Wesentlichen durch die Einmaligkeit der Bedingungen in ihrer Gesamtheit gekennzeichnet ist, z. B.

- Zielvorgaben,
- zeitliche, finanzielle, personelle oder andere Begrenzungen,
- Abgrenzungen gegenüber anderen Vorhaben,
- projektspezifische Organisation.

Dabei spielen der Umfang, die Komplexität oder auch der Schwierigkeitsgrad der Aufgabenstellung nur eine untergeordnete Rolle. Ob etwas als einfach oder schwer, als komplex oder simpel oder als umfangreich oder überschaubar wahrgenommen wird, ist sehr subjektiv. Selbst einfache Aufgaben können als Projekt bearbeitet werden. Die meisten Softwareprojekte weisen aber eine höhere Komplexität auf und erfordern die Zusammenarbeit verschiedener Fachbereiche. Softwareprojekte lassen sich auf unterschiedliche Art und Weise kategorisieren. Einige davon sind in der folgenden Tabelle aufgeführt.

Mögliche Einteilungen von Softwareprojekten	
Einteilung nach	Beispiele
Projektträger	Unternehmensprojekte, Projekte von öffentlichen Einrichtungen usw.
zu erstellender Leistung	Datenbankprojekte, Spieleprojekte usw.
Auftraggeber	interne Projekte (innerhalb des Unternehmens), externe Projekte (für Kunden)
Komplexität	Kleinprojekte, mittlere Projekte, Großprojekte
Anlass	Neuentwicklungs-, Anpassungs-, Migrationsprojekte

Die interne Projektstruktur hängt stark von der Art und dem Umfang des zu entwickelnden Softwareprojekts ab. Bei kleinen Projekten können beispielsweise das Management und die Durchführung des Projektes in einer Hand liegen und von nur einer Person übernommen werden. Bei mittleren und großen Projekten wird die Projektstruktur meistens, wie in der folgenden Abbildung dargestellt, gestaltet.



Der Lenkungsausschuss ist üblicherweise für die Erteilung des Projektauftrages sowie die Freigabe der Projektziele und -planung zuständig. Er verfolgt den Projektfortschritt, legt zudem Projektprioritäten fest und trifft Grundsatzentscheidungen im Projektablauf. Die Projektleitung berichtet dem Lenkungsausschuss und ist für das Projektmanagement zuständig. Im Falle von großen Projekten werden Teilprojekte mit Teilprojektleitern gebildet. In den Projektteams arbeiten dann ein oder mehrere Softwareentwickler an der Umsetzung eines Softwareprojekts. Dazu wird die Software in einzelne Einheiten aufgeteilt und von spezialisierten Entwicklern entworfen und programmiert. Einige Softwareentwickler übernehmen z.B. die Entwicklung von Algorithmen oder Hintergrunddiensten, andere Softwareentwickler kümmern sich um die Benutzerführung und die Gestaltung der grafischen Oberfläche und wieder andere gestalten und entwickeln die Datenbank. Um Projekte in eine bestehende Unternehmensstruktur zu integrieren, spielen noch andere Organisationsformen wie die reine, die Stab- oder die Matrix-Projektorganisation eine Rolle. Diese werden hier aber nicht weiter betrachtet.

Ein Projekt durchläuft in der Regel verschiedene Phasen. Die Einteilung von Projekten in Phasen hilft sie zeitlich zu strukturieren und die Übersicht zu behalten. Die einzelnen

Projektphasen stellen in sich abgeschlossene Zeiträume innerhalb des Projektes dar und können sehr unterschiedlich gestaltet sein. Die Gestaltung hängt dabei u.a. von dem Auftrag, der Größe des Projektes und von der Branche ab.

Phase	Beschreibung
Projektstart	In dieser Phase bewerten Entscheider den Bedarf und Nutzen des geplanten Projektes für ihr Unternehmen. Fällt die Bewertung positiv aus, wird das Projekt initiiert und der Projektleiter und das Projektteam werden benannt. Es liegt ein Projektauftrag vor, welcher die Ziele und Inhalte des Projektes beschreibt. Aus Sicht des Projektteams startet das Projekt mit dem ersten offiziellen Treffen, dem Kick-off Meeting.
Projektplanung	Die Planung beginnt teilweise schon vor dem eigentlichen Projektstart und wird nach diesem intensiviert. Zu den üblichen Planungsgegenständen gehören u.a. Inhalt, Zeit, Budget, benötigte Ressourcen, Qualität, Kommunikation, Risiko, Stakeholder, Teilprojekte. Der Projektleiter und sein Team planen alle Arbeiten, die notwendig sind, um das Projektziel zu erreichen.
Projektdurchführung	Sobald die Durchführungsphase gestartet ist, nimmt der Planungsanteil drastisch ab und das Projektteam beginnt mit der Realisierung. Bei späteren Änderungswünschen werden zunächst die Machbarkeit überprüft und der Realisierungsaufwand abgeschätzt. Je nachdem wie das Ergebnis ausfällt, fließen die Änderungen in die bereits gestartete Realisierung mit ein. In dieser Phase ist die Kommunikation zwischen den Teammitgliedern untereinander und mit dem Projektleiter am intensivsten. Es wird sich regelmäßig auf Teambesprechungen und Meetings über den Arbeitsstand ausgetauscht.
Projektkontrolle	Diese Phase erfolgt parallel zur Durchführungsphase und stellt gleichzeitig auch eine eigene Phase dar. Dabei steuert und überwacht der Projektleiter alle relevanten Projektaktivitäten. Er kontrolliert u.a. die Budgeteinhaltung, achtet auf das termingerechte Abliefern, priorisiert Aufgaben, aktualisiert den Projektplan und löst Ressourcenkonflikte. Bei umfangreichen Projekten werden auch autonome Kontrollteams gebildet.
Projektabschluss	Sobald die Software fertiggestellt wurde, wird sie zusammen mit allen nötigen Unterlagen, z. B. der Kundendokumentation, dem Kunden übergeben. Wenn dieser die Software abgenommen hat, wird das Projekt beendet. Zum internen Projektabschluss gehören folgende Tätigkeiten: Erstellen eines Abschlussberichtes Evaluieren des Projektablaufes Archivieren der Projektunterlagen Wertschätzen der Arbeit des Projektteams und dessen Auflösung

Lernzielkontrollen Tag 1

1. Welche Aussagen sind richtig?

- a. Softwareentwickler beschäftigen sich nur mit dem Programmieren von Software.
- b. Die Aufgabenbereiche von Softwareentwicklern sind sehr vielfältig.
- c. Neben fundierten fachlichen Kenntnissen benötigen Softwareentwickler auch noch andere Kompetenzen.
- d. Java, C# und Python sind heutzutage wichtige Programmiersprachen.
- e. Kenntnisse in mehreren Programmiersprachen sind für Softwareentwickler von Vorteil.
- f. Entwicklerdokumentationen werden nicht von Softwareentwicklern verfasst.
- g. Softwareentwickler müssen auch Fachkenntnisse im Testen und Debuggen von Software besitzen.
- h. Heutzutage müssen Softwareentwickler nicht mehr teamfähig sein.

2. Wählen Sie drei nicht fachliche Kompetenzen für Softwareentwickler aus, welche für Sie am wichtigsten erscheinen und begründen Sie kurz Ihre Entscheidung. Stellen Sie in Partnerarbeit Ihre Überlegungen einem anderen Auszubildenden vor.

3. Welche Aussagen sind richtig?

- a. Neben Programmen gehören auch Daten zur Software.
- b. Windows 10 fällt in den Bereich der Anwendersoftware.
- c. Betriebssysteme sind Software.
- d. Linux ist ein Betriebssystem.
- e. Computerspiele sind Bestandteil der Systemsoftware.
- f. Standardsoftware ist beim Kauf schon speziell an die Bedürfnisse des Kunden angepasst.
- g. Für die Erstellung von Software muss der Entwickler auch grundlegende fachliche Kenntnisse in dem Bereich besitzen, für den die Software entwickelt wird.
- h. Die Erstellung von Individualsoftware ist mit höheren Kosten verbunden als der Kauf von Standardsoftware.

4. Finden und diskutieren Sie jeweils drei konkrete Beispiele für System- und Anwendungssoftware.

5. Überlegen und notieren Sie sich mindestens vier Beispiele für Standardsoftware, welche Sie persönlich einsetzen.

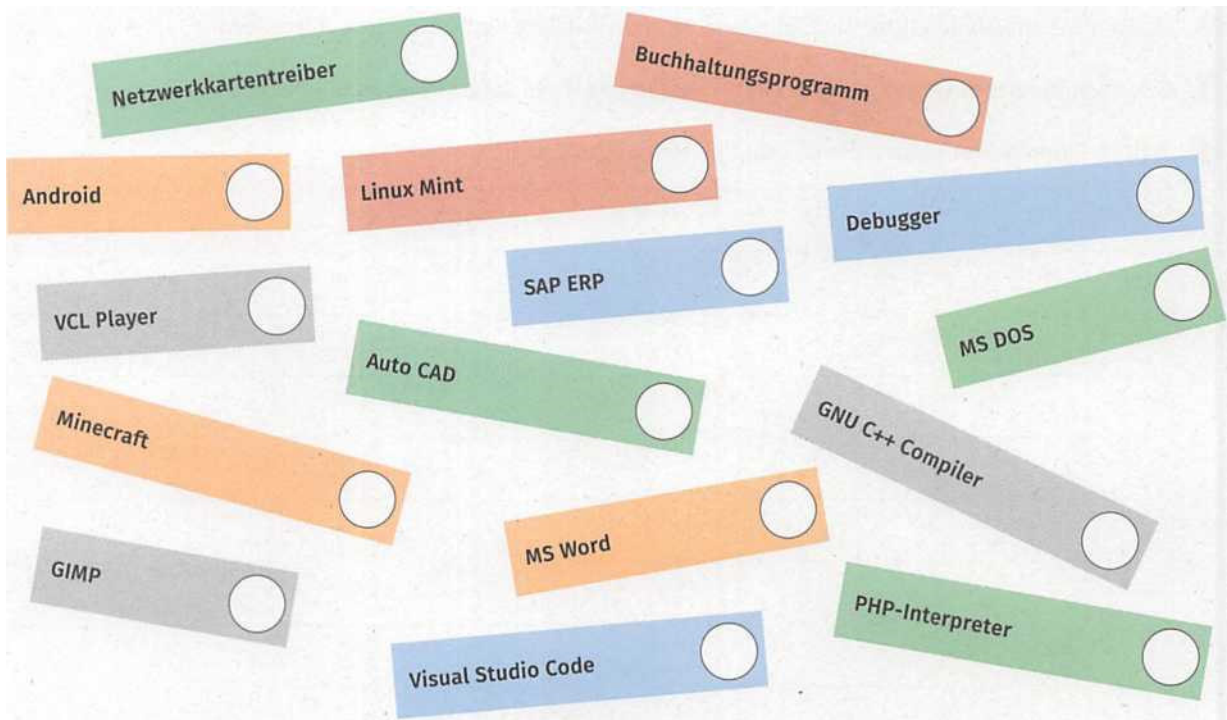
6. Recherchieren und präsentieren Sie weitere Einteilungsmöglichkeiten von Software und geben Sie jeweils zwei Beispiele an (z. B. Einteilung nach Nutzungsrecht, Freeware: Linux, Gimp).

7. Ordnen Sie Software der entsprechenden Softwareart zu.

Ordnen Sie die entsprechende Softwareart den einzelnen Software Produkten zu. Wenn einige Softwareprodukte nicht bekannt sein sollten, dann tauschen Sie sich bitte mit Ihren Mitschülern aus.

Softwarearten

- (A) Systemsoftware
- (B) Unterstützungssoftware
- (C) Anwendersoftware



8. Welche Aussagen sind richtig?

- a. Bei der Anpassung und Erweiterung von Individualsoftware wird in der Regel der Quellcode verändert.
- b. Beim Customizing von Standardsoftware wird auch der Quellcode des Programms verändert.
- c. Erweiterungsprogrammierung wird eingesetzt, wenn die Konfiguration und Parametrisierung von Software nicht ausreichen.
- d. Standardsoftware kann nicht angepasst werden.
- e. Anpassungen von Software dürfen nur der Softwarehersteller oder der Softwareanbieter vornehmen.
- f. Unter Datenmigration wird das Übertragen von Daten von einem Altsystem auf ein neues System verstanden.

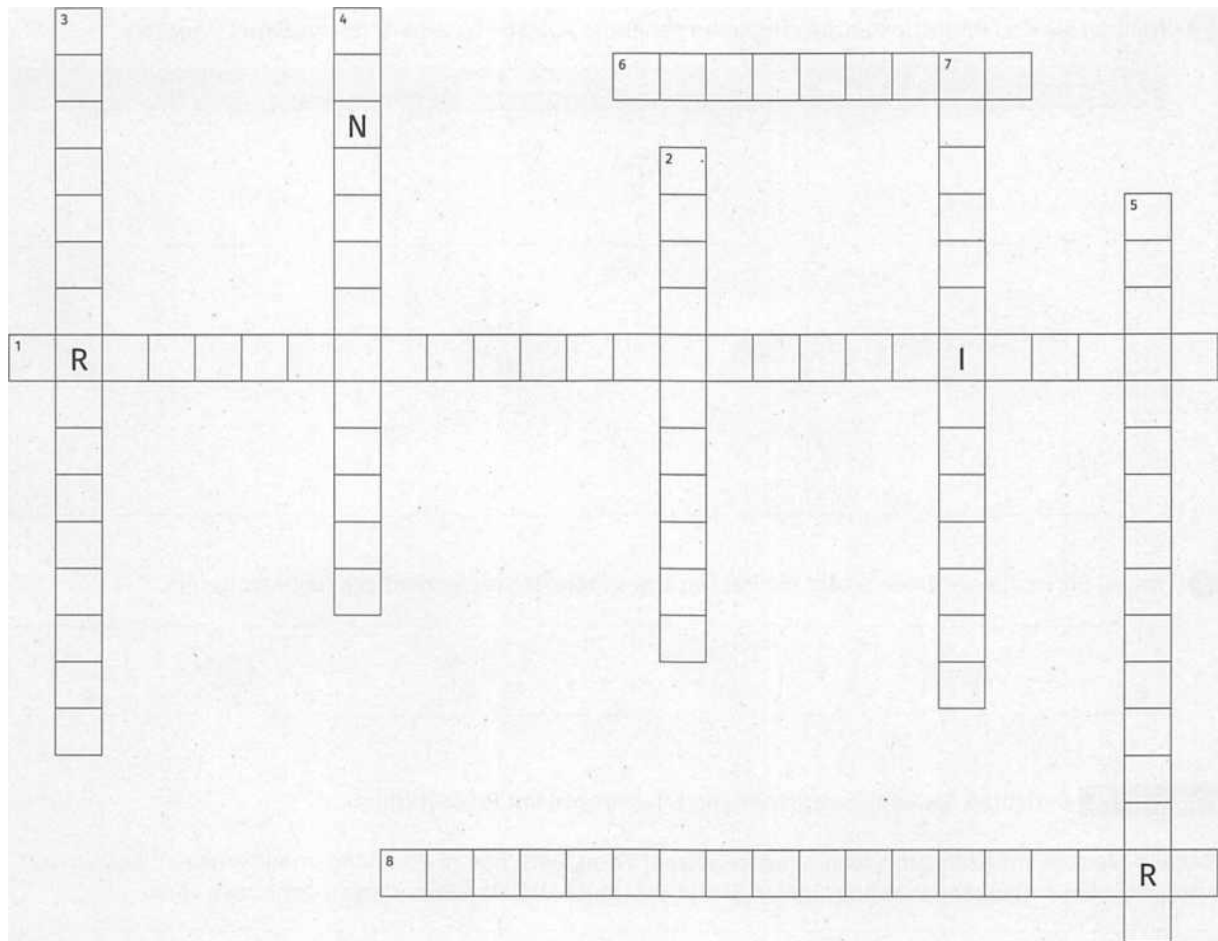
9. Erarbeiten Sie sich in Partnerarbeit ein konkretes Beispiel für Customizing einer Software, welches Ihnen bekannt ist. Das Beispiel kann aus Ihrer Firma oder aus anderen persönlichen Erfahrungen stammen. Notieren Sie sich dazu um welche Software es sich handelt, was und wie diese an konkrete Bedürfnisse angepasst wurde. Präsentieren Sie Ihre Ergebnisse der Klasse.

10. Überprüfen Sie Ihr Wissen über die Anpassung von Software.

Lösen Sie folgendes Kreuzworträtsel.

1. Wird zur Anpassung einer Standardsoftware genutzt, wenn die Möglichkeiten vom Customizing nicht ausreichen.
2. Anpassen von Standardsoftware an die individuellen Wünsche des Kunden; der Quellcode wird dabei nicht verändert.
3. Durch das Setzen von Parametern wird der Funktionsumfang von Standardsoftware im Rahmen vom Customizing angepasst.
4. Hier wird im Rahmen des Customizings durch die Auswahl und Konfiguration von entsprechenden Modulen die Software angepasst.

5. Diese Softwareart wird meistens durch Customizing an neue Anforderungen an die Bedürfnisse des Kunden angepasst.
6. Dieser Teil muss bei der Anpassung von Software durch Programmierung verändert werden.
7. Hierbei wird ein vorhandenes System zur Verwaltung von Daten durch ein neues System ersetzt.
8. Die Anpassung dieser Art von Software wird meistens durch Programmierung vorgenommen. Dabei kann sehr flexibel auf die Bedürfnisse des Kunden eingegangen werden.



11. Welche Aussagen sind richtig und welche falsch?

- a. Das Arbeiten in Projekten ist der Arbeit an Routineaufgaben sehr ähnlich.
- b. Projekte besitzen einen definierten Anfangs- und Endpunkt.
- c. Projekte lassen sich u.a. nach der Größe und nach dem Anlass einteilen.
- d. Ein Projekt kann aus mehreren Teilprojekten bestehen.
- e. Das Arbeiten in Projekten kann mehrere Tätigkeiten umfassen.
- f. Für das Projektmanagement ist in der Regel die Projektleitung zuständig.

Richtig	Falsch

- g. Für das Projektteam startet das Projekt mit dem Kick-Off-Meeting.
- h. In größeren Projekten übernehmen die Softwareentwickler spezielle Teilaufgaben wie die Programmierung von Schnittstellen.
- i. Neben der fachlichen Planung werden u.a. auch auftretende Kosten kalkuliert und das Risiko abgewogen.
- j. Die Kontrolle und Überwachung des Projektes können von speziellen Kontrollteams übernommen werden.
- k. Das Wertschätzen der Arbeit des Projektteams ist nicht besonders wichtig.

12. Sie sind einem Team zugeordnet worden, welches an einem Projekt zur Verwaltung und Auswertung von Daten für einen Kunden arbeitet. Sie sollen sich in einem Teammeeting dem Team vorstellen und über bisherige Erfahrungen bezüglich des Arbeitens in Projekten berichten.

Überprüfen Sie Ihr Wissen über die Mitarbeit in Softwareprojekten. Informieren Sie sich zunächst im Schülerbuch über Softwareprojekte und bearbeiten Sie im Anschluss die folgenden Aufgaben.

- a. Nennen Sie drei Merkmale, durch die ein Projekt gekennzeichnet ist.
- b. Beschreiben Sie kurz, wodurch sich ein Projekt von Routineaufgaben unterscheidet.
- c. Nennen Sie drei Projektinstanzen und deren generelle Aufgabe innerhalb von größeren Projekten.

Projektinstanz	Aufgabe

- d. Tragen Sie in die einzelnen Felder die Phasen eines Projektes in der richtigen Reihenfolge ein.



Tag 2 - Grundlagen zur Verwaltung von Daten in IT-Systemen erläutern

Sie sollen in einem Projekt mitarbeiten, welches der Anpassung einer Anwendungssoftware zur Verwaltung von Daten dient. Dazu erarbeiten Sie sich zunächst einmal grundlegendes Wissen über Informationen und Daten und wie diese in informationstechnischen Systemen abgebildet werden.

Dieses Kapitel beschäftigt sich mit den Begriffen „Daten“ und „Information“ und wie Informationen mithilfe von Daten dargestellt werden. Grundlage dafür bildet das duale Zahlensystem. Mithilfe von diesem werden beispielsweise numerische Daten, Text- und Bilddaten, aber auch Programme in einer für informationstechnische Systeme verständlichen Form dargestellt. Weiterhin wird auf die Art, die Herkunft und die Verfügbarkeit von Daten eingegangen. Dabei werden Aspekte des Datenschutzes und der Datensicherheit angesprochen und Speicheranforderungen und -lösungen betrachtet.



LE 1 - Informationen und Daten unterscheiden

Sie sollen sich Wissen darüber erarbeiten, wodurch sich Informationen von Daten unterscheiden und wie Informationen mithilfe von Daten verarbeitet, übertragen und gespeichert werden.

Im Alltagsgebrauch werden die Begriffe „Daten“ und „Information“ kaum voneinander abgegrenzt und teilweise sogar sinngleich verwendet. Dabei gibt es einige wichtige Unterschiede.

1. Definition der Begriffe „Information“ und „Daten“

Im allgemeinen Sprachgebrauch wird Information mit „Bedeutung“ oder „Wissen“ gleichgesetzt. Dagegen versteht man in der Informationstechnik unter Information das Wissen, welches ein Absender einem Empfänger übermittelt. Die Information führt dann bei Empfänger zu einem Zuwachs oder einer Veränderung an Wissen. Man kann auch sagen, dass eine Information die Beseitigung von Unwissenheit beim Empfänger ist.

„Information“ in der Informationstechnik

Information ist der Transfer von Wissen, welche beim Empfänger zu einer Veränderung bisherigen Wissens führt.

Informationen besitzen eine Reihe von Eigenschaften, welche den alltäglichen Umgang mit ihnen maßgeblich beeinflussen. So sind Informationen in der Regel

- immateriell, d.h., sie verbrauchen sich nicht,
- gebunden an einen „Träger“,
- nicht kostenlos, d.h., sie verursachen durch ihre Beschaffung, Bearbeitung, Speicherung usw. irgendeine Art von Kosten,
- leicht übertragbar, z. B. mündlich, schriftlich, optisch usw.,
- beliebig oft vervielfältigbar, z. B. Erstellung von Kopien usw.

Darüber hinaus besitzen Informationen einen bestimmten Wert (Nutzen). Dieser hängt maßgeblich von der Art ihrer Verwendung und von den potenziellen Empfängern ab. Daher

können Informationen auch als Ware auftreten, welche verkauft werden kann. Im Vergleich dazu werden Daten im allgemeinen Sprachgebrauch mit Werten oder Angaben zu Sachverhalten und Vorgängen verbunden, welche durch Messung, Beobachtung oder

Daten nach ISO/IEC 2382-1

Daten sind eine wiederverwendbare Darstellung der Information in einer formalisierten Art, passend gemacht für Kommunikation, Interpretation oder Prozesse.

Ähnliches gewonnen werden. In der Informationstechnik hingegen dienen Daten der Abbildung von Informationen. Die Daten können dann verarbeitet, übertragen und gespeichert werden. Die in den Daten enthaltene Information ist in der Regel jederzeit wieder reproduzierbar.

Für die Abbildung von Informationen mithilfe von Daten wurden in der Informationstechnik geeignete Formen entwickelt, durch welche auf der einen Seite eine automatische Verarbeitung der Daten ermöglicht wird und auf der anderen Seite diese auch vom Menschen interpretiert werden können. Als Grundlage dafür wird der binäre Code, oder auch Binärcode, verwendet. Dieser besteht aus den Zuständen „0“ und „1“.

Alle Daten, welche verarbeitet, übertragen oder gespeichert werden sollen, z. B. Zahlen, Text oder Bilder, müssen zunächst in eine Folge von Nullen und Einsen umgewandelt werden. Eine solche Folge wird auch Bit-Folge genannt. Für die Umwandlung selbst wird vorher festgelegt, welcher Binärcode für welche Zahl oder welches Zeichen verwendet wird. Nach der Verarbeitung muss der Binärcode wieder decodiert werden. Dazu wird wieder auf die festgelegten Codes für Zahlen, Zeichen oder Datenformate zurückgegriffen. Wie diese Codes aussehen, wird in den nachfolgenden Abschnitten dieses Kapitels behandelt.

Beispiele für die Darstellung von Daten in Binärcode	
Darstellung der Dezimalzahl 13	Darstellung des Buchstabens B
Zahl in Binärcode: 00001101	Dezimalcode in der ASCII-Tabelle: 66 Dezimalcode 66 in Binärcode: 01000010

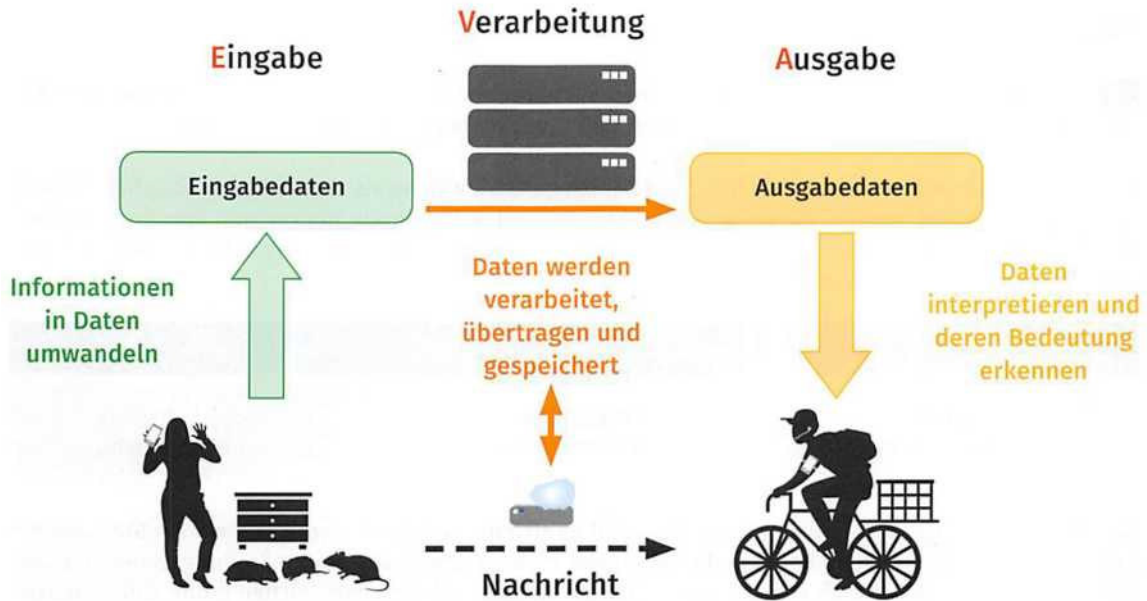
2. Abbildung und Verarbeitung von Informationen mithilfe von Daten

Informationen werden in Form von Nachrichten übertragen. Dazu muss zunächst die Information durch Daten dargestellt werden, welche danach verarbeitet, übertragen oder gespeichert werden. Eine Nachricht kann mehr Daten enthalten als nur eine Darstellung der eigentlichen Information. Dies können beispielsweise Absender- und Empfängerdaten sein oder weitere Daten, welche zur Übertragung der Information erforderlich sind. Die in den Daten enthaltene Information stellt die Bedeutung der Daten dar. Man kann an diese Information erst durch Interpretation der Daten gelangen. Dabei ist zu beachten, dass die Zuordnung zwischen Daten und Information nicht immer eindeutig ist und die Interpretation einer Nachricht stark von dem Bezugssystem abhängt.

EVA-Prinzip

EVA steht für „Eingabe - Verarbeitung - Ausgabe“ und ist ein Grundprinzip der Datenverarbeitung. Die drei Begriffe stehen stellvertretend für die Reihenfolge, in der die Daten verarbeitet werden.

So können beispielsweise die aktuellen Wetterdaten auf einer Webseite verschieden interpretiert werden und somit auch zu unterschiedlichen Schlussfolgerungen führen. Ohne ein entsprechendes Bezugssystem kann man nicht sagen, ob eine Tagesdurchschnittstemperatur von 20°C für einen Ort zu warm, zu kalt oder normal ist. Bei unterschiedlichen Bezugssystemen kann sowohl das eine als auch das andere als Information gewonnen werden.



Generell kann man sagen, dass die Abbildung von Information durch Daten und die Interpretation der Daten als Information eine gedankliche Leistung des Menschen ist und nicht im datenverarbeitenden System stattfindet. Das datenverarbeitende System kann allerdings durch Verarbeitung vorhandener Daten neue Daten erzeugen und bereitstellen. Durch die Interpretation dieser Daten können dann wiederum neue Informationen gewonnen werden. Hier können auch Systeme künstlicher Intelligenz helfen.

Beispiel für die Interpretation von Daten in Abhängigkeit vom Bezugssystem
Daten 1000001
Im Dezimalsystem würde es die Zahl „Eine Million eins“ bedeuten. Im Dualsystem würde so die Dezimalzahl 65 dargestellt. In der ASCII-Tabelle dagegen entspricht es dem Zeichen A.

LE 2 - Zahlensysteme der IT beschreiben und umrechnen

Sie sollen sich über die wichtigsten Zahlensysteme, welche in der Informationstechnik verwendet werden, einen Überblick verschaffen und diese ineinander umrechnen können. Ein Zahlensystem wird für die Darstellung von Zahlen genutzt, welche als eine Folge von Zeichen (Ziffern, Symbole) abgebildet werden. Die verwendeten Zeichen und deren Anzahl hängen von dem entsprechenden Zahlensystem ab. Einige wichtige Zahlensysteme, welche in der Informationstechnik verwendet werden, sind nachstehend aufgeführt.

Wichtige Zahlensysteme in der IT		
Dezimalsystem (Zehnersystem)	Dualsystem (Zweiersystem)	Hexadezimalsystem (Sechzehnersystem)

Bei den hier aufgeführten Zahlensystemen handelt es sich um sog. Stellenwertsysteme. Ein Stellenwertsystem ist ein Zahlensystem, bei dem die Wertigkeit eines Zeichens von seiner Stellung in der Zeichenfolge abhängig ist. Der Vorteil von solchen Systemen ist, dass mit wenigen Zeichen große Zahlen dargestellt werden können. Neben den aufgeführten gibt es noch weitere Zahlensysteme wie z.B. das Oktalsystem (Achtersystem) oder das

Vigesimalsystem (Zwanzigersystem) der Maya. Diese spielen in der IT aber nur eine untergeordnete oder keine Rolle. Deshalb werden sie hier nicht weiter betrachtet.

Dezimalsystem (Zehnersystem)

2371858₁₀

Das Dezimalsystem oder auch Zehnersystem ist das bekannteste Zahlensystem. Dieses ist der Tatsache geschuldet, dass das Dezimalsystem heutzutage im Alltagsgebrauch zur Darstellung von Zahlen verwendet wird. Die natürlichen Grundlagen des Dezimalsystems liegen in den zehn Fingern des Menschen, welche als Zähl- und Rechenhilfe benutzt wurden. Dementsprechend finden auch zehn Zeichen (Ziffern) Verwendung. Die von uns verwendeten Zeichen für das dezimale Zahlensystem sind arabischen Ursprungs, welche über Spanien nach Europa gekommen sind.

Kenngrößen und Beispiele des Dezimalsystems	
Basis	10
Zeichen	0,1,2, 3, 4, 5, 6, 7, 8,9
Mögliche unterschiedliche Zeichen pro Stelle	10
Stellenwerte	$10^0 = 1$, $10^1 = 10$, $10^2 = 100$ usw.
Beispiele	15, 22345, 701

Beispiel für die Darstellung einer Zahl im Dezimalsystem						
Beispielzahl	7	0	4	2	3	1
Stelle	4	3	2	1	1	2
Stellenwert	$10^3 = 1$ 000	$10^2 = 100$	$10^1 = 10$	$10^0 = 1$	$10^{-1} = 0,1$	$10^{-2} = 0,01$
Zahlenwert im Dezimalsystem	$(7 \cdot 1\,000) + (0 \cdot 100) + (4 \cdot 10) + (2 \cdot 1) + (3 \cdot 0,1) + (1 \cdot 0,01) = 7\,000 + 0 + 40 + 2 + 0,3 + 0,01$ $= 7\,042,31_{10}$					

Dualsystem (Zweiersystem)

01101111₂

Obwohl das Dualsystem der Menschheit schon sehr lange bekannt ist, erlangte es seine heutige Bedeutung erst mit der Entwicklung von Rechenmaschinen und der Digitaltechnik. In der Digitaltechnik werden nur zwei Zustände unterschieden. Zum Beispiel „Ein Signal liegt an“ oder „kein Signal liegt an“. Diese beiden Zustände lassen sich auch durch Zeichen beschreiben. In der Regel werden dafür die Ziffern 0 und 1 verwendet. Dadurch, dass nur zwei Zeichen zur Darstellung einer Zahl zu Verfügung stehen, wird eine Zeichenfolge im Dualsystem länger ausfallen als eine entsprechende Zeichenfolge im Dezimal- oder im Hexadezimalsystem. Das duale Zahlensystem ist das Basis-Zahlensystem in der Informationstechnik.

