Национальный Открытый Университет "ИНТУИТ": www.intuit.ru
Екатерина Лаврищева, Владимир Петрухин
Лекция 2. Области знаний программной инженерии и стандарты ЖЦ программного обеспечения

В данной лекции систематически изложены следующие взаимосвязанные аспекты инженерии *ПО*:

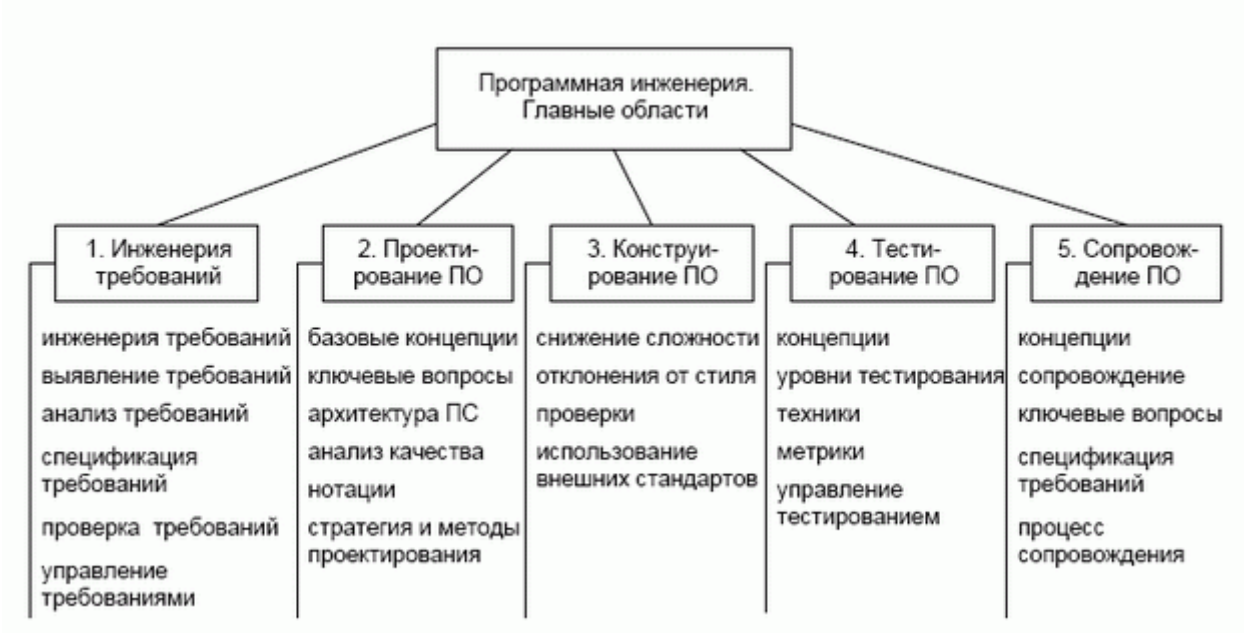
теоретический и интеллектуальный базис (методы, принципы, средства и методологии и др.) проектирования, представленный в ядре SWEBOK, способствующий созданию высококачественных программных продуктов, удовлетворяющих заданным заказчиком функциональным и нефункциональным требованиям;

стандартный подход к разработке программных проектов, состоящий в использовании моделей ЖЦ, в процессы которых встроены методы проектирования, верификации, тестирования и оценивания промежуточных *рабочих продуктов*, а также проверки планов и времени выполнения работ на этих процессах для того, чтобы регулировать сроки и затраты, а также возможные риски и недостатки.

1.1 Анализ и характеристика областей знаний SWEBOK

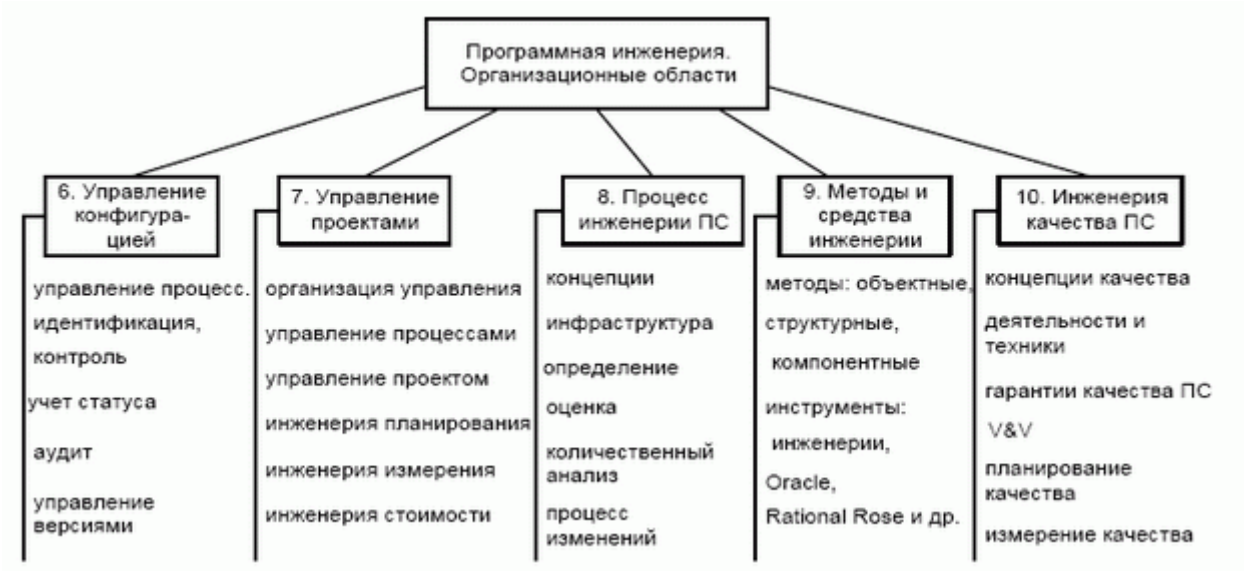
Ядро знаний SWEBOK является основополагающим научно-техническим документом, который отображает мнение многих зарубежных и отечественных специалистов в области программной инженерии [1.3–1.12] и согласуется с современными регламентированными процессами ЖЦ *ПО* стандарта *ISO*/IEC 12207. В этом ядре знаний содержится описание 10 областей, каждая из которых представлена согласно принятой всеми участниками создания этого ядра общей схемы описания, включающей *определение* понятийного аппарата, методов и средств, а также инструментов поддержки инженерной деятельности. В каждой области описывается определенный запас знаний, который должен практически использоваться в соответствующих процессах ЖЦ.

Для наглядного представления понятийного аппарата областей знаний SWEBOK проведем условное *разбиение* областей на основные (пять для проектирования ПС, рис. 1.1) и дополнительные организационные методы и подходы, которые отображают инженерию управления проектированием ПС (конфигурацией, проектами, качеством – рис. 1.2).



увеличить изображение

Рис. 1.1. Основные области знаний SWEBOK



увеличить изображение

Рис. 1.2. Организационные области знаний SWEBOK

В каждой области приведены ключевые понятия, подходы и методы проектирования разных типов ПС. Данное *разбиение* областей на основные и вспомогательные соответствует структуре разбиения процессов стандарта *ISO*/IEC 12207 (см. раздел 1.2), выполнение которых определяется знаниями, содержащимися в ядре SWEBOK.

Далее приводится обзор каждой области ядра знаний SWEBOK, определяется ее роль в проектировании и реализации программных продуктов. В некоторых подразделах показана *связь* с положениями соответствующих стандартов, которые регламентируют и регулируют выполнение процессов проектирования программных систем.

1.1.1. Требования к ПО (Software Requirements)

Требования – это свойства, которыми должно обладать ПО для адекватного определения функций, условий и ограничений выполнения ПО, а также объемов данных, технического обеспечения и среды функционирования.

Требования отражают потребности людей (заказчиков, пользователей, разработчиков), заинтересованных в создании ПО. Заказчик и разработчик совместно проводят сбор требований, их анализ, пересмотр, определение необходимых ограничений и документирование. Различают требования к продукту и к процессу, а также функциональные, нефункциональные и системные требования. Требования к продукту и к процессу определяют условия функционирования и режимы работы ПО в операционной среде, ограничения на структуру и память компьютеров, на принципы взаимодействия программ и компьютеров и т.п. Функциональные требования определяют назначение и функции системы, а нефункциональные – условия выполнения ПО и доступа к данным. Системные требования описывают требования к программной системе, состоящей из взаимосвязанных программных и аппаратных подсистем и разных приложений. Требования могут быть количественные (например, количество запросов в сек., средний показатель ошибок и т.п.). Значительная часть требований относится к атрибутам качества: безотказность, надежность и др., а также к защите и безопасности как ПО, так и данных.

Область знаний "Требования к ПО (Software Requirements)" состоит из следующих разделов:

- инженерия требований (*Requirement Engineering*),
- выявление требований (*Requirement Elicitation*),
- анализ требований (*Requirement Analysis*),
- спецификация требований (Requirement Specification).
- валидация требований (Requirement validation),
- управление требованиями (*Requirement Management*).

Инженерия требований к ПО – это дисциплина анализа и документирования требований к ПО, которая заключается в преобразовании предложенных заказчиком требований к системе в описании требований к ПО и их валидация. Она базируется на модели процесса определения требований и действующих лицах, обеспечивающих управление и формирование требований, а также на методах достижения показателей качества.

Модель процесса определения требований – это схема процессов ЖЦ, которые выполняются от начала проекта и до тех пор, пока не будут определены и согласованы требования. При этом процессом может быть маркетинг и проверка осуществимости требований в данном проекте.

Управление требованиями к ПО заключается в контроле за выполнением требований и планировании использования ресурсов (человеческих, программных, технических, временных, стоимостных) в процессе разработки промежуточных *рабочих продуктов* на этапах ЖЦ.

Качество и процесс улучшения требований – это процесс формулировки характеристик и атрибутов качества (надежность, реактивность и др.), которыми должно обладать ПО, методы их достижения на этапах ЖЦ и оценивания полученных результатов.

