BØRNE- OG
UNDERVISNINGS-
MINISTERIET
STYRELSEN FOR
UNDERVISNING OG KVALITET

Ordliste over fagbegreber i teknologiforståelse i folkeskolen

Udarbejdet for
Børne- og Undervisningsministeriet
af Michael E. Caspersen og Ole Sejer Iversen

November 2019

Indhold

Indledning

Denne ordliste indeholder korte beskrivelser af 54 udvalgte fagbegreber fra teknologiforståelse. Begreberne er udvalgt af formandskabet for ekspertskrivegruppen for teknologiforståelse efter opdrag fra Børne- og Undervisningsministeriet.

Målgruppen for ordlisten er

- medarbejdere på professionshøjskoler, som arbejder med teknologiforståelse i folkeskolen (herunder med grund-, efter og videreuddannelse af folkeskolelærere i teknologiforståelse samt med understøttelse af skolers arbejde med teknologiforståelse) og
- lærere i den danske folkeskole, der underviser i teknologiforståelse.

De udvalgte fagbegreber er færdigheds-/vidensområder fra Fælles Mål (for teknologiforståelse som selvstændigt fag) samt yderligere 30 fagbegreber fra to beslægtede fagfelter, datalogi og digital design. Ordlisten er relevant for både arbejde med teknologiforståelse som selvstændigt fag og teknologiforståelse integreret i eksisterende fag.

Fagbegreberne er defineret specifikt i kontekst af teknologiforståelse og er dermed ikke nødvendigvis fyldestgørende i forhold til den faglighed, som de stammer fra. En del af fagbegreberne har en almen betydning i det danske sprog, men en særlig og specifik betydning eller toning i teknologiforståelse. Det er sidstnævnte, der er beskrevet her.

Begreber, der beskrives i ordlisten, er markeret med kursiv, når de anvendes i beskrivelsen af ordlistens øvrige begreber.

Fagbegreber

Abstraktion
Aggregering
Algoritme
Analog
Argumentation
Automatisering
Begrebsmodel
Brug
Brugspraksis
Brugsstudier
Brugssituation
Computationel tankegang
Computersystem
Data
Dekomponering
Digital
Digital design og designprocesser
Digitale artefakter
Digitale teknologier
Digital myndiggørelse
Divergent tænkning
Eksemplificering
Eksternalisering
Formålsanalyse
Generalisering
Idégenerering
Inputteknologi
Intentionalitet
Introspektion
Iteration (algoritme)
Iteration (designproces)
Klassifikation
Komplekse problemer
Konsekvensvurdering
Konstruktion
Konvergent tænkning
Modellering
Mønstre
Netværk
Outputteknologi
Programmering
Programmeringssprog
Rammesættelse
Redesign
Selektion
Sikkerhed
Specialisering
Strukturering
Teknologisk handleevne
Teknologianalyse
Teknologiforståelse

Abstraktion
Abstraktion betegner en proces (eller resultatet heraf), hvor man fokuserer på bestemte facetter af en sag/et objekt og ignorerer andre. Formålet er at blive klogere på sagen, uden at man forstyrres af (midlertidigt) irrelevante detaljer.
Ved abstraktion bevæger man sig fra det (mere) konkrete til det (mere) abstrakte. Den modsatte proces er konkretisering, hvor man bevæger sig fra det (mere) abstrakte til det (mere) konkrete.
Et eksempel er at bevæge sig fra konkrete fænomener (for eksempel cykler) til begrebet cykel. Et andet eksempel er at bevæge sig fra begrebet cykel til det mere abstrakte begreb køretøj.
Abstraktion findes i tre varianter: *klassifikation*, *aggregering* og *generalisering*. Tilsvarende findes konkretisering i tre varianter (de modsatrettede processer): *eksemplificering*, *dekomponering* og *specialisering*.

Aggregering
Aggregering betegner sammensætning af dele (komponenter) til et hele (aggregat) – det modsatte af *dekomponering,* som er et hele, der opsplittes i dele. Man taler i den forbindelse om 'part-whole' og karakteriserer populært relationen mellem 'whole' og 'part(s)' som en 'has-a'-relation.
Et hverdagseksempel er en bil (aggregat), der er sammensat af motor, karosseri, hjul, mv. (komponenter). En motor kan naturligvis også betragtes som et aggregat, der er sammensat af komponenter.
Aggregering og *dekomponering* spiller en rolle i mange sammenhænge af *teknologiforståelse*, både i analyse- og designaktiviteter og i modellerings-, konstruktions- og programmeringsaktiviteter.
Se også *abstraktion*.

