УДК 681.03

К.М. Лавріщева

КОНЦЕПЦІЯ ІНДУСТРІЇ НАУКОВОГО СОФТВЕРА І ПІДХІД ДО ОБЧИСЛЕННЯ НАУКОВИХ ЗАДАЧ

Дано аналіз сучасного стану і застосування фабрик в індустріальному виробництві програмної продукції. Запропоновані базові складові фабрики наукового софтвера, обгрунтовано їх зміст і призначення у рішенні наукових завдань. Розглянуто підхід до індустрії обчислень різних наукових задач. Сформульовані нові дисципліни з індустріального виробництва програмних артефактів і reuses та навчання цим дисциплінам студентів за відповідними спеціальностями з інформатики і Computer Sciences.

Вступ

Виходячи з матеріалів державної цільової науково-технічної та економічної програми розвитку індустрії програмної продукції (ПП) України на 2012–2014 рр. (www.itdev.org.ua), внесок індустрії ПП у світовому виміру перевищує 100 млрд. дол. Але в Україні він містить десь 1.0 % за рахунок ринку програмістських послуг закордонним фірмам, обсяг експорту якого становить значно більший процент. Дана програма ставить задачу економічного та соціального розвитку суспільства за рахунок підвищення індустрії ПП шляхом консолідації наукових та підготовлених у вузах спеціалістів до потреб держави в індустрії національної ПП шляхом інвестицій. Це відкриє шлях до накопичення і продажу виготовлених цими спеціалістами різних видів наукових артефактів і ПП, починаючи з Вузів, аспірантур і завершаючи науково-дослідним інститутом.

Ідею індустрії комп'ютерів і ПП сформулював академік В.М. Глушков в Інституті кібернетики (1975 р.). За його ініціативою розвиток індустрії у СРСР ознаменувався побудовою низки комп'ютерів (Днепр–1, Днепр–2, серії машин «Мир» тощо), створенням фондів алгоритмів і програм (1978 р.), постановою про ПП як продукції промислово-технічного призначення (1984 р.) та про державне інвестування науково-дослідних інститутів усіх союзних республік ГКНТ СРСР, спрямоване на автоматизацію засобів виробництва ПП (1980–1990) та на обговорення різних шляхів організації індустрії ПП на всесоюзних конференціях (1980–1991) тощо.

В результаті були розроблені не тільки засоби автоматизації ПП, але і конкретні системи АСУ, АСУ ТП для різних галузей промисловості за методом збірки готових програм із фондів [1–3]. Перша фабрика (або програмно-будівельний інститут у Калініні – 1982 р.) проіснувала біля двох років. На той час не було визначено формального базису такого підприємства, готові програмні ресурси з фондів були частково незавершені, як програми повторного використання або невдосконалені для автоматизованої збірки. Поняття інтерфейсу різних програм ще не було визначено, а тільки сформувалося як необхідний елемент збірки.

Нині склалися усі необхідні умови, а саме, накопичено велику кількість різноманітних програмних ресурсів у електронних бібліотеках, зокрема, і наукових, як елементів індустрії наукового софтвера, дуже потрібних Європейському проекту Grid, почали діяти конкретні фабрики програм різного призначення та сформовані базові складові індустріального виробництва ПП з компонентів повторного використання (КПВ) [4, 5].

У роботі дається визначення основ фабрики – понять, змісту і сутності індустрії наукового софтвера (e–science – фізика, математика, біологія тощо), який створюється на різних математичних артефактах, розроблених на відповідних кафедрах університетів та науково-дослідних інститутів України з метою обчислювання наукових завдань з різних напрямів e–science глобального характеру. Для накопичення готових програмних ресурсів і артефактів для їх повторного використання розроблені

міжнародні стандарти з сертифікації програм, ПП та їх інтерфейсів. Вони розміщуються в бібліотеках, репозиторіях інститутів або Інтернету, як сховищ заготовок наукової продукції у галузі інформатики та Computer Science [5] для застосування їх як об'єктів індустрії ПП і обчислень.

1. Головні складові сучасних фабрик програм

Сутність виробництва продукції. Виробництво – це процес автоматизованого створення матеріальних благ, необхідних для задоволення індивідуальних і соціальних потреб деяких прошарків суспільства. Виробництво ПП орієнтоване на виготовлення продукції, потрібної науковцям, фахівцям комп'ютерних галузей підприємств тощо. Це виробництво ґрунтується на готових технічних і людських ресурсах. Одним з найважливіших технічних ресурсів фабрики програм є готові програмні артефакти, алгоритми і програми багаторазового використання (компоненти, сервіси, КПВ, reuses, assets і т. п.).

Рівень виробництва деякої продукції залежить від обсягу використаних ресурсів і обчислюється за функцією вигляду: $v = F(z, u)$,
де $v = (v_i)$ – вектор випуску продукції, $z = (z_i)$ – вектор витрат ресурсів, $u = (u_i)$ – матриця параметрів залежності функції витрат $z = F(w_j, u)$, $j = 1, 2, .., n$. Показники функції w_j відповідають обсягові виробництва, структурі виробничих фондів та рівню спеціалізації фабрики, їх значення для рівня спеціалізації фабрики формуються, як правило, статистично.

Загальна виробнича статечна функція випуску продукції Кобба–Дугласа

$$v = ne^{rt} L^{\alpha} K^{\beta},$$

де v – узагальнений випуск; n – нормативний множник; e – основа натурального алгоритму; t – показник рівня науково-технічного прогресу; L – витрати людської праці; K – величина капіталу; α, β – коефіцієнти еластичності [6].

Розрахунки цих функцій дають оцінки доходної частини, зокрема і саме фабрики програм. Економічність, системність та пропорційність випуску ПП залежить від принципів організації і керування ресурсами, визначення номен-клатури продукції, застосування комп'ютерів, людських ресурсів та персоналу з обслуговування фабрики (контролю вимог, специфікацій, процесів тестування, оцінювання ПП й ін.).

В Україні нині діють окремі невеликі організації і фірми з розроблення ПП, які орієнтовані на випуск в основному одиночних продуктів для окремих замовників, як правило, закордонних. Фірми з успадкування ПП, що покривають значний фрагмент ринку ПП, теж працюють в основному для закордонних держав, а не для потреб різних підприємств.

Для першого попиту концепції індустрії ПП запропоновано створити студентську фабрику програм на факультеті кібернетики в КНУ імені Тараса Шевченка. Ставиться мета залучити студентів для постачання на фабрику наукових артефактів і готових ресурсів відповідних кафедр, їх апробації на продуктовій лінії та розповсюдження не тільки на вітчизняному ринку, але з часом і закордонному.

