Computational thinking i gymnasiefag

Jonas Ørbæk Hansen, Silkeborg Gymnasium og Center for Computational Thinking og Design (CCTD), Aarhus Universitet, Frode Peulicke, Gefion Gymnasium, Allan Jensen, Silkeborg Gymnasium, Solveig Skadhauge, Nærum Gymnasium

I forbindelse med gymnasiereformen (2016) er "Informatik" indført som nyt fag, og styrkelse af elevernes digitale dannelse og digitale kompetencer er et fokusområde på tværs af alle fag.

Denne artikel handler ikke om informatik som et selvstændigt fag, men om hvordan man kan integrere den del af informatikken, der omtales som *computational thinking* (CT), i matematik og de naturvidenskabelige fag.

> *"Informatik og computational thinking er internationalt hastigt i færd med at blive en del af almendannelsen i skolen på alle klassetrin; mange mener, at det er (eller snart bliver) en lige så væsentlig grundlæggende kompetence som læsning, skrivning og matematik."*
>
> *Direktør Michael E. Caspersen i "Gymnasiepædagogik – En grundbog", Caspersen (2017).*

Vi er en gruppe gymnasielærere, der i samarbejde med Aarhus Universitet [1] og Danske Science Gymnasier (DASG) [2] i skoleåret 18/19 arbejdede med at undersøge, hvordan vi kunne integrere CT i matematikundervisningen og i den naturvidenskabelige undervisning på en sådan måde, at det styrker elevernes CT–kompetencer, og *samtidig* bidrager til et øget fagfagligt udbytte af undervisningen. Målet var at integrere CT i faget, således at (CCTD 2018, CTiMNAT 2019):

- de faglige fænomener, begreber og principper anvendes til at lære CT
- de CT–faglige fænomener, begreber og principper anvendes til at lære faget.

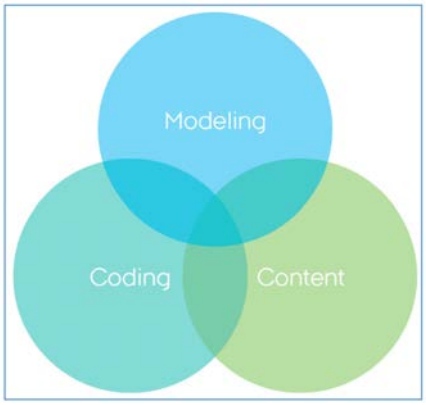
I den forbindelse udviklede og afprøvede vi flere undervisningsforløb i fagene matematik, fysik, kemi, biologi, bioteknologi og NV. I artikler i dette blad præsenterer vi et lille udpluk af disse forløb. De anvendte modeller og materialer kan findes på library.ct-denmark.org/lmfk.

Skal give faglig mening

Det var vigtigt for os, at CT–aktiviteterne understøtter den faglige læring, så CT ikke bliver et nyt undervisningsmæssigt vedhæng, men en sund faglige indsprøjtning. Vi valgte derfor at fokusere på *modellering og simulering*, da det at kunne modellere et fagligt fænomen og forholde sig kritisk til en models muligheder og begrænsninger er en central kompetence i mange fag – ikke mindst i matematik og de naturvidenskabelige fag. Dette fælles fokus på modelleringskompetencen betød, at vi kunne følge en fælles tilgang og udveksle erfaringer på tværs af fag og faglige emner.

I planlægningen af forløbene tog vi udgangspunkt i CMC–tilgangen (Figur 1), som er udviklet af medarbejdere ved Center for Computational Thinking og Design ved Aarhus Universitet (Musaeus 2019). Her kombineres fagligt indhold (*content*) med modellering (*modeling*) og kodearbejde (*coding*). Et undervisningsforløb kan typisk se ud som følger:

Læreren vælger et centralt fagligt fænomen og finder eller udvikler en simpel computermodel over fænomenet, som udleveres til eleverne. Det forudsættes ikke, at eleverne på forhånd kender til computerkode. De starter "forudsætningsløst" ved at lære at bruge computermodellen gennem dens interface. Derefter får de adgang til den bagvedliggende kode. Vi har anvendt programmeringsmiljøet NetLogo, der er udviklet til undervisningsbrug (se boksen nedenfor). Det er forholdsvis let for eleverne at lære at læse koden og trække mening ud af den, hvorefter de kan gå i gang med selv at ændre, tilrette eller forbedre modellen ved at ændre i koden ud fra egne faglige overvejelser. NetLogo ligner andre programmeringssprog tilpas meget til, at dét eleverne lærer i NetLogo (fx mht. et programs opbygning, programstrukturer, kodesyntaks og algoritmer) kan overføres til andre programmeringssprog. Dette indblik i koden bag computermodellen, som eleverne får ved at arbejde direkte med koden, bruges som et afsæt til en faglig vurdering og diskussion af modellens styrker og begrænsninger.



