УДК 004.4

Е.М. Лаврищева, О.А. Слабоспицкая, А.Ю. Стеняшин, А.Л. Колесник

ОБЪЕКТНО-КОМПОНЕНТНАЯ РАЗРАБОТКА ИЗМЕНЯЕМЫХ ПРОГРАММНЫХ СИСТЕМ

Объектно-компонентный метод (ОКМ) моделирования программных систем Лаврищевой – Грищенко развит согласованными моделями вариабельности систем и их вариантной конфигурационной сборки из компонентов, а также алгеброй операций изоморфного преобразования нерелевантных типов данных для этих компонентов. Введение в модель системы точек вариантности с их вариантами обеспечивает изменяемость систем и устойчивое взаимодействие их компонентов. Описана реализация формального аппарата в автоматизированном конфигураторе систем и его апробация в инструментально-технологическом комплексе ИПС НАНУ и экспериментальной фабрике программ КНУ им. Т. Шевченко.

Ключевые слова: объект, компонент, объектно-компонентный метод, модель вариабельности, артефакт, вариантная точка, готовый ресурс, управление моделями, конфигурационная сборка.

Введение

Большинство современных подходов к изменению сложной программной системы (ПС) все еще предполагают непосредственную корректировку ее кода. Она усложняет ПС, снижает ее качество и требует дополнительных затрат времени и ресурсов на устранение вносимых дефектов. Эти негативные последствия все более критичны в актуальных сегодня динамичных слабо формализованных предметных областях. Однако именно в них изменяемые ПС особенно востребованы из-за растущей нестабильности ожиданий потребителей и условий выполнения. Поэтому технологии разработки ПС, многократно изменяемых без доступа к коду, становятся актуальным вызовом программной инженерии.

Целый ряд подходов, предложенных в ответ на этот вызов, объединяет использование формализма Семейства ПС [1]. Это множество ПС с общим набором постоянных понятий и характеристик, а также поднаборами изменяемых характеристик отдельных ПС, названных вариантными. Изменения ПС, допустимые в их семействе, описываются моделью его вариабельности – пригодности ПС или артефакта к эффективному развитию, изменению, настройке или конфигурированию для использования в определенном контексте [1–3]. Стандартом де-факто служит модель характеристик (Feature Model). Характеристика – функция или показатель качества ПС, востребованный группой заинтересованных лиц, их требование либо ожидание.

Модель характеристик поддерживает проектирование изменяемых ПС как членов семейства. Однако их автоматизированную сборку из готовых ресурсов затрудняет отсутствие формализмов связывания ресурсов с характеристиками и композирования для их набора.

Альтернативный базис обеспечения изменяемости ПС предоставляет сборочное программирование [4, 5]. Заявленное в 1982 г. методом сборки больших систем из готовых модулей [6], сегодня оно интенсивно развивается в разнообразных формах: от сборочного конвейера (М. Фаулер, корпорация EPAM System) до фабрик индустриального производства программ (Дж. Гринфильд, И. Бей, Г. Ленц, технология AppFab) [7]. В Украине оно представлено, прежде всего, ОКМ моделирования ПС Лаврищевой – Грищенко [5, 8–12]. Преимущество ОКМ – это алгебры операций реинжиниринга компонентов повторного использования (КПИ) и сборки ПС из КПИ. Но ОКМ-разработке изменяемых ПС препятствует слабая предсказуемость характеристик такой сборки и ее реактивность (“with reuse”, а не “for reuse” [1]).

Цель работы – непротиворечивая интеграция технологий разработки семейств ПС, поддерживающих предсказуемые изменения наиболее эффективно,

и ОКМ для преодоления их ограничений. Статья обобщает результаты авторов в проекте ДР 0107U002205 ИПС НАН Украины под руководством доктора физико-математических наук, профессора Е.М. Лаврищевой.

Подходы к изменению систем

Конструирование линеек программных продуктов. На основании материалов [1] для развития ОКМ выбраны две технологии:

– конструирование линеек программных продуктов К. Пола (K. Pohl) [2];

– генерирующее программирование К. Чернецки (K. Czarnecki) [3].

Определяющая особенность первой технологии – представление процесса разработки семейства ПС взаимодействующими подпроцессами инженерии домена и приложений, которые координируются подпроцессами организационного и технического управления. Инженерия домена предназначена для определения обязательных и опциональных характеристик ПС в семействе (в виде модели характеристик) и создания готовых ресурсов их реализации (требований, архитектур, фрагментов кода и тестов, структур данных) вместе с правилами композирования ресурсов в ПС.

Модель характеристик формируется в подпроцессе задания границ семейства. Это иерархия опциональных и общих характеристик ПС с отношениями [2]:

– вариантного подчинения (подчиняющая характеристика реализуется в ПС только при реализации строгого подмножества подчиненных характеристик);

– импликации (реализация предпосылки влечет реализацию следствия);

– эквивалентности (характеристики реализуются в ПС одновременно);

– исключения (одновременная реализация характеристик недопустима).

Ресурсы создаются в подпроцессах анализа требований, реализации, проектирования и тестирования, образуя четырехуровневую платформу семейства.

В процессе инженерии приложений ресурсы платформы автоматизировано композируются, по заданным правилам, для реализации заданного набора обязательных и/или опциональных характеристик – подграфа модели характеристик.

В подпроцессах организационного и технического управления координируются операции инженерии домена и приложений, а также планируется и отслеживается создание и использование в ПС ресурсов платформы.

Генерирующее программирование изменяемых ПС. Вторая технология, интегрируемая с ОКМ, реализует парадигму разработки ПС и их семейств под девизом "от ручного труда к конвейерной сборке" с помощью специального формализма – генерирующей модели домена. В отличие от технологии К. Пола, для ее построения вводятся пространства *проблем* и *решений*, а также *база конфигурации* семейства ПС.

Пространство проблем содержит обязательные и опциональные понятия и характеристики ПС семейства. Для его представления широко используется модель характеристик.

В свою очередь, пространство решений объединяет объекты конвейерной сборки. Это готовые ресурсы реализации характеристик (каркасы, шаблоны, модули, КПИ, сервисы, артефакты разработки), представленные в современных языках программирования, предметно-ориентированных языках и языках описания интерфейсов. Наконец, элементами базы конфигурации являются механизмы описания, генерации и композирования ресурсов для реализации набора характеристик.

