SDU

sdu.dk

#c_lct

Computational thinking – teoretiske, empiriske og didaktiske perspektiver

Trin to i boglancering, første trin var et online seminar den 25. januar

Program (tilpasset)

12.15-12.50:	Præsentation af bogen
12.50-13.00:	Pause
13.00-13.50:	Hovedpointer fra kap. 1, 5 og 10
13.50-14:10:	Pause
14.10-14.55:	Hovedpointer fra kap. 3, 6 og 12
14.55-15.05:	Pause
15.05-15.50:	Diskussionsoplæg Lis Zacho, lærer og medlem af hovedbestyrelsen i Coding Pirates
15.50-16.05:	Pause
16.05-16.50:	Hovedpointer i kap. 2, 8 og 11
16.50-17.00:	Pause
17.00-17.45:	Diskussionsoplæg Anders Mørch, professor ved Institutt for pedagogikk, Universitetet i Oslo
17.45-18.00:	Afrunding, tak for i dag

Hvorfor en bog om Computational Thinking?

- Computational Thinking (CT) er i uddannelsespolitisk fremmarch
  - Et centralt punkt i de nye fag Teknologiforståelse (grundskole) og Informatik (gym)
  - Væsentligt tema i OECDs liste over "21. århundredes kompetencer"
  - Hævdes at være den 4. grundlæggende kompetence
    - Som læsning, skrivning og regning: en kompetence, der er nødvendig for at nå andre mål
    - Fortalere herfor er fx af Jeanette Wing og Michael Caspersen

- Samtidig er det til debat – med kun begyndende forskningsunderbygning
  - Hvordan CT skal karakteriseres i forhold til andre former for tænkning
  - Hvilken rolle det computationelle – og it – har i CT
  - Hvordan vi kan forstå andre fagområder med CT
  - Hvordan man lærer CT, og hvordan læring kan understøttes didaktisk
  - Hvordan vi didaktisk kan understøtte læring af andre fagområder med CT

Hvorfor har *vi* skrevet en bog om Computational Thinking?

Hvem er vi?

- Forskere ved IDK, tilknyttet Center for Learning Computational Thinking
  - CLCT er samarbejde mellem fire institutter på tre fakulteter, ca. 30 personer
  - CLCT har fokus på udvikling af CT-didaktik og involvering af ”computationelle ting” i læring af og med CT
  - Informationsvidenskab, webarkitektur, filosofi, læringsteori, didaktik, lingvister
  - Vi arbejder med kropslig læring, tænkning, it-didaktik, fagsystematik, it-design

Hvilke mål har vi for bogen?

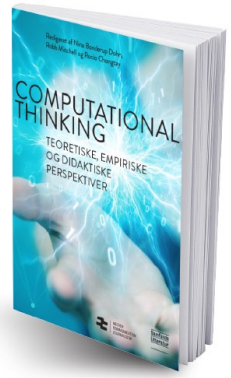
- Begrebsudvikle CT – idehistorisk og kritisk karakteristik af CT
- Anskueliggøre, om og hvordan CT kan bidrage på forskellige fagfelter
- Udvikle didaktik for og med CT

CT Historical Timeline

Although Wings's influential 2006 essay popularized the concept of Computational Thinking, one can trace some of the very central elements of CT back to ancient times (1800 BC).

Here we present a historical CT timeline that is divided by shifts on the relationship between *problems, tools and human thinking*, these shifts are as follows:

a) **The time of automated computation procedures before the electronic computer,**
b) **The time after the electronic computer,**
c) **The time after Wings popularization of CT.**



CT Historical Timeline

a) The time of automated computation procedures before the electronic computer.

Ancient precursors

Mathematicians devised step by step instructions for complex computations*
For other to use, similar to today's algorithms.

Early machines for automating computational procedures

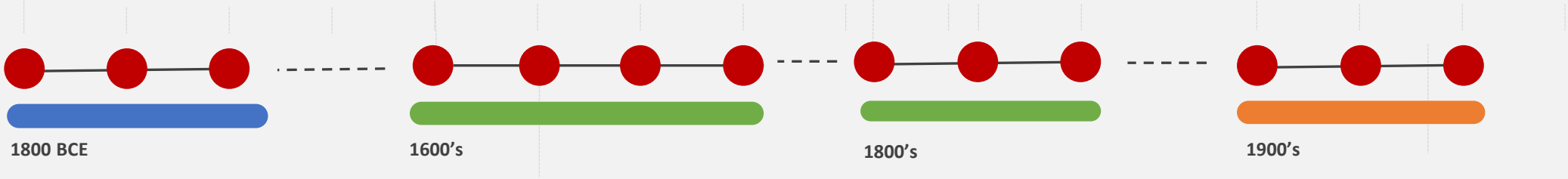
John Napier (1617),
Edmund Gunther (1620),
Blaise Pascal (1642),
Gottfried Leibniz (1671)

Charles Babbage (1819, 1840)

Ada Lovelace (1843)

Abstract computational machines and problem solving theorists.

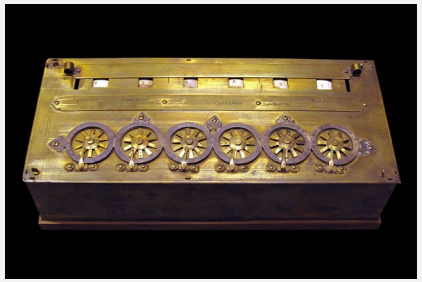
Alan Turing (1936),
Karl Duncker (1945),
George Polya (1945)



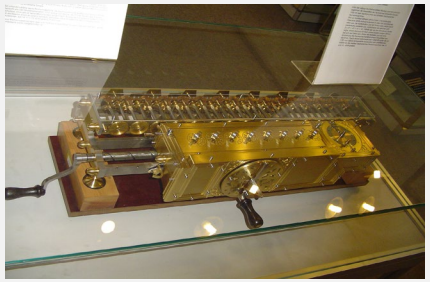
* computations – complex numerical calculations and symbol manipulations

Relationship between problems, tools and human thinking

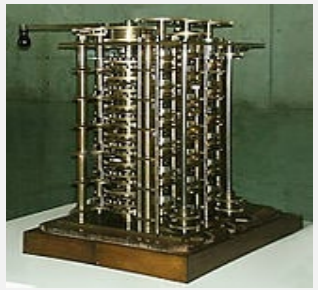
Napier's bones (1617) multiplication(additions) division (substractions) to help writing logarithmic tables



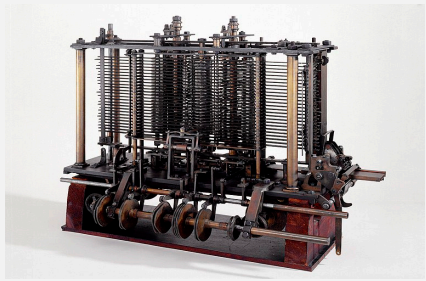
Pascal's pascaline (1642) add and substract and by repetition of these multiply and divide



Liebniz's wheels (1671) Added automatic multiplication to the pascaline

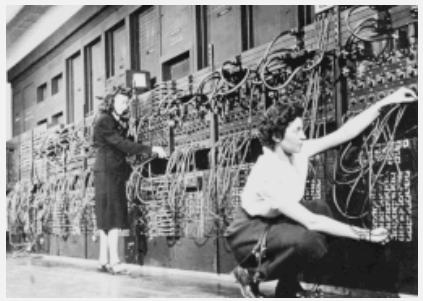


Babbage's difference engine (1819) Logarithms



Babbage's analytical engine (1840) Universal calculator + Ada Lovelace - Information processing machine (numbers, letters and symbols) (1843).

