DOI: 10.20535/kpi-sn.2020.1.197953
УДК 004.415.26

І.В. Редько*, П.О. Яганов
КПІ ім. Ігоря Сікорського, Київ, Україна
*corresponding author: redkoigor@ukr.net

КОНЦЕПТУАЛЬНА МОДЕЛЬ ТЕХНОЛОГІЧНОГО СЕРЕДОВИЩА ПРОГРАМУВАННЯ

Проблематика. Зміст розглядуваної концептуальної моделі розкритий через поняття "концепт", "монада" та засадниче для технологічного середовища програмування концептомонадне взаємодоповнення. Концептомонадне середовище є платформою, що реально імплементує розуміння програмування як суб'єкт-об'єктну продуктивну діяльність – взаємодоповнення процесу програмотворення і його наслідку в їх причинно-наслідковому зв'язку з домінуванням саме процесу. Це закладає основу для технологізації галузі та створення адаптивних і гнучких суб'єктоорієнтованих систем програмування. Можливості концептомонадної платформи продемонстровано на прикладі однієї з можливих її імплементацій – редукційної технології програмування.

Мета дослідження. Подальший розвиток концептомонадних засад суб'єкт-об'єктного технологічного середовища, націлений на створення адаптивних і гнучких систем програмування.

Методика реалізації. Методи введення, виключення абстракції та прагматико-обумовленої типізації, метод концептомонадної релятивізації, метод редукцій, метод програмних алгебр.

Результати дослідження. Розвинуто понятійну систему концептомонадного технологічного середовища та створено на цій основі продуктивну модель редукційного середовища програмування. Запропоновано концептомонадну модель відкрито-замкненого середовища програмування. Вперше редукційні моделі програм та редукційні методи програмування розглянуті як прагматико-обумовлена конкретизація цього середовища. Показано, що в рамках отриманої концептомонадної системи коректно ставляться та вирішуються засадничі для сучасного програмування задачі та проблеми.

Висновки. Будь-яка сучасна інформаційно-технологічна система повинна не стільки бути орієнтована на нотацію одержуваних рішень, скільки підтримувати (забезпечувати) продуктивну діяльність суб'єкта для їх отримання. Ефективна розробка таких систем можлива в рамках концептомонадного середовища, що володіє розвиненими засобами адаптації, орієнтованими на врахування активної ролі суб'єкта програмотворення.

Ключові слова: концепт; монада; композит; редукція; оракул.

Вступ

При аналізі стану справ у програмуванні прийнято посилатись на значні досягнення практичного програмотворення. Як докази наводять численні факти успішного використання програмних продуктів у пристроях і системах різноманітного призначення. Головним критерієм продуктивності тут традиційно вважається програма, що на змістовому рівні пов'язується з діяльністю програміста з її створення. Таким чином, вона набуває всіх ознак його авторського виробу, в якому той реалізує суб'єктивні уявлення про способи досягнення поставленої мети, керуючись власним досвідом спроб і помилок, інтуїтивними евристиками, запозиченими знахідками і т.п., що знаходить своє відображення у змісті терміна "мистецтво програмування" [1].

Проте поряд із бурхливим розвитком і примноженням часто дуже показових успіхів у програмістській діяльності їх загалом екстенсивна природа неминуче сприяє накопиченню принципових проблем, що чимдалі дужче проявляються і трактуються як кризові прояви. Такий стан справ, звісно, притаманний не тільки програмуванню, а й усім, особливо технічним, галузям на етапах їх розвитку. Однак і в програмуванні, і в інформатико-технологічній діяльності (ІТД) він проявляється найбільш контрастно. Пов'язано це насамперед із безпрецедентним проникненням інформаційних технологій у всі галузі діяльності сучасного соціуму, що спричиняє залучення до цієї сфери величезних матеріальних і нематеріальних ресурсів. Природно, що при цьому ціна будь-якої прихованої, випадкової помилки, на яку ще вчора можна було не звертати особливої уваги, сьогодні неймовірно зростає. Ризик появи техногенної аварії внаслідок розтиражованого мільйонами копій програмного продукту без можливості для користувача проконтролювати й оцінити усі етапи його створення стає неприйнятним. Тому в сучасному технологізованому світі будь-яка продуктивна

діяльність повинна спиратися не на "осяяння" окремих виконавців, а на чіткі процедури пошуку рішень, які окреслені строгими структурами і є знаряддями досягнення гарантованого результату необхідної якості.

Суперечливість окресленої ситуації зводиться до класичного діалектичного протиріччя між продуктивним процесом і продуктом, коли надмірна індивідна суб'єктність програмістської діяльності не дає змоги об'єктивізувати та дослідити причинно-наслідкові зв'язки всередині процесу програмування як реальності, невіддільної від його наслідку. Тому для подолання протиріччя доцільною видається принципова зміна парадигм. Традиційна, індивідно-суб'єктивна парадигма, що асоціює програмування виключно з програмою, повинна бути замінена на інтерсуб'єктивну парадигму [2], що розглядає програмування як діяльність, націлену на створення програми (плану) цієї діяльності.

