# Dette bilag er en del af inspirationsmaterialet til faget [Erhvervsinformatik](https://emu.dk/eud/erhvervsinformatik/digital-myndiggorelse) på emu.dk. ”[Teknologisk handleevne og computational tankegang](https://emu.dk/eud/erhvervsinformatik/teknologisk-handleevne-og-computationel-tankegang)” er et af de tre kompetenceområder i faget. . Nedenfor finder du en kort introduktion samt link til ressourcer om computationel tankegang.

Definition(er) af computationel tankegang

Oprindeligt stammer begrebet fra bogen Mindstorms af **Seymour Papert** fra 1980, hvor han anvender det i en enkelt sætning. Paperts brug af begrebet Computational Thinking hænger sammen med hans læringsteori om konstruktionisme og betegner anvendelsen af en computer til modellering, beregning og analyse for derigennem at udvikle problemløsende kompetencer og abstrakt tænkning, dvs. det at tænke gennem anvendelsen af en computer. Han skriver således om hobby-computerklubber og eksperimenter i den retning:

*“ Their computers simply did not have the power needed for the most engaging and shareable kinds of activities. Their visions of how to integrate computational thinking into everyday life was insufficiently developed.”  Papert, S: Mindstorms s. 182, 1980*I 2006 relancerede **Jeanette S. Wing** begrebet i en ny betydning, nemlig som en samlebetegnelse for en række tankeprocesser, der er i spil, når problemer og deres løsninger konkretiseres i digitale artefakter. (Wing 2006). Wing selv har siden da arbejdet på at forfine begrebet. En definition af begrebet kunne lyde som Cuny, Snyder og Wings (2010):

*“Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent”.*Siden Wings tekst fra 2006 har begrebet fået stor bevågenhed internationalt og dannet afsæt for en række “skoler”, som har søgt at udfolde det på forskellig vis. Af betydelige bidrag til udfoldelser af begrebet kan bl.a. nævnes;

**Tim Bell** m.fl, som står bag en stor samling af ressourcer, der sigter mod at bibringe forståelser for computationelle strukturer og koncepter gennem fysiske lege og øvelser (<https://csunplugged.org/en/>),

**Paul Curzon** og **Peter W McOwan**, som gennem en lang årrække har arbejdet ud fra legende tilgange til Computer Science, og som helt tilbage i 2005, og altså før Wings tekst, startede projektet CS4Fun (<http://www.cs4fn.org/>), samt

**Michal Armoni** og **Mordechai Ben-Ari**, som begge hører til The Weizmann Institute of Science i Israel, og som sammen og hver for sig har forsket i og udgivet en mængde artikler om sammenhænge mellem Computer Science koncepter og CT (se fx<http://www.weizmann.ac.il/ScienceTeaching/Armoni/publications-and-presentations> og<http://www.weizmann.ac.il/sci-tea/benari/sites/sci-tea.benari/files/uploads/vit.pdf>)

**Yasmin B. Kafai**, som især har markeret sig i forhold til computationel tankegang inden for spil, sociale medier og e-tekstiler.

**Karen Brennan** og **Mitchell Resnick**s arbejde, som anlægger en “maker-drevet” konstruktionistisk og eksplorativ vinkel på begrebet. I 2012 har de defineret CT ved hjælp af tre nøgledimensioner:

|  |  |  |
| --- | --- | --- |
| https://lh5.googleusercontent.com/zp0Z3AtBwhJxzAaF9jKg1klLZLcM_GRGls1mFnUHI42fmQoBGf4G-tZF8ZyXrT5E_9NBRSdpQTEabWcbSdjRLpHOMMGJjngfFUr3HL1dsGsKOgozPEbGNau0MIGwbKMuNICSUTXk | https://lh3.googleusercontent.com/BYZkLJP-WIQ1zTHI6XmDx2yDbawBi8-er8BePL51MdsjMHLmmuwugQHbdP0V5Ni0II0gJco-70TnXou9rxoGnAnl7odqgh3hb3TUrUqk4soqQ8-_cpv0Ct9Y7tAlrbsfGCyn3GiX | https://lh4.googleusercontent.com/fNAOGx3N4ua9NkjF0JlEPZ4pv48KjRWlyYzJDBMvtnpFnRsPf00A96uomRCgwMuliSnESpSQtaiurZdyXx7HOYEGQMESrmOryw4xPEUbIwpHl1r7SIvOJhKVPhP13b_Kh-4o0Vai |
| COMPUTATIONAL CONCEPTSFundamentale elementer og strukturer, der bruges i programmering, fx sekvenser, løkker, data, betingelser, osv. | COMPUTATIONAL PRACTICESTilgange, som anlægges i en programmeringsproces, fx abstraktion, dekomposition (modularisering), fejlsøgning, m.m. | COMPUTATIONAL PERSPECTIVESDet syn på verden, som kommer til udtryk gennem computationel problemløsning |

