УДК 681.3.06

І. В. РЕДЬКО, О. О. ТАТАРІКОВ

Національний технічний університет України «Київський Політехнічний Інститут»

Д. І. РЕДЬКО

Київський національний університет ім. Т. Г. Шевченка

СИСТЕМИ ПРОГРАМУВАННЯ: ДЕСКРИПТИВНІ ЗАСАДИ

В рамках концепції дескриптивної системи (ДС) розглянуто поняття системи програмування (СП) та платформи програмування (ПП). Розроблено універсальний метод еволюційного збагачення сутностей, основу якого складає прагматико-обумовлена типізація сутностей (ТОП), а головним інструментом типізації виступає поняття дефінітної системи (ДФС). Метод ТОП використано для розбудови понятійної структури ПП. Зокрема, дано визначення понять декомпозиції та композиції як адекватних збагачень відповідних ДФС – ключових елементів розуміння понять програмування та програми.

***Ключові слова**: дескриптивна система, дефінітна система, платформа програмування, прагматико-обумовлена типізація, дескрипція, дефініція, монада, поліада, композиція, декомпозиція.*

У першому наближенні системи програмування це дескриптивні системи [1–3]. Тому природно, що СП наслідують головні засадничі властивості ДС. Як відомо, ДС – це система, яка підтримує дескриптування сутностей як процеси побудови їх *дескрипцій*, тобто прагматико-обумовлених описів, що розуміються в широкому сенсі, зокрема і як процес, і як результат процесу. Така орієнтованість дескриптування на прагматику є парадигмною властивістю в цілому для ДС, що виділяє їх серед інших систем пізнання сутностей. Зміст її на загальному рівні можна сформулювати наступним чином: «*сутність адекватно визначається сутностями, що обумовлюються нею*». Розкриття глибинної суті СП як ***головної мети*** даної статті, таким чином, за необхідністю пов'язане з відповідним предметним збагаченням цієї властивості, в першу чергу, відношення обумовленості, як ***об'єкта дослідження***, в контексті розуміння програмування як виду роду дескриптування. У відповідності до дескриптивного підходу основою СП є *платформа програмування* – носій рішень програмістських задач. Тому вона, як ***предмет дослідження*** за необхідністю повинна спиратись на відповідну понятійну систему, яка підтримує ключову властивість СП – згадане прагматико-обумовлене розуміння програмістської проблематики. Побудова такої понятійної системи в рамках ПП здійснюється за допомогою методу сутесутнісної релятивізації, основу якого складає поняття дефінітної системи [1–3].

Постановка завдання

Дана робота присвячена прагматико-обумовленій побудові концептуальних засад ПП. Основна увага тут буде приділена дефінітній природі монадних, поліадних, інтроадних, екстраадних, композиційних та декомпозиційних систем, як систем, що наповнюють реальним змістом розуміння програмування як процесу породження та застосування композицій. Всі використані та невизначені в роботі терміни та поняття трактуються в сенсі [1].

Результати та їх обговорення

Прагматичні передумови ПП.

Враховуючи фундаментальну значимість ПП для пізнання природи СП, її розгляд за необхідністю розуміється тут максимально широко, представляючи собою взаємодоповнююче використання як аналізу вже наявного заділу, так і, що більш важливо, еволюційну побудову власне платформи СП із урахуванням проведеного аналізу й максимально можливим прагматико-обумовленим використанням і розвитком його результатів. Останнє зауваження є важливим у світлі широко

обговорюваної проблеми збереження інвестицій. Таким чином, згадану побудову проведемо в декілька етапів, кожний з яких являє собою прагматико-обумовлену конкретизацію результатів попереднього.

