ПРОГРАММНО-
ТЕХНИЧЕСКИЕ
КОМПЛЕКСЫ

Е.М. ЛАВРИЩЕВА

УДК 681.3.06

ТЕОРИЯ И ПРАКТИКА ФАБРИК ПРОГРАММНЫХ ПРОДУКТОВ

Ключевые слова: *производство программных продуктов, языки программирования, интерфейс, платформа, среда, взаимодействие, типы данных, трансформация программ и данных.*

ВВЕДЕНИЕ

Концепцию фабрики программ с конвейерным способом производства академик В.М. Глушков сформулировал на научном семинаре Института кибернетики (ИК) Академии наук Украины в 1975 г. [1, 2]. Содержание этой концепции формировалось постепенно (теоретически и практически) последние десятилетия. Были созданы разные средства поддержки многих аспектов сборочного производства программных продуктов (ПП). Объекты сборки — модули повторного использования в языках программирования (ЯП) 4GPL (Алгол, ПЛ-1, Кобол, Фортран, Модула-2 и др.) и их интерфейсы в языках (MIL, MESA, IDL, API и др.), которые служат источником генерации модулей посредников (Р-код, stub, skeleton и др.) для каждой пары модулей, размещаемых в библиотеках ЭВМ или в республиканских фондах алгоритмов и программ. Назовем системы автоматизации метода сборки модулей: в ИК — Дельтастат, Проект, Маяк, Мультипроцесист, РТК, АПРОП [1–4], в других институтах бывшего СССР — СИМПР (МГУ), ПРИЗ (АН Эстонии), Альфа (Новосибирск) и в зарубежных — SUN ONC IBM, Corba, Oberon, MS.net и др. [3–8]. Метод конвейерной сборки программ определен на основе анализа конвейера в автомобильной промышленности и включал технологические линии (ТЛ) процессов производства ПП на основе названных систем автоматизации [9]. Сформулировано понятие интерфейса ЯП и разноязычных модулей как необходимых элементов сборочного производства ПП [10–18].

Таким образом, к 1990 г. с участием специалистов ИК были определены базовые понятия и атрибуты первых фабрик программ: линии производства, системы сборки ПП, язык спецификации модулей, методы интеграции (в частности, сборки), интерфейсы для передачи данных и преобразования нерелевантных типов данных ЯП, библиотеки модулей и операционные среды.

Концепция Глушкова с конвейерным производством частично реализована в рамках автоматизированной информационной системы (АИС) «Юпитер» (1980–1991). Впервые было создано несколько видов ТЛ для изготовления прикладных программ (ввода и вывода данных, пакетов прикладных программ, задач статистики, численных методов и т.п.) [2, 11] для четырех технических объектов АИС. В связи с развалом СССР работы по индустрии ПП выполнялись теоретически специалистами отдела «Программная инженерия» Института программных систем (ИПС) НАН Украины.

Процесс разработки основ фабрик программ активно развивается за рубежом по многим направлениям: стандартизация качества (ISO/IEC 9000-1, 2, 3, 4, ISO/IEC 12598-1, 2, 3, 4, ГСТУ 9126, 9150 и др.), жизненного цикла (ЖЦ) ISO/IEC 12207 и типов данных общего назначения ISO/IEC 11404; систематизация разных видов деятельности (экономика, управление и др.) [19, 20]; индустриализация ПП (Product Lines, Fabric program); определение языков описания интерфейсов (IDL, API, XML, RDF, SIDL) и совершенствование сред — MS.VSTS, Corba, Java, Grid [12, 14] и т.п.

Описание основных положений фабрик производства программ начинается с характеристики теоретических работ, описания содержания и структуры фабрики, принципов ее организации, управления и функционирования, а также новых направлений индустриального производства программ.

1. РАЗВИТИЕ ТЕОРЕТИЧЕСКИХ ОСНОВ ПРОИЗВОДСТВА КАЧЕСТВЕННЫХ ПП

Концепция Глушкова последние десятилетия развивалась теоретически в фундаментальных и диссертационных исследованиях в рамках научных проектов ГКНТ (1992–1996 гг.) и НАН Украины (1997–2010 гг.). Сформулировано сборочное программирование, основанное на идее ТЛ с процессами проектирования, сборки разноязыковых программ со средствами преобразования несовместимых FDT типов данных. Разработана теория экспертиз ПП, тестирования и оценивания качества ПП. Специалисты института получили оригинальные научные результаты по многим аспектам производства ПП, которые опубликованы в статьях, монографиях и научных отчетах. Некоторые из них приводятся в коротком изложении [9–29].

Подход к построению ТЛ. Процессы и операции, технологический маршрут ТЛ создаются на этапе технологической подготовки разработки (ТПР) после анализа предметной области и выявления ее основных задач и функций [9, 14]. Для них выбираются готовые модули, методы, средства и инструменты, определяются процессы планирования, контроля и управления процессами ТЛ, формируются технологические объекты для фиксации проектных решений, изменения состояний объектов и процессов производства ПП, а также управления качеством ПП на ТЛ [9]. Процессы ТЛ соответствуют международному стандарту ISO/IEC 12207-96, 2007 и ДСТУ 3918–99, подкрепляются методами, средствами и инструментами программирования для осуществления изменений состояний объектов. Они описываются в специальном языке спецификации. Подход к построению ТЛ апробирован в АИС «Юпитер». Идеи ТЛ возникли вновь в *линии продуктов* (Product Line Practice) [21], принципы построения и выполнения аналогичны ТЛ и отображают индустрию производства коммерческих ПП.

Теория объектно-компонентного программирования. Компонентный подход исторически пришел на смену модульному — основе сборочного программирования [15–18, 22, 23]. Он является естественным продолжением определения программ с точки зрения сборочной концепции. Из-за отсутствия в этом подходе теоретических основ была поставлена задача — создать теорию компонентов, используя теорию объектного подхода Буча и сборочного программирования, реализующую теорию преобразования типов данных ЯП. В результате был разработан объектно-компонентный метод (ОКМ) [16, 22, 23], основанный на теории Фреге [24] для уточнения и обобщения свойств и характеристик объектов математическими формализмами.

Определение 1. Программный компонент — это независимый от ЯП программный объект, который обеспечивает выполнение некоторой совокупности прикладных функций (сервисов), доступ к которым возможен с помощью интерфейсов, определяющих порядок их вызова.

