**ROMÂNIA**

**MINISTERUL APĂRĂRII NAŢIONALE**

**ACADEMIA TEHNICĂ MILITARĂ „FERDINAND I”**

**FACULTATEA DE SISTEME INFORMATICE ȘI SECURITATE CIBERNETICĂ**

**Specializarea: Calculatoare și sisteme informatice pentru apărare și securitate națională**

**PLATFORMĂ EDUCATIVĂ PENTRU TESTARE CU ÎNTREBĂRI GENERATE PROCEDURAL**

ABSOLVENT:

**Std. Sg. Maj. Porfirie-Denissa PILIPĂUȚANU**

CONDUCĂTOR ŞTIINŢIFIC:

**Ș.L. dr. ing. Cristian CHILIPIREA**

Conține \_\_\_\_\_\_\_\_\_\_ file

Inventariat sub nr. \_\_\_\_\_

Poziția din indicator: \_\_\_\_\_

Termen de păstrare: \_\_\_\_\_\_

**BUCUREŞTI**

**2022**

# Abstract

Formarea unui număr mare de specialiști devine o provocare pentru resursa umană limitată din mediul universitar. Profesorii se confruntă în mod regulat cu sarcina de a crea itemi pentru testare din cauza publicării materialelor existente și accesului deschis la acestea. Astfel, apare nevoia de a crea într-un timp scurt un set amplu de exerciții.

Prin urmare, această lucrare își propune să prezinte un sistem de evaluare automatizat ce asigură varietatea de conținut prin intermediul generării procedurale a întrebărilor și a răspunsurilor aferente acestora. Ca o consecință a faptului că proiectul a fost conceput cu scopul de a inova modalitatea de evaluare din cadrul cursului de Structuri de Date și Algoritmi, acesta este focusat asupra generării exercițiilor de programare.

Există o multitudine de tipuri de exerciții care pot fi folosite pentru a evalua cunoștințele de programare ale unui student. În cadrul lucrării au fost abordate exercițiile ce evaluează cunoașterea și aplicarea unor algoritmi, precum și exercițiile semantice ce constau în înțelegerea fluxului unui program.

# Cuprins

[1. Introducere 1](#_Toc103326036)

[2. Literatură relevantă recentă 1](#_Toc103326037)

[3. Detalii privind implementarea practică 1](#_Toc103326038)

[4. Rezultate experimentale 1](#_Toc103326039)

[5. Concluzii 1](#_Toc103326040)

[Bibliografie 1](#_Toc103326041)

**Listă figuri**

[**Figură 1. Aliniament Justified** 2](#_Toc49881484)

**Listă tabele**

[**Tabel 1 Preferință citări** 1](#_Toc49881492)

**Listă de acronime**

AIG

API

AST

AQG

IEEE Institute of Electrical and Electronics Engineers

CPU Central Processing Unit

DBMS Database Management System

DST

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

# Introducere

În procesul educațional, testarea acumulării de cunoștințe prin întrebări este una dintre metodele fundamentale de evaluare. Acestea oferă beneficii precum stimularea memoriei pentru regăsirea și sedimentarea informației, focusarea pe conceptele importante din materialul studiat și în anumite circumstanțe obținerea de feedback atunci când există o nelămurire.

Ca o consecință a dezvoltării rapide și a nevoii de personal tot mai mari în industria tehnologiei informației, formarea unui număr mare de specialiști devine o provocare pentru resursa umană limitată din mediul universitar.

Profesorii se confruntă în mod regulat cu sarcina de a crea itemi pentru testarea cunoștințelor din cauza publicării materialelor existente și accesului deschis la acestea, iar compunerea manuală a întrebărilor poate deveni un proces complex ce necesită experiență, resurse și timp. Astfel, apare nevoia de a crea într-un timp scurt un set amplu de exerciții.

Printre sarcinile comune pe care profesorii trebuie să le îndeplinească se numără și corectarea testelor sau oferirea de feedback asupra rezolvărilor propuse de către studenți.

Caracterul repetitiv al sarcinilor expuse anterior subliniază importanța rezolvării problemei și impactul puternic pe care l-ar avea o soluție automatizată, precum reducerea resurselor necesare pentru producerea unui test cât și maximizarea timpului care poate fi dedicat de către profesori pentru sarcini mult mai importante.

În momentul de față soluțiile cele mai cunoscute și utilizate de evaluare automată au reușit să degreveze cadrele didactice de partea de corectare, folosind întrebările și răspunsurile furnizate de către acestea, fiind pretabile pentru orice domeniu. Pentru a preveni plagiatul, este necesară versatilitatea și complexitatea itemilor din cadrul testelor, iar din acest punct de vedere soluțiile menționate anterior necesită o bancă mare de itemi. Există și soluții particulare care au reușit să degreveze cadrele didactice de partea de creare de itemi și generare a răspunsurilor, precum și să asigure o versatilitate a itemilor, utilizând șabloane parametrizate ale întrebărilor și algoritmi specifici care să genereze răspunsul. Acestea nu mai sunt, în schimb, pretabile oricărui domeniu, iar utilizarea unor șabloane, deși previne plagierea, nu asigură întotdeauna înțelegerea materiei, rezolvarea acestora putând fi învățată mecanic.

Se poate observa că generarea procedurală are un rol important în cadrul automatizării sarcinii de creare a itemilor. Aceasta este una din ramurile creării de conținut multimedia, cunoscută pentru producerea, manipularea și modificarea datelor prin mijloace automate și se bazează pe fuziunea dintre conținutul furnizat de om și aleatorismul utilizat în cadrul algoritmilor[[1]](#footnote-1).

