КОЛЛЕКЦИЯ CASE-TOOLS ДЛЯ СБОРКИ ВАРИАБЕЛЬНЫХ СИСТЕМ ИЗ ГОТОВЫХ ПРОГРАММНЫХ РЕСУРСОВ

[1]Е.М.Лаврищева, ИСП РАН
[2]О.А.Слабоспитская, ИПС НАНУ

1. ВВЕДЕНИЕ

Технология разработки ПО начала развиваться одновременно с появлением ЭВМ и первая ее стадия – это ручное программирование общесистемного обеспечения для функционирования первых ЭВМ. Потом появился термин “Software Engineering” прозвучал на конференции НАТО (1968), который обозначал инженерные аспекты изготовления вычислительной продукции (ЭВМ и программного обеспечения – ПО) высокого качества (www.sei.com). Постепенно на практике формировались процессы разработки ПО, которые упорядочивали подходы к программированию как отдельных элементов ПО, так и математического обеспечения для решения научно-технических задач на ЭВМ.

В СССР создавались ЭВМ и ПО для них. Сформировалась технология программирования (ТП) прежде всего трансляторов с простых языков (типа адресного), языков программирования Алгол-60, Фортран, Кобол и др. для описания математических и технических задач, а также ОС для управления процессом вычислений этих задач. Кабинет Министров СССР принял постановление (1978) о фондах программ со статусом программной продукции производственно-технического назначения. ГКНТ СССР финансировал разработку средств автоматизации технологии создания сложных программ (ПРИЗ, АЛЬФА, АПРОП, ПРОЕКТ, ПРОТВА, ПРОМЕТЕЙ, и др.) [1].

В 2001г. международный комитет IEEE и ACM определил ядро знаний SWEBOK (Software Engineering body of Knowledge, www.swebok.com, 2001г.) программной инженерии. В нем дано такое определение – *это система методов, средств и дисциплин планирования, разработки, эксплуатации и сопровождения ПО, готового к внедрению*. Ядро включало в себя 10 разделов знаний (area knowledge). Первые пять разделов – это инженерия требований, проектирование, конструирование, тестирование и сопровождение ПО. Следующие пять разделов – это организационные процессы (управление проектом, конфигурацией, качеством, методы и средства инженерии ПО) [2]. Дальнейшее развитие SWEBOK – новые дисциплины SE [3–5], которые регламентируют инженерную, экономическую, управленческую и производственную деятельность процесса разработки программного продукта (ПП).

За рубежом создавались CASE-средства – SADT (Structured Analysis and Design Technique), SSADM (Structured Systems Analysis and Design Method), IDEF0, IDEF1, IDEF2 (Integrated Definition Functions), продуктовые линии (Software Product Line Engineering), самостоятельные

фабрики программ (Дж.Гринфильда, Г.Ленца, М.Фаулера и др.) и системные AppFab на платформах (IBM, VS.Net, Intel, CORBA, Java, Intel и др.) для сборки готовых элементов, шаблонов, кодов в системе автоматизации предприятий, бизнес приложений и др. [3].

В рамках международного комитета ISO/IEC созданы стандарты, регламентирующие процессы разработки качественного ПО, жизненный цикл (ЖЦ), качество, метрики и др. Процессы стандарта ISO/IEC 12207 ЖЦ ПО (1996, 2007) соответствуют областям знаний SWEBOK. Технология использования всех процессов ЖЦ ISO/IEC 12207 ЖЦ - 2007 для проектирования и разработки программ, комплексов программ и систем традиционными методами представлена в монографиях и учебных пособиях В.В.Липаева [например, 6, 7] и др.

В виду накопления в мировом информационном пространстве огромного количества готовых программ и сервисов (более 100мил.), как многоразовых объектов, КПИ и reuses главным принципом разработки больших и сложных программных систем (ПС) и информационных систем (ИС) стала **сборка** и/или **интеграция** (Assembling, Build, Continious Integration и др.). Для сборки многоразовых объектов – модулей, компонентов, объектов, сервисов, reuses assets и др., описываются паспорта в языке WSDL и их интерфейсы в языках (IDL, API, SIDL и др.).

Проектирование систем традиционными методами и SADT, SSADM и др. усложняли системы, что привело к кризису сложности и трудностям внесения в них изменений, так как они разрабатывались не модульным принципом. Технология сборки сложных систем из готовых КПИ и их интерфейсов проводилась в рамках модульном программировании приложений и систем. КПИ, как многоразовые reuses, упрощает процесс внесения изменений в сложные системы [8–10].

Сборочный принцип разработки систем и их семейств, присущий всем современным технологиям, обеспечивает комбинирование КПИ в конфигурацию новых вариантов приложений и систем.

2. АНАЛИЗ СОВРЕМЕННЫХ МЕТОДОВ РАЗРАБОТКИ СИСТЕМ

Начиная с 1990 годов, активно развивался объектно-ориентированный подход (ООП). Появились новые объектно-ориентированные ЯП (C++, VBasic, Java и др.) и CASE-средства (Rational Rose, UML, CORBA, COBRA и др.), системы поддержки процессов разработки объектных программ и оценки их качества. Они стали альтернативой традиционного программирования монолитных систем и способствовали представлению программных объектов в виде самостоятельных многоразовых продуктов со свойством изменяемости. Консорциум CORBA реализовал для ООП объектную модель, брокер объектных запросов через интерфейс к собранным объектам на сервере. Объекты описываются в ЯП этой системы. Rational Rose и UML предоставили пользователю богатый набор диаграмм (use case)

проектирования систем из объектов и компонентов и инструменты их реализации, тестирования и управления качеством продуктов. Эти системы можно считать системными фабриками программ, создаваемых из разнородных КПИ [11, 12].

С 2000 года в мировой индустрии сформировалась технология Software Product Line Engineering. Ее основу составляет стандартная модель архитектуры системы и модель вариабельности, задаваемая характеристиками (Feature Model) приложений и систем и точками изменяемости артефактов приложений и семейств. Эта модель реализует свойство вариабельности готового программного продукта (ПП), обеспечивает автоматизированную обработку функций системы и возникающих нерегулярных ситуаций, выдавая сведения для внесения изменений в терминах значений характеристик и вариантных точек [13–18].

Цель работы рассмотреть современные технологии изготовления ПП – приложений и систем из КПИ и определить сборочную технологию программирования теоретически и практически для использования на любых фабриках программ.

2.1 Продукты SPLE (Software Product Line Engineering)

Согласно «Systems and Software Engineering Vocabulary» ISO/IEC FDIS 24765:2009(E) **Product Line** – это "группа продуктов или услуг, которые имеют общее управляемое множество свойств, удовлетворяющие потребностям определенного сегменту рынка или вида деятельности". В технологии SPLE главную роль выполняет процесс управления вариабельностью отдельных артефактов (компонентов, reuses, assets и др.) систем и их семейств с учетом требований. Основные процессы создания программного продукта (ПП) SPLE – domain engineering and application engineering, которые отработанны практически при производстве коммерческих продуктов [13].

Процесс инженерии домена состоит в определении и реализации общих свойств (the commonality) обеспечения вариабельности и изготовления новых *вариантов* ПП. *Процесс инженерии приложений* состоит в определении специфики отдельных приложений домена и изменяемости артефактов.

На Product Line выполняется процесс инженерии домена, который называется разработкой «для обеспечения повторного использования» (for reuse), и процесс инженерии приложений, как разработку ПС «с использованием КПИ» (with reuse). Модель Product Line отражает оба процесса разработки и сборки КПИ в семейство ПП. Эти процессы повторяются при разработки новых КПИ и ПС с учетом требований.

В инженерию *домена* и приложений для обеспечения изменяемости артефактов (requirements, architecture, components, test cases и др.) и архитектуры ПП добавляется модель вариабельности домена (Domain

Variability Model) – набор функций, которые отмечаются вариантными точками и констрейнами для внесения по ним изменений. *Модель вариабельности* приложения (Application Variability Model) используется для получения разных вариантов членов семейства ПП. *Точка вариантности* – это место в артефакте ПП, определяющее выбор варианта артефакта и ПП.

Вариабельность – это способность семейства ПС или отдельного приложения и их артефактов к расширению, изменению, приспособлению или конфигурированию с целью использования в конкретном домене.

Процесс инженерии домена в SPLE

Технология разработки продукта для домена включает процессы обработки требований, проектирования архитектуры, реализации домена и тестирования домена.

Инженерия требований домена определяет общие свойства функциональности, качества ПП и документ на требования и техническое задание на разработку домена.

Проектирование домена заключается в создании стандартной архитектуры по модели требований домена. Процесс ориентирован на механизмы конфигурации архитектуры для включения в нее спецификаций приложения и характеристик изменчивости. Вариабельность архитектуры определяется областью реализации и точками вариантов артефактов.

Реализация домена – это детальное проектирование проектных решений (assets) и artifacts, которые разрабатываются для этого домена. Обеспечивается configurable артефактов приложения путем задания вариантных контекстов для конфигурации.

Тестирование домена – это процесс проверки артефактов и КПИ по заданным требованиям с помощью тестов испытаний в вариантных точках артефактов продукта. Составляется стратегический план испытаний artifacts и домена в целом. Основу плана составляют наборы тестов, тестовых случаев и сценариев их выполнения.

Процесс изготовления приложений в SPLE

Технология разработки приложения включает аналогичные процессы: обработка требований приложения, проектирование приложения, реализации приложения, тестирование приложения домена. Эти процессы идентичны процессам домена. Они базируются на характеристиках приложений и домена. Для обеспечения **адаптивности и изменчивости** отдельных артефактов домена используется FM (Model Feature) характеристик артефактов, вариантных точек приложений и домена. Модель вариабельности приложения - основа процесса управления конфигурацией (configuration management) и создания продукта из готовых

артефактов и reuses на процессе конфигурации (product configuration) [19, 20].