Сформулирован формальный механизм перехода от объектов к компонентам и их интерфейсам, а также определен компонент повторного использования (КПИ) в компонентном программировании:

$$\text{КПИ} = (T, I, F, R, S),$$

где T — тип компонента, I — интерфейс компонента; F — функциональность, R — реализация, S — сервис взаимосвязи с другими компонентами и средой [23].

Создана алгебра анализа $\Sigma = (E', \Psi, P)$, $E' = (E_1, E_2, \ldots, E_n)$ в ОКМ, которая развивает объектную теорию Буча формальными механизмами представления множества объектов, операциями $\Psi = \{decds, decdn, comds, comdn, conexp, connar\}$ над элементами E' и множеством предикатов $P = (P_1, P_2, \ldots, P_r)$. Система операций алгебры и функций детализации, экземпяризации и агрегации объектов является полной.

В компонентном программировании определен набор формальных моделей (*компонента, интерфейса, компонентной среды*), операций компонентной алгебры (внешней и внутренней) и усовершенствована теория преобразования несовместимых типов данных компонентов FDT в ЯП.

Внешняя алгебра $\Psi = \{CSet, CESet, \Omega 1\}$ определена на множестве компонентов $Cset = \{comp\}$, среде CESet из множества компонентов и интерфейсов из репозитория или библиотек и операциях $\Omega 1 = \{CE_1, CE_2, CE_3, CE_4\}$ объединения компонентов и сред $CE_3 = CE_1 \cup CE_2$, удаления компонента из среды $CE_2 = CE_1 \setminus Comp$, инсталляции и замещения компонента и его интерфейса $Comp_1$ на $Comp_2$ — $CE_2 = Comp_2 \oplus (CE_1 \setminus Comp_1)$. Доказана непротиворечивость операций и функциональная целостность компонентов и компонентной среды.

Внутренняя алгебра $\varphi = \{CSet, CESet, \Omega 2\}$ определена на операциях эволюции (замены) $\Omega 2 = \{O_{refac}, O_{Reing}, O_{Rever}\}$, где:

$O_{refac} = \{AddOImp, AddNImp, ReplImp, AddInt\}$ — операции рефакторинга,

$O_{reing} = \{rewrite, restruc, adop, supp, conver\}$ — операции реинженерии,

$O_{Rever} = \{visual, metric, restruc, design, rewrite\}$ — операции реверсной инженерии.

Эти операции предназначены для изменения имен, интерфейсов, реализаций компонентов, а также объединения интерфейсов и реализаций в одно целое (например, в контейнер).

Компонентная алгебра состоит из этих операций: $\Xi = \{\Psi \cap \varphi\} = \{CSet, CESet, \Omega 1\} \cap \{CSet, CESet, \Omega 2\} = \{CSet, CESet, \Omega 1, \Omega 2\}$, и обеспечивает механизмы взаимодействия и эволюции разнородных компонентов в современных средах, а также преобразование несовместимых в ЯП типов данных аналогично системе АПРОП [2]. Она регламентирует операции в индустриальном производстве ПП [9, 18, 22, 23].

Решение проблемы несовместимости типов данных. Отличие в описании типов данных в ЯП для переменных разноязычных программ в операторах типа CALL, неадекватность в трансформации данных системами программирования и архитектурные различия платформ реализации компонентов осуществляется изоморфным отображением. Задача сборки пары разноязычных модулей — определение взаимно однозначного соответствия между множествами фактических параметров $V = \{v_1, v_2, \ldots, v_n\}$ и формальных параметров $F = \{f_1, f_2, \ldots, f_k\}$ модулей.

Фундаментальные типы данных ЯП и способы их представления определены в теории структурной организации данных [15–18, 25–29] и включают два класса:

- простые типы данных $t = b(bool), c(char), u(int), r(real))$;
- сложные типы данных $t = a(array), z(record), u(union), e(enum)$ и др.

Каждый тип данных в классе предложено задавать алгебраической системой вида $Gt_\alpha = \{X^t_\alpha, \Omega^t_\alpha\}$ [22], в которой t — тип данных языков L, X^t_α — множество значений типа данных, Ω^t_α — множество операций над этим типом данных.

Простым и сложным типам данных поставлены в соответствие следующие два класса алгебраических систем:

$$\Sigma_1 = \{G^b_\alpha, G^c_\alpha, G^u_\alpha, G^r_\alpha\}, \quad \Sigma_2 = \{G^a_\alpha, G^z_\alpha, G^u_\alpha, G^e_\alpha\}.$$

В каждом классе этих систем преобразование типов данных $t \to q$ для пары языков l_t и l_q основано на таких свойствах отображений:

1) системы G_{α}^{t} и G_{β}^{q} изоморфны, если их типы данных q, t определены на одном и том же множестве типов данных;

2) между значениями X_{α}^{t} и X_{β}^{q} типов данных t, q существует изоморфизм, если множества операций Ω_{α}^{t} и Ω_{β}^{q} разные. Если множество $\Omega = \Omega_{\alpha}^{t} \cap \Omega_{\beta}^{q}$ непустое, то имеет место изоморфизм двух систем: $G_{\alpha}^{t'} = \langle X_{\alpha}^{t}, \Omega \rangle$ и $G_{\beta}^{q'} = \langle X_{\beta}^{q}, \Omega \rangle$. Если тип данных отличается, например, t — строка, а тип q — вещественное, то между множествами X_{α}^{t} и X_{β}^{q} не существует изоморфного соответствия.

Отображения сохраняют линейный порядок элементов в силу линейной упорядоченности элементов алгебраических систем из этих классов.

Методы оценивания качества ПП. Впервые идея обеспечения качества прозвучала в 1968 г. на конференции НАТО по программной инженерии. С того времени многие специалисты стали уделять внимание решению этой проблемы за рубежом и в ИПС в рамках научно-исследовательских проектов ГКНТ, Министерства науки и НАН Украины. Комитет ISO создал ряд стандартов качества ISO 9000-1, 2, 3, 4., ГОСТ 2844-94, 9126, 2850-94), в которых определены модели качества, терминология и подходы к обеспечению качества компонентов и ПП Основным направлением работы отдела продолжительное время была проблематика качества ПП, включающая аспекты проверки правильности компонентов и программ (тестирование, верификация, экспертизы и др.), и по результатам этой проверки разработаны методы оценивания качества компонентов и ПП. В результате были созданы оригинальные формальные методики проверки правильности, измерения и оценивания показателей качества компонентов и ПП в классе задач СОД, которые не имеют прототипа:

— модель качества с ориентацией на оценку надежности ПП;

— модель распределения количественных требований по отдельным компонентам и с учетом приоритета показателя, заданного пользователем;

— модель распределения надежности ПП компонентов с функцией полезности $Q_{nc} = \sum_{j=1}^{l} v_j^* \cdot q_j = \sum_{s=1}^{m} w_s^* \cdot r_s$ в зависимости от весовых коэффициентов w_s и надежности $q_j = \prod_{n \in E_j} r_n$ отдельного компонента;

— концептуальная модель принятия решений по управлению качеством, включая методы систематического контроля надежности, начиная из ранних стадий ЖЦ, измерения требований к надежности и прогнозированию возможных дефектов.