Выявление требований – это процесс извлечения информации из разных источников (договоров, материалов аналитиков по декомпозиции задач и функций системы и др.), проведения технических мероприятий (собеседований, собраний и др.) для формирования отдельных требований к продукту и к процессу разработки. Исполнитель должен согласовать требования с заказчиком.

Анализ требований – процесс изучения потребностей и целей пользователей, классификация и преобразование их к требованиям к системе, аппаратуре и ПО, установление и разрешение конфликтов между требованиями, определение приоритетов, границ системы и принципов взаимодействия со средой функционирования. Требования могут быть функциональные и нефункциональные, которые определяют соответственно внешние и внутренние характеристики системы.

Функциональные требования характеризуют функции системы или ее ПО, способы поведения ПО в процессе выполнения функций и методы передачи и преобразования входных данных в результаты. Нефункциональные требования определяют условия и среду выполнения функций (например, защита и доступ к БД, секретность, взаимодействие компонентов и др.). Разработка требований и их локализация завершается на этапе проектирования архитектуры и отражается в специальном документе, по которому проводится *согласование требований* для достижения взаимопонимания между заказчиком и разработчиком.

Спецификация требований к ПО – процесс формализованного описания функциональных и нефункциональных требований, требований к характеристикам качества в соответствии со стандартом качества ISO/IEC 9126-94, которые будут отрабатываться на этапах ЖЦ ПО. В спецификации требований отражается структура ПО, требования к функциям, качеству и документации, а также задается в общих чертах архитектура системы и ПО, алгоритмы, логика управления и структура данных. Специфицируются также системные требования, нефункциональные требования и требования к взаимодействию с другими компонентами и платформами (БД, СУБД, маршаллинг данных, сеть и др.).

Валидация требований – это проверка изложенных в спецификации требований, выполняющаяся для того, чтобы путем отслеживания источников требований убедиться, что они определяют именно данную систему. Заказчик и разработчик ПО проводят экспертизу сформированного варианта требований с тем чтобы разработчик мог далее проводить проектирование ПО. Один из методов валидации – прототипирование, т.е. быстрая отработка отдельных требований на конкретном инструменте и исследование масштабов изменения требований, измерение объема функциональности и стоимости, а также создание моделей оценки зрелости требований.

Верификация требований – это процесс проверки правильности спецификаций требований на их соответствие, непротиворечивость, полноту и выполнимость, а также на соответствие стандартам. В результате проверки требований делается согласованный выходной документ, устанавливающий полноту и корректность требований к ПО, а также возможность продолжить проектирование ПО.

Управление требованиями – это руководство процессами формирования требований на всех этапах ЖЦ и включает управление изменениями и атрибутами требований, а также проведение мониторинга – восстановления источника требований. Управление изменениями возникает после того, когда ПО начинает работать в заданной среде и обнаруживаются ошибки в трактовке требований либо в невыполнении некоторого отдельного требования и т.п. Неотъемлемой составляющей процесса управления является *трассирование требований* для отслеживания правильности задания и *реализации требований* к системе и ПО на этапах ЖЦ, а также обратный процесс отслеживания в полученном продукте реализованных требований. Для исправления некоторых требований или добавления нового требования составляется план изменения требований, который согласуется с заказчиком. Внесенные изменения влекут за собой и изменения в созданный продукт или в отдельные его компоненты.

1.1.2. Проектирование ПО (Software design)

Проектирование ПО – это процесс определения архитектуры, компонентов, интерфейсов, других характеристик системы и конечного состава программного продукта. Область знаний "Проектирование ПО (*Software Design*)" состоит из следующих разделов:

- базовые концепции проектирования ПО (*Software Design* Basic Concepts),
- ключевые вопросы проектирования ПО (Key Issue in *Software Design*),
- структура и архитектура ПО (Software Structure and Architecture),
- анализ и оценка качества проектирования ПО (*Software Design* Quality Analysis and Evaluation),
- нотации проектирования ПО (*Software Design* Notations),
- стратегия и методы проектирования ПО (Software Design Strategies and Methods).

Базовая концепция проектирования ПО – это методология проектирования архитектуры с помощью разных методов (объектного, компонентного и др.), процессы ЖЦ (стандарт ISO/IEC 12207) и техники – декомпозиция, абстракция, инкапсуляция и др. На начальных стадиях проектирования предметная область декомпозируется на отдельные объекты (при объектно-ориентированном проектировании) или на компоненты (при компонентном проектировании). Для *представления архитектуры* программного обеспечения выбираются соответствующие артефакты (нотации, диаграммы, блок-схемы и методы).

К ключевым вопросам проектирования ПО относятся: декомпозиция программ на функциональные компоненты для независимого и параллельного их выполнения, принципы распределения компонентов в среде выполнения и их взаимодействия между собой, механизмы обеспечения качества и живучести системы и др.

При проектировании архитектуры ПО используется *архитектурный стиль* проектирования, основанный на определении основных элементов структуры – подсистем, компонентов и связей между ними.

Архитектура проекта – высокоуровневое представление структуры системы и спецификация ее компонентов. Архитектура определяет логику отдельных компонентов системы настолько детально, насколько это необходимо для написания кода, а также определяет связи между компонентами. Существуют и другие виды представления структур, основанные на проектировании образцов, шаблонов, семейств программ и каркасов программных сред.

Одним из важнейших инструментов проектирования архитектуры является *паттерн* – типовой конструктивный элемент ПО, который задает взаимодействие объектов (компонентов) проектируемой системы, а также роли и ответственности исполнителей. Основным языком описания этого элемента является UML. Он может быть *структурным*, включающим типовые композиции структур из объектов и классов, объектов, связей и др.; *поведенческим*, определяющим схемы взаимодействия классов объектов и их поведение *диаграммами активностей*, взаимодействия, потоков управления и др.; *порождающим*, отображающим типовые схемы распределения ролей экземпляров объектов и способы динамической генерации структур объектов и классов.

Анализ и оценка качества проектирования ПО включает мероприятия по анализу сформулированных в требованиях атрибутов качества, оценки различных аспектов ПО – количества функций, структура ПО, качества проектирования с помощью формальных метрик (функционально-ориентированных, структурных и объектно-ориентированных), а также проведения качественного анализа результатов проектирования путем статического анализа, моделирования и прототипирования.

Нотации проектирования позволяют представить описание объекта (элемента) ПО и его структуру, а также поведение системы. Существует два типа нотаций: структурные, поведенческие и множество различных их представлений.

Структурные нотации – это структурное, блок-схемное или текстовое представление аспектов проектирования структуры ПО из объектов, компонентов, их интерфейсов и взаимосвязей. К нотациям относятся формальные *языки спецификаций* и проектирования: *ADL* (Architecture Description Language), UML (Unified Modeling Language), *ERD* (Entity-Relation Diagrams), IDL (Interface Description Language), Use Case Driven и др. Нотации включают языки описания

архитектуры и интерфейса, диаграммы классов и объектов, *диаграммы сущность-связь*, конфигурации компонентов, схем развертывания, а также структурные диаграммы, задающие в наглядном виде операторы цикла, ветвления, выбора и последовательности.

Поведенческие нотации отражают динамический аспект поведения систем и их компонентов. Таким нотациям соответствуют диаграммы потоков данных (Data Flow), таблиц принятия решений (*Decision Tables*), деятельности (Activity), кооперации (Collaboration), последовательности (Sequence), пред- и постусловия (Pre-Post Conditions), формальные *языки спецификации* (Z, *VDM*, RAISE), языки проектирования *PDL* и др.

Стратегия и методы проектирования ПО.К стратегиям относятся: проектирование снизу-вверх, сверху-вниз, абстрагирование, использование паттернов и др. методы – это функционально-ориентированные, структурные, которые базируются на структурном анализе, структурных картах, диаграммах потоков данных (Dataflow) и др. Они ориентированы на идентификацию функций и их уточнение сверху-вниз, после чего уточняются диаграммы потоков данных и проводится описание процессов.

В объектно-ориентированном проектировании ключевую роль играет наследование, полиморфизм и инкапсуляция, а также абстрактные структуры данных и отображение объектов. Подходы, ориентированные на структуры данных, базируются на методе Джексона [1.8] и используются для задания входных и выходных данных структурными диаграммами. Метод UML предназначен для сценарного моделирования проекта [1.19] в наглядном диаграммном виде. Компонентное проектирование ориентировано на использование готовых компонентов (reusing), их интерфейсов и интеграцию для формирования конфигурации, служащей основой развертывания компонентного ПО и взаимодействия компонентов в операционной среде.

Формальные методы описания программ основываются на спецификациях, аксиомах, описаниях некоторых условий, называемых предварительными (предусловиями), утверждениях и постусловиях, определяющих

заключительное условие получения правильного результата программой. Спецификация является формальным описанием функций и данных программы, с которыми эти функции оперируют. Различают входные и выходные параметры функции, а также данные, которые не привязаны к реализации и определяют интерфейс с другими функциями. По формальным спецификациям,

условиям и утверждениям выполняется доказательство правильности программы

1.1.3. Конструирование ПО (Software Construction)

Конструирование ПО – создание работающего ПО с привлечением методов верификации, кодирования и тестирования компонентов. К инструментам конструирования ПО отнесены языки программирования и конструирования, а также программные методы и инструментальные системы (компиляторы, СУБД, генераторы отчетов, системы управления версиями, конфигурацией, тестированием и др.). К формальным средствам описания процесса конструирования ПО, взаимосвязей между человеком и компьютером и с учетом *среды окружения* отнесены, например, структурные диаграммы Джексона.

Область знаний "Конструирование ПО (Software Construction)" включает следующие разделы:

- снижение сложности (*Reduction* in Complexity),
- предупреждение отклонений от стиля (Anticipation of *Diversity*),
- структуризация проверок (Structuring for Validation),
- использование внешних стандартов (Use of External Standards)

Снижения сложности – это минимизация, уменьшение и локализация сложности конструирования.