Kenngrößen und Beispiele des Dualsystems	
Basis	2
Zeichen	0,1
Mögliche unterschiedliche Zeichen pro Stelle	2
Stellenwerte	$2^0 = 1, 2^1 = 2, 2^2 = 4, 2^3 = 8$ usw.
Beispiele	1100, 10101111 0001 1001

Beispiel für die Darstellung einer Zahl im Dualsystem							
Beispielzahl	1	0	1	1	.	1	1
Stelle	4	3	2	1		1	2
Stellenwert	$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$		$2^{-1} = 0,5$	$2^{-2} = 0,25$
Zahlenwert im Dezimalsystem	$(1 \cdot 8) + (0 \cdot 4) + (1 \cdot 2) + (1 \cdot 1) + (1 \cdot 0,5) + (1 \cdot 0,25)$ $= 8 + 0 + 2 + 1 + 0,5 + 0,25$ $= 11,75_{10}$						

Hexadezimalsystem (Sechzehnersystem)

C3AA1F78₁₆

Wie schon festgestellt wurde, ist in der Informationstechnik das Dualsystem das vorherrschende Zahlensystem. Allerdings kann es bei der Darstellung von größeren Zahlen sehr schnell dazu kommen, dass die Zeichenfolgen aus Nullen und Einsen sehr lang und unübersichtlich werden. Um Abhilfe zu schaffen, hat man das Hexadezimalsystem eingeführt. Dabei werden 4 Bit einer Dualzahl durch ein hexadezimales Zeichen ersetzt. Auf diese Weise entsteht aus einer langen Folge von Nullen und Einsen eine kürzere hexadezimale Zahl. Somit dient das Hexadezimalsystem hauptsächlich einer übersichtlicheren Darstellung des Dualsystems und erhöht damit gleichzeitig die Lesbarkeit der Zahlen. Außerdem können auf diese Weise Daten mit weniger Zeichen abgebildet und somit komprimiert werden. Für die Darstellung von Zahlen im hexadezimalen System werden insgesamt 16 Zeichen benötigt. Dieses resultiert aus der Tatsache, dass sich mit 4 Bit insgesamt 16 Zahlen im dualen System darstellen lassen. Deswegen werden dem hexadezimalen System noch die sechs Buchstaben A bis F als Zeichen hinzugefügt. Anwendung finden Hexadezimalzahlen unter anderem als Farbcode für Farben oder zur Angabe von Speicheradressen.

Kenngrößen und Beispiele des Hexadezimalsystems	
Basis	16
Zeichen	0,1,2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
Mögliche unterschiedliche Zeichen pro Stelle	16
Stellenwerte	$16^0 = 1, 16^1 = 16, 16^2 = 256$ usw.
Beispiele	FF, A3 1E, BC F3 71 CO

Beispiel für die Darstellung einer Zahl im Hexadezimalsystem						
Beispielzahl	2	A	7	F	.	5
Stelle	4	3	2	1		1
Stellenwert	$16^3 = 4096$	$16^2 = 256$	$16^1 = 16$	$16^0 = 1$		$2^{-1} = 1/16$
Zahlenwert im Dezimalsystem	$(2 \cdot 4096) + (10 \cdot 256) + (7 \cdot 16) + (15 \cdot 1) + (5 \cdot 0,0625) + (12 \cdot 0,00390625) = 8192 + 2560 + 112 + 15 + 0,3125 + 0,046875$ $= 10879,359375_{10}$					

Kennzeichnung von Zahlensystemen

Es gibt 10 Arten von Menschen.
Diejenigen, die das Dualsystem verstehen und jene, die es nicht verstehen.

In dieser Aussage erkennt man sehr gut ein Problem, welches beim Interpretieren von Zahlen auftritt. Zahlensysteme benutzen teilweise die gleichen Zeichen für die Darstellung einer Zahl. In diesem Fall werden die Ziffern 0 und 1 sowohl vom Dezimal- als auch vom Dual- und vom Hexadezimalsystem verwendet. Je nachdem welches Zahlensystem man als Bezugssystem zum Interpretieren wählt, fällt die Information anders aus. Im Falle des Dezimal- oder des Hexadezimalsystems entsteht ein Widerspruch in der Aussage. Um solche Verwechslungen zu vermeiden, wird zur eigentlichen Zahl noch ein Index oder ein Buchstabe hinzugefügt, welcher das

System und Index	Beispiel	Zahl im Dezimalsystem
Dezimalsystem: 10 oder d	1100_{10} 1100_d	1100
Dualsystem: 2 oder b	1100_2 1100_b	12
Hexadezimalsystem: 16 oder h	1100_{16} 1100_h	4352

Darstellung der Zahlen 0 bis 15 in verschiedenen Zahlensystemen		
Dezimal-system	Dualsystem	Hexadezimal-system
00	0000	0
01	0001	1
02	0010	2
03	0011	3
04	0100	4
05	0101	5
06	0110	6
07	0111	7
08	1000	8
09	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Zahlensystem bestimmt. Die Indizierung sollte aber nur dann vorgenommen werden, wenn eine Verwechslungsgefahr besteht. Ansonsten wird auf den Index verzichtet. Korrekterweise sollte der Index als Zahl dargestellt werden und der Basis der Zahl entsprechen. Schreibweisen mit Buchstaben sind zwar erlaubt, aber wegen ihrer Ähnlichkeit zur hexadezimalen Darstellung zu vermeiden.

Umrechnen von Zahlensystemen

Manchmal ist es notwendig, die Zahl eines Zahlensystems in ein anderes Zahlensystem umzurechnen. Dazu kann man heutzutage einfach einen Rechner verwenden. Das Umrechnen ist aber nicht schwer, sodass schnell auch eine Umrechnung ohne Rechner vorgenommen werden kann, wenn man über die entsprechenden Grundkenntnisse verfügt.

Zahlensysteme umrechnen																																																																																															
Dezimalzahlen in Dualzahlen (Divisionsmethode)	Dualzahlen in Dezimalzahlen (Additionsmethode)																																																																																														
Die Dezimalzahl wird solange durch 2 geteilt bis nur noch eine dezimale 0 übrig bleibt. Bei jedem Divisionsergebnis ohne Rest wird eine 0 und bei einem Ergebnis mit Rest eine 1 notiert. Das Divisionsergebnis ist dann die nächste Ziffer. Beispiel: $233_{10} \rightarrow 11101001_2$ <table><tr><td>233</td><td>+ 2 =</td><td>116</td><td>Rest 1</td><td><i>niedrigstes Bit</i></td></tr><tr><td>116</td><td>+ 2 =</td><td>58</td><td>Rest 0</td><td></td></tr><tr><td>58</td><td>+ 2 =</td><td>29</td><td>Rest 0</td><td></td></tr><tr><td>29</td><td>+ 2 =</td><td>14</td><td>Rest 1</td><td></td></tr><tr><td>14</td><td>+ 2 =</td><td>7</td><td>Rest 0</td><td></td></tr><tr><td>7</td><td>+ 2 =</td><td>3</td><td>Rest 1</td><td></td></tr><tr><td>3</td><td>+ 2 =</td><td>1</td><td>Rest 1</td><td></td></tr><tr><td>1</td><td>+ 2 =</td><td>0</td><td>Rest 1</td><td><i>höchstes Bit</i></td></tr><tr><td>0</td><td></td><td></td><td></td><td></td></tr></table>	233	+ 2 =	116	Rest 1	<i>niedrigstes Bit</i>	116	+ 2 =	58	Rest 0		58	+ 2 =	29	Rest 0		29	+ 2 =	14	Rest 1		14	+ 2 =	7	Rest 0		7	+ 2 =	3	Rest 1		3	+ 2 =	1	Rest 1		1	+ 2 =	0	Rest 1	<i>höchstes Bit</i>	0					Hier wird jeder Stelle der Dualzahl eine dezimale Wertigkeit zugewiesen. Für die Umrechnung addiert man alle dezimalen Wertigkeiten von den Stellen der Dualzahl, welche eine 1 aufweisen. Beispiel: $11101001_2 \rightarrow 233_{10}$ <table><tr><td>1</td><td>110</td><td>10</td><td>0</td><td>1</td><td></td><td></td></tr><tr><td>2^7</td><td>2^6</td><td>2^5</td><td>2^4</td><td>2^3</td><td>2^2</td><td>2^1</td></tr><tr><td>128</td><td>64</td><td>32</td><td>16</td><td>8</td><td>4</td><td>2</td></tr><tr><td>128</td><td>1</td><td></td><td></td><td></td><td></td><td></td></tr><tr><td></td><td>+ 64</td><td>+ 32</td><td>+ 0</td><td>+ 8</td><td>+ 0</td><td>+ 0</td></tr><tr><td></td><td></td><td></td><td></td><td>+ 0</td><td>+ 1</td><td>+ 0</td></tr><tr><td></td><td></td><td></td><td></td><td></td><td></td><td>= 233</td></tr></table>	1	110	10	0	1			2^7	2^6	2^5	2^4	2^3	2^2	2^1	128	64	32	16	8	4	2	128	1							+ 64	+ 32	+ 0	+ 8	+ 0	+ 0					+ 0	+ 1	+ 0							= 233
233	+ 2 =	116	Rest 1	<i>niedrigstes Bit</i>																																																																																											
116	+ 2 =	58	Rest 0																																																																																												
58	+ 2 =	29	Rest 0																																																																																												
29	+ 2 =	14	Rest 1																																																																																												
14	+ 2 =	7	Rest 0																																																																																												
7	+ 2 =	3	Rest 1																																																																																												
3	+ 2 =	1	Rest 1																																																																																												
1	+ 2 =	0	Rest 1	<i>höchstes Bit</i>																																																																																											
0																																																																																															
1	110	10	0	1																																																																																											
2^7	2^6	2^5	2^4	2^3	2^2	2^1																																																																																									
128	64	32	16	8	4	2																																																																																									
128	1																																																																																														
	+ 64	+ 32	+ 0	+ 8	+ 0	+ 0																																																																																									
				+ 0	+ 1	+ 0																																																																																									
						= 233																																																																																									
Dualzahlen in Hexadezimalzahlen	Hexadezimalzahlen in Dualzahlen																																																																																														
Jeweils vier Binärstellen werden zu einer Hexadezimalstelle zusammengefasst. Daher ist die Umrechnung sehr einfach. Beispiel: $11101001_2 \rightarrow E9_{16}$ <table><tr><td>1110</td><td>1001</td></tr><tr><td>E</td><td>9</td></tr></table>	1110	1001	E	9	jeweils eine Hexadezimalstelle entspricht vier Binärstellen. Daher ist die Umrechnung sehr einfach. Beispiel: $E9_{16} \rightarrow 11101001_2$ <table><tr><td>E</td><td>9</td></tr><tr><td>1110</td><td>1001</td></tr></table>	E	9	1110	1001																																																																																						
1110	1001																																																																																														
E	9																																																																																														
E	9																																																																																														
1110	1001																																																																																														

Lernzielkontrollen Tag 2

1. Welche Aussagen sind richtig?

- a. Mithilfe von Daten werden Informationen abgebildet.
- b. Eine Information ist die Beseitigung von Unwissenheit beim Empfänger.
- c. Die Interpretation von Daten ist immer eindeutig.
- d. EVA ist eine Abkürzung für Eingabe - Verarbeitung - Ausgabe.
- e. Zur Übertragung von Nachrichten werden Daten verwendet.
- f. Daten werden in der Informatik mithilfe von Zeichen und Symbolen dargestellt.
- g. Der Binärcode kann drei Zustände abbilden.
- h. Daten können nur eine Information enthalten.
- i. Informationen besitzen einen gewissen Wert und sind relativ leicht übertragbar.

2. Finden und diskutieren Sie weitere Beispiele für verschiedene Interpretationen von gegebenen Daten.

3. Kreuzen Sie an, was zutrifft.

1. Welche Eigenschaften haben Informationen?

- a) Informationen sind an einen Träger gebunden.
- b) Informationsverbrauchensich.
- c) Informationen können nur digital übertragen werden.
- d) Informationen verursachen Kosten.
- e) Informationen sind leicht übertragbar.

2. Welche Aussagen über Daten sind korrekt?

- a) Daten sind eine wiederherstellbare Darstellung von Informationen.
- b) Daten können durch Messung gewonnen werden.
- c) Daten können nicht direkt übertragen werden.
- d) Daten werden mithilfe von Informationen abgebildet.

3. Welche Aussagen zum Binärcode sind falsch?

- a) Der Binärcode besteht aus den Ziffern 0 und 1.
- b) Informationen müssen in der Regel in Binärcodes überführt werden.
- c) Der Binärcode besteht aus den Ziffern 0 bis 9.
- d) Alle Daten liegen in Form von Binärcodes vor.

4. Welche Aussagen über die Darstellung von Daten in der Informationstechnik treffen zu?

- a) Texte werden in Form von Bitfolgen dargestellt.
- b) Bilder können nicht als eine Folge von Nullen und Einsen abgebildet werden.
- c) Dezimale Zahlen müssen nicht als Binärcode dargestellt werden.
- d) Es gibt festgelegte Binärcodes für Zahlen und Zeichen.

5. Welche Aussagen im Zusammenhang von Daten und Informationen sind korrekt?

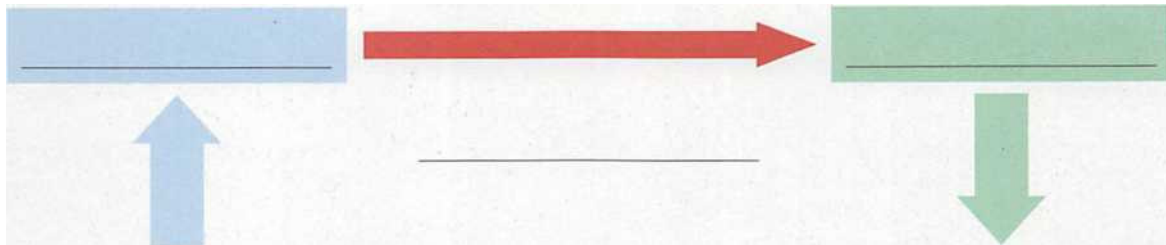
- a) Informationen, welche aus Daten gewonnen werden, sind immer eindeutig.
- b) Daten können neben der eigentlichen Information auch noch verschiedene andere Informationen enthalten.
- c) Informationen werden in Form von Nachrichten übertragen.
- d) In Daten ist immer nur eine Information enthalten.
- e) Durch die Interpretation von Daten können neue Informationen gewonnen werden.

6. Kreuzen Sie an, welche der folgenden Daten eindeutig interpretiert werden können.

- a) 1 000 als Dezimalzahl
- b) 1 000
- c) Binärfolge 1101 0000 als Buchstabe gemäß der ASCII-Tabelle
- d) 20,00 € sind zu billig.
- e) 19,73 Sekunden im Sprint sind eine sehr gute Leistung.

4. Ordnen Sie folgende Begriffe richtig in die Abbildung ein.

(A) Informationen in Daten umwandeln	(E) Daten interpretieren
(B) Ausgabe	(F) Ausgabedaten
(C) Eingabedaten	(G) Daten übertragen
(D) Verarbeitung	(H) Eingabe



5. Welche Aussagen sind richtig?

- a) Das Dualsystem benutzt zwei Ziffern zur Darstellung einer Zahl.
- b) Das letzte Zeichen des hexadezimalen Zahlensystems ist das „G“.
- c) Es gibt nur vier Zahlensysteme.
- d) Es gibt Zeichen, welche in mehreren Zahlensystemen verwendet werden.
- e) Zahlen des Dezimalsystems können zur Unterscheidung mit einem „d“ gekennzeichnet werden.
- f) Dualzahlen lassen sich nur sehr schwer in Hexadezimalzahlen umwandeln.

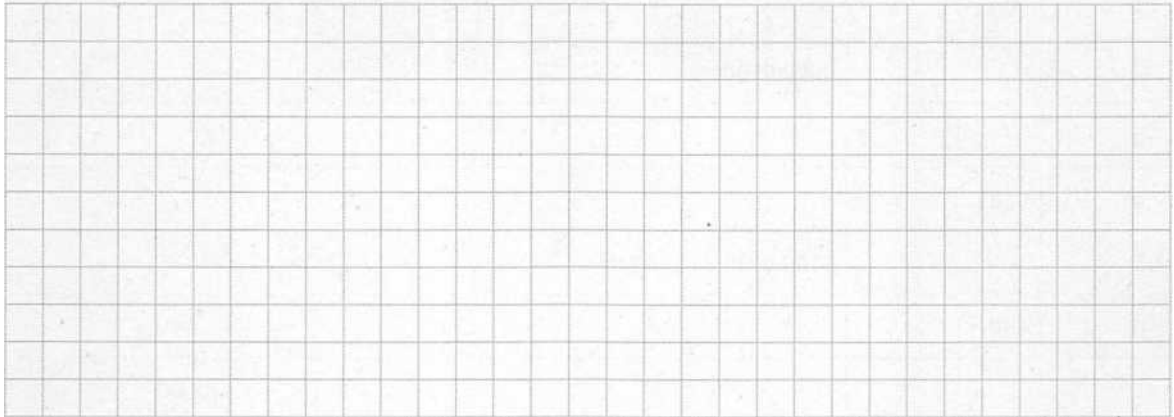
- | Kenngrößen es Oktalsystems | Suchkataloge |
|--|--------------|
| Basis | |
| Zeichen | |
| Mögliche unterschiedliche Zeichen pro Stelle | |
| Stellenwerte | |

- | Hexadezimalzahl | Dualzahl | Dezimalzahl |
|-----------------|----------|-------------|
| | | |
| | | |
| | | |
| | | |

- | Dezimalzahl | Dualzahl | Hexadezimalzahl |
|-------------|----------|-----------------|
| | | |
| | | |
| | | |
| | | |

12. Rechnen Sie folgende Dualzahl mithilfe der Additionsmethode in eine Dezimalzahl um. Schreiben Sie den vollständigen Rechenweg auf.

Dualzahl: 10001011_2



13. Rechnen Sie danach folgende Dezimalzahl mithilfe der Subtraktionsmethode in eine Dualzahl um. Schreiben Sie den vollständigen Rechenweg auf.

Dezimalzahl: 171_{10}



14. Rechnen Sie danach folgende Dualzahl mithilfe der Multiplikationsmethode in eine Dezimalzahl um. Schreiben Sie den vollständigen Rechenweg auf.

Dualzahl: 10001011_2



15. Stellen Sie Zahlen in unterschiedlichen Zahlensystemen dar.
Ergänzen Sie in den freien Kästen die entsprechenden Zahlen. Rechnen Sie ohne elektronische Hilfsmittel.

Nr.	Dezimalzahl	Dualzahl (Darstellung in ganzen Bytes)	Hexadezimalzahl
Beispiel	25	0001 1001	19
1	2000		
2		0000 0100	
3			A3
4	9103		
5		0100 0111 1111 0010	
6	430		
7			F2 BC
8			21 OF
9		1100 0011	
10	255		
11	10371		
12		0011 0000 1010 0011 1111 1000	
13			AA 37 2E E3
14	65000		
15			FFAE

16. Entwickeln Sie in Partnerarbeit Ihr eigenes Zahlensystem (z. B. Basis 3, 7 oder 11) und stellen Sie die nachfolgenden dezimalen Zahlen in Ihrem Zahlensystem dar. Die Zeichen oder Symbole Ihres Zahlensystems können Sie selbst festlegen. Präsentieren Sie im Anschluss Ihr Ergebnis der Klasse.

Daten des eigenen Zahlensystems	
Basis	
Zeichen	
Mögliche unterschiedliche Zeichen pro Stelle	
Stellenwerte	
Beispiele	

Dezimalzahl	Darstellung der Zahl in Ihrem Zahlensystem
2	
9	
11	
21	
63	
103	
555	
1001	
2370	

Tag 3 - Darstellungsformen von Daten in IT-Systemen beschreiben

Die Grundlage für die Darstellung von Daten in IT-Systemen bildet das duale Zahlensystem, weil IT-Systeme nur Nullen und Einsen verarbeiten können. Somit müssen alle erdenklichen Daten unserer Umwelt in dieser Form dargestellt werden. Die Umwandlung von Daten in Nullen und Einsen nennt man auch allgemein die Digitalisierung von Daten.

LE 1 - Darstellung von Zahlen und numerischen Daten

Man kann eine Zahl mithilfe von Zeichen eines definierten Zeichensatzes, z. B. dem ASCII-Code, oder gemäß dem Dualsystem direkt abbilden. Zahlen, welche mithilfe von Zeichen eines Zeichensatzes dargestellt werden, werden vor allem in Texten verwendet. Hierbei werden die Zahlen über den Umweg der Zeichentabelle in den Binärcode überführt. Für schnelle mathematische Berechnungen mithilfe eines Computers ist diese Form der Darstellung in der Regel nicht geeignet. Hierfür hat sich die direkte Darstellung der Zahlen mithilfe des dualen Zahlensystems als weitaus besser erwiesen. Dabei ist zwischen der Darstellung von positiven und negativen Ganz- und Fließkommazahlen zu unterscheiden.

Ganze Zahlen
71 135 -14

Ganze Zahlen sind Zahlen ohne Nachkommastellen. Entsprechend dem dualen Zahlensystem wird eine ganze Zahl mithilfe einer Folge von Nullen und Einsen abgebildet. Allerdings besitzt das Dualsystem kein (Vor-)Zeichen für negative Zahlen. Deshalb wird für die Darstellung von negativen Zahlen in der Regel das erste Bit einer Zahlenfolge als Hilfsmittel verwendet. Dabei steht eine „0“ für eine positive Zahl und eine „1“ für eine negative Zahl. Zahlen mit einem Vorzeichen werden meistens als Zweierkomplement dargestellt. Wird das erste Bit nicht zur Darstellung des Vorzeichens verwendet, dann sind alle dargestellten Zahlen nur positive Zahlen. Man spricht in diesem Zusammenhang auch von vorzeichenlosen ganzen Zahlen. In den meisten Programmiersprachen gibt es entsprechende Datentypen, z. B. Integer, unsigned int usw.

	Darstellung ganzer Zahlen ohne Vorzeichen	Darstellung ganzer Zahlen mit Vorzeichen (Zweierkomplement)
Beschreibung	Die Darstellung der Zahlen erfolgt gemäß dem dualen Zahlensystem (siehe auch Kapitel 5.2.2).	Hier wird das erste Bit als Vorzeichen-Bit verwendet. Eine „0“ steht für eine positive und eine „1“ für eine negative Zahl. Negative Zahlen werden in der Regel, um eine +0 und eine -0 zu vermeiden, als Zweierkomplement dargestellt.
Alle Zahlen, welche sich mit 4 Bit darstellen lassen	0000 = 0 1000 = 8 0001 = 1 1001 = 9 0010 = 2 1010 = 10 0011 = 3 1011 = 11 0100 = 4 1100 = 12 0101 = 5 1101 = 13 0110 = 6 1110 = 14 0111 = 7 1111 = 15	0000 = 0 1000 = -8 0001 = 1 1001 = -7 0010 = 2 1010 = -6 0011 = 3 1011 = -5 0100 = 4 1100 = -4 0101 = 5 1101 = -3 0110 = 6 1110 = -2 0111 = 7 1111 = -1
Weitere Beispiele	0000 0000 = 0 1111 1111 = 255 0111 1111 = 127 1000 0000 = 128	0000 0000 = 0 1111 1111 = -1 0111 1111 = 127 1000 0000 = -128

Gleitkommazahlen
13005,12 -14,23011

Gleitkommazahlen (Fließkommazahlen) sind Zahlen, welche Stellen vor und nach dem Komma besitzen. Die Grundlage für die Abbildung bildet wieder das duale Zahlensystem. Allerdings stellt sich hier die Frage, an welcher Stelle der Bit-Folge das Komma gesetzt wird. Da sich eine festgelegte Anzahl für die Vor- und Nachkommastellen nicht als praktikabel erwiesen hat, wurde eine Darstellung gewählt, bei der das Komma jede beliebige Stelle einnehmen kann. Es „gleitet“ oder „fließt“ sozusagen. Daher stammen auch die Wörter „Gleitkomma“ oder „Fließkomma“. Das bedeutet, wenn eine Zahl vor dem Komma viele Stellen hat, dann ist der Platz für die Anzahl der Stellen nach dem Komma geringer oder umgekehrt. Auch hierfür gibt es in den meisten Programmiersprachen entsprechende Datentypen, z. B. float (einfache Genauigkeit) oder double (doppelte Genauigkeit).

Gleitkommazahlen nach IEEE 754/IEC 60559

Die Darstellung der Gleitkommazahlen ist in der IEEE 754-2008 für 32, 64 und 128 Bit definiert. Die binäre Form einer Gleitkommazahl entspricht: Vorzeichen (V) + Exponent (E) + Mantisse (M) und ist als Standard festgelegt. Die Genauigkeit der Gleitkommazahl hängt davon ab, wie viele Bit die Mantisse umfasst.

32-Bit Gleitkommazahl: 1 Bit Vorzeichen - 8 Bit Exponent - 23 Bit Mantisse



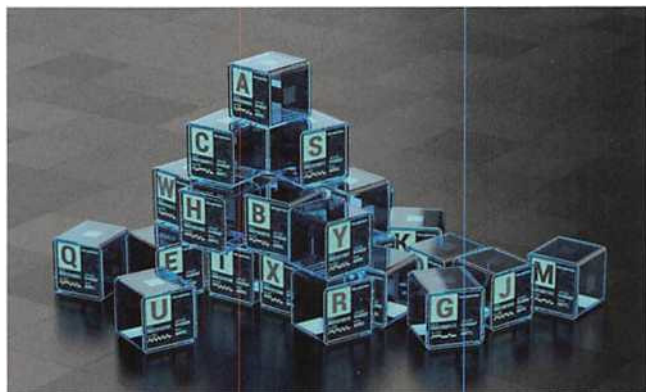
64-Bit Gleitkommazahl: 1 Bit Vorzeichen - 11 Bit Exponent - 52 Bit Mantisse



LE 2 - Darstellung von Zeichen und Text

Ein Text ist eine Folge von Zeichen, welche z. B. Buchstaben, Zahlen oder andere Symbole repräsentieren. Diese Zeichen müssen für die Verarbeitung in informationstechnischen Systemen wieder als Bit-Folge aus Nullen und Einsen dargestellt werden. Dazu wird ein sogenannter Zeichensatz definiert. Ein Zeichensatz ist eine Sammlung von Zeichen zur Darstellung von Zahlen, Buchstaben, Umlauten, Satzzeichen, Symbolen, Sonderzeichen, Steuerzeichen usw.

Jedem Zeichen in dem Zeichensatz ist ein eindeutiger Binärcode zugeordnet. Beispiele für solche Zeichensätze sind der ASCII-Zeichensatz oder der Unicode. Einige der wichtigsten Zeichensätze werden nachfolgend näher betrachtet.



ASCII-Zeichensatz

ASCII-Zeichensatz 012ABcabc<= >{,},.

Der **ASCII-Zeichensatz** (American Standard Code for Information Interchange) wurde 1963 entwickelt. ASCII ist eine 7-Bit-Zeichenkodierung

und enthält druckbare Zeichen und Steuerzeichen. Da jedes Bit zwei Werte annehmen kann, ergeben sich $2^7 = 128$ verschiedene Bit-Muster, mit denen sich dementsprechend viele Zeichen darstellen lassen. Das achte Bit ist das Paritätsbit (Ergänzungsbit zur Überprüfung auf Fehlern) und wurde früher nicht zur Zeichendarstellung verwendet. Heutzutage dient es aber oft zur Erweiterung vom ASCII-Zeichensatz auf 8 Bit, welcher 256 Zeichen enthält.

ASCII-Tabelle											
Zeichen	Dez	Hex	Zeichen	Dez	Hex	Zeichen	Dez	Hex	Zeichen	Dez	Hex
NUL	0	00	SP	32	20	@	64	40	`	96	60
SOH	1	01	!	33	21	A	65	41	a	97	61
STX	2	02	"	34	22	B	66	42	b	98	62
ETX	3	03	#	35	23	C	67	43	c	99	63
EOT	4	04	\$	36	24	D	68	44	d	100	64
ENQ	5	05	0/ /0	37	25	E	69	45	e	101	65
ACK	6	06	&	38	26	F	70	46	f	102	66
BEL	7	07	,	39	27	G	71	47	g	103	67
BS	8	08	(	40	28	H	72	48	h	104	68
TAB	9	09	)	41	29	I	73	49	i	105	69
LF	10	0A	*	42	2A	J	74	4A	j	106	6A
VT	11	0B	+	43	2B	K	75	4B	k	107	6B
FF	12	0C	,	44	2C	L	76	4C	l	108	6C
CR	13	0D	-	45	2D	M	77	4D	m	109	6D
SO	14	0E		46	2E	N	78	4E	n	110	6E
SI	15	0F	/	47	2F	O	79	4F	o	111	6F
DLE	16	10	0	48	30	P	80	50	P	112	70
DC1	17	11	1	49	31	Q	81	51	q	113	71
DC2	18	12	2	50	32	R	82	52	r	114	72
DC3	19	13	3	51	33	S	83	53	s	115	73
DC4	20	14	4	52	34	T	84	54	t	116	74
NAK	21	15	5	53	35	U	85	55	u	117	75
SYN	22	16	6	54	36	V	86	56	v	118	76
ETB	23	17	7	55	37	W	87	57	w	119	77
CAN	24	18	8	56	38	X	88	58	x	120	78

Zeichen	Dez	Hex	Zeichen	Dez	Hex	Zeichen	Dez	Hex	Zeichen	Dez	Hex
EM	25	19	9	57	39	Y	89	59	y	121	79
SUB	26	1A		58	3A	Z	90	5A	z	122	7A
ESC	27	1B		59	3B		91	5B	{	123	7B
FS	28	1C	<	60	3C	\	92	5C		124	7C
GS	29	1D	=	61	3D	]	93	5D	}	125	7D
RS	30	1E	>	62	3E	^	94	5E	—	126	7E
US	31	1F	7	63	3F		95	5F	DEL	127	7F

Die ersten 32 Zeichen sind Steuerzeichen. Diese Zeichen stellen keine Schriftzeichen dar, sondern dienen zur Steuerung von Geräten, die den ASCII verwenden. *SP* ist das Leerzeichen. Danach kommen die druckbaren Zeichen. Das letzte Zeichen *DEL* (Löschzeichen) ist ein Sonderzeichen.

Aufruf von ASCII-Zeichen

a) NUM-Tastatur rechts aktivieren b) ALT-Taste drücken und dabei Dezimalzahl der Tabelle über
Zahlenblock rechts eingeben (Alt + 64 zeigt @ z. B. im Textdokument an)

ISO-8859-Norm

ISO 8859-15: Latin-9 Westeuropäisch

Die ISO 8859 stellt eine Erweiterung des ASCII-Zeichensatzes dar. Sie umfasst insgesamt 15 Zeichensätze, welche alle 8-Bit-Zeichensätze sind.

Die ersten 128 Zeichen sind mit den Zeichen vom ASCII identisch. Die Belegung der restlichen Zeichen ist nicht definiert, wobei allerdings die letzten 96 Positionen regionale Sonderzeichen enthalten. Diese machen auch den Hauptunterschied zwischen den einzelnen Zeichensätzen aus.

Zeichensätze der ISO-8859-Norm

ISO 8859-1: Latin-1, Westeuropäisch

- ISO 8859-9: Latin-5, Türkisch

ISO 8859-2: Latin-2, Mitteleuropäisch

- ISO 8859-10: Latin-6, Nordisch

ISO 8859-3: Latin-3, Südeuropäisch

ISO 8859-11: Thai

ISO 8859-4: Latin-4, Nordeuropäisch

ISO 8859-12: wurde entwickelt, aber nie spezifiziert

ISO 8859-5: Kyrillisch

- ISO 8859-13: Latin-7, Baltisch

ISO 8859-6: Arabisch

ISO 8859-14: Latin-8, Keltisch

ISO 8859-7: Griechisch

ISO 8859-15: Latin-9, Westeuropäisch

ISO 8859-8: Hebräisch

ISO 8859-16: Latin-10, Südosteuropäisch

Latin-1 ist der Zeichensatz, welcher am häufigsten genutzt wird. Die 256 Zeichen dieses Zeichensatzes sind auch identisch mit den ersten 256 Zeichen des Unicode-Zeichensatzes. Eine etwas modernere Variante vom Latin-1-Zeichensatz ist der Latin-9-Zeichensatz. Dieser enthält nur acht andere Zeichen. So wurde z. B. das allgemeine Währungszeichen n durch das Eurozeichen € ersetzt. Es hat sich allerdings in der Praxis erwiesen, dass eine Anzahl von 256 Stellen zu gering ist, um alle internationalen Zeichen abzubilden. Um dieses Problem zu lösen, wurde der Unicode-Standard entwickelt und die ISO-8859-Norm durch die ISO/IEC 10646 ersetzt, welche praktisch dem Unicode-Standard entspricht. Aus diesem Grunde wird die ISO-8859-Norm auch nicht mehr aktiv weiterentwickelt.

Unicode



Unicode ist ein internationaler Standard, der jedes Schriftzeichen oder Textelement aller bekannten Schriftkulturen und Zeichensysteme in einem digitalen Code festlegt. Dementsprechend umfangreich ist die Anzahl der Unicode-Zeichen, welche zu Zeit etwa 230 000 Zeichen umfasst. Das Ziel des Unicode-Standards ist die Beseitigung unterschiedlicher und inkompatibler Zeichensätze aus verschiedenen Ländern und Kulturkreisen. Die erste Version erschien im Jahre 1990. Mit

Unicode Transformation Format (UTF)

Es gibt verschiedene Transformationsformate. Mit jedem Format lassen sich alle im Unicode-Standard enthaltenen Zeichen darstellen.

- **UTF-8:** Ein Zeichen wird mit 1 bis 4 Byte dargestellt. Die variable Zeichenlänge spart Speicherplatz, aber es ist ein intelligenter Algorithmus zur Kodierung der Zeichen notwendig.
- **UTF-16:** Hierbei handelte es sich um ein Kodierungsverfahren, bei dem 2 oder 4 Byte zur Kodierung eines Zeichens verwendet werden.
- **UTF-32:** Zur Darstellung eines Zeichens werden immer genau 4 Byte verwendet. Ist die einfachste Darstellungsform, welche aber auf Kosten der Speichergröße geht. So werden z.B. auch für einfache ASCII- Zeichen 4 Byte verwendet.

