УДК 04.004

БАЗОВІ ОСНОВИ ІНДУСТРІЇ ПРОГРАМ, ОБЧИСЛЕНЬ І ДАНИХ

К.М. Лавріщева

Інститут програмних систем НАН України
03187, Київ, проспект Академіка Глушкова, 40,
тел.: 526 3470, e–mail: Lavryscheva@gmail.com

Дано опис підходів з розвитку індустрії виробництва програм із компонентів повторного використання (КПВ). Запропоновані головні елементи – теоретичні аспекти індустрії програм, обчислень та даних. Розглянуто підхід до реалізації теоретичних аспектів щодо принципів interoperability взаємодії програм, систем і середовищ у сучасних операційних середовищах (Corba,Vs.Net, Java, Eclipse тощо). За новітніми ідеями створено першу фабрику програм, яка орієнтована на розроблення і збереження студентських програм і артефактів у стандартизованому виді з їхніх дипломних і магістрівських предметних робіт. На фабрики діє побудовані три технологічні лінії за набутою технологією автора, зокрема для подання КПВ і навчання студентів основам програмної інженерії. До нової фабрики звернулися вже більш 3000 користувачів.

Description of approaches is given from development of industry of production of the programs from the components of the repeated use – reuse. Staples – theoretical aspects of industry of the programs, computing and information are offered. Going is considered near realization of theoretical aspects in relation to principles of interoperability co-operation of the programs, systems and environments, in modern operating environments (Corba,vs.Net, Java, Eclipse and others like that). After the newest ideas the first factory of the programs, which is oriented to development and maintenance programs and artifact in the standardized kind from their diploma and magistrivski of subject works is created. On factories the built three technological lines operate on the purchased technology of author, in particular for presentation of reuses and studies of student's bases of the programmatic engineering. To the new factory 3000 users appealed already more.

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 містить значно більший процент. Ця програма ставить задачу підвищення індустрії ПП шляхом консолідації наукових і підготовлених в Вузах спеціалістів і інвестування нових розробок різних видів наукових і промислових продуктів, потрібних державі.

Ідею індустрії комп'ютерів і ПП сформулював академік В.М. Глушков в Інституті кібернетики в 60–70 роках 20 сторіччя. За його ініціативою розвиток комп'ютерної індустрії ознаменувався побудовою низки комп'ютерів (Днепр–1, Днепр–2, серії машин «Мир» тощо), створенням фондів алгоритмів і програм (1972 р.), постановою про ПП з промислове технічним призначенням (1982 р.) та державним інвестуванням науково-дослідних інститутів усіх союзних республік ГКНТ СРСР, спрямованих на автоматизацію засобів виробництва ПП (1980–1990рр.) та на обговорення різних шляхів до індустрії на всесоюзних конференціях (1980–1991рр.) тощо. В результаті створено не тільки засоби автоматизації ПП, але і конкретні системи АСУ, АСУ ТП для різних галузей промисловості шляхом використання готових програм із фондів [1, 2]. Перша фабрика (або програмо будівельний інститут у Калініні – 1982 р.) проіснувала біля двох років. На той час не було визначено формального базису індустрії таких ПП, зокрема, інтерфейсів готових програм, їх специфікації і вдосконалення якості для автоматизованого збирання в ПП.

Нині склалися всі необхідні умови, а саме, накопичено велику кількість якісних різноманітних програмних ресурсів у електронних бібліотеках, зокрема, наукових, як елементів індустрії наукового софтвера, дуже потрібних Європейському проекту Grid, діють конкретні фабрики програм системного і прикладного характеру та сформувалися базові основи індустріального виробництва ПП [3, 4]. Перехід до індустрії програм і систем пов'язаний також бурним розвитком високоефективної елементної бази, нової багатоядерної, процесорної конфігурації комп'ютерів, кластерів тощо. Тобто комп'ютерна індустрія більш ніж на порядок опереджає розвиток як з теоретичної, так і з практичної точки зору інші види індустрії, а саме, індустрію програм, обчислень і даних. Така індустрія потрібна для виконання великомасштабних обчислень дуже складних задач сучасності в e-sciences (фізики, математики та ін.), АСУ й інших галузей промисловості.

У роботі автором розглянуто базові основи індустрії програм, як механізмів фабрик програм, які зорієнтовані на індустріальне вирішення названих математичних задач наукового софтвера, що розробляються на кафедрах університетів та науково-дослідних інститутів України і зможуть бути використані при обчислюванні глобальних завдань. Наукові артефакти, як програмні ресурси, будуть накопичуватися на фабриках відповідно міжнародних стандартів їх специфікації, і стануть сховищами «заготовок» наукової продукції у галузі інформатики та Computer Science.

1. Сутність індустріального підходу

Термін індустрія визначає виробництво різних видів продуктів масового застосування і засобів їх вироблення промисловими підприємствами, фірмами та корпораціями, в нашому випадку – програмно-технологічного типу. Головним питанням деякої індустрії є не тільки випуск відповідної продукції, але й отримання прибутку від цього. Нині великі прибутки від випуску програмної продукції отримують такі світові фірми, як Microsoft, IBM, Corba, Intel та ін., а також індійські фірми по оновленню старіючих наслідуваних (legase) систем програм. У нас за останні десятиріччя розвиток індустрії ПП фактично відсутній, немає державної і наукової програми її підтримки. Одночасно діють невеличкі підприємства комерційного типу, які виконують роботи по виробленню не масових ПП на замовлення різних фірм, що фінансують їх. У деяких наукових інститутах НАН України проводяться дослідження щодо принципів індустрії ПП, а в університетах ВНЗ цим теоретичним і прикладним аспектам індустрії не навчають (не має окремих навчальних курсів в Міносвіті) і тому можу констатувати, що наша держава відстає від світового рівня розвитку індустрії ПП більш ніж на 20 років.

Під ***виробництвом*** **ПП** розуміються процеси, за допомогою яких виконавці використовують відповідну теорію і інструментальні засоби для вироблення ПП масового застосування. Ці лінії із процесів, орієнтовані на розроблення ПП. Прикладом є розвинуті технології – Engineering application, family, domain, а також сформовані засоби оновлення застарілих систем в офшорних організаціях (наприклад, в Індії, Росії), Нині для підтримки процесів виготовлення ПП використовуються системні сервісні засоби (ОС, загальносистемні сервіси горизонтального і вертикального типу, нові мови, транслятори, редактори, компоузери тощо), готові програмні ресурси – ГОР, КПВ та методики проведення робіт, до яких належать запропоновані нами *дисциплін програмної інженерії (ПІ),* які визначають різні аспекти діяльності по виробництву ПП, а саме наукові, інженерні, управлінські, економічні, виробничі. Вони потрібні при виробництві ПП й участі кваліфікованих фахових спеціалістів для виготовлення ПП і розробки нових засобів виробництва для фабрики програм [7–9].

Сутність виробництва програмної продукції. Виробництво ПП – це процес створення матеріальних благ, необхідних для задоволення індивідуальних і соціальних потреб деяких прошарків інформаційного суспільства. Одним з найважливіших таких ресурсів є готові програмні артефакти, алгоритми і компоненти багаторазового використання (КПВ, сервіси, reuses, assets і т. п.) [5], які розробляються у світі різними фахівцями цього суспільства.

Від обсягу використаних ресурсів залежить рівень виробництва деякої продукції і обчислюється за загальною функцією виду: $v = F(z, u)$,

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 статечна виробнича функція Кобба Дугласа випуску продукції має вигляд: $v = ne^{rt} L^{\alpha} K^{\beta}$,

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 [7].

Розрахунки цих функцій дозволяють визначити відповідні оцінки доходної частини, зокрема і на фабрики програм. Економічність, системність та пропорційність випуску ПП залежить від принципів організації і керування ресурсами, номенклатури продукції, застосування комп'ютерів та служб фабрики (контролю, тестування, оцінювання ПП й ін.).

В Україні нині діють окремі організації і фірми з розроблення ПП, які орієнтовані на випуск продуктів для окремих замовників, як правило, закордонних. Фірми з успадкування ПП, що покривають деякий фрагмент ринку ПП теж працюють в основному для закордонних держав.

Для першого попиту з індустрії запропоновано створити *студентську фабрику програм і артефактів* (Http://programsfactory.univ.kiev.ua) на факультеті кібернетики в КНУ імені Тараса Шевченка з готових артефактів відповідних кафедр, їх апробація на технологічних або продуктових лініях (Product Lines) та розповсюдження їх не тільки на вітчизняному ринку, але з часом і закордонному.

