**UML (Unified Modeling Language ):** Modellierungssprache für objektorientierte Systeme; Spezifikation, Konstruktion, Visualisierung, Dokumentation, keine Programmiersprache oder Entwicklungsmethode; Standard für objektorientierte Softwareentwicklung

**UML Modelle:** beschreiben Aspekt eines Softwaresystems mithilfe v. vollständiger Abstraktion dieses Aspektes, stehen untereinander in Beziehung

**Arten v. UML Modellen/ Diagramme:** Strukturmodelle (Klassen, Objekte, Komponenten,..); Verhaltensmodelle (Aktivitätsdiagramme, Zustandsdiagramme, Interaktionen, Zustandsübergänge, Anwendungsfälle,…) Sichten auf Modelle in grafischer Darstellungsform, Sammlung von Modellelementen

**Klassendiagramme:** Beschreibung der konzeptionellen Sicher der statischen Objektstruktur; Elemente sind Klassen (Repräsentanten ihrer Instanzen/ samt Attributen+ Methoden), Beziehungen zwischen Klassen (Vererbung), Beziehungen zwischen Instanzen v. K. (Assoziation), Eigenschaften dieser Elemente

**Klasseneigenschaften** (Klassifizierungseigenschaften)**: Generalisierung:** {incomplete}(nicht vollständig), {overlapping}(abgeleitet Klassen haben gemeinsame Unterklasse/ Gegenteil von disjoint), {complete}(Diagramm zeigt alles), {disjoint}(kann keine gemeinsame Klasse abgeleitet werden z.b. wenn oben Plattform steht und unten Windows und Unix!); **Assoziationen:** xor(?), subset(?); **Stereotypen** (Erweiterungen)**:** metaclass; **Kennzeichnungen:** {abstract}(darf keine Instanzen haben=> gut zum Aufbauen einer Struktur), {private}, {read only}, {add only}, {frozen}, {sorted}, {ordered}

**Datenmodellierung:** Realitätsausschnitt; Abstraktion realer Entitäten; Entwicklung eines Schemas; Interpretation der Realität; Entwurf von Datenbankschemata; (logische) Datenstrukturen

**Anwendungsmodellierung:** Entwurf von Anwendungs-(Geschäfts-)Modellen; Informationsstrukturen und Geschäftsprozesse

**Physische Datenunabhängigkeit:** Applikationsprogramme von physischen Speicherung der Daten unabhängig; internes Schema ändern ohne externe Schema zu beeinflussen; Trennung von interner und konzeptioneller Schicht

**Logische Datenunabhängigkeit:** Datenbasis ändern ohne Applikation signifikant zu beeinflussen; konzeptionelle Schema ändern ohne externe Schemata zu beeinflussen; Trennung von externen und konzeptioneller Schicht

**Datenmodelle:** diskreten Entitäten, die Eigenschaften haben, und zwischen denen Beziehungen sind

**Prinzip:** Entitätstypen mit Attributen, die durch Mengen von Attributen identifiziert werden; Beziehungstypen

**ER Modell:** Entitäten; Beziehungen; Attribute

**Entwurf:** Ergebnis einer Analysephase; Ergebnisse aus Folge von Interviews; schrittweise (iterativ), resultiert aus Folge von Schematransformationen (ausgehend von Startschema)

**EER Modell:** Zusammengesetzte Attribute; Generalisierungshierarchie

**Entitätstypen:** Klassen von Entitäten; z.b. Mann, Frau, Student; **Rechteck**

**Beziehungstypen:** Aggregate (Anzahl heißt Grad der Beziehung (minimale und maximale Kardinalität)) von mehreren an Beziehung teilnehmenden Entitäten; Entitäten spielen bestimmte Rollen; **Raute**