Важливим для подальших побудов є те, що розуміння як ДС, так і СП та ПП завжди було й залишається релятивним. Це забезпечує постійний розвиток розуміння цих сутностей в напрямку відповідності сучасному рівню знань. Природа релятивності визначається в першу чергу прагматичною обумовленістю розглядів цих сутностей. Тому розуміння природи такої прагматичної обумовленості є ключовим для побудови ПП. В роботах [4–9] проведено ґрунтовний аналіз еволюційного розвитку уявлень як про системи програмування, так і про відповідні їм платформи. З нього випливає, що на всьому протязі еволюційного розвитку ПП чільне місце в розглядах займала проблематика адекватної сучасним вимогам підтримки взаємозв'язку денотативних (не процедурних, орієнтованих на властивості) та конотативних (процедурних, орієнтованих на процес) специфікацій задач. В якості репрезентативного прикладу тут можна навести проблему взаємозв'язку «функція - програма» в контексті традиційного розуміння ПП як універсальної мови програмування. Адже попри потенційну універсальність засобів програмо творення, між сутностями «функція» та «програма» дистанція величезного розміру. На дану проблематику вказував задовго до появи перших ЕОМ К. Поппер в [10] в контексті вивчення проблем верифікації та фальсифікації. Суть його точки зору, проектуючи її на ПП сьогодні можна виразити так: стосовно програми прагматико-обумовленою є проблема побудови та дослідження методів коректного її створення, а не проблема перевірки коректності створеної програми. В цій роботі будемо дотримуватись саме цієї точки зору. Стосовно ПП як носія рішень програмістських задач це означає трактування її як несуперечливої логічної абстракції цілісного різноманіття програм. Безпосередньо з даного визначення випливає, що тут носій несуперечливо будується за допомогою прагматико-обумовленого застосування засобів уведення та виключення абстракції, що складають основу методу сутесутнісної релятивізації [4, 8, 11]. У той час, як традиційна точка зору на ПП як на наперед заданий універсальний носій програм (універсам програм) разом з максимально можливою загальністю такого розуміння неминуче втягує в розгляди проблеми, пов'язані з можливою суперечливістю розглядів. Адже добре відомо, що традиційний універсум як *актуально побудований* не може бути несуперечливим. Більше того, ця суперечливість непереборна. Не рятують положення численні спроби «роздрібного» усунення викритих антиномій або парадоксів у конкретних дослідженнях. Вони, очевидно, є потенційним джерелом нових протиріч. Тому точка зору на ПП як несуперечливу логічну абстракцію цілісного різноманіття програм суттєво переважає традиційну в контексті порушеної проблематики. Акцентація уваги тут на несуперечність, а не на актуальну заданість різноманіття програм, дозволяє, по-перше, звільнити розгляди від штучних спроб підтримки несуперечності конкретних розглядів і, по-друге, забезпечити досить високу їх змістовність.

Непряме збагачення ПП

Обрана точка зору на ПП суттєво розширює можливості збагачення уявлень про програми. В першу чергу за рахунок залучення до розглядів поряд з власне програмами також процесів їх генезису. Першорядне місце для розуміння природи цих процесів займає композиційна парадигма [1, 2, 5, 6, 9, 11]. Адже композиції як генетичні структури програм є тими первинними структурами, що складають «загальний знаменник» для всіх програмологічних розглядів та програмістських розробок. Це випливає з обгунтованої в роботах [8, 9, 11] точки зору на *програмування* як на *процес породження та застосування*

композицій. Таке розуміння програмування є ключовим для подальших розглядів у тому сенсі, що воно природним чином збагачує розуміння програмування, зводячи його до розуміння як процесів породження та застосування композицій, так і до їх взаємодії. Природа цих процесів принципово різна. Проявляється це хоча б у тому, що перш ніж застосовувати дещо, зокрема і композицію, це необхідно мати в наявності. І якщо вивчення питання застосування композицій сьогодні є досить просунутим як у теоретичному, так і в практичному аспекті [12], то відносно проблеми породження композицій такого сказати не можна. Адже добре відомо, що згідно з дескриптивним аналогом теореми Гьоделя про неповноту будь-який рекурсивно перераховний клас композицій є неповним [13]. Тобто, змістовно кажучи, процес породження композицій у принципі не може бути об'єктивізований, а значить необхідним є залучення до розглядів даного питання суб'єкта.

