ПРОГРАММНО-ТЕХНИЧЕСКИЕ КОМПЛЕКСЫ

Е.М. ЛАВРИЩЕВА

УДК 681.03.06

ГЕНЕРИРУЮЩЕЕ И СБОРОЧНОЕ ПРОГРАММИРОВАНИЕ. АСПЕКТЫ РАЗРАБОТКИ СЕМЕЙСТВ ПРОГРАММНЫХ СИСТЕМ

Ключевые слова: *генерирующее программирование, компонентное, объектно-ориентированное программирование, семейство программных систем, компоненты повторного использования, взаимодействие, вариабельность, инструментально-технологический комплекс.*

ВВЕДЕНИЕ

В начале 70-х годов прошлого века Е. Дейкстра ввел понятие семейства программ с общей «семейной» постановкой задачи, с которой они порождаются как программы, и некоторые из них изменяются в связи с недостаточно заданной функциональностью или уточнением постановки задач семейства. Позднее появился термин семейство систем (в Product Lines SEI), семейство программных систем (СПС) в сборочном программировании и на фабриках изготовления программ из готовых компонентов повторного использования (КПИ), которых накоплено огромное количество в различных всемирных библиотеках и репозиториях.

Под семейством программных систем понимается совокупность программных систем (членов семейства), связанных между собой общим множеством понятий предметной области (ПрО), их общими характеристиками, готовыми КПИ, предикатами их отбора для уточнения функциональности производных вариантов членов семейства, операциями сборки (конфигурирования) и изменения членов СПС.

К. Чернецки в работе «Генерирующее программирование» (ГП) [1] рассматривает СПС как целевую функцию технологии разработки программных продуктов (ПП) под девизом «от ручного труда к конвейерной сборке» СПС. Элементы СПС не строятся с нуля, а генерируются на основе модели GDM (Generative Domain Model) — модели членов семейства систем. Она задается тремя элементами: членами семейства; компонентами реализации, из которых собираются члены семейства; конфигурационными знаниями (configuration knowledge) о спецификации членов семейства для получения корректного и качественного конечного ПП.

Объекты конвейерной сборки в ГП — готовые ресурсы (reuses, assets, servises) или КПИ, которые представлены в современных языках программирования (ЯП), DSL (Domain Spesific Languages) или exe с интерфейсами в IDL, API, RDF и др. КПИ могут выбираться из существующих хранилищ и использоваться при реализации задач конкретного члена семейства, а также участвовать в конвейерной сборке на продуктовых линиях (Product lines — PL) СПС.

Для построения модели GDM и определения членов семейства К. Чернецки вводит базовые понятия: пространство задач (problem space), пространство решений (solution space) и базу конфигурации (configuration base) СПС. Пространство задач отображает понятия СПС, членов СПС и их характеристики в FM (Feature Model) модели, а также функции и задачи, которые описываются PL или предметно-ориентированными языками типа DSL, UML2. Пространство решений — это компоненты, каркасы, шаблоны и КПИ реализации задач членов семейства СПС. Механизмы, правила описания, генерации компонентов и подбора КПИ для отдельных задач СПС — основные средства базы конфигурации.

В ГП описана методология проектирования моделей ПрО в терминах и понятиях, которые задаются формальными механизмами и средствами моделирования базовых характеристик модели FM для членов семейства СПС ПрО в зависимости от возможностей среды разработки и методов трансформации описаний элементов систем к платформам, где они будут располагаться и выполняться. Модели GDM и характеристик FM позволяют представлять общие понятия и свойства объектов ПрО, их взаимосвязи, а также знания о конфигурации КПИ в СПС. При реализации КПИ могут применяться такие виды программирования: ООП, компонентное, сервисное, аспектное, UML, инженерия семейств систем и ПрО. Концепция продуктовых линий института SE США (www.sei.com) используется для создания коммерческих ПП массового потребления.

Аналогичный подход к технологии индустриального изготовления программных систем (ПС), членов семейства и СПС из готовых КПИ [2–4] на технологических линиях (ТЛ) сборочного конвейера Глушкова разработан в отечественном сборочном программировании (СбП), начиная с 80–х годов прошлого столетия [2–7]. Эта технология активно развивается как сборочный конвейер (М. Флауер), ЕПАМ (Белоруссия), фабрики программ в Украине (http://programsfactory.univ.kiev.ua) и за рубежом (Дж. Гринфильд, И. Бей, Г. Ленц) [8–10], которые развивают индустрию ПП в современном информационном мире. Вопросам сборки СПС по технологическим линиям фабричного типа посвящена данная статья.

1. ОБЩИЕ ЦЕЛИ И ОЪЕКТЫ АВТОМАТИЗАЦИИ ЛИНИЙ ПРОИЗВОДСТВА СПС

Идея сборочного конвейера изготовления ПС, членов СПС из готовых КПИ исследовалась и разрабатывалась в отделе программной инженерии ИПС НАН Украины в рамках фундаментальных проектов (1998–2011 гг.), в том числе последнего из них, связанного с ГП. На рис. 1 приведена базовая структура проекта ГП в ИПС.

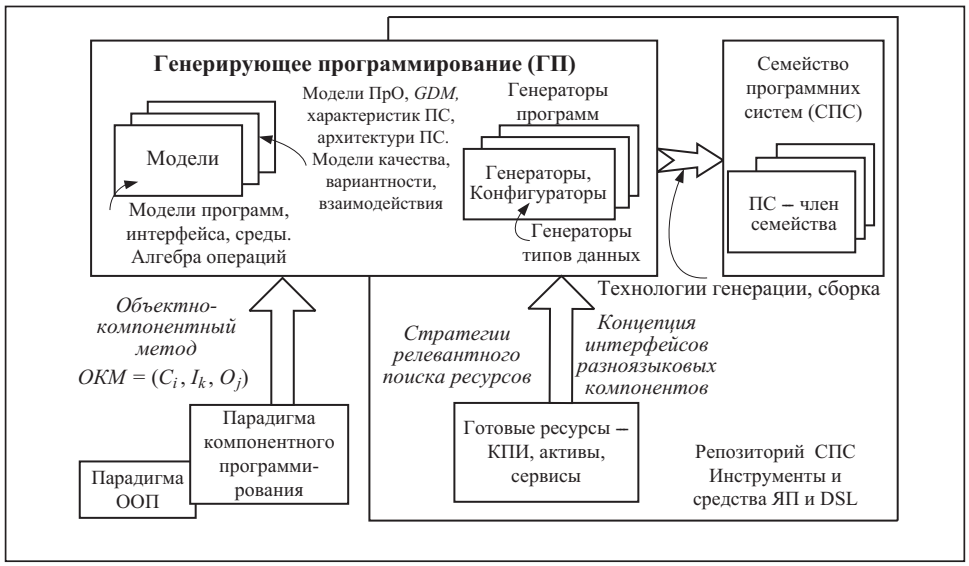

Рис. 1. Базовая структура проекта ГП

Основной целевой объект исследования в проекте ГП — семейство программных систем. Для его разработки изучались парадигмы ГП и КП, объектно-компонентный подход, КПИ, интерфейсы, методы проектирования и ведения процесса разработки СПС в репозитории и др. Формировалась научная концепция методологии разработки СПС, инструментальной среды и средств технологии разработки членов семейства, КПИ. В результате созданы оригинальные по своей сути отдельные теоретические и прикладные аспекты разработки СПС, которые расширяют методологию ГП К. Чернецки и сборочное программирования в частности. На основе практических экспериментов с КПИ и членами СПС в интегрованной среде ГП (VS.Net, Corba, IBM, Eclipse, Protégé и др.) были определены новые модели взаимодействия и вариабельности, технологические линии КПИ и реализован инструментально-технологический комплекс (ИТК). Результаты теории и практики разработки СПС изложены в ряде научных статей и в электронной монографии в контексте идей ГП [11]. В рамках проекта ГП разработано следующее:

