МИНОБРНАУКИ РОССИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технологический университет «СТАНКИН»
(ФГБОУ ВО МГТУ «СТАНКИН»)

Кафедра "Информационные технологии и вычислительные системы"

Новоселова Ольга Вячеславовна

Современные технологии и средства разработки программного обеспечения (раздел 1)

Курс лекций
по дисциплине «Современные технологии и средства разработки программного обеспечения»

для студентов МГТУ «СТАНКИН», обучающихся

по направлению: 09.03.01 "Информатика и вычислительная техника" ,

по профилю: "Программное обеспечение средств вычислительной техники и автоматизированных систем"

Москва 2020г.

Оглавление.
7 семестр

Раздел 1. Основы организации жизненного цикла программного продукта.

7 семестр.

Раздел 1. Основы организации жизненного цикла программного продукта.

Лекция 1.

Тема 1. Программный продукт, как рыночный продукт.
Особенности программного продукта как рыночного продукта.
Виды программных продуктов.

План лекции.

1) Понятие «программный продукт», деление на виды
2) Выделение компонент в описании предметной задачи, подлежащей автоматизации
3) История программирования
4) Проблематика автоматизации интеллектуального труда
5) Классификация программного обеспечения

Основная часть.

1) **Понятие «программный продукт», деление на виды в соответствии с характером использования и категориям пользователей**

Программный продукт (ПП) - комплекс взаимосвязанных программ для решения определенной задачи (проблемы) массового спроса, подготовленный к реализации как любой др. вид промышленной продукции.

ПП должен быть соответствующим образом подготовлен к эксплуатации, иметь необходимую техническую документацию, предоставлять сервис и гарантию надежной работы, иметь товарный знак производителя, дополнительно желательно наличие кода государственной регистрации. Только тогда программный комплекс может быть назван программным продуктом.

Все программы по характеру использования и категориям пользователей можно разделить на два класса:

- **утилитарные программы** ("программы для себя") - предназначены для выполнения задач их разработчиков. Чаще всего утилитарные программы выполняют роль сервиса в технологии обработки данных либо являются программами решения функциональных задач, не предназначенных для широкого распространения;
- **программные продукты** (изделия) - предназначены для выполнения задач пользователей, широкого распространения и продажи.

ПП могут создаваться как:

- **индивидуальная разработка** под заказ;
- разработка **для массового распространения** среди пользователей.

Говоря о создании программных продуктов, необходимо заметить, что в первую очередь они призваны облегчить умственный (интеллектуальный) труд человека. Любой программный продукт обрабатывает информацию различных видов. Выделяют три вида информации, которая может храниться и обрабатываться в вычислительной среде:

- текстовая (вербальная);
- образная;
- звуковая.

2) Выделение компонент в описании предметной задачи, подлежащей автоматизации

При создании программных продуктов необходимо учитывать следующие особенности:
- при автоматизации решается две задачи: *анализ предметной задачи* для того, чтобы понять, как ее выполняет специалист и *разработка программного обеспечения*.

То есть существует информация о предметной области (или предметной задаче) и информация о задаче разработки программного продукта.

Для последней первая является входной. А выходной будет уже разработанные структуры информации и алгоритмы для программного продукта, а также сам программный продукт.
- информация о любой задаче может отражать разные аспекты задачи: функциональную часть, информационную часть, поведение.

Функциональная часть – отражает структуру функций автоматизированной предметной задачи, алгоритм действий, структуру действий *(названия: функциональная, процедурная, динамическая*).

Информационная часть – отражает данные, переменные, структуры данных, модели данных (в том числе математическую), описывающих объекты и свойства объектов предметной области (*названия: информационная, декларативная, статическая*).

Поведение – отражает изменение состояния системы.

3) История программирования

Этапы развития программирования как науки:

1. **«Стихийное» программирование** (охватывает период от момента появления первых вычислительных машин до середины 60-х годов XX века).

В этот период практически отсутствовали сформулированные технологии, программирование фактически было искусством. Первые программы имели простейшую структуру. Они состояли из программы на машинном языке и обрабатываемых ею данных. Сложность программ в машинных кодах ограничивалась способностью программиста одновременно мысленно отслеживать последовательность выполняемых операций и местонахождение данных при программировании.

Сначала появились языки низкого уровня – ассемблеры, и стало возможным использовать символические имена данных вместо кодов (двоичного или 16-ричного) и мнемоники кодов операций. Программы стали более читаемыми. Далее появились языки высокого уровня (FORTRAN и ALGOL), это понизило уровень детализации операций, и упростило программирование вычислений. В дальнейшем появились средства, которые позволяют оперировать подпрограммами, это дало возможность создать библиотеки расчетных и служебных подпрограмм, которые по мере надобности вызывались из разрабатываемой программы. Программы в то время состояли из области глобальных данных и набора подпрограмм, обрабатывающих эти данных.

Проектирование программ выполнялось «снизу-вверх». Сначала проектировали и реализовывали простые подпрограммы, а затем пытались построить сложную программу. В связи с отсутствием четких моделей описания подпрограмм и методов их проектирования создание любой программы превращалось в непростую задачу. Так как программисту приходилось отслеживать процесс обработки данных, местонахождение данных, то программирование ограничивалось его возможностями.

2. **Структурный подход к программированию** (60-70 гг XX века).

Структурный подход к программированию – это совокупность рекомендуемых технологических приемов, охватывающих выполнение всех этапов разработки программного обеспечения (ПО). В его основе лежит декомпозиция сложных систем с целью последующей реализации в виде отдельных (небольших – 40-50 операторов) подпрограмм.

Проектирование программ «сверху-вниз» подразумевало реализацию общей идеи, и обеспечение проработки интерфейсов подпрограмм. Одновременно вводились ограничения на

конструкции алгоритмов, рекомендовались формальные модели их описания, а также специальный метод проектирования алгоритмов – метод пошаговой детализации.

На данном этапе возникли процедурные языки программирования – они включали основные «структурные» операторы передачи управления, поддерживали вложение подпрограмм, локализацию и ограничение области «видимости» данных. Наиболее известные – PL/1, ALGOL-68, Pascal, C.

Рост сложности и размеров разрабатываемого программного обеспечения потребовал развития структурирования данных. Появилась возможность определения пользовательских типов данных. Одновременно научились разграничивать доступ к глобальным данным программы.

В дальнейшем появилась технология модульного программирования. Это предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные в отдельно компилируемые модули (библиотеки подпрограмм) – модуль графических ресурсов, подпрограмм вывода на принтер. Связи между модулями осуществляются через специальный интерфейс, при этом доступ к реализации модуля (телам подпрограмм и некоторым «внутренним» переменным) запрещен. Эту технологию поддерживают современные версии языков Pascal и C (C++), языки Ада и Modula.

Использование модульного программирования существенно упростило разработку ПО несколькими программистами (появилась возможность каждому разрабатывать свои модули независимо, взаимодействие модулей обеспечивается через специально оговоренные межмодульные интерфейсы).

Фактически структурный подход в сочетании с модульным программированием позволяет получать достаточно надежные программы, размер которых не превышает 100000 операторов.

При увеличении размера программы обычно возрастает сложность межмодульных интерфейсов и с некоторого момента предусмотреть взаимовлияние отдельных частей программы становится невозможно. Помимо этого, ошибка в интерфейсе при вызове подпрограммы выявляется только при работе программы.

Для разработки ПО большого объема было предложено использовать объектный подход.

3. **Объектный подход к программированию** (с середины 80-х до конца 90-х годов XX в.).

Объектно-ориентированное программирование определяется как технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности объектов, каждый из которого является экземпляром определенного типа (класса), а классы образуют иерархию с наследованием свойств. Взаимодействие программных объектов в такой системе осуществляется путем передачи сообщений.

Объектная структура программы впервые была использована в языке имитационного моделирования сложных систем Simula (60 годы XX века). Естественный для языков моделирования способ представления программы получил развитие в другом специализированном языке моделирования – Smalltalk (70 годы XX века), а затем был использован в новых версиях универсальных языков программирования – Pascal, C++, Modula, Java.

Основным достоинством ООП по сравнению с модульным является «более естественная» декомпозиция ПО, которая существенно облегчает его разработку. Это приводит к более полной локализации данных и интегрированию их с подпрограммами обработки, что позволяет вести практически независимую разработку отдельных частей (объектов) программы. Кроме того, объектный подход предлагает новые способы организации программ, основанные на механизмах наследования, полиморфизма, композиции, наполнения. Эти механизмы позволяют конструировать сложные объекты из сравнительно простых. Поэтому существенно увеличивается показатель повторного использования кодов и появляется возможность создания библиотек классов для различных применений.

Развитие технологий программирования, основанных на объектном подходе, дало толчок к созданию визуальных сред программирования – Delphi, C++ Builder, Visual C++ и т.д. при использовании визуальной среды у программиста появилась возможность проектировать некоторую часть (например, интерфейсы будущего продукта) с применением визуальных средств добавления и настройки специальных библиотечных компонентов. Результат – заготовка будущей программы, в которую уже внесены соответствующие коды.

Тем не менее, при программировании сохраняется зависимость модулей ПО от структур и форматов данных. От методов и адресов экспортируемых полей. Модули должны взаимодействовать между собой, обращаясь к ресурсам друг друга. Связи модулей нельзя разорвать, но можно стандартизировать. На этом основан компонентный подход к программированию.

4. **Компонентный подход и CASE-технологии** (с середины 90-х годов до настоящего времени).

Компонентный подход (КП) предлагает построение ПО из отдельных компонентов физически отдельно существующих частей ПО, которые взаимодействуют между собой через стандартизованные двоичные интерфейсы. В отличие от обычных объектов объекты-компоненты можно собрать в динамически вызываемые библиотеки или исполняемые файлы, распространять в двоичном виде (без исходных текстов) и использовать в любом языке программирования, поддерживающем соответствующую технологию. Это позволяет программистам создавать продукты, хотя бы частично состоящие из повторно использованных частей, то есть использовать технологию, хорошо зарекомендовавшую себя в области проектирования аппаратуры.

КП лежит в основе технологий, разработанных на базе COM (Component Object Model – компонентная модель объектов), и технологии создания распределенных приложений CORBA (Common Object Request Broker Architecture – общая архитектура с посредником обработки запросов объектов). Эти технологии используют сходные принципы и различаются лишь особенностями их реализации.

Технология COM определяет общую парадигму взаимодействия программ любых типов, т.е. позволяет одной части ПО использовать функции, предоставляемые другой, независимо от того, функционируют ли эти части в пределах одного процесса, в разных процессах на одном компьютере или на разных компьютерах.

Отличительной особенностью данного этапа также является создание и внедрение автоматизированных технологий разработки и сопровождения ПО – CASE-системы (Computer-Aided Software / System Engineering – разработка ПО с использованием компьютерной поддержки). Без средств автоматизации разработка достаточно сложного ПО сегодня трудноосуществима – память человека не в состоянии фиксировать все детали, которые необходимо учитывать при разработке сложных программных комплексов. CASE-технологии и CASE-средства возникли при структурном подходе, когда были разработаны и автоматизированы методы, позволяющие строить модели для различных составляющих ПО (информационной, функциональной, поведения) на различных этапах проектирования сложного программного обеспечения. CASE-системы позволяют выполнить автоматизировано разработку проекта ПО, а также, при необходимости, генерацию программного кода. Сегодня существуют CASE-системы, поддерживающие структурный и объектный подходы к программированию.