Для поддержки доменной инженерии разработана библиотека **DEMRAL.** Она содержит: основные понятия Абстрактных Типов Данных (АТД) и алгоритмы, которые оперируют с АТД; свойства контейнера, как в АТД; основную математическую теорию реализации задач; разновидности алгоритмов работы с данными. Библиотека способствует моделированию домена – выявление сущностей, их идентификация, определение диаграмм характеристик и алгоритмов задач в среде OS Linux.

2.2. Типовые AppFab (фабрики программ) приложений на современных операционных платформах

В каждой операционной системе массового применения созданы фабрики программ.

В библиотеках системы содержатся наборы документов, заготовок, КПИ, наборы данных и др. Они используются для сборки и интеграции из них приложений и систем для разных целей, включая управление предприятиями, большими данными и др.[2]. К фабрикам системного типа относятся:

1) Фабрика ПО (Fabrics Software) для систем и сервисов;
2) AppFab в системе коллективной разработки VS.Net;
3) AppFab в системе Grid Европейского проекта;
4) AppFab IBM для создания доменов предприятий и бизнес-приложений;
5) AppFab в системе CORBA для сборки разнородных программных ресурсов;
6) AppFab в системе INTEL и др.

2.2.1 Фабрики ПО

Эта фабрика ПО разработана группой **p&p.** Она включает целую коллекцию разного рода ресурсов, блоков кода, документации, образцов приложений и т.п. [21]. В состав фабрики вошли схемы и стандарты, рекомендации, инструменты, шаблоны решений, упрощающие запуск новых приложений. Архитекторы модифицируют требуемые коды сервисных ресурсов модели SOA Visual Studio Industry Partners (VSIP), а затем повторно устанавливает их на фабрике ПО. Используется модели Guidance Automation eXtensions (GAX) и Guidance Automation Toolkit (GAT) в Visual Studio. GAX – это среда исполнения в VSIP с использованием пакетов рекомендаций. GAT – это набор инструментов реализации пакетов рекомендаций в GAX Visual Studio, которые имеются на любой другой фабрике, в том числе в IBM. Основу фабрики составляет **фабрики служб:** веб-служба ASP.NET (ASMX) для Windows Communication Foundation (WCF) в среде .NET Framework 3.0. Версии для

веб-службы WCF находится в сообществе разработчиков фабрики ПО и GotDotN. В ее состав входят процессы предприятия, договора и пакет документаций и рекомендаций для приложения-образца Global Bank агентами.

В версию ASMX входят два пакета рекомендаций – один для решения задач ASMX и другой для задач доступа к данным. В версию WCF также входит два пакета рекомендаций – один для задач WCF и один для задач, связанных с безопасностью. Диспетчер пакетов рекомендаций обеспечивает запуск из меню инструментов Visual Studio 2007, а также включение и отключение пакетов рекомендаций. Это аналогично модели SCA в IBM WebSphere. Взаимодействие со службами выполняется в соответствии с договором на обслуживание, в котором определены операции сервиса и служб. Для каждой операции определяется тип сообщения и тип данных. Адаптер реализует договор обслуживания и его адаптацию к уровню предприятия, отражающего его объекты, логику деятельности объектов предприятия и его рабочие процессы.

2.2.2 Системные фабрики интеграции разнородных компонентов и данных

К ним относятся: IBM WebSphere, Microsoft Biz Talk 2004, BEA WebLogic Oracle 10g, SAP NetWeaver, ИВК "Юпитер» (см. табл. 1.) [20-24].

Таблица 1. **CASE–системы интеграции программных ресурсов**

Платформа	**Разработчик**	**Содержание платформы**
IBM WebSphere	Корпорация IBM	Сервер приложений J2EE, брокеры обмена данными, КПИ, портал, workflow/BPM, средства EII, SCA
Microsoft Biz Talk 2004 и компоненты .Net	Корпорация "Майкрософт"	Сервер приложений COM, брокеры обмена данными, RGB доставки, портал, workflow/BPMN
BEA WebLogic	Корпорация "BEA Systems" (в 2008 г. присоединенная к корпорации "Oracle")	Сервер приложений J2EE, брокеры обмена данными, ГОР, сервер прикладных приложений, портал, workflow/BPMN
Oracle 10g	Корпорация "Oracle" http://www.oracle.com	Сервер приложений J2EE, брокеры обмена данными, ГОР, портал, workflow/BPM, средства EII
SAP NetWeaver	Корпорация SAP http://www1.sap.co	Сервер приложений J2EE/ABAP, брокеры обмена данными, портал,

	m/ www.sap.ru	инструменты BPMN
ИВК "Юпитер	Компания ИВК (Россия) http://www.ivk.ru/	Брокеры обмена данными, КПИ, среда выполнения, сертификация, защита данных

CASE корпорации IBM WebSphere [24]. Эта платформа позволяет работать на основе веб-технологий. Ядро WebSphere включает в себя WebSphere Application Server (WAS) и открытые стандарты J2EE, XML и веб-сервисы. Это многофункциональный набор (коллекцию) инструментов *интеграции приложений* в рамках предприятия (EAI) на уровнях: данных, обмена сообщениями, бизнес-процессов, B2B business to business; сервисных компонентов; бизнес-логики программ на JAVA [21]. В него входят:

1) сервер обработки очередей сообщений MOM (Message Oriented Middleware), операций интеграции Business Integration Interchange Server (ICS) и MQ Business Integration Message Broker (WSMB);

2) сервер приложений WAS;

3) Portal Server для поддержки функционирования в WAS;

4) система проектирования Workflow, совместимая с WSMB;

5) Сервисно-компонентная архитектура (SCA) для создания бизнес приложений.

В состав WebSphere входит Business Integration Workbench для проектирования бизнес-процессов и управления ими. Продуктами платформы являются образцы в классах, брокер сообщений WSMB, WAS и встроенные возможности для задания бизнес-правил и сценариев workflow, а также сборки компонентов Enterprise JAVA beans (EJB). Сервер WAS связывает ресурсы веб-сервисов в единый процесс и включает следующие средства:

– WebSphere Business Integration *for Automotive* для поддержки автоматизированного создания сервисно-ориентированных приложений деловых процессов;

– WebSphere Business Integration *for Financial Networks* обеспечивает консолидацию разнородных сетей обмена сообщениями в финансовой деятельности. Поддерживает интеграцию и оптимизацию операций проектирования, производства и выпуска новых продуктов и поддержку "унаследованных" систем и разнородных приложений;

– WebSphere Business Integration *for Energy* and Utilities обеспечивает оптимальную интеграцию процессов эксплуатации, управления активами и их обслуживания;

– WebSphere Business Integration Express *for Item Synchronization* обеспечивает связь информации из цепочек услуг для небольших компаний.

Модель SCA – это сервисно-компонентная модель для определения бизнес-сервисов, создания и настройки бизнес-решений через модель интерфейсов в языке WSDL, артефактов процессов в языке BPMN и реализацию продукта в Java ™.

Сервисный модуль WebSphere Process Server (эквивалентен EAR J2EE) объединяет модули SCA и SDO. Последний обеспечивает обмен данными между сервисными компонентами через объект данных. В нем содержатся ссылки к метаданным и информация о передаваемых данных, которые описываются в Java интрефейсе – commonj.sdo.DataObject и обрабатываются инструментами JMS, Enterprise JavaBeans или Web сервисы.

Таким образом, IBM WebSphere предоставляет набор средств по созданию приложений из компонентов, разного рода сервисов и сборки бизнес приложений.

CASE Microsoft.NET Framework [26] Эта платформа предоставляет разработчику функциональность в J2EE и в среде ОС Windows. Инструменты, необходимые для реализации разных интеграционных подходов, сгруппированы в виде нескольких продуктов, а отдельные функции возложены на ОС (например, управление транзакциями MTS, веб-сервер Internet Information Server, библиотеки и среда "управляемого кода" .Net) [22].

Основная функциональность BizTalk Server 2004 – сервер интеграции на базе XML, как брокер сообщений, осуществляет преобразование данных и коммутацию сообщений. Сервер приложений выполняет бизнес-логику – низкого (компоненты EJB) и высокого (через механизмы workflow) уровней. Этот сервер поддерживает высокоуровневую бизнес-логику и интеграцию систем, выполнение логики низкого уровня реализуется моделью COM+ или .Net. Модель Microsoft позволяет размежевать работу программиста и аналитика бизнес-процессов. Бизнес-аналитик может графически "рисовать" бизнес-процесс, задавая схемы обмена документами и передачи управления через workflow. Для интеграции определяются точки вызова внешней функции через COM-объекты, Веб-сервисы и т. п**.**

CASE WebLogic Integration [23]**.** Это инструмент создает бизнес-логики на языке JAVA и интегрирует приложения с обеспечением взаимодействия с бизнес-партнерами (B2B). Платформа включает в себя пять основных компонентов: виртуальная машина JAVA, сервер приложений, средство построения порталов, пакет инструментов интеграции, среда разработки. Ключевое преимущество платформы – возможность снижения требований к группе разработки за счет использования трехуровневого подхода к созданию приложений, подобно подходу корпорации Microsoft. Программы создаются на языках JAVA, Visual Basic или COBOL. Платформа BEA основывается на новейших стандартах XML (XSLT, XQuery и т. п.), веб-сервисах и средствах

доставки, совместимых со стандартом JMS и брокером сообщений. WebLogic имеет набор интерфейсов для интеграции приложений, файлов и баз данных разной природы. В нее входят набор готовых конвекторов, системы документооборота, анализаторы форматов файлов и средства обращения ко всем модулям программ Windows и JAVA для взаимодействия с другими интеграционными платформами.