В роботах [1– 6] докладно обгрунтовано прагматичну обумовленість переходу в інформатико-технологічній проблематиці від традиційних моноабстрактних підходів до поліабстрактних, зокрема біабстрактних розглядів. Біабстрактність є характерною особливістю дескриптивних систем, яка природним чином акцентує увагу не на окремому наперед заданому розумінні сутності, а *на прагматико-обумовленому взаємодоповненні* сутності та її розуміння. Для останнього, як окремого виду сутностей, які є обумовленими прагматикою, в ДС уведено спеціальний термін – *суть*. Розглядались такі корисні конкретизації ДС, як відкрито-замкнені, логіко-предметні, абстрактно-інкапсулятивні, динаміко-статичні, суб'єктно-об'єктні системи. Загальним для всіх таких систем є двоєдиність поглядів на суть розглядуваної сутності, що випливає з фундаментального значення *обумовлення* для ДС та тези про двоєдиність [8]. По відношенню до обумовлення ця теза акцентує увагу на взаємодоповнення двох точок зору на нього – як на процес та як на результат процесу. Причому обумовлення трактується тут максимально широко. Так, що навіть обезумовлення розуміється як вид роду обумовлення. Виходячи з зазначеного зрозуміло, що програмування це процес, що підтримує двоєдиність розуміння програми і є головною ланкою програмної системи. Тому подальше збагачення поняття програми природно здійснювати через виокремлення відповідного виду у роді дескриптивних систем. Відповідно до [2, 4, 8] дескриптивна система презентує собою розуміння сутності через виокремлення її суті та визначення двоєдиного зв'язку між ними у вигляді суте сутнісного відношення. Точніше, ДС формалізується у вигляді наступної системи:

$$\begin{cases} \textit{Дескрипціал ::=суть, що обумовлює сутність,} \\ \textit{Дескрипт::=суть, що обумовлена дескрипціалом} \end{cases}$$

Принциповою особливістю експлікації ДС є те, що в рамках концептуально-єдиної її формалізації явно використане обумовлення як процес (дія) у двох видах. Один з них недосконалий (обумовлює), а інший – досконалий (обумовлений). При цьому, взаємодоповнення цих видів дій підтримується суттю як сутністю другого порядку. Це дає всі підстави розглядати введену формалізацію як адекватну експлікацію взаємодії суб'єкта та об'єкта, домінуючу роль у яких, очевидно, відіграє суб'єкт.

Подальші збагачення ДС в напрямку програмних систем, які, що зазначалося вище, підтримують породження та застосування композицій, пов'язані, в першу чергу, з конкретизацією відповідних до даної прагматики обумовлень. Природа ж останніх, по-перше, обумовлена розумінням, як композицій, так і дій (процесів) їх породження та застосування. По-друге, обумовлення як дія природним чином пов'язана з відповідною, як завгодно складно структурованою, умовою. Цілком природно, що одні умови

є похідними інших. Останні є своєрідним «загальним знаменником», що складає саму основу обумовлень. Зрозуміло, що такий знаменник, у свою чергу, є похідним прагматики розглядів. Носієм же останньої, очевидно, є суб'єкт розгляду. У визначенні ДС він представлений дескрипціалом, який здійснює обумовлення розглядуваної сутності у відповідністю з її прагматико-орієнтованим розумінням. Залучення у розгляди дескрипціалу, таким чином, здійснюється, зокрема, у повній відповідності до зазначених вище загальних міркувань про природу породження композицій. Загалом, виходячи з наявного причино-наслідкового зв'язку між породженням та застосуванням композицій, корисно зазначити, що, по-перше, породження та застосування композицій це види обумовлень та, по-друге, породження композицій обумовлює їх застосування. Подальше роз'яснення природи породження композицій за необхідністю відноситься вже не до загальнозначущих, а до предметних розглядів програм та програмування. Вони будуть проведені у свій час та у потрібному місці. Зараз же звернемо увагу на композиції та їх застосування. Домінантою розглядів тут, зрозуміло, є композиція, як сутність, що є первинною у тандемі «сутність – її застосування». При цьому зауважимо, що композиція є сутністю, яка обумовлена процесом породження, природа якого є суб'єктивно-обумовленою.