Минимизация сложности определяется ограниченными возможностями исполнителей обрабатывать сложные структуры и большие объемы информации на протяжении длительного периода времени. Минимизация сложности достигается, в частности, использованием в процессе конструирования модулей и других более простых элементов а также рекомендациями стандартов.

Уменьшение сложности в конструировании ПО достигается при создании простого и легко читаемого кода для придания большей значимости этого кода, простоте *тестирования*, *производительности* и удовлетворению заданных критериев. Это влияет на функциональность, характеристики и ограничения проекта. Потребность в уменьшении сложности влияет на все аспекты конструирования и особенно критична для процессов верификации (проверки) и тестирования результатов конструирования элементов ПС.

Локализация сложности – это способ конструирования с применением объектно-ориентированного подхода, который лимитирует интерфейс объектов, упрощает их взаимодействие, также проверку правильности самих объектов и связей между ними. Локализация упрощает внесение изменений, связанных с обнаруженными ошибками в коде, либо источником ошибок является среда, в которой выполняется код.

Предупреждение отклонений от стиля.Для решения различных задач конструирования применяются разные стили конструирования (лингвистический, формальный, визуальный).

Лингвистический стиль основан на использовании словесных инструкций и выражений для представлений отдельных элементов (конструкций) программ. Он используется при конструировании несложных конструкций и приводится к виду традиционных функций и процедур, логическому и функциональному их программированию и др.

Формальный стиль используется для точного, однозначного и формального определения компонентов системы. В результате его применения обеспечивается конструирование сложных систем с минимальным количеством ошибок, которые могут возникнуть в связи с неоднозначностью определений или обобщений при конструировании ПО неформальными методами.

Визуальный стиль является наиболее универсальным стилем конструирования ПО, он позволяет разработчикам проекта представлять конструируемый элемент в наглядном виде. Визуальный *язык проектирования* UML представляет разработчику набор удобных диаграмм для задания статической и динамической структуры ПО [1.17]. При применении визуального стиля конструирования создается текстовое и диаграммное описание конструктивных элементов ПО, которое выводится на экран дисплея не только для их наглядного представления, но и корректировки.

Структуризация проверок предполагает, что построение ПС должно проводиться таким образом, чтобы сама система помогала вести поиск ошибок, дефектов и причин сбоев при применении различных методов проверки, как на стадии независимого тестирования (например, инженерами-тестировщиками), так и в процессе эксплуатации, когда особенно важно быстрое обнаружение и исправление возникающих ошибок. Структуризации проверок способствуют обзор, инспектирование, просмотр, модульное тестирование, применение автоматизированных средств тестирования и др.

Использование внешних стандартов. Конструирование ПО зависит от применения внешних стандартов, связанных с языками программирования, инструментальными средствами и интерфейсами. При конструировании должен быть определен достаточный и полный набор стандартов для руководства и

обеспечения координации между определенными видами деятельности, группами операций, минимизации сложности, внесения изменений, анализа рисков и др.

Иными словами, внешние и внутренние стандарты должны определить общие правила работы членов проектной команды с учетом процессов ЖЦ. Такими стандартами являются: языки программирования (например, Java Language Specification, C++), интерфейсы ЯП и прикладные интерфейсы платформ Windows и др. При конструировании используются стандарты: ЯП (Ада 95, C++ и др.), языков описания данных (XML, SQL и др.), средств коммуникации (COM, CORBA и др.), интерфейсных языков (POSIX, IDL, *APL*) [1.18], сценарии UML [1.17] и др.

Данная область пополнилась новым разделом (SWEBOK, 2004 г.), описываемым ниже.

Управление конструированием – это управление процессом конструирования ПО, которое базируется на моделях конструирования, планировании и внесении изменений.

Модели конструирования включают набор операций, последовательность действий и результатов. Виды моделей определяются стандартом ЖЦ, методологиями и практиками. Некоторые стандарты ЖЦ по своей природе ориентированы на конструирование, например, *экстремальное программирование* (XP – *eXtreme Programming*). Конструирование с помощью моделирования осуществляется в рациональном *унифицированном процессе* – RUP (Rational Unified Process) [1.19].

Планирование состоит в определении порядка операций и уровня выполнения заданных условий в процессе конструкторской деятельности. Определяется модель ЖЦ, включающая задачи и действия по созданию компонентов, а также по их проверке, включая достижение показателей качества на этапах ЖЦ. Исполнители распределяются по процессам ЖЦ и выполняют соответствующие задачи по реализации промежуточного продукта. Затем проводится измерение разных аспектов конструирования, например, количественная оценка объема кода, оценка степени использования reuse, вычисление вероятности появления дефектов и оценка количественных показателей качества ПО.

Внесение изменений связано с ошибками, выявленными путем разного рода проверок и тестирований, оно проводится с целью сохранения функциональной целостности системы. В случае обнаружения ошибок на этапе сопровождения принимается решение об изменении кода путем исправления обнаруженных ошибок или внесения изменений в требования к ПО.

1.1.4. Тестирование ПО (Software Testing)

Тестирование ПО – это процесс проверки готовой программы в статике (просмотры, инспекции, отладки исходного кода) и в динамике путем прогона конечного набора тестовых данных, проверяющих разные пути выполнения программы и сравнении полученных результатов с заранее запланированными.

Существует две формы проверки кода – модульное и интеграционное. При этом используются стандарты (IEEE 829-1996 и IEEE 1008-1987) проверки и тестирования модулей ПО. Затем проводится *интеграционное тестирование* модулей системы и их интерфейсов в динамике выполнения. В процессе разных видов проверок собираются данные об ошибках, дефектах, отказах и т.п. и оформляется соответствующая документация (таблицы типов ошибок, частота и время появления отказов и др.). Данные, собранные при отладке, просмотрах, инспекции и тестировании, используются для оценки характеристик качества готового ПО, например, атрибутов *показателя надежности – интенсивность отказов*, количество ошибок и др.

Область знаний "Тестирование ПО (Software Testing)" включает следующие разделы:

- основные концепции и определение тестирования (Testing Basic Concepts and definitions),
- уровни тестирования (*Test Levels*),
- техники тестирования (Test Techniques),
- метрики тестирования (Test Related Measures),
- управление процессом тестирования (Managing the *Test Process*).

Данная область знаний SWEBOK представляет разработчику методы проверки правильности ПО: верификация, валидация, тестирование. Определяются типы, уровни и техники тестирования ПО, методы *планирования процесса* тестирования и разработки тестовых наборов данных для прогонки ПО в режиме испытания конкретного модуля или системы в целом, с последующим измерением результатов тестирования.

Основная концепция тестирования описывает базовые термины, ключевые проблемы и их связь с другими областями знаний. Тестирование – это процесс проверки правильности программы в динамике ее выполнения на тестовых данных. При тестировании выявляются недостатки: отказы (faults) и дефекты (*defects*) как причины нарушения работы программы, сбои (failures) как нежелательные ситуации, ошибки (errors), как последствия сбоев и др. Базовым понятием тестирования является тест, который выполняется в заданных условиях и на заданных наборах данных. Тестирование считается успешным, если найден дефект или ошибка, и они сразу устраняются. Степень тестируемости определяется критерием покрытия системы тестами, проверки всех возможных путей выполнения алгоритмов программ и вероятности статического предположения того, что при тестировании может появиться сбой или ошибочная ситуация в системе.

Уровни тестирования:

тестирование отдельных элементов,которое заключается в проверке отдельных, изолированных и независимых частей ПО;

интеграционное тестирование,которое ориентировано на проверку связей и способов взаимодействия (интерфейсов) компонентов друг с другом, включая компоненты, расположенные на разных архитектурных платформах распределенной среды;

тестирование системы предназначено для проверки правильности функционирования системы в целом, с обнаружением отказов и дефектов в системе и их устранение. При этом контролируется выполнение сформулированных нефункциональных требований (безопасность, надежность и др.) в системе, правильность задания и выполнения внешних интерфейсов системы со средой окружения и др.

На всех уровнях тестирования применяются методы:

функционального тестирования, обеспечивающие проверку реализации функций, которые определены в требованиях, а также правильность их выполнения;

регрессионного тестирования, ориентированные на повторное выборочное тестирование системы или ее компонентов после внесения в них изменений на тех же тестах, что и до модификации;

тестирования эффективности, проверяющие производительность, пропускную способность, максимальный объем данных и системные ограничения в соответствии со спецификациями требований;

стресс-тестирования, проверяющие поведение системы при максимально допустимой нагрузке или при ее превышении;

альфа и бета тестирования, выполняющие внутреннее тестирование кодов системы и внешнее тестирование интерфейсов. Альфа - это внутреннее тестирование (функций и алгоритмов), а бета - внешнее (взаимосвязей с другими системами и средой)

конфигурационного тестирования, проверяющие структуры и идентификации системы на различных наборах данных, а также работу системы на различных конфигурациях аппаратуры и оборудования.

К методам тестирования относятся также методы проведения испытаний ПО, проверки *реализации требований* и обеспечения параметров настройки и размещения компонентов ПО на заданном количестве и типах компьютеров, среды и ОС.

Техники тестирования базируются на некоторых теоретических и практических положениях, например, на природе подхода к проектированию (компонентного, объектно-ориентированного, сервисного и т.п.), а также на следующих данных:

информация о структуре ПО или системы в документации ("белый ящик" *)* ;

подбор тестовых наборов данных для проверки правильности работы компонентов и системы в целом без знания их структуры ("черный ящик");

анализ граничных значений, таблиц принятия решений, потоков данных, статистики отказов и др.;

блок-схемы построения программ и составления наборов тестов для покрытия системы этими тестами;

обнаруженные и зафиксированные в таблицах системы дефекты, пред- и постусловия выполнения, структурные характеристики системы (количество модулей, объем данных и др.).