Таке трактування програмування природно позиціонувати як інтерсуб'єктивний об'єкт дослідження. Залучення його осучаснює розуміння програмування, адекватно збагачуючи його. Структури таких програм, планів, їх передумови, очікувані наслідки є суб'єктивними. Зокрема, поведінкові особливості причинно-наслідкових зв'язків, що їх індукують, способи, методи їх створення, засоби специфікації – все це становить предмет будь-якого інтерсуб'єктивного дослідження. Значимість обраних пріоритетів у тому, що саме через них забезпечується найголовніше – можливість розгляду програмування в контексті взаємодоповнення як процесу програмотворення, так і його наслідку в їх причинно-наслідковому зв'язку з домінуванням саме процесу. Це дає змогу відійти від "мистецтва програмування" і закласти основи його технологізації, коли надбання кожного суб'єкта програмування будуть реально досяжними для інших, взаємодоповнюючи фактографію галузі з її фактологією.

Постановка задачі

Метою роботи є подальший розвиток концептомонадних засад адаптивного технологічного середовища програмування на основі інтерсуб'єктивної парадигми.

Результати дослідження

Для досягнення поставленої мети необхідно предметно збагатити поняття концептування як суб'єкто-об'єктну взаємодію у програмотворенні, де активна роль суб'єкта програмування є визначальною. Це збагачення здійснюється на основі понять концепту, монади, обумовлення. Наступні кроки спрямовані на схематизацію концептування оракулами та композиціями як каркасу процесу, що відбувається в адаптивному технологічному середовищі програмування. Апробація інтерсуб'єктивної парадигми програмування буде продемонстрована на прикладі строгої специфікації важливого виду систем програмування – редукційних систем.

Концептомонадна парадигма програмування. Спонукальним мотивом будь-яких парадигмальних побудов є об'єктивна суперечливість між явищем і його розумінням. Нова парадигма завжди є компромісом – прийнятним узгодженням згаданої суперечності. Життєздатність такого узгодження забезпечується здатністю його до розвитку і саморозвитку, зважаючи на природну еволюційність розуміння. Тому зміна парадигми є об'єктивною необхідністю, яка виникає внаслідок нового розуміння процесу програмування.

Ґрунтовний аналіз різних парадигм, які пройшли серйозну перевірку часом, проведений у [3, 4], довів корисність їх інтеграційного розгляду, що дає змогу не стільки протиставляти різні точки зору, скільки акцентувати увагу на тому, що є спільним для них.

Основою такої об'єднавчої позиції є те, що будь-яку суб'єкто-об'єктну взаємодію слід сприймати як здійснюване суб'єктом обумовлення (зв'язування умовою) об'єкта [3]. Будь-які значущі парадигми розуміння суб'єкто-об'єктної взаємодії розглядаються як види роду обумовлення, а суб'єкто-об'єктне взаємодоповнення є індукований суб'єктом активно-пасивний зв'язок предмета впливу з наслідком цього впливу. Таке трактування дає можливість представити будь-яке програмотворення як суб'єкто-об'єктну відкрито-замкнуту систему (ВЗС) [3, 5] з домінуванням у ній саме активної ролі суб'єкта. Взаємодоповнення активної та пасивної форм програмування є ключовою характеристичною особливістю породжуваного ним причинно-наслідкового, а в разі обумовлення – сутесутнісного відношення (ССВ) типу $\langle$*сутність*, *суть*$\rangle$ [3, 5].

Найважливішою характерною рисою програмістської діяльності є націленість на створення її програми, плану. Панівна індивідно-суб'єктивна парадигма програмування не враховує актив-

ної ролі того, хто власне здійснює цю діяльність. Тому слід осучаснити цю парадигму залученням до розгляду діяльності вищого порядку – планування, розглядаючи програмування як діяльність, обумовлену планом (програмою). Таку програму (план) будемо називати концептом, а діяльність обумовлення, здійснювану з її допомогою, – концептуванням. Концепт дає можливість розглядати наслідок програмування як суть, що обумовлює сутність. А сутність, що обумовлюється концептом, будемо називати монадою. Сказане можна підсумувати такою активно-пасивною дефініцією:

$$\begin{cases} \textit{Концепт} \overset{def}{=} \textit{суть, що обумовлює сутність}, \\ \textit{Монада} \overset{def}{=} \textit{сутність, що обумовлюється концептом}, \end{cases}$$

де $\overset{def}{=}$ розуміється як "дорівнює за визначенням".