Композиція та декомпозиція в контексті ДФС

Виходячи з вищезазначеного, розуміння композиції, як і будь-якої іншої сутності зводиться до відповідної дескриптивної системи. Відповідність встановлюється за рахунок прагматико-орієнтованого збагачення дії обумовлення. Такому збагаченню за необхідністю повинні передувати змістовні роз'яснення його мотивів. Розгляд композиції в контексті ДС передбачає змістовне викриття природи відповідного сутесутнісного відношення, суттю якого є композиція, а рефлексивно-транзитивне замикання його адекватно підтримує саме дескриптивна система.

Передумовою появи композицій можна вважати виділення цілого та частки як найважливіших категорій пізнання у їх нерозривному зв'язку, що дозволило говорити не просто про генезис сутностей, а, що більш важливо, про його системність та викриття відомого епістемологічного результату, отриманого ще Арістотелем про незводимість цілого до простої суми його часток. Цей результат дозволив поряд із сутностями говорити про властиві їм структури. Виходячи з відомого епістемологічного закону про первинність генетичних структур та згаданої вище системності генезису випливає засадничий для композиційної парадигми *принцип обумовленості*: структури сутностей обумовлені їх генетичними структурами. Не виключенням тут є і програми та програмування як сутності. Тому саме генетичні структури програм та програмування як передумови композицій складають «загальний знаменник» для всіх програмологічних досліджень та програмістських розробок. Іншими словами, генетичні структури програм є сутями структур програм, а композиції – сутями генетичних структур. Даний приклад рефлексивно-транзитивного замикання «структурологічних» сутесутнісних відношень без сумніву становить самостійний інтерес і його можна було б продовжити. Але тут його значення полягає не стільки в ньому самому, скільки у ДС, що підтримує дане замикання. Виходячи з зазначеного є очевидним, що роз'яснення природи композиції зводиться до покрокового збагачення введеної вище дескриптивної системи, ключовими елементами якого є поняття цілого та частки, одиничного та загального, типу та представника, внутрішнього та зовнішнього і т.п.

Перший крок збагачення виділяє серед ДС такі системи, процеси обумовлення яких є типовими, тобто такими, що здійснюються в рамках прагматико-обумовленої типізації (ТОП) універсуму сутностей.

Такі обумовлення будемо називати дефініціями (визначеннями), а самі системи – *дефінітними системами* (ДФС). Змістовно кажучи, ДФС це ДС, обумовлення яких є в свою чергу обумовленим прагматикою. А саме обумовленість прагматикою, як зазначалося вище є характерною рисою композицій. Таким чином,

Дефініціал ::=суть, що визначає сутність,
Тип::=суть, що визначена дефініціалом,

де визначає (визначена) розуміється як типово обумовлює (типово обумовлена).

Уведення ДФС збагачує розуміння композиції як дескрипта в рамках ДС до розуміння її як типа в рамках ДФС. Наступний крок збагачує розуміння типу в контексті фундаментального в епістемології відношення одиничного та загального, яке розуміється максимально широко та включає такі його прояви, як «одиничне у загальному» та «загальне в одиничному» [4, 14]. Змістовно кажучи, відображає те, що тип може розумітися, як одноманітно, так і різноманітно, наприклад, як властивість та як розмаїття представників. Таке розуміння природно індукує збагачення ДФС так званими *монадними* та *поліадними системами* (МС та ПЛС, відповідно):

Монадіал ::=тип, що монадизує тип,
Монада::=тип, що монадизований монадіалом,
Поліадіал ::=тип, що поліадизує сутність,
Поліада::=тип, що поліадизований поліадіалом

де монадизує (монадизований) розуміється як одноманітно визначає (одноманітно визначений), а монадизує (поліадизований) – як різноманітно визначає (різноманітно визначений).

