УДК 519.683:681.3

Є.І. Моренцов

КОНЦЕПТУАЛЬНА АРХІТЕКТУРА ІНТЕГРОВАНОГО СЕРЕДОВИЩА ГЕНЕРУЮЧОГО ПРОГРАМУВАННЯ

Розглянуті концепції побудови архітектури інтегрованого середовища генеруючого програмування (ІС ГП). Визначені типові компоненти, базова платформа і представлена модель архітектури ІС ГП.

Вступ

Генеруюче програмування (generative programming) – це методи і засоби генерації сімейств систем і застосувань з окремих компонентів, аспектів, повторновикористовуваних компонентів (ПВК), каркасів й ін. [1]. Базис цього програмування складають методи і засоби об'єктно-оріентованого та об'єктно-компонентного програмування [2, 3], доповнені механізмами генерації багаторазових елементів і ПВК, а також властивостями їхньої мінливості (змінності), взаємодії й іншими. Головний продукт інженерії предметної області (ПрО) у генеруючому програмуванні (ГП) – це сімейство програмних систем (ПС), які генеруються на основі загальної генеруючій моделі GDM (Generative Domain Model), що включає у себе засоби визначення членів (представників) сімейства, а також методи генерації, зборки членів сімейства і базу конфігурації для розгортання сімейства в операційному середовищі.

У небагатьох публікаціях, присвячених дослідженню окремих аспектів ГП, до цього часу не розглядалося питання визначення та побудови архітектури ІС ГП. Ураховуючи, що дослідження цього відкриває шлях до практичного застосування ГП, в рамках теми фундаментальних досліджень НАН України у якості першого кроку нами була розроблена концептуальна архітектура ІС ГП.

Під ІС ГП будемо розуміти систему мовних, графічних та інструментальних засобів, яка призначена для розробки сімейства програм ПрО в парадигмі генеруючого програмування. При цьому для кожного унікального домену (ПрО) фактично створюється своє ІС ГП. Саме тому наше дослідження було сконцентровано на визначенні концепцій загальної або типової архітектури ІС ГП, яка містить класи компонентів ІС, а екземпляри цих класів добираються із наявних компонентів при побудові ІС для конкретного домену [1, 4].

Концептуальна архітектура ІС ГП була визначена виходячи з:

- парадигми генеруючого програмування;
- проекції життєвого циклу, точніше – технології розробки програмних систем (ПС), на парадигму ГП [5];
- функціональності інструментарію, необхідного для підтримки технологічних процесів ГП, внутрішніх і зовнішніх інтерфейсів інструментарію;
- базової платформи, на якій реалізується ІС ГП.

Парадигма генеруючого програмування

Генеруюче програмування – являє собою парадигму технології розробки програмного забезпечення, засновану на моделюванні сімейства програмних систем, використовуючи які, можна по конкретних технічних вимогах автоматично отримати спеціалізований та оптимізований проміжний або кінцевий програмний продукт з елементарних, багаторазово використовуваних компонентів реалізації за допомогою бази знань *про конфігурації*.

Генеруюче програмування фокусує увагу не на унікальних продуктах (об'єктах), а на сімействах програмних систем (класах об'єктів). Таким чином, генеруюче програмування має велику подібність з теорією універсальних та інтегративних моделей, застосовуваних для синтезу об'єктів, і які є моделями не окремо взятого об'єкта, а моделями всіх об'єктів, що нале-

жать розглянутому класові (моделлю сімейства об'єктів або систем).

Таким чином, генеруюче програмування – це автоматизоване виробництво програмних продуктів з окремих компонентів, на манер того, як це відбувається при виробництві промислових виробів. Ключовими моментами тут є *уніфікація*, *автоматизація* і *гнучкість*.

Для реалізації парадигми генеруючого програмування необхідні [6]:

- компоненти реалізації, з урахуванням загальної архітектури даної лінійки продуктів;
- модель знань про трансляцію абстрактних вимог у конкретні зв'язки компонентів;
- реалізація цих знань у генераторах.

Схематично, у контексті визначення архітектури ІС ГП на найвищому рівні абстракції, парадигма ГП представлена на рис. 1.