Під *фабрикою програм* розуміється інтегрована інфраструктура зі зборкою готових ресурсів у ПП, потрібних державним, науковим, комерційним та іншим замовникам. Фабрика обладнується продуктовими лініями, набором засобів, інструментів і сервісів для автоматизованого виконання процесів на цих лініях у операційному середовищі. Дане у [7] визначення фабрики таке: фабрика програмного забезпечення – це погоджений набір процесів, засобів і інших ресурсів для прискорення усього циклу створення тих чи інших програмних компонентів, застосувань і систем.

Фабрика надає середовище, орієнтоване на автоматизацію виробництва продукту. З погляду інформаційних технологій фабрика дає набір інструментів для переходу до індустрії ПП із метою збільшення продуктивності розробки продукту на кожному етапі життєвого циклу із заданими функціями, архітектурою і якістю. Фабрики повинні мати продуктові лінії й відповідний набір засобів розробки, виробництва простих і складних ПП. Лінія ро-

зробки простих продуктів, як правило, відповідає ЖЦ, наприклад, реалізованому в середовищі MS.Net з використанням рекомендацій, каркасів (framework), DSL–мов (Domain Specific Language) та інших інструментів. Лінія виготовлення складних ПП фактично є збіркою готових програмних ресурсів, що знаходяться в різних бібліотеках і репозиторіях та відповідають критеріям їх відбору.

Виходячи з отриманої практики автоматизованого збирання різнорідних програм у мовах програмування (МП) в середовищі ОС ЄС [1–3] і досвіду сучасних закордонних фабрик програм з індустрії ПП (IBM, OMG, Microsoft, Oberon тощо) [4, 5, 7] сформувалися загальні складові зборки, що характеризують в основному всі відомі й майбутні фабрики програм:

– готові програмні ресурси (артефакти, програми, компоненти, системи, reuses, assets, КПВ) тощо;

– інтерфейс, специфікований паспортними даними готових різнорідних ресурсів, незалежно від МП, особливою мовою специфікації інтерфейсу (наприклад, IDL, API, SIDL, WSDL, RAS тощо) [8, 9];

– операційне середовище, насичене системними програмними засобами і інструментами для підтримки індустріальної збірки різнорідних ресурсів [4];

– технологічні лінії (ТЛ) або продуктові лінії (Product Lines) [10, 11] з виробництва ПП.

Ці складові були визначені нами і розвинуті в рамках фундаментальних проектів Інституту кібернетики (1980–1991) і ІПС НАН України (1992–2010). В них уперше визначено концепцію інтерфейсу (1982, [3, 8]), метод збірки різномовних компонентів попередили появу індустріальних складових програм, ТЛ (1987–1991, [1, 10]) та засоби автоматизації випуску ПП [2, 3]. Вони попередили появу складових з виробництва ПП закордонними фірмами, наприклад, IBM, Microsoft, Oberon та ін.

Далі дається формальне тлумачення наведених складових індустрії ПП, підходів до навчання IT–спеціалістів усім проблемам індустрії у цілях застосування студентів у сфері наукового софтвера.

1.1. Готові компоненти і їх інтерфейси. Програмні ресурси, які можуть використовуватися багаторазово, називають reuse, assets, КПВ, програми тощо. Вони відображають реалізацію різних прикладних або математичних функцій деякої наукової (фізика, математика, біологія тощо) або прикладної предметної області. Головна парадигма КПВ – «писати один раз, виконувати багато разів, де завгодно». Архітектура, в яку вбудовується готовий ресурс, використовує стандартні механізми роботи з ним, як із будівельними блоками. Щоб забезпечити високу ефективність повторного використання, вони мають бути якісними і надійними при виконанні.

Як готові ресурси можуть використовуватися формалізовані артефакти діяльності розробників програм, які відображають певну функціональність. Під *артефактом* розуміється реальна порція інформації з програмування, яка може створюватися, змінюватися і використовуватися у діяльності, пов’язаної з виготовленням ПП різного призначення. Артефактами можуть бути:

– модель предметної області (ПрО) зі своїми термінами, поняттями та лексикою;

– готові КПВ або окремі частини (фрагменти) систем;

– проміжні продукти процесу розроблення (вимоги, завдання, моделі та ін.);

– специфікації (ресурсу, інтерфейсу, моделі і т.п.) окремих елементів, діаграм, паспортних даних і т. п.

Артефакти незалежні від платформ комп’ютерів, мають опис інтерфейсів і різних параметрів для взаємодії з іншими ресурсами.

Усі ресурси та їхні інтерфейси зберігаються у сховищах (бібліотеках, репозиторіях) для подальшого пошуку іншими фахівцями з метою подальшого застосування. Тобто, повторне використання стає капіталомістким видом діяльності на фабрики ПП.

Студентська фабрика, на наш погляд, повинна мати три базові лінії.

Перша лінія як схема процесів стандартного ЖЦ з побудови деякого програмного ресурсу шляхом:

– вивчення завдань ПрО, виявлення серед них загальних властивостей і функцій та методів породження з них програмних елементів;

– специфікація цих елементів МП і паспортних даних їх інтерфейсів;

– зберігання ресурсу в репозиторію.

Реалізація процесів такої лінії закінчується визначенням сформованих артефактів або ресурсів у класі задач ПрО, специфікацією паспортних даних та розміщенням описів ресурсів у репозиторії для подальшого застосування. Ця лінія потребує вкладення капіталу в розробку наукових артефактів. Такого типу лінії розробляється на студентській фабриці у вигляді життєвого циклу створення програм у середовищі Visual Studio .Net (рис. 1).

Друга лінія – це механізми підбору готових ресурсів із репозиторію за їхніми функціями і відповідними критеріями з метою оцінки можливості їх застосування у деякої предметної області як готових елементів.

Третя лінія – збіркова лінія, що забезпечує конструювання нових ПП методом збірки з застосуванням розроблених і підібраних ресурсів за шляхами:

– визначення цілей і вимог до майбутньої ПП та до окремих елементів збірки;

– аналіз готових ресурсів, перевірка їх життєздатності і результату для оцінки їх потреби у нової системи;

– прийняття рішень про доцільність застосування ресурсу та його збереження у репозиторію;

– зборка ресурсів за їхніми інтерфейсами у ПП, тестування зв'язків та системи.

Ця лінія буде давати прибуток за рахунок заощадження трудовитрат від застосування готових артефактів та КПВ за оцінкою ефективності їх використання і обсягів повернення цього вкладення в новий ПП. Ресурс може бути науковим, прикладним і загальносистемним. Перший ресурс – це метод або науковий алгоритм (з математики, фізики, біології тощо), що описаний загальною МП; другий ресурс – реалізація окремої задачі або функції ПрО (бізнесу, комерції, економіки і т. п.); загальносистемний ресурс – це готові засоби, потрібні всім – транслятор, редактор текстів, генератор, інтегратор, завантажувач, сервіс з обміну даних, передач повідомлень тощо. Зміст даної лінії в основному відповідає лінії збірки (рис.1).

