МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
МОСКОВСКИЙ ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
(ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ)

Кафедра информатики и вычислительной математики

Е. М. Лаврищева

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Тема 3. БАЗОВЫЕ ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Учебно-методическое пособие

МОСКВА
МФТИ
2016

УДК 681.3.06

Рецензент
Член-корреспондент РАН, доктор физико-математических наук,
профессор *И.Б. Петров*

Лаврищева, Е.М.

Программная инженерия. Тема 3. Базовые основы программной инженерии: учебно-методическое пособие. – М.: МФТИ, 2016. – 51 с.

Представлены школы по теории программирования (А.А. Ляпунова, Ю.И. Янова, А.П. Ершова, В.М. Глушкова, Е.Л. Ющенко, Г.Е. Цейтлина, В.Н. Редька и др.) на первых ЭВМ. Дана характеристика теории схем программ и автоматов, алгоритмического, алгебраического и синтезирующего программирования. Рассмотрены подходы к формальной спецификации программ и доказательства их правильности. Дана теория композиции и сборки модулей в сложные системы.

Предназначено для преподавания студентам 1–3 курсов, обучающихся в области информатики, программной инженерии и компьютерных наук.

Учебное издание

Лаврищева *Екатерина Михайловна*

ПРОГРАММНАЯ ИНЖЕНЕРИЯ

Тема 3. БАЗОВЫЕ ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Учебно-методическое пособие

Редактор *Л.В. Себова.* Корректор *В.А. Дружинина.* Компьютерная верстка *Л.В. Себова.*
Подписано в печать 15.09.2016. Формат 60 × 84 $^{1}/_{16}$. Усл. печ. л. 3,25. Уч.-изд. л. 2,9.
Тираж 50 экз. Заказ № 385.
Федеральное государственное автономное образовательное
учреждение высшего образования «Московский физико-технический институт
(государственный университет)»
141700, Московская обл., г. Долгопрудный, Институтский пер., 9
Тел. (495) 408-58-22. E-mail: rio@mipt.ru

Отдел оперативной полиграфии «Физтех-полиграф»
141700, Московская обл., г. Долгопрудный, Институтский пер., 9
Тел. (495) 408-84-30. E-mail: polygraph@mipt.ru

ВВЕДЕНИЕ

Программная инженерия (ПИ) с момента своего появления в 1968 г. заняла центральное место среди компьютерных наук, информатики, информационных систем и технологий. ПИ предназначена для разработки *программного обеспечения* (ПО) программных, прикладных и информационных систем разного назначения [1–5].

Основные положения ПИ представлены в ядре знаний SWEBOK (Software Engineering body of Knowledge, www.swebok.com, 2001 г.), созданном международным комитетом IEEE и ACM. Это ядро включает 10 разделов знаний (area knowledge). Первые пять разделов ядра – это определение требований, проектирование, конструирование, тестирование и сопровождение ПО. Следующие пять разделов – организационные. К ним относятся – управление проектом, конфигурацией, качеством, методы и средства инженерии ПО. Ядру знаний соответствуют процессы жизненного цикла (ЖЦ) стандарта ISO/IEC 12207, которые ориентированы на реализацию ПО, изложенную в пяти разделах SWEBOK.

В SWEBOK сформулировано определение понятия ПИ. *Это система методов, средств и дисциплин планирования, разработки, эксплуатации и сопровождения ПО, готового к внедрению*.

К основным концепциям ПИ отнесены – продуктивность, индустрия и качество.

Разделы знаний SWEBOK 2001 г. ориентированы на создание ПО. В них не определены такие целевые объекты, как программные системы (ПС), семейство СПС, семейство продуктов СПП и т.п. Все разделы аккумулируют вопросы теории и практики изготовления для ПО программного продукта (ПП) с учетом требований к создаваемому ПО. Кроме того, в нем не нашло отражение проблем и задач защиты данных и обеспечения безопасности работы изготовленного ПО для систем, а также не описана сущность индустрии ПО и систем, экономики затрат на их разработку и принципов внедрения и сопровождения ПП.

Основатели ядра SWEBOK 2001 г. не осветили подход к индустриальной разработке ПП, который практически сформировался.

Под *индустрией* понимается производство разных видов продуктов массового применения и средств их изготовления. То есть основная задача любой индустрии – массовый выпуск продукции и получение прибыли. Применительно к программным продуктам В.М. Глушков в 1974 г. выдвинул идею сборки ПП и систем из готовых программных элементов, подобно сборочному конвейеру в автомобильной промышленности Форда. И, как показала практика, эта идея оказалась плодотворной. Сборочный принцип

производства ПП является наиболее развитым и используется во всех фирмах производителей программного продукта.

Сегодня индустрия *программной продукции* мировых фирм-производителей базового программного обеспечения (Microsoft, IBM, Corba, Java, Intel, Apple и др.), а также индийской фирмы по обновлению и доработке наследуемых (legacy) систем дает огромные прибыли.

В период появления ПИ в СССР развивалась технология программирования (ТП) программ, систем и ПО в рамках создания новых вычислительных, управляющих и инженерных ЭВМ. ТП – это методы, средства и инструменты создания ПО для новых компьютеров и описания разных вычислительных задач для решения на ЭВМ. За рубежом формировалась компьютерная наука (Computer Sciences – CS), включающая теорию и эксперименты создания аппаратных и прикладных вычислительных систем.

Тема 3. БАЗОВЫЕ ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Основные компоненты CS-науки: Computer Engineering (компьютерная инженерия), System Engineering (системная инженерия), Software Engineering (SE – программная инженерия), соответствующая ТП. Согласно энциклопедии CS (1992) приведем краткое определение этих инженерных дисциплин CS.

Компьютерная инженерия – это теории, принципы и методы построения компьютеров (frameworks, суперкомпьютеров и т.п.), а также системного ПО ЭВМ (ОС, трансляторов, загрузчиков и т.д.). *Системная инженерия* – это теория, методы и принципы построения информационных систем, систем управления и Computer Systems. *Программная инженерия* – это система методов, способов и дисциплин планирования, разработки, эксплуатации и сопровождения ПО (www.swebok.com).

Принципам и методам построения информационных систем (ИС) В.М. Глушков посвятил свой последний научный труд «Безбумажная информатика» (1982) [8]. В нем *информационные системы* – это компьютерные системы обработки информации на предприятиях, в органах управления и в производственной сфере. Базис таких систем – документы, не бумажные, а электронные, и система документооборота на всех уровнях управления государственных предприятий и органов.

Информационные технологии (ИТ) – базис компьютерной инфраструктуры современных корпораций, предприятий и государственных органов управления, на которых главным источником деятельности является информация и решение различных задач обработки информации локального и глобального характера.

3.1. Определение базовых понятий программной инженерии

Занимаясь системно вопросами технологии программирования и CS, автор опубликовал в 1980-х годах ряд статей и сформулировал свое понимание SE. Оно представлялось в курсах лекций (1980–1991) и в Доме научно-технической пропаганды в Киеве (1990–1991). В результате был опубликован препринт под названием «Проблематика программной инженерии» (1991) [3]. В нем отображены сложившиеся точки зрения многих отечественных и зарубежных специалистов в области ПИ. Определены базовые понятия, смысл и методы создания ПП на начальном периоде развития ПИ, которые приводятся ниже.

Термин *программная инженерия* (software engineering – SE), появившись в 1968 г., означал «систематический подход к разработке, эксплуатации, сопровождению и прекращению использования программных средств». С этим термином связаны важные разработки и в области ТП и ПИ. Они публиковались в ряде научных зарубежных журналов: IEEE Transaction on Software engineering, Software Engineering Journal, Computer Technology, а также в советских журналах – Современные проблемы кибернетики, Программирование и др. Для ядра SWEBOK разработана международная программа обучения ПИ – Curricula 2001. Эта программа обучения ПИ уточнялась в 2004, 2007, 2014 годах.

ПИ является инженерно-научной дисциплиной [9–12], методы, средства и инструменты которой обеспечивают качественный и производительней труд лиц, занимающихся различными видами деятельности по созданию ПО и использованию программных систем (ПС) и ИС на инженерной основе. Появление ПИ знаменует переход от отдельного *мануфактурного* производства программ к промышленному. В нем для заданного предмета труда (программного объекта) определены: спецификация объекта (эскиз), действия (операции и процедуры), выполняемые исполнителями с целью создания по эскизу опытного образца ПС с заданными функциями и требуемым качеством. На основе опытного образца осуществляется его тиражирование для массового применения.

Появлению термина ПИ в СССР предшествовал термин *технология программирования*, который обозначал *методы, средства и инструменты, обеспечивающие процесс создания ПС* [13–15]. Эти определения настолько близки, что фактически их можно трактовать как дальнейшее развитие ТП в плане обеспечения программистского труда инженерными методами организации (планирование, учёт, контроль) выполнения работ. Такое развитие, по существу, означало переход от одиночного создания программ отдельными лицами к промышленному их производству со всеми вытекающими проблемами организации труда.

На первом этапе создания сложных систем главную роль в процессе программирования играли интуиция и знания отдельных программистов. Они вкладывали свои знания и опыт, творческие способности в методы и средства создания ПС для выполнения на ЭВМ. Тем самым налагались некоторые ограничения на свободу действий рядовых программистов в виде определенных регламентаций, правил и стандартов.

В результате подошли, как и в промышленности, к технологиям, при которых не только определяется предмет труда, выполняемый на операциях и процессах, но и проводится оценка этого труда на трудоемкость и стоимость, а также делается оценка показателей качества (надёжность, эффективность и др.).

В становлении ТП и ПИ сложилось три основных подхода [3]:

– инженерный;

– управленческий;

– инженерный.

Инженерный подход аналогичен технологии промышленного производства, в которой рассматриваются регламентированные последовательности технологических операций (ТО), призванные гарантировать получение продукции с заданными свойствами. ТО связаны с индивидуальными особенностями исполнителей, которые, как правило, не учитывались, так как не отражали их творческие способности. Поэтому некоторые специалисты в области ПИ считают, что этот подход не приведет к большому прогрессу.

Однако в сфере ПИ существуют виды работ, относящиеся к этому подходу: правильное копирование программ и документации; настройка и генерация программ; ввод и контроль данных и др. Эти виды работ наиболее близки к производству. В нём также применяются специализированные работы (отладка, анализ, тестирование), которые могут выполняться по жестко отработанной, многократно проверенной технологии.

Вместе тем творческие способности одних специалистов можно заложить в более трудные процессы создания ПС, сделав труд разработчика программ более инженерным и производительным. Процесс разработки ПО и систем включает операции, требующие элементов творчества программистов, которые не ограничены определенными рамками процесса создания ПС, а именно: заданной последовательностью работ и результатом, зависящим от заданных в техническом задании требований к готовому программному продукту.

Управленческий подход ставит во главу угла четкие организационные структуры групп разработчиков программ и обязательное выполнение нормативных документов и стандартов при разработке программ и систем. Данный подход не учитывает основ ПИ, ориентированных на опти-

мизацию отношений руководителя с подчиненным и с потребителями в зависимости от специфики ПС. Инструментом оптимизации их отношений являются инструкции по созданию продукта и обеспечению его качества. Для управления качеством разработан стандарт ГОСТ 2844-89 «Оценка качества программных средств» (1987). Он предусматривал большой объем рутинных работ по экспортированию отдельных свойств и характеристик ПС в продукт на этапах жизненного цикла. Оценка качества ПО проводилась по многим оценочным элементам (50 критериев и 6 факторов). Однако экспертная оценка недостаточно точно отражала реальное качество ПО, и большие накладные расходы на эту оценку не оправданны. Методы управления таким необычным продуктом, как ПО, должны влиять не только на взаимные договоренности между заказчиком и исполнителем ПО, но и между менеджером и программистами.

Инженерный подход основывается на автоматизации средств создания ПО и методов программирования.

Известны попытки свести решение различных проблем повышения производительности труда и качества разработки ПО к созданию или освоению технологических ПС простейшего или общего характера, автоматизирующих все программистские работы с помощью программных инструментов (*software tools*). Первоначально возникала опасная возможность замены профессионального мастерства исполнителей готовыми инструментами и средствами. При этом инструменты оказывались ненадёжными, трудоёмкими в использовании с имеющейся документацией. Такие обстоятельства зачастую вели к игнорированию хороших сторон рассматриваемого подхода, связанных с методологическим характером программирования, включающим:

– разработку ПО и программной документации на него;

– комплексное управление качеством ПО;

– технику модульного и структурного программирования.

В практике программирования каждый из указанных подходов самостоятельно не использовался. Как правило, программист выбирает подход, подходящий к особенностям среды, в которой он работает. Программный объект является средством ПИ. Он обладает следующими признаками:

– многократной применяемостью;

– надёжной работоспособностью;

– спецификой, выгодно выделяющей данное средство от всех других, функционирования объекта разработки, его внедрения и эксплуатации.

В связи с изменчивостью условий ПО и среды функционирования объекта на ЭВМ первый признак недостижим без обеспечения соответствующего качества продукта, гарантирующего передачу его другим лицам и организациям.

Продуктом деятельности программиста может быть не только программа, но и технология ее изготовления, представленная методически или инструментально.

Далее дается определение основных элементов ПИ.

3.2. Основные объекты разработки программной инженерии

Основным понятием ПИ является объект разработки – программа во всех ее проявлениях на этапах жизненного цикла (ЖЦ), а также программная система, методы, средства и инструменты их изготовления [16].