Algoritme
Algoritme betegner en *utvetydig* beskrivelse af løsning af et problem. Utvetydig betyder, at algoritmen er udformet i et sprog, der entydigt kan forstås af den (eller det), der skal udføre algoritmen. En algoritme fungerer således som en opskrift for, hvordan man i et antal trin kan løse problemet.
Et eksempel er at finde vej fra et sted til et andet, Et andet eksempel er løsning af en ligning. Et tredje eksempel er at finde et bestemt ord i et leksikon, og et fjerde eksempel er en opskrift for, hvordan man laver for eksempel risotto.
Algoritmer ligger til grund for programmering og udtrykkes typisk i et tekstbaseret eller grafisk sprog, hvor de algoritmiske strukturer er tydelige (sekvens, løkker, forgreninger og eventuelt "under- eller delalgoritmer").

Analog
Analog står i modsætning til *digital* og betegner processer eller systemer, hvor *data* præsenteres/overføres/behandles i form af kontinuerte fysiske størrelser (f.eks. lyd, hastighed, temperatur og alder).
For eksempel kan *data* præsenteres analogt ved et "gammeldags" instrument med visere (et ur, et speedometer eller et termometer).

Argumentation
Argumentation betegner de begrundelser, der kan gives i forhold til valg og fravalg i en designproces. I designprocessen kan eleven argumentere for sine valg med henvisning til den viden, der er skabt i designprocessen gennem egne eller andres undersøgelser. Eleven kan som eksempel argumentere for et design gennem reference til elementer i deres egen undersøgelse af en eksisterende *brugspraksis* eller til anerkendte kriterier og konventioner for godt design.

Automatisering
Automatisering betegner den proces, hvor man organiserer en fysisk eller kognitiv proces, så den kan udføres af en automatisk virkende enhed (for eksempel maskiner eller robotter) i stedet for menneskelig arbejdskraft.
Den industrielle revolution har affødt en lang række af maskiner, som helt eller delvist kan automatisere fysiske processer, for eksempel at generere energi, fragte mennesker og gods fra A til B, samle komponenter, osv.
Den computationelle revolution har resulteret i opfindelsen af computeren – en maskine der i sig selv intet kan, men som når den "fodres" med programmer og udstyres med passende ydre enheder (*input-/outputteknologier*), kan bringes til at kunne næsten alt. Specielt kan computeren ved hjælp af programmer helt eller delvist automatisere ikke bare fysiske, men komplekse kognitive processer.

Begrebsmodel
Begrebsmodel er en model af et problemfelt, for hvilket man ønsker at udvikle et digitalt artefakt. Gennem en analyse og *abstraktion* identificeres og beskrives centrale begreber og deres indbyrdes relation i et problemfelt. Modellen er nyttig i en dialog med fremtidige brugere om artefaktets ydre design/grænseflade. Modellen vil typisk afspejle sig i det *digitale artefakts* interaktionsdesign samt i den underliggende datamodel for systemet.
For eksempel vil man for et system til elevadministration finde begreber som elev, lærer, klasse, fag, lokale, fravær, prøve, prøveform, karakter, udtalelse osv. For en musikafspiller vil man finde begreber som musiknummer, kunstner, komponist, album, genre, spilleliste, afspiller osv.

Brug
Brug betegner de måder, hvorpå et *digitalt artefakt* anvendes. Den faktiske brug af et *digitalt artefakt* vil ofte adskille sig fra den brug, som designeren har forestillet sig. Når den faktiske brug af et *digitalt artefakt* forstås, giver det en bedre forudsætning for at komme med kvalificerede forslag til *redesign* eller en vurdering af, om det *digitale artefakt* er egnet til en given situation. Viden om brug opnås som oftest gennem *brugsstudier*.

Brugspraksis
Brugspraksis er, sammenlignet med *brug*, en mere generel beskrivelse af de aktiviteter, vaner, adfærd og handlinger, som fremkommer når et *digitalt artefakt* har været i anvendelse over en længere periode. Det er vigtigt at forstå brugspraksis for at kunne foretage konsekvensanalyser og vurdere, hvad et givet *digitalt artefakt* betyder for individet, fællesskabet eller samfundet. Viden om brugspraksis opnås som regel gennem *brugsstudier*.