— теория объектно-компонентного программирования [4–8] разнородных КПИ и метод сборки их по ТЛ с обеспечением обмена данных между ними через примитивные функции преобразования типов данных GDT стандарта ISO/IEC 11404–2007 к фундаментальным типам FDT;

— модели взаимодействия (Interconnection) программ, систем и сред, основанных на общих интерфейсах и средствах адаптации для миграции (переноса) их за пределы целевой среды изготовления систем в другую среду, в том числе в MS.Net, Grid, Vshpere, Cloud Computing, которые включают механизмы доступа к глобальным данным [10];

— оригинальная технология конструирования вариабельных (variability) систем и СПС с использованием готовых КПИ, которые оцениваются на качество, могут заменяться новыми, более функциональными и надежными [12–15];

— инструментальная среда ИТК ГП с новыми средствами и инструментами поддержки технологии сборочного производства СПС из КПИ, представленного спектром ТЛ, а именно — разработка и обслуживание КПИ в репозиториях, описание специфики доменов DSL-языками, генерация членов семейства, обеспечение взаимодействия КПИ между собой и оценка их качества [16–18, 21];

— подход к электронному обучению программирования в современных ЯП (C#, Java, C++, VBasic) и дисциплинам программной инженерии [16–20].

Эти идеи, концепции и инструменты дополняют ГП средствами создания индустриальных методов изготовления СПС из КПИ по технологическим и продуктовым линиям в современных операционных средах и Интернете [9–18].

Базовые определения объектов ГП: Product family — «группа продуктов или услуг, которые имеют общее управляемое множество свойств, которые удовлетворяют потребности определенного сегменту рынка или вида деятельности» («product family/product line — group of products or services sharing a common, managed set of features that satisfy specific needs of a selected market or mission», www.sei.com);

модель КПИ = (T, I, F, R, S), где T — тип компонента, I — интерфейс компонента, F — функциональность; R — реализация; S — сервис взаимосвязей с другими компонентами и средой [18–20];

интерфейс — спецификация программ и данных КПИ в языках IDL, API, WSDL и других для обмена данными с другими КПИ и программами, в том числе и членами СПС [21–23];

сборка компонентов и готовых КПИ по линиям подобно инженерии приложений, доменов, семейств систем и СПС в ГП [3–5];

репозиторий — сервис представления, накопления КПИ в электронном и стандартном представлениях для использования их при изготовлении новых систем семейств из готовых КПИ для разных ПрО [20].

2. РАЗВИТИЕ ОБЪЕКТНО-КОМПОНЕНТНОГО ПРОГРАММИРОВАНИЯ

Компонентное проектирование СПС из готовых КПИ — наиболее продуктивный путь разработки сложных ПС. В нем объекты рассматриваются на логическом уровне проектирования модели ПрО и ПС, с переходом к компонентам при физической реализации методов объектов. Программный компонент — некоторая абстракция, которая включает информационный раздел (назначение, дата изготовления, условия применения и т.п.) и собственно артефакт в виде реализации (implementation), интерфейса (interface) и схемы развертывания (deployment). Компонент описывается в любом ЯП, не зависящем от операционной среды и от реальной платформы, может иметь несколько реализаций и интерфесов исходя из сущности метода и условий операционной среды. Интерфейс определяет данные для связи одного компонента с другими. Развертывание — это выполнение физического файла компонента соответственно конфигурации [5–8].

Понятие компонента расширено и включает шаблон, каркас, контейнер. Они используются для представления общей структуры сложного объекта или для погружения в них готовых КПИ. Все типы компонентов запоминаются в репозитории компонентов, а их интерфейсы — в репозитории интерфейсов.

Сущность теории объектно-компонентного программирования. Основу теории составляет метод комплексного анализа ПрО и компонентного построения ПС. Объектно-компонентный метод (ОКМ) [6, 7] обобщает понятие объекта с помощью математических формализмов и теории Фреге. Он устанавливает теоретическую и практическую связь между объектным анализом и компонентным методом создания ПС. ОКМ представлен алгеброй объектного анализа, внешней и внутренней компонентной алгеброй и алгебраической системой преобразования неэквивалентных типов данных, собираемых из разноязыковых КПИ в сложные структуры ПС [5, 7].

Алгебра объектного анализа — это множество базовых функций объектного анализа для изменения денотат и концептов объектов (О) ПрО: $\Sigma = (O', \Psi)$, где $O = (O_1, O_2, \ldots, O_n)$ — множество объектов ПрО; $O_i = O_i(\text{Name}_i, \text{Den}_i, \text{Con}_i)$ — соответственно знак (имя), денотат и концепт объекта; концепт $\text{Con}_i = (P_{i1}, P_{i2}, \ldots, P_{is})$; $P = (P_1, P_2, \ldots, P_r)$ — множество предикатов.

Рассмотрим базовые операции объектного анализа:

— изменение денотата путем декомпозиции однородных объектов $\text{decds}(O_i)$: $O_i \to (O_{i1}, \ldots O_{ik})$, где $O_{ij} = O_{ij}(\text{Name}_{ij}, \text{Den}_{ij}, \text{Con}_{ij})$; $\forall j\ \text{Con}_{ij} = \text{Con}_i$; $\text{Den}_i = \text{Den}_{i1} \cup \ldots \cup \text{Den}_{ik}$ и неоднородных объектов $\text{decdn}(O_i): O_i \to (O_{i1}, \ldots, O_{ik})$, где $O_{ij} = O_{ij}(\text{Name}_{ij}, \text{Den}_{ij}, \text{Con}_{ij})$; $\forall j\ \text{Con}_{ij} = \varnothing$; $\text{Den}_i = \text{Den}_{i1} \cup \ldots \cup \text{Den}_{ik}$;

— изменение денотатов путем композиции однородных объектов $\text{comds}(O_{i1}, \ldots, O_{ik}): (O_{i1}, \ldots O_{ik}) \to O_i$, где $O_i = O_i(\text{Name}_i, \text{Den}_i, \text{Con}_i)$; $\forall j\ \text{Con}_i = \text{Con}_{ij}$; $\text{Den}_{i1} \cup \ldots \cup \text{Den}_{ik} = \text{Den}_i$ и неоднородных объектов $\text{comdn}(O_{i1}, \ldots, O_{ik}): (O_{i1}, \ldots, O_{ik}) \to O_i$, где $O_i = O_i(\text{Name}_i, \text{Den}_i, \text{Con}_i)$; $\text{Con}_i = \varnothing$; $\text{Den}_{i1} \cup \ldots \cup \text{Den}_{ik} = \text{Den}_i$.

Аксиома 1. Если предикат $P_t \in P$, $P_t \notin \text{Con}_i$ и $P_t(O_i)$ принимает значение истины, то $\text{conexp}(O_i, P_t): O_i \to O_i'$, где $O_i' = O_i'(\text{Name}_i, \text{Den}_i, \text{Con}_i')$; $\text{Con}_i \cup P_t = \text{Con}_i'$ является расширением концепта существующего объекта.

Аксиома 2. Если предикат $P_t \in \text{Con}_i$, то $\text{connar}(O_i, P_t): O_i \to O_i'$, где $O_i' = O_i'(\text{Name}_i, \text{Den}_i, \text{Con}_i')$; $\text{Con}_i' = \text{Con}_i \setminus P_t$, является сужением концепта существующего объекта.

Каждая из операций имеет определенный приоритет и арность, а также связана с соответствующими изменениями денотатов и концептов.

Утверждение. Множество операций Ψ алгебры Σ является полной системой операций для функций объектного анализа, обеспечивает адекватность перехода от функций к операциям алгебры объектного анализа.

Теория компонентного программирования включает модель компонента, интерфейса, компонентной среды и компонентную алгебру построения сложных программ из КПИ с преобразованием типов данных, передаваемых между ними [7].