Программа – это объект разработки, который не является осязаемым (нельзя пощупать, взвесить и т.п.) человеком, а доступен пониманию ЭВМ, для которой написан. Готовая программа – это программный продукт, реализующий определённые функции (задачи) некоторой предметной области (ПрО), процесс проектирования и разработки которого осуществляется соответствующими методами, средствами и инструментами ПИ на технологических процессах (ТП).

Объектами разработки могут быть: модуль, программа, комплекс программ, пакет прикладных программ, система и др. То есть объект либо сам является отдельной конструктивной единицей разработки, реализующей элементарную функцию ПрО, либо состоит из взаимосвязанного их набора.

Для объекта разработки первоначально формируется его модель, в которой специфицируются реализуемые функции и элементы данных, устанавливаются их связи и отношения. То есть для объекта определяется его спецификация или прототип.

В техническом задании (ТЗ) на разработку объекта, программы выдвигаются определенные требования к составу реализуемых функций и их характеристик – работоспособность (свойство надежности), быстродействие, удобство общения (свойство эргономичности), мобильность (свойство переносимости из одной среды в другую) и др.

В зависимости от требований к условиям функционирования объекта (работать с большим быстродействием в условиях реального времени или с большой точностью в условиях космических, военных систем и т.п.) требования к его свойствам отличаются и влияют на организацию процессов разработки систем.

С точки зрения теории автоматов любой объект имеет начальное (исходное), промежуточное и конечное состояния. Начальное состояние – это исходная модель объекта (эскиз опытного образца). Промежуточное состояние – это изменённое состояние, отличное от начального и конечного состояний объекта, полученное на определенном этапе ЖЦ под воздействием соответствующих данному этапу программных методов и средств.

Промежуточным состоянием объекта, например, является эскизный, технический и рабочий проекты.

Конечное состояние объекта – это программный продукт, готовый для исполнения требуемых от него функций. Конечным состоянием считается опытный образец, который передаётся в опытную эксплуатацию либо на тиражирование (производство).

В качестве аппарата задания начального состояния объекта используются языки спецификации (SADT, SSADM, PSA, PSI и т.д.), а для задания промежуточных состояний используются языки программирования (ЯП).

Метод разработки (программный метод) – это способ или планомерный подход к достижению той цели, которая ставится перед объектом разработки.

Наиболее распространенные методы проектирования и разработки: нисходящий, восходящий, модульный, метод расширения ядра, метод сборочного программирования. Метод определяет стратегию проектирования и разработки.

При нисходящем (сверху вниз) проектировании после определения требований к объекту формируется его функциональная и системная архитектуры, при которых декомпозированы все задачи ПрО и построена модель объекта. Задачи, в свою очередь, развиваются до понятий и функций объекта, выражаемых в базовых терминах рассматриваемой ПрО. Каждая функция объекта разрабатывается последовательным уточнением до определения элементов системной архитектуры.

Восходящий метод (снизу верх) является обратным нисходящему и начинается с определения элементарных базовых понятий ПрО и формирования из них более крупных понятий, приводящих в конечном итоге к определению некоторой функции ПрО. Для сформированной функции подбирается или вновь разрабатывается программный элемент (модуль, модель программы, макрос, заготовка и т.п.). В этом плане данный метод является методом от готового.

Развитием восходящего метода является метод прототипирования, при котором для объекта вначале создаётся «грубый» его прототип из готовых компонентов, от которых требуется, чтобы они соответствовали функциям объекта без учета эксплуатационных характеристик. Цель прототипирования состоит в том, чтобы отработать «каркас» объекта и его функциональные возможности, а затем улучшать характеристики свойств программных элементов.

Метод расширения ядра характеризуется начальным выделением множества вспомогательных функций. Наиболее эффективным методом выделения является метод, использующий анализ данных и определение

модулей, обрабатывающих различные информационные структуры. Для этого может применяться метод Джексона, в основе которого лежит принцип соответствия организации программы этапам преобразования обрабатываемых данных.

Несмотря на различие этих методов в стратегиях проектирования, общее, что их объединяет – это модульное или блочное (программное) представление объекта разработки. Метод модульного программирования обеспечивает декомпозицию задач ПрО на отдельные функции, вплоть до элементарных, каждой из них сопоставляется отдельная программа пли модуль. Каждый модуль должен обладать интерфейсом для его связи с другими модулями и программами. Применение данного метода предоставляет по сравнению с другими значительные преимущества в плане организации и управления разработкой, но вместе с тем требует создания определенных механизмов языкового и программного характера для решения проблем интерфейса при сборке модулей в более сложные программные структуры. Расширением данного метода является метод сборочного программирования, обеспечивающий интеграцию программных элементов различной степени сложности.

Таким образом, указанные методы проектирования и программирования взаимосвязаны, они используют друг друга. Так, при применении восходящего метода проектирования для систем обработки данных на ранних этапах ЖЦ программного объекта используются методы модульного программирования, расширения ядра и др.

Технологический процесс (ТП) – это взаимосвязанная последовательность операций, выполняемых при разработке объекта. Процесс предназначен для перевода объекта из одного состояния в другой соответствующими программными методами и средствами.

Со временем сформировался набор типовых функций автоматизируемых ПрО (СОД, АСНИ, САПР, АСУ и др.) [8]. Им соответствуют типовые ТП, которые вместе со специализированными образуют линию программ функционально-ориентированного типа для каждой системы.

Технологическая линия (ТЛ) задает набор ТП разработки функций объекта, представленных совокупностью технологических операций (ТО), которые последовательно и систематически преобразуют состояния объектов, включая заключительное его состояние – готовый программный продукт. Особенность линии – отражение определенных функций ПрО, реализуемых в виде программ с заданными показателями качества.

Для одиночных программ количество операций в ТП меньше, чем для сложных. Независимо от сложности программного объекта основным условием выполнения процессов является их автоматизация. Процессы имеют модельное представление жизненного цикла объекта разработки.

Динамика изменения объекта может быть представлена в ТЛ последовательностью операций, переводящих объект из одного состояния в другое путем использования средств автоматизации операций процесса. Каждая конкретно разработанная ТЛ определяет множество допустимых состояний объектов, а на средства их автоматизации возлагается функция перевода и слежения за переходом в недопустимые состояния.

Инструмент – это программное, языковое или методическое средство, применяемое для получения состояния объекта в некотором законченном виде. В зависимости от этапа жизненного цикла и состояния объекта для работы с ним могут быть языки, трансляторы, генераторы и т.п.

Процесс преобразования модели объекта из одного состояния в другое не является полностью автоматизированным и имеется ряд CASE-систем, специально предназначенных для реализации конкретных ПрО.

Управление разработкой. Каждая инженерная дисциплина базируется на принципах и методах конструирования (разработки) и промышленного производства продуктов, которые затрагивают как организационные, так и управленческие аспекты производства. Основными вопросами управления разработкой объектов как инженерии ведения разработки являются:

– организация коллектива разработчиков (состав, структура, квалификация и др.);

– планирование работ, трудозатрат и обеспечение роста производительности труда;

– контроль хода разработки и оценка проектных решений в ходе разработки ПП;

– экономические вопросы (стоимость, ценообразование, стимулирование и др.);

– управление качеством.

Жизненный цикл – это совокупность последовательных процессов и всех действий (операций), связанных с превращением программы в продукцию производственно-технического назначения, начиная с анализа потребностей в автоматизации функций ПрО до создания ПС и кончая его моральным износом.

С ЖЦ связаны сроки и период разработки и эксплуатации ПС.

По срокам эксплуатации ПС разделяются на два класса:

– с краткосрочной эксплуатацией;

– с долгосрочной эксплуатацией.

К краткосрочным ПС относятся продукты научного творчества студентов, аспирантов, а также стажёров, занимающихся их созданием для проверки научной идеи. ЖЦ таких продуктов включает этапы проектирования и разработки. Этапу эксплуатации ПП соответствует процесс апробации созданного продукта.

Программным объектам второго класса соответствуют регламентированные процессы проектирования, разработки и эксплуатации. Они создаются в проектных и промышленных организациях, занимающихся реализацией на ЭВМ крупных народнохозяйственных задач. Объекты данного класса изменятся в диапазоне 100–110 тысяч команд, строк операторов ЯП со свойством изменяемости и модификации при их сопряжении и использовании. Такие объекты тиражируются и вместе с документацией передаются потребителю, отчуждаясь от разработчика. Срок жизни таких программ 10–20 лет, 70–80% этого срока приходится на эксплуатацию и сопровождение.

Программы с долгосрочным характером эксплуатации требуют применять в процессе разработки индустриальные методы организации и планирования разработки. На первое место выступают задачи экономического характера, способствующие повышению производительности, качества ПС и уменьшению сроков разработки. Главная задача – оценивание затрат на разработку ПС и установление конечной стоимости ПП.

ТЛ ориентированы на производства ПП и требуют технико-экономического обоснования процессов разработки, достижения высокого качества ПС, методов оценивания стоимости ПС и учета человеческого фактора в процессе разработки ПС [17].

Предложенная Б.У. Боемом [6] модель стоимости (COCOMO) позволяет провести оценку стоимости и затрат на этапах ЖЦ с учётом ряда стоимостных атрибутов, показателей квалификации исполнителей, степени использования современных методов программирования и др.

Основные элементы этой модели:

– соотношение текущих и будущих расходов на изготовление ПП;

– эффективность ПС;

– методы расчета доходов и чистой стоимости;

– способы оценивания стоимости, основанные на планировании ресурсов, уточнения требований, учета плановых сроков и др.;

– методы экспертных оценок свойств ПП, рассматриваемых как факторы (надежность, сложность, быстродействие и др.), влияющие на стоимость [22];

– средства создания ПС (стили и методы программирования, инструменты и др.).

Модель жизненного цикла ПС. Процесс разработки ПС определяется моделью ЖЦ, этапы которой будем отождествлять с ТП. Для каждого ТП рассматриваемого ПС фиксируются его начальное (S_0) и промежуточные (S_j) и конечное (S_R) состояния. Переход объекта из состояния S_j в S_{j+1} из $ТП_j$ обеспечивается выполнением ТО, входящих в этот ТП. При этом для каждого типа ПС может быть свой набор ТП и состояний S, определяемых

на соответствующем множестве исходных данных, характеризующих эти состояния [18]. Известны такие модели: каскадная, спиральная, итерационная и др. Каждая модель задает схему разработки программы и ПС.

Схема проектирования соответствует методу проектирования ПС сверху вниз, при которой понятия $(i + 1)$-го уровня ($ТП_{i+1}$) описывается через элементарные понятия этого уровня и представляют собой данные, входящие в класс понятий состояния ПС. При этом для каждого типа ПС не исключаются переходы с одного уровня абстракции на другой (сверху вниз и снизу вверх).

Основная цель выделяемых $ТП_i \subset ТЛ$ состоит в получении некоторого полуфабриката ПП (S_i – состояние ПС), фрагментов документации для проведения экспертизы уровня качества в соответствии с моделью качества $М_{кач}$. Каждый ТП модели ЖЦ в общем виде определяет состояние элементов ПС, состав технологических операций, обеспечивающих преобразование исходного состояния ПС и получение его конечного состояния. Таким образом, общая технологическая модель (схема) $М_{пр}$ процесса разработки является отражением модели ЖЦ, способов преобразования состояний ПС и представлена в виде

$$М_{пр} = (S, ТП_i, (ТО_j, ТМ_j), i = \overline{0,7}, j = \overline{1,k}.$$

Множество состояний S рассматриваемой модели включает:

S_0 – исходное (начальное) состояние – описание требований заказчика, предъявляемых к ПС;

S_1 – состояние, включающее набор элементарных состояний, а именно описаний функций (S_1^1), архитектуры (S_1^2), структуры данных (S_1^3) и т.п. средствами языков спецификаций L_1;

S_2 – соответствует техническому проекту и включает описание в классе языков L_2 алгоритмов функции (S_1^2), данных (S_2^2), интерфейсов (S_2^3), гипертекстов документации (S_2^4), оценочных элементов модели качества (S_2^5) и др.;

S_3 – соответствует рабочему проекту и включает описание в ЯП (класс L_3) текстов программ (S_3^1), модулей (S_3^2), тестов (S_3^3) и т.п.;

S_4 – соответствует отлаженным элементам программного продукта;

S_5 – это ПС после сборки объектов;

S_6 – состояние ПС, соответствующее ПП, которое испытывается, проверяется на соответствие заданным функциям и значениям показателей качества;

S_7 – состояние ПС на процессе сопровождения.

Состояние объекта S_i определяется набором частичных его состояний $S_i^1, ..., S_i^k$, подаваемых на вход $ТП_i = \{ТО_k\}$. Этот набор для ПС конечен.

Управление процессом преобразования состояний объекта S_0 в S_k может быть задано в виде графовой модели (или маршрута), в вершинах которой находятся процессы (или операции), а ребра задают всевозможные переходы из одного состояния объекта в другое. Графовая модель отображает способ ведения разработки с распараллеливанием работ и возвратами (при ошибках) в предыдущие процессы.

Каждый частичный процесс ТП является абстрактным *конечным автоматом*, граф которого совпадает с технологическим маршрутом процесса, множество состояний S_i которого определено на совокупности операций $\{\text{ТО}_i\} \subset$ ТО данного процесса.

Задача выполнения i-го процесса заключается в переводе автомата из некоторого промежуточного состояния объекта в другое S_{i+1} с выполнением операций преобразования, качественной или количественной оценки результата разработки объекта и формирования фрагментов эксплуатационной документации (ЭПД).

