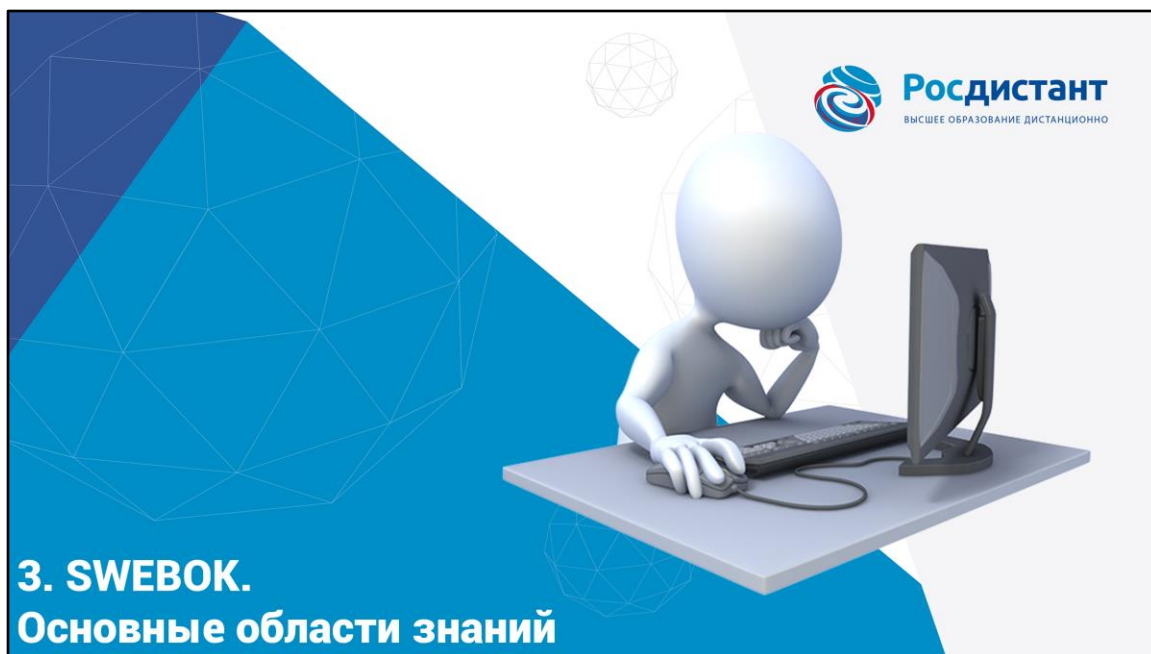




Росдистант
ВЫСШЕЕ ОБРАЗОВАНИЕ ДИСТАНЦИОННО

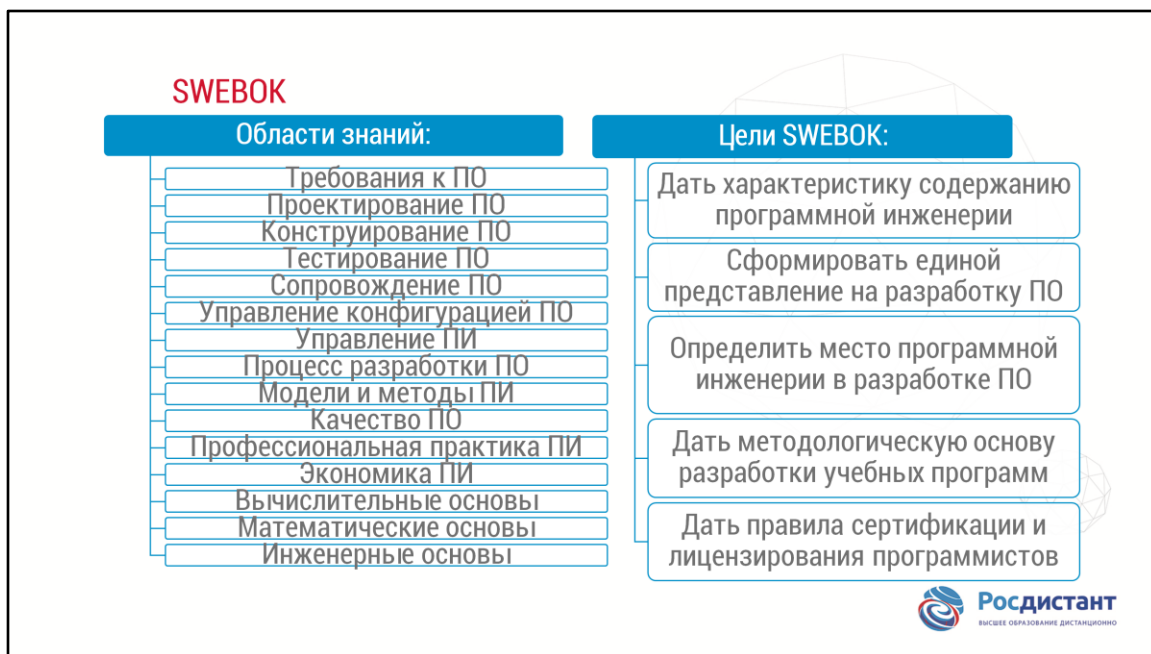
ВВЕДЕНИЕ В ПРОГРАММНУЮ

ИНЖЕНЕРИЮ 3 ЧАСТЬ



Тема 3. SWEBOOK. Основные области знаний

В ходе изучения данной темы вы должны ознакомиться с руководством по своду знаний в области программной инженерии. Вы рассмотрите общие подходы и стратегии к разработке программных продуктов, узнаете о тех задачах, которые предстоит решать на каждом этапе разработки программного обеспечения.



SWEBOOK – руководство по своду знаний в области программной инженерии. Свод знаний разделен на области знаний КА по программной инженерии, в которых приводится краткое изложение тем. SWEBOOK преследует следующие цели.

- Охарактеризовать содержание дисциплины программной инженерии;
- продвигать единый взгляд на разработку программного обеспечения во всем мире;
- прояснить место и установить границы программной инженерии по отношению к другим дисциплинам.
- А также обеспечить основу для учебных материалов и разработки учебных программ;
- обеспечить основу для сертификации и лицензирования инженеров-программистов.

Области знаний, характеризующие практику программной инженерии, следующие:

- требования к программному обеспечению;
- проектирование программного обеспечения;
- конструирование программного обеспечения;
- тестирование программного обеспечения.