Эти результаты изложены в ряде статей и в монографии [30–39], которые пользуются спросом в странах СНГ. Они развиваются в новом проекте по генерирующему программированию применительно к семействам систем [32].

Новые средства проверки правильности компонентов. Тестирование компонентов и ПП стали следующим главным направлением исследований, нацеленных на выявление ошибок, поиск дефектов, отказов и сбоев программ, которые вызываются разными нерегулярными ситуациями в ПП или аварийным прекращением функционирования некоторого компонента или всей компонентной системы. В работах [30, 33, 34] предложены новые результаты в области тестирования:

— модель процесса тестирования с учетом рисков отказов и их оценки во время эксплуатации ПП;

— математическая модель определения оптимального времени тестирования компонентов t_e^* с максимальной прибылью $K(t_0 | t_e) = \Delta R(t_0 | t_e) - C(t_e) = C_{\text{м}}(\mu(t_0) - \mu(t_0 + t_e) + \mu(t_e)) - c_1 t_e - c_2 \mu(t_e)$, с производной $K(t_0 | t_e) = C_{\text{м}}(\lambda(t_e) - -\lambda(t_0 + t_e)) - c_1 - c_2 \lambda(t_e)$ в зависимости от функции возрастания надежности, интенсивности $\lambda(t)$ и риска $C_{\text{м}}$;

— технология тестирования ПП и сбора информации о всех видах ошибок и использование их при расчете надежности программ СОД.

Результаты в области тестирования апробированы в прикладных проектах Министерства обороны Украины, они совершенствуются применительно к ПП, изготовляемых в среде проекта по генерирующему программированию (2007–2011) [33].

Подход к экспертному оцениванию процессов обеспечивает решение формальных задач оценивания объектов производства и эксплуатации ПП с помощью экспертиз и новой модели процесса управления рисками ПП на основе дерева ценности. Разработанный метод группового экспертного оценивания компонентов и ПП базируется на соответствующей экспертной теории применительно к оценке процессов, адаптированных из стандарта ЖЦ ПП:

$$T = \langle G, et, ch(et) \rangle;\ \varnothing \neq G \in GT;\ et \in ET;\ ch(et) \in CH^{et};\ CH = \bigcup\nolimits_{et \in ET} CH^{et};$$

$$ES(T) = \langle A, d(ca_1, cn_1, \ldots, ca_n, cn_n), mg, vf \rangle,\ d \in \Delta_{et,ch},$$

где G — цель разработки ПС; et — тип оцениваемых объектов ЖЦ; $ch(et)$ — целевая характеристика; $A = \{ a = (ud, t) \}$, $|A| > 1$ — альтернатива; d — аналитическая зависимость $ch(et)$ от ее оцениваемых подхарактеристик, $ca_l \in CH^{et}$, $l = 1, \ldots, n$, из множества $\Delta_{et,ch}$; $\varnothing \subseteq cn_l$ — источник контекста оценивания ca_l; mg — модель экспертной группы; $\varnothing \subseteq vf$ — основа верификации оценок $\{ch(a), a \in A\}$.

Метод оценивания встроен в новый процесс ЖЦ с общей информационной средой, адекватной потребностям и специфике производственной деятельности, связанной с изготовлением ПП. Математический аппарат включает методический каркас (целевые функции и механизмы реализации); модель и методы оценивания процессов и ПП с помощью формализмов, повышающих качество экспертизы результатов, и их дальнейшее использование при сборке на основе процессов управления изготовлением ПП [34].

Методика управления программным проектом. Менеджмент программного проекта стал одной из проблем индустрии ПП в ряде зарубежных работ, в том числе в стандарте PMBOK (Project Mamagerment Body of Knowladge, 2005). Стандарт регламентирует руководство работами команды исполнителей проекта с использованием общих методов управления, планирования и контроля работ (стартовые операции, планирование итераций, мониторинг и отчетность), управление рисками и конфигурацией ПП. При исследовании менеджмента появились новые теоретические и прикладные результаты [35, 36], а именно:

— формальная модель управления проектированием информационных систем с учетом материальных, финансовых и трудовых ресурсов, необходимых для производства ПП;

— метод формирования варианта плана X работ по сетевому графику включает последовательность работ ($l_i \in L$) с объемом q_i и типом W_i ресурсов $R = \{R_L, R_S\}$, норму их использования ($NR_i \in NR$) в соответствии с законом распределения случайных величин $F = \{F_1, \ldots, F_r\}$, времени t в плановом периоде $[t_0, T]$ и с вероятностью окончания работ, исходя из критерия оптимального плана $K(X^*) = \min K(X)$.

Данный метод прошел апробацию в системах автоматизации образовательных процессов («Документооборот в образовании Украины», порталы «Дети Украины», «Учитель новатор», 2004–2010) в Академии педагогических наук Украины.

Классификация дисциплин для производства ПП. Структуризация программной инженерии (ПИ) из 10 knowledge areas of SWEBOK (Software Egineering of Body Knowledge, www.swebok.com) соответствует процессам стандарта ISO/IEC 12207, программе обучения Computing Curricula (2001, 2004) и направлена на получение знаний в этой области.

Производство разных видов ПП и семейств систем требует новой структуризации ПИ, ориентированной не только на процессы ЖЦ и общие методы их поддержки, но и на теоретические и прикладные методы решения важных технологических задач производства ПП (инженерия, управление и т.п.). Предложена новая классификация дисциплин, которая обеспечивает регламентацию разных видов работ в производстве ПП на фабриках программ: научная дисциплина, инженерная дисциплина, дисциплина управления, дисциплина экономики и производство ПП. Сущность и назначение каждой из дисциплин представлены в [18, 39–50]. Здесь дается их краткая характеристика.

