Didaktisk guideline til programmeringsundervisning for novicer: Den taktile Notional Machine i praksis

Anvendelse af RT-Metoden (Reality Testing) med RT-Vinduet og RAM-Papiret i Novice Programmering

Abstract

This practice guide serves as a concrete, actionable roadmap for implementing the Reality Testing (RT) methodology in educational settings, building upon the foundational theories established in Working Paper 1 (Hansen, 2026). Addressing the critical need for cognitive load reduction among novice programmers, the guide operationalizes the Notional Machine through tactile and visual tools: the RT-Window for physical inhibition and line-by-line tracing, and RAM-paper structured as an isolated sandbox for stack, heap, and dynamic references. By shifting the focus exclusively to program comprehension (tracing) before code generation, the methodology successfully bridges the gap between template dependency and independent mental model calibration. Empirical test results demonstrate a dramatic cognitive progression, with intervention cohorts achieving an average score of 96.7% and significantly reduced completion times, establishing a robust defense against student dropout.

1. Introduktion og formål

Hvor det første working paper (Hansen, 2026) beskriver det teoretiske og aktionsforskningsbaserede fundament samt de indledende digitale interventioner (herunder visualiseringsværktøjet *CodeToModel* (CtM), *CodeExercises (CE)* og det numeriske *QuizSystem (QS)*), fungerer denne guide som en konkret, praksisnær køreplan til den fysiske og taktile videreudvikling af **RT-metoden** i klasselokalerne. Målet er at fjerne den kognitive overbelastning, som nybegyndere oplever, ved at omsætte indsigterne om dynamiske pegepinde og didaktiske "hastigheds-vejbump" til et håndgribeligt, analogt redskabssæt, der erstatter usikker "gæt-og-test"-adfærd.

1.1 Fokus på programforståelse (Comprehension Before Generation)

Det er afgørende at skelne mellem to grundlæggende dimensioner i programmering:

- **Programforståelse (Code Comprehension / Tracing):** Evnen til at læse, spore og simulere kode linje for linje for at forstå dens dynamiske tilstand.

- **Programgenerering (Code Generation / Problem-solving):** Evnen til selv at skrive kode og løse komplekse problemer fra bunden.

Forskning i datalogididaktik (herunder Sorva, 2013 og Lister et al., 2004) fastslår, at mange novicer fejler, fordi de mangler de helt basale forudsætningsfærdigheder i at læse og spore kode, længe før de overhovedet kan forventes at løse problemer. Den taktile træning med RT-vinduet og RAM-papiret er derfor eksklusivt designet til at mestre **programforståelse** (dekonstruktion og aktiv Reality Testing på Level 3) – og fungerer som det nødvendige fundament, der fjerner den kognitive barriere, så de studerende opbygger den rette mentale model, før de kaster sig over selvstændig programgenerering (rekonfiguration på Level 4).