O constrângere în domeniul generării automate de întrebări este reprezentată de faptul că nu poate fi construită o soluție universal valabilă. Odată stabilite constrângerile particulare ale unui generator destinat unui domeniu, șabloanele parametrizate pot asigura o oarecare versatilitate a itemilor, însă nu asigură înțelegerea acestora de către student. Prin urmare, problema dificilă care trebuie rezolvată constă în furnizarea unui grad și mai mare de versatilitate.

Lucrarea de față propune o soluție la această problemă, țintind domeniul programării. Proiectul a fost conceput cu scopul de a dezvolta un sistem de testare automatizat și de a inova modalitatea de evaluare din cadrul cursului de Structuri de Date și Algoritmi. Predarea programării se confruntă cu probleme specifice, cum ar fi înțelegerea conceptelor de programare, dar și a algoritmilor pentru rezolvarea sarcinilor de programare. În cadrul cursurilor introductive, studenții încearcă să evite înțelegerea conceptelor urmând niște scurtături, cum ar fi învățarea codului prin memorare sau copierea programelor de la colegi.

Prin urmare, se profilează obiectivul principal pe care generatoarele de exerciții de programare trebuie să îl îndeplinească: furnizarea varietății necesare de exerciții pentru studenți astfel încât să le testeze abilitățile practice de-a lungul cursului și să rezolve problema plagiatului. Pentru a ajuta la rezolvarea problemei de scalabilitate a cursurilor, generatoarele de exerciții ar trebui să îndeplinească și alte obiective adiționale. O varietate de exerciții nu asigură faptul că acestea reflectă conceptele și tehnicile de programare urmărite în evaluarea cunoștințelor. Prin urmare, trebuie asigurată și îndeplinirea acestui obiectiv. De asemenea, testarea corectitudinii soluției studentului și furnizarea de feedback în mod automat reprezintă alte modalități prin care profesorul poate fi degrevat de o sarcină repetitivă și niște componente necesare în cadrul unui sistem de evaluare.

Există o multitudine de tipuri de exerciții care pot fi folosite pentru a evalua cunoștințele de programare ale unui student. În cadrul lucrării au fost abordate exercițiile ce evaluează cunoașterea și aplicarea unor algoritmi, precum și exercițiile semantice ce constau în înțelegerea fluxului unui program.

Pentru primul tip de exerciții soluția propusă constă în furnizarea tiparului întrebării, a unui fișier de configurare cu ajutorul căruia se determină termenii ce vor fi înlocuiți în cadrul acesteia cu datele generate și un fișier producător care încapsulează algoritmul vizat de către întrebare, care compilat și rulat va genera datele întrebării și rezultatul aferent.

Al doilea tip de exerciții a necesitat o soluție mai complexă deoarece o parte din întrebare constă dintr-o secvență de cod care trebuie să reflecte conceptele urmărite în evaluarea cunoștințelor. Pentru a produce acel cod a fost indispensabilă o abordare originală asupra unor concepte de design a compilatoarelor precum gramaticile sau tabela de simboluri. Spre deosebire de tipul precedent de exerciții acesta utilizează și un fișier cu gramatica folosită pentru generarea secvenței de cod. O altă diferență constă în faptul că fișierul producător este inițial incomplet și necesită inserarea secvenței de cod generate pentru a putea fi compilat și rulat.

Întrebarea și fișierele sunt introduse în cadrul platformei prin intermediul unei interfețe administrator. Ulterior, profesorul poate crea un test care să cuprindă întrebarea și să poată fi susținut de către student. O imagine mai detaliată asupra întregului sistem este furnizată în capitolele următoare.

Primul capitol prezintă succint importanţa temei, utilitatea unui astfel de sistem, precum și obiectivele lucrării ce se doresc a fi îndeplinite prin elaborarea proiectului. De asemenea, este furnizată o imagine de ansamblu asupra conținutului acestei lucrări și a modului în care aceasta este structurată alături de o prezentare sumară a rezultatelor obţinute.

Capitolul al doilea își propune să ofere prin intermediul literaturii relevante recente o perspectivă generală asupra contextului actual în care se află domeniul generării procedurale de întrebări și o analiză detaliată a soluțiilor similare și a implementărilor existente, cu un focus pe exercițiile de programare. Adițional, acesta va expune API-uri și componente software relevante.

//capitol cu gramatici

Cel de-al patrulea capitol cuprinde detaliile privind implementarea practică precum definirea cerinţelor complete pentru sistemul propus, prezentarea cazurilor de utilizare, definirea arhitecturii și proiectarea sistemului, tehnologiile sau API-urile folosite pentru dezvoltarea acestuia, descrierea cazurilor de testare și elaborarea raportului de testare, precum și problemele întâmpinate pe parcursul întregului proces.

Al cincilea capitol este dedicat prezentării rezultatelor experimentale, în urma măsurătorilor de performanță, iar ultimul capitol face o sinteză asupra principalelor idei din lucrare, furnizând direcţii și idei pentru continuarea acesteia.

# Literatură relevantă recentă

Pentru a reduce costurile asociate procesului manual de generare al întrebărilor și pentru a rezolva problema unui flux constant de conținut, au fost dezvoltate tehnici de generare procedurală ale acestora.

Acest capitol își propune să furnizeze o imagine de ansamblu asupra contextului actual în care se află domeniul generării procedurale de întrebări, o analiză detaliată a soluțiilor similare și a implementărilor existente, precum și dezavantajele fiecărei propuneri, punând accent pe cele care vizează exercițiile de programare.