Научная дисциплина ПИ — это совокупность таких теоретических дисциплин, как теория алгоритмов, множеств, доказательства, математическая логика и т.п., теория программирования, ЯП, абстрактные модели и архитектуры программных объектов, методы программирования, основанные на процессах сборки ПП из готовых программ и их интерфейсов.

Инженерная дисциплина ПИ — это совокупность технологических средств и методов проектирования ПП из фундаментальных и стандартных моделей ЖЦ, техника анализа предметной области, формулирования требований, моделей системы, разработка исходного кода, сопровождение, внесение изменений (реинженерия, реверсная инженерия, рефакторинг), перенос ПП на другие компьютерные платформы и среды.

Дисциплина управления в ПИ основана на общей теории управления, PMBOK и содержит базовые методы управления программным проектом с помощью графиков работ, наблюдений за их выполнением, а также управление рисками, версиями (конфигурационный файл) ПП и сопровождением.

Экономическая дисциплина в ПИ — это совокупность методов экспертного, качественного и количественного оценивания промежуточных объектов, артефактов и конечного результата процессов ЖЦ, а также методов расчета времени, объема, трудоемкости и стоимости изготовления ПП для передачи его на рынок.

Дисциплина индустрии ПП — это набор ТЛ для производства прикладных систем, семейств систем с применением готовых ПП, накопленных в современных информационных хранилищах, библиотеках и репозиториях, одиночные готовые программы (сервисы, аспекты, агенты и т.п.). Готовые ПП проверяются методами верификации, тестирования и оцениваются на показатели качества и надежности.

Таким образом, специалисты ИПС своими новыми научными достижениями сделали важный вклад в теорию производства ПП по основным вопросам разработки, тестирования и управления изготовлением ПП. Они положены в основу создания экспериментальной фабрики программ в новой среде Eclipse.

2. ОБЩАЯ ХАРАКТЕРИСТИКА КОНЦЕПЦИЙ ПРОИЗВОДСТВА ПП НА ИЗВЕСТНЫХ ФАБРИКАХ ПРОГРАММ В ИНФОРМАЦИОННОМ ПРОСТАНСТВЕ

Анализ фабрик программ начнем с характеристики первых фабрик программ, концепцию которых сформулировал академик В.М. Глушков (1975), а именно:

- система АПРОП (ИК) [2];
- система Sun Microsystems (IBM) [41];
- OMA-архитектура, или система CORBA (OMG) [42];
- фабрика «ручной» сборки разноязычных программ Инга Бейя [43];
- фабрики программ по методу UML Дж. Гринфильда [44];
- среда для групповой разработки ПП — MS.VSTS [45];
- инфраструктура вычисления Grid [46, 47].

Анализ сред систем — АПРОП, IBM Sun Microsystems, CORBA. Это основные системы на начальном этапе конвейерного производства ПП, проложивших путь к созданию современных фабрик программ.

АПРОП — это фабрика, которая работала в среде ОС ЕС и обеспечивала сборку разноязычных модулей в монолитную структуру на ЕС ЭВМ через ин-

терфейсных посредников, сгенерированных с помощью специальной библиотеки функций преобразования нерелевантных FTD типов данных (например, символьного к целому и т.п.), которые передаются операторами CALL в ЯП модулях и реализуют методы численного анализа и статистики, которые расположены в специальном Банке модулей.

IBM — среда со сборкой разноязычных программ в 4GPL (1980-е годы) с помощью внешних интерфейсных данных, которые трансформируются функциями XDR-библиотеки, Sun Workshop, Toolbox и т.п. к соответствующей платформе. Дальнейшим развитием новых направлений производства ПП является модель архитектуры SOA, веб-сервисы, языки C, C++, Java, Ruby, Script, которые обеспечивают связь программ в ЯП и передачу данных через порт системы AIX. Интеграция разнородных программ выполняется на общей платформе IBM в средах ONC (Sun Microsystems), MVS, VM, OS/2, AIX, Open source, а также на сервере (WebSphere Application Server Compunity Edition).

CORBA, или OMA-архитектура (Apple, IBM, Win-NT, x-Open, Dec), ее можно считать фабрикой программ с обеспечением связи разноязычных объектов в ЯП (C, C++, Smalltalk, Java, Cobol, Visual, Ada-96) через интерфейсных посредников (stub, skeleton, dill), которые описываются в языке IDL для клиент-серверной архитектуры (Client–interface, Server–Interface) с использованием сред (COSS, DCE/RPC, PCTE, ToolTalk, Java2SDK, NetPilot CCS и т.п.). Данные между клиентом и сервером передаются протоколами TCP/IP, IIOP через брокер ORB, который обеспечивает их разным сервисом, в том числе по преобразованию несовместимых типов данных разноязычных объектов, устранению неоднородности платформенных данных взаимодействующих объектов клиента и сервера. Эта среда поддерживает связи с другими средами: CORBA, OLE/DCOM, SOM/DSOM, IBM OS и т.п.

Фабрика «ручной» сборки по И. Бею. Базисом этой фабрики производства разноязычных программ есть интерфейсный посредник, командные строки, конфигурационные файлы, проверенные в операционных средах (VC++, VBasic, Matlab, Java, Visual Works Smalltalk и т.п.) для разных платформ (Microsoft.Net, HP, Apple, IBM и т.п.). Автор разработал разные варианты связей пар разноязычных программ в названных ЯП. Они передавали друг другу данные, для некоторых из них проводилось преобразование данных, типы которых нерелевантные или их форматы зависят от особенностей платформы, механизмов передачи данных (через протоколы, вызовы, интерфейсные карты MIO-16E-2 и др.) и средств RMI, Java Native Interface и его реализации в виде exe-файла. И. Бей апробировал Domain and Application Model, Model Interconnection, Microsoft Foundation, а также современные визуальные средства (панели, сценарии, иконки и т.п.) для внесения изменений типов данных вызывающей и вызываемой программ.

Фабрика программ с UML Дж. Гринфильда. Для будущей фабрики сформулированы методологические и технологические аспекты производства ПП по методу UML с использованием моделей архитектур систем и компьютеров, механизмов интеграции разнородных компонентов с типами данных FDT и описанием интерфейса в языках (IDL, XML, RDF и т.п.). Главные концептуальные объекты производства: reuse, которые накоплены в общепринятых хранилищах (библиотеках, репозиториях и т.п.) Интернет и имеют сертификаты качества; активы (assets), программы, приложения и системы; модели, шаблоны и инструменты UML при реализации ПП на линии производства; веб-сервисы, процессы линий; методики измерения и контроля качества ПП и т.п.