4) Проблематика автоматизации интеллектуального труда

Опыт автоматизации проектной и других видов интеллектуальной деятельности выявил следующие проблемы:

1) проблема интеграции как отдельных автоматизированных задач, так и подсистем в систему в целом. Как правило, данная проблема появляется при отсутствии согласования данных (наименований, типов) между подпрограммами при проектировании и разработке ПО;

2) проблема ошибок, сделанных при разработке ПО. Ошибки, сделанные на начальных этапах (формулирования требований, проектирования автоматизированной системы или программного обеспечения) наиболее дорого стоят при их устранении.

Затраты = e^t, где t - время обнаружения ошибки.

3) проблема объективизации знаний. При проектировании и программировании необходимо знать, как автоматизируемая задача выполнятся специалистом в реальном мире. Для этого информацию можно получить от самого специалиста, который не всегда знает, что именно он должен рассказать, и из документальных источников. Информация, которая распространена в документах (книгах, стандартах, методиках организация и предприятий) недостаточна, ведь многое специалист получает из опыта. Таким образом автоматизировать на основе документов можно небольшую часть знаний специалиста. Необходимо извлечь знания и памяти специалиста на основе методов анкетирования, интервьюирования, бесед и пр.

4) Главная проблема автоматизации проектной деятельности, интеллектуального труда - переход от естественно-языкового представления информации и знаний предметных задач, зафиксированного в традиционных документальных источниках и в памяти специалиста к формально-языковому представлению тех же знаний, но в качественно иной среде - среде вычислительных машин (в памяти ЭВМ).

5) Классификация программного обеспечения

1. Системное
2. Прикладное
3. Инструментальные технологии программирования

I. Системное ПО – совокупность программ и программных комплексов для обеспечения работы компьютера и вычислительных сетей.

Системное ПО направлено на:

- создание операционной среды функционирования других программ;
- обеспечение надежной и эффективной работы самого компьютера и вычислительной сети;
- проведения диагностики и профилактики аппаратуры компьютера и вычислительных сетей;
- выполнение вспомогательных технологических процессов (копирование, архивация, восстановление файлов программ и БД и т.д.).

Состав системного ПО:

1. **Базовое ПО** – минимальный набор программных средств, включающий операционную систему и систему программирования, поставляемый вместе с ЭВМ

В него входят: операционная система, операционные оболочки (текстовые, графические), сетевая операционная система.

Операционная система – это совокупность программных средств, обеспечивающих управление аппаратной частью компьютера и прикладными программами, а также их взаимодействием между пользователями. Она поддерживает типовые операции, такие как копирование информации, проверка работы конкретных блоков и т.д. Это программы, организующие работу устройств и не связанные со спецификой работы решаемой задачи.

Сетевая ОС – это комплекс программ, обеспечивающий обработку, передачу и хранение данных в сети. Сетевая ОС предоставляет пользователям различные виды сетевых служб (управление файлами, электронная почта, процессы управления сетью и т.д.), поддерживает работу в абонентских системах.

Операционные оболочки – специальные программы, предназначенные для облегчения общения пользователя с командами ОС. ОО имеют текстовый и графический варианты интерфейса конечного пользователя.

2. **Сервисное ПО** (утилиты) – набор сервисных, дополнительно устанавливаемых программ, являющихся расширением базового ПО: диагностика работоспособности компьютера, антивирусные программы, программы обслуживания дисков, программы архивирования данных, программы обслуживания сети.

II. Прикладное ПО служит программным инструментарием для решения прикладных задач, состоит из отдельных прикладных программ и пакетов прикладных программ. Прикладное ПО является самым многочисленным классом ПО. В данный класс входят программные продукты, выполняющие обработку информации различных предметных областей. То есть прикладное ПО – это комплекс взаимосвязанных программ для решения задач определенного класса предметной области.

Проблемно-ориентированные ППП – ППП одинакового функционального назначения (типизация функций управления, структуры данных, алгоритмов обработки): финансовая деятельность, автоматизированный бухгалтерский учет, управление производством, управление персоналом и т.д.

ППП автоматизированного проектирования предназначены для поддержки работы конструкторов и технологов, связанных с разработкой чертежей, схем, графическим моделированием и конструированием (высокие требования к аппаратному обеспечению, наличие библиотек встроенных функций, объектов, интерфейсов с графическими системами и БД).

Методо-ориентированные ППП - это программные продукты, обеспечивающие независимо от предметной области математические, статистические и другие методы решения задач (методы математического программирования, решения дифференциальных уравнений, имитационного моделирования, исследования операций).

Офисные ППП - это программные продукты, обеспечивающие управление деятельностью офиса (планировщики, программы-переводчики, средства проверки орфографии, распознавание текста, коммуникационные пакеты – предназначены для организации взаимодействия пользователей с удаленными абонентами или информационными ресурсами сети, средства электронной почты и пр.).

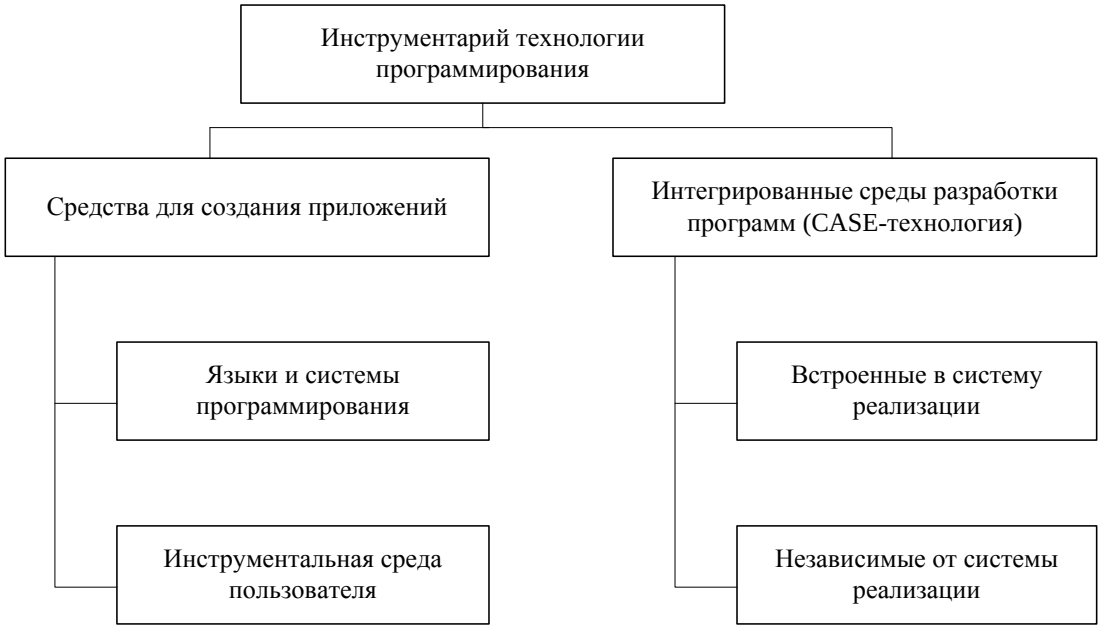
ППП общего назначения включают:

- СУБД (обеспечивают организацию и хранение локальных БД на автономно работающих компьютерах либо централизованное хранение БД на файл-сервере и сетевой доступ к ним);
- серверы БД – это ПО, предназначенное для создания и использования при работе в сети интегрированных БД в архитектуре «клиент-сервер»;
- генераторы отчетов (серверы отчетов) обеспечивают реализацию запросов и формирование отчетов в печатном или экранном виде в условиях сети с архитектурой «клиент-сервер»;
- текстовые процессоры (для работы с текстовыми документами);
- табличные процессоры – среда для вычислений конечным пользователем, содержат средства деловой графики, средства специализированной обработки;
- средства презентационной графики – специализированные программы, предназначенные для создания изображений и их показа на экране, подготовки слайд-фильмов, мультфильмов и их проектирования;
- интегрированные пакеты - набор нескольких программных продуктов, функционально дополняющих друг друга, поддерживающие единые информационные технологии, реализованные на единой операционной и вычислительной платформе. Компоненты интегрированных пакетов могут работать изолированно друг от друга, имеют общий интерфейс.

Программные средства мультимедиа – применяют для создания и использования аудио- и видеоинформации для расширения информационного пространства пользователя (БД компьютерных произведений искусства, библиотеки звуковых записей и т.д.).

Системы искусственного интеллекта включают программы-оболочки для создания экспертных систем путем наполнения баз знаний и правил логического вывода, готовые экспертные системы для принятия решений в рамках определенных предметных областей, системы анализа и распознавания речи, текста и т.д.

III. Инструментарий технологии программирования обеспечивает процесс разработки программ и включает специализированное ПО, которое является инструментальным средством разработки. ПО данного класса поддерживает все технологические этапы процесса проектирования, программирования, отладки и тестирования создаваемых программ. Пользователями данного ПО являются системные и прикладные программисты.



Средства для создания приложений – совокупность языков и систем программирования, инструментальные среды пользователя, а также различные программные компоненты для отладки и поддержки создаваемых программ.

Языки программирования – это формализованный язык для описания алгоритма решения задач на компьютере.

Системы программирования включают:

- компилятор (транслятор);
- интегрированная среда разработки программ;
- отладчик;
- средства оптимизации кода программ;
- набор библиотек;
- редактор связей;
- сервисные средства (утилиты) (для работы с библиотеками, текстовыми и двоичными файлами);
- справочные системы;
-систему поддержки и управления продуктами программного комплекса.

Инструментальная среда пользователя – это специальные средства, встроенные в пакеты прикладных программ:

- макрокоманды;
-клавишные макросы;
- языковые макросы;
- генераторы приложений;

- языки запросов высокого уровня;
- конструкторы экранных форм и объектов;
- библиотека функций, процедур, объектов и методов обработки;
- конструкторы меню и т.д.

Интегрированные среды разработки программ объединяют набор средств для их комплексного применения на технологических этапах создания программы.

Средства для создания информационных систем (ИС) и технологий поддерживают полный цикл проектирования сложной информационной системы от исследования объекта автоматизации до оформления проектной и прочей документации на систему. Они позволяют вести коллективную работу над проектом.

CASE-системы автоматизируют весь технологический процесс анализа, проектирования, разработки и сопровождения сложных программных систем.

Средства **CASE-систем** делятся на**:**

- встроенные в систему реализации – все решения по проектированию и реализации привязки в выбранной СУБД;

- независимые от системы реализации – все решения по проектированию ориентированы на унификацию начальных этапов ЖЦ программы и средств их документирования, обеспечивают большую гибкость в выборе средств реализации.

По способу распространения и использования программное обеспечение делится на:

- свободное
- открытое
- несвободное (закрытое).

Лекция 2.

Тема 2. Организация жизненного цикла программного продукта. Процессы жизненного цикла программного продукта по ГОСТ Р ИСО/МЭК 12207-2010.

План лекции.