CASE Oracle Integration [22]. Данная платформа предоставляет полный набор средств корпорации "Oracle" для интеграции приложений из разнородных приложений [24]. Платформа соединяет технологии нескольких классов со стилями интеграции: данных (технология Transparent Gateways и конвекторы базы данных), интерфейса пользователя, сервера и системы MOM. В Oracle Application реализованы новейшие возможности COA и веб-сервисов управления их координацией и композицией приложений для любых сред разработки приложений. В ней выполняется конструктивное развитие двух современных парадигм конструирования приложений с корпоративными вычислениями и использованием сервисов (Service-Oriented Computing) и сетевых вычислений (Grid Computing). Предлагается аспектная инфраструктура реализации SOA и объединения (federate) приложений с доступом к ним и к сервисам. Сетевые вычисления предусматривают использование собираемых из модулей серверов и блоков памяти, которые имеют низкую стоимость при эксплуатации приложений поддержки деловых процессов вида;

1) Business Intelligence для анализа бизнес-информации предприятия, построения ПС путем сборки, анализа и распределения разноуровневой информации;

2) Business Integration для интеграции разнородных приложений, включая объединение отдельных подсистем и автоматизацию деловых процессов;

3) Identity Management - управление идентификацией личности, что позволяет администрировать средствами защиты и снижать стоимость владения точками уязвимости.

Для конструирования приложений используется пакет – Oracle Integration Interconnect. В нем имеются средства композиции и координации сервисов (служб) предприятия и ESB- развертывание интеграционных приложений на предприятии. Для интеграции связей используются Repository метаданных SAP и Directory. Особенность платформы SAP – это интеграция не только на уровне конектора к шине обмена данными, но и на высших уровнях, совместимых с системой управления контентом, и порталом среды, включающей Eclipse, IBM WebSphere, SAP Web Application Server через SAP JAVA Connector и SAP .NET Connector.

Платформа ИВК "Юпитер" [23]. Этот продукт обеспечивает функции интеграции на уровне данных и обмена сообщениями для

потребностей государственных структур, которые выдвигают повышенные требования к защите информации и интеграции унаследованных и устаревших ПС. Эта платформа соединяет характеристики виртуальной машины, транспортной магистрали, отдельные свойства систем документооборота и средства защиты информации. Она состоит из двух высокоуровневых логических блоков:

1) обеспечения стандартной функциональности на отдельном компьютере;

2) определения связи между разными компьютерами.

На каждом компьютере представлена унифицированная модель вычислительного процесса, которая соединяет среду выполнения ИВК "Юпитер" и набор библиотек для создания приложений, которые могут выполняться в средах типа Cloud Computing через модуль API ИВК "Юпитер". Продукт обеспечивает контроль целостности вычислительного процесса в начале загрузки и при выполнении. Имеется встроенная возможность эмуляции IP поверх многих унаследованных транспортных протоколов с гарантированной доставкой сообщений в гетерогенной сети и реализованы возможности документооборота.

Таким образом, из приведенного описания современных платформ индустрии приложений следует, что разработаны типичные фабрики в среде общих систем с базовыми средствами создания, компоновки и выбора специальных КПИ и сервисов, а также сборки КПИ и сервисов в новые приложения для функционировать в любой среде, в том числе в Grid и Cloud Computing.

2.3. Современные фабрики программ

Фабрика программ – это интегрированная инфраструктура сборочного производства из готовых КПИ новых ПП (систем, семейств систем, ИС, АСУ, АСУТП и др.).

Фабрика включает в себя комплекс системных средств, инструментов и сервисов, а также репозитории и библиотеки программ для накопления и выбора готовых КПИ. Ядро фабрики – операционная среда и методы изготовления (UML, компонентный, структурный, модульный, сервисный и др.) отдельных КПИ. В среду фабрики вводится набор веб-сервисов и веб-семантики для управления выбором необходимых сервисов из Интернета и их использования при сборки и композировании новых ПП.

К фабрикам программ отнесены:

1) система АПРОП (ИК) [25];

2) система Sun Microsystems (IBM) [26];

3) OMA–архитектура или система CORBA (OMG) [11];

4) фабрика "ручной" сборки разноязычных программ Инга Бейя [27];

5) фабрики программ по методу UML Дж. Гринфильда [28];

6) среда для групповой разработки ПП – MS.VSTS [26];

7) инфраструктура вычислений Grid [29];

8) фабрика Г. Ленца на основе Use Case [30];
9) каркас фабрики программ Авдошина [31];
10) Фабрика «continiuos Integration» Фаулера [32];
11) Фабрика программ ЕПАМ [33] и др.

Системы АПРОП, IBM, CORBA проложили первый путь к созданию современных фабрик программ. Рассмотрим их положения.

АПРОП – это фабрика, которая работала в среде ОС ЕС (1980) и обеспечивала сборку разноязычных модулей в монолитную структуру на ЕС ЭВМ через интерфейсных 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 (4GPL) модулях и реализуют методы численного анализа и статистики, которые расположены в специальном банке модулей и библиотеки из 64 функций преобразования отличающихся типов данных [26] .

IBM – среда сборки 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, веb-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, на сервере (WebSphere Application Server Compunity Edition).

CORBA или OMA-архитектура (Apple, IBM, Win-NT, x-Open, Dec) обеспечивает связи разноязычных объектов в ЯП (C, C++, Smalltalk, JAVA, Cobol, ADA-96 и др.) через интерфейсные посредники (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 через модули посредники (stub, skeleton) в языке IDL или протоколами TCP/IP, IIOP через брокер ORB, который обеспечивает их разным сервисом, в том числе по преобразованию несовместимых типов данных разноязычных объектов, устранения 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.

Фабрики программ на платформе Microsoft. Среда MS.NET содержит большое количество средств и инструментов: готовые ресурсы (компоненты, сервисы) Интернета, языки – JAVA, C++, Basic, JAVA, Pascal, C#, библиотеки CLR и FCL, сборка кодов (exe, dill) в ПП, веб-обслуживание, управление проектом PM–2007 в виртуальной среде VSTS и MSF. Среда предназначена для коллективной разработки систем. В ее

состав входят: пакет инструментов VSTS–2007 (Visual Studio Teams Systems); MSF (Microsoft Solution Arhitecture) для построения производственной архитектуры предприятия; модели процессов и систем; Professional Studio и Foundation Server для поддержки процессов проектирования, кодирования, тестирования и формирования версий ПП (SDLC, IDE, MS Office, MS SQL server, MS Visual Studio 2007); модель усовершенствования процессов (CMMI Process Impovement), процесс сборки с использованием CLR-библиотеки (Common Language Routine), FCL-типы, трансляторы, General code (exe), Portable Executable code и т. п. В среду обработки находятся средства определения сроков работ, трудозатрат, оценки показателей качества готовых ПП и др.

Visual Studio Teams Systems – это семейство продуктов для коллективной разработки ПС, их взаимодействия, а также для отбора, распределения и выполнения работ в программном проекте. Команда исполнителей проекта разделены на четыре категории с учетом их уровня знаний и навыков программирования систем (рис.1). Специалисты четвертой группы - это квалифицированные и ответственные за функциональную правильность разработки ПС. К средствам поддержки процесса разработки элементов ПС относятся IDE (Integrated Development Environment).

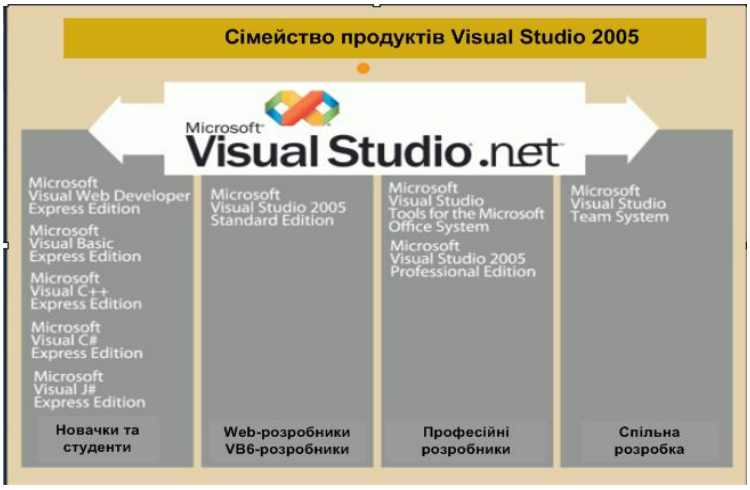


Рис.1 – Категории разработчиков проекта ПП

В нем проводится проверка требований, отслеживание процессов и результатов разработки с оцениванием степени их готовности и выполнения в Microsoft Office Project и Microsoft Office Excel. Тестирование элементов ПС осуществляется специальными средствами, которые интегрируются с Visual Studio.

В среде VSTS имеются такие средства: *Microsoft SQL Server 2007, Microsoft Project 2007* для руководства проектами и *Microsoft Excel 2007.*

Таким образом, среда VSTS предназначена для коллективной разработки систем и разных ПП посредством инструментов управления процессами ЖЦ на проекте и графиком работ, а также для отслеживания и

оценивания результатов на качество, стоимость и сроки выполнения проекта.

Средства и инструменты методологии MSF [18]. Эта методология – система стратегий, принципов и управления проектом построения производственной архитектуры предприятия с учетом: объемов работ в проекте, времени и стоимости, количества персонала, коммуникаций, закупок и контрактов, рисков. В ней разработана модель архитектуры предприятия с такими аспектами: *области приложения*, *бизнес, информация, технология*. Для разработки ПП создается скоординированный технологический план, который отвечает приоритету архитектуры и получению максимального эффекта при минимуме расходов.

