АВТОНОМНАЯ НЕКОММЕРЧЕСКАЯ ОРГАНИЗАЦИЯ

«ПРОФЕССИОНАЛЬНАЯ ОБРАЗОВАТЕЛЬНАЯ ОРГАНИЗАЦИЯ»

«ОТКРЫТЫЙ ТАВРИЧЕСКИЙ КОЛЛЕДЖ»

УТВЕРЖДАЮ

Директор

АНО «ПОО» «Открытый

Таврический колледж»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_Г.П. Узунова

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_20\_\_ г.

ПЛАНЫ (КОНСПЕКТЫ) ЛЕКЦИЙ

ПО междисциплинарному курсу

**МДК.02.02 Инструментальные средства разработки программного обеспечения**

ПО СПЕЦИАЛЬНОСТИ

09.02.07 Информационные системы и программирование

Квалификация: программист

БАЗОВЫЙ УРОВЕНЬ ПОДГОТОВКИ

ФОРМА ОБУЧЕНИЯ

ОЧНАЯ

Симферополь, 2025 г.

**Лекционное занятие № 1-2**

**Тема:** «КлассификацияИТ-проектов. Понятие репозитория проекта, структура проекта»

**Количество часов**: 4

**Цель**:

– учебные: сформировать информационно-наглядное представление о предмете и содержании «МДК.02.02 Инструментальные средства разработки программного обеспечения», взаимосвязь со смежными дисциплинами и курсами; изучить основные понятия, связанные с инструментальными средствами разработки программного обеспечения; познакомиться с классификацией ИТ-проектов; рассмотреть необходимость применения репозитория проекта при работе с ИТ-проектами;

– воспитательные: подвести к пониманию важности изучения «МДК.02.02 Инструментальные средства разработки программного обеспечения» для специальности «Информационные системы и программирование»; содействовать воспитанию деловых и профессиональных качеств, профессиональному развитию;

– развивающие: развивать интерес к профессии, формировать умение обобщать, делать выводы.

**План:**

1. Основные понятия связанные с инструментальными средствами разработки программного обеспечения.

2. Классификация ИТ-проектов.

3. Понятие репозитория проекта, структура проекта.

**Вступление**

Профессиональный модуль **«ПМ.02 Осуществление интеграции программных модулей»** направлен на овладение студентами следующего вида профессиональной деятельности: осуществление интеграции программных модулей. Профессиональные компетенции в соответствии с ФГОС СПО по специальности 09.02.07 Информационные системы и программирование:

ПК 2.1. Разрабатывать требования к программным модулям на основе анализа проектной и технической документации на предмет взаимодействия компонент;

ПК 2.2. Выполнять интеграцию модулей в программное обеспечение;

ПК 2.3. Выполнять отладку программного модуля с использованием специализированных программных средств;

ПК 2.4. Осуществлять разработку тестовых наборов и тестовых сценариев для программного обеспечения;

ПК 2.5. Производить инспектирование компонент программного обеспечения на предмет соответствия стандартам кодирования.

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся в ходе освоения профессионального модуля должен:

1) знать:

* модели процесса разработки программного обеспечения;
* основные принципы процесса разработки программного обеспечения;
* основные подходы к интегрированию программных модулей;
* основы верификации и аттестации программного обеспечения.

2) уметь:

* использовать выбранную систему контроля версий;
* использовать методы для получения кода с заданной функциональностью и степенью качества.

3) иметь практический опыт в:

* интеграции модулей в программное обеспечение;
* отладке программных модулей.

Структура «ПМ.02 Осуществление интеграции программных модулей»:

– МДК.02.01 Технология разработки программного обеспечения;

– МДК.02.02 Инструментальные средства разработки программного обеспечения;

– МДК.02.03 Математическое моделирование;

– УП.02 Учебная практика;

– ПП.02 Производственная практика;

– ПМ.2.ЭК Квалификационный экзамен (по модулю).

Междисциплинарный курс «МДК.02.02 Инструментальные средства разработки программного обеспечения» 10 л. + 15 пр.з. = дифференцированный зачет.