Ця дефініція суттєво розширює зміст ІТД, оскільки вводить важливі види сутностей – концепти, монади, обумовлення. Їх особливість полягає в тому, що вони є носіями інтенсіональних (якісних) суб'єктивних обумовлень, які, використовуючи термінологію Б.А. Трахтенброта [7, 8], будемо називати оракулами. Отже, наведена вище дефініція є оракульною ВЗС, що презентує найбільш загальну схему ІТД і міститься в основі усіх інших відомих схем або парадигм ІТД.

Для конкретизації інтерсуб'єктивної парадигми оракульною парадигмою на виконання мети нашої роботи необхідно її збагатити через дослідження згаданих вище оракулів як оракульних відкрито-замкнутих взаємодій, залучивши до розгляду оракули вищих типів – оракульні структури або схеми. Це дасть змогу використати можливості, зокрема, традиційного математичного апарату для нотації результату, а також інтегрувати його з методами, що розвиваються в рамках денотативних підходів. Для цього слід розглянути концептування як універсальний інструмент розв'язання задач ІТД, конкретизувавши цей універсалізм розглядом його суб'єктно-об'єктної природи.

Оракульна природа концептування. Весь досвід, накопичений людиною у пошуках розв'язку задач, свідчить про те, що, незважаючи на різноманіття методів досягнення мети, реально вони зводяться до парадигми "розділяй і володарюй". Тому цілком виправдано, що подальше збагачення як власне концептування, так і властивих йому оракулів буде проводитися в контексті цієї парадигми ІТД.

Суб'єктно-об'єктна природа концептування тісно пов'язана з активним і пасивним видами (формами) обумовлень. Активна форма є діяльністю суб'єкта концептування і тому сама є концептуванням. Вона проявляється безпосередньо у використанні для досягнення мети активних видів обумовлень. Пасивна форма пов'язана з наслідком концептування, а активна роль суб'єкта в ній реалізується через концепт. Суб'єкт концептування безпосередньо формує концепт (програму, план) розв'язання задачі та опосередковано впливає на наслідок цього концептування.

Концептування є схемою взаємодії оракулів: обумовлення, концепту, монади, сутності, суті. За допомогою цих оракулів суб'єкт здійснює свою активну роль у концептуванні, актуалізуючи всі або деякі з них і тим самим конкретизуючи вихідну схему "розділяй і володарюй". Оракул "монада" відіграє залежну роль, презентуючи собою наслідок, обумовлений тим, як суб'єкт реалізував свою активну роль.

Конкретизація цієї схеми взаємодії оракулів призводить до конкретизації породжуваного концептуванням ССВ. З точки зору прагматики ІТД, найбільш цікавими є конкретизації, пов'язані з об'єктивізацією концепту. Вони зводять вихідне ССВ до відношення особливого виду – функціональної монадосутнісної залежності. Це зведення є істотно інтенсіональним, оскільки спирається на відповідний інтенсіонал, яким є актуалізований концепт. Екстенсіональне ж зведення такої функціональної залежності у вигляді заданого об'єму всіх монадосутнісних зв'язків трактується в контексті її обумовленості інтенсіоналом. Обидва зведення відображають одну й ту саму сутність, але під різними кутами зору. Підтвердимо цей висновок наочним прикладом.

При позначенні операцій і предикатів використовують як операторну, так і термальну форми запису. Так, записи

$$S^2(\times, S^2(+, I_1^3, I_2^3), I_3^3)_{|N^3 \to N} \text{ і } (x+y) \times z_{|x,y,z \in N}$$

позначають одну й ту ж саму арифметичну операцію [9]. Перша, операційна форма, відображає генезис цієї операції з простіших, обумовлений інтенсіоналом – оператором суперпозиції. Друга, термальна, описує цю ж арифметичну операцію екстенсіонально як функцію через зазначення об'єму властивих їй арифметич-

них дій. Форми цих текстів, у певних сенсах, взаємозамінні, але у першій занотована абсолютно природна домінанта генезису відносно його результату, друга ж є описом деякої сутності – актуально заданої множини дій, про генезис якої можна говорити лише опосередковано і частіше за все недетерміновано як про "функцію об'єму дій".

Отже, для здійснення ІТД суб'єкт свідомо чи підсвідомо об'єктивізує концепт схемою взаємодії оракулів, підтверджуючи оракульну природу концептування. Свідомо (інтенсіонально, змістовно) – якщо володіє методами концептування ІТД, підсвідомо (екстенсіонально, об'ємно) – якщо ні.

Об'ємна точка зору на ІТД та програмування достатньо ґрунтовно вивчена і становить основу індивідно-суб'єктивного його розуміння. Репрезентативними прикладами тут є відомі об'ємні парадигми програмування: функціональна – орієнтована на об'єм функції [10, 11], об'єктно-орієнтована – на об'єм об'єкта [12], модульна – на об'єм модуля [13, 14] і т.д. У нашій роботі зосередимо увагу на розгляді інтенсіональних засад концептування як інтеграційного середовища будь-яких, як існуючих, так і тих, що можуть бути побудовані, об'ємних і змістовних парадигм програмування.