- Кроме того, сопровождение программного обеспечения;
- управление конфигурацией программного обеспечения;
- управление программной инженерией;
- процесс разработки программного обеспечения.

А также в области знаний, характеризующие практику программной инженерии, входят:

- модели и методы программной инженерии;
- качество программного обеспечения;
- профессиональная практика программной инженерии.
- Кроме того, экономика программной инженерии;
- вычислительные основы;
- математические основы;
- инженерные основы.

Освоив основные методы и инструменты выделенных областей, вы сможете разрабатывать качественное программное обеспечение.

ТРЕБОВАНИЯ К ПО / ПРОЕКТИРОВАНИЕ ПО

Область знания «Требования к программному обеспечению»
выявление + согласование + анализ + спецификация + проверка
требований к программному обеспечению

- потребности и ограничения → программный продукт →
решение реальных проблем

Область знания «Проектирование программного обеспечения»
архитектура + компоненты + интерфейсы + другие характеристики
+ результат

- результат → описание архитектуры программного обеспечения



А теперь попробуем определиться с пониманием того, чем занимается каждая из выделенных областей.

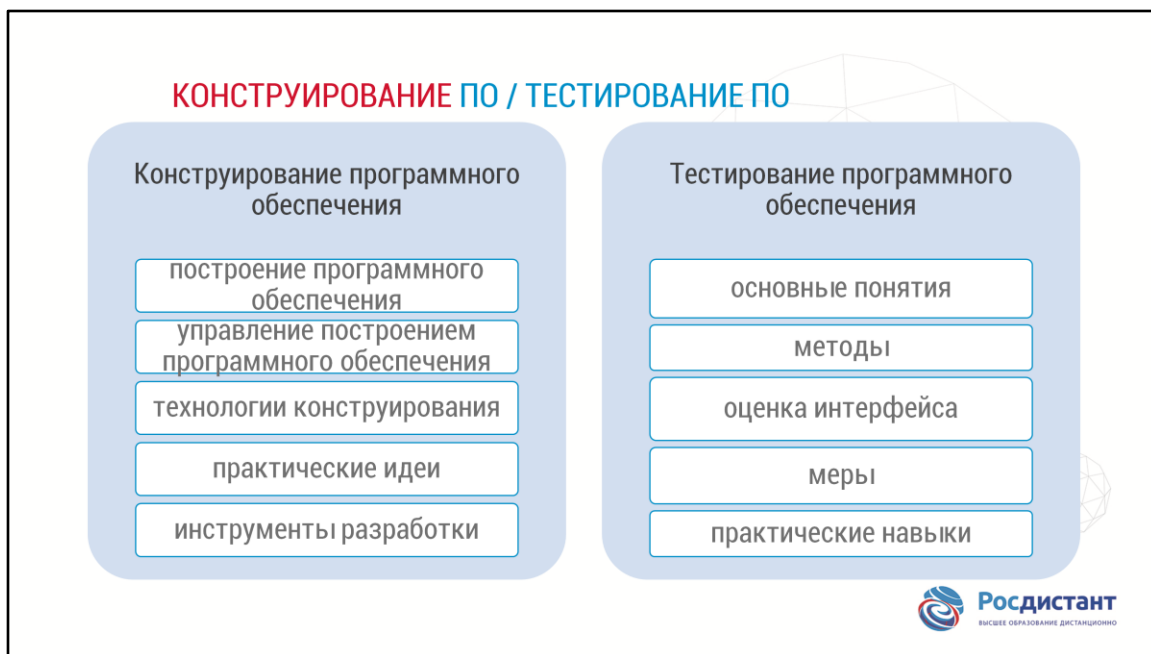
Область знания «Требования к программному обеспечению» занимается выявлением, согласованием, анализом, спецификацией и проверкой требований к программному обеспечению.