Анализ моделирования UML на данной фабрике показал следующее: язык UML является языком эскиза ПП, который не допускает использования компонентов reuse и интерфейса на разных ЯП; проблема модификации ПП не решена в визуальном языке UML и др. Фактически автор разработал меморандум современной фабрики ПП, которая базируется на reuses, применяемых на производственных линиях, моделях систем, каркасах процессов и продуктов. Конкретной фабрики построить не удалось.

Фабрики программ фирмы Microsoft. Базис данной фабрики — среда MS.NET, содержащая большое количество средств и инструментов, а именно: готовые ресурсы (компоненты, сервисы) Интернета, языки – JAVA, C++, Basic, Java, Pascal, C#, библиотеки CLR и FCL, сборка кодов (exe, dill) в готовый ПП, веб-обслуживание разного назначения, управление PM-2007 группами разработчиков ПП в виртуальной среде VSTS, MSF, которые могут располагаться в разных местах мира. В состав средств автоматизации входят: пакет инструментов VSTS-2005 (Visual Studio Teams Systems); MSF (Microsoft Solution Arhitecture) для построения производственной архитектуры предприятия, стандарт PMBOK для управления группой; модели процессов и систем; Professional Studio и Foundation Server для поддержки процессов проектирования, кодирования тестирования, формирования версий ПП (SDLC, IDE, MS Office, MS SQL server, MS Visual Studio 2005); модель усовершенствования процессов (CMMI Process Impovement), в частности процесса сборки, который использует CLR-библиотеки (Common Language Routine), классы, FCL-типы, трансляторы, General code (exe), Portable Executable code и т.п. Среда может определять сроки, трудозатраты и показатели качества изготовленных ПП.

Инфраструктура вычислений научных задач в среде Grid. Европейский проект Grid ориентирован на организацию распределенных вычислений задач в разных научных направлениях (физика, математика, медицина, биология и др.) [14, 15]. Он включает ряд подпроектов: Gcube — операционная среда, ETICS — как сборочная среда и т.п., и предназначен для производства распределенных систем разного назначения методом интеграции (сборки) существующих готовых КПИ и ПП с применением веб-порталов и многоплатформенных ресурсов.

Технология создания пакетов из исходных программ или комбинаций перекомпилированных программ в двоичном представлении обеспечивается процессом доступа к репозиторию для выбора компонентов системы в альтернативной сетевой среде Grid. Задача представления типов данных объектов среды: Проект, Подсистема и Компонент, реализуется с использованием типового формата CIM как механизма связи между разными компонентами с помощью системы MySQL на основе аннотации ряда свойств компонентов (имя, лицензия, URL-репозиторий и т.д.), глобального уникального идентификатора — ID (GUID) и скомпилированного компонента с отображением его в конфигурационном файле ПП.

Сервисы — главные ресурсы инфраструктуры вычислений. Они обеспечивают услуги, ориентированные на вычисления научных задач глобального масштаба. Ресурсы задаются в протоколе, в котором передаются данные по распределенной сети. Сервисы строятся с помощью стандартных протоколов, интерфейсов API и инструмента SDK (Software Development Kits). Сборка разных программ, компонентов, подсистем и систем научного назначения реализуется глобальными протоколами (Global Protocol). Подсистема ETICS содержит средства разработки, тестирования новых пакетов и услуг, а также сборку и конфигурирование программных элементов с помощью механизмов плагинов [46, 48] и открытых интерфейсом услуг для потребителей или поставщиков, как на фабрике программ.

Главные особенности рассмотренных фабрик программ — это методы производства, ТЛ и инструментальная поддержка операционной среды для автоматизации процессов производства ЖЦ или ТЛ.

3. СТРУКТУРА И НАЗНАЧЕНИЕ РЕСУРСОВ ФАБРИКИ ПРОГРАММ

Исходя из теоретических достижений, закономерностей развития технологии программирования и анализа зарубежных вариантов фабрик программ на современных платформах компьютеров, новых идеях обеспечения взаимодействия разнородных программ с использованием теории организации фундаментальных и общих типов данных (Fundamental Data Types — FDT, GDT (General Data Types – ISO/IEEC 11404), установлены общие черты, свойственные разным фабрикам [26–39]. Это прежде всего операционная среда (типа

SUN ONC, MS.Net, Oberon, Babel, Grid, Eclipse и др.), метод программирования (компонентный, структурный, сервисный и др.), средства (ЯП, Rational Rose, CLR, и др.) и инструменты, поддерживающие процессы линий изготовления ПП или разработки отдельных компонентов, а также библиотеки готовых продуктов, сервисное обслуживание разных аспектов производства (данных интерфейса, качества, управления, контроля, планирования, расчета разных затрат и др.). Главный ресурс фабрики — специалисты по производству программ (аналитики, программисты, инженеры, тестировщики, контролеры и т.п.).

Определение 2. Фабрика программ — это интегрированная инфраструктура сборочного производства ПП (компонентов, подсистем, систем, модулей (блоков) и семейств систем, АСУ, АСУТП и др.), предназначенная для выполнения государственных, коммерческих и других заказов на ПП [49].

Фабрика программ включает комплекс средств, инструментов и сервисов для производства ПП на процессах ТЛ. Ядро фабрики — операционная среда и метод изготовления ПП (UML, компонентный, структурный, модульный, сервисный и др.). Обязательное условие сборочного конвейера — средства связи разноязыковых программ, аналогично тому, как это реализовано в MS.Net (CRL,

При ориентации на сервисный метод в среду вводятся дополнительные возможности по созданию набора сервиса и обслуживания, связанного с выбором и использованием его при производстве ПП.

Ресурсы для производства ПП на фабриках. К ним относятся следующие.

Технические ресурсы: платформы, процессоры (Intell, IBM, Apple, MS; коммуникации (OSI, TCP/IP; компьютеры пользователей; файлы и серверы; локальные и глобальные сети; электронная почта (e-mail); тестеры и т.п.

Технологические ресурсы: библиотеки, репозитории готовых ПП (КПИ, Reuses, Assets, Applications, Domains, Systems); методики методов программирования сборочного типа; руководства, методики с языков интерфейсов объектов (IDL, API, DII, SIDL, XML, RDF и др,); стандарты (каркасы, шаблоны, контейнеры, процессы, проекты, системы и др.).