Turing's electro-mechanical deciphering machine (1938)



ENIAC, considered the first programmable, electronic, general-purpose digital computer (1945)

Electronic computer elements: input, processing functions, memory and output

Newell et al. (1958) modelled human problem-solving behaviour as an analogy with computers as information processing systems.

* Images from Wikimedia.org

CT Historical Timeline

b) The time after the electronic computer,

c) The time after Wings popularization of CT.

Problems, tools and human thinking, Computer Sciences - Datalogi, Programming languages and formal thinking, CS problem-solving practices

Peter Naur (1965),
Seymour Papert,
Wallace Feurzeig and
Daniel Bobrow (1966),
George Forsythe (1968)

1960's

Elements of Computational Thinking (reduction, generalization, abstraction and algorithms). Mathematical- CS problem solving

Symour Papert (1980),
Donald Knuth (1981)

1980's

Peter Denning (1999),
Problem solving: Algorithmic thinking, representation, programming and design.

1990's

Computational Thinking, reuse, remix

Mitchel Resnick, Yasmin Kafai, and John Maeda (2003),

Jeannette M. Wing (2006),

2000's

"There is a fundamental interaction and a mutual determination between problems, tools and human thinking". (Naur, 1965).

Overblik over bogen

Formål med bogen (igen)

- Begrebsudvikle CT – idehistorisk og kritisk karakteristik af CT
- Anskueliggøre, om og hvordan CT kan bidrage på forskellige fagfelter
- Udvikle didaktik for og med CT

Bogen har 3 dele, hver med særligt fokus på ét af disse formål

- Del 1: CT på landkortet – historisk og systematisk
  - Teoretiske perspektiver, adresserer især begrebsudviklingsformål
- Del 2: Computationelle problemer og hvordan man løser dem
  - Empiriske perspektiver, adresserer især anskueliggørelsesformål
- Del 3: Didaktisk design for Computational Thinking
  - Didaktiske perspektiver, adresserer især formål om udvikling af didaktik

Overblik over bogen

Del 1: CT på landkortet – historisk og systematisk

- Vi definerer CT som *de kognitive processer, som er involveret i udviklingen af it-artefakter og programmer til at leve i verden i dag*.
- CT har analoge, digitale, fysiske og kropsligt funderede former
  - Illustreres med cykling: en kropsligt baseret, dynamisk og relationel proces
- CT som sekventiel informationsbehandling har begrænsninger
  - Fx har en robotingeniør brug for en lang række andre kompetencer også
    - (sekventiel informationsbehandling udtømmer ikke ”thinking like a computer scientist”)

1. Computational Thinking – indplacering i et landskab af it-begreber
2. Computational Thinking – et idehistorisk rids
3. Cyklling som kropsbaseret computationel praksis
4. Computational Thinking og uddannelsen af fremtidens dataloger

Overblik over bogen

Del 1: CT på landkortet – historisk og systematisk

- Vi definerer CT som *de kognitive processer, som er involveret i udviklingen af it-artefakter og programmer til at leve i verden i dag*.
- CT har analoge, digitale, fysiske og kropsligt funderede former
  - Illustreres med cykling: en kropsligt baseret, dynamisk og relationel proces
- CT som sekventiel informationsbehandling har begrænsninger
  - Fx har en robotingeniør brug for en lang række andre kompetencer også
    - (sekventiel informationsbehandling udtømmer ikke ”thinking like a computer scientist”)

I dag præsenterer vi:

- Systematisk indplacering af CT ift. it-kompetencer (kapitel 1)
- Idehistorisk indplacering af CT ift. tænkning (kapitel 2)
- Analyse af cykling som CT (kapitel 3)

Overblik over bogen

Del 2: Computationelle problemer og hvordan man løser dem

- Anskueliggørelse af CT som problemløsningsmetode
  - På makroplan: oversigt over implementering af CT i forskellige fag de sidste 10 år
  - På mikroplan: sammenstilling af CT med fagspecifikke metoder
    - Tekstkonstruktion og terminologisk begrebsarbejde
- Vi viser anvendeligheden af CT inden for humaniora
  - – og at CT ikke kan stå alene
- Kritisk stillingtagen til problemer og løsninger er nødvendig

5. Computational Thinking som redskab til problemløsning på tværs af fagområder
6. For mennesker, maskiner og medier – indholdsforfatteren som computational thinker
7. Computational Thinking i terminologisk begrebsarbejde
8. Algoritmisk dannelse

Overblik over bogen

Del 2: Computationelle problemer og hvordan man løser dem

- Anskueliggørelse af CT som problemløsningsmetode
  - På makroplan: oversigt over implementering af CT i forskellige fag de sidste 10 år
  - På mikroplan: sammenstilling af CT med fagspecifikke metoder
    - Tekstkonstruktion og terminologisk begrebsarbejde
- Vi viser anvendeligheden af CT inden for humaniora
  - – og at CT ikke kan stå alene
- Kritisk stillingtagen til problemer og løsninger er nødvendig

I dag præsenterer vi:

- Oversigt over CTs implementering i forskellige fag (kapitel 5)
- Sammenstilling af CT med tekstkonstruktion (kapitel 6)
- Nødvendigheden af algoritmisk dannelse (kapitel 8)

Overblik over bogen

Del 3: Didaktisk design for Computational Thinking

- Skelnen mellem læring *af* CT og læring *med* CT
- Analyse af implicitte didaktiske begrundelser for begge dele
- Identifikation af didaktiske fokus for CTs konkretisering i forskellige fag
- Præsentation af didaktiske designs for læring af CT, der
  - anvender kropsligt, socialt og fysisk medierede læringsaktiviteter
  - sigter på at etablere forståelse af computerens perspektiv og algoritmiske processer