Таким образом, рассмотренные принципы и метод разработки ТЛ для сборки элементов процесса предопределили специфику языка формального описания ТЛ, а также структуру модели качества, форматов проектно-технологических и эксплуатационных документов.

3.3. Принципы программной инженерии

К принципам программной инженерии отнесены: принцип производственной организации, принцип обеспечения технологичности, принцип планирования трудозатрат на разработку программ и систем [17].

Принцип производственной организации. При технологической подготовке (ТПР) ТЛ проводится анализ ПрО, выбор подходящих средств описания функций ПрО, корректировка и выбор ограничений на проект, а также определение инженерных методов для программирования прикладных программ. Этот принцип отличается строгой последовательностью ТП и ТО, представленных как производственные процессы. В них имеется специализация операций применяемых методов и инструментов, а также видов исполнителей, деятельность которых учитывается в соответствии с экономическими методами оценки труда [1–6]. Необходимым условием успешного применения данного принципа является планирование программных работ, управление которыми осуществляет технологическая служба предприятия, выполняющая контроль соблюдения технологии и оценку объекта разработки, планирование работ с элементами нормирования, совершенствование технологических процессов и др.

Принцип планирования. Для соблюдения сроков разработки программного объекта во многих проектах проводится планирование. Известно около 20 различных моделей планирования, включающих расчет

трудозатрат, сроков и требуемого числа специалистов. Каждая модель разрабатывалась индивидуальным способом на основе использования накопленных данных о проведенных разработках. В качестве исходных данных и критериев в моделях применяются разные характеристики продукта и среды разработки. Большинство методов основано на прогнозировании объема продукта, выражаемого в числе строк (операторов, команд).

Предполагаемый объем делился на среднестатистическое значение производительности (число строк) одного программиста в год. В результате определялась планируемая трудоемкость. Производительность разработчиков ПС колеблется в больших диапазонах в зависимости от применения ЯП высокого уровня (производительность повышается в 4–5 раз по сравнению с языком типа Ассемблер), форм организации работ в коллективе, режима работы программистов на ЭВМ, производительность которых повышается на 20% при диалоговом режиме доступа к ЭВМ.

Для определения числа разработчиков планируемая трудоемкость делилась на время разработки, которое определялось в зависимости от заданных сроков разработки объекта. Естественно, что такая модель расчета является поверхностной и, кроме того, зависит от организации управления разработкой и требований к созданию высокого качества ПП. Данное обстоятельство приводит к увеличению сроков разработки и к снижению производительности труда, что в конечном итоге увеличивает общие трудозатраты.

В производственных условиях проводится *текущее планирование работ*, при котором решается задача составления выполнимого (достижимого) плана работ Y по ТЛ, основными данными его являются:

– общий плановый срок разработки $[t_0, T]$, где t_0 и T – начальный и конечный сроки разработки;

– объем работ $W = \{W_i\}$ с учетом переделок;

– требуемые ресурсы $R = \{R_l, R_m\}$, где R_l – людские; R_m – материальные (технические и программные);

– нормы потребления людских ресурсов по ТП $\left(i = \overline{1, N}\right)$, NR_l и др.

Формально план работ записывается в следующем виде:

$$Y = Y(t_0, T, W, R_l, R_m, NR_l, f),$$

где f – случайные факторы (ошибки при выполнении плановых работ на ТП, сбои технических средств и др.), а также факторы, связанные с появлением средств новой техники и программного обеспечения.

В качестве механизма управления разработкой может использоваться сетевой план-график работ и контроль его выполнения. В условиях производственной организации ведения разработки на основе ТЛ его отсутствие зачастую приводит к проблемам управления разработкой.

Принцип обеспечения технологичности. Понятие технологичности целиком и полностью связано с наличием технологии и с полным соблюдением всех ее требований и правил. Технологичность – это понятие, включающее технологичность ПП и технологичность процесса разработки.

Технологичность ПП – это соответствие ПП потребительским свойствам и определенным функциям ПрО. Ее обеспечение основывается на заложенных в ТЛ элементах типизации, унификации и стандартизации конструктивных элементов ПП, применяемых моделях и заготовках, а также готовых, повторно используемых программных объектах из фондов коллективного пользования. Технологичность определяет способность ПП к эксплуатации.

Технологичность разработки – это регламентированный (упорядоченный набор процессов, операций и процедур их выполнения), конструктивный (методом сборки из готового) и инструктивный (методическое и инструктивное обеспечение процессов ТЛ) порядок работы, а также организация управления разработкой ПП. Ее обеспечение основывается на применении эффективных методов ведения разработки ПП, воплощенных в ТЛ (или ТП) и направленных на повышение качества и производительности труда, минимизации затрат и времени на разработку ПП [1, 6].

Принцип планирования трудозатрат. Исходя из результатов исследований [8], основное распределение трудозатрат на процессах разработки приходится на сопровождение и поддержку проекта.

Поэтому работы по планированию и нормированию в условиях производственной организации являются актуальными.

Взятый за основу подход разработки программ по ТЛ позволяет определить трудозатраты (в человеко-днях) на разработку программ исходя из следующих исходных данных:

N – количество процессов на ТЛ;

J_i – количество операций на ТП$_i$ $\left(i = \overline{1, N}\right)$;

$\sum_{i=1}^{N} P_i + P$ – директивная продолжительность разработки всей ПС, где P_i – минимально требуемая продолжительность (в днях) для i-го процесса; P – резерв времени, который остается до планового срока и используется для варьирования исходными продолжительностями процессов;

x_{ij} $\left(i = \overline{1, N}\right)$, $j = \overline{1, J_i}$ – объем ПС (в операторах), задаваемый экспертно. ПС должно разрабатываться на i-м процессе и j-й операции;

T_{ij} – производительность труда (в операторах в день) при выполнении j-й операции на процессе i.

За искомые величины примем: γ_i – оптимальная продолжительность i-го процесса, m_i – количество программистов, необходимых дли выполнения j-й операции на i-м процессе. Тогда $\sum_{j=1}^{j_i} m_{ij}$ задает количество исполнителей, необходимых для разработки ПС на i-м процессе, $F = \max \sum_{i=1,N} \sum_{j=1}^{j_i} m_{ij}$ – максимальное количество специалистов, необходимых для реализации всей прикладной системы на основе ТЛ.

Исходя из введенных обозначений, математическую модель задачи планирования разработки ПС определяется следующим образом:

$$F^0 = \max \sum_{i=1,N} \sum_{j=1}^{j_i} m_{ij} \Rightarrow \min,$$

$$x_{ij} \le \gamma_i m_{ij} T_{ij}, \ \gamma_i \ge P_i, \qquad (1)$$

$$\sum_{i=1}^{N} \gamma_i \le \sum_{i=1}^{N} P_i + P. \qquad (2)$$

Величины x_{ij}, T_{ij}, P_i, P известны, а γ_i и m_{ij} – целевые переменные. Ограничения на m_{ij} выражаются в виде $m_{ij} > 1$.

Формула (1) означает, что x_{ij} – количество операторов в ПП, должно быть создано m_{ij} специалистами за γ_i дней, а неравенство (2) означает, что разработка ПП должна быть выполнена в плановый срок.

Задача имеет много вариантов с большим перебором, требующих некомбинаторных методов решения, поэтому делается переход от этой математической модели к задаче линейного программирования путем изменения критериев и линеаризации ограничения (1). Для этого в F^0 потребуем приведения всех сумм $\sum_{j=1}^{j_i} m_{ij}$ к минимально возможной величине. В этом случае

$$F^0 = d_1 \sum_{i=1}^{N} \left(\sum_{j=1}^{j_i} m_{ij} - \frac{\sum_{i=1}^{N} \sum_{j=1}^{j_i} m_{ij}}{N} \right) + d_2 \sum_{i=1}^{N} \sum_{j=1}^{j_i} m_{ij} \Rightarrow \min, \qquad (3)$$

где d_1, d_2 – управляющие параметры.

Введем дополнительные переменные $y_i \geq 0$ и $z_i > 0$ такие, что

$$y_i - z_i = \sum_{j=1}^{j_i} m_{ij} - \frac{\sum_{i=1}^{N} \sum_{j=1}^{j_i} m_{ij}}{N}. \quad (4)$$

Тогда после линеаризации (1) получим

$$x_{ij} - \gamma_i^0 m_{ij} T_{ij} - m_{ij}^0 \gamma_i T_{ij} \leq -\gamma_i^0 m_{ij}^0 T_{ij}. \quad (5)$$

Здесь γ^0, m_{ij}^0 соответствуют начальным приближениям, γ_i^0 – управляющим параметрам, которые выражены следующим соотношением:

$$m_{ij}^0 = \frac{x_{ij}}{\gamma_i^0 T_{ij}}.$$

Исходя из этих предположений, получаем задачу линейного программирования:

$$F^0 = d_1 \sum_{i=1}^{N} \left(\sum_{j=1}^{j_i} m_{ij} - \frac{\sum_{i=1}^{N} \sum_{j=1}^{j_i} m_{ij}}{N} \right) + d_2 \sum_{i=1}^{N} \sum_{j=1}^{j_i} m_{ij} \Rightarrow \min,$$

$$x_{ij} - \gamma_i^0 m_{ij} T_{ij} - m_{ij}^0 \gamma_j T_{ij} \leq -\gamma_i^0 m_{ij} T_{ij}, \ \gamma_i \geq P_i, \quad (6)$$

$$\sum_{i=1}^{N} \gamma_i \leq \sum_{i=1}^{N} P_i + P,$$

$$m_{ij}^0 = \frac{x_{ij}}{\gamma_i^0 T_{ij}}, m_{ij} \geq 1,$$

$$\frac{\sum_{i=1}^{N} \sum_{j=1}^{j_i} m_{ij}}{N} - \sum_{j=1}^{j_k} m_{ij} - y_i + z_i = 0.$$

Решение этой задачи состоит в получении γ_i, m_{ij} (нецелочисленных), с помощью которых определяются целочисленные значения базовой модели, как пробного варианта решения задачи планирования трудозатрат.

Варьируя значениями управляющих параметров d_1, d_2 и γ_i, можно получить окончательное решение задачи планирования, т.е. искомое число

специалистов, необходимых для разработки ПС по заданной технологической линии и в заданный срок.

3.4. Определение ПИ с научной точки зрения

Программная инженерия – это наука, связанная с построением целевых объектов путем планирования, управления работами специалистов, оценивания трудозатрат, стоимости и вклада каждого из них в достижение правильности и качества ПП [9–13].

С инженерной точки зрения в ПИ решаются задачи формирования требований, разработки и сопровождения построенного ПП для массового воспроизводства, а также проверки операций базового процесса на правильность реализации проекта, ответственности за выполнение в заданный срок, стоимость и функциональность.

Программная инженерия определяется как наука инженерного построения программных продуктов [9, 11].

Характеристика ПИ как науки

В отличие от математической науки, целью которой есть получение новых знаний, знание в программной инженерии – это способ построения и получения некоторой пользы от созданной программы.

Наука ПИ – это теория, методы и средства построения программ. Построение целевых программных объектов – это процесс анализа автоматизируемой предметной области и образования выходного кода для выполнения на компьютере.

Основа науки ПИ – теория алгоритмов, математическая логика, теория вычислений, методы искусственного интеллекта, теория управления и др. Наука ПИ включает:

– понятия и исходные объекты;

– теории, методы программирования и доказательства (верификация, валидация, тестирование) правильности, оценки надежности и качества ПП;

– средства и инструменты.

Понятия и объекты ПИ. К ним относятся: данные и их структуры, функции и композиции, базовые объекты (модуль, компонент, каркас, повторно используемый компонент – КПИ и др.), целевые объекты (ПО, программная система, семейство систем, ИС, программный проект, ПП и др.).

Разработка простых объектов и программ осуществляется с помощью элементарных операций формального их описания, а изготовление из них целевых объектов осуществляется путем применения инженерных методов проектирования и управления коллективом, их работами, сроками и стоимостью.

Дадим определение целевым объектам ПИ.

Программная система (Application) – комплекс интегрированных программ и средств, которые реализуют набор функций некоторой ПрО в заданной среде. В нее могут входить: ПО, прикладные системы (зарплата, учет и др.), общесистемные компоненты (транслятор, редактор, СУБД и т.п.), системы защиты, обеспечения безопасности функционирования и др. *Способ изготовления* – инженерия ПС (application engineering), которая включает процессы ЖЦ, методы разработки, управления, оценивания ПП и процессов их усовершенствования.

Программное обеспечение – совокупность программных средств, которые реализуют функции компьютерной системы (или функции аппаратно-программной системы), включая общесистемные средства (например, ОС, СКБД, системы защиты и т.п.), подсистемы контроля (показателей процессов, обработки сигналов) и прикладные программные системы. Так, например, функции некоторой ОС – это управление задачами, программами, данными и т.п. ПО может входить в состав ПС или быть идентичным функциям ПС. *Способ изготовления* – инженерия реализации задач ПО.

Семейство систем (systems family) – совокупность ПС с общим (неизменным для всех членов семейства) и управляемым изменяемым набором ее характеристик, которые удовлетворяют определенным потребностям прикладной области (домену). *Способ изготовления* – инженерия домена (Domain Engineering) или конвейерное производство однотипных ПП с единой схемой на основе каркаса и специально разработанных отдельных членов семейства и других готовых ресурсов с помощью базового процесса или линий продукта (Product line).