Генерирующая модель – отображение пространства проблемы в пространство решений, сопоставляющее допустимому набору обязательных и опциональных характеристик ПС конфигурационный файл. Он задает набор программных ресурсов для первичной реализации либо изменения характеристик ПС. Эти ресурсы автоматически порождаются (в случае трансформационной модели) либо выбираются (соответственно, при конфи-

гурационной модели) в пространстве решений.

Вместе с ресурсами конфигурационный файл описывает также и механизм(ы) их композирования из базы конфигурации. При этом формальное описание набора характеристик преобразуется в спецификацию программных ресурсов в языках программирования, задаваемых архитекторами семейства. При необходимости, для промежуточных преобразований могут использоваться релевантные предметно-ориентированные языки.

Технология сборки изменяемых программных систем

Формальный аппарат обеспечения изменяемости ПС. Сопоставительный анализ технологий конструирования линеек программных продуктов и генерирующего программирования, приведенный в предыдущем разделе, показывает критическую роль формализма связывания характеристик ПС с ресурсами их реализации для обеспечения изменяемости ПС. Именно его отсутствие требует ручного задания правил композирования ресурсов в ПС (в линейках продуктов) либо определения генерирующей модели (в генерирующем программировании).

Предлагается введение такого формализма за счет:

– определения модели вариантных характеристик семейства ПС, распространяющей модель характеристик на базовые артефакты процесса разработки (требования, архитектуру, программные ресурсы, тесты, структуры данных [13, 14]);

– выбора КПИ как ресурсов и конструктивного уточнения полученной модели с помощью ОКМ [14–16];

– введения интенсиональной объектно-компонентной модели семейства ПС, явно включающей полученное уточнение [15, 17].

Зафиксируем содержательные определения основных проявлений вариабельности [1–3], используемые далее.

Точка вариантности – представление артефакта процесса разработки ПС элемента поддерживаемого делового процесса, который может реализоваться несколькими способами.

Вариант для точки вариантности – способ реализации элемента делового процесса, который она описывает.

Зависимость (dependance) – отношение на множестве точек вариантности и вариантов, ограничивающее выбор вариантов для одних точек вариантности в зависимости от их выбора для других точек.

Ограничение (constraint) – зависимость, определенная только для точек вариантности. Типичные ограничения – отношения импликации/исключения.

Формализацию приведенных содержательных определений задает

Определение 1. Модель вариантных характеристик семейства ПС – это пара

$$M_{var} = (SV;\ AV); \qquad (1)$$

$$SV = \langle G_1; \langle G_t; TR_t \rangle, t=2,...,5; Con; Dep \rangle; \qquad (2)$$

$$AV(id_m) = \langle g_1; \langle \langle g_t, tr_t \rangle; t=2,...,m \rangle; \qquad (3)$$

$$\langle \langle p_t, tr_t \rangle; t=m+1,...,5 \rangle;\ cn;\ dp \rangle,$$

где SV – подмодель вариабельности в структуре семейства ПС;

AV – подмодель вариабельности в артефактах ПС;

$G_t = (F_t, L_t)$ в SV (2) – граф, вершины которого – идентификаторы артефактов типа t (требований, $t=1$; элементов архитектуры, $t=2$; программных ресурсов, $t=3$; тестов, $t=4$; структур данных, $t=5$), а дуги – бинарные отношения на F_t, обусловленные моделью характеристик;

TR_t – двусторонние связи между артефактами типов $t-1$ и t;

Con и Dep – предикаты на $\otimes_{t=1,...,5} F_t$, задающие ограничения и зависимости для артефактов;

g_t и p_t – подграфы G_t, описывающие артефакты, реализуемые артефактом типа m с идентификатором id_m при разра-

ботке ПС;

остальные элементы $AV(id_m)$ (3) – сужения соответствующих элементов *SV*.

Подмодель SV (2) включает вариантные точки артефактов всех типов для внесения и отслеживания изменений в них самих и в структуре создаваемых ПС семейства, определяя спектр согласованных изменений характеристик ПС в соответствии с требованиями и реализующих их артефактов в вариантных точках.

В свою очередь, модель AV (3) задает унифицированное представление артефакта разработки ПС и ее самой как "сквозного" вертикального фрагмента SV.

Дальнейшее уточнение M_{var} (1) с помощью ОКМ преобразует ее в объектно-компонентную модель семейства ПС.

Метод ОКМ основан на обобщении понятия объекта с помощью теории Фреге [5, 8–11] и использовании в компонентном методе создания ПС результатов объектного анализа предметной области. Метод предполагает логико-математическое моделирование задач предметной области с помощью четырех специальных графов ее объектов. Для их формирования и использования он включает алгебру объектного анализа, внешнюю и внутреннюю компонентную алгебры и алгебраическую систему операций преобразования неэквивалентных типов данных, передаваемых между разнородными объектами в структуре ПС [5, 9, 11].

Модель вариантных характеристик M_{var} последовательно уточняется и встраивается в *объектную подмодель вариабельности* семейства ПС на четырех уровнях проектирования метода ОКМ, выделенных в ОКМ. Ее вид устанавливает

Определение 2. Объектная подмодель вариабельности – четверка графов, детализирующих друг друга:

$$OM = \langle G_1; (G_2, TR_2);$$

$$(G_3, TR_3); (G_4, TR_4) \rangle, \qquad (4)$$

где G_1 – граф объектов предметной области на обобщающем уровне проектирования;

G_2 – представление модели характеристик на характеристическом уровне;

G_3 – архитектурно-компонентная модель семейства структурного уровня;

G_4 – интерфейсная модель взаимодействия КПИ на поведенческом уровне.

Связи трассируемости TR_t в OM (4) сопоставляют объектам функций моделируемых ПС (т. е. вершинам графа G_1) методы и данные (отображаемые вершинами графов G_2 и G_3), необходимые для взаимодействия этих объектов в составе ПС.

В поддержку сборки ПС, моделируемых с помощью OM (4), предложено дальнейшее преобразование OM в компонентную подмодель вариабельности семейства. Его суть – реализация методов объектов, представленных в OM, за счет КПИ с входными и выходными интерфейсами специального вида, описанными в [15, 16]. При этом терминальным объектам в G_3 и их интерфейсам в G_4 соответствует один и только один терминальный КПИ.