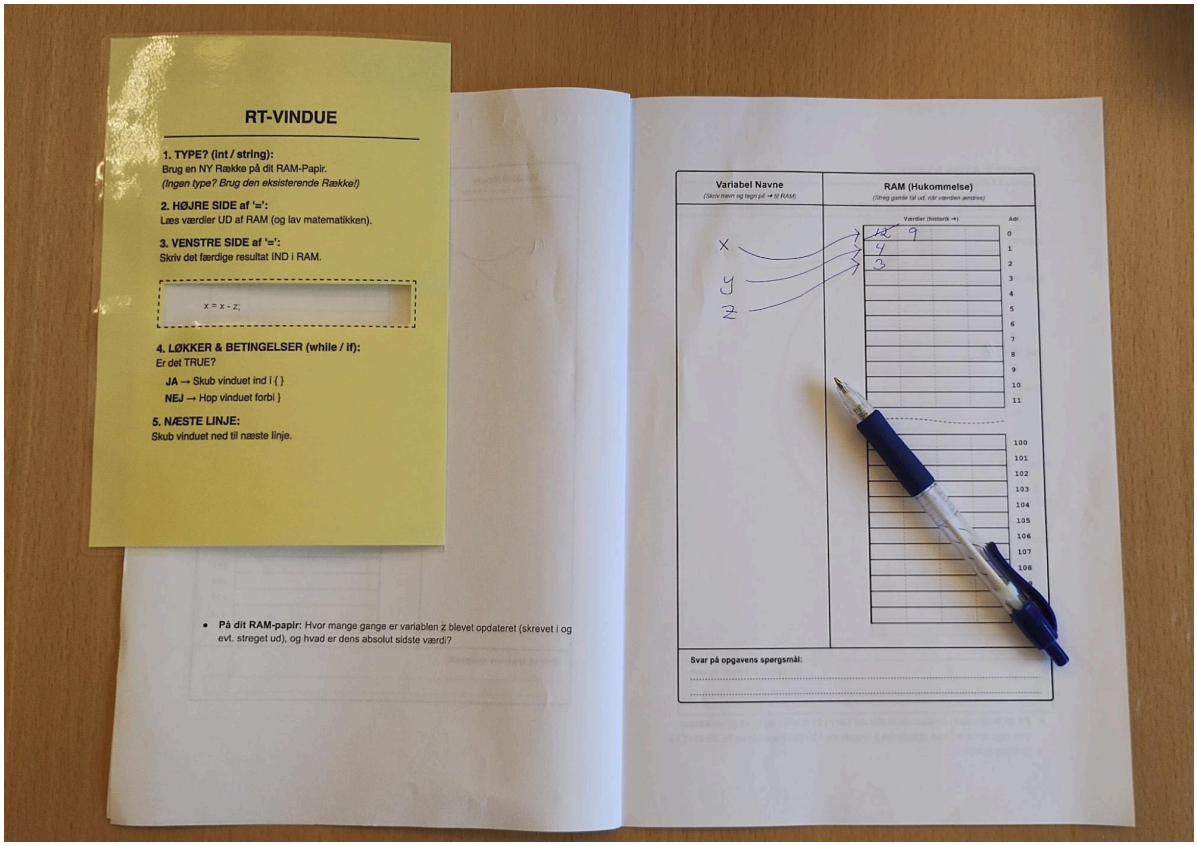
2. De centrale elementer i den taktile Notional Machine

RT-metoden hviler på det fysiske redskabssæt, der tilsammen udgør en eksternaliseret computer (*Notional Machine*):

2.1 Det fysiske filter og den mekaniske algoritmeproces (RT-Vinduet og de 5 faste trin)

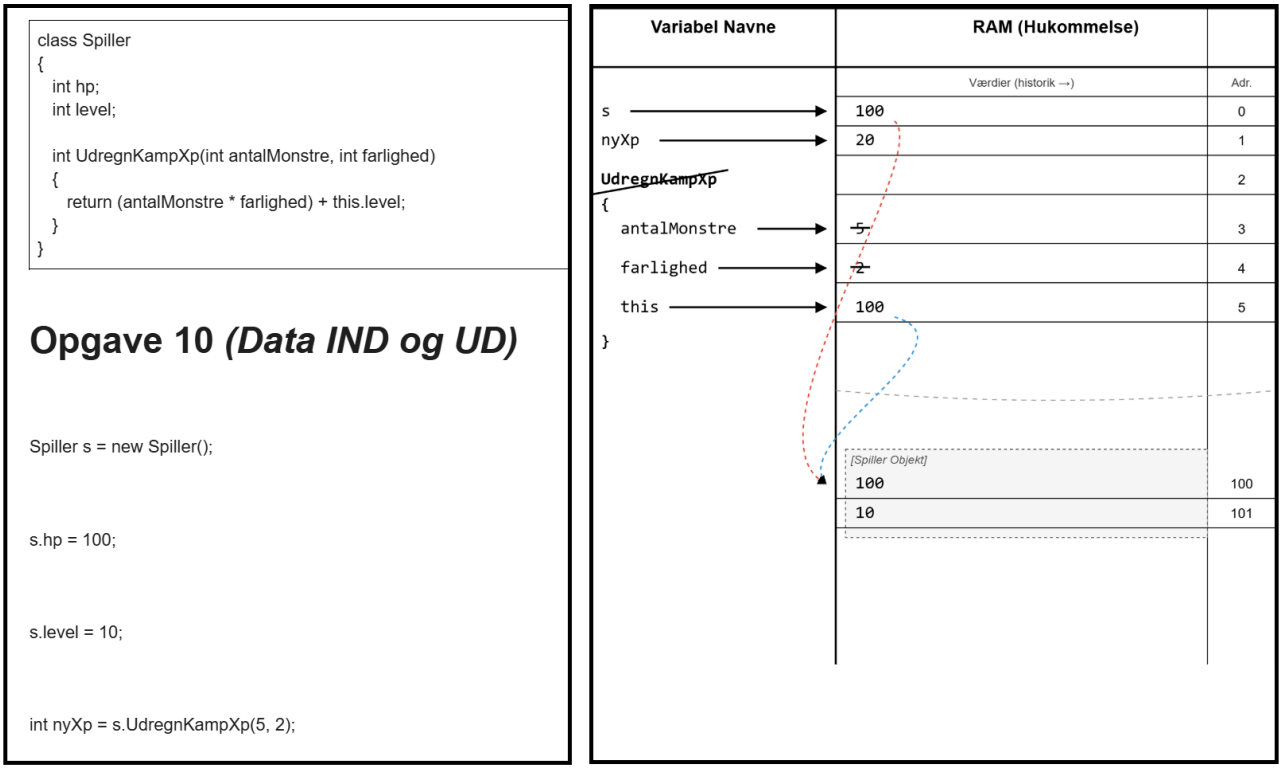
- **RT-Vinduet:** Det gule papvindue med den smalle udskæring blokerer alt overflødigt information og tvinger den studerende til at fokusere på **én enkelt linje kode ad gangen**. Dette forhindrer ukontrolleret skanning op og ned i koden og gennemtvinger en isoleret virkelighedsprøvning (*Reality Testing*).
- **De 5 faste trin:** Når vinduet er placeret på en linje, følger den studerende en fast procedure:
    1. ***TYPE?:*** Opret ny række eller brug eksisterende.
    2. ***HØJRE SIDE AF "=":*** Læs værdier fra RAM først og udfør matematikken.
    3. ***VENSTRE SIDE AF "=":*** Skriv resultatet i RAM og streg den gamle værdi over.
    4. ***LØKKER & BETINGELSER:*** Tjek om betingelsen er sand (`true` / `false`) – gå ind i `{}` eller spring over.
    5. ***NÆSTE LINJE:*** Flyt vinduet ned til næste linje.