Figur 1
Ved CMC–tilgangen kombineres faglig viden (Content) med modellering (Modeling) og programmering (Coding).

Fordelen ved CMC–tilgangen er, at faget gør arbejdet med CT relevant og meningsfyldt for eleverne, mens CT–kompetencerne muliggør at arbejde med faget på en anderledes måde og undersøge komplekse sammenhænge, problemstillinger og fænomener, som eleverne ikke normalt har mulighed for at arbejde i dybden med (fx på grund af svær matematik). Desuden bliver det muligt for eleverne at arbejde aktivt og konstruktivt med modellering, så de får en tydeligere opfattelse af, hvorfor modeller er så vigtige i forbindelse med at forstå og beskrive faglige fænomener. Der arbejdes altså med alle delene i Figur 1 sideløbende, således at de forskellige dele integreres og understøtter hinanden undervejs. Det står i kontrast til den tilgang, som vi selv har mødt i vores egne universitetsstudier, hvor programmering blev inddraget som et selvstændigt kursus, som var af-

[1] Center for Computational Thinking og Design, Aarhus Universitet, cctd.au.dk
[2] Støttet af Villum Fonden.

koblet fra resten af indholdet på studiet, og som det kunne være svært at se relevansen af i den givne situation.

Vi har i udviklingen af forløbene tilstræbt at følge nogle generelle designprincipper. De vigtigste er ridset op her:

- *Fagligt enkle modeller:* Computermodellerne skal være enkle. Det betyder ofte, at de kun omfatter et enkelt eller få aspekter ved det modellerede fænomen. Det gør det muligt at holde koden *kort og lettilgængelig*. De fagligt enkle modeller og den lettilgængelige kode er en forudsætning for, at eleverne kan finde ud af, hvordan de enkelte dele af koden styrer modellens opførsel. De fagligt enkle modeller har desuden den fordel, at eleverne selv kan indse, at der er truffet nogle valg om forsimpling i forbindelse med modellen. Det kan give anledning til en sund faglig diskussion i klassen om modellens begrænsninger. En diskussion der naturligt kan munde ud i den vigtige pointe, at den slags valg træffes i forbindelse med al modellering – men at det kan være svært at gennemskue, eller man glemmer at tænke på det. I mange tilfælde vil det være muligt for eleverne efterfølgende at forbedre modellen i forhold til diskussionen af dens begrænsninger. Det har stor betydning for elevernes motivation, at de føler, at de ændringer, de laver i koden, fører til reelle forbedringer af modellen. Det giver dem en stor følelse af ejerskab over modellen. Således bliver det, der set med faglige øjne i første omgang kan være problematisk ved en model – at den er overdrevent forenklet – pædagogisk set en styrke, der ikke spænder ben for den faglige læring – *tværtimod*.
- *Use–modify–create:* Undervisningsaktiviteterne følger progressionen vist i Figur 2. Eleverne starter med at bruge en udleveret computermodel gennem dens interface (*use*) – fx ved at undersøge hvordan forskellige startparametre har indflydelse på modellens opførsel. Derefter får de adgang til den bagvedliggende kode. De første ændringer i koden (*modify*) er simple ændringer, typisk nogle der ændrer den visuelle repræsentation af fænomenet. Det er kommet som en positiv overraskelse for mange af os, hvor stor en effekt det kan have på eleverne, når de laver selv simple ændringer i koden, så en del af modellen ændrer udseende. Der er ofte stor tilfredshed og begejstring at spore, og denne indledende mestringsoplevelse er et vigtigt trinbræt for de næste, mere krævende kodeaktiviteter. Gradvist bliver kodeaktiviteterne mere omfattende. De kræver, at eleverne

NetLogo og agentbaseret modellering (ABM)