Оракульні структури інтенсіонально збагачують концептування, формують точку зору на монаду, зокрема на програму, як на структурну сутність, а на концепт – як на відповідну структуру. Однак таке розуміння концептування все ще залишається надто загальним, а значить, недостатньо змістовним. Адже залежно від цілей до розгляду можуть залучатись найрізноманітніші структури. Необхідне подальше прагматико-обумовлене збагачення концептування через виділення серед усього розмаїття структур таких, що складають своєрідний "спільний знаменник", стосовно якого всі інші структури природно розглядати як похідні від нього.

Програми, як і монади в цілому, характеризуються перед усім генезисом. Як випливає з самої значущості генезису, він визначає основну парадигму концептування, зокрема програмування. Тому як згаданий "загальний знаменник" концептування логічно вибрати саме генетичні структури [15]. Серед усього різноманіття таких структур особливе місце посідають композиційні структури.

Композиційне збагачення концептування. Інтенсіонально-екстенсіональна двоєдиність концептування обумовлена значенням активної ролі суб'єкта у програмуванні та її місцем у парадигмі "розділяй і володарюй". Кожен крок активності суб'єкта в будь-якому конкретному програмотворенні націлений на необхідну, з точки зору суб'єкта, деталізацію розглядуваної сутності, а активність же в цілому має на меті достатню, з точки зору суб'єкта, її деталізацію. При цьому оракул "активна роль суб'єкта" в кожному конкретному концептуванні може бути наповнений, залежно від суб'єкта, інтенсіональним або екстенсіональним змістом. Інтенсіональне трактування оракула підтримує взаємодоповнення покрокового процесу деталізації та його результату, екстенсіональне – нотацію результату. У контексті згаданої парадигми в першому випадку активність суб'єкта обумовлена необхідністю "розділяти", тобто дивитися на сутність як на структурну, складену сутність – підґрунтя для обумовленого таким розділенням її розуміння та його нотації. У другому – виключно можливістю "панувати", розглядаючи сутність із точки зору її актуального, заздалегідь заданого об'єму.

Забезпечити інтенсіональність побудов у рамках інтерсуб'єктивної парадигми програмування означає розглянути складену сутність як наслідок її генезису, в цьому випадку – концептування, концептом якого виступає згадувана вище генетична структура – композиція. Особливістю цих структур є те, що вони не можуть бути задані заздалегідь, актуально та вичерпно. Вони є наслідками суб'єктивних обумовлень і носіями суб'єктивних розумінь розглядуваних сутностей. Адже композиції як концепти є носіями інтенсіоналів – актуалізованими змістами оракула "активна роль суб'єкта". Такі структури можуть бути як завгодно складно влаштовані. Тому серед усіх генетичних структур особливе місце займають базові засоби генезису – генні структури або композити [3, 4], на тлі яких інші композиції породжуються як похідні покроковості таких базових структур. Таким чином, концептування конкретизується як покроковість активностей суб'єкта, концептом якої є композиція. При цьому кожен крок покроковості теж є концептуванням, концепт якого презентується деяким композитом, а відповідна монада зводиться до представлення досліджуваної сутності як складеної сутності виду композиту сутностей-складових.

Сказане дає змогу прагматико-обумовлено збагатити об'єм (рід) оракула "композит", зокрема виділити два основних види цього роду. До першого належать композити, що підтримують

розуміння композиції як покроковості композитів. До другого – значущі для суб'єкта базисні композити, що формують носій його активної ролі в концептуванні. При цьому концептування обумовлене розумінням згаданої вище покроковості як концепту композиції.

Сказане надає досить багате змістовне уявлення про концептування як покроковість генних (композитних) обумовлень. Видається доцільним здійснити його адекватну схематизацію.

Схематизація концептування. Уведемо низку важливих для подальшого викладення позначень та визначень. Найбільш загальним оракулом концептування є "сутність", що позначатиметься надалі *Ent*. Серед усього розмаїття сутностей, які складають носій *Ent*, виділяються оракули, що презентують наслідки обумовлень, які коротко будемо називати "суть" і позначати *Gst*, "концепт" *Con* і "монада" *Mon*. Активну та пасивну форми обумовлення – "обумовлює" та "обумовлюється" – позначатимемо $\succ$ та $\prec$ відповідно. Тобто запис $Ent_1 \succ Ent_2$ означає, що сутність Ent_1 обумовлює сутність Ent_2, а запис $Ent_1 \prec Ent_2$ – що сутність Ent_1 обумовлюється сутністю Ent_2. Тоді концептування може бути представлене такою схемою, яка становить загальнозначущу основу інтерсуб'єктивної парадигми ІТД – відкрито-замкнуту систему оракульних обумовлень:

$$\begin{cases} Con \overset{def}{=} Gst \succ Ent, \\ Mon \overset{def}{=} Ent \prec Con. \end{cases}$$

