АВТОМАТИЗАЦІЯ НАВЧАННЯ ПРОГРАМУВАННЮ В КОНТЕКСТІ КОНСТРУЮВАННЯ ІЄРАРХІЧНИХ ПРОГРАМНИХ СТРУКТУР

Головін М.Б., Сомик О.І.

Волинський національний університет ім. Лесі Українки

Стаття містить опис досліджень та програмних розробок у області застосування графів у процесі навчання та на етапі перевірки знань шляхом тестування. У роботі описані підходи до створення ефективного тестового програмного засобу, що оперує завданнями, котрі передбачають конструювання ієрархічних структур. Приводиться опис нового авторського автентичного програмного продукту, котрий є прототипом, що реалізує інноваційні ідеї використання графів при тестуванні учнів та загалом реалізує підтримку роботи з ієрархічними конструкціями.

Ключові слова: *програмне забезпечення, тестування, графи, інформаційні технології.*

Основою автоматизованої діагностики процесу навчання в комп'ютерному класі є періодичне виконання учнями серій завдань у межах відповідних тестових програм. За допомогою цих програм здійснюється якісний диференційований зворотній зв'язок між викладачем та групою суб'єктів навчання. Зрозуміло, що цей зв'язок не безпосередній, а опосередкований: він реалізується через комп'ютери, зв'язані в локальну мережу. Останнє дозволяє в значній мірі автоматизувати та індивідуалізувати процес навчання. Ефективність автоматизованого навчального процесу залежить від програмних засобів, що для цього використовуються, та структури завдань, закладених у цей процес. У цьому контексті важливим є інформаційне наповнення тренажерної програми: зміст завдань, структурна, їх організація та можливі послідовності пред'явлення. Реалізація пакетів завдань, однорідних за кількісними та якісними показниками, дозволяє в значній мірі спростити механізм керування навчанням. У цьому сенсі важлива формалізація понять „кількість" та „складність" розумовій роботі стосовно кожного завдання.

Отримання кількісних параметрів будь-якого процесу, у тому числі пізнавального, передбачає розуміння закономірностей його протікання. У когнітивно орієнтованих течіях теоретичної психології активно розроблюється ідея про те, що інтелектуальна діяльність у значній мірі детермінується структурною організацією пізнавальної сфери [1].

На жаль, реальний типовий, тестологічний підхід у навчанні, як правило, акцентований не на психологічному механізмі процесу навчання, а на вибірковій перевірці засвоєння декларативних знань. Такий підхід дає мало користі для формування спеціалістів у сфері високих технологій.

Практична діяльність у цій технологічній галузі пов'язана з багатокроковим, абстрактно-логічним, причинно-наслідковим, продуктивним мисленням, яке детермінує діяльність. Це стосується, діяльності по створенню, модернізації, відлагодженню складно-організованих об'єктів, наприклад, таких як програмні засоби та електронні пристрої. Діяльність первинна в цій сфері. Вона формує процеси сприйняття, спосіб диференціації та структурування знань, особливості зв'язків у понятійній сфері.

Специфіка практичної навчальної діяльності стосовно складних добре формалізованих штучних об'єктів полягає в тому, що людина не може оперувати одночасно всіма багаточисельними функціональними вузлами складного об'єкта, які вона сприймає і розрізняє. Їх число навіть в навчальних задачах набагато перевищує ту кількість, яку можна одночасно утримувати в полі уваги. Суб'єкт навчання змушений у процесі інтелектуальної роботи масштабувати поняттями, не виходячи в процесі ментальних дій за межі магічного числа Міллера 7±2 [2]. Саме це число характеризує об'єм короткочасної пам'яті людини. У результаті тривалої інтелектуальної діяльності, суть якої абстрактно-логічне масштабування поняттями, у довготривалій пам'яті поступово формується ментальна вербальна конструкція

ієрархічного типу - пізнавальна (когнітивна) схема об'єкта навчання (наприклад, програми). Особливості навчальних дій стосовно програмних об'єктів у контексті ієрархічних ментальних структур були розглянуті в роботі [3].