În ciuda faptului că volumul cercetărilor științifice în domeniul generarării procedurale de întrebări a cunoscut o creștere în ultimii ani, munca privind generarea automată de exerciții pentru programare a fost relativ limitată în comparație cu celelalte subdomenii. Au apărut mai multe propuneri pentru dezvoltarea de software educațional specializat în a ajuta studenții să înțeleagă conceptele de bază de programare și să dezvolte abilități de rezolvare a problemelor. Soluțiile propuse în prezent se bazează în mare măsură pe tipare parametrizate și nu oferă gradul dorit de versatilitate și complexitate, augmentând astfel utilitatea prezentei lucrări.

## Metode generative pentru întrebările cu răspuns liber

Articolul ‚,A Systematic Review of Automatic Question Generation for Educational Purpose” (Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler & Salam Al-Emari) a analizat peste 90 de lucrări științifice legate de generarea întrebărilor în domeniul educațional, furnizând o imagine de ansamblu asupra comunității AQG, direcțiile curente și progresele realizate. Acesta prezintă etapele procesului generator și metodele generative pentru diferite tipuri de întrebări printre care se numără și întrebările cu răspuns liber vizate în cadrul proiectului, precum și modalitățile prin care se poate face evaluarea calității itemilor.

În domeniul evaluării automate există mai multe categorii de întrebări care pot fi utilizate în cadrul testelor. Exercițiile care pot fi folosite cu precădere în domeniul programării pot fi împărțite în trei categorii de bază: întrebări cu răspunsuri multiple, întrebări cu răspuns liber și sarcini de programare. Simplitatea întrebărilor cu alegere multiplă le-a făcut foarte populare în sistemele de management al învățării, cum ar fi Moodle. Cu toate acestea, în predarea programării, întrebările cu răspunsuri multiple pot fi utile doar pentru adoptarea unor elemente teoretice de bază, nu și pentru dobândirea de abilități practice în rezolvarea sarcinilor de programare. De aceea, proiectul curent este focusat exclusiv pe întrebările cu răspuns liber.

Alcătuirea întrebării (setul întrebare-răspuns corect) este principala sarcină a întregului proces de generare. În funcție de tipul acesteia și de formatul răspunsului sunt implicate diverse abordări. În cadrul clasificării metodelor de generare, metoda folosită în mod uzual pentru producerea întrebărilor cu răspuns liber este cea bazată pe șabloane ce constă în folosirea unei structuri de bază, ale cărei goluri sunt customizate cu valori care întrunesc caracteristicile sintactice sau semantice precizate. Șabloanele necesită intervenția umană prin construirea manuală a acestora, ceea ce face ca din perspectiva costului, această metodă să fie considerată scumpă. Un alt dezavantaj al folosirii acestei metode este dat de structură limitată a întrebărilor generate din punct de vedere al diversității lingvistice.

Din perspectiva costului nu a putut fi identificată o alternativă mai bună pentru generarea întrebărilor cu răspuns liber, din cauză că și restul metodelor prezentate au în acest moment un cost ridicat (ex. necesitatea unei cantități mari de date etichetate corespunzător). Metodele alternative nu au însă dezavantajul structurii rigide a întrebării. Deși acestea produc întrebări mult mai diversificate din punctul de vedere al topicii propoziției și al limbajului, nu sunt potrivite exercițiilor ce urmăresc să reflecte anumite concepte și tehnici de programare. Se justifică, astfel, folosirea uzuală a metodei șabloanelor în acest domeniu.

Ca practică generală, în urma generării, se realizează evaluarea calității întrebării. Criteriile de evaluare se orientează asupra calității lingvistice și educaționale. Din punct de vedere lingvistic se urmărește corectitudinea gramaticală, fluența, ambiguitatea semantică și lipsa erorilor. Din punct de vedere educațional se urmărește utilitatea întrebării, relevanța în cadrul domeniului și obiectivul didactic atins de subiectul evaluat prin răspunsul la acea întrebare. De asemenea, mai sunt și alte metrici relevante pentru calitate precum dificultatea sau nivelul cognitiv țintă. Determinarea dificultății întrebării se poate face prin determinarea numărului de persoane evaluate care au răspuns corect din totalul celor examinați, compararea dificultății prezise cu performanțele obținute de studenți sau prin utilizarea unor mecanisme de rezolvare automată[[2]](#footnote-2).

Ținând cont de observațiile făcute de către autori pe baza analizei compilației de lucrări științifice, implementarea acestui proiect folosește la bază ca metodă de generare șabloanele. Prin utilizarea acestei metode, întrebările întrunesc criteriile de calitate menționate anterior deoarece șablonul trece prin filtrul unei persoane cu experiența necesară a-l întocmi.

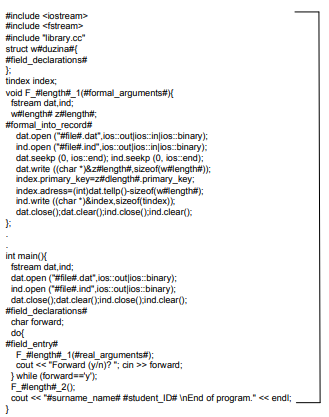
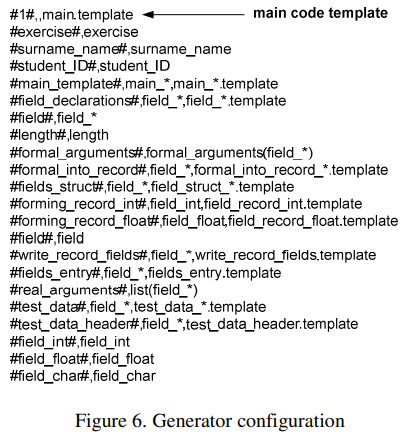
## Soluții ce utilizează șabloane parametrizate

Generatorul de exerciții de programare implementat în Python de D. Radošević, T. Orehovački și Z. Stapić, se bazează pe o listă de specificații, un fișier de configurare și tipare de cod în limbajul C++.