Программный проект (Program project) – уникальный и интегрированный продукт, в котором отображены совокупности реализованных целей, задач и результатов деятельности, удовлетворяющих требованиям заказчика проекта. *Способ изготовления* – инженерия разработки и менеджмента проекта.

Сложные программные объекты – совокупность взаимосвязанных целевых объектов разных типов, которые выполняют необходимые функции системы, представленные вновь разработанными объектами или выбранными с репозитария готовых ресурсов и КПИ.

3.5. Компьютерные технологии разработки ПС и ИС

В рамках новой дисциплины Computer Science за рубежом определены следующие технологии: компьютерная инженерия (Computer Engineering); системная инженерия (System Engineering) и программная инженерия (Software Engineering), информационные системы и информа-

ционные технологии. Среди этих технологий центральное место отведено ПИ (рис. 1). Она задает парадигмы, теории и концепции для каждой технологии Computer Science, определяет процесс разработки каждой из них и применение, конфигурацию и развертывание созданного для них ПО.

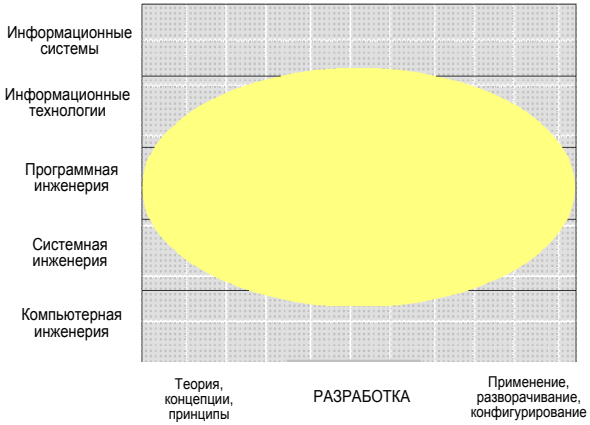

Рис. 1. Место программной инженерии в информатике

Сущность этих инженерных технологий приводится ниже.

Компьютерная инженерия – это теория и принципы построения компьютеров, фреймворков, суперкомпьютеров, вычислительных кластеров и их системного программного обеспечения. Основы этой инженерии – теории Тьюринга, фон Неймана, автоматов, алгоритмов и концепций кибернетики, разработанных В.М. Глушковым [8, 20]. Она использует математику, логику, теории анализа и систем. С помощью этих средств строятся компьютеры, фреймворки, многопроцессорные, макроконвейерные машины, устройства, микросхемы и т.п.

Системная инженерия – это теория, методы и принципы построения информационных, компьютерных АС и систем управления ими. Она представляет собой междисциплинарный подход, объединяющий теоретические положения, методы и средства, направленные на создание и объединение системных решений для поддержки сложно организованных объектов и систем. Данный подход базируется на технологиях компьютерных систем для разных областей применений и новых средствах управления ИС (ОС, БД, СУБД и другие). Системная технология включает теорию АСУ и информатики Глушкова [8, 13, 20], а также математику, компьютерные науки и методы, которые применяют в экономике, финансовой деятельности и других. Системная инженерия или технология программ-

ных систем получили развитие при создании АС различного назначения разными международными организациями.

Программная инженерия – это система методов, способов и дисциплин планирования, разработки, эксплуатации и сопровождения ПО, предназначенного для его промышленного производства (www.swebok.com). Она охватывает все аспекты создания ПО от начала формулирования требований и до разработки, сопровождения и окончательного списания его. Теоретическую основу программной инженерии составляет: теории алгоритмов, теория программирования, методы вычислений и распределенных коммуникаций [12–25]. Массовое сопровождение готовых ПП основывается на теориях менеджмента, планирования процессов и необходимых для этого ресурсов, верификации и тестирования, оценивания рисков и качества [23, 24]. Технологии создания ПП развиваются в направлении создания и совершенствования программных и информационных технологий.

Информационные технологии (ИТ) с 1990-х годов получили широкое развитие в современных корпорациях, предприятиях и государственных органов управления. Они обеспечивают решение задач, связанных с обработкой информации, в том числе размещенной в глобальных сетях. Для разработки таких технологий подготавливаются высококвалифицированные ИТ-специалисты в университетах. Цели и задачи информационных технологий В.М. Глушков сформулировал в своей монографии [8]. Они и сегодня имеют большое значение. Сегодня ИТ-технологии широко представлены в сети Интернет для широкого применения.

Информационные системы – это компьютерные системы обработки разнообразной информации, которая относится к различным сферам деятельности человека. В их числе бухгалтерский учет, документооборот на всех уровнях управленческой деятельности [17, 21]. Информационные системы предоставляют средства для управления и обработки информации с целью обеспечения эффективной работы разных организаций. В настоящее время информационно-поисковые и аналитические системы стали главным инструментом поиска и отбора, накопления и анализа различных информационных ресурсов для их массового использования.

Информационные системы, технологии и дисциплины CS пространства информатики наполняются инженерными методами Software Engineering, которые обеспечивают технологию программирования сложных систем из готовых ресусов средствами новых теорий, парадигм, методов и средств конвейерной разработки, их конфигурирования и развертывания для выполнения в современных гетерогенных средах.

3.6. Индустриальные основы программной инженерии

Термин **индустрия** определяет производство разных видов продуктов массового применения и средств их изготовления промышленными предприятиями, фирмами и корпорациями, в нашем случае – программно-технологического типа. Главным вопросом любой индустрии является не только выпуск соответствующей продукции, но и получение прибыли от нее. Сейчас большие прибыли от выпуска ПП получают такие мировые фирмы, как Microsoft, IBM, Corba, Intel и др., а также индийские фирмы по адаптации устаревших (legacy) систем программ. У нас за последние десятилетия развитие индустрии ПП фактически развивалась коммерческим способом. Государственные и научные программы ее поддержки в основном отсутствуют. Вместе с тем действуют предприятия коммерческого типа, которые выполняют работы по выработке не массовых ПП, а на заказ разных фирм, которые финансируют их.

Под **производством ПП** понимаются процессы ТЛ, посредством которых исполнители используют соответствующую теорию и инструментальные средства для выработки ПП массового применения. Эти линии из процессов ориентированы на разработку ПП. Примером являются развитые технологии – Engineering application, family, domain, а также сформированные средства обновления устаревших систем в оффшорных организациях (например, в Индии). В настоящее время для поддержки процессов изготовления ПП используются системные сервисные средства (ОС, общесистемные сервисы горизонтального и вертикального типа, новые языки, трансляторы, редакторы, компоузеры и т.п.), готовые программные ресурсы, КПИ и методики проведения робот, к которым относятся предложенные нами *дисциплины ПИ.* Они определяют разные аспекты деятельности по производству ПП. Это научные, инженерные, управленческие, экономические, производственные дисциплины, которые выполняют квалифицированные профессиональные специалисты в рамках системных средств разработки ПП и фабрики программ [7–9].

Сущность производства продукции. Производство продукции – это процесс создания материальных благ, необходимых для удовлетворения индивидуальных и социальных потребностей некоторых слоев общества. Продукция производится из готовых элементов и ресурсов.

От объема использованных ресурсов зависит уровень производства некоторой продукции и вычисляется с помощью функцией вида [6]

$$v = F\left(z, u\right),$$

где $v = (v_i)$ – вектор выпуска продукции, $z = (z_i)$ – вектор расходов ресурсов, $u = (u_i)$ – матрица параметров зависимости функции расходов $z = F\ (w_j, u)$, $j = 1, 2, ..., n$. Показатели функции w_j отвечают объему про-

изводства, структуре производственных фондов и уровню специализации фабрики программ.

Общая производственная функция Кобба–Дугласа выпуска продукции имеет вид

$$v = ne^{rt} L^{\alpha} K^{\beta},$$

где v – обобщенный выпуск, n – нормативный множитель, e – основа натурального алгоритма, t – показатель уровня научно-технического прогресса, L – расходы человеческого труда, K – величина капитала α, β – коэффициенты эластичности.

Расчеты этих функций позволяют определить соответствующие оценки доходной части, в частности на фабрики программ. Экономичность, системность и пропорциональность выпуска ПП зависит от принципов организации и управления ресурсами, номенклатуры продукции, применения компьютеров и служб фабрики (контроля, тестирования, оценивания ПП и др.).

Фабрика как структура производства продукции базируется на линиях, оборудованных набором средств, комплектующих «деталей» – готовых ресурсов, инструментов и сервисов для автоматизированного выполнения процессов на этих линиях в конкретной операционной среде. То есть ИТ дают набор инструментов для перехода от разработки отдельных продуктов к индустрии сложных программных и компьютерных систем с повышением производительности создания отдельных элементов на процессах ЖЦ. Линии разработки для разных продуктов реализованы, например, в среде MS.Net с использованием каркасов, языка DSL (Domain Specific Language), WorkFlow и др. Элементы фабрики накапливаются в разных библиотеках и репозитариях и используются при сборке из них сложных ПС.

Методика производства ПП. В ядре SWEBOK нет дисциплин, связанных с производством программной продукции на фабриках программ, принципами комплектации фабрики техническими и человеческими ресурсами, методами создания необходимых *продуктовых линий*, служб поддержки средств и управления соответствующими специалистами. В индустрии ПП недостаточно решены проблемы оценивания сложности объектов и процессов изготовления ПП на *линиях*. В случае создания больших систем еще не реализуются методы преодоления сложности. В последние годы в мировой информатике сформировалось понятие вариабельности систем и продуктов, которое теоретически и программно направленно на снижение сложности ПП путем конфигурационной сборки больших систем из разных готовых программных ресурсов (КПИ, reuses, assess и др.) [18].

Методическое обеспечение ПИ развивается по пути развития и использования новых дисциплин SE (наука, инженерия, экономика, менеджмент, производство). Эти дисциплины могут выступать в качестве нормативного регламентированного выполнения разных видов работ в процессе разработки КПИ, их сборки и проведении экспертиз ПП, измерения и оценки качества продукта и его отдельных артефактов. Эти научно-технические дисциплины (*Di*) входят в состав ПИ [9, 11]:

$$\text{ПИ} = \{DiSc, DiEn, DiEc, DsMa, DiDe\},$$

где *DiSi* – научная, *DiEn* – инженерная, *DiEc* – экономическая, *DiMa* – управленческая (менеджерская) дисциплины и *DiDe* – производственная.

3.7. Характеристика дисциплин программной инженерии

В [9, 11] дано описание дисциплин ПИ (ДПИ). С помощью этих дисциплин, разные специалисты (аналитики, менеджеры, программисты, тестировщики, анализаторы, оценщики и др), участвующие в производстве ПП, выполняют соответствующие работы. Сформулированы их задачи и цели (рис. 2). В них определены методы и принципы разработки, их сроки, стоимость, производительность и оценка качества по результатам тестирования ПП.

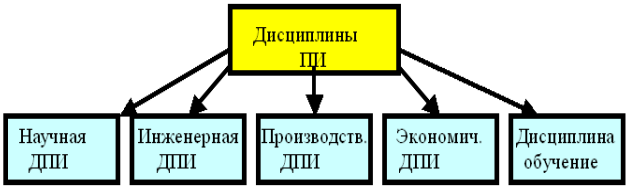

Рис. 2. Набор дисциплин ДПИ программной инженерии

Определим сущность этих дисциплин.

Научная дисциплина

Эта дисциплина основывается на существующих классических науках: теория алгоритмов, теория множеств, теория доказательства, математическая логика, теория программирования, теория абстрактных моделей и архитектур, теория управления и др. Она определена на множестве формализованных базовых понятий и объектов, формальных подходов, методах программирования, теории данных, методах управления (Management Project) изготовления ПП [6–9] и CASE-инструментах.

К *основным понятиям* этой дисциплины относятся типы и структуры данных, функции и композиции, простые и сложные целевые объекты. Разработка простых объектов выполняется посредством их формального описания, спецификации, а составных целевых – посредством применения

инженерных методов проектирования и организации управления процессов их изготовления.

Теория программирования включает следующие методы:

– языки, средства спецификации и проектирования целевых объектов, методы доказательства их правильности (верификация, тестирование);

– формальные методы управления (персоналом, материальными и финансовыми ресурсами) проектом и отдельными его характеристиками;

– методы оценивания промежуточных и конечных результатов проектирования для достижения заданных показателей качества ПП (надежность, корректность и т.п.).

Таким образом, научная дисциплина – это теоретический фундамент ПИ, и учить ему необходимо не только для повышения уровня квалификации будущих компьютерщиков и программистов, но и для поддержки и развития теорий, возможностей и средств программирования, которые усовершенствуют соответствующие направления производства ПИ. Одна из важных научных проблем индустриального производства ПП – это интеграция (композиция, сборка) составных элементов будущего продукта, основанная на совместимости их интерфейсов при сборке этих элементов в сложные программные структуры. Интеграция решается путем фундаментальной теории синтеза и сборки как ветвей научной дисциплины ПИ.

Главным курсом обучения в вузах должна стать научная дисциплина, которую необходимо, на наш взгляд, представить общими дисциплинами теоретического курса, курсами систематического программирования (объектного, компонентного, агентного и т.п.), отдельными классическими курсами программы Curricula-2004 в области ПИ [4], а также CASE-инструментами поддержки методов программирования.