Brugsstudier
Brugsstudier betegner de undersøgelser, hvor der tilvejebringes viden om den specifikke *brug* af *digitale artefakter* eller om den generelle *brugspraksis*. Brugerstudier vil omfatte indsamling og analyse af *data*, der belyser et *digitalt artefakts* betydning og *brug*. Dataindsamlingen vil ofte indbefatte observationer af, hvordan et digitalt design bruges, samt interviews, der søger at afdække oplevelsen af det *digitale artefakt* og dets betydning.

Brugssituation
Brugssituation betegner en situation, hvori et *digitalt artefakt* bruges. En brugssituation omfatter personer og de fysiske omgivelser, men også den kultur og de omstændigheder, der har betydning for brugen. Brugssituationer er som oftest det primære genstandsfelt for *brugsstudier*. Som eksempel vil den *brugspraksis* børn har med mobiltelefoner være del af en bredere brugssituation, der også rummer forældre, venner og de institutioner, hvor børnene færdes.

Computationel tankegang
Computationel tankegang er et af *teknologiforståelse* som fags fire kompetenceområder. Det er en samlet betegnelse for de processer, som indgår i at modellere et problem, så det kan behandles effektivt af en computer. Dette omfatter analyse, *modellering* og *strukturering* af *data*, datarepræsentationer og dataprocesser. Centrale begreber i den forbindelse er *data*, *algoritmer*, *strukturering* og *modellering*.
Teknologiforståelse som fags fire kompetenceområder er hinandens forudsætninger, og de beriger hinanden. Se video om computationel tankegang på emu.dk (Teknologiforståelse → "Kompetenceområder")

Computersystem
Computersystem er en betegnelse for en komplet computer med operativsystem, software og ydre enheder (f.eks. tastatur, skærm, printer, højtalere, lagringsenheder osv.), samt den infrastruktur der muliggør, at computere af alle mulige slags kan "samarbejde" via internettet (og andre *netværk*).
I relation til *teknologiforståelse* handler det dels om selve computeren og dens principielle virkemåde, dels om måden computere fungerer på i *netværk*.

Data
Data er "*enhver repræsentation af fakta eller ideer på en formaliseret måde, som kan kommunikeres eller manipuleres ved en eller anden proces"* (Peter Naurs definition).
Et banalt eksempel er en persons navn og alder, som simpelt kan repræsenteres i form af en tekststreng og et tal – i modsætning til udsagn om, at en person er venlig eller pæn.
I takt med at man ønsker at repræsentere flere og flere aspekter af virkeligheden i digitale modeller, er man tvunget til at formalisere disse aspekter af virkeligheden som data. Det åbner for slagkraftige analyser (data science), men også for "overformalisering" hvor virkeligheden struktureres mere end godt er. Dette er et vigtigt aspekt af computationel *modellering*.

Dekomponering
Dekomponering betegner opdeling/nedbrydning af et hele i dele – det modsatte af *aggregering*.
Se også *abstraktion*.

Digital
Digital står i modsætning til *analog* og betegner processer eller systemer, hvor *data* dybest set præsenteres/overføres/behandles i form af talrepræsentationer.
For eksempel kan *data* præsenteres digitalt ved et "moderne" instrument med tal (et digitalur, et digitalt speedometer eller et digitalt termometer). Computere, som vi kender dem i dag, er digitale apparater, hvori både *data* og programmer er repræsenteret digitalt (dvs. indkodet i tal).

Digital design og designprocesser
Digital design og designprocesser er et af *teknologiforståelse* som fags fire kompetenceområder. Digital design og designprocesser er en samlet betegnelse for de processer, hvori *digitale artefakter* tilvejebringes. Dette omfatter *rammesættelse*, *idégenerering*, *konstruktion, argumentation* og *introspektion*, ofte udført i flere *iterationer*.
Teknologiforståelse som fags fire kompetenceområder er hinandens forudsætninger, og de beriger hinanden. Se video om digital design og designprocesser på emu.dk (Teknologiforståelse → "Kompetenceområder").

Digitalt artefakt
Digitalt artefakt betegner en af mennesket tilvejebragt genstand, som indeholder et væsentligt element af *digital teknologi*. Til forskel fra betegnelsen *digital teknologi*, betoner betegnelsen digitalt artefakt de produktkvaliteter, der er blevet til gennem design og *programmering*, hvorved *intentionalitet* og formål er blevet indlejret i artefaktet.
En app, en programmeret robot, en simulering af fotosyntese eller en programmeret micro:bit indlejret i et fysisk artefakt er eksempler på digitale artefakter.