9. Teknologiforståelsesdidaktik – problemløsning, empowerment, eksistens, udfordring og innovation
10. Didaktiske fokuspunkter i design for læring af og læring med Computational Thinking
11. Kropsbaseret Computational Thinking
12. Læring af Computational Thinking gennem Storycoding

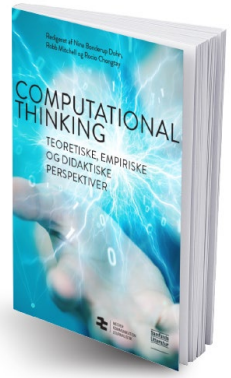
Overblik over bogen

Del 3: Didaktisk design for Computational Thinking

- Skelnen mellem læring *af* CT og læring *med* CT
- Analyse af implicitte didaktiske begrundelser for begge dele
- Identifikation af didaktiske fokus for CTs konkretisering i forskellige fag
- Præsentation af didaktiske designs for læring af CT, der
  - anvender kropsligt, socialt og fysisk medierede læringsaktiviteter
  - sigter på at etablere forståelse af computerens perspektiv og algoritmiske processer

I dag præsenterer vi:

- Skelnen mellem læring *af* CT og læring *med* CT (kapitel 10)
- Didaktiske fokus for CTs konkretisering (kapitel 10)
- Nogle af de didaktiske designs (kapitel 11 og 12)



Præsentation af kapitlerne 1, 5 og 10

Nina Bonderup Dohn,

Professor i læring og IT

Rocio Chongtay,

Lektor i informationsvidenskab

- Systematisk indplacering af CT ift. it-kompetencer
  - Definition af CT
  - Faser og kompetencer for CT
  - Model over CT-former
- Oversigt over CTs implementering i forskellige fag
- Didaktiske fokus for CTs konkretisering i fag
  - Skelnen mellem læring *af* CT og læring *med* CT
  - Helhedsforståelse versus algoritmisk tænkning
  - Skabelon til design af CT-forløb

Bogens definition af CT

Computational thinking er de kognitive processer, som er involveret i udviklingen af it-artefakter og programmer til at leve i verden i dag.

Bogens definition af CT

Computational thinking er de kognitive processer, som er involveret i udviklingen af it-artefakter og programmer til at leve i verden i dag.

- Lægger op til at se på it-udvikling i kontekst: udvikling af *passende* it

Bogens definition af CT

Computational thinking er de kognitive processer, som er involveret i udviklingen af it-artefakter og programmer til at leve i verden i dag.

- Algoritmisk tænkning
- Forståelse for de roller, som algoritmer spiller positivt og negativt
- Embodied cognition, distribueret tænkning, systemtænkning…

Definition gennem udpegning

- De samme kognitive processer kan potentielt være til stede i andre sammenhænge også

Bogens definition af CT

Computational thinking er de kognitive processer, som er involveret i udviklingen af it-artefakter og programmer til at leve i verden i dag.

- **Bred definition**
- Flere af bogens kapitler afgrænser fokus til CT som *sekventiel problemløsning*

CT som sekventiel problemløsning – faser og kompetencer

Fase	Kompetencekarakteristik
Problemformulering	
Datagenerering og -behandling	
Modellering	
Algoritmedesign	
Automatisering	
Generalisering	

CT som sekventiel problemløsning – faser og kompetencer

Fase	Kompetencekarakteristik
Problemformulering	- Abstraktion af problem fra konkret situation - Dekomposition af problem i mindre, håndterbare dele
Datagenerering og -behandling	
Modellering	
Algoritmedesign	
Automatisering	
Generalisering	

CT som sekventiel problemløsning – faser og kompetencer

Fase	Kompetencekarakteristik
Problemformulering	- Abstraktion af problem fra konkret situation - Dekomposition af problem i mindre, håndterbare dele
Datagenerering og -behandling	- Dataskabelse og data indsamling, klargøring af data til analyse - Dekomposition af data, dvs. logisk dataanalyse og -organisering
Modellering	
Algoritmedesign	
Automatisering	
Generalisering	

CT som sekventiel problemløsning – faser og kompetencer

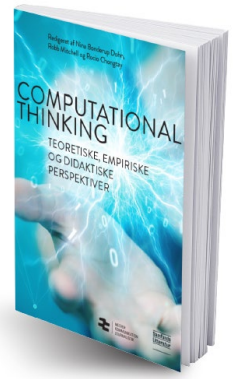
Fase	Kompetencekarakteristik
Problemformulering	- Abstraktion af problem fra konkret situation - Dekomposition af problem i mindre, håndterbare dele
Datagenerering og -behandling	- Dataskabelse og data indsamling, klargøring af data til analyse - Dekomposition af data, dvs. logisk dataanalyse og -organisering
Modellering	- Abstraktion af træk som de vigtigste - Mønstergenkendelse/-skabelse ud fra disse træk - Modelskabelse
Algoritmedesign	
Automatisering	
Generalisering	

CT som sekventiel problemløsning – faser og kompetencer

Fase	Kompetencekarakteristik
Problemformulering	- Abstraktion af problem fra konkret situation - Dekomposition af problem i mindre, håndterbare dele
Datagenerering og -behandling	- Dataskabelse og data indsamling, klargøring af data til analyse - Dekomposition af data, dvs. logisk dataanalyse og -organisering
Modellering	- Abstraktion af træk som de vigtigste - Mønstergenkendelse/-skabelse ud fra disse træk - Modelskabelse
Algoritmedesign	- Opstilling af trin-for-trin-handlingssekvens
Automatisering	
Generalisering	

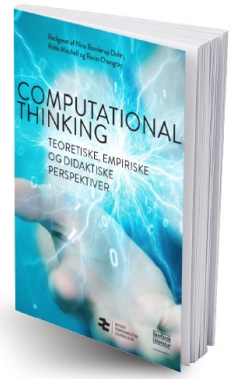
CT som sekventiel problemløsning – faser og kompetencer

Fase	Kompetencekarakteristik
Problemformulering	- Abstraktion af problem fra konkret situation - Dekomposition af problem i mindre, håndterbare dele
Datagenerering og -behandling	- Dataskabelse og data indsamling, klargøring af data til analyse - Dekomposition af data, dvs. logisk dataanalyse og -organisering
Modellering	- Abstraktion af træk som de vigtigste - Mønstergenkendelse/-skabelse ud fra disse træk - Modelskabelse
Algoritmedesign	- Opstilling af trin-for-trin-handlingssekvens
Automatisering	- Kodning af algoritmen til at udføres automatisk - Fejlsøgning og iterativ afprøvning
Generalisering	

