ASOCIAȚIA techsoup

Programarea trebuie evaluată ca proces

Analiză a propunerii de standarde naționale de evaluare la Informatică și TIC, clasele a V-a și a VI-a, cu accent pe unitățile care introduc programarea și exersează gândirea computațională

1. Punctul de plecare

Introducerea unor standarde naționale unitare de evaluare în învățământul public obligatoriu din România este necesară și așteptată de multă vreme. Standardele pot reduce diferențele de evaluare dintre clase și școli, pot face criteriile transparente, pot susține capacitatea elevilor de a-și evalua propria muncă și pot orienta feedbackul către progres. Sunt, totodată, o ocazie de a modera anumite limitări ale curriculumului de Informatică și TIC la gimnaziu, unde progresul în spirală de la un an la altul este slab articulat.

Arhitectura comună a standardelor este necesară pentru coerența sistemului, dar aplicarea ei trebuie să păstreze particularitățile competențelor evaluate în fiecare disciplină. În Informatică, procesul de gândire și de lucru contează cel puțin la fel de mult ca produsul final. Itemii standardizați scurți nu pot rămâne unica modalitate de a surprinde competența în programare; sunt necesare și sarcini practice în care elevul proiectează sau modifică un program, îl testează și justifică o decizie, pe baza unei rubrici comune.

Utilitatea standardelor depinde deci de măsura în care surprind nu doar produsul final sau utilizarea unei interfețe, ci procesul specific programării: înțelegere și predicție; proiectare; implementare; testare și depanare; explicare și îmbunătățire. Cadrele internaționale de referință converg asupra acestui punct: produsul final, singur, este o dovadă insuficientă de competență (Brennan & Resnick, 2012).

Ne bazăm aceste observații pe experiența noastră de lucru și de cercetare, precum și pe repere aflate în practică în sisteme educaționale cu tradiție de cercetare în didactica informaticii, detaliate în secțiunea 5.

2. Studiu de caz: un program care rulează nu demonstrează singur competența

Doi elevi prezintă proiecte la finalul unei unități de programare din clasa a VI-a, în care trebuiau să dezvolte într-un mediu vizual interactiv de programare, cum ar fi Scratch, un proiect care folosește repetiția, pornind de la un exemplu de proiect propus de profesor.

Elevul A: Schimbă decorurile, sunetele și personajele, dar păstrează aceleași blocuri-cod. Proiectul rulează perfect. Întrebat cum ar putea modifica codul pentru un alt rezultat, nu știe să răspundă.

Elevul B: Păstrează decorurile, sunetele și personajele proiectului-exemplu, dar reorganizează codul și rescrie repetițiile. Proiectul nu rulează inițial, însă elevul explică exact ce a urmărit; după o întrebare ajutătoare a profesorului, localizează eroarea și o corectează. Proiectul final are o dinamică diferită.

Situații de acest tip apar constant în observațiile noastre de clasă, iar un standard național trebuie să poată distinge între ele. Nu este suficient să constate că blocurile sunt așezate potrivit sau că programul rulează; standardul trebuie să surprindă ce înțelege elevul, cum proiectează o soluție, cum verifică rezultatul și cum învață din eroare. Diferențierea doar între „situație familiară" și „context nou" sau între „independent" și „cu sprijin" este un început util, dar insuficient singur.

3. Observații de fond

3.1. Standardele trebuie să spună ce face elevul, nu doar ce conținut a parcurs

La clasa a V-a, coloana „Standard de învățare" enumeră frecvent liste de conținuturi, fără să precizeze ce dovadă trebuie să producă elevul. În cazul variabilelor, de exemplu: elevul trebuie să le recunoască, să le urmărească valoarea, să le explice rolul sau să le folosească într-un program? Fiecare acțiune reprezintă o cerință cognitivă diferită. Clasa a VI-a urmărește mai atent acest aspect, cu standarde care încep cu acțiuni: „Elevul aplică", „Elevul elaborează".