Общесистемные ресурсы: ОС, инструменты: клиент/серверные технологии; офисные системы (ридери/райтери форматов PDF, PS, HTML и т.п.); системы документооборота; утилиты (архиваторы, программы записи на носитель, конфигураторы и т.п.); средства защиты информации (антивирусные, парольные и др.); CASE-инструменты, трансляторы; графические инструменты; СКБД.

Человеческие ресурсы: группы разработчиков, служб управления и выполнения проектных работ (по планам, сетевым графикам), связанных с достижением качества, выявления рисков, формирования конфигурации, проверки правильности реализации проекта и т.п. Стандарт ISO/IEC 12207 определяет следующие группы:

— технико-технологической поддержки (изучение рынка, приобретение CASE, ПП, консультации и т.п.);

— защиты информации (паролей, ключей защиты, проверки и т.п.);

— технологической службы (сопровождение, поддержка ЖЦ, контроль и т.п.);

— качества (SQA-группа) с функциями планирования и выполнения ЖЦ (проверка работ, контроль качества рабочих продуктов, документов ПП и т.п.);

— верификации, валидации и тестирования компонентов или ПП на правильность задания требований, координации планов работ менеджером, проверки правильности ПП в тестовой среде системы и др.;

— руководителя проекта, который отвечает за финансовые и технические ресурсы, а также за выполнение проектных соглашений заказчика и управление разработкой ПП;

— менеджера проекта, ответственного за разработку программного проекта фабрики, согласующего требования, решения и планы работ и реализации по всем группам, срокам и стоимости;

— проектировщиков и программистов, которые отвечают за разработку проектных решений и программирование, разработку документов и разных выходных результатов;

— руководителя конфигурации (ответственные за версию) ПП, который регистрирует версии ПП, сохраняет твердые копии и версии с размежеванием доступа к ним.

Эти группы необходимы при любом индустриальном коллективном производстве ПП.

Стандартные ресурсы. Комитет по стандартизации ISO разработал ряд стандартов программной инженерии, которые регламентируют порядок разработки ПП, управления методами программирования на фабрике программ [44]. Рассмотрим основные из них.

Базовый процесс (БП) предназначен для «процессного продуцирования» ПП как вид инженерной деятельности по изготовлению ПП, включающий операции оценки, измерения, управления изменениями и усовершенствованием БП, прописанному в стандарте ISO/IEC 15504-7 («Оценивания процессов ЖЦ ПЗ. Установки на усовершенствование процесса»). Оценка зрелости организации или фабрики программ осуществляется с помощью модели зрелости CMM (Capability Maturity Models) [16] института SEI США, модели Bootstrap, Trillium и т.п. Согласно этим стандартам уровень зрелости организации зависит от наличия ресурсов, стандартов, методик и способностей (зрелости) членов коллектива фабрики изготавливать ПП в заданные сроки и от стоимости.

Жизненный цикл стандарта ISO/IEC 12207 «Процессы ЖЦ ПЗ» регламентирован разными направлениями деятельности по разработке, проектированию и управлению ПП, организации процессов (планирования, управления и сопровождения), измерения, оценивания продуктов и процессов. Наиболее важные из них — серия стандартов: ДСТУ ISO/IEC 14598 «Оценивания программного продукта», стандарт ДСТУ ISO 15939 «Процесс измерения», серия стандартов ISO/IEC 15504 «Оценивания процессов ЖЦ ПО», базовые стандарты качества — ISO 9001 «Системы управления качеством. Требования», ГОСТ 2844-94, ГОСТ 2850-94 и 9126 регламентируют разные аспекты обеспечения качества ПП.

Ядро знаний SWEBOK — это стандарт из десяти разделов программной инженерии, распределенных по двум категориям. Первая — это методы и средства разработки (формирование требований, проектирование, конструирование, тестирование, сопровождение), вторая — методы управления проектом, конфигурацией, качеством и БП [44]. Методы ядра знаний соответствуют стандартным процессам ЖЦ с учетом потребностей конкретной фабрики программ и регламентированной последовательностью процессов, начиная от требований, разработки проектных решений, определения каркасов ПП и выбора готовых компонентов для «наполнения» его соответствующим содержанием.

Ядро знаний менеджмента проекта — это стандарт по управлению проектом — PMBOK (IEEE Std.1490 «IEEE Guide adoption of PMI Standard. A Guide to the Project Management Body of Knowledge), разработанный институтом PMI [44, 45], содержит описание лексики, структуры процессов и областей знаний: *управление содержанием проекта* (планирования с распределением работ); *управление качеством* и контроль результатов на соответствие стандартам качества; у*правление человеческими ресурсами* согласно их квалификации и профессионализму.

Среда фабрик программ. Рассмотренные в разд. 2 разные виды операционных сред — необходимый атрибута для некоторой фабрики программ. Принятие решения об их полноте и функциональности для производства ПП зависит от наличия финансов и знаний менеджеров, которые будут заниматься изготовлением ПП определенного назначения. Экспериментальным вариантом фабрики программ в ИПС НАНУ является бесплатная система Eclipse [48], которая имеет базовые инструмен-

тальные средства для производства ПП из готовых КПВ, а именно:

— механизм плагинов в формате XML в средстве Plug Development Environment, которая обеспечивает автоматизированное подключение плагинов и новых инструментов (например, Protege, Java, RMI), репозиториев и готовых программ;

— автоматизированное подключение новых меню к интерфейсу пользователя, иконок, сценариев и т.п.;

— использование языка Java и механизма вызова RMI для описания разных программ и их объединения в выходном коде и т.п.

Эта система дополнена нами алгеброй операций компонентного программирования, средствами сборки, генерации и конфигурирования КПИ в семейство систем [23, 34]. Метод порождения и генерации моделируется на процессах создания репозиториев компонентов, КПИ (сертификация, накопление, выбор, интеграция и др.) и сборки разнородных программных объектов применительно к сгенерированным членам семейства СПС в среде Eclipse. Изучается среда Grid для внедрения в нее необходимых средств сборки ПП .

4. ПОСТРОЕНИЕ ФАБРИКИ ПРОГРАММ МЕТОДОМ ГЕНЕРАЦИИ КОМПОНЕНТОВ

Среди рассмотренных фабрик программ наиболее развитой является сетевая среда Grid. В ней решаются задачи, которые не могут решить обычные компьютеры, — это интенсивные вычисления и обработка больших объемов информации. Для этого требуются процессорные мощности и системы хранилищ данных, схемы взаимодействия гетерогенных платформ, расположенных в географически отдаленных административных доменах и т.п. Среда Grid станет самой мощной фабрикой для сборки разнородных и разноплатформенных программ для организации вычислений научных задач глобального масштаба. Сущность обеспечения задач сборки заключается в формализации и преобразовании несовместимых типов данных на разных платформах вычислений (от 16-разрядной до 64-разрядной), так как при обмене данными программ на разных платформах гетерогенной среды Grid могут возникнуть всевозможные коллизии.

