П. Андон, К. Лавріщева

РОЗВИТОК ФАБРИК ПРОГРАМ В ІНФОРМАЦІЙНОМУ СВІТІ

Концепцію фабрик програм уперше сформулював академік В.М. Глушков на науковому семінарі 5 березня 1975 р. в Інституті кібернетики (ІК) у присутності К.Л. Ющенко, І.М. Молчанова, І.В. Сергієнка, Ю.В. Капітонової, О.А. Летичевського, А.І. Нікітіна, К.М. Лавріщевої, І.В. Вельбіцького та інших [1]. Суть нової парадигми Глушкова — прискорити перехід від мистецтва програмування до промислових методів виробництва програмних продуктів (ПП) для розв'язання різних господарських задач у межах нової системи ОГАС, що охоплювала всі республіки СРСР. Основними аргументами академіка Глушкова на користь переходу до промислового програмування були:

— обчислювальні, керівні й математичні машини, виготовлені самостійно в Україні;

— готові програми у мовах програмування (МП) 4GL (Алгол, ПЛ-1, Кобол, Фортран, Модула-2 та ін.) й алгоритми, накопичені в республіканських фондах алгоритмів і програм для повторного використання у нових розробках за конвеєрною збіркою;

— побудовано важливі автоматизовані системи, деякі АСУ керування підприємствами господарського та військово-промислового комплексу, АСУТП тощо.

Для реалізації цієї концепції ГКНТ СРСР відкрив державні наукові програми для академічних інститутів, аби створити засоби автоматизації (наприклад, Проект, Дельтастат, АПРОП, Альфа, Симп, Приз, Прометей та ін.) [2–12] різних комплексів програм, необхідних для виконання важливих науково-технічних завдань у різних галузях промисловості й господарства СРСР за допомогою комп'ютерів. Одночасно ідея індустріалізації складних програмних систем (ПС) із готових простіших модулів і програм у МП 4GL виникла і в крупних західних фірмах (SUN ONC IBM, CDC, SDS, UCSD, General Electric, Oberon тощо) [10–21]. Тобто зусиллями багатьох працівників ІК і закордонних спеціалістів до 1990 р. було розроблено цільові засоби, необхідні для виробництва ПП на перших «фабриках»:

— системи автоматизованого виробництва програм із модулів повторного використання (МПВ) за принципом збирання частин ПП або в цілому, подібно до того, як це робили на той момент в автомобільній чи авіаційній промисловості;

— лінії виробництва окремих ПП або їхніх сукупностей, що регламентують випуск деякого виду продукту [9];

— окремі технічні, програмні інструменти, засоби й механізми для підтримки збірки, а саме: паспорти модулів із описом зовнішніх даних у мовах інтерфейсу MIL, MESA, АЛМО тощо, необхідних у «стиковці» (збірці) різномовних програм між собою із використанням бібліотек програм і функцій перетворення нерелевантних у МП і модулях типів даних тощо;

© *АНДОН Пилип Іларіонович. Академік НАН України. Директор Інституту програмних систем НАН України.*

ЛАВРІЩЕВА Катерина Михайлівна. Доктор фізико-математичних наук. Завідувач відділу програмної інженерії цієї ж установи (Київ). 2010.

— середовища підбору з бібліотек або фондів готових МПВ, їх автоматизоване збирання чи інтеграція у нові програмні конструкції (комплекси, агрегати, системи тощо) спеціального призначення для виконання деяких задач підприємств.

Прикладом апробації ідеї ліній виробництва як фабрики програм (уводу, виводу даних, пакетів прикладних програм, розв'язання наукових задач статистики, числення тощо) була АІС «Юпітер» із чотирьох технічних об'єктів Морського флоту СРСР, розроблення якої очолював П.І. Андон. На основі спроектованих ліній виробництва для цієї системи побудовані за методом збірки багато програм обробки даних. Але у зв'язку із розвалом СРСР роботи з удосконалення ліній виробництва для інших галузей з 1991 р. були зупинені, але продовжувались теоретичні дослідження [13–21].

При розробленні цих засобів чимало спеціалістів дійшли висновку, що загальною ще слабко розв'язаною проблемою збірки різнорідних програмних об'єктів у МП є **інтерфейс** (системний, прикладний, теоретичний), який повинен забезпечувати обмін даних між різнорідними програмами й комп'ютерами. Тоді зміст інтерфейсу складали практично в кожній системі по-різному. Нині триває його розвиток і вдосконалення, вчені створюють нові мови інтерфейсу (IDL, API, SIDL, DII), стандарти фундаментальних (Fundamental Data Types — FDT [20–25]), загальних (General Data Types — GDT стандарту ISO/IEEC 11404 [26]) типів даних (ТД) МП і форматів даних (OSI, RDF, XML, OSWL тощо [27]).

Період 1992–2010 рр. характеризується розвитком не лише мов опису інтерфейсів, але й засобів їхньої підтримки для забезпечення обміну даних між модулями в різних середовищах шляхом узагальнення раніше сформульованого інтерфейсного посередника новими його видами (P-код, stub, skeleton, adaptor, glue, slim, DSI), а також через конфігураційні, спискові файли, інтерфейсні карти й т.п. Удосконалено розподілені середовища (COM, CORBA, MS. VSTS, JAVA-машина, Grid, Babel) [21], відновлено лінії виробництва (Product Lines, Programs Factories), розроблено нові програмні ресурси (reuses, assets, Domains, Applications) й інформаційні ресурси в Інтернеті, чимало інструментів їхньої підтримки [20, 21, 27, 28].

Усе це сприяє переходу до індустріального виробництва різноманітних видів ПП конвеєрним методом, появу якого прогнозував акад. В.М. Глушков.

Таким чином, загальними практичними зусиллями багатьох програмістів було визнано, що головне місце в індустрії ПП займає **операційне середовище**, яке устатковане сучасними *комп'ютерами, МП, інтерфейсами, типами даних FDT і GDT; інструментами трансформації і взаємодії різнорідних програм.*

1. АНАЛІЗ ДИНАМІКИ РОЗВИТКУ ФАБРИК ПРОГРАМ У СВІТІ

Аналіз фабрик програм проведено, починаючи з того часу, коли цю ідею висловив акад. В.М. Глушков. Особливу увагу було приділено розробленим середовищам і системам з автоматизованим виробництвом різного роду ПП із різних елементів збірки, створених «ручними», автоматизованими або частково автоматизованими засобами. Це насамперед:

1. Система АПРОП в ІК, що працювала в середовищі ОС ЄС і об'єднувала за методом збірки різномовні модулі в МП 4GL через інтерфейсні міжмовні й міжмодульні посередники [2, 6, 9, 10, 19–21];

2. Інтегроване середовище Sun Microsystems IBM зі збіркою різномовних про-

грам у 90-х рр. XX ст. і подальшим розвитком нових напрямів виробництва ПП, зокрема, моделлю архітектури SOA, Web-сервісами, новими мовами Ruby, Script [29];

3. Клієнт-серверне розподілене середовище CORBA OMG із віддаленою взаємодією клієнта й сервера у мережі через модулі-посередники, які отримали назву «заглушок» — stub (для клієнта); skeleton, adaptor, dynamic interface Dill (для сервера) і передають зовнішні дані із запитом брокеру від клієнта до сервера і зворотнім [30];

4. Фабрика «ручної» збірки різномовних програм Інг Бей із використанням кількох засобів взаємодії різнорідних програм (інтерфейсні посередники, командні рядки, конфігураційні файли, інтерфейсні карти) у середовищах (VC++, VBasic, Matlab, Java, Visual Works Smalltalk тощо) [30];

5. Фабрики програм Дж. Грінфільда з технологією побудови складних бізнесних програм із використанням методології use case (UML), ліній виробництва й модельного підходу SOA [32, 36];

6. Колективне міжнародне середовище MS.VSTS для виробництва ПП широкого призначення за участю виконавців, що можуть звертатись через Інтернет до цього середовища із різних місць [33];

7. Інфраструктура розроблення, тестування, збірки наукових програм, систем і пакетів для подальшого обчислення у міжнародній мережі Європейського проекту Grid [35, 36] тощо.

Проаналізовано ці фабрики програм із визначенням різних технологічних джерел, інструментів і важливих атрибутів у табличному поданні середовищ (*платформа, МП, зв'язки, засоби, FDT, трансформація, інтерфейс* тощо).

1.1. Аналіз автоматизованої збірки ПП у вітчизняній системі АПРОП

Перші системи автоматизованої побудови ПП в Україні були системи [1–10], зокрема АПРОП [2, 6], яка забезпечувала збірку різномовних програм у агрегатну монолітну структуру з інтерфейсними посередниками в мові типу MIL для кожної пари таких програм, призначених для розв'язання різних наукових, прикладних і господарчих задач.

Табл. 1. **Основні засоби середовища АПРОП**

1980 –1991	
Середовище	ОС ЄС
Платформа	ЄС ЕОМ
Мови	Алгол, ПЛ/1, Кобол, Фортран, Асемблер
Зв'язки	CALL, ММК
Засоби	АПРОП, Банк модулів, збірка
Типи даних	FDT і ТД МП
Трансформація	Бібліотека функцій міжмодульного і міжмовного інтерфейсу, Банк готових модулів
Інтерфейс	Посередники із зовнішніми ТД у ММК АПРОП

Генерацію посередників виконував технологічний модуль за допомогою спеціально розробленої бібліотеки інтерфейсних функцій (64) та перебудови нерелевантних FDT типів даних, що передаються через CALL-виклик в усіх МП 4GPL [2, 6, 9, 10].

Було вироблено зміст поняття *інтерфейсу*: міжмовного й міжмодульного для FDT МП і паспорта готових модулів, опис яких був надалі вдосконалений у мовах IDL і API. Як фабрика система базувалась на готових модулях, які були розроблені для методів чисельного аналізу і зберігалися в Банку модулів. Крім того, аналогічні рішення відносно концепції інтерфейсного посередника були отримані і в інших системах, що були створені в ІК. Але розроблені в межах системи АПРОП функції відображення нерелевантних типів даних різномовних модулів в обраному класі МП ОС ЄС були практично використані в 52 організаціях СРСР [2, 6, 20], а метод збірки, розробленої в системі АПРОП, став основою збіркового програмування, застосованого в сучасному виробництві ПП на фабриках програм [10, 9–21].

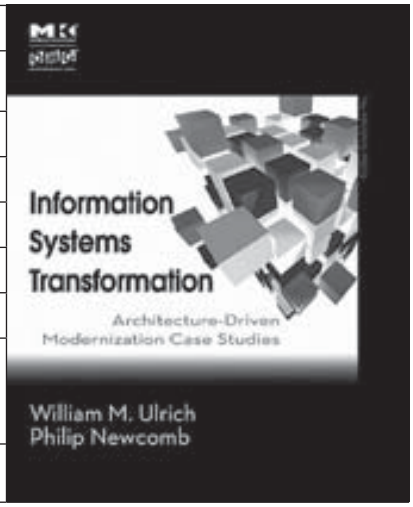
1.2. Аналіз середовища збірки ПП у Sun Microsystems IBM

IBM-середовище забезпечує взаємодію тих самих МП, що АПРОП. Воно використовує ОС: MVS, VM, OS/2, AIX. У ньому механізм виклику і принципи об'єднання різномовних програм аналогічні. Опис зовнішніх інтерфейсних FDT різномовних програм виконано мовою MIL та Асемблера як проміжного для багатьох ПП [29].