В индустрии программного обеспечения широко признано, что проекты разработки программного обеспечения критически уязвимы, когда действия в области знаний «Требования к программному обеспечению» выполняются плохо. Требования к программному обеспечению выражают потребности и ограничения, налагаемые на программный продукт, которые способствуют решению некоторых реальных проблем.

Следующая область – это область проектирования программного обеспечения, которая занимается вопросами определения архитектуры, компонентов, интерфейсов и других характеристик системы или компонентов, а также получаемого результата. Программное обеспечение Software Design KA охватывает процесс проектирования и сам конечный продукт.

Процесс проектирования программного обеспечения – это деятельность жизненного цикла программной инженерии. В ходе этой деятельности требования к программному обеспечению анализируются для создания

описания внутренней структуры программного обеспечения и его поведения. Это должно послужить основой для построения программного обеспечения. Проект программного обеспечения должен описывать архитектуру программного обеспечения. То есть то, как программное обеспечение декомпозировано и организовано на компоненты, а также интерфейсы между этими компонентами. Он также должен описывать компоненты на таком уровне детализации, который позволяет их компоновку.



Особое внимание заслуживает область, которая называется «Конструирование программного обеспечения».

Конструирование программного обеспечения относится к детальному созданию рабочего программного обеспечения. Этот процесс основан на комбинировании детального проектирования, кодирования, модульного тестирования, интеграционного тестирования, отладки и проверки.

Программа Software Construction KA включает темы, связанные с разработкой программного обеспечения, которое будет удовлетворять их требованиям и ограничениям проектирования.

Эта область знаний охватывает:

- основы построения программного обеспечения;
- управление построением программного обеспечения;
- строительные технологии;
- практические соображения;
- инструменты для создания программного обеспечения.

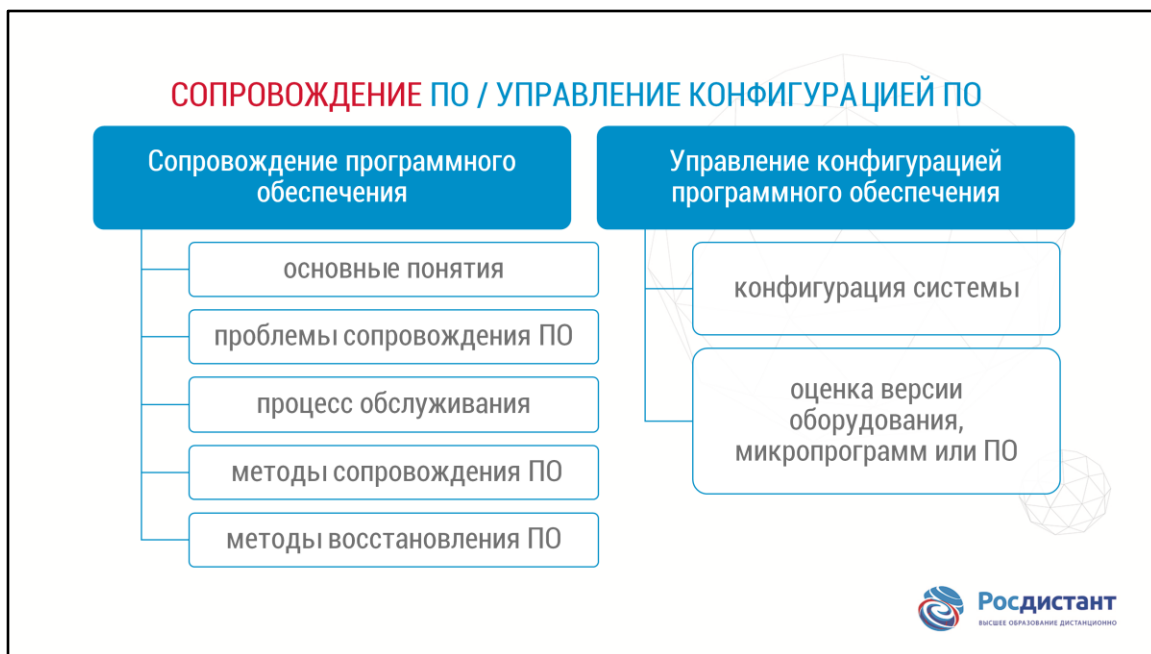
Огромное значение в разработке качественного программного обеспечения играет тестирование.