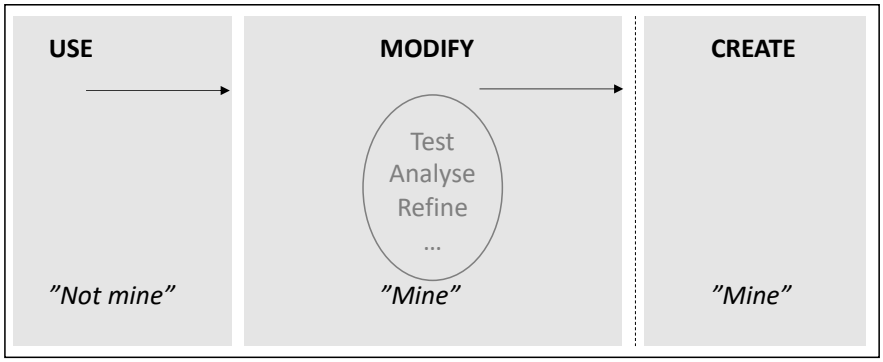
Computermodellerne i forløbene er udviklet i programmeringsmiljøet NetLogo (Tisue 2004). NetLogo er velegnet til CMC–tilgangen, hvor modellering og kodearbejde spiller en central rolle. NetLogo er et tekstbaseret programmeringssprog, der er designet, så det er forholdsvist simpelt og let at gå til for både elever og lærere – også uden forudgående kendskab til programmering. Syntaksen er relativ let at forstå, og det kræver en minimal mængde kode at lave grafer og animationer. NetLogo egner sig især godt til at modellere fænomener, der udvikler sig over tid.

Netlogo er *agentbaseret*, hvilket vil sige, at programmeringen af en computermodel tager udgangspunkt i de enkelte "agenter" i modellen (det kan være atomkerner, kanonkugler, fugle, molekyler, etc.). Det centrale i programmeringen består i at definere agenterne og udstyre dem med de rette "egenskaber" (fx udseende, position, retning og fart) og "adfærd" (fx bevægelse, frastødning/tiltrækning, kemisk reaktion, henfald, etc.). Agenternes *egenskaber* og *adfærd* definerer, hvordan de ser ud og opfører sig, samt hvordan de interagerer med hinanden og deres omgivelser.

Med agentbaseret modellering kan store, komplekse fænomener reduceres til et spørgsmål om, hvordan de enkelte agenter agerer i bestemte situationer. Det gør arbejdet med og programmeringen af modeller meget intuitivt og let at gå til for eleverne og giver eleverne mulighed for at arbejde med relativt komplekse fænomener, som ellers ville være vanskelige at behandle i undervisningen.

NetLogo er dermed et ideelt værktøj til at arbejde med modellering og til at undersøge, hvilke interessante *makroskopiske fænomener der opstår ud fra agenternes mikroskopiske adfærd* (CCTD 2018), og dermed belyse sammenhængen mellem mikroskopiske og makroskopiske modeller (emergerende fænomener).

NetLogo kan installeres fra ccl.northwestern.edu/netlogo. Når du åbner NetLogo, kan du under *File* og *Models Library* finde et bibliotek af modeller til forskellige fag. De NetLogo–modeller, der omtales i artiklerne i dette blad, kan findes på library.ct-denmark.org/lmfk. Her ligger også de anvendte undervisningsmaterialer. Endnu flere færdige forløb udviklet af danske gymnasielærere kan findes på library.ct-denmark.org.



Figur 2
Illustration af use–modefy–create–tilgangen (inspireret af Lee 2011). Nederst i figuren er angivet, hvordan elevernes følelse af ejerskab for computermodellen ændres undervejs i læringsaktiviteterne.

forholder sig til selve modelleringen af fænomenet, fx agenternes egenskaber og adfærd (se boksen nedenfor) og hvordan de hænger sammen med modellens opførsel. I løbet af *modify*–fasen ser vi ofte, at computermodellen i elevernes bevidsthed går fra at være noget, andre har lavet (*not–mine*), til at være noget, de føler ejerskab for (*mine*) – nogle elever føler nærmest, at de har lavet hele modellen selv. Det er sjældent, at eleverne når til at lave en computermodel selv (*create*). Men det er som regel inden for rækkevidde at snakke med eleverne om, at det, som de nu har lært om modeller og modellering, kan generaliseres. Meget af det, de har lært, gælder for *al* modellering, også i andre fag og på andre områder.