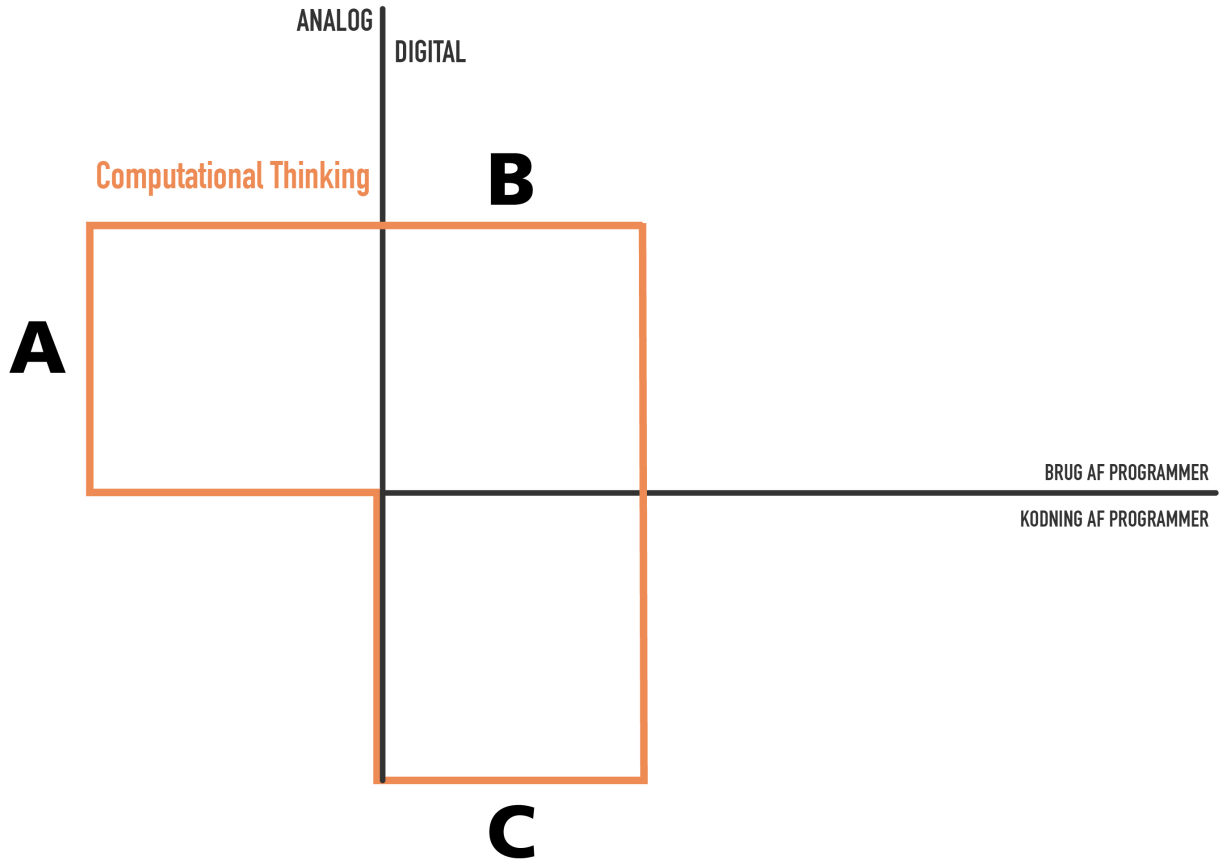


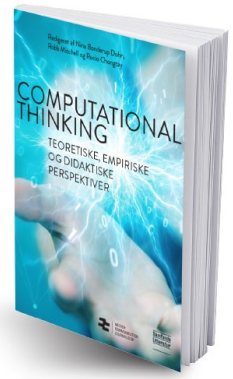
CT som sekventiel problemløsning – faser og kompetencer

Fase	Kompetencekarakteristik
Problemformulering	- Abstraktion af problem fra konkret situation - Dekomposition af problem i mindre, håndterbare dele
Datagenerering og -behandling	- Dataskabelse og data indsamling, klargøring af data til analyse - Dekomposition af data, dvs. logisk dataanalyse og -organisering
Modellering	- Abstraktion af træk som de vigtigste - Mønstergenkendelse/-skabelse ud fra disse træk - Modelskabelse
Algoritmedesign	- Opstilling af trin-for-trin-handlingssekvens
Automatisering	- Kodning af algoritmen til at udføres automatisk - Fejlsøgning og iterativ afprøvning
Generalisering	- Abstraktion af mønster for problemløsning - Generalisering af problemløsningsmønstret til andre områder