Тестирование – это деятельность, выполняемая для оценки качества продукта и его улучшения путем выявления дефектов. Тестирование программного

обеспечения включает динамическую проверку поведения программы относительно ожидаемого поведения на конечном наборе тестовых примеров. Эти тестовые примеры выбираются обычно из очень большой области выполнения.

Software Testing КА включает:

- основы тестирования программного обеспечения;
- методы тестирования;
- тестирование и оценку интерфейса человек – компьютер;
- меры, связанные с тестированием;
- практические навыки.



Сопровождение программного обеспечения включает расширение существующих возможностей, адаптацию программного обеспечения для работы в новых и измененных операционных средах и исправление дефектов. Эти категории называются усовершенствованным, адаптивным и корректирующим сопровождением программного обеспечения.

КА по обслуживанию программного обеспечения включает следующее.

- Основы обслуживания программного обеспечения, определяющие характер и необходимость обслуживания, категории обслуживания, затраты на обслуживание.
- Ключевые вопросы в сопровождении программного обеспечения, такие как: технические вопросы, вопросы управления, оценка стоимости обслуживания, оценка обслуживания программного обеспечения.
- Процесс обслуживания, методы сопровождения программного обеспечения, описывает понимание программ, реинжиниринг, обратный инжиниринг, рефакторинг, изъятие программного обеспечения.
- Методы аварийного восстановления и инструменты обслуживания программного обеспечения.

Следующая область – «Управление конфигурацией программного обеспечения». Здесь рассматривается конфигурация системы, включающая

функциональные или физические характеристики оборудования, программного обеспечения или их комбинации. Также можно рассматривать набор конкретных версий оборудования, микропрограмм или программного обеспечения, объединенных в соответствии с определенными процедурами сборки для определенной цели.

Таким образом, управление конфигурацией программного обеспечения SCM представляет собой дисциплину идентификации конфигурации системы для систематического контроля изменений конфигурации. Здесь же рассматриваются вопросы поддержания целостности и отслеживаемости конфигурации на протяжении всего жизненного цикла программного обеспечения. Область знания «Управление конфигурацией программного обеспечения» охватывает:

- управление процессом SCM ; идентификацию конфигурации программного обеспечения, контроль, учет состояния и аудит;
- управление выпуском программного обеспечения и его доставку.

УПРАВЛЕНИЕ ПРОГРАММНОЙ ИНЖЕНЕРИЕЙ / ПРОЦЕСС РАЗРАБОТКИ ПО

Управление программной инженерией

инициирование предметной области
планирование
внедрение программного проекта
приемка продукции
оценка выполнения проекта
закрытие проекта
инструменты управления ПО

Процесс разработки ПО

внедрение и изменение
процессов
определение процесса
модели и методы оценки
процессов
измерения
программные средства
обработки



Управление программной инженерией включает планирование, координацию, измерение, отчетность и контроль над проектом или программой.

КА по управлению программной инженерией охватывает следующее.

- Инициирование и определение области, то есть определение и согласование требований, технико-экономический анализ, а также анализ и пересмотр требований.
- Планирование программных проектов, рассматривающее планирование процессов; оценка усилий, затрат и графика, распределение ресурсов, анализ рисков, планирование качества.
- Введение в действие программного проекта, охватывающее измерение, отчетность и контроль; управление закупками и контрактами с поставщиками.
- Приемка продукции.
- Обзор и анализ выполнения проекта.
- Закрытие проекта.
- Инструменты управления программным обеспечением.

КА процесса разработки программного обеспечения занимается определением, реализацией, оценкой, измерением, управлением и улучшением процессов жизненного цикла программного обеспечения. Охватываемые темы включают:

- внедрение и изменение процессов, охватывающие инфраструктуру

процессов, модели внедрения и изменения процессов, а также управление процессами программного обеспечения.

- Кроме того, определение процесса, которое включает модели и процессы жизненного цикла программного обеспечения, нотации для определения процесса, адаптацию процесса и автоматизацию процесса;
- модели и методы оценки процессов;
- измерения, включающие измерение процесса, измерение продуктов, методы измерения и качество результатов;
- программные средства обработки.