Faglig og almen dannelse

Fordi computermodellering er så vigtig en metode for den forskningsmæssige udvikling i fagene, er det en forudsætning for elevernes faglige dannelse, at de opnår en basal forståelse for, hvad computermodeller er, og hvordan de benyttes i moderne naturvidenskab.

Det er vigtigt, at eleverne bliver bevidste om, at computermodeller er *tilnærmelser* af virkeligheden og bliver klædt på til at kunne vurdere en computermodels muligheder og begrænsninger og reflektere over de antagelser, forsimplinger, valg og fravalg, der er gjort i modelleringen af fænomenet – om en model er "god" afhænger udelukkende af, hvad man anvender den til. Skal eleverne kunne dette, er det nødvendigt med et grundlæggende kendskab til programmering og en forståelse for algoritmer, og de skal arbejde med computermodeller, hvor de har adgang til koden bag computermodellen (ikke tilfældet for de fleste online simuleringsværktøjer).

Det er vores overbevisning, at aktiviteterne i de beskrevne forløb er med til at styrke elevernes faglige dannelse og studieforberedende kompetencer. Men der er også et almendannede perspektiv i aktiviteterne:
Det er ikke kun i de naturvidenskabelige og matematiske fag, at computermodeller spiller en stor rolle. Computermodeller har afgørende betydning for, hvordan beslutninger tages i vores samfund. Der er en pointe i at gøre eleverne opmærksomme på dette og præsentere dem for eksempler på, hvor og hvordan computermodeller bliver anvendt i samfundet – fx økonomiske modeller som DREAM og ADAM, klima– og vejrmodeller, styrkeberegningsmodeller til byggekonstruktioner eller modeller anvendt i sundhedsvæsenet. Som i faget har det afgørende betydning for computermodellens resultater, hvad man valgt at tage med i computermodellen, og hvad man har valgt at udelade.

De CT–kompetencer, som eleverne (forhåbentligt) udvikler, vil være med til at øge deres mulighed for at kunne forstå og forholde sig vurderende og kritisk til – samt medvirke til at forme – den digitaliserede virkelighed, som de er en del af – både fagligt og alment.

Computermodeller har afgørende betydning for, hvordan beslutninger tages i vores samfund. Vores sundhed, sociale relationer, adgang til nyheder, politiske beslutninger, m.m. påvirkes af de resultater, som computermodeller spytter ud.

Vi har her præsenteret, hvordan vi med udgangspunkt i CMC–tilgangen har arbejdet med at integrere CT i den faglige undervisning i gymnasiet. Det centrale har været, at det skal gøres på en måde, så det giver fagligt mening. At eleverne træner væsentlige CT–kompetencer, *samtidig* med at de styrker deres faglige læring. Udgangspunktet i faglig modellering har vist sig at være et godt fundament for at arbejde med at udvikle elevernes CT–kompetencer, og for at give dem en forståelse af computermodellers rolle og enorme betydning både for faget og for samfundet. I artikler her i bladet gives nogle konkrete bud på, hvordan det kan gøres, og hvad eleverne får ud af det – både fagfagligt og CT–fagligt.

De NetLogo–modeller, der omtales i artiklerne i dette blad, kan findes på library.ct-denmark.org/lmfk. Her ligger også de anvendte undervisningsmaterialer. Endnu flere færdige forløb udviklet af danske gymnasielærere kan findes på library.ct-denmark.org.

Kan man som lærer anvende NetLogo i undervisningen, hvis man ikke i forvejen

Computational thinking (CT)

Begrebet computational thinking (CT) stammer fra Seymour Papert, der anvendte begrebet første gang i 1980 (Papert 1980, Papert 1996). Papert beskriver CT som:

Computational Thinking is the use of programming – as an extension of our mind – to experience and understand the world, to manipulate the world, and to create things that matter to us.

Papert ser altså CT som de kompetencer, der skal sætte os i stand til at bruge en computer til at *tænke*, *lære* og *skabe* med. Han ser CT som en ny erkendelses– og udtryksform, en udvidelses af vores mentale evner.

I 2006 relancerede Jeannette M. Wing begrebet (Wing 2006, Wing 2014), og det er siden da blevet udbredt, ikke mindst i undervisningssystemet verden over (Caspersen 2017, Caspersen 2018). Wing definerer CT som:

Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer can carry it out.