Інтеграцію (збірку) різних програм виконано на загальній платформі IBM. Для роботи з даними різної структури застосовано системи БД (IMS, CICS) з ієрархічною і мережною структурою. Багато питань сумісності типів даних розв'язано транзакціями з БД. Нині середовище *IBM* поповнено новими засобами, зокрема мовою Ruby й Web-сервісами. Основу технології збірки становить механізм заглушок, подібно до системи CORBA. Ці системні можливості адаптовані на платформах процесорів Intel, Linux з мовою Scilab і пакетом відповідних розширень (Toolbox). До середовища IBM додано новий засіб — WebSphere Application Server Compunity Edition, який забезпечує сервісні застосування на сучасних компонентах для виконання різних

Табл. 2. **Основні засоби IBM-середовища**

1980–1991	
Середовище	ОС 360-390, ONC (Sun Microsystems)
Платформа	IBM Solaris Sparc, Intel, Linux Itanium
Мови	Алгол, ПЛ/1, Кобол, Фортран, Асемблер
Зв'язки	CALL, CALLP, MIL-генератор, SAG
Засоби	MVS, VM, OS/2, AIX, Open source
Типи даних	FDT і ТД МП
Трансформація	XDR-бібліотеки, бібліотека готових модулів, Sun Workshop, Toolbox
Інтерфейс	Посередники в MIL, Асемблері



завдань і сервісів. Ураховуючи популярність програм з Windows, у системі Unix адаптували програми у мовах C, C++ з портуванням їхніх даних через AIX. Зараз система IBM постачає новітні технології та засоби IBM Factories для індустріального виробництва програм більш ніж у 100 країн, зокрема в Росію.

1.3. Аналіз механізму збірки у клієнт-серверному середовищі CORBA

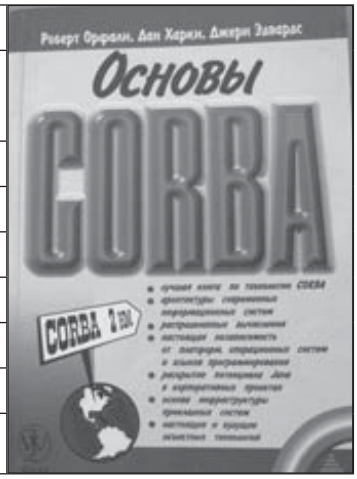
OMA-архітектура посилила можливості зв'язків у звичайних програмних об'єктах у МП і в об'єктно-орієнтованому програмуванні. В межах цієї архітектури виник ширший клас інтерфейсних посередників (stub, skeleton, adaptor, dill, service, client-interface, server-interface), опис яких подано за сучасними мовами інтерфейсу — IDL, API, DII для різних МП, програми з яких можуть виконуватись у клієнт-серверній архітектурі системи з передачею даних через протокол IIOP об'єктному брокерові ORB, який забезпечує обмін інформацією між клієнтом і сервером, а також подання різних видів сервісів цим програмам із горизонтальним і вертикальним призначенням (табл. 3). Принципи конструювання розподілених об'єктів із забезпеченням механізмів взаємодії в середовищах OMG/CORBA, Microsoft/COM і Java/RMI подано в [29] з обґрунтуванням проміжного прошарку брокера ORB як інструмента керування розподіленими об'єктами і завданнями з усунення неоднорідності різнорідних платформ при вирішенні на них інтерфейсів програм між клієнтом і сервером засобами брокера ORB.

1.4. Фабрика напівавтоматизованої збірки програм за Інг Бей

На цій фабриці використано різні середовища (VC++, VBasic, Matlab, Java, Visual Works тощо) з відповідною взаємодією різномовних компонентів, частково автоматизованими інструментами й засобами (Domain and Application Models, Model Interconnection, Microsoft Foundation, асемблування) виготовлення ПП (табл. 4) [31]. Розглянуто різні варіанти зв'язків і конкретні приклади для кожної пари програм у МП з відповідними функціями перетворення обмінних типів даних шляхом звернення до них операторами на різних МП як механізмах прямої та оберненої взаємодії різномовних і різноплатформених програм. Для кожної пари компонентів у МП викладено техніку передачі даних через віддалений

Табл. 3. **Основні засоби CORBA-середовища**

1991–2010	
Середовище	ORB, COSS, DCE/RPC, PCTE, ToolTalk, Java2 SDK, NetPilot CCS
Платформа	OMA-архітектура (Apple, IBM, Win-NT, x-Open, Dec)
Мови	C, VC++, VC#, Smalltalk, Java, Кобол, Visual Basic, Ada-96
Зв'язки	IDL, API, DII, Client-interface, Server-interface, TCP/IP
Засоби	CORBA, OLE/ DCOM, SOM/ DSOM (IBM), OSF DCE
Типи даних	FDT, ТД сучасних МП
Трансформація	MIL, IDL, DLL, ORB, Borland Jbuilder
Інтерфейс	Посередники — stub, skeleton, service, adaptor



виклик і з можливим перетворенням несумісних переданих типів даних за відповідними фактичними параметрами і принципами розгортання цих компонентів.

На відміну від загальної схеми взаємодії різномовних програм за інтерфейсними посередниками в даному керівництві подано нові технологічні засоби перетворення типів даних за допомогою користувальних панелей, сценаріїв, іконок, зразків інтерфейсних програм тощо. Особливості зв'язків пар різнорідних програм у різних МП наведено далі.

Інтерфейс між Visual Basic та іншими *МП* реалізовано операторами звернення з такими параметрами, як текстові рядки, значення, масиви тощо. Їх обробку виконано спеціальними функціями Windows API, DLL і функціями відтворення нерелевантних типів даних, що передаються різним програмам у цих мовах.

Matlab як середовище розв'язання задач лінійної та нелінійної алгебри, матричних операцій, різних математичних методів обчислень містить такі інструменти: Matlab Compiler, Matlab C++, Matlab Library, Matlab Graphic Library. Інтерфейс за зверненням реалізовано MatlabCompiler шляхом відображення типів даних у відповідні формати мови С — М-файли чи М-функції, які перетворюються до формату даних архітектури інших комп'ютерів.

Smalltalk — інтегрована система програмування, що забезпечує створення різних застосувань у середовищі VisualWorks за допомогою моделей застосувань із функціями DLL із класу зовнішнього інтерфейсу бібліотеки С++, методів об'єктів і різних повідомлень, що містять фактичні значення параметрів для передачі іншим об'єктам.

Система LabView — набір виробничих процесів розроблення, тестування, збирання даних із вимірюванням та оцінкою взаємодіючих даних з апаратурою платформи за допомогою інструментів ANS C, Visual Basic, Visual C++, Lab Windows/CV, а також вимірювання параметрів термометрів і перемикачів у реальному часі для їх передачі через мережу.

Табл. 4. **Основні механізми «ручної» збірки програм**

1996–2005	
Середовище	VC++, Vbasic, Matlab, Java, Visual Works
Платформа	MS.Net, HP, Apple, IBM...
Мови	C, C++, Visual C++, Visual Basic, Matlab, Smalltalk, Java, LabView, Perl
Зв'язки	RPC, Active, рядковий формат, dll, External interface class
Засоби	Domain and Application Models, Model Interconnection, Microsoft Foundation, інструкції збірки для різних середовищ
Типи даних	FDT, нові ТД, Setup, Start, Exit
Трансформація	Збірка (лінковка), інтеграція кодів
Інтерфейс	RMI, JNI, exe-файл, інтерфейсна карта

Система Java містить набір інструментів зв'язку з мовами JAVA, C, C++ і засобами Java Native Interface на віртуальній машині VM з використанням інтерфейсних функцій і бібліотек класів C і C++. У межах цього середовища наведено приклади взаємодії різномовних компонентів у цих МП.

Мова Perl включає засоби опису сценаріїв для взаємодії з Інтернетом, керування завданнями, створення CGI-сценаріїв для сервера системи Unix, інтерфейс із мовами C, C++, Visual Basic, Java, а також з іншими мовами за процедурами, операторами виклику в C або C++ і функціями з динамічної бібліотеки. Перетворені дані у спеціальному коді зберігають у бібліотеці інтерпретатора Perl.

На цій фабриці використано найсучасніші засоби й інструменти різних середовищ підтримки взаємодії різномовних програм у МП, надано рекомендації щодо прямої та оберненої передачі даних через параметри звернення до інших готових програм.

1.5. Фабрики бізнесних програм Дж. Грінфільда

Для майбутньої фабрики автор сформулював методологічні й технологічні аспекти виробництва ПП за методом UML, моделями архітектури систем і комп'ютерів, механізмами інтеграції різнорідних компонентів з використанням багатьох типів даних FDT і мов взаємодії IDL, XML, RDF тощо [32]. Головні концептуальні ідеї цієї фабрики такі (табл. 5):

— еволюція, змінювання готових reuses, активів, програм і систем під цілі замовника ПП;

— програми із класифікаційними ознаками категорій ПП, які накопичують на фабриці в загальноприйнятих сховищах (бібліотеках, репозитаріях);

— економіка систематичного повторного використання;

— моделі, шаблони, інструменти, принципи побудови ПП з використанням методу UML;

— лінії ПП за технологічними діями (аналіз проблеми, вимоги, специфікації, дизайн, реалізація, тестування, інсталяція, сертифікація ПП), сервісне обслуговування фабрик (веб-сервіси, служби якості, контролю робіт і планів);

— демонстрація результатів фабрики розроблення ПП за Product line на прикладі бізнес-магазину.

Табл. 5. **Основні засоби фабрики програм за методом UML**

2004–2010	
Середовище	Web-сайти, MS VS, IBM Web Sphere, MS Business Framework, Product Lines, Web-служби й сервіси
Платформа	Класи, каркаси, шаблони різних комп'ютерів
Мови	UML, DSL, GSS, OOP, OSL, XML, RDF
Зв'язки	IBM Rarional Rose XDE, MDD, MDA, класи, Core J2EE Design
Засоби	Генерації, збірки, розгортання, тестування інструментами фабрики, Java-VM
Типи даних	FDT
Трансформація	Збірка, генерація ПП на ТЛ
Інтерфейс	RMI, RPC, IDL, API

Результати практичного моделювання UML для бізнес-застосувань дозволили авторові зробити такі висновки:

— мова UML — це мова ескізу ПП за різними сценаріями;

— відсутність можливості використання в UML компонентів повторного використання (КПВ) і сценаріїв із програм на інших МП;

— неможливість застосовувати нові моделі MDA, SOA, посилання до типів даних FDT, прийнятих у сучасних МП, і звернення до новітніх загальних типів даних GDT [21—26];

— UML важкий для модифікування ПП фабричними засобами через відсутність механізмів забезпечення їх змінюваності.

Отже, запропоновано макет фабрики програм із використанням сучасних Product lines, які повинні мати:

— мови, шаблони, каркаси, моделі;

— інструменти, автоматизовані засоби наповнення каркасів;

— збірку програм незалежно від платформ комп'ютерів із використанням МП і ТД (XML, WSDL, RDF тощо);

— сертифікацію залежностей, перевірку функціональних і нефункціональних вимог до ПП;

— використання аспектів безпеки, захисту, синхронізації задля їх додавання до побудованого ПП;

— можливість зміни окремих компонентів, архітектури, даних, що використовують різні ТД або БД.

Фактично меморандум сучасної фабрики ПП Дж. Грінфільда такий:

— архів МПВ чи бібліотек повинен бути великим для відбору потрібних артефактів на фабрики програм;

— збірка за виробничими лініями повинна бути як у Чернецкі [37, 44] стосовно сімейств систем, простору проблем, рішень і забезпечення їхньої якості;

— застосування нових моделей систем, каркасів процесів і ПП відповідають новим рисам проектування й виробництва продуктів на сучасних фабриках програм.

Головний висновок автора — фабрика програм за методом UML має багато проблем при індустріальному виробництві ПП та їхньому супроводі, особливо це торкається забезпечення механізмів змінювання й еволюції створених за UML продуктів.