Проекція технології розробки програмних систем на парадигму генеруючого програмування

Основними складовими концепції генеруючого програмування, що випливають з парадигми ГП є:

- формалізація предметної області і побудова простору рішень;
- мови опису предметних областей та мовно-орієнтоване програмування;
- методи реалізації застосувань на основі мов описів предметних областей і повторного використання генерованих, параметризованих і конфігурованих об'єктів;
- спеціальне мовно-орієнтоване середовище реалізації застосувань та інструментарій розробки і застосування повторно використовуваних об'єктів.

Парадигма і, відповідно, концепції ГП впливають на технологію розробки ПС у середовищі ГП. Щоб визначити основні елементи даної технології виконаємо проекцію технології розробки ПС на парадигму ГП (рис. 2).

Основними її елементами є:

- знання конфігурації;
- DSL-технології;
- технології генерації;
- технології компонентів.

Склад кожної з них представлений на рис. 2.

DSL-технології містять:

- мови програмування (наприклад, C++, Java, C#, Ada 2005);
- розширювані мови;
- нові текстуальні мови;
- графічні мови;
- інтерактивні помічники та інтерфейси користувача;
- також будь-які комбінації з вище перерахованих компонентів.

Рис. 1. Схематичне представлення парадигми генеруючого програмування

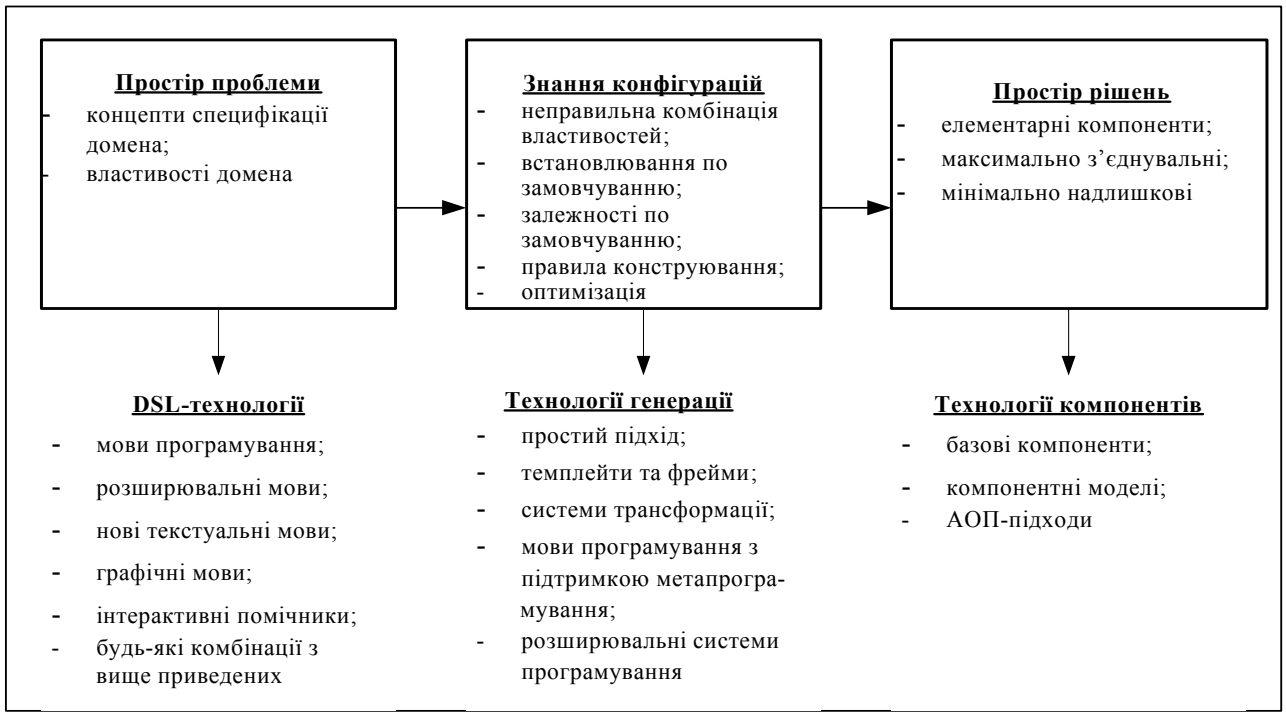

Рис. 2. Проекція технології на парадигму генеруючого програмування

Технології генерації включають:

- простий прохід;
- шаблони та фрейми;
- системи трансформації, що забезпечують трансформацію елементів опису домену з мови вищого рівня до мови нижчого рівня (аж до виконуваного коду);
- мови програмування з підтримкою мета програмування;
- розширювані системи програмування.