Такі технології програмування, як метод низхідної покрокової деталізації та модульного програмування, використовуються у відповідній професійній діяльності завжди [4, 5]. Цей момент опосередковано підтверджує описаний вище механізм мислення, що спирається на ієрархічну конструкцію. Так, при низхідній покроковій деталізації проблеми, ієрархічна конструкція вибудовується з верху вниз, тобто від поодиноких узагальнених понятійних одиниць до великої кількості конкретизованих одиниць – операторів мови програмування. Саме коли кожний окремий крок вирішення деталізованої проблеми можна представити окремим оператором і починається процес безпосереднього програмування. Технологія модульного програмування сповідує альтернативний шлях мислення. Від великої кількості конкретизованих одиниць операторів до поодиноких узагальнених, що характеризують програмні блоки. По суті остання технологія дозволяє представити в мисленні велику сукупність логічно зв'язаних елементів, як монолітну поодиноку відокремлену сутність, що характеризується окремим поняттям або короткою концепцією.

Актуальною проблемою, на думку авторів, є розробка програмних засобів для підтримки навчальних процесів, у яких відбувається не тільки просте понятійне відтворення, а реалізуються навчальні причинно-наслідкової та абстрактно - логічної дії.

Метою цієї роботи є розгляд підходів до створення ефективного тестового програмного засобу, що оперує завданнями на конструювання ієрархічних структур. У цьому контексті мета має дві складові: з одного боку, існує необхідність в розробці та апробації відповідних завдань; з іншого боку, є потреба в розробці самої програми, що підтримує роботу з ієрархічними конструкціями.

У роботі розглядались завдання двох видів.

Перший вид завдань може бути реалізований на базі різноманітних ієрархічних конструкції, які вказують на відношення між понятійними одиницями. У деяких конструкціях це відношення загальне – конкретне, а також ціле – складова частини. У завданнях цього виду, немає різниці, у якому порядку слідують однорівневі елементи структури. Апробація цього виду завдання була проведена на базі відтворення структури обчислювальної системи, схеми типів файлів файлової системи Windows та дерева типів даних мови програмування Delphi.

Другий вид оригінальних завдань охоплював напрям конструювання алгоритмів та програм в вигляді ієрархічних структур. Як і в першому випадку, ці структури містять як конкретні так і узагальнені поняття.

Під конкретними поняттями в останньому пункті розуміються оператори програми, а під узагальненими - понятійні одиниці, що об'єднують сукупності логічно зв'язаних операторів. Ключова відмінність завдань другого виду від першого полягає в тому, що при перевірці виконання цих завдань відслідковується не тільки наявність окремих гілок дерева, а і їх порядок. У завданнях цього виду важливу роль має порядок, у якому слідують однорівневі елементи структури, адже ці елементи є окремими кроками в подоланні проблемної ситуації.

Ілюстрація формалізації об'єктів навчальної діяльності за допомогою деревоподібних структур саме на прикладі програмування має свої ексклюзивні переваги перед будь-якими іншими напрямками навчальної діяльності. Програма, як навчальний об'єкт, легко масштабується за розміром і складністю; вона добре формалізована і структурована. Діяльність стосовно програми відбувається у вікні текстового редактора і завжди доступна для автоматизованого моніторингу за допомогою діагностичної програми, що працює паралельно із засобами програмування в багатозадачному операційному середовищі Windows.

Розглянемо перший тип завдання на наступному прикладі. Нехай нам необхідно утворити структуру обчислювальної системи з таких компонентів як процесор, оперативна та

постійна пам’ять, комп’ютер, програмна та апаратна складова обчислювальної системи, різноманітні контролери тощо.

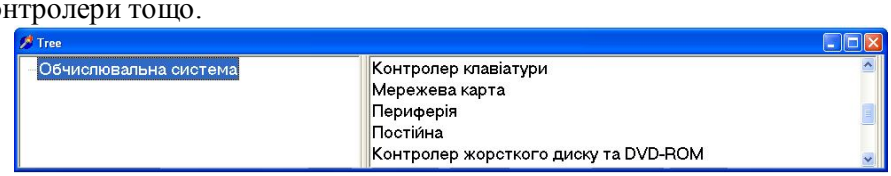

Рис.1. Інтерфейс начальної програми, що оперує завданням першого типу. Стартова ситуація в виконанні завдання.