Модель компонента: $\mathrm{Comp} = (\mathrm{CName}, \mathrm{CInt}, \mathrm{CFact}, \mathrm{CImp}, \mathrm{CServ})$, где CName — уникальное имя компонента; $\mathrm{CInt} = \mathrm{CInt}_i$ — множество интерфейсов компонентов; CFact — управление экземплярами компонента; $\mathrm{CImp} = \mathrm{CImp}_j$ — множество реализаций компонента; $\mathrm{CServ} = \mathrm{CServ}_r$ — множество системных сервисов.

Модель интерфейса: $\mathrm{CInt}_i = (\mathrm{IntName}_i, \mathrm{IntFunc}_i, \mathrm{IntSpec}_i)$, где $\mathrm{IntName}_i$ — имя интерфейса, $\mathrm{IntFunc}_i$ — интерфейс функции, $\mathrm{IntSpec}_i$ — спецификация интерфейса.

Модель компонентной среды: $CE = (\mathrm{NameSpace}, \mathrm{IntRep}, \mathrm{ImpRep}, \mathrm{CServ}, \mathrm{CServImp})$, где $\mathrm{NameSpace} = \mathrm{CName}_m$ — множество имен компонентов; $\mathrm{IntRep} = \mathrm{IntRep}_i$ — репозиторий интерфейсов компонентов; $\mathrm{ImpRep} = \mathrm{ImpRep}_j$ — репозиторий реализаций; $\mathrm{CServ} = \mathrm{CServ}_r$ — интерфейс множества сервисов; $\mathrm{CServImp} = \mathrm{CServImp}_r$ — множество реализаций сервисов.

Модель объединения компонентов. Свойства и характеристики объектов модели FM отображаются в описании интерфейса компонентов IDL (параметры In, Out) и операциях принадлежности:

- $O_k \in O, \mathrm{In}(O_k)$ — множество входных (In) интерфейсных объектов;
- $O_k \in O, \mathrm{Out}(O_k)$ — множество выходных (Out) интерфейсных объектов.

Результатом объединения двух объектов будет компонентный объект, у которого множество входных интерфейсов $(O_k \leftarrow O_l)$ совпадает со множеством выходных интерфейсов объекта-приемника, а множество выходных интерфейсов совпадает со множеством выходных интерфейсов объекта–передатчика:

$$O_k = (\mathrm{Out}(O_k), \mathrm{In}(O_k)), \quad O_l = (\mathrm{Out}(O_l), \mathrm{In}(O_l)),$$
$$O_k \cdot O_l = (\mathrm{Out}(O_k), \mathrm{In}(O_l)).$$

Аксиома 3. Композиция объектов $O_k \cdot O_l$ корректна, если объект-передатчик полностью обеспечивает сервис, необходимый объекту-приемнику, т.е. имеем $\forall I_m \in \mathrm{In}(O_k) \Rightarrow \exists I_n \in \mathrm{Out}(O_l) \wedge I_m = I_n$.

Компонентные объекты могут иметь несколько интерфейсов, наследующих интерфейсы других объектов $(O_k \leftarrow O_l)$, тогда последние предоставляют сервис для всего множества выходных интерфейсов: $O_k \leftarrow O_l \Rightarrow \mathrm{Out}(O_k) \subseteq \mathrm{Out}(O_l)$.

Если объект наследует другой объект, у которого множество входных интерфейсов содержит все его интерфейсы, а множество выходных интерфейсов содержит только интерфейсы, которые необходимы для задания сервиса, то получаем:

$$O_k \leftarrow O_l = \begin{pmatrix} \mathrm{Out}(O_k) \cup \mathrm{Out}(O_l), \\ \{I_m : (I_m \in \mathrm{In}(O_k) \cup \mathrm{In}(O_l)) \wedge \exists I_n \in \mathrm{Out}(O_k \leftarrow O_l) : \mathrm{exec}(I_n, I_m)\} \end{pmatrix}.$$

Объект, который наследуется, передает все интерфейсы и имеет свойства:

- транзитивности $\forall O_{1,2,3} \in O: O_1 \leftarrow O_2, O_2 \leftarrow O_3 \Rightarrow O_1 \leftarrow O_3$;
- симметричности $\forall O_k \in O \Rightarrow O_k \leftarrow O_k$.

Компонентная алгебра: $\Sigma = \{\varphi 1, \varphi 2, \varphi 3\}$, где $\varphi 1$ — внешняя алгебра, $\varphi 2 = \{\mathrm{CSet}, \mathrm{CESet}, \Omega_2\}$ — внутренняя алгебра компонентов, $\varphi 3 = \{\mathrm{Set}, \mathrm{CESet}, \Omega_3\}$ — алгебра сборки компонентов.

Внешняя алгебра: $\varphi 1 = \{\mathrm{CSet}, \mathrm{CESet}, \Omega_1\}$, где $\Omega_1 = \{CE_1, CE_2, CE_3, CE_4\}$, содержит операции: инсталляции, объединения компонентных сред, удаления компонента из компонентной среды, замещения компонента в ПС (PS).

Внутренняя алгебра: $\varphi 2 = \{\mathrm{CSet}, \mathrm{CESet}, \Omega_2\}$, где $\Omega_2 = \{O_{\mathrm{refac}}, O_{\mathrm{Reing}}, O_{\mathrm{Rever}}\}$ содержит операции рефакторинга, реинженерии и реверсной инженерии компонентов.

Алгебра сборки: $\varphi 3 = \{\mathrm{CSet}, \mathrm{CESet}, \Omega_3\}$, где $\Omega_3 = \{\mathrm{incon}, \mathrm{redev}, \mathrm{linkconf}, \mathrm{makeaw}, \mathrm{add}, \mathrm{insert}, \mathrm{redo}\}$ — операции сборки и взаимодействия КПВ с данными, обработка которых проводится с помощью алгебраических систем преобразования ти-

пов и форматов данных, передаваемых между КПИ, с помощью примитивных функций генерации типов GDT к FDT и обратно.

К операциям сборки отнесены следующие:

• Interconnect (incon), program system — $PS\ (A, B, C, \mathrm{Int}_A, \mathrm{Int}_B, \mathrm{Int}_C)$ — операции взаимодействия программ, компонентов A, B, C и их интерфейсов Int_A, Int_B, Int_C;

• redev $PS\ (\mathrm{Int}_A, \mathrm{Int}_B)$ — операции трансформации типов данных компонентов A, B;

• linkconf $PS(A, B, C, (\mathrm{Int}_{\mathrm{IDL}}, \mathrm{Int}_{\mathrm{IDL}}, \mathrm{Int}_{\mathrm{IDL}})$ — операции сборки путем конфигурирования программ A, B, C в одном языке L и параметрами интерфейса в языке IDL;

• linkconf $SPF(A_{L1}, B_{L2}, C_{L3}, (\mathrm{Int}_{\mathrm{IDL}}, \mathrm{Int}_{\mathrm{IDL}}, \mathrm{Int}_{\mathrm{API}})$ — операции сборки разноязычных компонентов, которые представлены ЯП ($L1, L2, L3$) с соответствующими интерфейсами в языках IDL и API;

• makeaw $AS(A)$ — операции удаления компонента A из системы AS;

• add $AS(A, C)$ — операции добавления компонентов A, C к системе AS;

• insert $F \Rightarrow AS$ — операция вставки компонента F в систему AS;

• rename $A \Rightarrow B$ — операция изменения имени компонента;

• redo x, $y \Rightarrow BD$ — операции передачи данные x, y в БД с соответствующим форматом.

Теория преобразования данных. В рамках КП и ОКМ разработана теория преобразования сложных FDT к простым в разных ЯП, которая изложена в [3–5]. В ней образованы алгебраические системы для простых и структурных типов данных и функции преобразования структурных типов к простым. Если имеет место нерелевантность в передаче данных от одного ЯП к другому, то используются примитивные функции преобразования (например, тип integer к character и наоборот).