1.6. Фабрика програм фірми Microsoft

Середовище MS.NET — сучасна фабрика програм з усіма атрибутами збіркового виробництва ПП. Насамперед, середовище найліпше оснащено, в ньому забагато готових системних і прикладних ресурсів (компонентів, сервісів, Domains, Applications тощо); МП (JAVA, C++, Basic, Java, Pascal, C#, SpecC#); механізмів RMI, RPC виклику компонентів із загальними бібліотеками CLR (Common Language Routine) і FCL (Framework class Language); інструментів збірки проміжного чи вихідного кодів (exe, dill) у готовий ПП; Веб-сервісів різного призначення для виробництва програмних проектів; пакетів VSTS з керування конвеєрним методом командами розробників, члени яких можуть бути в різних місцях інформаційного світу (табл. 6) [33].

Головні засоби середовища MS.NET для автоматизованого виробництва ПП:

— пакет інструментів VSTS (Visual Studio Teams System), орієнтований на розроблення великих проектів за участю різних спеціалістів (аналітиків, менеджерів, тестувальників, програмістів, кодувальників тощо) із різних організацій;

— методологія MSA (Microsoft Solution Architecture), призначена для побудови виробничої архітектури підприємства за допомогою ЖЦ розроблення ПП (Software Development Life Cycle), стандарту PMBOK, моделей перспектив і процесів;

Табл. 6. **Оргструктура середовища конвеєрного розроблення ПП у MS.VSTS**

2003–2010	
Середовище	OS 2000 XP/ME, MS.NET-server, SQL-server, Visual Studio Teams System.NET, MCSD, MSF
Платформа	MS.NET (8, 16, 32, ..., 128 байт)
Мови	C, C++, VBasic, Java, Pascal, C#, SpecC#
Зв'язки	SQL-request, Web-Forms, Web-services, EXE-файли, dll
Засоби	SDLC, IDE, MS Office, MS SQL server, MS VS
Типи даних	FTD, GDT, CTS (Common Type Systems)
Лінкування Збірка	CLR-бібліотеки, класи, FCL-типи, трансформатори
Інтерфейс	General code (EXE), Portable Executable code

— системи Professional Studio, Foundation Server для підтримки процесів проектування, кодування, тестування, формування версій ПП тощо;

— CMMI Process Improvement для регулювання термінів розроблення за технологією Agile тощо [58].

Пакет VSTS — це сімейство системних засобів проектування ПП, забезпечення взаємодії виконавців різних частин ПП і продукту в цілому, а також підбору КПВ, розподілу й виконання окремих робіт із виготовлення ПП. Виконавці проектів поділені за чотирма категоріями з урахуванням рівня знань, здібностей у програмуванні, проектуванні, вимірюванні, оцінюванні ПП тощо. Перехід із нижчої категорії на вищу можливий при якісному виконанні завдань або додатковому навчанні. Спеціалісти четвертої групи — головні в команді, відповідають за функціональну правильність виконання вимог і ПП.

До основних компонентів продукування ПП у середовищі VSTS належать:

— *Microsoft SQL, Server* зберігають усі робочі результати, вихідний код і дані інтеграції Team команди за допомогою засобів Веб-порталу з сервісами, а також інструментів *Analysis*, *Reporting Services* і *SharePoint Portal Services*;

— *Microsoft Project*, *Excel* використовують як альтернативні засоби для керівників проекту ПП.

Засобом підтримки процесу розроблення окремих елементів ПП є IDE (Integrated Development Environment), Microsoft Office Project, Microsoft Office Excel, що перевіряють вимоги, відстежують проміжні процеси вироблення й кінцеві результати з оцінюванням ступеня готовності. Тестування окремих елементів ПП здійснюють засобами, що інтегрує Visual Studio.

Таким чином, середовище VSTS виступає прикладом колективної фабрики виробництва ПП. Для цього є багатий набір інструментів керованої підтримки процесів ЖЦ за графіком робіт, а також його відстеження, оцінювання результатів на якість, вартість і терміни виготовлення ПП.

1.7. Інфраструктура збірки й обчислень у системі Grid

Європейський проект Grid — мережна інфраструктура для організації розподілених обчислень задач із різних наукових напрямів (фізика, математика, медицина, біологія) [34, 35]. До його складу входить GCube — операційне середовище, ETICS — збіркове середовище тощо.

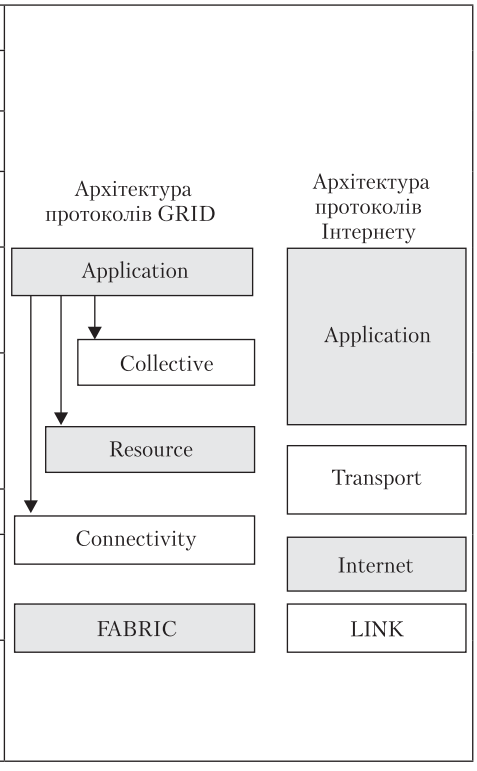
ETICS за функціями найближчий до сучасної фабрики програм, базується на наборах характеристик, послуг і процедур виготовлення пакетів. Вони можуть об'єднуватись плагінами (plugins) [34] з описом послуг для споживачів або постачальників, засобами керування завданнями з робочих місць, а також доступу до ОС, архітектури CPU, компіляторів з МП і засобами специфікації залежностей між різними пакетами та їхніми тестами при збірці й розгортанні. Множина функціональних плагінів забезпечує перевірку контрактів, тестів виконання різних елементів систем, генерацію документації, ведення готових програм в оперативному або постійному репозитарії ETICS.

Технологію створення великих наборів пакетів із вихідних або сукупностей перекомпільованих програм забезпечено процесами доступу до репозитаріїв, формування розподілених версій та їх репродукцій. Головним гальмом є перебудова деяких компонентів систем для альтернативної платформи гетерогенного середовища комп'ютерів шляхом посилань з 16-, 32-розрядної платформи на 64-розрядну платформу середовища Grid.

Головна проблема споруджених у системі ETICS об'єктів (Проект, Підсистема й Компонент) у стандартизації опису типів даних. Модель даних із типовим форматом CIM для взаємин між цими об'єктами

Табл. 7. **Система збірки й обчислення — Grid**

2003–2010	
Середовище	Grid (Collection Instuments and Tools)
Платформа	Intell, Sun, IBM, Apple, MS.Net, Кластери
Мови	4-5 GPL C, C++, Visual C++, Visual Basic, Smalltalk, Java, LabView, Perl, Java, Pyton
Зв'язки	RPC, Active, RMI, CRL, SiDL, API, інтерфейсний, конфігураційний файли, dll-data
Засоби	OGSA, SDK, Protocol NFS, User Domain and Application, HTTP, Model Interconnection, MS Foundation, репозитарії ресурсів, OSI
Типи даних	FDT, GDT (стандарт ISO/IEC 11404)
Збірка	Взаємодія комп'ютерів, мережні ресурси, протоколи, МПВ і збірка вихідних кодів систем, підсистем, компонентів і пакетів
Інтерфейс	RMI, JNI, exe й конфігураційні файли, інтерфейсні карти, SIDL (Babel), FDT, GDT



включає їхній опис зі зверненнями між ними за таких умов:

— кожен компонент має властивості (ім'я, ліцензію, URL, репозитарії), глобальний унікальний ідентифікатор — GUID, команди перевірки скомпільованого компонента, тестові команди, зв'язки з конфігураційним файлом версії системи;

— конфігураційний файл окремого компонента чи системи містить інформацію про версію, зв'язки з репозитарієм, GUID, про види платформ і зв'язків із глобальними компонентами і пакетами для обчислень;

— між двома різними конфігураціями встановлено статичну й динамічну залежність даних і компонентів.

У середовищі Grid на першому плані ресурси в широкому розумінні, а саме: динамічне керування застосуваннями й сервісами в кожний момент часу при глобальному обчислюванні наукових завдань. Ресурси — це різні програмні артефакти, засоби комунікації, інформаційні системи, СПС, системи збереження даних, СКБД, програмні фонди із різних доменів, технічні, комп'ютерні, людські ресурси тощо. Ресурс подає логічний файл або фізичний об'єкт (кластер, комп'ютер, суперкомп'ютер тощо). Їхні протоколи (NFS) завантажуються системою Grid і забезпечують моніторинг, керування, доступ до локальних ресурсів фабрики програм з метою отримання інформації про безпеку, захист, взаємозв'язки тощо.

Глобальні задачі в Grid взаємодіють між обчислювальними вузлами мережі через протоколи (Resource, Connectivity) при виконанні пакетів завдань, якими керує служба диспетчеризації системи. На основі стандартних протоколів будують сервіси, інтерфейси в API, засоби розроблення систем SDK (Software Development Kits). Збірку програм на канальному рівні з різних фабрик, а також виконання компонентів, підсистем і систем наукового призначення в різних МП реалізують глобальні протоколи (Global Protocols). Тобто ця мережа зорієнтована на майбутні обчислення з побудованих різними науковцями ПП.

Крім розглянутих видів фабрик є й інші: інтегроване середовище **Eclipse** [45]; система **Oberon** [46], яка з 90-х рр. виконує збірку різнорідних програм і підключила на цей час нові МП для опису й виготовлення ПП на комерційній основі; а також поповнення цього середовища новою мовою наукового інтерфейсу **SIDL** (Scientifically IDL) для забезпечення взаємодії наукових програм у мовах C, C++, Pyton, Java через прошарок типу glue для платформ Linux, AIX, Solaris, IBM з орієнтацією названої мови на майбутнє використання в інфраструктурі проекту Grid.

Підсумком виконаного аналізу фабрик програм є опис фундаментальних робіт із теорії і практики виробництва ПП за участю співробітників ІПС НАН України протягом багатьох років, визначення структури, змісту й організації керування сучасною фабрикою програм, а також упровадження глобального проекту фабрики в Grid для перспективного її застосування у наукових підрозділах НАН України.

2. РОЗВИТОК ТЕОРЕТИЧНИХ ОСНОВ ВИРОБНИЦТВА ПРОГРАМ В ІПС НАН УКРАЇНИ

Концепція виробництва програм протягом останніх десятиріч отримала теоретичний розвиток у дисертаційних і фундаментальних дослідженнях проектів ДКНТ (1992–1996) і наукових проектах ІПС НАН України (1997–2011). Дослідження й розробки забезпечили розвиток теоретичних методів проектування, збирання різнорідних компонентів, проведення експертиз, тестування й оцінювання якості кінцевого продукту. Саме тут спеціалісти інституту отримали оригінальні

результати, поповнили теорію виробництва ПП новими формальними засобами й механізмами, яких бракує зарубіжним публікаціям.

Створення теорії об'єктного й компонентного програмування. У межах наукових проектів інституту [10, 21, 48, 49] і збіркового програмування побудовано об'єктно-компонентний метод (ОКМ) із використанням теорії Фреге як апарату узагальнення поняття об'єкта формальними властивостями, характеристиками і з математичними формалізмами для уточнення окремих і загальних рис об'єктів та їх відмінностей.