Определение 3. Компонентная подмодель вариабельности – пятерка

$$CM = \langle RC; In; ImC; Fim; D \rangle, \qquad (5)$$

где RC – терминальные КПИ для терминальных объектов OM (4);

In – интерфейсы КПИ, параметры которых содержат точки вариантности;

ImC – реализации терминальных КПИ в заданной среде;

$Fim(\cdot)$ – функции преобразования входных параметров терминальных КПИ;

D – структуры данных в сигнатурах интерфейсов терминальных КПИ.

Наконец, искомая модель фиксирует

Определение 4 [15–17]. Модель семейства изменяемых ПС – кортеж

$$M_{SF} = \langle M_{var}; (KP,\ PR); PC; M_{FM}; MK \rangle, \quad (6)$$

где M_{FM} – модель характеристик семейства ПС;

KP – готовые ресурсы семейства;

PR – предикат принадлежности KP;

PC – сборочный предикат, определяющий операции сборки ресурсов;

MK – модель их конфигурации.

Замена в кортеже M_{SF} (6) модели M_{var} ее объектно-компонентным уточнением (OM, CM) (4), (5) преобразует (6) в *объектно-компонентную* модель семейства изменяемых ПС, где характеристики отображаются объектами, а готовыми ресурсами служат КПИ.

Разработка изменяемых ПС в их семействе (6) осуществляется за счет конфигурирования ресурсов, прежде всего КПИ, согласно универсальной модели M_{var} (1)–(3) либо объектно-компонентной модели (OM, CM).

При этом выполняется ряд операций управления вариантами ПС в их семействе:

– выделение общих и вариантных характеристик ПС для заданной предметной области;

– построение модели характеристик M_{FM};

– преобразование M_{FM} в объектно-компонентную модель вариабельности семейства;

– формирование артефактов и ресурсов ПС с подбором готовых ресурсов из базы конфигурации (при ее наличии);

– планирование многократного использования ресурсов, частности КПИ, для (пере)сборки ПС с заданными характеристиками, представленными подграфом G_1 в модели OM (4);

– рефакторинг КПИ и ПС с использованием функционально эквивалентных КПИ для адаптации ПС к новым условиям выполнения и/или требованиям потребителей;

– вариантная сборка КПИ по модели CM (5).

Метод сборки и преобразования данных интерфейса. Метод сборки, интерфейс и готовые модули – базис сборочного программирования [4, 5, 11].

Метод сборки – это способ соединения разноязычных объектов в языках программирования, который базируется на теории спецификации и отображения типов и структур данных этих языков с помощью алгебраических систем, включающих типы данных и функции их эквивалентного преобразования.

Интерфейс (межмодульный и межъязыковый) выступает в качестве главной доминанты взаимодействующих компонентов и объектов в современных глобальных и сетевых средах.

Межмодульный интерфейс – модуль-посредник между двумя взаимодействующими объектами, который выполняет функции передачи и приема данных между ними.

Межъязыковый интерфейс – совокупность средств и методов взаимно однозначного преобразования структур и типов данных между языками программирования с помощью алгебраических систем, а также функций (и макроопределений) библиотеки интерфейса для обмена данными между разноязычными модулями. Библиотека интерфейсов для языков программирования операционной системы ЕС, разработанная В.Н. Грищенко, включала 64 интерфейсные функции и была передана в 52 организации СССР.

Развитие интерфейса для новых типов неструктурированных данных выполнено аспирантом ИПС НАНУ А.Ю. Стеняшиным [12].

Концепция интерфейса, как средства связи разных типов объектов в языках программирования, получила развитие в 90-х годах в языках Application Program Interface (API) и Interface Definition Language (IDL).

При практическом использовании интерфейс включает описание формальных параметров и оператор вызова (CALL, RPC, RMI и т. п.) со списком параметров и

их значениями. Значения параметров проверяются на соответствие базовым типам данных с помощью специальных аксиом и операций преобразования типов данных в классе языков программирования. Результат отображения – сгенерированные функции эквивалентного преобразования типов, которые записываются в интерфейсном модуле-посреднике связываемых объектов.

Как отметил в 1975 г. академик В.М. Глушков [6, 18], метод сборки из КПИ аналогичен конвейеру фабрики (например, фабрики Р. Форда) сборки автомобилей из готовых комплектующих и стыковочных деталей. В нем роль комплектующих выполняют терминальные КПИ и ресурсы различной степени сложности, а роль стыковки – их интерфейсы.

Алгебра сборки КПИ имеет вид:

$$\varphi = \{C, CE, \Omega\}, \qquad (7)$$

где C и CE – множества КПИ и компонентных сред [5];

Ω = {*incon, redev, link, makeaw, add, insert, redo*} – операции сборки КПИ и преобразования обмениваемых между ним данных.

Обработка этих данных производится с помощью примитивных функций генерации общих типов данных (general data types стандарта ISO/IEC 11404:2007) для фундаментальных типов данных языков программирования (и наоборот).

Выделены следующие операции сборки:

$incon(A, B, Int_A, Int_B)$ – организация взаимодействия КПИ A, B с интерфейсами Int_A, Int_B;

redev(A, B) – трансформация типов данных КПИ A, B;

$link_{PS}$ – сборка одноязычных КПИ с параметрами интерфейса в языке IDL;

$link_{SF}$ – сборка разноязычных КПИ с интерфейсами в IDL или API;

makeaw(AS , A) – удаление компонента A из системы AS ;

add(AS, A) – добавление компонента A к системе AS;

insert(A , AS) – вставка компонента A в систему AS ;

redo(x y, BD) – передача данных x , y в БД с соответствующим форматом.

Теория преобразования данных. В сборочном программирования разработана теория преобразования простых и структурных фундаментальных типов данных различных языков программирования [4, 6]. Она включает алгебраические системы для функций преобразования структурных типов к простым типам в множестве фундаментальных типов данных. При нерелевантности передаваемых данных, представленных в различных языках программирования, используются примитивные функции преобразования (например, типа *integer* к типу *character* и наоборот).

Эта теория развита для общих типов данных (general data types стандарта ISO/IEC 11404:2007), которые отображаются в типы данных современных языков программирования путем генерации для них соответствующих фундаментальных типов [12]. Генерация использует набор функций (процедур) в языке XML:

– преобразование типов данных для последовательности языков;

– формальное описание фундаментальных типов данных;

– представление данных общего типа в формате соответствующего фундаментального типа для обработки и верификации его схемы данных;

– отображение между общими и фундаментальными типами данных.