Der er generelt en opfattelse af, at CT ligger ud over “programming”, forstået således, at der er tale om en robust kompetence, som er generisk, mens der med programmering menes en specifik færdighed i et eller flere sprog, i vores terminologi altså “kodning”. I sammenhænge, hvor der søges at opøve en generel programmerings**forståelse,** er det derfor relevant at fokusere på CT frem for specialiserende forløb i et eller flere programmeringssprog.

Der er fortsat ikke en helt fast definition af begrebet, men der fokuseres gennemgående på problemløsningskompetencer, og begrebet søges ofte defineret gennem forskellige underbegreber og tankeprocesser på samme måde, som Wings oprindelige tekst gjorde:
*“While computational thinking draws upon concepts that are fundamental to computing and computer science, it also includes practices such as problem representation, abstraction, decomposition, simulation, verification, and prediction (Sengupta et al., 2013). These practices, in turn, are also central to modelling, reasoning and problem-solving in a large number of scientific and mathematical disciplines (National Research Council, 2010). Einhorn (2012) states that computational thinking develops a variety of skills (logic, creativity, algorithmic thinking, modelling/simulations), involves the use of scientific methodologies and helps develop both inventiveness and innovative thinking.” Bower,M., Wood, L.N et al. s. 54 (2017)*

Som ovenstående viser, er det ikke helt enkelt at vælge underbegreber. I de fleste definitioner er der dog fire, som går igen, nemlig:

|  |  |  |  |
| --- | --- | --- | --- |
| https://lh4.googleusercontent.com/hlLK4REJF7PyG4MBH-O9BUeKu8RH5CbKtbKeK8AWZCSa8zWLqknJku1cFIsKo5v9AUEkhO3b19ozYqRqqKbfgAny_5kRUkj5WjIrZ-C-FqIL5b5i0bTxpJKun7LUPcIq6mXYAz1k | https://lh6.googleusercontent.com/Ekz3q4zp9cX8XCZPlNLLPdWFjsRQSkxJaefTqrOnvoZl77WFd_U4O2RdrjTyb8fzKHLYBBDV0mfehiVMxUXgMps_FuE5orhzZdkE0JmH2hofkib9H3BpktXVo_VZWA7IKdvpxXlN | https://lh3.googleusercontent.com/OWmisIa3TbfuYtFoag8zzoDToAQPihjjUMEKFM6ClYIsg4tRQCFb580GGRHe9be9jtM4aGxPAmeHvdeuj4wE62XHLUHfCsIf3oAx1K1EaHlRzlYea1p27WLbz_gXse4vJXyok6ye | https://lh3.googleusercontent.com/eQ1tBsakTFTeQ2cKP0jLE9YXYlZ8VeVewbb6QiLbFs9eoOM9s44T-Q9NPCGhKaxATobbAf62hHcj3qnWg_URlh0oEGl8I-l6piOMnNwEsKArM88I_0i3e-rXR9LkpoCpv-1AmMZe |
| Her arbejdes der grundlæggende med at forstå og udarbejde trinvise løsninger på problemer, dvs. algoritmer. Værktøjer som pseudokode og flowcharts kan anvendes til at beskrive en algoritme, før den kodes. | Her arbejdes med nedbrydning af et komplekst problem eller en problemløsning i mindre, mere håndterbare dele. Denne tankeproces har stærk sammenhæng med den algoritmiske tænkning og abstraktionen. | Mønstergenkendelse anvendes både i gentagelser i det enkelte program, fx løkker og procedurer, og på tværs af løsninger, hvor der fokuseres på ligheder mellem problemer og de algoritmer, der kan løse dem. | Her fokuseres på de vigtigste elementer, idet man ignorerer irrelevante detaljer. Virkelige fænomener abstraheres til modeller og måder at repræsentere disse på som data. |