Основу *об'єктної теорії* становить алгебра аналізу $\Sigma=(E', \Psi, P)$ *з* $E'=(E_1, E_2, ..., E_n)$ — множини об'єктів, операції $\Psi=\{decds, decdn, comds, comdn, conexp, connar\}$ над елементами E' і множина предикатів $P=(P_1, P_2, ..., P_r)$ завдання концептів об'єктів. Алгебра Σ — система операцій аналізу, функцій деталізації, екземпляризації, агрегації об'єктів. Розроблено формальний механізм переходу від об'єктів до їхньої програмної реалізації, тобто до компонентів та інтерфейсів. У результаті компонентне програмування розширено формальними моделями (*компонент, інтерфейс, компонентне середовище),* операціями зовнішньої та внутрішньої компонентної алгебри із засобами перетворення нерелевантних типів даних різномовних компонентів [14, 20, 21].

Зовнішня алгебра $\Psi=\{CSet, CESet, \Omega 1\}$ визначена на множині компонентів *Cset,* середовищі *CESet*, операціях $\Omega 1=\{CE_1, CE_2, CE_3, CE_4\}$ розгортання, об'єднання компонентних середовищ, видалення компонента із середовища, заміщення компонентів.

Внутрішня алгебра $\varphi=\{CSet, CESet, \Omega 2\}$ визначена на операціях еволюції (рефакторингу, реінженерії, реверсної інженерії) компонентів $\Omega 2=\{O_{refac}, O_{Reing}, O_{Rever}\}$.

Компонентна алгебра $\Xi=\{\Psi\cap\varphi\}=\{CSet, CESet, \Omega 1\}\cap\{CSet, CESet, \Omega 2\}=\{CSet, CESet, \Omega 1, \Omega 2\}$ — необхідний інструмент взаємодії, еволюції різнорідних компонентів для сучасних середовищ. Взаємодію забезпечено оригінальною теорією перебудови типів даних різномовних програм, створеною в рамках системи АПРОП [10] і розвинутої стосовно індустрії виробництва ПП на фабриках програм [21].

Формально модель збірки різномовних компонентів, заданих на множині компонентів $CSet=\{Comp_i\}$, забезпечує взаємодію компонентів через обмін даних, кожне з яких є трійкою з: ім'ям змінної, типом даного, значенням. При обміні дані кожної пари компонентів можуть бути еквівалентними, коли вони мають однакову семантичну структуру й тип даних, або нееквівалентними, тоді необхідне їх перетворення за наступними функціями відображень:

$FN_{ij}: N_u \rightarrow N_j$, $FT_{ij}: T_i \rightarrow T_j$, $FV_{ij}: V_i \rightarrow V_j$,

де FN_{ij} — задає відповідність між іменами змінних на множині формальних і фактичних параметрів, FT_{ij} — це опис еквівалентних відображень типів даних, FV_{ij} реалізує необхідні перетворення значень даних нееквівалентних типів.

Задачу побудови перетворень FN_{ij} вирішують шляхом упорядкування імен змінних. Відображення між типами даних FT_{ij} подано як абстрактна алгебраїчна система $T=(X, \Omega W)$, де X — множина значень для цього типу, а ΩW — множина операцій над цими змінними. Відображення FV_{ij} використовують у випадку нееквівалентності типів T_i і T_j.

У випадку багаторазового виклику компонентів виконують пряме і зворотне перетворення формальних і фактичних параметрів за умови ізоморфізму відображень між алгебраїчними системами опису типів даних.

Під *перетворенням типу* $T_i=(X_i, \Omega W_i)$ у тип $T_j=(X_j, \Omega W_j)$ розуміють таке перетворення, при якому семантичний зміст операцій із ΩW_j еквівалентний змістові опера-

Рис. 1. Інженерія тестування, якості й оцінки процесів ПП

ції із ΩW_j. У випадку односторонньої перебудови типу T_i у тип T_j еквівалентність перетворення не існує.

Задача забезпечення взаємозв'язку пари компонентів, розроблення котрих виконане на різних МП, полягає у побудові сукупностей відображень для всіх викликів з метою встановлення однозначної відповідності між множиною фактичних параметрів $V=\{ v_1, v_2 ,..., v_k\}$ і множиною формальних параметрів $F=\{f_1, f_2 ,..., f_l\}$ компонентів.

Для типів даних FDT МП у роботах [14, 20, 21] визначені алгебраїчні системи для FDT даних, доведені ізоморфні відображення між ними і сформульовані нові умови генерування деяких типів даних GDT.

Нові засоби перевірки правильності компонентів. Тестування компонентів і ПП як метод виявлення помилок, дефектів, відмов і збоїв, викликаних різними нерегулярними ситуаціями, аварійним припиненням роботи деяких компонентів чи всієї компонентної системи, стало наступним головним напрямом досліджень. Так, у дисертаційній роботі Т.М. Коротун [50, 16] запропоновано (рис.1):

— математичну модель визначення оптимального часу тестування компонентів t_e^* з максимальним прибутком $K(t_0|t_e)= \Delta R(t_0|t_e) - C(t_e)=C_м(\mu(t_0) - \mu(t_0+t_e)+\mu(t_e)) - c_1t_e - c_2\mu(t_e)$, з похідною $K(t_0|t_e)$, яка дорівнює $K'(t_0|t_e)=C_м(\lambda(t_e) - \lambda(t_0+t_e)) - c_1 - c_2\lambda(t_e)$ в залежності від функції зростання надійності, інтенсивності $\lambda(t)$, ризику $C_м$;

— метод оцінювання ризиків відмов компонентів під час експлуатації системи;

— технологічний модуль тестування ПП і збору інформації про всі помилки для посилення надійності програм СОД.

Дослідження перевірено у проектах СОД МО України й удосконалено щодо СПС у проекті генерувального програмування (2007–2011).

Методи оцінювання якості ПП. Уперше про забезпечення якості заговорили в 1968 р. на конференції НАТО з програмної інженерії. Відтоді дослідження отримання ПП із гарантованою якістю виконували за кордоном і в Україні в межах науково-дослідних проектів ДКНТ, Міністерства науки, НАН. Одночасно створювали стандарти з якості в рамках інформаційних технологій та програмної інженерії, зокрема, в ІПС гармонізовано стандарти ISO/IEC 12207, серія ISO/IEC 14598 «Оцінювання програмного продукту», ДСТУ ISO 15939 «Процес вимірювання», серія ISO/IEC 15504 «Оцінювання процесів ЖЦ ПЗ», базові стандарти з якості — ISO 9001 «Системи управління якістю. Вимоги», ДСТУ 2844–94, ДСТУ 9126 «Якість ПП», ДСТУ 2850–94 і стандарти, що регламентують різні аспекти забезпечення якості ПП. Багаторічні дослідження в галузі якості дозволили спеціалістам [13, 16, 51], зокрема Г.І. Коваль [51], розробити в межах дисертації нові формальні методики вимірювання й оцінювання показників якості ПП у класі задач СОД для МО України, отримати такі оригінальні результати:

— модель якості з орієнтацією на оцінку надійності ПП;

— модель розподілу надійності системи з компонентів за функцією корисності ПП $Q_{nc} = \sum_{j=1}^{l} v_j \cdot q_j = \sum_{s=1}^{m} w_s^* \cdot r_s$ залежно від вагомих коефіцієнтів w_s і надійності $q_j = \prod_{n \in E_j} r_n$ окремих компонентів;

— концептуальну модель прийняття рішень із керування якістю, включаючи байєсівські методи, методи систематичного контролю надійності на ранніх стадіях ЖЦ, вимірювання кількісних вимог до надійності, прогнозування дефектів.

Результати досліджень із проблеми забезпечення якості ПП систематизовано колективом авторів у монографії «Основы инженерии качества программных систем» [16], яка є бестселером у СНГ.

Підхід до експертного оцінювання. Забезпечує розв'язання формалізованих задач оцінювання об'єктів виробництва й експлуатації ПП шляхом експертиз. Базується на теоретичному обґрунтуванні методу *експертного оцінювання* об'єктів ЖЦ ПС, а саме:

$$T=<G, et, ch(et)>;\ \varnothing \neq G \in GT;\ et \in ET;$$
$$ch(et) \in CH^{et};\ CH = \cup_{et \in ET} CH^{et};$$
$$ES(T)=<A, d(ca_1, cn_1, ..., ca_n, cn_n), mg, vf>,$$
$$d \in \Delta_{et,ch},$$

де G — цілі розроблення ПС; et — тип оцінюваних об'єктів ЖЦ; $ch(et)$ — цільова характеристика; $A=\{a=(ud,\ t)\}$, $|A| \geq 1$ — альтернативи; d — аналітична залежність $ch(et)$ від її оцінюваних підхарактеристик; $ca_l \in CH^{et}$, $l=1,..., n$ з множини $\Delta_{et,ch}$; $\varnothing \subseteq cn_l$ — джерела контексту оцінювання ca_l; mg — модель експертної групи; $\varnothing vf$ — підстави верифікації оцінок $\{ch(a), a \in A\}$.

Метод оцінювання створює новий процес ЖЦ зі спільним інформаційним середовищем, адекватним потребам і специфіці виробничої діяльності з виготовлення ПП. Розроблений Слабоспицькою О.О. [52] математичний апарат експертиз містить: методичний каркас (цільові функції, механізми реалізації); модель і методи процесу оцінювання (формалізми підвищення якості результатів та їхнього повторного використання); засоби збірки процесів керування розробленням ПП [58–60].

Методи керування проектом. Менеджмент проекту протягом багатьох років був однією з головних проблем індустрії ПП у закордонних роботах, у тому числі стандарті PMBOK (Project Management Body of Knowledge, 2005), який регламентує керівництво командою виконавців програмного проекту як продукту з використанням

загальних методів управління, планування й контролю робіт (стартові операції, планування ітерацій, моніторинг і звітність), керування менеджером проекту ризиками й конфігурацією ПП. Дослідження в цьому напрямі Задорожної Н.Т. дали нові наукові результати [53, 54], а саме:

— формальну модель керування проектуванням інформаційних систем, що враховує матеріальні, фінансові, трудові ресурси, необхідні для розроблення ПП;

— метод формування варіанта плану X робіт проекту за сітковим графіком В, що включає: послідовність робіт $(l_i \in L)$, їхній обсяг q_i і вид W_i, ресурси $R=<R_L, R_S>$ і норми їхнього використання $(NR_i \in NR)$, закон розподілу випадкових величин $F=\{F_1,..., F_r\}$, часу t у плановому періоді $[t_0,T]$, ймовірність закінчення робіт з урахуванням критерію оптимального плану $K(X^*) = \min K(X)$.

Ці результати апробували в системах автоматизації освітніх процесів («Документообіг в освіті України», портали «Діти України», «Вчитель новатор», 2004–2009), а також викладають у вищих навчальних закладах Академії педагогічних наук України.

Розвиток ліній ПП. Технологічні прийоми виробництва ПП із готових компонентів (КПВ, reuses, assets тощо) розроблено ст. науковими співробітниками відділу Коваль Г.І., Моренцовим Є.І., Коротун Т.М. під керівництвом авторів у 80-х рр. XX ст. у проекті «Юпітер» [9, 16, 17, 19], удосконалено в сучасному представленні *лінії продуктів* (Product Lines Practice) [36]. Нині вони наближають до конвеєрного виробництва ПП на задоволення потреб деякого ринку за такими головними вимогами їх побудови:

— завдання обмежень властивостей і характеристик параметрів збірки ПП;

— підбір каркасів, заготовок, засобів, інструментів підтримки технологічного процесу лінії;

— створення онтологій ПрО для об'єктів репозитаріїв КПВ у мовах OSWL, RDF, IDL, адаптованих для збірки готових специфікованих для фабрики компонентів у нові ПП [55, 56] тощо.