**Starke Entität:** existieren unabhängig von anderen Entitäten; haben mindestens einen internen Identifikator

**Schwache Entität:** existieren nur in Beziehung zu anderen Entitäten und können nur über Identifikation dessen bestimmenden Entität identifiziert werden; Instanz von „schwach" ist es existenziell wesentlich in Beziehung zu „stark" zu stehen; „schwach" ist nicht nur durch seien Attribute identifizierbar; haben nur externe oder gemischte Identifikatoren

**Attribute:** repräsentieren Eigenschaften oder Beziehungen der realen Welt; minimal und maximale Multiplizität; **Kreis**

**Generalisierung:** E= Generalisierung einer Menge von Entitätstypen $E_1, E_2 ..., E_n$ wenn jede Entität des Typen $E_i$ auch Entität von Typ E ist; **Pfeil** (Richtung allgemeinere Entität); $E_i$ erben alle Eigenschaften von E+ haben spezielle Eigenschaften.

**Überdeckungsarten v. Generalisierungen: Total/ Partiell:** Total (t) wenn E auf mindestens eine Entität von $E_i$ abgebildet wird (Person auf Frau/ Mann), sonst partiell (p)(Person auf Manager/ Mitarbeiter). **Exklusiv/ Überlappend:** exklusiv (e)(Person auf Frau/ Mann)wenn jede Entität von E auf höchstens ein $E_i$ abgebildet wird, sonst überlappend (o) (Manager auf Technischer Manager/ Administrativer Manager)

**Zusammengesetzte Attribute:** Attribut „Gruppe" die Gemeinsamkeiten bezüglich Bedeutung+ Verwendung haben; **Ellipse** (Oval)

**Identifikatoren**, Schlüssel: Eine Entität wird identifiziert durch eine Teilmenge ihrer eigenen Attribute und durch ihre in Beziehung stehenden Entitäten; Seien A1, …, An jeweils einwertige Attribute eines Entitätstyps E und E1, …, Em Entitätstypen, die mit E durch (1..1) - oder (n..1) - Beziehungen in Verbindung stehen, so dass jede Entität des Typs E genau einen Wert jedes Attributes Ai und je eine Entität der Typen Ej zugeordnet hat.; Eine Menge $I \subseteq \{A1, …, An, E1, …, Em\}$ $(i \leq n, j \leq m)$ ist ein Identifikator, wenn gilt:
(1) Der Identifikator- Wert ist für jede Entität vom Typ E eindeutig
(2) Keine Teilmenge von I hat diese Eigenschaft; **ausgefüllter Kreis**

**„Entwurfarten":** Top- Down (Grobstruktur=> in Entwurfsschritten Verfeinerung); Bottom-Up (unstrukturierte Einzelelemente=> viele Entwurfsschritte=> neue Strukturen; Konzepte; Eigenschaften); Inside- Out (Kernbereich=> Entwurfsschritte=> weitere Randbereiche einbeziehen)

**Objektorientiert**: Objekte (ko)existieren in Objektraum, kommunizieren miteinander, senden und empfangen Botschaften, Programme (Methoden) mit Objekttypen (Klassen) verbunden

**Datenorientierte Prozesse:** Prozeduren existieren unabhängig von Daten, Prozeduren (Programme) müssen bei Aktivierung mit Daten (sind von ihnen getrennt)versorgt werden und koordiniert!

**Objekte:** eindeutig Identifizierbar; veränderlichen Zustand (Instanzvariable (Namen für innere Objekte); Zustandsänderung durch Zuweisung von neuen Objekten an Instanzvariable oder Zustandsänderung der zugewiesenen Objekte); vorgegebenes Verhalten (reagieren auf Botschaften; empfangen Botschaften aktivieren Methoden die auf Zustand (Instanzvariable) zugreifen können)

Objekte (die einer Klasse angehören): sind Instanzen einer Klasse, durch Instanzierung der Klasse erzeugt, gemeinsame Struktur, individuelle Belegungen der Instanzvariable=> individuellen Zustand