Інтерфейс програмних ресурсів. Під *інтерфейсом* розуміється зв'язок двох окремих сутностей. Інтерфейси є програмні, апаратні, мовні, користувальницькі, цифрові й т.п. Програмний (API) і/або апаратний інтерфейс (port) є способом перетворення вхідних/вихідних даних під час зв'язку комп'ютера з периферійним устаткуванням. У програмуванні інтерфейсом є програма або її частина, в якій визначаються дані (константи, змінні, параметри й структурні типи даних тощо) для передачі їх іншим програмам зі значеннями їхніх параметрів [8].

У ролі операцій інтерфейсу виступає виклик процедур і функцій у МП з завданням їх імен і списку формальних параметрів для передачі фактичних значень для виконання. Послідовність і число формальних параметрів мають відповідати фактичним параметрам. Коли програма, процедура або функція специфіковані різними МП і вони розташовані на різних комп'ютерах, то вирішується проблема неоднорідності поданих їх типів даних, відмінності архітектур комп'ютерів або операційних середовищ та несумісності параметрів за їх кількістю та порядком розташування.

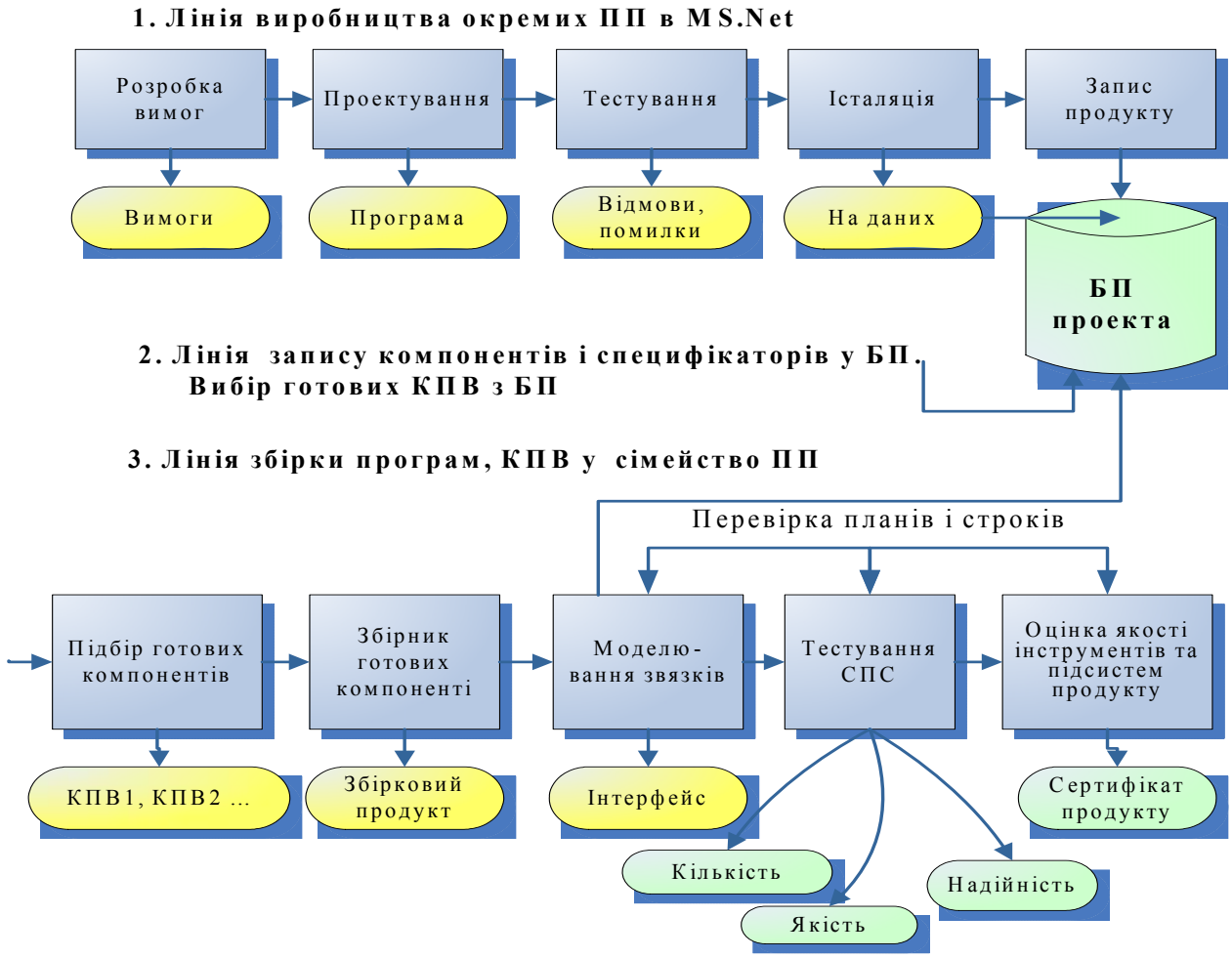

Рис. 1. Структура загальних ліній фабрики програм

Інтерфейс – фактично посередник або “перехідний мостік” між двома модулями. Він передає дані і виконує необхідне пряме й зворотне перетворення даних у випадку їх неоднорідності та невідповідності.

Кожний програмний ресурс має специфікуватися паспортними даними за деякими стандартами [12–14]:

– назва ресурсу;
– ID – ідентифікатор ресурсу;
– зміст програми (функції);
– параметри виклику інших ресурсів;
– інструменти підтримки виконання програмного ресурсу тощо.

Необов'язкові атрибути паспорту: дата, стан, версія, автор, дата створення, термін придатності і тому подібно. Інтерфейс може містити опис точок зі зв'язками з деякими послугами, точками для внесення змін і взаємодії ресурсів у середовищі та додавання деяких аспектів (наприклад, синхронізації, захисту) тощо. Специфікації інтерфейсів зберігаються у бібліотеці або репозитарію інтерфейсів для майбутньої збірки з них програм.

Для включення студентських програмних артефактів у репозитарій розроблено спеціальний шаблон специфікації інтерфейсів, позиції якого заповнюються студентами при запису кожного артефакту в деякий його розділ.

1.2. Операційне середовище з засобами і інструментами для збірки ПП. Виходячи з аналізу діючих фабрик програм [4], дамо класифікацію середовищ з залежності від типу взаємодії, що обирається для фабрики, як базове:

1) через брокер запитів між клієнтом і сервером, щодо отримання даних від stub клієнта, їх обробку під формати комп’ютера, передачу їх серверу і зворотно від skeleton сервера, що обробляє результати обчислювань до форматів даних клієнта;

2) через проміжний прошарок загальносистемних засобів, сучасних середовищ або віртуальних серверів хмарних обчислень [15, 16];

3) через пряму зборку трансльованих програм за різними МП і їх інтерфейсами у бібліотеках, наприклад, MS.Net.