Новий підхід до виробництва звичайних ПП на лінії — це використання системи Protégé [57] для подання онтологій нових компонентів і КПВ, а також необхідних плагінів зв'язку із сучасним середовищем Eclipse при їх збірці у складні ПП з контролем робіт і відстеженням ходу побудови продукту; виявлення ризиків, прогнозування вартісних і технічних ресурсів; керування конфігурацією ПП; вимірювання, оцінки якості і сертифікації ПП.

Поповнення середовища Eclipse новими можливостями виробництва СПС. Інтегроване середовище Eclipse базове для фундаментального проекту ІПС (2007–2011) «Розробка теоретичного фундаменту генеруючого програмування та інструментальних засобів його підтримки» [44, 45] й орієнтоване на виробництво застосувань і сімейств програмних систем (СПС з компонентів повторного використання — КПВ) із забезпеченням їхньої інтероперабельності шляхом генерації інтерфейсних посередників при збірці різномовних програм у СПС або ПП (рис. 2).

Основні завдання проекту цієї системи як фабрики програм:

— використання готових компонентів, інформаційних ресурсів Інтернету, модулів, сервісів, аспектів, Legacy-systems і т.п.;

— створення нової концепції генерації КПВ за моделлю GDM [37, 44], збірка ПП за описом специфіки ПрО у мовах окремих доменів DSL СПС;

— представлення онтологічними моделями процесів тестування, експертизи, оцінки якості компонентів і КПВ [57–59];

— орієнтація на МП (C/C++, C#, JAVA, Pascal, Basic, Clear, Ruby), мови опису даних

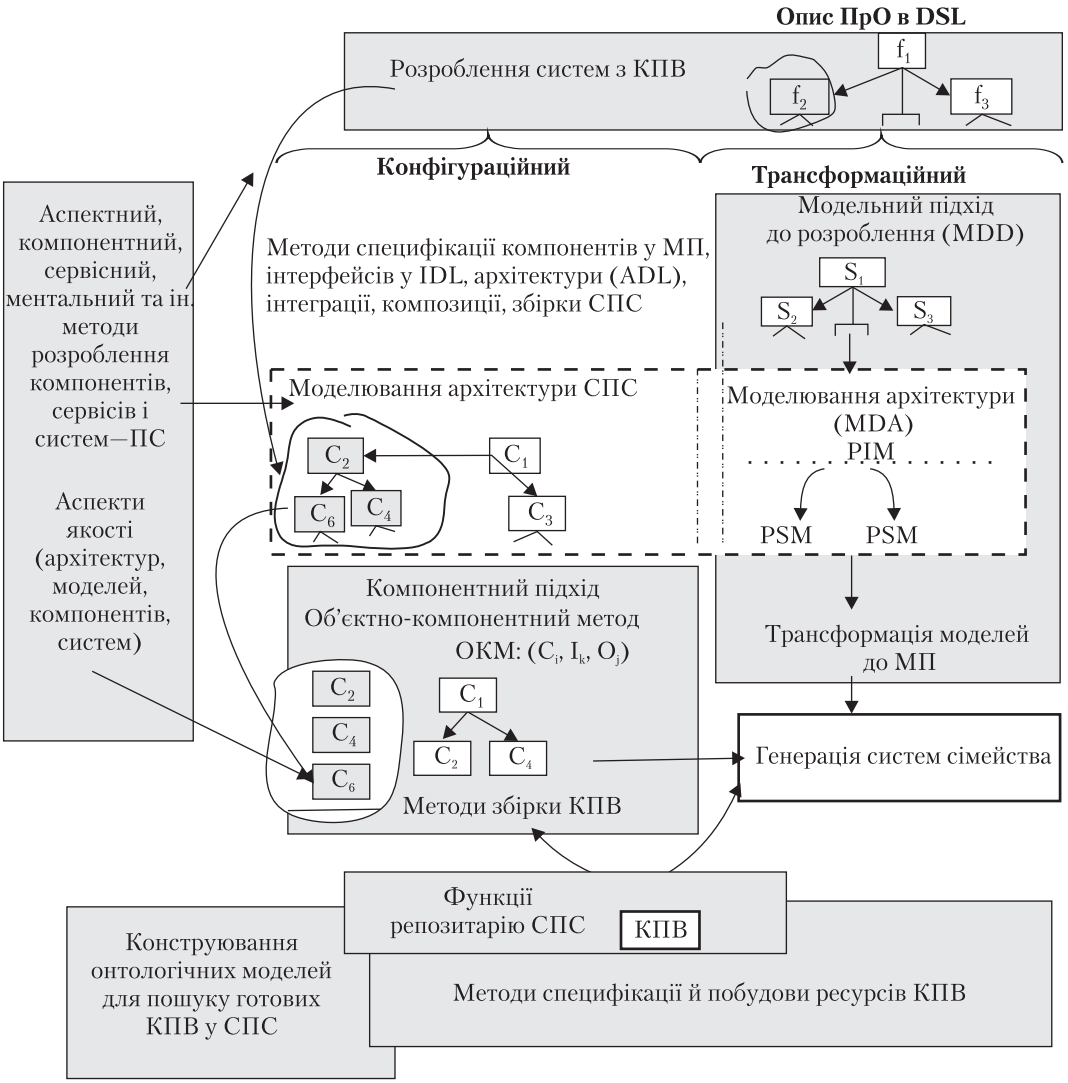


Рис. 2. Інтегроване середовище ГП

(XML, UML, RDF, FDT, GDT), мови взаємодії (IDL, APL, SIDL, протоколи, ін.), застосування сучасних моделей систем (MDA, MDD, PIM, PSM, GDM, DSML, Feature);

— розширення інструментального середовища генерувального програмування репозитарієм із онтологій ПрО, системними засобами (Protégé, Eclipse, Aspectj, Java), прикладними інструментами (ТМ тестування, надійності, експертизи, збірки, конфігурації тощо);

— дороблення теорії експертного та байєсівського оцінювання КПВ, процесів ЖЦ, окремих об'єктів і членів СПС.

На рисунку 2 наведено основні процеси створення ПП (трансформаційний, конфігураційний, або збірковий). Вони базуються на відповідних методах ОКМ перебудови компонентів та їх збірки в СПС.

У цьому середовищі головні технологічні модулі (ТМ) ліній виробництва ПП такі:

ТМ тестування КПВ і різних програм;

ТМ збірки різнорідних програм із забезпеченням відображення несумісних за типом даних;

ТМ експертизи і вимірювання компонентів і членів СПС;

ТМ оцінки надійності за результатами тестування окремих програм і СПС;

ТМ оцінки якісних і кількісних показників компонентів і систем у сімействі СПС;

ТМ сертифікації компонентів і СПС для розміщення їх у глобальному середовищі Grid;

ТМ керування різними ресурсами (технічними, програмними, людськими тощо) середовища.

Цей проект побудови сучасної фабрики програм має загальні риси з інфраструктурою системи Grid і в майбутньому може стати її складовою.

3. ЗАГАЛЬНЕ ВИЗНАЧЕННЯ ФАБРИКИ ЗБІРКИ ПРОГРАМ

З урахуванням досвіду роботи, виконання наукових і прикладних програмних проектів, експериментальних і виробничих фабрик індустріального виробництва ПП з аналізом концепцій збірки ПП у низці монографій [2, 5–21, 28–35] дано визначення фабриці збірки програм. Це інтегрована інфраструктура збіркового виробництва окремих ПП (компонентів, підсистем, систем, модулів, блоків, сімейств систем, АСУ, АСУТП та ін.) із застосуванням:

— ресурсів (програмних, наукових, інженерних, технічних, технологічних, економічних, фінансових, людських);

— середовищ збірки типу SUN ONC, MS.Net, Oberon, Grid, Eclipse.

Охарактеризуємо базові елементи інфраструктури сучасної фабрики програм.

3.1. Ресурси для виробництва ПП на фабриці

Цих ресурсів та їх типів багато, і вони необхідні для побудови й функціонування різних фабрик виробництва програм. Розглянемо їх.

Програмні ресурси. Програма — це послідовність команд (операторів) у будь-якій МП або опис алгоритму вирішення завдання для виконання на комп'ютері. Програма для використання на фабриці може бути:

— модульною (в деякій МП, як частина ОС або ПС);

— монолітною (кілька модулів або програм як одна програма);

— локальною (частина деякої програми);

— резидентною (перебуває в основній пам'яті ОС або ПС, готова до запуску чи постійно працює);

— розподіленою (розміщена і працює на кількох комп'ютерах або платформах із обміном даними через мережу);

— функціональною/прикладною (реалізує задачі предметної області — ПрО);

— сервісною (виконує послугу чи сервіс для користувача);

— інтероперабельною (взаємодіє з іншими програмами);

— реентерабельною (паралельне виконання з іншими програмами);

— заготовкою (шаблон, каркас, патерн, макрос, контейнер, контракт, assets, reuses, artifact тощо).

Наведені типи програм повинні мати паспортні дані з описом зовнішніх характеристик (у мовах IDL, API, SIDL тощо) чи специфікацією у стандартній мові OSWL для подальшого застосування у збірці різномовних програм і перетворення нерелевантних типів даних.

Цільові програми для виготовлення ПП на фабриці:

— програмна (прикладна) система (Application) — комплекс інтегрованих програм і засобів, що реалізують набір функцій деякої ПрО в заданому середовищі;

— програмне забезпечення (Software) — сукупність програмних засобів, які реалізують функції комп'ютерної чи технічної апаратно-програмної системи, включаючи

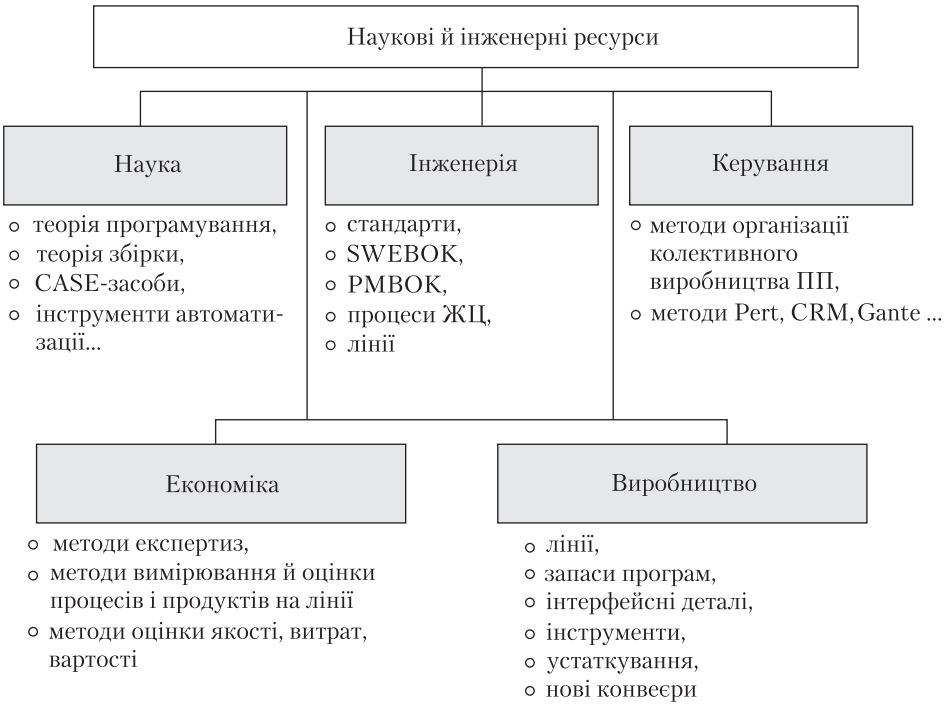


Рис. 3. Дисципліни виробництва ПП

загальносистемні засоби (ОС, СКБД, контроль технологічних процесів, обробку сигналів тощо) і прикладні програмні системи;

— сімейство систем (Systems family) — сукупність програмних систем із загальним і керованим (змінним) набором характеристик, що задовольняють потреби ПрО чи домену;

— програмний проект (Program Project) — унікальний інтегрований комплекс взаємозалежних заходів, орієнтованих на досягнення цілей і задач конкретного ПП за визначеними вимогами до термінів, бюджету, результатів його функціювання;

— складні програмні об'єкти — сукупність взаємопов'язаних цільових об'єктів різних типів (bissnes, web-services applications), які виконують необхідні функції, послуги, сервіси, виготовлені самостійно як прості цільові об'єкти з використанням інших готових компонентів з бібліотек або репозитаріїв. Великий клас становлять сервіси, Веб-сервіси, прикладні сервісні застосування. Для виготовлення з них застосувань мають специфічні особливості як деякі готові послуги, що обираються й виконуються за запитом різних користувачів. Цей клас засобів у майбутньому матиме також індустріальні риси виробництва з них ПП.

