Фундаментальные основы программной инженерии. Парадигмы, технологии, CASE-средства

Е.М. Лаврищева

доктор физ.-мат. наук, профессор,
Институт системного программирования им. В.П. Иванникова РАН,
109004, Россия, г. Москва, ул. А. Солженицына, д. 25.
Московский физико-технический институт, 141700, Московская область,
г. Долгопрудный, Институтский пер., 9
lavr@ispras.ru, lavryscheva@gmail.com

Abstract. Define the basic concepts and fundamental foundations of Software Engineering (SE). The basic concepts are objects, modules, programs, systems; processes. The fundamental basis of the SE are: assembly method (configuration) of modules; SE disciplines (scientific, engineering, economic, management, etc.); paradigms of programming modules, objects, components, etc.; life cycle (ISO/IEC 12207 Life Cycle); technological and production lines; the factory programs and AppFab; logical-mathematical theory of object-component modeling (OKM) changing systems; verification and testing of systems.

Аннотация. Определяются базовые понятия и фундаментальные основы Software Engineering (SE). Базовые понятия – это объекты, модули, программы, системы; процессы разработки объектов. Фундаментальную основу SE составляют: метод сборки (конфигурирования) модулей; дисциплины SE (научная, инженерная, экономическая, управленческая и др.); парадигмы программирования модулей, объектов, компонентов и др.; жизненный цикл (стандарт ISO/IEC Life Cycle 12207); линии изготовления систем; фабрики программ и AppFab; логико-математическая теория объектно-компонентного моделирования (ОКМ) изменяемых систем; верификация и тестирование систем.

1 Вступление

Начиная с появления SE (1968), многие ученые и специалисты Computer Sciences стали создавать методы разработки программ и систем для решения задач в разных областях знаний (математика, физика, биология, архитектура, промышленность и др.). Первая книга Б.Боэма [1] посвящена описанию техники проектирования и оценивания качества программного обеспечения (ПО). В [2, 3] дается описание методики проектирования и оценки надежности и качества систем. В связи с появлением ООП в [4, 5] представлен унифицированный язык моделирования UML (1996) систем с помощью use case прецедентов и процессов RUP, В Software Engineering Body of Knowledge (SWEBOK) (www.swebok.com) (2001) дано определение предмета – **программная инженерия**

– система методов, способов и дисциплин планирования, разработки, эксплуатации и сопровождения ПО, предназначенного для промышленного производства ПО. SE охватывает все аспекты создания ПО от начала формулировки требований вплоть до разработки, сопровождения и окончательного списания. В [6-9] сформулированы базовые понятия, парадигмы их программирования и сборки в системы или семейство систем. Далее автор приводит формальные основы производства систем, их верификацию и тестирование [10, 11].

2 Общая характеристика фундаментальных основ SE

2.1 Базовые понятия программной инженерии

Программа – это объект разработки, который доступен пониманию ЭВМ, для которой написан. Готовая программа – это программный продукт (ПП), реализующий определённые функции предметной области (ПрО) [4,5]. Объектами разработки могут быть: модуль, программа, система, семейство систем и др. Т.е. объект либо сам является отдельной конструктивной единицей разработки систем либо состоит из взаимосвязанного их набора.

Метод разработки (программный метод) – это способ или планомерный подход к достижению той цели, которая ставится перед созданием объекта разработки. Наиболее распространенным методом является метод модульного программирования, обеспечивающий декомпозицию задачи на отдельные функции, вплоть до элементарной, каждая из которых представляется модулем или программой.

Модель жизненного цикла ПС – это каскадная, спиральная, итерационная и др. На основе этих моделей разработан первый вариант стандарта ISO/IEC Life Cycle 1996, а потом 2007. Этот стандарт регламентирует набор процессов разработки ПО и процессный подход к разработке ПО.

Технологический процесс – это взаимосвязанная последовательность операций, выполняемых при разработке объекта. Процесс предназначен для перевода объекта из одного состояния в другое вплоть до получения конечного продукта [10].

Линия – технологическая (ТЛ) или *продуктовая* (ЛП) задает набор процессов разработки функций некоторого объекта, которые последовательно и систематически преобразуют объекты к готовому ПП.

Инструмент (CASE-средство) – это программное или методическое средство для получения объекта в законченном виде. Это трансляторы, отладчики, генераторы, сборщики, тестеры и т. п.