Для реализации процедур разработаны:

– библиотека функций преобразования общих типов данных (примитивных, агрегатных и генерированных) к фундаментальным типам данных (простым, структурным и сложным), необходимым в гетерогенной среде взаимодействия разноязычных КПИ и ПС;

– спецификация внешних типов данных компонентов, подсистем и систем в различных языках программирования;

– форматы данных новых модулей-посредников с операциями обращения к соответствующим процедурам из числа перечисленных для передачи данных, имеющих нерелевантный или перестроенный тип, между взаимодействующими компонентами.

Отдельные операции программно реализованы аспирантом ИПС НАНУ А.Ю Стеняшиным в разделе «Трансформация ТД» Инструментально-технологического комплекса ИПС НАНУ (http://sestudy.edu-ua.net).

Предложенная теория преобразования типов данных может использоваться для организации взаимодействия КПИ в облачных вычислениях.

Конфигурационная сборка изменяемых ПС из КПИ. Задачи конфигурационной сборки КПИ в ПС включают [14, 17, 19]:

1) трассировку требований к вариантам КПИ на уровнях архитектуры ПС;

2) порождение вариантов КПИ и ПС в точках вариантности M_{var} (1)–(3) или (OM, CM) (4), (5) в поддержку новых требований к ПС с учетом ограничений на качество и ресурсы;

3) управление конфигурированием ПС за счет выбора КПИ для них;

4) управление вариабельностью семейства ПС путем контроля полноты и непротиворечивости его состава для удовлетворения потребностей разработчиков и потребителей ПС за счет адекватной и своевременной актуализации вариантов КПИ и ПС.

Управление конфигурацией КПИ предполагает следующие операции для решения задач 1)–4):

– сборку КПИ по M_{var} или *CM*, т. е. реализацию операций $link_{PS}$ и $link_{SF}$ из (7);

– аудит уровня удовлетворительности вариабельности семейства ПС для их потребителей и разработчиков;

– аудит целостности конфигурации ПС на этапах их сопровождения и эксплуатации;

– планирование корректирующих действий по восстановлению удовлетворительного уровня вариабельности и/или целостности конфигурации ПС.

Для поддержки эффективного выполнения выделенных операций предлагается интегрировать процесс управления конфигурацией КПИ в процесс проактивного обоснованного управления вариабельностью семейства ПС [16, 20], показанный на рис. 1. Он представлен композицией четырех функций, выделенных как обобщения действий в известном цикле управления Э. Деминга (Plan, Do, Check, Act) в проанализированных индустриальных технологиях разработки семейств ПС [1–3].

Согласно рис. 1, эта композиция выполняется в единой информационной среде, структурированной на основании модели вариантных характеристик M_{var} или ее объектно-компонентного уточнения (OM, CM).

Предлагаются следующие целевые функции *F* управления вариабельностью:

– планирование состава ПС и КПИ с их вариантами для целевой предметной области (F_1);

– непосредственная разработка терминальных КПИ и их автоматизированная сборка в целевые ПС по моделям M_{var} и/или ($OM; CM$) (F_2);

– анализ соответствия состава ПС и КПИ потребностям потребителей с диагностикой неадекватности и выработкой корректирующих действий (F_3);

– актуализация моделей M_{var} и/или ($OM; CM$) либо текущего состава ПС и КПИ для устранения неадекватности (F_4).

Функции F_1 – F_4 дополнены сервисной функцией инициализации их информационной среды, в частности M_{var} и/или ($OM; CM$), и комплексной организационно-технологической подготовки.

Согласно рис. 1 стандартизированные представления КПИ (например, в языке WSDL) накапливаются в репозитории