**Содержание «МДК.02.02 Инструментальные средства разработки программного обеспечения»:**

Тема 1. Современные технологии и инструменты интеграции.

Тема 2. Инструментарий тестирования и анализа качества программных средств.

**1. Основные понятия связанные с инструментальными средствами разработки программного обеспечения**

*Задача* (problem, task) – проблема, подлежащая решению. Задача означает проблему, подлежащую реализации с использованием средств информационных технологий.

С позиций специфики разработки и вида программного обеспечения будем различать два класса задач:

1) *технологические задачи* ставятся и решаются при организации технологического процесса обработки информации на компьютере. Технологические задачи являются основой для разработки сервисных средств программного обеспечения в виде утилит, сервисных программ, библиотек процедур и др., применяемых для обеспечения работоспособности компьютера, разработки других программ или обработки данных функциональных задач;

2) *функциональные задачи* требуют решения при реализации функций управления в рамках информационных систем предметных областей. Например, управление деятельностью торгового предприятия, планирование выпуска продукции, управление перевозкой грузов и т.п. Функциональные задачи в совокупности образуют предметную область и полностью определяют ее специфику.

*Предметная (прикладная) область (application domain)* – совокупность связанных между собой функций, задач управления, с помощью которых достигается выполнение поставленных целей.

*Постановка задачи* (problem definition) – это точная формулировка решения задачи на компьютере с описанием входной и выходной информации.

**ГОСТ 19.101-2024 «Единая система программной документации. Виды программ и программных документов»** распространяется на программы для средств вычислительной техники, программные документы и устанавливает виды программ и программных документов независимо от их назначения и области применения. Согласно данному стандарту: **программа** – совокупность команд и данных, обеспечивающая выполнение заданной последовательности действий средствами вычислительной техники. Программа может применяться самостоятельно и (или) в составе других программ. Выделяют следующие виды программ:

– *программный компонент* – программа, рассматриваемая как единое целое, выполняющая законченную функцию;

– *программный комплекс* – программа, состоящая из двух или более программных компонентов и/или программных комплексов, выполняющих взаимосвязанные функции;

– *комплекс программ* – совокупность программ, выполняющих не взаимосвязанные или не посредственно не связанные функции.

Программный компонент может применяться самостоятельно, в составе программного комплекса или комплекса программ. Программный комплекс может применяться самостоятельно, в составе другого программного комплекса или комплекса программ. Комплекс программ может объединять любые виды программ.

*Программное обеспечение (software)* – совокупность программ обработки данных и необходимых для их эксплуатации документов.

ГОСТ Р 59853-2021 «Информационные технологии. Комплекс стандартов на автоматизированные системы. Автоматизированные системы. Термины и определения» распространяется на автоматизированные системы, используемые в различных видах деятельности (исследования, проектирование, управление и т. п.), включая их сочетания. Настоящий стандарт устанавливает термины и определения основных понятий в области автоматизированных систем (АС). Термины, установленные настоящим стандартом, обязательны для применения во всех видах документации и литературы по автоматизированным системам, входящих в сферу работ по стандартизации и использующих результаты этих работ, и рекомендуются для применения в научно-технической, справочной и учебной литературе. Согласно данному стандарту:

– *автоматизированный процесс* – процесс, осуществляемый при совместном участии человека и средств автоматизации;

– *автоматизированная система* (АС) – система, состоящая из комплекса средств автоматизации, реализующего информационную технологию выполнения установленных функций, и персонала, обеспечивающего его функционирование;

– *автоматизированное рабочее место* (АРМ) – программно-технический комплекс АС, предназначенный для автоматизации деятельности определенной категории пользователей или определенного вида деятельности;

– *автоматизированный производственный комплекс* – автоматизированный комплекс, согласованно осуществляющий автоматизацию информационных процессов подготовки производства, производства и управления им.