Интерфейс (1976) – это модуль-связник объектов друг с другом для обмена данными между ними.

2.2 Метод сборки

Основу метода сборки модулей составляет интерфейс. Он обеспечивает связь модулей друг с другом и обмен данными. Концепция интерфейса впервые опре-

делена в программировании и реализована в 1975–1982 г. в рамках системе АПРОП [6-9].

Интерфейс (межмодульный и межъязыковый) стал базовым понятием технологии программирования и программной инженерии. *Межмодульный интерфейс* – это интерфейсный модуль-посредник (stub, skeleton в терминологии CORBA) между двумя взаимодействующими модулями, обеспечивающий передачу и прием данных между ними. *Межъязыковый интерфейс* – совокупность средств и методов преобразования структур и типов данных языков программирования (ЯП) с помощью алгебраических систем и функций библиотеки интерфейса для взаимно однозначного обмена данных .между разноязыковыми модулями. Для класса ЯП ЕС ЭВМ разработана библиотека интерфейса (64 функции) для преобразования разных типов данных ЯП и была передана в 52 организации СССР для сборке разноязыковых программ в ОС ЕС. Метод сборки системы АПРОП и библиотекой интерфейса внедрены в комплекс РУЗА и ПРОТВА (В.В. Липаева) и стал основой создания ПО для разных ЭВМ ВПК. Этот комплекс был награжден Государственной премией Кабинета министров СССР (1985).

Метод сборки программ и методология ТПР (1987) использовались при формирования шести ТЛ для автоматизации задач Военно-морского флота СССР (АИС «Юпитер»). Сборка программ основывается на теории преобразования фундаментальных типов данных (FDT) и общих типов GDT. Теория FDT возникла в 70-х годах прошлого столетия в работах Дейкстра, Хоара, Вирта, Агафонова, Ершова и др., а теория общих типов данных GDT определена в стандарте ISO/IEC GDT 11404 – 2006 (General Data Types) и в нем представлен аппарат генерации GDT↔FDT.

В 1990–1995 годах появились языки описания интерфейсов – MIL (Model Interface Language), API (Application Program Interface), IDL (Interface Definition Language) и др.

3 Дисциплины SE

Классификация дисциплин программной инженерии предложена автором в 2008г. как доклад на конференцию «40 лет SE». Одновременно с этим статья была передана в журнал «Кибернетика» и опубликована на двух языках (рус., анг.). [13, 14, 10]. Предложенные дисциплины используются в программе обучения Curicula-13. Далее дается характеристика этих дисциплин.

3.1 Научная дисциплина SE

Основу этой дисциплины составляют классические науки (теория алгоритмов, теория множеств, теория доказательств, математическая логика и др.), теория программирования, теория абстрактных данных, теория управления и др. Эта дисциплина задает базовые понятия и объекты (структуры данных, функции и композиции), формализмы для описания систем и теорию преобразования данных для организации вычислений [13, 14] и др.

3.2 Инженерная дисциплина SE

Инженерия – это способы применения технологических правил и процедур, процессов ЖЦ, методик измерения и оценки качества разработки ПП. Данная дисциплина задает набор инженерных приемов, средств и стандартов, ориентированных на изготовление целевых ПП. Базовые понятия инженерии SE:

1. ядро знаний SWEBOK – набор методов, средств и процессов разработки и управления проектом;
2. базовый процесс SE, как стержень процессной деятельности разработчиков ПП;
3. стандарт конструирования артефактов на процессах ЖЦ;
4. инфраструктура – условия среды обеспечения базового процесса SE;
5. общесистемные и инструментальные средства поддержки процессов изготовления ПП.

3.3 Дисциплина управления SE

Базис этой дисциплины – классическая теория менеджмента проектов и стандарт IEEE Std.1490 PMBOK (Project Management Body of Knowledge); метод CRM (Critical Path Method) для графического представления работ, операций и времени их выполнения; метод сетевого планирования PERT (Program Evaluation and Review Technique) и др. В стандарте PMBOK определены процессы ЖЦ проекта и главные области знаний, сгруппированные по таким задачам, как планирование, мониторинг, управление и завершение. Основная область знаний этого ядра – управление деятельностью коллектива исполнителей проекта, контроля правильности выполнения задач проекта и финансовой его стоимости.