**Klassen:** Zusammenfassungen v. Objekten gleicher Art; Menge gleichartiger Objekte (gleiche Struktur+ Verhalten); Konstruktionsbeschreibungen v. O., Einrichtungen zur Erzeugung v. O.; Charakterisierung v. O. Typen

**Vererbung:** Jede Klasse (außer Wurzelklasse) ist Unterklasse einer Oberklasse und erweitert Oberklasse; Instanzen der Unterklasse sind speziellere Objekte, besitzen Eigenschaften der Oberklasse+ neue (Spezialisierung)

**Klassenhierarchie** (Einfachvererbung): zeigt hierarchische Struktur der Vererbungsbeziehungen einer Menge von Klassen; folgt aus Vererbungsbeziehungen zwischen Klassen; an Wurzel steht die Klasse Object, diese hat als einzige keine Superklasse

**Subtyp Beziehungen:** Wo A Instanzen erwartet werden, können auch B Instanzen ohne Probleme auftreten; B muss alle A Operationen anbieten und wie A Operationen anwendbar sein (+ kompatibles Ergebnis liefern) ; ergTypA oder ergTypB ist Supertyp von ergTypA (Kontravarianz); B „is-a“ A=> Ausprägungen von B sind spezielle Ausprägungen von A

**Implementationsvererbung:** Dieser Vererbungsbeziehung liegt kein Konzept zugrunde.; Vordergrund steht Wiederverwendung von Programmcode.; Implementationsvererbung kann sinnvoll durch Aggregation ersetzt werden!

**Assoziation:** Beziehung (Relation) zwischen (den Instanzen von) mehreren (meist zwei) Klassen; Gemeinsame Struktur (und Semantik) einer Menge von Objektbeziehungen; Jeder Partner einer Assoziation hat eine bestimmte Rolle, diese wird durch einen Rollennamen bezeichnet, die (Instanzen der) Partnerklassen werden durch ihre Rolle repräsentiert; Bezüglich der Navigierbarkeit ist zu unterscheiden: **bidirektionale Assoziation**: von den Objekten mit jeder Assoziationsrolle kann das Objekt mit der jeweils anderen Assoziationsrolle direkt erreicht werden, **Linie (oder Doppelpfeil); gerichtete Assoziation**: von dem Objekt mit einer Assoziationsrolle kann das die andere Rolle spielende Objekt direkt erreicht werden, nicht aber umgekehrt, **Pfeil mit Spitze zum direkt erreichbaren Objekt;** Assoziationen können (wie andere UML-Elemente) mit vordefinierten oder benutzerdefinierten Eigenschaften versehen werden

**Mengen von Assoziationen:** können Einschränkungen auferlegt werden: Einschränkung für Paare von Assoziationen mit gleichen Partnern: {subset};Die Menge der Ausprägungen einer Assoziation muss eine Teilmenge der Ausprägungen der anderen Assoziation sein;

Einschränkung für Mengen von Assoziationen mit einem gemeinsamen Partner: {xor}; Für eine Instanz der gemeinsamen Partnerklasse darf jeweils
nur eine der möglichen Assoziationen realisiert sein.

**Qualifizierte Assoziation:** Binäre Assoziation, die an einem Ende ein qualifizierendes Attribut trägt**;** Referenzierte Objektmenge wird durch die Werte des qualifizierenden Attributs partitioniert, wobei jede Partitiondes Zielobjektes nur einmal vorkommen kann.; Das qualifizierende Attribut muss im Zielobjekt definiert(oder ableitbar) sein.