В связи с этим мы исследуем подобную Grid генерационную среду Eclipse и в ней моделируем новое решение задачи преобразования передаваемых по глобальной сети данных, основанную на создании системы генерации общих типов данных GDT в FTD [22–29] современных ЯП, которая может использоваться при изготовлении программ для физических, биологических и других экспериментов.

Подход к генерации FDT $\Leftrightarrow$ GDT. Основы генерации общих типов данных GDT изложены в стандарте ISO/IEC 11404-2007. Тип данных рассматривается как математическое понятие, обозначающее множество значений элементов. Значения базового типа (целое, действительное и др.) определяются аппаратурой, компиляторами программ с ЯП и т.п. Операции над значениями типа — это аксиомы, которые отображают значение одного типа в значение другого типа. FDT-тип данных включает простые, структурные и сложные типы данных, множество операций и значений типов данных и связи с другими типами данных. Они рассмотрены в разд. 2 и более подробно в [12, 17, 18].

GDT в данном стандарте включают LI-тип данных (LI — Language Independed), независимый от ЯП. Это примитивные типы, независимые от других типов, и непримитивные, которые генерируются к FDT с помощью других типов данных этого стандарта. Аппарат генерации типов данных включает: выбор (choice), указатель (pointer), процедуру (procedure), запись (record), набор (set), портфель (bag), последовательность (sequence), массив (array), таблицу (table) и т.п. Генерация осуществляется специальным набором процедур (функций), которые необходимо разработать и использовать в разных прикладных системах.

Предлагается схема генерации GDT $\Leftrightarrow$ FDT , которая базируется на библиотеке следующих функций (процедур) в языке XML:

— преобразование типов данных $ЯП_1, \ldots, ЯП_n$;

— представление типов данных FDT к виду специальных функций;

— преобразование GDT к виду FDT;

— эквивалентные отображения GDT $\Leftrightarrow$ FDT.

Теория представления FDT, разработанная с примерами описания для класса ЯП 4GL, описана в [6, 18, 22]. Она будет использоваться и для реализации приведенного набора функций путем создания:

1) библиотеки функций для преобразования всех типов данных GDT (примитивных, агрегатных и генерированных) к FDT типам данных (простым, структурным и сложным) ЯП, которые обеспечат взаимодействие разноязычных компонентов в системах Eclipse и Grid;

2) спецификаций внешних типов данных компонентов средствами GDT и сохранение их в репозиториях или библиотеках среды фабрики;

3) форматов посредников взаимосвязи компонентов подобных stub с использованием функций GDT $\Leftrightarrow$ FDT для передачи данных от одного стандартного объекта к другому;

4) системы генерации компонентов и преобразования в них типов данных с учетом платформ компонентов, систем и семейств систем.

Реализованная библиотека типов данных GDT $\Leftrightarrow$ FDT станет необходимой в гетерогенной среде Grid для решения проблемы сборки разнородных компонентов в новых ЯП.

Развитие среды Eclipse для производства ПП. Интегрированная среда генерации Eclipse принята в качестве базовой в фундаментальном проекте института (2007–2011) [30, 47] и ориентирована на производство приложений и семейств программных систем (СПС) из компонентов и КПИ. Основным механизмом взаимодействия компонентов в ПП являются сгенерированные интерфейсные посредники при сборке разноязычных программ. Как фабрика программ с методом генерации она включает:

— готовые компоненты, информационные ресурсы Интернета, модули, сервисы, Legacy-systems и т.п.;

— систему генерации членов семейства из КПИ на основе генерирующей модели (GDM) с использованием языка описания специфики предметной области в языке DSL;

— онтологическую модель набора программных ресурсов предметной области ПИ, тестирования, экспертизы и оценки их качества;

— ЯП (C/C++, C#, JAVA, Pascal, Basic, Clear, Ruby и др.) для описания разноязычных компонентов, язык описания данных (XML, UML, RDF, FDT, GDT, ...), язык интерфейса (IDL, APL, SDL, протоколы, др.) компонентов и современные модели (MDA, MDD, PIM, PSM, GDM, DSML, Feature);

— инструментальную среду программирования, репозитории с разными онтологиями, системные средства (Protege, Eclipse, Aspectj, Java и др.) и технологические модули поддержки процессов (тестирования, надежности, экспертизы, сборки и др.;

— теория экспертного и байесовского оценивания КПИ, процессов ЖЦ и отдельных объектов, входящих в состав ПП.

Главными процессами линий производства ПП являются: тестирование КПИ, программ, приложений и др.; сборка разнородных программ, основанная на теории отображения типов данных FDT и GDT; экспертизы, измерения компонентов и членов семейства СПС; оценка надежности по результатам проверки программ; оценка показателей компонентов и ПП; сертификация компонентов и СПС; управление командой исполнителей и др.

Эти методы и процессы реализации ПП расширяют возможности современных фабрик производства разного вида ПП в современных интегрированных средах.

ЗАКЛЮЧЕНИЕ

Автор и некоторые сотрудники отдела «Программная инженерия» стояли в истоке идеи фабрик программ. В настоящей статье рассмотрены основные позиции концепции академика В.М. Глушкова и первоначальные пути ее развития в ИК, проанализированы действующие фабрики программ с интегрированными средами, методы программирования и другие аспекты производства ПП. В результате разработаны оригинальные методы и средства поддержки фабрики программ, ее оргструктура и ресурсы, необходимые для ее функционирования. Предложен набор средств сборки и преобразования общих типов данных к фундаментальным GDT $\Leftrightarrow$ FDT для применения соответствующих функций в гетерогенных средах для обеспечения новых видов интерфейсов в модулях посредниках и в конфигурационном файле ПП.