Компонентні технології містять:

- базові компоненти;
- компонентні моделі;
- АОР-підходи (тобто підходи, властиві аспектно-орієнтованому програмуванню).

Типові компоненти інтегрованого середовища генеруючого програмування

У технології ГП можна виділити дві основних фази:

- моделювання і специфікування домену (ПрО);
- автоматичну генерацію коду ПС.

При моделюванні і специфікації домену двома фундаментальними проблемами, які необхідно досліджувати, є склад проблемних *мов моделювання та специфікацій домену* і склад *інструментів для моделювання і специфікацій домену*.

Автоматичну генерацію коду програмної системи виконують *генератори і перетворювачі моделей.*

Базовим елементом технології ГП і ядром ІС ГП, навколо якого будуються усі компоненти, є *репозитарий*, у якому накопичуються *повторно використовувані компоненти* та *знання конфігурації*. ПВК представляють собою параметризовані компоненти сімейства систем (програм) домену, які реалізують деякі методи обробки об'єктів домену. Під знаннями конфігурації розуміється сукупність правил, які визначають виконувані комбінації, залежності і значення за замовчуванням для розроблюваних компонентів.

До складу мов специфікації та моделювання домену входять:

- мови специфікації домену (DSL);
- метамови специфікації DSL для опису потрібної мови специфікації конкретного домену;
- проміжні мови (XMI, XML, XSLT та ін.);

- графічні мови (моделі та метамоделі);
- мови програмування.

Інструменти для моделювання і специфікації домену включають:

- редактори мов;
- графічні редактори;
- інструменти проектування DSL (редактор метамови опису DSL та генератор мови DSL за її описом).

Окремий клас компонентів ІС ГП складають методи проектування DSL та методи моделювання і специфікації домену, які використовує розробник ПС.

У даний час у рамках генеруючого програмування розроблено декілька способів генерації коду [7]. Серед них можна виділити наступні:

- підстановки;
- підстановки з виконанням коду;
- оброблювачі даних регулярної структури.

Генерація коду, що заснована на підстановках, припускає, що розроблювач створює шаблон коду і набір даних у спеціальному форматі, а потім, за допомогою допоміжної програми, виконує підстановку цих даних у шаблон. Набір використовуваних у шаблоні підстановок може визначатися як статично, так і динамічно. Такий підхід досить наочний і простий у використанні, однак має досить обмежену область застосування і вимагає попередньої підготовки переданих для підстановки даних.

Класичний приклад підстановок (з деякими застереженнями) – шаблони (templates) мови C++.

Набагато ширше можливості при генерації коду з використанням підстановки з виконанням коду. Цей вид генерації відрізняється від попереднього можливістю застосовувати в шаблоні не тільки підстановки, але також вставки виконуваного коду, що оперує переданими в шаблон даними. Код, що виконується, найчастіше використовує мову, спеціально створену для конкретного типу шаблонів і яка містить основні алгоритмічні конструкції (прості розгалуження – IF, виконання ітерацій за переданими як параметри спискам – WHILE, цикли з заданою кількістю ітерацій – FOR).

При ускладненні конструкцій, застосовуваних у шаблоні, і збільшення можливостей мови, зменшується наочність самих шаблонів.

Третій спосіб генерації коду заснований на оброблювачах даних регулярної структури і припускає повний поділ даних і їхнього представлення. У цьому випадку шаблон відіграє роль оброблювача даних і пишеться на спеціальній метамові. Прикладом може послужити XSLT-обробка даних, представлених у XML-форматі.

Основною перевагою цього способу генерації коду є можливість обробки даних складної структури без попередньої підготовки цих даних. Існування перерахованих видів генерації коду обумовлено неоднорідністю розв'язуваних у рамках генеруючого програмування задач і структури метаданих.