Инженерная дисциплина

Инженерия ПИ – это способы применения теории программирования, технологических правил и процедур, процессов ЖЦ, методик измерения и оценки выполнения деятельности по изготовлению ПП. К ним относятся: набор областей знаний; БПО – схема деятельности; стандарты – набор правил и положений; модели ресурсов проекта; стандарт PMBOK – управление проектом.

В эту *дисциплину* входит совокупность инженерных приемов, средств и стандартов, ориентированных на изготовление целевых объектов ПП с применением научной дисциплины ПИ [9–13]. Базовыми компонентами этой дисциплины являются:

– ядро знаний SWEBOK как стандартный набор методов и средств разработки ПП и управления проектами;

– базовый процесс ПИ как стержень процессной деятельности разработчиков ПП;

– стандарты как набор регламентированных правил конструирования промежуточных артефактов на процессах ЖЦ;

– инфраструктура – условия среды, методическое и организационное обеспечение базового процесса ПИ и поддержки деятельности исполнителей ПП;

– общие системные средства и инструментальные CASE-средства поддержки процессов изготовления ПП.

Инженерия ПП базируется на повторном использовании компонентов КПИ, готовых средствах, ресурсах и инструментах их построения. К таким технологиям относятся: инженерия КПИ (Reuse Engineering), инженерия приложений (Application Engineering), доменная инженерия (Domain Engineering) и инженерия семейств систем (Family of systems Engineering) [19].

Инженерия КПИ сформировалась как систематическая и целенаправленная деятельность по поиску и подбору готовых КПИ, которые размещаются в современных хранилищах (репозиториях или библиотеках) [3, 16]. Базис изготовления из них ПП – каркас, набор вновь реализованных компонентов и готовых функциональных и сервисных КПИ. В технологии изготовления новых ПС могут использоваться готовые прикладные системы (бизнеса, коммерции, экономики, сервиса и т.п.), а также системы общего назначения (трансляторы, редакторы, ОС, СУБД, интеграторы, генераторы и др.).

Инженерия приложений и *инженерия доменов* также основываются на многократном использовании разных КПИ и других программных элементах. Основная задача этих видов инженерной деятельности – построение прикладных систем или семейств систем, которые реализуют задачи приложения или домена с учетом общих и изменяемых характеристик составляющих их элементов (членов семейства). Технология изготовления доменов вплотную подошла к современным принципам конвейерного производства продуктов из готовых «деталей» типа КПИ по модели домена в DSL (Domain Specific Language) и спецификациям каждого члена семейства [19]. Основная суть этой технологии – управление изготовлением ПП, основанное на план-графиках работ, контроле результатов работ и оценивании степени применимости готовых ресурсов в процессе реализации специфических задач домена. Компоненты данной инженерной дисциплины совершенствуются и адаптируются к условиям производственной среды (CMM, SPICE, Trillium и др.).

Таким образом, в процессе исследований научных и инженерных аспектов компьютерных наук создан фундамент инженерной дисциплины, который включает стандартные принципы инженерии, современные языки

спецификации доменов, членов семейств и процессов производства ПП с помощью средств и инструментов инженерных технологий.

Дисциплина управления в ПИ

Базис этой дисциплины – классическая теория менеджмента производства проектов и стандарт IEEE Std.1490 PMBOK (Project Management Body of Knowledge) [11].

Теория управления, а также теория организационного управления разработана академиком В.М. Глушковым (1970 г.) [13, 20] и другими учеными. Эта теория проверена практикой при построении технологических процессов в металлургической, судостроительной и химической промышленностях, а также при внедрении для целей массового производства (например, АСУ «Львов», АСУ металлургическими процессами в ГДР и др.). После смерти Глушкова (1982 г.) математическая теория управления сложными системами развивалась заграницей, в особенности в части теории планирования производством. Так, на фирме «Dupon» с целью планирования и создания планов-графиков больших комплексов работ для модернизации заводов был разработан метод CRM (Critical Path Method), базис которого – графическое представление работ, соответствующих видов операций и времени их выполнения [21]. Другой метод – сетевое планирование PERT (Program Evaluation and Review Technique) – был апробирован при реализации проекта разработки ракетной системы «Polaris», которая объединяла около 3800 подрядчиков с числом операций более чем 60 тыс. Применение этого метода было настолько успешным, что проект был завершен на два года раньше запланированного срока. Каждый из этих методов возник в недрах промышленного производства. Они адаптированы к среде программирования и стали базовыми в индустрии программных продуктов.

Элементы теории управления и планирования нашли отображение в стандарте PMBOK. В нем определены процессы ЖЦ проекта и главные области знаний, сгруппированные по таким задачам, как инициация, планирование, мониторинг, управление и завершение. Основная область знаний этого ядра – интеграция средств управления организационной деятельностью коллектива исполнителей проекта. Она базируется на методах принятия решений для используемых ресурсов, общих задачах проектирования, контроля правильности выполнения проекта и вкладывания в заданную заказчиком стоимость проекта [3–6].

Эти базовые теории управления и планирования, стандартные положения PMBOK, серия стандартов ISO/IEC 9001 (1–4), регламентирующие управление качеством продукта, и соответствующее методическое обеспечение проекта стали базисом дисциплины управления в ПИ. Сформированный курс обучения (Curricula SE – 2014) этой дисциплины будет обес-

печивать подготовку в вузах будущих высококвалифицированных менеджеров проектов и других специалистов в области организационного управления выпуском программной продукции.

Экономическая дисциплина в ПИ

Экономика ПИ является самостоятельной дисциплиной предмета ПИ и связана с экономическими аспектами индустрии ПП. В ее основе лежат методы проведения экономических расчетов разных сторон деятельности на проекте с учетом знаний специалистов всех факторов и затрат в выполняемом проекте. Эта дисциплина имеет свою теорию и практику решения задач по проведению экспертизы проекта, оценке стоимости, сроков и экономических показателей, устанавливаемых в требованиях к ПП при заключении контракта на его разработку [22]. В рамках этой дисциплины проводится оценка требований, проектных решений, рисков, связанных с имеющимися материальными и людскими ресурсами, показателями качества ПП, а также финансовые расчеты с каждым исполнителем на всех этапах выполняемых договоров.

Эта дисциплина наиболее развита с точки зрения методов экономических расчетов в ПИ, а именно: методология прогнозирования размера ПП (FPA – Function Points Analyses, Feature Points, Mark-II Function Points, 3D Function Points и др.); оценки трудозатрат на разработку ПП с помощью семейства моделей COCOMO 1, 2 и ряда других математических моделей оценки трудозатрат на разработку ПП (Angel, Slim, Seer–SEM и др.), а также моделей, связывающих экономические показатели ПП с характеристиками качества [23, 24]. В рамках этой дисциплины могут выполняются следующие расчеты:

– измерение показателей качества продукта и производительности;

– анализ риска создания проектов и процессов разработки;

– сбор данных для проведения оценки стоимости ПП;

– определение размера ПП и др.

На основе этих показателей проводятся оценки трудозатрат на разработку проекта и оценка качества ПП и проведения его сертификации.

При формировании задач дисциплины используются фундаментальные экономические методы, связанные с принципами распределения и экспертизы работ в сложных системах, а также современные методы расчета стоимости отдельных частей систем с помощью данных, полученных при определении размера КПИ и системы в целом. Привлекаются стандарты, обеспечивающие оценку качества и сертификацию готового продукта. Систематизированный и научно обоснованный курс экономической дисциплины ПИ, определенный в Curricula-2014, закроет тот пробел, который имеется при выпуске и производстве ПП.

Производственная дисциплина ПИ

Главный вопрос индустрии – выпуск ПП и получение прибыли.

В области ПИ продукты массового производства, создаваемые известными фирмами Microsoft, IBM, Intel и др., а также результаты аутсорсинга (обновление устаревшего, унаследованного ПО) приносят владельцам большие *прибыли*. Этим подтверждается (в соответствии с толкованием понятия «производство»), что виды ПП этих фирм выпускаются на *индустриальной* основе. Производство ПП базируется на ТП изготовления определенных видов продуктов с применением теории проектирования, методах и инструментальных средствах поддержки выпуска ПП.

Первыми попытками поддержки индустриального производства являются технологическая подготовка разработки (ТПР) для любого производства, а для ПП продуктовые линии продуктов (Product line, Product family) института SEI США [1, 25]. В их задачу входит удовлетворение рыночных потребностей пользователей на некоторые виды программной продукции. Более продвинутые инженерные технологии производства ПП – это инженерия приложений, доменов, семейств систем, а также средства поддержки их производства (ОС, общесистемные средства, новые языки, интегральные среды и т.п.).

ТПР было применено при разработке АИС «Юпитер» для производства программ обработки данных на нескольких объектах проекта автоматизации военно-морского флота СССР. ТПР способствовала разработке шести ТЛ по изготовлению программ обработки данных, решения аналитических задач, формирования БД информационного обеспечения объектов АИС и др.

Производство ПП на упомянутой *линии продуктов* осуществлялось из готовых КПИ и программ, информационных ресурсов БД, средств и инструментов ТЛ, в которую включались необходимые методы разработки, тестирования и оценивания конечного результата. Конструирование на линии выполнялось на основе каркаса ПП и встраивания разработанных или подобранных КПИ. Инструментальная среда их разработки содержит необходимые средства и инструменты производства, а также механизмы отслеживания процесса построения отдельных продуктов АИС в соответствии с установленным заказчиком планом.

Данная дисциплина включает методы и технологии производства разных видов продуктов, методы анализа сложности структуры ПС, средства описания специфических особенностей целевых объектов, оценки готовых ресурсов и применения инструментальных сред, а также языков спецификации этих объектов и стандартов предприятия по документированию готового продукта.

Каждая из приведенных дисциплин – это теоретические основы, понятия, объекты и методы их выполнения при изготовлении ПП в произ-

водственных, фабричных условиях. Особенность идеи производства ПП – использование готовых программных продуктов (reuses, services, assets и др.), используемых как готовые «детали» на складе выпуска продукции в автомобильной промышленности и т.п.

Связывающее звено процесса сборки ПП из готовых продуктов (или КПИ) – это интерфейс, который определяет типы и виды данных, передаваемых между объектами сборки КПИ в ПС или семейства ПС [26]. Для их автоматизации разрабатываются ТЛ, включающие последовательность действий, ТО и форм документов, ориентированных на регламентированное выполнение определенной деятельности при изготовлении ПП и методов оценки разных сторон ПП.

Таким образом, предложенные дисциплины ДПИ детализируются и уточняются при производстве разного рода ПП. Они в будущем станут предметом подготовки студентов для их участия в индустриальном производстве качественных ПП. Ими могут быть такие специалисты: аналитики – *DiSi* Sciences, инженеры – *DiEn* Engineers, экономисты – *DiEc* Economics, руководители – *DiMa* Managers, разработчики – *DiDe* Developer и т.д.

3.8. Фабрики производства программ

Исходя из опыта разработки и анализа технологии программирования и зарубежных вариантов фабрик программ на современных платформах компьютеров установлены общие характеристики, свойственные разным видам фабрик ПП [27–34]. Это, прежде всего, операционные среды (типа SUN ONC, MS.Net, Oberon, Babel, Grid, Eclipse и др.), методы программирования (компонентный, структурный, сервисный и др.), средства (ЯП, Rational Rose, CLR, Z, RAISE и др.) и CASE-инструменты (RUP. UML, MSF, CORBA и др.), а также идей взаимодействия разнородных программ, основанной на теории организации фундаментальных и общих типов данных (Fundamental Data Types – FDT, GDT (General Data Types – ISO/IEEC 11404)). Они поддерживают процессы линий изготовления ПП отдельных компонентов и систем, используют библиотеки готовых продуктов, сервис разных аспектов производства (данные, интерфейс, качество, управление, контроль, планирование, расчет разных затрат и др.). Главный ресурс фабрики – специалисты по производству программ (аналитики, программисты, инженеры, менеджеры, тестировщики, контролеры и т.п.).

Фабрика программ – это интегрированная инфраструктура сборочного производства ПП (компонентов, подсистем, систем, семейств систем, АСУ, АСУТП и др.), предназначенная для выполнения государственных, коммерческих и других заказов на ПП [35]. Фабрика включает комплекс средств, инструментов и сервисов для производства ПП на процессах ТЛ.

Ядро фабрики – операционная среда и метод изготовления ПП (UML, компонентный, структурный, модульный, сервисный и др.). Обязательное условие сборочного конвейера – средства обеспечения связи разноязыковых программ, аналогично реализации в MS.Net. При использовании сервисов в среду вводятся современные средства (веб-сервисы, веб-семантики и др.) для управления выбором и сборкой необходимых ресурсов при производстве ПП. В состав фабрики входят ТЛ или продуктовые линии, набор КПИ и методических средств, а также инструментов и сервисов автоматизированного выполнения процессов на линии изготовления продукта в операционной среде.

Фабрика софтвера – это согласованный набор процессов, средств и разных видов ресурсов для всего цикла создания тех или иных программных артефактов, компонентов, приложений и систем [36].

Линии фабрики обеспечивают переход от ремесла к индустрии ПП, повышают производительность разработки продукта на процессах линии с заданными функциями, архитектурой и качеством. Эти линии содержат соответствующий набор средств разработки простых и сложных ПП из готовых программных элементов. Им соответствует ЖЦ, например, реализованный в среде MS.Net с использованием каркасов (framework), DSL-языков и др. Линии сложных ПП могут быть сборочного типа из готовых программных ресурсов, которые находятся в разных библиотеках и репозитариях Интернета.