Lista de specificații conține elemente de tipul atribut:valoare, fiind constituită inițial din numele fișierului de output. Aceasta este ulterior modificată prin adăugarea mai multor elemente alese aleator din liste predefinite ce conțin nume de fișiere șablon sau diverse denumiri pentru identificatorii variabilelor. Lista finală de specificații conține atribute precum fișierul de output, numele studentului, ID-ul studentului, tiparul principal și alți parametrii specifici exercițiului.

Autorii lucrării menționează că au definit mai multe tipuri de tipare: un tipar cu codul principal al programului și cu exercițiile sub formă de comentarii și tipare cu dimensiuni mai mici ce conțin secvențe de cod. Exemplul furnizat de aceștia arată o posibilă structură pentru tiparul principal.

În cadrul procesului generativ, porțiunile delimitate de caracterul „#” din tipare sunt înlocuite prin intermediul valorilor asociate din fișierul de configurare, care referențiază fie un atribut din lista de specificații fie numele unui fișier cu tiparul de dimensiune mai mică. Înlocuirea se face, astfel, fie prin obținerea directă a valorii atributului din lista de specificații fie prin generarea tiparului menționat[[3]](#footnote-3).



Program code template

O problemă a acestei soluții sunt cerințele relativ mari pentru profesori, din cauză că fiecare exercițiu necesită un set particular de tipare. Deși acestea ar putea fi moștenite parțial de la exercițiile anterioare, iar configurația generatorului ar putea fi, de asemenea, moștenită parțial, întregul proces necesită un control granular asupra mulțimii de fișiere, ceea ce nu este fezabil pentru o diversitate de tipuri de exerciții. În implementarea proiectului, această problemă a multitudinii de tipare ce trebuie întocmite, a fost rezolvată prin generarea unei diversități de secvențe de cod prin definirea unei gramatici.

O altă problemă este lipsa furnizării unei soluții automate de notare a exercițiilor și metoda de furnizare a acestora prin comentarii. Acest aspect necesită timp suplimentar din partea profesorilor și ar putea fi îmbunătățit. Prin urmare, lucrarea de față a definit în procesul generator, un mecanism de producere a răspunsurilor, pentru ca ulterior soluția studentului să poată fi corectată în mod automat.

În cadrul lucrării ,,Automatic Generation of Programming Exercises for Learning Programming Languages’’ (Akiyoshi Wakatani, Toshiyuki Maeda), autorii au pornit de la premiza că la baza învățării unui limbaj de programare se află atât dobândirea cunoștințelor de gramatică a limbajului cât și capabilitatea de a înțelege fluxul unui program. Prin urmare, aceștia au pregătit două tipuri de exerciții: exerciții de sintaxă prin care cursanții să corecteze erorile programelor furnizate, și exerciții de semantică prin care cursanții să estimeze rezultatul programelor, fără a le executa. Acestea pot fi accesate prin intermediul unei aplicații web în limbajul PHP, care le generează automat folosind șabloane în limbajul C. Exercițiile generate sunt evaluate automat.

Pentru primul tip de exerciții, interfața grafică permite vizualizarea unui program cu erori și a descrierilor erorilor furnizate de compilator. Prin intermediul acesteia, cursanții corectează programul utilizând cunoștințele despre gramatica limbajului și mesajele de eroare până când toate erorile sunt complet eliminate.

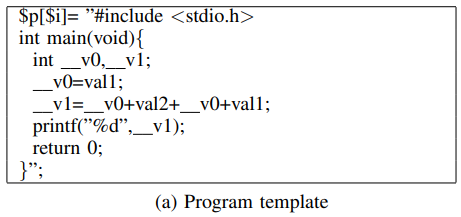
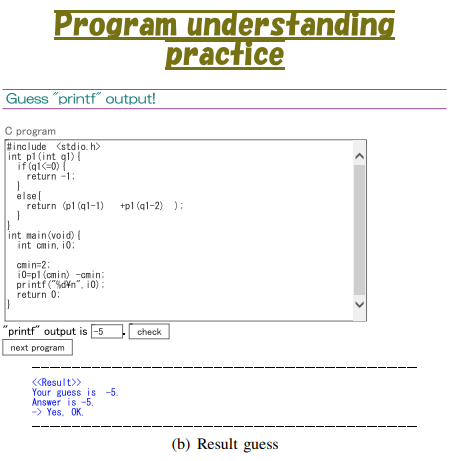
Exercițiile sunt generate prin adăugarea de bug-uri unor programe șablon și prin înlocuirea numelor de variabile. Au fost definite trei metode de adăugare a erorilor la programul șablon. În primul rând, numele variabilelor sunt înlocuite cu valori alese aleator dintr-o listă predefinită de nume. Astfel, pot apărea în program variabile nedeclarate. A doua metodă se bazează pe scrierea greșită a unor șiruri de caractere definite în prealabil (de exemplu, stdio.h poate fi schimbat în stdi.h). A treia metodă înlocuiește un șir de caractere cu un alt cuvânt desemnat, pentru a genera o eroare întâlnită des (de exemplu, înlocuirea „;” cu „:”).

După ce un cursant corectează un program care are erori, scriptul PHP salvează programul corectat ca fișier cu un nume unic, îl compilează și trimite înapoi mesajul de la compilator către cursant. Șirul de caractere „OK” este trimis înapoi, dacă compilatorul nu produce niciun mesaj.