Тут замкнутість забезпечується активно-пасивною природою концептування, а (суб'єктивна) відкритість – оракулами Ent, Gst, Con, Mon, $\succ$ і $\prec$ – "реперними точками" реалізації активної ролі суб'єкта в концептуванні. Резервом наведеної схеми є властива їй недостатня змістовність як зворотний бік її загальності, зокрема з надто широким трактуванням оракулів. Адже активність суб'єкта є концептуванням, концептом якого є вже не просто композиція, а композит – базова (для суб'єкта) генна структура. З урахуванням цього приходимо до обумовленої конкретизації як властивих концептуванню оракулів, так і самої оракульної схеми:

$$\begin{cases} Comt \overset{def}{=} Gst \overset{B}{\succ} Ent, \\ CtMon \overset{def}{=} Ent \prec Comt, \end{cases}$$

де $Comt$ – оракул "композит", $\overset{B}{\succ}$ – оракул "базисно обумовлює".

"Базисно обумовлює" означає, що обумовлює "за один крок", природа якого не вимагає від суб'єкта додаткової деталізації, а $CtMon$ – збагачення оракула Mon за рахунок актуалізації концепту композитом. Звідси можемо конкретизувати вже і саму схему концептування:

$$\begin{cases} Comp \overset{def}{=} Gst \overset{G}{\succ} Ent, \\ CMon \overset{def}{=} Ent \prec Comt, \end{cases}$$

де $Comp$ – оракул "композиція", $\overset{G}{\succ}$ – оракул "генезисно обумовлює", тобто обумовлює покроковістю (у зазначеному вище сенсі), а $CMon$ – збагачення оракула Mon за рахунок актуалізації концепту композицією.

Наведені схеми є концептами для різних композитних і композиційних концептувань. Останні отримуються за рахунок актуалізації оракулів, що входять до схем, з урахуванням того, що оракули $CtMon$, $CMon$, $\overset{B}{\succ}$, $\overset{G}{\succ}$ і їх носії залежать від актуалізації оракулів Ent, $Comt$ і $Comp$. Зокрема, залежності вигляду $Ent \overset{Comt}{\to} CtMon$ або $Ent \overset{Comp}{\to} CMon$ є функціональними. А значить, і композитні, і композиційні монади можуть бути експліковані як композиції сутностей:

$$CtMon \Rightarrow Comt(Ent_{i_1}, ..., Ent_{i_k})$$

і

$$CMon \Rightarrow Comp(Ent_{j_1}, ..., Ent_{j_p}),$$

де Ent_{i_1}, ..., Ent_{i_k}, Ent_{j_1}, ..., Ent_{j_k} – деякі сутності, зокрема, можливо, і оракули, а $\Rightarrow$ означає "експлікативно зводиться". Розглядаючи ці експлікативні зведення в контексті концептування і проектуючи їх на відповідні функціональні залежності, отримуємо такі оракульні схеми:

$$Ent \overset{Comt}{\to} Comt(Ent_{i_1}, ..., Ent_{i_k})$$

і

$$Ent \overset{Comp}{\to} Comp(Ent_{j_1}, ..., Ent_{j_p}).$$

Ці схеми дуже схожі за зображенням, але смисли їх принципово різні. Перша використовує базову композицію, але при цьому на сутності $Ent_{i_1}, ..., Ent_{i_k}$ ніяких додаткових обмежень не

накладається. У другій же, навпаки, композиція є похідною від композитів, зате сутності $Ent_{j_1},...,Ent_{j_k}$ є елементарними, тобто достатньо деталізованими з точки зору суб'єкта. $Ent \overset{Comt}{\to} Comt(Ent_{i_1},...,Ent_{i_k})$ є концептом реалізації активної ролі суб'єкта в будь-якому кроці концептування, $Ent \overset{Comp}{\to} Comp(Ent_{j_1},...,Ent_{j_p})$ являє собою рефлексивно-транзитивне замикання породжуваної концептуванням функціональної залежності. Тобто ця схема відображає наслідок концептування як похідної від композитів композиції елементарних, з точки зору суб'єкта, сутностей.

У наведених схемах відображено взаємодоповнення двох полюсів концептування – синтезу й аналізу, композиції та декомпозиції. Синтез представлений у них оракулами *Comt* і *Comp*, а аналіз (декомпозиція) – $Ent_{i_1},...,Ent_{i_k}$ і $Ent_{j_1},...,Ent_{j_k}$ відповідно. Зв'язок стадій композиції та декомпозиції в концептуванні добре представлений схемою $Ent \overset{Comt}{\to} Comt(Ent_{i_1},...,Ent_{i_k})$, у якій використано саме композит – генну структуру, що дає можливість здійснити відповідний аналіз (декомпозицію) сутності *Ent* "в один крок". Будемо казати, що кортеж $\langle Ent_{i_1},....,Ent_{i_k}\rangle$ є *Comt*-редукцією сутності *Ent*, якщо існує функціональна залежність $Ent \overset{Comt}{\to} Comt(Ent_{i_1},...,Ent_{i_k})$.