Вона базується на лініях, обладнаних набором засобів, комплектуючих «деталей», інструментів і сервісів для автоматизованого виконання процесів на цих лініях у операційному середовищі.

З погляду інформаційних технологій фабрики дають набір інструментів для переходу від індустрії окремих ПП, до індустрії складних програмних систем з підвищенням продуктивності простих елементів продукту на кожному процесі життєвого циклу (ЖЦ). Лінії розробки простих продуктів на фабрики реалізовані в середовищі MS.Net з використанням, каркасів, мови DSL (Domain Specific Language) й ін. Тобто, до головних основ фабрики програм належать лінія розробки окремих ПП, з накопиченням їх в різних бібліотеках і репозитаріях, для зборки в цільові ПС.

Методика виробництва ПП. Ядро знань SWEBOK і відповідна програма навчання Curricula-2004 з програмної інженерії не містить дисциплін, пов'язаних з виробництвом програмної продукції на фабриках програм, принципами комплектації фабрики технічними і людськими ресурсами, методами створення необхідних *продуктових ліній,* служб підтримки, засобів, інструментів, керування відповідними фахівцями. Тобто в індустрії ПП не вирішені проблеми оцінювання складності об'єктів і процесів виготовлення ПП на

лініях, замало робіт, в яких пропонуються шляхи подолання складності, особливо при зборці великих систем з різних готових КПВ.

Тому потрібна **методика** розроблення і виконання процесу виробництва ПП. До неї належить набор нових дисциплін підтримки індустрії, які забезпечують нормативне і регламентоване виконання різних робіт з розробки, зборки, керування експертизами, вимірами і оцінками артефактів. Однією із форм методики для фабрики науково-технічні дисципліни (*Di*) ПІ, запропоновані в [10, 11]:

ПІ = DiSc, DiEn, DiEc, DsMa, DiDe,

де *DiSi* – наукова, *DiEn* – інженерна, *DiEc* – економічна, *DiMa* – управлінська (менеджерська) дисципліни та *DiDe* – виробнича.

Сутність цих дисциплін для виробництва ПС така:

– *наукова дисципліна* це класичні науки (теорія алгоритмів, множин, доказу, математична логіка тощо), теорія програмування та відповідні загальні мовні засоби проектування на відповідному рівні абстракції моделей і архітектури цільових програмних об'єктів а також стандартизовані методи програмування та процеси виготовлення ПП. Тобто вона є теоретичним фундаментом індустріального виробництва ПП у напрямку інтеграції (зборки) складних програм за їхніми інтерфейсами;

– *інженерна дисципліна* – це сукупність технологічних засобів і методів проектування ПП з фундаментальних і стандартних моделей життєвого циклу (ЖЦ), техніка аналізу предметної області, формулювання вимог, моделей системи, розробка вихідного коду, супровід, змінювання ПП (реінженерія, реверсна інженерія, рефакторінг) та адаптування ПП до інших комп'ютерних платформ і середовища;

– *дисципліна керування* застосовує загальну теорію управління і містить базові методи керування програмним проектом за допомогою графіків робіт, спостережень за їх виконанням, керує ризиками, версіями (конфігураційний файл) ПП та супроводженням;

– *економічна дисципліна* складається з сукупності методів експертного, якісного і кількісного оцінювання проміжних артефактів і кінцевого результату процесів ЖЦ, а також економічних методів розрахунку часу, обсягу, трудомісткості й вартості виготовлення ПП для постачання їх замовнику або на ринок;

– *виробнича дисципліна* – це лінії виробництва комп'ютерних і прикладних систем, сімейств систем із застосуванням готових програмних ресурсів (КПВ, сервіси, аспекти, агенти і т. п.) з інформаційних сховищ, бібліотек і репозиторіїв, а також одиночних готових програм, які перевіряються методами верифікації, тестування та оцінювання характеристик якості й надійності.

Ці дисципліни після відповідного їх подання для вивчання і освоєння як дисциплін виробництва різного роду програм у майбутньому стануть предметом підготовки студентів для їх участі в індустріальному виробництві якісних ПП за такими спеціальностями з дисциплін ПІ: аналітики – *DiSi* Sciences, інженери – *DiEn* Engineers, економісти – *DiEc* Economics, керівники – *DiMa* Managers, розробники – *DiDe* Developer тощо [11].

Фабрики програм. Під *фабрикою програм* розуміється інтегрована інфраструктура зі зборкою готових ресурсів у ПП, потрібних державним, науковим, комерційним й іншим замовникам. Фабрика обладнається продуктовими лініями, набором засобів, інструментів і сервісів для полу – чи автоматизованого виконання процесів на цих лініях в операційному середовищі. Визначення фабрики дано в [2, 5] таке: фабрика софтвера – це погоджений набір процесів, засобів й інших ресурсів для прискорення всього циклу створення тих чи інших програмних компонентів, застосувань і систем.

Фабрика базується на середовище, орієнтованим на автоматизацію виробництва ПП за відповідними лініями. З погляду інформаційних технологій фабрика дає набір інструментів для переходу від ремесла до індустрії ПП із метою збільшення продуктивності розробки продукту на кожному процесі ЖЦ лінії із заданими функціями, архітектурою і якістю. Фабрики містять лінії й відповідний набір засобів розробки простих і складних ПП. Лінія розробки простих елементів продуктів, як правило, відповідає ЖЦ, наприклад, реалізованому в середовищі MS.Net з використанням рекомендацій, каркасів (framework), DSL-мов й ін. Лінія розробки складних ПП може бути зборочного типу із готових програмних ресурсів, що знаходяться в різних бібліотеках і репозиторіях Інтернету.

2. Індустрія програм – базові основи

Виходячи з отриманої практики автоматизованого збирання різнорідних програм у мовах програмування (МП) в середовищі ОС ЄС [1] і аналізу сучасних закордонних фабрик програм з індустрії ПП (IBM, OMG, Microsoft, Oberon тощо) [3–8] нами сформовані загальні елементи індустрії виробництва програм, а саме:

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

– *інтерфейс,* як специфікатор готових ресурсів, незалежно від мов програмування (МП), в мові інтерфейсу (IDL, API, SIDL, WSDL, RAS тощо) [6, 7];

– *технологічні лінії (ТЛ), продуктові лінії* (Product Lines) [1] з виробництва ПП;

– *зборочний конвеєр* фабрики програм;

– *методики з прийомів* проведення робіт на фабрики програм;

– *середовище* виробництва програм на фабрики.

Ці складові елементи сформульовані нами і розвиті в рамках фундаментальних проектів Інституту кібернетики (1980–1991рр.) і ІПС НАНУ (1992–2010рр.) [1, 8]. В них уперше визначено концепцію

інтерфейсу (1982 р.) [10]), метод зборки різномовних програм, технологічні лінії (ТЛ, 1987–1991рр.)) та засоби автоматизації випуску ПП [1–5]. Все це попередило появу індустріальних складових елементів з виробництва ПП закордонними фірмами, наприклад, IBM, Microsoft, Oberon та ін.

Далі дається формальне тлумачення наведених складових елементів індустрії ПП і обчислень та індустрії даних.

2.1. Опис складових елементів

Готові програмні ресурси (ГОР)

Програмні ресурси, які можуть використовуватися багаторазово, звуться reuse, assets, компоненти, 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 КПВ, вони мають бути якісними і надійними при обчисленні [8].

Готові ресурси відображають деякі артефакти з діяльності розробників програм. *Артефакт* – це реальна порція інформації з програмування, яка може створюватися, змінюватися і використовуватися при виготовленні ПП, їм може бути:

– модель ПрО зі своїми термінами, поняттями та лексикою;

– готові КВП або окремі частини (фрагменти) систем;

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

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

Артефакти незалежні від платформ комп'ютерів мають опис інтерфейсів і параметрів для взаємодії з іншими готовими ресурсами. Усі ресурси і їх інтерфейси зберігаються у сховищах (бібліотеки, репозиторії) для подальшого їх пошуку іншими фахівцями з метою подальшого використання. Повторне використання є капіталомістким видом діяльності з ПП на фабрики.

Інтерфейси ГОР

Під *інтерфейсом* розуміється зв'язок двох окремих сутностей. Інтерфейси бувають 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 з завданням значень типів даних їх параметрів [6, 7].