Este necesar un audit al alinierii dintre competența specifică, standard, descriptor și dovada solicitată; acolo unde competența spune „identifică", iar descriptorul spune „proiectează", diferența de nivel cognitiv trebuie justificată sau corectată.

Standardele ar trebui să facă explicită progresia cumulativă și să solicite reutilizarea achizițiilor anterioare: la clasa a VI-a, repetiția se evaluează împreună cu secvența, selecția și variabilele învățate anterior, nu ca un concept care le înlocuiește.

3.2. Programarea trebuie evaluată ca proces, nu doar ca implementare

Formulări precum „implementează algoritmi dați", „plasează blocuri adecvate" sau „utilizează facilități" descriu doar o parte a activității. Ele nu arată dacă elevul a înțeles problema, a anticipat comportamentul, a ales structura, a testat programul și a corectat erorile. Un elev poate transforma mecanic un algoritm formulat de altcineva în blocuri, fără să poată proiecta sau explica o soluție.

Cercetarea în didactica programării susține aceeași concluzie. Metodologia PRIMM (Predict, Run, Investigate, Modify, Make), pe care o folosim în propriile noastre unități de învățare, structurează învățarea astfel încât predicția și citirea codului preced scrierea lui, cu efecte pozitive demonstrate într-un studiu cvasi-experimental cu circa 500 de elevi (Sentance, Waite & Kallia, 2019). În lectura noastră, aceste etape descriu și tipurile de dovezi pe care evaluarea le poate observa separat: înțelegere și predicție; proiectare; implementare; testare și depanare; explicare și îmbunătățire. Nu fiecare sarcină trebuie să le acopere pe toate, dar evaluarea anuală nu poate reduce întregul proces la implementare.

În întregul document, pentru toate cele patru clase, verbele testează, depanează și substantivul eroare nu apar nici măcar o dată în descriptori; rulare-execuția programului apare doar ca facilitate a mediului de dezvoltare, nu ca act de verificare pe care elevul îl realizează și îl consemnează. Unul dintre puținele elemente de proces formulate explicit, urmărirea pas cu pas a execuției, apare la clasele a V-a și a VI-a, dar nu este continuat suficient de clar la clasele a VII-a și a VIII-a, când trecerea la limbajul de programare o face mai necesară.

Testarea și depanarea nu sunt propuse ca noi conținuturi curriculare, ci ca dovezi prin care elevul demonstrează că înțelege și aplică algoritmii prevăzuți deja de programă.

3.3. „Familiar" și „nou" nu pot fi singurele criterii care separă nivelurile

În mai multe standarde se schimbă simultan structura algoritmică, complexitatea problemei, gradul de sprijin și noutatea contextului; din nivelul acordat nu mai rezultă ce știe elevul și ce trebuie învățat în continuare. La clasa a V-a, structura alternativă lipsește de la nivelul De bază în toate competențele de programare, apărând abia la Consolidat; la clasa a VI-a, repetiția condiționată apare de asemenea abia la Consolidat. Nivelul De bază poate fi astfel atins fără demonstrarea unei părți importante din conținutul anual.

În plus, noutatea poveștii nu înseamnă automat transfer computațional: elevul care schimbă pisica și mărul cu un robot și o stație folosește același algoritm. Iar sprijinul pentru citirea cerinței nu este același lucru cu sprijinul pentru alegerea structurii de programare: elevul care primește cerința citită cu voce tare și apoi proiectează, implementează și depanează singur nu a primit ajutor la programare. Sprijinul de acces, cel procedural și cel conceptual trebuie consemnate separat.

4. Recomandările noastre

4.1. Modificări imediate: principii de reformulare

Propunerile de mai jos disting între modificările necesare în textul standardelor și elementele care pot fi dezvoltate ulterior în ghidurile de aplicare, instrumentele de evaluare și programele de formare.