Наукові й інженерні ресурси фабрики

У межах досліджень розвитку програмної інженерії запропоновано класифікацію нових дисциплін, спрямованих на індустріальне виробництво ПП [28, 38–40] (рис. 3).

Сутність цих ресурсів як дисциплін збіркового виробництва ПП у наступному:

— **наука** базується на класичних дисциплінах (теорія алгоритмів, множин, доказу, математична логіка, теорія програмування) і відповідних загальних мовних засобах проектування абстрактних моделей і архітектур цільових програмних об'єктів, стандартизованих методів програмування

із процесами збірки ПП з готових програм та їхніх інтерфейсів;

— **інженерія** поєднує сукупність технологічних засобів і методів проектування ПП із фундаментальних і стандартних моделей ЖЦ, техніку аналізу ПрО, формулювання вимог, моделей системи, розроблення вихідного коду, його вимірювання, супровід і змінювання (реінженерія, реверсна інженерія, рефакторинг), адаптування ПП до інших комп'ютерних платформ і різних середовищ;

— **керування** ПП застосовує загальну теорію керування, містить базові методи керування програмним проектом за допомогою графіків робіт, спостережень за їхнім виконанням, а також ризиками, версіями (конфігураційний файл) ПП і супроводженням;

— **економіка** складається із сукупності методів експертного, якісного, кількісного оцінювання проміжних артефактів і кінцевого результату процесів ЖЦ, економічних методів розрахунку часу, обсягу, трудомісткості, вартості виготовлення ПП для постачання замовникові чи на ринок;

— **виробництво** базується на лініях виробництва комп'ютерних і прикладних систем, сімейств систем із застосуванням готових програм (КПВ, сервісів, аспектів, агентів і т.п.), що накопичені в інформаційних сховищах, бібліотеках і репозитаріях, а також одиночних готових програм, які перевіряють і сертифікують на якість і надійність.

Ці дисципліни призначені для систематизації процесів виготовлення ПП на деякій фабриці програм. У майбутньому вони стануть предметом підготовки студентів для участі в індустріальному виробництві ПП.

Технічні, технологічні й загальносистемні ресурси фабрики

Набір технічних, технологічних і загальних ресурсів організації-розробника чи фабрики ПП необхідний для виконання підпроцесів базового процесу програмної інженерії, спрямований на виконання договорів із замовником на ПП.

Технічні ресурси — платформи, процесори (Intel, IBM, Apple, MS тощо); комунікації (OSI, TCP/IP; комп'ютери; файли, сервери; локальні, глобальні мережі; електронна пошта; техніка налагодження тощо).

Технологічні ресурси — бібліотеки, репозитарії готових ПП (КПВ, reuses, assets, applications, domains, systems); методики програмування збіркового типу (модульного, компонентного, сервісного, UML); керівництва й методики з мов інтерфейсів (IDL, API, DII, SIDL, XML, RDF); стандартний опис (каркасів, шаблонів, контейнерів, процесів, проектів, систем, СПС).

Загальносистемні ресурси — ОС, клієнт-серверні технології, інструменти; офісні системи (рідери/райтери форматів pdf, ps, html); системи документообігу; утиліти (архіватори, записувачі інформації); засоби захисту (антивірусні, парольні); CASE-інструменти, транслятори; графічні інструменти; СКБД тощо.

Людські ресурси фабрики

Це групи розробників і служб керування/виконання проектних робіт за планами, контролю якості, ризиків, правильності реалізації проекту тощо.

В інфраструктурі людських ресурсів згідно зі стандартом ISO/IEC 12207 поєднано групи за таким призначенням (рис. 4):

— техніко-технологічної підтримки (вивчення ринку, придбання Case, ПП, консультації, навчання тощо);

— захисту інформації (паролі, ключі захисту, перевірки);

— технологічної служби (супроводу, підтримки ЖЦ, контролю дій/ удосконалення ТЛ тощо);

— якості (SQA-група) із функціями планування і виконання ЖЦ, перевірки робіт, контролю якості робочих продуктів і документів ПП;

— верифікації, валідації (V&V), тестування компонентів чи ПП на правильність

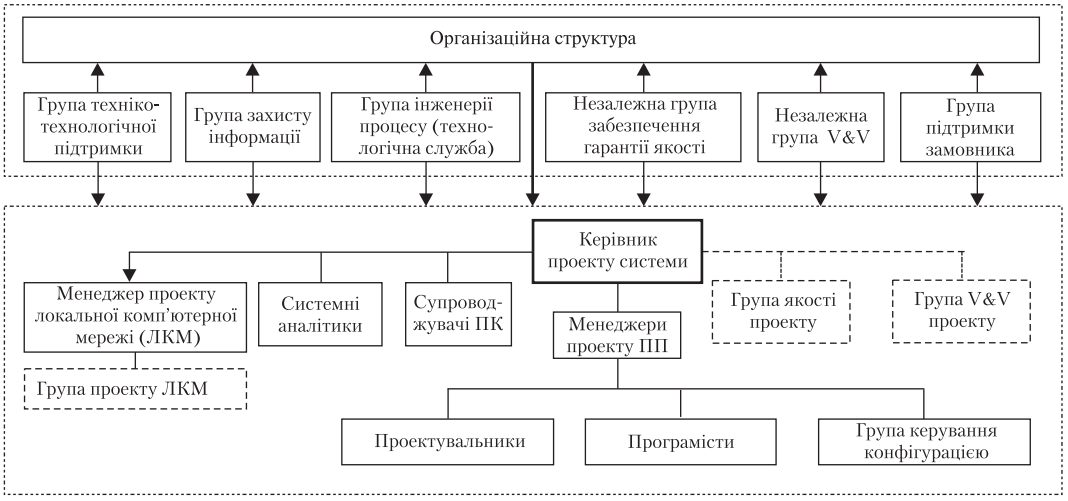


Рис. 4. Структура груп виконавців фабрики

виконання вимог, координування планів робіт із менеджером, перевірки правильності ПП у тестовому середовищі системи;

— керівників проекту, що відповідають за фінансові й технічні ресурси проекту, а також за виконання проектних угод замовника й керування розробленням ПП;

— менеджера проекту, відповідального за розроблення програмного проекту фабрики відповідно до вимог, проектних рішень і планів робіт;

— проектувальників і програмістів, що відповідають за розроблення проектних рішень, їхню реалізацію у вигляді програм, документів, інших вихідних результатів;

— керівника конфігурації, який реєструє версії ПП, зберігає тверді копії та конфігурацію з розмежуванням доступу до них.

Наведені ресурси необхідні для будь-якого індустріального колективу виробників ПП. Роль і призначення різних фахівців наведено в низці стандартів із завдань програмної інженерії.

Стандартні ресурси

Міжнародний комітет зі стандартизації розробив чимало стандартів програмної інженерії, що регламентують порядок розроблення ПП з керованими методами для деякої фабрики програм. Ці стандарти створюють важливі ресурси фабрики.

Базовий процес забезпечує «процесне продукування» ПП як вид інженерної діяльності з виготовлення ПП з операціями оцінки, вимірювання, керування змінами, вдосконаленням самого БП відповідно до стандарту ISO/IEC 15504-7 («Оцінювання процесів ЖЦ ПЗ. Настанови з удосконалення процесу»). Оцінку зрілості організації чи фабрики програм здійснюють за моделлю зрілості CMM (Capability Maturity Models) [16] інституту SEI США, а також моделями Bootstrap, Trillium тощо. Рівень зрілості визначає наявність фінансових ресурсів, стандартів, методик і здібностей (зрілості) членів колективу фабрики, здатних виготовляти ПП у визначені час і вартість.

Життєвий цикл у стандарті ISO/IEC 12207 «Процеси ЖЦ ПЗ» регламентовано різними напрямами діяльності щодо розроблення, проектування, керування ПП, організації процесів (планування, керування й супроводу), вимірювання, оцінювання продуктів і процесів. Найважливіші стандарти ДСТУ наведені в підрозділі «Методи оцінювання якості».

Ядро знань SWEBOK — стандарт SEI США містить опис 10 розділів (knowledge areas) програмної інженерії за двома категоріями. Перша — методи й засоби розроблення (формування вимог, проектування, конструювання, тестування, супровід), друга — методи керування проектом, конфігурацією, якістю, БП [43]. Методи ядра знань відповідають стандартним процесам ЖЦ з урахуванням потреб конкретної фабрики програм з регламентованою послідовністю розроблення і супроводу ПП, починаючи з вимог, вироблення проектних рішень, каркаса майбутнього продукту, вибору готових компонентів для «наповнення» цього каркаса [15, 39, 40, 55].

Ядро знань менеджменту проекту — стандарт керування проектом PMBOK, розроблений PMI США, що містить у собі опис лексики, структури процесів, галузі знань: *керування змістом проекту (*планування із розподілом робіт); *якістю* з контролем результатів на відповідність стандартам якості; *людськими ресурсами* відповідно до кваліфікації та професіоналізму. Нині — це стандарт IEEE Std.1490 «IEEE Guide Adoption of PMI Standard. A Guide to the Project Management Body of Knowledge». Крім цих стандартів, є багато інших, які потрібно використовувати, виготовляючи ПП.

3.2. Середовище фабрик програм

Розглянуті в розділі 1 різні види середовищ можуть бути базовими для певної фабрики. Рішення залежить від фінансів замовників і знань фахівців середовища для виготовлення відповідного ПП (наприклад, із засобами інтероперабельності компонентів, засобами замовника ПП тощо). У наступному часі експериментальним варіантом фабрики програм у ІПС є система Eclipse [45], яка безкоштовно подається через Інтернет і має новітні концепції виробництва ПП із готових КПВ, а саме:

— механізм задання плагінів у форматі XML засобами Plug Development Environment, що забезпечує автоматизоване їх підключення та нові інструменти (наприклад, Protégé, Java, RMI) репозитаріїв і готових програм;

— автоматизоване підключення нових меню до інтерфейсу користувача, іконок, сценаріїв тощо;

— використання мови Java. виклику RMI для опису, об'єднання різних програм у вихідному коді тощо.

Ці можливості моделюють на репозитаріях компонентів, КПВ і методі збірки в сімействі СПС. Згодом ця система генерації буде використана в середовищі Grid.

4. БАЗОВІ ОСНОВИ МАЙБУТНЬОЇ ФАБРИКИ ПРОГРАМ В ІПС

Серед розглянутих фабрик програм найрозвинутішою є інфраструктура обчислень глобального масштабу — Grid та загальна структура Eclipse.