2.2 RAM-papiret som eksternaliseret hukommelsesmodel (Stack & Heap)



Figur 1: Samlet oversigt over RAM-papir, Stack & Heap samt RT-vinduet som integreret redskab, der viser faste adresser, dynamiske pegepinde (pile) og tidslinjens bevarelse.

- **Faste adresser:** Hver opgave har sin egen isolerede "sandbox" med faste adresser (f.eks. 0, 1, 2 på Stack og 100, 101 på Heap).
- **Dynamiske pegepinde:** Variabelnavnene placeres *ikke* inde i hukommelsestabellen, men står til venstre og peger via en pil ind på en række med rå data og historik. Dette er et helt centralt didaktisk greb: Novicen har en tendens til at betragte en variabel som en "magisk kasse", der *er* værdien i sig selv (at "A er 5"). Men det er en sandhedskrise for den studerendes mentale model; A *er* ikke 5 – A er blot et navn, der peger på en fysisk adresse i RAM, hvor værdien befinder sig. Ved at adskille deklaration fra værditildeling tvinges den studerende til at opbygge den nødvendige, korrekte mentale model frem for blot at læse koden som en overfladisk fortælling.
- **Den tidsmæssige dimension:** Når krøllede parenteser `{}` lukkes, streges variabelnavnet over – det "dør" fysisk for øjnene af den studerende, hvilket gør lokal scope og call stack visuelt gennemskuelig.



Figur 2: Visuel trin-for-trin gennemgang af almindelige variabler, objekter på heapen samt metoder med parameter.

3. De fire gyldne principper for underviseren

For at sikre, at metoden lykkes i praksis, arbejdes der ud fra fire faste pædagogiske principper:

- **Fysisk inhibering:** Fjern visuel støj og tving fokus på én linje.
- **Højre side først:** Beregn altid værdien fra RAM, før du opdaterer.
- **Ét RAM-papir per opgave:** Hold den ydre hukommelsesmodel ren og isoleret.
- **Reversibilitet:** Bevæg dig konsekvent frem og tilbage mellem koden og RAM-papiret, så den interne mentale model forankres korrekt.