Перший тип взаємодії практично забезпечується брокером ORB для класу МП у середовищі CORBA. Зборка компонентів базується на Client class з викликами інших, Stub class з конвертування даних, ORB class з передачі даних; Server class зі створення сервентів та посилання даних ORB; Skeleton class з конвертування форматів даних та породження різних видів серверів.

Другий тип забезпечує взаємодію між компонентами, розміщеними на різних комп'ютерах через проміжний прошарок (middleware) шляхом повідомлень (наприклад, Java Message Queue), віддаленого виклику процедур RPC в MS Windows, ONS IBM, CORBA та виклику методів RMI у Java. Модель ISO/OSI також забезпечує проміжну взаємодію на рівнях (фізичному, канальному, мережному і транспортному) мережної ОС. Верхній рівень моделі (прикладний) забезпечує перетворення даних до транспортного рівня шляхом їх маршалінгу й демаршалінгу за інтерфейсним stub. Сеансовий рівень моделі передає дані іншим рівням через транспортний канал за механізмами посилань. Технологія розподіленої обробки отримала новий віртуальний прошарок між Internet і засобами Cloud Computing [16], що подає сервіс з обробки даних великих розмірів у режимі online з віртуальних серверів Google Apps, Windows Cloud, Amazon Web–services, Sky–driven, Azure (www.cmswire.com).

Третій тип реалізований в MS.Net за допомогою таких бібліотек системи:

CLR (Common Language Runtime), CTS (Common Type System) і CLS (Common Language Specification). CLR забезпечує виявлення і завантаження даних, керування ними у відповідності до завдань розробника програми. CTS визначає принципи взаємодії із іншими, відповідно представлення форматів мета-даних. Збірка різномовних програм виконується засобами CLS бібліотеки. Усі компілятори з МП створюють модулі DLL або EXE у проміжній мові MSIL (Microsoft Intermediate Language) як механізм збірки програми незалежно від платформи, на якій вона буде виконуватися. При її запуску вона конвертується в специфічний код CPU, що виконується на різних архітектурах комп'ютерів.

Для побудови програм на студентській фабриці програм обрано Visual StudioMS.Net. У цієї системи є багато різних інструментів, зокрема, Веб-сервіси різного призначення, пакет VSTS для збірки і керування конвеєрним методом розробки програм, вимірювання й оцінювання їх якості.

Таким чином, у залежності від цілей фабрики програм як середовище вибирається те середовище, що найбільш підходить для організації процесу побудови і зборки відповідних програмних ресурсів у межах деякої ПрО. Коло проблем звужується, коли різні середовища можуть взаємодіяти між собою, створюючи одне велике, гнучне середовище з поширеними можливостями щодо виробництва кінцевого ПП.

1.3. Структура і зміст ліній виробництва ПП. Підхід до побудови ТЛ. Після аналізу ПрО й виявлення її основних функцій будується деяка загальна ТЛ за процесами ЖЦ. Для ній підбираються готові програмні ресурси, засоби й інструменти породження із функцій програм та планування і контролю процесами щодо формування шаблонів фіксації проектних рішень, зміни станів і оцінювання якості ПП [3, 7, 10].

Процеси ТЛ будуються з урахуванням міжнародного стандарту ISO/IEC 12207–96, 2007 і ДСТУ 3918–99, підкріплюються відібраними методами, засобами й інструментами для здійснення змін станів об'єктів на процесах. Цей підхід до побудови ТЛ апробований в АІС «Юпітер» (1987–1991). На новому витку історії розвитку індустрії виникли *продуктові лінії* [11], принципи побудови й виконання яких аналогічні ТЛ, що слугують індустрії виробництва студентських програмних артефактів за лініями, наведеними на рис. 1.

На ньому лінія 1 ініціює первинний процес з виробництва ПП із готових ресурсів з метою задоволення потреб деякого фрагменту ринка ПП. Лінія 2 призначена для накопичення КПВ з паспортними даними (специфікаторами) у бібліотеки і спрощення їх пошуку з метою використання на лінії 3 зборки з них ПП.

На кожної лінії враховуються:

– умови та обмеження ресурсів;

– шаблони та набір готових ресурсів (КПВ, reuses, assets тощо);

– стратегії та МП;

– необхідні засоби і інструменти;

– плани робіт з використання технічних ресурсів;

– методики виміру та оцінювання показників ПП якості;

– сертифікат ПП.

Наведені на рис. 1 типи ліній виробництва ПП для фабрики не обмежені. Деяка фабрика може бути зорієнтована на спеціальні лінії функціонального типу, наприклад, на створення програм із класу задач статистичної обробки, деяких чисельних методів, принципи розробки яких було подано в [2, 4, 10].

1.4. Обслуговування фабрики. До складу фабрики входять групи розробників і спеціалістів сервісного обслуговування, що наведені в таблиці. Крім цих спеціалістів можуть бути й інші, а також технічний персонал, що запускає інструменти і засоби підтримки ТЛ. Керівництво програмними проектами і ресурсами на фабрики виконує менеджер або директор.

Таблиця. Персоналу фабрики

Розробники	Обслуговуючий персонал
Аналітики, програмісти, тестери, верифікатори, експерти, вимірники якості	Плановики, економісти, бухгалтери, контролери, оцінювачі процесів і ПП, сертифікатори, пакувальники ПП

2. Нові індустріальні підходи щодо обчислення наукових задач

Нові індустріальні підходи з'явилися у зв'язку з бурним розвитком високоефективної елементної бази, нової багатоядерної, процесорної конфігурації комп'ютерів. Це сприяє великомасштабним обчисленням дуже складних задач сучасності в e-sciences (геології, біології, фізики, математики і др.), АСУ і інших галузях промисловості. Комп'ютерна індустрія більш ніж на порядок опереджає розвиток як з теоретичної, так і з практичної точки зору інші види індустрії, а саме, індустрія обчислень, систем і програм.

Індустрія обчислень у теоретичному плані іде за шляхом застосування нових комп'ютерних можливостей щодо швидкості дій і розподілення пам'яті для обчислення задач з великими обсягами даних. Вона отримала також новітні базові засоби з організації прозорих обчислень різного роду складних задач за рахунок розвитку нових моделей: *взаємодії* OSI, *генерації* GDM, *архітектури* SOA,MDA, *розробки* DDM тощо, так званих, *хмарних обчислень* (Cloud Computing, SkyDriven), що підтримуються Веб-стандартом HTML5 з високо пропускними протоколами доступу до загальних on-line сховищ даних з деякого місця нашої планети. Архітектура системи за моделлю MDD (Model Driven Development) моделюється на двох рівнях – платформи незалежного рівня PIM (Platform Independent Model) і платформи залежного рівня PSM (Platform Specific Models). Концепція дворівневого моделювання архітектури MDA (Model Driven Architecture) і відображення PIM→PSM відповідає ідеології побудови сімейства систем СПС мовою DSL для опису понять і задач з простору проблем предметної області [4, 9].