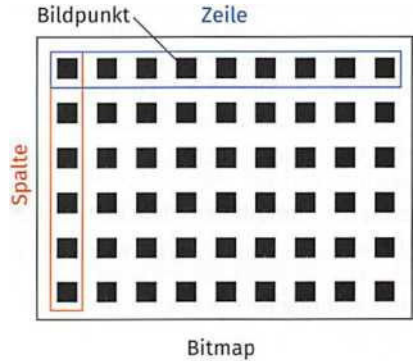
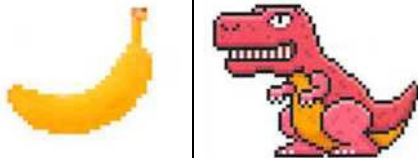
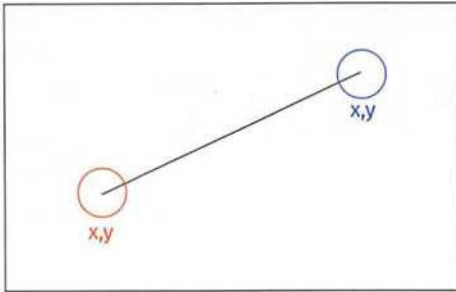
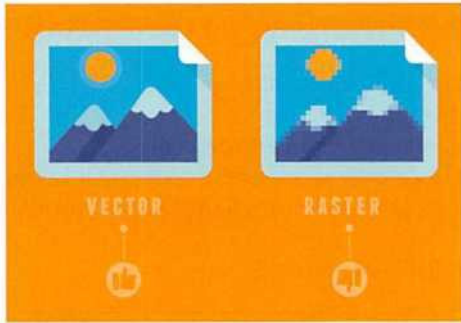
derzeit wurde der Unicode um weitere Zeichen ergänzt und hatte schnell die 8-Bit-Grenze durchbrochen. Heutzutage gibt es den Unicode auch auf der Basis von 16 bzw. 32 Bit und ist in der ISO/IEC 10646 genormt.

Beispiele für Unicode-Zeichen



LE 3 - Darstellung von grafischen Daten

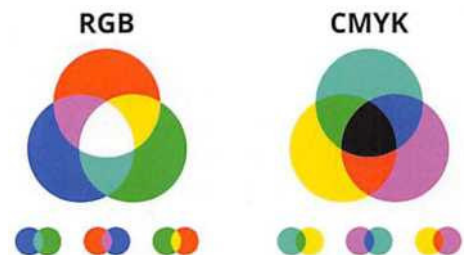
Um die grafischen Daten von Bildern, Fotos oder Grafiken digital darzustellen, gibt es prinzipiell zwei Möglichkeiten: zum einen als Vektorgrafik und zum anderen als Rastergrafik (Bitmap).

Darstellungsmöglichkeiten von grafischen Daten		
Rastergrafik	Eine Rastergrafik (Bitmap) besteht aus einer Folge von Bildpunkten (Pixel). Jeder Bildpunkt wird mit einer Bit-Folge beschrieben, in der die Helligkeit und die Farbe codiert sind. Je nach Farbtiefe kann die Anzahl der Bit pro Bildpunkt variieren. Aus der Höhe und Breite eines Bildes kann die Gesamtanzahl der Bildpunkte ermittelt werden. Zum Beispiel $600 \cdot 400 = 240\,000$ Pixel. Rastergrafiken lassen sich nicht ohne Qualitätsverlust skalieren.	
		
Vektorgrafik	 Vektor	Bei der Vektorgrafik wird das Bild aus Linien, Kreisen, Polygonen usw. zusammengesetzt. So kann eine Linie beispielsweise über die Koordinaten des Anfangs- und Endpunktes, der Linienstärke und der Farbe vollständig beschrieben werden. Da nur diese Parameter gespeichert werden, benötigen Vektorgrafiken in der Regel weniger Speicherplatz als Rastergrafiken. Ein weiterer Vorteil von Vektorgrafiken ist ihre verlustfreie Skalierbarkeit.
		

Raster-/Pixelgrafiken werden z.B. in folgenden Dateien einbezogen:

- .gif (graphic interchange format)
- .tif oder .tiff (tagged image file format)
- .jpg oder .jpeg (joint photographic experts group)
- .png (portable network graphics)
- .psd (Adobe Photoshop)
- Vektorgrafiken werden z.B. in folgenden Dateien einbezogen:
- .ai (Adobe Illustrator)
- .eps (Encapsulated PostScript)
- .svg (Scaleable Vector Graphics)
- .pdf (Portable Document Format)

Für die Abbildung von grafischen Daten ist es auch unbedingt notwendig, Farben digital darzustellen. Dazu wurden verschiedene Farbmodelle entwickelt, welche alle ihre Vor- und Nachteile besitzen. Bei den jeweiligen Farbmodellen werden die Einzelfarben aus Grundfarben zusammengesetzt. Es gibt z.B. die additive Farbmischung der Lichtfarben Rot, Grün und Blau (RGB) oder die Subtraktive Mischung der Druckfarben Cyan, Magenta, Gelb und Schwarz (CMYK). Normalerweise wird für die Intensität jeder einzelnen Grundfarbe ein Wert angegeben.



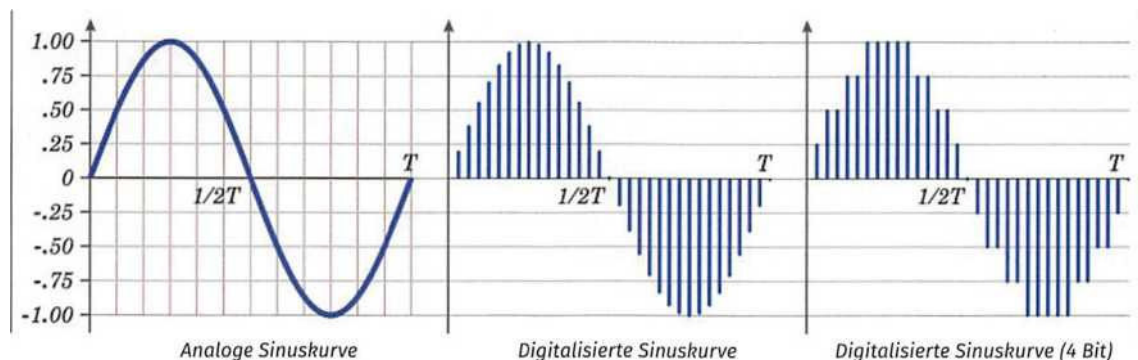
Dabei wird durch die Farbtiefe bestimmt, wie viele Bit zur Speicherung der Farbinformation verwendet werden. Je nach Farbtiefe kann nur eine bestimmte Anzahl verschiedener Farben abgebildet werden. So ermöglicht eine Farbtiefe von 8 Bit nur die Darstellung von 256 verschiedenen Farben. Mit 16 Bit können schon 65 536 und mit 24 Bit sogar mehr als 16,7 Millionen (True Color) dargestellt werden.

Farbtiefe

Die Farbtiefe bestimmt, wie viele Bit zur Speicherung der Farbinformation verwendet werden.

LE 4 - Darstellung von Audiodaten

Töne werden mithilfe eines Verfahrens digitalisiert, das man Sampling nennt. Beim Sampling wird das Audiosignal (die Amplitude) in bestimmten Zeitabständen gemessen und die Werte der einzelnen Abtastvorgänge als numerischer Wert digital abgespeichert. Die Frequenz (Tonhöhe) ergibt sich dabei aus der zeitlichen Verteilung der Amplituden. Wie bei grafischen Daten die Qualität und die Menge der gespeicherten Informationen beispielsweise durch die Farbtiefe beeinflusst wird, so gibt es auch bei Audiodaten entsprechende Kenngrößen, z.B. die Sampling-Rate, die Sampling-Tiefe und die Anzahl der Tonkanäle.



Kenngrößen der digitalen Abbildung von Audiodaten	
Sampling-Rate	Die Sampling-Rate gibt die Anzahl der Messungen pro Sekunde an. Je höher die Frequenz dieser Messungen, desto höher ist die Tonqualität. Audio-CDs haben beispielsweise eine Sampling-Rate von 44,1 kHz. Es wird also 44 100 Mal pro Sekunde gemessen. Computerspiele verwenden in der Regel eine Sampling-Rate von 22,05 kHz.
Sampling-Tiefe	Die Sampling-Tiefe gibt an, wie viele Bit pro gespeichertem Ton verwendet werden. Dadurch wird die Anzahl der möglichen Amplituden unterschieden. Bei Audio-CDs sorgt eine Sampling-Tiefe von 16 Bit (65 536 unterschiedliche Amplituden) für eine gute Tonqualität.
Anzahl der Tonkanäle	Die Anzahl der Tonkanäle bestimmt, ob Raumklänge gespeichert werden oder nicht. Je höher die Anzahl der Kanäle ist, umso realistischer wird das Hörerlebnis. Bei der Verwendung von nur einem Kanal werden die Audiodaten mono gespeichert. Für Stereo werden zwei Kanäle verwendet. Noch mehr Kanäle finden bei Quadrophonie, Dolby Surround oder 5.1-Sound Anwendung.

LE 5 - Darstellung von Algorithmen und Programmen

Algorithmus

Ein Algorithmus ist eine definierte und endliche Vorgehensweise, mit der ein Problem gelöst werden kann. Er besteht aus Anweisungen, welche Schritt für Schritt und in der definierten Reihenfolge abgearbeitet werden.

Dem Begriff "Algorithmus" begegnet man heutzutage sehr häufig. In vielen Fällen wird er mit den Begriffen „Programm“ oder sogar „Software“ synonym verwendet. Dabei gibt es jedoch einige Unterschiede. Ein Algorithmus ist erst einmal nichts weiter als eine Handlungsvorschrift, mit der man ein Problem lösen kann. Algorithmen werden dann mithilfe von Programmen auf IT-Systemen umgesetzt. Dazu bedient man sich einer Programmiersprache, in welcher das Programm geschrieben wird. Der Algorithmus selbst ist aber programmiersprachenunabhängig. Der älteste bekannte Algorithmus ist z.B. der Euklidische Algorithmus, mit dessen Hilfe der größte gemeinsame Teiler zweier natürlicher Zahlen ermittelt werden kann.

Unter einem Programm kann man einen Text (Quellcode) verstehen, der aus Befehlen und Speicheradressen, in welchen Daten gespeichert werden, besteht. Ein Programm wird deswegen in Textform geschrieben, weil der Mensch so das Programm besser lesen und verstehen kann. Ein Compiler übersetzt diesen Text, welcher in einer bestimmten Programmiersprache verfasst ist, in Maschinenbefehle. Jedem Maschinenbefehl ist wie einem Zeichen in einem Zeichensatz eine bestimmte Folge aus Nullen und Einsen zugeordnet. Diese Bit-Folge kann dann vom Prozessor verarbeitet werden. Durch die Ausführung von Programmen können wiederum andere Daten in IT-Systeme eingegeben, verarbeitet und ausgegeben werden.

```
# -----
# Euklidischer Algorithmus
# -----
def ggt(a, b):
    if a == 0: return b while b != 0:
    if a > b: a = a - b else:
        b = b - a return a
```

Quellcode für den Euklidischen Algorithmus in Python

Lernzielkontrollen Tag 3

1. Welche Aussagen sind richtig?
Für die Darstellung eines negativen Vorzeichens wird das erste Bit einer Bit-Folge verwendet.
 - a. Ganze Zahlen und Gleitkommazahlen werden auf die gleiche Weise dargestellt.
 - b. Zur Darstellung von negativen ganzen Zahlen wird überwiegend das Zweierkomplement verwendet.
 - c. Die Genauigkeit von Gleitkommazahlen hängt von der Anzahl der Bit der Mantisse ab.
2. Stellen Sie folgende dezimale Zahlen als Dualzahl dar: -71,-126, -3, -16.
3. Stellen Sie folgende duale Zahlen als Dezimalzahl dar. Dabei wird das erste Bit als Vorzeichen-Bit verwendet: 0110, 1010, 1001 1111, 0110 0000,1111 0000
4. Stellen Sie vorzeichenbehaftete Dezimalzahlen als Dualzahlen dar.
Ergänzen Sie in den freien Zellen der Tabelle die entsprechenden Zahlen. Rechnen Sie ohne elektronische Hilfsmittel.

Nr.	Dezimalzahl	Vorzeichenbehaftete Dualzahl (Zweierkomplement)
Beispiel	-2	1111 1110
1	-15	
2		00001100
3		1111 0011
4	-6007	
5		0100 0111 1111 0010
6	30	
7	-30	
8		0100 0011
9	<	1100 0011
10	-155	
11	103	
12		1011 0000 1010 0011 1111 1000
13	222	
14		1000 1000
15	-222	
16		0001 0001
17		1001 0001
18	-800	
19	800	•
20		1111 1011 1010 1001

5. Ermitteln Sie die numerischen Datentypen für eine Programmiersprache.

Notieren Sie sich in Partnerarbeit alle numerischen Datentypen für eine Ihnen bekannte Programmiersprache und kennzeichnen Sie im Anschluss, ob es sich um einen Fließkomma- oder um einen Ganzzahldatentyp handelt. Tragen Sie danach den Wertebereich und die Anzahl der verwendeten Bytes für diesen Datentyp in die Tabelle ein. Falls Ihnen nicht alle Datentypen und die zugehörigen Informationen bekannt sind, recherchieren Sie die fehlenden Informationen im Internet.

Hinweis: Für den Fall, dass Sie noch keine Programmiersprache kennen, ermitteln Sie die entsprechenden Daten entweder für Java, C# oder Python.

Datentyp	Fließkomma oder Ganzzahldatentyp	Wertebereich	Anzahl der verwendeten Bytes

6. Welche Aussagen sind richtig?

- Ursprünglich wurden für den ASCII-Zeichensatz nur 7 Bit für die Darstellung der Zeichen verwendet.
- Das Zeichen A steht an der Stelle 41₁₆ in der ASCII-Tabelle.
- Die ISO 8859 besteht aus 15 Zeichensätzen.
- Die Zeichensätze der ISO 8859 werden noch aktiv weiterentwickelt.
- Der Latin-I-Zeichensatz und der Latin-9-Zeichensatz unterscheiden sich kaum.
- Ziel der Einführung vom Unicode war es, einen einheitlichen Standard für die Darstellung aller bekannten Zeichen zu schaffen.
- Um alle Zeichen vom Unicode darzustellen, reichen 16-Bit aus.

7. Wandeln Sie mithilfe der ASCII-Tabelle folgende Wörter in Binärcode um.

Text	IT-Berufe	Python
Dezimalzahlen laut ASCII-Tabelle		
Hexadezimalzahlen laut ASCII-Tabelle		
Binärcode		

8. Wandeln Sie mithilfe der ASCII-Tabelle folgenden Binärcode in Text um.

Binärcode	01000001 01010011 01000011 01001001 01001001 00101101 01000011 01101111 01100100 01100101	01000100 01100001 01110100 01100101 01101110 01100010 01100001 01101110 01101011
Hexadezimal- zahlen		
Dezimalzahlen		
Text laut ASCII- Tabelle		

9. Werten Sie einen englischen Text über Unicode aus.

Ein älteres Programm soll auf Unicode umgestellt werden. Sie sollen an der Projektbesprechung teilnehmen. Deshalb informieren Sie sich im Vorfeld über Unicode im Schülerbuch. Außerdem liegt Ihnen noch folgender englischer Text über Unicode vor:

Unicode is an information technology standard for representation of text expressed in most of the world's writing systems. Before the Unicode standard was developed, there were many different systems, called character encodings. These earlier character encodings were limited and did not cover characters for all the world's languages. Pictographic languages, such as Japanese, were a challenge to support with these earlier encoding standards. Another option with Unicode is to send emoji to express love, thanks, and congratulations all over the world. Emoji are pictographs that are typically presented in a colourful form and used inline in text. They represent things such as faces, weather, vehicles or icons that represent emotions or feelings. To the computer they are simply another character from the Unicode.

Übersetzen Sie den englischen Text sinngemäß ins Deutsche.

10. Welche Aussagen sind richtig?

- a. Die gewählte Farbtiefe einer Rastergrafik hängt von der Anzahl der Farben ab, welche dargestellt werden sollen.
 - b. Die Anzahl der Bildpunkte einer Bitmap kann aus deren Höhe und Breite in Pixel berechnet werden.
 - c. Vektorgrafiken benötigen weniger Speicherplatz als Rastergrafiken.
 - d. Vektor- und Rastergrafiken skalieren ohne Qualitätsverlust.
 - e. Es gibt additive und Subtraktive Farbmodelle.

11. Informieren Sie sich über ein weiteres Farbmodell und präsentieren Sie Ihre Ergebnisse.
12. Stellen Sie Farben digital dar und berechnen Sie Farbwerte.
Ergänzen Sie in der Tabelle die fehlenden Einträge. Als Grundlage dient das RGB-Farbmodell.
Hinweis: Wenn Ihnen zu den Farbwerten die entsprechende Farbe nicht bekannt sein sollte, ermitteln Sie diese im Internet.

Name der Farbe	Dezimalwerte R G B	Hexadezimalwert	Binärcode
Blau	0 0 255	#0000FF	0000 0000 0000 0000 1111 1111
Rot			
	0 128 0		
		#800080	
			1100 0000 1100 0000 1100 0000
Gelb			
	255 255 255		
		#FFA500	
			0100 0001 0110 1001 1110 0001
Schwarz			
	190 190 190		
		#D3D3D3	
			1010 0101 0010 1010 0010 1010

13. Welche Aussagen sind richtig?
- Das Verfahren, mit dem Audiodaten digitalisiert werden, nennt sich Sampling.
 - Die Tonhöhe ergibt sich aus der zeitlichen Verteilung der Amplituden.
 - Die Sampling-Rate bestimmt, wie viele Bit zur Speicherung von einem Ton verwendet werden.
 - Die Anzahl der Tonkanäle beeinflusst das Hörerlebnis.
 - Für Stereo braucht nur ein Kanal verwendet zu werden.

14. Erläutern Sie folgende Begriffe:

Sampling-Rate:

Sampling-Tiefe:

15. Vergleichen Sie digitale Tonformate.
Recherchieren Sie im Internet folgende Tonformate und machen Sie sich Notizen zur Herkunft, Ihrem Einsatz, den Kenngrößen (z.B. Anzahl der Tonkanäle) und den Vor- und Nachteilen. Fügen Sie in der letzten Zeile der Tabelle noch ein weiteres, selbstgewähltes Tonformat hinzu.

Tonformat	Notizen
Dolby Digital	

Dolby True HD	
Dolby Digital Plus	
DTS Digital	
DTS ES	
SDDS	

16. Welche Aussagen sind richtig?

- a. Algorithmus und Programm sind identische Begriffe.
- b. Algorithmen sind an Programmiersprachen gebunden.
- c. Programme bestehen aus Maschinenbefehlen und Speicheradressen.
- d. Mithilfe von Programmen werden Algorithmen abgebildet.

17. Erläutern Sie folgende Begriffe:

Algorithmus:

Programm:

18. Überprüfen Sie Ihr Wissen über die Darstellungsformen von Daten in IT-Systemen.

Darstellungsformen von Daten in IT-Systemen

1. Welche Aussagen über Gleitkommazahlen sind richtig?

- a) Gleitkommazahlen haben eine festgelegte Anzahl an Vorkomma- und Nachkommastellen.
- b) Gleitkommazahlen haben kein Vorzeichenbit.
- c) Gleitkommazahlen werden u.a. durch einen Exponenten und eine Mantisse abgebildet.
- d) Gleitkommazahlen haben immer die gleiche Genauigkeit.

2. Welche Aussagen über die Darstellung von Zeichen und Text in IT-Systemen sind richtig?

- a) Buchstaben werden als Bitfolge von Nullen und Einsen dargestellt.
- b) Alle Zeichen eines Textes sind in sogenannten Zeichensätzen codiert.
- c) Es gibt genau einen Zeichensatz für die Darstellung von Text.

- d) Unicode ist ein internationaler Standard, der jedes Schriftzeichen oder Textelement aller bekannten Schriftkulturen und Zeichensysteme in einem digitalen Code festlegt.

☐

3. Welche Aussagen über die digitale Darstellung von Grafiken sind falsch?

- a) Es gibt mindestens zehn verschiedene Möglichkeiten, um Grafiken digital abzubilden.
b) Eine Rastergrafik besteht aus einer Folge von Bildpunkten.
c) Vektorgrafiken benötigen in der Regel weniger Speicherplatz als Rastergrafiken.
d) Rastergrafiken sind ohne Qualitätsverlust skalierbar.

☐
☐
☐
☐
☐

4. Welche Aussagen über Farbmodelle sind richtig?

- a) Es gibt genau zwei Farbmodelle.
b) Das RGB-Farbmodell ist eine Subtraktive Mischung aus den Farben Rot, Grün, Blau.
c) Das Farbmodell CMYK ist eine Subtraktive Mischung aus den Farben Cyan, Magenta, Gelb und Schwarz.
d) Eine Farbtiefe von 8 Bit ermöglicht eine Darstellung von 256 verschiedenen Farben.

☐
☐
☐
☐
☐

5. Welche Aussagen über die digitale Abbildung von Audiodaten sind richtig?

- a) Die Sampling-Rate gibt die Anzahl der Messungen pro Millisekunde an.
b) Die Sampling-Tiefe gibt an, wie viele Bytes für die Speicherung eines einzelnen Tons verwendet werden.
c) Das Verfahren zur digitalen Darstellung von Audiodaten heißt Audiodigit.
d) Bei Dolby Digital werden mehrere Tonkanäle verwendet.

☐
☐
☐
☐
☐

6. Welche Aussagen zu Algorithmen und Programmen sind falsch?

- a) Algorithmen und Programme sind dasselbe.
b) Ein Algorithmus ist eine definierte Vorgehensweise, mit der ein Problem gelöst werden kann.
c) Programme werden in einer Programmiersprache geschrieben.
d) Algorithmen sind programmiersprachenunabhängig.
e) Programme werden durch Algorithmen dargestellt.

☐
☐
☐
☐
☐
☐

Tag 4 - Daten hinsichtlich ihrer Art und ihrer Herkunft vergleichen

Heutzutage sind Daten sehr wertvoll und täglich nutzen wir eine Vielzahl davon, um uns zu informieren und daraus Entscheidungen für unser Handeln abzuleiten. Das gilt sowohl für Privatpersonen als auch für Unternehmen. Für die letzteren ist es, neben der Auswertung von Daten, auch notwendig eigene Daten, über Unternehmens- und Standortgrenzen hinweg, global zugänglich und abrufbar zu machen. Daraus folgt, dass die Beschaffung, Aufbereitung und Bereitstellung von Daten auch wichtige Bestandteile ihrer Tätigkeit im Unternehmen sind. Aber aus welchen Quellen kommen all diese Daten und welche Arten von Daten gibt es überhaupt?

LE 1 - Einteilung von Daten

Daten können auf unterschiedliche Art und Weise eingeteilt werden. Einige Kategorien sind in folgendem Schaubild aufgeführt. Die Art der Daten wirkt sich sehr stark auf ihre Darstellung, Verarbeitung und Speicherung aus und beeinflusst damit viele Teile der Informationstechnik. Wenn zum Beispiel die Daten im Unternehmen ordentlich klassifiziert sind, ist es für IT-Abteilungen deutlich einfacher, die kritischen Daten richtig zu schützen.

Die Einteilung der Daten nach der Erscheinungsform - wie Ton, Bild, Text - wurde in den Abschnitten davor behandelt. Nachfolgend sind die weiteren Einteilungsformen etwas konkretisiert. Bei der Einteilung nach der Repräsentation spielen im IT-Bereich nur die digitalen Daten eine Rolle.



Einteilung nach Repräsentation	
Analoge Daten	Analoge Daten werden durch kontinuierliche Funktionen dargestellt, d.h. die Funktion kann im Prinzip jeden beliebigen Zwischenwert annehmen. Ein Beispiel für die analoge Ausgabe von Daten sind Zeigerinstrumente, z. B. Tachometer in Autos oder Zeigeruhren. Mit analogen Daten können Computer nicht ohne Weiteres umgehen. Dazu müssen die Daten erst digitalisiert werden.
Digitale Daten	Digitale Daten sind all jene Daten, die aus einer Folge von diskreten Werten bestehen und sich damit maschinell verarbeiten lassen. Digitale Daten können nur endlich viele Werte annehmen. So können die Zeiger einer Analoguhr beliebig viele Stellungen auch zwischen den einzelnen Stunden, Minuten und Sekunden einnehmen, während in einer Digitaluhr nur diskrete Werte für die Stunde, Minute und Sekunde angezeigt werden können.

Eine weitere wichtige Einteilung ist die Einteilung nach der Struktur. Diese ist für die Verarbeitung und Speicherung von Daten von großer Bedeutung. So lassen sich unstrukturierte Daten schwer verarbeiten, stellen aber in einigen Unternehmen die Hauptmasse der Daten dar.

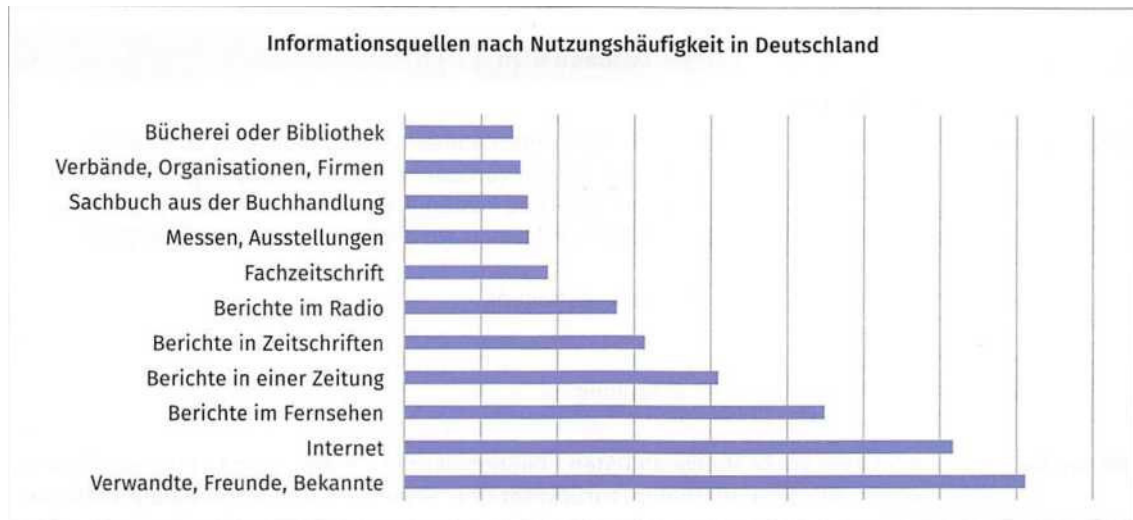
Einteilung nach Struktur	
Unstrukturierte Daten	Wie der Name es schon vermuten lässt, ist bei unstrukturierten Daten keine einheitliche Struktur zu bestimmen. Dadurch ist es schwer diese Daten zu analysieren und zu verarbeiten. Daraus ergibt sich, dass die Nutzbarkeit von unstrukturierten Daten mithilfe von Software und automatisierten Prozessen sehr eingeschränkt ist. Beispiele für unstrukturierte Daten sind Textdateien, Präsentationen, Videos, Audiodaten, aufgezeichnete Sprache oder Bilder und ähnliches.
Semistrukturierte Daten	Gegenüber unstrukturierten Daten weisen semistrukturierte Daten eine gewisse Grundstruktur auf. Als gutes Beispiel für semistrukturierte Daten gilt die E-Mail. Sie besitzt mit Absender, Empfänger und Betreff sowie weiteren Informationen des Nachrichtenkopfes eine Grundstruktur. Der eigentliche Inhalt der Mail besteht jedoch aus Text, welcher wiederum strukturlos ist.
Strukturierte Daten	Strukturierte Daten haben ein vorgegebenes Format, in das sich alle Daten gezielt einordnen lassen. Dadurch können diese Daten auch leicht maschinell verarbeitet werden.

Auch die Sicherheitsstufen, welche die Daten verlangen, sind von immenser Bedeutung. Besonders betrifft dieses den Bereich des Datenschutzes und der Datensicherheit.

Einteilung nach Sicherheitsstufen	
Öffentliche Daten	Öffentliche Daten sind für jedermann zugänglich. Dabei handelt es sich beispielsweise um Informationen (Daten), die auf einer öffentlichen Webseite im Internet dargeboten werden.
Interne Daten	Interne Daten werden lediglich den Mitarbeitern des Unternehmens zugänglich gemacht. Dies können beispielsweise Telefonverzeichnisse, Weisungen oder allgemeine Strategiedokumente sein. Eine Veröffentlichung dieser Daten könnte eventuell zu Nachteilen für das Unternehmen führen.
Vertrauliche Daten	Hierbei handelt es sich in der Regel um Informationen (Daten), die sehr wichtig für das Unternehmen sind, z. B. Gehaltslisten. Die Herausgabe dieser Daten ist teilweise sogar gesetzlich geregelt und ein Verstoß gegen die Vorschriften würde juristische Folgen nach sich ziehen. Deswegen sind vertrauliche Daten nur einer begrenzten Anzahl an Mitarbeiter zugänglich.
Geheime Daten	Die höchste Sicherheitsstufe weisen geheime Daten auf. Solche Daten sind unmittelbar für den Bestand des Unternehmens von Bedeutung, z. B. die Kundeninformationen eines Nummernkontos einer Bank. Deswegen hat zu diesen Daten nur ein sehr kleiner Personenkreis Zugang.

LE 2 - Datenherkunft

Zur Beantwortung der Frage, woher Daten stammen bzw. Informationen bezogen werden, kann allgemein die nachfolgende Statistik (Stand 2019) herangezogen werden.



Diese Statistik zeigt das Ergebnis einer Umfrage in Deutschland zu Informationsquellen, welche von der Bevölkerung hauptsächlich genutzt werden. Dabei ergab sich, dass rund 70% der Bevölkerung das Internet nutzen, wenn sie sich über etwas informieren wollen. Am meisten werden aber mit über 80% Verwandte und Freunde als Informationsquelle genannt. Die Bezugsquellen von Unternehmensdaten sind ähnlich gelagert, auch wenn Freunde und Bekannte hier weniger eine Rolle spielen, sondern eher die Kunden, andere Unternehmen und die Mitarbeiter des eigenen Unternehmens. Digitale Kommunikationsmittel erlangen hierbei eine immer größere Bedeutung.



Die für die Informationsbeschaffung potenziell nutzbaren Informationsquellen kann man in unternehmensinterne und unternehmensexterne Quellen unterscheiden. Hierbei sollten zuerst die unternehmensinternen Quellen genutzt werden. Der Zugang ist in der Regel problemloser, schneller und preiswerter. Beispiele dafür sind Auftrags- und Bestellstatistiken, Kundendienst- oder Außendienstberichte. Gute unternehmensexterne Daten sind meist sehr teuer, aufwendig zu beschaffen und müssen sorgfältig geprüft werden.

Eine weitere Unterscheidung der Datenquellen ist die Unterscheidung in Primärdaten und Sekundärdaten. Auch hier bestimmt die Aufgabe, auf welche Datenquelle man zugreifen sollte, da sich Primär- und Sekundärdaten sowohl in Verfügbarkeit, Kosten, Qualität und Zuverlässigkeit unterscheiden.

Datenquellen	
	Beschreibung
Primärdaten	Primärdaten sind Daten, die vom Unternehmen eigenständig erhoben werden. Meistens sind diese Daten noch nicht vorhanden und müssen erst gesammelt werden. Dieses ist teilweise sehr zeit- und kostenintensiv. Dafür sind die Daten dann in aller Regel auch sehr aktuell und genau auf die Problemstellung bezogen. Quellen - Mündliche oder schriftliche Befragung - Interview - Fragebogen - Computergestützte Befragung - Beobachtung
Sekundärdaten	Wird eine große Menge an Daten benötigt, wird in der Regel auf bereits verfügbare Daten, sog. Sekundärdaten, zurückgegriffen. Sekundärdaten wurden bereits früher für ähnliche oder andere Zwecke erhoben. Sie haben den Vorteil, schnell verfügbar zu sein und sind oft kostengünstiger und von einer Qualität, die eine eigene Analyse nicht erreichen kann. Häufig sind diese Daten jedoch nicht aktuell oder für das Problem nur bedingt geeignet. Auch die Zuverlässigkeit der Daten ist nicht immer einfach zu beurteilen. Quellen - Unternehmensinterne Datenquellen - Datenbanken - Vertriebserfolgsanalyse - Berichte der Außendienstmitarbeiter und des Kundendienstes, Kundenkorrespondenz - Archive - Unternehmensexterne Datenquellen - Statistisches Bundesamt - Statistische Landesämter - Ministerien und sonstige staatliche Institutionen - Wirtschaftsverbände - Branchenberichte und -Statistiken, Betriebsquellen vergleiche - Aufbereitung von Daten amtlicher und nicht amtlicher Quellen - Informationen von wissenschaftlichen Instituten - Bibliografien - Fachliteratur - Zeitungen und Zeitschriften - Informationen von internationalen Organisationen - Statistische Ämter der Europäischen Union und anderer supranationaler Organisationen

Lernzielkontrollen Tag 4

1. Welche Aussagen sind richtig?

- a) Die Verarbeitung von analogen Daten spielt in der IT eine Hauptrolle.
- b) Unstrukturierte Daten lassen sich leichter verarbeiten als strukturierte Daten.
- c) Vertrauliche Daten sind nur bestimmten Mitarbeitern des Unternehmens zugänglich.
- d) Öffentliche Daten sind nur für den internen Gebrauch bestimmt.
- e) Freunde und Bekannte sind im privaten Bereich die Hauptinformationsquelle für die Menschen.
- f) Im Unternehmensbereich unterscheidet man unternehmensinterne und -externe Daten.
- g) Primärdaten werden von Unternehmen selbst erhoben.
- h) Sekundärdaten können aus Fachliteratur oder über statistische Ämter bezogen werden.
- i) Sekundärdaten sind immer auf dem aktuellsten Stand.