У ролі інтерфейсу виступають оператори виклику до процедур і функцій у МП з завданням імен процедур або функцій і списку формальних параметрів з фактичними значеннями для виконання. Послідовність і число формальних параметрів відповідають 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.

Інтерфейсу фактично відповідає *посередник* або *перехідник* між двома модулями чи програмами. Він передає дані і виконує необхідне пряме й зворотне перетворення даних у випадку їх неоднорідності та невідповідності. Він визначається мовою IDL або APL для об'єкта-клієнта і об'єкта-сервера, має окрему реалізацію і доступний різномовним програмам. Інтерфейсний посередник включає опис формальних і фактичних параметрів програм, їх типів і порядок завдання операцій передачі параметрів і отримання результатів після виконання. Іншими словами, такий опис є не що інше, як специфікація інтерфейсу двох різномовних програм, які взаємодіють між собою через виклик, який реалізований практично в різних загальних системах (наприклад, COM, CORBA, JAVA тощо). У функції інтерфейсного посередника (stub у системі CORBA) для клієнта входить:

– підготовка зовнішніх параметрів клієнта для звернення до сервісу сервера;

– посилка параметрів серверу і його запуск для отримання результату або відомостей про помилки.

Загальні функції інтерфейсного посередника (skeleton) сервера:

– отримання повідомлення від клієнта, запуск видаленої процедури, обчислення результату і підготовка (кодування або перекодування) даних з формата клієнта;

– повернення результату клієнтові через параметри повідомлення з результату виконання) та ін.

Таким чином, інтерфейсні посередники задають зв'язок між клієнтом і сервером (*stub – для клієнта і skeleton* – для сервера).

Сучасні підходи до реалізації інтерфейсу. До підходів з вирішення проблеми інтерфейсу належать наступні:

1) зв'язок готових, різнорідних програм за допомогою інтерфейсу IDL, в якому визначені вхідні й вихідні дані цих програм;

2) спеціальний інтерфейс – JNI (Java Native Interface), що допускає звернення з Java-класів до функцій і бібліотек на інших МП з пошуком прототипів звернень до функцій на C/C++, генерацією заголовкових файлів компіляторами C/C++ і звернення з Java-класів до COM-компонентам;

3) технологія Bridge2Java, що генерує оболонку для COM-компонентів у вигляді проксі-класа і забезпечує необхідне перетворення даних для різних МП засобами стандартної бібліотеки перетворень типів;

4) зв'язок за допомогою мови CLR(Common Language Runtime) платформи .Net для будь-яких її МП, в якій транслюються об'єкти в ЯП (C#, Visual Basic, C++, Jscript) з використанням бібліотеки стандартних класів і засобів генерації в представлення .Net-компонентів;

5) стандартне рішення ISO/IEC 11404–1996 щодо специфікації типів даних незалежно від МП за допомогою мови LI (Language Independent) для різномовних компонентів, що містить усі існуючі типи даних МП або засоби їх конструювання. У мові LI описуються параметри виклику, як елементи інтерфейсу, вони перетворюються у типи даних конкретних МП спеціальними правилами і операціями генерації складних типів даних цього стандарту до більш простих фундаментального типу;

6) XDL-стандарт опису структур даних довільної складності і перетворення форматів даних, що передаються з однієї платформи комп'ютера на іншу за допомогою спеціальних процедур;

7) XML-стандарт забезпечення взаємозв'язків і перетворення типів даних ЯП до єдиного формату XML, зрозумілого багатьом розподіленим середовищам.

Рішення по конкретному перетворенню даних цими варіантами не вичерпуються, вони ще з'являтимуться при впровадженні нових платформ комп'ютерів і середовищ гетерогенного типу.

Визначення і тлумачення інтерфейсу модулів по-перше в 80 роках дано нами у монографії 1976 р. (В.М. Глушков, К.М. Лавріщева, А.О. Стогній та інші, «Система автоматизации производства программ – АПРОП»), в доповіді на конференції "Інтерфейс СЄВ" з технологічного інтерфейсу (1987 р.), потім П. Вегнер, сформулювавши парадигму *взаємодії* об'єктів, як перехід до обчислення модулів і доказом, що обчислення і взаємодія об'єктів – є дві ортогональні концепції (1997 р.), в якій взаємодія – як деяка дія запуску на проведення обчислення, а *повідомлення*, як передавач такої дії з даними. Пропонувалося розглядати операції інтерфейсу неалгоритмічними, а повідомлення – діями для взаємодії об'єктів у операційному середовищі.

Подальший розвиток ідеї взаємодії, ґрунтованої на діях мови AL(Action language), дано А.А. Летічевським і Гілбертом (1997), які визначають виклики процедур (локальних або розподілених) і їх розгортку у вигляді операторів дій, що розглядаються як обмежена множина операторів кінцевої програми для взаємодії з середовищем, де вони занурені.

Кожний інтерфейс специфікується паспортними даними МП сучасного стандарту WSDL виду:

– *назва* інтерфейсу;

– *ID* – ідентифікатор ресурсу;

– *зміст* програми (функції) з інтерфейсом;

– *параметри* виклику інших програм;

– *інструменти* підтримки виконання програми тощо;

– *необов'язкові атрибути* (дата, стан, версія, право доступу, автор, дата створення, термін придатності, правила придбання і т. п.).

Специфікації інтерфейсу зберігаються у бібліотеки інтерфейсів для майбутньої зборки різнорідних програм за інтерфейсами. Для включення студентських програмних артефактів розроблено шаблон специфікації їх інтерфейсів за WSDL, позиції якого заповнюється розробниками програм..

Прикладом розвитку різних типів взаємодії різнорідних програм є керівництво І. Беєві («Взаимодействие разноязыковых программ», 2005), представив більше 100 спроектованих варіантів модулів-посередників для програм в класі сучасних мов : C/C++, Visual C++, Visual Basic, Matlab, Smalltalk, Lava, LabView, Perl їх різних предметних областей. Ці варіанти практично перевірені автором у відповідних середовищах функціонування без врахування проблем, пов'язаних з новими архітектурами комп'ютерів і середовищ. Нами додані нові сучасні підходи автоматизації взаємодії різномовних програм у новому програмному інструментально-технологічному комплексі (ІТК), названому веб-сайтом з програмної інженерії (http://sestudy.edu-ua/net).

Роль цих концепцій інтерфейсу для взаємодії КПВ і шляхів їх реалізації зростає завдяки накопиченню величезного запасу цих КПВ у світі і інтерфейсів для різних предметних областей, використання яких без визначення їх інтерфейсів, принципів їх взаємодії і засобів їх підтримки не можливо.

2.2. Технологічні і продуктові лінії

Технологічна лінія – ТЛ будується з процесів ЖЦ після аналізу ПрО й виявлення її основних задач і функцій. Для процесів підбираються готові ресурси, а також засоби й інструменти породження для функцій програм. Крім того додається планування, контроль й керування процесами ТЛ з формуванням шаблонів і заготовок фіксації проектних рішень, зміни станів і оцінювання якості ПП [2, 4].

Набір процесів ТЛ будуються з урахуванням міжнародних стандартів ISO/IEC 12207– 2007 і ДСТУ 3918–99, що підкріплюються відібраними методами, засобами й інструментами для здійснення змін станів проміжних об'єктів. Цей підхід до побудови ТЛ апробований в АІС "Юпітера" (1987–1991 рр.) на прикладі шості ТЛ обробки даних.

На новому витку історії розвитку ліній індустрії програм виникли поняття продуктових *ліній* в SEI США (SEI), принципи побудови й виконання яких аналогічні ТЛ і відображають індустрію виробництва систем, сімейств систем комерційного типу.

Поняття *продуктової лінії (*product line (лінія продуктів) і product family (сімейство продуктів, надалі СПС) визначені у словнику ISO/IEC FDIS 24765:2009(E) – Systems and Software Engineering Vocabulary як

«група продуктів або послуг, які мають спільну керовану множину властивостей, що задовольняють потреби певного сегмента ринку або виду діяльності» («*product line* – group of products or services sharing a common, managed set of features that satisfy specific needs of a selected market or mission. Synonym: *product family*»).