3.4 Экономическая дисциплина SE

Экономика – это самостоятельная дисциплина SE, обеспечивающая расчет разных сторон деятельности на проекте с учетом знаний специалистов для определения затрат, экспертизы, оценки стоимости, сроков и экономических показателей, заданных в требованиях к ПП. На практике используются экономические методы: прогнозирование размера ПП (FPA – Function Points Analyses, Feature Points, Mark-II Function Points, 3D Function Points и др.); оценка трудозатрат на разработку ПП с помощью семейства моделей COCOMO [1]; математические модели оценки трудозатрат на разработку ПП (Angel, Slim, Seer-SEM и др.) и методы оценки показателей качества ПП (Стандарт ISO/IEC 9126). Показатель качества – основа сертификации программного продукта.

3.5 Производственная дисциплина SE

Главная задача – 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 ПО большие *прибыли*. Производство ПП базируется на технологических процессах изготовления определенных видов продуктов с применением теории проектирования и использования инструментальных сред [12].

4 Парадигмы программирования для разработки программ и систем

4.1 Отечественные и зарубежные парадигмы программирования

Разработан формальный аппарат для представления парадигмы программирования (модульной, объектной, компонентной, сервисный, аспектный и др.). Аппарат представлен теоретическими и прикладными аспектами проектирования соответствующих ресурсов в этих парадигмах и их сборку (конфигурацию) в программные компьютерные системы. В [10, 11, 13, 14] описаны следующие парадигмы:

- объектная программирования;
- компонентное программирование;
- объектно-компонентное программирование;
- генерирующее программирование;
- аспектно-ориентированное программирование и др.

Изготовленные на парадигмах элементы стандартизируются согласно WSDL и собираются в сложные программ с помощью метода сборочного (конфигурационного) программирования, который обеспечивает общий механизм взаимодействия элементов этих парадигм в новых системах и семействах [15, 16].

4.2 Характеристика сборочных технологий

Модульная сборочная технология – базируется на процедурах и функциях в виде модулей и методологии программирования в среде ЕС [8-10].

Объектное сборочное программирование базируется на методологии ООП, ОКМ и обеспечивает использование библиотек методов, классов и средств CORBA, Rational Rose, ИТК и др.

Компонентное сборочное программирование обеспечивает связь компонентов и интерфейсов, преобразование несовместимых данных и смену версий классов без перекомпиляции. Поддерживают это программирование системы: COM ,DCOM, .NET, OBERON и др.

Аспектное, сборочное программирование дополняет компонентное программирование концепцией аспекта для изменения варианта реализации критических по эффективности процедур и программ. Оно дополняет собранную систему новыми функциями (безопасности, синхронизации, надежности и др.).

Сервисное сборочное программирование обеспечивает интеграцию сервисов и обслуживания ПС. Общие сервисы выполняют связь, управление, каталогизацию и др.; объектные сервисы поддерживают объекты, классы и операции их выполнения; веб–сервисы Интернет для решения бизнес-задач.

5 ЖЦ стандарта ISO/IEEE. Подход к автоматизации

Стандарт ISO/IEC ЖЦ 12207–2007 является основным инструментом планомерного. процессного изготовления ПС и представлен тремя категориями процессов:

1. основные процессы (рис. 1);
2. процессы поддержки (рис. 2);
3. организационные процессы (рис. 2).

В этом стандарте приведен перечень процессов, способ их выполнения и форм представления результатов. Основные процессы – это процессы разработки, эксплуатации и сопровождения ПС (рис. 2).

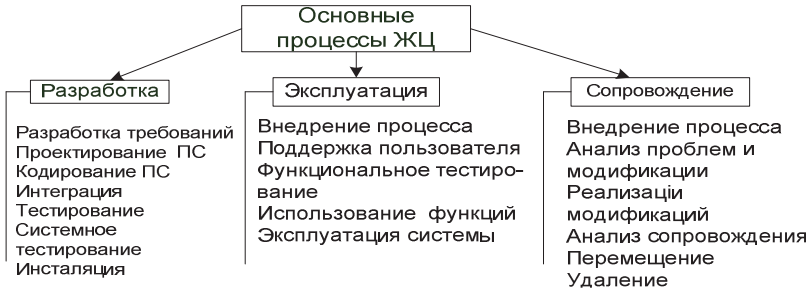

Рис. 1. Схема основных процессов ЖЦ ПС

Процесс разработки начинается с требований, проектирования элементов ПС (модулей, объектов, компонентов и др.), интеграции (сборку) отдельных элементов, тестирование отдельных элементов и системы в целом; эксплуатация готового ПП. Процессы поддержки и организационные процессы (рис. 2) используются для управление качеством ПС и усовершенствования процессов ЖЦ.