Propunem o formulă care poate fi aplicată sistematic standardelor de învățare și descriptorilor de programare, decurgând direct din observațiile de la secțiunea 3. O formulare coerentă respectă cinci reguli:

1. **Pornește de la acțiunea elevului și păstrează constructul central al competenței.** Standardul trebuie să arate ce face elevul cu ceea ce a învățat, nu doar ce conținut a parcurs. Verbele descriptorilor trebuie să rămână în aceeași familie de operații cognitive și să nu înlocuiască, fără o justificare explicită, competența evaluată cu o cerință diferită.
2. **Numește dovada observabilă.** Descriptorul precizează ce poate observa și consemna profesorul: elevul urmărește execuția unui program, anticipează un rezultat, implementează o soluție, testează, explică sau corectează o eroare. În funcție de competență, același descriptor poate integra mai multe dovezi relevante.
3. **Face explicită dimensiunea prin care crește performanța.** Diferențierea dintre niveluri trebuie să arate clar dacă progresul privește autonomia, complexitatea soluției, combinarea conceptelor, calitatea explicației, testarea sau transferul. Conținuturile prevăzute pentru anul respectiv rămân prezente de la nivelul De bază; nivelurile diferențiază performanța, nu materia evaluată.
4. **Precizează tipul de sprijin primit.** Sprijinul de acces, precum citirea cerinței sau clarificarea vocabularului, trebuie distins de sprijinul procedural, legat de utilizarea mediului, și de sprijinul conceptual, care indică structura algoritmică sau soluția. Acestea trebuie consemnate separat, în locul formulării generale „cu sprijin”.
5. **Include cel puțin o dovadă privind procesul de programare.** Evaluarea nu trebuie să se limiteze la existența unui produs care rulează. Urmărirea execuției, predicția, testarea, depanarea, explicarea sau revizuirea soluției oferă informații despre ceea ce înțelege elevul și despre următorul pas de învățare.

Aplicăm formula pe două cazuri, ilustrativ, cu mențiunea că reformulările sunt exemple de arhitectură, nu text normativ, și trebuie validate prin pilotare.

Exemplul 1: din listă de conținuturi în rezultat observabil.

La clasa a V-a, unele standarde sunt formulate ca liste de conținuturi. Profesorul vede temele asociate competenței, dar nu și acțiunea prin care elevul demonstrează că le-a înțeles. La CS.3.3, competența privind crearea de jocuri digitale, coloana „Standard de învățare" enumeră „Noțiunea de algoritm; proprietăți ale algoritmilor; clasificarea datelor [...]; constante și variabile; expresii; structura secvențială; structura alternativă; medii grafice interactive". Lista precizează materia, dar nu definește dovada.

Reformulare, standard de învățare. Elevul creează, testează și explică, într-un mediu grafic interactiv, un joc digital simplu în care o decizie schimbă desfășurarea jocului, folosind operații succesive și o condiție.

Notă de aplicare: sprijinul de acces, procedural sau conceptual se consemnează separat. Sprijinul conceptual este luat în considerare în interpretarea performanței, dar nu este inclus nediferențiat în formula nivelului.

De bază. Creează un joc scurt pornind de la o structură indicată, în care o condiție simplă produce două situații diferite; verifică, folosind două cazuri de test date, că ambele situații pot fi declanșate și descrie comportamentul observat.

Consolidat. Creează independent jocul, alege valorile sau acțiunile necesare pentru a testa ambele rezultate și explică legătura dintre condiție și comportamentul observat.

Avansat. Modifică independent jocul pentru o cerință cu structură nouă, de exemplu prin adăugarea unei reguli sau combinarea a două condiții, selectează cazuri de test relevante și justifică de ce soluția produce comportamentul dorit.

Exemplul 2: descriptori cu sprijinul consemnat separat, nu inclus în nivel.

Formulări precum „cu sprijin" și „în pași ghidați" nu spun ce fel de ajutor a primit elevul. Citirea cerinței cu voce tare sau explicarea unui cuvânt nu sunt echivalente cu indicarea condiției, a variabilei ori a ordinii blocurilor; fără această distincție, evaluarea poate măsura literația sau accesul la cerință în locul programării.

La clasa a VI-a, CS.2.1 descrie nivelul De bază prin utilizarea „cu sprijin" a facilităților mediului și implementarea algoritmilor „în pași ghidați", fără să precizeze dacă sprijinul privește citirea cerinței, interfața sau soluția algoritmică.