1. Жизненный цикл программного продукта
2. Организация жизненного цикла программного продукта
3. Процессы жизненного цикла программного продукта по ГОСТ Р ИСО/МЭК 12207-2010

Основная часть.

Жизненный цикл программного продукта

В основе разработки и дальнейшего применения программного обеспечения пользователем лежит понятие ЖЦ, который является моделью его создания и использования, отражающей различные состояния, начиная с момента осознания необходимости появления данного ПО и заканчивая моментом его полного выхода из употребления у всех пользователей.

Длительность ЖЦ для различного ПО неодинакова и для большинства современных программных продуктов измеряется в годах (2-3 года). Хотя достаточно часто встречаются на компьютерах и давно снятое с производства программное обеспечение.

Стандарт **определяет структуру ЖЦ**, содержащую процессы, действия и задачи, которые должны быть выполнены во время создания и использования ПО. В России создание ПО первоначально, в 70-е гг., регламентировалось стандартами **серии ГОСТ 19.XXX** - Единая система программной документации (**ЕСПД**) и **ГОСТ 34.XXX** - Комплекс стандартов на

автоматизированную систему (АС). Однако создание, сопровождение и развитие современного прикладного ПО высокого качества в этих стандартах отражено недостаточно, а отдельные их положения устарели. Эти стандарты вынуждены использовать предприятия, выполняющие государственные заказы при создании ПО для внутреннего применения. Однако в экспортных заказах зарубежные клиенты требуют соответствия технологии проектирования, производства и качества продукции современным международным стандартам.

Основным зарубежным нормативным документом, наиболее полно и подробно регламентирующим ЖЦ ПО, является **международный стандарт ISO/IEC 12207**. (**ISO** - International Organization of Standardization - Международная организация по стандартизации, **IEC** - International Electrotechnical Commission - Международная комиссия по электротехнике.)

Под моделью ЖЦ понимается структура, определяющая **последовательность выполнения** и взаимосвязи процессов, действий и задач, выполняемых на протяжении ЖЦ ПО.

Модель ЖЦ зависит от специфики ПО и специфики условий, в которых оно создается и функционирует. **Стандарт** ISO/IEC 12207 **не предлагает конкретную модель ЖЦ и методы разработки ПО**. Его регламенты являются **общими для любых моделей** ЖЦ, методологий и технологий разработки. Стандарт ISO/IEC 12207 **описывает структуру процессов** ЖЦ ПО, но не конкретизирует в деталях, как реализовать или выполнить действия и задачи, включенные в эти процессы.

Определение понятий по стандарту ГОСТ Р ИСО/МЭК 12207-2010:

Жизненный цикл (life cycle): Развитие системы, продукта, услуги, проекта или других изготовленных человеком объектов, начиная со стадии разработки концепции и заканчивая прекращением применения.

Модель жизненного цикла (life cycle model): Структура процессов и действий, связанных с жизненным циклом, организуемых в стадии, которые также служат в качестве общей ссылки для установления связей и взаимопонимания сторон.

Модель ЖЦ любого конкретного ПО определяет характер процесса его создания. **Процесс создания ПО представляет собой совокупность упорядоченных во времени, взаимосвязанных и объединенных в этапы работ**, выполнение которых необходимо и достаточно для создания ПО, соответствующего заданным требованиям. **Под этапом создания ПО понимается часть процесса создания ПО**, ограниченная некоторыми временными рамками и заканчивающаяся выпуском конкретного продукта (моделей ПО, программных компонентов, документации), определяемого заданными для данной стадии требованиями. Этапы создания ПО выделяются по соображениям рационального планирования и организации работ, заканчивающихся заданными результатами. В состав ЖЦ ПО обычно включаются следующие этапы: 1. Формирование требований, предъявляемых к ПО. 2. Проектирование структуры ПО. 3. Реализация. 4. Тестирование. 5. Ввод в действие. 6. Эксплуатация и сопровождение. 7. Снятие с эксплуатации.

- *Анализ требований* состоит в сборе требований к продукту. Результатом анализа, как правило, является некоторый текст, а котором приводятся требования к программному продукту.
- *Проектирование* описывает внутреннюю структуру продукта. Обычно такое описание дается в форме диаграмм и текстов (архитектура, детальное проектирование).
- *Реализация (программирование)*. Результатом реализации является программный код всех уровней, будь то код, генерируемый высокоуровневой системой программирования, компилятором языка четвертого поколения или какой-либо другой. *Интеграция* — это процесс сборки всего продукта из отдельных частей.
- *Тестирование* – проверка правильности работы ПО.
- *Ввод в действие* – передача в эксплуатацию.
- *Эксплуатация и сопровождение* – использование ПО и его модификация или модернизация при необходимости.
- *Утилизация* – снятие с использования.

В действительности перечисленные фазы не следуют строго последовательно друг за другом, а частично перекрываются. На практике любую из фаз можно начинать до того, как будет полностью завершена предыдущая.

На каждом этапе могут выполняться несколько процессов, определенных в стандарте 1SO/IEC 12207, и, наоборот, один и тот же процесс может выполняться на различных этапах.

К настоящему времени сложилось несколько моделей ЖЦ, каждая из которых определяет различную методологию создания систем, тем не менее все без исключения модели ЖЦ включают в себя не менее четырех-пяти этапов и связей между ними с детальным описанием действий, моделей и результатов каждого этапа.

Стадии разработки программ и программной документации (ГОСТ 19.102).

ЖЦ в соответствии с ГОСТ 19.102-77 включает следующие стадии:

1. Техническое задание:

Этапы работ	Содержание работ
Обоснование необходимости разработки программы	- Постановка задачи. -Сбор исходных материалов. -Выбор и обоснование критериев эффективности и качества разрабатываемой программы. -Обоснование необходимости проведения научно-исследовательских работ.
Научно-исследовательские работы	-Определение структуры входных и выходных данных. - Предварительный выбор методов решения задач. - Обоснование целесообразности применения ранее разработанных программ. - Определение требований к техническим средствам. - Обоснование принципиальной возможности решения поставленной задачи.
Разработка и утверждение технического задания	- Определение требований к программе. - Разработка технико-экономического обоснования разработки программы. - Определение стадий, этапов и сроков разработки программы и документации на неё. - Выбор языков программирования. - Определение необходимости проведения научно-исследовательских работ на последующих стадиях. - Согласование и утверждение технического задания.

2. **Эскизный проект**

Этапы работ	Содержание работ
Разработка эскизного проекта	- Предварительная разработка структуры входных и выходных данных. - Уточнение методов решения задачи. - Разработка общего описания алгоритма решения задачи - Разработка технико-экономического обоснования.
Утверждение эскизного проекта	- Разработка пояснительной записки. - Согласование и утверждение эскизного проекта.

3. **Технический проект**

Этапы работ	Содержание работ
Разработка технического	- Уточнение структуры входных и выходных данных.

проекта	- Разработка алгоритма решения задачи. - Определение формы представления входных и выходных данных. - Определение семантики и синтаксиса языка. - Разработка структуры программы. - Окончательное определение конфигурации технических средств.
Утверждение технического проекта	- Разработка плана мероприятий по разработке и внедрению программ. - Разработка пояснительной записки. - Согласование и утверждение технического проекта.

4. **Рабочий проект**

Этапы работ	Содержание работ
Разработка программы	- Программирование и отладка программы.
Разработка программной документации	- Разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.
Испытания программы	- Разработка, согласование и утверждение порядка и методики испытаний. - Проведение предварительных государственных, межведомственных, приёмо-сдаточных и других видов испытаний. - Корректировка программы и программной документации по результатам испытаний.

5. **Внедрение**

Этапы работ	Содержание работ
Подготовка и передача программы.	- Подготовка и передача программы и программной документации для сопровождения и (или) изготовления. - Оформление и утверждение акта о передаче программы на сопровождение и (или) изготовление. - Передача программы в фонд алгоритмов и программ (ФАП).

ГОСТ Р ИСО/МЭК 12207: Процессы жизненного цикла программных средств

В ГОСТ ИСО_МЭК 12207 приведен перечень всех процессов, которые могут быть использованы при разработке программного обеспечения. Технологию организации процесса разработки ПО определяет модель жизненного цикла ПО, для которой из стандарта 12207 формируют набор необходимых процессов.

ГОСТ ИСО_МЭК 12207-99 «Процессы жизненного цикла программных средств» содержит в себе описание всех процессов жизненного цикла программного продукта от идеи создания до процесса сопровождения.

Взамен ГОСТ Р ИСО/МЭК 12207-99 принят ГОСТ Р ИСО/МЭК 12207-2010 «Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств».

Построение стандарта 12207-2010

Настоящий стандарт группирует различные виды деятельности, которые могут выполняться в течение жизненного цикла программных систем, в семь групп процессов.

Каждый из процессов жизненного цикла в пределах этих групп описывается в терминах цели и желаемых выходов, списков действий и задач, которые необходимо выполнять для достижения этих результатов.

a) процессы соглашения — два процесса;

b) процессы организационного обеспечения проекта — пять процессов;

c) процессы проекта — семь процессов;

d) технические процессы — одиннадцать процессов;

e) процессы реализации программных средств — семь процессов;

f) процессы поддержки программных средств — восемь процессов;

д) процессы повторного применения программных средств — три процесса.

Цели и результаты процессов жизненного цикла образуют эталонную модель процессов.

В настоящем стандарте принята следующая нумерация:

6.a и 7.a — указывают на группу процесса;

б.a.Ь и 7.a.Ь — указывают на процесс (или процесс более низкого уровня) в пределах этой группы;

6.a.Ь.1 и 7.a.Ь.1 — описывают цели процесса;

6.a.Ь.2 и 7.a.Ь.2 — описывают выходы процесса;

б.a.Ь.3.c и 7.a.Ь.3.c — перечисляют виды деятельности процесса и пункты;

6.a.b.3.c.d и 7.a.b.3.c.d — перечисляют задачи в пределах деятельности (c).

Группы процессов жизненного цикла представлены на рисунке 1.

Эталонная модель процесса не представляет конкретного подхода к осуществлению процесса, как и не предопределяет модель жизненного цикла системы (программного средства), методологию или технологию. Вместо этого эталонная модель предназначается для принятия организацией и базируется на деловых потребностях организации и области приложений. Определенный организацией процесс принимается в проектах организации в контексте требований заказчиков.

Результаты процесса используются для демонстрации успешного достижения цели процесса, что помогает оценщикам процесса определять возможности реализованного процесса организации и предоставлять исходные материалы для планирования улучшений организационных процессов.