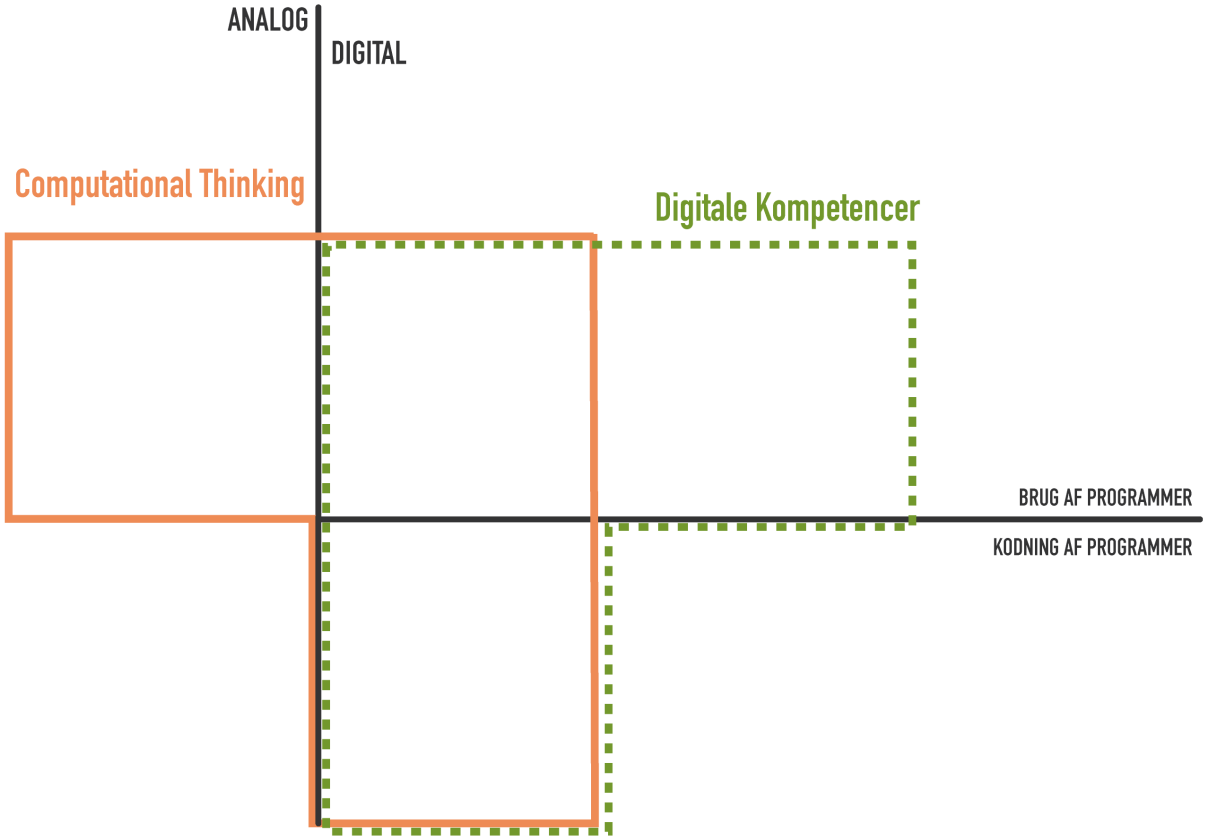


Model for CT-former

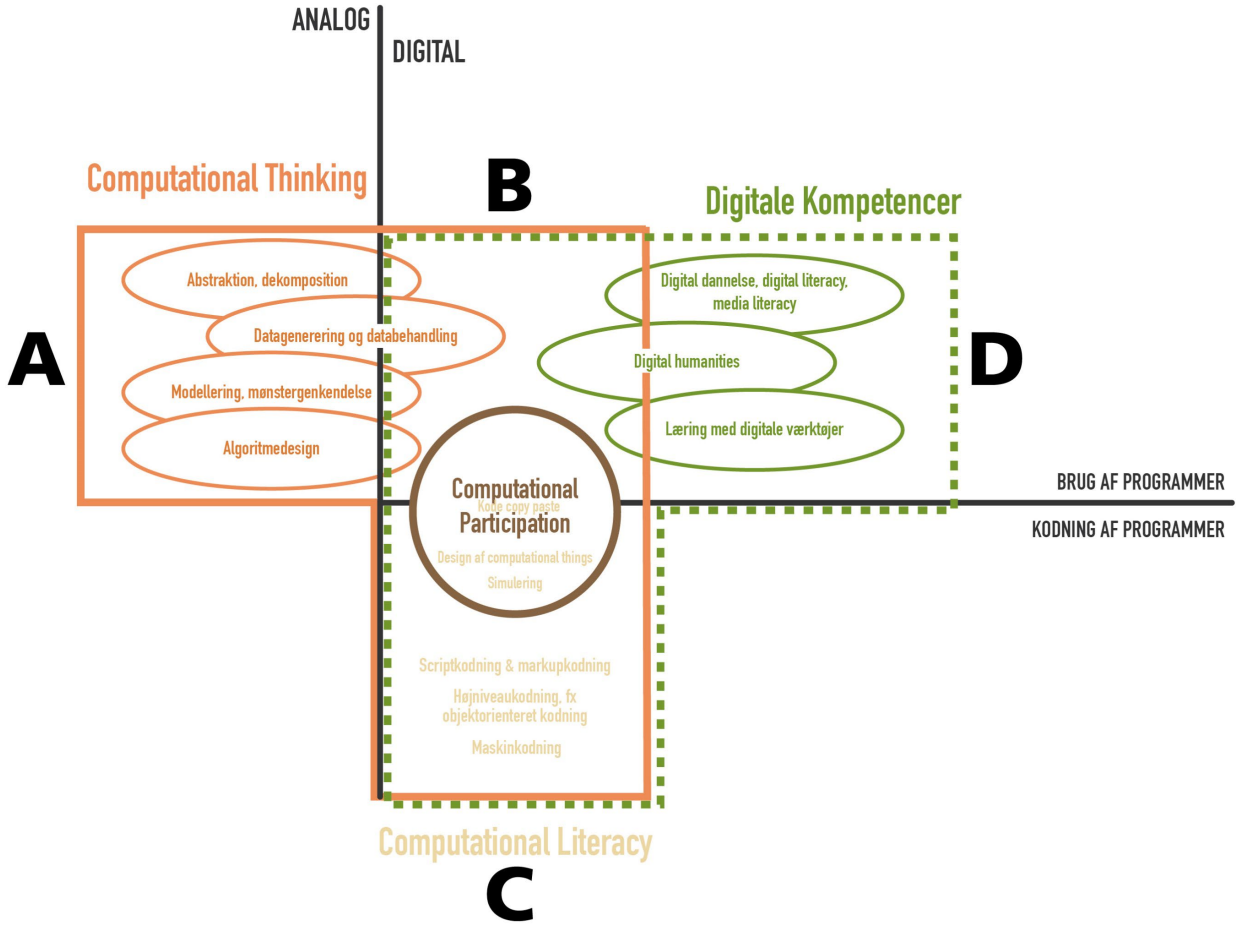




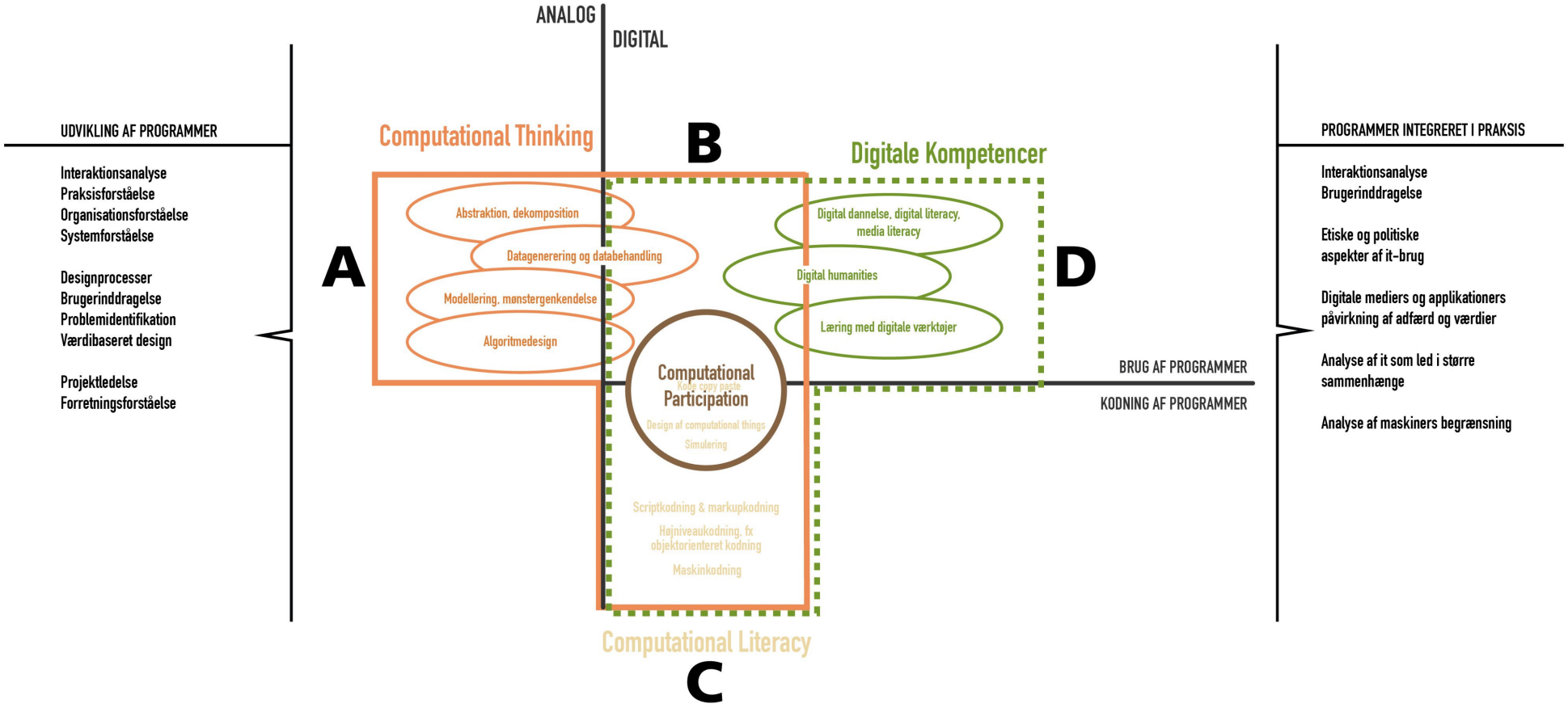
Model for CT-former – i relation til ”digitale kompetencer”