Эта теория развита нами для общих типов данных GDT. Они отображаются в типы данных современных ЯП путем генерации GDT $\Leftrightarrow$ FDT [24, 25]. Генерация базируется на наборе функций (процедур) со следующим содержанием:

— преобразование типов данных для последовательности языков $\mathrm{ЯП}_1, \ldots, \mathrm{ЯП}_n$;

— формальное представление описания типов данных FDT;

— представление GDT в формат FDT для обработки;

— отображение типов данных GDT $\Leftrightarrow$ FDT.

Для реализации набора функций разрабатываются:

— библиотека функций преобразования типов данных GDT (примитивных, агрегатных и генерированных) к FDT (простым, структурным и сложным);

— спецификация внешних типов данных компонентов, подсистем и систем с использованием GDT и накопление их в одном из репозиториев интегрированной среды ИТК ГП [21];

— форматирование данных для новых посредников, подобных stub, с операциями обращения к соответствующим функциям GDT $\Leftrightarrow$ FDT.

Данная теория преобразования типов данных может применяться при вычислении компонентов в среде Cloud Computing.

Вариант реализации отдельных аспектов теории КП выполнен на веб-сайте http://sestudy.edu-ua.net. В нем представлен спектр простых линий технологии разработки и взаимодействия программ, систем и КПИ, а также на линии разработки КПИ, поиска КПИ в репозитории, программирования КПИ в языках C#, Java и е-обучения дисциплине — программная инженерия.

3. АСПЕКТЫ ТЕОРИИ ВЗАИМОДЕЙСТВИЯ ПРОГРАММ, СИСТЕМ И СРЕД

Сущность теории. Проблема взаимодействия систем реализована в стандартной модели OSI (Open System Interconnection) открытых систем. Она опреде-

ляет различные уровни взаимодействия систем в компьютерных сетях. Средства взаимосвязей определены на семи уровнях: прикладной, представительный, сеансовый, транспортный, сетевой, канальный и физический. Они обеспечивают системные средства взаимодействия, реализуемые операционной системой и аппаратными средствами. Модель OSI не включает средства взаимодействия приложений в разных средах.

Для взаимодействия приложений используются механизмы типа RPC/RMI, которые реализуют связь между компонентами приложения и могут обращаться с запросом к прикладному уровню модели OSI. Такие механизмы взаимодействия реализованы в современных операционных средах (VS.Net, Corba, Eclipse, Java и др.) и поэтому приложения, изготовленные в одной из сред, не могут переноситься в другую среду [9–11, 26–32]. В связи с этим предложен подход к взаимодействию членов семейства между собой, если они располагаются в разных средах (рис. 2).

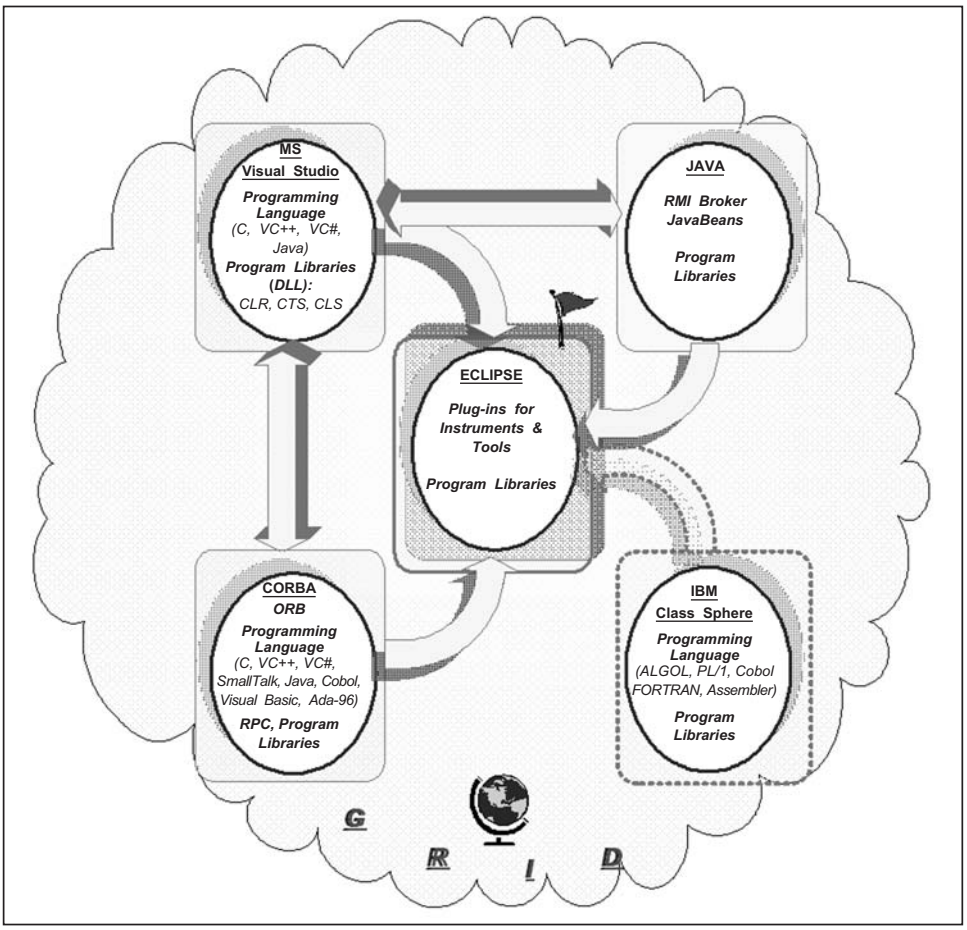

Рис. 2. Взаимодействие между современными средами

Модель взаимодействия M_{inter} имеет такой вид:

$M_{\text{inter}} = \{M_{\text{pro}}, M_{\text{sys}}, M_{\text{env}}\}$, где $M_{\text{pro}} = \{\text{Com, Int, Pr, Pro}\}$ — модель программы или ПС; Com— компонент, Int — интерфейс, Pr — программа, Pro — тип запроса (RPC, RMI, Icontract);

$M_{\text{env}} = \{\text{Env, Int, Pro}\}$ — модель среды, в которой Int, Pro, Icontract / IP задают совокупность внешних интерфейсов, программ с помощью протокола Icontract / IP, через который передаются данные между распределенными программами и средами.

Модель среды M_{inter} по отношению к стандартной модели OSI — модель верхнего уровня, включает программные элементы, интерфейс и среду.

Программный элемент специфицируются ЯП, DSL, IDL, API и сетевыми языками XDL, RDF и т.п. Элементы сохраняются в библиотеках и репозиториях среды ИТК ГП [21] и используются при выполнении разных функций технологии объединения и взаимодействия КПИ, программ и систем.

Интерфейс [22, 23] как объект модели взаимодействия включает параметры, набор операций и предикатов, которые определяют необходимые действия по обработке данных, переданных от одного программного элемента к другому. Он играет роль посредника между программами, которые обмениваются между собой данными. Если типы данных оказывались нерелевантными, выполнялось их прямое и обратное преобразования. Проблему взаимодействия между клиентными и серверными программами выполняют Service consumer и Service provider через протокол Icontract в системе WCF (http://127.0.01:4000/icontract). При взаимодействии пользователей и провайдеров с помощью протокола Icontract могут возникнуть «конфликтные ситуации», связанные с нерелевантностью типов и форматов данных.

Для поддержки процессов разработки программ в ЯП (MS.Net, CORBA, Java) и сборки их в члены СПС в ИТК ГП включена система Eclipse, на базе которой разработаны специальные механизмы взаимодействия для следующих пар сред: Visual Studio $\leftrightarrow$ Eclipse, CORBA $\leftrightarrow$ VS.Net, IBM VSphere $\leftrightarrow$ Eclipse.