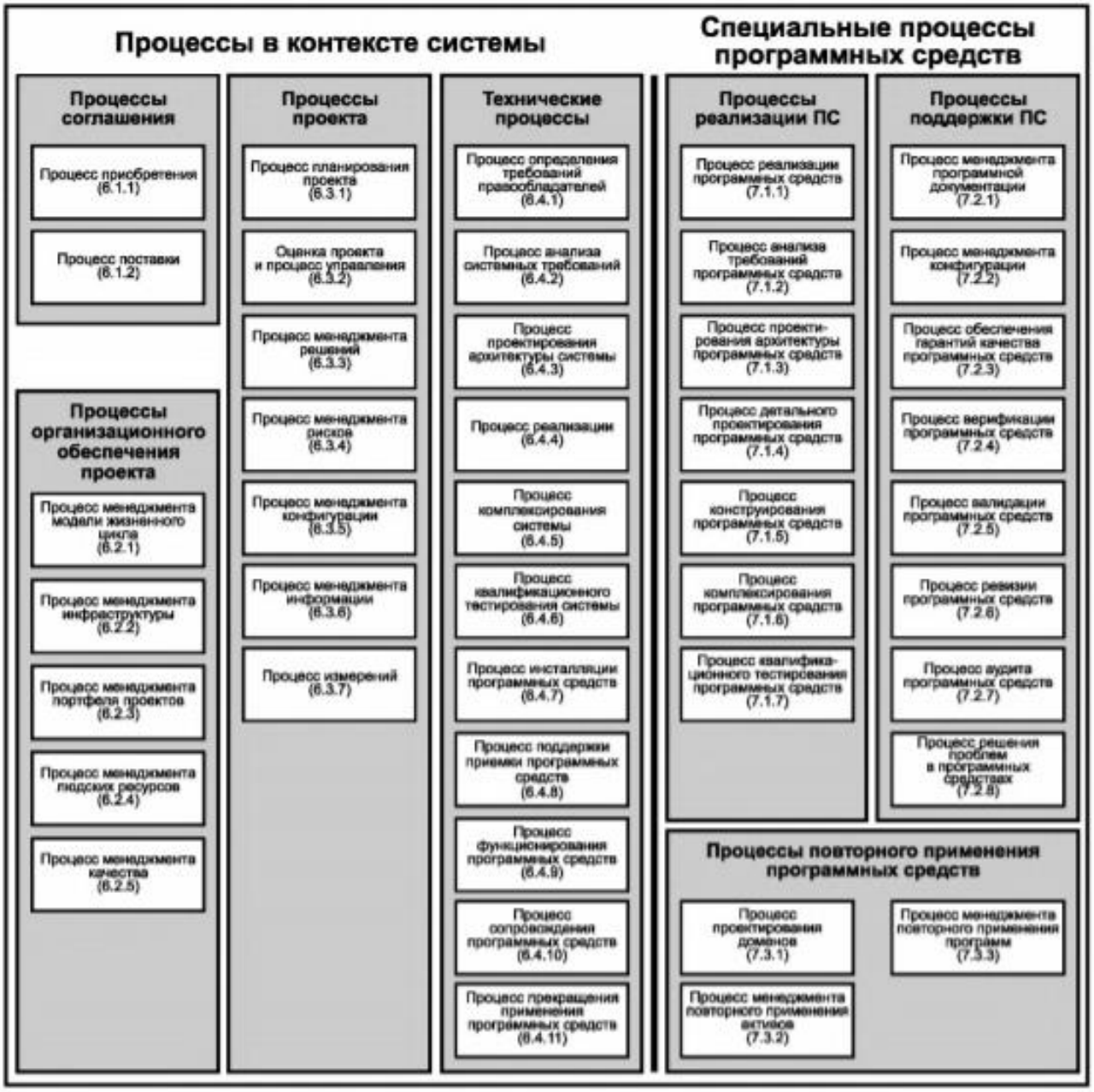


Рисунок 1. Группы процессов жизненного цикла

В настоящем стандарте существует два важных подразделения процесса. В разделе 6 представлен системный контекст для работы с автономным программным продуктом или услугой, или программной системой. Раздел 7 содержит специальные процессы программных средств для использования в реализации программного продукта или услуги, которые являются некоторым элементом более крупной системы.

Примечание: Для помощи в одновременном использовании ИСО/МЭК 15288:2007 «Программная инженерия. Процессы жизненного цикла систем» и настоящего стандарта соответствующие процессы раздела 6 имеют одинаковые обозначения подразделов. В общем случае совокупность процессов, представленная в настоящем стандарте, приспособлена к программным средствам или вкладам в результаты процессов, предусмотренных в ИСО/МЭК 15288:2007. Многие процессы из данного нормативного документа подобны реализациям процессов, специфических для программных средств, однако они сохраняют в ГОСТ Р

ИСО/МЭК12207—2010 важные различия, базирующиеся на целях, результатах и аудиториях. Пользователям как ИСО/МЭК 15288:2007, так и настоящего стандарта следует обязательно рассматривать пояснения и примечания в каждом таком специфическом процессе.

6. Процессы в контексте системы

6.1.Процессы соглашения

Процессы соглашения определяют действия, необходимые для выработки соглашений между двумя организациями. Если реализуется процесс приобретения (6.1.1), то он обеспечивает средства для проведения деловой деятельности с поставщиком продуктов, предоставляемых для применения в функционирующей системе, услугах поддержки этой системы или элементах системы, разработанных в рамках проекта. Если реализуется процесс поставки (6.1.2), то он обеспечивает средства для проведения проекта, в котором результатом является продукт или услуга, поставляемые приобретающей стороне. Таким образом, процессы соглашения, приведенные в настоящем стандарте, ориентированы на программные средства процессами соглашения из ИСО/МЭК 15288:2007.

6.2. Процессы организационного обеспечения проекта

Процессы организационного обеспечения проекта осуществляют менеджмент возможностей организаций приобретать и поставлять продукты или услуги через инициализацию, поддержку и управление проектами. Эти процессы обеспечивают ресурсы и инфраструктуру, необходимые для поддержки проектов, и гарантируют удовлетворение организационных целей и установленных соглашений. Они не претендуют на роль полной совокупности деловых процессов, реализующих менеджмент деловой деятельности организации.

Процессы организационного обеспечения проекта включают в себя:

a) процесс менеджмента модели жизненного цикла (6.2.1);

b) процесс менеджмента инфраструктуры (6.2.2);

c) процесс менеджмента портфеля проектов (6.2.3);

d) процесс менеджмента людских ресурсов (6.2.4);

e) процесс менеджмента качества (6.2.5).

В общем случае процессы организационного обеспечения проекта, предусмотренные настоящим стандартом, являются процессами, ориентированными на программные средства из соответствующей совокупности процессов в ИСО/МЭК 15288:2007.

6.3. Процессы проекта

В настоящем стандарте проект выбран как основа для описания процессов, относящихся к планированию, оценке и управлению. Принципы, связанные с этими процессами, могут применяться в любой области менеджмента организаций.

Существуют две категории процессов проекта. Процессы менеджмента проекта используются для планирования, выполнения, оценки и управления продвижением проекта. Процессы поддержки проекта обеспечивают выполнение специализированных целей менеджмента. Обе категории процессов проекта описаны ниже.

Процессы менеджмента проекта применяются для создания и развития планов проекта, оценки фактического выполнения и продвижения относительно плановых заданий и управления выполнением проекта вплоть до полного его завершения. Отдельные процессы менеджмента проекта могут привлекаться в любое время жизненного цикла и на любом уровне иерархии проекта в соответствии с планами проекта или возникновением непредвиденных событий. Процессы менеджмента проекта применяются на уровне строгости и формализации, зависящих от риска и сложности проекта:

a) процесс планирования проекта (6.3.1);

b) процесс управления и оценки проекта (6.3.2).

Процессы поддержки проекта формируют специфическую совокупность задач, ориентированных на выполнение специальных целей менеджмента. Все эти процессы очевидны при осуществлении менеджмента любой инициируемой деятельности, располагаясь по нисходящей от организации в целом вплоть до отдельного процесса жизненного цикла и его задач:

a) процесс менеджмента решений (6.3.3);

b) процесс менеджмента рисков (6.3.4);

c) процесс менеджмента конфигурации (6.3.5);

d) процесс менеджмента информации (6.3.6);

e) процесс измерений (6.3.7).

В общем случае процессы поддержки проекта, представленные в настоящем стандарте, идентичны процессам поддержки проекта, приведенным в ИСО/МЭК 15288:2007, за исключением некоторых отличий в форме их представления. В некоторых случаях процессы поддержки программных средств могут иметь взаимосвязи с процессами поддержки проектов.

6.4. Технические процессы

Технические процессы используются для определения требований к системе, преобразования требований в полезный продукт, для разрешения постоянного копирования продукта (где это необходимо), применения продукта, обеспечения требуемых услуг, поддержания обеспечения этих услуг и изъятия продукта из обращения, если он не используется при оказании услуги.

Технические процессы определяют деятельность, которая дает возможность реализовывать организационные и проектные функции для оптимизации пользы и снижения рисков, являющихся следствием технических решений и действий. Эта деятельность обеспечивает возможность продуктам и услугам обладать такими свойствами, как своевременность и доступность, результативность затрат, а также функциональность, безотказность, ремонтопригодность, продуктивность, приспособленность к применению, и другими качественными характеристиками, требуемыми приобретающими и поддерживающими организациями. Она также предоставляет возможность продуктам и услугам соответствовать ожиданиям или требованиям гражданского законодательства, включая факторы здоровья, безопасности, защищенности и факторы, относящиеся к окружающей среде.

Технические процессы состоят из следующих процессов:

a) определение требований правообладателей (специальный случай процесса определения требований правообладателей, приведенного в ИСО/МЭК 15288:2007) (6.4.1);

b) анализ системных требований (специальный случай процесса анализа требований) (6.4.2);

c) проектирование архитектуры системы (специальный случай процесса проектирования архитектуры, приведенного в ИСО/МЭК 15288:2007) (6.4.3);

d) процесс реализации (специальный случай процесса реализации элементов системы, приведенного в ИСО/МЭК 15288:2007 и далее разработанного в разделе 7 настоящего стандарта как процесса реализации программных средств) (6.4.4);

e) процесс комплексирования системы (специальный случай процесса комплексирования, приведенного в ИСО/МЭК 15288:2007) (6.4.5);

f) процесс квалификационного тестирования системы (процесс, который способствует достижению результатов процесса верификации, приведенного в ИСО/МЭК 15288:2007) (6.4.6);

д) процесс инсталляции программных средств (процесс, который способствует достижению результатов процесса передачи, приведенного в ИСО/МЭК 15288:2007) (6.4.7);

h) процесс поддержки приемки программных средств (процесс, который способствует достижению результатов процесса передачи, приведенного в ИСО/МЭК 15288:2007) (6.4.8);

i) процесс функционирования программных средств (специальный случай процесса функционирования, приведенного в ИСО/МЭК 15288:2007) (6.4.9);

j) процесс сопровождения программных средств (специальный случай процесса сопровождения, приведенного в ИСО/МЭК 15288:2007) (6.4.10);

k) процесс изъятия из обращения программных средств (специальный случай процесса изъятия и списания, приведенного в ИСО/МЭК 15288:2007) (6.4.11).

В общем случае технические процессы, представленные в настоящем стандарте, ориентированы на программные средства специальными случаями или вкладами в результаты технических процессов, представленных в ИСО/МЭК 15288:2007. Большинство из них схожи с процессами реализации программных средств, но сохраняют важные различия, например, анализ системных требований и анализ требований к программным средствам начинаются с разных исходных позиций и имеют разные предназначения.

7. Специальные процессы программных средств

7.1. Процессы реализации программных средств

Процессы реализации программных средств используются для создания конкретного элемента системы (составной части), выполненного в виде программного средства. Эти процессы преобразуют заданные характеристики поведения, интерфейсы и ограничения на реализацию в действия, результатом которых становится системный элемент, удовлетворяющий требованиям, вытекающим из системных требований. Специальным процессом является процесс реализации программных средств, выражающий специфически программную особенность процесса реализации, приведенного в ИСО/МЭК 15288:2007.