Det er to forskellige vinkler, der komplementerer hinanden. Wing m.fl. argumenterer for, at CT dækker over nogle almene basiskompetencer, som alle – børn som voksne – bør tilegne sig fra barnsben (Wing 2006, Wing 2014, Caspersen 2017, Caspersen 2018, Google 2018). Der er forskellige bud på, hvad disse almene kompetencer/tankeprocesser dækker over, men disse går igen mange steder:

- **Nedbrydning:** Nedbryde data, processer eller problemer i mindre, håndterbare dele.
- **Mønstre og generaliseringer:** Finde mønstre, tendenser og regelmæssigheder i data.
- **Abstraktion:** Identificere de generelle principper, der genererer disse mønstre. Kunne oversætte mellem forskellige repræsentationsformer.
- **Algoritmisk tænkning:** Udvikle trin–for–trin instruktioner til at løse dette og lignende problemer.
- **Evaluering:** Teste, analysere, vurdere og forbedre sit produkt løbende.

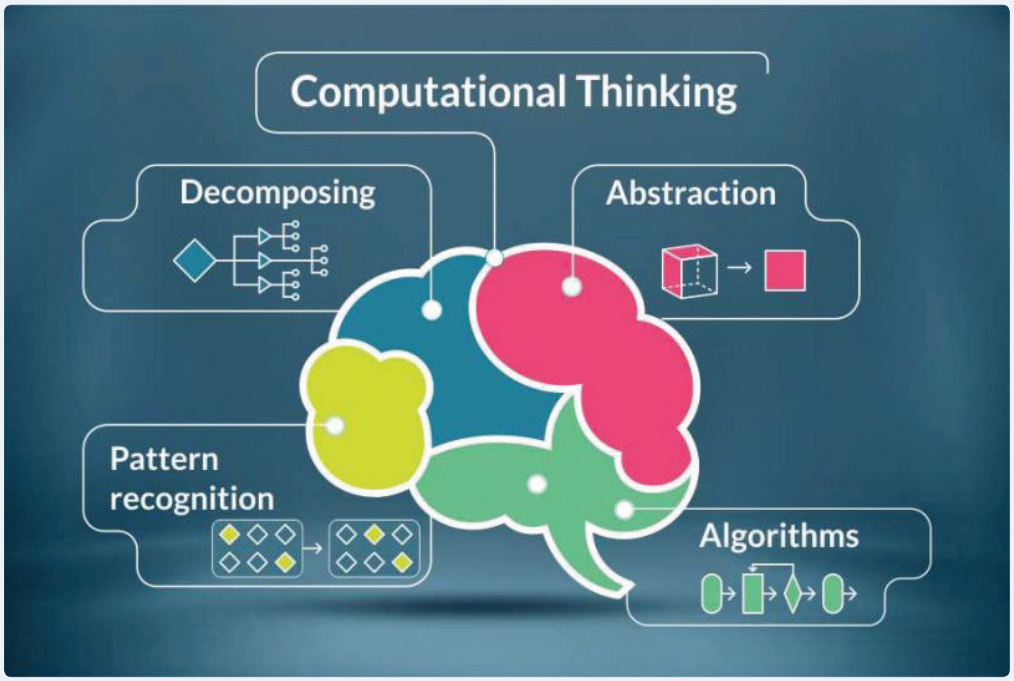
Hertil kunne man tilføje evnen til at beskrive problemer og problemløsningsstrategier meget præcist, detaljeret og struktureret.

Computational thinking er således andet og mere end blot programmering. Begrebet dækker over en bred vifte af kompetencer, der gør os i stand til at løse problemer. Michael E. Caspersen beskriver det som (Caspersen 2017):

"… evnen til at repræsentere, analysere, bearbejde og præsentere data og dataprocesser gennem passende abstraktioner i modeller og simuleringer, samt automatiseret problemløsning gennem algoritmisk tænkning."

Computational thinking skal altså forstås som en arbejdsform og en tænkeform, som kan indgå i alle fag og ikke kun i de matematiske og naturvidenskabelige. Computational thinking handler ikke om at lære elever at programmere for programmeringens skyld, men at eleverne – gennem det at lære at programmere – udvikler deres problemløsende kompetencer og abstrakte tænkning og lærer at se på problemer, emner og verden på en ny måde (Petropouleas 2018).