Її поява зумовлена наявністю суперобчислювальних ресурсів (кластерів, supercomputers, frameworks), високопродуктивних мереж і низки глобальних задач, вирішення яких не під силу звичайним комп'ютерам. У цьому середовищі будуть використані світові ресурси: комп'ютери, системи програм, розподілені процесорні потужності, системи сховищ даних, схеми взаємодії гетерогенних платформ, розташованих у географічно віддалених адміністративних доменах світу. Тобто середовище Grid стане новітньою фабрикою збірки різнорідних і різноплатформених програм для обчислення за ними наукових задач глобального масштабу. Ця інфраструктура буде корисна багатьом інститутами НАН України. Її впроваджуватиме ІПС. Це насамперед наукові проекти в інститутах фізики, космічних досліджень і програмних систем. Для першого інституту проект спрямований на поліпшення середовища

GCube новими завданнями, які вдосконалюють наукові експерименти, пов'язані із фізичними явищами колайдера. Інститут космічних досліджень застосовує системи Grid для вирішення задач [34, 35, 42], орієнтованих на адаптацію, розвиток системних і програмних основ збірки та взаємодії різнопланових і різноплатформених комплексів програм і систем у межах інфраструктури Grid з подальшим застосуванням їх у системі НАН України і наукових центрах світового товариства.

Одним з важливих завдань збірки програм є форматизація і перебудова типів даних шляхом приведення до виду відповідної платформи (наприклад, 16-розрядної до 64-розрядної структури комп'ютерів) або до вихідного скомпільованого коду транслятора з МП. Такі проблеми вирішують по-різному в сучасних програмних середовищах. Коли готові програми обмінюються даними для інших або нових платформ у гетерогенному середовищі, виникають колізії з їхнім обчисленням із можливим отриманням неправильного кінцевого результату.

ІПС пропонує нове фундаментальне вирішення завдань передачі даних по мережі, суть його полягає у розробленні оригінальної системи генерації GDT загальних типів даних із FDT [22–25] сучасних МП, на яких описують програми для фізичних, біологічних та інших експериментів.

Основи системи генерації GDT<=>FDT викладено у стандарті ISO/IEC 11404–2007 (General Data Types) [26].

4.1. Загальна характеристика типів даних FDT і GDT

Фундаментальні типи даних — FDT визначають типи, операції над значеннями і формати їх подання на комп'ютері. Тип є математичним поняттям, що позначає множину значень елементів. Базовий тип — елементарний тип (ціле, дійсне та ін.), значення якого визначає апаратура, компілятори програм з МП. Тип присвоєно змінній у програмі з МП, що визначає клас значень, кожне з яких належить одному й тільки одному типу. Операції над значеннями типу — аксіоми, що перетворюють відображення значень одного типу в значення іншого типу. FDT включають прості, структурні, складні дані, множину операцій і значень типів даних, їх властивості та зв'язки з іншими типами даних. Прості типи — перераховані й числові; структурні — масиви, записи тощо; складні — множини, об'єднання, динамічні дані, списки, послідовності, стеки, дерева та ін.

Типи даних призначені для опису функцій програм у МП. Вони реалізовані системами програмування на різних платформах комп'ютерів у вихідний код, який є джерелом не лише для виконання програми на цій МП, але й для забезпечення взаємодії у різних сучасних середовищах, коли вони відрізняються між собою. Кожна реалізована програма відображає тип даних конкретної МП, значення якої передано іншій програмі шляхом виклику (звернення) або протоколу.

Типи даних загального призначення — GDT подані у стандарті ISO/IEC 11404–2007 за номенклатурою і семантикою наборів типів даних, що найчастіше використовують у МП та інтерфейсах ПП. LI-типи даних (*LI* — Language Independed) незалежні від МП, вони специфіковані як примітивні (базові, незалежні від інших) типи даних і непримітивні, що повністю/частково визначені через інші типи даних і становлять класи типів даних, реальні представники яких використані в МП, що ґрунтуються на концепції FDT. Механізм звернення — виклик процедур із завданням інтерфейсу до кожного стандартного сервісного засобу чи деякої системи.

Типи даних стандарту створюють простір значень — сукупність (колекцію) значень деякого типу, що визначають одним із та-

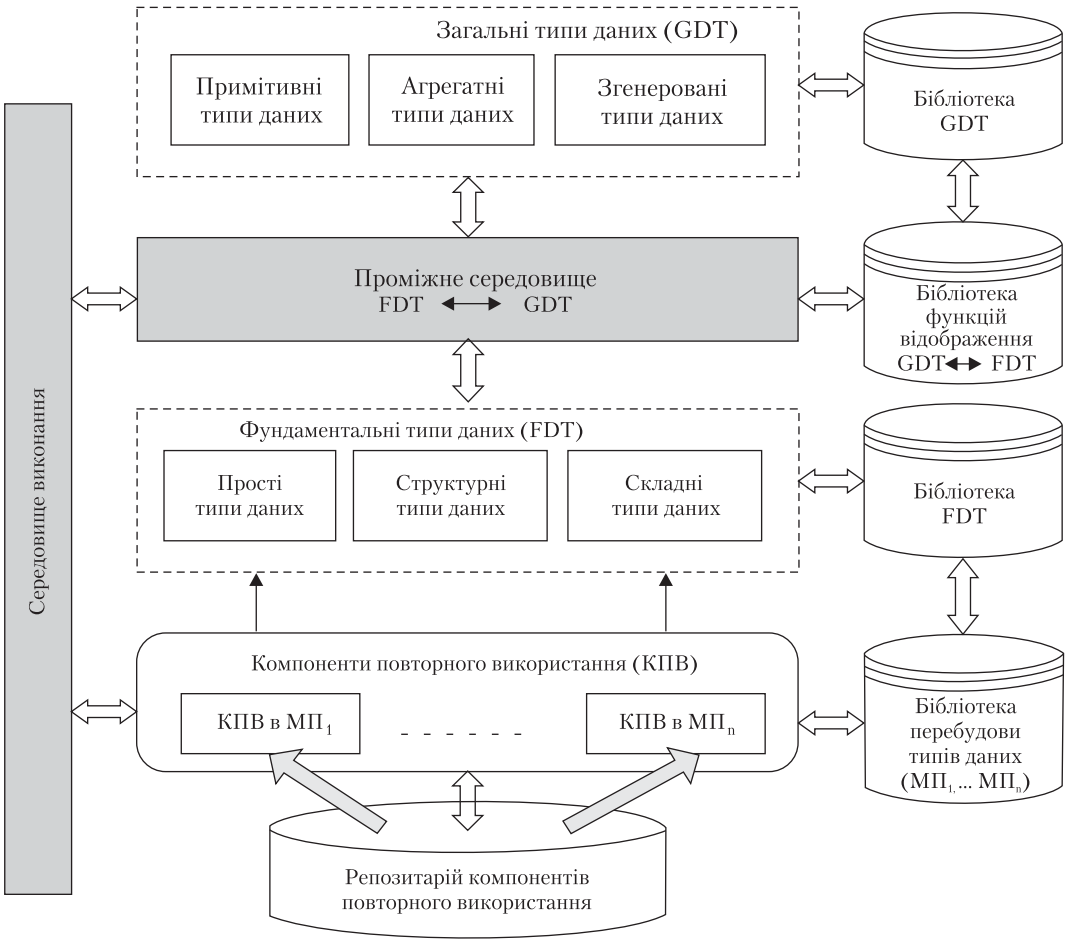


Рис. 5. Схема перебудови FDT<=>GDT

ких способів: перелічення; аксіоматичність згідно з основними положеннями; підмножина вже визначеного простору значень; комбінація будь-яких значень визначеного простору значень шляхом конструювання нових значень. Кожне окреме значення належить тільки одному типові даних або кільком підтипам.

Модель типів даних обчислювальна, абстрактна. Вона має справу із властивостями одиниць інформації для визначення характерних операцій щодо типів даних та їхніх генераторів, типи даних яких можуть бути подані в комп'ютерні системи. Типи даних можуть належати одному сімейству, якщо існує заміна, яка перетворює весь простір значень одного типу даних (domain) у підмножину (діапазон) простору значень іншого типу так, щоби зберігалися значення відношень і характеристичних операцій із домену й діапазону.

Генератор типів даних (datatype generator) — концептуальна операція над одним або кількома типами даних, яка створює новий тип даних. Він формує згенерований чи параметричний тип даних. Згенеровані агрегатні типи даних можуть отримувати значення параметричних типів даних. Він має назву компонента типу даних, значення яких відрізняються між собою властивостями і відношеннями між кожним компонентом і агрегатним значенням.

Згенеровані типи даних (generated datatypes) — типи даних, які отримано в результаті застосування генератора типів даних як операції для створення нового типу з одного чи кількох типів даних. Цей тип даних семантично залежить від різних параметричних типів і має власні характеристичні операції. Стандарт має такі генератори типів даних: вибір (choice), покажчик (pointer), процедура (procedure), запис (record), набір (set), портфель (bag), послідовність (sequence), масив (array), таблиця (table) тощо.

Із практичної точки зору типи даних генерують за допомогою спеціального набору процедур (функцій), які треба розробляти для використання в різних комп'ютерних системах. Для цього пропонується нова схема генерації типів даних GDT<=>FDT.

4.2. Основні функції схеми генерації типів даних

Відповідно до схеми генерації (рис. 5) слід розробити набір бібліотечних функцій (процедур) у загальноприйнятій мові XML для застосування їх при відображенні різних типів даних у програмах на сучасних або майбутніх МП за такими функціями:

— перебудови типів даних $МП_1$,..., $МП_n$;

— подання опису типів даних FDT;

— представлення GDT у формі для оброблення й апробування схеми FDT;

— відображення типів даних GDT<=> FDT.

Теорію подання FDT розроблено із прикладами опису для класу МП 4GL [2–25, 60]. Для реалізації вказаного набору функцій необхідно провести:

1) створення бібліотек функцій для перетворення типів даних GDT (примітивних, агрегатних і генерованих) до FDT (простих, структурних і складних) МП як необхідних елементів середовища взаємодії різномовних компонентів, підсистем і проектів системи Grid;

2) специфікацію зовнішніх типів даних компонентів, підсистем і систем у МП засобами мови GDT з накопиченням їх в одному з репозитаріїв середовища фабрики;

3) розроблення формату нових посередників, подібних до stub, з операціями звертання до відповідних функцій GDT<=>FDT з метою передачі взаємодіючому компоненту і зворотно нерелевантних і перебудованих типів даних.

Отже, проблему збирання різнорідних компонентів у нових МП з урахуванням архітектури платформ і гетерогенних середовищ у майбутньому, на наш погляд, буде розв'язано на інструментах і засобах перебудови типів даних GDT<=>FDT.

ВИСНОВКИ

Автори і співробітники ІПС НАН України стояли при витоках ідеї фабрик програм. У статті розглянуто основні позиції концепції академіка Глушкова, попередні шляхи її розвитку в Інституті кібернетики, основи чинних фабрик програм, базисом яких є інтегровані середовища збірки ПП, та сформульовано перспективні напрями. Наведено головні методи підтримки фабрики програм у ІПС, а також подано оргструктуру сучасної фабрики програм для збірки окремих і готових компонентів, визначено ресурси, необхідні для її функціонування.

Проект ІПС з інфраструктури системи Grid орієнтований на створення засобів перетворення загальних типів даних стандарту GDT<=>FDT відповідно до нової концепції, що відрізняється перебудовою типів даних сучасних і нових МП у майбутніх гетерогенних середовищах із використанням оригінальної бібліотеки функцій відображення GDT до FDT, а також функцій забезпечення невідповідностей і відмінностей різних платформ комп'ютерів. Буде побудовано і звичайні види інтерфейсів — конфігураційні файли, а також нові види

зв'язків і взаємодій компонентів, підсистем, СПС для виконання ПП у середовищі Європейського проекту Grid за участю різних інститутів НАН України.