– *алгоритм* – конечный набор предписаний для решения задачи посредством конечного количества операций;

– *алгоритм функционирования автоматизированной системы* – алгоритм, определяющий условия и последовательность действий компонентов автоматизированной системы в процессе её функционирования;

– *взаимодействие автоматизированных систем* – обмен информацией, данными, командами, сигналами между функционирующими АС;

– *автоматический процесс* – процесс, выполняемый техническими средствами по ранее заданному алгоритму без участия человека;

– *решение задач автоматизированных систем* – функция или часть функции АС, представляющая собой формализованную совокупность автоматических действий, выполнение которых приводит к результату заданного вида;

– *информационная технология* – приемы, способы и методы применения средств вычислительной техники при выполнении функций сбора, хранения, обработки, передачи и использования данных;

– *интегрированная автоматизированная система –* совокупность двух или более взаимоувязанных АС, в которой функционирование одной из них зависит от результатов функционирования другой (других) так, что эту совокупность можно рассматривать как единую АС.

В зависимости от вида деятельности выделяют, например, следующие виды АС: автоматизированные системы управления (АСУ), системы автоматизированного проектирования (САПР), автоматизированные системы научных исследований (АСНИ) и др.

В зависимости от вида управляемого объекта (процесса) АСУ подразделяют, например, на АСУ технологическими процессами (АСУТП), АСУ предприятиями (АСУП) и т. д.

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

– утилитарные программы – предназначены для удовлетворения нужд их разработчиков. Чаще всего утилитарные программы выполняют роль сервиса в технологии обработки данных либо являются программами решения функциональных задач, не предназначенных для широкого распространения;

– программный продукт – комплекс взаимосвязанных программ для решения определенной проблемы (задачи) массового спроса, подготовленный к реализации как любой вид промышленной продукции.

Программное средство, предназначенное для поддержки разработки других ПС, будем называть *программным инструментом разработки* *программных средств*, а устройство компьютера, специально предназначенное для поддержки разработки ПС, будем называть *аппаратным инструментом разработки ПС*.

Инструменты разработки ПС могут использоваться в течение всего жизненного цикла ПС для работы с разными программными документами. Так текстовый редактор может использоваться для разработки практически любого программного документа. С точки зрения функций, которые инструменты выполняют при разработке **ПС**, их можно разбить на следующие четыре **группы**:

– *редакторы* – поддерживают конструирование (формирование) тех или иных программных документов на различных этапах жизненного цикла. Для этого можно использовать один какой-нибудь универсальный текстовый редактор. Однако, более сильную поддержку могут обеспечить специализированные редакторы: для каждого вида документов – свой редактор. В частности, на ранних этапах разработки в документах могут широко использоваться графические средства описания (диаграммы, схемы и т.п.). В таких случаях весьма полезными могут быть графические редакторы. На этапе программирования (кодирования) вместо текстового редактора может оказаться более удобным синтаксически управляемый редактор, ориентированный на используемый язык программирования;

– анализаторы производят либо статическую обработку документов, осуществляя различные виды их контроля, выявление определенных их свойств и накопление статистических данных (например, проверку соответствия документов указанным стандартам), либо динамический анализ программ (например, с целью выявление распределения времени работы программы по программным модулям);

– *преобразователи* позволяют автоматически приводить документы к другой форме представления (например, форматеры) или переводить документ одного вида к документу другого вида (например, конверторы или компиляторы), синтезировать какой-либо документ из отдельных частей и т.п.;

– *инструменты, поддерживающие процесс выполнения программ* позволяют выполнять на компьютере описания процессов или отдельных их частей, представленных в виде, отличном от машинного кода, или машинный код с дополнительными возможностями его интерпретации. Примером такого инструмента является эмулятор кода другого компьютера. К этой группе инструментов следует отнести и различные отладчики. По существу, каждая система программирования содержит программную подсистему периода выполнения, которая выполняет программные фрагменты, наиболее типичные для языка программирования, и обеспечивает стандартную реакцию на возникающие при выполнении программ исключительные ситуации (такую подсистему будем называть *исполнительной поддержкой*). Такую подсистему также можно рассматривать как инструмент данной группы.