Digitale teknologier
Digitale teknologier betegner i denne sammenhæng et materiale, der har et væsentligt digitalt element. Til forskel fra *digitalt artefakt* betegner digital teknologi det potentiale, som det digitale materiale rummer, i forhold til at kunne indgå i en designproces, hvor digital teknologi bruges til at udforme et *digitalt artefakt*. Et *programmeringssprog*, en database, et arduinoboard eller en Makey-Makey er typiske eksempler på digitale teknologier.

Digital myndiggørelse
Digital myndiggørelse er et af *teknologiforståelse* som fags fire kompetenceområder. Digital myndiggørelse handler om evnen til analytisk og refleksivt at forstå *digitale artefakters* betydning i hverdags- og arbejdslivet. Gennem teknologiforståelsesfaglige analyser af *digitale artefakter*, artefaktets indlejrede *intentionalitet* og artefaktets *brug* (via *teknologianalyse*, *formålsanalyse* og *brugsstudier*) får eleven det nødvendige grundlag for at kunne *redesigne digitale artefakter,* hvor artefakterne synes uhensigtsmæssige. Eleven får også grundlag for at vurdere artefaktets betydning for individ, fællesskaber og samfund (*konsekvensvurdering*).
Teknologiforståelse som fags fire kompetenceområder er hinandens forudsætninger, og de beriger hinanden. Se video om digital myndiggørelse på emu.dk (Teknologiforståelse → "Kompetenceområder").

Divergent tænkning
Divergent tænkning betyder – modsat *konvergent tænkning* – at udvide og åbne mulighedsrummet og søge nye inputs eller ny viden. I designprocessen beskriver divergens de aktiviteter, hvor der afsøges nye ideer og forståelser for dermed at åbne for nye muligheder i forhold til et design. Divergens kan for eksempel opnås gennem at inddrage nye *brugsstudier* eller ved at søge ny inspiration i brugskontekst eller gennem overvejelser om alternative teknologiske muligheder.

Eksemplificering
Eksemplificering er at angive et konkret eksempel på et fænomen, der falder under et begreb – det modsatte af *klassifikation*.
Hverdagseksempler er Folketinget, som falder under begrebet "parlament", eller Paul McCartney, som sammen med Ringo Starr, John Lennon og George Harrison falder under begrebet 'beatle'. De to sidstnævnte falder endvidere under det mere specielle begreb 'afdød beatle'.
Eksemplificering og *klassifikation* spiller en rolle i mange sammenhænge af *teknologiforståelse*, både i analyse- og designaktiviteter og i modellerings-, konstruktions- og programmeringsaktiviteter.
Se også *abstraktion*.

Eksternalisering
Eksternalisering betegner den proces, hvor en idé eller en tanke, gives fysisk form eller på anden måde gøres tilgængelig og konkret for andre. I design vil eksternaliseringer ofte være modeller, skitser eller mock-ups af et *digitalt artefakt*, der udarbejdes i tilgængelige materialer, for eksempel papir, træ eller i digitale omgivelser som App Lab. Disse skabes med henblik på at kommunikere idéer til andre og modtage feedback på disse idéer.

Formålsanalyse
Formålsanalyse betegner den proces, hvori et digitalt artefakt undersøges med henblik på at forstå dets funktioner, anvendelsesmuligheder og intention. Formålsanalyse belyser, hvordan en designidé kommer til udtryk gennem fysiske og digitale egenskaber, som for eksempel menuer, knapper og grafiske elementer. Hvor *brugsstudier* beskæftiger sig med den faktiske *brug*, kan man sige, at formålsanalyse beskæftiger sig med den tiltænkte *brug*.

Generalisering
Generalisering er at danne et (mere) generelt begreb ud fra et eller flere (mere) specielle begreber – det modsatte af *specialisering*. Man taler i den forbindelse om overbegreber og underbegreber og karakteriserer populært relationer mellem under- og overbegreber som en 'is-a'-relation.
Et hverdagseksempel er begreberne køretøj og transportmiddel, hvor begrebet transportmiddel er en generalisering af begrebet køretøj.
Se også *abstraktion*.