Поліадні системи (ПЛС) збагачують розуміння композиції як типа в рамках ДФС до розуміння її як поліади в рамках ПЛС. Подальше збагачення пов'язане з розумінням різноманіття як внутрішньо властивого природі розглядуваної сутності як поліади (наслідок поліадизації) та із зовнішніми причинами такого розуміння сутності (причинами поліадизації) у їх нерозривному причинно-наслідковому зв'язку. Це обґрунтовує збагачення ПЛС так званими *інтроадними* та *екстрааднимисистемами* (ІС та ЕС, відповідно):

Інтроадіал ::=поліада, що інтроспектує сутність,
Інтроада::=поліада, що інтроспектована інтроадіалом,
Екстраадіал ::=поліада, що екстраспектує сутність,
Екстраада::=поліада, що екстраспектована екстраадіалом,

де інтроспектує (інтроспектована) розуміється як внутрішньо поліадизує (внутрішньо поліадизована), а екстраспектує (екстраспектована) – як зовнішньо поліадизує (зовнішньо поліадизована).

ІС та ЕС збагачують розуміння композицій зовнішньо-внутрішнім зв'язком, який відображає прагматичну обумовленість композиції як наслідку відповідною причиною. Подальше збагачення якраз і направлене на адекватну підтримку взаємодоповнення внутрішніх та зовнішніх аспектів поліадизації. Це надає всі підстави для збагачення ПС через уведення *композиційних* та *декомпозиційних систем* (*КС* та *ДКС*. відповідно).

Композиціал ::= екстраада, що композитизує сутність,
Композиція::= інтроада, що композитизована композиціалом,

$$\left\{\begin{array}{l} \textit{Декомпозиціал ::= інтроада, що декомпозитизує сутність,} \\ \textit{Декомпозиція::= екстраада, що декомпозитизована декомпозиціалом,} \end{array}\right.$$

де композитизує (композитизована) розуміється як композитно інтроспектує (композитно інтраспектована), а декомпозитизує (декомпозитизована) – як декомпозитно екстраспектує (декомпозитно екстраспектована). Так уведені КС та ДКС адекватним чином підтримують взаємодоповнення в програмуванні засобів композиціонування та декомпозиціонування, у якому, змістовно кажучи, на рівні концепції рішення програмістських задач композиційні структури домінують над декомпозиційними, а на рівні конкретних представників таких рішень – навпаки.

Породження композицій в контексті взаємодії КС та ДКС

Уведення КС та ДКС в їх взаємодоповнюючому зв'язку суттєво збагачує розгляди програмування саме в аспекті породжень композицій. Ключовим об'єктом збагачення тут є двоєдиність цих породжень, яка проявляється у можливості прагматико-обумовленого вибору точки зору на породження композицій як на процес та як на наслідок останнього. У першому наближенні саме КС підтримує у взаємодоповненні біфуркаційність точок зору на породження композицій. При цьому, «процесну» точку зору максимально широко, аж до ототожнення в розглядах процесу з його наслідком, підтримує композиціал, який «поєднує» в собі знову ж таки у взаємодоповненні, як процес композитизації, так і його передумову. Композиція ж трактується тут як наслідок процесу. Вище вже зазначалось про неможливість повної об'єктивізації процесу (!) породження композицій та необхідність залучення до розглядів суб'єкта. Рішення, що лежить «на поверхні» – повністю суб'єктивізувати композиціал, тобто, виключивши з розглядів сам процес породження, фактично постулювати його результат.