**Инструментальные средства разработки программного обеспечения** – это программные инструменты, предназначенные для обеспечения полного цикла проектирования программного продукта (написание текста программы, компиляция, компоновка, отладка, тестирование, сопровождение и др.). Они могут представлять собой набор отдельных программ для выполнения специальных задач проектирования программного обеспечения или интегрированную среду разработки с графическим интерфейсом со встроенными инструментами проектирования.

Рассмотрим какие существуют **инструментальные средства**:

– специализированные редакторы текстов предназначены для ввода и редактирования исходного текста программы, а также графические средства описания;

– программы-компиляторы транслируют текст программы с языка программирования в машинный код (исполняемый файл) без её выполнения;

– программы-компоновщики производят компоновку программы из нескольких модулей, подключают нужные библиотеки, определяют ссылки между модулями и связывают их между собой;

– программы-отладчики предназначены для анализа выполнения и выявления ошибок в работе программы;

– программы для создания инсталляторов требуются для разработки дистрибутивов программ;

– программы создания справочной системы позволяют организовывать файлы помощи с нужной структурой, содержанием, возможностью поиска, контекстными подсказками, перекрёстными ссылками;

– интегрированные среды разработки (IDE) – это программное обеспечение, которое объединяет в себе редактор кода, компилятор, отладчик и другие инструменты, необходимые для разработки программного обеспечения. Существует множество IDE для различных языков программирования, например, Visual Studio для C#, IntelliJ IDEA для Java и PyCharm для Python:

– системы контроля версий позволяют отслеживать изменения в исходном коде и возвращаться к предыдущим версиям при необходимости. Наиболее распространенные системы контроля версий – Git, SVN и Mercurial. Git является самой популярной и широко используемой системой контроля версий, которая предоставляет множество возможностей для совместной работы над проектом и управления изменениями;

– средства автоматической сборки и тестирования помогают автоматизировать процессы сборки, тестирования и развертывания программного обеспечения. Они также позволяют обнаруживать и устранять ошибки на ранних этапах разработки. Наиболее распространенные инструменты для автоматической сборки – Maven и Gradle. Для автоматического тестирования используются такие инструменты, как JUnit для Java, pytest для Python и NUnit для C#;

– средства анализа кода позволяют выявлять ошибки, несоответствия стандартам кодирования, утечки памяти и другие проблемы в исходном коде. Наиболее распространенные инструменты анализа кода - SonarQube и ReSharper. Они предоставляют множество возможностей для анализа кода, создания отчетов о проблемах в исходном коде.

– средства документирования кода позволяют автоматически генерировать документацию на основе комментариев в исходном коде. Наиболее распространенные инструменты документирования кода – Javadoc для Java, Sphinx для Python и Doxygen для C++;

– средства управления проектами позволяют планировать, отслеживать и координировать работу команды разработчиков. Они также позволяют управлять бюджетом проекта, контролировать сроки и ресурсы. Наиболее распространенные инструменты управления проектами – Jira, Trello и Asana.

– CASE-технологии – программные комплексы, автоматизирующие весь технологический процесс жизненного цикла программного обеспечения.

**2. Классификация ИТ-проектов**

Термин проект, происходит от латинского слова projectus, что в буквальном переводе означает «брошенный вперед». **Проект** – совокупность действий, в которых человеческие, материальные и финансовые ресурсы организованы для выполнения уникального набора работ определенного содержания в условиях ограниченной стоимости и времени. Проект имеет жизненный цикл, в пределах которого происходят необходимые изменения в соответствии с установленными количественными и качественными целями.

К **характерным чертам проекта**, которые отличают его от текущей деятельности фирмы, относят:

1) Временность означает, что каждый проект имеет определенное начало и конец. Началом проекта является его одобрение инвесторами и, соответственно, открытие финансирования проекта. Концом считается время, когда цель достигнута, или когда становится понятной последующая бесперспективность продолжения работ.

Из временности выплывает, что непосредственно проект или проектная стадия внедрения бизнес-идеи не достигает периода окупаемости вложенных средств, а тем более получение прибылей. Периоды, предусмотренные в бизнес-плане и касающиеся проекта, представлены на рис. 1.1.