МОДЕЛИ И МЕТОДЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ / КАЧЕСТВО ПО

Модели и методы программной инженерии

моделирование

виды моделей

анализ предметной области
на предмет возможности
разработки

методы разработки
программного обеспечения

Качество программного обеспечения

основные понятия

процессы управления

практические навыки



Область знания «Модель и методы разработки программного обеспечения» обращается к методам, охватывающим несколько стадий жизненного цикла; методы, специфичные для конкретных стадий жизненного цикла, охватываются другими КА. Обсуждаемые темы включают следующее.

- Моделирование. Здесь рассматриваются принципы и свойства моделей программной инженерии; предусловия, постусловия и инварианты;
- типы моделей: информационные, структурные и поведенческие;
- анализ на предмет правильности, полноты, согласованности, качества и взаимодействия.
- А также методы разработки программного обеспечения, такие как эвристические методы, формальные методы, методы прототипирования и гибкие методы.

Качество программного обеспечения – это распространенная проблема жизненного цикла программного обеспечения, которая решается во многих областях знаний. Кроме того, оценка качества программного обеспечения включает следующее.

- Основы качества программного обеспечения. К ним относятся культура разработки программного обеспечения, характеристики качества, стоимость качества, а также повышение качества программного обеспечения.

- Процессы управления качеством программного обеспечения. Например, обеспечение качества программного обеспечения, верификация и валидация, обзоры и аудиты.
- Практические навыки, такие как определение характеристики дефектов, измерение качества и инструменты качества программного обеспечения.

ПРОФЕССИОНАЛЬНАЯ ПРАКТИКА ПРОГРАММНОЙ ИНЖЕНЕРИИ / ЭКОНОМИКА ПРОГРАММНОЙ ИНЖЕНЕРИИ

Профессиональная
практика
программной
инженерии

профессионализм
кодексы этики
групповая динамика
коммуникативные навыки

Экономика
программной
инженерии

основы экономики программной инженерии
некоммерческое принятие решений
экономический риск и неопределенность
принятие решений по множеству атрибутов



Профессиональная практика программной инженерии связана со знаниями, навыками и отношением. Ими должны обладать программные инженеры, чтобы заниматься программной инженерией профессионально, ответственно и этично. Профессиональная практика программной инженерии КА охватывает:

- профессионализм: профессиональное поведение, профессиональные сообщества, стандарты программной инженерии, трудовые договоры и юридические вопросы;
- кодексы этики.
- Сложность когнитивных проблем, взаимодействие с заинтересованными сторонами, работу с неопределенностью и двусмысленностью и другие;
- коммуникативные навыки.

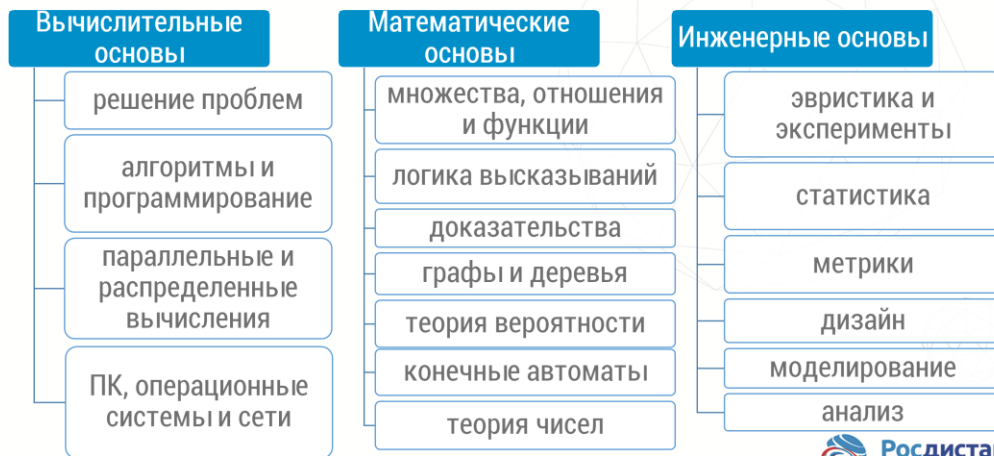
Следующая часть – это области знаний, характеризующие образовательные требования программной инженерии.

Первая область КА – «Экономика программной инженерии» занимается принятием решений в рамках бизнес-контекста для согласования технических решений с бизнес-целями организации. Охватываемые темы включают:

- основы экономики программной инженерии;
- некоммерческое принятие решений;
- предварительный расчет; экономический риск и неопределенность;

- принятие решений по множеству атрибутов.

ВЫЧИСЛИТЕЛЬНЫЕ ОСНОВЫ / МАТЕМАТИЧЕСКИЕ ОСНОВЫ / ИНЖЕНЕРНЫЕ ОСНОВЫ



Рассмотрим оставшиеся области знаний, представленные на слайде.