2. Werten Sie einen englischen Text über quantitative und qualitative Daten aus.

Daten werden nach verschiedenen Merkmalen unterschieden. Eine Form ist die Unterscheidung in quantitative und qualitative Daten. Lesen Sie sich den folgenden englischen Text durch und beantworten Sie im Anschluss die Fragen.

Quantitative data are measures of values or counts and are expressed as numbers. Quantitative data are data about numeric variables (e.g. how many, how much, how often). Qualitative data are measures of "types" and may be represented by a name, symbol, or a number code. Qualitative data are data about categorical variables (e.g. what type). Quantitative and qualitative data provide different outcomes and are often used together to get a full picture of a facts. Quantitative and qualitative data can be gathered from the same data unit depending on whether the variable of interest is numerical or categorical.				
Data unit	Numeric variable	Quantitative data	Categorical variable	Qualitative data
A person	How many children do you have?	2	In which country do you live?	Germany
	How much do you earn?	50000,00 € p.a.	What is your job?	IT-Professional
	How many hours per week do you work?	40	Do you work full-time or part-time?	Full-time

- a) Was verstehen Sie unter quantitativen Daten?
- b) Was verstehen Sie unter qualitativen Daten?
- c) Warum ist es von Vorteil sowohl quantitative als auch qualitative Daten zu erheben?
- d) Nennen Sie jeweils zwei Beispiele für quantitative und qualitative Daten einer Person.

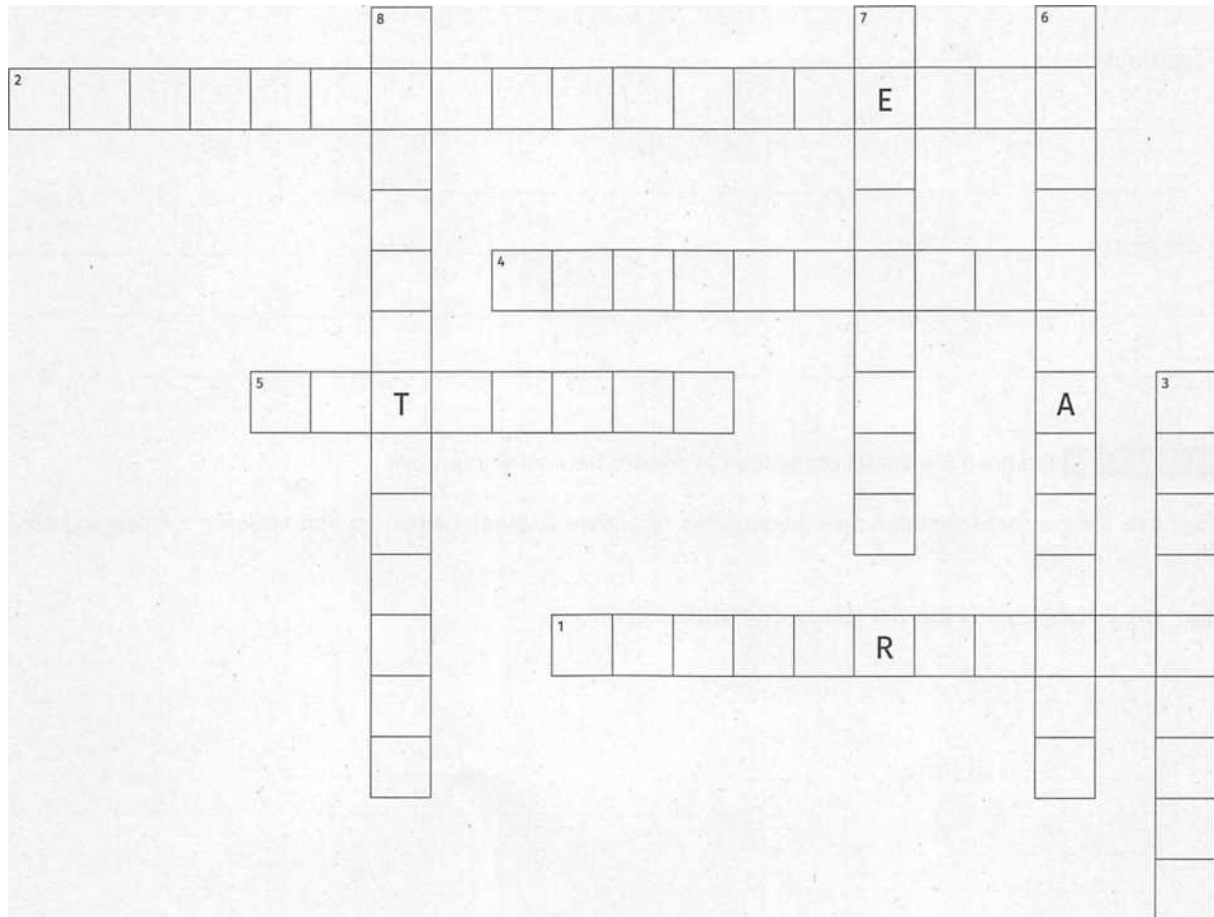
3. Überprüfen Sie Ihr Wissen über die Art und die Herkunft von Daten.

Lösen Sie folgendes Kreuzworträtsel.

- 1. Daten, welche noch nicht vorhanden sind und vom Unternehmen eigenständig erhoben werden.
- 2. Abstrakter Begriff für Personen, Orte, Prozesse, Gegenstände ect., von denen man Auskunft, Wissen oder Erläuterungen erhält.
- 3. System, mit dessen Hilfe große Mengen an Daten digital gespeichert und verwaltet werden.
- 4. Daten, welche die Grundinformationen über betrieblich relevante Objekte wie Produkte,

Kunden und Mitarbeiter enthalten.

5. Ist heutzutage für die meisten Menschen eine der wichtigsten Informationsquellen.
6. Daten, welche vorliegen und schon für ähnliche oder andere Zwecke erhoben wurden.
7. Methode, um Primärdaten zu erheben.
8. Datenquelle für Sekundärdaten.



4. Unterscheiden Sie Primär- und Sekundärdaten und nennen Sie jeweils drei Beispiele für entsprechende Quellen.

Tag 5 - Aspekte der Speicherung von Daten präsentieren

Heute hängt die Wettbewerbsfähigkeit vieler Unternehmen davon ab, wie schnell Daten verarbeitet und analysiert werden können. Dabei spielen die Speicherung von Daten und die damit verbundenen Aspekte eine große Rolle. Um Daten bestmöglich zu nutzen, ist eine effiziente Storage-Strategie notwendig, die konsequent an den individuellen Anforderungen der Unternehmen ausgerichtet ist. Grundlage für ein tragfähiges Speicherkonzept ist die genaue Analyse der Daten.

LE 1 - Verfügbarkeit von Daten, Datenschutz und Datensicherheit

Datenschutz, Datensicherheit und Informationssicherheit spielen eine große Rolle bei der Speicherung und Verwaltung von Daten. An dieser Stelle soll es nur einen generellen Überblick über die Begriffe geben. Für tiefgehende Betrachtungen in diesem Bereich wird auf das Lernfeld 4 verwiesen.

Begriffserläuterungen	
Datenschutz	Bei dem Datenschutz geht es um den Schutz der Privatsphäre eines Menschen. Datenschutz garantiert allen Bürgerinnen und Bürgern ein Recht auf informationelle Selbstbestimmung und schützt sie vor missbräuchlicher Verwendung ihrer Daten. Für die Verarbeitung personenbezogener Daten gibt es Regeln, die hauptsächlich im BDSG bzw. den Datenschutzgesetzen der Länder niedergelegt sind. Hier wird also danach gefragt, ob personenbezogene Daten überhaupt verarbeitet werden dürfen.
Datensicherheit	Im Unterschied zum Datenschutz befasst sich die Datensicherheit mit dem Schutz von Daten, unabhängig davon, ob diese einen Personenbezug aufweisen oder nicht. Unter den Begriff Datensicherheit fallen daher grundsätzlich auch Daten, die keinen Personenbezug haben. Datensicherheit soll Sicherheitsrisiken begegnen und die Daten vor Manipulation, Verlust oder unberechtigter Kenntnisnahme schützen. Hier geht es also nicht um die Frage, ob Daten überhaupt erhoben und verarbeitet werden, sondern um die Frage, welche Maßnahmen zur Sicherheit der Daten ergriffen werden müssen.
Informationssicherheit	Informationssicherheit hat den Schutz von Informationen als Ziel. Hier ist es unerheblich, ob es sich um digitale oder analoge Informationen handelt oder ob diese einen Personenbezug haben. Teilweise wird die Datensicherheit als ein Teil der Informationssicherheit angesehen, da Letzteres umfassender ist.

Informationssicherheit		
Vertraulichkeit	Integrität	Verfügbarkeit
Informationen dürfen nicht in die falschen Hände gelangen.	Sensible Informationen dürfen nicht verfälscht werden.	Notwendige Informationen müssen abrufbar sein.

Einige Aspekte der Informationssicherheit sollen an dieser Stelle nochmals etwas genauer betrachtet werden.

Schutzziele	
Verfügbarkeit	Der Zugriff auf Daten muss innerhalb eines vereinbarten Zeitrahmens gewährleistet sein. Verfügbarkeit bedeutet, dass Daten und IT-Systeme zur Verfügung stehen und von autorisierten Personen genutzt werden können, wenn dies benötigt wird. Eine 99-prozentige Datenverfügbarkeit klingt zwar gut, ist aber im kommerziellen Einsatz nicht ausreichend. Die Datenverfügbarkeit kann durch geeignete Maßnahmen wie RAID-Systeme oder Cluster-Rechner deutlich verbessert werden. Es wird eine Verfügbarkeit von 99,999% oder besser angestrebt.
Integrität	Integrität bedeutet, dass die Daten über einen bestimmten Zeitraum vollständig und unverändert sind. Sie umfasst Datensicherheit (Schutz vor Verlust) und Fälschungssicherheit (Schutz vor vorsätzlicher Veränderung). Die Integrität ist gewährleistet, wenn die Daten vom angegebenen Absender stammen und vollständig sowie unverändert an den Empfänger übertragen worden sind.
Vertraulichkeit	Vertraulichkeit bedeutet, dass nur bestimmte Personen von der Existenz der Daten oder dem Inhalt der Daten Kenntnis haben. Bei der Vertraulichkeit müssen Sicherheitsmaßnahmen erhoben werden, damit kein Unbefugter Zugriff auf gespeicherte sowie auch nicht auf übermittelte Daten bekommt.

LE 2 - Speicheranforderungen

Die Speicheranforderungen ergeben sich zum einen aus der Menge und zum anderen aus der Art der zu speichernden Daten. Dieses hat Auswirkungen auf die Größe und Art des Speichermediums und die Speicherkonzepte, z.B. was die Verfügbarkeit von Daten betrifft. Durch die Datenmenge wird bestimmt, wie viele Daten zu speichern sind. Dabei ist zu beachten, dass die Datenmenge nicht mit der Informationsmenge, welche übertragen werden soll, gleichzusetzen ist. Der Informationsgehalt von Daten kann nicht aus der Datenmenge bestimmt werden. Die Datenmenge, die benötigt wird, um eine Information zu speichern, hängt einerseits von der Komplexität der Information und andererseits vom Verfahren ab, wie diese Information binär abgebildet wurde.

Datenmenge

Die Datenmenge ist ein Maß für die Menge von Daten. Die Grundeinheit der Datenmenge ist das Bit. Durch die Datenmenge kann aber nicht der Informationsgehalt der Daten bestimmt werden.

Die Datenmenge wird durch die Anzahl der digitalen Dateneinheiten bestimmt. Die kleinste darstellbare binäre Dateneinheit ist das Bit. Ein Bit kann auf einem Speicherplatz nur zwei Zustände (0,1) darstellen. Wenn einzelne Bit aneinandergereiht werden, dann spricht man von einer Bit-Folge. Bit-Folgen mit einer bestimmten Anzahl an Bit haben wiederum eigene Namen, welche weitere Dateneinheiten bezeichnen. Eine Bit-Folge von 8 Bit bezeichnet man typischerweise als Byte. Weitere grundlegende Dateneinheiten sind in der folgenden Tabelle aufgeführt. Die Bezeichnungen und die Anzahl der Bit hängen teilweise aber sehr stark von dem benutzten System ab.

Bit

Ein Bit ist die kleinste darstellbare binäre Dateneinheit. Bit ist die Kurzform für „binary digit“ (Binärziffer).

Grundlegende binäre Dateneinheiten	
Bit	Kleinste Einheit; kann nur zwei Zustände annehmen
Byte	Bit-Folge aus 8 Bit; neben dem Bit die gebräuchlichste Dateneinheit
Halb-Byte oder Nibbel	Bit-Folge aus 4 Bit oder ein halbes Byte
Word	Entspricht auf den meisten Systemen einer Länge von 2 Byte oder 16 Bit
Double Word	Entspricht auf den meisten Systemen einer Länge von 4 Byte oder 32 Bit

Um den Umgang mit großen Zahlen zu erleichtern, existieren im gewohnten metrischen System sog. SI- Präfixe, wie Kilo, Mega, Zenti, Milli usw. Zum Beispiel kann die Länge in Meter, aber auch in Zentimeter oder Kilometer angegeben werden. Alle SI-Präfixe des metrischen Systems sind jeweils ein Vielfaches oder ein Teiler von 10 hoch 3 (1 000). Diese Präfixe können und werden auch für die Maßeinheiten Bit und Byte verwendet. Heutzutage allerdings nur noch in Verbindung mit der dezimalen Angabe von Speichergrößen. Daraus ergibt sich aber ein Problem, weil den Dateneinheiten Bit und Byte das binäre Zahlensystem zugrunde liegt und nicht das dezimale. Um in diesem Bereich Verwechslungen zu vermeiden, wurden 1996 neue Einheitenvorsätze entwickelt, die nur in der binären Bedeutung verwendet werden. Dabei wird eine den SI-Präfixen ähnlich lautende Vorsilbe ergänzt um die Silbe „bi“, die klarstellt, dass es sich um binäre Vielfache handelt, z. B. Kibi, Mebi, Gibi usw.

Dezimalpräfixe	Binärpräfixe laut IEC	Beispiele
Kilobyte (KB) 10^3 Byte = 1 000 Byte	Kibibyte (KiB) 2^{10} Byte = 1 024 Byte	ca. 1 bis 2 KB: eine Normseite als Text (ANSI/ASCII- kodiert)
Megabyte (MB) 10^6 Byte = 1 000 000 Byte	Mebibyte (MiB) 2^{20} Byte = 1 048 576 Byte	1 MB bis ca. 500 Textseiten
Gigabyte (GB) 10^9 Byte = 1 000 000 000 Byte	Gibibyte (GiB) 2^{30} Byte = 1 073 741 824 Byte	128 GB Speicherplatz auf modernem Smartphone; ca. 250 Filme oder 500 Stunden HD-Videos
Terabyte (TB) 10^{12} Byte = 1 000 000 000 000 Byte	Tebibyte (TiB) 2^{40} Byte = 1 099 511 627 776 Byte	1 TB Speicherplatz von ungefähr acht Smartphones
Petabyte (PB) 10^{15} Byte = 1 000 000 000 000 000 Byte	Pebibyte (PiB) 2^{50} Byte = 1 125 899 906 842 624 Byte	Für YouTube werden 1,5 Petabyte Speicherplatz pro Tag benötigt
Exabyte (EB) 10^{18} Byte = 1 000 000 TB	Exbibyte (EiB) 2^{60} Byte = 1 152 921 504 606 846 976 Byte	Größenordnung des „menschlichen Wissens“, beträgt heutzutage geschätzt mehrere Hundert Exabyte
Zettabyte (ZB) 10^{21} Byte = 1 000 000 000 TB	Zebibyte (ZiB) 2^{70} Byte = 1 180 591 620 717 411 303 424 Byte	Geschätzte weltweite Datenmenge bis ins Jahr 2020 44 Zettabyte.

Dezimalpräfixe	Binärpräfixe laut IEC	Beispiele
Yottabyte (YB) 10^{24} Byte = 1 000 000 000 000 TB	Yobibyte (YiB) 2^{80} Byte = 1 208 925 819 614 629 174 706 176 Byte	100 Milliarden Festplatten zu 10 Terabyte

Beispielaufgabe für die Berechnung von Speicherbedarf

Es soll der Speicherbedarf für 500 Bilder in GiB berechnet werden. Gegeben sind folgende Daten pro Bild: Höhe: 1 500 Pixel, Breite: 2 000 Pixel, Farbkanäle pro Pixel: 3 (RGB), Farbtiefe = 8 Bit pro Farbkanal = 1 Byte.

Der Speicherbedarf für ein Bild in Byte berechnet sich aus der Anzahl der Pixel pro Bild und dem Speicherbedarf für einen Pixel.

Lösung:

Pixelanzahl pro Bild	= 1 500 • 2 000 Pixel = 3 000 000 Pixel/Bild
Byte pro Pixel	= (Anzahl Farbkanäle • Bit pro Farbkanal) ÷ 8 = (3 • 8 Bit/Pixel) ÷ 8 Bit/Byte = 3 Byte/Pixel
Speicherbedarf pro Bild	= 3 000 000 Pixel/Bild • 3 Byte/Pixel = 9 000 000 Byte/Bild
Speicherbedarf gesamt	= 9 000 000 Byte/Bild • 500 Bilder = 4 500 000 000 Byte = 4 500 000 000 Byte ÷ 1 024 = 4 394 531,25 KiB ÷ 1024 = 4 291,53 MiB ÷ 1024 = 4,19 GiB

Weitere Speicheranforderungen ergeben sich aus den Anforderungen an die Verfügbarkeit der Daten, dem Datenschutz und der Datensicherheit.

LE 3 - Speicherlösungen

Die Grundlage für die Auswahl einer Speicherlösung sind die Speicheranforderungen. Große Datenmengen im Big-Data-Bereich erfordern andere Speicherlösungen als die Speicherung und Verwaltung einiger Worddokumente. Im Zusammenhang mit Speicherlösungen spricht man auch von Storage (Lagerung oder Speicherung). Man kann Speicherlösungen danach gestalten, wo z. B. die Daten gespeichert werden, welche Speichermedien zum Einsatz kommen oder auch in welcher Form die Daten abgelegt werden. Bei der Form hat, neben der klassischen Variante des File-Storage und dem etwas neueren Block-Storage, z.B. das Object-Storage viel Beachtung erhalten. Bei den Speichermedien spielen u.a. Flashlösungen eine Rolle. Für weitergehende Informationen dazu siehe auch Lernfelder 2 und 3. Einige verschiedene Speicherlösungen aus allen Rubriken sollen hier kurz vorgestellt werden.

Mögliche Speicherlösungen	
Cloud-Storage	Beim Cloud-Storage werden die Daten an einem Cloud-Standort gepflegt, verwaltet, gesichert und Benutzern über ein Netzwerk (meist das Internet) zur Verfügung gestellt. Es werden drei grundlegende Varianten unterschieden: öffentliche Dienste für Cloud-Storage, private Cloud-Storage-Dienste, hybrides Cloud-Storage.
Full-Flash-Storage	Flash-Speicherlösungen bieten einen deutlich schnelleren Datenzugriff als klassische Plattensysteme, erleichtern die Datenverwaltung und bringen spürbare Kostenvorteile.
All-Flash- und Hybrid-Flash-Storage	Moderne All-Flash-Speicher bieten Funktionen für Hochverfügbarkeit und Disaster-Recovery und sind eine kostengünstige Speicherlösung.
File-Storage	Ein File-Storage speichert die Daten in Form von Dateien. Diese Dateien sind in einer hierarchischen Struktur in Verzeichnissen abgelegt. Verzeichnisse können ein oder mehrere Unterverzeichnisse haben. Im sogenannten File-System eines Speichermediums sind die Pfadinformationen gespeichert, über die der Rechner auf die Dateien zugreift.
Block-Storage	Beim Block-Storage werden Daten in Blöcke aufgeteilt und separat gespeichert. Jeder dieser Blöcke erhält eine eindeutige Kennzeichnung, was es dem Storage-System ermöglicht, kleinere Datenmengen dort abzulegen, wo es am praktischsten ist.
Object-Storage	Während File-Storage bei der Speicherung von Daten auf komplette Dateien setzt und Block-Storage Dateien in gleich große Blöcke aufteilt, werden Daten im Object-Storage als Objekte gesichert. Hier werden neben den eigentlichen Daten noch zahlreiche Metadaten gespeichert.

Lernzielkontrollen Tag 5

1. Welche Aussagen sind richtig?

- a) Datenschutz und Datensicherheit spielen eine große Rolle bei der Verwaltung von Daten.
- b) Der Datenschutz bezieht sich auf den Umgang mit personenbezogenen Daten.
- c) Eine Datenverfügbarkeit von 90% ist heutzutage ausreichend.
- d) Ein Bit besteht aus 8 Byte.
- e) Ein Halb-Byte besteht 12 Bit.
- f) Ein DWORD besteht aus 32 Bit.
- g) GB und GiB sind unterschiedliche Dateneinheiten.
- h) Die Umrechnungszahl von TiB in MiB ist 1 000000.
- i) Datenmengen im Bereich von Zettabyte werden erst in einigen Jahrzehnten erreicht werden.
- j) Flash-Speicherlösungen sind in der Regel sehr langsam.
- k) File-Storage ist eine klassische Speicherlösung von Daten.

2. Es sollen von 1 000 Kunden die Bestellungen über einen Zeitraum von 20 Jahren gespeichert werden. Gegeben sind folgende Fakten:

- Ein Kunde gibt pro Monat im Schnitt fünf Bestellungen auf.
- Daten pro Bestellung: ID (4 Byte), Datum (4 Byte), Artikelname (30 Byte), Menge (8 Byte), Kundenname (30 Byte), Bemerkungen (100 Byte)

Berechnen Sie den Speicherbedarf in MB und in MiB.

3. Überprüfen Sie Ihr Wissen über die Schutzziele bei der Speicherung von Daten. Erläutern Sie folgende Begriffe im Zusammenhang mit Daten:

- a. Verfügbarkeit:
- b. Vertraulichkeit:
- c. Integrität:

4. Rechnen Sie die nachfolgenden Speicherangaben in andere Dateneinheiten um und notieren Sie den vollständigen Rechenweg.

- a. Wie viel MByte und wie viel MiB sind 10000000 Byte?
- b. Wie viel Bit entsprechen 5 KiB?
- c. Wie viel TiB sind 79 234 MiB?
- d. Wie viel TB sind 15 GiB?
- e. Wie viel MB sind 14 PB?
- f. Wie viel Byte sind 4 TiB?

5. Lösen Sie folgende Aufgaben und geben Sie jeweils den vollständigen Rechenweg mit den entsprechenden Einheiten an.

- a. Es soll der Speicherbedarf für 1 000 Rastergrafikbilder berechnet werden. Gegeben sind pro Bild die folgenden Daten:

Abmaße des Bildes	Höhe: 17,78 cm; Breite: 33,02 cm (1 Inch = 2,54 cm)
Auflösung	300 ppi (Pixel per Inch)
Farbmodell	RGB (drei Kanäle pro Pixel)
Farbtiefe	Deep Colour (12 Bit pro Farbkanal)

Berechnen Sie den Speicherbedarf in TiB.

- b. Es soll die benötigte Speicherkapazität eines Streaming-Servers ermittelt werden. Dazu liegen Ihnen die folgenden Angaben für ein Video vor:

Auflösung in Pixel	1024 • 768	Bilderje Sekunde	30
Farbtiefe	12 Bit/Pixel	Videodauer	15 min

Der Komprimierungsfaktor für ein einzelnes Video beträgt 1 : 100. Das bedeutet, dass die Dateigröße nach der Kompression einhundertstel der vorherigen Dateigröße beträgt. Hinweis: Für die Aufgabe wird die Speicherung der Tonspuren vernachlässigt. Die Speicherkapazität des Streaming-Servers soll für insgesamt 5 000 Videos von identischer Größe ausgelegt sein. Ermitteln Sie die erforderliche Gesamtspeicherkapazität des Streaming-Servers in TiB.

- c. Es sollen Versicherungsdaten von 200000 Versicherungsnehmern in einer Datenbank gespeichert werden. Ein Versicherungsnehmer besitzt im Schnitt zwei Versicherungen.

Daten pro Versicherungsnehmer und Versicherung					
Versicherungsnehmer			Versicherung		
Daten	Datentyp	Byte	Daten	Datentyp	Byte
ID	Ganzzahl		ID	Ganzzahl	
Name	Text[30]		Bezeichnung	Text[30]	
Vorname	Text[30]		ID_Versicherungsnehmer	Ganzzahl	
Straße	Text[50]		Abschlussdatum	Ganzzahl	
PLZ	Text(5]		Laufzeit	Ganzzahl	
Ort	Text[50]		Betrag pro Monat	Fließkommazahl	

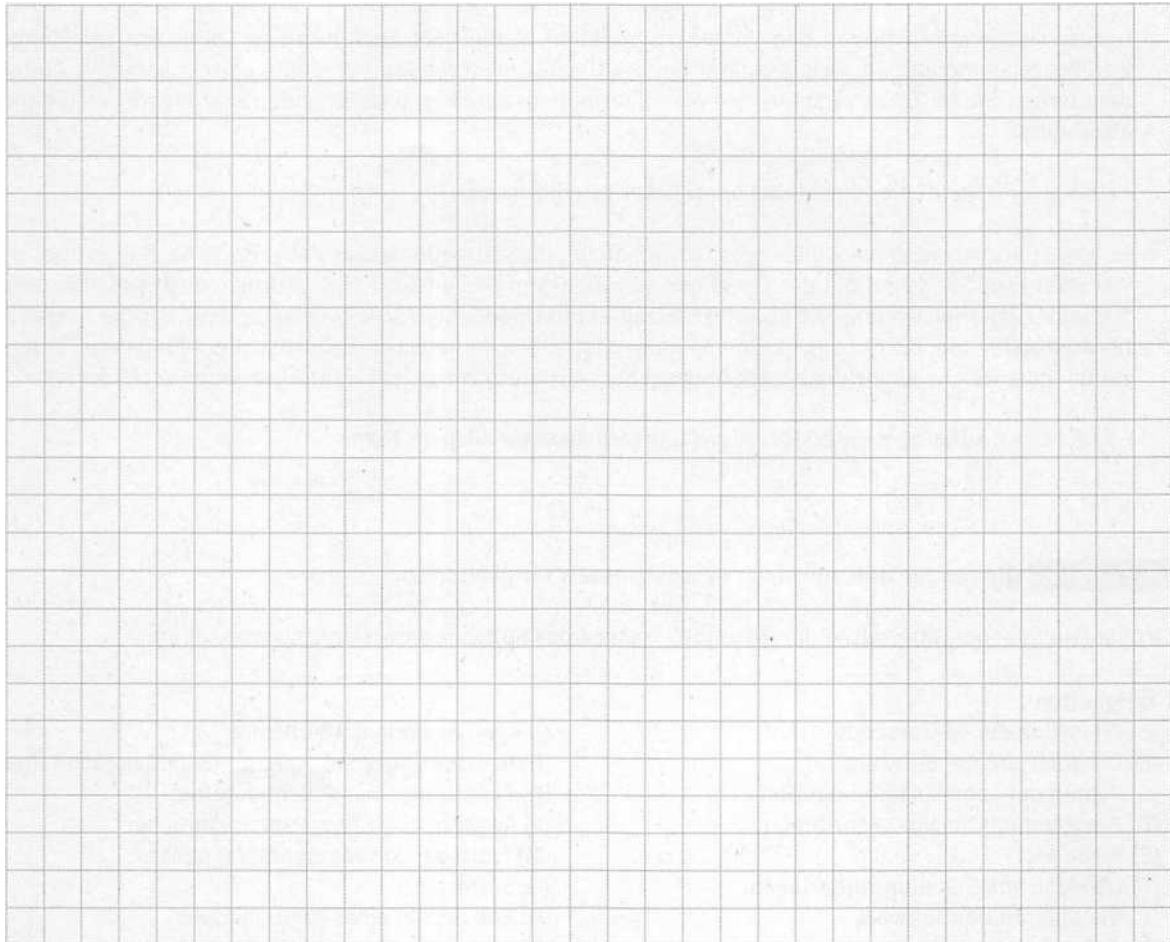
Tel	Text[20]	
Bemerkungen	Text[250]	

Ermäßigung	Fließkommazahl	
Bemerkungen	Text[250]	

d. Es wird folgender Speicherbedarf angenommen:

Ganzzahl: 4 Byte
Fließkommazahl: 8 Byte
Text - pro Zeichen: 2 Byte (Beispiel: Text[50] = 100 Byte)

Berechnen Sie die Speichieranforderungen für die Versicherungsdaten in GiB, wenn noch eine Speicherreserve von 20% auf das berechnete Ergebnis aufgeschlagen werden soll.



Tag 6 - Den Prozess der Softwareentwicklung analysieren

Sie sollen in einem Projekt mitarbeiten, in dem eine Anwendungssoftware zur Verwaltung von Daten angepasst wird. Dazu erarbeiten Sie sich zunächst einmal grundlegendes Wissen über den Prozess der Softwareentwicklung im Allgemeinen. Außerdem wollen Sie Kenntnisse darüber erwerben, welche Modellierungs- und Programmiersprachen und welche Entwicklungswerkzeuge in diesem Prozess zum Einsatz kommen.



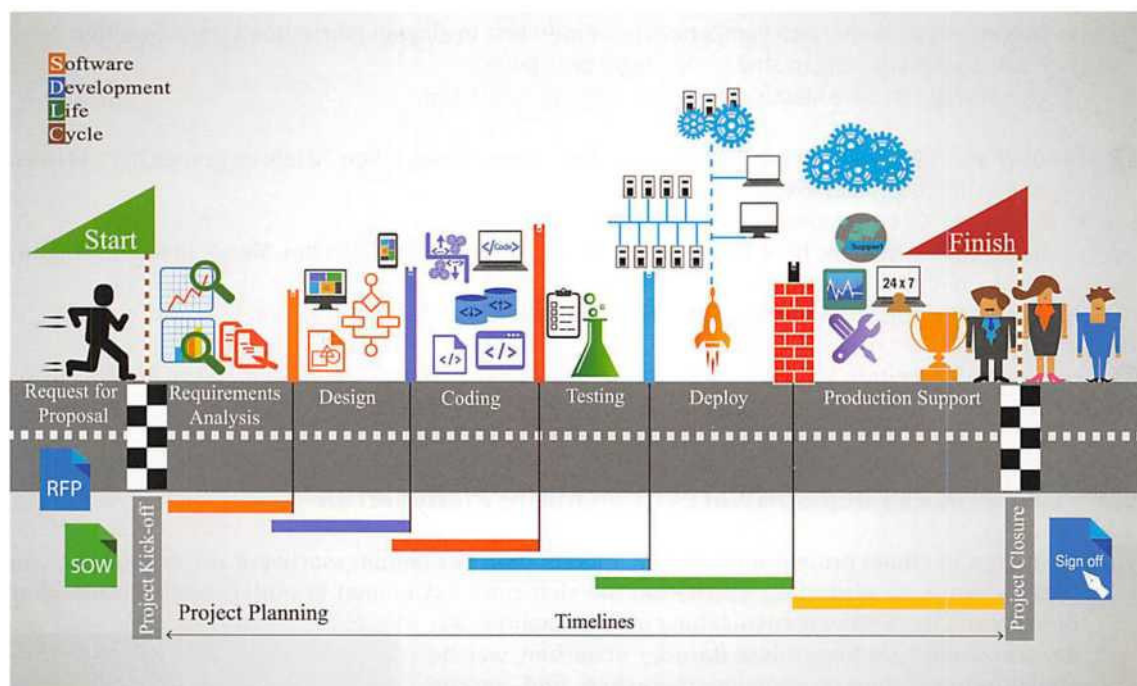
In der Regel wird angenommen, dass die Entwicklung von Software ein Prozess ist, der einen klar definierten Anfang und ein klar definiertes Ende hat. Das ist heutzutage aber selten noch der Fall. Der Softwareentwicklungsprozess ist wesentlich komplexer geworden und erstreckt sich meistens über einen langen Zeitraum. Dabei werden die einzelnen Phasen dieses Prozesses in der Regel nicht mehr geradlinig durchlaufen, sondern eher spiralförmig und die Entwicklung erfolgt größtenteils in Form von Softwareprojekten. Dabei kommen die unterschiedlichsten Programmiersprachen und Softwareentwicklungswerkzeuge zum Einsatz. Alle diese Themen werden in diesem Kapitel näher betrachtet.

LE 1 - Die Prozessphasen beschreiben

Egal, wie komplex oder übersichtlich die zu entwickelnde Software auch sein mag, der Prozess der Entwicklung enthält immer mehrere grundlegende Phasen. Grundlage für den Softwareentwicklungsprozess bildet der Softwarelebenszyklus. Dieser beschreibt insgesamt die Phasen, welche ein Softwareprodukt bei seiner Herstellung und dem späteren Einsatz durchläuft. Die einzelnen Entwicklungsphasen können auf verschiedene Art und Weise, zu unterschiedlichen Zeitpunkten und in unterschiedlicher Ausprägung durchlaufen werden.

Prozess

Unter einem Prozess versteht man die Gesamtheit aller aufeinander einwirkenden Vorgänge innerhalb eines Systems.



Mögliche und bewährte Varianten werden in den sog. Vorgehensmodellen beschrieben. Grundsätzlich steht am Anfang einer jeden Neuentwicklung oder Anpassung von Software eine Idee, welche ein Problem lösen soll. Allerdings sind von der Idee bis zum fertigen Softwareprodukt viele Überlegungen und Handlungsschritte notwendig, welche man einzelnen Prozessphasen zuordnen kann.