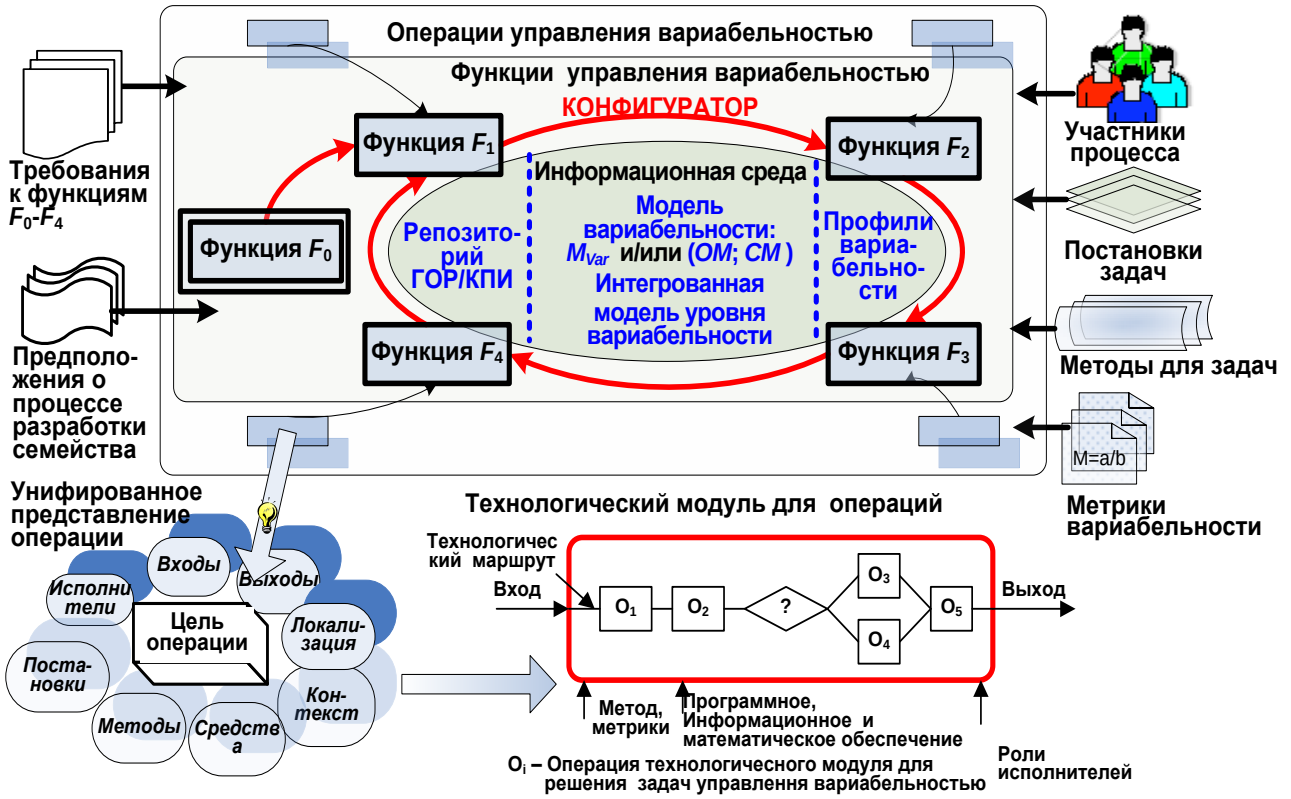

Рис. 1. Схема процесса управления вариабельностью семейства ПС

семейства изменяемых ПС. В подпроцессах реализации функции F_2 – их отбора, настройки и сборки в ПС – ключевую роль должно выполнять специализированное инструментальное средство – конфигуратор ПС. Именно конфигуратор должен обеспечивать объединение КПИ и их интерфейсов с вариантами артефактов ПС из репозитория в соответствии с моделями их вариабельности (AV) (3) либо ее объектно-компонентными уточнениями (OM, CM) (4), (5).

Реализация конфигуратора ПС

Макет конфигуратора и технологическая схема сборки КПИ с его помощью реализованы аспирантом ИПС НАНУ А.Л. Колесником в среде MS VS.Net и описаны в [14, 19].

Конфигуратор предоставляет заинтересованным лицам интерфейс для автоматизированного создания изменяемой ПС с заданными характеристиками из заранее разработанных КПИ. Его функциональные аналоги предлагаются в Интернет-магазинах автомобилей, ноутбуков и т. п., где покупатели могут заказать продукт с необходимыми им функциями.

Создаваемая ПС описывается в интерфейсе конфигуратора диаграммой характеристик – визуальным представлением графа G_1 из модели M_{var}, в котором перечисленные выше отношения между характеристиками обозначаются специальными пиктограммами, показанными на рис. 2. В среде конфигуратора элементы этой диаграммы реализуются средствами Windows Workflow Foundation в VS.Net и описываются предметно-ориентированным языком. Его словарь включает инструкции технологии, термины предметной области и артефакты процесса создания ПС.

Основной этап сборки КПИ конфигуратор – компиляция их исходного кода, преобразующая его в промежуточный код, или код выполнения в среде VS.Net. Специальная программа-посредник заменяет относительные адреса функций внешних библиотек их реальными адресами, используемыми при выполнении.

На рис. 3 показана схема работы конфигуратора. В качестве демонстраци-

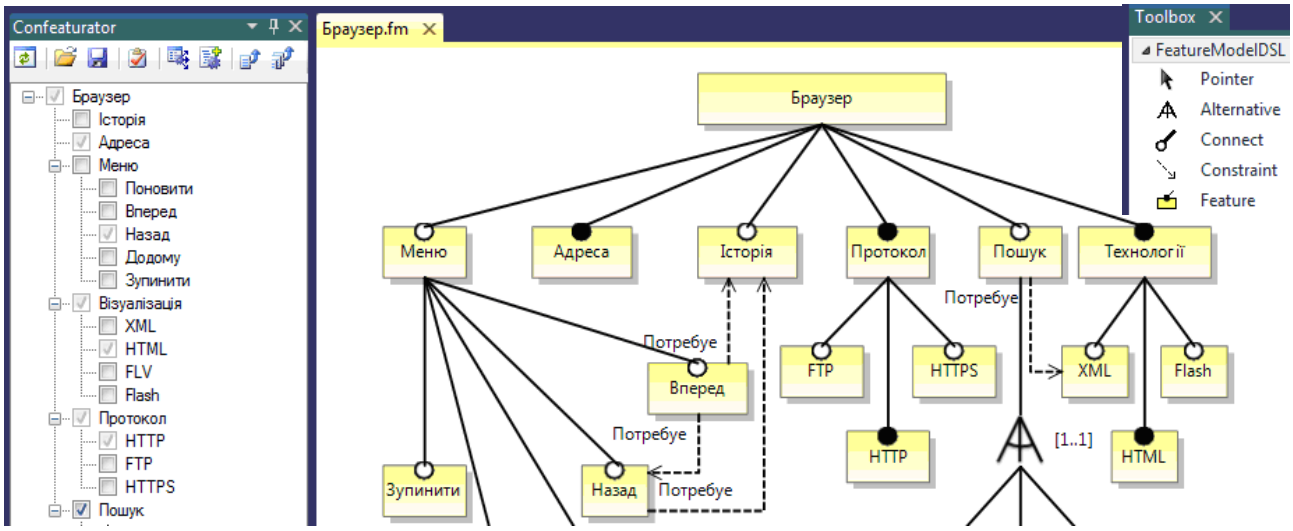

Рис. 2. Фрагмент диаграммы характеристик для семейства браузеров в конфигураторе

онного примера рассмотрена разработка семейства ПС для решения квадратного уравнения при различных значениях дискриминанта. Как показано на рис. 3, вначале разработчик получает задание на доработку/создание КПИ в среде VS.Net и помещает созданные КПИ в репозиторий семейства (стрелка 1 на рис. 3). В общем случае КПИ представлен парой файлов: ***.cs**, фиксирующих реализуемую бизнес-логику, и ***.xoml**, описывающих атомарные или составные объекты предметной области и задающих алгоритм выполнения бизнес-логики из ***.cs** файлов.