Очевидно, що такі рішення можуть буди адекватно підтримані самою композиційною системою. Змістовно кажучи, це випадки, коли програмування у кращому випадку розглядається як застосування заданих композицій, а частіше навіть, ототожнюється з результатом такого застосування. Зрозуміло, що така трактовка програмування, точніше її абсолютизація, при всій її загальності йде у розріз зі згадуваним вище принципом обумовленості і вже тому не може розглядатись у якості «загального знаменника» розглядів програм та програмування. Причина у недостатній змістовності такого рішення. Тому необхідне його прагматико-обумовлене збагачення. В першому наближенні таке збагачення повинно бути направленим на розкриття природи взаємодоповнюючого зв'язку КС та ДКС. Адже значимість взаємовпливу процесів декомпозиції та композиції не тільки в програмуванні, але й узагалі в розв'язанні будь-яких реальних задач не викликає сумніву (от лат. *divide et impera*). Природа зв'язку КС та ДКС, зокрема і з огляду на притаманну їй суб'єктивність, за необхідністю складно влаштована. Змістовно кажучи її зміст відображає прагматико-обумовлений, а значить суб'єктно-орієнтований зв'язок процесів породження композицій з процесами розв'язання задач. І якщо розв'язання будь-якої задачі як породження та застосування композиції (до підзадач) обумовлюється відповідною прагматико-обумовленою декомпозицією вихідної задачі, то сама декомпозиція, у свою чергу, обумовлюється відповідними (наявними) засобами композиціювання. Іншими словами, аналіз розв'язання різних задач показує, що пов'язані з ними композиції є структурованими сутностями, тобто їм притаманні ті чи інші структури. При цьому зрозуміло, що всі вони в своїй основі спираються на деякі базові композиційні структури, по відношенню до яких самі є похідними. Ці базові структури складають змістовно кажучи

прагматико-обумовлений «загальний знаменник» бачення процесу розв'язання задачі суб'єктом її розв'язання. З наведених змістовних роз'яснень випливає, що для будь-якої КС природа композиціалу як сутності, що підтримує породження композицій є обумовленою відповідною декомпозицією (результат процесу декомпозитизації). Побічно це відображено у визначеннях КС та ДКС, де композиціал та декомпозиція є сутностями одного роду – екстраад. З визначення ж ДКС стає очевидним, що декомпозиція як сутність, що обумовлює композиціал відповідної КС у свою чергу є обумовленою декомпозиціалом – сутністю, що підтримує декомпозитизацію задачі. З тих же змістовних роз'яснень, а також з того, що декомпозиціал та композиція є сутностями одного роду – інтроад слідує, що природа декомпозиціалу в свою чергу є обумовленою композиціями, що складають вищезгаданий «загальний знаменник» прагматико-обумовленого (суб'єктно-орієнтованого) розуміння процесу розв'язання задач, а значить є базою цього процесу. Такі міркування можна було б продовжувати як завгодно довго. Але з прагматичної точки зору можна цілком обмежитись вже наведеними. Таким чином, пара <КС, ДКС> адекватно підтримує прагматико-обумовлене програмування в частині суб'єктно-орієнтованого породження композицій. Композиційно-декомпозиційна взаємодія цих систем обумовлює логіку породження композицій. Що ж стосується предметної складової даного процесу, то її природа обумовлена багатьма чинниками, зокрема, наприклад, розглядуваною областю задач та відповідним до неї суб'єктно-орієнтованим вибором базових композицій.

Висновки

Головна ідея, що неодноразово нами підкреслювалась на протязі всієї роботи це мотивація об'єктивної необхідності переходу від екземпляр них дефініцій до дефініції самих дефініцій, зокрема, від екземпляр них дефініцій програм до визначення загальної суті самих таких дефініцій. Ключове значення в такому переході відіграють суте сутнісні відношення, які адекватно експлікують зв'язок між сутностями та їх сутями, зокрема між процесом програмування та його результатом. В якості універсального прагматико-обумовленого засобу, що підтримує рефлексивно-транзитивне замикання цих відношень виступають дескриптивні системи та їх онона прагматико-обумовлена конкретизація – дефінітні системи. У рамках ДС та ДФС цілком природно реалізується взаємодоповнення одиничного й загального понять (взаємодія МС та ПЛС [1,4,15–17]), розкриття суті причинно-наслідкового зв'язку, властивого процесу поліадизації (взаємодія ІС та ЕС) як адекватні збагачення ДФС. Останні складають дефінітологічну основу сутнісної платформи, що підтримує композиційну парадигму систем програмування.

Список використаної літератури

1. Редько В.Н. Дескриптологические основания программирования // Кибернетика и системный анализ. – 2002. – № 1. – С.3 – 19.