Autorii menționează că programele șablon sunt pregătite în conformitate cu progresul învățării și sunt necesare între 10 și 20 programe șablon pentru a produce o varietate de programe.

Al doilea tip de exerciții, cele de estimare a output-ului unui program, au scopul de a îmbunătăți capacitatea de înțelegere a programelor din punct de vedere semantic. Interfața grafică a aplicației web permite vizualizarea unui program care conține diverse instrucțiuni. Cursantul citește programul și estimează rezultatul funcției prin înțelegerea semanticii acestuia. După scrierea valorii estimate în zona de răspuns și apăsarea butonului de verificare, este afișată comparația răspunsului corect cu estimarea.

Autorii menționează că programele date pot include apeluri recursive de funcții și calcule complicate, ceea ce face mai dificil de estimat rezultatul. Pentru a crea programe corecte, este indispensabilă înțelegerea comportamentului programelor care includ instrucțiuni condiționale, bucle și apeluri de funcții. Prin urmare, aceștia susțin că aceste exerciții sunt utile pentru a îmbunătăți capacitatea cursanților de a înțelege programe. Prin schimbarea adecvată a numelor variabilelor și a valorilor constantelor și menținând lungimea programului suficient de compactă pentru estimare, cursanții pot efectua exerciții în mod repetat pentru a spori eficacitatea educației. Parametrii constanți și modelele de calcul trebuie să fie date corect, deoarece parametrii greșiți pot provoca excepții precum împărțirea la zero sau erori de acces la memorie[[4]](#footnote-4).



Justificarea obiectivelor didactice atinse de către exercițiile menționate, le fac pretabile pentru a fi produse de către generatorul implementat în cadrul proiectului. Ținând cont de faptul că primul tip vizează noțiuni de bază introductive și nu reprezintă o provocare pentru studenții acomodați cu un limbaj de programare, a fost ales doar al doilea tip de exerciții. Pentru acesta, soluția curentă furnizează o gamă largă de secvențe de cod unice fără a mai fi necesară definirea a 10-20 de tipare.

## Soluții ce utilizează noduri de instrucțiuni

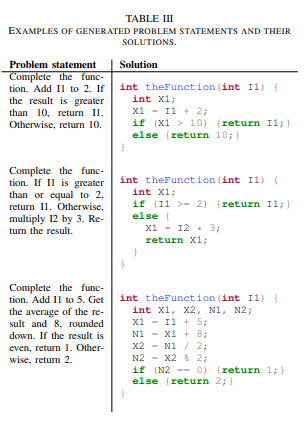
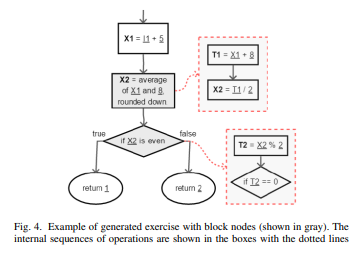
Deși întrebările parametrizate pot fi utile pentru a preveni plagiatul, ele necesită destul de mult timp din partea cadrelor didactice pentru crearea tiparelor. Un potențial răspuns la această problemă este furnizat de lucrarea ,,Procedural Generation of Programming Exercises with Guides Based on the Student’s Emotion” (Thomas James Tiam-Lee, Kaoru Sumi), ce prezintă o abordare pentru generarea procedurală a exercițiilor de programare fără a utiliza șabloane parametrizate. Pentru a reprezenta exercițiile într-o manieră structurată, fiecare exercițiu este tratat ca o succesiune de instrucțiuni pe care elevul trebuie să le efectueze. Fiecare instrucțiune este reprezentată de un singur nod, care poate fi asociat cu o operație, o condiție sau returnarea unei valori. Prin urmare, exercițiul este o colecție de noduri, care va fi parcursă succesiv.

Un nod de operație, se referă la o singură operație aritmetică și este urmat întotdeauna de un singur nod. Un nod de condiție, se referă la o singură expresie condițională și este întotdeauna urmat de două noduri, unul pentru când condiția este adevărată și unul pentru când condiția este falsă. Un nod de returnare, se referă la un punct final al funcției și nu este urmat de un alt nod. Operații de nivel superior pot fi adăugate prin abstractizarea lor ca blocuri de operații sau de condiții. Blocurile constau într-o secvență internă de operații elementare.

Pornind de la o valoare definită a complexității codului dorit, se generează o structură aleatorie. Complexitatea unei structuri este o metrică definită ca numărul de noduri asociate cu o operație sau cu o condiție. Această definiție se bazează pe ideea că un exercițiu devine mai complex pe măsură ce crește numărul de instrucțiuni necesare pentru a-l finaliza.

Primul nod este inițializat fie ca nod de operație fie ca nod de condiție. Apoi, într-unul din locurile disponibile din cadrul structurii este atașat aleatoriu, în mod asemănător, fie un nod de operare, fie un nod de condiție. Această operațiune este efectuată până când structura conține un număr de noduri egal cu complexitatea. În locurile libere rămase, sunt atașate nodurile de returnare.

După definirea structurii exercițiului, parametrii fiecărui nod sunt asignați utilizând un algoritm ce a luat în calcul considerațiile detaliate în cadrul lucrării. Odată construită secvența de instrucțiuni, fiecare nod este parcurs și în funcție de tipul acestuia este generat atât un text în limbaj natural pentru crearea exercițiului, cât și codul Java asociat pentru a evalua soluția elevului. Pentru crearea cerinței, există mai multe texte predefinite pentru fiecare tip de nod, iar din acestea este ales unul, în mod aleator. Ulterior, porțiunile marcate goale sunt înlocuite cu variabile și parametrii din structura nodului[[5]](#footnote-5).