Idégenerering
Idégenerering omhandler systematisk behandling af viden med henblik på at skabe løsningsforslag, der gennem *eksternalisering* gøres til genstand for kollektiv bearbejdning og vurdering. Med idégenerering giver eleverne specifikke svar på en problemstilling. Dette kan finde sted på mange tidspunkter i en designproces, men vil som oftest bygge på elevens undersøgelser. I idégenereringen

kan eleverne eksempelvis skitsere designidéer, bygge dem i pap og papir eller skrive scenarier, der angiver en måde, hvorpå et fremtidigt *digitalt artefakt* kan bringes i anvendelse.

Inputteknologi
Inputteknologi er en samlet betegnelse for de dele af et *digitalt artefakt*, som lader brugeren interagere med artefaktet. Det kan eksempelvis være et tastatur, et kamera eller en sensor, som kan registrere et input fra en bruger og omsætte det til en handling i en computer. Det kan være vigtigt at forholde sig til inputteknologier i forhold til dels egne *digitale artefakter* og dels i forhold til analytisk at vurdere andres *digitale artefakter*.

Intentionalitet
Intentionalitet betegner de holdninger eller værdier, som designere har indlejret i et *digitalt artefakt*. Alle *digitale artefakter* sigter efter noget særligt eller har noget særligt til hensigt. Designere har gennem valg og fravalg i designprocessen besluttet, hvilke egenskaber et *digitalt artefakt* skal have. Disse egenskaber sigter mod at give fremtidige brugere særlige oplevelser eller muligheder gennem interaktion med det *digitale artefakt*. Den eller de mennesker, som har designet et *digitalt artefakt*, har således gjort det ud fra en intention, og artefaktet bærer dermed en intentionalitet.

Introspektion
Introspektion betyder at kigge indad. I designprocessen er introspektion en refleksion, som primært foretages efter, at designprocessen er afsluttet. Her kan eleverne på andres eller eget initiativ reflektere over, hvad de har lært i en designproces. Introspektion betegner den erkendelsesproces, gennem hvilken eleverne bliver bevidste om viden, færdigheder og kompetencer, som de har erhvervet sig gennem en designproces. Introspektion er eksempelvis at overveje, hvordan fremtidsscenarier bidrog til input fra brugere i en given designproces, eller hvordan man som designer kan indhente viden om et helt nyt område gennem observationsstudier eller interviews.

Iteration (ift. algoritmer)
Iteration (ift. algoritmer) er en betegnelse for en kontrolstruktur i algoritme- og *programmeringssprog*, hvor man gentager en handling et bestemt eller ubestemt antal gange. Iteration udtrykkes ved hjælp af såkaldte løkker, for eksempel ”for”-løkker og ”while”-løkker.
Det arketypiske eksempel på en ”for”-løkke er: **for** 1...100 **do** <handling>.
Det arketypiske eksempel på en ”while”-løkke er: **while** <betingelse> **do** <handling>.
Iteration findes naturligvis også i blokbaserede sprog. I Scratch beskrives de med "forever"-, "repeat"- eller "repeat until"-blokke.

Iteration (ift. designprocesser)
Iteration (ift. designprocesser) betyder gentagelse. I *digital design og designprocesser* bruges iteration, når man skal beskrive en designproces, hvor man gentager aktiviteter, men baserer dem på en ny viden. Det kunne for eksempel være, når eleverne på baggrund af ny viden skaber et nyt scenarium, eller når eleverne skaber en ny prototype på baggrund af input fra fremtidige brugere. At arbejde iterativt i designprocessen har den fordel, at svære beslutninger om det fremtidige *digitale artefakt* kan omgøres i takt med, at vidensmængden opbygges gennem designprocessen.

Klassifikation
Klassifikation er at samle en mængde fænomener til et begreb – det modsatte af *eksemplificering*. Hverdagseksempler på *klassifikation* er at samle fænomener som Folketinget og Stortinget under begrebet "parlament" eller at samle Paul McCartney, Ringo Starr, John Lennon og George Harrison under begrebet "beatle".
Se også *abstraktion*.

