Kunnskapssenter for utdanning
Universitetet i Stavanger

# Utvikling av algoritmisk tenking gjennom programmering med Scratch i grunnskolen

REFERANSE

Fagerlund, J., Häkkinen, P., Vesisenaho, M. & Viiri, J. (2020). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education*, 2021;29:12–28. DOI: https://doi.org/10.1002/CAE.22255

**Det finst fleire digitale ressursar og verktøy som lærarar bruker for å støtte opp om elevers læring og utvikling av algoritmisk tankegang. Eit digitalt verktøy som er mykje brukt er «Scratch», eit gratis tilgjengeleg, visuelt programmeringsspråk som kan brukast online eller lastast ned. Det systematiske kunnskapsoversynet som blir presentert i dette forskingsnotatet er ei oppsummering av internasjonal forsking på korleis Scratch kan lære elevar algoritmisk tenking. Resultata viser at Scratch både fremmer og gjer synleg elevane si evne til algoritmisk tenking. I tillegg gir verktøyet lærarane moglegheit til å evaluera korleis elevane løyser problem ved hjelp av algoritmisk tenking.**

**Læreplan for kunnskapsløftet 2020 (LK20) har integrert algoritmisk tenking matematikkfaget, både i kjerneelement og i kompetansemål. I kjerneelementa er algoritmisk tenking knytt til utforsking og problemløysing. I kompetansemåla er det konkrete programmeringsmål. Til dømes skal elevar i andre klasse kunne lage og følgje reglar og trinnvise instruksjonar i leik og spel og i femte klasse skal dei kunne lage og programmere algoritmar med bruk av variablar, vilkår og løkker.**

**Men korleis kan lærarar vurdere algoritmisk tenking som kompetansemål? Ifølge ein internasjonal rapport om integrering av algoritmisk tenking i europeiske land (Bocconi et al., 2022[1]), er vurdering av algoritmisk tenking i Noreg mest basert på lærarens observasjonar når elevar jobbar prosjektbasert i grupper. Læring ved å gjøre feil blir vektlagt i vurderingsprosess. Kunnskapsoversikten som blir presentert i dette forskingsnotatet handlar om vurdering av algoritmisk tenking i Scratch-prosjekter. Vurdering kan være basert på observasjonar og intervju, men også for eksempel på analyse av koder skrevet av elever. Kunnskapsoversikten introduserer konkrete verktøy for dette.**

[1] https://op.europa.eu/en/publication-detail/-/publication/bbf875ec-a5a2-11ec-83e1-01aa75ed71a1/language-en

## Bakgrunn

Det er ikkje einigheit om korleis algoritmisk tenking skal definerast (på engelsk: computational thinking). Nokre definisjonar kan vere meir programmeringsorienterte, medan andre tar med meir generelle ferdigheiter. Forfattarane av dette kunnskapsoversynet viser til 14 kjerneelement som inngår i algoritmisk tenking.[2] Døme på nokre av kjerneelementa er:

> Abstraksjon: Å kunne redusere unødvendige detaljar og fokusere på element som betyr noko
>
> Algoritmar: Å kunne skape instruksjonar ved å sekvensere, selektere, repetere.
>
> Samarbeid: Å kunne fordele oppgåver, roller, bygge på kvarandre sine prosjekt
>
> Kreativitet: Å kunne «tenke utanfor boksen»
>
> Effektivitet: Å kunne designe noko som er enkelt i bruk, utan unødvendige steg
>
> Logikk: Å kunne analysere situasjonar, sjekke fakta, verifisere hypotesar, ta avgjerder og konkludere
>
> Mønster og generaliseringar: Å kunne identifisere gjentakande mønster basert på likskapar og forskjellar, vurdere overføringsverdi

Korleis kan digitale læringsverktøy hjelpe lærarar i deira arbeid med å hjelpe elevar å auke slik kompetanse? I dette tilfellet tek forskarane for seg det digitale læringsverktøyet Scratch. Scratch er eit gratis, nettbasert og blokkbasert programmeringsverktøy som gjør det mogeleg for barn heilt frå barnehagealder av å skape spel, animasjonar og digitale forteljingar basert på deira eigne erfaringar og interesser. Innhaldskomponentar i Scratch er studert tidlegare, men det er få som har vurdert innhaldet i forhold til kjerneelement i algoritmisk tenking.