Одним з практичних рішень забезпечення *індустрії обчислень* є поява системи Grid у 2006 р. в межах Європейського проекту. Цю систему було розроблено як інфраструктуру для глобальних обчислень суперскладних задач з області e-science. Для обчислень таких задач застосовуються знов розроблені або готові програмні й системні ресурси виду: розподілені процесорні потужності; системи сховищ даних; новітні засоби комунікації; фонди reuses з багатьох доменів; нові технічні устаткування і загальні «тули» (кон-

вертори, генератори, трансформатори, верифікатори тощо); моделі та схеми взаємодії програм у середовищі гетерогенних платформ, географічно розташованих у віддалених адміністративних доменах тощо. Середовище *Grid* як інфраструктура обчислень різних наукових задач пропонує підсистему ETICS для організації зборки різнорідних і різноплатформених програм рішення задач за їх описом МП або мовою DSL (рис. 2).

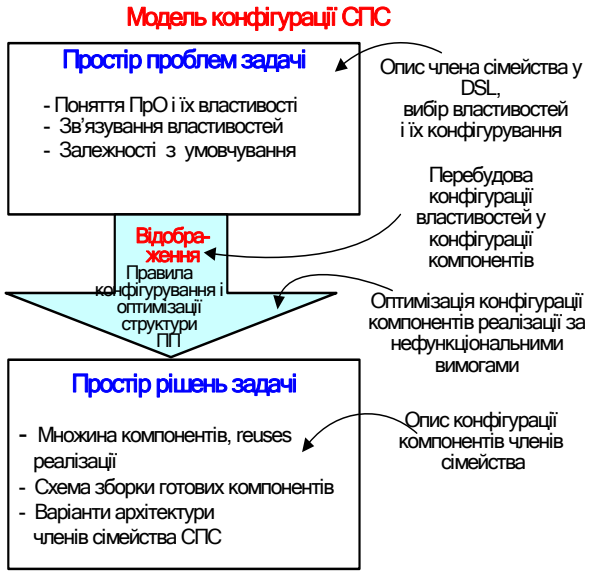

Рис. 2. Модель завдання опису і рішення задач

Завдання з обчислення задач виконуються із застосуванням різних системних і прикладних сервісів Інтернету і досягнення максимальної продуктивності рішення задач глобального масштабу [15, 16].

Іншим напрямом практичного обчислення задач глобального типу є *хмарні обчислення,* які підтримуються новими засобами Cloud Computing системами IBM-VSphere та системами Microsoft – WCloud, Azure, Amazon, Mech, WApps, SkyDriven (Http://lenta.ru/articles/2010).

Вони зорієнтовані на збереження даних, доступ до глобальних сховищ даних on-line, синхронізацію даних великих розмірів і викладання їх на віддалені сервери тощо. Тут головна проблема організації обчислень потребує визначення нових методів, таких як координація, кооперація та взаємодія різних сервісів і інших готових ресурсів через конфігураційний файл для виконання обчислень відповідних задач.

Накопичення готових ресурсів у різних глобальних сховищах для доступу до них різних наукових користувачів потребує розвитку індустріальних методів і моделей з урахуванням особливостей і специфіки наукових задач. Індустрія ПП, концепція якої подано у попередніх розділах роботи, теж базується на готових звичайних компонентах (артефактах, reuses, assets і даних) багаторазового використання, що знаходяться у різних бібліотеках, репозиторіях та нових «хмарних» сховищах Інтернету. Головним методом індустрії наукових продуктів буде удосконалений метод зборки наукових різнорідних ресурсів, що належить до напрямів e-science, в структури ПП, котрі після їхнього виготовлення можуть бути подані на різні глобальні сервери для їх застосування при рішенні відповідних наукових задач.

Іншим індустріальним підходом є *модельний підхід*, якій широко використовується нині в програмної інженерії. Нами запропоновано набор моделей в межах фундаментального проекту ІПС НАН України (ІІІ–1–07 “Розробка теоретичних основ генеруючого програмування (ГП) та інструментальних засобів його підтримки” (2007–2011) [17] для організації процесів з виробництва і виконання ПП. Цей набір включає: *модель взаємодії, модель варіабельності і модель життєздатності*.

У межах проекту розроблено їх зміст і місце в СПС, а також методи і засоби для їх використання на сучасних фабриках програм.

Розглянемо сутність цих моделей у просторі індустріальних проблем сучасного наукового софтвера.

Модель взаємодії або інтероперабельності призначена для обміну інформацією між різними компонентами при організації по них обчислень [4, 9]. Взаємодія це інтерфейс зв'язків різномовних і різноплатформних програм, а саме посередник у мові MIL (Module Interconnection Language) або IDL (Inter-face Definition Language). Нині він реалізований різними способами у системах Windows Server,

Microsoft.Net, IBM Web Sphere і т.п. Головні поняття інтерфейсу – фундаментальні типи даних, що специфікуються при описі простих і структурних даних компонентів, що об'єднаються. Модель взаємодії найбільш пов'язана з використанням у процесі розробки на фабриках нових програмних систем з готових програм, за допомогою яких вирішуються різного роду задачі зборки. Організація обчислень з зібраних програм буде більш ефективною, якщо реалізований інтерфейс враховує формати даних платформ сучасних комп'ютерів. Реально існуючі розходження в апаратній частині платформ відображаються у вихідному коді систем програмування з МП, у конфігураційному файлі готових ПС, а також у принципах забезпечення механізмів взаємодії програм у сучасних обчислювальних або гетерогенних середовищах.

Розвиток нових технічних засобів (майнфреймів, кластерів, суперкомп'ютерів й ін.) породжує нові структури даних (графи, контейнери, портфелі й ін.) та відповідні формалізми генерації загальних типів даних (стандарт ISO/IEC 11404–2007 General Data Types) до фундаментальних. Деякі питання їх перебудови підтримуються спеціальними засобами середовищ (Sun IBM, Microsofts, CORBA, COM, JAVA і ін.) шляхом побудови проміжного прошарку (stub, skeleton) брокерного типу або інтеграції вихідного коду програм у системах Linux, Windows Server, MS.Net, IBM Web Sphere і т. п.