Model for CT-former – i relation til ”digitale kompetencer”



Model for CT-former – i den brede forståelse af CT



Chapter 5

Computational Thinking as a problem-solving method across disciplines

In recent years, much research has been done into the benefits of using Computational Thinking (CT) as part of problem-solving methods. At the same time, CT has been recognized as an essential 21st century skill.

However, there are also constant discussions about the following two questions:

1. What CT is?
2. Can CT be used as a generic problem-solving method in areas other than Computer Science?

Chapter 5

Computational Thinking as a problem-solving method across disciplines

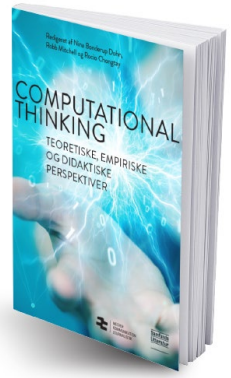
This chapter focuses on:

- A brief historical timeline of the mutual influence between the development of problem-solving computation procedures and the development of machines that automate these procedures.
- Problem-solving context
- A metareview ("Umbrella review") to analyze the prospects of Computational Thinking as a possible generic problem-solving method

Problem-solving context

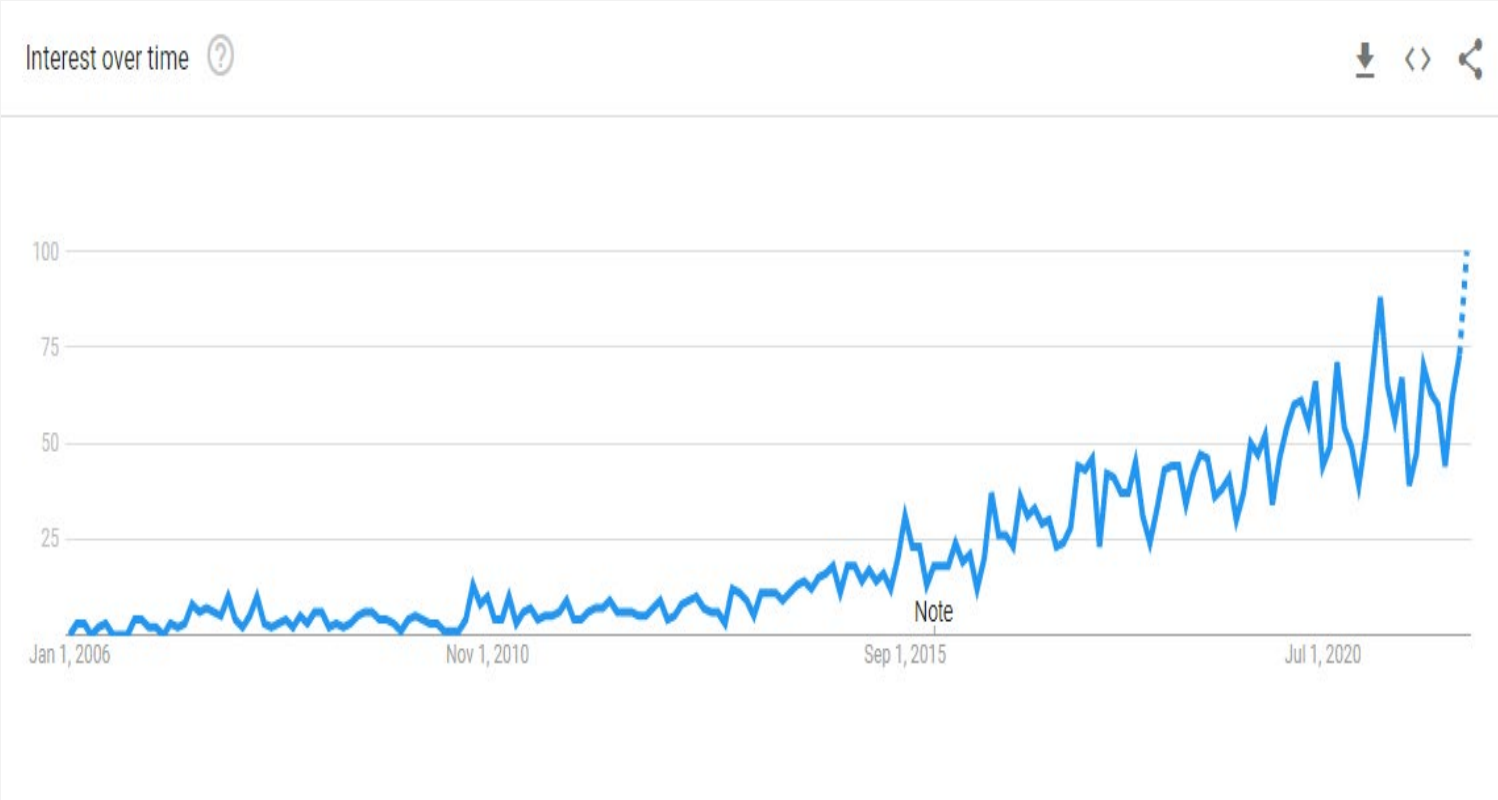
Although the theorists have tried to develop generic definitions of problem solving, the proposed definitions may vary depending on the domain for which they are designed.
Mayer (2012) has formulated a broad definition of problem solving, which can be applied to a wide range of problems and areas – from solving personal dilemmas to laying puzzles or solving mathematical problems, etc.