Характеристика общих ресурсов фабрики

Технические ресурсы: платформы, процессоры (Intell, IBM, Apple, MS; коммуникации (OSI, TCP/IP; компьютеры пользователей; файлы и серверы; локальные и глобальные сети; электронная почта (e-mail); тестеры и т.п. [18].

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

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

Человеческие ресурсы: группы разработчиков, служб планирования и управления выполнением проектных работ (по планам, сетевым графи-

кам), связанных с достижением качества, выявления рисков, формирования конфигурации, проверки правильности реализации проекта и т.п.

В стандарте ISO/IEC 12207 определены следующие виды групп:

– технико-технологической поддержки (изучения рынка, приобретения CASE, ПП, консультации и т.п.);

– технологической службы (сопровождения, поддержки ЖЦ, контроля и т.п.);

– качества (SQA-группа) с функциями планирования и выполнения ЖЦ, проверки работ, контроля качества рабочих продуктов, документов ПП и т.п.;

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

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

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

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

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

Эти группы необходимы при индустриальном и коллективном производстве ПП, как это есть в крупнейшей системе VS.Net.

3.8.1. *Стандартные ресурсы фабрик*

Комитет по стандартизации ISO разработал ряд стандартов программной инженерии, которые регламентируют порядок разработки ПП, управления методами программирования на фабриках программ [16–18]. К ним относятся следующие.

Базовый процесс (БП) предназначен для «процессного продуцирования» ПП как вида инженерной деятельности по изготовлению ПП. Он включает операции оценки, измерения, управления изменениями и усовершенствованием ПП и БП в соответствии со стандартом ISO/IEC 15504-2007 («Оценивания процессов ЖЦ ПЗ. Установки на усовершенствование процесса»). Оценка зрелости организации или фабрики программ осуществляется с помощью модели зрелости CMM (Capability Maturity Models) [16] института SEI США, модели Bootstrap, Trillium и т.п. Согласно этим стандартам уровень зрелости организации зависит от наличия в

ней ресурсов, стандартов, методик и способностей (зрелости) членов коллектива фабрики изготавливать ПП в заданные сроки и определенной стоимости.

Жизненный цикл стандарта ISO/IEC 12207 «Процессы ЖЦ ПС» регламентирован разными направлениями деятельности по разработке, проектированию и управлению ПП, организации процессов (планирования, управления и сопровождения), измерения, оценивания продуктов и процессов. Наиболее важные из них – серия стандартов: ISO/IEC 14598 «Оценивание программного продукта», стандарт ISO/IEC 15939 «Процесс измерения», серия стандартов ISO/IEC 15504 «Оценивание процессов ЖЦ ПО», базовые стандарты качества – ISO/IEC 9001 (1–4) «Системы управления качеством. Требования», ГОСТ 2844-94, ГОСТ 2850-94 и 9126 – регламентируют аспекты обеспечения отечественного качества ПП.

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

Ядро знаний менеджмента проекта – стандарт по управлению проектом – PMBOK (IEEE Std.1490 «IEEE Guide. Adoption of PMI Standard. A Guide to the Project Management Body of Knowledge»), разработанный институтом PMI [25]. Этот стандарт содержит описание лексики, структуры процессов и областей знаний: *управления содержанием проекта* (планирования с распределением работ); *управление качеством* и контроль результатов на соответствие стандартам качества; *управление человеческими ресурсами* согласно их квалификации и профессионализма.

3.8.2. *Базовые компоненты фабрик программ*

Рассмотренные виды операционных сред позволили определить необходимые атрибуты, свойственные любой фабрике программ. Принятие решения о их полноте и функциональности для производства ПП зависит от наличия финансов и знаний менеджеров, которые намерены заниматься изготовлением ПП определенного назначения. Экспериментальный вариант фабрики программ был сделан в ИПС НАНУ на основе бесплатной системы Eclipse [37], содержащей базовые инструментальные средства для производства ПП из готовых КПИ, а именно:

– механизм плагинов в формате XML (Plug-in Development Environment), который обеспечивает автоматизированное подключение плагинов и новых инструментов (например, Protege, Java, RMI), а также КПИ из репозиториев;

– автоматизированное подключение соответствующих меню к интерфейсу пользователя (иконок, сценариев и т.п.);

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

Эта система дополнена алгеброй операций компонентного программирования, средствами сборки, генерации и конфигурирования КПИ в семейство систем [38, 39]. Метод генерации отдельных элементов моделирует процессы занесения в репозиторий компонентов, КПИ и проведение сертификации, выбора, интеграции и сборки разнородных программных объектов совместно с сгенерированными членами семейства СПС в среде Eclipse.

Исходя из полученной практики автоматизированной сборки разнородных программ в ЯП в среде ОС ЕС и анализа современных и зарубежных фабрик программ в системах (IBM, OMG, Microsoft, Oberon и т.п.) [27–33], сформированы общие составные элементы фабрики производства программ, а именно:

– *готовые программные ресурсы* (артефакты, программы, системы, reuses, assets, КПИ) и т.п.;

– *интерфейс* как спецификатор готовых ресурсов независимо от языков программирования в языке интерфейса (IDL, API, SIDL, WSDL, RAS и т.п.) [6, 7, 26];

– *технологические линии* (*ТЛ*)*, продуктовые линии* (Product Lines) [1, 37–40] производства ПП;

– *сборочный конвейер* фабрики программ;

– *методики и приемы* проведения работ на фабрики программ;

– *операционная среда* разработки программ на фабрики.

Эти составные элементы в основном имеются во всех системах производства программ.

3.8.3. *Современные действующие фабрики программ*

В [40] проведен анализ фабрик программ и дана их характеристика:

– система АПРОП [41];

– система Sun Microsystems (IBM) [42];

– OMA-архитектура, или система CORBA (OMG) [27];

– фабрика «ручной» сборки разноязычных программ Инга Бейя [28];

– фабрики программ по методу UML Дж. Гринфильда [29];

– среда для групповой разработки ПП–MS.VSTS [30];

– инфраструктура вычисления компонентови систем Grid [43];

– фабрика Г. Ленца на основе Use Case [32];

– каркас фабрики программ Авдошина [33] и др.

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

АПРОП – это фабрика, которая работала в среде ОС ЕС и обеспечивала сборку разноязычных модулей в монолитную структуру на ЕС ЭВМ через 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, и располагаются в специальном Банке модулей.

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

CORBA, или OMA-архитектура (Apple, IBM, Win–NT, x-Open, Dec), ее можно считать фабрикой программ с обеспечением связи разноязычных объектов в ЯП (C, C++, Smalltalk, Java, Cobol, Visual, Ada-96) через 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 протоколами TCP/IP, IIOP через брокер ORB, который обеспечивает сервис преобразования несовместимых типов данных разноязычных объектов, устранение неоднородности платформенных данных взаимодействующих объектов клиента и сервера. Эта среда поддерживает связи с другими средами CORBA, OLE/DCOM, SOM/DSOM, IBM OS и т.п.

Фабрика сборки И. Бея. Базисом этой фабрики производства разноязычных программ есть интерфейсный посредник, командные строки, конфигурационные файлы, проверенные в операционных средах (VC++, VBasic, Matlab, Java, Visual Works, Smalltalk и т.п.) для разных платформ (Microsoft.Net, HP, Apple, IBM и т.п.). Были разработаны разные варианты связей пар разноязычных программ в названных ЯП [45]. Они передают данные между собой. Для некоторых типов данных разработаны функции

преобразования, типы которых – нерелевантные или их форматы – зависят от особенностей платформы, механизмов передачи данных (через протоколы, вызовы, интерфейсные карты MIO–16E–2 и др.). Апробированы Domain and Application Model, Model Interconnection, Microsoft Foundation, а также современные визуальные средства (панели, сценарии, иконки и т.п.) для внесения изменений типов данных вызывающей и вызываемой программ.

Фабрика программ Дж. Гринфильда. Для фабрик программ сформулированы методологические и технологические аспекты производства ПП по методу UML с использованием моделей архитектур систем и компьютеров, механизмов интеграции разнородных компонентов с типами данных FDT и описанием интерфейса в языках (IDL, XML, RDF и т.п.). Главные концептуальные объекты производства: *reuse*, которые накоплены в общепринятых хранилищах (библиотеках, репозитариях и т.п.) Интернет и имеют сертификаты качества; *активы* (assets), программы, приложения и системы; *модели*, шаблоны и инструменты UML при реализации ПП на линии производства; *веб-сервисы*, процессы линий; методики измерения и контроля качества ПП и т.п. Анализ моделирования UML на данной фабрике показал следующее: язык UML является языком эскиза ПП, который не допускает использования компонентов reuse и интерфейса на разных ЯП; проблема модификации ПП не решена в визуальном языке UML и др. Фактически автор разработал меморандум современной фабрики ПП, которая базируется на reuses, применяемых на производственных линиях, моделях систем, каркасах процессов и продуктов. Конкретная фабрика строится.

Фабрики программ фирмы Microsoft. Основа данной фабрики – среда MS.Net. Она содержит большое количество средств и инструментов, а именно: готовые ресурсы (компоненты, сервисы) Интернета, языки – JAVA, C++, Basic, Java, Pascal, C#, библиотеки CLR и FCL, сборку кодов (exe, dill) в готовый ПП, веб-обслуживание разного назначения, управление проектом PM-2007 группами разработчиков ПП в виртуальной среде VSTS, MSF, которые могут работать по проекту, располагаясь в разных точках мира. В состав средств автоматизации входят: пакет инструментов VSTS-2005 (Visual Studio Teams Systems); MSF (Microsoft Solution Architecture) для построения производственной архитектуры предприятия; управление группами разработчиков по стандарту PMBOK; Professional Studio и Foundation Server для поддержки процессов проектирования, кодирования тестирования, формирования версий ПП (SDLC, IDE, MS Office, MS SQL server, MS Visual Studio 2007); модель усовершенствования процессов (CMMI Process Impovement), в частности процесса сборки, который использует CLR-библиотеки (Common Language Routine), классы,

FCL-типы, трансляторы, General code (exe), Portable Executable code и т.п. В среду включены средства определения сроков работ, трудозатрат, оценки показателей качества изготовления разных видов ПП.

Фабрики для вычисления задач в Grid. Европейский проект Grid ориентирован на организацию распределенных вычислений задач в разных научных направлениях (физика, математика, медицина, биология и др.) [43]. Он включает ряд подпроектов: Gcube – операционная среда, ETICS – как сборочная среда и т.п. 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 комбинаций перекомпилированных программ в двоичном представлении обеспечивается процессом доступа к репозитарию для выбора компонентов системы в альтернативной сетевой среде Grid. Задача представления типов данных объектов среды: Проект, Подсистема и Компонент – реализуется с использованием типового формата CIM как механизма связи между разными компонентами с помощью системы MySQL на основе аннотации ряда свойств компонентов (имя, лицензия, URL в репозиторий и т.д.), глобального уникального идентификатора – ID (GUID) и скомпилированного компонента с отображением его в конфигурационном файле ПП.

Сервисы – главные ресурсы инфраструктуры вычислений Grid. Они обеспечивают процесс вычислений научных задач глобального масштаба. Ресурсы задаются в протоколе для передачи данных по распределенной сети. Сервисы задаются 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 и открытых интерфейсов для обслуживания потребителей или поставщиков, как на фабрике программ.

Фабрика ПО Ленца включает 4 строительных блока. Ключевые моменты Software Factory – схемы и шаблон Software Factory. Схема Software Factory – это описание того, как реализовать продукты, которые могут быть произведены на этой фабрике. Типичным примером шаблона является завод ПО, а именно инструмент Visual Studio, обеспечивающей разработку проектов из многократно используемых компонентов при реализации проектных решений и требований. Требования и решения описываются в языке DSL в виде задания архитектуры, блоков и документов, кото-

рыми заказчики приложения будут пользоваться при реализации соответствующей линии продукта или ее экземпляров.

Архитектурный каркас фабрики Авдошина. Основным ресурсом и активом данной фабрики – архитектурный каркас как отправная точка в разработке любого продукта на линии. Каркас покупается или разрабатывается организацией-разработчиком ПП в среде фабрики. В нем аккумулируются ресурсы: классы, компоненты, конфигурации, образцы и т.п. Активы выбираются на стадии разработки линии фабрики. База знаний фабрики включает разные пособия, справочники, примеры программ и программ-образцов. Проводится модельно-ориентированная разработка на основе абстрактной модели, которая оперирует набором понятий ПрО, заданных в языке DSL с помощью графического дизайнера. Моделирование понятий и основных концепций осуществляется генерацией кода и конфигурированием продукта. Для реализации фабрики задается схема фабрики, которая охватывает активы, точки зрения и связи между ними. Из схемы берется шаблон фабрики и уточняется набор необходимых ресурсов для автоматизации работы среды разработки.

Разработчик на фабрике задает схему продукта или модель описания рабочих продуктов в заданной ПрО, а также процессы и активы в виде коллекции избранных и тщательно подобранных активов и рекомендаций для реализации применения на линии. Пользователь фабрики работает с шаблоном, который специфицирует процесс разработки. Любая схема фабрики состоит из набора взаимозависимых точек зрения, каждая из которых разрешает посмотреть на систему с разных сторон. Точки зрения помогают понять логическую и физическую стороны системы, набор используемых компонентов и документирование архитектуры семейства ПП. Точки зрения управляют созданием типов и средств проекта. Каждая точка зрения носит имя и описание моделей, средств, шаблонов и др. Ей соответствует стандартные блоки схемы фабрики, которые формализуют действия по созданию и изменению рабочих продуктов. Одна точка зрения может иметь несколько видов, которые могут использовать разные механизмы (языки, модели, сценарии) и документировать разные аспекты производства ПП. Ими могут быть:

– идентификатор вида и его описание;

– описание системы соответственно заданной точке зрения;

– конфигурация продукта.