Розглянемо перший тип завдання на наступному прикладі. Нехай нам необхідно утворити структуру обчислювальної системи з таких компонентів, як: процесор, оперативна та постійна пам’ять, комп’ютер, програмна та апаратна складова обчислювальної системи, різноманітні контролери, тощо.

Інтерфейс вікна виконання складається з двох частин рис.1 – 5. З лівої сторони знаходиться дерево, а з правої список компонентів для конструювання цього дерева. На початковому етапі виконання завдання ліва сторона майже порожня. На старті, з лівої сторони тільки корньовий елемент дерева, а з правої повний список компонентів. Виконання завдання полягає в тому, що необхідно будувати деревоподібну структуру перетягуючи компоненти з списку, тобто справа, наліво.

На рис.1 зображена саме така ситуація. Список компонентів справа доступний через відповідну смугу прокрутки. Порядок слідування окремих елементів у списку випадковий.

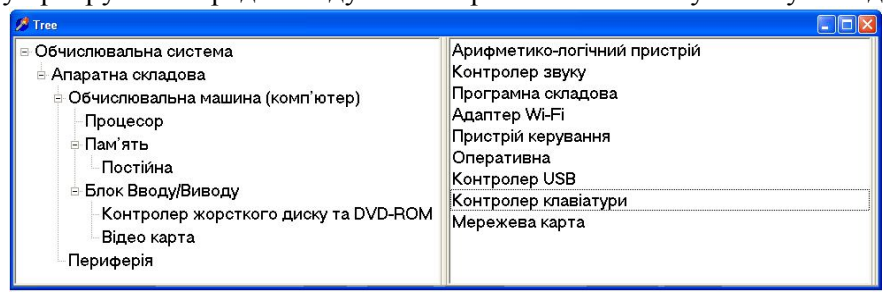

Рис. 2. Інтерфейс начальної програми, що оперує завданням першого типу. Завдання виконано наполовину. Праве вікно програми, ще заповнено невикористаними елементами конструювання

На рис.2 представлена ситуація, коли завдання на конструювання виконано не повністю. З лівої сторони утворена ієрархічна конструкція, а кількість елементів справа значно менша, ніж на початку роботи.

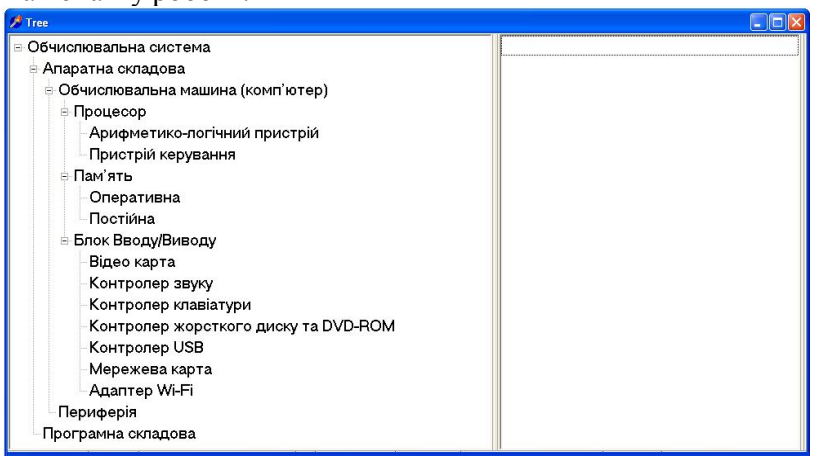

Рис.3. Інтерфейс начальної програми, що оперує завданням першого типу. Завдання виконано повністю. Усі елементи конструювання в лівому вікні. Вони включені у відповідну структуру. Праве вікно програми порожне

На рис. 3 представлена ситуація, коли завдання виконане повністю. В завданні цього типу, як зазначалось вище, немає різниці, у якому порядку слідують одно-рівневі елементи структури. Тому правильним переліком вузлів процесора може бути список альтернативний, показаному на рис. 2. Тобто список, у якому спочатку записаний пристрій керування, а потім арифметико–логічний пристрій. Можливі різні альтернативні рис. 2 правильні послідовності вузлів комп'ютера, наприклад така: пам'ять, блок вводу/виводу, процесор.