Зі сказаного випливає, що редукція природним чином імплементує парадигму "розділяй і володарюй" у розумінні активної ролі суб'єкта в концептуванні, підтримуючи реальне взаємодоповнення декомпозиційної та композитної складових концептування [3, 4, 6]. Таким чином, редукція реально, а не номінально технологізує концептування, експлікативно зводячи його до редукційного концептування.

Підтвердимо цей висновок простим репрезентативним прикладом.

Редукційне збагачення концептування. Проведені збагачення обґрунтовують точку зору на концептування як на покроковість редукувань. При цьому редукування зводиться до пошуку відповідної *Comt*-редукції, тобто до розв'язання індукованого відповідною функціональною залежністю рівняння $Ent = Comt(E_{i_1},...,E_{i_k})$. Під розв'язкам тут розуміється кортеж $\langle Ent_{i_1},...,Ent_{i_k}\rangle$ такий, що справедливою є тотожність

$$Ent \equiv Ap(Comt,\langle Ent_{i_1},...,Ent_{i_k}\rangle),$$

де *Ap* розуміється традиційно як аплікація [16].

Розв'язання задачі, як відомо, суть інтеграція розв'язків її підзадач. Якщо задача "проста", то інтеграція тривіальна і, як правило, явно не виділяється. У разі ж коли задача є "складною", інтеграційний аспект її розв'язання стає все більш домінуючим. Редукційне концептування реально враховує цю притаманну реальним задачам домінантність, акцентуючи увагу на процесі покрокового породження суб'єктом інтеграційної складової розв'язання таких задач. Продемонструємо це на прикладі редукційного концептування репрезентативного класу задач числового аналізу. Розглянемо задачу пошуку наближеного розв'язку рівнянь вигляду $x = \varphi(x)$, де функція $\varphi(x)|_{x\in R}$ задовольняє такі дві умови:

1) вона визначена і неперервно диференційовна на всій числовій прямій;

2) існує таке дійсне число $p < 1$, що для всіх x справедливо $|\varphi'(x)| \le p$.

Концептування проведемо, виходячи з того, що для таких рівнянь метод простих ітерацій збігається. Причому розв'язком є границя послідовності $\{x_i\}_{i=0,1,2,...}$, де x_0 – будь-яке дійсне число, а $x_{i+1} = \varphi(x_i), i = 0,1,2,...$

Як платформу концептування використаємо композиційне програмування та іменну модель даних, функцій і операцій, а як композити – операції мультиплікування $\circ$, розгалуження *IF*, циклування *WD* і найпростіші похідні від них (у сенсі операцій аплікації *Ap* та *n*-арної суперпозиції $S^n|_{n\in N}$) композиції, що уточнюють найбільш вживані способи синтезу одних програм з інших [3, 15, 17]. Надалі під даними, функціями та операціями, якщо не зазначено інше, розуміємо іменні дані, іменні функції та іменні операції відповідно. Як редукційний апарат, виходячи із зазначеного вище, цілком достатньо буде вибрати просту *WD*-редукцію. Така редукція являє собою кортеж $\langle g, p\rangle$, що є розв'язкам рівняння $f = WD(g, p)$, де g, p – деякі функція та предикат [15, 18]. Звідси безпосередньо випливає корисна необхідна умова *WD*-редукційності: $\langle g, p\rangle$ є *WD*-редукцією функції f, якщо справедлива рівність $g \circ f = f$.

Із зазначеного випливає, що концептування зводиться до деталізації функції

$$f : \{(v, x_0),(u,\varepsilon)\} \to \{(v, x_n)\},$$

де x_n – перший із членів послідовності наближень, для якого виконується умова $|x_{n-1} - x_n| < \varepsilon$. Безпосередньо перевіряється, що кортеж $\langle g, p \rangle$, де

$$g : \{(v_{pr}, a), (v, b)\} \to \{(v_{pr}, b), (v, \varphi(b))\},$$

а

$$p : \{(v_{pr}, a), (v, b), (u, \varepsilon)\} \to \begin{cases} True, |a - b| \geq \varepsilon, \\ False, |a - b| < \varepsilon, \end{cases}$$

є розв'язком рівняння $f = WD(E_1, E_2)$. Тобто $f \equiv WD(g, p)$.