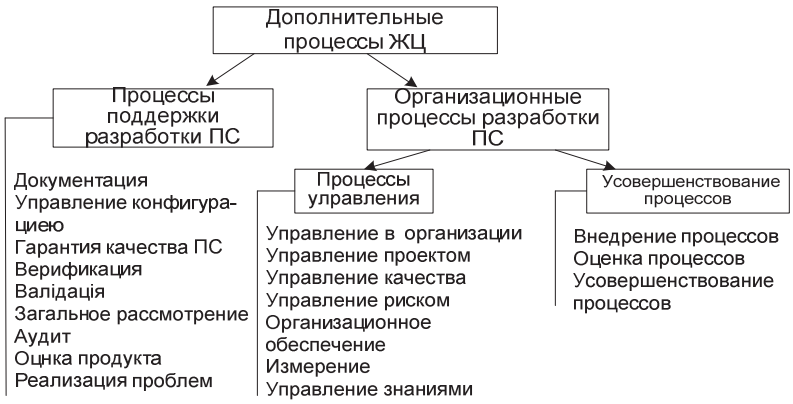

Рис. 2. Схема вспомогательных процессов

Стандарт ISO/IEC ЖЦ 2007 (табл. 1) включает в себя 17 процессов, 74 подпроцессов и 232 технологических задач (действий).

Таблица 1. Процессы, подпроцессы и задачи стандарта ЖЦ

Классы	**Процесс**	**Действие**	**Задача**
Основные процессы	5	35	135
Процессы поддержки	8	25	70
Организационные процессы	4	14	27
Всего	17	74	232

Этих процессов необходимо и достаточно для проектирования и изготовления систем. Некоторые системные фирмы реализуют отдельные фрагменты, т.е. отдельные варианты этого стандарта или пользуются моделями ЖЦ (спиральной, каскадной, итерационной и др.).

Автоматизация стандарта ISO/IEC ЖЦ средствами онтологии является новой и оригинальной. В основе реализации лежит структура процессов ЖЦ (рис 1, 2) и их взаимосвязи, а также язык онтологический

концептуализации отдельных вариантов процессов ЖЦ для конкретных применений [23, 24]. Средствами описания процессов ЖЦ могут быть: языки OWL (Web Ontology Language), ODSD (Ontology-Driven Software Development), XML (Extensible Markup Language); системы моделирования доменов – ODM (Organizational Domain Modeling), FODA (Feature-Oriented Domain Analysis), DSSA (Domain-Specific Software Architectures), DSL (Domain Specific Language), Eclipse-DSL Tools VS.Net, Protégé и т.п. К ним также относится язык BPMN описания процессов ЖЦ и язык DSL для описания семантики доменов. Проведено онтологическое описание основных процессов, процессов поддержки и

организационных процессов в языке DSL и получено описание этих процессов на Protege 2.3 в XML виде.

Подход к автоматизации ЖЦ докладывался на двух конференциях, в том числе “Science and Information- 2015” [24].

6 Подход к созданию новых технологий

Основу технологии производства ПП составляют линии продуктов и технологий, создаваемые с помощью метода технологической подготовки разработки (ТПР, 1987) [12]. Этот метод апробирован в проекте Института Кибернетики АИС «Юпитер–470» для автоматизации военно-морского флота СССР (1982–1991). В нем подготовлено шесть ТЛ для создания ПО и представлены в конкретные формы, документы и процессы этой АИС. По этим ТЛ было реализовано около 500 программ обработки данных для разных объектов АИС. Процессы ТЛ выполняют операции над готовыми ресурсами (модулями, компонентами, данными, метриками качества и др.). Работы в области мета технологий ТПР начали выполняться с помощью языков UML, DSL, Workflows, BPMN (Basic Process Modeling Notation) и др. Эти средства используются для создания продуктовых линий (Product Lines/Product Family), как инфраструктуры производства ПП из готовых ресурсов.