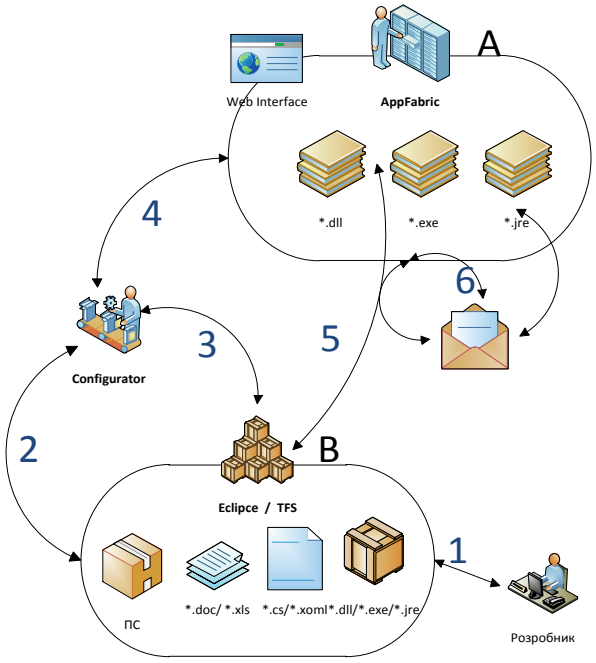

Рис. 3. Схема сборки КПИ в AppFab

Интерфейс конфигуратора обеспечивает связь с репозиторием и отображение в соответствующей экранной форме списка доступных КПИ (стрелка 2). Каждый элемент в окне дизайнера конфигуратора является атомарным КПИ, заранее разработанным и размещенным в репозитории. Под списком КПИ расположены детальные сведения о них из файлов ***.cs** (имя, родительский класс, описание, статус, привязка к конкретному методу).

Пользуясь графическим дизайнером, заинтересованные в ПС лица могут модифицировать или задать новую конфигурацию характеристик и построить соответствующую ПС из доступных КПИ.

Конфигурацию характеристик можно разместить на сервере или в репозитории семейства для последующего использования (стрелка 4 на рис. 3).

Класс КПИ представляет собой набор методов, выполняемых последовательно или параллельно согласно файлу ***.xoml**. Это позволяет манипулировать в среде конфигуратора теми КПИ, язык(и) реализации которых не поддерживаются в VS.Net, а также КПИ в форме сервисов.

В последнем случае для запуска программы AppFabric (блок A на рис. 3) следует задать команды: Пуск→ Программы →Internet Information Server (IIS). Выбирается нужный сервис для получения необходимой информации (адрес, связи, контракт) и занесения данных в код одного из методов файла ***.cs**.

Результатом является код:

```
1: private void codeActivi-
ty1_ExecuteCode_1
(object sender, EventArgs e)
2: {
3:  Uri address = new
Uri("http://localhost:63632/
/CountDiscriminant.
svc");//адреса
4:  WSHttpBinding binding =
= new WSHttpBinding();
//прив'язка
5:  EndpointAddress endpoint
= new EndpointAddress(address);
6:  ChannelFactory
<ICountDiscriminant>
factory = new
      ChannelFactory
<ICountDiscriminant>(binding,
endpoint); //контракт
7:  ICountDiscriminant
channel =
= factory.CreateChannel();
8:  D =
channel.GetDiscriminant
(a, b, c);  //D = b2 – 4ac:
9: }
```

Строки 3–7 обеспечивают создание специального объекта *channel* для отдаленного вызова, а строка 8 реализует его (стрелка 6 на рис. 3).

В среде конфигуратора можно запускать на выполнение алгоритм, загруженный в его редактор. Для этого нужно задать команды: Workflow → CompileWorkflow. Конфигуратор проанализирует и скомпилирует файлы ***.cs** и ***.xoml, сформирует библиотеку (***.dll**) и выведет сообщение о выполнении компиляции. Полученный файл загружается и извлекается из созданной библиотеки КПИ в оперативную память и запускает требуемый КПИ на выполнение.

Технологический комплекс сборки готовых ресурсов

Предложенная технология объектно-компонентной разработки изменяемых ПС с помощью CASE-инструментов (трансляторы для языков программирования, каркасы тестирования, генераторы и т. п.), а также интегрированных инструментальных средств (Eclipse, Protege, VS.Net, Corba, Java и т. п) реализована в Инструментально-технологическом комплексе (ИТК) ИПС НАНУ (http://sestudy.edu-ua.net) [21, 22].

В нем готовыми ресурсами являются КПИ. Они отображают функции и данные предметной области, представленные в модели M_{var} в виде функциональных объектов и объектов данных. Каждый КПИ специфицируется согласно соответствующим стандартам в языке WSDL, а его интерфейс – в языках IDL, API, SDIL и др. Это дает возможность собирать КПИ на единой основе, общей для всех видов разнородных ресурсов.

Технология разработки изменяемых ПС из готовых ресурсов в ИТК включает:

– проектирование ПС с использованием стандартного жизненного цикла;

– онтологическое проектирование доменов с заданием модели характеристик и архитектуры ПС из готовых компонентов;

– спецификацию разнородных программных ресурсов, прежде всего КПИ, в языках программирования, их реализацию и верификацию;

– отбор функционально готовых КПИ в репозитории;

– сборку разнородных КПИ и преобразование передаваемых между ними данных, нерелевантных по типу, формату, размеру и т. д.;

– изменение ранее созданных КПИ и ПС для адаптации под конкретные цели;

– описание специфики предметной области в предметно-ориентированных языках с использованием DSL TooLs VS.Net для получения исполняемого кода;

– тестирование КПИ и ПС, сбор данных для оценки качества ПС;

– сохранение результатов проектирования в репозитории КПИ;

– документирование КПИ.

В ИТК реализованы элементы основных современных парадигм программирования, необходимые для проектиро-

вания ПС из КПИ в различных предметных областях. В нем нашли отражение фундаментальные положения этих парадигм программирования, включая теорию взаимодействия и вариантности ПС, а также теорию моделирования и адаптации ПС, спроектированных за пределами ИТК. В среде ИТК представлены:

– технология изготовления ПС из КПИ на ряде технологических линиях;

– средства поддержки процессов жизненного цикла ПС по стандарту ISO/IEC 12207:2008 и оценки качества ПС согласно его рамочной модели в ISO/IEC 9126:2001;

– онтология вычислительной геометрии;

– линия обучения современным языкам программирования C#, Java и CASE-инструментам (Protégé, Eclipse, VS.Net, Java).

К технологическим линиям можно обращаться на Web-сайте ИТК. Для обучения дисциплине «Программная инженерия» можно воспользоваться электронным учебником: на сайте www.intuit.ru на русском языке; на сайте экспериментальной фабрики программ КНУ им. Т. Шевченко (programsfactory.univ.kiev.ua) украинском языке.

Выводы

Выявлены взаимодополнительные ограничения индустриальных технологий разработки семейств ПС и объектно-компонентного метода Лаврищевой – Грищенко для разработки изменяемых ПС: отсутствие формализма сборки программных ресурсов для характеристик ПС и, соответственно, слабая предсказуемость характеристик этой сборки.