Схема фабрики – это набор взаимозависимых точек зрения, необходимых для построения продуктов. Она помогает определить и обрабатывать данные, а также организовать вариант фабрики, связанный с конкретными сервисными ресурсами создания и управления предприятием. Шаблон фабрики – это пакет с ресурсами и со схемой фабрики, включающей:

– библиотеки и каркасы с многократно КПИ, вновь разработанными при проектирования линии (.NET Enterprise Library);

– рекомендации, технологии, распоряжения и руководства по автоматизации процесса построения ПП;

– языки предметной области и дизайнеры, которые задают необходимый уровень абстракции при разработке приложений и генерации кода по модели.

Построенный шаблон фабрики может быть загружен в интегрированную среду разработки некоторого ПП и обеспечивать конфигурацию средств и КПИ в новый ПП.

3.8.4. *Типовые фабрики программ на современных платформах*

В каждой операционной и распределенной системе массового применения созданы фабрики программ, выполняющие функцию изготовления сложных ПП. Они работают на всех современных компьютерных и кластерных платформах:

– AppFab в системе коллективной разработки VS.Net;

– AppFab в системе Grid рамочного Европейского проекта;

– AppFab в системе WebSphereIBM для создания бизнес-доменов;

– AppFab в системе CORBA для сборки разнородных программных ресурсов;

– AppFab в системе Intel;

– Product Line SEI USA;

– фабрики программ Дж. Гринфильда, Г. Ленца, М. Фаулера и др.;

– коммерческие фабрики программ типа ЕПАМ;

– другие системные фабрики.

К *системным фабрикам* относятся следующие: IBM WebSphere, Microsoft Biz Talk 2004, BEA WebLogic Oracle 10g, SAP NetWeaver, ИВК «Юпитер». Дадим краткую им характеристику.

Платформа корпорации IBM – WebSphere позволяет работать на основе веб-технологий. Ядро WebSphere включает WebSphere Application Server (WAS) и открытые стандарты J2EE, XML и веб-сервисы.

Платформа WebSphere – это многофункциональный набор инструментов интеграции приложений в рамках предприятия (EAI) на уровнях: данных, обмена сообщениями, сквозных бизнес-процессов, B2B business to business; бизнес-логики программ на Java. В этот набор входит:

– сервер приложений WAS;

– Portal Server средства, функционирующие на WAS;

– система обработки очередей сообщений (Message Oriented Middleware, MOM), Business Integration Interchange Server (ICS) и MQ Business Integration Message Broker (WSMB);

– система проектирования Workflow, совместимая с WSMB.

В состав WebSphere входит Business Integration Workbench для проектирования бизнес-процессов и управления ими. Продуктами платформы являются образцы в своих классах, брокер сообщений WSMB, WAS и встроенные возможности для задания бизнес-правил и сценариев workflow, а также компоненты Enterprise Java beans (EJB). Сервер WAS позволяет использовать ресурсы веб-сервисов с компоновкой в единый процесс. IBM WebSphere объединяет следующие средства:

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

– WebSphere Business Integration *for Banking* предоставляет для банков средства обслуживания клиентов и предоставление финансовых услуг;

– WebSphere Business Integration *for Financial Networks* позволяет руководить обработкой платежей путем консолидации разнородных сетей обмена сообщениями;

– WebSphere Business Integration *for Electronics* ориентирован на компании, которые производят электронику и обеспечивают интеграцию и оптимизацию операций проектирования и производства. Кроме того, этот продукт позволяет соединять между собой «унаследованные» системы и разнородные приложения;

– WebSphere Business Integration *for Energy* and Utilities обеспечивает оптимальную интеграцию процессов эксплуатации (в частности, бесперебойного электроснабжения), управление активами и их обслуживанием;

– WebSphere Business Integration Express *for Item Synchronization* помогает компаниям среднего размера получить информацию из их цепочек услуг.

Таким образом, IBM WebSphere предоставляет набор средств по созданию приложений разного типа методом их интеграции.

Платформа Microsoft .NET Framework. Она предоставляет разработчику ту же функциональность, что и J2EE, но в среде ОС Windows. Инструменты, необходимые для реализации разных интеграционных подходов, сгруппированные в ней в виде нескольких продуктов, а отдельные функции возложены непосредственно на ОС (например, компонент управления транзакциями MTS, веб-сервер Internet Information Server, библиотеки и среда выполнения «управляемого кода» .Net).

Основную функциональность EAI выполняет BizTalk Server 2004 – сервер интеграции на базе XML как брокер сообщений, осуществляя преобразование данных и коммутацию сообщений.

Сервер приложений выполняет бизнес-логику – низкого (компоненты EJB) и высокого (через механизмы workflow) уровней, BizTalk Server поддерживает только высокоуровневую бизнес-логику и интеграцию систем,

а выполнение логики низкого уровня реализуется моделью COM+ или .Net.

Модель Microsoft позволяет размежевать работу программистов и аналитиков бизнес-процессов. Бизнес-аналитик может графически «рисовать» бизнес-процесс, задавая схемы обмена документами и передачи управления через workflow. Интеграция проводится через точки вызова внешней функциональности через COM-объекты, веб-сервисы и т.п.

Платформа WebLogic Integration. Данная платформа позволяет осуществлять интеграцию приложений и обеспечить их информационное взаимодействие с бизнес-партнерами (B2B), а также создавать бизнес-логику программ в языке Java. В эту платформу входит пять основных компонентов: виртуальная машина Java, сервер приложений, средства построения порталов, пакет инструментов интеграции и среда разработки.

Ключевым преимуществом платформы считается возможность снижения требований к группе разработки за счет использования трехуровневого подхода к созданию приложений, подобно подходу корпорации Microsoft. Программы создаются на языке Java, Visual Basic или COBOL. В платформе BEA есть поддержка новейших стандартов XML (XSLT, XQuery и т.п.), веб-сервисов и средства гарантированной доставки, совместимых со стандартом JMS и брокером сообщений. WebLogic предоставляет один из самых полных наборов интерфейсов для интеграции приложений, файлов и баз данных разной природы. Эта платформы включает много готовые конекторы, систему документооборота, синтаксический анализатор форматов файлов, средства обращения ко всем выполняемым модулям программ Windows и Java, а также взаимодействия с интеграционными платформами других разработчиков.

Платформа Oracle Integration предоставляет полный набор средств корпорации «Oracle» для интеграции приложений из разнородных приложений. Платформа соединяет технологии классов со стилями интеграции: данных (технология Transparent Gateways и коннекторы базы данных), интерфейса пользователя, сервера и системы MOM. В этой платформе реализованы COA (Component-oriented architecture) и веб-сервисы управления координацией и композицией приложений в любых средах разработки приложений. Проводится конструктивное развитие двух современных парадигм конструирования приложений с корпоративными вычислениями и с использованием сервисов (Service-Oriented Computing) и сетевых вычислений (Grid Computing). Рассматривается аспектная инфраструктура реализации SOA и объединение (federate) приложений с необходимой функциональностью. В свою очередь, корпоративные вычисления на базе сервисов обеспечивают гибкую инфраструктуру корпоративных приложений.

Сетевые вычисления предусматривают координированное использование значительного количества собираемых из модулей серверов и блоков памяти, которые имеют низкую стоимость. Для эксплуатации и поддержки деловых процессов представлены три базовые технологические решения:

1) Business Intelligence – средства построения и анализа бизнес-информации предприятия для сборки и распределения разноуровневой информации;

2) Business Integration – средства интеграции разнородных приложений, которые делают возможным объединение отдельных подсистем и автоматизацию деловых процессов;

3) Identity Management (управление идентификационными параметрами личности), что позволяет консолидировать средства администрирования защиты, чтобы снизить полную стоимость владения ими и сократить количество точек уязвимости.

Для поддержки конструирования приложений предлагается три основных пакета.

1. Oracle Integration Interconnect. Пакет обеспечивает многоаспектные функциональные возможности композиции и координации сервисов предприятия через соответствующую шину ESB для быстрого развертывания интеграционных приложений на предприятии. SAP предлагает два репозитория метаданных для интеграционных связей: Repository с заполненными отношениями SAP и развертыванием (Directory). Это позволяет максимально приблизить условия разработки и тестирования создаваемой ПС к условиям ее эксплуатации. SAP поддерживает интеграцию не только на уровне коннектора к шине обмена данными, но и на высших уровнях, совместимыми с системою управления контентом и порталом среды разработки. Эта платформа использует среду разработки Eclipse, IBM WebSphere среду выполнения SAP Web Application Server через SAP Java Connector, а также среду MS.NET через SAP .NET Connector.

Платформа ИВК «Юпитер» – российский продукт. Обеспечивает функции интеграции на уровне данных и обмена сообщениями в государственных структурах, которые выдвигают повышенные требования к защите информации и возможностям интеграции унаследованных и устаревших ПС. Эта платформа поддерживает современные вычисления путем соединения характеристик виртуальной машины, транспортной магистрали, систем документооборота и средств защиты информации. Состоит из двух высокоуровневых логических блоков:

– первый обеспечивает стандартную функциональность на изолированном компьютере;

– второй – связь между разными компьютерами.

На каждом компьютере обеспечивается реализация унифицированной модели вычислительного процесса, которая соединяет среду выполнения системы ИВК и набор библиотек. Полученный продукт может вычисляться в средах типа Cloud Computing и обеспечивать контроль целостности вычислительного процесса.

Таким образом, из приведенного описания известных современных платформ индустриального изготовления приложений следует, что это типичные фабрики с базовыми средствами создания, компоновки и координации специальных КПИ и сервисов. Эти платформы позволяют выбирать необходимые готовые ресурсы, сервисы и инструментальные средства для реализации и конструирования новых приложений, способных функционировать в этих же средах или в других средах организации вычислений типа Grid и Cloud Computing.

3.9. Пути развития технологий компьютерных систем

В.М. Глушков один из первых подхватил идею Норберта Винера (1948) «Кибернетика или управление и связь животных и машин». В это время уже появились первые ЭВМ за рубежом и в СССР. Термин употреблялся как управление живыми организмами, машинами и обществом с помощью информации, передаваемой между ними по каналам связи. Информация стала источником жизнедеятельности всех сфер жизни. Начиная с 1980-х годов кибернетика положила начало научной дисциплине информатике, которая изучает общие свойства, методы и системы создания, сохранения и использования информации во всех сферах жизни общества с помощью вычислительной техники и связей. Информатика фактически соответствует Computer Sciences.

В Computer Sciences началась эволюция научных идей, связанных с техникой разработки ПС, технологий и других форм концентрации знаний. Развитие информатики началось с первых экспериментов ЭВМ до появления индустрии их выпуска.

Кибернетика способствовала развитию теории автоматов, математической логики, теории программирования, теории компьютеров, информационных систем, баз данных и знаний и др.

Основным способом передачи знаний в производственной и управленческой сфере являются информационные технологии. Такие технологии реализуют научные идеи на потребности человека, включают метод (способ) и определенный порядок их применения в производстве разных видов продуктов (материальных и нематериальных). Элементы компьютерных ИТ-технологий включают технические ресурсы повторного использования (микро-, макроэлементы, микротехника, материалы, данные и

т.п.) и нанотехнологии, которые объединяют микроресурсы в адаптивные биокибернетические ресурсы предметных областей [19].

В своих работах А.С. Лебедев, а потом В.М. Глушков предлагали новые виды рекурсивных, многопроцессорных, микро- и макроконвейерных ЭВМ и новую элементную базу для организации высокоэффективных вычислений сложных математических и народнохозяйственных задач, а также для управления оборудованием с приоритетным прерыванием объектов в системах АСУ и АСУ ТП. По его мнению, структурные элементы ЭВМ и систем необходимо постоянно модернизировать и стандартизировать. Он считал, что процесс изготовления компьютерных систем будет приближаться к автоматизированной сборке малых микроэлементов в более крупные элементы, как на конвейере Форда.

Сборочный конвейер основан на ТЛ сборки отдельных частей компьютерных систем в целом. Это предвидение фактически сбылось.

И сегодня мы видим, что компьютеры разных вариантов, типов и размеров собираются массово по принципу сборки. То есть в схеме и системотехнике ЭВМ созданы высокие технологии типа *нанотехнологий*, которые обеспечивают сборку новых моделей компьютеров с элементов подобно атомам и молекулам. Схемоэлементы компьютеров стали настолько мизерными, что их варианты встраивают в мобильные телефоны, микроэлементы медицинских приборов и т.п.

Позже, в 2000-х годах, появилась альтернатива ТЛ – продуктовые линии (Product Lines) института SE США (http://sei.cmu.edu/productlines/frame_report/ 2004 г.). И, кроме того, корпорация W3C разработала стандарт BPMN для нотации процессов и действий продуктовых линий. Среди линий имеются линии сборки семейств ПП с разработанных ПП. Настоящий период развития индустрии ПП в мире характеризуется усовершенствованием объектов сборки и линий сборочного конвейера на фабриках программ. Каждая линия повышает производительность и снижает себестоимость выпуска ПП (Lavrischeva K.M. Theory and practice of software factories // Cybernetics and Systems Analysis. Springer, 2011. V. 47, N 6. P. 961–972). Каждый из названных способов идет по пути построения гибких и высоких технологий, приближающих нас до нанотехнологий, когда элементы линии автоматически синтезируются в большие приложения.