6.1 Фабрики программ – основа индустрии программ

Фабрика – это интегрированная архитектура сборочного производства ПП из готовых программных элементов (модулей, объектов, компонентов, сервисов, аспектов и др.), которые стандартно оформлены и размещены в системных библиотеках и репозиториях [17, 18]. Анализ имеющихся фабрик программ (Гринфильда, Вей, Ленц и др.) и опыт создания конкретной студенческой фабрики в КНУ (http://programsfactory.univ.kiev.ua) позволил сформулировать следующий набор необходимых элементов на фабрике программ:

1. готовые программные ресурсы (артефакты, программы, системы, reuses, assets, КПИ) др.;
2. интерфейсы – спецификаторы готовых ресурсов в языках IDL, API, SIDL, WSDL;
3. 3) ТЛ, продуктовые линии (Product Lines) производства ПП;
4. сборочный конвейер;
5. методики и приемы планирования и выполнения работ на линии по созданию системы;
6. общесистемная среда разработки отдельных программ.

К действующим фабрикам программ относятся:

1. AppFab в системе коллективной разработки VS.Net;
2. AppFab в системе Grid Европейского проекта;

3. AppFab IBM для создания бизнес-систем;
4. AppFab в системе CORBA для сборки разнородных программных ресурсов;
5. Product Line SEI USA;
6. фабрики потоковой сборки программ Дж. Гринфильда, Г. Ленца,
7. фабрика continuous integration М. Фаулера;
8. фабрика программ ЕПАМ и др.

Некоторые фабрики представлены на сайте http://www.7dragons.ru/ru.

7 Теория моделирования изменяемых систем

7.1 Сущность моделирования систем и семейств

Понятие вариабельности (изменяемости) представлено в модели FM (Feature Model) на Product Line, основанной на наборе готовых компонентов повторного использования (КПИ, типа reuses), которые в ПС отмечаются вариантными точками [19, 20]. Вариабельность – это свойство системы к расширению, изменению, приспособлению или конфигурированию с целью использования в определенном контексте и обеспечении последующей его эволюции . Модель FM формируется в процессе разработки ПП и включает общие функциональные и нефункциональные характеристики элементов, которые могут использоваться членами семейства ПС при создании разных вариантов ПС или ПП с учетом точек вариантности.

Точка вариантности – это место в системе, по которому осуществляется выбор варианта ПС. Эта точка транслируется в коллекцию вариантов, присоединяемых к ядру изготавливаемой системы. При производстве ПС из КПИ создается ПС и семейство ПС. Модель FM используется в инженерии предметной области и инженерии приложений. Инженерия предметной области (domain engineering) состоит в определении и реализации общих артефактов-функций вариабельного ПО для изготовлении нового *варианта* продукта. Артефакты – это архитектура, требования, компоненты, тесты и др. Инженерия приложений (application engineering) состоит в определении артефактов, необходимых пользователю и внесению изменений в семейство на уровне приложений.

7.2 Моделирование систем и семейств методом ОКМ

В последние годы широкое распространение получило проектирование моделей предметных областей (GDM, SOA, SCA и др.) и модели MF из готовых программных ресурсов и КПИ. Одним из методов моделирования является объектно-компонентный метод (ОКМ) [21, 22]. Этот метод обеспечивает четырехуровневое логико-математическое проектирование семейств СПС с помощью функциональных ($Fo = fo_1,\ldots,fo_n$) и интерфейсных элементов ($Io= io_1,\ldots,io_m$) домена или предметной области.

Совокупности функциональных элементов домена соответствует множество $F_O=(f_{O1}, f_{O2}, \ldots f_{On})$ и их интерфейсов *Io*. Для них определяется множество унар-

ных предикатов $P'=(P_1, P_2, \ldots P_r)$, с помощью которых устанавливаются характеристические свойства элементов f_{On} доменов.

Функциональный объект (f_{oi}) задает формальное описание прикладной функции ПС, которая обеспечивает решение задачи заданной предметной области/домена. Объект задается тройкой: имя, типы данных и области их значений. Интерфейсный объект (i_o) задает формальное описание операций вызова методов и данных функциональных объектов, которое является посредником взаимодействующих функциональных объектов.

Логико-математическое проектирование СПС на уровнях из функциональных и интерфейсных объектов сводится к построению подграфов уровней и к окончательному формированию графа [21]: $G = \{O, I, R\}$, где O – множество функциональных элементов, I – множество интерфейсных объектов, R – множество отношений (relations) между объектами (рис. 3). Вершины графа G задают функциональные элементы СПС – $f_{O1}, f_{O2}, f_{O3}, f_{O4}, f_{O5}, f_{O6}, f_{O7}, f_{O8}$ и интерфейсные элементы – $i_{O5}, i_{O6}, i_{O7}, i_{O8}$.