Метод MSF базируется на анализе и разработке требований к ПС, проектировании проектных решений, которые учитывают базовые концепции предприятия и приоритетность архитектуры. Этот метод содержит в себе набор моделей: производственная архитектура; проектная группа; процесс разработки ПС; управление рисками; процесс проектирования применения. *Модель производственной архитектуры* – это набор принципов для создания версии производственной архитектуры предприятия согласно четырех перспектив. Основная задача этой модели – приспособление производственной архитектуры к бизнес-целям организации для поэтапного выпуска последовательных версий, отмеченных приоритетами для получения целевой производственной архитектуры. Члены проектной группы на процессе разработки создают: код системы, конфигурацию, функциональную спецификацию и сценарии тестирования. *Модель процесса* проектирования определяет цель и задачи производственной архитектуры с помощью концептуальной, логической и физической фаз, т.е. выполняется систематический переход от абстрактных концепций к конкретным техническим решениям. *Модель применения* – это трехуровневая структура, создаваемая сценарным методом разработки системы.

Таким образом, методология MSF ориентирована на проектирование программного и информационного обеспечения предприятия с помощью рассмотренных принципов, моделей и методов решения задач предприятия.

Фабрика сборки И. Бея. Базис этой фабрики – разноязычные программы, интерфейсные посредники, конфигурационные файлы. Программы описываются в ЯП (VC++, VBasic, Matlab, JAVA, Visual Works Smalltalk и т. п.) и выполнялись на платформах (MS.Net, HP, Apple, IBM и др.). В интерфейсном посреднике данные передаются данные друг другу. Для некоторых данных, типы которых нерелевантные или и их форматы зависят от другой платформы или от механизмов передачи данных (протоколы, вызовы, интерфейсные карты MIO–16E–2 и др.) проводилось ручное их преобразование. Описание интерфейсных связей проводилось с

помощью RMI, RPC, Java Native Interface. Затем они реализовывались в виде exe-файлов. Принцип взаимодействия разноязычных программ апробирован на Domain, Application Models, Model Interconnection, Microsoft Communication Foundation. Для внесения изменений в типы данных использовались современные визуальные средства (панели, сценарии, иконки и т. п.).

Фабрика программ Дж. Гринфильда. Для любой строящейся фабрики сформулированы методологические и технологические аспекты производства ПП методом 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 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 для реализации ПП на линиях производства; *веб-сервисы,* процессы линий; методики измерения и контроля качества ПП и т.п. Анализ системы моделирования UML для применения на фабрике показал, что UML удобен для задания эскиза ПП, в который используются компоненты и интерфейсы. Проблема модификации ПП не решена в визуальном языке UML и др. Фактически автором разработан меморандум современной фабрики ПП, которая базируется на reuses, линиях, моделях систем, каркасах процессов и продуктов.

Фабрики для вычислений в Grid [29]. 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. Она предназначена 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 или перекомпилированных программ в двоичном представлении обеспечивается репозиторием при выборе компонентов системы в альтернативной сетевой среде Grid. Паспорта КПИ представляются в стандарте WSDL (табл. 1) и сохраняются в репозитории системы.

Таблица 1. **Формат стандарта описания паспорта КПИ**

Дата создания	Дата создания КПИ автором (дата аттестации специфицированной версии продукта)
Дата изменения	Дата внесения изменений в КПИ
Версия	Версия КПИ
Платформа	Платформа, для которой создавался КПИ и на которой проверена его работоспособность

Операционная система	Операционная система, для которой создавался КПИ и на которой проверена его работоспособность
Размер	Общий размер КПИ (продукта, документации, и т.п.)
Описание	Короткое описание КПИ, список внесенных изменений, системные требования, требования к пользователям, список необходимых программ для корректной работы, справка, и т.п.
Правила использования	Описание операций выполнения КПИ, правил эксплуатации и т.п.

Типы данных Grid – проект, подсистема и компонент реализуется с использованием формата CIM для связи между разными компонентами с помощью системы MySQL и паспортных аннотаций КПИ, которые включают характеристики (имя, лицензия, глобальный идентификатор – ID (GUID), URL и т.д,).

Компоненты компилируются и собираются в конфигурационном файле ПП. Сервисы – главные ресурсы инфраструктуры вычислений. Они обеспечивают процесс вычислений научных задач глобального масштаба. Ресурсы задаются в протоколе для передачи данных по распределенной сети. Сервисы предоставляются стандартным протоколом, интерфейсом API и инструментом SDK (Software Development Kits). Сборка программ, компонентов, подсистем и систем научного назначения реализуется протоколом (Global Protocol). Подсистема ETICS содержит средства разработки, тестирования пакетов и услуг, а также сборку и конфигурирование программных элементов на основе механизмов плагинов [46, 50] и открытых интерфейсов для обслуживания потребителей или поставщиков.

Фабрика ПО Ленца [30]. Ключевые элементы этой Software Factory – схемы и шаблоны. Схема Software Factory – это описание того, как реализовать продукты, которые могут быть произведены на этой фабрики. Технология базируется на MDD и поддерживается архитектурным фреймворком. Типичный пример шаблона – завод ПО, а именно, инструмент Visual Studio, обеспечивающий разработку проектов из многократно используемых компонентов при реализации проектных решений и требований. Эти требования и решения описываются на языке DSL в виде задания архитектуры, блоков и документов, которыми заказчики приложения будут пользоваться при реализации соответствующей линии продукта или ее экземпляра.

Архитектурный каркас фабрики Авдошина [31]. Основной ресурс и активы данной фабрики – архитектурный каркас, являющийся отправной точкой, в разработке любого продукта на линии. В нем аккумулируются активы и ресурсы: классы, компоненты, конфигурации, образцы, проектные решения и т. п. Активы выбираются на стадии

разработки линии фабрики [53]. База знаний фабрики, включает в себя разные пособия, справочники, статьи, примеры программ и программы-образцы. Моделирование понятий и основных концепций в языке DSL способствует генерации кода и конфигурации структуры продукта. Для реализации фабрики задается схема фабрики, которая охватывает все активы, точки зрения и связи между ними. Из схемы берется шаблон фабрики и уточняется набор необходимых ресурсов для автоматизации работы среды разработки. Любая схема фабрики состоит из набора взаимозависимых точек зрения, каждая из которых дает возможность посмотреть на систему с разных сторон. Точки зрения отражают логическую и физическую стороны системы, набор используемых компонентов и документирование архитектуры семейства ПП. Каждая точка зрения имеет имя и описание, она указывает разработчику, что строить, как строить и из чего (моделей, средств, шаблонов и т. п.).

Шаблон фабрики – это пакет с ресурсами фабрики и экземплярами схемы категорий:

1) библиотеки и каркасы с КПИ, разработанными на линии (.NET Enterprise Library);

2) рекомендации, технологии, распоряжения и руководства процесса построения ПП;

3) языки предметной области и дизайнеры, которые задают необходимый уровень абстракции при разработке приложения и генерации кода по модели.

Построенный шаблон фабрики загружается в интегрированную среду разработки для автоматизации продукта.

Фабрика программ ”Continiuos integration” (непрерывная интеграция, повторная сборка) М.Фаулера (M.Fowler, K.Beck) [32]. Это автоматизированный процесс, обепечивающий разработку, поиск изменений в коде и в системе, сборку, контроль v собранных версий, развертывание и тестирование приложения. На фабрике используется гибкие (Agile) методологии, включая такие практики, как модульное (unit) тестирование, рефакторинг, стандарт кодирования. Каждое изменение версий (production enviroment) интегрируется продукт, который будет развернут согласно конфигурации программного и аппаратного обеспечения. Процесс интеграции выполняется на сервере интеграции (continuous integration server) – CruiseControl, который написан на Java и CruiseControl.NET. К основным процессам сервера интеграции относятся: **Build** – это компиляция (трансляция) исходных кодов в исполнимые файлы и их сборка средствами Ant, Maven в Java и NAnt в.NET. **UnitTest** предназначен для тестирование модулей приложения с помощью автоматизированных тестов. Проект **Deploy** развертывается после проведения Build и всех модульных тестов UnitTest. Развернутое приложение тестируется процессом Test с помощью автоматических функциональных тестов. После прохождения регрессионных тестов

считается, что интеграция прошла успешно. В противном случае требуется вносить исправления и процесс интеграции повторяется.

Фабрика EPAM [33] предоставляет широкий спектр технологий и решений, процессов, сертифицированных по стандартам качества CMMI Level 4 и ISO 9001, высокоэффективные инструменты управления проектами (http://www.epam-group.ru/#_sthash.zPWSiln4.dpuf) аутсорсинга для разработки, внедрения, интеграции, тестирования и сопровождения программного обеспечения. Практически создано более 15 филиалов в СНГ.

3. ТЕХНОЛОГИЧЕСКИЕ CASE-СРЕДСТВА СБОРКИ СИСТЕМ ИЗ КПИ

Разработка разнородных артефактов, reuses и КПИ осуществляется в рамках парадигм программирования (модульной, объектной, компонентной, сервисной, генерирующей). Цель каждой парадигмы – создать соответствующий базовый элемент и интерфейсный объект для связи разнородных базовых элементов парадигм в структуру ПС. Разработан формальный аппарат каждой парадигмы программирования. Он включает теоретические и прикладные аспекты проектирования соответствующих элементов и операции их интеграции в сложные ПС. Сборка разноязычных элементов основана на теории преобразования фундаментальных типов данных (FDT), возникших еще в 70–х годах прошлого столетия в работах Дейкстра, Хоара, Вирта, Агафонова, Ершова и др. и стандарте общих типов данных ISO/IEC GDT 11404 –2006 (General Data Types) для генерации типов данных GDT ↔FDT [8–10, 34].