Phasen des Softwareentwicklungsprozesses nach IEEE-Standard 12207

Anforderungsanalyse und Erstellung einer Spezifikation

In der Analysephase werden die Anforderungen an die zu entwickelnde Software festgelegt. Dabei ist es das Ziel, die konkreten Wünsche und Vorstellungen des Auftraggebers zu erfassen und festzuhalten. Wenn kein Kundenbezug vorhanden ist, wird präzise beschrieben, was die zukünftige Software leisten soll. In der Analysephase werden u.a. Ist-Zustände erfasst und Geschäftsprozesse und Daten, welche für zukünftige Software von Bedeutung sind, analysiert und beschrieben. Dabei kann beispielsweise ein Pflichtenheft entstehen, welches genau angibt, welche Anforderungen auf welche Art umgesetzt werden. Auch die Erstellung eines Lastenheftes kann zu einem frühen Zeitpunkt in der Analysephase erfolgen, wenn es nicht schon im Vorfeld geschehen ist. Außerdem wird eine Aufwandsabschätzung vorgenommen.

Design (Entwurf)

In der Designphase wird die Software mithilfe von Modellen entworfen. Es werden beispielsweise mithilfe der grafischen Modellierungssprache UML Klassendiagramme erstellt oder wichtige Algorithmen mithilfe von Struktogrammen beschrieben. Handelt es sich um eine Anwendung, welche Daten verwalten soll, dann sind die relevanten Daten abzugrenzen und das Datenmodell zu beschreiben. Auch eine Spezifizierung des Datenbanksystems ist notwendig. Wenn eine Einbindung in ein bestehendes System erfolgen soll, dann ist auch eine Planung der Schnittstellen von Bedarf. Außerdem sollte in dieser Phase auch schon das Testen der Anwendung geplant werden.

Umsetzung (Implementierung)

In der Umsetzungsphase erfolgt die eigentliche Programmierung der Software in der gewählten Programmiersprache. In der Endphase der Implementierung geht dieser Prozess fließend in die Testphase über.

Testen

In der Testphase wird überprüft, ob die entwickelte Software die vorgegebenen Anforderungen erfüllt. Dabei werden verschiedene Arten von Tests, z.B. Modul-, Integrations-, System- und Akzeptanztests, durchgeführt.

Dokumentation

In der Dokumentationsphase werden alle notwendigen Dokumente erstellt bzw. geordnet. Solche Dokumente sind das Lasten- und Pflichtenheft, Planungsmodelle und Testprotokolle. Es zählen aber auch Bedienungsanleitungen, Schulungsunterlagen u.a. dazu.

Auslieferung

In der Auslieferungsphase wird die Software dem Auftraggeber bzw. Anwender zur Verfügung gestellt und gegebenenfalls installiert und konfiguriert.

Wartung und Support

In der Wartungsphase werden Veränderungen an der Software vorgenommen, um nach deren Auslieferung Fehler zu beheben, die Effizienz zu verbessern oder Anpassungen an neue Anforderungen vorzunehmen.

LE 2 - Software im Rahmen eines Projektes entwickeln

Software wird in der Regel im Rahmen eines Software Projektes entwickelt. Dieses ist der Tatsache geschuldet, dass es sich bei Entwicklung und Anpassung von Software meistens um einmalige Vorhaben handelt, welche durch eine zeitliche Befristung und eine relative Neuartigkeit gekennzeichnet sind. Dieses entspricht der allgemeinen Definition von Projekten. Eine wichtige Rolle bei der Durchführung von Softwareprojekten spielt dabei das Projektmanagement. Egal, welchen Umfang ein Softwareprojekt hat, die Aufgaben des Projektmanagements bleiben im Prinzip die gleichen.

Projektmanagement

Unter Projektmanagement ist die Planung, die Überwachung, die Steuerung und der Abschluss eines Projekts zu verstehen. Zu den Aufgaben des Projektmanagements gehören u.a.:

- Projektziele abstimmen,
- organisatorische Gestaltung des Projektes,
- Planung der Projektleistung, der Termine, der Ressourcen und der Kosten,
- Aufgaben-, Kompetenz- und Verantwortungsverteilung,
- Risikoabschätzung,
- Teambildung,
- Rollen, Werte und Regeln festlegen,
- Personalmanagement, Mitarbeiterführung,
- Koordination und Kommunikation intern und extern,
- Dokumentation und Berichte,
- Projektsteuerung und Projektcontrolling: Termine, Kosten und Leistungsfortschritt überwachen, das Budget einhalten,
- Vorbereitung und Durchführung von Projektbesprechungen,
- Projekt abschließen.



Jedes Projekt ist immer ein Balanceakt zwischen Umfang, Zeit und Kosten. Die Änderung eines dieser Faktoren bewirkt unweigerlich auch Änderungen bei den anderen Faktoren. Ob Projekte erfolgreich sind oder scheitern hängt aber noch von vielen anderen Faktoren ab. Ein professionelles Projektmanagement ist das zentrale Erfolgskriterium von Projekten. Es gibt andererseits aber auch viele Gründe, warum Projekte scheitern können.

Faktoren für den Projekterfolg	
Erfolgsfaktoren	Risikofaktoren
• Genaue Definition von Projektzielen • Entwicklung präziser Projektpläne und Kontrolle von deren Einhaltung • Projektspezifische Gestaltung von Projektorganisation und Projektkultur • Zusammenstellen eines geeigneten Projektteams • Optimale Kommunikationsbedingungen	• Schlechte Kommunikation (z.B. Sprachbarrieren, unvereinbare Persönlichkeiten, schlechte Kommunikationskanäle) • Schlechte Planung (z. B. knappe Ressourcen, zu wenig Zeit) • Mangelnder Überblick über die Projektdetails • Verwendung der falschen Projektmanagementtools (z. B. Software nicht flexibel genug für das Projekt) • Mangelnde Überwachung und Kontrolle • Kein Risikomanagement • Änderungen im Umfang (z. B. Fristen werden verschoben, Aufgaben geändert)

Die Projektplanung entscheidet wesentlich darüber, ob das Softwareprojekt ein Erfolg wird. Wie ein Softwareprojekt geplant und durchgeführt wird, hängt wiederum stark vom ausgewählten Vorgehensmodell der Softwareentwicklung ab. Auf verschiedene Vorgehensmodelle soll im nächsten Abschnitt näher eingegangen werden.

LE 3 - Vorgehensmodelle unterscheiden

Sie sollen wichtige Vorgehensmodelle der Softwareentwicklung kennen und unterscheiden.

Der Prozess der Softwareentwicklung kann verschiedene Phasen umfassen, z. B. Analyse, Planung, Implementierung, Testen (siehe Kapitel 5.1.4). Je nach Art der Problemstellung und Dynamik der Anforderungen werden diese Phasen unterschiedlich ausgeprägt sein und notfalls häufiger durchlaufen werden. Abläufe, welche sich in der Praxis bewährt haben, werden in den sog. Vorgehensmodellen beschrieben. In der Regel bedient man sich eines oder mehrerer dieser Vorgehensmodelle, um eine Software zu entwickeln.

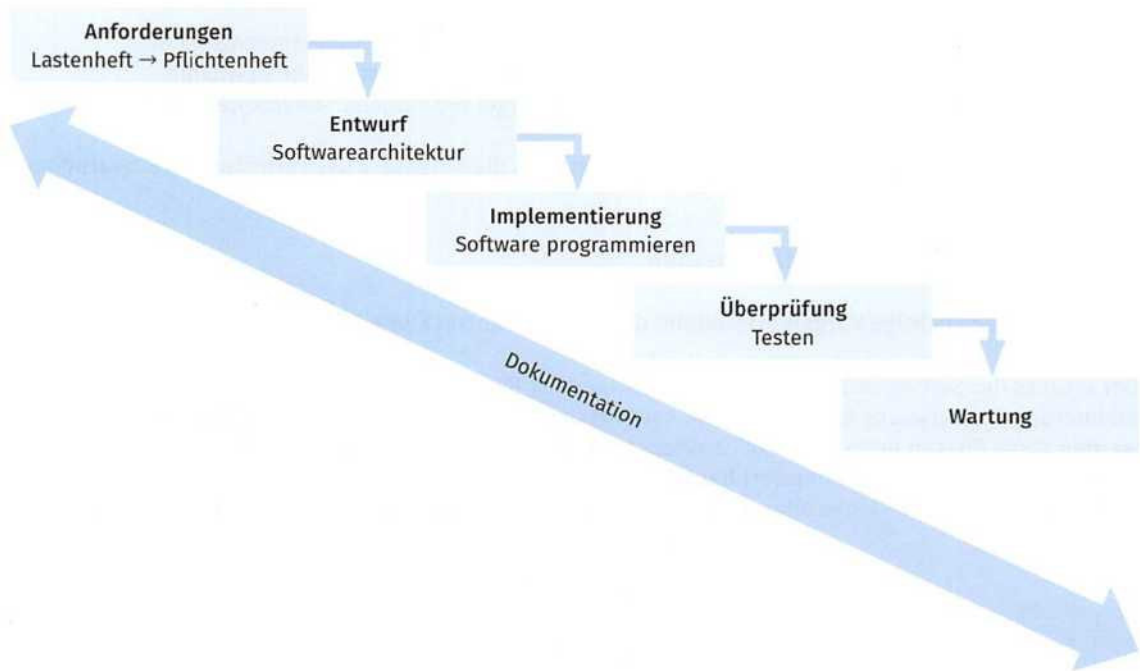
Vorgehensmodelle (Beispiele)	
Klassische Vorgehensmodelle	Agile Vorgehensmodelle
• Wasserfallmodell • Spiralmodell • V-Modell • Rational Unified Process (RUP)	• Serum • DevOps-Ansatz • Extreme Programming(XP) • Kanban

Die klassischen Vorgehensmodelle entsprechen am ehesten der Idee eines Entwicklungsplanes für das Erstellen einer Software. Dabei geht man davon aus, dass alle Anforderungen bekannt sind und sich über den Entwicklungszeitraum hinweg kaum verändern. Der Einsatz agiler Vorgehensmodelle bietet dagegen mehr Flexibilität und die Möglichkeit schneller Anpassung an neue Anforderungen. Dabei sind Codeentwicklung und -ausführung eng miteinander verzahnt, damit Fehler zeitnah gefunden und behoben werden können.

Wasserfallmodell

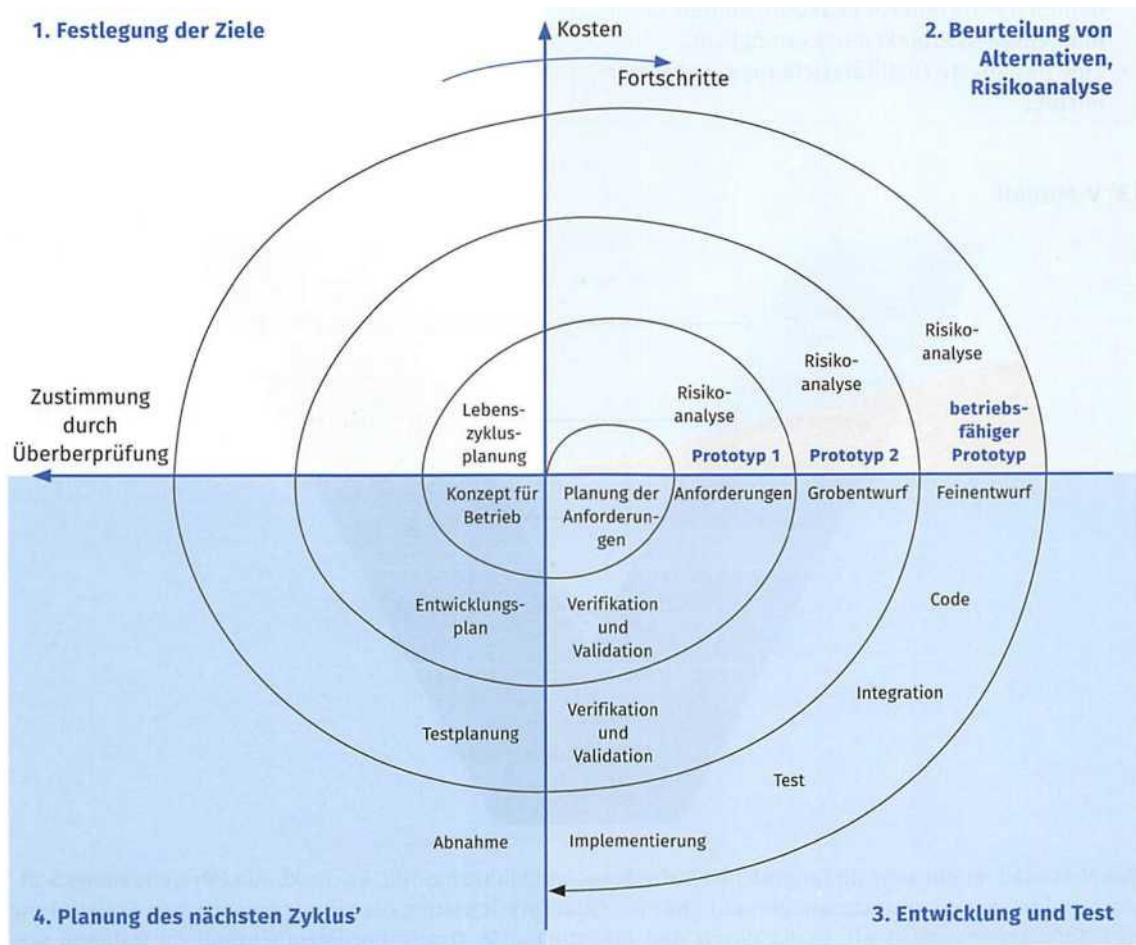
Das Wasserfallmodell ist ein lineares Vorgehensmodell, welches seinen Namen der grafischen Darstellung der einzelnen Phasen als Kaskade verdankt. Es ist in verschiedene Phasen

unterteilt, welche nacheinander abgearbeitet werden. Jeder Phase werden zu Beginn ein Startpunkt, verschiedene Meilensteine und ein Endpunkt zugewiesen, wodurch eine klare Kontrolle des Entwicklungsablaufes möglich ist. Die Dokumentation wird über den gesamten Entwicklungszeitraum parallel zu den einzelnen Phasen erstellt. In der Praxis ist es ein weitverbreitetes Vorgehensmodell, von dem es viele Varianten gibt. Es wird meistens dort eingesetzt, wo sich die Anforderungen an die Software relativ präzise beschreiben lassen. Allerdings kann man beim Wasserfallmodell nur sehr eingeschränkt auf geänderte Anforderungen reagieren. Da größere Softwareprojekte meist eine längere Entwicklungszeit aufweisen, kann es vorkommen, dass die „neue“ Software bereits zum Zeitpunkt ihrer Einführung inhaltlich veraltet ist.



Wasserfallmodell	
Vorteile	Nachteile
- Anforderungen sind präzise definiert. - Phasen sind klar voneinander abgegrenzt. - Relativ einfache Planung und Kontrolle möglich. - Anfallende Kosten und benötigte Zeiten lassen sich relativ einfach kalkulieren. - Entwicklung wird ausführlich dokumentiert.	- Komplexe Softwareprojekte lassen sich nur schwer in klar abgegrenzte Phasen unterteilen. - Geringer Spielraum für Anpassungen aufgrund veränderter Anforderungen. - Der Auftraggeber ist nicht in den Entwicklungsprozess eingebunden. - Fehler werden teilweise erst am Ende des Entwicklungsprozesses erkannt.

Spiralmodell



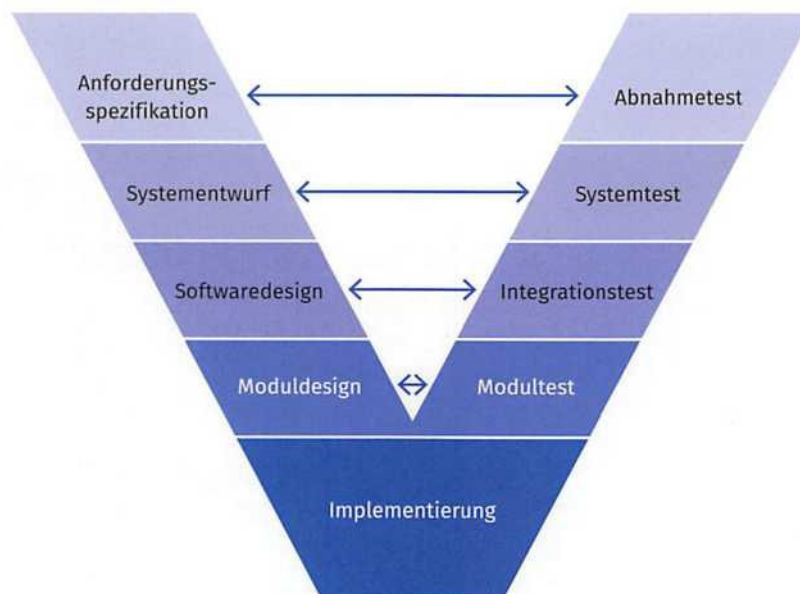
Das Spiralmodell ist im Gegensatz zum Wasserfallmodell ein iterativer Prozess, d.h. ein Prozess, in dem sich bestimmte Phasen stets wiederholen. Damit sollen die Projektrisiken minimiert werden. Es werden folgende vier Grundphasen unterschieden:

1. Festlegung von Zielen und Rahmenbedingungen
2. Betrachtung von Alternativen und Erkennen und Reduzieren von Risiken
3. Entwicklung und Überprüfung (Verifikation, Validierung, Tests) des Zwischenprodukts
4. Auswertung des aktuellen und Planung des nächsten Zyklus'

Das Spiralmodell setzt zwar voraus, dass alle Phasen linear durchlaufen werden. Es können aber auch Phasen übersprungen werden, wenn es sich als notwendig erweist. Das Spiralmodell gilt als eine Weiterentwicklung des Wasserfallmodells. Es ist flexibler als dieses und auch für große Projekte geeignet. Allerdings kommen umso mehr agile Vorgehensmodelle als Alternative infrage, je mehr Flexibilität das Softwareprojekt erfordert.

Spiralmodell	
Vorteile	Nachteile
Es besitzt eine hohe Flexibilität hinsichtlich geänderter Anforderungen. Eine unabhängige Planung und Budgetierung der einzelnen Zyklen sind möglich. Es wird eine flexible, rein risikoorientierte und dennoch kontrollierte Reaktion auf den momentanen Projektstand ermöglicht. Eine integrierte Qualitätssicherung ist gewährleistet.	Auch dieses Modell ist durch die linearen Zyklen schlecht zu parallelisieren. Die Risikobewertung ist schwierig durchzuführen. Es besteht ein hoher Managementaufwand.

V - Modell



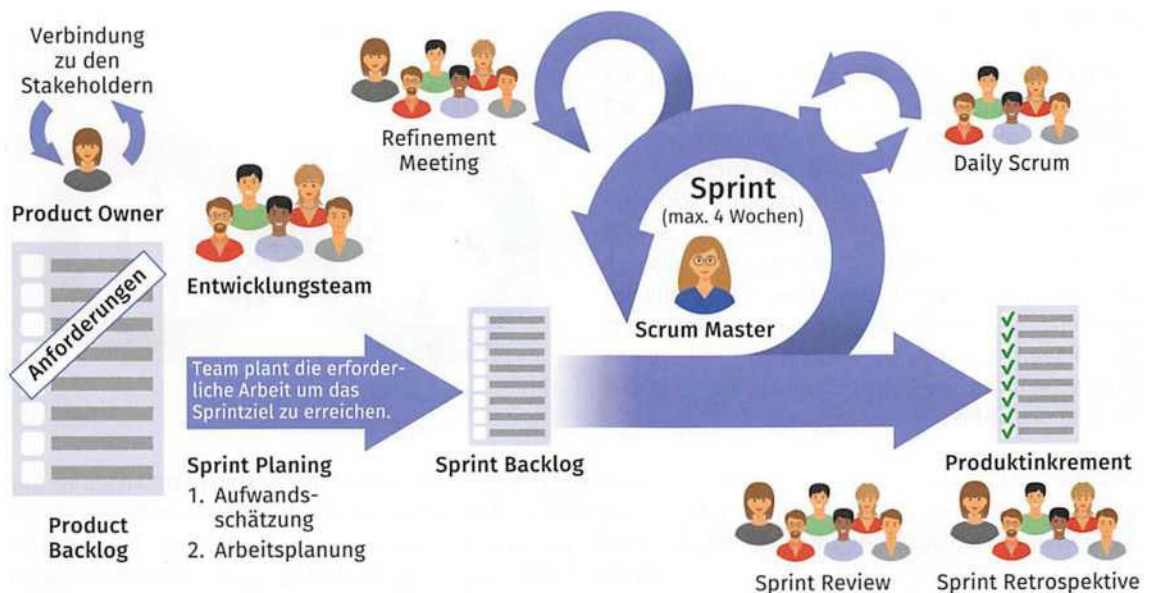
Das V-Modell ist ein sehr umfangreiches Softwareentwicklungsmodell. Es deckt alle Bereiche eines Softwareprojektes vom Projektmanagement über die Qualitätssicherung bis hin zur eigentlichen Entwicklung des Softwareproduktes ab. Es ist durch den öffentlichen Auftraggeber verpflichtend im Rahmen von Ausschreibungen anzuwenden und besitzt somit auch für die Wirtschaft eine hohe Relevanz.

Das V-Modell strukturiert den Softwareentwicklungsprozess ähnlich wie das Wasserfallmodell in einer sequenziellen Abfolge von Phasen. Ein wesentlicher Unterschied besteht in der Zusammengehörigkeit von Entwicklungsphasen und Tests. Auf der linken Seite des „Vs“ sind die Entwicklungsphasen aufgeführt und auf der rechten Seite die dazugehörigen Testphasen. Zu den Testphasen zählen der Unit-, der Integrations-, der System- und der Akzeptanztest (siehe Kapitel 5.8.1).

Alle Tests werden bereits während der Entwicklungszeit spezifiziert. Die Qualitätssicherung steht somit im Mittelpunkt der Überlegungen zum V-Modell. Im V-Modell werden alle Entwicklungs- und die dazugehörigen Testphasen umfangreich dokumentiert. Die Entwicklung des V-Modells begann Ende der 1970er-Jahre und wurde bis zum dem aktuell gebräuchlichen V-Modell XT weiterentwickelt. Das V-Modell XT wird in Unternehmen, Behörden und im militärischen Bereich für die Planung und Durchführung von Systementwicklungsprojekten genutzt und berücksichtigt auch moderne Entwicklungsansätze.

V-Modell	
Vorteile	Nachteile
• Verbesserung der Kommunikation zwischen den Beteiligten • Minimierung der Projektrisiken • Gewährleistung von Qualität • Eindämmung der Gesamtkosten für das gesamte Projekt • Verbesserung der Informationssicherheit • Für große und sehr große Projekte geeignet	• Komplexität und Umfang des Modells • Parallelisierung von Aufgaben ist schwierig • Änderungswünsche lassen sich nicht leicht integrieren • Abgrenzung verschiedener Projektphasen funktioniert schlecht • Überbetonung der Dokumentenerstellung • Für kleine Projekte ungeeignet

Scrum



Scrum stellt eine agile Vorgehensweise der Softwareentwicklung dar. Der Begriff „Scrum“ kommt ursprünglich aus dem Rugby und bedeutet etwa so viel wie „Gedränge“. Das Vorgehensmodell von Scrum ist einfach zu verstehen, jedoch schwierig zu meistern. Die Grundannahme bei diesem Modell ist, dass die Entwicklung einer Software nicht von Anfang bis Ende präzise durchgeplant werden kann. Deshalb erfolgt die Produktentwicklung iterativ (= wiederholend) in Feedback-Schleifen von wenigen Wochen, den sog. Sprints.

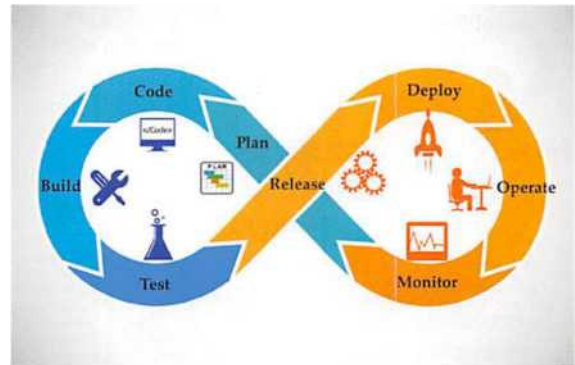


Am Anfang eines Sprints steht eine Liste mit Anforderungen (Product Backlog). Daraus werden vom Entwicklungsteam bestimmte Anforderungen ausgewählt (Sprint Backlog), welche dann in einem Sprint umgesetzt werden. Die Verteilung der Arbeit wird durch das Team selbst organisiert. Am Ende von jedem Sprint soll eine verwendbare Produktversion entstehen. Danach beginnt der Zyklus von vorn und die nächsten Anforderungen werden abgearbeitet. Scrum ist für Teams mit einer Größe von drei bis neun Entwicklern konzipiert. Für größere Projekte werden weitere Organisationsformen benötigt, damit die Arbeit mehrerer Teams koordiniert werden kann.

Scrum	
Vorteile	Nachteile
• Wenige Regeln und leicht verständlich • Kurze Kommunikationswege • Hohe Flexibilität • Hohe Effektivität durch Selbstorganisation • Hohe Transparenz durch regelmäßige Meetings und Backlogs • Kontinuierlicher Verbesserungsprozess • Geringer Administrations-Und Dokumentationsaufwand	• Kein Gesamtüberblick über die komplette Projektstrecke • Hoher Kommunikations- und Abstimmungsaufwand • Wenige konkrete Handlungsempfehlungen • Erschwerte Koordination mehrerer Entwicklungsteams bei Großprojekten • Potenzielle Unvereinbarkeit mit bestehenden Unternehmensstrukturen

DevOps-Ansatz

Üblicherweise beschränken sich agile Modelle wie Scrum auf die reine Softwareentwicklung. Mit dem DevOps-Ansatz wird die agile Vorgehensweise weiterentwickelt und diese auch auf den Betrieb der Software bezogen, um noch schneller auf Änderungen reagieren zu können. Der Begriff „DevOps“ ist ein Kunstwort, welches sich aus den Wörtern „Development“ und „Operations“ zusammensetzt. Der DevOps-Ansatz wird von integrierten IT-Teams umgesetzt, welche die Verantwortung sowohl für die Entwicklung als auch für den Betrieb der Software übernehmen.



DevOps-Prozesse und -Methoden kombinieren somit den Ansatz der agilen Softwareentwicklung mit weiteren Prinzipien, z.B. dem des Systems Management, oder anders ausgedrückt, es werden die agilen Methoden der Entwicklung mit den stabilen Methoden des Betriebs von Software vereint. Als Ergebnis entsteht ein hochagiles Gesamtkonzept, welches das regelmäßige Ausliefern von Softwareversionen ermöglicht, welche sich trotz ihrer kurzen Entwicklungszeit durch eine hohe Qualität auszeichnen. DevOps ist ein großer Schritt zu einem Wandel in der Softwareentwicklung und ein wichtiger Baustein auf dem Weg in die Cloud.

DevOps-Ansatz	
Vorteile	Nachteile
• Schnelleres Time-to-Market • Hohe Liefertreue • Kurze Kommunikationswege • Transparentes Arbeiten • Kürzere Problemlösungszeiten • Erwartungen des Auftraggebers können präzise erfüllt werden • Verbesserter Support	• Potenzielle Unvereinbarkeit mit bestehenden Unternehmensstrukturen • Mitarbeiter müssen teilweise massiv umdenken und sich an neue Methoden und Tools gewöhnen

Lernzielkontrollen Tag 6

1. Welche Aussagen sind richtig?

- a. Grundlage der Softwareentwicklung ist der Softwarelebenszyklus.
- b. In der Analysephase werden die Anforderungen an die zu entwickelnde Software beschrieben.
- c. In der Planungsphase kommen Modellierungssprachen zu Einsatz.
- d. In der Umsetzungsphase wird die Software programmiert und notfalls an eine Datenbank angebunden.
- e. Die Testphase kann neben Modultests auch andere Tests enthalten.
- f. Die Kommentierung von Quellcode gehört zur Dokumentationsphase.
- g. Die Phasen der Softwareentwicklung werden immer auf die gleiche Art und Weise durchlaufen.

2. Ordnen Sie den einzelnen Prozessphasen Tätigkeiten zu.

Ordnen Sie folgende Tätigkeiten den einzelnen Phasen des Softwareentwicklungsprozesses zu:

Tätigkeiten

- | | |
|--|--|
| (A) Erfassung der Ist-Zustände | (I) Konfiguration der Software |
| (B) Optimierung der Software | (J) Anpassung der Software an neue Gegebenheiten |
| (C) Planen der Schnittstellen der Software | (K) Erstellung eines Pflichtenheftes |
| (D) Programmierung von Algorithmen | (L) Auswahl einer Programmiersprache |
| (E) Modultest | (M) Erstellen von Klassendiagrammen |
| (F) Erstellen von Schulungsunterlagen | (N) Systemtest |
| (G) Installieren der Software | (O) Entwickeln eines Datenmodells |
| (H) Aufwandsabschätzung | (P) Schreiben der Bedienungsanleitung |

Phasen des Softwareentwicklungsprozesses nach IEEE-Standard 12207	
Anforderungsanalyse und Erstellung einer Spezifikation	
Design (Entwurf)	
Umsetzung (Implementierung)	
Testen	

Dokumentation	
Auslieferung	
Wartung und Support	

3. Welche Aussagen sind richtig?

- a. Ein professionelles Projektmanagement ist wesentlich für die Durchführung von Projekten.
- b. Bei Projekten wird in der Regel eine Risikoabschätzung vorgenommen.
- c. Teambildung ist eine Aufgabe des Projektmanagements.
- d. Die Ressourcen- und Ablaufplanung spielt eine große Rolle im Projektmanagement.
- e. Die Kontrolle des Projektfortschritts ist in der heutigen Zeit nicht mehr so wichtig.
- f. Softwareprojekte können nach verschiedenen Vorgehensmodellen bearbeitet werden.

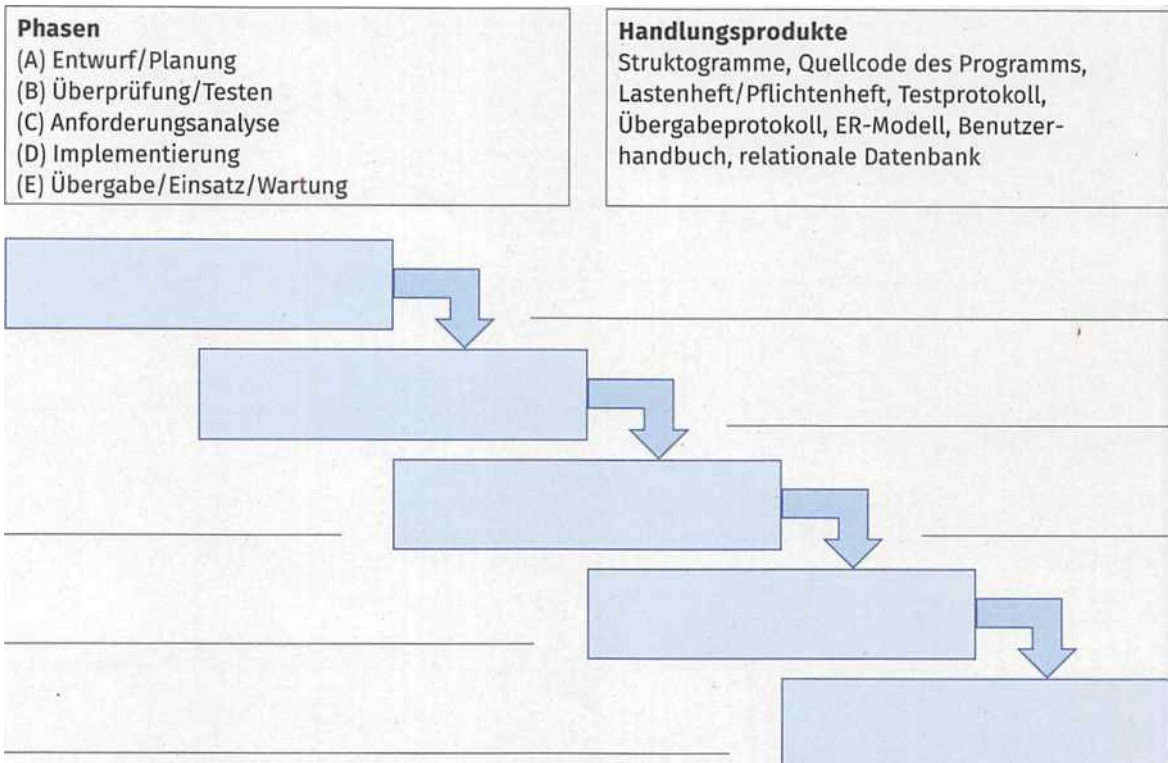
4. Erläutern Sie kurz, was Sie unter Projektmanagement verstehen und nennen Sie fünf Aufgaben des Projektmanagements.

5. Nennen und bewerten Sie jeweils zwei Erfolgs- und Risikofaktoren des Projektmanagements.

6. Welche Aussagen sind richtig?

- a. Mit agilen Vorgehensmodellen lässt sich nicht sehr flexibel auf veränderte Anforderungen reagieren.
- b. Im Wasserfallmodell werden die einzelnen Phasen linear durchlaufen.
- c. Ein großer Vorteil des Wasserfallmodells ist, dass der Auftraggeber in den Entwicklungsprozess stark eingebunden ist.
- d. Beim Spiralmodell werden einzelnen Phasen auch linear durchlaufen.
- e. Das Spiralmodell ist nicht für größere Projekte geeignet.
- f. Kleine Softwareprojekte werden am besten mit dem V-Modell umgesetzt.
- g. Für größere Softwareprojekte im öffentlichen Dienst ist das V-Modell XT von großer Bedeutung.
- h. Mit Scrum kann man flexibel auf geänderte Anforderungen reagieren.
- i. Ein Sprint bei Scrum dauert mindestens vier Wochen.
- j. Der DevOps-Ansatz verbindet die Softwareentwicklung mit dem Betrieb der Software.
- k. Beim DevOps-Ansatz wird auf das Testen der Software meistens ganz verzichtet.