Тобто с*ім'я продуктів* або *лінія продуктів* для деякої ПрО – це множина ПС, розроблених з використанням набору ГОР, які мають спільний набір властивостей і рис для деякої ПрО. Під *ресурсами* розуміються артефакти, КПВ, проектні рішення, коди, вимоги тощо. Кожна лінія ініціює первинний процес або зборочний конвеєр з виробництва ПП із ГОР з метою задовольняння потреб деякого ринка ПП. Наведені типи ліній виробництва ПП не обмежені. Деяка фабрика може бути зорієнтована на спеціальні лінії функціонального типу, наприклад, на створення програм із класу задач статистичної обробки, чисельних методів тощо.

Розробка методів побудови нових технологій. Дослідження показали, що необхідним етапом програмування нових систем є *технологічна підготовка розробки* (ТПР), аналогічно тому, як це має місце в промисловості при визначенні технології виробництва деякого типу виробів. Мета ТПР – визначити спеціалізовану технологічну лінію методом «складання» необхідних засобів, нотацій, мов в єдину ТЛ для генерації (побудови) по ній конкретної системи чи ПС [1, 5].

Вітчизняна технологія побудови ліній. Ця робота виконувалася, коли ГКНТ СССР спільно з членами СЄВ розробив Комплексну програму науково-технічного прогресу країн-членів СЄВ до 2000 року. Водночас програмна продукція уперше була оголошена продукцією виробничо-технічного призначення. Була сформульована програма ГКНТ СРСР «Розвиток технології розробки і промислового виробництва програмних засобів обчислювальної техніки». У рамках цієї програми в інституті були розроблені основи ТПР програмних виробів як етапу, який передує безпосередньому створенню СПС і призначений для побудови базової ТЛ в організації з подальшою її адаптацією для виробництва множини конкретних ПП різного призначення, що висвітлені у препринтах з ТПР (1982 р.) [1] і модульна інженерія (2008 р.) [5].

Такий підхід до визначення конкретних технологій був оригінальним, не мав аналогу, він реалізував класи програм в прикладних системах (АСУ, СОД, АСНІ та ін.). Процеси і операції ТЛ визначалися за допомогою спеціальної мови специфікацій процесів. Визначений набір документів ТПР (методик і стандартів підприємства), які регламентують побудову ТЛ. Уперше сформульовані завдання ТПР, рішення яких спрямоване на визначення ТП для шості розроблених співробітниками відділу (Г.І. Коваль, Т.М. Коротун, К.М. Лавріщева, Є.І. Моренцов) ТЛ з описом ТП та змісту програмно-технологічних документів (карт ТЛ і процесів, технологічних маршрутів проектування і вихідних форм).

До основних об'єктів ТПР віднесені: об'єкт розробки (початкове, проміжні і кінцеві стани) і відповідні процеси ЖЦ; методи програмування, засоби й інструменти, що забезпечують зміну станів на процесах ЖЦ; моделі ТП і ТЛ; інженерні методи (планування, контролю) керування розробкою програм на процесах ТЛ.

Один з процесів ТПР – аналіз предметної області для виявлення класів об'єктів, які специфікуються в процесах ТЛ, яким зіставляються типові завдання, для яких будуються моделі функцій предметної області. Основа створення ПС за ТЛ із класу технологічних об'єктів по моделі формалізованого представлення станів об'єктів і процесів побудови нових програм на ТЛ. У рамках проекту ТПР розроблений набір моделей для відображення проміжних результатів проектування по ТЛ (моделі фіксації проектних рішень у ході розробки; моделі процесів і ліній для відображення діяльності виконавців; модель якості для оцінки показників якості на усіх процесах ЖЦ; модель експлуатаційних документів тощо).

Таким чином, при побудові ТЛ використовувався процесний підхід, в якому приділялася увага на забезпечення процесів створення якісного продукту шляхом проведення заходів по своєчасному виявленню і усуненню помилок в ПС і оцінці результатів розробки на досягнення заданих показників якості. Цей підхід фактично зумовив визначення процесів створення ПС в Міжнародному стандарті ISO/IEC 12207 і у відповідному ДСТУ 3918–99 "Інформаційних технологій. Процеси ЖЦ програмного забезпечення".

На відміну від вказаних стандартів процеси ТЛ підкріплюються методами, засобами й інструментами програмування. Вони вказуються в специфікації ТЛ для тих процесів, які здійснюють автоматизоване перетворення станів об'єкта розробки. *Мова специфікації* процесів ТЛ є першою в СРСР спробою формалізовано описати ТЛ за ТП і їх складові – технологічні операції.

Основні атрибути операції: стан об'єкта, вхідні і вихідні дані, метод і інструментальний засіб розробки результату за цією операцією. При описі операції контролю, наприклад, це найменування показника якості, оцінний елемент (поточний робочий продукт розробки), метрика і метод оцінки.

Даний метод побудови ТЛ на той час був новий і захищений у ряді публікацій, апробований при створенні конкретних функціонально-орієнтованих технологій програмування застосовних програм, працюючих з базами даних у рамках АІС "Юпітер-470" Міноборони СРСР за ініціативою головного конструктора П.І. Андона, а також розроблено шість ТЛ, докторська дисертації К.М. Лавріщевої.

Крім того, була проведена експериментальна реалізація ТЛ у рамках ГКНТ СССР: конструювання технологічних інтерфейсів – КОНТИ, оцінки надійності – ТМНАД; розробки пакетів застосовних програм – АПФОРС тощо.

Сучасні лінії виробництва програмних продуктів. Ідея ТПР побудови технологічних процесів і ТЛ постійно обговорювалася в публікаціях і продовжувала розвиватися. Останніми роками з'явилися нові технологічні напрями [8**,** 10]:

– інженерія КПВ (Reusible Engineering);

– інженерія застосувань (Application Engineering);
– інженерія предметної області (Domain Engineering);
– інженерія продуктових ліній (Product lines – SEI США).

Інженерія КПВ – це систематична і цілеспрямована діяльність по підбору реалізованих програмних артефактів і їх подання КПВ у сховищах (репозиторії) системи підтримки ТЛ, аналізу функцій КПВ з метою їх застосування як готових у системі, що проектується, для інтеграції з іншими компонентами**,** характеризує систематичну і цілеспрямовану діяльність з використання КПВ.

Інженерія застосувань – це процес виробництва продукту для прикладної системи з готових КПВ (модулів, програм, підпрограм та ін.), які раніше створені самостійно або в середовищі окремої ПС, або як елементи багаторазового використання в інженерії КПВ чи деякої предметної області (ПрО).

Інженерія ПрО (домену) – це технологія розробки частин або систем сімейства ПС, збір, систематизація і їх збирання на конвеєрній основі. Необхідні умови цієї інженерії – системні інструментальні системи підтримки методів накопичення КПВ і генерації з них окремих членів сімейства.

Створення сімейства ПП в інженерії ПрО вимагає наявності певної компонентної платформи, що включає сукупність КПВ для генерації окремих членів сімейства ПС.

Інженерія продуктових ліній. Створення *ліній виробництва продуктів,* запропоноване інститутом SEI з систем, каркасів, готових програм і КПВ, з яких формується кінцевий продукт, що задовольняє певним потребам ринку програмної продукції [10, 12].

Поняття фреймворка для лінії програмних продуктів (Framework for Product Line Practice) сформувалося як деяка автоматизована реалізація інженерії Про, в завдання якої входить побудова різних видів програмних продуктів для ринку за допомогою методів і засобів лінійок виробництва продуктів.

Для побудови ліній виробництва досліджуються ринок і потреби покупців, створюється виробничий план, визначаються процеси, організація їх виконання і взаємодії. На основі аналізу потреб ринку і інтересу до певного виду продукту будується технологічна лінія продукції, в яку включаються необхідні методи розробки, тестування і оцінки процесів, продуктів лінійки.

У інфраструктуру розробки лінії продуктів, окрім необхідних методів і засобів побудови, експлуатації ліній продуктів, входять матеріали і методики по керівництву. Хоча лінії і сімейство продуктів частенько ототожнюють, лінії продуктів (на ринок) може бути побудована на базі певного представника сімейства продуктів (що розробляється, наприклад, щодо певного замовлення).

При побудові конкретної лінії для деякого представника сімейства у ПрО визначаються:
– технологічні і виробничі обмеження, властиві продуктової лінії;
– зразки і каркаси, які можуть використовуватися на лінії;
– набір засобів, методів і інструментів для розробки продукту на лінії.

Цей напрямок ґрунтований на використанні КПВ, які відбираються і накопичуються в репозиторіях ПрО, що покращує процес складання готових функцій, компонентів і КПВ в єдиний продукт на лінії виробництва.