Процессы реализации программных средств включают в себя несколько специальных процессов более низкого уровня:

a) процесс реализации программных средств (7.1.1);

b) процесс анализа требований к программным средствам (7.1.2);

c) процесс проектирования архитектуры программных средств (7.1.3);

d) процесс детального проектирования программных средств (7.1.4);

e) процесс конструирования программных средств (7.1.5);

f) процесс комплексирования программных средств (7.1.6);

д) процесс квалификационного тестирования программных средств (7.1.7).

7.2. Процессы поддержки программных средств

Процессы поддержки программных средств предусматривают специально сфокусированную совокупность действий, направленных на выполнение специализированного программного процесса. Любой поддерживающий процесс помогает процессу реализации программных средств как единое целое с обособленной целью, внося вклад в успех и качество программного проекта. Существует восемь таких процессов:

a) процесс менеджмента документации программных средств (7.2.1);

b) процесс менеджмента конфигурации программных средств (7.2.2);

c) процесс обеспечения гарантии качества программных средств (7.2.3);

d) процесс верификации программных средств (7.2.4);

e) процесс валидации программных средств (7.2.5);

f) процесс ревизии программных средств (7.2.6);

д) процесс аудита программных средств (7.2.7);

h) процесс решения проблем в программных средствах (7.2.8).

7.3. Процессы повторного применения программных средств

Группа процессов повторного применения программных средств состоит из трех процессов, которые поддерживают возможности организации использовать повторно составные части программных средств за границами проекта. Эти процессы уникальны, поскольку, в соответствии с их природой, они используются вне границ какого-либо конкретного проекта. Процессами повторного применения программных средств являются:

a) процесс проектирования доменов (7.3.1);

b) процесс менеджмента повторного применения активов (7.3.2);
c) процесс менеджмента повторного применения программ (7.3.3).

Примечание: При разработке ГОСТ Р ИСО/МЭК12207—2010 учитывались следующие нормативные документы: ГОСТ Р ИСО/МЭК12207—1995; ГОСТ Р ИСО/МЭК12207—2008; ИСО/МЭК 15288:2002; ИСО/МЭК 15288:2007.
Также были разработаны руководящие указания для обоих международных стандартов (12207 и 15288) - технический отчет ИСО/МЭК 24748.

Лекция 3.

Тема 3. Модели жизненного цикла программного продукта.
Каскадная, итерационная, спиральная модели жизненного цикла программного продукта.

План лекции.

1. Каскадная модель ЖЦ
2. Итерационная модель ЖЦ
3. Спиральная модель ЖЦ

Основная часть.

Модели жизненного цикла программного продукта.

Существующие **модели ЖЦ определяют порядок исполнения этапов** в ходе разработки, а также **критерии перехода** от этапа к этапу.

Исторически утвердились три модели ЖЦ:

1. Каскадная
2. Итерационная
3. Спиральная

1.Каскадная модель (или модель водопада) (70-80 гг.) - модель жизненного цикла программного изделия, основным свойством которой является завершение каждой фазы цикла (планирования, проектирования, кодирования, внедрения и т.д.) верификацией и подтверждением и циклическое повторение реализованных фаз. Предполагает переход на следующий этап после полного завершения работ предыдущего этапа. Каскадная модель ЖЦ ПО представлена на рис. 1.1.

Рис. 1.1. Каскадная модель ЖЦ ПО

Такой метод, называемый каскадным, имеет следующие преимущества:

- разбиение всей разработки на этапы;
- четкое разделение данных и процессов их обработки;
- переход с одного этапа на следующий происходит **только после полного завершения работы** на текущем этапе;
- возможность **планировать сроки завершения всех работ и** соответствующие **затраты**;
- результатами каждого этапа являются технические решения и **полный комплект проектной документации**, отвечающей критериям полноты и согласованности, достаточной для того, чтобы разработка могла быть продолжена **другой командой** разработчиков;
- исходными для каждого этапа являются документы и решения, полученные на предыдущем этапе,
- использование процедурных языков программирования.

Прим.: Каскадная модель хорошо зарекомендовала себя при разработке несложного ПО, когда каждая программа представляет собой единое целое. При построении такого ПО в самом начале разработки можно достаточно точно и полно сформулировать все требования, с тем, чтобы предоставить разработчикам свободу реализовать их как можно лучше с технической точки зрения.

Недостатки каскадного метода:

- Главный из них – последовательное выполнение этапов. Например, программирование можно начать только по завершении анализа и проектирования.
- Это приводит к большим потерям времени, не позволяет быстро разрабатывать прототипы программной системы.

Каскадный принцип не согласуется с итеративным характером разработки программной системы, поскольку на последних этапах может выясниться необходимость внесения изменений в решения, принятые на предыдущих этапах.

- Требования к ПО "заморожены" в виде технического задания на все время его создания. Пользователи могут внести свои замечания только после того, как работа над ПО будет полностью завершена. В случае неточного изложения требований или их изменения в течение

длительного периода создания ПО, пользователи получают систему, не удовлетворяющую их потребностям.

- Модели (как функциональные, так и информационные) автоматизируемого объекта могут устареть одновременно с их утверждением.
- Выявление и устранение ошибок производится только на стадии тестирования, которая может растянуться во времени или вообще никогда не завершиться.

2. Итерационная модель

Следующая стадия развития теории проектирования стала **Поэтапная модель с промежуточным контролем** (80-85 гг.) или **итерационная модель** разработки ПО с циклами обратной связи между этапами. Итерационная модель ЖЦ ПО представлена на рис. 1.2.

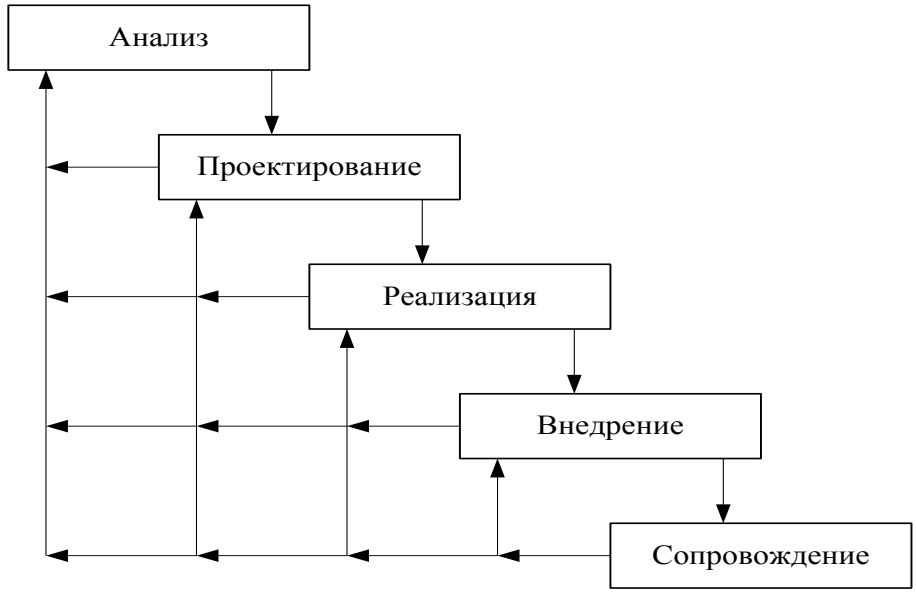


Рис. 1.2. Итерационная модель ЖЦ ПО

Преимущество данной модели заключается в том, что межэтапные корректировки обеспечивают меньшую трудоемкость по сравнению с каскадной моделью, большие гибкость и надежность, однако, время жизни каждого из этапов растягивается на весь период разработки. Это достигается за счет появившейся возможности проводить проверки и корректировки проектируемого программного продукта на каждой стадии разработки.

Прим.: Итерационность модели проявляется в обработке ошибок, выявленных промежуточным контролем. Если на каком-либо этапе в ходе промежуточной проверки обнаружена ошибка, допущенная на более ранней стадии, то необходимо повторить весь цикл работ этой стадии. При этом анализируются причины ошибки и корректируются в случае необходимости исходные данные этапа или его содержание (последовательность действий).

Недостатки: к сожалению, в процесссе разработки системы могут измениться начальные требования, и в этом случае итерационная модель может оказаться неэффективной.

3. Спиральная модель

Для устранения этого недостатка Боэм предложил **спиральную модель** процесса разработки ПО (86-90 гг.). Она заключается в том, что разработка проекта выглядит как последовательное создание развиваемых релизов (прототипов системы). При этом для каждого релиза выполняются последовательно перечисленные выше этапы, на которых уточняется проект (см. рис. 1.3). Таким образом, каждый релиз может рассматриваться как очередной виток спирали, расширяющий охватываемую область проекта за счет добавления новых функций.

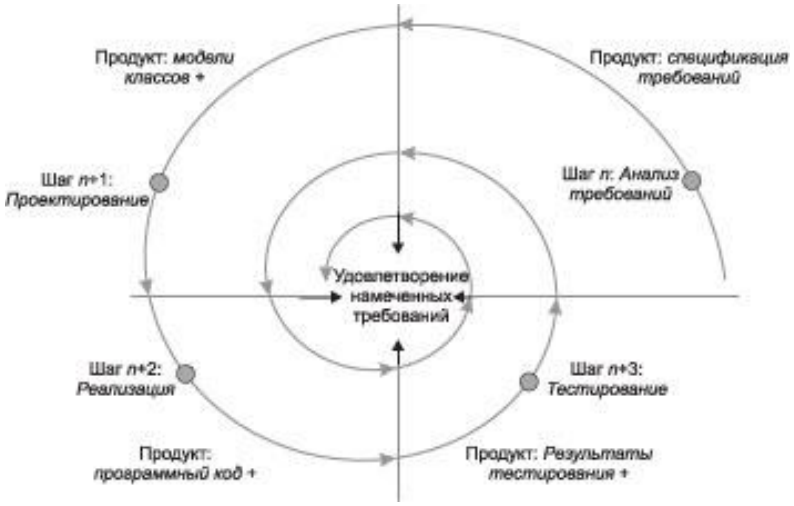


Рис. 1.3. Спиральная модель ЖЦ ПО

Данная модель ЖЦ ПО делает упор на начальные этапы ЖЦ (анализ требований, проектирование спецификаций, предварительное и детальное проектирование). На этих этапах обосновывается и проверяется реализуемость технических решений путем создания прототипов. Каждый виток спирали соответствует поэтапной модели создания фрагмента или версии программного изделия, на нем уточняются цели и характеристики проекта, определяется его качество, планируются работы следующего витка спирали. Таким образом, углубляются и последовательно конкретизируются детали проекта и в результате выбирается обоснованный вариант, который доводится до реализации.

На этих этапах реализуемость технических решений проверяется **путем создания прототипов приложений**, которые демонстрируются заказчику, обсуждаются. Под прототипом обычно понимают набор программ, моделирующих (изображающих, эмулирующих) работу готовой системы.