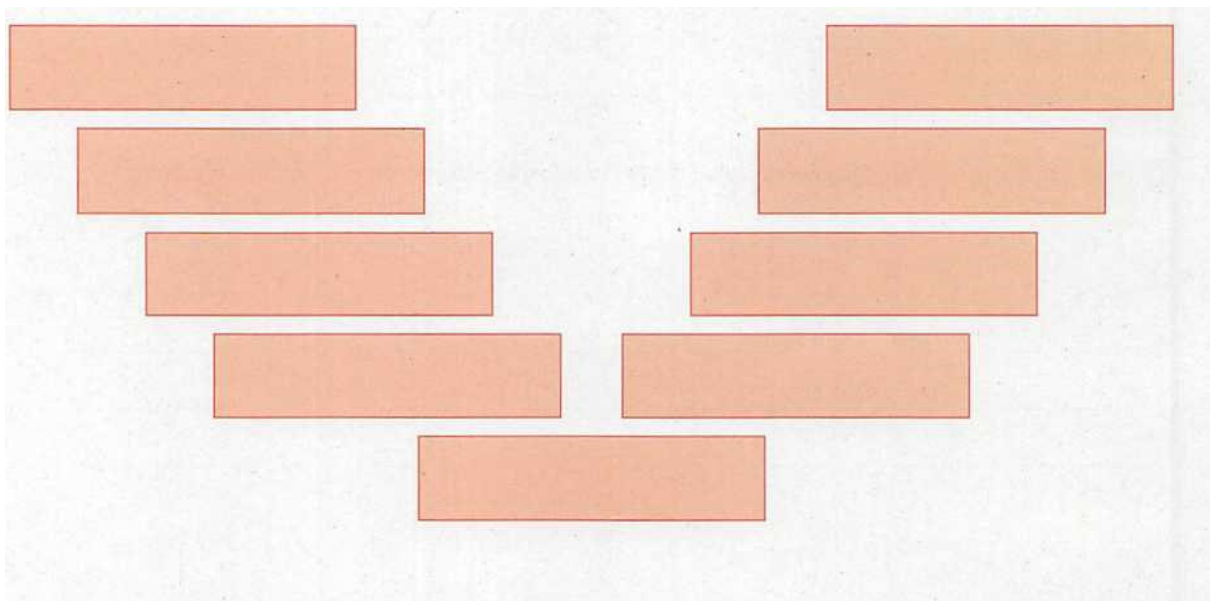
7. Ordnen Sie folgende Phasen des Wasserfallmodells in einer zeitlich sinnvollen Reihenfolge und ergänzen Sie auf den Linien daneben dazu passende Handlungsprodukte.



8. Stellen Sie den Entwicklungsprozess in einem V-Modell dar. Tragen Sie dazu in nachfolgendes Schema ein:
- die Bezeichnungen der Phasen,
 - die Verlaufsrichtung, wenn eine Phase erfolgreich abgeschlossen wurde,
 - die Verlaufsrichtung, wenn ein Test nicht bestanden wurde.

Phasen

- (A) Systemtest
- (B) Softwareentwurf
- (C) Unittest
- (D) Moduldesign
- (E) Implementierung
- (F) Anforderungsspezifikation
- (G) Integrationstest
- (H) Systementwurf
- (I) Abnahmetest



9. Erläutern Sie den DevOps-Ansatz und nennen Sie jeweils zwei Vor- und Nachteile.
10. Übersetzen Sie den folgenden englischen Text sinngemäß ins Deutsche.

Scrum is designed for small teams of developers who break their work into actions that can be completed within fixed duration cycles (called "sprints"). The members of the team track progress and replan in daily 15-minute stand-up meeting, and collaborate to deliver workable software every sprint. There are three core roles in the Scrum framework: the product owner, the development team and the scrum master.

11. Informieren Sie sich in Teamarbeit im Internet über weitere Vorgehensmodelle, z.B. Rational Unified Process (RUP), Kanban oder Extreme Programming (XP), und präsentieren Sie Ihre Ergebnisse.

Tag 7 - Programmiersprachen und -Werkzeuge unterscheiden

Seit der ersten höheren Programmiersprache „Plankalkül“, welche Konrad Zuse in den Jahren 1942 bis 1946 entwickelte, sind bis heute mehr als 1 000 Programmiersprachen entstanden. Einige sind in Vergessenheit geraten, andere dagegen erfreuen sich seit Jahrzehnten ungebrochener Beliebtheit. Mithilfe von Programmiersprachen lassen sich Algorithmen und Daten zu Programmen zusammenfassen, welche nach der Übersetzung in Binärcode vom Rechner ausgeführt werden können.



LE 1 - Aufbau von Programmiersprachen

Programmier Sprache und Quelltext

Unter einer Programmiersprache versteht man nach DIN 44300, Teil 4, eine formale Sprache zum Abfassen (Formulieren) von Verarbeitungsanweisungen für Rechnersysteme. Die mithilfe einer Programmiersprache ausgedrückte, von einem Menschen lesbare Beschreibung heißt Quelltext (oder auch Quellcode bzw. Programmcode).

Programmier Sprachen sind den natürlichen Sprachen wie Deutsch oder English sehr ähnlich. Sie werden allerdings im Unterschied zu den natürlichen Sprachen künstlich entworfen. Der Aufbau einer Programmiersprache unterliegt strengen Regeln und lässt sich anhand verschiedener Informationen genau definieren. Zu diesen Informationen zählen die Syntax, die Schlüsselwörter und der Zeichensatz. Die Schlüsselwörter der meisten Programmiersprachen sind heutzutage aus dem Englischen entlehnt.

Bestandteile einer Programmiersprache	
Syntax	Die Syntax einer Programmiersprache ist deren „Grammatik“. Sie beschreibt sowohl die zulässigen Sprachelemente als auch, wie diese Sprachelemente zu verwenden sind.
Schlüsselwörter	Schlüsselwörter sind Bezeichner, welche von der Programmiersprache für ihre Syntaxausdrücke verwendet werden. Diese dürfen somit auch nicht als Namen für Variablen oder Funktionen gewählt werden. Schlüsselwörter von weitverbreiteten Programmiersprachen sind meist an die englische Sprache angelehnt.
Zeichensatz	Der Zeichensatz einer Programmiersprache besteht in der Regel aus festgelegten alphabetischen und numerischen Zeichen sowie aus Sonderzeichen.

Wie leistungsfähig eine Programmiersprache ist, wird durch den Umfang ihres Befehlssatzes bestimmt. Im Allgemeinen sind mindestens die folgenden Grundfunktionen vorhanden:

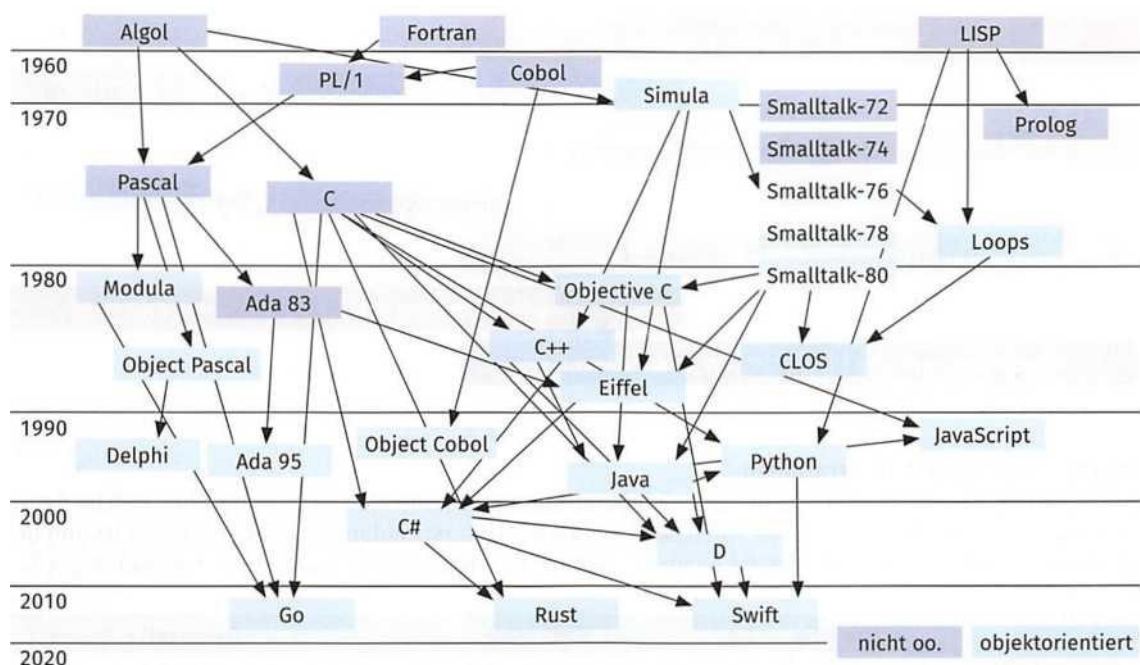
- Befehle zur Ein- und Ausgabe von Daten,
- Befehle zur Deklaration von Variablen und Datenstrukturen,

- Funktionen zur Verarbeitung von Zeichenketten,
- mathematische Grundfunktionen,
- Kontrollstrukturen für die bedingte Ausführung oder die wiederholte Ausführung bestimmter Programmteile.
- Aus der Kombination dieser Grundfunktionen lassen sich dann komplexere Programme erstellen. Bei modernen Sprachen wird diese Grundfunktionalität meist noch um moderne Sprachbestandteile, z.B. objektorientierte Elemente, erweitert.

LE 2 - Historische Entwicklung

Die Anfänge der Programmiersprachen liegen bereits mehr als 200 Jahre zurück und haben ihren Ursprung in den Lochstreifen für mechanische Webstühle. Auf den Lochstreifen waren Informationen über das zu webende Muster enthalten. Das erste mathematisch logische Programm hat die britische Mathematikerin Ada Lovelace im Jahre 1843 geschrieben. Nach ihr wurde auch später die Programmiersprache „Ada“ benannt. Die Programmiersprache, die sie benutzte, war den heutigen Assemblersprachen sehr ähnlich. Allerdings entwickelte sich diese Programmiersprache nicht weiter, weil die entsprechende Hardware fehlte. Erst mit der Erfindung des ersten Computers von Konrad Zuse und der damit verbundenen Entwicklung der Informatik entwickelten sich auch die Programmiersprachen weiter.

Die ersten Programme wurden in Maschinensprache geschrieben, d.h., die Daten und Befehle wurden durch eine lange Reihe von Nullen und Einsen angegeben. Allerdings zeigte sich schnell, dass komplexere Programme in dieser Form nicht umgesetzt werden konnten. So wurden Auswege gesucht und die Entwicklung der höheren Programmiersprachen begann. Höhere Programmiersprachen wurden mit dem Ziel entworfen komplexe Programme in einer für den Menschen leicht verständlichen Sprache schreiben zu können. Allerdings entfernte sich dadurch die Syntax dieser Sprachen stark von der für den Computer benötigten Maschinensprache. So war es notwendig, weitere Programme zu entwickeln, deren Aufgabe darin bestand, den Quelltext der höheren Programmiersprache in Maschinensprache zu übersetzen. Das war die Geburtsstunde der Compiler. Das folgende Schaubild zeigt die Vielfalt der Programmiersprachen, die aus unterschiedlichen Zielvorstellungen entwickelt und weiterentwickelt wurden.



Geschichte ausgewählter Programmiersprachen

Im Jahr 1953 wurde die erste höhere Programmiersprache von John Warner Backus bei IBM entwickelt. Sie erhielt den Namen Fortran (Formular Translator). Ihr folgten dann ziemlich schnell weitere Programmiersprachen wie Algol, COBOL oder Lisp, welche die Grundlagen für viele der modernen Programmiersprachen legten. So auch für die Programmiersprache „C“. In C sind sehr viele Programme geschrieben worden, z.B. Betriebssysteme wie Linux oder

Datenbankmanagementsysteme. Programmiersprachen wie C++, Java, C#, PHP, Python usw. basieren alle auf der C-Syntax, auch wenn diese sehr stark weiterentwickelt wurde.

Die bekannteste und am weitesten verbreitete Sprache der Welt ist heutzutage Java. Trotz der Vielfalt an vorhandenen Sprachen werden aus den unterschiedlichsten Gründen immer noch neue Programmiersprachen entwickelt, meistens, um Unzulänglichkeiten von etablierten Programmiersprachen zu verbessern. Bekannte Neuentwicklungen in diesem Bereich sind die Sprachen Go, Rust und Swift.

LE 3 - Einteilung von Programmiersprachen

Um die Vielzahl der Programmiersprachen unterscheiden und bewerten zu können, werden diese nach unterschiedlichen Kriterien eingeteilt. Die Einteilung hilft dabei, die optimale Sprache für die Lösung der jeweiligen Programmieraufgabe herauszusuchen.

Unterscheidung der Programmiersprachen		
Anwendungsgebiete	Generationen	Programmierparadigmen

Unterscheidung nach Anwendungsgebieten

Programmiersprachen können sehr allgemein oder speziell für ein bestimmtes Gebiet ausgelegt sein. Spezialisierte Sprachen sind für das entsprechende Gebiet optimiert, d.h., es lassen sich mit relativ wenig Quelltext komplexe Sachverhalte aus diesem Gebiet beschreiben. Universellere Sprachen dagegen lassen sich auf vielen Gebieten einsetzen, auch wenn sie nicht immer optimal zum Problem passen. Zu den universell einsetzbaren Sprachen gehören u.a. C, C++, Java, Pascal oder Python. Einige Beispiele für verschiedene Anwendungsgebiete mit den entsprechenden Programmiersprachen sind nachfolgend aufgeführt.



Unterscheidung nach Generationen

Die Einteilung der Programmiersprachen nach Generationen ist eine klassische Einteilung, welche heute nur noch eine untergeordnete Rolle spielt, weil die Art der Unterscheidung zu unübersichtlich ist und bei der Auswahl der Programmiersprache nicht viel weiterhilft. Die Übersicht dient nur der Vollständigkeit.

Name	Art der Sprache	Merkmale	Beispiel
1. Generation	Maschinensprache	Binärer Code; maschinennahe Programmierung; direkt auf dem Computer lauffähig; schwer verständlich	Maschinencode für IBM-Großrechner OS/390 0001 1010 0011 0100 0001 0000 1101
2. Generation	Assemblersprache	Leichter verständliche Symbole (Mnemonics); nicht direkt auf dem Computer lauffähig	Verschiedene Assembler: mov ds, ax mov ax, 09h int 21h
3. Generation	Höhere prozedurale Sprachen	Prozeduraler Aufbau in Funktionen; der menschlichen Sprache angenähert; unabhängig vom Computersystem	C, Fortran, Java, C#, Python u.a. z1 = 1.5 z2 = 6.3 summe = float(z1) + float(z2) print(summe)
4. Generation	Sprachen für Datenbanken	Einsatz z. B. für DB-Abfragen, Tabellenkalkulation	SQL, Natural SELECT * FROM Artikel
5. Generation	Sprachen der künstlichen Intelligenz	Wurden dazu entwickelt, um Programme schreiben zu können, die menschliches Denken nachahmen	LISP, PROLOG, HASKELL defun f(x) (/ (+ (* x x) 3) (- a x)))

Unterscheidung nach Programmierparadigmen

Die Unterscheidung der Programmiersprachen nach Generationen ist nicht mehr hilfreich bei der Auswahl einer geeigneten Sprache. Die Unterscheidung nach Programmierparadigmen bietet wesentlich mehr Unterstützung.

Programmierparadigma

Ein Programmierparadigma ist das grundlegende Konzept (Grundauffassungen, Methoden und Prinzipien), welches einer Programmiersprache zugrunde liegt.

Unterscheidung nach Programmierparadigmen				
Imperative maschinennahe Sprachen	Höhere imperative Sprachen	Objektorientierte Sprachen	Deklarative logische Sprachen	Deklarative funktionale Sprachen
z. B. Assembler	z.B. C, Fortran, COBOL, Pascal	z. B. C#, Java, Smalltalk	z. B. Prolog	z.B. Lisp, Haskell

Man kann das imperative, das objektorientierte und das deklarative Programmierparadigma unterscheiden. Bei **imperativen Programmiersprachen** besteht ein Computerprogramm aus einer Folge von Anweisungen, die genau vorgibt, zu welchem Zeitpunkt was vom Rechner erledigt werden soll. Um die Befehlsausführung zu steuern, werden bestimmte Kontrollstrukturen wie Schleifen oder Verzweigungen verwendet. Die imperative Programmierung ist das älteste Paradigma. Im Gegensatz dazu betrachtet man in der deklarativen Programmierung nicht, *wie* etwas berechnet werden soll, sondern was berechnet wird. Das heißt, es wird nicht mehr der Lösungsweg programmiert, sondern nur noch

angegeben, welches Ergebnis gewünscht ist. Bei den **objektorientierten Programmiersprachen** wiederum werden Daten und Programmcode in übersichtlichen, wiederverwendbaren Objekten gekapselt. Dies reduziert den Programmieraufwand und hilft, Fehler zu vermeiden. Diese Grundparadigmen lassen sich noch weiter unterteilen. Darauf wird hier aber nicht weiter eingegangen. Viele Programmiersprachen lassen sich klar in ein Programmierparadigma einordnen. Andere stellen eine Art Mischform dar und können mehreren Paradigmen zugeordnet werden. Die Beispiele in der Übersicht sind nur dem Hauptparadigma zugeteilt.

LE 4 - Auswahl einer Programmiersprache

Bei der Auswahl einer geeigneten Programmiersprache für ein Softwareprojekt spielen mehrere Faktoren eine Rolle. Zunächst sollte sich die Sprachwahl nach der zu programmierenden Anwendung und dem Anwendungsgebiet richten. In den meisten Fällen ergibt sich die Programmiersprache schon von selbst, entweder, weil es für die Zielplattform konkrete Empfehlungen von dem Hersteller gibt oder wenn eine Anwendung erweitert und angepasst werden soll und die Programmiersprache somit schon feststeht.

Anwendungsbeispiele für Programmiersprachen

Zum Beispiel wird man für eine Android-App eher Java und für eine iOS-App Objective-C oder das moderne Swift verwenden. Anwendungssoftware für Windows wird meistens mithilfe von C#, C++ oder Visual Basic entwickelt. Für systemnahe Programme ist C die Sprache der Wahl. Web-Anwendungen werden heutzutage meist in JavaScript geschrieben, aber auch PHP findet in diesem Bereich noch seine Verwendung. Die Sprache „R“ ist spezialisiert auf die statistische Auswertung von Daten und SQL eignet sich perfekt zur Abfrage von Datenbanken. Python ist momentan die angesagte Sprache für das Programmieren auf dem Raspberry Pi, eignet sich aber auch sehr gut zum Erlernen des Programmierens als solches und findet auf vielen Plattformen ihre Anwendung.

In vielen Fällen hat man aber auch einige Alternativen zur Auswahl. In diesem Fall spielt die Erfahrung, welche man selbst oder das gesamte Team in den einzelnen Programmiersprachen besitzt, eine große Rolle. Der Einarbeitungsaufwand in eine neue Programmiersprache schwankt mitunter sehr und kann das Projekt unnötig verzögern. Hier sollte, wenn möglich, immer eine Sprache gewählt werden, mit der man sich gut genug auskennt. Allerdings sind auch das Erlernen und Einarbeiten in neue Sprachen zu empfehlen, nur sollten dafür nicht unbedingt wichtige Softwareprojekte gewählt werden.

Platz	RedMonk (Q1/2020)	TIOBE (6/2020)	PYPL (2/2020)
1	JavaScript	C	Python
2	Java	Java	Java
3	Python	Python	JavaScript
4	PHP	C++	C#
5	C#	C#	PHP
6	C++	VB.net	C/C++
7	Ruby	JavaScript	R
8	C	PHP	Objective-C
9	Swift	Swift	Swift
10	Objective-C	Go	Matlab

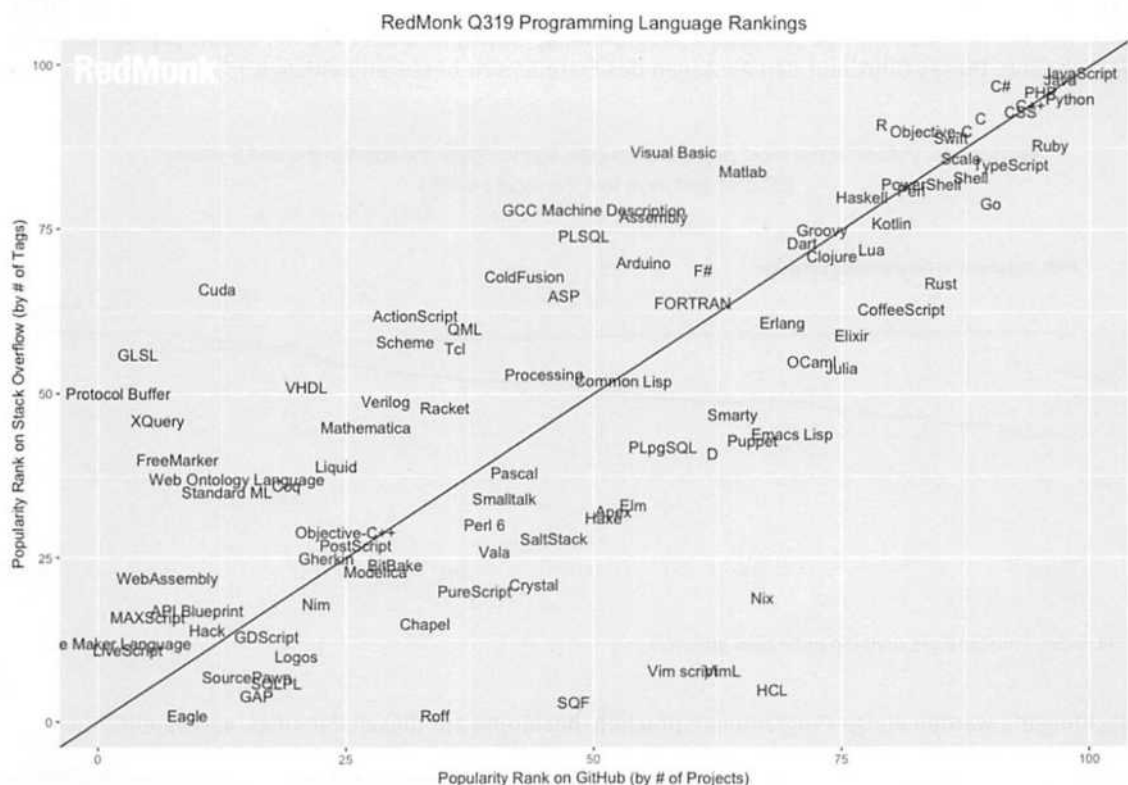
Meistgenutzte Programmiersprachen nach unterschiedlichen Rankingverfahren

Generell kann man aber sagen, dass das Programmieren mit jeder Sprache erlernt werden kann. Die Programmiersprachen sind sich teilweise sehr ähnlich, besonders wenn sie dem gleichen Programmierparadigma zugeordnet sind. Wer etwa die Grundlagen in Python beherrscht, findet sich schnell in C, JavaScript, Visual Basic oder PHP zurecht. Auch die Methoden zum Testen des Codes und zur Fehlersuche sind bei allen Sprachen ähnlich. Außerdem findet die meiste Arbeit bei der Programmierung, völlig unabhängig von der Sprache, im Kopf statt. Das heißt, das Schreiben eines Programms in der konkreten Syntax einer Programmiersprache spielt beim Programmieren keine so große Rolle, wie meist angenommen wird. Wenn das zu lösende Problem nicht verstanden wird und man auch keine Idee für einen Algorithmus hat, dann kann man noch so gut eine Programmiersprache beherrschen, ein zufriedenstellendes Programm wird man nicht zustande bekommen.

Für Programmieranfänger ist es aber nicht unerheblich, mit welcher Programmiersprache sie beginnen. Die einzelnen Sprachen sind nämlich unterschiedlich komplex und damit mehr oder weniger leicht zu erlernen. Bei einigen Sprachen, z.B. Python, kann man einfach drauflostippen und hat sehr schnell ein Programm fertig. Bei anderen Sprachen muss man sich im Vorfeld einige grundlegende Konzepte aneignen, bevor man beginnen kann. Dementsprechend sind die Lernkurven auch unterschiedlich steil. Deswegen empfiehlt es sich, mit einer einfachen, aber mächtigen Sprache wie Python zu starten, weil man dort schnell Ergebnisse erzielt, motiviert bleibt und trotzdem viel erreichen kann.

Ein weiterer Aspekt beim Erlernen einer Sprache ist deren Verbreitung und Popularität, weil man mit einer Sprache, welche sehr häufig Anwendung findet, an vielen Softwareprojekten mitarbeiten kann. Um einen Überblick über aktuelle und häufig verwendete Sprachen zu erhalten, können Programmier- Sprachen-Rankings wie RedMonk, TIOBE-Index oder PYPL genutzt werden (siehe auch S. 505).

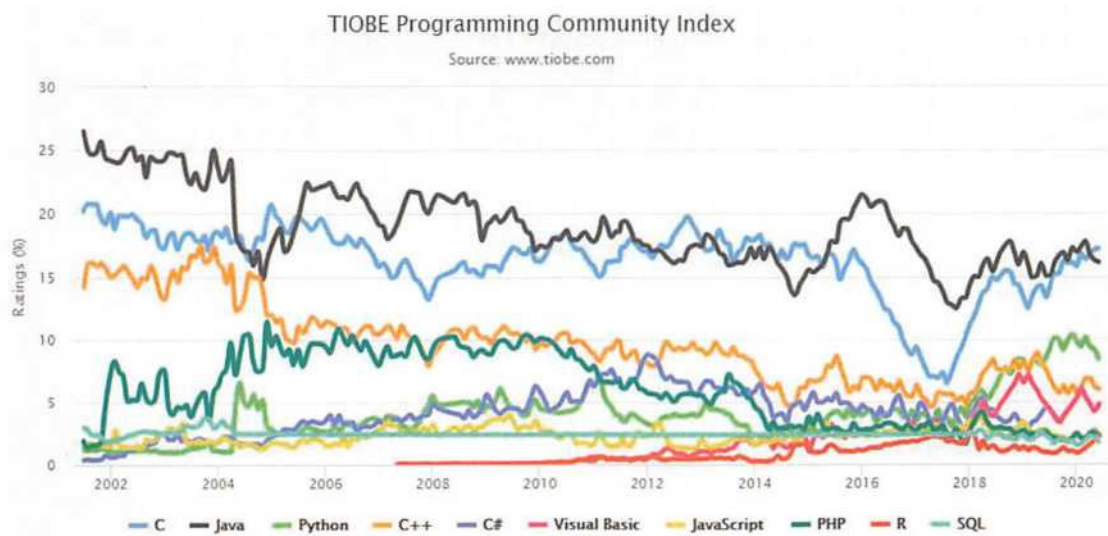
RedMonk benutzt für das Ranking ein Verfahren, dass sich besonders stark an der Entwickler-Community orientiert. So werden die Hashtags bei Stackoverflow und die Anzahl an GitHub-Projekten, die eine bestimmte Sprache nutzen, miteinander in Bezug gesetzt und auf einer Skala von 1 bis 100 gewertet.



RedMonk Ranking

Der TIOBE-Index nutzt als Grundlage seiner Auswertung die Zahl der Treffer für die Suchanfrage „Sprache + programming“ in den populärsten Suchmaschinen und Internetportalen

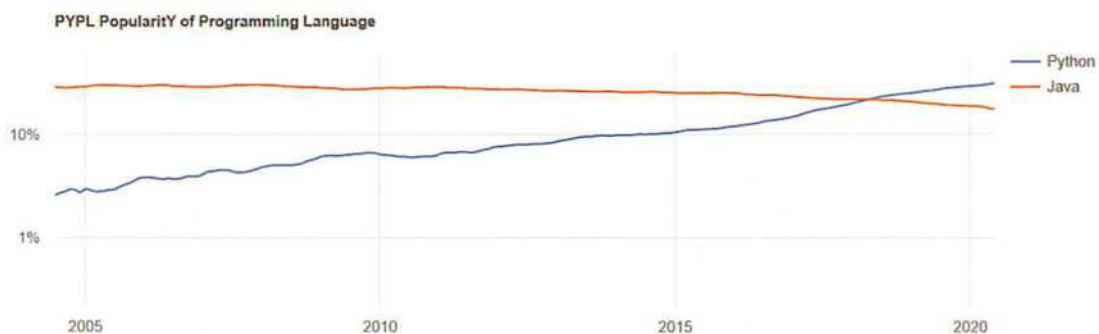
wie Google, Wikipedia oder YouTube. Damit wird versucht, eine Übersicht über den aktuellen Markt zu geben und Trends aufzuzeigen.



TIOBE Index, vollständiges Ranking unter www.tiobe.com/tiobe-index/

PYPL (Popularity of Programming Languages) verfolgt einen anderen Ansatz als TIOBE. Während TIOBE nach Treffern der Suchanfrage „Sprache + programming“ sucht, wird bei PYPL der Zusatz „Sprache + tutorial“ gewählt. Dieses führt laut den Aussagen des Erstellers zu besseren Rankings.

Worldwide, Python is the most popular language. Python grew the most in the last 5 years (20.3%) and Java lost the most (-8.0%)



PYPL Index, vollständiges Ranking unter pypl.github.io

Nachfolgend werden einige Programmiersprachen, basierend auf diesen Rankings, vorgestellt.

Programmierersprachen

Python

Python ist eine der meist verwendeten Programmiersprachen. Die Syntax der Sprache ist benutzerfreundlich und leicht verständlich und eignet sich somit sehr gut zum Erlernen des Programmierens. Die Sprache selbst ist sehr vielseitig und kann für viele Anwendungsbereiche eingesetzt werden. Dank verschiedener Bibliotheken kann die Sprache auch im Bereich der künstlichen Intelligenz und bei Big-Data-Anwendungen eingesetzt werden. Auch bei der Web-Entwicklung, insbesondere durch die Frameworks Django oder Flask, spielt Python eine Rolle.

```
x = 10
y = 20
summe = x + y
print("Summe: ", summe)
```

Java

Java ist eine universell einsetzbare Programmiersprache, die objektorientiert ist und auf möglichst geringe Abhängigkeiten bei der Implementierung ausgelegt wurde. Mit dieser Programmiersprache können Anwendungen für verschiedenste Plattformen, u.a. mobile Endgeräte, Terminals oder Desktop-PCs, entwickelt werden. Java gehört noch immer zu den meist verwendeten Sprachen, auch wenn die objektorientierte Programmiersprache Kotlin an Popularität gewinnt.

```
int x = 10;
int y = 20;
int summe = x + y;
System.out.println("Summe : " + summe);
```

C#

C# (C-Sharp) wurde von Microsoft entwickelt und dient hauptsächlich der Erstellung von Anwendungen für Windowsplattformen. Die Grundlage dafür bietet das .NET-Framework, welches explizit für Windows entwickelt wurde. Trotzdem wird die Sprache vielfältig in den verschiedensten Bereichen eingesetzt. Klassische Desktop-Anwendungen lassen sich damit genauso gut erstellen wie Webseiten (ASRNET-Framework) oder mobile Anwendungen. Aufgrund von Unity spielt C# auch bei der Entwicklung von Spielen eine sehr große Rolle.

```
int x = 10;
int y = 20;
int summe = x + y;
Console.WriteLine("Summe: " + summe);
```

C/C++

C wurde schon sehr früh entwickelt und ist eine rein prozedurale Sprache. Die Anwendungsbereiche von C sind auch sehr vielfältig. Sie reichen vom Einsatz in der Systemprogrammierung (z. B. Linux Kernel) über Datenbanken bis hin zur Spieleprogrammierung. Teilweise ist die Popularität dieser recht „alten“ Sprache auch historisch bedingt, da früher sehr viele Programme in C geschrieben wurden, welche bis heute gewartet und weiterentwickelt werden müssen. Viele andere Sprachen orientieren sich an der Syntax von C. C++ ist eine Programmiersprache, die als Erweiterung der C-Sprache entstanden ist, um objektorientiert programmieren zu können. Zum Einsatz kommt C++ in den gleichen Bereichen wie C, bietet aber durch seine objektorientierten Elemente zusätzliche Möglichkeiten. C++ gilt als eine sehr komplexe Sprache und es bedarf einiger Zeit, diese vollständig zu erlernen.

```
int x = 10;
int y = 20;
int summe =x+y;
printf("Summe: %d", summe)
```

JavaScript

JavaScript ist eine Prototypenbasierte Programmiersprache, die als integrierter Bestandteil eines jeden Web-Browsers verwendet wird. Das bedeutet, dass keine Kompilierung erforderlich ist, da der Browser selbst den Code liest und die angegebenen Aktionen ausführt. Dies macht JavaScript zu einer der besten Sprachen für die Erstellung von Websites. Aufgrund der Funktionen von HTML5 und von 2D- und 3D-Grafikbibliotheken spielt JavaScript auch bei der Entwicklung von Browserspielen eine wichtige Rolle.

```
<script>
var x = 10;
var y = 20;
var summe =x+y;
alert("Summe: " + summe);
</script>
```

PHP

PHP ist eine an C und Perl angelehnte Programmiersprache für Server-Side-Codes. Diese Sprache garantiert eine gute Kommunikation zwischen Webseiten und Servern, was bedeutet, dass die damit entwickelten Webseiten stabil sind und gute Performance bieten. So konnte sich die Scriptsprache PHP zur einer der beliebtesten Programmiersprachen für das Erstellen von Webseiten und Webanwendungen entwickeln. Viele Content-Management-Systeme wie WordPress oder Joomla sowie auch Shopsysteme für E-Commerce-Anwendungen beruhen auf PHP.

```
<?php
$x = 10;
$y = 20;
$summe = $x + $y;
echo "Summe: $summe";
```

LE 5 - Frameworks

Für häufig sich wiederholende Aufgaben gibt es Frameworks, die einem viel Arbeit abnehmen. Mit einem Framework wird das Grundgerüst für eine spätere Softwarelösung geschaffen. Es stellt eine Struktur zur Verfügung, die von sämtlichen Anwendungen gleichermaßen genutzt werden kann.