Питання про використання того або іншого способу генерації залежить від умов конкретної задачі і даних, використовуваних для такої генерації.

Генератори і перетворювачі моделей належать до класу інструментів ІС ГП. Серед них можна виділити наступні:

- інструменти трансформації (DSL-трансформатори), які трансформують специфікацію домену на мові DSL у еквівалентний опис на одній із проміжних мов (XML, XSLT, XMI) або на мові програмування;
- DSL-компілятори виконуваного коду, які із специфікації домену генерують виконуваний код;
- перетворювачі моделей;
- компілятори з мов програмування;
- компілятори з проміжних мов.

Останні два види генераторів також генерують виконуваний код.

Важливою складовою ІС ГП є *інтерфейси*, в тому числі інтерфейс користувача, інтерфейси між інструментами й інтерфейс із базовою платформою реалізації ІС ГП.

На рис. 3 показана концептуальна архітектура репозитарію ІС ГП, а на рис. 4 – класифікація компонентів типового ІС

ГП.

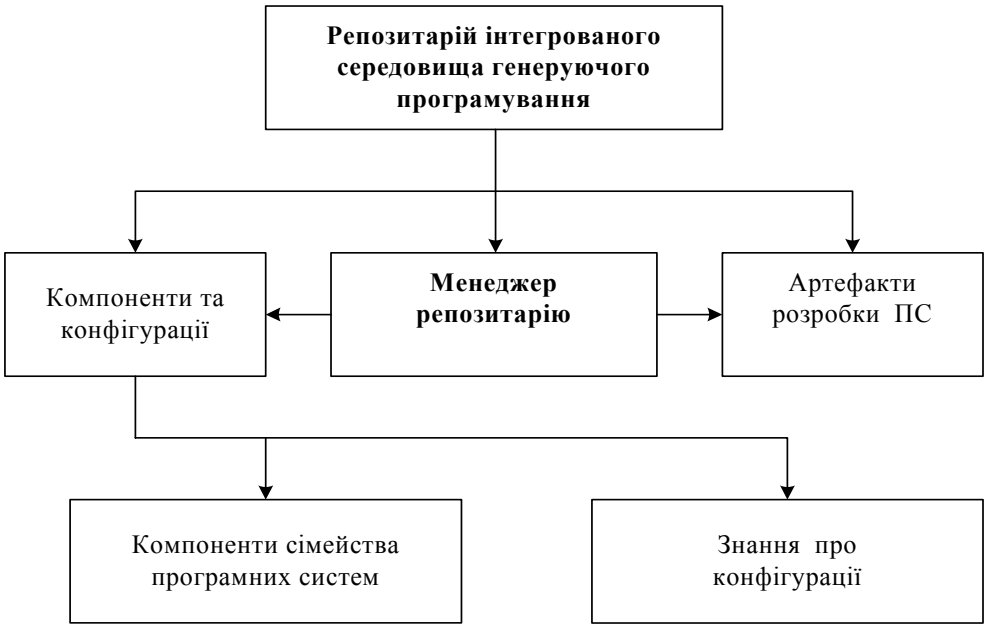

Рис. 3. Концептуальна архітектура репозитарію інтегрованого середовища генеруючого програмування

Базова платформа інтегрованого середовища генеруючого програмування

У якості базової платформи реалізації ІС ГП нами запропоновано середовище Eclipse [8]. Eclipse має повністю відкритий вихідний код і розвинутий механізм розширення середовища. Віртуальна машина мови Java дозволяє використовувати середовище Eclipse на різних платформах, включаючи Windows і UNIX. У вихідних текстах надається середовище розробки для мови Java – основної мови програмування Eclipse. Надаються також засоби інтеграції в Eclipse компіляторів мови C++ зовнішніх розроблювачів. Паралельно ведуться проекти з відкритим вихідним кодом і для ряду інших мов програмування. Усі ці проекти використовують стандартні *механізми розширення середовища Eclipse* .

На вибір Eclipse як базового середовища вплинули дві важливі властивості Eclipse.