Необхідно відмітити, що список справа може бути і надлишковим. Тоді наприкінці виконання завдання права частина не є повністю порожньою. Такі завдання надають побудові структури додаткової складності.

Програмна структура виконана в форматі ієрархії – зовні дуже схожа на структуру класифікації понятійних одиниць. Однак у випадку тексту програми – зміст окремих елементів і всієї конструкції дещо інший. Вузли графу об'єднують в єдине ціле оператори мови програмування. У такій конструкції важливий порядок елементів.

Приклади таких зображень можна побачити на рис. 4 та 5. Конструкція на рис. 3 не завершена, а на рис. 4 має закінчений вигляд.

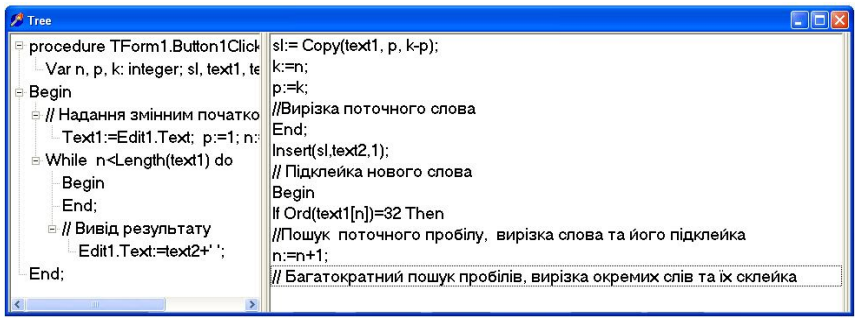

Рис.4. Інтерфейс начальної програми, що оперує завданням другого типу. Завдання виконано не повністю. Праве вікно програми, ще заповнено невикористаними програмними компонентами.

Для ілюстративного прикладу взята задача розрізки речення на окремі слова та запису їх в зворотному порядку. З рис. 4 видно, що проблема розрізки тексту передбачає в себе допоміжні задачі: внесення змін до початкових значень; багатократний пошук пробілів, вирізка окремих слів та їх конкатенація; пошук поточного пробілу, вирізка слова та його підклейка; вирізка поточного слова; підклейка нового слова; вивід результату.

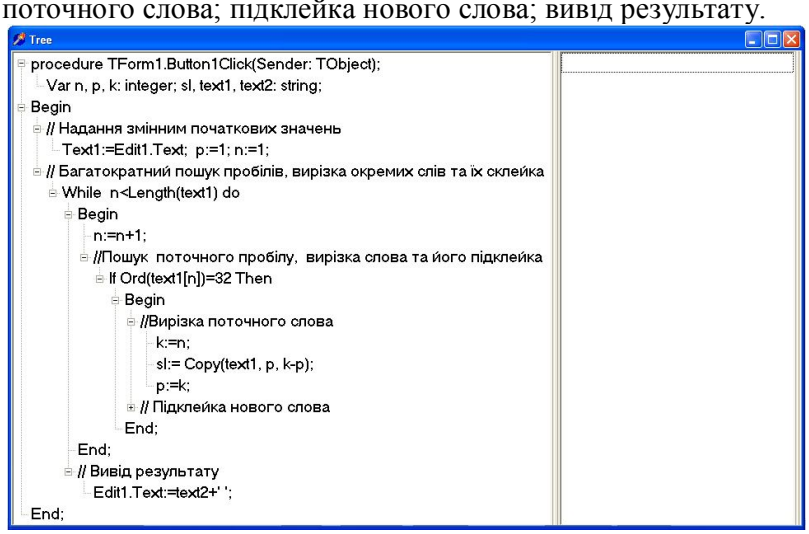

Рис.5. Інтерфейс начальної програми, що оперує завданням другого типу. Завдання виконано повністю. Всі елементи конструювання в лівому вікні. Вони включені в відповідну структуру. Праве вікно програми порожне

Змінними text1, text2 позначено відповідно початковий та кінцевий текст. Змінна N відслідковує номер поточної букви при пошуку пробілу в тексті. Змінними P і K позначені

відповідно положення пробілу на початку слова і в кінці його. Змінна Sl містить поточне вирізане слово.