Die genaue Ausgestaltung eines Frameworks ist jedoch nicht eindeutig definiert. In der Praxis gibt es eine Vielzahl von Frameworks, die in unterschiedlichen Bereichen zum Einsatz kommen. Dazu gehören u.a.:

- **Application Frameworks:** Sie dienen als Programmiergerüst für eine bestimmte Klasse von Anwendungen, die ihrerseits der Bereitstellung von Funktionen und Programmstrukturen dienen.
- **Domain Frameworks:** Sie dienen als Programmiergerüst für einen bestimmten Problembereich. Sie enthalten typischerweise Funktionen und Strukturen, die zu einer Lösung des Problems benötigt werden.
- **Test Frameworks:** Sie dienen der Ausführung von Softwaretests, etwa für Programm- und Webentwicklungen.
- **Web Frameworks:** Sie dienen als Grundlage für die Entwicklung von Webanwendungen. Sie sind in der Regel dynamisch verwendbar.

Durch den Einsatz von Frameworks wird Programmierern Routinearbeit abgenommen, sodass sie sich auf die eigentliche Aufgabe konzentrieren können. Man kann sich beispielsweise durch den Einsatz von Spiele-Engines wie Unity oder der Cry-Engine um den eigentlichen Inhalt des Spiels kümmern, ohne sich mit den konkreten Belangen der Grafikausgabe auseinandersetzen zu müssen. Web-Frameworks wiederum erleichtern das Entwickeln von Web-Anwendungen. Bekannte Frameworks in diesem Bereich sind Symfony für PHP, Ruby on Rails, Django für Python und Play für Java. Für grafische Anwendungen auf dem Desktop empfiehlt sich für Windows das .NET-Framework. Für plattformunabhängige Anwendungen bieten sich Gtk+ oder Qt-Framework an.

LE 6 - Programmierwerkzeuge

Um ein Programm zu schreiben, bedarf es nicht vieler Programmierwerkzeuge. Im einfachsten Fall reicht ein Editor, ein Compiler oder Interpreter.

Programmierwerkzeuge

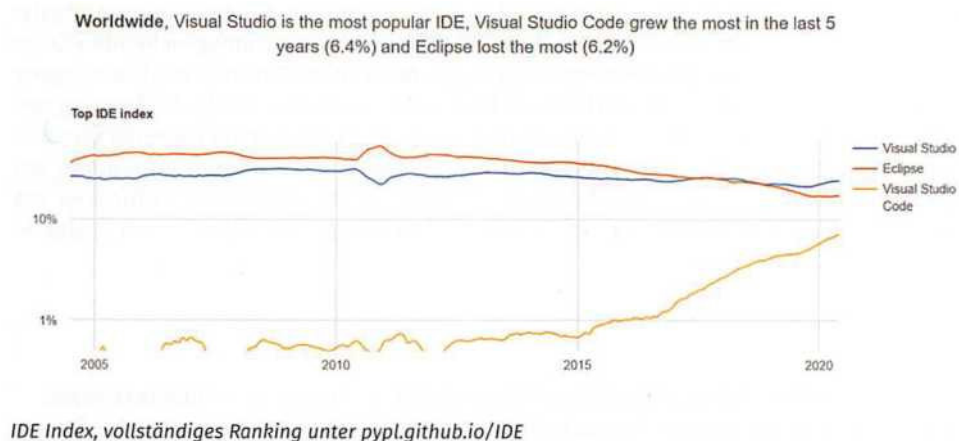
Unter Programmierwerkzeugen werden Softwareanwendungen verstanden, durch welche die Erstellung von Computerprogrammen unterstützt wird.

Wenn die Programme umfangreicher werden, dann sollte über den Einsatz einer integrierten Entwicklungsumgebung (IDE) nachgedacht werden. In einer IDE sind neben einem Texteditor mit Syntaxhervorhebung, einem Compiler oder Interpreter auch Debugger, Profiler, Projekt- und Versionsverwaltung unter einer gemeinsamen Oberfläche zusammengefasst. Durch Plugin-Schnittstellen lassen sich die meisten Entwicklungsumgebungen auch um weitere Funktionalitäten erweitern.

Integrierte Entwicklungsumgebung (IDE)

Bei einer IDE sind die wichtigsten Softwareentwicklungswerkzeuge in einem Programm unter einer einheitlichen Oberfläche zusammengefasst. Das hat das Ziel, die Arbeit der Programmierer bei der Programmerstellung zu vereinfachen, z. B. indem viele Aufgaben schon vorkonfiguriert und teilweise automatisiert sind.

So ist es mit modernen IDE möglich, in mehreren Programmiersprachen und für verschiedene Plattformen Programme zu entwickeln. Bekannte IDE sind z.B. Eclipse, VisualStudio oder PyCharm. Für manche Entwickler sind die heutigen IDE schon mit zu vielen Funktionen überladen, von denen sie nur einen Bruchteil benötigen. Deshalb hat ein Trend eingesetzt, wieder auf leichtgewichtige Editoren zurückzugehen, welche durch Plugins an die persönlichen Bedürfnisse angepasst werden können. Ein Beispiel dafür ist Visual Studio Code, ein moderner Texteditor, welcher plattformübergreifend eingesetzt werden kann. PYPL führt neben einem Programmiersprachenranking auch einen Index über die am häufigsten verwendeten IDEs.



Einige wichtige Programmierwerkzeuge sind in der nachfolgenden Tabelle näher erläutert.

Wichtige Programmierwerkzeuge	
Texteditor	Texteditoren werden zur Eingabe und Änderung des Quelltexts von Programmen verwendet. Für eine bessere Lesbarkeit des Quelltextes bieten die meisten für die Programmierung ausgelegten Texteditoren Syntaxhervorhebung (Syntax-Highlighting) an.
GUI-Editor (GUI-Builder)	GUI-Editoren erleichtern das Design von grafischen Benutzeroberflächen, indem sie das Anordnen von Steuerelementen per Drag-and-drop ermöglichen. Auch wird in der Regel das Programmieren von Events unterstützt.
Compiler, Interpreter, Assembler	Die Hauptaufgabe dieser Werkzeuge ist die Übersetzung des Quellcodes in ausführbare Maschinensprache. Es gibt für viele Programmiersprachen sowohl Compiler als auch Interpreter oder eine Kombination aus beiden.
Debugger	Der Debugger (von engl. „Entlausen“, Bug = Schädling, Fehlfunktion) ist ein Entwicklungswerkzeug, welches den Entwickler bei der Fehlersuche unterstützt. Mithilfe des Debuggers kann das Programm Schritt für Schritt ausgeführt und die einzelnen Programmzustände untersucht werden.
Versionsverwaltung	Eine Versionsverwaltung ist eine Software zur Speicherung und Verwaltung von Dateien und Dokumenten, welche bei der Entwicklung von Programmen erstellt werden.

An dieser Stelle werden die Entwicklungswerkzeuge Compiler und Interpreter noch einmal genauer betrachtet, weil sie essenziell für die Erstellung von Programmen sind. Alle Programme müssen vor der Laufzeit in Maschinenbefehle umgewandelt werden, die der Prozessor versteht und schnell ausführen kann. Diese Aufgabe übernehmen Compiler oder Interpreter. Ein Compiler übersetzt den Quelltext einmal in ein fertiges Programm und wird danach nicht mehr benötigt.

Compiler Ein Compiler übersetzt den Quelltext von einem Programm in einen Maschinencode, ohne jedoch die Befehle auszuführen. Die Programmiersprache, aus der die übersetzten Programme stammen, wird als Quellsprache bezeichnet. Dementsprechend nennt man das zu übersetzende Programm Quelltext.

Ein Nachteil besteht darin, dass die kompilierten Programme maschinenabhängig sind und somit nicht auf jeder Computerplattform ausgeführt werden können. Wenn das Programm auf einer anderen Plattform laufen soll, dann muss es angepasst und neu für diese Plattform von einem entsprechenden Compiler übersetzt werden. Interpreter hingegen übersetzen das Programm erst zur Laufzeit. Die Übersetzung wird bei jedem neuen Start des Programms wiederholt. Deshalb wird der Interpreter für Ausführung des Programms immer zwingend benötigt. Der Vorteil von Interpretern besteht darin, dass die Programme in der Regel plattformunabhängig sind, d.h., sie müssen nur einmal geschrieben werden und laufen auf allen Plattformen, auf denen ein entsprechender Interpreter installiert ist. Dieser Vorteil wird aber mit einer geringeren Ausführungsgeschwindigkeit erkauft, weil der Übersetzungsvorgang etwas Zeit kostet. In den meisten Fällen fällt das aber gar nicht auf.

Interpreter

Ein Interpreter übersetzt während der Laufzeit eines Programms den vorhandenen Quelltext Anweisung für Anweisung in eine maschinenverständliche Form, d. h., er bearbeitet ein Programm zeilenweise, wobei jede Zeile einer Syntaxprüfung unterzogen wird.

Für viele Sprachen gibt es sowohl Compiler als auch Interpreter, wobei es immer, je nach Einsatzgebiet, eine bevorzugte Art gibt. Andere Programmiersprachen wie Java setzen auf eine Kombination aus Compiler und Interpreter. Hier wird der Quelltext in einen Bytecode kompiliert, welcher dann von einem Interpreter ausgeführt wird.

Neben diesen Werkzeugen, welche die Programmentwicklung unmittelbar unterstützen, kommen auch weitere Werkzeuge zum Einsatz, welche viele andere Tätigkeiten im Softwareentwicklungsprozess vereinfachen. Beispielhaft werden hier nur Werkzeuge zur grafischen Datenmodellierung oder Softwareprogramme zur Unterstützung des Projektmanagement genannt.

Lernzielkontrollen Tag 7

1. Welche Aussagen sind richtig?
 - a. Es gibt nur sehr wenige Programmiersprachen.
 - b. Programmiersprachen besitzen eine Syntax und Schlüsselwörter.
 - c. Es gibt imperative und deklarative Programmiersprachen.
 - d. Die Auswahl einer Programmiersprache für eine Programmieraufgabe ist in aller Regel unabhängig vom jeweiligen Aufgabengebiet.
 - e. Um die Popularität der Programmiersprachen zu messen, gibt es Indizes wie RedMonk oder den TIOBE-Index.
 - f. Ein Framework stellt dem Programmierer verschiedene Bibliotheken und Module für bestimmte Programmieraufgaben zur Verfügung und erleichtert somit die Programmierarbeit.
 - g. C ist eine neue Programmiersprache.
 - h. Python ist leicht zu erlernen und eignet sich sehr gut für den Einstieg in die Programmierung.
 - i. Eine IDE ist ein Spezieller Texteditor.
 - j. Ein Debugger findet logische Fehler in einem Programm.
 - k. Interpreter übersetzen den Quelltext bei jedem Programmstart neu.
 - l. Kompilierte Programme sind in der Regel langsamer in der Programmausführung als interpretierte Programme.
2. Überprüfen Sie Ihr Wissen über Programmiersprachen und -Werkzeuge.

Programmierersprachen und -Werkzeuge

1. Ein Verein möchte sich im Internet präsentieren und dafür eine Webseite entwickeln lassen. Wählen Sie für die Implementation der Seiten eine geeignete Sprache aus.

- a) C++
- b) Visual Basic
- c) HTML/CSS
- d) PCL
- e) SQL

2. Eine Software soll in einer rein objektorientierten Sprache erstellt werden. Welche Sprachen können verwendet werden?

- a) C
- b) Java
- c) Assembler
- d) C++
- e) JavaScript

3. Sie sollen in einem Programm einen Fehler suchen. Dabei sollen Sie sich die in den Variablen gespeicherten Werte während der Laufzeit anzeigen lassen. Welches der folgenden Werkzeuge ist dazu geeignet?

- a) Compiler
- b) Editor
- c) Interpreter
- d) Linker
- e) Assembler
- f) Debugger

4. Welche Aussage über Interpreter-Sprachen sind richtig?

- a) Interpreter-Sprachen benötigen zur Laufzeit ein Programm, das den Quellcode einliest und ausführt.
- b) Interpreter-Sprachen korrigieren fehlerhafte Eingaben vom Benutzer.
- c) Interpreter-Sprachen können keine Betriebssystemfunktionen aufrufen.
- d) Interpreter-Sprachen können keine Programmbibliotheken benutzen.
- e) Interpreter-Sprachen werden nur für Programme verwendet, welche Echtzeitdaten verarbeiten.

5. Für die Programmerstellung stehen Ihnen einige Tools zur Verfügung. In welchen Aufzählungen werden die Werkzeuge in der richtigen Reihenfolge verwendet?

- a) Interpreter - Listengenerator - Editor
- b) Debugger - Datenbankgenerator- Editor
- c) Struktogrammgenerator - Debugger - Compiler
- d) Editor-Compiler-Debugger
- e) Interpreter - Tracer - Projektgenerator
- f) Gui-Editor-Editor-Compiler

6. Bei der Auswahl einer Programmiersprache haben Sie die Wahl zwischen einer Compiler- und einer Interpreter-Version. Welche Aussage trifft zu?

- a) Compiler haben eine längere Programmausführungszeit.

--

- b) Der Compiler kann semantische Fehler finden.
- c) Der Interpreter verarbeitet den Quelltext während der Laufzeit des Programms.
- d) Interpreter erzeugen als Ergebnis einen Objektcode.
- e) Compilerprogramme können schlecht getestet werden.

3. Erläutern Sie folgende Begriffe und nennen Sie jeweils zwei Beispiele.

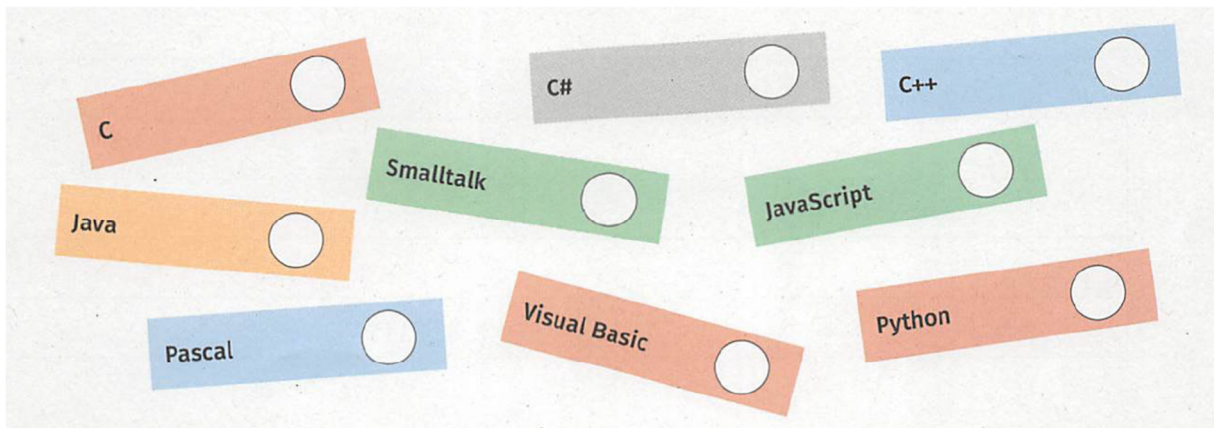
IDE:

Versionsverwaltung:

Framework:

4. Ordnen Sie den Programmiersprachen folgende Programmierparadigmen zu:

- (A) Imperatives (prozedurales) Programmierparadigma
- (B) Objektorientiertes Programmierparadigma
- (C) Imperatives (prozedurales) und objektorientiertes Programmierparadigma



Sie beschäftigen sich in Ihrer Ausbildung mit Programmiersprachen. Zur Übersetzung des Quelltextes kommen sowohl Compiler als auch Interpreter zum Einsatz. Bei der Programmiersprache „Java“ fällt Ihnen auf, dass in diesem Prozess erst ein Compiler und dann ein Interpreter zum Einsatz kommt. Dadurch hebt sich Java von anderen Programmiersprachen ab. Bei der Programmiersprache „C#“ kommt auf der anderen Seite ein JIT-Compiler zum Einsatz.

Wiederholen und erweitern Sie Ihr Wissen über Compiler und Interpreter.

5. Beschreiben Sie die Arbeitsweise von Compilern und Interpretern. Nennen Sie jeweils einen Vor- und einen Nachteil.

Compiler:

Interpreter:

6. Erkundigen Sie sich im Internet über JIT-Compiler und notieren Sie ein Beispiel.

NOTIZEN

7. Erkundigen Sie sich im Internet, wie Java-Programme übersetzt und ausgeführt werden. Notieren Sie sich den Weg und die Werkzeuge, welche dabei zum Einsatz kommen.

NOTIZEN

Unter den Auszubildenden ist eine lebhafte Diskussion über die „beste“ IDE oder den „besten“ Texteditor ausgebrochen. Am Ende dieser hitzigen Diskussion wurde beschlossen, dass jeder seinen „Favoriten“ objektiv den anderen präsentiert und sich im Anschluss daran kritischen Fragen stellt.

8. Stellen Sie eine IDE oder einen Texteditor vor.

Präsentieren Sie eine IDE oder einen Texteditor, welchen Sie aktuell zur Softwareentwicklung benutzen oder bereits benutzt haben. Beleuchten Sie dabei die Vorzüge dieses Tools und gehen Sie auch auf die Nachteile ein. Beantworten Sie in Vorbereitung dazu u.a. folgende Fragen:

Name der IDE oder des Texteditors	
Aktuelle Version	
Für welche Betriebssysteme ist sie/er verfügbar?	
Seit wann ist dieses Werkzeug in Nutzung?	
Wie viel Erfahrungen haben Sie damit schon gesammelt?	
Wurde bereits Erfahrung mit anderen IDE OderTexteditoren gemacht? Wenn ja, mit welchen?	
Welche IDE oder Texteditoren verwenden Kolleginnen und Kollegen?	
Vorzüge	
Nachteile	
Warum wird diese IDE oder dieser Texteditor bevorzugt?	

Tag 8 - Den Prozess der Anforderungsspezifikation und des Softwareentwurfes beschreiben

Sie informieren sich über die Anforderungsspezifikation und wie man diese Anforderungen zum Entwurf der Software verwendet.

Bevor über das Vorgehensmodell oder die zu verwendende Programmiersprache entschieden wird, müssen zunächst die Anforderungen an die zukünftige Software genau spezifiziert werden. Dabei spielt es keine Rolle, ob ein Problem für einen externen Auftraggeber gelöst oder eine eigene Idee umgesetzt werden soll.

Eines der wichtigsten Instrumente für eine gute Planung ist die Anforderungsspezifikation. In diesem Zusammenhang spielt, vor allem im deutschsprachigen Raum, das sog. Lasten- und Pflichtenheft eine große Rolle. Im Anschluss an die Anforderungsspezifikation erfolgt der Entwurf oder das Design der Software. Welche Prozessschritte beim Entwurf einer Software durchlaufen werden, wird in diesem Kapitel näher betrachtet.



LE 1 - Anforderungen an eine Software spezifizieren

Anforderungsspezifikation

Der Prozess der Anforderungsspezifikation umfasst das Ermitteln, Analysieren, Spezifizieren und Validieren (Überprüfen) aller Eigenschaften und Rahmenbedingungen eines Softwaresystems.

Das IEEE (Institute of Electrical and Electronic Engineers) hat den Prozess der Anforderungsspezifikation von Softwaresystemen in der IEEE 29148-2018 standardisiert. Je nach Art und Umfang des Projektes können Einzelpersonen bis hin zu ganzen Projektteams in diesen Prozess involviert sein. Die Aktivitäten in diesem Prozess sind nicht starr und laufen in aller Regel auch nicht linear ab, sondern sind durch Rücksprünge und Wiederholungen gekennzeichnet. Unabhängig von der Anforderungsart und den an diesem Prozess Beteiligten sollten die Anforderungen sehr präzise formuliert werden, um spätere Unstimmigkeiten zu vermeiden. Allgemeingültige Kriterien bei der Formulierung der Anforderungen sind:

Korrektheit: Die Anforderung muss sehr genau den Bedürfnissen des Auftraggebers entsprechen. Außerdem muss die Gesamtheit der Anforderungen vollständig sein.

Eindeutigkeit: Die Formulierung der Anforderung erlaubt nur eine gültige Interpretation. Es darf zu keinen Überschneidungen kommen und Anforderungen dürfen sich nicht widersprechen.

Prüfbarkeit: Es muss Möglichkeiten geben, um nachzuweisen, ob die Anforderung erfüllt ist oder nicht. **Nachverfolgbarkeit:** Der Umsetzungsprozess der Anforderung muss nachvollziehbar sein. Die Anforderungen als solche lassen sich in verschiedene Kategorien einteilen.

Anforderungen			
Funktionale Anforderungen		Nicht funktionale Anforderungen	
		Qualitätsanforderungen	Rahmenbedingungen

Funktionale Anforderungen

Funktionale Anforderungen beziehen sich darauf, was das System leisten soll. Diese sind von Software zu Software sehr unterschiedlich und sehr spezifisch. In den funktionalen Anforderungen wird beschrieben, welche einzelnen Funktionen und Funktionskomplexe die zu entwickelnde Software haben soll. Die dazu benötigten Eingabedaten, zwingend erforderliche Verarbeitungsschritte und die erwarteten Ausgaben werden dazu genau spezifiziert.

Nicht funktionale Anforderungen

Nicht funktionale Anforderungen beziehen sich darauf, wie das System seine Leistungen erbringen soll. Sie werden in der Anforderungsspezifikation genau wie die funktionalen Anforderungen beschrieben. Allerdings unterliegen manche dieser Anforderungen einer gewissen Subjektivität und deren Umsetzung und die Abnahmekriterien lassen sich schwerer beschreiben. Bei den nicht funktionalen Anforderungen wird zwischen Qualitätsanforderungen und Randbedingungen unterschieden. Die Qualitätsanforderungen werden dabei meistens an den Softwarequalitätskriterien ausgerichtet. Leider wird die Bedeutung der nicht funktionalen Qualitätsanforderungen meistens unterschätzt. Dabei spielen sie eine entscheidende Rolle, hinsichtlich der Kundenzufriedenheit. Wenn diese Anforderungen nicht spezifiziert wurden, dann wird meistens zu spät bemerkt, dass das System zu umständlich zu bedienen ist oder die Performance nicht ausreicht.

Nichtfunktionale Qualitätsanforderungen	
Benutzbarkeit	Hier werden die Anforderungen an die Benutzerfreundlichkeit der Software definiert. Dabei wird auch berücksichtigt, wie gut der Umgang mit der Software zu erlernen ist. In den meisten Fällen wird auch schon eine Skizze der grafischen Oberfläche angefertigt.
Zuverlässigkeit	Hier werden beispielsweise Anforderungen beschrieben, welche Fehlertoleranz die Software haben muss oder wie bei einem Softwareversagen die Daten und alten Zustände wiederhergestellt werden können.
Effizienz	Hier werden die Anforderungen an das Zeit- und Verbrauchsverhalten der Software definiert.
Änderbarkeit	Hier werden die Anforderungen beschrieben, wie gut und schnell die Software in der Zukunft an neue Anforderungen angepasst werden kann.
Übertragbarkeit	Hier werden Anforderungen beschrieben, wie schnell und mit wie viel Aufwand die Software auf neue Umgebungen übertragen werden kann.

Weitere wesentliche nicht funktionale Anforderungen werden durch die Rahmenbedingungen vorgegeben und müssen damit Beachtung finden. Diese Faktoren sind kaum beeinflussbar, schränken aber die möglichen Lösungen in den meisten Fällen ein.

Rahmenbedingungen	Beispiele
Technologisch	Vorhandene, technische IT-Infrastruktur (Plattformen, Schnittstellen, Nachbarsysteme usw.)
Organisatorisch	Aufbau und Prozessabläufe der Abteilungen, welche die Software nutzen sollen, unverrückbare Projekttermine
Normativ	Einzuhaltende Gesetze, Verordnungen oder Normen, kulturelle Aspekte: Sprache, Gebräuche, Traditionen

LE 2 - Lasten- und Pflichtenheft unterscheiden

Sie sollen sich grundlegende Kenntnisse über das Lasten- und Pflichtenheft erarbeiten und dessen Inhalte unterscheiden.

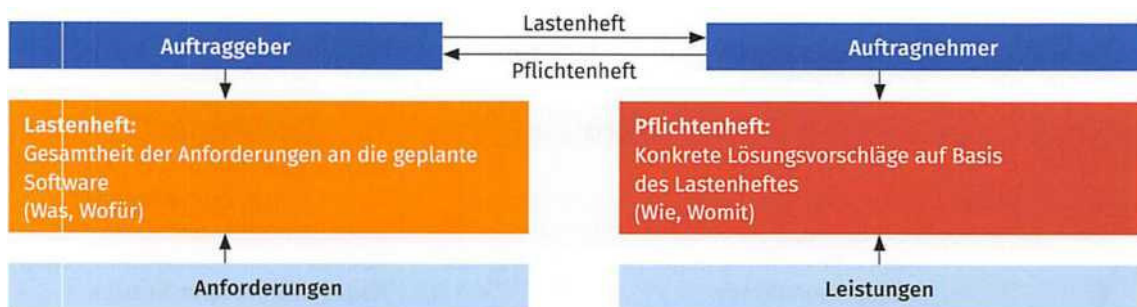
Im deutschsprachigen Raum werden für die Beschreibung der Anforderungen oft Lasten- und Pflichtenhefte verwendet. Die Begriffe „Lasten- und Pflichtenheft“ sind in der deutschen Norm zum Projektmanagement DIN 69901-5 definiert. Die Gliederung und die Inhalte sind aber nicht vorgeschrieben. In der Regel hat jedes Unternehmen seinen eigenen Standard.

Beschreibung		Synonyme
Lastenheft (User Specification)	„ Was und Wofür “ Vom Auftraggeber festgelegte Gesamtheit der Forderungen an die Lieferungen und Leistungen eines Auftragnehmers innerhalb eines (Projekt-)Auftrags.	Leistungsverzeichnis, Anforderungskatalog, Produktskizze, Kundenspezifikation, Customer Requirements Specification (CRS), Requirements Specification
Pflichtenheft (Functional Specification)	„ Wie und Womit “ Vom Auftragnehmer erarbeitete Realisierungsvorgaben auf der Basis des vom Auftraggeber vorgegebenen Lastenheftes.	Systemspezifikation, Fachspezifikation, Sollkonzept, Gesamtsystemspezifikation, System Requirements Specification (SRS), Design Specification

Beispiel für Inhalte und Gliederung	
Lastenheft	Pflichtenheft
- Ausgangssituation und Zielsetzung - Ausgangssituation - Zielsetzung - Funktionale Anforderungen - Muss-Kriterien - Kann-Kriterien - Abgrenzungskriterien - Nicht funktionale Anforderungen - Einsatzumgebung Hardware - Einsatzumgebung Software - Einsatzumgebung Orgware - Datenvolumen - Verarbeitungsgeschwindigkeit - Lebenszyklus des Gesamtsystems - Zeitpunkt des Einsatzes - Abzulösende Systeme - Zukünftige Anforderungen	- Ausgangssituation und Zielsetzung - Ausgangssituation - Zielsetzung - Funktionale Anforderungen - Softwarearchitektur, Leistungen der Komponenten - Anwendungsfälle (Use Cases) - Datenmodell - Abgrenzungskriterien - Nicht funktionale Anforderungen - Einsatzumgebung Hardware - Einsatzumgebung Software - Einsatzumgebung Orgware - Datenvolumen - Benutzerführung - Verarbeitungsgeschwindigkeit - Lebenszyklus des Gesamtsystems

5. Lieferumfang 5.1. Zu übergebende Dokumente 5.2. Softwarebereitstellung 5.3. Installation und Datentransfer 5.4. Schulungsleistungen 5.5. Supportleistungen 6. Abnahmekriterien 6.1. Qualitätsziele 6.2. Testszenarien	5. Schnittstellenübersicht 6. Lieferumfang 6.1. Zu übergebende Dokumente 6.2. Softwarebereitstellung 6.3. Installation und Datentransfer 6.4. Schulungsleistungen 6.5. Supportleistungen 7. Abnahmekriterien 7.1. Qualitätsziele 7.2. Testszenarien
--	--

Der Aufbau eines Lastenheftes ist in der Regel immer ähnlich. Der Inhalt unterscheidet sich jedoch stark, je nachdem, ob es sich um ein reales Produkt, eine Dienstleistung oder eine Software handelt. Vor dem eigentlichen Projektstart, d.h. während der Initialisierungs- und Definitionsphase, wird auf Basis des Lastenheftes ein Pflichtenheft erstellt. Dieses Pflichtenheft ist die Grundlage des Abnahmekatalogs und führt zum Kundenvertrag. Erst mit Unterzeichnung des Kundenvertrags startet das Projekt. Das Pflichtenheft kann jedoch im Laufe des Projekts weiter ergänzt werden.



Ein idealtypischer Ablauf von der Erstellung des Lastenhefts über die Erstellung des Pflichtenheftes bis zum Projektstart kann in folgenden Einzelschritten ablaufen.

Idealtypischer Ablauf bis zum Projektbeginn

1. Der Auftraggeber erstellt das Lastenheft.
2. Der Auftraggeber übergibt das Lastenheft an den Auftragnehmer.
3. Der Auftragnehmer überprüft das Lastenheft und übergibt es dem Projektmanagement.
4. Das Projektmanagement erstellt ein Pflichtenheft.
5. Das Pflichtenheft wird dem Auftraggeber übergeben.
6. Auf der Basis der Angaben im Pflichtenheft wird ein Vertrag durch den Auftragnehmer erstellt.
7. Der Vertrag wird an den Auftraggeber weitergereicht.
8. Der Vertrag wird vom Auftraggeber geprüft, unterzeichnet und zurückgegeben.
9. Nach Freigabe des Vertrags wird der Projektauftrag freigegeben und das Projekt wird gestartet.

In der Praxis läuft dieser Vorgang aber nicht ganz so reibungslos und linear ab. Es müssen viele Details geklärt werden und so finden viele Rücksprünge und Wiederholungen statt.

LE 3 - Den Entwurfsprozess beschreiben

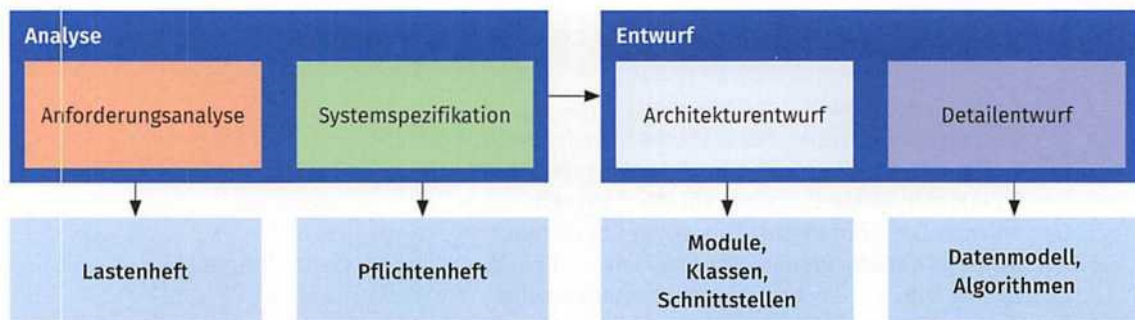
Sie sollen die Aufgaben des Entwurfsprozesses präsentieren.

Der Softwareentwurf gehört zu den Hauptaktivitäten der Softwareentwicklung. Nach der Problemanalyse und der Anforderungsspezifikation wird hierbei die Grundlage für die Implementierung der Software geschaffen.

Softwareentwurf

Nach IEEE ist unter einem Softwareentwurf ein Prozess zu verstehen, in dem die Architektur, die Komponenten, die Schnittstellen und andere Merkmale eines Softwaresystems definiert werden.

Dabei baut der Entwurfsprozess auf den Ergebnissen der Anforderungsspezifikation auf, z. B. dem Lasten- und Pflichtenheft. Der Schwerpunkt der Aufgaben beim Softwareentwurf liegt auf der Festlegung der Architekturbausteine (z.B. Module, Klassen oder Softwarekomponenten) und deren Beziehungen zueinander, der Spezifikation von Schnittstellen, der Modellierung eines Datenmodells und dem Entwurf von einzelnen Funktionen und Algorithmen.



Der Entwurf der Software erfolgt unabhängig von der gewählten Programmiersprache. Erst in den nachfolgenden Schritten wird der Entwurf nach und nach verfeinert und an die spezifischen Belange der gewählten Programmiersprache angepasst. Beim Entwurf kommen die unterschiedlichsten Modellierungssprachen zum Einsatz. Zum Beispiel wird UML, insbesondere das Klassendiagramm, zur Modellierung von objektorientierten Programmen eingesetzt oder zum Entwurf des Datenmodells für relationale Datenbanken das ERM.

Lernzielkontrollen Tag 8

1. Welche Aussagen sind richtig?
 - a. Der Prozess der Anforderungsspezifikation erfolgt in streng abgegrenzten Schritten.
 - b. Die Anforderungen müssen eindeutig formuliert sein.
 - c. Es werden funktionale und nicht funktionale Anforderungen unterschieden.
 - d. Qualitätsanforderungen sind nicht besonders wichtig, können aber vollständigkeitshalber erwähnt werden.
 - e. Die Fehlertoleranz gehört zu den nicht funktionalen Anforderungen.
 - f. Anforderungen an die Effizienz gehören zu den funktionalen Anforderungen.
 - g. Kulturelle Aspekte spielen bei der Anforderungsanalyse keine Rolle.