Det deterministiske øjeblik – opgøret med koden som "magi": Svage studerende lider ofte af en ubevidst forestilling om, at programmering er et uigennemtrængeligt, kaotisk spind, hvor hele koden lever sit eget liv på skærmen på samme tid. De kigger hvileløst op og ned ad linjerne og tror ikke på, at det kan være så simpelt. RT-vinduets fysiske blokering gennemtvinger derfor et *ontologisk chok*: Det lærer den studerende, at computeren er fundamentalt "dum" og nådesløst lineær. Der findes kun dét, der eksekveres på den ene synlige linje i vinduet, og dén tilstand, der fysisk er gemt i RAM i dette præcise mikrosekund.

Når den studerende overgiver sig til denne enkle sandhed, brydes den magiske tænkning, og den kognitive overbelastning forsvinder.

4. Konkrete testresultater og kvantitative effekter af metoden

For at validere effekten af det taktile RAM-papir, RT-vinduet og de didaktiske interventioner i praksis blev der foretaget en systematisk evaluering og komparativ analyse af studerendes præstationer:

- **Komparativ kohorte-analyse af variabler:** Som en supplerende kvantitativ kohorte-evaluering (ud over de 15 studerende i hovedundersøgelsen i Working Paper 1) blev der gennemført en test med to forskellige studenterkohorter på de samme diagnostiske quiz-spørgsmål om variabler:
    - **Kohorte 1 (Traditionel undervisning, n=22):** Viste stor spredning, hvor flere studerende scorede mellem 1 og 4 ud af 6 mulige point, og et tidsforbrug på 3 til 7 minutter, hvilket indikerede en skrøbelig forståelse.
    - **Kohorte 2 (Intervention med RT-metoden og RAM-papir, n=10):** Opnåede et markant gennemsnit på **96,7 %** (5,8 ud af 6 mulige point), hvor **90 % af deltagerne svarede fejlfrit**, og tidsforbruget faldt til blot 2 minutter og 25 sekunder.

Resultaterne af quiz-testene viser samlet set en markant forbedring i både nøjagtighed og tidsforbrug for interventionsgruppen, hvilket dokumenterer, at de fysiske redskaber effektivt understøtter den kognitive progression og sikrer en langt mere robust programforståelse.

5. Anvendelse i den faglige undervisning og praksis

Dette formidlingsprodukt og de tilhørende redskaber kan anvendes bredt af undervisere i programmering:

- **Fælles didaktisk sprog:** Det giver underviserne et fælles begrebsapparat og en fælles ramme til at identificere og håndtere de kognitive barrierer, hvor studerende sidder fast i overfladisk skabelon-tænkning (Zone 1) versus aktiv dekonstruktion (Zone 2).
- **Praksisnært værn mod AI-afhængighed:** Værktøjerne (RT-vinduet og RAM-papiret) kan implementeres direkte i undervisningsforløb som et fysisk og mentalt værn, der bryder "magisk tænkning" og hjælper studerende med at opbygge uafhængige, robuste mentale modeller frem for blindt at stole på ukritiske AI-værktøjer og code-completion.
- **Klar-til-brug opgavesæt:** Der er allerede udarbejdet en omfattende, køreklar samling af opgavesæt, der dækker alt grundlæggende programmering (herunder variabler, arrays, objekter, referencer og funktioner). Disse gennemtestede materialer kan andre undervisere og institutioner let få adgang til og implementere direkte i deres undervisning.

6. Erklæring om brug af AI

I forbindelse med udarbejdelsen af dette Working Paper er der anvendt kunstig intelligens (Gemini 3.1 Pro). AI er specifikt blevet brugt som en strukturel og sproglig sparringspartner til at omskrive forfatterens egne noter, observationer og empiri til sammenhængende akademisk prosa. Som forfatter bærer jeg det fulde ansvar for alt indhold i dokumentet, og al AI-genereret tekst og bearbejdning er blevet kritisk gennemlæst, vurderet og verificeret for faglig korrekthed i overensstemmelse med de gældende retningslinjer.

Referencer

Hansen, J. W. (2026). *Fra magisk tænkning til aktiv virkelighedsprøvning: Udvikling af en taksonomisk model og didaktisk intervention mod kognitive barrierer hos novicer i programmering* [Working Paper 1]. Erhvervsakademi Dania. https://www.eaviden.dk/wp-content/uploads/2026/11/Working-Paper-1_-Fra-magisk-taenkning-til-aktiv-virkelighedsproevning.pdf (tilgået 16.11.2026)

Lister, R., Adams, E. S., Fitzgerald, S., Fones, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J. E., Sanders, K., Seppälä, O., Simon, B., & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bulletin*, 36(4), 119-150.

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education*, 13(2), Article 8.