При конструюванні інтерфейсу програми автори намагались зробити маніпуляції в ньому максимально швидкими. Редагування текстів у цьому інтерфейсі мінімізоване. Робота реалізована переважно через маніпулятор «мишка». Крім перенесень окремих компонентів між деревом і списком, а також всередині списку і дерева, реалізована можливість перенесення цілих віток в дереві.

Максимальне пришвидшення роботи - ключовий принциповий момент. Мінімізація, часу на матеріалізовані дії і обмеження часу на виконання завдання, приводить до ситуації, коли весь час витрачений на виконання завдання переважно заповнений інтелектуальними діями. Останнє дає перспективи на дослідження зміни швидкості виконання завдань в процесі навчання. Аналіз конструкції деревоподібної структури, що є результатом виконання завдання дозволяє досить просто визначати, який відсоток виконаного завдання, виконаний правильно. Дослідження поступових, плавних змін відсотка більш тонко описує процес навчання ніж результат перевірки, що передбачає два значення правильно і неправильно. На рис.5 представлена схема роботи програми.

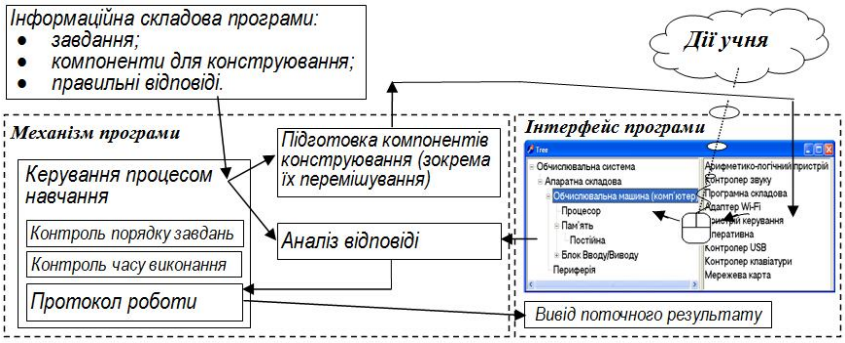

Рис.6. Механізм роботи програми в загальних рисах

Важливим моментом, який впливає на адекватну діагностику навчальних процесів при проведенні комп'ютеризованого навчання, є однорідність кожного з пакетів завдань, що використовується в навчанні.

Однорідність завдань у пакеті можна оцінити за двома факторами, а саме: за кількістю дій, необхідних для виконання кожного з завдань та за мірою складності окремих завдань у пакеті. Зрозуміло, що в однорідному пакеті кожне завдання повинно бути каліброване, як за складністю так і кількістю необхідних дій.

Розглянемо способи досягнення однорідності пакетів завдань для випадку, коли ці пакети укомплектовані саме завданнями на конструювання ієрархічної структури.

Калібровка кількість необхідних дій у завданнях на конструювання деревоподібної структури досягається досить просто. Кількість дій для завдань цього типу корелює з кількістю елементів конструювання. Для досягнення однорідності за кількістю дій достатньо запроектувати ці завдання так, щоб кількість елементів конструювання в різних завданнях була однакова. Довільно вибирати потрібну кількість дій можливо із-за того, що базисна для формування завдання ієрархічна структура, як правило, значно більша, ніж потрібно, для створення одного завдання і вона включає, елементи різної степені узагальненості. Ієрархічні структури добре масштабуються, тому при проектуванні завдань на конструювання гілки цих деревоподібних утворень можна згортати і розгортати на свій розсуд, не втрачаючи змістовної цілісності, але регулюючи кількість елементів у структурі. Інший цінний момент полягає в тому, що окрема гілка дерева є теж ієрархія. Тому в якості базису для завдання можна вибрати не деталізацію кореня всього дерева, а деталізацію окремої гілки. Це дозволяє досягнути найвищої степені деталізації та конкретизації матеріалу, не втрачаючи при цьому однорідність завдання за кількістю необхідних дій для виконання завдання. Великий об'єм базової для завдань ієрархії дозволяє утворювати десятки завдань навколо однієї деревоподібної структури, масштабуючи її при формуванні завдання або

пересуваючись нею. Якщо інтерпретувати програму, як ієрархію останнє твердження теж дійсне. Тут, з одного боку, можна акцентуватись при формуванні завдання на виготовлені окремих процедур відповідних локальним віткам дерева. З другого боку, використання готових бібліотек процедур може дозволити пересунутись ближче до «стовбура» ієрархії базової для завдань програми.