În acest studiu au fost folosite doar operații simple de calcul pentru producerea exercițiilor de programare precum operațiile aritmetice (de exemplu, adunarea, înmulțirea), operațiile condiționale (de exemplu, mai mare, mai mic) și câteva blocuri. De asemenea, tipurile de date sunt limitate doar la numere întregi. Din acest motiv, exercițiile generate, în ciuda faptului că nu sunt derivate dintr-un șablon parametrizat, arată în continuare similare între ele. Prin urmare, soluția propusă în lucrarea de față și-a propus și a reușit să obțină secvențe de cod mai complexe care conțin și bucle și care nu se limitează doar la un anumit tip de date. De asemenea, ideea că un exercițiu devine mai complex pe măsură ce crește numărul de instrucțiuni necesare pentru a-l finaliza a contribuit la definirea metricii asociate complexității folosite în cadrul proiectului.

## Soluții ce utilizează gramatici (biblioteca Tracery.js)

Generarea procedurală este utilizată într-un spectru larg de domenii, însă cel mai popular dintre acestea este cel al jocurilor, unde texturile, efectele sonore și povestea sunt produse cu ajutorul gramaticilor.

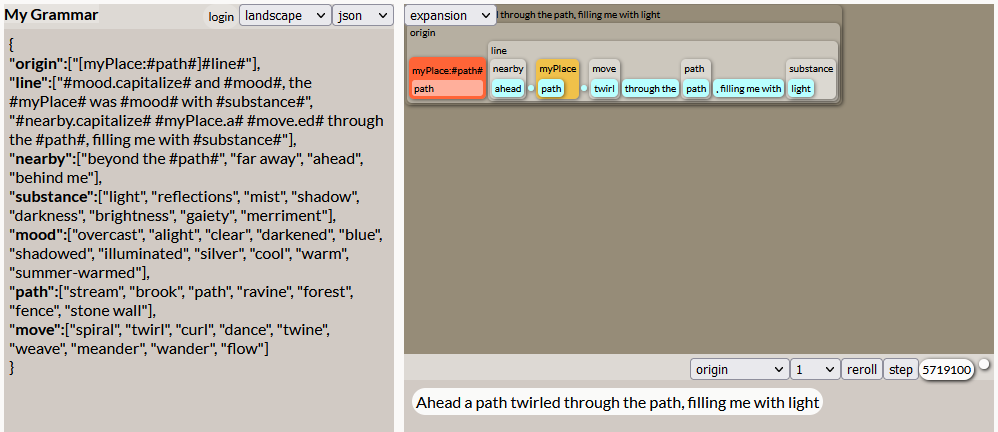
//Gramatici

Această metodă de generare nu este comun folosită în domeniul educațional. Pornind de la faptul că întrebările sunt de fapt un text, iar gramaticile pot produce o gamă diversificată de texte utilizând un set de reguli de producție, este foarte probabil ca această metodă să se preteze domeniului. De asemenea, pentru o particularizare a întrebărilor de natură generală la întrebări ce vizează exerciții de programare, această abordare e o opțiune la fel de viabilă. Astfel, utilizând această euristică, lucrarea de față și-a propus să obțină întrebări cu un grad mare de versatilitate prin intermediul gramaticilor.

Tracery este un instrument generativ de text conceput cu scopul de a fi folosit de către autori începători și experți sau de către persoane care nu se pot autoidentifica drept „programatori” pentru crearea unor combinații surprinzătoare de text în mod algoritmic. În practică, această bibliotecă a fost utilizată pentru crearea unei game largi de povești, poezii, dialoguri și chiar imagini în format SVG sau pagini HTML cu gramatici profund imbricate.

Utilizarea șabloanelor și gramaticilor pentru a crea text structurat, dar variabil în același timp reprezintă o tehnică comună în generarea de povești. Generatorul primește la intrare o gramatică formală scrisă ca obiect JSON, ce conține simbolurile și regulile de expandare asociate acestora. Aceasta va trece printr-un sistem modular de părți interfuncționale: un parsator și un motor de expandare. Prin înlocuirea treptată a simbolurilor cu una din regulile aferente acestora aleasă în mod aleator, se asigură variabilitatea în timp și se ajunge la forma finală a textului. Sintaxa limbajului Tracery este simplă și lizibilă, permițând utilizatorului să definească o gramatică în limbaj natural. Pentru a semnala simbolurile care vor fi extinse recursiv sunt folosite hashtag-uri.

Această tehnică a fost îmbunătățită cu ajutorul unui mecanism care generează text într-un mod mai riguros prin utilizarea unor funcții care pot fi aplicate după ce un simbol este expandat. Aceste funcții sunt numite modificatori. Astfel, au fost rezolvate problemele generative comune precum pluralizarea, conjugarea sau scrierea cu literă mare[[6]](#footnote-6).