Метрики тестирования.Для измерения результатов тестирования ПО, а также при проведении анализа качества используются метрики. Измерение как часть планирования и разработки тестов базируется на размере программ, их структуре и количестве обнаруженных ошибок и дефектов. Метрики тестирования обеспечивают измерение процесса планирования, проектирования и тестирования; а также результатов тестирования на основе таксономии отказов и дефектов, покрытия границ тестирования, проверки потоков данных и др. Процесс тестирования документируется и согласно стандарту IEEE 829-98 включает описание тестовых документов, их связи между собой и с задачами тестирования. Без документации по процессу тестирования невозможно провести сертификацию продукта по модели CMM [1.20]. После завершения тестирования рассматриваются вопросы стоимости и оценки рисков, вызванных сбоями или недостаточно надежной работой системы. Стоимость тестирования является одним из ограничений, на основе которого принимается решение о прекращении или его продолжении.

Управление тестированием:

планирование процесса тестирования (составление планов, тестов, наборов данных) и оценивание показателей качества ПО;

проведение тестирования компонентов повторного использования и паттернов как основных объектов сборки ПО;

генерация необходимых тестовых сценариев, соответствующих среде выполнения ПО;

верификация правильности реализации системы и валидация правильности *реализации требований* к ПО;

сбор данных об отказах, ошибках и др. непредвиденных ситуациях при выполнении программного продукта;

подготовка отчетов по результатам тестирования и оценка характеристик системы.

Заметим, что стандарт ISO/IEC 12207 и гармонизированный ГОСТ 12207 не выделяет деятельность по тестированию в качестве самостоятельного процесса, а рассматривает тестирование как неотъемлемую часть всего ЖЦ.

1.1.5. Сопровождение ПО (Software maintenance)

Сопровождение ПО – совокупность действий по обеспечению работы ПО, а также по внесению изменений в случае обнаружения ошибок в процессе эксплуатации, по адаптации ПО к новой среде функционирования, а также по повышению производительности или улучшению других характеристик ПО. В связи с решением проблемы 2000 года сопровождение стало рассматриваться как более важный процесс, который должны осуществлять разработчики. Новая версия системы должна решать те же самые задачи, иметь план переноса информации в другие обновленные БД и учета стоимости сопровождения. Сопровождение (в соответствии со стандартами ISO/IEC 12207 и ISO/IEC 14764) считается модификацией программного продукта в процессе эксплуатации при условии сохранения целостности продукта.

Область знаний "Сопровождение ПО (*Software maintenance*)" состоит из следующих разделов:

основные концепции (Basic Concepts),

процесс сопровождения (Process Maintenance),

ключевые вопросы сопровождения ПО (key Issue in *Software Maintenance*) ,

техники сопровождения (Techniques for Maintenance)

Сопровождение рассматривается с точки зрения удовлетворения требований к созданному ПО, корректности его выполнения, процессов обучения и оперативного учета процесса сопровождения.

Основные концепции описывают базовые определения и терминологию, подходы к эволюции и сопровождению ПО, а также к оценке стоимости сопровождения и др.

К основным концепциям можно отнести ЖЦ ПО (стандарт ISO/IEC 12207) и составление документации. Главное назначение этой области знаний состоит в выполнении готовой программной системы, фиксации возникающих ошибок при выполнении, исследовании причин ошибок, анализа необходимости модификации системы в целях устранения ошибок, оценки стоимости работ по проведению изменений функций и системы в целом. Рассматриваются проблемы, связанные с увеличением сложности продукта при большом количестве изменений и методы ее преодоления.

Процесс сопровождения включает: модели процесса сопровождения и планирование деятельности людей, которые проводят запуск ПО, проверку правильности его выполнения и внесения в него изменений. Процесс сопровождения согласно стандарту ISO/IEC 14764 проводится путем:

корректировки, т.е. изменения продукта для устранения обнаруженных ошибок или нереализованных задач;

адаптации, т.е. настройки продукта в изменившихся условиях эксплуатации или в новой среде выполнения данного ПО;

улучшения, т.е. эволюционного изменения продукта для повышения производительности или уровня сопровождения;

проверки ПО с целью поиска и исправления ошибок, обнаруженных при эксплуатации системы.

Ключевые вопросы сопровождения ПО.Основными из этих вопросов являются управленческие, измерительные и стоимостные. Сущность управленческих вопросов состоит в контроле ПО в процессе модификации, совершенствовании функций и недопущении снижения производительности системы. Вопросы измерения связаны с оценкой характеристик системы после ее модификации, а также повторного тестирования и оценки показателей качества. Стоимостные вопросы связаны с оценкой затрат на сопровождение ПО в зависимости от его типа, квалификации персонала, платформы и др. Знание этих факторов часто позволяет уменьшить затраты.

Эволюция ПО.Известный специалист в области ПО Дж. Леман (1970г.) предложил рассматривать сопровождение как эволюционную разработку программных систем, поскольку сданная в эксплуатацию система не всегда является полностью завершенной, ее надо изменять в течение срока эксплуатации. В результате программная система становится более сложной и плохо управляемой, возникает проблема уменьшения ее сложности. К технологиям эволюции ПО относятся реинженерия, реверсная инженерия и рефакторинг.

Реинженерия – это усовершенствование устаревшего ПО путем его реорганизации или реструктуризации, а также перепрограммированием отдельных элементов или настройки параметров на другую платформу или среду выполнения с сохранением удобства его сопровождения.

Реверсная инженерия состоит в восстановлении спецификации (*графов вызовов*, потоков данных и др.) по полученному коду системы для наблюдения за ней на более высоком уровне. Восстанавливается

идентификация программных компонентов и связей между ними для обеспечения перепрограммирования системы к новой форме.

Чаще всего реверсная инженерия применяется после того, как в код ПО было внесено много изменений и оно стало неуправляемым.

Рефакторинг – это реорганизация кода для улучшения характеристик и показателей качества объектно-ориентированных и компонентных программ без изменения их поведения. Этот процесс реализуется путем постепенного изменения отдельных операций над текстами, интерфейсами, средой программирования и выполнения ПО, а также настройки или внесения изменений в инструментальные средства поддержки ПО. Если сохраняется форма существующей системы при изменении, то рефакторинг – один из вариантов реверсной инженерии.

1.1.6. Управление конфигурацией ПО

Управление конфигурацией (Software Configuration Management – *SCM*) состоит в идентификации компонентов системы, определении функциональных и физических характеристик аппаратного и программного обеспечения для контроля за внесением изменений и трассированием конфигурации на протяжении ЖЦ. Это управление соответствует одному из вспомогательных процессов ЖЦ (ISO/IEC 12207), выполняется техническим и административным руководством проекта; составляются отчеты об изменениях, внесенных в конфигурацию, и степени их реализации, а также проводится проверка соответствия внесенных изменений заданным требованиям.

Конфигурация системы – состав функций, программного и технического обеспечения системы, возможные их комбинации в зависимости от наличия оборудования, общесистемных средств, обозначенных в технической документации системы, и требования к продукту.

Конфигурация ПО включает набор функциональных и технических характеристик ПО, заданных в технической документации и реализованных в готовом продукте. Это сочетание разных элементов продукта вместе с заданными процедурами сборки и настройки на среду в соответствии с назначением системы. Примерами элементов конфигурации являются график разработки, проектная документация, исходный и исполняемый код, библиотека компонентов, инструкции по установке и развертыванию системы.

Область знаний "Управление конфигурацией ПО" состоит из следующих разделов:

- управление процессом конфигурации (Management of *SCM* Process),
- идентификация конфигурации ПО (Software *Configuration Identification*),
- контроль конфигурации ПО (Software *Configuration Control*),
- учет статуса (положение конфигурации в ПО или состояние) конфигурации ПО (Software Configuration *Status Accounting*),
- аудит конфигурации ПО (Software *Configuration Auditing*),
- управление версиями ПО и доставкой (Software Release Management and Delivery).

Управление процессом конфигурации.Это деятельность по контролю эволюции и целостности продукта при идентификации, контроле изменений и обеспечении отчетной информацией, касающейся конфигурации. Включает:

- систематическое отслеживание вносимых изменений в отдельные составные части конфигурации, выполнение аудита изменений и автоматизированного контроля за внесением изменений в конфигурацию системы или ПО;
- поддержку целостности конфигурации, ее аудит и обеспечение внесения изменений в элементы конфигурации;
- ревизию конфигурации в целях проверки наличия разработанных программных или аппаратных средств и согласования версии конфигурации с заданными требованиями;
- трассировка изменений в конфигурацию на этапах сопровождения и эксплуатации ПО.

Идентификация конфигурации ПО заключается в документировании функциональных и физических характеристик элементов конфигурации ПО, а также в оформлении технической документация на элементы конфигурации ПО.

Контроль конфигурации ПО состоит в проведении работ по координации, утверждению или отбрасыванию реализованных изменений в элементах конфигурации после формальной ее идентификации, а также в анализе входящих компонентов в конфигурацию и соответствия их идентификации.

Учет статуса или состояния конфигурации ПО проводится с помощью комплекса мероприятий, позволяющих определить степень изменения конфигурации, полученной от разработчика, а также правильность внесенных изменений в конфигурацию ПО при ее сопровождении. Информация и количественные показатели накапливаются в соответствующей БД и используются при управлении конфигурацией, составлении отчетности, оценивании качества и выполнении других процессов ЖЦ.

Аудит конфигурации – это деятельность, которая выполняется для *оценки продукта* и процессов на соответствие стандартам, инструкциям, планам и процедурам. Аудит определяет степень, в которой элемент конфигурации удовлетворяет заданным функциональным и физическим (аппаратным) характеристикам системы. На основе функционального и физического аудита конфигурации фиксируется базовая линия изготовленного продукта.

Управление версиями ПО – это отслеживание имеющейся версии элемента конфигурации; сборка компонентов; создание новых версий системы на основе существующих путем внесения изменений в конфигурацию; согласование версии продукта с требованиями и проведенными изменениями на этапах ЖЦ; обеспечение оперативного доступа к информации об элементах конфигурации и системы, к которым они относятся. Управление выпуском охватывает идентификацию, упаковку и передачу элементов продукта и документации заказчику. При этом используются следующие основные понятия.