Вычислительные основы. КА Computing Foundations охватывает фундаментальные темы, которые обеспечивают вычислительную базу, необходимую для разработки программного обеспечения. Охватываемые темы включают:

- методы решения проблем, абстракцию;
- алгоритмы и сложность, основы программирования;
- основы параллельных и распределенных вычислений;
- организацию компьютеров, операционные системы и сетевое взаимодействие.

Математические основы. КА охватывает фундаментальные темы, которые обеспечивают математические основы, необходимые для практики разработки программного обеспечения. Охватываемые темы включают:

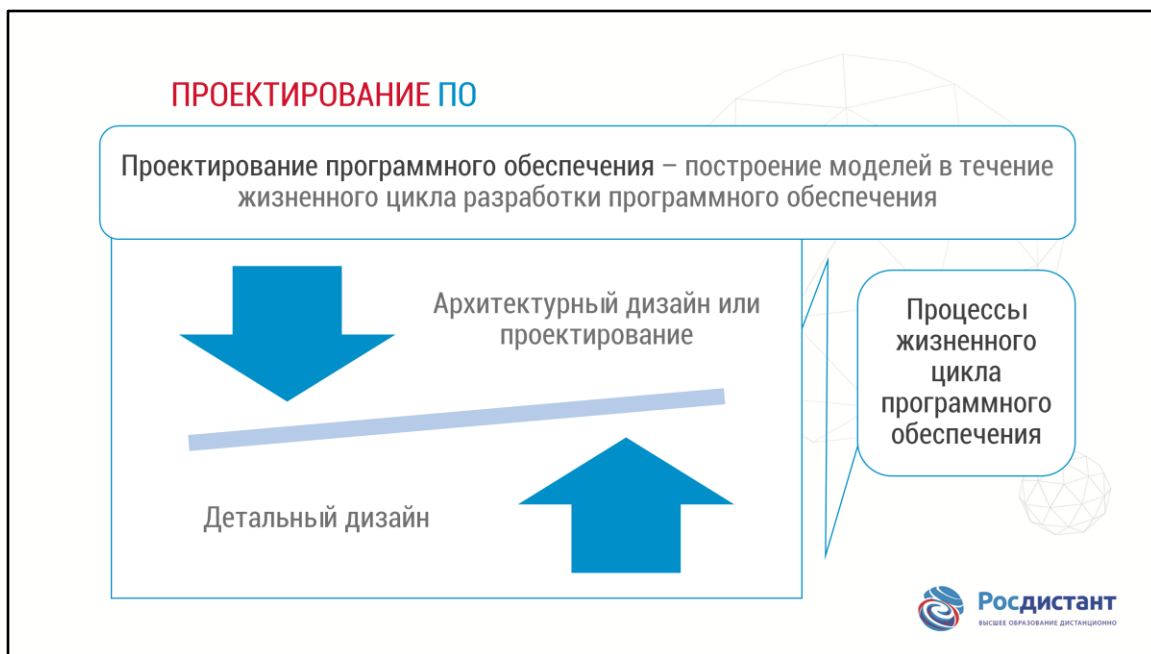
- множества, отношения и функции;
- основная логика высказываний и предикатов;
- методы доказательства;
- графы и деревья;
- дискретная вероятность;
- грамматики и конечные автоматы;

- теория чисел.

Инженерные основы. Engineering Foundations KA охватывает фундаментальные темы, которые обеспечивают инженерный фон, необходимый для практики разработки программного обеспечения. Охватываемые темы включают:

- эмпирические методы и экспериментальные методы;
- статистический анализ;
- измерения и метрики;
- инженерный дизайн;
- моделирование;
- анализ первопричин.

Таким образом, мы рассмотрели основные области знания, которые нам нужны будут в процессе разработки программного обеспечения. Остановимся более подробно на отдельных параметрах этих областей для детального представления круга задач, которые вам нужно будет освоить в процессе обучения.



Проектирование программного обеспечения представляет собой деятельность в течение жизненного цикла разработки программного обеспечения. В данном процессе требования к программному обеспечению анализируются с целью создания описания внутренней структуры программного обеспечения. Проект программного обеспечения описывает архитектуру программного обеспечения. То есть то, как программное обеспечение декомпозировано и организовано на компоненты, а также интерфейсы между этими компонентами. Он также должен описывать компоненты на уровне детализации, показывающей их конструкцию.

Проектирование программного обеспечения играет важную роль в разработке программного обеспечения. Во время разработки программного обеспечения инженеры-программисты создают различные модели, которые образуют своего рода план решения, которое необходимо реализовать. Можно проанализировать и оценить эти модели, чтобы определить, позволят ли они выполнить различные требования.

Также можно изучить и оценить альтернативные решения. Наконец, можно использовать полученные модели для планирования последующих действий по разработке. Процессы жизненного цикла программного обеспечения в проектировании включают два действия.