Цель прототипирования - более ясно представить себе будущую систему, предугадать ее недостатки на этапе проектирования, внести необходимые коррективы в техническое задание и технический проект, если он уже готов. Прототип системы удобно демонстрировать сотрудникам предприятия-заказчика, чтобы они могли понять, насколько удобно им будет пользоваться системой, какие функции следует добавить или исключить.

Главная задача, которая решается в данной модели ЖЦ ПО - **к**ак можно быстрее показать пользователям работоспособный продукт, тем самым активизируя процесс уточнения и дополнения требований, исправления ошибок, обусловленных неопределенностью или некорректностью технических заданий и спецификаций требований. Спиральная модель не исключает использования каскадного подхода на завершающих стадиях проекта в тех случаях, когда требования к системе оказываются полностью определенными.

Прим.: В случае спирального ЖЦ разработка программного обеспечения: последовательность анализ требований — проектирование — реализация — тестирование выполняется более одного раза. Для этого может быть несколько причин. Основная причина обычно связана с необходимостью предупреждения рисков. Другой причиной может быть необходимость предоставить заказчику частичную версию проекта для получения отзывов и пожеланий. Если разрабатываемая программа достаточно сложна, необходимо выполнять промежуточные интеграции, не откладывая эту фазу на самый конец, как это предписывает каскадная модель. Общая же идея спирального процесса заключается в том, чтобы на каждой итерации строить очередную версию программы, используя в качестве основы ее предыдущую версию. В этом случае процесс приобретает спиралевидный характер.

Дополнительное преимущество итеративных процессов заключается в возможности собирать на каждой итерации метрические характеристики процесса. Например, располагая

данными о времени, которое потребовалось для выполнения первой итерации, можно уточнить план-график дальнейшей работы. Такая возможность особенно полезна для организаций, имеющих небольшой опыт планирования разработок.

Недостатки:

Основная проблема спирального цикла - определение момента перехода на следующий этап. Для ее решения необходимо ввести временные ограничения на каждый из этапов ЖЦ. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа закончена. План составляется на основе статистических данных, полученных в предыдущих проектах, и личного опыта разработчиков.

Другими недостатками спиральной модели являются:

- трудоемкость внесения изменений;
- большой объем документации по проекту, затрудняющий программирование;
- серьезные ограничения возможностей сборки системы из готовых компонент;
- сложность переноса на другие платформы.

Прим.: Хотя спиральная модель отражает типичную схему процесса разработки, она требует более искусного управления, нежели простая каскадная модель. Одна из трудностей заключается в поддержке целостности документации, которая должна быть полностью обновлена и дополнена к концу каждой итерации. В частности, каждая версия программного кода должна реализовывать документированный проект и удовлетворять документированным же требованиям. Управление документацией еще более усложняется, когда с целью повышения производительности команды очередная итерация процесса начинается до завершения предыдущей итерации.

И все же для большинства программных проектов преимущества спирального процесса перевешивают его недостатки. Количество итераций, требуемых в случае применения спиральной модели, зависит от ситуации. Типичный проект, трудоемкость которого оценивается в три человеко-месяца, а продолжительность — в четыре месяца, вероятнее всего, потребует две-три итерации. Затраты на проведение большего числа шагов могут просто перевесить выгоду от дополнительных итераций.

Случай, когда число итераций возрастает настолько, что каждая новая итерация предоставляет слишком малое количество новых возможностей по сравнению с предыдущей, называется инкрементальной разработкой.

Дополнительно:

Унифицированный процесс разработки программного обеспечения (*Unified Software Developement Process* - **USDP)**

Первые модели впервые были предложены в книге Якобсона, Буча и Рамбо, изданной в 1999 году. Этот процесс является детищем более ранних методологий, разработанных этими тремя авторами: «Объектная методология» Якобсона, «Методология Буча» и «Техника объектного моделирования» Рамбо.

Поскольку итеративные подходы частично или полностью повторяют каскадный процесс, их иногда трудно описать. USDP — это итеративный процесс разработки, пытающийся разрешить эту проблему путем классификации итераций и отнесения их к одной из четырех групп.

- Начальные итерации.
- Итерации проектирования.
- Итерации конструирования.
- Итерации перехода.

Заинтересованные лица включают в себя всех, кто так или иначе заинтересован в реализации проекта. Это, конечно же, заказчики и пользователи (которые сами могут и не являться заказчиками), а также инвесторы, пользовательские группы и сами разработчики.

Итерации проектирования задают ключевую техническую цель в выборе и утверждении (принятии) архитектуры. Итерации конструирования представляют базовый продукт, но еще требуется проделать работу, чтобы подготовить продукт к выпуску. Целью итераций перехода является подготовка приложения к выпуску (к отправке заказчику).

С учетом приведенной выше классификации итераций, USDP может быть представлен матрицей (рис. 1 и рис. 2). Как и большинство процессов объектно-ориентированного анализа и проектирования, фазы каскадного процесса, представленные Якобсоном, также включают в себя фазу анализа. На рис. 1 показано, где эта фаза совпадает с классическими фазами каскадного процесса.

Анализ состоит из той части процесса анализа требований, в которой выбираются и соотносятся между собой базовые классы приложения. Кроме того, в USDP отсутствует своя фаза интеграции, обычно представленная в классическом каскадном процессе. Это произошло в связи с убежденностью Буча в том, что объектно-ориентированные приложения могут и должны использовать непрерывную интеграцию. Другими словами, сразу после добавления новых частей исходное приложение интегрируется.

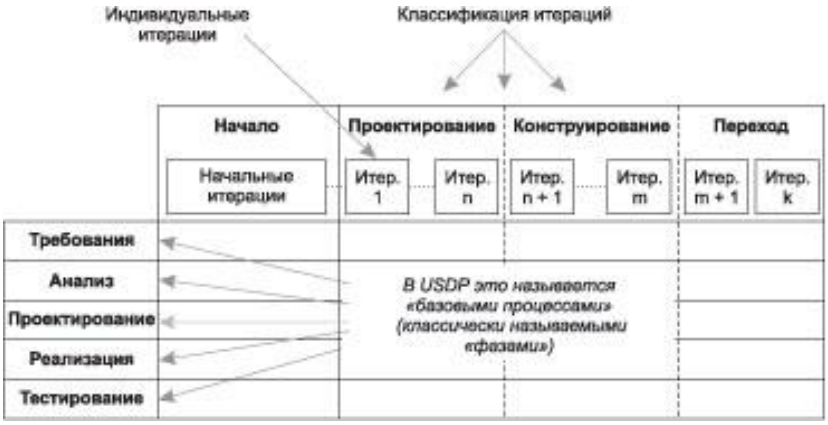


Рис. 1. Различия между USDP и стандартной терминологией (1)

Рис. 2. Различия между USDP и стандартной терминологией (2)

Приблизительный тип и уровень работ для каждого базового рабочего итерационного процесса USDP показан на рис. 3. Большинство итераций в разной степени затрагивают практически все фазы каскадного процесса (базовые рабочие процессы). Начальные итерации содержат в основном анализ требований, непосредственно анализ, а также могут включать проектирование и реализацию для создания предварительного прототипа, который может быть использован для обсуждения проекта с заинтересованными сторонами. Итерации проектирования затрагивают в основном анализ требований, а также некоторую часть проектирования и реализации. Итерации конструирования включают в себя проектирование и реализацию, а итерации перехода — реализацию и тестирование.

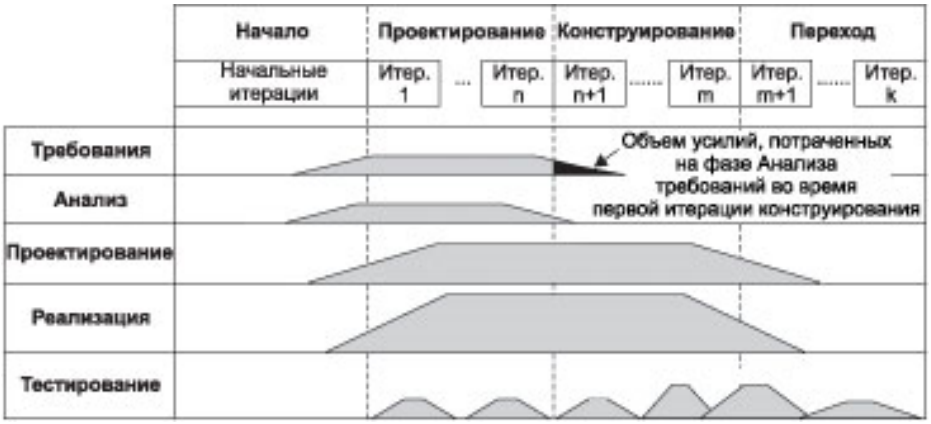


Рис. 3. Матрица USDP

Процесс разбит на циклы, каждый из который состоит из четырех фаз: начало - начальная стадия; проектирование – разработка; конструирование – конструирование; переход – ввод в эксплуатацию. Результатом работы каждого такого цикла является своя версия программного продукта. Каждая стадия завершается в четко определенной конкретной точке. В этот момент времени должны достигаться важные результаты и приниматься критически важные решения о дальнейшей разработке.

На начальной стадии выполняется начальный анализ оценки проекта.

На стадии разработки выявляются детальные требования к системе, выполняется высокоуровневый анализ предметной области и проектирование базовой архитектуры будущей системы.

Результатом стадии конструирования является продукт, готовый к передаче пользователям, содержащий руководство пользователей и готовый к интеграции на требуемых платформах.

Ввод в эксплуатацию – это передача готового продукта в полное распоряжение конечных пользователей.

Достоинства подхода:

- *распараллеливание работ – сокращение времени выполнения проекта,*
- *простота внесения изменений – локальный, а не глобальный характер,*
- *гибкая архитектура и переносимость,*
- *повторное использование программных компонент,*
- *естественность описания.*

Недостатки (в области программирования):

-динамическое связывание (поиск операций в классе) – обращение к методу класса занимает больше времени, чем к обычной подпрограмме,

- *излишняя многочисленность операций и их вызовов.*

Лекция 4.

Тема 4. Сертификация и оценка процессов жизненного цикла программного продукта. Модель оценки зрелости процессов CMM.

План лекции.

1. Качество программного обеспечения
2. Сертификация и системы обеспечения качества
3. Модель оценки зрелости процессов CMM

Переход от штучной разработки программ к промышленному способу разработки сложного программного обеспечения заключается в разделении жизненного цикла на этапы и выполнении процессов на данных этапах группами специалистов, при этом каждая группа обладает знаниями и навыками уже в определенной профессиональной области. Это определило повышение требований к качеству разрабатываемого программного обеспечения. В настоящее время существует несколько процедур и стандартов, которые связаны с оценкой качества процессов, выполняемых разработчиками при работе над созданием ПО:

- международные стандарты серии ISO 9000 (9000-9004);
- CMM (Capability Maturity Model) –модель зрелости. Предложена SEO (институт программирования при университете Карнеги-Меллон);
- процесс сертификации программ.

1) Качество программного обеспечения

Понятия качества программного обеспечения

Основой для формирования необходимых показателей программного обеспечения является анализ свойств характеризующих качество его функционирования с учетом технологических возможностей разработчика.

Под качеством функционирования понимается множество свойств, обусловливающих пригодность ПО обеспечивать надежное и своевременное представление требуемой информации потребителю для ее дальнейшего использования по назначению.