Базис (baseline) – формально обозначенный набор элементов ПО, зафиксированный на этапах ЖЦ ПО.

Библиотека ПО – контролируемая коллекция объектов ПО и документации, предназначенная для облегчения процесса разработки, использования и сопровождения ПО.

Сборка ПО – объединение корректных элементов ПО и конфигурационных данных в единую исполняемую программу.

1.1.7. Управление инженерией ПО

Управление инженерией ПО (Software *Engineering Management*) – руководство работами команды разработчиков ПО в процессе выполнения плана проекта, определение критериев эффективности работы команды и оценка процессов и продуктов проекта с использованием общих методов управления, планирования и контроля работ.

Как любое управление, менеджмент ПО базируется на планировании, координации, измерении, контроле и учете процесса управления проектом. Координацию людских, финансовых и технических ресурсов при реализации задач программного проекта выполняет менеджер проекта, аналогично тому, как это делается в технических проектах. В его обязанности входит соблюдение запланированных бюджетных и временных характеристик и ограничений, стандартов и сформулированных требований. Общие вопросы управления проектом содержатся в ядре знаний PMBOK [1.21] в разделе Management Process Activities, а также в стандарте ISO/IEC 12207 – *Software life cycle* processes [1.14], где управление проектом рассматривается как дополнительный и организационный процесс ЖЦ.

Область знаний "Управление инженерией ПО (Software *Engineering Management*)" состоит из следующих разделов:

- организационное управление (Organizational Management),
- управление процессом и проектом (Process/Project Management),
- инженерия измерений ПО (Software Engineering Measurement).

Организационное управление – это планирование и составление графика работ, оценка стоимости работ, подбор и управление персоналом, контроль за выполнением работ согласно принятым стандартам и планам. Главными проблемами организационного управления проектом являются: управление персоналом (обучение, мотивация и др.), коммуникации между сотрудниками (сценарии, встречи, презентации и др.), а также риски (минимизация риска, техники определения риска и др.). Для управления проектом создается определенная структура коллектива, специалисты распределяются по работам и решают задачи проекта под руководством менеджера с учетом заданной стоимости и сроков разработки. Для задач проекта подбираются также необходимые программные, инструментальные и аппаратные средства.

Управление проектом/процессом включает: составление плана проекта, построение графиков работ (сетевых или временных диаграмм) с учетом имеющихся ресурсов, распределение персонала по работам проекта, исходя из заданных сроков и стоимости их выполнения. Для эффективного управления проектом проводится анализ финансовой, технической, операционной и социальной политики организации разработчика для выбора правильной стратегии выполнения плана, контроля процесса управления планами и выпуском промежуточных продуктов (проектных решений, диаграмм UML, алгоритмов и др.).

В задачи управления проектом входят также уточнение требований, проверка их на соответствие заданным спецификациям характеристик качества, а также верификация функций отдельных продуктов проекта. Процесс управления проектом базируется на плановых сроках выполнения работ. Результаты планирования отображаются в *сетевых диаграммах PERT* (Program Evaluation and Review Technique), CPM (Critical Path Method) и др., предназначенных для отображения всех аспектов работ, в частности, времени их выполнения и связей между разными работами в проекте.

На сегодняшний день наиболее распространенным представлением сети для управления разными видами работ является сетевая диаграмма *PERT* – граф, в вершинах которого располагаются работы, а дуги задают взаимные связи между этими работами. Другой тип сетевой диаграммы, CPM, является событийным. В вершинах такой диаграммы указываются события, а работы задаются линиями между двумя узлами событиями.

Ожидаемое время выполнения работы для сетевых диаграмм оценивается с помощью среднего весового значения трех оценок: оптимистической, пессимистической и ожидаемой, т.е. вероятностной. Эти оценки берутся из заданного времени на разработку и заключений экспертов, оценивающих как отдельные работы, так и весь комплекс работ. Есть и другие методы оценок.

После составления плана решается вопрос управления проектом и контроля работ в соответствии с планом, выбранным процессом и сущностью проекта. Корректно составленный план обеспечивает выполнение требований и целей проекта. Контроль осуществляется при внесении изменений в проект, направлен на оценку риска и принимаемых решений по минимизации рисков.

Важной проблемой выполнения проекта является процесс определения рисков и разработки мероприятий по уменьшению их влияния на ход выполнения проекта. Под риском понимается вероятность проявления неблагоприятных обстоятельств, которые могут повлиять негативно на управление разработкой (например, увольнение сотрудника и отсутствие замены для продолжения работ и др.). При составлении плана проекта проводится идентификация и анализ риска, планирование непредвиденных ситуаций, касающихся рисков. Предотвращение риска заключается в выполнении действий, которые снимают риск (например, увеличение времени разработки и др.), что уменьшает вероятность появления нового риска при реорганизации проекта, БД или транзакций, а также при выполнении ПО.

Инженерия измерений ПО проводится в целях определения отдельных характеристик продуктов и процессов, инженерии планирования и измерения этих характеристик (например, количество строк в продукте, ошибок в спецификациях и т.п.). Предварительно проводятся работы по выбору метрик процессов и продуктов с учетом обстоятельств и зависимостей, влияющих на измерение их характеристик. К аспектам инженерии измерений относятся совершенствование процессов управления проектом; оценки временных затрат и стоимости ПО, их регулирование; определение категорий рисков и отслеживание факторов для регулярного расчета вероятностей их возникновения; проверка заданных в требованиях показателей качества отдельных продуктов и проекта в целом [1.15].

Проведение разного рода измерений является важным принципом любой инженерной деятельности. В программном проекте результаты измерений необходимы заказчику и потребителям, чтобы установить, действительно ли проект был реализован правильно. Без измерений в инженерии ПО процесс управления становится неэффективным и превращается в самоцель.

1.1.8. Процесс инженерии ПО (Software Engineering Process)

В некотором смысле это *метауровень*, который связан с определением, реализацией, оценкой, измерением, управлением изменениями и совершенствованием самого процесса. Однако такой процесс не является единственно правильным способом выполнения задач программной инженерии. На самом деле стандарты о процессах (ГОСТ 12207) говорят о том, что процессов много, например основной процесс разработки (*Development Process*), процесс управления конфигурацией (Configuration Management Process) и т.п.

Для оценивания и совершенствования процессов программной инженерии используется модель зрелости (Capability *Maturity* Models – CMM), эту концепцию разработал институт программной инженерии *SEI* (Software Engineering Institute) США. Модель описывает существенные атрибуты, которыми должен обладать процесс, находящийся на определенном уровне зрелости, а также указывает практические приемы обеспечения уровня абстракции без ограничений на способы реализации процесса в конкретном проекте. Разновидностями этой модели являются: CMM – SW (software) для оценки ПО, CMMI – вариант CMM для учета потребностей крупных государственных структур США, *Bootstrap* – вариант CMM для малых и средних коммерческих компаний, ISO – 15504 (Software Process Improvement and Capability – *SPICE*), *ISO 9000*-3 как приложение к общей модели качества *ISO 9001* и др.

Концепция зрелости процесса ПО основывается на интеграции концепции процесса ПО (software process), широты возможностей процесса ПО (software process capability), результативности процесса ПО (software process performance) и зрелости процесса ПО (software process *maturity*). Процесс ПО в модели CMM – это множество деятельностей (activities), методов (methods), практических приемов и процедур (practices), используемых при разработке ПО и связанных с ним продуктов (например, планов проекта, проектных документов, кода, тестов, руководства пользователя и др.).

Зрелость процесса – это степень его четкости определения, управления, измерения, контроля.

CMM предоставляет шаблон для улучшения процесса в виде пяти уровней зрелости, которые создают основу непрерывного улучшения процесса. Эти пять уровней зрелости определяют шкалу упорядочения для измерения зрелости процесса ПО организации и оценивании возможностей этого процесса. Уровень зрелости – это определенные средства для достижения зрелого процесса, а именно, множество целей процесса, которые, в случае их достижения, стабилизируют процесс в *проектной организации*.

Область знаний "Процесс программной инженерии (Software *Engineering Process*)" состоит из следующих разделов:

- концепции процесса инженерии ПО (Software *Engineering Process* Concepts),
- инфраструктура процесса (Process Infrastructure),

определение процесса (Process Definition),

оценки процесса (Process Assessments),

количественный анализ процесса (Qualitative Process Analysis),

выполнение и изменение процесса (Process Implementation and Change).

Концепции процесса инженерии ПО – задачи и действия, которые связаны с управлением, реализацией, оценкой, изменениями и совершенствованием процесса и/или ПО. Цель управления процессом – это создание инфраструктуры процесса, выделение необходимых ресурсов, планирование реализации и изменения процесса в целях внедрения его в практику и, наконец, оценка преимущества от его внедрения при реальной практике проектирования конкретного проекта.

Инфраструктура проекта – это его ресурсы (человеческие, технические, информационные и программные), стандарты, службы управления качеством и риском, а также форма организации производства проектов: бригада, экспериментальная фабрика (*Experimental* Factory), линейка программных продуктов (Framework for Product Line Practice) и др. Она также включает управление и коммуникации в коллективе, инженерные методы организации и производства программного продукта. Главная задача инфраструктуры – совершенствование процесса с учетом опыта разработки ПО.

Определение процесса основывается на: типах процессов и моделей (водопадная, спиральная, итерационная и др.); моделях ЖЦ процессов и средств, стандартах ЖЦ ПО ISO/IEC 12207 и 15504, IEEE std. 1074-91 и 1219-92; а также методах и нотациях задания процессов и автоматизированных средствах их поддержки. Основной целью процесса является повышение качества получаемого продукта, улучшение различных аспектов программной инженерии, автоматизация процессов и др.

Оценка процесса проводится с использованием соответствующих моделей и методов оценки. Например, оценка потенциальной способности специалиста к выполнению соответствующей работы. SWEBOK обращает внимание на процедуру оценки потенциальной возможности заключения контракта на разработку, организации разработки ПС на основе модели оценки зрелости (CMM) и процессов, согласно которым проводится разработка ПО.