Reformulare, standard de învățare. Elevul utilizează un mediu grafic interactiv pentru a implementa, testa și explica un algoritm care include structuri repetitive.

Notă de aplicare: sprijinul de acces, procedural sau conceptual se consemnează separat. Sprijinul de acces nu reduce automat nivelul competenței de programare.

De bază. Implementează un algoritm pornind de la o structură repetitivă indicată, verifică rezultatul pentru un caz de test dat și descrie comportamentul observat.

Consolidat. Selectează și organizează independent blocurile necesare, verifică programul folosind cel puțin două cazuri de test și corectează o eroare simplă observată la rulare.

Avansat. Planifică și implementează independent soluția pentru o problemă cu structură nouă, selectează o strategie de testare și explică modul în care modificările realizate rezolvă cerința.

O precizare de aplicare rămâne valabilă indiferent de standardul reformulat. „Context nou" nu ar trebui redus la schimbarea personajelor, a decorului ori a temei proiectului, dar nici definit exclusiv prin distanța structurală. Distanța de transfer structural, cât de mult diferă structura problemei, a datelor și a relațiilor față de sarcinile exersate, este una dintre dimensiunile noutății și ar trebui făcută explicită; noutatea rămâne însă parțial relativă la parcursul elevului, iar o sarcină identică structural poate fi nouă pentru un elev care nu a mai întâlnit reprezentarea respectivă.

4.2. O propunere de reconstrucție pentru procesul de programare

Propunem ca standardele de programare să fie construite în jurul unui profil simplu, comun claselor a V-a și a VI-a: înțelegere și predicție; proiectare; implementare; testare și depanare; explicare și îmbunătățire. Profilul urmează distincția, consacrată în cercetarea gândirii computaționale, dintre conceptele folosite de elev, practicile prin care lucrează cu ele și perspectiva asupra propriului program (Brennan & Resnick, 2012), și este propus ca instrument formativ pentru profesor și elev, nu ca o nouă scală administrativă: o etichetă globală poate fi păstrată pentru raportare, iar profilul servește la identificarea următorului pas de învățare.

Exemplu de sarcină integrată, clasa a V-a: elevul proiectează un quiz cu două rezultate, îl implementează, testează ambele ramuri și explică de ce condiția produce rezultatul ales.

Un profil posibil, „proiectare: Consolidat; implementare: Consolidat; testare și depanare: De bază; explicare: Avansat; sprijin de acces: cerința citită cu voce tare", arată profesorului nu doar unde se află elevul, ci și pasul următor: selectarea independentă a două cazuri de test și, într-o sarcină în care este introdusă o eroare, localizarea cauzei înainte de modificarea codului.

4.3. Un singur pachet de sprijin pentru profesori, evaluare și progresie

Propunem ca standardele să fie însoțite de un pachet unitar de implementare, nu de documente separate fără legătură între ele: nivelul de referință așteptat la finalul fiecărei clase, semnificația nivelului De bază și intervenția pedagogică asociată, o hartă cumulativă V-VIII și o componentă practică evaluată prin rubrică, cu produse și răspunsuri-ancoră pentru calibrarea profesorilor. Nivelul Avansat se acordă tuturor elevilor care demonstrează criteriile, fără cote prestabilite.

Concret: pentru De bază, variații scurte ale aceluiași concept și eliminarea graduală a indiciilor; pentru Consolidat, transfer și testare independentă; pentru Avansat, comparația soluțiilor, abstracție și proiecte deschise.

Componenta practică poate include două sau mai multe sarcini scurte, pentru a reduce dependența rezultatului de o singură performanță; durata, condițiile de administrare și variantele pentru școlile cu acces limitat la dispozitive trebuie stabilite prin pilotare, iar evaluarea trebuie însoțită de rubrici analitice, răspunsuri-ancoră și calibrarea evaluatorilor.