Першою з них є надання засобів для розробки інтерфейсу користувача, орієнтованого на використання конкретної мови програмування. Сукупність таких елементів інтерфейсу користувача для перегляду (views) і редагування (editors) інформації отримало назву *перспективи середовища Eclipse*. Всі елементи перспективи представляються на окремій закладці середовища. Існуючий у середовищі Eclipse редактор програм може бути легко розширений у синтаксично орієнтований редактор конкретної мови програмування. Для файлів проекту можуть бути задані фільтри, що дозволяють визначити файли з текстами програм й інші ресурси, що належать до перспективи мови програмування.

Другою важливою для нас властивістю Eclipse є можливість доповнення вже існуючих модулів середовища як новими, так і раніше розробленими інструментами через стандартні точки розширення.

Платформа Eclipse також має розширювану архітектуру модулів, що підключаються (плагинів), яка дозволяє працювати в ІС ГП практично з будь-яким типом даних. Розроблювач також може розробляти свої власні модулі, що підключаються, для роботи з будь-якими іншими об'єктами або використовувати готові модулі “третіх” виробників – в обох випадках

нові модулі, що підключаються, прозорим образом цілком інтегруються з існуючими інструментами.

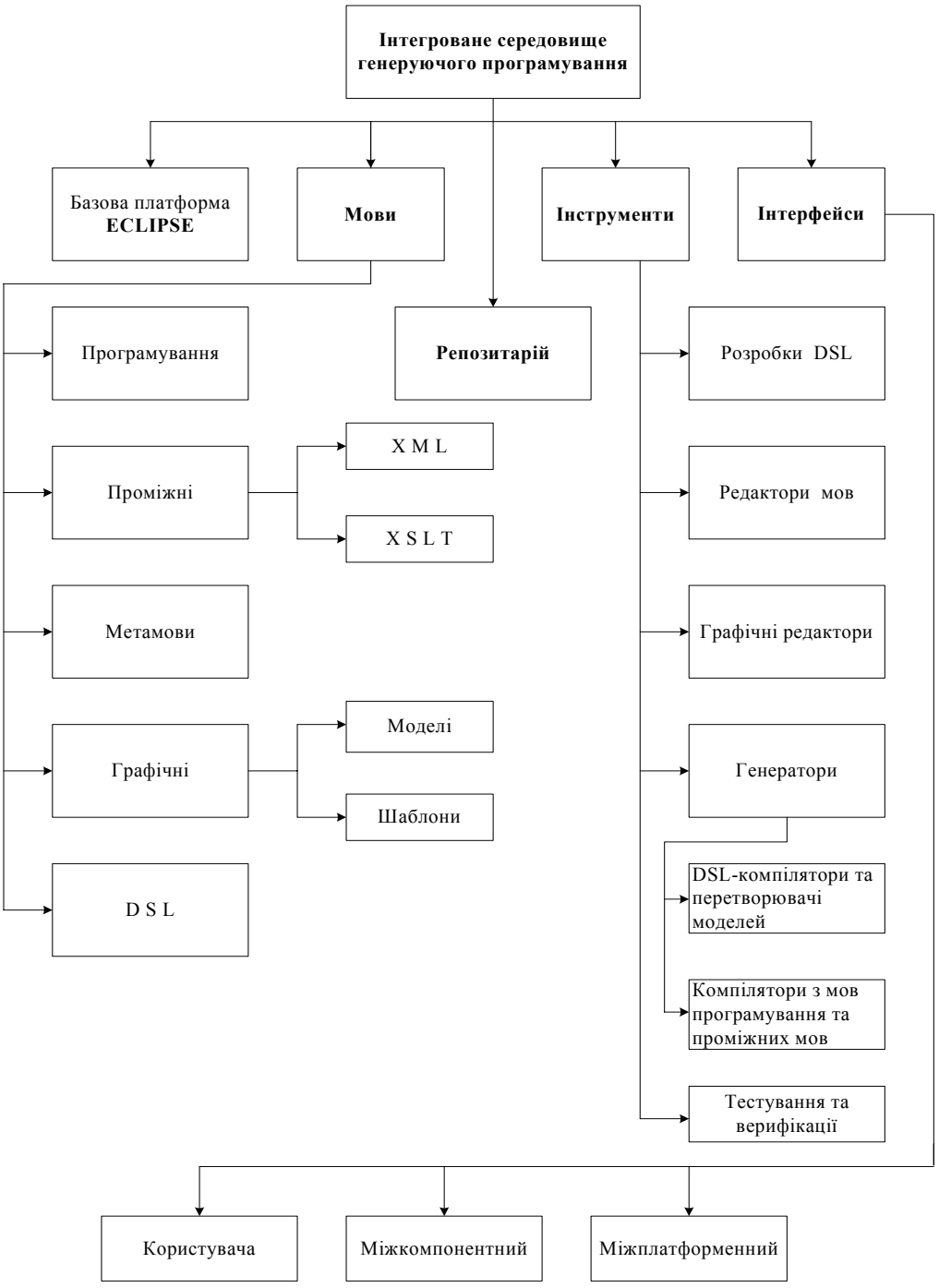

Рис. 4. Компоненти інтегрованого середовища генеруючого програмування

Саме тому нами пропонується у якості базової платформи для ІС ГП використовувати середовище Eclipse – як відкриту платформу для інтеграції інструментарію, яка підтримується великим і постійно розширюваним співтовариством компаній-виробників інструментів. При цьому платформа Eclipse забезпечує інтерфейси (точки розширення) для прозорої інтеграції інструментів. У результаті весь інструментарій інтегрованого середовища генеруючого програмування матиме єдиний графічний стиль і користувачеві не прийдеться заново вивчати користувальниць-