## Føremål

Forskarane har gjennomført eit systematisk kunnskapsoversyn[3] som undersøker korleis bruken av Scratch i grunnskolen kan hjelpe elevane å utvikle (a) grunnleggande algoritmisk forståing, (b) forståing av begrep knytt til programmering, og (c) forståing for korleis ein kan anvende desse kunnskapane og ferdigheitene.

Forskingsspørsmåla er:

1) Kva for nokre innhaldskomponentar og aktivitetar i Scratch er blitt vurdert?

[2]Utdanningsdirektoratet viser til ei forenkla oversikt basert på Barefoot Computing UK, som skildrar algoritmisk tenking ved hjelp av seks kjerneelement og fem arbeidsmåtar. Sjå: https://www.udir.no/kvalitet-og-kompetanse/profesjonsfaglig-digital-kompetanse/algoritmisk-tenkning/

[3] **Systematisk kunnskapsoppsummering/-oversyn:** Ei systematisk kunnskapsoppsummering tar utgangspunkt i et klart definert forskingsspørsmål eller mål og bruker systematiske og eksplisitte metodar for å identifisere, velje ut og kritisk vurdere relevant forsking, samt for å samle inn og analysere data frå studiane som er inkluderte i oppsummeringa. Kunnskapsoppsummeringa resulterer ofte i eit systematisk kunnskapsoversyn, som regel ein artikkel eller ein rapport, som gir eit oversyn over temaet eller svar på forskingsspørsmålet.

2) Korleis har innhaldskomponentar og aktivitetar i Scratch blitt vurdert?
3) Korleis kontekstualiserar ulike innhaldskomponentar og aktivitetar i Scratch kjerneelement og praksisar i algoritmisk tenking?

## Inkluderte studiar

For å bli inkludert i den systematiske kunnskapsoppsummeringa måtte studien (a) vere publisert mellom 2007 og november 2019 (b) på engelsk i (c) eit fagfellevurdert tidsskrift, (d) undersøke bruk av Scratch eller veldig like program i (e) grunnskole (Kindergarten – niende klasse). Artikkelen nemnar ikkje nokon geografisk avgrensing. Til saman 30 studiar blei til slutt inkluderte i den systematiske kunnskapsoppsummeringa.

## Resultat

### Kva er vurdert og korleis?

Svar på spørsmål om kva for nokre innhaldskomponentar og aktivitetar som var blitt vurdert i tidlegare forsking, syner at det er «code constructs» eller programmeringsstrukturar som er mest vurdert (20 studiar). Programmeringsstrukturar er logikken bak programmet, korleis programmet er oppbygd, f.eks. rekkefølgja av blokkene, kva for nokre programmeringsprinsipp (løkker, vilkår, funksjoner) er brukt osb. 10 studiar var opptekne av kodingsmønster. Kodingsmønster vil seie korleis programmeringsstrukturane er blitt kombinert i programmet for eksempel for å animere bevegelse. I seks studiar blei «annan programmeringsinnhald» vurdert.

### Programmeringsstrukturar i Scratch og vurdering av kjerneelement i algoritmisk tenking

Fleire studiar som var opptekne av programmeringsstrukturar brukte digitale vurderingsverktøy for å vurdere korleis elevane arbeidde og lærte. Døme på slike digitale verktøy er «Dr. Scratch»[4], «CodeMaster»[5], og «Ninja Code Village».[6] Vurderingsverktøya målte blant anna kva for eit nivå elevane brukte blokkene på: Elevar som bruker logiske operasjonar i staden for «if»-blokkene demonstrerer t.d. høgare kompetansenivå i kjerneelementet logikk i algoritmisk tenking. Elevar som kan bruke løkker i staden for sekvensiell bruk av repeterande blokker demonstrerer kompetanse i algoritmisk tenking.