Данная теория первоначально реализована в системе АПРОП (1980), АИС «Юпитер–470» (1982–1991) и на экспериментальной фабрике программ Киевского национального университета (КНУ) 2011 (http://programsfactory.univ.kiev.ua) [35-43]. В состав фабрики программ включен набор CASE–средств, который поддерживает аппарат формального построения отдельных объектов и компонентов данных парадигм. К ним относятся средства поддержки спецификации элементов в ЯП, описания интерфейсов в языке IDL и сохранения их в библиотеке готовых элементов. Сборка элементов парадигм в новые разные виды ПС осуществляется с помощью специальных операций сайта ИТК (http://sestudy.edu-ua.net) [3, 39–48]. В их разработке принимали участие аспиранты и студенты Киевского Национального университета (КНУ) и филиала МФТИ при институте Кибернетики имени академика Глушкова.

Стили парадигм сборочного типа. К настоящему времени сформировались [44-50]:

1) стили программирования по принципу модульности и стандартом модулей, которые собираются в более сложную структуру;

2) метод повышения эффективности межмодульных интерфейсов при передачи данных на компьютеры с разными форматами данных;

3) базы КПИ (библиотеки, репозитории), их идентификация, выбор и проверка пригодности применения исходя из стандартизованного описания элементов и их интерфейсов.

Базовые элементы и КПИ являются программными «кирпичиками", которые соединяются интерфейсами, как «болтами и гайками» технические изделия. Программные элементы могут быть в исходном и двоичном видах. Методология сборки позволяет подключать новые элементы, ресурсы, которые определяются в модульном, объектном, компонентном и сервисном программировании. Эти ресурсы должны разрабатываться стандартизировано в WSDL любых операционных средах IBM, MS, Microsystems, CORBA, COM, Oberon и др. [45-47].

Модульное сборочное программирование. Этот подход был исторически первым и базировался на процедурах и функциях, разноязыковых модулях и методологии императивного программирования в среде ЕС ЭВМ прототипа IBM–360.

Объектное сборочное программирование. Подход базируется на методологии объектно–ориентированного программирования и предполагает использование библиотек методов и классов (исходные коды или упаковки классов) в динамическую библиотеку. Существуют конкретные подходы, поддерживающие это программирование, например CORBA, Rational Rose, ОКМ и др.

Компонентное сборочное программирование. Основные идеи подхода – распространение классов в двоичном виде и предоставлении доступа к методам класса через строго определенные интерфейсы, которые позволяют снять проблему несовместимости компиляторов и обеспечивать смену версий классов без перекомпиляции. Существуют конкретные технологические подходы, поддерживающие компонентное сборочное программирование: COM (DCOM, COM+, .NET, OBERON и др.).

Аспектное сборочное программирование. Оно дополняет компонентное программирование концепцией аспекта для изменения варианта реализации критичных по эффективности процедур и программ. Это программирование заключается в сборке приложений из аспектных компонентов, инкапсулирующих различные варианты реализации (безопасность, синхронизация, надежность и др.). Существуют конкретные технологические подходы, поддерживающие данное программирование

Сервисное сборочное программирование. Это новая концепция интеграции сервисов и обслуживания ПС. К сервисам относятся: общие системные сервисы для поддержки реализации и выполнения ПС (связь, управление, каталогизация и др.); объектные сервисы, которые поддерживают объекты и классы, операции их выполнения и др.; веб–сервисы Интернет для быстрого решения поставленных задач, а также сервисно-компонентные службы SCA и SDO для создания приложений предприятий. Существуют конкретные системы, поддерживающие данное программирование.

Сборочный конвейер для поддержки стилей сборки

Сборка первоначально была представлена как способ объединения разноязычных объектов в ЯП и преобразования типов данных (ТД) с помощью теории спецификации и отображения типов и структур данных ЯП средствами алгебраических систем с операциями и функциями релевантного преобразования одних ТД в другие [8–10].

Сборка это:

1) метод программирования, который подчиняется общим закономерностям и функциям, используется для поддержки КПИ программных элементов, объектов, КПИ и интерфейсов;

2) механизм оценки стандартизованных КПИ и их интерфейсов и сборки продукта. Этим он отличается от процессов синтеза, композиции и от других методах программирования.

Базовые элементы сборки обладают свойствами (наследования, полиморфизма и инкапсуляции). Они включают описание данных и операций метода выполнения базового элементов для обеспечения связи разнязычных КПИ. Для технологичности сборки все базовые элементы сборки должны иметь паспорта в стандарте WSDL, в которых специфицированы данные, необходимые для информационной связи в более сложные структуры и выполнения в операционной среде.

Важное условие сборки – наличие большого количества разнообразных комплектующих КПИ, которые обеспечивают решение широкого спектра задач из разных предметных областей. Для их сборки задается схема сборки и операторы вызова (CALL, RPC, RMI и т. п.) в модуле, связанным отношением связи с другим модулем. В вызове задается список параметров и значений, которые при их передачи другому модулю проверяются на соответствие ТД исходя из аксиом и утверждений системы преобразования одних ТД к другим в классе ЯП. Результат отображения – сгенерированные интерфейсные модули–посредники для эквивалентных преобразований ТД в процессе выполнения.

Процесс сборки любых КПИ как готовых ПП – это линия сборочного конвейера, в котором роль "деталей" выполняют КПИ разной степени сложности, а роль "стыковщика" – интерфейсы–посредники. Последние присутствуют во многих методах и стилях программирования. На фабрики программ разработчики работают с КПИ как с деталями и выбирают те из них, которые могут быть комплектующими, т. е. повторно используемыми. Интерфейс каждой пара объектов сборки зависит от использования данных, их ТД и передаваемых значений, а также от наличия библиотек классов и функций преобразования ТД. В системе MS.Net имеются специальные библиотеки – CLS (Common Language Specification), стандартных процедур CLR (Common Language Runtime), типов CTS (Common Type System), которые реализуют типы данных в языке спецификации CLS в системе типов .NET.

3.1 Интерфейс сборки. Типы интерфейса

Для сборки разноязыковых модулей был придуман термин интерфейс (1976), соответствующий стыковочному элементу (болту, гайке) в промышленной сборке стандартных деталей. Интерфейс – это связь двух отдельных сущностей [34, 47]. В компьютерной области интерфейс определяется на разных уровнях: от уровня видимых коммуникаций между людьми до аппаратных, программных, пользовательских, языковых и других интерфейсов. Аппаратный интерфейс – это разъемы, конекторы и другие устройства для объединения компонентов в компьютерную систему и обеспечения перемещения информации с одного компьютера в другой. На *программном уровне* интерфейс между программами и ОС, между ОС и аппаратурой обеспечивает передачу и преобразование входных/выходных данных взаимодействующих программ в ОС или во время объединения компьютера с периферийным оборудованием. В ЯП интерфейс – типы данных и описание констант, переменных, параметров и сложных структур данных, которые образуют межъязыковой интерфейс ЯП и способ эквивалентных преобразований.

В программировании, интерфейс содержит паспортные данные программного модуля, включающие сведения о передаваемых данных и их типах. Каждый модуль специфицировался в одном из ЯП ОС ЕС. Стыковочный элемент (интерфейс) описывался в специальном языке MIL прототипе IDL (Interface Definition Language). Для каждой пары разноязычных модулей формировался интерфейс в виде модуля связника. По оператору Link A, B, C осуществлялась сборка модулей A, B, C и их интерфейсов в продукт на ЕС ЭВМ в рамках специально разработанной системы АПРОП [19]. Полученный продукт отличался от продукта традиционного программирования наличием самостоятельных отдельно разработанных модулей и их интерфейсов. Тем самым имеется возможность изменять модули и интерфейсы, которые осуществляют обмен данными и преобразование нерелевантных ТД с помощью (64) функций библиотеки интерфейса.

Идея интерфейса получила развитие к 1987–1984 годам, когда были созданы специальные языки описания программных интерфейсов IDL, API, SIDL и др.

Описание интерфейсов в IDL OMG CORBA

Язык описания интерфейсов IDL разработан в OMG CORBA [11]. В этом языке описываются интерфейсы как посредники (stub, skeleton). В них задается описание интерфейса, начиная с ключевого слова **interface**, имени интерфейса, типов параметров и операций вызова объектов. Параметрами обмена могут быть: **in** — входной параметр, **out** — выходной параметр, **inout** — совместный параметр. Пример описания параметров интерфейсов в виде stub F_1 и skeleton F_2 приведен ниже.

interface F_1 {
void f (in float s [l]); }
interface F_2 {
const long l=3 }
interface P_3: F_1 ( ) A, F_2( ) B.

Описание параметров интерфейса модуля в IDL CORBA имеет вид:

Request Operations
module CORBA {
interface Request {
Status add-arg (
in Identifier name
in Flags arg flags
);
Status invoke (
in Flags invoke flags // invocation flags
);
Status send();
Status get response (
out Flags response flags // response flags
);
};

В ТП сформировались следующие **виды интерфейсов**:

– модульный, программный (RPC, RMI, IDL, API, ISO и т.п.);

– межъязыковый интерфейс (Java Native Interface, SIDL– Scientifically IDL, Fundamental Data Types, GDT – General Data Types);

– сервисный интерфейс (IContract, Web, ISO);

– промежуточный интерфейс (Middleware, Virtualware, ISO, межсистемный, и между клиентом и сервером и т.п.);

– технический интерфейс (стандарт интерфейсной карты МПО–16Е–2).

Новый вид интерфейса в WCF *Icontract* содержит описание атрибутов и операций передачи данных от одного сервисного объекта клиента (*Service consumer*) к другому (*Service provider*). Их описание задается в языке XML. Передача интерфейсов между ними выполняет протокол, в котором задаются атрибуты и операции интерфейса.