кий інтерфейс кожного окремого інструмента або кожної інструментальної платформи.

Архітектура інтегрованого середовища генеруючого програмування

Як основний результат дослідження нами визначена модель архітектури інтегрованого середовища генеруючого програмування, яка показана на рис. 5.

У ній можна виділити три великих блоки:

- інструментальне середовище розробки;
- репозитарій;
- базову платформу.

Склад кожного з них відображений на рис. 5, як і типові компоненти ІС ГП, інтерфейси між компонентами і між інструментами та користувачем-розробником ПС, схема створення опису ПрО та його перетворення у виконуваний код ПС.

Зрештою подамо *принципи побудови середовища генеруючого програмування на компонентній основі*, запропоновані В.М. Грищенко.

Як вищезазначено, для кожного сімейства застосувань (систем і програм) має створюватися своє ІС ГП, що включає конкретні компоненти. Процес побудови ІС ГП складається з 11 наступних кроків:

1) визначення змісту і границь предметної області, для якої будується середовище;

2) визначення моделей для обраної предметної області, їхніх властивостей і взаємозв'язків;

3) визначення формального апарата представлення предметної області;

4) визначення мовних засобів опису і рішення задач у предметній області;

5) визначення основних архітектурних і проектних рішень для застосувань із предметної області;

6) вибір frameworks для застосувань із предметної області;

7) реалізація templates і компонентів для застосувань із предметної області;

8) визначення формальних правил і мовних засобів генерації, параметризації, конфігурування для templates і компонентів;

9) вибір і/або розробка генераторів для templates;