Komplekse problemer
Komplekse problemstillinger betegner en særlig kategori af problemfelter, der ikke kan beskrives entydigt, og hvortil der ikke kan skabes en entydig rigtig løsning. Komplekse problemer er ofte kendetegnet ved mangelfulde eller modsatrettede informationer, som gør det svært at forstå problemets omfang og format. Dermed kan komplekse problemer give sig udtryk i et dilemma. Eksempler på nyere komplekse problemer kunne være global migration, børns skærmtid, god opdragelse, global opvarmning eller ulandsbistand.
Et komplekst problem kendetegnes ved den måde, vi arbejder med dets løsning. Problemet rammesættes i én og samme proces, som vi finder dets mulige løsning(er). Det kræver, at man iterativt arbejder med at rammesætte, undersøge og idéudvikle og gradvist nærme sig en problemstilling, der kan gøres til genstand for en mulig løsning.
Problemer kan godt være svære uden at være komplekse. Det gælder eksempelvis beregninger af andengradsligninger eller kasusbøjninger i tyskundervisningen. Komplekse problemer er en særlig kategori, som ikke er kendetegnet ved problemernes sværhedsgrad, men ved problemernes dilemmafyldte karakter.

Konsekvensvurdering
Konsekvensvurdering omhandler overvejelser over *digitale artefakters* betydning for individ, fællesskab og samfund, herunder etiske dilemmaer, der knytter sig til deres anvendelse. En konsekvensvurdering bygger på en teknologiforståelsesfaglig analyse (*teknologianalyse*, *formålsanalyse* og *brugsstudier*) af et *digitalt artefakt*. Analysen gøres til genstand for en personlig vurdering i forhold til det *digitale artefakts* betydning for individ, fællesskaber eller samfund. Konsekvensanalyse er altså en fortolkende aktivitet, i hvilken eleverne trænes i at forstå og vurdere *digitale artefakters* betydning.

Konstruktion (at konstruere)
Konstruktion omhandler den aktivitet, hvor idéer finder udtryk i et konkret *digitalt artefakt*, som kan gøres til genstand for en efterprøvning af form, funktion og interaktion. Konstruktion i *digitale teknologier* rummer aktiviteter såsom *computationel tankegang*, valg af *programmeringssprog*, *programmering*, design af grænseflade, konstruktion af en tidlig papirmodel eller diagram over det fremtidige artefakt, at bygge den fysiske udformning af et *digitalt artefakt* og iterative evalueringer af *digitale artefakter* under udvikling.

Konvergent tænkning
Konvergent tænkning betyder – modsat *divergent tænkning* – at indsnævre eller dedikere sine tankeprocesser i en særlig retning. I designprocessen omfatter konvergent tænkning de aktiviteter,

hvor eleverne fravælger mulige løsninger eller frasorterer designidéer og dermed fokuserer arbejdet i en særlig retning.

Modellering
Modellering har flere betydninger:

a. Begrebs- og brugsmodellering i forbindelse med analytiske aktiviteter, der har til formål at give forståelse af et problemfelt (se *abstraktion* og *begrebsmodel*).
b. Computationel modellering af et problemfelt, der har til formål at frembringe abstrakte modeller af *data* og dataprocesser (for eksempel *algoritmer* samt data- og klassemodeller, der udarbejdes som mellemformer på vejen til realisering af et *digitalt artefakt*) eller konkrete computationelle modeller realiseret i et *programmeringssprog* – for eksempel AgentCubes eller NetLogo – hvor man simulerer et system ved at lave simple regler for de enkelte komponenters adfærd. Den form for computationel modellering kaldes agentbaseret modellering.

Mønstre/mønstergenkendelse
Mønstre/mønstergenkendelse er nært knyttet til *abstraktion*, idet mønstergenkendelse typisk forudsætter evnen til at abstrahere fra "støj", således at mønsteret træder frem. Begrebet har flere betydninger:

a. Ved analyseaktiviteter er det vigtigt at kunne identificere/genkende mønstre i problemfelter, *brugssituationer*, *data* og processer. Dette er vigtig for at kunne abstrahere og forstå essentielle egenskaber ved problemer og *brugssituationer*, som igen er en forudsætning for at kunne designe enkle, nyttige og brugervenlige *digitale artefakter*.
b. Ved design- og konstruktionsaktiviteter gør man brug af mønstre (skabeloner) i form af standardløsninger på ofte forekommende problemer i forbindelse med interaktionsdesign, *strukturering* af *data* og dataprocesser. Dette gør, at man ikke behøver at genopfinde "den dybe tallerken", og det er medvirkende til at skabe løsninger med en genkendelig struktur (for de modtagere, der kender mønstrene). Det kan for eksempel være mønstre for *algoritmer*/programstruktur, mønstre for brug af variabler, mønstre for at tilgå *data* i en database, mønstre for grænsefladedesign osv.