Реализация моделей взаимодействия в ИТК. Модели взаимодействия программ, систем и сред практически реализованы студентами КНУ имени Тараса Шевченко и МФТИ [26–28] в операционных средах Visual Studio, CORBA, VSphere, Eclipse. На примере этих сред продемонстрированы механизмы взаимодействия программ и систем, а также перенос программы из одной среды в другую. Принципы реализации данных моделей приведены ниже.

Модель взаимодействия систем Visual Studio $\leftrightarrow$ Eclipse реализована на примере построения программ в языке C# Visual Studio, размещения их в репозитории Eclipse и выполнении средствами плагина Emonic и компонента NAnt платформы VS.Net. Для этих программ создается соответствующий архив, который разархивируется при интеграции в аналогичные папки среды Eclipse. Для выполнения программ в среде Eclipse создается проект, для которого из контекстного меню выбирается пункт *Import* $\rightarrow$ *FileSystem* и из файловой системы импортируется выбранный проект. В конфигурационном файле проекта (.build) изменяется его содержание путем задания имен исходных файлов, библиотек и файлов ресурсов. При переходе в среду Eclipse используются исходные файлы программы, dll-библиотеки VS.Net и файлы ресурсов (.resx). Переход из этой среды в Visual Studio использует импорт проекта в среду Eclipse.

Модель взаимодействия сред CORBA $\leftrightarrow$ VS.Net реализована на примере программы в языке Java системы CORBA, которая после соответствующей компиляции и IDL-описания интерфейса передается брокеру ORB. Перенос этой программы на платформу MS.Net проведен с помощью пакета IIOPNet. Реализация задачи взаимодействия компонентов, разработанных в MS.Net (клиентская часть) и Java (серверная часть), выполнено системой CORBA. Вычисленные значения данных на платформе MS.Net через маршалинг MarshalByRefObject возвращаются исходному объекту. Взаимодействие программ между двумя средами Java и MS.Net выполнено процедурами: компиляция серверной части приложения; создание серверного Skeleton и описание интерфейсного объекта в языке IDL через утилиты rmic; преобразование параметров IDL-описания средствами CLS-библиотеки и пакета IIOPNet, присоединяющего клиентскую часть к библиотеке соответствующим сервисом.

Модель взаимодействия VSphere $\leftrightarrow$ Eclipse изучается для дальнейшей ее реализации в заданной среде VSphere. Для эксперимента также разрабатывается програм-

ма в ЯП, а описание интерфейса — в API. Реализованная программа будет выполняться сначала в виртуальной среде VSphere и Eclipse ↔ VS.Net.

4. ОРИГИНАЛЬНЫЙ ПОДХОД К КОНСТРУИРОВАНИЮ ВАРИАБЕЛЬНІХ СПС

Проблема изменения функциональности члена СПС или КПИ связана с изменением требований заказчика, необходимостью переноса СПС в новую среду функционирования или замены некоторого КПИ на другой (например, с более лучшими показателями качества и т.п.). Поэтому исследование проблемы вариабельности систем и семейств СПС, ориентированной на производство вариантов члена семейства, проводилось в рамках проекта ГП. Создана теория моделирования разных вариантов членов СПС в инженерии ПрО. Результаты теории и практики опубликованы в ряде научных публикаций [11–14, 32, 33], электронной монографии [15], а также представлены в ИТК ГП.

Определение вариабельных систем. Вариабельность — способность семейства систем, отдельной систем СПС или артефактов к расширению, замене и конфигурации их для решения задач конкретной ПрО [32]. Идея вариабельности СПС появилась в рамках работ, связанных с Product Lines SEI, как механизм поддержки вариантов систем семейства, изготовленных из ПП (компонентов, асетов, КПИ, сервисов) по требованию заказчика. Варианты СПС обеспечиваются «извлечением» из него отдельных КПИ либо заменой, вставкой в определенные точки новых функций КПИ для удовлетворения требований заказчика (рис. 3).

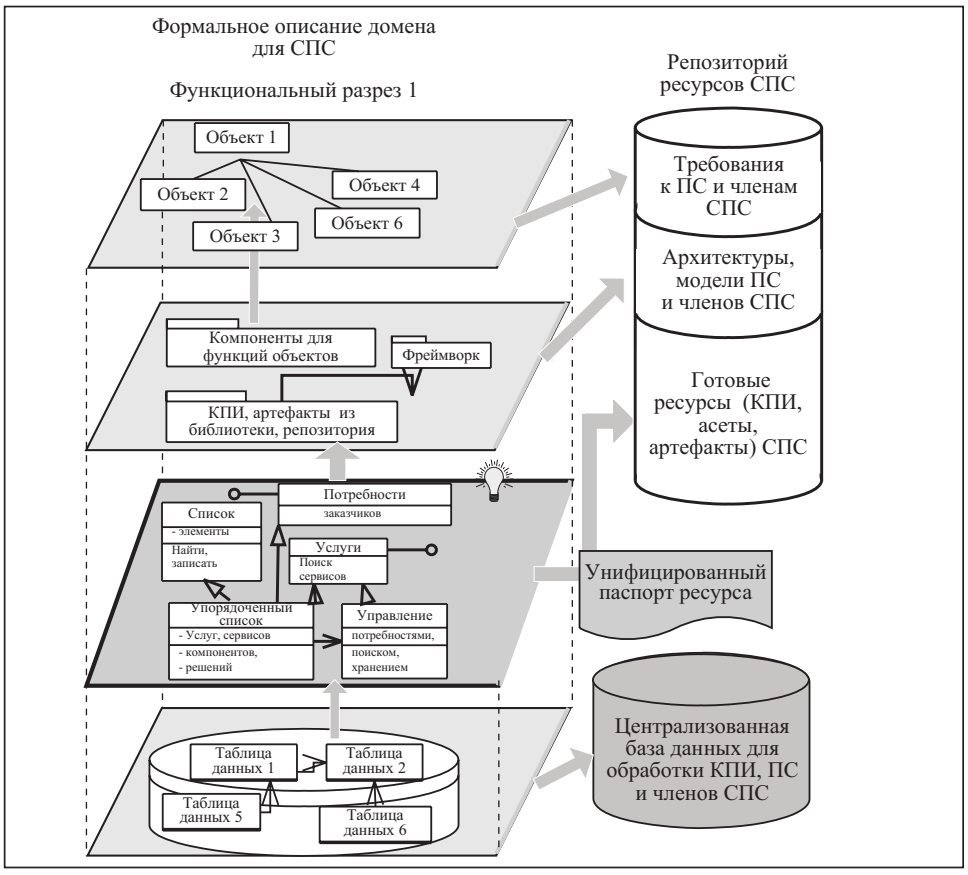

Рис. 3. Структура модели проектирования СПС в вариантах

Семейство СПС (SPS) — это кортеж из совокупности моделей:

$$M_{\text{SPS}} = \{M_{\text{PrO}}, KPV, PRG, RPC, M_{FM}, M_{\text{var}}, M_{\text{Config}}, M_{\text{ref}}\},$$

где M_{PrO} — модель ПрО; M_{FM} — модель характеристик членов семейства СПС; *KPV* — множество КПИ; *PRG* — предикат принадлежности к *KPV*; *RPC* — сборочный предикат, определяющий операции сборки КПИ в отдельные члены семейства СПС; M_{var} — модель вариабельности ПС, СПС; M_{Config} — модель процесса сборки (конфигурации); M_{ref} — модель рефакторинга.

Модель M_{var} определим с помощью двух связанных между собой моделей: объектной (ОМ) и компонентной (КМ). Их представление проведено как усовершенствование модельной среды и компонентной алгебры ПС, описанных в теории КП [4].

Составляющие элементы модельной среды КП ИТК — объектная, компонентная вариантная модели СПС, задаваемые на четырех уровнях проектирования.