Проект

Стадия эксплуатации проекта

Период прибыльности

Период окупаемости

Период затрат

Период доходности

Рисунок 1.1 – Стадии реализации бизнес-плана

Проект прекращается, когда поставленная цель достигнута, а текущие работы имеют циклический характер, то есть они длятся, пока фирма занимается предпринимательской деятельностью.

2) У*никальность* –проект является таким, пока работы по его реализации отличаются от тех, которые осуществлялись фирмой в своей текущей деятельности. Уникальность проекта не значит, что действия по его проведение имеют неповторимый характер, то есть для одних фирм конкретный комплекс работ может быть уникальным, а для других нет.

3) *Целевая направленность* проекта означает, что круг задач, которые решаются в ходе реализации проекта, четко очерчен теми из них, которые непосредственно ведут к цели проекта. Другими словами, все функции, хоть бы какими важными они были, но если они непосредственно не определяют сроки выполнения проекта, остаются снаружи проектной среды. Кроме того, целевая направленность помогает точнее установить временные рамки проекта.

Цель проекта должна быть максимально конкретной.Успешная реализация проекта во многом зависит от формирования содержания проекта. В его состав должны входить только те действия, которые непосредственно ведут к созданию продукта проекта и отвечают отмеченным характеристикам проекта. Все другое необходимо решать в ходе текущей деятельности предприятия.

**ИТ-проект** – это процесс создания, внедрения и поддержки информационных технологий или цифровых продуктов. IT-проекты могут отличаться по масштабу и сложности, а также направлению и сфере деятельности – начиная от разработки ПО и создания веб-сайтов заканчивая интеграцией корпоративных систем и внедрением облачных решений.

ИТ-проекты можно классифицировать по различным критериям, в зависимости от их целей, масштабов и используемых технологий. Рассмотрим основные виды ИТ-проектов:

1. **Разработка ПО**. Направлены на создание новых программных продуктов, приложений или систем, удовлетворяющих конкретные потребности пользователей или бизнеса. Примером могут быть разработка CRM-системы, мобильного приложения или ERP-системы для управления ресурсами предприятия.
2. **Модернизация и обновление инфраструктуры**. Проект такого типа может быть направлен на улучшение существующих IT-систем и инфраструктуры для повышения их надежности и безопасности. Примеры — обновление серверного оборудования, переход на облачные технологии или улучшение сетевой безопасности.
3. **Интеграция систем**. Цель таких проектов — объединение различных информационных систем и приложений для обеспечения их совместной работы. Например, возможна интеграция баз данных, программного обеспечения и сетевых инфраструктур для создания единой системы, которая упрощает бизнес-процессы и улучшает эффективность работы.
4. **Разработка сайтов и приложений**. Этот тип проектов может включать создание корпоративных сайтов, интернет-магазинов, порталов для взаимодействия с клиентами и многое другое.
5. **Автоматизация бизнес-процессов**. В этих проектах создаются или внедряются системы, которые автоматизируют различные бизнес-процессы, такие как управление запасами, бухгалтерский учёт, обработка заказов. Это позволяет сократить время на выполнение рутинных задач, снизить количество ошибок и повысить общую эффективность работы компании.
6. **Обеспечение информационной безопасности**. Эти проекты направлены на защиту информационных систем и данных от различных угроз, таких как хакерские атаки, вирусы, утечки данных и другие риски. В рамках таких проектов могут внедряться системы шифрования, разработки политики безопасности, настройка систем мониторинга и другие меры.
7. **Миграция данных**. Категория проектов, связанных с переносом данных с одной платформы на другую, например, при переходе на новые системы или облачные сервисы. Миграция данных требует тщательной подготовки, чтобы избежать потери данных и обеспечить их целостность.
8. **Облачные проекты**. В последние годы облачные технологии становятся всё более популярными, и многие IT-проекты связаны с переходом на облачные платформы. Сюда можно отнести направленные на создание облачных хранилищ данных, разработку облачных приложений и сервисов, а также миграцию существующих систем в облако.
9. **ИИ и машинное обучение**. В этой категории проектов создаются системы, которые используют методы ИИ и машинного обучения для решения сложных задач, таких как анализ данных, прогнозирование, автоматизация процессов. Например, разработка чат-ботов, систем рекомендаций, аналитических платформ и других решений на базе AI.
10. **Разработка IoT-решений**. Интернет вещей (IoT) — это технология, позволяющая устройствам взаимодействовать друг с другом через интернет. IT-проект в этой области может включать разработку умных устройств, систем мониторинга, управления и автоматизации, а также интеграцию IoT с существующими системами.
11. **Создание и управление сетевой инфраструктурой**. В рамках таких проектов создаются или модернизируются сети передачи данных, включая локальные сети (LAN), глобальные сети (WAN), беспроводные сети (Wi-Fi) и другие. Важными аспектами таких проектов являются обеспечение надёжности, безопасности и высокой пропускной способности сети.
12. **Разработка игр и развлечений**. IT-проект в игровой индустрии представляет собой сложный и многоэтапный процесс, включающий разработку сценариев, графики, анимации, программирование игровых движков и тестирование. Игры могут быть как для настольных компьютеров и консолей, так и для мобильных устройств.