Netværk
Netværk betegner den infrastruktur, som tillader computere at "tale" sammen. Netværk kan være trådede og trådløse.

Outputteknologi
Outputteknologi er en samlet betegnelse for de dele af et *digitalt artefakt*, som giver brugeren feedback på en interaktion med artefaktet. Det kan eksempelvis være en skærm (visuel feedback), en højttaler (auditiv feedback) eller en vibration (taktil feedback).

Programmering
Programmering betegner det at programmere, det vil sige at udvikle programmer.
Programmering er meget mere end at kunne kode i et programmeringssprog. For at kunne programmere skal man til en vis grad mestre det programmeringssprog, som man skal udtrykke sig i,

men endnu vigtigere er det at kunne "problemløse", det vil sige arbejde på problemniveau, hvor man skal forstå og løse problemet. Og så skal man have udtrykt sin (idé til en) løsning i et programmeringssprog.
En konkret måde at adskille de to aktiviteter på er ved at arbejde med "mellemprodukter" i form af forskellige mere eller mindre formelle modeller for *data* og dataprocesser, først og fremmest *algoritmer* og figurer/tegninger, der indfanger elementer i problemløsningen.
Kort sagt kan man sige, at forskellen på problemløsning og programmering svarer til forskellen på at komponere en melodi og skrive melodien ned i nodesystemet.
For at kunne programmere skal man først og fremmest kunne problemløse, men den kompetence følger ikke af at kende et (eller flere) programmeringssprog.

Programmeringssprog
Programmeringssprog er en formel notation (et kunstigt tekstbaseret eller grafisk sprog) til beskrivelse af beregninger, der kan udføres af en computer. 'Beregninger' skal forstås bredt. Der findes forskellige typer af programmeringssprog. De mest udbredte er såkaldt imperative programmeringssprog, hvor man udtrykker sig i bydeform overfor computeren: gør det, så det, så det osv.
I imperative sprog beskriver man eksplicit den række beregningstrin, som man ønsker, at computeren skal udføre.
Langt de fleste programmeringssprog er kendetegnet ved, at man som programmør kan udvide sprogets ordforråd (udtryk, kommandoer og datatyper), det vil sige skabe nye, som man så kan bruge til at skrive programmer på et højere abstraktionsniveau.
Eksempler på programmeringssprog er Scratch, AgentCubes, Java, NetLogo, Python og JavaScript. De to førstnævnte er blokbaserede (visuelle) programmeringssprog, de øvrige er tekstbaserede (leksikalske) programmeringssprog. Der findes tusindvis af programmeringssprog.

Rammesættelse
Rammesættelse omhandler de processer, hvor eleven, gennem empiriske undersøgelser af et komplekst problem, bliver i stand til at afgrænse og formulere en problemstilling, der kan løses gennem design.

Redesign
Redesign omhandler design af en alternativ løsning på baggrund af forudgående analyser og vurderinger. Redesign rummer samme aktiviteter som *digital design og designprocesser*. I redesign betones dog, at designaktiviteterne er foranlediget af en analyse og *konsekvensvurdering* af et eksisterende *digitalt artefakt*. Ud fra *konsekvensvurderingen* initierer eleven en designproces, som har til hensigt at forandre de utilsigtede konsekvenser ved det eksisterende artefakt. Et eksempel på dette kunne være et redesign af en digital assistent, så den kan betjenes, uden at den auditivt overvåger samtalerne i rummet.

Selektion
Selektion er betegnelse for en kontrolstruktur i algoritme- og *programmeringssprog*, hvor man på basis af en betingelse vælger mellem alternative "veje" gennem programmet. Selektion udtrykkes ved hjælp af såkaldte forgreninger, primært "if"-sætninger.

Det arketypiske eksempel på en ”if”-sætning er: **if** <betingelse> **then** <handling-1> **else** <handling-2>. Selektion findes også i blokbaserede sprog. I Scratch beskrives de med "it-then"- eller "if-then-else"-blokke.