ТПР і інфраструктури побудови ліній продуктів SEI [5, 10] ї показує, що ТПР (1982–1999 рр.) не лише уперше визначив ТЛ, але мав формалізований опис під певні типи продукту (ППП, АСНИ та ін.). Продуктова лінія – це та ж ТЛ, яка будується експериментально і задовольняє вимогам ринку до певного виду продукту (2002–2005 рр.). Обидва напрямки мають спільні технологічні цілі, хоча і визначають різні шляхи побудови ліній виробництва програм.

Нами розроблені три базові лінії загального призначення для **фабрик програм** [2].

Перша лінія для 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 мовами програмування (МП) і паспортних даних інтерфейсів;
– зберігання їх у репозиторію.

Реалізація процесів такої лінії закінчується описом сформованих артефактів або ресурсів у класі задач ПрО, специфікацією паспортних даних та розміщенням їх і описів ресурсів у репозиторію для подальшого застосування. Ця лінія потребує вкладення капіталу в розробку наукових артефактів. Такого типу лінії використані в студентської фабрики, яка реалізує ЖЦ програм у середовищі Visual Studio .Net.

Друга лінія – це механізми підбору готових ресурсів із репозиторію з метою їх застосування у відповідної предметної області

Третя лінія – це лінія об єднання (зборка) різних КПВ, що забезпечує конструювання нових ПП методом зборки з готових підібраних ресурсів.

Ця лінія буде давати прибуток за рахунок заощадження трудовитрат від застосування готових артефактів та КПВ за оцінкою ефективності їх застосування і обсягів повернення цього вкладення в ПП. ГОР може бути науковим, прикладним і загальносистемним. Перший ресурс – це метод, алгоритм як артефакт (з математики, фізики, біології тощо), що описаний загальною МП, другій ресурс – це реалізація окремій задачі або функції ПрО (бізнесу, комерції, економіки і т. п.) і загальносистемний ресурс – це готові засоби середовища виконання ліній (транслятор, редактор текстів, генератор, інтегратор, завантажувач, сервіс тощо). Зміст даної лінії відповідає лінії зборки програм.

2.3. Зборочний конвеєр

Зборочний конвеєр (assembly line/belt assembly line – в великому словнику) визначає автоматизовану одну чи більш ліній зборки на фабрики. Кожна лінія підвищує продуктивність праці виконавців, поліпшує умови їх роботи, скорочує число збиральників, підвищує якість продукції та знижує собівартість випуску продукції. Про такий конвеєр говорив В.М. Глушков у 1975 р. на науковому семінарі – "Пройде 20–30 років і складні програми будуть випускатися по принципу зборочного конвеєра, як в автомобільної промисловості", який уперше з'явився в 19 сторіччі це конвеєр на фабрики Форда. Ця ідея Глушкова щодо програм реалізована нами з теоретичної і практичної точки зору. Нині можливо говорити, що ми вже вирішили забагато щодо такого конвеєра на фабрики програм, базисом якої є лінії виробництва окремих елементів і з них складних програм.

Відповідно словнику, конвеєр чи його лінія може бути автоматизованою (як станок) або частково автоматизованою фабрикою за технологічними лініями. В індустрії ПП розглядаються тільки частково автоматизовані лінії, тому що «комплектуючі деталі» фабрики **не матеріальні**, вони належать до класу творчих «невідчутних» об'єктів, які невідомими у внутрішньому виді комп'ютера при виконанні, а лише зовнішнє, у вигляді опису деякого артефакту, КПВ на фабрики програм, специфікованих стандартизовано різними МП.

Кожний виконавець чи збиральник КПВ у нашому випадку виконує процеси лінії по отриманню проміжного або кінцевого продукту, в залежності від призначення лінії і типу продукту, що можна призвести на ній (наприклад, лінія розробки КПВ, зборки КПВ тощо).

Оскільки лінія програмних продуктів складається з процесів, які виконуються за участю одного чи більше розробників, кожний з них завдає необхідні вхідні дані для процесу виробництва програм (наприклад, готовий КПВ при зборці його з іншими).

У промисловості можуть бути неперервні лінії, що діють без участі робочих на проміжних діях конвеєра, а також з зупинками в установлені проміжки часу або в залежності від результату. Кожний робітник виконує закріплену за ним роботу. В нашому випадку виконавцем може бути розробник, верифікатор, тестіровщик, чи оцінювач якості тощо. Але вони повинні розмірковувати про виконання дій на лінії.

2.4. Середовище виробництва програм на фабрики

Середовище виробництва – це засоби, інструменти і методики забезпечення роботи фабрики програм при виробництві ПП методом зборки, функції яких забезпечують їх віртуалізацію, синхронізацію, виконання та підтримку продуктових та технологічних ліній з виробництва різнорідних програм в МП.

Засоби – це різні МП, які використовуються для опису програм і інструменти, типу транслятори, редактори обов'язково будуть на фабрики, як необхідний засіб виробництва програм.

Інструменти – це системні, прикладні програмні компоненти (Eclipse, Protégé, Eclipse-DSL тощо) підтримки процесів побудови прикладних ПС чи СПС та сучасні фреймворки (Jaspect, Jbeans, Ant, WCF, Web-service, Amazon, Sky-Driven тощо), які мають комплекс засобів для побудови окремих програмних об'єктів і даних в технології виробництва окремих особливих елементів, та загальносистемні засоби (VS.Net, Corba, Java, IBM Vsphere тощо), які самі як фабрики з великим набором різних Tools. І засобів підтримують командну розробку складних програм і проектів.

Методики розроблення і виконання процесів виробництва ПП включає документацію і інструкції по організації ТЛ і ТП і принципам проектування на них різних окремих елементів. Сюди відноситься створений нами набір нових дисциплін підтримки індустрії, які забезпечують нормативне і регламентоване виконання різних робіт з розробки, зборки, керування експертизами, вимірами і оцінками артефактів. Однією із форм методики для фабрики науково-технічні дисципліни (*Di*) програмної інженерії (ПІ), сутність яких розглянуто у пункті 1 –"Методика виробництва ПП".

Таким чином, розглянуті основні базові елементи індустрії програм, які з нашої точки зору відносяться до базових основ виробництва ПП.

3. Індустрія обчислень програм і даних

Нові індустріальні підходи до обчислення програм даних з'явилися у зв'язку з бурним розвитком високоефективної елементної бази, нової багатоядерної, процесорної конфігурації комп'ютерів тощо. Це сприяє великомасштабним обчисленням дуже складних задач сучасності в e-sciences (геології, біології, фізики, математики та ін.), АСУ й 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, індустрію обчислень систем і програм [3–5].

Індустрія обчислень у теоретичному плані іде по шляху застосування нових комп'ютерних можливостей щодо швидкості дій, розподілення пам'яті для обчислення задач з великими обсягами даних та інтероперабелності [6–8].

Індустрія обчислення програм у Grid. 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, *хмарних обчислень* (Cloud Computing), 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 Model Driven Development (MDD) моделюється на двох рівнях – платформи незалежного рівня Platform Independent Model (PIM) і платформи залежного рівня Platform Specific Models (PSM). Концепція дворівневого моделювання архітектури Model Driven Architecture (MDA) і відображення PIM→PSM відповідає ідеології побудови сімейства систем СПС мовою DSL для опису понять і задач з простору проблем предметної області [8].

Одним з практичних рішень забезпечення *індустрії обчислень* є поява системи Grid у 2005 р. в межах Європейського проекту. Цю систему було розроблено як інфраструктуру для глобальних обчислень суперскладних задач з області e–science. Підсистема Etics в Grid підтримує репозиторії програм і даних, виконує збирання готових КПВ, програм, модулів, систем у конфігураційний файл для виконання обчислень. Нами виконано опис специфікації КПВ з Etics для подальшого їх подання у глобальному репозиторії Grid. Для систем розроблено загальну модель обчислень, як розвиток моделі взаємодії програм для забезпечення інтероперабельності в конфігураційному файлі різнорідних програм і використання даних з on-line Cloud [3, 4].

Для обчислень таких задач застосовуються знов розроблені або готові програмні і системні ресурси виду: розподілені процесорні потужності; системи сховищ даних; новітні засоби комунікації; фонди reuses з багатьох доменів; нові технічні устаткування і загальні «тули» (конвертори, генератори, трансформатори, верифікатори тощо); моделі та схеми взаємодії програм у середовищі гетерогенних платформ, географічно розташованих у віддалених адміністративних доменах тощо.