"Problem solving refers to cognitive processing directed at achieving a goal when the problem solver does not initially know a solution method. A problem exists when someone has a goal but doesn't know how to achieve it."



Computational Thinking

Google search trends Worldwide 2006-2022



The interest in Computational Thinking and number of publications has been increasing since Wing's 2006 publication.

Computational Thinking

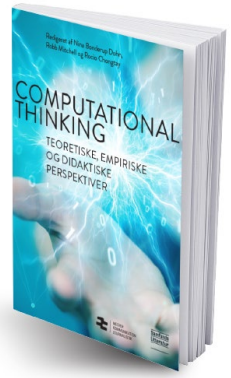
The umbrella review focuses on systematic reviews that have been published during 2013 and 2020 to address the challenge of analyzing large amounts of relevant CT research.

Chosen databases (giving most relevant results):

Web of Science (WoS),
ACM Digital Library,
Scopus and
Academic Search Premier Ebsco

Keywords (as subject and in title) such as:

Computational AND Thinking AND review;
Computational AND Thinking AND problem-solving AND review;
Computational AND Thinking AND problem AND review.



Computational Thinking

Results: 24 CT sytematic review articles that can be grouped into the following main categories:

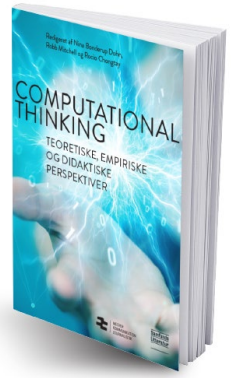
- Didactic designs where teaching and learning had CT as the *domain*. With two subcategories:
  - Teaching and learning CT as part of a specific subject or subject (mathematics, programming, robotics/educational robots, etc.)
  - Teaching and learning CT as a general competence across subjects, e.g. in the so-called STEM subjects (Science, Technology, Engineering, Mathematics)
- Didactic designs where CT was used to teach and learn specific subjects and topics (learning with CT as a *tool)*
- Other CT reviews, frameworks, trends, skills, etc.

Computational Thinking

Target Educational level

- 12 elementary and secondary education (K-12)
- 2 primary school up to 9th grade (K-9)
- 1 at 6th – 12th grade (secondary school)
- 1 Higher education
- 3 wide spectrum from elementary to higher education
- 3 unspecified level

The level of education may be relevant as it may reflect the complexity of the problems where CT has been used.



Computational Thinking

CT uses across disciplines

According to the reviews examined, the most common subjects to learn CT and subjects in which CT is used to learn those (some reviews mention more than one subject):

- programming (16 reviews),
- mathematics (6 reviews)and
- robotics/educational robots (6 reviews)

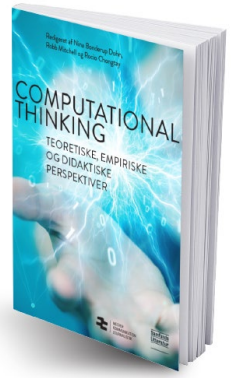
Computational Thinking

CT used in other disciplines

Many of the reviews mention that CT can be used across disciplines outside the STEM and computer science area. But they provide very few details about this.

Some of the other subject areas mentioned is the following (the number of reviews in brackets, and categories as stated in the reviews):

Arts (11), Biology (8), Physics (5), Humanities (5), Storytelling (5), Social Studies (5), Journalism (4), Music (4), Animation (4), Ecology (2), Kinematics (3), Literature (1), English Language (1), Geography (1), Archeology (1), Astronomy (1), Linguistics (1), Pharmacy (1), Medicine (1), Economics (1) and Meteorology (1).



Computational Thinking

In conclusion:

- Computational thinking (CT) is discussed as a problem-solving method across disciplines.
- The understanding of problem solving has historically evolved in line with the mutual development of computational procedures and of devices for automating such procedures.
- There is a link and mutual influence between understanding problem solving, computational procedures and IT automation.
- A systematic metareview of the reviews from 2013-2020 shows that there are:
  - remarkably few studies of CT applied to real problems outside of school
  - under-representation of areas other than computer science and science.

The hope is that this chapter stimulates interest among researchers across disciplines and other stakeholders to begin/continue a thorough discussion of the opportunities CT offers, both now and in the future – and perhaps more importantly, to explore new approaches to address the challenges identified here, as well as to publish concrete practical CT applications in relation to real-world problems.

Didaktisk fokuspunkt: Læring *af* og læring *med* CT

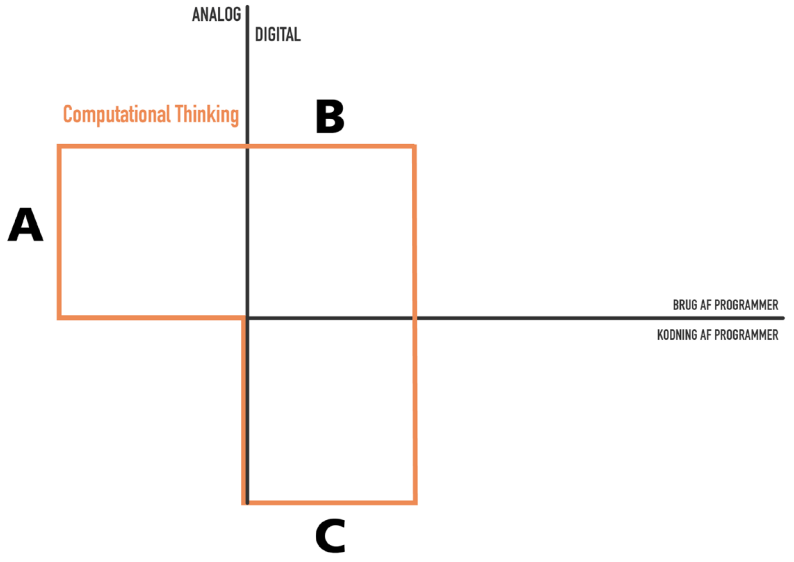
Skelnen mellem

- CT som indholdsdomæne
  - Svar på det fagdidaktiske *hvad*, dvs. fagdidaktisk begrundet indhold
- CT som redskab til læring af andre indholdsdomæner
  - Svar på det fagdidaktiske *hvordan*, dvs. fagdidaktisk begrundet metode
  - Instrumentalistisk begrundelse for CT