Особливість цього розв'язку полягає в тому, що він є оракульною схемою розв'язання класу задач, або оракульною схемою монади. Оракулом тут виступає функція

$$\varphi : \{(v, b)\} \to \{(v, \varphi(b))\} \; -$$

іменна специфікація $\varphi(x)\big|_{x \in R}$. Схема перетворюється на конкретну монаду після заміни φ у схемі конкретною функцією, що задовольняє вказані вище дві умови і за потреби може бути редукована. Такими є, наприклад, функції

$$\varphi : \{(v, b)\} \to \left\{\left(v, \frac{\cos(b)}{2}\right)\right\},$$

$$\varphi : \{(v, b)\} \to \left\{\left(v, \frac{\sin(b) + \cos(b)}{3}\right)\right\}$$

тощо.

Приклади можна було б продовжити, але виходячи з прагматики роботи, думається, що все наведене вище вже достатньою мірою демонструє переваги інтерсуб'єктивної парадигми програмування та роль редукції як основного інструменту технологізації програмування.

Висновки

У роботі в рамках інтерсуб'єктивної парадигми обгрунтовано методологічні засади технологізації програмування, в якій активна роль суб'єкта програмування є визначальною. Програмування визначено як концептування – діяльність обумовлення, здійснювану на основі концепту, що дає можливість розглядати концептування як універсальний інструмент розв'язання інформаційно-технологічних задач. Досліджено оракульну природу концептування в контексті розуміння активної ролі суб'єкта у ньому. На цій підставі здійснено оракульну схематизацію концептування як платформи технологізації програмування. Концептування конкретизовано покроковістю композитних обумовлень, концептом якої є похідна генетична структура – композиція. Обґрунтовано, що експлікативне зведення концептування до редукціювання (редукційного концептування) є продуктивним саме у прагматиці технологізації програмування – побудові програмологічних засад програмування і розвитку на цьому фундаменті адаптивних інформатико-технологічних середовищ. Засадничою особливістю таких середовищ програмування є те, що вони дають змогу реально, а не лише номінально підтримувати домінанту програмування у причино-наслідковому зв'язку $\langle$*програмування*, *програма*$\rangle$. Це своєю чергою дає змогу адекватно ставити та вирішувати найбільш значимі для сучасного програмування проблеми управління якістю програм, ефективності їх розробки та збереження інвестицій.

Подальші дослідження проводитимуться у напрямі розвитку композиційних засад адаптивного середовища програмування та створення редукційних моделей низки високорівневих мов програмування.

References

[1] D.E. Knut, *The Art of Computer Programming. Fundamental Algorithms.* Moscow, Russia: Vyliams, 2006, 650 p.

[2] E. Husserl, *Logical Studies. Cartesian Meditations. Sciences and Transcendental Phenomenology. Philosophy and the Crisis of European Man Philosophy.* St. Petersburg, Russia: Nauka, 2006, 320 p.

[3] I.V. Redko *et al.*, *Conceptual-Logical Foundations of Design*. Kyiv, Ukraine: Comprint, 2016, 154 p.

[4] V. Redko, "The theory of descriptive environments and its application", Ph.D. dissertation, Dept. Elect., National Technical University of Ukraine "Kyiv Polytechnic Institute", Kyiv, Ukraine, 2008.

[5] V.N. Redko and I.V. Redko, "Descriptive systems: Perspectives and retrospectives", *Visnik Kiïvs'kogo Natsional'nogo Universitetu im. T. Shevchenka*, sp. issue phys.-math. sci., pp. 68–75, 2004.

[6] V.N. Redko *et al.*, "Conceptological foundations of essential platform", in *Proc. TAAPSD'2012*, Kyiv, Ukraine, Dec. 3–7, 2013.

[7] B.A. Trakhtenbrot, *Algorithms and Computational Mashines.* Moscow, SU: Sovetskoe Radio, 1974, 200 p.

[8] V.A. Ganov and V.R. Karymov. (2009). *Computer Simulation of Computations with Oracles. Presented at the EAGU 2009* [Online]. Available: http://izvestia.asu.ru/2009/1/info-comp/07.en.html

[9] A.I. Maltsev, *Algorithms and Recursive Functions.* Moscow, SU: Nauka, 1965, 391 p.

[10] J.W. Backus, "Algebra of functional programs: functional level thinking, linear equations and generalized definitions", in *Mathematical Logic in Programming.* Moscow, Russia: Mir, 1991, pp. 8–53.

[11] J. McCarthy. (1960). *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I. Presented at the Communication of the ACM 1960* [Online]. Available: https://aiplaybook.a16z.com/reference-material/mccarthy-1960.pdf

[12] *G.* Buch, *The Object-Oriented Analysis and Design*, 2nd ed. Moscow, Russia: Binom, 1998, 560 p.