Хмарні обчислення

Іншим напрямком сучасного обчислення задач глобального типу є Cloud Computing – *хмарні обчислення,* які підтримуються новими системними засобами, які підтримуються системами Google Apps, IBM-VSphere та системами Microsoft – WCloud, Azure, Amazon, Mech, WApps, SkyDriven (Http://lenta.ru/articles/2010). Функції деяких систем розглядаються далі. Google Apps забезпечує обчислення бізнесу в режимі on-line за допомогою Інтернет-браузера до великих обсягів даних на серверах Google, виконує веб-застосування з сховищами даних через інтерфейс API .в мовах Java і Python за допомогою стандартних протоколів [3, 4].

Amazon Machine Image (AMI) підтримує застосування, бібліотеки за інструментами зберігання AMI і образів у сховищі, вибір ОС для запуску і контролю декількох AMI з використанням Веб-сервісу, інструментів управління та платою за надані для вживання ресурси, такі як час за кількість переданих даних. Azure (Windows Cloud, 2008) для створення розподілених «хмарних» веб-застосувань і Інтернет-сервісів з використанням технології .Net.

Amazon Web Services — це інфраструктура Web Services для надання таких послуг: зберігання даних (файловий хостінг, розподілені сховища даних), надання обчислювальних потужностей тощо. При чому, час обчислення 12 центів, гігабайт даних на сервері – 15 центів, кожні 10 тисяч транзакцій – 10 центів.

Ці системи 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. Ця індустрія 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 напрямків e–science в структури ПП, після виготовлення вони подаються на різні глобальні сервери для їх застосування.

Для хмарних обчислень основним припущенням є нерівномірність запиту ресурсів з боку клієнта (ів). Для згладжування цієї нерівномірності в плані надання сервісу між реальним залізом і middleware розміщується ще один шар — віртуальний сервер, який виконує застосування шляхом віртуалізації і балансування навантаженням засобами програмного забезпечення, так і засобами розподілу віртуальних серверів по реальних платформах і комп'ютерах..

Індустрія даних

Відповідно коментареві (http://lenta.ru/articles/2010/12/webservice) ринок сервісів для збереження даних on-line через п'ять років стануть більш прозорим, буде вбудований в продукти Microsofts з можливістю синхронізації даних за допомогою Windows Live Mech і передачі даних на віддаленні сервери. Сервіси SkyDriven і Mech взаємодіють зі хмарою у цілях вибору необхідного сервіса. Хмарне сховище є базисом при роботі сервісів. Користувачі зможуть мати різні обсяги даних з сховищ, а також створювати резервні копії даних. Поряд з таким механізмом доступу до хмарних даних з'являються нові безпровідної технології типу WiMAX подібно безпровідному Інтернету. Значні можливості дають веб-технології щодо збереження та доступу даних (Amazon, Flash, HTMLS, тощо).

Сервіс даних наведених on-line сховищ включає:

– величезну кількість накопичених даних з новим видом доступу до них засобами Amazon, Flash, HTML5) з деякого місця планети;

– синхронізацію даних засобами Windows Live Mech, SkyDriven тощо;
– передачу даних на віддаленні сервери і резервування різних копій даних (Office WebApps);
– використання нових веб-технологій (Flash, Silverlight, Amazon);
– компоузер завдань, робота з БД (Windows Azure, MS SQL, Services, Live Services);
– бібліотеки примітивів з перетворення типів даних GDT<=>FDT із бібліотек.

Бібліотека примітивів для перетворення типів даних GDT (примітивних, агрегатних і генерованих) до FDT типів даних (простих, структурних і складних) МП використовується при передачі даних між різномовними компонентами, підсистемами і проектами. Бібліотека складних типів даних GDT (контракт, портфель тощо) використовується для генерації простих типів даних FDT і накопичення у базі даних. Бібліотека функцій для перебудови форматів не релевантних даних інтерфейсних посередників (stub, skeleton), що передають їх на інші платформи взаємодіючим компонентам і зворотно. Бібліотека перетворення даних з on-line сховищ містить.

Організація обчислень

Для функціонування різних інструментів і засобів фабрики використовуються різні моделі організації зв'язків між компонентами на лінії [3, 4]. До них відносяться такі:

1) брокер запитів між програмами клієнта і сервера з отримання даних від stub клієнта, їх обробку під формати даних сервера складених через skeleton, що дає напрям обробки результатів обчислювань з доведенням їх до типу даних клієнта;

2) проміжний прошарок загальносистемних середовищ підтримки процесів виконання або взаємодії об'єднаних різнорідних програм;

3) пряма зборка транслюваних програм за різними МП і інтерфейсами з бібліотеки MS.Net.

Перша модель взаємодії практично реалізується брокером ORB для класу МП у середовищі CORBA. Зборка компонентів базується на класах: Client class з викликами інших, Stub class з конвертування даних, ORB class з передачі даних за викликами; Server class зі створення сервентів та посиланням даних ORB; Skeleton class з конвертування форматів даних та адаптера для породження різних видів серверів.

Друга модель забезпечує взаємодію між компонентами, розміщеними на різних комп'ютерах, через проміжний прошарок (middleware), та неоднорідність їхніх форматів даних 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) через мережну ОС. Верхній рівень моделі (прикладний) забезпечує перетворення даних до транспортного рівня шляхом їх маршалінгу й демаршалінгу за інтерфейсним stub. Сеансовій рівень моделі передає дані через транспортний канал за механізми посилань іншим рівням.

Третя модель реалізована в MS.Net і базується на бібліотеках: CLR (Common Language Runtime), CTS (Common Type System) і CLS (Common Language Specification). CLR забезпечує виявлення і завантаження типів даних, керування ними у відповідності до завдань розробника програми. CTS визначає принципи взаємодії із іншими, відповідно представлених форм метаданих. Зборка різномовних програм виконується за правилами CLS бібліотеки. Усі компілятори з МП цього середовища створюють модуль DLL або EXE у проміжну мову MSIL (Microsoft Intermediate Language) для зборки отриманого коду, незалежно від платформи, на якій він буде виконуватися рішення. При запуску цей код конвертується в специфічний код CPU івиконується на різних архітектурах комп'ютерів.

Для побудови програм для студентської фабрики програм використовуються Visual Studio, Веб-сервіси різного призначення, пакети VSTS для керування конвеєрним методом зборки та засобів вимірювання і оцінювання якості створених програмних продуктів.

Таким чином**,** в залежності від цілей фабрики програм в якості середовища вибирається те середовище, що найбільш підходить для організації процесу побудови і зборки відповідних програмних ресурсів у межах деякої ПрО. Коло проблем зменшується, коли різні середовища взаємодіють між собою, створюючи одне велике середовище з поширеними можливостями щодо виробництва кінцевого об'єднаного продукту.

Взаємодія програм, систем і середовищ – базис функціонування програм

У межах проекту проведено дослідження, орієнтовані на вирішення задач взаємодії програм, систем і середовищ, як базових процедур індустріального виробництва програмних продуктів [4].

Під *взаємодією* розуміють сумісність двох і більше об'єктів. Даний термін має спеціальний спектр використання в програмістській діяльності (наприклад, взаємодія програм і середовищ між собою) [4, 5, 12]. Здатність до взаємодії двох і більше програм або систем пов'язана з обміном інформацією і використанням її для організації обчислень. Для забезпечення локальної взаємодії програм у МП існує апарат зв'язку підпрограм і функцій через оператори звернення типу CALL, а для розподілених програм у різних середовищах є, наприклад, такі засоби – RPC, RMI, ORB (stub, skeleton), IContract тощо. Відповідно до цих засобів, зв'язки різномовних та різноплатформенних програм здійснює інтерфейс, який специфікується загально прийнятою мовою Interface Definition Language (IDL). На загальному рівні опис інтерфейсу з типами даних, що передаються, слугує для різних середовищ механізмом забезпечення взаємодії (interconnection) різнорідних програм.