[[7]](#footnote-7)

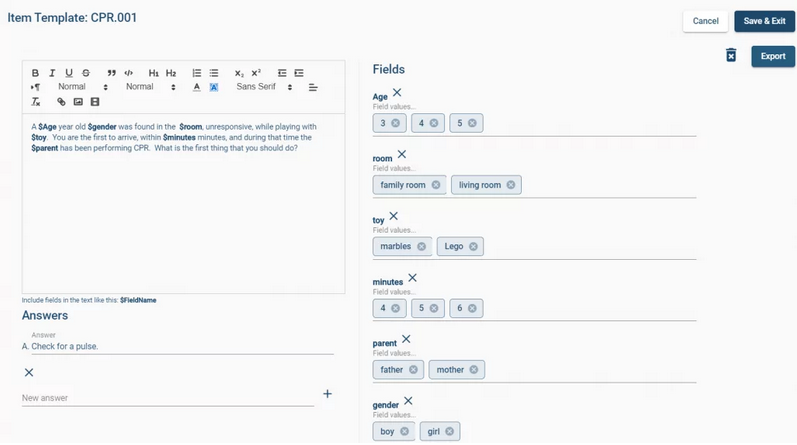
Cu această structură de date simplistă, autorii pot produce un text structurat și interesant din prisma faptului că poate fi lipsit pe alocuri de logică, însă în cazul exercițiilor de programare, acest lucru nu este de dorit. Deși biblioteca prezentată se pretează generării de text, nu permite un control mai fin al procesului generator pentru a putea produce un cod fără erori. Prin urmare, proiectul a implementat un generator bazat pe conceptele menționate anterior, adăugând un set de filtre care să verifice dacă o instrucțiune are sens în raport cu instrucțiunile generate anterior.

## Soluții comerciale similare (ASC, Brio)

Există mai multe instrumente folosite în procesul de evaluare, variind de la instrumente folosite doar de anumite universități până la instrumente care au fost dezvoltate pentru uz comercial.

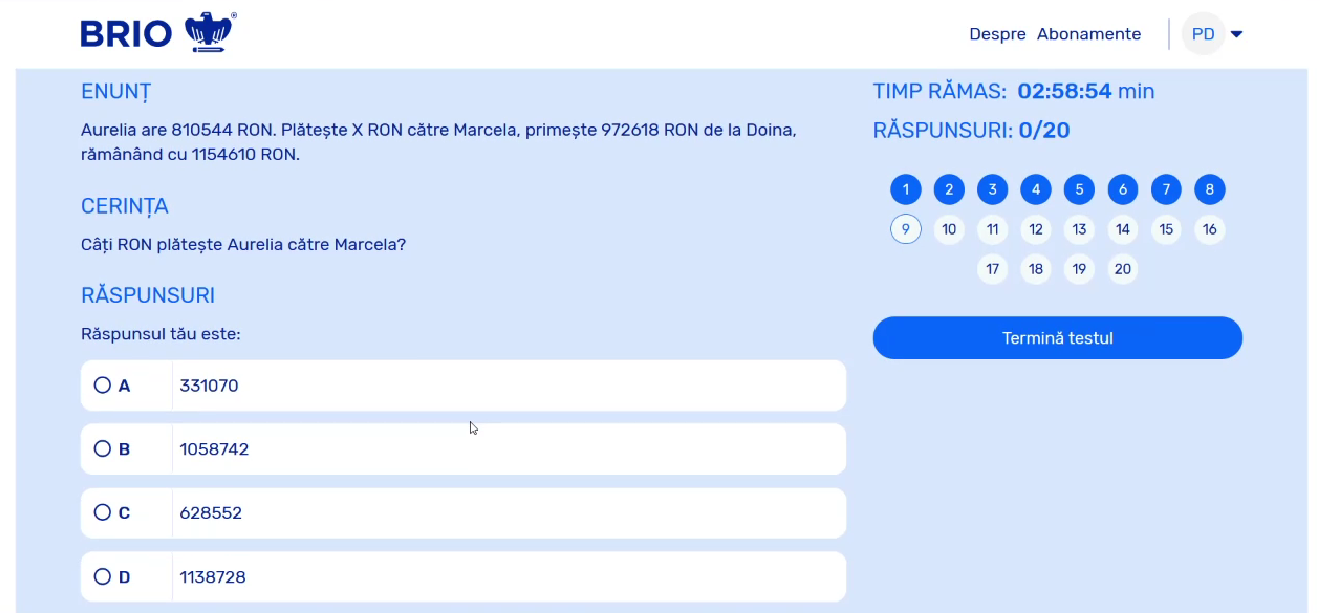
Platforma de testare ASC utilizează generarea automată de itemi (AIG), încercând să automatizeze o parte din efortul implicat în crearea articolelor, din cauză că acesta este unul dintre aspectele cele mai consumatoare de timp ale dezvoltării unui test.