Элементы графа $f_{O1} - f_{O8}$ описываются в языке программирования, а интерфейсные объекты i_{O5}–i_{O8} в языке интерфейса IDL (Interface Definition Language).

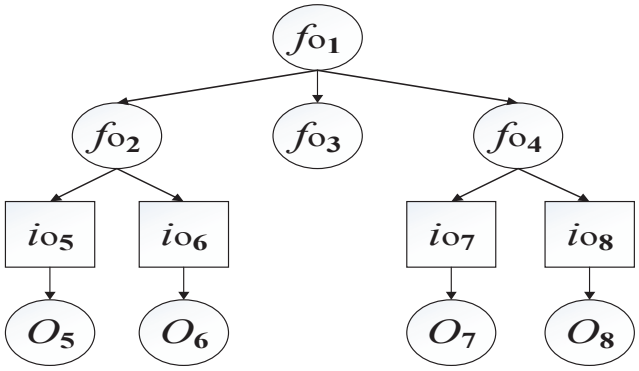

Рис. 3. Граф G на множестве функциональных и интерфейсных объектов

Параметры внешних характеристик интерфейсных объектов передаются между объектами через интерфейсы и помечаются знаками – *In* (входной), *Out* (выходной), *Inout* (входной и выходной).

Взаимодействие функциональных объектов, например, O_k, O_l обеспечивается интерфейсным объектом из множества входных интерфейсов *In*:

$$fo_k = (In(fo_k), Out(fo_k));$$
$$fo_l = (In(fo_l), Out(fo_l));$$
$$fo_k \cdot fo_l = (Out(fo_l), In(fo_k))$$

Теорема. Взаимодействие двух функциональных объектов является корректным, если первый объект полностью обеспечивает функции и передачу данных, необходимых другому объекту: $In(fo_k) \subseteq Out(fo_l)$.

По графу G можно собрать отдельные программы $P_0 - P_5$ с использованием математической операции ∪, соответствующей *link*:

1. $P_0 = (P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5)$;
2. $P_1 = f_{O2} \cup f_{O5}$, *link* $P_1 = In\ i_{O5}\ (f_{O2} \cup f_{O5})$;
3. $P_2 = f_{O2} \cup f_{O6}$, *link* $P_2 = In\ i_{O6}\ (f_{O2} \cup f_{O8})$;
4. P_3 :
5. $P_4 = f_{O4} \cup f_{O7}$, *link* $P_4 = In\ i_{O7}\ (f_{O4} \cup f_{O7})$;
6. $P_5 = f_{O4} \cup f_{O8}$, *link* $P_4 = In\ i_{O8}\ (f_{O4} \cup f_{O8})$.

Эти программы входят в состав ПС и могут отмечаться вариантными точками для проведения изменений в отдельные функциональные элементы модели ПС, соответствующей графовой модели G. Представлен переход от объектов к компонентам. В нем задается компонентная модель адекватная объектной в коде машины и модифицированная модель вариабельности [25].

Модель вариабельности ПС – $MF_{var} = (SV, AV)$ [22], где

SV – подмодель вариабельности артефактов в структуре ПС;

AV – подмодель вариабельности готового продукта ПС.

Модель MF_{var} обеспечивает изменение артефактов ПС, снижение затрат и уменьшение стоимости разработки системы.

Подмодель $SV = ((G_t, TR_t), Con, Dep)$,

где $G_t = (F_t, LF_t)$ – граф артефактов F_t и языка описания LF_t типа t; TR_t – связь артефактов типа *t*; *Con* і *Dep* – предикаты на декартовом произведении множества артефактов, которые задают ограничения и зависимости между функциональными элементами F_t и их показателями качества.

Подмодель *AV* определяет готовые КПИ, которые хранятся в репозитории. Эта подмодель отображает характеристики КПИ и системы, а также интерфейсы между ними на разных уровнях модели. Точки вариантности обрабатываются конфигуратором и производится замена некоторых КПИ другими, более корректными или новыми.

Модель вариабельности СПС – это совокупность FM моделей ПС, заданных на множестве артефактов, отмеченных точками вариантности для последующего изменения отдельных элементов [22].