Принципи забезпечення взаємодії різнорідних компонентів (різномовних, різноплатформенних, різносередовищних) розроблені у рамках генеруючої моделі ПС. Система породжується з окремих елементів (компонентів, сервісів, каркасів тощо), які специфікуються різними МП і відображають базові поняття певної ПрО, що реалізується у середовищі ГП. Модель взаємодії доповнює генеруючу модель Generative Domain Model (GDM) засобами опису КПВ і операціями розміщення в інформаційному просторі репозиторію ГП. Головне її призначення – забезпечувати взаємодію різнорідних компонентів, що побудовані в обраному інтегрованому середовищі Eclipse, MS.Net, CORBA, COM, працюють в них і можуть адаптуватися до іншого середовища, наприклад, Grid, призначеного для обчислення різнорідних програм глобального масштабу в області e–science.

Модель взаємодії систем M_{inter} забезпечує інтероперабельність і міграцію систем, розроблених у одному гетерогенному середовищі в інше, розширюючи межі їх виконання [12]. Вона має такий загальний вигляд:

$$M_{inter} = \{M_{pro}, M_{sys}, M_{env}\},$$

де $M_{pro} = \{Com, Int, Pro\}$ – модель програми, *Com* – компонент, *Int* – інтерфейс, *Pro* – програма;

$M_{sys} = \{PS, Int, Prot\}$ – модель програмної системи *PS*, *Int* – інтерфейс, *Prot* – протокол зв'язку для передачі даних;

$M_{env} = \{ENVIR, INT, PRO\}$ – модель середовища, в якому *INT, PRO* відображають сукупність зовнішніх інтерфейсів та протоколів, що передають дані між програмами через мережу, відповідно.

Тобто, параметрами моделі взаємодії M_{inter} є програма (компонент), інтерфейс і повідомлення.

Фактично M_{inter} щодо стандартної семирівневої моделі відкритих систем OSI, є моделлю верхнього рівня.

Програмні елементи моделі M_{inter} – це *Com*-компонент, *SYS*-система, *ENVIR*-середовище специфікуються відповідною МП, інтерфейсом мовою IDL, протоколом – мовою XDL, RDF тощо. Програмні компоненти і системи й інтерфейси зберігаються в бібліотеках і репозиторіях інтегрованої системи ГП, які використовуються для організації пошуку елемента типу КПВ, аналізу його функції і можливості застосування в новій системі *PS*. Розроблення самостійних програмних елементів виконується в інтегрованому середовищі Eclipse, MS.Net, CORBA, Java, COM з використанням взаємозв'язків між складними елементами.

Інтерфейс, як самостійний об'єкт, відігравав роль посередника між програмами, що викликаються і викликають. Оператор виклику задає список фактичних параметрів, який перевіряється на відповідність (кількість і порядок розташування) формальним параметрам програм. Якщо типи даних параметрів виявлялися не релевантними, необхідно виконувати пряме й обернене перетворення даних з урахуванням структури пам'яті комп'ютерів. У розподілених програмах, що розташовані у різних середовищах, використовуються такі засоби взаємодії – RPC, RMI, ORB (stub, skeleton).

Новим способом взаємодії між програмами типу клієнт і сервер забезпечується інтерфейсом *Icontract* у системі WCF (www. wcf.org), який призначений для опису атрибутів та операцій з передачі даних від сервісного об'єкта (*Service consumer*) до клієнтського (*Service provider*). Передача інтерфейсів виконується за контрактами:

– *сервісів* опису операцій з виклику клієнта і *передачі даних* за типами даних (*int*, *float*, *string* тощо) між клієнтом і сервером;

– *повідомлень, як* механізми забезпечення взаємодії об'єктів;

– *помилок*, що утримуються сервісною службою для передачі їх клієнтам.

Повідомлення специфікується мовою XML і подається протоколом SOAP.

При взаємодії середовищ через протоколи WCF виникають конфлікти (див. е-сайт wcf), пов'язані з:

– несумісністю типів даних, переданих в інтерфейсах клієнтів;

– різниця в МП опису окремих програм;

– відмінності в форматах архітектури платформ об'єктів клієнта і сервера;

– відмінність протоколів передачі повідомлень через мережу між IBM, SunMicrosystems, Microsoft;

– відмінність порядку параметрів у протоколі *Icontract* передачі даних.

На вказаному сайті пропонуються механізми відокремлення конфліктної ситуації.

Модель взаємодії програм – це схема зв'язків окремих частин програм між собою. Як зв'язки виступають оператори звернення (типу CALL, RPC й ін.) до процедур і функцій програм з формальними параметрами, які записуються в одній МП [4, 13, 14].

Оператори звернення містять імена об'єктів (процедур і функцій), що викликаються, список фактичних параметрів, з завданням їх значень, для отримання результатів. Послідовність і число формальних параметрів мають відповідати фактичним параметрам. Виконання функції у програмі на одній МП не викликає проблем, тоді типи даних параметрів збігаються.

Моделі взаємодії систем найбільш пов'язані з використанням готових програм у процесі розроблення нових ПС методом зборки. Якщо готові програми подані в різних МП і розташовані на різних комп'ютерах мережі, при їх збиранні можуть виникати проблеми взаємодії між ними, пов'язані з неоднорідністю типів даних, структур пам'яті комп'ютерів і середовищ, де вони виконуються. Зібрані КПВ разом враховують формати даних готових програм на платформах сучасних комп'ютерів, особливості структури вихідного коду програм після компіляторів з МП, який залежить від використання спеціальних бібліотек *data types* і *routines* або без них. Реально існуючі розходження в апаратній частині платформ і у вихідному коді програм відмічається у конфігураційному файлі ПП, який використовується при організації його виконання у сучасних обчислювальних або гетерогенних середовищах. Питанням перебудови загальних типів даних до

фундаментальних типів даних присвячений стандарт ISO/IEC 11404–2007 General Data Types (GDT), що пропонує механізми генерації складних типів даних (контракт, портфель тощо) до більш простих у МП. Але фундаментальні типи даних МП вже підтримуються спеціальними засобами сучасних операційних середовищ (Sun IBM, Microsofts.Net, CORBA, COM, JAVA тощо). Проміжний посередник типу stub, skeleton брокерного типу виконує перебудову типів даних від клієнта для передачі серверу і навпаки, а також вихідний код типу MISL ((Microsoft Intermediate Language) у проміжному вигляді або EXE виконується в таких системах, як Linux, Windows Server, MS.Net, IBM Web Sphere тощо [4, 13, 14].

Модель взаємодії операційних середовищ між собою розглядається нами як послідовність дій з перенесення програми, створеної у одному середовищі, в інше для їх виконання. Загальна схема розроблення програм у середовищі ГП на основі інтерфейсів програм і моделей взаємодії для середовищ IBM, VS.Net, Java, CORBA показано на рисунку.

Деякі середовища, наприклад, VS.Net і Java вже мають безпосередні зв'язки і створюють одне інтегроване середовище. В межах фундаментального проекту проведено реалізацію моделей взаємодії пар наступних операційних середовищ Visual Studio ↔ Eclipse та CORBA↔ Visual Studio за участю студентів КНУ імені Тараса Шевченка та МФТІ [8, 13, 14]. Дамо стислий опис процесів проектування програм і забезпечення їх взаємодії на прикладі вказаних пар середовищ.

Кожне з середовищ базуються на своїх інтерфейсах взаємодії і включає загальні методи і засоби доступу до даних мережного середовища. Програми, виготовлені в одному з середовищ, можуть бути перенесені з одного середовища в інше через репозиторій, наприклад, Eclipse, який виконує, по суті, функцію інтегратора програм у репозиторій з інших середовищ Інтернету. Ці операційні сучасні середовища забезпечують відповідні процеси ЖЦ з розроблення різнорідних і різноплатформенних програм та методи їх об'єднання у різні програмні структури ПП через відповідні механізми взаємодії, реалізовані в них [13, 14].