2. Редько В.Н. Основания дескриптологии // Кибернетика и системный анализ. – 2003. – № 5. – С.16 – 36.

3. Редько В.Н.,Редько И.В, Гришко Н.В. Дескриптивные системы: концептуальный базис// Проблеми програмування. – 2006. – № 2–3. – С. 75 – 80.

4. Редько В.Н., Редько И.В., Гришко Н.В. Дескриптологические основания сущностной платформы // Проблеми програмування. – 2010. – № 2–3. – С. 13–21.

5. Редько В.Н. Основания программологии // Кибернетика и системный анализ. – 2000. – № 1. – С.35–57.

6. Редько В.Н., Редько И.В, Гришко Н.В. Программологические основания сущностной платформы// Проблеми програмування. – 2008. – № 2–3. – С. 72–78.

7. Редько В.Н. Основания программологии // Кибернетика и системный анализ. – 2000. – № 1. – С.35–57.

8. Редько И.В. Теория дескриптивных сред и ее применения. Докт.диссерт. – К.: НТУУ «КПІ», 2008. – 403 с.

9. В.Н. Редько, І.А. Басараб, М.С. Нікітченко Композиційні бази даних.– К.: Либідь,1992.–192 с.

10. Поппер К. Объективное знание. Эволюционный подход. – М.: Изд-во «Эдиториал УРСС», 2002. – 384 с.

11. Редько В.Н., Редько И.В. Экзистенциальные основания композиционной парадигмы //Кибернетика и системный анализ.– 2008.– №2.– С. 3–12

12. Барендрегт Х. Ламбда-исчисление. – М.: Мир. – 1985. – 606с.

13. Редько В.Н. Композиции программ и композиционное программирование // Программирование. – 1978. – № 5. – С. 3–24.

14. Хинтикка Я. Логико-эпистемологические исследования. – М.: «Прогресс». – 1980. – 448 с.

15. Черч А. Введение в математическую логику. – М.: ИИЛ. – 1960. – 485 с.

16. Карнап Р. Значение и необходимость. – М.: Изд.проект «Тривиум», 1958. – 380 с.

17. Пап А. Семантика и необходимая истина. – М.: Изд-во «Идея-Прес», 2002. – 418 с.

Стаття надійшла до редакції 11.07.2012

Системы программирования: дескриптивные основания

Редько И.В., Татариков А.О.
Национальный технический университет Украины «Киевский политехнический институт»
Редько Д.И.
Киевский национальный университет им. Т.Г. Шевченко

В рамках концепции дескриптивной системы (ДС) рассмотрено понятие системы программирования (СП) и платформы программирования (ПП). Разработан универсальный метод эволюционного обогащения сущностей, основу которого составляет прагматико-обусловленная типизация сущностей (ТОП), а главным инструментом типизации выступает понятие дефинитной системы (ДФС). Метод ТОП использован для построения понятийной структуры ПП. В частности, дано определение понятий декомпозиции и композиции как адекватных обогащений соответствующих ДФС – ключевых элементов понимания понятий программирования и программ.

Ключевые слова: дескриптивная система, дефинитная система, платформа программирования, прагматико-обусловленная типизация, дескрипция, дефиниция, монада, полиада, композиция, декомпозиция.

The systems of programming: descriptive foundations

Redko I., Tatarikov A.
National Technical University of Ukraine «Kyiv Polytechnic Institute»
Redko D.
Taras Shevcchenko National University of Kyiv

Within the concept of descriptional system (DS) the notions of programming system (PS) and programming platform (PP) have been considered. A universal evolutional enrichment method has been worked

out. The methods base composed by pragmatics-conditioned typification (PCT). And the main instrument of last-mentioned is notion of definitical system (DFS). The PCT method has been used to build the notional structure of PP. Particularly the definition of decomposition and composition as adequate enrichment of according DFS – key elements of comprehension of programs and programming has been given.

Keywords: descriptive system, definitive system, programming platform, pragmatic-conditioned typification, description, definition, monad, polyad, composition, decomposition.