7.3 Управление вариабельностью систем

Управление вариабельностью СПС проводится по точкам вариантности, вариантным артефактам ПС, ограничениям и зависимостями с помощью предикатов P, определенных на множестве вариантов ПС. Для управления вариабельностью используется метод Е. Деминга, основанный на функциях F1- F4:

F1 – операция, действие для подготовки артефактов СПС (Act);

F2 – планирование системы СПС из артефактов (Plan) для инженерии предметной области и инженерии приложений;

F3 – системный контроль и проверка состояния изменений СПС (Check);

F4 – актуализация (выполнение) систем СПС (Do).

Управление вариабельностью СПС с учетом требований R (Requriment) состоит в:

1. обосновании решений для функции F1 (R1);
2. согласовании способа реализации артефактов на процессах СПС (R2);
3. реализации проверки правильности создания СПС (R3);
4. отслеживании связей между характеристиками ПС и СПС (R4).

Соответствие требований R1 – R4 функциям F1 – F4 модельной среды процесса обработки модели вариабельности СПС – основа формирования и реализации вариабельной системы [].

8 Верификация и тестирование ПС и семейств СПС

Для проведения верификации объектов систем используется темпоральная логика (Linear Temporal Logic (LTL) или логика деревьев вычислений CTL (Computational Tree Logic) [10,11]. Метод дедуктивного анализа LTL обеспечивает логический вывод по модели, сделанной вручную. Он применяется только для тех объектов, которые являются критическими (например, обеспечивается безопасность функционирования или защиту информации). Верификация по модели model checking применима только к объектам с конечным числом состояний. Особенность метода верификации на модели заключается в том, что проверка проходит автоматически и для ее выполнения не нужны особые знания и время. Суть метода верификации – математическая формулировка требований к создаваемым программам и алгоритмы формальной проверки требований.

Тестирование рабочих продуктов (планов, наборов тестов, тестовых данных) основывается на использовании КПИ и готовых продуктов. Продукты тестирования должны быть пригодны для других ПП и являются частью повторно используемых компонентов семейства СПС.

Для задач тестирования ПС и СПС от требований используется сценарный подход (Scenario-based test derivation), метод анализа деревьев *FCTA (Fault Contribution Tree Analysis)* и комплекс PLUTO (Product Lines Use case Test Optimization).

8.1 CASE –инструментарий

В качестве средства реализации процессов ЖЦ ISO/IEC 2007 избраны средства языка Protégé и DSL Tool VS.Net и др. В них онтологическое описание трансформируется к языку XML, который является языком реализации размеченных функций доменов ЖЦ, определяющих связи и обмен данных ними. С участием студентов были разработаны программы обработки FDT и GDT, вариант онтологии ЖЦ с помощью инструментов – DSL Tools VS.Net и Protege [22] и др. Они представлены на веб-сайте http:///7dragons.ru/ru. Данный сайт обеспечивает изготовление ПС из готовых КПИ, сборку и тестирование объектов в средах (VS.Net, Corba, Java, Eclipse). В нем устанавливается связь с сайтом фабрики программ КНУ http://programsfactory.univ.kiev.ua. В нем накапливаются научные артефакты студентов КНУ. На сайт включен также курс обучения языкам Java, C # VS.Net и «Программная инженерия» для обучения студентов МФТИ. К сайту обращалось более 150000 студентов и преподавателей ВУЗов.

Заключение

В работе представлены базовые понятия и фундаментальные основы программной инженерии, изложенные в монографии [10] и ее копии в [11]. Приводятся базовые понятий SE (объект, программа, ТЛ, инструмент, метод, интерфейс и др.) и метод сборки на основе интерфейсов. Описан метод ОКМ, включающий логико-математический аппарат проектирования систем на обобщающем, структурном, характеристическом и поведенческом уровнях. В нем строится объектная модель и модель характеристик MF для управления вариабельностью систем и их семейств. Дано формальное описание парадигм программирования – компонентного, сервисного, генерирующего и аспектного программирования. Показан стандарт описания объектов, интерфейсов и метода сборки (конфигурации) для получения изменяемых сложных систем. Представлен новый подход к созданию сложных систем с использованием модели вариабельности и механизма управления изменением систем из разнородных элементов парадигм программирования. Дано описание идеи автоматизации процессов ЖЦ стандарта ISO/IEC 12207 с помощью средств онтологии Protege в среде сайта ИСП РАН http:///7dragons.ru/ru.