Оценки относятся также и к техническим работам в сфере программной инженерии, к управлению персоналом и качеству ПО. Для этого проводятся экспериментальные исследования среды, наблюдение за выполнением процесса, сбор информации, моделирование, классификация полученных ошибок и дефектов, а также статический анализ недостатков процесса в сравнении с существующими стандартами (например, ГОСТ 12207) и потенциальных аспектов совершенствования процесса.

Качественный анализ процесса состоит в идентификации и поиске слабых мест в процессе создания ПО до начала его функционирования. Рассматривается две техники анализа: обзор данных и сравнение процесса с основными положениями стандарта ISO/IEC 12207; сбор данных о качестве процессов; анализ главных причин отказов в функционировании ПО, откат назад от точки возникновения отклонения до точки нормальной работы ПО для выяснения причин изменения процесса. На качество результатов проекта и процесса влияют применяемые инструменты и опыт специалистов.

Выполнение и изменение процесса. Существует ряд фундаментальных аспектов измерений в программной инженерии, лежащих в основе детальных измерений процесса. Оценка совершенствования процессов проводится для установления количественных характеристик процесса и продуктов. После процесса развертывания ПО, выполняются вычисления функций и инспекция результатов выполнения. Результаты процесса могут касаться оценки качества продукта, продуктивности, трудозатрат и др. Если результаты не удовлетворяют пользователя ПО, то проводится анализ и принятие решений о необходимости изменения или усовершенствования процесса, организационной структуры проекта и некоторых инструментов управления измерениями.

1.1.9. Методы и инструменты инженерии ПО

Методы обеспечивают проектирование, реализацию и выполнение ПО. Они накладывают некоторые ограничения на инженерию ПО в связи с особенностями применения их нотаций и процедур, а также обеспечивают оценку и проверку процессов и продуктов. Инструменты являются программной поддержкой отдельных методов инженерии ПО и обеспечивают автоматизированное выполнение задач процессов ЖЦ.

Область знаний "Методы и инструменты инженерии ПО (Software Engineering Tools and Methods)" состоит из разделов:

инструменты инженерии ПО (Software Engineering Tools),

методы инженерии ПО (Software Engineering Methods).

Методы инженерии ПО – это эвристические методы (*heuristic methods*), формальные методы (*formal methods*) и методы прототипирования (prototyping methods),.

Эвристические методы включают: *структурные методы*, основанные на функциональной парадигме; методы, ориентированные на структуры данных, которыми манипулирует ПО; объектно-ориентированные методы, которые рассматривают предметную область как коллекцию объектов, а не функций; методы,

ориентированные на конкретную область применения, например, на системы реального времени, безопасности и др.

Формальные методы основаны на формальных спецификациях, анализе, доказательстве и *верификации программ*. Формальная спецификация записывается на языке, синтаксис и семантика которого определены формально и основаны на математических концепциях (алгебре, теории множеств, логике). Различаются следующие категории формальных методов:

- *языки и нотации спецификации* (specification languages and notations), ориентированные на модель, свойства и поведение;
- *уточнение спецификации* (refinement specification) путем трансформации в конечный результат, близкий к конечному исполняемому программному продукту;
- *методы доказательства/верификации* (verification/*proving* properties), использующие утверждения (теоремы), пред- и постусловия, которые формально описываются и применяются для установления правильности *спецификации программ*.

Эти методы применялись в основном в теоретических экспериментах и более 25 лет их практическое применение было ограничено из-за трудоемкости и экономической невыгодности. В 2005 г. проблема верификации приобрела вновь актуальность в связи с разработкой нового международного проекта по верификационному ПО "Целостный автоматизированный набор инструментов для проверки корректности ПС" (идея Т. Хоара, "Открытые системы", 2006, № 6), ставящим следующие *перспективные задачи*:

- разработка единой теории построения и анализа программ;
- построение многостороннего интегрированного набора инструментов верификации на всех производственных процессах – разработка формальных спецификаций, их доказательство и проверка правильности, генерация программ и тестовых примеров, уточнение, анализ и оценка;
- создание репозитария формальных спецификаций, верифицированных программных объектов разных типов и видов.

Предполагается, что формальные методы верификации будут охватывать все аспекты создания и проверки правильности программ. Это приведет к созданию мощной верификационной производственной основы и значительному сокращению ошибок в ПО (доказательству и верификации будет посвящена тема 7.)

Методы прототипирования (Prototyping Methods) основаны на прототипировании ПО и подразделяются на:

- стили прототипирования, включающие в себя создание временно используемых прототипов (throwaway), эволюционное прототипирование – превращение прототипа в конечный продукт и разработка исполняемых спецификаций;
- техники оценки/исследования (evaluation) результатов прототипирования.

Инструменты инженерии ПО обеспечивают автоматизированную поддержку процессов разработки ПО и включают множество разных инструментов, охватывающих все процессы ЖЦ.

Инструменты работы с требованиями (Software Requirements Tools) – это:

- инструменты разработки (*Requirement Development*) управления требованиями (*Requirement Management*) для анализа, сбора, специфицирования и проверки требований. Например, в модели CMMI *Staged* на 2-м уровне зрелости находится управлении требованиями, а на 3-м уровне – разработка требований;
- инструменты *трассировки требований* (Requirement *traceability* tools) являются неотъемлемой частью работы с требованиями, их функциональное содержание зависит от сложности проектов и уровня зрелости процессов.

Инструменты проектирования (Software Design Tools) – это инструменты для создания ПО с применением базовых нотаций (*SADT/IDEF*, UML, Microsoft DSL, Oracle и т.п.).

Инструменты конструирования ПО (Software Construction Tools) – это инструменты для производства, трансляции программ и машинного выполнения. К ним относятся:

- редакторы (*program editors*) для создания и модификации программ, и редакторы "общего назначения" (UNIX и UNIX-подобные среды);
- компиляторы и генераторы кода (compilers and *code generators*) как самостоятельные средства объединения в интегрированной среде программных компонентов для получения выходного продукта с использованием препроцессоров, сборщиков, загрузчиков и др.;
- интерпретаторы (interpreters) обеспечивают исполнение программ путем эмуляции, предоставляя для исполнения программ контролируемое и наблюдаемое окружение. Наметилась тенденотладчики (debuggers) для проверки правильности описания исходных программ и устранения ошибок;
- интегрированные среды разработки (IDE – integrated *developers environment*), библиотеки компонент (libraries components), без которых не может проводится процесс разработки ПС, программные

платформы (Java, J2EE и Microsoft .NET) и платформа распределенных вычислений (CORBA и WebServices).

Инструменты тестирования (Software Testing Tools) это:

генераторы тестов (*test generators*), помогающие в разработке сценариев тестирования;

средства выполнения тестов (*test execution* frameworks) обеспечивают выполнение тестовых сценариев и отслеживают поведение объектов тестирования;

инструменты оценки тестов (test evaluation tools) поддерживают оценку результатов выполнения тестов и степени соответствия поведения тестируемого объекта ожидаемому поведению;

средства управления тестами (*test management tools*) обеспечивают инженерию процесса тестирования ПО;

инструменты анализа производительности (*performance analysis* tools), количественной ее оценки и оценки поведения программ в процессе выполнения.

Инструменты сопровождения (Software Maintenance Tools) включают в себя:

инструменты облегчения понимания (*comprehension* tools) программ, например, различные средства визуализации;

инструменты реинжинерии (reengineering tools) поддерживают деятельность по реинжинерии и обратной инженерии (*reverse engineering*) для восстановления (артефактов, спецификация, архитектуры) стареющего ПО и генерации нового продукта.

Инструменты конфигурационного управления (Software Configuration Management Tools) – это:

инструменты отслеживания (tracking) дефектов;

инструменты управления версиями;

инструменты управления сборкой, выпуском версии (конфигурации) продукта его инсталляции.

Инструменты управления инженерной деятельностью (Software Engineering Management Tools) состоят из:

инструментов планирования и отслеживания проектов, количественной оценки усилий и стоимости работ проекта (Microsoft Project 2003);

инструментов управления рисками используются для идентификации, мониторинга рисков и оценки нанесенного вреда;

инструментов количественной оценки свойств ПО путем ведения измерений и расчета окончательного значения надежности и качества.

Инструменты поддержки процессов (Software Engineering Process Tools) разделены на:

инструменты моделирования и описания моделей ПО (например, UML и его инструменты);

инструменты управления программными проектами (Microsoft Project 2003);

инструменты управления конфигурацией для поддержки версий и всех артефактов проекта.

Инструменты обеспечения качества (Software Quality Tools) делятся на две категории:

инструменты инспектирования для поддержки просмотра (review) и аудита;

инструменты статического анализа программных артефактов, данных, потоков работ и проверки свойств или артефактов на соответствие заданным характеристикам.

Дополнительные аспекты инструментального обеспечения (Miscellaneous Tool Issues) соответствуют таким аспектам:

техники интеграции инструментов (платформ, представлений, процессов, данных и управления) для естественного их сочетания в интегрированной среде

метаинструменты для генерации других инструментов;

оценка инструментов при их эволюции.

1.1.10. Качество ПО (Software Quality)

Качество ПО – набор свойств продукта (сервис или службы), которые характеризуют его способность удовлетворить установленные или предполагаемые потребности заказчика. Понятие качества имеет разные интерпретации в зависимости от конкретной системы

и требований к программному продукту. Кроме того, в разных источниках таксономия (классификация) характеристик в модели качества отличается.

Каждая модель имеет различное число уровней, и все известные модели качества имеют полное или частичное совпадение набора характеристик качества. Например, модель качества МакКолла на самом

высоком уровне имеет три характеристики: функциональность, модифицируемость и переносимость, а на более нижних уровнях модели, 11 подхарактеристик качества и 18 критериев (атрибутов) качества.