Предложено преодоление ограничений за счет модели вариантных характеристик семейства ПС, распространяющей модель их характеристик на базовые артефакты процесса разработки. Для случая, когда ресурсами служат компоненты повторного использования, разработана объектно-компонентная модель вариабельности ПС, уточняющая модель вариантных характеристик, и модель семейства изменяемых ПС. Описана алгебра операций конфигурационной сборки компонентов по модели вариабельности и преобразования типов данных, обмениваемых между ними.

Предложена интеграция этих операций в процесс проактивного обоснованного управления вариабельностью семейства изменяемых ПС и технологическая схема последнего. Он представлен композицией функций планирования, реализации, контроля вариабельности и актуализации модели/состава семейства по его результатам. Функции реализуются в единой информационной среде, структурированной на основании модели вариантных характеристик или ее объектно-компонентного уточнения.

Проведена экспериментальная реализация конфигуратора компонентов в предложенном процессе. Описана апробация формального аппарата и конфигуратора в технологических линиях разработки изменяемых систем из компонентов, реализованных в ИТК ИПС НАН Украины.

1. *Product* Line Engineering [Электронный ресурс]. – Режим доступа: http://www.productlineengineering.com/. – Название с экрана.
2. *Pohl K., Bockle G., Linden F.J.* Software Product Line Engineering: Foundations, Principles and Techniques. – New York: Springer–Verlag, 2005. – 437 p.
3. *Чернецки К., Айзенекер У.* Порождающее программирование. Методы, инструменты, применение. – М.: ИД «Питер», 2005. – 730 с.
4. *Лаврищева Е.М.* Парадигмы программирования сборочного типа в программной инженерии. – Сб. трудов межд. конф. УкрПРОГ-2014. – С. 76–92.
5. *Лаврищева Е.М., Грищенко В.Н.* Сборочное программирование. Основы индустрии программных продуктов. – К.: Наук. думка, 2009. – 372 с.
6. *Лаврищева Е.М., Грищенко В.Н.* Связь разноязыковых модулей в ОС ЕС. – М.: Финансы и статистика, 1982. – 127 с.
7. *Lavrischeva K.M.* Theory and practice of software factories // In: Cybernetics and Sys-

tems Analysis. – 2011. – Vol. 47, N 6. – P. 961–972.
8. *Лаврищева Е.М., Грищенко В.Н.* Методы и средства компонентного программирования // Кибернетика и системный анализ. – 2003. – № 1. – С. 39–55.
9. *Грищенко В.Н.* Теоретические и прикладные аспекты компонентного программирования: автореф. дис. … док. физ.-мат. наук; Институт кибернетики им. В.М. Глушкова. – К., 2007. – 34 с.
10. *Лаврищева Е.М., Колесник А.Л., Стеняшин А.Ю.* Объектно-компонентное проектирование программных систем. Теоретические и прикладные вопросы // Вісник КНУ, серія фіз-мат. науки. – 2013. – № 4. – С. 150–162.
11. *Lavrischeva K., Stenyashin A., Kolesnyk A.* Object-Component Development of Application and Systems. Theory and Practice [Electronic resourse] // Journal of Software Engineering and Applications. – 2014. Mode of access: http://www.scirp.org/journal/jseaUSA.
12. *Лавріщева К.М., Стеняшин А.Ю.* Підхід щодо трансформації загальних типів даних стандарту ISO/IEC 11404 для використання в гетерогенних середовищах // 2nd International Conference on High Performance Computing-2012. – К.: КПІ, 2014. – С. 227–234.
13. *Лавріщева К.М., Слабоспицька О.О., Колесник А.Л., Коваль Г.І.* Теоретичні аспекти керування варіабельністю в сімействах програмних систем / // Вісник КНУ, серія фіз-мат. науки. – 2011. – № 1. – С. 151–158.
14. *Колесник А.Л.* Модели и методы разработки семейства вариантных программных систем: автореф. дис. … канд. физ.-мат. наук: КНУ им. Т.Г. Шевченко, 2013. – 22 с.
15. *Лавріщева К.М., Слабоспицька О.О.* Підхід до побудови об'єктно-компонентної моделі сімейства програмних продуктів / // Проблеми програмування. – 2013. – № 3. – С. 14–24.
16. *Slabospickaya O.* Feature Model of Software Product Line Enhancing to Enable Product Adaptability [in Ukrainian] // In: Bulletin of University of Kiev. Series: Physics & Mathematics, special issue, Kiev. – 2014. – P. 151–158.
17. *Лавріщева К.М., Колесник А.Л.* Концептуальні моделі розподілених компонентних систем // Проблеми програмування. – 2013. – № 1. – С. 21–33.
18. *Lavrischeva K., Aronov A., Dzubenko A.* Programs Factory – A conception of Knowledge Reprezentation of Scientifical Artifacts From Standpoint of Software Engineering / // Comp. and Inf. Sci., Canadian Center of Sci. and Edu. – 2013. – P. 21–28.
19. *Колесник* А.Л. Підходи до конфігурування компонентів повторного використання // Проблеми програмування. – 2011. – № 4. – С. 57–66.
20. *Kolesnyk A., Slabospitskaya O.* Tested Approach for Variability Management Enhancing in Software Product Line // In: Ermolayev V., Mayr H.C., Nikitchenko M. et al. (eds.): Proc. 8-th Int. Conf. ICTERI 2012, Kherson, Ukraine, June 6–10, 2012. – Vol. 848. – P. 125–133. [Электронный ресурс]. – Режим доступа: ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-31-p–155–162.pdf. – Название с экрана.
21. *Лаврищева Е.М.* Software Engineering компьютерных систем. Парадигмы, Технологии, CASE-средства программирования. – Киев: Наук. думка, 2014. – 284 с.
22. *Лавріщева К.М., Зінькович В.М., Колесник А.Л.* Інструментально-технологічний комплекс розробки и навчання прийомам виробництва програмних систем. Свідоцтво про реєстрацію авторського права на твір № 45292 від 27.08.2012 // Державна служба інтелектуальної власності України. – К.; 2012.