Любой проект в IT проходит через такие ключевые этапы:

1. Инициация. Определяется основная идея, цели в проекте, задачи и предполагаемые результаты. Также проводится предварительная оценка стоимости, ресурсов и времени, которые нужно потратить на создание уникального продукта.
2. Планирование. Разрабатывается развернутый план проекта, включающий график, бюджет, распределение ресурсов и определение рисков. Важно учитывать все возможные варианты развития событий и подготовить план на случай возникновения непредвиденных обстоятельств.
3. Разработка и реализация. Непосредственное создание продукта или внедрение системы. На этом этапе многое зависит от типа проекта — каждый уникален по-своему и, исходя из этого, определяется, какие технологии и методологии лучше задействовать.
4. Тестирование и контроль качества. После завершения разработки продукт проходит через серию тестов, чтобы убедиться в его работоспособности и соответствии исходным требованиям. Необходимо выявить и устранить все возможные ошибки и баги, которые могут помешать успешному внедрению продукта.
5. Внедрение. После завершения тестирования продукт внедряется в рабочую среду. На этом этапе в проектах может проходить обучение пользователей, интеграция с существующими системами и настройка программного обеспечения.
6. Сопровождение. После завершения внедрения начинается этап поддержки, на котором обеспечивается стабильная работа системы, исправление возникающих проблем и обновление продукта по мере необходимости.

Все шесть этапов являются логичными для ИТ-проектов различной направленности, масштаба и специфики. Далее рассмотрим, какие виды ИТ-проектов бывают и чем они отличаются друг от друга.

Результатами ИТ-проекта могут быть:

* продукт, представляющий собой компонент другого изделия, улучшение изделия или конечное изделие;
* услуга или способность предоставлять услугу (например, бизнес-функция, поддерживающая производство или дистрибуцию);
* улучшение существующей линейки продуктов или услуг;
* уникальный результат, например, исследовательского проекта, приносящий новые знания для общества.

**3. Понятие репозитория проекта, структура проекта**

Основные компоненты инструментальных систем технологий программирования: репозиторий, инструментарий, интерфейсы

**Инструментальная система технологии программирования** – это интегрированная совокупность программных и аппаратных инструментов, поддерживающая все процессы разработки и сопровождения больших ПС в течение всего его жизненного цикла в рамках определенной технологии. Она помимо интегрированности обладает еще свойствами комплексности и ориентированности на коллективную разработку. Это означает, что она базируется на согласованности продукции технологических процессов. Тем самым, инструментальная система в состоянии обеспечить, по крайней мере, контроль полноты (комплектности) создаваемой документации (включая набор программ) и согласованности ее изменения (версионности). Поддержка инструментальной системой фазы сопровождения ПС, означает, что она должна обеспечивать *управление конфигурацией ПС*. Кроме того, инструментальная система поддерживает управление работой коллектива и для разных членов этого коллектива обеспечивает разные права доступа к различным фрагментам продукции технологических процессов и поддерживает работу менеджеров по управлению коллективом разработчиков. Инструментальные системы технологии программирования представляют собой достаточно большие и дорогие ПС, чтобы как-то была оправданна их инструментальная перегруженность. Поэтому, набор включаемых в них инструментов тщательно отбирается с учетом потребностей предметной области, используемых языков и выбранной технологией программирования.