[13] D.L. Parnas. (1972). *On the Criteria to be Used in Decomposing Systems into Modules. Presented at the Communication of the ACM* [Online]. Available: https://www.win.tue.nl/~wstomv/edu/2ip30/references/criteria_for_modularization.pdf

[14] N. Wirth, "A Personal computer for the software engineer", in *Proc. ICSE 81*, San Diego, March 9–12, 1981.

[15] V.N. Redko, "Foundations of programmology", *Kibernetika i Sistemnyj Analiz*, no. 1, pp. 35–57, 2000.

[16] H.P. Barendregt, *The Lambda Calculus. Its Syntax and Semantics.* Moscow, SU: Mir, 1985, 606 p.

[17] V.N. Redko, "Compositions of programs and composite programming", *Programmirovanie*, no. 5, pp. 3–24, 1978.

[18] D.I. Redko *et al.*, "Compositional basis in programmer activity", *Systemni Doslidzhennia ta Informatsiini Tekhnolohii*, no. 4, pp. 83–96, 2015.

И.В. Редько, П.А. Яганов

КОНЦЕПТУАЛЬНАЯ МОДЕЛЬ ТЕХНОЛОГИЧЕСКОЙ СРЕДЫ ПРОГРАММИРОВАНИЯ

Проблематика. Содержание рассматриваемой концептуальной модели раскрыто посредством понятий "концепт", "монада" и основополагающего для технологической среды программирования концептомонадного взаимодополнения. Концептомонадная среда является платформой, которая реально имплементирует понимание программирования как субъект-объектную продуктивную деятельность – взаимодополнение процесса программостроения и его последствия в их причинно-следственной связи с доминированием именно процесса. Это закладывает основу для технологизации отрасли и создания адаптивных и гибких субъекториентированных систем программирования. Возможности концептомонадной платформы продемонстрированы на примере одной из возможных ее имплементаций – редукционной технологии программирования.

Цель исследования. Дальнейшее развитие концептомонадных основ субъект-объектной технологической среды, нацеленных на создание адаптивных и гибких систем программирования.

Методика реализации. Методы введения абстракции, исключения абстракции и прагматическо-обусловленной типизации, метод концептомонадной релятивизации, метод редукций, метод программных алгебр.

Результаты исследования. Развита понятийная система концептомонадной технологической среды, и создана на этой основе продуктивная модель редукционной среды программирования. Предложена концептомонадная модель открыто-замкнутой среды программирования. Впервые редукционные модели программ и редукционные методы программирования рассмотрены в качестве прагматико-обусловленной конкретизации этой среды. Показано, что в рамках полученной концептомонадной системы корректно ставятся и решаются важные для современного программирования задачи и проблемы.

Выводы. Любая современная информационно-технологическая система должна не столько быть ориентирована на нотацию получаемых решений, сколько поддерживать (обеспечивать) продуктивную деятельность субъекта для их получения. Эффективная разработка таких систем возможна в рамках концептомонадной среды, обладающей развитыми средствами адаптации, ориентированными на учет активной роли субъекта программостроения.

Ключевые слова: концепт; монада; композит; редукция; оракул.

I.V. Redko, P.O. Yahanov

CONCEPTUAL MODEL OF TECHNOLOGICAL ENVIRONMENT OF PROGRAMMING

Background. The content of the considered conceptual model is explained through the concepts of "concept", "monad", and concept-monadic complementation that is fundamental for the technological programming environment. The concept-monadic environment is a platform that really implements the understanding of programming as a subject-object productive activity – the complementarity of the programming process and its consequence in their cause-effect relation with the dominance of the actual process. It sets the stage for the technologization of programming industry and the creation of adaptive and flexible subject-oriented programming systems. The capabilities of the concept-monadic platform are exemplified by one of its possible implementations – reduction programming technology.

Objective. The purpose of the paper is further development of the concept-monadic foundations of the subject-object technological environment, aimed at creating adaptive and flexible programming systems.

Methods. Methods of introduction, exclusion of abstraction and pragmatic-driven typing, concept-monadic relativization method, reduction method, program algebra method.

Results. Based on the developed conceptual system of concept-monadic and technological environment productive model of reduction environment of programming is created. The concept-monadic model of open-closed programming environment is proposed. For the first time, reduction models of programs and reduction methods of programming are considered as pragmatically conditioned concretization of this environment. It is shown that within the framework of the received concept-monadic system, the tasks and problems that are fundamental for modern programming are correctly set and solved.

Conclusions. Any modern information-technology system should be oriented not so much to the notation of the decisions received, but to support (provide) the productive activity of the subject to obtain them. Effective development of such systems is possible within the framework of a concept-monadic environment that has advanced adaptation tools oriented to take into account the active role of the subject of programming.

Keywords: concept; monad; composite; reduction; oracle.

Рекомендована Радою
факультету прикладної математики
КПІ ім. Ігоря Сікорського

Надійшла до редакції
25 листопада 2019 року

Прийнята до публікації
04 лютого 2020 року