В системе WCF реализовано четыре вида интерфейсов-контракта:

1) службы операций, вызываемых клиентом;

2) фундаментальне типы данных (int, float, string и др.) для передачи их другим модулям;

3) ошибки, которые могут содержаться при передачи контрактов клиенту;

4) сообщения для взаимодействия объектов между собой.

Каждый интерфейс IContract задает атрибуты и операции обмена данными между клиентом и сервером.

3.2. Модели взаимодействия систем и продуктов в современных средах

В связи с постоянным изменением архитектур компьютеров, появлением распределенных, клиент-серверных и гетерогенных сред выявилась неоднородность ЯП, в плане представления типов данных, подходов к их реализации в системах программирования, а также форматов представления и передачи данных [47]. Стандарт ISO/IEC 11404–1996 определил подход к решению вопросов интерфейса ЯП с помощью независимого языка LI (Language Independent) от ЯП. До настоящего времени отсутствует реализация LI и пользователи разных ЯП выбирают реализацию интерфейса в разных средах [2, 3, 45, 49].

Под взаимодействием *понимают совместимость двух и больше объектов. Данный термин имеет специальный спектр применения в программистской деятельности (например, взаимодействие программ и сред между собой). Способность к взаимодействию двух и больше программ в разных средах зависит от способа обмена информацией между ними. К способам обмена данными относятся RPC, RMI, ORB (stub, skeleton), IContract и др. Эти операции связывают разнородные программы через интерфейс в языке IDL, APL, SIDL и т. п. Именно интерфейс стал механизмом обеспечения взаимодействия (interconnection) разнородных программ и систем.*

Один из способов взаимодействия систем представлен в стандартной модели OSI (Open System Interconnection). Модель определяет виды взаимодействия систем на семи уровнях: прикладном, представления данных, сеансовом, транспортном, сетевом, канальном и физическом. Системные средства взаимодействия реализованы операционными системами в разных средах (DCE, OSF, ONS, OLE/COM, OMG CORBA, WCF, Eclipse и др.). В них связь между компонентами системы происходит через запрос к прикладному уровню модели OSI. Механизмы взаимодействия обеспечивают связь компонентов и систем, которые реализованы в соответствующих операционных средах.

Моделью взаимодействия – это набор параметров для обмена разнородной информацией между разными системами. Модель отображает систему отношений, которая существует между программами и системами в виде интерфейса и сообщений.

Каждая среда базируется на своих интерфейсах взаимодействия и включает в себя общие методы и средства доступа к данным. Программы, изготовленные в одной из сред, могут быть перенесены из одной среды в другую через интеграцию в репозиторий Eclipse. Эти среды обеспечивают реализацию процессов ЖЦ и объединение результатов ЖЦ в разные структуры ПП с помощью рассмотренных способов взаимодействия программ.

В рамках курса по ТП студентами проведена экспериментальная реализация принципов взаимодействия программ для следующих пар системных сред [39, 40]:

1) Visual Studio.Net↔Eclipse – это виртуальная среда для технологии разработки отдельных программ в языке C# и спецификации интерфейса для переноса готового продукта в репозиторий системы Eclipse. Эта система отображает связь с исходной средой разработки программ с помощью плагинов и конфигурационного файла с параметрами и операциями обработки данных в среде и среде выполнения Eclipse.;

2) CORBA↔JAVA↔MS.Net обеспечивают разработку программ на ЯП этой среды и устанавливают связи между этими средами с целью размещения разработанных программ в репозитории Eclipse для предоставления доступа другим разработчикам к этим программам;

3) IBM VSphere↔ Eclipse предоставляет средства для разработки новых программ с использованием ЯП, которые допустимы в среде или в VSphere виртуального варианта системы.

Кроме этих моделей взаимодействия программ и сред исследованы системные средства взаимодействия WCF (Windows Communication Foundation) с применением протокола IContract.

3.3 Конфигурация (сборка) программных артефактов парадигм

Конфигурационная сборка построена на основе концепции вариабельности Product Line и включает [45, 50-59]:

модель инженерии домена для разработки отдельных КПИ, артефактов и их сборки;

процессную модель «для обеспечения повторного использования» (for reuse) артефактов и разработки ПС и СПП «с использованием КПИ» (with reuse);

модель управления вариабельностью для реализации процесса конфигурации продукта из КПИ с учетом точек в модели вариабельности.

Модель вариабельности семейства систем СПС представляет собой пару взаимно согласованных моделей: $VM = \{SV, AV\}$,

где SV – модель СПС структуры;

AV – модель активов СПС процесса разработки.

Модель VM используется для:

– последовательного управления изменчивостью интеграционной структуры СПС;

– представления уровня изменчивости продукции СПС;

– учета затрат и времени для изготовленных систем семейства.

Процесс управления конфигурации включает следующий набор действий:

– систематическое отслеживание внесенных изменений в отдельные КПИ конфигурации, проведения аудита изменений и контроля изменений в каждый отдельный компонент;

– поддержки целостности конфигурации, ее аудит и обеспечение внесения изменений в один из объектов конфигурации, а также в связанный с ним другие объекты;

– *контроль конфигурации путем проверки программных или аппаратных элементов в соответствии с вариантом конфигурации и требованиями;*

– трассировка изменений конфигурации на этапах сопровождения и эксплуатации ПО.

Компонентная конфигурация - это архитетура компонентной модели из совокупности компонентов и КПИ, взаимодействующих согласно требований и правил этой модели.

Следствием этого определения является:

1. Наличие интерфейсов каждой пары компонентов, взаимодействующих между собой.

2. Количество компонентов у компонентной конфигурации может быть произвольным, но, согласно требований для конкретной компонентной ПС минимальное значение равняется единице, а максимальное - числу классов эквивалентности для всей коллекции компонентов.

Конфигурация СПС с использованием КПИ сокращает время и повышает уровень качества ПС. Ее экспериментальный вариант реализован на сайте ИТК (http://sestudy.edu-ua.net), а также в системных фабриках AppFab IBM, .Net, Intel и др. Управление конфигурацией ПС в СПС из элементов разных парадигм программирования приведен на рис. 2 [45, 54].

К операциям управления конфигурацией ПС в СПП относится:

– идентификация конфигурации, элементов конфигурации и данных;

– функции управления процессом изменений;

– модели, методы и метрики оценки вариабельности;

В модельную среду конфигуратора КПИ и артефактов входят:

– схемы формального описания артефактов:

–пример решения квадратного уравнения с визуальным представлением модели характеристик с отмеченными точками вариантов;

– база данных, включающая требования, архитектуру, набор базовых КПИ и ПС;

– конфигуратор, объединяющий артефакты в ПС и СПП.

В процессе сборки происходит преобразование исходного кода в код или КПИ в СПС, которые могут быть запущены на компьютере, или преобразованы в код выполнения. Одним из шагов создания сборщика является процесс компиляции исходного кода, где файлы превращаются в промежуточный код или в код выполнения простых программ.

Для СПС после компиляции ПС происходит процесс связи специальной программой — связником. Процесс связи представляет собой

замену относительных адресов функций внешних библиотек в реальные адреса, используемые в программе при выполнении.

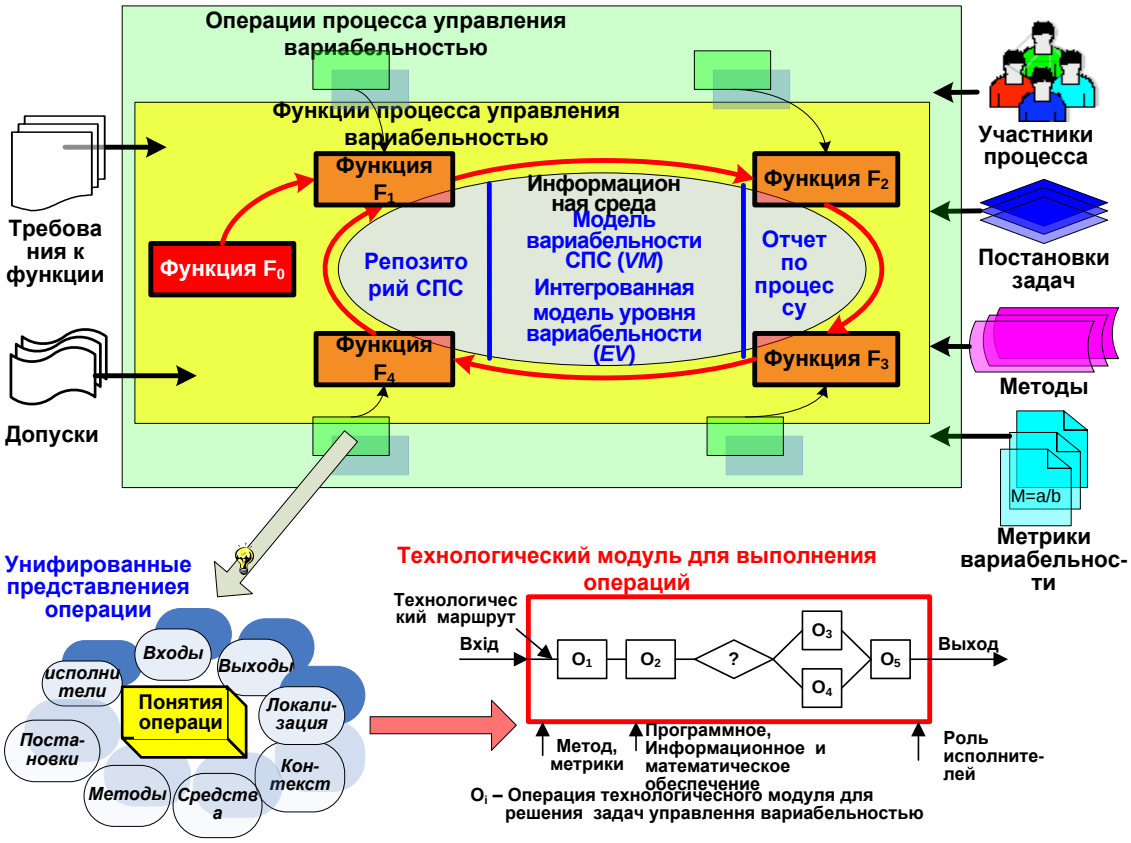


Рис. 2 – Схема процесса управления вариабельностью СПП в среде сайта

Реализация *модели вариабельности* – компактное представление всех продуктов СПП с точки зрения характеристик (функциональности). Эта модель визуально представляется диаграммой, которая содержит характеристики ПС их зависимости, ограничения в виде (деревьев) характеристик или возможных комбинаций (рис.3).