References

1. *Product* Line Engineering [Electronic resourse]. – Mode of access: http://www.product lineengineering.com/.
2. *Pohl K., Bockle G., Linden F.J.* Software Product Line Engineering: Foundations, Principles and Techniques. – New York: Springer–Verlag, 2005. – 437 p.
3. *Czarnecki K., Eisenecker U.* Generative Programming: Methods, Tools, and Applications. – Addison-Wesley, Reading, MA, USA, 2000 – 864 p.
4. *Lavrischeva E.M.* Paradigms of programming assembling type in software engeneering // Problems in Programming, 2014. – N 2–3. – P. 121–132. (in Russian).
5. *Lavrischeva E.M., Grischenko V.N.* Assembly Programming. Basics of Software Industry. – Kyiv: Naukova Dumka, 2009 (2nd ed.). – 372 p. (in Russian).

6. *Lavrischeva E.M., Grischenko V.N.* Interconnection of Multilingual Modules in OS ES. – M.: Finansy i Statystika, 1982. – 127 p.
7. *Lavrischeva K. M.* Theory and practice of software factories // In: Cybernetics and Systems Analysis, 2011. – Vol. 47. – N 6. – P. 961–972.
8. *Lavrishcheva E.M. Grishchenko V.N.* Methods and Tools of Component Programming // Cybernetics and Systems Analysis. – 2003. – Vol. 39. – № 1. – P. 33–45. (in Russian).
9. *Grischenko V.N.* Theoretical and Applied aspects of Component Programming: Ph. D. theses: spec. 01.05.03; V.M. Glushkov Institute of Cybernetics. – K., 2007. – 34 p. (in Russian).
10. *Lavrishcheva E., Stenyashin A., Kolesnyk A.* Object-Component Design. Theoretical and Applied Issues // Visn. Ser. Fiz.-Mat. Nayky, Kyiv St. Univ. im. Tarasa Shevchenka. Special Issue, 2013. – № 4. – C. 150–162. (in Russian).
11. *Lavrischeva K., Stenyashin A., Kolesnyk A.* Object-Component Development of Application and Systems. Theory and Practice [Electronic resourse] // Journal of Software Engineering and Applications, 2014. Mode of access: http://www.scirp.org/journal/jseaUSA.
12. *Lavrischeva K., Stenyashin A.* An approach for Standard ISO/IEC 11404 General Data Types transformation for use in heterogeneous environments // Second Int. Conf. on High Performance Computing-2012. – K.: KPI, 2012. – P. 227–234. (in Ukrainian).
13. *Lavrischeva K., Slabospitskaya O., Kolesnik A. at all.* The Theoretical View for Software Family Variability Management // Visn., Ser. Fiz.-Mat. Nayky, Kyiv Univ. im. Tarasa Shevchenka. – 2011. – N 1. – P. 45–53. (in Ukrainian).
14. *Kolesnik A.L.* Models and Methods for Variable Software Systems Familly Development: Ph. D. theses: Kyiv St. Univ. im. Tarasa Shevchenka. – 2013. – 22 p. (in Ukrainian).
15. *Lavrischeva K., Slabospickaya O.* An approach for Software Product Family Object-Component Model Elaborating // Problems in Programming. – 2013. – N 3. – P. 14–24. (in Ukrainian).
16. *Slabospickaya O.* Feature Model of Software Product Line Enhancing to Enable Product Adaptability [in Ukrainian] // In: Bulletin of University of Kiev. Series: Physics & Mathematics, special issue, Kiev. – 2014. – P. 151–158. (in Ukrainian).
17. *Lavrischeva K., Kolesnik A.* Conceptual Models for Distributed Software Systems // Problems in Programming, 2013. – N 1. – P. 21–33. (in Ukrainian).
18. *Lavrischeva K., Aronov A., Dzubenko A.* Programs Factory – A conception of Knowledge Reprezentation of Scientifical Artifacts From Standpoint of Software Engineering // Comp. and Inf. Sci., Canadian Center of Sci. and Edu. – 2013. – P. 21–28.
19. *Kolesnik A.* Approaches for Configuring Reusable Components // Problems in Programming. – 2011. – N 4. – P. 57–66. (in Ukrainian).
20. *Kolesnyk A., Slabospitskaya O.* Tested Approach for Variability Management Enhancing in Software Product Line // Proc. 8-th Int. Conf. ICTERI 2012, Kherson, Ukraine, June 6–10, 2012, CEUR-WS.org/Vol-848, urn:nbn:de:0074-848-8. – P. 125–133. [Electronic resource]. – Mode of access: ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-31-p–155–162.pdf.
21. *Lavrischeva E.M.* Software Engineering for Computer Systems. Paradigms, Technologies, CASE tools for Programming. – Kiev: Naukova Dumka, 2014.– 284 p. (in Russian).
22. *Lavrischeva K.M., Zinkovich V.M., Kolesnik A.L.* Instrumental-Technological Complex for Software Development and Production Scills Learning. A Certificate for authors' intellectual property N 45292 at 27.08.2012 // State Intellectual Property Service of Ukraine. – Kyiv, 2012.

Получено 06.10.2015

Об авторах:

Лаврищева Екатерина Михайловна,
доктор физико-математических наук,
профессор.
Количество публикаций в украинских изданиях – более 150.
Количество публикаций в иностранных индексированных изданиях – более 40,
http://orcid.org/0000-0002-1160-1077,

Слабоспицкая Ольга Александровна,
кандидат физико-математических наук,
старший научный сотрудник.
Количество публикаций в украинских изданиях – более 50,
Количество публикаций в иностранных индексированных изданиях – 5,
http://orcid.org/0000-0001-6556-0947,

Стеняшин Андрей Юрьевич,
аспирант.
Количество публикаций в украинских изданиях – более 10.
Количество публикаций в иностранных индексированных изданиях – 3.
http://orcid.org/0000-0001-7615-9024

Колесник Андрей Леонидович,
кандидат физико-математических наук, младший научный сотрудник.
Количество публикаций в украинских изданиях – более 10.
Количество публикаций в иностранных индексированных изданиях – 3.
http://orcid.org/0000-0001-1672-9201

Место работы авторов:

Московский физико-технический институт (государственный университет),
Россия, 141700, Московская область,
г. Долгопрудный,
Институтский переулок, д. 9.
Тел.: +7(495) 408 4554

Институт программных систем
НАН Украины,
03187, Киев-187,
Проспект Академика Глушкова, 40.
Тел. +38(044) 526 4286.

E-mail: lavryscheva@gmail.com,
ols.07@mail.ru,
andrey.stenyashin@gmail.com,
swabber@gmail.com