Figur 3
Computational thinking dækker over nogle almene basiskompetencer, som kan hjælpe eleverne til at løse problemer på tværs af fag og discipliner.



har erfaring med programmering? Ja, det kan man godt, især hvis man benytter eller tager udgangspunkt i de færdige forløb, der allerede er udviklet til gymnasiebrug. NetLogo har en relativ lav indlæringstærskel. Vores kollegaer har ved flere lejligheder anvendt vores forløb i deres egen undervisning.

Interesseret i at lære mere om CT i gymnasiefag?
I skoleåret 2020/21 tilbyder DASG i samarbejde med CCTD igen kurset *Computational Thinking i Matematik og Naturfag*, som vi var på i 2018/19 – her vil NetLogo blive introduceret sammen den didaktik, som ligger til grund for de omtalte forløb. Du kan læse kursusbeskrivelsen her: sciencegym.dk/kurser/20202021/ct3.htm. Deadline for tilmelding er midt i marts.

Du kan læse mere om CT i gymnasieundervisningen i udgivelserne *Computational Thinking – hvorfor, hvad og hvordan*? (Caspersen 2018) og *Didaktik for Computational Thinking i Gymnasiefag* (Nielsen, 2020). Sidstnæsnte er et didaktikhæfte for CT–i–fag rettet mod gymnasiet.

Vi vil afslutningsvis henlede opmærksomheden på et tilsvarende projekt i fagene samfundsfag, historie, musik, engelsk og dansk (MCTiG 2019).

Litteraturliste:

Caspersen, Michael E. (2017): "Computational thinking", in: Dolin, Jens (red.) m.fl. (2017): *Gymnasiepædagogik – En grundbog*, Hans Reitzels Forlag, 3. udg., s. 470 – 478.

Caspersen, Michael E. m.fl. (2018): *Computational Thinking – hvorfor, hvad og hvordan*? Link: it-vest.dk/fileadmin/user_upload/pdf/2018-12-18--Computational-Thinking--hvorfor-hvad-og-hvordan--PRINT-2-sided.pdf.

CCTD (Center for Computational Thinking and Design), Aarhus Universitet (2018): *CT I Gymnasiefag (afsluttende rapport)*. Link: cctd.au.dk/fileadmin/user_upload/CT_i_Gymnasiefag-afslutningsrapport-ur.pdf.

CTiMNAT (2019): Projektet *Computational thinking i matematik og naturfag*. Link: cctd.au.dk/projects/ctimnat-computational-thinking-in-math-and-science

Google (2018): *Computational Thinking for Educators*. Link: computationalthinkingcourse.withgoogle.com/course.

Lee, Irene m.fl. (2011): "*Computational Thinking for Youth in Practice*", in: *ACM In–roads*, Vol. 2, No. 1, s. 32 – 37.

MCTiG (2019): *Projektet Modellering og computational thinking i gymnasiefag*. Link: cctd.au.dk/projects/mctig-computational-thinking-in-humanities-arts-and-social-sciences

Musaeus, Line H. og Musaeus, Peter (2019): "*Computational Thinking in the Danish High School: Learning Coding, Modeling, and Content Knowledge with NetLogo*", in: *SIGCSE '19: Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, s. 913 – 919.

Nielsen, Keld m.fl. (2020): *Didaktik for Computational Thinking i Gymnasiefag* (udgives i april og kommer til at ligge sammen med de andre online materialer).

Papert, Seymour (1980): *Mindstorms: Children, computers, and powerful ideas*, Basic Books.

Papert, Seymour (1996): *"An exploration in the space of mathematics educations"*, in *International Journal of Computers for Mathematical Learning*, Vol. 1, No. 1, s. 138 – 142. Link: papert.org/articles/AnExplorationintheSpaceofMathematicsEducations.html.

Petropouleas, Eva (2018): *Computational Thinking*. Link: mv-nordic.com/dk/bloggen/artikel/computational-thinking.

Tisue, Seth og Wilensky, Uri (2004): *"Netlogo: A simple environment for modeling complexity"*, in: *International conference on complex systems*, Vol. 21, s. 16 – 21.

Wing, Jeannette M. (2006): *"Computational thinking"*, in: *Communications of the ACM*, Vol. 49, No. 3, s. 33 – 35.

Wing, Jeannette M. (2014): *"Computational Thinking Benefits Society"*, in: *Social Issues in Computing* (2014). Link: http://socialissues.cs.toronto.edu/index.html%3Fp=279.html.