С учетом обсужденных свойств инструментальных систем технологии программирования можно выделить три их основные компоненты:

* 1. репозиторий,
  2. инструментарий,
  3. интерфейсы.

Общее значение термина «репозиторий» в контексте информационных систем – актуализируемое хранилище электронных данных.

**Репозиторий** – хранилище информации, связанной с проектом разработки программного продукта в течение всего его жизненного цикла.

*Репозиторий* – это хранилище файлов, предназначенное для организации совместной работы программистов по созданию какой-либо программы.

Большинство технологических подходов к разработке ПО предполагает работу с **тремя основными типами информации** – модельными спецификациями, интерфейсом прикладного программиста и окружением проекта. В соответствии с этими типами выделяют и три класса уровней репозиториев:

1. Модельный.

2. Программного интерфейса.

3. Окружения.

**Уровень моделирования** достаточно хорошо может быть описан универсальным языком UML. Данный язык является абстрактным, не привязанным к конкретной модели. Язык дает возможность описать зависимости элементов, иерархию, взаимосвязи, свойства.

**Уровень программного интерфейса** разумно описывать с помощью языка определения интерфейсов IDL, обеспечивающего независимость спецификации интерфейсов от их реализации. Уровень играет не только роль промежуточного слоя – его средства также поддерживают распределенное программирование.

**Репозиторий окружения** программного проекта предназначен для хранения информации, разделяемой компонентами и подкомпонентами систем программирования в процессе их работы.

Основными группами и подгруппами полезной информации считаются следующие:

*1. Языково-независимая группа:*

* информация для отладчика;
* информация для анализатора исходных текстов.

*2. Языково-зависимая группа:*

- информация для шаблонов;

- коды встроенных функций;

* виртуальные функции.

*3. Группа контроля репозитория:*

- контроль информации о версиях;

- контроль отношения к проекту;

- тип параллельной обработки;

- тип управления репозиторием;

- проверка цифровой подписи.

**Главными достоинствами применения репозиториев окружения являются:**

* эффективность работы с информацией;
* использование информации для целей оптимизации;
* распределенность (из которой следуют доступность, параллелизм и специализация);
* модульность, включающая независимость от конкретных инструментов (например, компиляторов);
* возможность работы с репозиторием как в архитектуре «клиент-сервер», так и в «связанном» с инструментом режиме.

**Виды репозиториев:**

1) **Системы контроля версий.** Использование репозитория позволяет программистам придать процессу коллективной работы организованный характер. С помощью репозитория ведется учет того, кем и когда внесены изменения в хранящиеся файлы, репозиторий позволяет определить, в чем именно заключались эти изменения, а в случае необходимости – возвратить файлы в исходное состояние. В таком случае в качестве репозитория рассматриваются **системы контроля версий (системы управления версиями)**, которые позволяют как сохранить все изменения, внесенные в код, так и нескольким разработчикам работать над одним проектом. Например, Git – распределенная система управления версиями. Принцип работы:

* есть один общий проект, который лежит удаленно (на сервере);
* в проекте участвует команда разработчиков. Каждый из них перед началом работы делает копию с удаленной площадки к себе на ПК, и работает с ней, не затрагивая работу других разработчиков;
* периодически разработчик отправляет свои изменения на удаленную площадку (push) и забирает оттуда изменения других разработчиков (pull).

Git был разработан в 2005 году Линусом Торвальдсом, для того чтобы другие разработчики могли вносить свои изменения в ядро Linux. Напрямую – с помощью командной строки и/или через спец программы с графическим пользовательским интерфейсом.