Для примера взят алгоритм решения квадратного уравнения в наглядном представлении набора возможных точек вариантности, вариантов и ограничений на их использование. На данном рисунке представлена схема работы алгоритма задачи в виде КПИ, который описан средствами Windows Workflow Foundation (WWF). По точкам вариантов в схеме WWF выполняется управление вариабельностью ПС. WWF среды Visual Studio разрешает разработчику вставить любой КПИ в заданную точку вариантности. Получен проверенный продукт, который задает необходимую функциональность и возможность изменять конфигурацию ПС из заданной коллекции компонент.

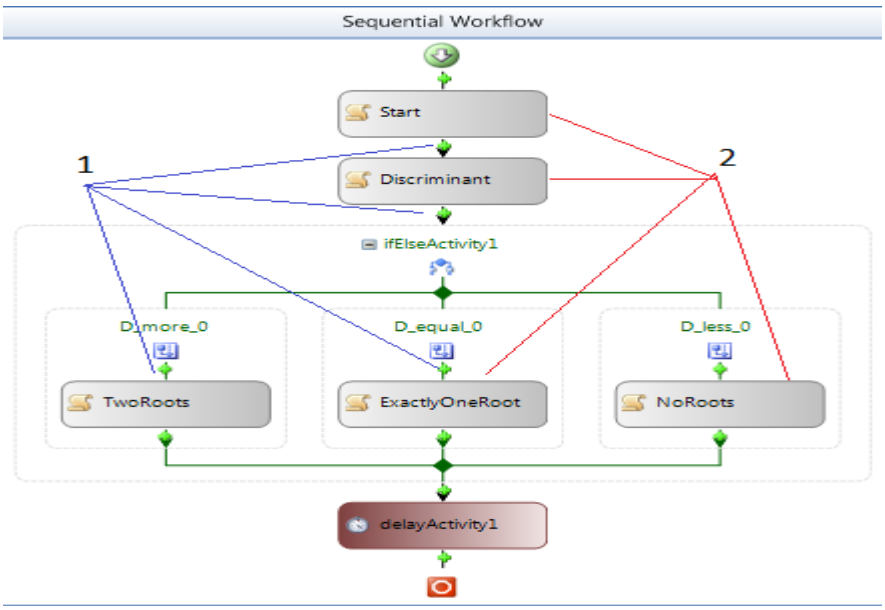


Рис. 3 – Схема решения кважратного уравнения с точками вариантов

Конфигуратор реализован в среде ИТК. Он обеспечивает сборку КПИ и компонент, которые могут изменяться. Обеспечивает добавление и поддержку модификации структуры системы по точкам вариабельности [40, 51]. Приведенная схема алгоритма обработана на инструменте WWF в среде Visual Studio. Все компоненты сохраняются в репозитории объектов данной программы. Описанный пример решения задачи нахождения корней квадратного равнения в xoml код есть не что иное, как обычный расширенный xml, который описывает последовательность вызовов команд бизнес логики.

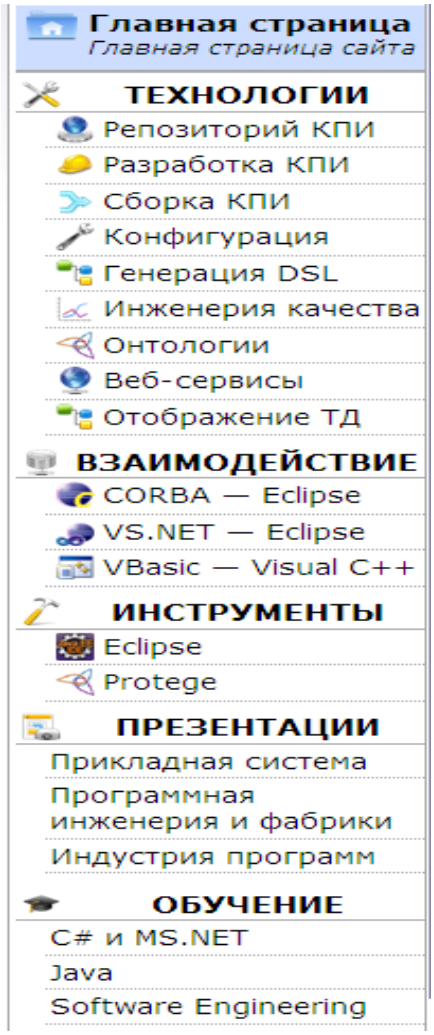
Технологии ИТК, включая «Конфигурацию» (рис.4).

Реализованные в ИТК средства и технологии ориентированы на производство ПС и СПС из готовых КПИ по простым линиям обработки КПИ в разных ЯП, которые фактически отображают идею сборочного создания современных программ, систем и их семейств. В ИТК входит фабрика программ КНУ (http://programsfactory.univ.kiev.ua), реализованная студентами под руководством автора курсов "Программная инженерия" и "Технологии программирования ИС". В 2007г. опубликован учебник по методам и средствам программной инженерии, который читался в МФТИ РАН и используется на сайте www.intuit.ru.

В состав сайта ИТК входят линии, которые разработаны студентами при обучении технологии программирования и программной инженерии:

1. Линия проектирования программ в MS.NET.
2. Представление артефактов в языке WSDL.
3. Сертификация КПИ для хранения а репозитории.
4. Конвейерная сборка КПИ в системы.
5. Учебник по «Программная инженерия»

К сайту обратилось более 35000 из разных стран мира. Есть намерения развивать ее в направлении технологий для изготовления новых механизмов и веществ в рамках e-sciences (биология, физика, химия и др.).



а

Сайт ИТК содержит следующие технологические разделы.

ТЕХНОЛОГИИ ПОСТРОЕНИЯ ПС:

Технология обслуживания репозитория КПИ,
Технология разработки КПИ,
Технология сборки КПИ,
Технология конфигурирования КПИ,
Технология генерации описания КПИ в языке DSL,
Технология оценка затрат и качества,
Технология онтологии вычислит. геометрии, ЖЦ ISO/IEC12207,
Технология веб-сервисов,
Технология генерация типов данных ISO/IEC 11404.

ВЗАИМОДЕЙСТВИЕ программ, систем и сред:

Модель CORBA – Eclipse–JAVA,
Модель VS.Net C# – Eclipse,
Модель Basic – C++.

ИНСТРУМЕНТЫ ИТК:

Система Eclipse,
Система Protégé.
Система оценки стоимости и качества.

ПРЕЗЕНТАЦИИ ПС В ИТК:

Система ведения зарубежных командировок НАНУ,
Слайды про ИТК, фабрики программ
Методологии построения ТЛ
Фабрика программ КНУ

ОБУЧЕНИЕ:

Технологии программ в C# VS.Net и JAVA,
Электронный учебник "Программная инженерия"
на сайте КНУ http://programsfactory.univ.kiev.ua.

б

Рис. 4 Главная страница веб-сайта
а – название разделов веб- сайта, б – перечень технологий веб-сайта

4. ЗАКЛЮЧЕНИЕ

В работе рассмотрены коллекции CASE-инструментов и средств, включающих системные AppFab и прикладные (Application Fabric) фабрики, специальные фабрики программ (Дж.Гринфильда, Г.Ленца, М.Фаулера, С.Авдошина и др.), продуктовые (Product Line /Product Family) и технологические линии. Все они обеспечивают построение ПС и ИС из многоразовых компонентов и reuses. Представлен отечественный вариант метода сборки, модели вариабельности, обеспечивающие взаимодействие и изменяемость систем и их семейств.

Рассмотрены формальные аспекты создания КПИ, описания их паспортов в языке WSDL и интерфейсов в языке IDL. Они конфигурируются в изменяемую структуру систем для выполнения на современных системных платформах. Представлена коллекция технологий ИТК для поддержки процессов разработки КПИ, систем и их семейств, взаимодействующих между собой и способных вносить изменения в готовую систему.

Библиографический список

1. Лаврищева Е.М. Развитие отечественной технологии программирования.–Кибернетика и системный анализ, 2014, т50, № 3.– С. 121–143.
2. Лаврищева Е.М. Методы программирования. Теория, инженерия, практика. Наук. Думка, 2006.– 451 С. (www.twirpx.com).
3. Лаврищева Е.М. Software Engineering компьютерных систем. Парадигмы, Технологии, CASE-Средства программирования.–Научная Думка, 2014.–284с.
4. Software engineering as a scientific and engineering discipline E. M. Lavrishcheva 2008, Volume 44, Number 3, Pages 324–332
5. Classification of software engineering disciplines, E. M. Lavrischeva 2008, Volume 44, Number 6, Pages 791–796, Software–Hardware Systems.
6. Липаев В.В. Программная инженерия сложных заказных программных продуктов. Учебное пособие.–Мак-Пресс, Москва, 2014.–308с.
7. Липаев В.В. Программная инженерия сложных заказных программных продуктов. Учебное пособие.–Мак–Пресс, Москва, 2014.–308с.
8. Лаврищева Е.М. , Грищенко В.Н. Связь разноязыковых модулей.– Москва.– Финансы и статистика.–1982.– 136 с.
9. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. Основы индустрии программных продуктов.– К.: Наук. Думка, 2009.–371 С. (www.twirpx.com).
10. Лаврищева Е.М., Петрухин В.А. Методы и средства инженерии программного обеспечения. – М.: МОН РФ, 2007. – 415 с.– Aviable (www.intuit.ru).