1. *Капитонова Ю.В., Летичевский А.А.* Парадигмы и идеи академика В.М. Глушкова. — К.: Наук. думка, 2003. — 454 с.
2. *Глушков В.М., Лаврищева Е.М., Стогний А.А.* й інші. Система автоматизации производства программ (АПРОП). — К.: Ин-т кибернетики АН УССР, 1976. — 134 с.
3. *Сергиенко И.В., Парасюк И.П., Тукалевская Н.И.* Автоматизированные системы обработки данных. — К.: Наук. думка, 1976. — 256 с.
4. *Глушков В.М., Капитонова Ю.В., Летичевский А.А.* О применении метода формализованных технических заданий к проектированию программ обработки структур данных // Кибернетика. — 1978. — №6. — С. 31–43.
5. *Вельбицкий И.В., Ходаковский В.Н., Шолмов Л.И.* Технологический комплекс производства программ на машинах ЕС ЭВМ и БЭСМ-6. — М.: Статистика, 1980. — 264 с.
6. *Лаврищева Е.М., Грищенко В.Н.* Связь разноязыковых модулей в ОС ЕС. — М.: Финансы и статистика, 1982. — 127 с.
7. *Кахро М.И., Калья А.П., Тыугу Э.Х.* Инструментальная система программирования ЕС ЭВМ (ПРИЗ). — М.: Финансы и статистика, 1982. — 157 с.
8. *Волховер В.Г., Иванов Л.А.* Производственные методы разработки программ. — М.: Финансы и статистика, 1983. — 208 с.
9. *Лаврищева Е.М.* Основы технологической подготовки разработки прикладных программ СОД. — К., 1987. — 30 с.
10. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. — К.: Наук. думка, 1991. — 213 с.
11. *Лаврищева Е.М.* Проблематика программной инженерии. — К.: Знання, 1991. — 17 с.
12. *Липаев В.В., Позин Б.А., Штрик А.А.* Технология сборочного программирования. — М.: Радио и связь, 1992. — 272 с.
13. *Андон Ф.И., Лаврищева Е.М.* Методы инженерии распределенных компьютерных приложений. — К.: Наук. думка, 1997. — 229 с.
14. *Лаврищева Е.М.* Сборочное программирование. Некоторые итоги и перспективы. // Проблемы программирования. — 1999. — №2. — С. 20–31.
15. *Лаврищева Е.М.* Методы программирования. Теория, инженерия, практика. — К.: Наук. думка, 2006. — 451 с.
16. *Андон Ф.И., Коваль Г.И., Коротун Т.М., Лаврищева Е.М., Суслов В.Ю.* Основы инженерии качества программных систем: 2-е изд. — К.: Академпериодика, 2007. — 680 с.
17. *Коваль Г.И., Коротун Т.М., Лаврищева Е.М.* Об одном подходе к решению проблемы межмодульного и технологического интерфейсов // Диалоговые системы: Межотрасл. сб. АН СССР и Минвуза СССР. — 1988.
18. *Лаврищева Е.М.* Интерфейс в программировании // Проблеми програмування. — 2007. — №2. — С. 126–139.
19. *Лаврищева Е.М.* Становление и развитие модульно-компонентной инженерии программирования в Украине. — 33 с.
20. *Лаврищева Е.М.* Сборочное программирование. Теория и практика // Кибернетика и системный анализ. — 2009. — №6. — С. 1–12.
21. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии программных продуктов: 2-изд. — Доп. и перераб. — К.: Наук. думка, 2009. — 370 с.
22. *Гоар К.О.* Структурная организация данных // Структурное программирование. — М.: Мир, 1975. — С. 92–197.
23. *Турский В.* Методология программирования. — Пер. с анг. — М.: Мир, 1981. — 265 с.
24. *Агафонов В.Н.* Типы и абстракция данных в языках программирования // Данные в языках программирования. — М.: Мир, 1982. — С. 267–327.
25. *Замулин А.В.* Типы данных в языках программирования и базах данных. — М.: Наука, 1987. — 152 с.
26. Стандарт ISO/IEC 11404–2007. General Data Types.
27. *Лаврищева Е.М., Коваль Г.И., Коротун Т.М.* Подход к управлению качеством программных систем обработки данных // Кибернетика и системный анализ. — 2006. — №5. — С. 174–175.
28. *Лавріщева К.М.* Перспективні дисципліни програмної інженерії // Вісник НАН України. — 2008. — №9. — С. 12–17.
29. *Corbin J.* The Art of Distributed Applications. Programming Tech. for Remote Procedure Calls. — Berlin: Springer Verlag, 1992. — 305 p.
30. *Эммерих В.* Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft/COM и Java/RMI. — М.: Мир, 2002. — 510 с.
31. *Бей И.* Взаимодействие разноязыковых программ. — М., СПб., К.: Изд. дом «Вильямс», 2005. — 868 с.
32. *Гринфильд Дж.* Фабрики разработки программ. — М., СПб., К.: Изд. дом «Вильямс», 2007. — 591 с.
33. *Guckenheimer S., Perez J.I.* Software Engineering with Microsoft Studio Team System. — Crawfordsville: Adison–Wesley, 2006. — 304 p.

34. *Meglio A., Bégin M.E., Couvares P., Ronchieri E., Takacs E.* ETICS: the International Software Engineering Service for the Grid // Journal of Physics Conference. — 2008.
35. *Таковицкий О.* Технология Grid computing. — С. 1–9.
36. *Norhrop L.M.* Software SEI's Product Line Tenets // IEEE Software. — 2002. — V. 19. — №4. — P. 32–39.
37. *Чернецки К., Айзенекер У.* Порождающее программирование. Методы, инструменты, применение. — М., СПб., Харьков, Минск: Издательский дом «Питер», 2005. — 730 с.
38. *Лаврищева Е.М.* Классификация дисциплин программной инженерии // Кибернетика и системный анализ. — 2008. — №6. — С. 3–9.
39. *Лавріщева К.М.* Визначення предмету — програмна інженерія // Проблеми програмування. — 2008. — №2–3. — С. 191–204.
40. *Лавріщева К.М.* Програмна інженерія — напрями розвитку // Праці міжнародної конференції «50 років Інституту кібернетики імені В.М. Глушкова НАН України». — К., 2008. — С. 336–345.
41. *Лаврищева Е.М.* Проблема интероперабельности разнородных объектов, компонентов и систем. Подходы к ее решению // Матер. 7 міжнародної конференції з програмування. — С. 28–41.
42. *Castelli D., Candela L., Pagano P., Simi M.* 2005 2nd IEEE — CS International Symposium Global Data Interoperability (IEEE Computer Society). — P. 56–99.
43. http.//swebok.com
44. *Лавріщева К.М.* Генерувальне програмування програмних систем і сімейств // Проблеми програмування. — 2009. — №1. — С. 3–16.
45. *Карлсон Д.* Eclipse. — Изд. «Лори», 2004. — 335 с.
46. http.//www.oberon_ethz.ch
47. http.//en.wikipedia.org/babel-middleware
48. *Грищенко В.Н., Лаврищева Е.М.* Методы и средства компонентного программирования // Кибернетика и системный анализ. — 2003. — №1. — С. 39–55.
49. *Грищенко В.М.* Метод об'єктно-компонентного проектування програмних систем // Проблеми програмування. — 2007. — №2. — С. 113–125.
50. *Коротун Т.М.* Моделі й методи інженерії тестування програмних систем в умовах обмежених ресурсів. Автореф. дисер. — К.: Інст. кібернетики ім. академіка Глушкова, 2005. — 21 с.
51. *Коваль Г.І.* Моделі й методи інженерії якості програмних систем на ранніх стадіях життєвого циклу. Автореф. канд. дисер. — К.: Інст. кібернетики ім. академіка Глушкова, 2005. — 19 с.
52. *Слабоспицька О.О.* Моделі та методи експертного оцінювання у життєвому циклі програмних систем. Автореф. канд. дисер. — К.: Інст. кібернетики ім. академіка Глушкова, 2008. — 21 с.
53. *Задорожна Н.Т.* Кероване проектування документообігу в управляючих інформаційних системах. Автореф. канд. дисер. — К.: Інст. кібернетики ім. академіка Глушкова, 2004. — 21 с.
54. *Задорожна Н.Т., Лавріщева К.М.* Менеджмент документообігу в інформаційних системах освіти. — К.: Педагогічна думка, 2007. — 220 с.
55. *Бабенко Л.П., Лавріщева К.М.* Основи програмної інженерії. Посібник. — К.: Знання, 2001. — 269 с.
56. *Бабенко Л.П.* Проблемы повторного использования в программной инженерии // Кибернетика и системный анализ. — 1999. — №2. — С. 155–166.
57. Protégé-OWL API Programmer's Guide <URL: http://protege.stanford.edu/doc/owl/guide.html
58. *Коваль Г.І., Колесник А.Л., Лавріщева К.М., Слабоспицька О.О.* Удосконалення процесу розроблення сімейств програмних систем елементами гнучких методологій // Проблеми програмування. — 2010. — №2–3. — С. 261–270.
59. *Лаврищева Е.М., Слабоспицкая О.А.* Подход к экспертизе оценивания в программной инженерии // Кибернетика и системный анализ. — 2009. — №4. — С. 151–168.
60. *Лавріщева К.М.* Програмна інженерія. Підручник. — К.: Академперіодика, 2008. — 450 с.

П. Андон, К. Лавріщева

РОЗВИТОК ФАБРИК ПРОГРАМ
В ІНФОРМАЦІЙНОМУ СВІТІ

Резюме

У 1975 р. академік Глушков запропонував концепцію конвеєрного способу виробництва ПП з готових програм. У статті проаналізовано розвиток цієї концепції на прикладі попередніх і сучасних фабрик програм; засвідчено появу двох основних понять виробництва: інтерфейс як стиковочний елемент із передачі і трансформації програм, що збираються, та інтегроване середовище збірки готових різнорідних програм із деяких МП. Упродовж останніх 35 років вони постійно вдосконалювались і стали базисом сучасної фабрики програм, наприклад, в інфраструктурі Європейського проекту Grid, призначеного для обчислювання наукових завдань. Автори оприлюднюють результати наукових досліджень Інституту програмних систем НАН України, дають визначення фабрики програм за збіркою ПП із різнорідних і різноплатформених програм з використанням людських, технологічних та інструментальних ресурсів. Інститут планує упроваджувати ці результати в систему Grid і в інститутах НАН України. В межах системи будуть розроблені нові засоби інтерфейсу різнорідних програм із перетворення стандартних ISO/IEC 11404–2007 типів даних до тих, що є в багатьох мовах програмування, процедури генерації яких зба-

гатять майбутні гетерогенні середовища сучасними засобами збірки програм.

Ключові слова: системи автоматизації, індустрія, виробництво програмних продуктів, інтерфейс, передача типів даних, трансформація програм..

Andon P., Lavrischeva K.

PROGRAM FACTORIES DEVELOPMENT IN THE INFORMATION WORLD

Abstract

In 1975, academician Hluskov proposed assembly line way in program products (PP) from ready programs manufacturing concept. The article analyzes that concept development on previous and modern program factories example; demonstrates the appearance of two main notions in production — 1) interface as butt-joint element in assembling programs assignation and transformation; 2) integrated medium for ready various programs of certain program languages assembling. During last 35 years they are being constantly improved and become a modern program factory basis, exempli gratia in the intended for scientific tasks calculation European Project Grid infrastructure. Authors expose the results of Program Systems Institute of Ukrainian NAS scientific explorings; give definition for the program factory assembling PP of various programs having different platforms with the usage of human, technology and instrument resources. Institute plans to instill those results into Grid system and Ukrainian NAS institutes. Within system creases, new interface means will be treated. They belong to various programs transforming data types of ISO/IEC 11404–2007 standard to ones existing in the large number of program languages whose generation procedures enrich future heterogenous environments with modern program assembling means.

Keywords: automatization systems, industry, program products production, interface, data type assignation, program transformation.