2) **Репозиторий-портал –** веб-портал, совмещающий в себе функции каталога ПО, инструмента для организации сообществ по разработке ПО, среды информационного обмена для разработчиков, технической инфраструктуры разработки проектов по созданию ПО и инструмента для загрузки разработанного ПО. Например, SourceForge, Google code.

3) **Репозиторий-каталог** – каталог программного обеспечения, направленный на конечных пользователей ПО, который может выступать как хранилищем файлов, так и хранилищем ссылок на другие сайты, где можно загрузить соответствующие файлы. Например, СОФТ@mail.ru.

4) **Репозиторий пакетов** – репозитории как инфраструктура разработки операционных систем, включающих, помимо системного ПО, любые программы пользовательского и серверного назначения. Основная задача репозиториев этого рода – интеграция разных пакетов программ в единую систему. Объектом хранения в таких репозиториях выступают пакеты программ, где каждое наименование ПО представлено в виде отдельного пакета. Например, Ubuntu.

**Структура системы** – это способ организации связей и отношений между элементами (подсистемами). При разработке структуры системы задается описание множества элементов системы и связей между ними, распределение задач по уровням и элементам системы, выбор комплекса средств, обеспечивающих их эффективное решение.

**Структура проекта** представляет собой дерево ориентированных на продукт компонентов, представленных оборудованием, работами, услугами и информацией, полученными в результате реализации проекта.

***Структура проекта призвана определить*** продукцию, которую необходимо разработать или произвести, и связать элементы работы, которую предстоит выполнить, как между собой, так и с конечной целью проекта. Формирование структуры начинается с разделения целей проекта на значительно меньшие блоки работ, вплоть до достижения самих мелких позиций, подлежащих контролю.

***Структурирование проекта должно включать разделение проекта по следующим признакам:***

* компоненты продукции проекта;
* этапы жизненного цикла;
* элементы организационной структуры.

***К основным задачам структуризации относятся:***

* разбивка проекта на поддающиеся управлению блоки;
* распределение ответственности за различные элементы проекта с учетом структуры организации и имеющихся ресурсов;
* точная оценка необходимых затрат – финансовых, временных, материальных;
* создание единой базы для планирования, составления смет, контроля за затратами;
* увязка работ по проекту с системой ведения бухгалтерского учета;
* конкретизация целей проекта по подразделениям компании;
* определение подрядов.

Основная задача структуризации – определить вещественные компоненты проекта. Структуризация проекта по этапам жизненного цикла называется **структурой процесса**. Процесс структуризации проекта может быть представлен в виде следующей совокупности стандартных шагов:

1. **Определение проекта.** Подразумевает четкую формулировку характера, целей и содержания проекта, а также определение всех конечных продуктов проекта с их характеристиками. Здесь же должна быть составлена иерархия целей.
2. **Уровень детализации.** На этом шаге рассматриваются различные уровни детализации плана и выявляется оптимальное количество уровней элементов структуры.
3. **Структура процесса.** Разрабатывается схема жизненного цикла проекта.
4. **Организационная структура,** т.е. организационная схема проекта, охватывающая все группы участников или отдельных участников, занятых в проекте, включая лиц из внешнего окружения, заинтересованных в реализации проекта.
5. **Структура продукта** – схема разбивки конечного продукта по подсистемам или компонентам, включая материальное, программное, информационное, сервисное обеспечение, если необходимо, то и территориальное распределение.
6. **Система бухгалтерских счетов организации** – предполагает разработку системы кодов, применяемых при структурировании проекта. Должна основываться на существующем в организации плане бухгалтерских счетов и на системе внутреннего учета.
7. **Структура разбивки проекта. Результаты шагов с 3 по 6 объединяются в единую структуру проекта.**
8. **Генеральный сводный план проекта.** Может в дальнейшем подвергаться дальнейшей детализации в процессе поиска критического пути.
9. **Матрица распределения ответственности.** В результате анализа отношений между элементами структуры проекта и организационной структурой строится матрица, где элементы структуры проекта становятся строками, а элементы организационной структуры компании – столбцами. В ячейках матрицы уровни ответственности обозначаются при помощи условных различных обозначений или кодов.