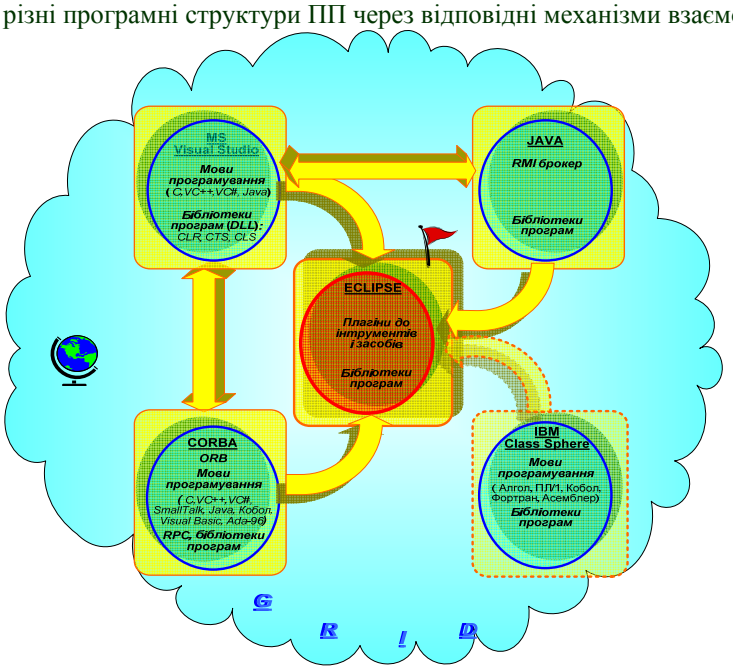

Рисунок. Взаємодія сучасних загальних середовищ

Практичні приклади пар взаємодії середовищ:

– *Visual Studio.Net, Eclipse* застосовують технологію розроблення окремих програм мовою C# за стандартними процесами ЖЦ, включаючи специфікацію інтерфейсу і паспортних даних та перенесення готового КПВ в репозиторій системи Eclipse, що відображає зв'язок з даним середовищем через механізм плагинів або конфігураційний файл з параметрами і операціями оброблення даних в тому чи іншому середовищі;

– *Corba*, *Java*, *MS.Net* забезпечують розроблення програм на МП цих середовищ та встановлення зв'язків між ними в цілях розміщення розроблених програм в репозиторій та надання доступу іншим розробникам його програм;

– *IBM VSphere*, *МП, Eclipse*, де розробляються нові програми з використанням можливих МП, що допускаються у цьому середовищі або у VSphere.

Eclipse виконує функцію адміністратора репозиторію по збереженню, відбору і застосуванню готових програмних ресурсів типу КПВ, а також для виконання ПП, виготовлених у цих середовищах.

Тобто при переході в середовище Eclipse використовуються вихідні файли програми, *dll*–бібліотки VS та файли ресурсів (*.resx*). Коли необхідно знов перейти із цього середовища в Visual Studio, весь проект імпортується туди. Обчислення програми та її зміни можна виконувати як у середовищі Eclipse, так і у Visual Studi*o*. Модель IBM Eclipse буде продовжено надалі при використанні сервісів для взаємодії програм і проведення обчислень за ними у розширеному середовищі.

Підхід до практичної реалізації ідеї індустрії

Виходячи з розглянутих концепцій і теорії індустрії, а також викладання лекцій автором в КНУ імені Тараса Шевченка по курсам «Технологія програмування» і «Програмна інженерія» студентами була підтримана ідея створення експериментальної фабрики програм з декількома лініями по поданню наукових артефактів, що розробляються як нові артефакти і програми, які кращі і їх можливо стандартизовано оформляти і накопичувати у репозиторії фабрики програм (http://programsfactory.univ.kiev.ua). Крім того, за участю студентів підготовлені курси навчання з дисциплін за такими технологічними лініями:

– розроблення засобами C# VS.Net (опис цієї технології на наведеному сайті фабрики) консольних програм на мові C#, бібліотек компонентів DLL, локальних Windows-застосувань;

– розроблення програм мовою Java (самовчитель І.Ш. Хабібуліна);

– електронне навчання курсу "Програмна інженерія" (SE) за підручниками на сайті КНУ фабрики програм (укр.), також раніше створеного цього підручника російською мовою на сайті Інтернету (www.intuit.ru).

На фабрики використано ліцензійну в КНУ VS.Net як фундамент функціонування фабрики програм і можливості платформи MS.Net щодо *багатомовної розробки* ПС із компонентів в МП (C#, C++, VBasic тощо) та засобів підтримки колективного виробництва програмних проектів. Як результат, зовнішній розробник програм для фабрики не обмежується вибором однієї якої-небудь МП, а може в межах однієї ПС використовувати різні МП використовуючи сайт фабрики програм [9, 13, 14].

Лінії навчання на фабрики корисні для навчання студентів, аспірантів, співробітників ІПС НАНУ та різних користувачів Інтернету. Підготовлений також новий сайт (http://sestudy.edu-ua.net), розроблений у рамках ІТК [8, 9] і в якому подані побудовані в межах фундаментального проекту ПС НАНУ 15 ліній розроблення різномовних програм і КПВ, що зберігаються у репозиторії за стандартом WSDL, а також у сучасної мови DSL, мові онтологічного Protégé для подання знань про предмети навчання (наприклад, обчислювальна геометрія, домен програмної інженерії – ЖЦ тощо).

Висновки

Визначені базові основи індустрії виробництва програм із готових ГОР і КПВ. Запропоновані головні елементи індустрії програм, обчислень і даних. До них віднесені *набор КПВ, інтерфейс опису* параметрів КПВ, *технологічні і продуктові лінії, зборочний конвеєр* та *операційне середовище* системного і методичного забезпечення процесів взаємодії різних компонентів і систем. Описано теорію індустрії програм, обчислень та індустрії даних. Розглянуті практичні варіанти моделі взаємодії складних систем і операційних середовищ (Corba,Vs.Net, Java, Eclipse тощо). Сформульовані нові дисципліни SE з підтримки виробництва програм на всіх його етапах та технологічних лініях електронного навчання студентів на фабрики програм в КНУ імені Тараса Шевченка (http://programsfactory.univ.kiev.ua). На фабрики відображено технологію розроблення програмних артефактів засобами Visual Studio .Net за стандартно прийнятими специфікаціями КПВ та запропоновано дистанційний електронний курс навчання студентів програмної інженерії. Також розроблений ІТК, в якому реалізовано нові теоретичні і прикладні аспекти технології виробництва програм за 15 простішими лініями та лінії взаємодії програм, систем і середовищ з використанням сучасних розподілених систем (VS.Net, Corba, Java, Protégé, Eclipse). Відповідний сайт цього ІТК – http://sestudy.edu-ua.net.

За короткий час до фабрики звернулися більш 3000 абонентів. Вона буде надалі доповнюватися новими дисциплінами, наприклад, обчислювальна геометрія, онтологія знань тощо. Рекомендується для навчання в ВНЗ за спеціальністю «програмна інженерія», інформатика і Computer Sciences.

1. *Лаврищева Е.М.* Становление и развитие модульно-компонентной инженерии программирования в Украине. – Київ, 2008. – 33С. – (Препр. / Ин-т кибернетики им. В.М. Глушкова; 2008–1).
2. *Андон П.І., Лавріщева К.М.* Розвиток фабрик програм в інформаційному світі // Вісник НАН України. – 2010. – № 10. – С. 15–41.
3. *Лаврищева Е.М.* Концепція індустрії наукового софтвера і підхід до обчислення наукових задач // Проблеми програмування. – 2011. – № 1. – С. 3–17.
4. *Лаврішева К.М.* Взаємодія програм, систем й операційних середовищ // Проблеми програмування. – 2011. – №3. – С. 13–24.
5. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии программных продуктов. – Київ: Наук. Думка, 2009.– 371 с.
6. *Лаврищева Е.М.* Интерфейс в программировании // Проблеми програмування. – 2007. – № 2. – С. 126–139.
7. *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.
8. *Лавріщева К.М., Коваль Г.І., Бабенко Л.П., Слабоспицька О.О., Ігнатенко П.П.*. Нові теоретичні засади технології виробництва сімейств програмних систем у контексті ГП .– Електронна монографія, ДРНТІ.– № 67–УК–2011 від 05.10.11. – 377 с.
9. *Лавріщева К.М.* Інструментально-технологічний комплекс для виробництва програмних систем.– Вісник НАН України, 2012.– № 3.– С. 19–23.
10. *Лавріщева К.М.* Програмна інженерія.– Підручник.– Академперіодика. – 2008. – 319 с.
11. *Лаврищева Е.М.* Классификация дисциплин программной инженерии // Кибернетика и системный анализ. – 2008. – № 6. – С. 3–9.
12. *Лавріщева К.М.* Генерувальне програмування програмних систем і сімейств // Проблеми програмування. – 2009.– № 1. – С. 3–16.
13. *Радецький І.О.* Один з підходів до забезпечення взаємодії середовищ MS.Net і Eclipse // Проблеми програмування. – 2011. – № 2. – С. 45–52.
14. *Островський А.И.* Подход к обеспечению взаимодействия программных сред JAVA и Ms.Net // Проблеми програмування. – 2011.– № 2.– С. 37–44.