**Aggregation** (Teile/Ganzes -Beziehung)**:** Beziehung zwischen einem Ganzen (Aggregat) und einem Teil (Komponente); Teile sind vom Ganzen existentiell unabhängig: wenn eine Instanz des Ganzen zerstört wird, dann existieren seine Teile weiter; Die Klasse des Ganzen heißt Aggregatklasse, ihre Ausprägungen Aggregatobjekte; Teile können einem Aggregatobjekt nicht exklusiv oder exklusiv zugeordnet sein; **nichtausgefüllte Raute** bei der Aggregatklasse

**Komposition:** Strenge Form der Aggregation**;** Teile sind vom Ganzen existentiell abhängig: wenn eine Instanz des Ganzen zerstört wird, dann auch seine Teile; Die Klasse des Ganzen heißt Kompositionsklasse, ihre Ausprägungen Kompositionsobjekte; Teile können einem Kompositionsobjekt nur exklusiv zugeordnet sein; **ausgefüllte Raute** bei der Aggregatklasse

In einer **Kompositionsbeziehung** gelten folgende **Einschränkungen:** Exklusivität: ein Objekt darf Kompositionsteil von höchstens einem Ganzen auftreten. Bei der Kompositionsraute sind nur die Multiplizitäten „1“ oder „0..1“ möglich!; Existenzabhängigkeit: ein Teil existiert höchstens so lange wie sein Ganzes (er kann jedoch während der Lebensdauer des Ganzen an ein anders Ganzes übertragen werden). Im Falle der Komposition mit Multiplizität „0..1“ kann der Teil so lange unabhängig existieren, bis er (erstmals) einem Ganzen zugeordnet wurde.

Ein **Zustandsdiagramm** ist auf *ein* Objekt bezogen: es beschreibt einen wesentlichen Aspekt des Intraobjektverhaltens; Festlegung von Objektzuständen und deren zulässige Abfolgen, die sowohl während des gesamten Objektlebenszyklus als auch innerhalb einer Aktivität durchlaufen werden können; Charakterisierung durch einen Graph, dessen Knoten die Zustände und dessen gerichtete Kanten die möglichen Zustandsübergänge (Transitionen) darstellen. In der Folge werden Zustandsdiagramme nur in einfacher Form einführend behandelt.

**Zustand:** Benannte Bedingung (Situation) in der Lebenszeit eines Objektes, die eine endliche Zeit lang besteht; Zustände können durch Zustandsvariable (diese müssen in
der Klasse definiert sein) charakterisiert werden; Zustände können zugeordnet haben:
- Eintrittsaktivität (entry activity)
- Austrittsaktivität (exit activity)
- Zustandsaktivität, die andauert und den Zustand nicht verändert(do activity)
- Invarianten
- Zustände können in Unterzustände (substates) aufgeteilt sein.

**Zustandsübergang, Transition:** Gerichtete Beziehung zwischen einem Quellzustand und einem Zielzustand; Einem Zustandsübergang sind zugeordnet: ein auslösendes Ereignis, Trigger (event, trigger), eine Bedingung, Wächter (guard condition)
(eines davon kann fehlen);,optional eine Aktivität, die beim Durchlaufen des Überganges ausgeführt wird; Ein Zustandsübergang findet statt (feuert), wenn die

Bedingung (Wächter) erfüllt ist und das auslösende Ereignis eingetreten ist. Eine bedingungslose Transition feuert, wenn das Ereignis eintritt, eine ereignislose Transition findet statt, wenn die Bedingung wahr ist.

**Arten von Triggern** und zugeordneten Ereignissen: Nachrichtentrigger (message trigger);signal trigger (Ereignis = beobachtetes Signal); call trigger (Ereignis = Methodenaufruf); any trigger (Ereignis = jedes Ereignis, sofern es nicht an einer anderen Transition des aktiven Zustandes zugeordnet ist); Zeittrigger (time trigger) (Ereignis = Ablauf einer Zeitspanne oder Erreichen eines Zeitpunktes); Veränderungstrigger (change trigger) (Ereignis = Änderung des Wertes eines Boole´schen Ausdruckes von false auf true)

**Zusammengesetzte Zustände:** enthalten einen oder mehrere (einfache oder zusammengesetzte) Zustände; die enthaltenen Zustände heißen Unterzustände; Interne Transaktionen zusammengesetzter Zustände werden auf die Unterzustände vererbt, sie werden innerhalb des aktiven (Unter)zustandes ausgeführt; Ein zusammengesetzter Zustand ist aktiv, wenn er (direkt oder transitiv) einen aktiven Zustand enthält.