I andre studiar blei elevar spurt om å svare på oppgåver om det dei hadde lært (testar). Dei blei også bedne om å beskrive kva dei hadde gjort og å vurdere sjølv kor vanskeleg det var å lære. I andre tilfelle vurderte forskarane det arbeidet som elevane hadde gjort manuelt, utan digitale hjelpemiddel.

### Kodingsmønstera i Scratch og vurdering av kjerneelement i algoritmisk tenking

Implementering av repeterende kodemønster i Scratch kontekstualiserar kjerneelementa mønster og generaliseringar samt dekomposisjon (å kunne bryte problemet ned i mindre delar) i algoritmisk tenking .

For å vurdere elevar si læring om kodingsmønster og om andre aspekt ved programmering, blei det også brukt ulike tilnærmingar. Elevar blei bedne om å vurdere vanskegrad, om å teikne resultata av ei «script» (ein instruksjon) og dermed å visualisere ein algoritme. I andre tilfelle blei elevane bedne

[4] http://www.drscratch.org/
[5] https://www.researchgate.net/publication/324762596_CodeMaster_-_Automatic_Assessment_and_Grading_of_App_Inventor_and_Snap_Programs
[6] http://ik1-325-22639.vs.sakura.ne.jp/ncv4s/about/

om å utvikle taksonomiar som forskarane brukte til å vurdere arbeidet deira. Forskarane bak det systematiske kunnskapsoversynet framhevar også ei vurderingsform som er utvikla av Seiter & Foreman (2013).[7] Seiter og Foreman utvikla eit vurderingsverktøy som dei kallar «Progression for Early Computational Thinking» (PECT). Denne gir retning for manuelle vurderingar av algoritmisk tenking.

### Anna programmeringinnhald i Scratch og vurdering av kjerneelement i algoritmisk tenking

I nokre studiar blei det vurdert kor mange prosjekt elevane hadde laga og remiksa, mens andre studiar kategoriserte prosjektenes sjangrar. Elevar som kan bruke ulike typar løysingar i deira design og som kan remikse design i ei rekke prosjekt for spesifikke formål demonstrerer kompetanse i automasjon, som er eit kjerneelement i algoritmisk tenking.

### Aktivitetar med Scratch i vurdering av kjerneelement i algoritmisk tenking

Berre ein av studiane fokuserte direkte på aktivitetar som kunne demonstrere algoritmisk tenking. Slike aktivitetar var t.d. prøving og feiling, gjenbruk og remiksing samt abstraksjon.

Dei andre studiane som handla om aktivitetar, fokuserte på planleggingsfasar i elevane sine prosjekt og på interaksjonane i klasserommet for å undersøkje om elevane deltok aktivt, kor mykje hjelp elevane hadde behov for, samt elevane sine programmeringsvanar.

Mange studiar påpeikte at aktivitetane kan evaluerast ved hjelp av observasjon, intervju eller sjølvevaluering i tillegg til ei ferdigheitsbeskriving eller ei beskriving av prestasjonsnivået til elevane.

### Relasjonar mellom Scratch og algoritmisk tenking

Elevar som bruker Scratch arbeider med utvikling av algoritmisk tenking. Vurderingane som blei gjort i dei studiane som er inkludert i kunnskapsoversynet koplar innhald og aktivitetar i Scratch til kjerneelement i algoritmisk tenking som t.d: abstraksjon, logisk tenking, mønster og generaliseringar, å kunne dekomponere problem, algoritmar, og meir.