СПИСОК ЛИТЕРАТУРЫ

1. Капитонова Ю.В., Летичевский А.А. Парадигмы и идеи академика В.М. Глушкова. — Киев: Наук. думка, 2003. — 454 с.
2. Глушков В.М., Лаврищева Е.М., Стогний А.А. и др. Система автоматизации производства программ. (АПРОП). — Киев: Ин-т кибернетики АН УССР, 1976. — 134 с.
3. Сергиенко И.В., Парасюк И.П., Тукалевская Н.И. Автоматизированные системы обработки данных. — Киев: 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. — Киев, 1987. — 30 с. — (Препр. АН УССР, Ин-т кибернетики им. В.М. Глушкова).
10. 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. — С. 121–136.
11. Лаврищева Е.М. Интерфейс в программировании // Проблеми програмування. — 2007. — № 2. — С. 126–139.
12. Лаврищева Е.М. Проблема интероперабельности разнородных объектов, компонентов и систем. Подходы к ее решению // Матеріали 7 Міжнар. конф. з програмування УКРПрог’2010. — Киев: ИПС НАНУ, 2010. — С. 28–41.
13. Castelli D., Candela L., Pagano P., Simi M. // IEEE 2005 — CS Intern. Symp. Global Data Interoperab. (IEEE Comput. Soc.). — 2005. — P. 56–99.
14. Лаврищева Е.М. Становление и развитие модульно-компонентной инженерии программирования в Украине. — Киев, 2008. — 33 с. Препр. / Ин-т кибернетики им. В.М. Глушкова; 1.
15. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. — Киев: Наук. думка, 1991. — 213 с.
16. Лаврищева Е.М. Сборочное программирование. Некоторые итоги и перспективы // Проблемы программирования. — 1999. — № 2. — С. 20–31.
17. Лаврищева Е.М. Сборочное программирование. Теория и практика // Кибернетика и системный анализ. — 2009. — № 6. — С. 1–12.
18. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов: 2-изд. доп. и перераб. — Киев: Наук. думка, 2009. — 370 с.
19. Лаврiщева К.М. Перспективні дисципліни програмної інженерії // Вісн. НАН України. — 2008. — № 9. — С. 12–17.
20. Лаврищева Е.М. Классификация дисциплин программной инженерии // Кибернетика и системный анализ. — 2008. — № 6. — С. 3–9.
21. Norhrop L.M. Software SEI’s Product lineTenets // IEEE Software. — 2002. — **19**, N 4. — P. 32–39.

22. Грищенко В.Н., Лаврищева Е.М. Методы и средства компонентного программирования // Кибернетика и системный анализ. — 2003. — № 1. — С. 39–55.
23. Грищенко В.М. Метод об'єктно-компонентного проектування програмних систем // Проблеми програмування. — 2007. — № 2. — С. 113–125.
24. Фреге Г. Логика и логическая семантика. — М.: Аспект Пресс, 2000. — 512 с.
25. Лаврищева Е.М. Методы программирования. Теория, инженерия, практика. — Киев: Наук. думка, 2006. — 451 с.
26. Hoar K. О Структурной организации данных // Структурное программирование. — М.: Мир, 1975. — С. 92–197.
27. Турский В. Методология программирования: Пер. с англ. — М.: Мир, 1981. — 265 с.
28. Агафонов В.Н. Типы и абстракция данных в языках программирования // Данные в языках программирования. — М.: Мир, 1982. — С. 267–327.
29. Замулин А.В. Типы данных в языках программирования и базах данных. — М.: Наука, 1987. — 152 с.
30. Основы инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, Е.М. Лаврищева, Ю.В. Суслов: 2-е изд. — Киев: Академпериодика, 2007. — 680 с.
31. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение. — М.; СПб.; Харьков; Минск: Издательский дом Питер, 2005. — 730 с.
32. Лаврiщева К.М. Генерувальне програмування програмних систем і сімейств // Проблеми програмування. — 2009. — № 1. — С. 3–16.
33. Коротун Т.М. Моделі и методи інженерії тестування програмних систем в умовах обмежених ресурсів. — Автореф. дис. ... канд. фіз.-мат. наук / ІК НАНУ. — Київ, 2005. — 21 с.
34. Слабоспіцька О.О. Моделі та методи експертного оцінювання у життєвому циклі програмних систем. — Автореф. дис. ... канд. фіз.-мат. наук / ІК НАНУ. — Київ, 2008. — 21 с.
35. Задорожная Н.Т. Кероване проектування документообігу в управляючих іінформаціних системах. — Автореф. дис. ... канд. фіз.-мат. наук / ІК НАНУ. — Київ, 2004. — 21 с.
36. Задорожна Н.Т., Лаврiщева К.М. Менеджмент документообігу в інформаційних системах освіти. — К.: Педагогічна думка, 2007. — 220 с.
37. Бабенко Л.П. Проблемы повторного использования в программной инженерии // Кибернетика и системный анализ. — 1999. — № 2 — С. 155–166.
38. Бабенко Л.П., Лаврiщева К.М. Основи програмної інженерії: Посібник. — К.: Знання, 2001. — 269 с.
39. Лаврiщева К.М. Програмна інженерія — напрями розвитку // Пр. міжнар. конф. «50 років Інституту кібернетики імені В.М. Глушкова НАН України». — Київ, 2008. — С. 336–345.
40. Corbin J. The art of distributed applications. Programming Tech. for remote procedure calls. — Berlin: Springer-Verlag, 1992. — 305 p.
41. 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.
42. Бей И. Взаимодействие разноязыковых программ. — М.; СПб; Киев: Изд. дом. «Вильямс», 2005. — 868 с.
43. Гринфильд Дж. Фабрики разработки программ. — М.; СПб.; Киев: Изд. дом. «Вильямс», 2007. — 591 с.
44. Guckenheimer S., Perez J.I. Software engineering with Microsoft studio team system. — Crawfordsville, USA: Adison-Wesley, 2006. — 304 p.
45. ETICS: the International software engineering service for the Grid / A. Meglio, M.E. Bégin, P. Couvares, E. Ronchieri, E. Takacs // J. of Physics Conf. Ser. 119. — 2008. — P. 1–11.
46. Таковицкий О. Технология Grid computing. — Htm.-Grid /doc/Byte Magazine Online.
47. Лаврiщева К.М. Програмна інженерія. — Київ: Академперіодика, 2008. — 313 с.
48. Карлсон Д. Eclipse. — Лори, 2004. — 335 с.
49. Лаврiщева К.М. Теоретичні, прикладні та організаційні основи фабрик програм // Сб. праць міжн. конф. TAARST-2010. — 2010. — С. 151–160.
50. Коваль Г.І., Колесник А.Л., Лаврiщева К.М., Слабоспицька О.О. Удосконалення процесу розроблення сімейств програмних систем елементами гнучких методологій // Проблеми програмування. — 2010. — № 2–3. — С. 261–270.

Поступила 17.09.2010