Стандарт ISO 9126-01 рассматривает *внешние и внутренние характеристики* качества. Первые отображают требования к функционирующему программному продукту. Для количественного задания критериев качества, по которым будет осуществляться проверка и подтверждение соответствия ПО заданным требованиям, определяются соответствующие внешние измеряемые свойства (внешние атрибуты) ПО и метрики. Ими могут быть конкретные значения (например, время выполнения отдельных компонентов), диапазоны изменения значений для некоторых внешних атрибутов и модели их оценивания. Метрики, применение которых возможно только для ПО, функционирующего на компьютере, используются на стадии тестирования или функционирования. Они называются *внешними метриками* и представляют собой модели оценивания атрибутов.

Внутренние характеристики качества и внутренние атрибуты ПО используются при составлении плана достижения необходимых внешних характеристик качества для конечного программного продукта. Для квантификации внутренних характеристик качества специфицируются внутренние метрики, использующиеся при проверке соответствия промежуточных продуктов спецификациям внутренних требований к качеству, которые формулируются на этапах, предшествующих тестированию (определение требований, проектирование, кодирование).

Внешние и внутренние характеристики качества отображают свойства самого ПО (работающего или не работающего), а также взгляд заказчика и разработчика на это ПО. Непосредственного конечного пользователя ПО интересует эксплуатационное качество ПО – совокупный эффект от достижения характеристик качества, который измеряется в сроках использования результата, а не свойств самого ПО. Это понятие шире, чем любая отдельная характеристика (например, удобство использования или надежность).

Окончательная оценка качества проводится в соответствии со стандартом ISO 15504-98 [1.15]. Качество может повышаться за счет постоянного улучшения используемого продукта в связи с процессами обнаружения, устранения и предотвращения сбоев/дефектов в ПО.

Область знаний "Качество ПО (*Software Quality*)" состоит из следующих разделов:

- концепция качества ПО (*Software Quality* Concepts);
- определение и планирование качества (Definition & Planning for Quality);
- техники и активности, обеспечивающие гарантию качества, валидацию и верификацию (Activities and Techniques for Software *Quality Assurance*, Validation -V & Verification – V);
- измерение при анализе качества ПО (Measurement in *Software Quality* Analysis).

Концепция качества ПО – это внешние и внутренние характеристики качества, их метрики, а также модели качества, определенные на множестве этих характеристик, которые представлены в стандартах качества [1.16] – это шесть характеристик и каждая из них имеет несколько атрибутов. К характеристикам качества относятся:

- функциональность (functionality);
- надежность (reliability);
- удобство применения (*usability*);
- эффективность (*efficiency*);
- сопровождаемость (maitainnability);
- переносимость (portability).

Базовая модель качества включает эти характеристики и относится к любому типу программных продуктов. При разработке требований заказчик формулирует те требования к качеству, которые наиболее подходят для заказываемого программного продукта.

Определение и планирование качества ПО основывается на положениях стандартов в этой области, составлении планов и графиков работ, процедурах проверки и др. План обеспечения качества включает набор действий для проверки процессов обеспечения качества (верификация, валидация и др.) и формирование документа по управлению качеством.

Планирование качества обеспечивает управление процессами обеспечения качества продуктов проекта (в частности промежуточных *рабочих продуктов*) и ресурсов – программных, технических,

исполнительских и др., а также включает управление требованиями к процессам и продуктам. Планирование качества включает:

- определение продукта в терминах заданных характеристик качества;
- *планирование процессов* для гарантии получения требуемого качества;
- выбор методов оценки планируемых характеристик качества и установления соответствия продукта сформулированным требованиям.

В стандарте 12207 определены специальные процессы: обеспечения качества, верификации, аттестации (валидации), совместного анализа и аудита.

Деятельности и техники гарантии качества включают: инспекцию, верификацию и валидацию ПО.

Инспекция ПО – анализ и проверка различных представлений системы и ПО (спецификаций, архитектурных схем, диаграмм, исходного кода и др.). Выполняется на всех этапах ЖЦ разработки ПО.

Верификация ПО – процесс обеспечения правильной реализации ПО, которое соответствует спецификациям, выполняется на протяжении всего жизненного цикла. Верификация дает ответ на вопрос, правильно ли создана система.

Валидация – процесс проверки соответствия ПО функциональным и нефункциональным требованиям и ожидаемым потребностям заказчика.

Верификация и валидация начинают выполняться на ранних стадиях ЖЦ и ориентированы на качество. Они планируются и обеспечиваются определенными ресурсами с четким распределением ролей. Проверка основывается на использовании соответствующих техник тестирования для обнаружения тех или иных дефектов и сбора статистики. В результате собранных данных проводится оценка правильности *реализации требований* и работы ПО в заданных условиях.

Измерение в анализе качества ПО основывается на метриках продукта и данных, собранных в процессе создания продукта при заданных ресурсах, на оценках процессов, ПО и его моделях, а также на документировании измерений. Оценка качества продукта – это измерение и оценивание качественных показателей с помощью данных о разных типах ошибок и отказах во время тестирования ПО и исполнения кода на тестовых данных. Эти данные анализируются,

проверяются и используются при качественной и количественной оценке ПО.

Для имитации работы системы в режиме тестирования разрабатываются тесты с реальными входными данными для проверки правильности работы ПО на разных фрагментах программы и путях следования в них операторов. В процессе тестирования ПО обнаруживаются разного рода ошибки (отказы, дефекты, ошибки и т.п.), количество которых в значительной степени может повлиять на получение правильного результата.

С учетом типов обнаруженных ошибок можно установить наличие (или отсутствие) соответствия реализованных и нереализованных функций, заданных в требованиях к системе, а также оценить принципы реализации нефункциональных требований (производительность, надежность и др.). Проводятся также *оценки процессов* управления планами, инспекциями, прогонами и т.п. По этим оценкам принимаются решения о завершении разработки продукта проекта и передаче его заказчику в опытную эксплуатацию или о необходимости внесения изменений для устранения ошибок, определения адекватности планов и требований, оценки рисков на переделку ПО и др.

Целью инспекций является обнаружение различных аномальных состояний в ПО независимыми специалистами команды экспертов и с привлечением авторов промежуточного или конечного продукта. Эксперты инспектирует выполнение требований, интерфейсы, входные данные и т.п., а затем документируют обнаруженные отклонения в проекте.

Назначением аудита является независимая *оценка продуктов* и процессов на соответствие регулирующим и регламентирующим документам (планам, стандартам и др.), формулирование отчета о случаях несоответствия и предложений для их корректировки.

1.2. Жизненный цикл ПС, связь с ядром знаний SWEBOK

Программная инженерия, как инженерная дисциплина, охватывает все аспекты создания ПС от начальной стадии разработки системных требований до реализации программного продукта и его использования.

Под *программной системой* (ПС) понимается комплекс интегрированных программ и средств, которые реализуют функции *предметной области* в заданной среде. В комплекс могут входить: прикладные системы (зарплата, учет и др.), общесистемные компоненты (отладчик, редактор, *СУБД*), системы защиты и безопасности и др.

Программное обеспечение – это некоторая конкретная функции системы (например, программный *модуль* решения одной общей задачи, ОС – функционирование программ и систем, *управление данными* и др.). *ПО* может входить в состав ПС или быть идентичным ПС.

Каждая ПС на протяжении своего существования проходит определенную последовательность *процессов* (этапов), начиная от постановки задачи до ее воплощения в готовую программу, эксплуатации и изъятия из эксплуатации. Такая последовательность этапов называется *жизненным циклом (ЖЦ)* разработки ПС. На каждом этапе ЖЦ выполняется определенная совокупность процессов и/или подпроцессов, каждый из которых порождает соответствующий промежуточный продукт, используя при этом результаты предыдущего процесса и продукта.

Все продукты программной инженерии представляют собой некоторые описания, а именно, тексты требований к разработке ПС, согласованные договоренности с заказчиком, архитектуру, структуру данных, тексты

программ, документацию, инструкции *по* эксплуатации и т.п. Главные *ресурсы* разработки ПС в программной инженерии – это сроки, время и *стоимость*. Правильное управление и использование этих ресурсов при выполнении задач и действий процессов ЖЦ определяет эффективность разработки ПС.

Разновидности действий и задач, представленные в процессах ЖЦ ПС, отображены в международном стандарте *ISO*\IEC 12207 (*таблица* 1) и связаны содержательно с областями знаний SWEBOK.

Данный стандарт устанавливает общую структуру и содержание ЖЦ ПС, начиная от разработки концепции до утилизации системы. Структура представляет собой множество процессов, взаимосвязей между ними и определений действий и задач на процессах ЖЦ. Иными словами, ЖЦ определяет, что надо делать, а не как выполнять те или иные действия.

Стандарт не обязывает использовать определенную модель ЖЦ *ПО* или конкретную методологию разработки *ПО* и не предъявляет требования к формату и содержанию создаваемых документов. Поэтому организация-*пользователь* этого стандарта должна для своей работы разработать стандарты предприятия, методики и процедуры, определяющие разные детали процесса разработки *ПО*.

Таблица 1.1. Процессы ЖЦ в стандарте ISO/IEC 12207

№ п/п	Наименование процессов (подпроцессов)
	1. Категория "Основные процессы"
1.1	Заказ (договор)
1.1.1	Подготовка заказа, выбор поставщика
1.1.2	Мониторинг деятельности поставщика, Приемка потребителем
1.2	Поставка (приобретение)
1.3	Разработка
1.3.1	Выявление требований
1.3.2	Анализ требований к системе
1.3.3	Проектирование архитектуры системы
1.3.4	Анализ требований к ПО системы
1.3.5	Проектирование ПО
1.3.6	Конструирование (кодирование) ПО
1.3.7	Интеграция ПО
1.3.8	Тестирование ПО
1.3.9	Системная интеграция
1.3.10	*Системное тестирование*
1.3.11	Инсталляция ПО
1.4	Эксплуатация
1.4.1	Функциональное использование
1.4.2	Поддержка потребителя
1.5	Сопровождение
	2. Категория "Процессы поддержки"
2.1	Документирование
2.2	Управление конфигурацией
2.3	Обеспечение гарантии качества
2.4	Верификация
2.5	Валидация
2.6	Общий просмотр
2.7	Аудит
2.8	Решение проблем

Отметим, что *ISO* выпустил ряд руководств и процедур, дополняющих стандарт *ISO*\IEC 12207. Основная идея данного стандарта – разработка и сопровождение ПС я так, как этого требует инженерная дисциплина. В процессе разработки создается каркас системы (абстрактная *архитектура* с вы деленными объектами), для которой определены среда, виды обеспечения, исполнители и сроки.