Набор показателей качества программ зависит от функционального назначения и свойств каждого программного средства.

Как правило, качество изменяется в течение жизненного цикла программного обеспечения - требуемое значение вначале почти всегда отличается от фактически достигнутого при завершении проекта и качества поставленного продукта. Для различных стадий ЖЦ определены следующие представления качества программного продукта, например:

- целевое качество – необходимое и достаточное качество, отражающее реальные потребности заказчика или пользователя (как правило, вначале не могут быть полностью установлены характеристики, так как пользователь не знает их целиком)
- требуемое качество продукта - значения характеристик, фактически установленных в требованиях (спецификациях) к качеству. Это цель для начального утверждения в спецификациях (оптимальные и минимально допустимые значения)
- качество поставленного продукта - набор характеристик качества поставленного и готового к применению продукта, обычно прошедшего испытания в моделированной среде с имитированными или реальными данными, поскольку некоторые недостатки могут еще оставаться после тестирования.
- качество в использовании - качество системы, содержащей программное средств, которое воспринимается пользователями. Оно измеряется скорее в терминах результата использования программ, чем внутренних свойств самого программного обеспечения.

Хорошее ПО должно обладать целым рядом свойств, позволяющим успешно его использовать в течение длительного периода, т.е. обладать определенным качеством. Качество (quality) ПО − это совокупность его черт и характеристик, которые влияют на его способность удовлетворять заданные потребности пользователей. Это не означает, что разное ПО должно обладать одной и той же совокупностью таких свойств в их наивысшей степени. Этому препятствует тот факт, что повышение качества ПО по одному из таких свойств часто может быть достигнуто лишь ценой изменения стоимости, сроков завершения разработки и снижения качества этого ПО по другим его свойствам. Качество ПО является удовлетворительным, когда оно обладает указанными свойствами в такой степени, чтобы гарантировать успешное его использование. Совокупность свойств ПО, которая образует удовлетворительное для пользователя качество ПО, зависит от условий и характера эксплуатации этого ПО. Поэтому

при описании качества ПО, прежде всего, должны быть фиксированы критерии отбора требуемых свойств ПО. В настоящее время критериями качества ПО принято считать:

− функциональность;
− надежность;
− легкость применения;
− эффективность;
− сопровождаемость;
− мобильность.

Функциональность − это способность ПО выполнять набор функций, удовлетворяющих заданным или подразумеваемым потребностям пользователей. Набор указанных функций определяется во внешнем описании ПО.

Надежность (reliability) ПО − это его способность безотказно выполнять определенные функции при заданных условиях в течение заданного периода времени. Надежность является наиболее важной характеристикой качества ПО. Это свойство объекта выполнять заданные функции, сохраняя во времени значения установленных эксплуатационных показателей в заданных пределах, соответствующих заданным режимам и условиям использования, технического обслуживания, ремонта, хранения и транспортирования.

Относительно невысокая физическая надежность компонент, их способность к разрушению, старению или снижению надежности в процессе эксплуатации привели к тому, что этот фактор оказался доминирующим для большинства комплексов аппаратуры. Этому способствовала также невысокая сложность многих технических систем, вследствие чего дефекты проектирования проявлялись относительно редко и вуалировались физическими отказами компонент.

Надежность сложных ПО определяется этими же факторами, однако доминирующими являются дефекты и ошибки проектирования, так как физическое хранение программ на современных носителях характеризуется очень высокой надежностью. Программа любой сложности и назначения при строго фиксированных исходных данных и абсолютно надежной аппаратуре исполняется по однозначно определенному маршруту и дает на выходе строго определенный результат. Однако случайное изменение исходных данных и накопленной при обработке информации, а также множество условных переходов в программе создают огромное число различных маршрутов исполнения каждого сложного ПО. Источниками ненадежности являются непроверенные сочетания исходных данных, при которых функционирующее ПО дает неверные результаты или отказы. В результате комплекс программ не соответствует требованиям функциональной пригодности и работоспособности.

2) Сертификация и системы обеспечения качества

Потребителей – заказчиков интересует, прежде всего, качество готового конечного продукта и обычно не очень беспокоит, как оно достигнуто. Однако это качество должно быть ответственно удостоверено и гарантировано компетентными, желательно независимыми организациями путем достаточно широких, регламентированных испытаний. Гарантирование качества продукции возможно посредством сертификационных испытаний процессов производства комплексов программ и/или испытаний их результатов – программных продуктов. Сертификация – процедура подтверждения соответствия требованиям, посредством которой независимое от изготовителя и потребителя предприятие юридически удостоверяет в письменной форме, что состояние продукции и/или производства и системы менеджмента качества способно обеспечить стабильность характеристик изготовляемой продукции и соответствует установленным заказчиком требованиям и стандартам. Качество при проектировании и производстве программных продуктов можно обеспечивать двумя методами:
1) посредством применения регламентированных технологий и систем обеспечения качества процессов проектирования и производства, предотвращающих дефекты и гарантирующих высокое качество продуктов в процессе их создания;

2) путем использования заключительного контроля и испытаний готовых продуктов и исключения из поставки или направлением на доработку изделий, не соответствующих требуемым показателям качества.

Первый метод обеспечивает высокое качество выполнения всего процесса проектирования и производства, обеспечивая тем самым минимум экономических потерь от брака, что рентабельно при создании сложных систем. Многие руководители осознали, что для создания современных прикладных высококачественных информационных систем необходимы не менее качественные технологии их производства.

При этом качество тех и других должно удостоверяться регламентированными поэтапными и заключительными испытаниями. Результаты испытаний процессов производства трудно измерять количественными критериями, их обычно характеризуют рядом требований к качественному выполнению стандартизированных производственных процессов. Они оцениваются набором свойств, непосредственно отражающихся на характеристиках качества конечного программного продукта, однако при этом нет гарантии адекватного и однозначного на них влияния.

Второй метод акцентирован на анализ и контроль качества готового программного продукта, которое удостоверяется при сертификационных испытаниях. Достижение при производстве необходимого качества продукции за счет только выходного контроля, при отсутствии или недостатках системы обеспечения качества в технологическом процессе разработки, может приводить к длительному итерационному процессу массовых доработок и повторных испытаний. При сертификационных испытаниях качества готового программного продукта могут использоваться стандартизированные количественные и качественные критерии и характеристики, которые непосредственно отражают функции и свойства продукции, интересующие заказчика и потребителей, их можно измерить и достоверно установить.

Таким образом, задача обеспечения и удостоверения высокого качества сложных программных продуктов сводится к испытаниям технологий процессов проектирования и производства программных средств, поддержанных системой качества, и конечного продукта, созданного на базе таких технологий. Соответственно можно выделить два вида сертификационных испытаний: технологий обеспечения жизненного цикла программных средств, поддержанных регламентированными системами качества и/или готового программного продукта с полным комплектом эксплуатационной документации.

Наиболее полно в России стандартизирована сертификация производства продукции различных видов. Она поддержана стандартами: Временный порядок сертификации производств с учетом требований ГОСТ Р ИСО 9001; ГОСТ Р 40.003 и ГОСТ Р ИСО 19011.

При этом сертификация производства определена как процедура подтверждения соответствия, посредством которой независимая от изготовителя (продавца, исполнителя) и потребителя (покупателя) организация удостоверяет в письменной форме, что состояние производства (системы менеджмента качества производства) способно обеспечить стабильность характеристик изготовляемой продукции и соответствует требованиям ГОСТ Р ИСО 9001. К работе по сертификации привлекаются эксперты (аудиторы) по сертификации производств, зарегистрированные в Регистре системы сертификации персонала – сертифицированные специалисты. Область сертификации производства определяет заказчик по согласованию с председателем комиссии органа по сертификации продукции.

Квалифицированным специалистам по сертификации следует учитывать юридическую ответственность за качество производства и продуктов, за результаты сертификации и достоверность документации при применении программных продуктов пользователями. Соответствие продукции требованиям и документам они должны уметь оценивать на основе данных об испытаниях продукции в процессе и/или при завершении производства. Если в соответствии с действующим в стране законодательством к определенной продукции предъявляют обязательные специальные требования качества, установленные национальными стандартами или другими нормативными документами, то при сертификации производства

следует уметь проверять способность и гарантии специалистов предприятия обеспечивать соблюдение этих требований.

Результатом положительных испытаний является сертификат соответствия – документ, изданный в соответствии с правилами системы сертификации, удостоверяющий, что обеспечивается необходимая уверенность в том, что должным образом идентифицированная продукция, процесс или услуга соответствует конкретным стандартам или другим нормативным документам.

Основной целью сертификации технологий проектирования и производства систем и программных средств является защита интересов пользователей, государственных и ведомственных интересов на основе контроля качества продукции, обеспечения их высоких потребительских свойств, повышения эффективности затрат в сфере их производства, эксплуатации и сопровождения, повышения объективности оценок характеристик и обеспечения конкурентоспособности конечного продукта. Проведение сертификации систем качества предприятия обычно планируется для достижения одной или нескольких целей:

- определения соответствия или несоответствия элементов системы качества установленным требованиям производства;
- определения эффективности внедренной системы качества предприятия с точки зрения соответствия поставленным целям для обеспечения качества продукции;
- обеспечения возможности предприятию улучшить свою систему качества;
- определения соответствия системы качества производства регламентирующим требованиям.

Добровольная сертификация применяется с целью повышения конкурентоспособности продукции, расширения сферы ее использования и получения дополнительных экономических преимуществ. Результаты сертификации должны оправдывать затраты на ее проведение вследствие получения пользователями продукции более высокого и гарантированного качества при некотором повышении ее стоимости. Таким сертификационным испытаниям подвергаются компоненты операционных систем и пакеты прикладных программ широкого применения, повышение гарантии качества, которое выгодно как для поставщиков, так и для пользователей программного продукта. В этих случаях разработчики и поставщики добровольно предоставляют ПС для сертификации с учетом экономических оценок выгодности ее проведения для их продуктов.

3) Модель организационной зрелости предприятий Capability Maturity Model

В зависимости от организационной зрелости предприятия различна степень использования информации и IT в его бизнес-процессах. Уровни организационной зрелости тесно связаны с проблемой обеспечения и контроля качества. Наибольшую известность здесь имеют серия международных стандартов ISO 9000, теория TQM (Total Quality Management) и модель CMM (Capability Maturity Model).

В конце 1987 года Институт программной инженерии США (Software Engineering Institute, SEI) при университете Карнеги-Меллона (Carnegie Mellon University) в сотрудничестве с корпорацией Mitre в рамках работы с Министерством обороны США предложил CMM в качестве методики, позволяющей крупным правительственным организациям США выбирать наилучших поставщиков ПО. Для этого предполагалось создать исчерпывающее описание способов оценки процессов разработки ПО и методики их дальнейшего усовершенствования. В итоге авторам удалось достичь такой степени подробности и детализации, что стандарт оказался пригодным и для обычных компаний-разработчиков, стремящихся качественно улучшить существующие процессы разработки, привести их к определенным стандартам. Документ назывался «Модель зрелости. Пять уровней зрелости процесса создания программного обеспечения» («Capability Maturity Model. The Five Levels of Software Process Maturity»).