Модель варіабельності подається у проекті ІІІ–1–07 як поточна або прогнозована структура СПС, що забезпечує необхідні зміни та додавання нових можливостей у складну СПС. Тобто варіабельність визначає здатність сімейства ПС, чи окремої її підсистеми або артефакту до розширення, змінювання, пристосування до інших умов конфігурування її компонентів у моделі обчислень конкретній ПрО [18, 19]. Варіабельність забезпечується на рівні подання вимог, побудови моделі характеристик СПС, архітектури, коду, тестів тощо. Варіабельність може бути побудована засобами продуктової лінії, за якою будується сімейство, як множина ПС або інших готових ресурсів, що містяться у репозиторії системи або Інтернету. Варіабельність складових ПС забезпечує не тільки здатність окремої ПС до еволюції після її породження і відділення від складу СПС, але і життєздатність ПС та СПС у цілому.

Основні об'єкти цієї моделі такі:

– точки варіантності у формальному поданні для деяких ПС сімейств;

– варіант як елементарний артефакт СПС деякого типу;

– предикат, що визначає множину точок варіантності та їх варіантів для СПС;

– предикат припустимості взаємозв'язків між точками варіантності і множиною варіантів.

Модель варіабельності моделюється за допомогою діаграм UML або іншими засобами продуктової лінії, до якої додається наступне:

– план її реалізації за допомогою артефактах СПС;

– опис та її реалізація для СПС з фіксацією у файлі конфігурації;

– відповідність моделі в структурі СПС, файлі конфігурації СПС та параметрах її настроювання на виконання деяких функцій СПС.

Концепція варіабельності щодо СПС у системі генерувального програмування проекту ІІІ–1–07 розроблено фахівцями відділу Коваль Г.І, Слабоспицької О.О. та аспірантом Колесник А.П. [18, 19].

Модель життєздатності [20, 21] розробляється для забезпечення змін й корегування проектних характеристик архітектури системи з використанням теорії живучості систем LST (Living System Theory) [22] з моделюванням архітектури СПС засобами MDA. Сутність теорії LST міститься у інтегрованому концептуальному процесі аналізу системи LSPA (Living Systems Process Analysis) для забезпечення життєздатності майбутній СПС в процесі її супроводу. Основні положення цієї теорії застосовані у модель життєздатності для сімейства систем. Модель VSM (Viable System Model) удосконалюється додаванням параметрів функціонування СПС та деякими показниками стандартної моделі якості – адаптивність, змін-

ність, реактивність тощо, які ініціюють коригування діючій системи у випадку виникнення нерегулярних ситуацій при функціонуванні системи. Тобто вхідні параметри та відповідні характеристики моделі життєздатності адаптуються для продовження роботи системи. Для цього модель має бути відображена у конфігураційному файлі СПС, яка керує розгортанням і виконанням окремих її членів.

Запропоновані у роботі моделі взаємодії, варіабельності і життєздатності мають реалізуватися в рамках фундамент-тального проекту з ГП. Вони є спробою вперше забезпечити моделювання програм СПС в процесі динаміки їх виконання у середовищі Eclipce. Головним механізмом переходу від опису моделей до вихідного результату є трансформація понять ПрО до проміжних мов простору рішень (рис. 3), та відображення їх характеристик у наведених моделях та конфігураційному файлі СПС.

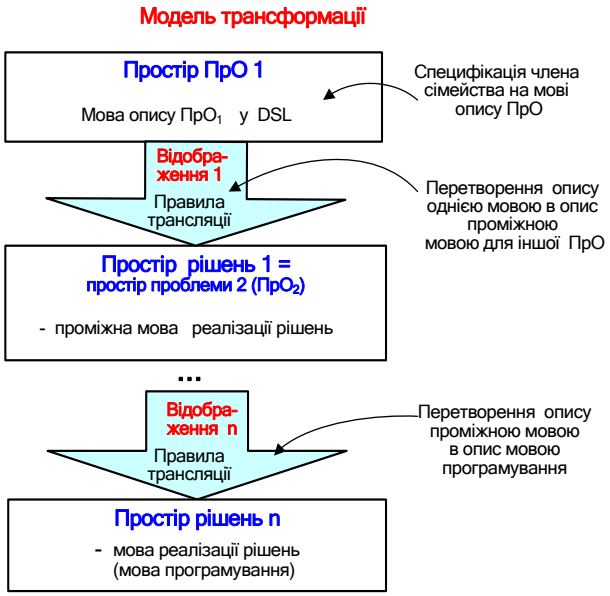

Рис. 3. Схема трансформації опису домену

Основною вимогою до нових моделей варіабельності і життєздатності є створення їх такими, щоб їх можливо було використати при супроводі СПС під час їх виконання, проведення необхідних змін у різні частини системи, корегування параметрів цих моделей у зв'язку з різними відмовами та продовження функціо-нування системи з зафіксованого варіанта або з деякої іншої точки моделі варіабельності СПС.

3. Концепція навчання студентів з метою їх участі в індустрії софтвера

Більше трьох років автор викладає курс лекцій в КНУ ім. Тараса Шевченка по курсу "програмна інженерія" відповідно до міжнародної програми Curricula-2004. Ця програма не містить дисциплін, пов'язаних з виробництвом програмної продукції на фабриках програм, принципами комплектації фабрики технічними і людськими ресурсами, методами створення продуктових ліній та служб підтримки засобів виготовлення та керування фахівцями на фабрики ПП.

3.1. Дисципліни для навчання індустрії виробництва програм. Індустрія ПП за продуктовими лініями має ґрунтуватися на дисциплінах, які забезпечують нормативне і регламентоване виконання різних робіт з розробки, зборки та керування роботами щодо експертиз, вимірів і оцінки різних артефактів.

До дисциплін навчання віднесені такі науково-технічні дисципліни (*Di*) програмної інженерії (ПІ) [7, 23–25]:

$$ПІ = \{DiSc, DiEn, DiEc, DsMa\},$$

де *DiSc* – наукова, *DiEn* – інженерна, *DiEc* – економічна, *DiMa* – менеджерська (управлінська) дисципліни.

Сутність цих дисциплін ПІ наведено у [24, 25], а короткий їх огляд дивитися далі.

Наукова дисципліна є теоретичним фундаментом ПІ і навчати їй необхідно не тільки для підвищення рівня кваліфікації майбутніх фахівців з програмної інженерії, але і для підтримки та розвитку нових можливостей і засобів програмування, які удосконалюють відповідні напрямки індустрії ПІ. Однією з важливих наукових проблем індустріального виробництва ПП є інтеграція (композиція, інтеграція) складових елементів майбутнього продукту за їх інтерфейсами. З урахуванням новітніх можливостей сучасних середовищ необхідно удосконалити теорію композиції різнорідних програм [2, 7].

Головним напрямом навчання у вищих навчальних закладах має стати саме наукова дисципліна, як теоретичний курс з

програмування (об'єктно-орієнтованого, компонентного, сервісного тощо) поряд з діючими класичними курсами та додатковими курсами, наприклад, з теорії моделей.