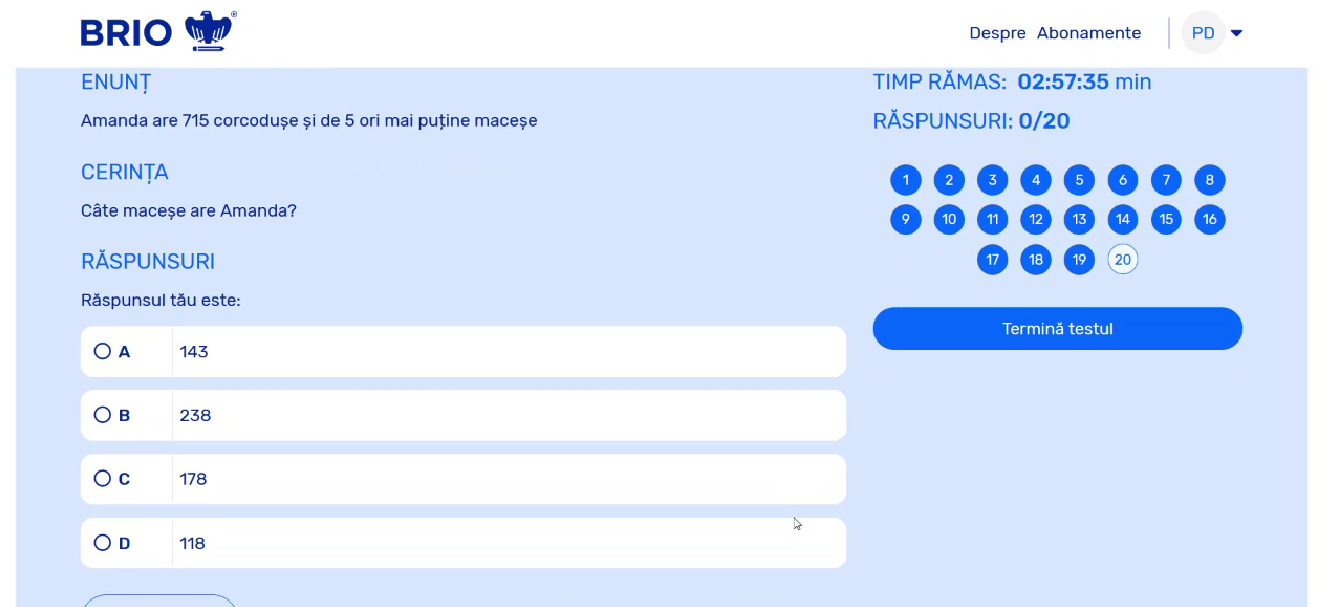
Tehnologia curentă de generare automată a unei game largi de itemi se bazează pe șabloane. Utilizatorul marchează prin intermediul simbolului $ porțiunile din text care sunt inserabile și definește pentru fiecare porțiune niște valori ce vor fi folosite în momentul înlocuirii. Algoritmul va produce toate permutările posibile ale șablonului. Itemii rezultați nu sunt mutați automat în banca de itemi, ci sunt revizuiți de persoane autorizate, din motive precum crearea de scenarii puțin probabile sau irelevante. Cu toate acestea, eficiența generală în procesul de creare al întrebărilor s-a îmbunătățit considerabil, iar itemii generați sunt de o calitate mai bună decât itemii scriși în mod tradițional, deoarece procesul i-a făcut pe autori să se gândească mai profund la ceea ce evaluează și cum. Platforma nu recomandă utilizarea mai multor instanțe ale aceluiași tip de item generat procedural în cadrul unui formular de testare[[8]](#footnote-8).



//răspunsurile trebuie furnizate de către autorul întrebării, un șablon parametrizat are o structură rigidă în timp ce o gramatică are atât o structură rigidă prin regulile ei, dar și variabilitate prin modul de alegere al regulilor.

Platforma de testare Brio furnizează elevilor din ciclul primar (I-IV) teste de antrenament bazate pe generarea programatică, dinamică și randomizată de itemi. Această platformă deține un număr foarte mare de itemi calibrați prin testarea pe eșantioane reprezentative de elevi români, garantând acoperirea minuțioasă a întregii programe școlare anuale, în cadrul fiecărui test[[9]](#footnote-9).





//răspunsurile sunt generate automat, tipare simpliste, deși nu e precizată explicit metoda pare că e bazată pe șabloane, o gramatică ar permite obținerea unor soluții mai versatile din punctul de vedere al poveștii. În loc să fie construiți un anumit număr de itemi, o gramatică permite generarea tuturor tiparelor prin definirea doar a ei

# Gramatici

# Detalii privind implementarea practică



**Figură 1. Aliniament Justified**

# Rezultate experimentale

# Concluzii

# Bibliografie

|  |  |
| --- | --- |
| [1] | J. Peat, E. Elliott, L. Baur and V. Keena, Scientific writing: easy when you know how, 2013. |
| [2] | Google, "Google Scholar," [Online]. Available: https://scholar.google.com/. [Accessed 01 September 2020]. |

PAGINA GOALĂ

1. https://en.wikipedia.org/wiki/Procedural\_generation [↑](#footnote-ref-1)
2. Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, Salam Al-Emari, ,,A Systematic Review of Automatic Question Generation for Educational Purposes’’, International Journal of Artificial Intelligence in Education, vol. 30, 2020, p. 121-204 [↑](#footnote-ref-2)
3. Danijel Radošević, Tihomir Orehovački, Zlatko Stapić, Automatic On-line Generation of Student's Exercises in Teaching Programming, Central European Conference on Information and Intelligent Systems, CECIIS 2010, Varaždin, 22-24 September 2010, p. 87-93 [↑](#footnote-ref-3)
4. Akiyoshi Wakatani, Toshiyuki Maeda, Automatic Generation of Programming Exercises for Learning Programming Languages, 2015 IEEE International Conference on Computer and Information Science, Las Vegas, USA, 28 June 2015 - 01 July 2015, p. 66-71 [↑](#footnote-ref-4)
5. Thomas James Tiam-Lee, Kaoru Sumi, ,,Procedural Generation of Programming Exercises with Guides Based on the Student’s Emotion”, IEEE International Conference on Systems, Man and Cybernetics, Miyazaki, Japan, 7-10 Oct. 2018, p. 1465-1470 [↑](#footnote-ref-5)
6. Kate Compton, Ben Kybartas, Michael Mateas, ,,Tracery: An Author-Focused Generative Text Tool”, International Conference on Interactive Digital Storytelling, Copenhagen, Denmark, November 30 - December 4 2015, p. 154–161 [↑](#footnote-ref-6)
7. https://tracery.io/editor/ [↑](#footnote-ref-7)
8. https://assess.com/what-is-automated-item-generation/ [↑](#footnote-ref-8)
9. https://brio.ro/teste-scolare [↑](#footnote-ref-9)