MVC — архітектурний шаблон, який використовується під час проектування та розробки програмного забезпечення.

Цей шаблон передбачає поділ системи на три взаємопов'язані частини: модель даних, вигляд (інтерфейс користувача) та модуль керування. Застосовується для відокремлення даних (моделі) від інтерфейсу користувача (вигляду) так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

Мета шаблону — гнучкий дизайн програмного забезпечення, який повинен полегшувати подальші зміни чи розширення програм, а також надавати можливість повторного використання окремих компонентів програми. Крім того використання цього шаблону у великих системах сприяє впорядкованості їхньої структури і робить їх більш зрозумілими за рахунок зменшення складності.

1. Електронний ресурс. – Режим доступу: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C-D0%B2%D0%B8%D0%B4-%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D1%80
2. Електронний ресурс. – Режим доступу: http://ukrbukva.net/page,7,92896-Sozdanie-mobil-nogo-prilozheniya-kolledzha-Ugresha-dlya-operacionnoiy-sistemy-iOS.html. 3. Електронний ресурс. – Режим доступу:http://science.donntu.edu.ua/ks/rantyuk/diss/indexu.htm.

Зербіно Д.Д.
Національний університет «Львівська політехніка»

ІНВЕСТИЦІЙНА ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ГЕНЕРУВАННЯ ІДЕЙ

Генерування ідей на перший погляд здається прерогативою живої істоти, але чим сильніше ми заглиблюємося в зміст обробки інформації, тим більше переконуємося в існуванні формальних правил підстановки одних даних іншими, які лежать в основі будь-якої обробки. Відтепер, настав час з’ясувати, чи можна звичайними операціями підстановки даних реалізувати фантазію, здогадку, припущення і якщо можна, то як це зробити?

Для цього необхідно порівняти, чим відрізняється звичайна обробка інформації від фантазії, що основана на створенні нової ідеї, яка може здаватися дивною з погляду повсякденного життя [1]. Формально кажучи, комп’ютерна система і зараз від реального життя достатньо віддалена. Тому оцінити, «дивна» ідея чи «не дивна» для неї досить важко. Насправді, комп’ютерна система може не лише підставляти одні дані в інші, але і перебирати варіанти, що і лежить в основі генерування ідей.

Основна проблема, яку необхідно розв’язати в цьому напрямку – це вибирати такі варіанти ідей, які заздалегідь не є абсурдними. Абсурдність – це логічна перешкода, яку неможливо подолати будь-якими припущеннями. Якщо можливо зробити деяке припущення, яке може «врятувати» ідею, то така ідея не є абсурдною, і її треба розвивати далі.

Розвиток ідей засобами комп’ютерного перебору та аналізу – це основний зміст даного проекту. Для його реалізації необхідно формально описати у вигляді тверджень ту галузь знань, для якої генеруються ідеї. Твердження містять змінні та їх властивості. Властивість може бути сформульована для одної змінної, двох, трьох або чотирьох змінних. Якщо змінними позначати {X,Y,Z,P}, то можуть бути введені наступні властивості: «об’єкт (X)», «(X) має особливість (Y)», «(X) впливає на (Y) способом (Z)», «(X) відноситься до (Y) як (Z) до (P)», «(X) виконує (Y) в ситуації (Z) для здійснення (P)».

Змінні можуть приймати довільні числові значення, які означають ідентифікатори об'єктів, особливостей, способів, операцій, ситуацій, тощо. Генерування ідеї на такому рівні означає деяке припущення, що «(X) може мати особливість (Y)», «(X) може вплинути на (Y) способом (Z)», «(X) може відноситися до (Y) як (Z) до (P)», «(X) може виконати (Y) в ситуації (Z) для здійснення (P)». Припущення відрізняється від факту тим, що факт реально існує, а припущення лише теоретично може існувати, хоча мають однакову структуру. Для того, щоб здійснити припущення, необхідно зробити реальними інші припущення, які логічно залежать від заданого припущення, яке на даний момент розглядається.

Для того, щоб генеровані ідеї були корисними, необхідно починати з мети. Згідно принципу функціонально-вартісного аналізу, необхідно не просто аналізувати шаблон «(X) може виконати (Y) в ситуації (Z) для здійснення (P)», але і розраховувати вартість такого припущення, тобто, від умовного виграшу відняти умовні витрати. Для цього необхідно ввести додаткові твердження типу «для (X) наближена вартість є (W)».

Такі неконкретні міркування стають доволі конкретними, якщо використовувати мову програмування PROLOG. В сучасних версіях цієї мови реалізована система припущень за допомогою оператора «SUPPOSE». Суть цього оператора полягає у тому, що ви можете тимчасово додати деякі данні до деякої оголошеної бази даних, або тимчасово їх видалити. Основна відмінність PROLOG-операторів від звичайних мов програмування полягає у тому, що будь-який оператор можна відмінити, тобто, вернути програму на той стан, коли оператор не був застосований, а ж до початку виконання програми. Це ж стосується і роботи з базою даних, тобто, при відміні оператора «SUPPOSE» тимчасово видалені або виправлені дані вертаються назад. Тому алгоритми, що написані з врахуванням такого «бектрекінгу» є по-перше безпечними (програма може дуже довго перебирати варіанти, але ніколи не зависне), а по-друге, алгоритми стають простішими, тобто, формулюються у вигляді тверджень.

1. SCRUM Product Ownership Balancing Value from the Inside Out [Електронний ресурс]. – Режим доступу: http://synchronit.com/downloads/Scrum-PO-From-the-Inside-Out-FINAL-Version2-and-6x9.pdf).