- Архитектурный дизайн программного обеспечения или проектирование высокого уровня. Здесь разрабатываются структура верхнего уровня и организация программного обеспечения, а также определяются различные компоненты.
- Детальный дизайн программного обеспечения включает подробное описание каждого компонента, чтобы облегчить его построение.

ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ ПО

Абстракция

- представление объекта на основе конкретной цели, оставляя без внимания дополнительную информацию
- представление именованных параметров данных
- использование трех видов абстракции: абстракция процедур, абстракция данных и абстракция управления

Связь и сплоченность

- связь - «мера взаимозависимости между модулями в компьютерной программе»
- сплоченность - «мера силы связи между элементами внутри модуля»



В общем смысле проектирование можно рассматривать как форму решения проблем. Проектирование программного обеспечения – важная часть процесса разработки программного обеспечения.

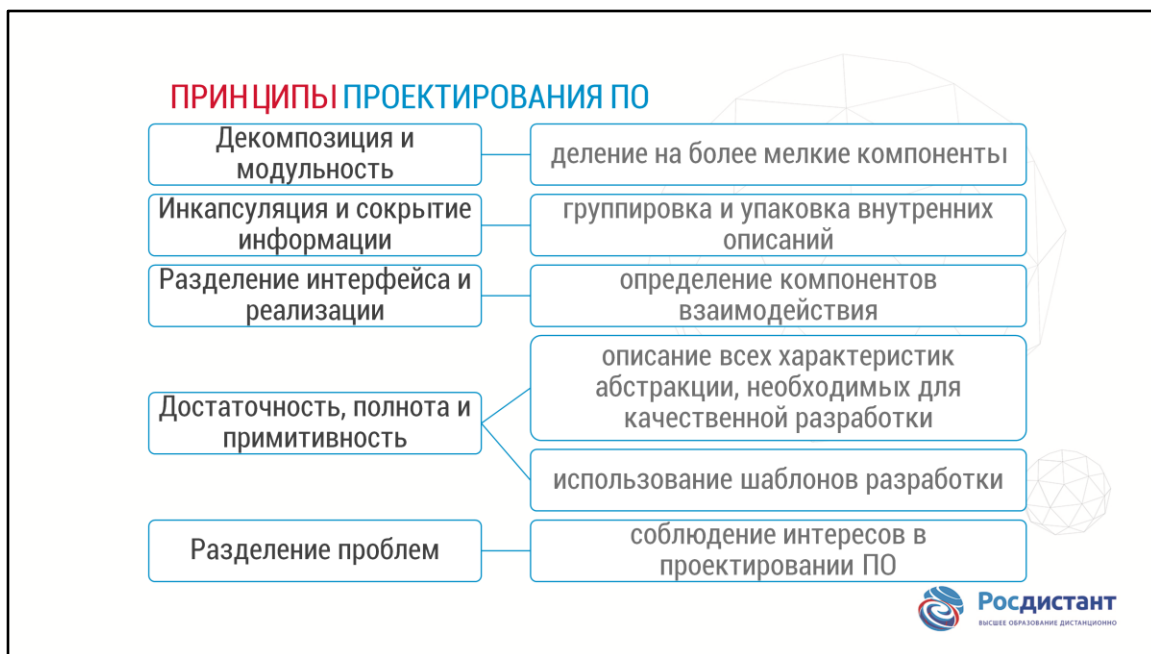
Результатом проектирования является набор моделей и артефактов, в которых записаны основные принятые решения, а также объяснение причин каждого нетривиального решения. За счет записи обоснования улучшается возможность обслуживания программного продукта в долгосрочной перспективе.

Принципы проектирования программного обеспечения являются ключевыми понятиями, лежащими в основе множества различных подходов и концепций.

Принципы разработки программного обеспечения включают следующее.

- Абстракция – это представление объекта, которое фокусируется на информации, относящейся к конкретной цели, и игнорирует остальную информацию. В контексте разработки программного обеспечения двумя ключевыми механизмами абстракции являются параметризация и спецификация. Абстракция путем параметризации представляет данные как именованные параметры. Абстракция по спецификации приводит к трем основным видам абстракции: абстракция процедур, абстракция данных и абстракция управления.
- Связь и сплоченность. Связь определяется как мера взаимозависимости

между модулями в компьютерной программе, тогда как сплоченность определяется как мера силы связи между элементами внутри модуля.



К принципам проектирования программного обеспечения также относится следующее.