Інженерна дисципліна – це сукупність інженерних прийомів, засобів і стандартів, орієнтованих на інженерне виготовлення ПП із застосуванням наукової дисципліни ПІ. До них належить:

– ядро знань SWEBOK (www.swebok.org) з розділами побудови та керування програмними проектами;

– базовий процес з організації процесної діяльності на фабрики ПП;

– стандарти з регламентованими правилами конструювання артефактів на процесах ЖЦ;

– умови середовища з забезпечення базового процесу і підтримки дій виконавців з вироблення ПП на ЖЦ;

– загальні засоби і інструментальні середовища підтримки виготовлення ПП.

Склалися різновиди інженерної дисципліни: інженерія КПВ, застосувань (Application Engineering), доменів (Domain Engineering), сімейств систем (Family Engineering) [2, 7]. Усі ці інженерії ґрунтуються на багаторазовому використанні різнотипних КПВ. Побудова СПС пов'язана з розв'язанням задач домену, загальними і змінюваними характеристиками програм сімейства. Технологія автоматизації СПС набуває конвеєрного виробництва із КПВ, в основі якого лежить опис моделі домену в DSL, специфікація членів сімейства МП, керування планами робіт, контролю результатів та оцінювання рівня застосування готових ресурсів при реалізації різних задач. Тобто, без інженерії не мислиться побудова жодного промислового продукту.

Дисципліна керування. Базисом цієї дисципліни є класична теорія керування складними системами, сучасний менеджмент проекту та відповідний стандарт IEEE Std.1490 – настанова до ядра знань менеджменту PMBOK (Project Management Body of Knowledge). Ця теорія отримала розвиток у нас і закордоном, особливо в частині планування виробництвом. За теорією планування Глушкова побудова телевізорів на Львівському заводі значно було підвищено протягом декілька років. Метод CRM (Critical Path Method) з графічним поданням робіт, різних видів операцій та часу їхнього виконання (на фірмі «Dupon») та техніка планування PERT (Program Evaluation and Review Technique), що розроблені у надрах промислового виробництва, на даний час адаптовані до менеджменту ПП [7].

Стандарт з менеджменту – PMBOK відображає задачі планування, моніторингу і керування програмними проектами. В ньому концепція керування організаційною діяльністю виконавців проекту забезпечується методами прийняття рішень, контролем правильності проекту та оцінки ризиків [2].

Базові теорії керування, стандартні положення PMBOK, серія стандартів ISO–9001 з якості та відповідне методичне забезпечення мають стати основою дисципліни керування ПІ. Створений з цих питань курс навчання буде слугувати підготовки у Вузах майбутніх висококваліфікованих менеджерів проектів та інших фахівців з організаційного керування ПП.

Економічна дисципліна має стати самостійною дисципліною ПІ зі своєю теорією і практикою оцінювання вартісних, часових і експертних показників щодо зборки ПП з готових ресурсів, прийняття проектних рішень, подання вимог, розроблення архітектури, визначення ризиків проектування за наявними ресурсами, проведення розрахунків за роботи виконавців та отриману ними якість ПП. Ця дисципліна є найбільш розвинутою з точки зору наявності методів економічних розрахунків у ПІ, а саме методологій прогнозування розміру ПП (FPA – Function Points Analyses, Feature Points, Mark–H Function Points, 3D Function Points тощо), оцінювання витрат на ПП за допомогою сімейства моделей COCOMO або інших математичних моделей (Angel, Slim, Seer тощо) [5, 26].

При визначенні цієї дисципліни необхідно використати фундаментальні економічні методи, пов'язані з принципами розподілу робіт у складних системах, методи розрахунків вартості окремих частин

систем залежно від розміру і системи у цілому, існуючі стандарти щодо оцінювання ПП тощо. Систематизований і науково обґрунтований курс економічної дисципліни ПІ закриє економічну прогалину, яка нині існує в індустрії ПП.

Виробнича дисципліна включає методи побудови ТЛ**,** продуктових ліній**,** технологію виготовлення продукту на цих лініях з застосуванням відповідних теорій проектування та інструментів операційного середовища побудови ПП масового використання. За останні роки у нас в основному не розробляються такі лінії і вітчизняні інструменти для виготовлення різних видів ПП. Найбільш апробований в Україні аутсорсинг готових систем і засобів становить більше 35 % від загального обсягу робіт з індустрії ПП. Але виникають труднощі при супроводженні готових програм, які вирішуються за допомогою новітніх методів реінженерії і реверсної інженерії [7, 23]. При цьому залишаються деякі проблеми оцінювання складності об'єктів і процесів виготовлених ПП для їхній зміни і перебудови.

Ця дисципліна, як предмет навчання, включає класичні методи і технології виготовлення ліній та виробництва за ними різних ПП, методи аналізу особливостей вибору готових ресурсів, стандартів виробництва та документування продукту.

Наведеними дисциплінами ПІ мають володіти учасники процесів виробництва ПП (аналітики – Scientists, інженери – Engineers, економісти – Economists, керівники – Managers тощо) [2, 7].

Виробнича діяльність (*PA* – production activity) цих спеціалістів на процесах лінії задається кортежем:

$$PA = < ТЛ, Ap, Rp >,$$

де $ТЛ = \{Tp \cup CP\}$, *Tp* – навчальний і управлінський процес *CP* (Control process); $Ap = \{DiSi \cup DiEn \cup DiEc \cup DiMa\}$ – дисципліни програмної інженерії; *Rp* – кваліфікаційні вимоги до фахівців ліній з виробництва ПП.

Усі вищерозглянуті базові теорії і дисципліни ПІ, а також методи і технології побудови ліній і ПП, оброблення даних і організації обчислювань в сучасних середовищах є базові для навчання студентів Вузів з інформатики для майбутнього розвитку індустрії ПП.

3.2. Про навчання студентів проблемам генерації типів даних для on-line сховищ. Основним видом практичної діяльності студентів при навчанні дисциплінам, орієнтованим на індустрію ПІ є лабораторні і дипломні роботи по створенню деяких «тулів» (конструкторів, збиральників, конвекторів програм і даних, генераторів тощо), бібліотек примітивів з перебудови типів даних, що подаються у інтерфейсах ресурсів, які можуть не відповідати форматам платформ комп'ютерів або бути відсутніми серед фундаментальних (FDT) типів МП. У роботі [4] запропоновано нове вирішення задачі перетворення типів даних різних ресурсів Інтернету у вигляді оригінальної системи генерації загальних типів даних (ТД) GDT до FDT, що є у сучасних МП, на яких специфікуються програми для фізичних, біологічних та інших наукових завдань і експериментів. Базовим фундаментом цього рішення є стандарт ISO/IEC 11404–2007 – General Data Types, який визначає генерацію загальних типів даних до фундаментальних.