В настоящее время стандарт претерпел ряд изменений. Более современная версия стандарта носит название CMMI (Integrated Capability Maturity Model)

CMM/CMMI — модель зрелости процессов создания ПО, или эволюционная модель развития способности компании разрабатывать качественное программное обеспечение. Модель CMM не противоречит международным стандартам ИСО и близка по духу концепции и теории TQM (Total Quality Management).

Ключевым понятием стандарта является зрелость организации. ***Незрелой*** считается организация, в которой процесс разработки программного обеспечения зависит только от конкретных исполнителей и менеджеров, а решения зачастую просто импровизируются «на ходу» — то, что на современном языке называется творческим подходом, или искусством. В этом случае велика вероятность превышения бюджета или выхода за рамки сроков сдачи проекта, поэтому менеджеры и разработчики вынуждены заниматься только разрешением актуальных проблем, становясь тем самым заложниками собственного программного продукта. К сожалению, на данном этапе развития находится большинство компаний (по градации CMM этот уровень обозначается числом 1).

В ***зрелой*** организации, напротив, имеются четко определенные процедуры создания программных продуктов и отработанные механизмы управления проектами. Все процедуры и механизмы по мере необходимости уточняются и совершенствуются в пилотных проектах. Оценки времени и стоимости выполнения работ основываются на накопленном опыте и достаточно точны. Наконец, в компании существуют стандарты на процессы разработки, тестирования и внедрения, а также правила оформления конечного программного кода, компонентов, интерфейсов и т.д. Все это составляет инфраструктуру и корпоративную культуру, поддерживающую процесс разработки программного обеспечения. Технология выпуска будет лишь незначительно меняться от проекта к проекту на основании абсолютно стабильных и проверенных подходов.

Выделяют 5 уровней организационной зрелости по CMM/CMMI (рис. 1):

Рисунок 1- Принцип последовательного повышения уровня зрелости

1.Начальный уровень

Этот уровень присущ большинству начинающих и малых компаний. Ведение бизнеса здесь носит хаотичный характер, что напрямую связано с борьбой за выживание. К данному уровню, как правило, относится любая компания, которой удалось получить заказ, разработать и передать заказчику программный продукт. Предприятия первого уровня не отличаются стабильностью разработок. Как правило, успех одного проекта не гарантирует успешность следующего. В компании, как правило, отсутствует стратегия развития: основное внимание уделяется решению сиюминутных тактических задач. Свойственны неравномерность процесса разработки — наличие авралов в работе. К этой категории можно отнести любую компанию, которая хоть как-то исполняет взятые на себя обязательства.

Одной из характерных черт начального уровня организационной зрелости являются спонтанные информационные связи в компании, которые аккумулируются в руководящем звене и носят в основном справочный характер. Эффективность управления в значительной степени зависит от небольшой группы единомышленников и от личности руководителя — от того, насколько четко он понимает цели и задачи развития компании. Через этот уровень проходят все предприятия и организации — кто быстрее, кто медленнее — но в конечном итоге они подступают вплотную к следующему уровню.

2.Уровень повторяемости

Данному уровню соответствуют предприятия, обладающие определенными технологиями управления проектами. На этом уровне зрелости в компании уже возможна успешная реализация задуманных проектов, что достигается благодаря жесткому управлению, оперативному планированию и контролю. Основные бизнес-процессы становятся повторяемыми и управляемыми, они приобретают устойчивый характер. Компании начинают искать пути снижения издержек, и, прежде всего, за счет оптимизации повторяющихся процессов.

Планирование и управление в большинстве случаев основывается на имеющемся опыте. Как правило, в компании данного уровня уже выработаны внутренние стандарты и организованы специальные группы проверки качества. В компании начинают формироваться корпоративные традиции и культура, однако по-прежнему отсутствует интеграция информации, а сами информационные потоки остаются неформализованными.

3. Уровень регламентируемости

На этом уровне процессы (как в управлении, так и в производстве) становятся формализованными и настолько повторяемыми, что их можно описать и задокументировать (то есть описаны все типичные действия, необходимые для многократного повторения: роли участников, форматы документов, производимые действия и пр.). Для создания и поддержания подобного стандарта в актуальном состоянии в организации уже подготовлена специальная группа.

В компаниях появляются описания ролевых функций сотрудников внутри организации или список задач, которые должен выполнять сотрудник внутри того или иного подразделения. Компания постоянно проводит специальные тренинги для повышения профессионального уровня своих сотрудников. Начиная с этого уровня организация перестает зависеть от личностных качеств конкретных разработчиков и не имеет тенденции скатываться на нижестоящие уровни. Абстрагирование от разработчиков обусловлено продуманным механизмом постановки задач и контроля исполнения.

Все процессы стандартизированы, документированы и объединены в общий информационный поток. Благодаря этому в организации появляется возможность анализа информации по всем аспектам управленческой деятельности, а также получения оперативной информации о степени использования ресурсов. Тем не менее, в таких компаниях практически отсутствует процесс постановки долгосрочных целей, а планирование основывается на принципе «от достигнутого» (т. е. на показателях прошлых периодов). Следует отметить, что в обработке информации при этом преобладает ретроспективный анализ.

Для предприятий, находящихся на этом уровне, характерно формирование стратегии развития. Как только такие решения начинают приниматься на основе анализа, это означает, что предприятие переходит на следующий этап организационной зрелости.

4. Уровень управляемости

Здесь приоритетным направлением становится повышение качества продукции или предоставляемых услуг, а целью — достижение рыночной привлекательности и увеличение доли рынка, т. е. именно то, к чему стремится любая компания, добивающаяся успеха в том сегменте рынка, где она работает. Это уровень, при котором устанавливаются количественные показатели качества. В организации формируются внутрикорпоративные стандарты качества, касающиеся не только собственной продукции или процессов производства, но и всей цепочки поставки — от партнеров (контрагентов) до клиентов.

Наличие и сохранение постоянных клиентов дает возможность долгосрочного планирования бизнеса и прогнозирования будущих продаж. В компании налажены стратегические и оперативные взаимосвязи, а для принятия решений активно используются обратные связи, в частности данные от клиентов.

Попытки принимать решения не только на основе анализа предыдущего опыта, но и на основе прогнозов будущего развития, стратегическое планирование с учетом тенденций (для чего необходимы корпоративные базы знаний) обуславливают постепенный переход организации на последний, высший уровень организационного развития.

5. Уровень оптимизируемости

Достичь этого уровня чрезвычайно трудно, и удается это лишь немногим компаниям, лидирующим в индустрии. Здесь управление качеством по количественным показателям происходит по всей цепи взаимосвязанных процессов, а модификация или совершенствование системы — по результатам обратной связи. Для организации характерно не только построение стратегических планов, но и оптимизация путей их достижения. Стратегия компании направлена на достижение организационного, финансового, технологического преимущества. Мероприятия по совершенствованию рассчитаны не только на существующие процессы, но и на оценку эффективности ввода новых технологий. Основной задачей всей организации на этом уровне является постоянное совершенствование существующих процессов, которое в идеале призвано способствовать предупреждению возможных ошибок или дефектов.

Для трех последних уровней организационной зрелости компании — регламентируемости, управляемости и оптимизируемости - такие бизнес-задачи, как достижение успеха, увеличение доли на рынке, улучшение отношений с заказчиками, контроль затрат, являются первоочередными. Для того чтобы компания могла успешно решать эти задачи и добиваться успеха в своем бизнесе, она должна правильно и эффективно использовать информацию о протекающих процессах и внешней среде.

Из градации уровней видно, что технологические требования сохраняются только до 3-го уровня, далее же в основном следуют требования к административному управлению. То есть уровни 4 и 5 по большей части управленческие и для их достижения важно не только выпустить программный продукт, но и проанализировать ход проекта, а также построить планы на будущий проект, основываясь на текущих шаблонах.

Контрольные вопросы:

1. Понятие «программный продукт»: дать определение и охарактеризовать его особенности.
2. В чём состоят особенности формирования себестоимости и цены программного продукта?
3. Какие виды программных продуктов выделяются с точки зрения рынка?
4. Понятие «модель предметной задачи».
5. Какие аспекты (части) необходимо моделировать при проектировании ПО?
6. Жизненный цикл программного обеспечения.
7. Группы процессов жизненного цикла программного продукта (ГОСТ Р ИСО МЭК 12207-2010).
8. К какой группе относятся технические процессы в ГОСТ Р ИСО/МЭК 12207-2010? Назовите процессы, входящие в данную группу.
9. Перечислите процессы в группе «Процессы реализации программных средств»» по ГОСТ Р ИСО/МЭК 12207-2010. Поясните, что они включают.
10. Какие процессы относят к группе «Процессы поддержки программных средств» по ГОСТ Р ИСО/МЭК 12207-2010. Почему?
11. Определите понятие «Модель жизненного цикла программного обеспечения».
12. Каковы область эффективного применения, сильные и слабые стороны каскадной модели жизненного цикла программного продукта и почему?
13. Каковы область эффективного применения, сильные и слабые стороны итерационной модели жизненного цикла программного продукта и почему?

14. Какова область эффективного применения, сильные и слабые стороны спиральной модели жизненного цикла и почему?
15. Сертификация программного продукта.
16. Модели оценки зрелости процессов СММ.
17. На оценку каких способностей коллективов-разработчиков программных продуктов ориентированы различные уровни модели СММ?
18. Как соотносятся требования СММ с требованиями стандартов ИСО 9000?

Список литературы:

1. Орлов С. А. Программная инженерия: Технологии разработки программного обеспечения. Учебник для вузов. 5-е издание обновлённое и дополненное [Текст] – СПб.: Питер, 2016. – 640 с.: ил. – ISBN 978-5-496-01917-0 Режим доступа URL: https://www.studmed.ru/orlov-sa-tehnologiya-razrabotki-programmnogo-obespecheniya_fc460ac2b04.html
2. Зубкова, Т.М. Технология разработки программного обеспечения: учебное пособие / Т.М. Зубкова; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего образования «Оренбургский государственный университет», Кафедра программного обеспечения вычислительной техники и автоматизированных систем. - Оренбург: ОГУ, 2017. - 469 с.: ил. - Библиогр.: с. 454-459. - ISBN 978-5-7410-1785-2; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=485553.
3. Антамошкин, О.А. Программная инженерия. Теория и практика : учебник / О.А. Антамошкин ; Министерство образования и науки Российской Федерации, Сибирский Федеральный университет. - Красноярск : Сибирский федеральный университет, 2012. - 247 с. : ил., табл., схем. - Библиогр.: с. 240. - ISBN 978-5-7638-2511-4 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=363975.
4. Гагарина Л.Д., Кокорева Е.В., Виснадул Б.Д. Технология разработки программного обеспечения: учебное пособие / М.: ИД «ФОРУМ»: ИНФРА-М, 2008 г. – 400 с. (Высшее образование).
5. Липаев В.В. Обеспечение качества программных средств. Методы и стандарты, М.: СИНТЕГ, 2001 г.
6. Стандартизация и разработка программных систем [Текст]: учеб. пособие / В.Н. Гусятников, А.И. Безруков. - М.: Финансы и статистика: ИНФРА-М, 2010 (Смоленск). - 286 с.: ил. - Библиогр.: с. 283 - 286 (47 назв.)