Литература

1. Боэм Б.У. Инженерное проектирование программного обеспечения.- М.-Радио и связь. 1986.- 510 р.
2. Липаев В.В. Надежность программного обеспечения АСУ, Энергоиздат, 1981;
3. Липаев В.В. Качество программного обеспечения, Финансы и статистика, 1983
4. Jacobson Ivar. Object-Oriented Software Engineering: A Use Case Driven Approach, 1992. ISBN 0- 201-54435-0.
5. Pfleeger Shari Lawrence, Software engineering: theory and practice, London : Prentice-Hall, 1996.- 676 p.
6. Лаврищева Е.М. Грищенко В.Н. Связь разноязыковых модулей в ОС ЕС – М.: Финансы и статистика , 1982.-137 с.
7. Лаврищева .Е.М. Проблематика программной инженерии.-Знание.- К., 1991.
8. Лаврищева Е.М., Грищенко В.Н. Сборочное программирование. – Киев: Наук. думка, 1991. – 213 с.
9. Лаврищева Е.М., Петрухин В.А. Методы и средства инженерии программного обеспечения. – М.: МОН РФ, 2007. – 415 с.
10. Лаврищева Е.М. Software Engineering компьютерных систем. Парадигмы, технологии, CASE- средства программирования. К.: Наук. думка.- 2014.-285 с.
11. Лаврищева Е.М. Программная инженерия. Парадигмы, технологии, CASE- средства.- 2 издание книги в п.10.- Юрайт, 2016.-280 с.
12. Лаврищева Е.М. Основы технологической подготовки разработки прикладных программ СОД.- Препринт 87-5 ИК АН УССР.—1987.-30 с.
13. Software engineering as a scientific and engineering discipline E. M. Lavrishcheva, 2008, Volume 44, Number 3, Pages 324-332
14. Classification of software engineering disciplines. E. M. Lavrischeva 2008, Volume 44, Number 6, Pages 791-796, Software–Hardware Systems.

15. Ekaterina M.Lavrischeva. Assemblling Paradigms of Programming in Software Engineering. – 2016, 9, 2016. – p. 296-317, http://www.scrip.org/journal/jsea, http://dx.do.org/10.4236/jsea.96021
16. Lavrischeva E. Generative and composition programming: aspects of developing software system families.- Cybernetics and Systems Analysis, Springer Volume 49, Issue 1 (2013), Page 110-123.
17. Лаврищева Е.М. Теория и практика фабрик программных продуктов .- Кибернетика и системный анализ .- № 6, 2011.- с.145-158.
18. Theory and practice of software factories K. M. Lavrischeva, 2011, Volume 47, Number 6, Pages 961-972.
19. Pohl K., Böckle G., van der Linden F. J. Software Product Line Engineering: Foundations, Principles and Techniques. Springer-Verlag, 2005. DOI: 10.1007/3-540-28901-1.
20. Berger T., She S., Lotufo R., Wąsowski A., Czarnecki K. A study of variability models and languages in the systems software domain. IEEE Transactions on Software Engineering, 39(12):1611-1640, 2013. DOI: 10.1109/TSE.2013.34.
21. Ekaterina Lavrischeva, Andrey Stenyashin, Andrii Kolesnyk. Object-Component Development of Application and Systems. Theory and Practice /Journal of Software Engineering and Applications, 2014, http://www.scirp.org/journal/jsea
22. Лаврищева Е.М. Теория объектно-компонентного моделирования программных систем. препринт ИСП РАН, 2016, 48 с. www.ispras.ru/preprints/docs/prep_29_ 2016_pdf
23. Lavrischeva E.M. Ontology of Domains. Ontological Description Sofware Engineering Domain – The Standard Life Cycle, Journal of Software Engineering and Applications, July 24, 2015.
24. Lavrischeva Ekaterina. Ontological Approach to the Formal Specification of the Standard Life Cycle, “Science and Information Conference-2015", Jule 28-30, London, UK,www.conference.thesai.org. – p.965-972.
25. Е.М..Лаврищева. Компонентная теория и коллекция технологий для разработки индустриальных приложений из готовых ресурсов, Труды 4-научно-практической конференции «Актуальные проблемы системной и программной инженерии», АПСПИ-2015, 20-21мая 2015, с .101-119.