2. Erläutern Sie, was eine Anforderungsspezifikation ist und welche allgemeingültigen Kriterien bei der Erstellung berücksichtigt werden sollten.
3. Erläutern Sie folgende nicht funktionale Qualitätsanforderungen und finden Sie zu jedem Punkt mindestens zwei konkrete Beispiele.

Nicht funktionale Qualitätsanforderungen	
Benutzbarkeit	
Zuverlässigkeit	
Effizienz	
Änderbarkeit	
Übertragbarkeit	

4. Bestimmen Sie, ob die nachfolgenden Rahmenbedingungen (1) organisatorisch, (2) technologisch oder (3) normativ sind. Tragen Sie dazu die entsprechende Nummer in den Kreis ein.

Die OpenGL-Grafikschnittstelle ist Standard in der Firma.

Die kulturellen Aspekte der Sorben sollen berücksichtigt werden.

Die Software soll von zehn Abteilungen der Firma und dem Vertrieb benutzt werden.

Die Reifen-EU-Norm muss eingehalten werden.

Verbindung zu Auto CAD ist erforderlich.

Es besteht in der Firma eine IT-Abteilung, welche später die Wartung der Software übernehmen soll.

Das Programm muss sich in die vorhandene Apple-Infrastruktur problemlos einfügen.

Die Firma orientiert sich an den Ferienregelungen der einzelnen Bundesländer.

Die DSGVO muss berücksichtigt werden.

Die Firma ist auch in Mittel- und Südamerika aktiv.

Die Importmöglichkeit von Word-dokumenten besteht bereits.

Der Messetermin am 30. März 20.. muss unbedingt gehalten werden.

Die Firma ist in ganz Europa vertreten.

Es sind nur Linux-Server vorhanden.

Der bisherige Quellcode liegt in Python vor.

5. Übersetzen Sie den englischen Text sinngemäß ins Deutsche.

What is the System Requirements Specification (SRS)?

Every software has specific goals and serves particular purposes. Each goal and purpose translate a process or several processes that the software aims to solve or to automate. To deliver the right software product, we should define well the software from the beginning. System requirement specification or SRS frameworks software development, it documents every operation and dictates how software should behave, it can be as detailed as what a button should do and should be as complete and correct as possible. The purpose of a specification document is to describe the behaviour as well as the different functionalities of an application or software in a specific environment.

6. Welche Aussagen sind richtig?

- a. Das Lastenheft wird vom Auftraggeber erstellt.
- b. Das Pflichtenheft wird vom Auftragnehmer erstellt.
- c. Das Lastenheft wird auf Grundlage des Pflichtenheftes erstellt.
- d. Nicht funktionale Anforderungen werden nur im Pflichtenheft formuliert.
- e. Das Pflichtenheft enthält konkrete Lösungsvorschläge.
- f. Das Pflichtenheft ist die Grundlage für den Vertrag zwischen dem Auftraggeber und dem Auftragnehmer.

7. Beschreiben Sie, was ein Pflichten- und Lastenheft ist und wer diese erstellt.

Pflichtenheft:

Lastenheft:

8. Ausgangssituation: Es soll ein Programm entwickelt werden, mit dem die Daten von Schiffen und deren Ladungen verwaltet werden können. Nennen Sie jeweils fünf Inhalte, welche Pflichten- und Lastenhefte enthalten können. Ordnen Sie jedem Inhalt ein konkretes Beispiel zu, welches zur Ausgangssituation passen könnte.

Lastenheft	Pflichtenheft
1.	
2.	
3.	
4.	
5.	

9. Führen Sie eine Anforderungsanalyse durch und erstellen Sie ein Pflichtenheft.

Führen Sie für eine selbstgewählte Aufgabe aus dem Bereich Softwareentwicklung mithilfe eines IPERKA-Plans (siehe Anhang) eine Anforderungsspezifikation durch und erstellen Sie danach ein Lasten- und ein Pflichtenheft. Die Aufgabe sollte als Kernziel eine Verwaltung von Daten beinhalten. Dabei können das Thema und die Daten, welche verwaltet werden, frei bestimmt werden. Diese können aus dem beruflichen oder dem privaten Umfeld kommen. Bilden Sie dazu Selbstständig Teams.

Entscheiden Sie sich im Team für eine Aufgabenstellung.

Beschreiben Sie die Ausgangssituation und die Rahmenbedingungen. Machen Sie sich dazu im nachfolgenden Formular Notizen.

Erstellen Sie auf Grundlage der Notizen in einem extra Dokument ein Lasten- und ein Pflichtenheft. Präsentieren Sie Ihre Ergebnisse den anderen Teams und diskutieren Sie diese.

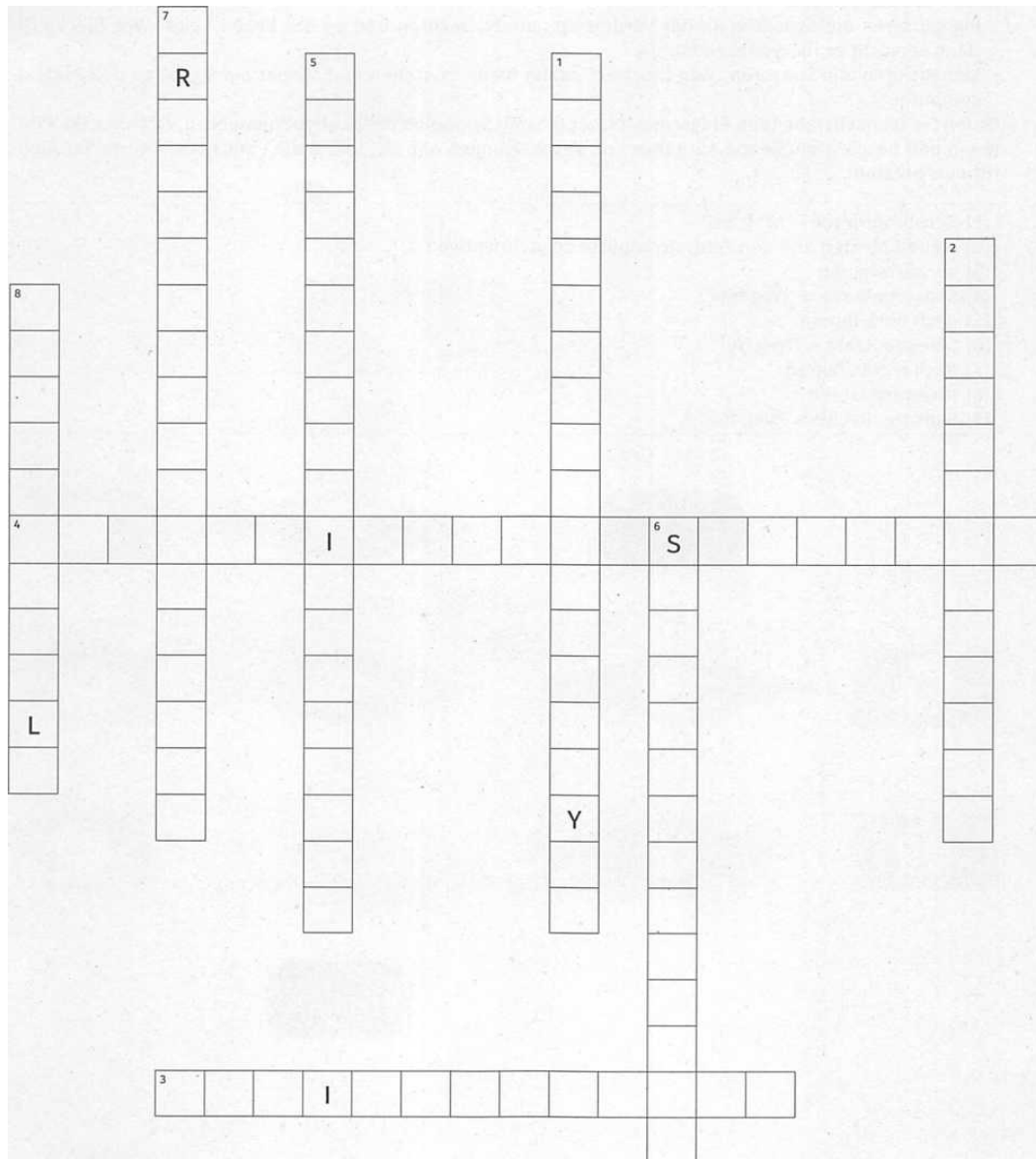
Stichwortsammlung	
Ausgangssituation	
Rahmenbedingungen	
Funktionale Anforderungen	
Nicht funktionale Anforderungen	
Schnittstellen	
Lieferumfang	
Abnahmekriterien	

10. Welche Aussagen sind richtig?

- a. Beim Entwurf einer Software wird u.a. die Architektur der Software festgelegt.
- b. Mit dem Design der Software kann schon vor der Analysephase begonnen werden.
- c. Die Entwicklung von Algorithmen gehört zum Detailentwurf.
- d. Beim Softwareentwurf kommen Modellierungssprachen zum Einsatz.

11. Lösen Sie folgendes Kreuzworträtsel.

- 1. Ist der erste Schritt im Analyse- und Entwurfsprozess.
- 2. In dieser Phase werden z. B. Algorithmen entworfen.
- 3. Grundlage für dieses Dokument bildet das Lastenheft.
- 4. Begriff (Mehrzahl) für alle Beschreibungsmittel (z. B. UML), welche u.a. im Entwurfsprozess zum Einsatz kommen.
- 5. Wird u.a. im Pflichtenheft beschrieben.
- 6. Architekturbaustein (Mehrzahl) einer Software.
- 7. In dieser Entwurfsphase wird u.a. die Modulstruktur festgelegt.
- 8. Wird im Detailentwurf festgelegt.



Tag 9 - Modellierungssprachen unterscheiden

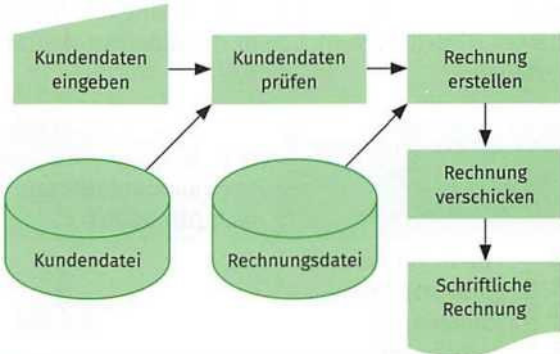
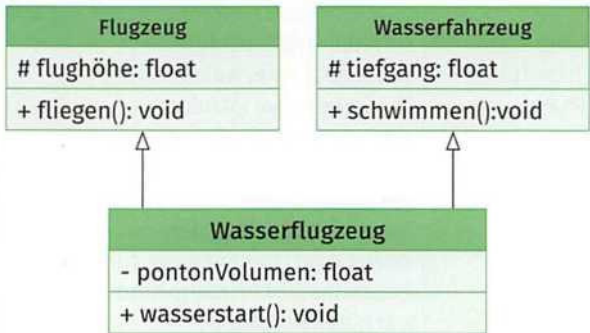
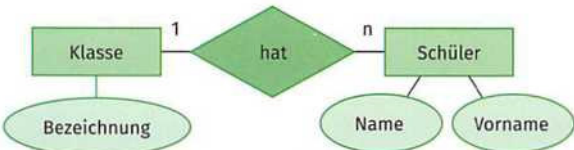
Sie diskutieren Möglichkeiten, den Entwurf und die Dokumentation von Software mithilfe von Beschreibungsmitteln zu unterstützen.

Sowohl für die Planung und Dokumentation von Softwareprojekten als auch für die Kommunikation zwischen Auftraggeber und Entwickler hat es sich als vorteilhaft erwiesen, Sachverhalte mithilfe von grafischen Modellen zu beschreiben und darzustellen. Damit ist es möglich, eigene Gedankengänge kompakt und übersichtlich dem Kommunikationspartner zu vermitteln, was wiederum die Kommunikation sehr erleichtert. Außerdem werden Missverständnisse bei der Planung und Dokumentation erheblich reduziert.



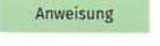
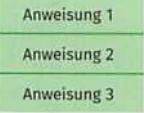
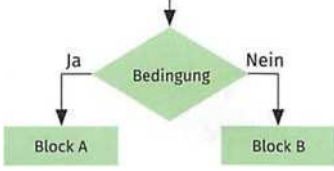
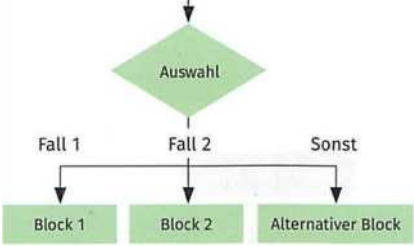
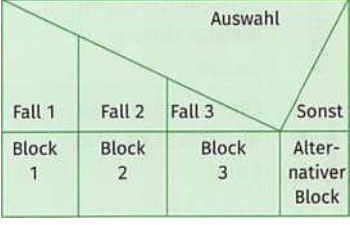
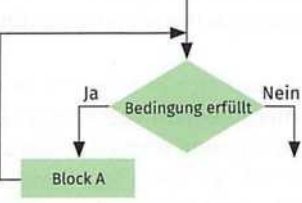
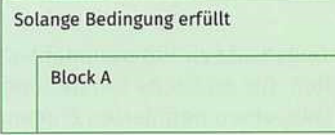
Um grafische Modelle für die Softwareentwicklung zu erstellen, wurden verschiedene grafische Beschreibungsmittel entwickelt, welche sich weit verbreitet haben und meistens auf wenigen eindeutigen Symbolen basieren. Für einige wurden sogar DIN-Normen entwickelt. Andere dieser Beschreibungsmittel sind bereits veraltet und werden nach und nach von modernen Modellierungssprachen abgelöst. Nachfolgend sind einige grafische Modelle für die Planung und Dokumentation von Software aufgeführt.

Grafisches Beschreibungsmodell		
Programmablaufplan (PAP)	Darstellungsform für Programmabläufe, welche für die imperative (strukturierte) Programmierung entwickelt wurde. Es wird auch als Flussdiagramm oder Programmstrukturplan bezeichnet. Die Symbole für Programmablaufpläne sind nach der DIN 66001 genormt.	<pre> graph TD Start([Start]) --> Init[i := 0] Init --> Cond{i < 10} Cond -- ja --> Output[Ausgabe i] Output --> Inc[i := i + 1] Inc --> Cond Cond -- nein --> End([Ende]) </pre>
Struktogramm	Es dient zur Darstellung von Programmentwürfen im Rahmen der imperativen (strukturierten) Programmierung. Es wird nach seinen Entwicklern auch als Nassi-Schneiderman-Diagramm bezeichnet. Die Symbole sind nach der DIN 66261 genormt.	<pre> graph TD Loop1[Solange a>0 UND b>0] Loop2[a>b] Loop3[b=0] Loop1 -- ja --> Loop2 Loop2 -- ja --> AssignA[a:=a-b] Loop2 -- nein --> AssignB[b:=b-a] AssignA --> Loop3 AssignB --> Loop3 Loop3 -- ja --> OutputA[Ausgabe a] Loop3 -- nein --> OutputB[Ausgabe b] </pre>

Grafisches Beschreibungsmodell																																									
Datenfluss-plan	Diese alte Darstellungsform stellt den Fluss von Daten durch ein Programmsystem dar und kommt heutzutage in der betrieblichen Praxis selten bis gar nicht vor.																																								
Entscheidungs-tabellen	Sie werden eingesetzt, um Abhängigkeiten zwischen verschiedenen Bedingungen und den jeweils auszuführenden Aktionen übersichtlich darzustellen. Sie unterstützen beispielsweise den Entwurf von Computerprogrammen oder speicherprogrammierbaren Steuerungen (SPS). Entscheidungstabellen sind in der DIN 66241 genormt.	<table><tr><th rowspan="2">Entscheidungs-tabelle</th><th colspan="4">Regeln</th></tr><tr><th>R1</th><th>R2</th><th>R3</th><th>R4</th></tr><tr><td>Bedingungen</td><td></td><td></td><td></td><td></td></tr><tr><td>Es regnet</td><td>J</td><td>J</td><td>N</td><td>N</td></tr><tr><td>Es ist kalt</td><td>J</td><td>N</td><td>J</td><td>N</td></tr><tr><td>Aktionen</td><td></td><td></td><td></td><td></td></tr><tr><td>Regenschirm mitnehmen</td><td>x</td><td>x</td><td>-</td><td>-</td></tr><tr><td>Jacke anziehen</td><td>x</td><td>-</td><td>x</td><td>-</td></tr></table>	Entscheidungs-tabelle	Regeln				R1	R2	R3	R4	Bedingungen					Es regnet	J	J	N	N	Es ist kalt	J	N	J	N	Aktionen					Regenschirm mitnehmen	x	x	-	-	Jacke anziehen	x	-	x	-
Entscheidungs-tabelle	Regeln																																								
	R1	R2	R3	R4																																					
Bedingungen																																									
Es regnet	J	J	N	N																																					
Es ist kalt	J	N	J	N																																					
Aktionen																																									
Regenschirm mitnehmen	x	x	-	-																																					
Jacke anziehen	x	-	x	-																																					
Unified Modeling Language (UML)	UML ist eine grafische Modellierungssprache, welche zur Planung von objekt-orientierter Software eingesetzt wird. Sie ist heute das am meisten eingesetzte Modellierungsmittel für die Softwaresystemmodellierung. Sie wird von der Object Management Group (OMG) entwickelt und ist in der ISO/IEC 19505 genormt.																																								
Entity-Relationship-Modell (ERM)	ERM ist ein in der Praxis weitverbreiteter Standard für die Datenmodellierung. Es bildet meist die Grundlage für die Entwicklung einer relationalen Datenbank.																																								

LE 1 - Programmablaufplan und Struktogramm

Der Programmablaufplan und das Struktogramm gehören zu den ältesten grafischen Modellen für die Softwareentwicklung. Beide sind durch eine DIN-Norm standardisiert. Die wichtigen Darstellungselemente beider Modelle lassen sich in die Ablaufstrukturen Verarbeitung, Folge, Auswahl und Wiederholung unterteilen.

Name Beschreibung		Symbol	
		Programmablaufplan nach DIN 66001	Nassi-Schneidermann-Struktogramm nach DIN 66261
Verarbeitung (Prozess)	Dient zur Darstellung von Anweisungen, Unterprogrammen		
Folge (Sequenz)	Aneinanderreihung von mehreren Anweisungen		
Auswahl, Alternative (Verzweigung)	Mit einer Verzweigung wird eine Bedingung dargestellt. Ist die Bedingung erfüllt, dann wird der Block A ausgeführt, ansonsten der Block B.		
Fallunterscheidung (Mehrfachauswahl)	Bei der Mehrfachauswahl wird aus mehreren Vorgaben ein Fall ausgewählt und der entsprechende Block abgearbeitet. Ist der Fall in den Vorgaben nicht vorhanden, dann wird ein Alternativblock durchlaufen.		
Wiederholung (kopfgesteuerte Schleife)	Ein Anweisungsblock wird wiederholt, solange die Schleifenbedingung erfüllt ist. Die Abfrage der Bedingung erfolgt vor der Verarbeitung. Ist die Bedingung schon bei der ersten Abfrage nicht erfüllt, dann wird die Schleife nicht durchlaufen.		

Name Beschreibung		Symbol	
		Programmablaufplan nach DIN 66001	Nassi-Schneidermann-Struktogramm nach DIN 66261
Wiederholung (fußgesteuerte Schleife)	Ein Strukturblock wird wiederholt, solange die Schleifenbedingung erfüllt ist. Die Abfrage der Bedingung schon vorher nicht erfüllt, wird diese Schleife trotzdem einmal durchlaufen.	<pre> graph TD Entry(()) --> A[Block A] A --> B{Bedingung erfüllt} B -- Ja --> Entry B -- Nein --> Exit(()) </pre>	
Grenzstelle	Dient zum Darstellen von Start und Ende		Nicht definiert

In Programmablaufplänen werden die Schritte symbolisch durch Sinnbilder dargestellt und der Steuerungsablauf durch Ablauflinien hergestellt. Der Steuerungsablauf kann beliebig gestaltet werden, da die Verwendung der Pfeile zur Darstellung des Steuerungsablaufes keinerlei Einschränkungen unterliegt. Daher sind auch Abläufe möglich, welche nicht den gängigen Regeln der strukturierten Programmierung entsprechen. Deshalb werden Programmablaufpläne im Bereich der Softwareerstellung nur noch selten verwendet. Pseudocode bietet einen ähnlichen Abstraktionsgrad, ist jedoch einfacher zu erstellen und in der Regel sehr viel einfacher zu verändern als ein Programmablaufdiagramm.

Im Gegensatz zu Programmablaufplänen lassen sich in Struktogrammen die Abläufe nicht beliebig darstellen. Die grafische Darstellung erfolgt hier in einem geschlossenen Block. Ziel dieser Darstellung soll es sein, einen definierten Eingangspunkt (oben) und einen definierten Ausgangspunkt (unten) zu haben. Bei der Entwicklung eines Struktogramms wird nach dem Top-Down-Prinzip ein komplexes Ausgangsproblem Ebene für Ebene in übersichtlichere Teilprobleme zerlegt. Auch Struktogramme kommen nur noch sehr selten zum Einsatz. Grund dafür ist die Verfügbarkeit komplexerer Beschreibungssprachen, z.B. UML. Trotzdem sind Struktogramme besonders beim Einstieg in die Programmierung sehr nützlich, um den Ablauf von Algorithmen zu verstehen oder selbst welche zu planen. Deswegen sind sie auch noch ein fester Bestandteil des Lehrstoffes an Schulen und Universitäten. Der Einsatz von

Programmablaufplänen und Struktogrammen wird nachfolgend anhand eines Beispiels dargestellt.

Beispielauftrag

Es soll ein Algorithmus für die automatische Berechnung eines Rabattes bei einer Online-Bestellung entwickelt werden. Es gelten folgende Rabattregelungen:

10% Preisnachlass, wenn der Bestellwert 100,00 € übersteigt

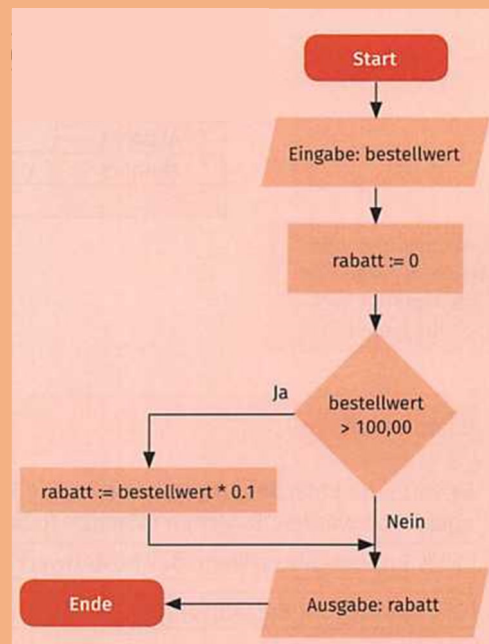
Die Eingabe ist der Bestellwert und die Ausgabe ist der gewährte Rabatt in Euro. Der Algorithmus soll als grafisches Modell dargestellt werden.

Lösung als Programmablaufplan

Jeder Programmablaufplan hat ein Start- und mindestens ein Ende-Symbol. Nach der Eingabe des Bestellwertes wird zunächst der Rabatt auf 0 gesetzt. Danach wird überprüft, ob der Wert der Bestellung 100,00 € überschreitet. In diesem Fall werden vom Bestellwert 10% berechnet ($\text{Bestellwert} * 0.1$) und als Rabatt gespeichert. Wenn der Bestellwert nicht über 100,00 € liegt, wird das Programm einfach weiter abgearbeitet. Zum Schluss wird der Rabatt ausgegeben.

Lösung als Struktogramm

Im Gegensatz zum Programmablaufplan hat das Struktogramm kein Start- und Ende-Symbol, da Einstiegs- und Ausstiegspunkte klar definiert sind. Nach der Eingabe des Bestellwertes wird zunächst der Rabatt auf 0 gesetzt. Danach wird überprüft, ob der Wert der Bestellung 100,00 € überschreitet. In diesem Fall werden vom Bestellwert 10% berechnet ($\text{Bestellwert} * 0.1$) und als Rabatt gespeichert. Wenn der Bestellwert nicht über 100,00 € liegt, wird das Programm einfach weiter abgearbeitet. Ein leerer Nein-Zweig wird durch ein „%“ oder durch vollständiges Leerlassen des Zweiges gekennzeichnet. Am Ende wird der Rabatt ausgegeben.



LE 2 - Entscheidungstabelle

Entscheidungstabellen sind eine weitere Möglichkeit, komplexe Sachverhalte übersichtlich darzustellen. Sie finden überall dort Anwendung, wo eine rein verbale Beschreibung des Systems zu ungenau, unvollständig und mehrdeutig ist. So werden Entscheidungstabellen etwa beim Design von Softwaresystemen oder im Zusammenhang mit speicherprogrammierbaren Steuerungen (SPS) eingesetzt.

Entscheidungstabellen beschreiben immer Sachverhalte, bei denen eine Folge von Bedingungen zu gewissen Aktionen führen. Diese „Vorschriften“, unter welchen Bedingungen welche Aktionen erfolgen, werden in der Entscheidungstabelle „Regeln“ genannt. Nachfolgend ist der grundlegende Aufbau einer Entscheidungstabelle dargestellt.

Hier werden die Bedingungen aufgelistet, z.B.: Ist das Material verfügbar?

Entscheidungstabelle		Regeln							
Bedingungen	R1	R2	R3	R4	R5	R6	R7	R8	
Bedingung 1	J	J	J	J	N	N	N	N	
Bedingung 2	J	J	N	N	J	J	N	N	
Bedingung 3	J	N	J	N	J	N	J	N	
Aktionen									
Aktion 1	-	x	-	-	x	-	-	x	
Aktion 2	-	-	x	-	x	x	x	-	
Aktion 3	-	-	-	x	x	x	-	x	
...									

Hier werden alle Aktionen aufgelistet, z.B. Material nachbestellen.

Hier werden die Regeln festgelegt. Dabei steht ein „J“ für Bedingung erfüllt und ein „N“ für Bedingung nicht erfüllt. Alternativ kann auch „0“ und „1“ verwendet werden.

In diesem Bereich wird durch ein „x“ gekennzeichnet, welche Aktion unter welchen Bedingungen ausgeführt wird.

Beispielauftrag

Es soll eine Entscheidungstabelle für die Bestimmung eines Rabattes bei einer Online-Bestellung entwickelt werden. Es gelten folgende Rabattregelungen:

10% Preisnachlass, wenn der Bestellwert 100,00 € übersteigt

3% Rabatt vom Bestellwert bei Vorkasse

Lösung als Entscheidungstabelle

Für die Beispielaufgabe in diesem Abschnitt würde die Entscheidungstabelle aus zwei Bedingungen und drei Aktionen bestehen. Aus der Anzahl der Bedingungen ergibt sich auch die Anzahl der Regeln. Diese ermittelt sich aus der Formel:

Anzahl Regeln = 2^{Anzahl Bedingungen}

Damit hat die Entscheidungstabelle der Beispielaufgabe vier Regeln und sieht wie folgt aus:

Entscheidungstabelle	Regeln			
Bedingungen	R1	R2	R3	R4
Bestellwert > 100,00 €	J	J	N	N
Zahlungsart: Vorkasse	J	N	J	N
Aktionen				
kein Rabatt	-	-	-	x
Rabatt: 10% vom Bestellwert	x	x	-	-
Rabatt: 3% vom Bestellwert	x	-	x	-

LE 3 - Pseudocode

Der Pseudocode ist die umgangssprachliche Beschreibung eines Programmablaufs, der sich syntaktisch an eine Programmiersprache anlehnt. Pseudocode ist nicht genormt und kann deshalb relativ frei formuliert werden. In der Regel sollte sich die Syntax vom Pseudocode aber an die Programmiersprache anlehnen, in der das Programm geschrieben wird. Dabei können Variablen- und Konstanten-Deklarationen schon mit enthalten sein. Möchte man seinen Pseudocode stärker programmiersprachenunabhängig formulieren, dann kann man auch die Wörter der eigenen Muttersprache verwenden. Da erst in den nächsten Abschnitten die Syntax von Programmiersprachen behandelt wird, soll hier kurz ein programmiersprachenunabhängiger Pseudocode vorgestellt werden.

Struktur	Pseudocode	Beispiel
Verarbeitung	Anweisung1	durchschnitt := gesamt / anzahl
Folge (Sequenz)	Anweisung1 Anweisung2	anzahl := 20 gesamt := 200 durchschnitt := gesamt / anzahl
Auswahl (Verzweigung)	Wenn Bedingung dann sonst	Wenn anzahl > 30 dann x := anzahl *2 y := anzahl /2 sonst Ausgabe: „Fehler“ Ende Verzweigung
Wiederholung (kopfgesteuerte Schleife)	Wiederhole solange Bedingung	Wiederhole solange x < 10 Ausgabe: x x := x + 1 Ende Schleife
Wiederholung (fußgesteuerte Schleife)	Wiederhole solange Bedingung	Wiederhole Ausgabe: x x := x + 1 solange x < 10
Wiederholung (Zählschleife)	Von Anfangswert Bis Endwert	Von i:=0 Bis 10 Ausgabe: x Ende Schleife

Beispielauftrag

Es soll die Summe aller ganzen Zahlen, welche durch 3 teilbar sind, aus einem bestimmten Bereich berechnet und ausgegeben werden. Der Bereich beginnt mit der Zahl 1 und endet mit der Zahl 100. Es soll ein entsprechender Algorithmus in Pseudocode entwickelt werden.

Lösung als Pseudocode

```
summe := 0
Von zahl:=1 Bis 100
  Wenn zahl modulo 3=0 dann
    summe := summe + zahl
  Ende Verzweigung
Ende Schleife
Ausgabe: summe
```

Hinweis: Modulo liefert den ganzzahligen Rest einer Division

LE 4 - Unified Modeling Language (UML)

Die Unified Modeling Language (UML) ist eine moderne grafische Modellierungssprache, welche zur Beschreibung von Softwaresystemen eingesetzt wird. UML besitzt mehrere Diagrammtypen, um verschiedene Aspekte von Softwaresystemen zu beschreiben. Deshalb gilt UML auch als eine der umfassendsten Modellierungssprachen in der Softwareentwicklung.

Besonderen Einsatz findet UML im Rahmen der objektorientierten Programmierung und der erste Kontakt besteht häufig darin, dass einzelne UML-Diagramme im Rahmen von Softwareprojekten zu verstehen oder zu beurteilen sind. An dieser Stelle wird nur eine kurze Übersicht über die UML-Diagrammart gegeben. Eine ausführliche Beschäftigung mit UML erfolgt im zweiten Ausbildungsjahr.

UML-Diagrammart			
Verhaltensdiagramme		Strukturdiagramme	
Anwendungsfalldiagramm	Sequenzdiagramm	Klassendiagramm	Objektdiagramm
Zustandsdiagramm	Zeitverlaufdiagramm	Komponentendiagramm	Verteilungsdiagramm
Aktivitätsdiagramm	Kommunikationsdiagramm	Paketdiagramm	Profildiagramm
Interaktionsübersichtdiagramm		Kompositionsstrukturdiagramm	

Lernzielkontrollen Tag 9

1. Welche Aussagen sind richtig?
 - a. PAP und Struktogramm sind grafische Modellierungssprachen.
 - b. UML ist eine veraltete grafische Modellierungssprache.
 - c. Das ER-Modell findet bei der Planung von relationalen Datenbanken Anwendung.
 - d. Bei Struktogrammen werden die einzelnen Elemente durch Pfeile verbunden.
 - e. Entscheidungstabellen haben immer drei Bedingungen.
 - f. Pseudocode ist an die Programmiersprache Java angelehnt.
 - g. Das Klassen- und das Anwendungsfalldiagramm sind UML-Diagramme.
2. Entwickeln Sie einen Programmablaufplan, welcher das Betanken eines Autos an der Tankstelle darstellt.
3. Entwickeln Sie ein Struktogramm für folgende Aufgabenstellung: Nach Eingabe von drei Zahlen soll der höchste Wert dieser Zahlen ausgegeben werden.
4. Es soll ein Algorithmus entwickelt werden, welcher den Speicherbedarf für digitale Bilder berechnet und in MiB ausgibt. Eingabewerte sind: Anzahl der Bilder, Höhe und Breite eines einzelnen Bildes in Pixeln und die Farbtiefe pro Pixel in Bit. Erstellen Sie für den Algorithmus ein Struktogramm.
5. Ein Unternehmen beschließt, seinen Mitarbeitern Aktien zur Vorzugskonditionen anzubieten. Folgende Bezugsberechtigungen wurden festgelegt. Mitarbeiter mit mehr als zehn Jahren Betriebszugehörigkeit können 50 Aktien erwerben. Mitarbeiter, welche mehr als zwei Jahre zum Betrieb gehören, können max. 20 Aktien beziehen. Mitarbeiter, welche außertariflich bezahlt werden oder sich in einem gekündigten Arbeitsverhältnis befinden, haben in keinem Fall einen Anspruch auf den Erwerb der Aktien. Entwerfen Sie eine Entscheidungstabelle.

TAG 10 - Vertiefung des Wissens und Modulprüfung

LE 1 – Einleitung

Der letzte Tag in diesem Modul bietet Ihnen die Möglichkeit der Wiederholung und Vertiefung Ihres Wissens. Sollten es noch offene, nicht vollständig bearbeitete Modul-Inhalte geben, nutzen Sie den heutigen Tag zur Komplettierung.

LE 2 - Modulprüfung

120 Minuten zu denen wir Ihnen recht viel Erfolg wünschen!