Студентам 4 курсу кафедри ІС факультету кібернетики викладалась лекція щодо ТД GDT і FDT поза програмою дисципліни. Зокрема, декілька студентів переклали цей стандарт, як лабораторні роботи, зробили наукові доповіді про ТД, сутність ідеї генерації загальних ТД до більш простих, що є в МП. Студенти Є. Забєлін і І. Чорний захистили бакалаврські роботи щодо завдань з перетворення деяких ТД вказаного стандарту (2010 р.).

Основні задачі і підхід до генерації GDT<=>FDT запропоновані в [4] і рекомендуються для реалізації у системі Grid. Значимість задач генерації даних під нові платформи, гетерогенні середовища, Cloud Computing зростає у зв'язку зі створенням Інтернет on-line сховищ даних і їх застосування при обчисленні складних наукових задач глобального типу. При цьому перебудова даних є дуже важливою і потребує розроблення спеціального набору примітивів (функцій), які будуть використовуватися як в індустрії розроблення ПП, так і в глобальних обчисленнях.

Вирішення деяких задач генерації ТД виконується в межах фундаментального проекту з генерувального програмування [17] і орієнтовані на включення примітивів щодо оброблення сховищ даних.

Висновок

Запропонована в роботі концепція побудови фабрики софтвера є загальною. Вона містить базові складові за конвеєрною зборкою – лінії, бібліотеки програмних ресурсів і інтерфейсів, операційне середовище. Розглянуті основні питання індустрії комп'ютерів і пов'язаної з ними індустрії ПП і обчислень, а саме модельний підхід, включаючи моделі взаємодії, варіабельності та життєздатності, які визначені в межах фундаментального проекту III–1–07.

Підкреслимо, що концепція фабрики програм почала реалізуватися в КНУ ім. Тараса Шевченка на факультеті кібернетики. Базовими артефактами фабрики є дипломні і аспірантські роботи з методів і алгоритмів наукових кафедр цього факультету. Прикладом побудови студентської фабрики є створення сайту (http://www.programfactory.unv.kiev.ua), що демонструє декілька технологій розроблення програмних артефактів за ЖЦ засобами Visual Studio.Net, додавання нових КПВ, інших готових програмних ресурсів в бібліотеку сайту з описом стандартно прийнятих специфікацій та паспортів інтерфейсів.

1. *Лаврищева Е.М.* Сборочное программирование. Теория и практика // Кибернетика и системный анализ.– 2009.– № 6. – С. 3–12.
2. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии программных продуктов.– Киев.: Наук. думка, 2009. – 371 с.
3. *Лаврищева Е.М.* Становление и развитие модульно-компонентной инженерии программирования в Украине // Препринт 2008. – 1.– Институт кибернетики имени В.М. Глушкова, 33 с.
4. *Андон П.І., Лавріщева К.М.* Розвиток фабрик програм в інформаційному світі // Вісник НАН України. – 2010. – № 10. – С. 15–41.
5. *Гринфильд Дж.* Фабрики разработки программ. – М., СПб., К.: Изд. дом «Вильямс», 2007. – 591 с.
6. Словарь по кибернетике. – Под ред. академика Глушкова, Украинская энциклопедия, Киев.– 1979. – 620 с.
7. *Лавріщева К.М.* Програмна інженерія.– Академперіодика. – 2008. – 319 с.
8. *Лаврищева Е.М.* Интерфейс в программировании // Проблеми програмування.– 2007. – № 2. – С. 126–139.
9. *Лаврищева Е.М.* Проблема интероперабельности разнородных объектов, компонентов и систем. Подходы к ее решению // Матер. 7 Міжнар. конф. з програмування "УкрПрог–2008". – С. 28–41.
10. *Лаврищева Е.М.* Основы технологической подготовки разработки прикладных программ СОД // Препринт 1987. –АН УССР, Институт кибернетики им. В.М. Глушкова. – 30 с.
11. Norhrop L.M. Software SEI's Product line Tenets // IEEE Software. – 2002. – V. 19. – N 4.– P. 32–39.
12. http://www.w3.org/
13. Reusable Asset Specifications (RAS) OMG Available Specifications Version 2.2., Date: November 2005: http://www.omg.org
14. Semantic Annotations for WSDL and XML Schema. W3C Recommendation. http://www.w3.org/TR/sawsdl/
15. *Таковицкий О.* Технология Grid computing // Byte. – 2003. – № 7(59). – P. 1 – 9. Available at http://www.bytemag.ru/articles/
16. *Ильин В.А.* Сетка с облаками для интернета.– В мире науки.– 2010.– С. 83–85.
17. *Лавріщева К.М.* Генерувальне програмування програмних систем і сімейств // Проблеми програмування. – 2009.– № 1. – С. 3–16.
18. *Коваль Г.І., Колесник А.Л., Лавріщева К.М., Слабоспицька О.О.* Удосконалення процесу розроблення сімейств програмних систем елементами гнучких методологій // Проблеми програмування (Спецвипуск конф. УкрПрог–2010). – 2010. – № 2-3. – С. 261 – 270.
19. *Колесник А.Л.* Механізми забезпечення варіабельності в сімействах програмних систем // Проблеми програмування. – 2010. – № 1. – С. 35 – 44.
20. *Ігнатенко П.П.* Життєздатні програмні системи. Концептуалізація підходу до автоматизації систем організаційного керування // Проблеми програмування. – 2006. – № 3. – С. 33–44.

21. *Ігнатенко П.П., Бистров В.М.* Особливості забезпечення життєздатності програмних систем в умовах генеруючого програмування // Проблеми програмування. – 2008. – № 2–3. – С. 270–278.
22. *Miller J.G. Living Systems*.–1995, Niwot, Colorado: University Press of Colorado. – 1102 p.
23. *Основы* инженерии качества программных систем / *Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, Е.М. Лаврищева, В.Ю. Суслов.* 2–е изд. – Киев: Академпериодика, 2007. – 672 с.
24. *Лавріщева К.М.* Перспективні дисципліни програмної інженерії // Вісник НАН України. – 2008. – № 9. – С. 12–17.
25. *Лаврищева Е.М.* Классификация дисциплин программной инженерии // Кибернетика и системный анализ. – 2008. – № 6. – С. 3–9.
26. *Лаврищева Е.М., Слабоспицькая О.А.* Подход к экспертному оцениванию в программной инженерии // Кибернетика и системный анализ. – 2009. – № 4. – С. 151–168.

Отримано 18.12.2010

Про автора:

Лавріщева Катерина Михайлівна,
доктор фізико-математичних наук,
професор, завідуюча відділом.

Місце роботи автора:

Институт программных систем
НАН Украины,
03680, Киев-187,
Проспект Академика Глушкова, 40.
Тел.: (044) 526 3470.