Объектная модель имеет вид: $\text{ОМ} = \langle G_1^t; G_2^t, G_3^t, G_4^t \rangle$, где G_1^t — граф объектов ПрО, создаваемый на общем уровне проектирования ($t = 1$); G_2^2 — FM характеристического уровня ($t = 2$); G_3^3 — архитектурно-компонентная модель структурного уровня ($t = 3$); G_4^t — интерфейсная модель взаимодействия компонентов СПС на поведенческом уровне ($t = 4$).

Объектам функций G_1^t и их характеристикам соответствуют методы и данные (уровня 2, 3), необходимые для их реализации в СПС и обеспечения их взаимодействия.

Компонентная модель СПС — развитие ОМ, методы объектов которой реализуются КПИ для одного и только одного ее объекта и соответственно интерфейсов между ними. Модель имеет следующий вид:

$$\text{КМ} = \langle \text{RC, In, ImC, Fim} \rangle,$$

где RC — базовые компоненты множества компонентов C, которые соответствуют базовым объектам модели ОМ; In — интерфейс компонентов, среди параметров которого задается имя точки вариантности; ImC — реализация базового компонента в заданной среде; Fim (·) — функции преобразования входных и выходных параметров интерфейса во множестве данных сигнатуры интерфейса.

Компонентная модель ориентирована на обеспечение вариантности систем семейства на разных уровнях модели и платформ выходного кода, а также оценки полученного уровня вариабельности потребностям заказчика.

Разработка вариабельных СПС состоит в реализации вариантов членов СПС, а также процессов конфигурирования и рефакторинга КПИ или членов СПС. Основную роль в этих процессах выполняет конфигуратор в среде ИТК ГП. Он обеспечивает объединение КПИ и их интерфейсов с вариантами отдельных рабочих продуктов ПС, которые находятся в репозитории и реализуют логическую вариантность.

СПС с механизмами вариантов становится способным к расширению, изменению и настройки на новую среду реализации.

Для управления конфигурацией СПС из КПИ репозиторий ИТК ГП дополнен механизмами хранения, введения (изъятия) вариантов КПИ и сопоставления их между собой, а также принятия решений по сборке, объединению разных вариантов КПИ и с учетом требований к показателям качества ПС, стоимости и др.

Управление конфигурацией СПС начинается с обслуживания КПИ (подбор, поиск, выбор) в репозитории, проведения сборки КПИ и создания конфигурационного файла для выполнения ПС или СПС в среде ИТК ГП.

5. ОРИГИНАЛЬНЫЙ ПОДХОД К ИНЖЕНЕРИИ КАЧЕСТВА СПС

Сущность данного подхода состоит в обосновании процесса конструирования качественных членов семейств ПС и генерации новых ПС с регулированием показателей качества СПС из готовых КПИ и обязательными оценками показателей качества [12, 13]. Данный подход отсутствует в ГП. Инженерия качества СПС основывается на определении задач для данной ПрО с точки зрения качества. К инженерии качества СПС отнесены [14, 15]:

— измеряемые атрибуты артефактов семейства ПС, показатели качества которых необходимо оценивать;

— измеряемые атрибуты рабочих продуктов СПС с прогнозированными показателями качества для разработки члена семейства;

— метрики, модели и методики для принятия и обоснования решений разработчиков семейства ПС и СПС по отношению к заданным показателям качества;

— элементы поддержки метрик, моделей и методов разработки семейств ПС в среде ИТК ГП.

В инженерии качества СПС решаются такие задачи:

1) моделирование качества семейства ПС и оценивание качества сгенерированных артефактов на каждом процессе инженерии ПрО;

2) подбор компонентов из репозитория, которые удовлетворяют требованиям ПС;

3) верификация КПИ, которые помещены в репозиторий ИТК и используются в разрабатываемой СПС;

4) тестирования компонентов и собранных в ПС или СПС, выбранных из репозитория КПИ;

5) проверка качества членов ПС с точки зрения реализации функциональных требований к СПС.

Инженерия разработки качественных СПС представлена следующими образом.

Задача 1. Моделирование набора совместных нефункциональных требований к семейству ПС и их спецификация. Они могут конфликтовать с заданными показателями качества (например: высокая надежность $\rightarrow$ низкая эффективность, высокая эффективность $\rightarrow$ низкая модификация, высокое качество $\rightarrow$ высокие затраты).

Для спецификации конфликтных требований к качеству строится «матрица конфликтов», используется метод анализа иерархий для приведения в порядок, взвешивания характеристик и оптимизации их целевых значений.

Задача 2. Оценивание спецификации ПрО можно выполнять методом формальной инспекции, прототипирования, построения сценариев тестов и т.п. Инспекция проводится вручную опытным экспертом (или экспертами) с использованием опросных карт и форм данных измерений. Результаты анализа можно моделировать средствами XML-link.

Задача 3. Принятие решений при управлении вариантами ПС и обеспечения качества разрабатываемой СПС из КПИ и ПС в среде ИТК ГП.

Оценка качества ПС. В рамках работ по инженерии качества созданы новые формальные методики количественного измерения и оценки показателей качества программ в классе задач СОД для МО Украины [12, 15]. Получены такие оригинальные результаты:

— модель качества с ориентацией на оценку надежности ПП;

— концептуальная модель принятия решений с управления качеством, включая байесовские методы, методы системного контроля надежности на ранних процессах ЖЦ, количественного измерения требований к надежности и прогнозирования дефектов;

— модель распределения надежности системы из компонентов на основе функции полезности ПП $Q_{nc} = \sum_{s=1}^{m} w_s^* \cdot r_s$ в зависимости от приоритетов w_s^* и надежности $q_j = \prod_{n \in E_j} r_n$ отдельных компонентов.

Оценка качества включена в линию ИТК ГП для обеспечения расчета показателей качества и затрат на разработку ПС.

6. РЕАЛИЗАЦИЯ ТЕОРЕТИЧЕСКИХ АСПЕКТОВ СОЗДАНИЯ СПС В СРЕДЕ ИТК ГП

Методология изготовления СПС из готовых КПИ включает: теорию компонентного программирования; модели взаимодействия и вариабельности; комплекс технологических линий для построения разных элементов СПС; средства обучения фундаментальным основам программной инженерии. Предложенная методология отображает новые теоретические и прикладные аспекты технологии изготовления программных продуктов с учетом ядра SWEBOK (www.swebok.com) [11–14], ЖЦ стандарта ISO/IEC 12207, процессы которого адаптированы к отдельным ТЛ, включенных в ИТК ГП (рис. 4), как инструментов регламентированного производства СПС [18, 24].

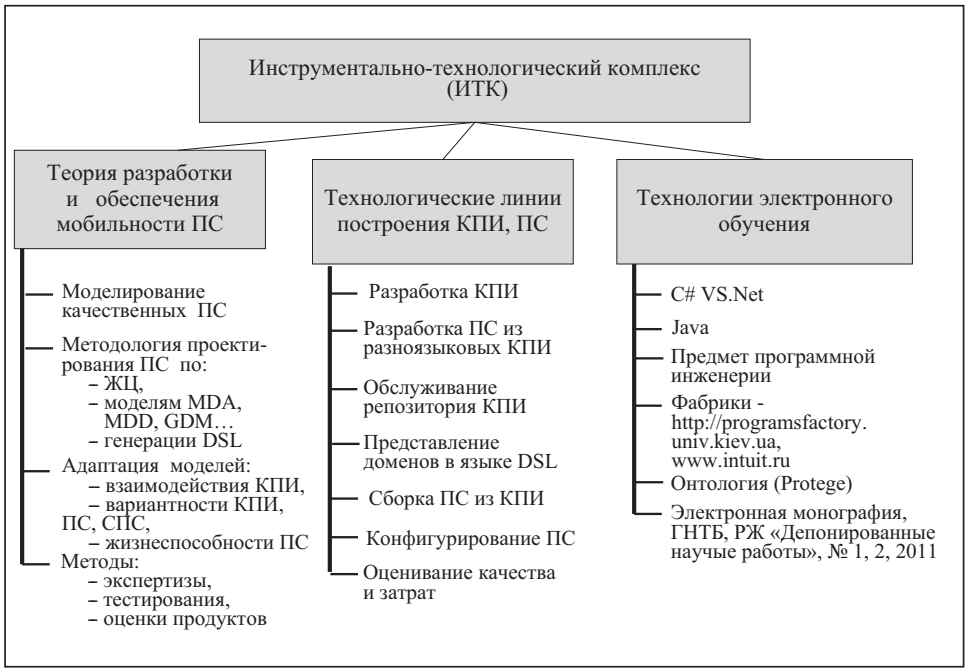

Рис. 4. Структура ИТК ГП

Основными задачами методологии изготовления СПС в среде ИТК ГП являются:

— разработка модели ПрО и ее характеристик для основных понятий домена, архитектуры СПС из готовых КПИ;

— спецификация разнородных программных ресурсов в ЯП или DSL средствами инструмента Eclipse-DSL, DSL Tool Microsoft, трансформация этих описаний к XML, формирование полученного описания к стандартному виду и запись в репозиторий ИТК ГП;

— поиск и отбор из репозитория необходимых КПИ;

— использование модели взаимодействия КПИ и ПС для адаптации в другой среде обработки;

— методика применения языка DSL для представления члена семейства;

— генерация проектных решений или артефактов к выходному коду и адаптация к конкретным целям ПС;

— оценка показателей качества с использованием соответствующих метрик программного продукта;

— сохранение результатов проектирования в репозитории компонентов для апробации и выбору КПИ пользователями;

— документирование программ и компонентов, входящих в состав СПС.

В данной методологии используется инженерия ПрО, в которой инженерная деятельность основывается на моделях проектирования (DDD, DSL, MDA, GDM и др.) СПС. В парадигме ГП основным аспектом производства программ является генерация, базирующаяся на представлении знаний о специфике ПрО и знаний, накопленных о методах, средствах и инструментах программной инженерии, которые необходимы для линий автоматизированного производства СПС из КПИ. В данной методологии КП члены СПС описываются ЯП и предметно-ориентированными языками DSL. Процесс генерации этих описаний рассматривается как последовательная их трансформация от одного исходного ЯП до промежуточного и так до получения готового продукта.

Процесс генерации компонентов начинается из поиска готовых КПИ согласно требованиям заказчика к их функциям и принятия решения о достаточности реализованных свойств КПИ для сборки (конфигурации) их в системы или СПС.

Реализация новых теоретических основ ГП в ИТК. Индустрия производства ПС на основе ТЛ разработана в [17, 41] для изготовления класса программ обработки данных в информационной системе АИС «Юпитер» для военно-морского флота СРСР. Вначале была проведена технологическая подготовка разработки линий, технологических маршрутов и готовых программных элементов, а также создано методическое и программное обеспечение, регламентирующее инженерную, экономическую и производственную деятельность на линии. В дальнейшем эта идея развивалась до создания фабрики программ, которые поддерживают дисциплины по производству ПП на линиях [24, 29, 30].

Комплекс ИТК ГП включает линии: разработки КПИ и их сервисного обслуживания в репозитории; представления описания КПИ и интерфейсов в стандартном виде (WSDL, IDL); сборки или конфигурации разноязычных КПИ в члены семейства или СПС; оценку качества КПИ и затрат на их разработку; преобразования GDT в FDT; использования веб-сервиса, обучения студентов теоретическим и прикладным аспектам производства ПП.

ИТК ГП построен как сайт http://sestudy.edu-ua.net в Интернете (рис. 5).

На главной странице сайта в левой части дан перечень функций ИТК, в центре — схемы их реализации с учетом научных и проектных решений методологии производства программ, включая обслуживание репозитория, методику реализации линий сборочного производства, взаимодействия программ, систем и технологии представления некоторых доменов средствами систем среды ГП и КПИ. Далее дается описание разделов и подразделов данного сайта.

- Технологии — фабрика программ КНУ, репозиторий КПИ, разработка КПИ, сборка КПИ, конфигурирование КПИ, генерация описания программ в языке DSL, онтология вычислительной геометрии, оценка затрат, веб-сервисы, генерация типов данных.
- Взаимодействие программ, систем и сред по моделям интероперабельности этих целевых объектов: Corba–Eclipse–Java; VS.Net C#–Eclipse; Basic–C++.
- Инструменты — Eclipse, Protégé и технология их использования.
- Презентации — прикладная система ведения зарубежных командировок, слайды докладов на международных конгрессах.
- Обучение — технологии разработки программ в языке C# VS.Net, Java и электронного учебника «Программная инженерия», выполненных на сайте КНУ http://programsfactory.univ.kiev.ua.

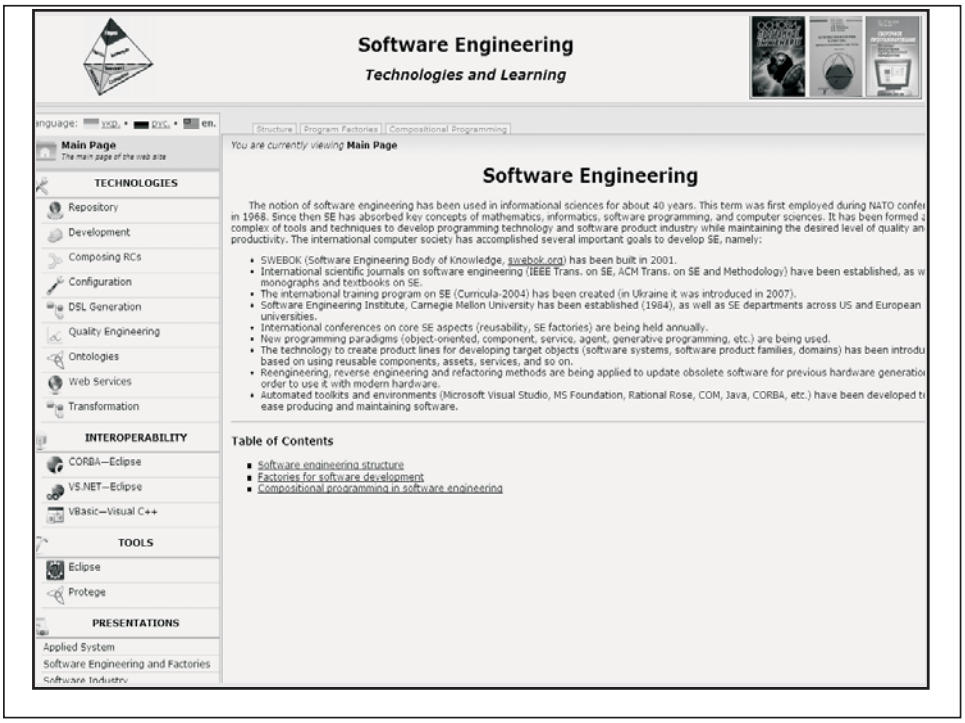

Рис. 5. Главная страница веб-сайта ИТК ГП

Все разделы (подразделы) сайта построены по одной рамочной схеме для нажатия в них текста, который демонстрирует:

— описание смысла технологии на конкретной линии;

— метод реализации на примерах программ;

— загрузку exe-кода для выполнения на рабочем столе;

— возврат в сайт.

Технологии и взаимодействие на сайте реализованы по одной схеме — описание, пример, выполнение, закрытие.

В реализации простых линий технологии производства программ в ИТК участвовали студенты КНУ имени Тараса Шевченко и МФТИ, особенно в части создания экспериментальной фабрики программ [27–30] и программных средств поддержки моделей взаимодействия программ и систем [11, 26–28] и вариабельности ПС [14, 15].