11. Эммерих В. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и Java RMI. – М.: Мир, 2002. – 510с.
12. Буч Г., Рамбо Д., Джекобсон А. Язык UML. Руководство пользователя: Пер. с англ. – М.: ДМК, 2000. – 432 с.
13. Pohl K., Böckle G., Linden F.*J.* Software Product Line Engineering: Foundations, Principles and Techniques. – New York: Springer–Verlag. – 2005. – 437 p.
14. Computer Journal of IEEE Computer Society, Face Recognition Technology, November 2013.
15. Ведеров А.М. CASE–технология. Современные методы и средства проектирования информацмонных систем. –М.: Финансы и статистика, 1998.–176с.
16. Соммервилл И. Инженерия программного обеспечения.– Изд. Дом «Вильямс», Москва *Санкт-Петербург*Киев.– 2002.–623с.
17. Чернецки К., Айзенекер У. Порождающее программирование. Методы, инструменты, применение.– Издательский дом Питер. – М. – СПб. – Харьков. – Минск.– 2005.– 730 с.
18. Уилсон С.Ф., Мєйлс Б., Ленгрев Т. Принципы проектирования и разработки программного обеспечения. Учебный курс MCSD.–Пер. с анг.– М.: Из–во торговый дом «Русская редакция», 2000.–608с.
19. Domain-Specific Languages: An Annotated Bibliography. Arie van Deursen – Paul Klint – Joost Visser. CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands http://www.cwi.nl/arie,paulk,jvisser/ Domain-Specific Languages Languages:An Annotated Bibliography1.htm.
20. Корпорация SAP – http://www1.sap.com/ www.sap.ruvs.net
21. P&P – www.ibm.eebsphereю.bisint
22. CASE "Oracle" – http://www.oracle.com
23. Компания ИВК (Россия) http://www.ivk.ru
24. IBM Sphere ПО portal www.01.ibm/com/ software/ru/webspere
25. Глушков В.М., Лаврищева Е.М., Стогний А.А. и др. Система автоматизации производства программ м (АПРОП). – К.: ИК АН УССР, 1976. – 134 С.
26. Guckenheimer S., Perez J.I. Software Engineering with Microsoft Studio Team System.–Crawfordsville, USA: Adison–Wesley, 2006.–304 P.
27. Бей И. Взаимодействие разноязыковых программ.–Диалектика.–М.:С-Петербург–Киев.– Изд. дом. "Вильямс".– 2005.–868с.
28. Гринфильд *Дж.* Фабрики разработки программ.–Диалектика.–М–С–Петербург–Киев.–Изд.дом. "Вильямс".– 2007.–591с.
29. Grid – Distributed Computing at Scale.An overview of Grid and the Open Grid Forum.GWD-I Mark Linesch, HP. Marketing April 23, 2007. http://www.ogf.org/ Public_Comment_Docs/Documents/Apr.
30. Lenz G. Wienands. *C.* Practical Software Factories in .NET. From theory in practice – a primer reference and case study. – Apress, 2007. – 205 p.

31. Авдошин С.М. Фабрики программного обеспечения. – ttp://www/ewwek.com./on
32. Duval P., Matyas, Grover A. Continuous integration Împroving Software Quality and Reducing Risk, Addison Wesley, 2009. – 691 p.
33. EPAM– systems – разработчик информационных технологий, www.epam.com.
34. Лаврищева Е.М., Коваль Г.И., Бабенко Л.П., Слабоспицька О.А., Игнатенко П.П. Новые теоретические основы технологии производства семейств программных систем в контексте генерирующего программирования, ИПС НАНУ, ВИНИТИ,2011.– 277 с.
35. Аронов А. А. Дзюбенко А. И. Подход к созданию студенческой фабрики программ // Проблемы программирования, №3, 2011.– с.42–49.
36. Lavrischeva E., Aronov A., Dzubenko A. Programs factory – a conception of Knowledge Representation of Scientifical Standpoint of Software Engineering. Jornal of Computer Science, Canadian Senter of Science and Education, ISSN1913-8989, 2013, p.21 – 27.
37. Lavrischeva E., Ostrovski A., Radetskyi I. Approach to E–Learning Fundamental Aspects of Software Engineering.– // 8– th international Conf. ICTERI– 2011 "ICT In Education, Research and Industrial Applications".– Publ. springer –sbm.com/ocs/home/
38. Лаврищева Е.М., Зинькович В.М., Колесник А.Л. и др. Инструментально-технологический
комплекс разработки и обучения приемаи производства программных систем.– Гос. Служба Интеллектуальной собственности Украины.–Свидетельство о регистрации авторского права на продукт – № 45292, от 27.08.2012.
39. Островський А.В. Подход к обеспечению взаимодействия программных сред JAVA и MS.Net.– Проблемы программирования, 2011.– №2.–с.37–44.
40. Радецкий ИА. Один из подходов к обеспечению взаимодействия сред MS.Net и Eclipse // Проблемы программирования, №2, 2011.– с.45–52.
41. Ekaterina Lavrischeva, Alexei Ostrovski. New Theoretical Aspects of Software Engineering for Development Applications and E-Learning, Journal of Software Engineering and Applications, 2013, 6, 34–40 – http://dx.doi.org/10.4236/ jsea.2013,.69A004 Published Online September 2013 (http://www.scirp.org/journal/jsea)
42. Lavrischeva E.: Generative and Composition Programming: Aspects of Developing Software System Families. In: Cybernetics and Systems Analysis, vol. 49, no.1, pp. 110–123. Springer, Heidelberg (2013) http://link.springer.com/article/10.1007%2Fs10559-013-9491-6.
43. Lavrischeva E.: Formal Fundamentals of Component Interoperability in Programming. In: Cybernetics and Systems Analysis, vol. 46, no. 4, pp. 639–652. Springer, Heidelberg (2010) http://link.springer.com/article/10.1007%2Fs10559-010-9240-z

44. Ekaterina Lavrischeva, Andrey Stenyashin, Andrii Kolesnyk, Object-Component Development of Application and Systems. Theory and Practice, Journal of Software Engineering and Applications, 2014, 7, Published Online August 2014 in SciRes http://www.scirp.org/journal/jsea
45. Лаврищева Е.М. Парадигмы программирования. УкрПро–2014, Спец. выпуск.– с.94–112.
46. Lavrischeva, E.: Theory and Practice of Software Factories. Software–Hardware Systems. In: Cybernetics and Systems Analysis, vol. 47, no. 6, pp. 961–972. Springer (2011).m http://link.springer.com/article/10.1007%2Fs10559-011-9376-5
47. Лаврищева Е.М. Взаимодейсвие программ, систем и операционных сред // Проблемы программирования, №3, 2011.– с.13–24.
48. Аронов А. О. Дзюбенко А. І. Подход к созданию студенческой фабрики программ // Проблемы программирования, № 3, 2011. – с. 42–49.
49. Колесник А.Л. Модели и методы разработки семейства вариантных программных систем. – Автореф. – КНУ, 2013. – 22 С.
50. Лаврищева Е.М., Слабоспицька О.А., Коваль Г.І., Колесник А.А. Теоретические аспекты управления вариабельностью в семействах программных систем. – Весник КНУ, серия физ.–мат.наук. – 2011. – №1. – С. 151–158.
51. Гамма Э., Бек К. Расширение Eclipse: принципы, шаблоны и подключаемые модули / Пер. с англ. – М.: КУДИЦ–ОБРАЗ, 2005. – 384 с.
52. Protégé-Frames User's Guide: protégé.Stanford.edu/doc/sydex /php /prf-ug
53. Lavrischeva E., Ostrovski A. General Disciplines and Tools for E–Learning Software Engineering.– 9– th International Conf. ICTERI–2012 Springer.com, Communication in Computer and Information Sciences, ISSN 1865-0929.- http://senldogo0039.springer- sbm.com/ocs/
54. Kolesnik, A, Koval, G.: The Theoretical View for Software Family Variability Management [in Ukrainian]. In: Bulletin of University of Kiev. Series: Physics & Mathematics, vol. 1, pp. 151–158. Kiev (2011)
55. Lavrischeva, E., Slabospitcka, O.: An Approach for Expert Assessment in Software Engineering. In:
Cybernetics and Systems Analysis, vol. 45, no. 4, pp. 638–654.
56. Kolesnik, A, Koval, G.: The Theoretical View for Software Family Variability
Management [in Ukrainian]. In: Bulletin of University of Kiev. Series: Physics &
Mathematics, vol. 1, pp. 151–158. Kiev (2011)
57. 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, CEUR-WS.org/Vol-848, ISSN 1613–0073,

urn:nbn:de:0074-848-8. – P. 125–133. – Available at http://ceur-ws.org/Vol-848/ICTERI-2012-CEUR-WS-paper-31-p-155-162.pdf
58. Ekaterina Lavrischeva, Andrey Stenyashin, Andrii Kolesnyk, Object-Component Development of и
Application and Systems. Theory and Practice, Journal of Software Engineering and Applications, 2014, 7,
Published Online August 2014 in SciRes http://www.scirp.org/journal/jsea
59. Lavrischeva, E., Slabospitcka, O.: An Approach for Expert Assessment in Software Engineering. In:
Cybernetics and Systems Analysis Cybernetics, 2008. vol. 45, no. 4, pp. 638–654.