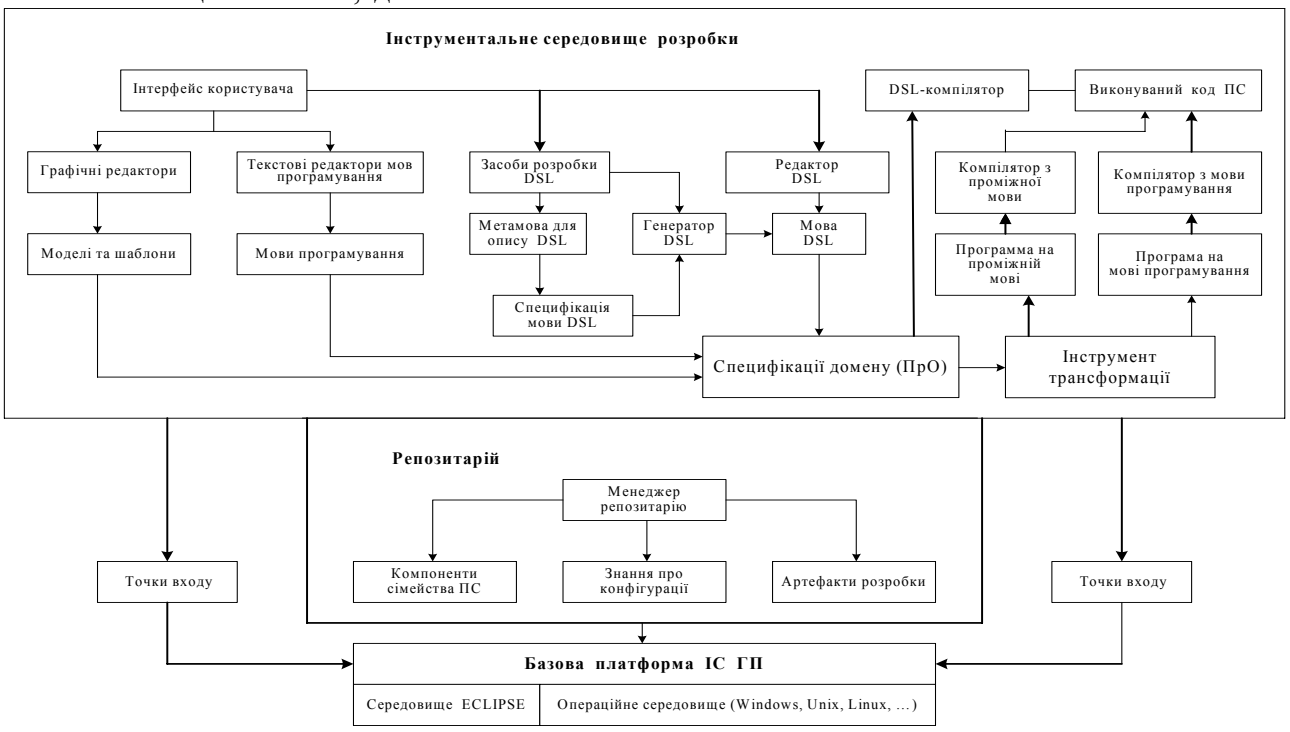

Рис. 5. Модель архітектури інтегрованого середовища генеруючого програмування

10) вибір і реалізація засобів для конфігурування і розгорнення застосувань;

11) вибір і реалізація засобів обробки вимог до створюваних застосувань.

Основою для побудови інтегрованого середовища генеруючого програмування є, по-перше, типова архітектура ІС ГП, по-друге, наявність відповідних компонентів (включаючи екземпляри інструментарію) із класів типових компонентів ІС ГП. Саме визначення необхідних екземплярів компонентів у залежності від специфіки ПрО, їх добір та/або розробка (у випадку відсутності потрібних) є основним стрижнем процесу побудови інтегрованого середовища генеруючого програмування на компонентній основі.

1. *Чернецки К., Айзенекер У.* Порождающее программирование // Методы, инструменты, применение. – М. – СПб. – Харьков. – Минск: Издательский дом Питер, 2005. – 730 с.
2. *Грищенко В.Н.* Метод об'єктно-компонентного проектування програмних систем // Проблеми програмування. – 2007. – № 2. – С. 113–125.
3. *Лаврищева Е.М.* Методы программирования // Теория, инженерия, практика. – Киев: Наук. думка, 2006. – 451с.
4. *Crarnetcki K.* Owerview of Generative Software Development // Canada, www.crarnetcki.fcm.org
5. *Лавріщева К.М.* Програмна інженерія. – К., 2008. – 319 с.
6. *Bordin Matteo.* Generative Programming and Components: theory and practice. http://www.studenti.math.unipd.it/~mbordin/proj/gpc.pdf.
7. *Канжелев С.Ю., Шалыто А.А.* Автоматическая генерация автоматного кода // Информационно-управляющие системы. – 2006. – № 6. – С. 35–42.
8. *Гамма Э., Бек К.* Расширения Eclipse: принципы, шаблоны и подключаемые модули / Пер. с англ. – М.: КУДИЦ-ОБРАЗ, 2005. – 384 с.

Отримано 17.05.2010

Про автора:

Моренцов Євген Іванович,
кандидат технічних наук,
старший науковий співробітник.

Місце роботи автора:

Інститут програмних систем
НАН України.
03187, Київ-187,
Проспект Академіка Глушкова, 40.
Тел. 38 044 526 4286.
e-mail: yevhen18@diawest.net.ua