К линиям относятся:

— фабрика программ (http://programsfactory.univ.kiev.ua) — спецификация КПИ и их паспортов, программирование VS.Net и обучение программной инженерии;

— обслуживание репозитория компонентов, КПИ и артефактов;

— сборка разноязычных программ и компонентов в ПС с возможным конвертированием несовместимых типов данных интерфейсов;

— конфигурация КПИ в сложную структуру ПС по точкам вариантности [31, 32];

— описание домена (на примере домена ЖЦ) в языке DSL и реализация на платформе Eclipse-DSL;

— описание веб-сервисов;

— оценивание качества и затрат на разработку ПС и др.

Экспериментальная фабрика программ в КНУ базируются на объектном анализе ПрО и компонентном методе реализации объектов и КПИ с интерфейсами, необходимыми при сборки компонентов и взаимодействии КПИ в средах VS.Net, Java, CORBA, Eclipse. Данная студенческая фабрика программ ориенти-

рована на разработку студентами научных артефактов и программ при выполнении лабораторных и дипломных работ с помощью специальных линий продуктов, разработанных по методике ТЛ. На данный момент к фабрике обращались больше четырех тысяч студентов, преподавателей и научных сотрудников.

ЗАКЛЮЧЕНИЕ

В рамках проекта ГП получено:

— теорию объектно-компонентного программирования разнородных КПИ и их сборку в средах (VS.Net, IBM, CORBA, Eclipse и др.);

— теорию взаимодействия систем и сред, которая включает развитый аппарат интерфейсов и механизмы обработки данных, которые передаються по сети в общие и глобальные хранилища, для интероперабельности и миграции систем из одной среды в другую;

— теорию конструирування вариантов СПС, управления конфигурациями СПС, начиная с требований и кончая продуктом СПС, а также инженерию качества СПС в процессе создания систем семейства из готовых КПИ и оценки качественных показателей, стоимости и затрат на разработку;

— новую методологию производства СПС, представленной спектром простых технологических линий по изготовлению и обслуживанию КПИ, сборки или конфигурации разноязыковых КПИ, тестирования, оценки качества и затрат на разработку;

— веб-сайт (http://sestudy.edu-ua.net) для производства СПС из готовых КПИ в среде ГП (VS.Net, CORBA, Java, Eclipse) и обучение индустриальным методом.

Представленные в работе новые теории и методы расширяют технологию программирования и способствуют развитию индустрии ПС и СПС из готовых КПИ в современных операционных средах.

СПИСОК ЛИТЕРАТУРЫ

1. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение. — М.; СПб.; Харьков: Минск: Издательский дом «Питер», 2005. — 730 с.
2. Лавріщева К.М. Генерувальне програмування програмних систем і сімейств // Проблеми програмування. — 2009. — № 1. — С. 3–16.
3. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. — Киев: Наук. думка, 1991. — 213 с.
4. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов: Второе изд. — Киев: Наук. думка, 2009. — 371 с.
5. Лаврищева Е.М. Сборочное программирование. Теория и практика // Кибернетика и системный анализ. — 2009. — № 6. — С. 3–12.
6. Грищенко В.Н., Лаврищева Е.М. Методы и средства компонентного программирования // Там же. — 2003. — № 1. — С. 39–55.
7. Грищенко В.М. Метод об'єктно-компонентного проектування програмних систем // Проблеми програмування. — 2007. — № 2. — С. 113–125.
8. Гринфильд Дж. Фабрики разработки програм. — М.; СПб.; К.: Изд. дом «Вильямс», 2007. — 591 с.
9. Бей И. Взаимодействие разноязыковых программ. — М.; СПб.; К.: Изд. дом «Вильямс», 2005. — 868 с.
10. Лаврішева К.М. Взаємодія програм, систем й операційних середовищ // Проблеми програмування. — 2011. — № 3. — С. 13–24.
11. Теоретичні аспекти керування варіабельністю в сімействах програмних систем / К.М. Лавріщева, О.О. Слабоспицька, Г.І. Коваль, А.О. Колесник // Вісн. КНУ. Сер. фіз.-мат.наук. — 2011. — № 1. — С. 151–158.

12. Основы инженерии качества программных систем / Ф.И. Андон, Г.И. Коваль, Т.М. Коротун, Е.М. Лаврищева, В.Ю. Суслов: 2-е изд. — Киев: Академпериодика, 2007. — 672 с.

13. Коваль Г.І. Підхід до моделювання якості сімейств програмних систем // Проблеми програмування. — 2009. — С. 49–58.

14. Удосконалення процесу розроблення сімейств програмних систем елементами гнучких методологій / Г.І. Коваль, А.Л. Колесник, К.М. Лавріщева, О.О. Слабоспицька // Там же. — 2010. — № 23. — С. 261–270.

15. Нові теоретичні засади технології виробництва сімейств програмних систем у контексті ГП / К.М. Лавріщева, Г.І. Коваль, Л.П. Бабенко, О.О. Слабоспицька, П.П. Ігнатенко. — ДРНТІ. № 67УК2011 від 5.10.11. — 377 с.

16. Лавріщева К.М. Становлення і розвиток модульно-компонентного програмування // Стан та перспективи розвитку інформатики в Україні (Розд. 4.4). — Київ: Наук. думка, 2010. — 1006 с.

17. Лаврищева Е.М. Основы технологической подготовки разработки прикладных программ СОД. — Киев: Ин-т кибернетики, 1987. — 30 с.

18. Ларищева Е.М. Методы программирования. Теория, инженерия, практика. — Киев: Наук. думка, 2006. — 451с.

19. Лаврищева Е.М., Петрухин В.А. Методы и средства инженерии программного обеспечения. — Уч. пособие. — М.: МОН Российской федерации, 2007. — 415 с.

20. Лавріщева К.М. Програмна інженерія. — Київ: Академперіодика, 2008. — 319 с.

21. Лавріщева К.М. Інструментально-технологічний комплекс для виробництва програмних систем // Вісн. НАН України. — 2012. — № 1. — С. 15–41

22. Лаврищева Е.М. Интерфейс в программировании // Проблеми програмування. — 2007. — № 2. — С. 126–139.

23. Лаврищева Е.М. Проблемы интероперабельности разнородних объектов, компонентов и систем. Подходы до ее решения // Спец. выпуск междунар. конф. «УКРПрО – 2010», Проблеми програмування. — 2010. — № 2–3. — С. 28–41.

24. Андон П.И., Лаврищева К.М. Розвиток фабрик програм в інформаційному світі // Вісн. НАН України. — 2010. — № 10. — С. 15–41

25. Лаврищева Е.М. Теория и практика фабрик программных продуктов // Кибернетика и системный анализ. — 2011. — № 6. — С. 145–158.

26. Лаврищева Е.М. Концепція індустрії наукового софтвера і підхід до обчислення наукових задач // Проблеми програмування. — 2011. — № 1. — С. 3–17.

27. Островський А.И. Подход к обеспечению взаимодействия программных сред JAVA и Ms.Net. // Там же. — 2011. — № 2. — С. 37–44.

28. Радецький І.О. Один з підходів до забезпечення взаємодії середовищ MS.Net і Eclipse // Там же. — 2011. — № 2. — С. 45–52.

29. Анісімов А.В., Лавріщева К.М., Шевченко В.П. Про індустрію наукового софтвера // Conf. Theoretical and Appl. Aspects of Cybernetics. — Kiev, 2011. — P. 7–9.

30. Аронов А.О., Дзюбенко А.І. Підхід до створення студентської фабрики программ // Проблеми програмування. — 2011. — № 3. — С. 42–49.

31. Колесник А. Підхід до конфігурування компонентів повторного використання // Там же. — 2011. — № 4. — С. 57–66.

32. Колесник А.Л. Механізми забезпечення варіабельності в сімействах програмних систем // Там же. — 2010. — № 1. — С. 35–44.

Поступила 10.01.2012