Sikkerhed
Sikkerhed betegner risici og forholdsregler ved brug af *computersystemer*, herunder sikring af *data*, privatliv og anonymitet. På den ene side vil man gerne sikre tilgængelighed af sine *data* (at de bevares og er tilgængelige). På den anden side vil man gerne beskytte sine *data*, så de kun er tilgængelige for autoriserede brugere.
Der er både organisatoriske og tekniske aspekter af it-sikkerhed i forhold til trusler og forholdsregler. Et konkret eksempel er databeskyttelsesforordningen, som har til formål at styrke og harmonisere beskyttelsen af personoplysninger i EU.
Eksempler på tekniske forordninger er passwords, backupsystemer og kryptering.
Eksempler på trusler er hacking, phishing og skadelig software (såkaldt malware), som virus, orme og trojanske heste.

Specialisering
Specialisering er at danne et (mere) specielt begreb ud fra et (mere) generelt begreb – det modsatte af *generalisering*.
Et hverdagseksempel er begreberne køretøj og transportmiddel, hvor begrebet køretøj er en specialisering af begrebet transportmiddel.
Se også *abstraktion*.

Strukturering
Strukturering af *data* og dataprocesser (*data*, programmer, modeller osv.) er vigtig af såvel kvalitative som kvantitative årsager. *Data* og programmer kan struktureres med henblik på effektivisering af programudførelsen, men i denne sammenhæng vil vi specielt fremhæve de kvalitative årsager til at prioritere strukturering.
Beskrivelser generelt, og programmer i særdeleshed, struktureres af hensyn til blandt andet læsbarhed/transparens, modificerbarhed og modularitet. Dette er specielt vigtigt i forbindelse med programmer til undervisningsbrug.
Programmer bliver hurtigt forholdsvis komplekse, og det er umuligt at overskue alle detaljer på en gang. Det er derfor vigtigt at opdele et program i mindre dele, der hver især kan overskues og forstås som et hele.
Som følge heraf skal man som programmør designe og konstruere programkomponenter (moduler), der løser veldefinerede simple opgaver. Disse kan kombineres til mere komplekse komponenter, som løser mere komplekse opgaver osv.
Det er lettere sagt end gjort, men eksemplariske eksempler, specielt såkaldte ‘Worked Examples’, er et vigtigt middel til at lære eleverne værdien af strukturering. Et andet vigtigt middel er brug af skabeloner eller *mønstre* i form af standardløsninger på ofte forekommende problemer (se *mønstre*).

Teknologianalyse
Teknologianalyse er en aktivitet, hvor eleven analyserer et *digitalt artefakts* fysiske og digitale kvaliteter, herunder artefaktets form, farve, komposition, funktionalitet, *inputteknologi* og *outputteknologi*. Teknologianalysen kan være mere eller mindre detaljeret og teknisk i forhold til elevernes forudsætninger for at foretage analysen. Forudsætningerne kan være elevernes alder eller tekniske kompetenceniveau. En teknologianalyse kan indeholde aktiviteter som analyse af systemspecifikationer for et *digitalt artefakt* eller hands-on-aktiviteter, hvor et artefakt afprøves eller skilles ad for på den vis at forstå dets komposition.

Teknologisk handleevne
Teknologisk handleevne er et af *teknologiforståelse* som fags fire kompetenceområder. Teknologisk handleevne betegner evnen til at kunne forstå og artikulere muligheder og begrænsninger ved *digitale teknologier* samt kunne udtrykke sig computationelt ved hjælp af disse. Dette omfatter først og fremmest *programmering*, men også mestring af *computersystemer*, *digitale teknologier* og tilhørende sprog.
Kompetenceområdet består af fire færdigheds-/vidensområder: *computersystemer*, *netværk*, *programmering* og *sikkerhed*.
Teknologiforståelse som fags fire kompetenceområder er hinandens forudsætninger, og de beriger hinanden. Se video om teknologisk handleevne på emu.dk (Teknologiforståelse → ”Kompetenceområder”).

Teknologiforståelse
Teknologiforståelse forener humanistiske, kreative og datalogiske fagfelter i bestræbelserne på at uddanne eleverne til at kunne undersøge og forstå menneskers brug af *digital teknologi*, og forudsætninger for selvstændigt og i samarbejde med andre at *redesigne* eller nyudvikle *digitale artefakter*. Læs om teknologiforståelses fagidentitet i læseplanen for teknologiforståelse som selvstændigt fag og se video om teknologforståelse på emu.dk (Teknologiforståelse → ”Om forsøget”).

Teknologiforståelse som selvstændigt fag har fire kompetenceområder: d*igital myndiggørelse, digital design og designprocesser*, *computationel tankegang* og *teknologisk handleevne*. Kompetenceområder er hinandens forudsætninger, og de beriger hinanden.