Ein **Aktivitätsdiagramm** ist auf einen Prozess bezogen; es beschreibt wahlweise:
- die Realisierung einer Operation (als Folge von Anweisungen)
- die Ausführung eines Anwendungsfalls
- das Zusammenspiel mehrerer Anwendungsfälle
- einen Geschäftsprozess (Folge von Aktivitäten zur Erreichung eines Geschäftsziels)
- es beschreibt die Details der Realisierung eines Systemverhaltens
- beschreiben Aktivitäten in Form von gerichteten Graphen, deren Knoten sind:
- Aktivitätsknoten (zusammengesetzte, komplexe Abläufe)
- Aktionsknoten (einzelne Elementaraktionen)
- Objektknoten (beteiligte Objekte, Daten)
- Kontrollknoten (Ablaufsteuerung) deren (gerichteten und optional beschrifteten) Kanten geben erlaubte Abfolgen von Aktionen an. (Hinweis auf ähnliche Beschreibungsformalismen: Datenflussdiagramme, Struktogramme (Nassi- Shneiderman-Diagramme), Petri-Netze)

**Aktionsknoten** beschreiben Aktionen, **abgerundetes Rechteck**

**Objektknoten** beschreiben Typen von Objekten/Daten, die als Eingangs- oder Ausgangsparameter von Aktivitäten wirken, **Rechteck;** Kontrollknoten beschreiben die Steuerung möglicher Ablaufwege (Symbole ähnlich jenen für Pseudozustände bei Zustandsdiagrammen)

Zu Beschreibung **sequentieller und nebenläufiger Abfolgen** von Aktionen dienen Token (Marken), die entlang der Kanten „fließen“.; Zwischen Aktionsknoten fließende Token „steuern“ den Kontrollfluss (Kontrolltoken); Zwischen Aktionsknoten und Objektknoten fließende Token beschreiben zusätzlich eine Übertragung von Objekten/Daten (Datentoken).; Token „verweilen“ in Aktions- und Objektknoten, ihre Position markiert die jeweils aktive Aktion, ihr Fluss durch Kanten und Kontrollknoten ist zeitlos.

In einen Objektknoten hineingehende **Token** sind mit Daten beladen, die in der Aktion entstehen, von der Art des Objektknotens (Objekte/Werte) sind und im Objektknoten „abgeladen“ werden (Ausgangsparameter); Objektknoten dienen zur Darstellung des Daten-/Objektflusses, sie sind nicht als Objekte (Klasseninstanzen) zu verstehen; Aus einem

Objektknoten herausgehende Datentoken tragen das Objekt oder Teile in die folgende Aktion (Eingangsparameter).

Ein Token bleibt so lange in einer Aktion, bis diese zu einem Ende gekommen ist, **abgerundetes Rechteck**; Ein Token bleibt so lange in einer Signalsendeaktion, bis das Signal gesendet ist, dann wird die Aktion abgebrochen und der Kontrollfluss fortgesetzt, **Großer Pfeil**; Ein Token bleibt in einer Signalempfangsaktion, bis das erwartete Ereignis eintritt, dann wird dieAktion abgebrochen und der Kontrollfluss fortgesetzt, **Rechteck wo „Pfeilspitze“ (hinten) ausgeschnitten („leer“) ist**

**Stereotypen:** Definition neuer Modellierungselemente auf Basis bestehender Elemente; Charakterisierung einer besonderen Art des Modellelementes: Stereotypen von Klassen («controller», «actor», «interface», …), Stereotypen von Objektknoten («centralBuffer», «datastore»); Schreibweise: «stereotyp» über dem Elementnamen