5. Repere comparative care pot susține reformularea

Am ales trei tipuri de repere, pentru a susține cu argumente propunerile noastre: o disciplină din același pachet național (Fizică), un sistem educațional cu informatica obligatorie și tradiție de cercetare în didactica informaticii (Anglia) și cadrele de cercetare care fundamentează evaluarea procesului.

5.1. Ce poate fi transferat din standardele de Fizică

Standardele de Fizică (Anexa nr. 13), parte a aceluiași pachet aflat în consultare, oferă câteva soluții de articulare a performanței care pot fi transferate și adaptate.

Un standard general, formulat ca acțiune observabilă, este urmat de particularizări concrete pentru fiecare nivel, cu instrumentele folosite: la programare, echivalentul ar fi particularizări pentru un quiz, un joc sau un proiect interactiv.

Nivelurile se diferențiază prin completitudinea și corectitudinea lucrului, prin alegerea formei de prezentare și prin capacitatea ei de a susține concluzii, nu doar prin „context nou": la programare, calitatea testării, justificarea structurii, claritatea explicației.

Procesul de investigație din Fizică include explicit interpretarea datelor, argumentarea, limitele investigației și sursele de eroare. Este transferabilă nu echivalarea directă dintre eroarea experimentală și eroarea de programare, ci tratarea erorii ca dovadă relevantă despre proces. În Informatică, testarea, localizarea unei erori, explicarea cauzei și verificarea corectării pot deveni componente explicite ale standardului, nu activități presupuse.

5.2. Ce poate fi transferat din curriculumul de Computing din Anglia

Curriculumul de Computing din Anglia tratează programarea ca proces: proiectarea, scrierea, testarea și depanarea programelor, utilizarea secvenței, selecției, repetiției și variabilelor, precum și explicarea logică a soluțiilor.

Teach Computing face aceste procese observabile în activități și evaluare. De exemplu, în unitatea Sensing movement (Year 6), elevul își revizuiește planul, implementează codul, îl testează și îl depanează; conceptele introduse anterior sunt reluate și combinate cumulativ.

Din aceste repere pot fi transferate în standardele românești trei elemente: formularea rezultatelor prin acțiuni observabile, includerea explicită a testării și depanării și construirea unei progresii în care achizițiile anterioare sunt reutilizate, nu înlocuite de conținuturile nou introduse.

5.3. Cadre de cercetare pentru evaluarea procesului

Două repere din cercetarea în didactica informaticii susțin direct reconstrucția propusă la 4.2.

Cadrul Brennan–Resnick arată limitele evaluării bazate exclusiv pe produsul final și propune observarea conceptelor, practicilor și perspectivelor elevului asupra propriei creații.

Metodologia PRIMM structurează învățarea în etape care pot genera dovezi observabile distincte și pot oferi un limbaj util pentru formularea descriptorilor de proces.

Folosim ambele cadre în propria noastră muncă de dezvoltare de curriculum și de cercetare a rezultatelor de învățare, inclusiv într-un pilot în derulare de evaluare a unor lecții dedicate gândirii computaționale la clasa a V-a, în patru școli din mediul rural, cu instrumente pre/post și observație la clasă. Este o cercetare pe eșantion mic, dar ne dă o imagine directă despre ce pot observa și consemna realist profesorii în clasă.

6. Concluzie

Standardele propuse fac un pas important: afirmă că evaluarea la Informatică și TIC trebuie să se raporteze la competențe și la criterii comune, nu la preferințele individuale ale evaluatorului. Acest principiu trebuie păstrat și întărit.

Pentru programare, standardul devine însă util doar dacă distinge între reproducere și proiectare, între rulare și testare, între observarea unei erori și explicarea ei. Propunerile noastre nu cer rescrierea filosofiei generale a standardelor; ele urmăresc ca promisiunile declarate de claritate, comparabilitate, funcție formativă și orientare spre progres să devină vizibile în fiecare descriptor de programare și în fiecare decizie pe care profesorul trebuie să o ia după evaluare.