Didaktisk fokuspunkt: Læring *af* og læring *med* CT

CT som indholdsdomæne, fx

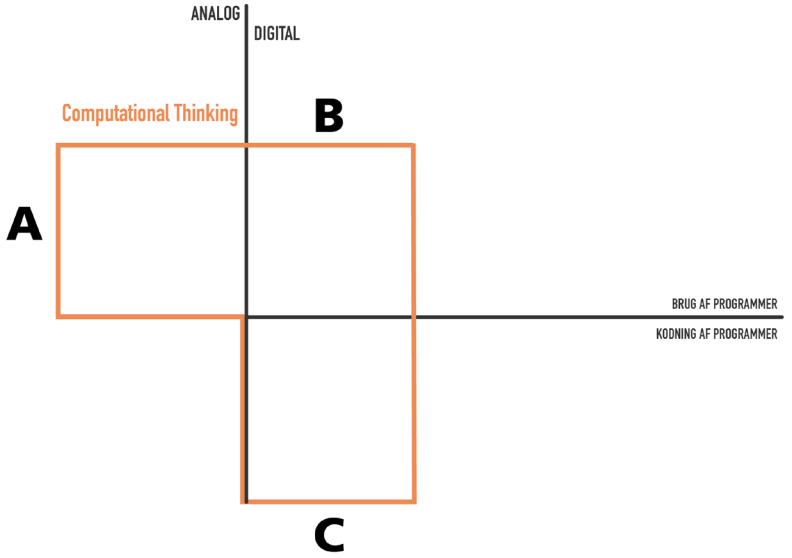
- Sekventiel problemløsning som metode
  - Forløb på EUD med modellering af kundestrømme og kødannelse i supermarked
- Datakvalitet og dataklargøring
  - Teknisk behandling og kritisk stillingtagen
- Programmering
- Design af it-artefakter



Didaktisk fokuspunkt: Læring *af* og læring *med* CT

CT som redskab til læring af andre indholdsdomæner, fx

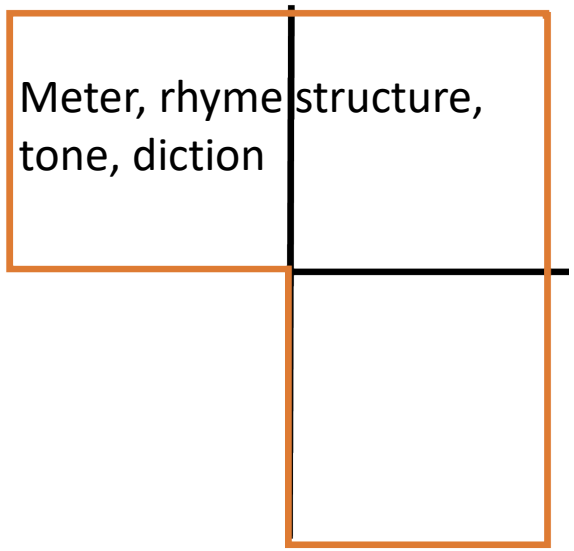
- Sekventiel problemløsning som en metode til løse problem i andet fag
  - Forløb på EUD: udvikling af systematisk fejlfinding i el-styring
- Algoritmisk beskrivelse som synliggørelse af sammenhænge og valg
  - Økonomi, flygtninges livssituation



Didaktisk fokuspunkt: Helhedsforståelse versus algoritmisk tænkning

Et eksempel til illustration, **CT-decomposition inden for litteratur:**

- "Break down the analysis of a poem into analysis of meter, rhyme, imagery, structure, tone, diction, and meaning." (tidligere på Googles online CTkursus)



Den hermeneutiske cirkel

- CT bruges til analyse af delene
- Helhedsforståelsen giver mening til delene
  - Udpeger CTs 'plads'
  - Men informeres også af CT-analysen

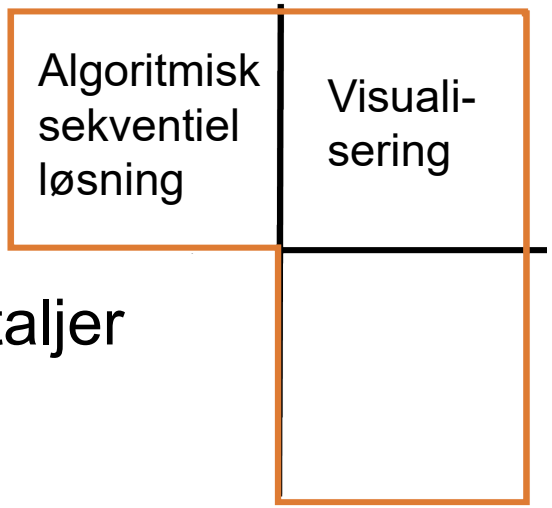
Didaktisk fokuspunkt: Helhedsforståelse versus algoritmisk tænkning

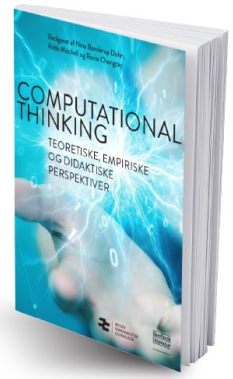
Ideel relation: Den hermeneutiske cirkel

- CT bruges til analyse af delene
- Helhedsforståelsen giver mening til delene
  - Udpeger CTs 'plads'
  - Men informeres også af CT-analysen

Erfaringer:

- Trin-for-trin-opdeling kan være en
  - hjælp for elever, der har svært ved at få overblik
  - udfordring for elever, der har blik for kompleksiteten
- Kompleksitet reduceres tit til envejsrelation
- Pointe i analysen kan blive væk pga. fokus på detaljer

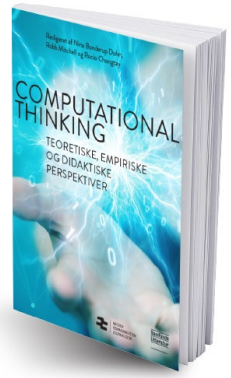




Skabelon til didaktisk design af CT-forløb

Hvilket fag og uddannelsesniveau?
Hvad er fokus for CT-undervisning: **læring *af* eller *med* CT?**
Hvad er **formål** med CT i undervisning? (didaktisk hvorfor – eller hvordan)
Hvilke **læringsmål** adresserer CT-forløbet? (didaktisk hvorfor)
Hvilket **indhold** skal der arbejdes med? (didaktisk hvad)

Hvilke **rammer** har forløbet? (didaktisk hvor og hvornår)
Hvilke **aktiviteter** skal eleverne arbejde med? (didaktisk hvordan)
Hvordan skal aktiviteterne **organiseres** (socialt, i rum, i tid) (didaktisk 'med hvem')
Hvilken **teknologi** skal anvendes (om nogen)? (didaktisk hvormed)
Hvilke **øvrige læringsressourcer** skal anvendes (didaktisk hvormed)



Tak for opmærksomheden!