Как видно из табл. 1.1, все процессы в данном стандарте разделены на три категории:

основные процессы;

обеспечивающие (поддерживающие) процессы;

организационные процессы.

Для каждого из процессов определены виды деятельности (действия – *activity*), задачи, совокупность результатов (выходов) видов деятельности и задач, а также некоторые специфические требования. Стандарт дает перечень *работ* для основных обеспечивающих и организационных процессов. Пункты 1.1.1, 1.1.2, а также категории 2 и 3 процессов определяют виды деятельности, цели и задачи которых оговорены в стандарте, но не определяют форму их представления.

К основным процессам относятся:

процесс приобретения инициирует ЖЦ ПС и определяет действия организации покупателя (или заказчика), которая приобретает автоматизированную систему, программный продукт или сервис. Этот процесс включает следующие виды деятельности: инициирование и подготовка запроса, оформление контракта и его актуализация; мониторинг поставщиков, приемка и завершение;

процесс поставки определяет действия поставщика, по снабжению покупателя программным продуктом или сервисом. Данный процесс включает в себя следующие виды деятельности: подготовку предложений (ответов на запросы); оформление контракта; планирование, выполнение и контроль поставляемого продукта; анализ и оценку продукта; поставку и завершение работ по поставке. Процесс поставки начинается тогда, когда устанавливаются договорные отношения на поставку ПС между заказчиком и поставщиком. В зависимости от условий договора с заказчиком процесс поставки может включать процесс разработки ПО, процесс эксплуатации и обеспечения служб эксплуатации ПО или сопровождения для исправления и улучшения ПС;

процесс разработки определяет действия предприятия разработчика программного продукта: анализ требований к системе; проектирование архитектуры системы; детальное проектирование компонентов ПС; кодирование и тестирование ПС; интеграцию системы; квалификационное тестирование; установку ПС; обеспечение приемки ПС;

процесс эксплуатации определяет действия предприятия оператора, которое обеспечивает обслуживание системы в процессе ее эксплуатации пользователями (консультирование пользователей, изучение потребностей оператора с точки зрения удовлетворения их системой и т.д.). Этот процесс определяет задачи и действия по функциональному тестированию, проверочной эксплуатацией системы; предоставление пользователю необходимых инструкций и документации по запуску и эксплуатации ПО;

процесс сопровождения определяет действия организации, выполняющей сопровождение программного продукта (управление модификациями, поддержку текущего состояния и функциональной пригодности,

инсталляцию и удаление программного продукта на вычислительной системе пользователя). Данный процесс включает задачи и действия по анализу проблем сопровождения и модификации; разработке планов и реализации модификации системы; анализу результатов сопровождения после изменений системы; миграции (перемещению) ПС в другую среду или ее удалению из употребления.

К обеспечивающим процессам создания ПС относятся: *документирование*, управление версиями, *верификация* и *валидация*, просмотры, аудиты, *оценивание* продукта и др. Процессы управления версиями соответствуют управлению конфигурацией системы, которая так же, как и продукты процессов должна проверяться на правильность реализации целей проекта и соответствия требованиям заказчика. Задачи проверки рекомендуется выполнять специальным контролерам, обладающим знаниями методов и процессов проектирования ПС.

К организационным процессам отнесены: *управление проектом* (*менеджмент* разработки), качеством и рисками и др. Эти процессы выполняются специальными службами, выполняющими планирование *работ* на проекте, *контроль* процессов, *определение* метрик для измерения продуктов, проверку показателей качества, соблюдение стандартных положений и др.

Процессы, определенные в стандарте *ISO*/IEC 12207, охватывают все возможные задачи и действия *по* проектированию и разработке ПС. *Пользователь* стандарта может выбрать из них соответствующее *подмножество* для достижения конкретной цели, стоящей перед данным проектом. Процессы, действия и задачи приведены в стандарте в наиболее общей естественной последовательности. В зависимости от целей конкретного проекта процессы, действия и задачи выбираются, упорядочиваются и применяются итерационно или рекурсивно. Главный разработчик и *менеджер* должны определить задачи проекта, выбрать под их реализацию модель ЖЦ *ПО*, которая позволит учитывать ресурсы, *стоимость* и временные характеристики программного проекта.

Данный стандарт определяет содержание деятельности в сфере технологии разработки ПС, а знания, которые необходимы исполнителям для выполнения всех видов деятельности *по* проектированию и реализации поставленных задач перед проектом, определяют методы и средства областей ядра знаний SWEBOK, которые распределяются *по* отдельным процессам и подпроцессам при формировании модели ЖЦ для конкретного проекта.

Как уже отмечалось, между стандартом *ISO*\IEC 12207 и ядром знаний SWEBOK существует *связь* и взаимовлияние друг на друга, тем более что в разработке обоих документов примерно в одно время принимали участие высококвалифицированные специалисты в области программирования и информатики.

Общие идеи и методы программирования, сложившиеся в 90-х годах прошлого столетия, проникли в оба направления и оказали влияние на их структуру и содержание. Программисты-профессионалы систематизировали накопившиеся знания и создали 10 разделов, которые близки процессам ЖЦ *по* целям, задачам и видам деятельности. В ядре знаний SWEBOK они изложены как фундаментальные знания и инженерные методы управления разработкой *ПО*, а в стандарте как общие положения, структура и регламентированные процессы проектирования как рекомендации *по* разработке проекта, начиная от процесса постановки требований до эксплуатации *ПО*.

Следует отметить, что структура ядра знаний SWEBOK не лишена недостатков. Так, между областями знаний в этом ядре имеются пересечения, а некоторые важные направления в области программирования вообще не отражены в нем. Например, методы доказательства правильности программ, *эволюция* программ, распределенные и неоднородные среды, взаимодействие систем, некоторые методы программирования (сервисные, аспектные, агентные и др.), CASE-системы и др.

Технологические процессы в стандарте структурированы системно. Они не устанавливают *связь* с существующими методами и средствами программной инженерии, что создает возможность выбирать подходящие процессы стандарта и сопоставлять им привычные методы (объектные, компонентные, сервисные и др.), т.е. те, которые уже использовались коллективом разработчиков и для которых есть инструменты поддержки. Это и отражает идею совершенствования процесса инженерии *ПО*.

Процессы стандарта отвечают на вопрос, какие действия и задачи процессов ЖЦ надо выбрать, чтобы построить конкретную ПС. *Ядро* знаний SWEBOK отвечает на вопрос, какими методами, средствами и инструментами надо выполнять регламентированные действия и задачи процессов ЖЦ, чтобы построить ПС.

Программная инженерия сформировалась как инженерная дисциплина, которая базируются на теоретических и прикладных методах и средствах разработки ПС и стандартах (*ISO*/IEC 12207, 15404, *ISO* 9126 и др.), содержащих рекомендации, правила и методики управления разработкой ПС. Эти два базиса объединяет инженерия оценивания результатов на процессах ЖЦ, *управление качеством* ПС, оценка затраченных ресурсов на создание и учет стоимости *работ* участников разработки.

Инженерия программирования делает акцент на принципы, методы и подходы к управлению проектом, конфигурацией и качеством ПС. Стандарты регламентируют основные процессы, определяющие порядок проведения *работ по* реализации ПС, рекомендуют процессы организационной деятельности *по* управлению, планированию и оцениванию *работ*, выполняемых при проектировании и разработке ПС.

Ядро знаний SWEBOK, а также многочисленные монографии и статьи *по* методам и средствам программной инженерии предоставляют всю необходимую информацию для выбора наиболее подходящего метода, средства, инструмента. Вместе с тем стандарт – это руководство к построению процессов в сочетании с методами для реализации конкретного программного проекта на стандартизованной инженерной основе.

Естественно, что в небольших программных проектах всегда будет *место* творческим и неформальным подходам, вносимым отдельными профессионалами, при создании разного рода уникальных продуктов, процесс разработки которых не всегда вкладывается в общее стандартное русло.

Таким образом, *программная инженерия* базируется на ядре SWEBOK и стандартах ЖЦ. Инженерия производства ПС основывается на стандартных организационных процессах – планирования, управления и сопровождения. Цель планирования – составление планов и графиков *работ по* реализации проекта и распределение *работ* между разными категориями специалистов с учетом их квалификации и уровня знаний проблематики программной инженерии. Второй процесс обеспечивает привнесение методов управления в процесс выполнения *работ по* программированию, а именно, управление временем, стоимостью и сроками. Третий процесс – это выполнения проекта, обнаружение и устранение найденных недостатков и замена или внесения новых функций в систему.

Контрольные вопросы и задания

1. Назовите цели и задачи программной инженерии.
2. Назовите области знаний SWEBOK инженерии разработки ПО.
3. Приведите базовые понятия области знаний "Тестирование ПО".
4. Определите цели и задачи области знаний "Управление проектом".
5. Определите цели и задачи области знаний "Инженерия качества ПО".
6. Дайте определение ЖЦ разработки ПО.
7. Назовите три основные группы процессов жизненного цикла и перечислите процессы каждой из групп.
8. Назовите организационные процессы ЖЦ и перечислите их.
9. Дайте характеристику процесса управления качеством ЖЦ.
10. Какой международный стандарт определяет перечень и содержание процессов ЖЦ ПО?
11. Все ли процессы, указанные в стандарте, должны быть выполнены при каждой разработке ПО?

Внимание! Если Вы увидите ошибку на нашем сайте, выделите её и нажмите Ctrl+Enter.