7. Cine suntem și de unde vin aceste observații

Asociația Techsoup, www.asociatiatechsoup.ro, lucrează din 2016 pentru și împreună cu profesorii de informatică și TIC din învățământul public obligatoriu. Programul nostru Predau Viitor lucrează

la intersecția dintre educația informatică și educația civică, cu profesori care formează elevi cu discernământ și putere civică față de tehnologiile digitale.

Sprijinim o comunitate profesională de peste 2.100 de profesori de informatică și TIC, mai mult de 40% din totalul cadrelor didactice din domeniu; în multe județe, unul din doi profesori de informatică face parte din comunitățile noastre de învățare. Am susținut implementarea programei de Informatică și TIC din 2017 prin formare gratuită pentru peste 1.000 de cadre didactice în programare vizuală și gândire computațională, cu accent pe Scratch și pe scenarii de predare adaptate infrastructurii reale a școlilor. Am dezvoltat primul program de introducere în inteligența artificială pentru profesori din România, urmat de peste 3.000 de cadre didactice, iar Platforma de Cursuri Predau Viitor oferă în prezent resurse pedagogice pentru peste 20.000 de profesori. Festivalul Digital Predau Viitor aduce anual, către peste 5.000 de profesori, practici și instrumente pedagogice actualizate în educația digitală și didacticile STEM.

Suntem partenerii din România ai Raspberry Pi Foundation și Google DeepMind în programul internațional Experience AI, distins în 2025 cu Premiul UNESCO King Hamad Bin Isa Al-Khalifa pentru ICT în Educație; prin el, peste 1.500 de profesori au fost formați să predea lecții despre bazele AI, siguranța digitală și utilizarea responsabilă a modelelor de limbaj. Am pilotat în România metode de dezvoltare a produselor digitale elaborate de Universitatea din Cambridge și am creat Accelerator, un program prin care profesorii de informatică și TIC își transformă clasele în incubatoare de produse digitale. Din 2023 facem parte din consorțiul proiectului Incubatorul Civic Digital, care formează 4.500 de profesori și intervine pedagogic în 30 de școli, cu echipe de profesori de TIC și de educație socială care proiectează împreună activități interdisciplinare. Predau Viitor Fellowship, programul nostru de pedagogie în acțiune desfășurat pe durata unui an școlar, rezervă mai mult de jumătate dintre locuri profesorilor de informatică și TIC.

Desfășurăm în prezent propria cercetare de clasă privind evaluarea gândirii computaționale la clasa a V-a, în patru școli din mediul rural. Asociația Techsoup este singura organizație din România laureată a Digital Skills in Education Award al Comisiei Europene (2017).

8. Referințe

Analiza se raportează la documentele puse în consultare de Ministerul Educației și Cercetării prin Centrul Național pentru Curriculum și Evaluare (2026): Anexa nr. 15 (standardele pentru Informatică și TIC, gimnaziu), Anexa nr. 1 (nota de prezentare), Anexa nr. 2 (documentul de politică educațională privind descriptorii de performanță) și Anexa nr. 13 (standardele pentru Fizică).

Referințe externe:

Brennan, K., Resnick, M. New frameworks for studying and assessing the development of computational thinking (2012). Lucrare prezentată la reuniunea anuală AERA, Vancouver. https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Department for Education. National curriculum in England: computing programmes of study (2013). https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

National Centre for Computing Education. Teach Computing: Key Stage 2 curriculum. https://teachcomputing.org/curriculum/key-stage-2

Sentance, S., Waite, J., Kallia, M. Teaching computer programming with PRIMM: a sociocultural perspective (2019). Computer Science Education, 29(2-3), 136-176. https://doi.org/10.1080/08993408.2019.1608781

Teach Computing. Programming B: Sensing movement, Year 6.
https://teachcomputing.org/curriculum/key-stage-2/programming-b-sensing

Sursele online au fost consultate între 15-19 iulie 2026.

Documentul a fost redactat cu sprijinul unor instrumente de inteligență artificială, utilizate pentru documentare primară și editare intermediară. Fondul conceptual, structura argumentativă, verificarea surselor și editarea finală reprezintă contribuția autorilor, membri ai echipei Asociației Techsoup.