На пути перехода к нанотехнологиям

Впервые идею синтеза атомов в макроатомы с помощью специального программного сборщика предложил Р. Фейнман (1959 г.) как манипулятора атома, на который не действуют силы гравитации, а действуют межмолекулярные ван-дер-ваальсовы силы [45]. Он считал, что может быть произвольное число таких механизмов, представленных манипулятором из элементов, уменьшенных в четыре и более раза копии «руки»

оператора, который может закручивать маленькие болтики и гайки, сверлить очень маленькие отверстия, выполнять работы в масштабе 1:4, 1:8, 1:16 (что близко к идее подковать блоху). К маленьким элементам относятся микроэлементы и хирургические макроприборы, позднее – чипы для микростимуляторов и т.п.

Р. Фейнман считал, что будут созданы миллионы миниатюрных заводиков, на которых «крошечные станки будут непрерывно сверлить отверстия, штамповать маленькие детальки» для макроприборов, собирать их в макромеханизмы, макровещества и т.п.

Эта идея привела к современной идеи миниатюризации и получению новых веществ из очень маленьких частиц других элементов со свойствами, необходимыми для конкретного применения. Такая маленькая частица названа «нано», что означает «карлик».

Нанотехнология – это технология производства, ориентированная на получение веществ и устройств с заранее заданной атомарной архитектурой (Э. Дрекслер).

Атом – это 10^{-10} = 1 нанометр (нм), а бактерии – это 10^{-9} нм.

Частицы от 1 до 100 нанометров называют «наночастицами».

Наночастицы имеют одно свойство – слипание друг с другом – которое приводит к образованию новых агломератов (в медицине, керамике, металлургии и др.).

Один из важнейших вопросов, стоящих перед нанотехнологией – как заставить молекулы группироваться определённым способом, самоорганизовываться, чтобы в итоге получать новые материалы или устройства. Например, белки, которые могут образовывать комплексные структуры путем синтеза нескольких молекул белков ДНК с новыми специфическими свойствами.

Одна из нанотехнологий создана в Московском дворце творчества «Интеллект» с участием учеников, посещающих его. Они в течение нескольких лет изучали физико-химические свойства некоторых материалов для создания сверхчастотного инфракрасного материала в видимом диапазоне частот из наноматериалов для получения новых веществ и материалов. Новый материал предназначался для маскировки военной техники и инженерных сооружений от оптических и радиолокационных средств разведки на растительные, горные и пустынные фоны. Полученный ими материал представляет собой волокна SiO_2. В материал внедрены ферромагнитные наночастицы, обеспечивающие коэффициент отражения в 15–80 раз по сравнению с металлической пластинкой в диапазоне частот А–37 ТЦ.

Эта нанотехнология была выставлена по многим городам как передвижной учебный класс «Нанотехнологии и материалы» (www.intelltct-cit.ru). Он демонстрировал:

– передовое качественное образование в сфере новейших разработок мирового уровня в области нанотехнологий с использованием уникальных методик и технологий;

– создание условий для развития школьников, студентов и их профессионального самоопределения в области научных дисциплин физики, математики, биологии и др.

На выставке был представлен системный комплекс, включая:

– сканирующий туннельный микроскоп «Умка» для изучения поверхности материалов с атомарной разрешающей острой иглой, скользящей по исследуемому материалу;

– устройство заточки иглы;

– смеситель материалов получения сверхчастотного материала;

– образцы материалов (астробетон – ЛБ, радиопоглощающий материал РПМ, гидрофобные покрытия);

– процесс получения наночастиц золота размером 15–20 нм и др.

Опыты и лабораторные работы проводились под научным руководством специалистов МГУ, институтов стали и сплавов, электронной техники, концерна наноиндустрии и др. Экспонаты демонстрировали результаты наноразвития таких дисциплин, как физика, химия, биология и др.

После изучения направлений и достижений нанотехнологий пришла идея обобщения компьютерных микроэлементов к наноэлементам в областях e-science (биологии, химии, физики, медицины, генетики и др.). Существующие ИТ-технологии достигли своего развития и дают мировому сообществу новые возможности. Например, Skype-технологии позволили визуально общаться людям, видя друг друга. Требуется разработать такие технологии, чтобы «поезд нано постепенно набирал обороты» в ближайшее десятилетие. То есть нанотехнологии будут способствовать поддержке жизни на земле, улучшать качество жизни и здоровье людей, а также исследовать недра земли, океанов и атмосферы, чтобы создать жизненноважные новые вещества из природных и наукоемких частиц.

В области компьютерной технологии создаются маленькие технические элементы в виде «чипов» и транзисторов и др. для использования разных предметных областей e-sciences. Они будут накапливаться как запас и средствами компьютерной нанотехнологии из них можно собирать, синтезировать новые технические и программные продукты.

В перспективе готовые программные элементы будут уменьшаться и приводиться к виду маленьких частиц с заранее атомарной структурой и функциональностью. Синтез таких маленьких частиц в более сложные

структуры программ и системы уже начал проводиться в рамках e-science (биология, медицина, генетика, физика, и др.), которые будут способствовать улучшению здоровья общества и жизни на земле. В ближайшее десятилетие ожидается развитие таких компьютерных нанотехнологий из новых минимизированных микроэлементов программ.

Литература к теме 3

1. *Лаврищева Е.М.* Технологическая подготовка и программная инженерия // УСиМ. 1988. – № 1. – С. 35–43.
2. *Лаврищева Е.М.* Основы технологической разработки прикладных программ СОД: препринт / ИК АН УССР. Киев, 1987. – № 5. – 29 с.
3. *Лаврищева Е.М.* Проблематика программной инженерии. – Киев: Знание, 1991. – 29 с.
4. *Лаврищева Е.М.* Модель процесса разработки ПО // УСиМ. – 1988. – № 5. – С. 53–60.
5. *Лаврищева Е.М.* Программная инженерия. Основные понятия и определения // Методы и средства программной инженерии: сб. ст. – Киев: РИО ИК АН УССР, 1989. – С. 30–36.
6. *Боем Б.У.* Инженерное проектирование программного обеспечения. – М.: Радио и связь, 1986. – 510 с.
7. *Лаврищева Е.М.* Концепція індустрії наукового софтвера і підхід до обчислення наукових задач // Проблеми програмування. – 2011. – № 1. – С. 3–17.
8. *Глушков В.М.* Основы безбумажной информатики. – М.: Наука, 1982. – 552 с.
9. *Lavrishcheva E.M.* Software engineering as a scientific and engineering discipline // Cybernetics and Systems Analysis. – 2008. – V. 44, N 3. – P. 324–332.
10. Лавріщева К.М. Програмна інженерія – напрями розвитку // Пр. міжнар. конф. «50 років Інституту кібернетики імені В.М. Глушкова НАН України». – Київ, 2008. – С. 336–345.
11. *Лаврищева Е.М.* Классификация дисциплин программной инженерии // Кибернетика и системный анализ. – 2008. – № 6. – С. 3–9.
12. *Андон П.І., Лавріщева К.М.* Розвиток фабрик програм в Інформаційному світі // Вісник НАН України. – 2010. – № 10. – С. 15–41.
13. *Глушков В.М.* Фундаментальные основы и технология программирования // Программирование. – 1980. – № 2. – С. 13–24.
14. *Вельбицкий И.В., Ходаковский В.Н., Шолмов Л.И.* Технологический комплекс автоматизации программ на машинах ЕС ЭВМ и БЭСМ-6. – М.: Статистика, 1980. – 263 с.
15. *Вельбицкий И.В.* Технология программирования. – Киев: Техника, 1984. – 278 с.
16. *Лаврищева Е.М.* Методы программирования. Теория, инженерия, инструменты. – Киев: Наукова думка, 2006. – 451 с.
17. *Mills H.D.* The management of Software Engineering. Part 1. Pinsiples of SE // IBM Systems. – 1980. – V. 19, N 4. – P. 415–418.

18. Лавріщева, К.М. Програмна інженерія: підручник / К.М. Лавріщева. – Київ: Академперіодика, 2008. – 319 с.
19. *Лаврищева Е.М.* Software engineering компьютерных систем. Парадигмы, технологи, CASE-средства. – Киев: Наукова думка, 2013. – 284 с.
20. *Глушков В.М.* Кибернетика, ВТ, информатика (АСУ). – Избран. труды в 3-х томах. – Киев: Наукова думка, 1990.
21. *Задорожная Н.Т., Лавріщева К.М.* Менеджмент документооборота в информационных системах образования. – Киев: Педагог. думка, 2007. – 220 с. (http://lib.iitta.gov.ua/view/creators)
22. *Lavrischeva E.M., Slabospickaya O.A.* An approach to expert assessment in software engineering // Cybernetics and Systems Analysis. – 2009. – V. 45, N 4. – P. 638–654.
23. *Lavrischeva E.M., Koval G.I., Korotun T.M.* An approach to the software quality management // Cybernetics and Systems Analysis. – 2006. – V. 2, N 5. – P. 758–768.
24. *Коваль Г.І.* Байєсівські мережі як засіб оцінювання та прогнозування якості програмного забезпечення // Проблеми програмування. – 2005. – № 2. – С. 15–23.
25. *Pohl K., Böckle G., Linden F.J.* Software Product Line Engineering: Foundations, Principles and Techniques. – New York: Springer–Verlag, 2005. – 437 p.
26. *Lavrischeva K.M.* Formal Fundamentals of Component Interoperability in Programming // Cybernetics and Systems Analysis. – 2010. – V. 46, N 4. – P. 639–652.
27. *Эммерих В*. Конструирование распределенных объектов. Методы и средства программирования интероперабельных объектов в архитектурах OMG/CORBA, Microsoft COM и JAVA RMI. – М.: Мир, 2002. – 510 с.
28. *Бей И.* Взаимодействие разноязыковых программ. – М.: Диалектика; М.–СПб–Киев. – Вильямс, 2005. – 868 с.
29. *Гринфильд Дж.* Фабрики разработки программ. – М.: Диалектика; М.–СПб–Киев. – Вильямс, 2007. – 591 с.
30. *Guckenheimer S., Perez J.I.* Software Engineering with Microsoft Studio Team System. – Crawfordsville, USA: Adison–Wesley, 2006. – 304 p.
31. *Чернецки К., Айзенекер У.* Порождающее программирование. Методы, инструменты, применение. – М.–СПб.–Харьков–Минск: Питер, 2005. – 730 с.
32. *Lenz G., Wienands C.* Practical Software Factories in .Net: From Theory to Practice – a Primer Reference and Case Study. – S.I.: Apress, 2007. – 205 p.
33. *Авдошин С.М.* Фабрики программного обеспечения. – http://www/ewwek.com./on
34. *Duval P., Matyas, Grover A.* Continuous integration Împroving Software Quality and Reducing Risk. – Addison Wesley, 2009. – 691 p.
35. *Lavrischeva K.M.* Theory and practice of software factories // Cybernetics and Systems Analysis. – 2011. – V. 47, N 6. – P. 961–972.
36. *Анісімов А.В., Лавріщева К.М., Шевченко В.П.* Про індустрію наукового софтвера // Conf. Theoretical and Applied Aspects of Cybernetics. Kiev. Febrary, 2011, Ukraine.

37. *Лаврищева Е.М., Зинькович В.М., Колесник А.Л.* и др. Инструментально-технологический комплекс разработки и обучения приемам производства программных систем (укр.). – Гос. служба интеллектуальной собственности Украины. – Свид. о регистрации авторского права. – № 45292 от 27.08.2012. – 103 с.
38. *Lavrischeva E., Stenyashin A., Kolesnyk* A. Object-Component Development of Application and Systems. Theory and Practice // Journal of Software Engineering and Applications. – 2014. V. 7, N 9. – Published Online August 2014 in SciRes. – http://www.scirp.org/journal/jsea
39. *Лавріщева К.М., Коваль Г.І., Бабенко Л.П. і др.* Нові теоретичні засади технології виробництва сімейств ПС у контексті ГП. – Електронна монографія. – ДНТІ України, ВИНИТИ Росії та ДНТБ, 2012. – 277 с. – Available at http://www.nbuv.gov.ua.
40. *Андон П.І., Лавріщева К.М.* Розвиток фабрик програм в інформаційному світі // Вісник НАН України. – 2010. – № 10. – С. 15–41.
41. *Лаврищева Е.М., Вишня А.Т., Грищенко В.Н. и др.* Система автоматизации производства программ с режимом мультидоступа (АПРОП-2). – ЕрНУЦ, Ереван, 1981. – № П005508, ЯЩ 15001–33–01, 09.12.81. – 587 с.
42. *Соммервилл Н.* Инженерия программного обеспечения. – М.–СПб.–Киев: Вильямс, 2002. – 623 с.
43. 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.
44. *Лаврищева Е.М., Грищенко В.Н.* Связь разноязыковых модулей в ОС ЕС. – М.: Финансы и статистика, 1982. – 136 с.
45. *Лаврищева Е.М.* От технологии программирования к компьютерным нанотехнологиям программ и систем // Международный научный конгресс «Інформаційне суспільство в Україні – 2013». – С. 56–60.

СОДЕРЖАНИЕ

Для заметок