Når det gjeld innhaldet i Scratch-prosjekt, kan evalueringar av algoritmisk tenking hos elevane vere basert på korleis dei bruker kodespråket til å utføre spesielle handlingar, korleis dei bruker kodemønster og korleis dei bruker anna programmeringsinnhald som dei har implementert i Scratch-prosjekt. Å studere om, kor ofte og kor korrekt elevane utfører særskilte handlingar, og om dei fullfører oppgåva, kan gje eit bilete av elevane si evne til algoritmisk tenking. For å få eit så nøyaktig bilete av elevane sine evner til algoritmisk tenking som mogeleg, er det hensiktsmessig å studere dei individuelle programmeringsoperasjonane som elevane utfører framfor kodeprosjekt dei gjer.

Programmeringsaktivitetar i Scratch kan fremme evner til algoritmisk tenking som ikkje er synlege i innhaldet som elevane produserer, men er ein del av programmeringsprosessen. Slike prosessar involverer til dømes å beskrive problemet, abstrahere problemet, bryte ned problemet, designe algoritmen og prøve ut løysinga. Heilskaplege vurderingssystemer er ikkje utvikla ennå for å fange opp alle dei spesifikke og generelle kjerneelementa i algoritmisk tenking.

Forfattarane er opptekne av formativ vurdering og definerer dette slik: prosessane involverer (1) klåre læringsmål og kriterium for måloppnåing, (2) informasjon om elevane si noverande forståing

[7] L. Seiter and B. Foreman, Modeling the learning progressions of computational thinking of primary grade students. The 9th annual international ACM conference on International computing education research. New York, NY: ACM. 2013, pp. 59–66.

og (3) tilbakemeldingar som gjer det mogeleg for elevane å bevege seg mot måla. Programmering, til dømes med bruk av Scratch, er ei synleggjering av elevane si evne til algoritmisk tenking. Sjølve programmeringsprosjektet gir ikkje lærarar direkte tilgang til elevar si tenking, men det kan likevel vere mogleg å vurdere elevane si algoritmiske tenking. Heilskaplege evalueringar av elevane si evne til algoritmisk tenking bør ta omsyn til kompleksiteten i problemløysing.

### Utfordringar

Kunnskapsoversynet peiker på fleire område som er utfordrande. Ikkje alle kjerneelementa i algoritmisk tenking er kontekstualisert i Scratch. Dette gjeld t.d. «effektivisering» eller det å kunne unngå å ha med unødige steg, å lage programmer som er enkle å bruke. Det å finne og bruke data frå ulike kjelder kan også vere vanskeleg i Scratch fordi det er primært eit media design-verktøy, ikkje eit generelt programmeringsverktøy. Det er også viktig at elevar forstår at computarar, operasjonssystem, applikasjonar og programmeringsspråk er abstraksjonar, og dei må kunne vurdere og identifisere reelle bruksområde for algoritmisk tenking. Slik kompetanse blir nok utvikla best ved hjelp av andre digitale læringsverktøy enn Scratch.

Vurderingsformene som er anvendt er også til dels mangelfulle fordi dei ikkje er komplekse nok til å seie noko om dei intrikate vurderingane som skjer, og kjerneelement som «å tenke utanfor boksen» er forholdsvis vage. Kva vil det seie i ulike situasjonar?

## Implikasjonar

Scratch er eit døme på eit digitalt læringsverktøy som blir brukt av lærarar i mange land og som kan fremme algoritmisk tenking og hjelpe elevar å auke sin kompetanse. Kunnskapsoversynet gir oversikt over korleis innhald i Scratch heng saman med kjerneelement i algoritmisk tenking, men også at det er fleire område av algoritmisk tenking som ikkje er så framtredande i Scratch. Lærarar må ha innsikt i ulike digitale læringsverktøy og må kunne vurdere kva for digital læringsverktøy dei skal bruke, kva dei skal bruke det til, til kva tid og for kven.