Каліброва складності теж непогано формалізується для завдань на конструювання ієрархічних конструкцій. У програмах існують фрагменти тексту достатньо прості для сприйняття і створення. Це, зокрема, лінійні частини програми, порядок виконання окремих кроків яких є незмінним. Ускладненими можна вважати лінійні фрагменти, які сприймаються тільки сукупно, в певному комплекті складових частин. Окрема частина такого фрагменту не має змісту в сенсі загальної мети роботи програми. Повтори та розгалуження в тексті програми, а також, складні математичні вирази, ексклюзивні функції і процедури (підпрограм) користувача підвищують складність програми. Такі фрагменти програми жорстко задають межі уваги при їх обмірковуванні. Кожен з перерахованих вище фрагментів формує свою змістовну складову, а значить спричинює локалізацію уваги на собі, ініціює стосовно себе відповідні логічні дії, формує стосовно себе нову цілісну узагальнену понятійну одиницю або коротке концептуальне твердження. На рис.5 представлені такі цілісні концептуальні твердження. (Багатократний пошук пробілів, вирізка окремих слів та їх склейка. Пошук поточного пробілу, вирізка слова та його підклейка. Вирізка поточного слова.) Зрозуміло, що складність розумових висновків ускладнюється в тих випадках, коли структура має вкладення. Понятійна одиниця верхнього рівня не може бути отримана без о концептуальної одиниці нижнього рівня. Як видно з рис.5 ми не можемо отримати багатократну вирізку слів без однократної вирізки, а однократна вирізка не може бути реалізована без пошуку чергового пробілу.

У контексті приведеного вище будемо вважати, що міра вкладеності ієрархічних конструкції ментальних структур одна в одну є мірою складності. Структура показана на рис. 5 має міру складності (вкладеності) 3. За аналогією структура на рис. 3 має таку ж міру вкладеності. Детальніше це розглянуто в роботі [6].

Підводячи висновок, можна сказати, що створення навчальних програм, що мають механізм тестування на базі конструювання ієрархічних структур, перспективні для розвитку. Вони дозволяють забезпечити суб'єктів навчання якісними завданнями, що ексклюзивно підтримують розвиток здатностей до причинно-наслідкового і абстрактно-логічного мислення. Витрати часу на матеріалізовану діяльність при цьому можуть бути мінімізовані. Крім того, ці завдання направлені на зменшення дифузності мислення, вони підвищують диференціацію понятійного базису розумових дій. Важливим моментом є і те, що пакети цих завдань можна зробити однорідними. Останнє має особливе значення як для автоматизації процесу навчання, так і для досліджень в галузі психології та педагогіки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Холодная М.А. Психология интеллекта: парадоксы исследования. – СПб.: Питер, 2002. – 272с.
2. Дж. Андерсон Дж. Когнитивная психология. 5-е изд. – СПб.: Питер, 2002. – 496 с.
3. Головін М.Б. Зміст підготовки висококваліфікованого фахівця з інформаційних комп'ютерних технологій у контексті когнітивних процесів (на прикладі програмування) //Інформаційні технології в освіті. Випуск 2. Херсон, 2008. – С. 66-73.
4. Дал У., Дейкстра Э., Хоар К. Структурное программирование. – М.: Мир, 1975. – 246 с.
5. Хьюз Дж., Митчом Дж. Структурный подход к программированию. – М.: Мир, 1980. – 276 с.
5. Головін М.Б. Кількість і складність розумових дій у контексті діагностики когнітивних процесів, що детермінують практику навчального програмування //Вісник черкаського університету. Серія педагогічні науки. Випуск 125. – Черкаси, 2008. – С. 34 – 41.