- Декомпозиция и модульность означают, что большое программное обеспечение делится на ряд меньших поименованных компонентов, имеющих четко определенные интерфейсы, которые описывают взаимодействия компонентов.
- Инкапсуляция и сокрытие информации – это группировка и упаковка внутренних деталей абстракции, что делает эти детали недоступными для внешних объектов.
- Разделение интерфейса и реализации включают определение компонента путем указания общедоступного интерфейса, который отделен от деталей реализации компонента.
- Достаточность, полнота и примитивность. Достижение достаточности и полноты означает обеспечение того, чтобы программный компонент отражал все важные характеристики абстракции и ничего более. Примитивность означает, что процесс должен быть основан на шаблонах, которые легко реализовать.
- Разделение проблем – это область интересов в отношении проектирования программного обеспечения. Проблема – это область дизайна, которая имеет

отношение к одному или нескольким заинтересованным сторонам. Каждое представление об архитектуре создает одну или несколько проблем. Рассмотренные принципы являются основополагающими для проектирования программных приложений.



В разработке программного обеспечения можно выделить две ключевые проблемы.

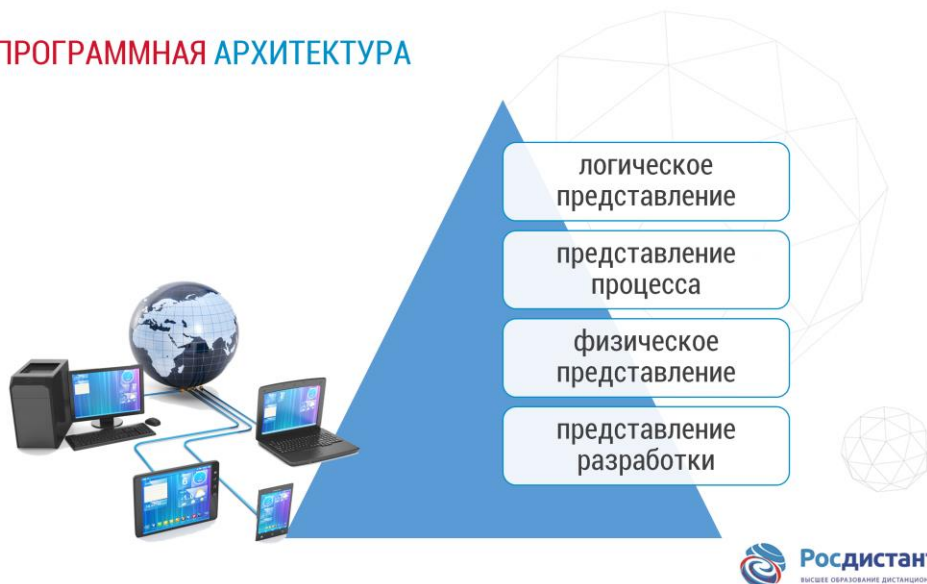
1. При разработке программного обеспечения необходимо решить ряд ключевых вопросов. Некоторые из них связаны с качеством, которое должно решать все программное обеспечение, например, производительность, безопасность, надежность, удобство использования и т. д.
2. Другой важный вопрос – как разложить, организовать и упаковать программные компоненты. Это настолько фундаментально, что все подходы к проектированию так или иначе решают эту проблему. Напротив, другие вопросы касаются некоторых аспектов поведения программного обеспечения, которые не относятся к домену приложения, но относятся к некоторым из поддерживающих доменов.

Эти проблемы часто пересекают функциональность системы. Они называются аспектами, которые обычно не являются единицами функциональной декомпозиции программного обеспечения. Они могут быть свойствами, которые системным образом влияют на производительность или семантику компонентов.

Следовательно, для проектирования программных приложений нужно придерживаться правила, что любая проблема может быть переложена в

проектное решение. Главное, грамотно декомпозировать обозначенную проблему и понять, что является именно решением, а что – свойством рассматриваемого объекта.

ПРОГРАММНАЯ АРХИТЕКТУРА



В строгом смысле программная архитектура – это набор структур, необходимых для размышлений о системе, которые включают элементы программного обеспечения, отношения между ними и их свойства.

Однако в середине 1990-х годов архитектура программного обеспечения начала формироваться как более широкая дисциплина, которая включала изучение структур и архитектур программного обеспечения в более общем плане. Это привело к появлению ряда интересных концепций проектирования программного обеспечения на разных уровнях абстракции. Некоторые из этих концепций могут быть полезны во время архитектурного проектирования, а также во время детального проектирования. Эти концепции дизайна также можно использовать для разработки семейств.

Различные высокоуровневые аспекты разработки программного обеспечения могут быть описаны и задокументированы. Эти аспекты часто называют представлениями. Представление обозначает собой частичный аспект архитектуры программного обеспечения, который показывает определенные свойства программной системы. Представления относятся к различным вопросам, связанным с проектированием программного обеспечения, например, к следующим.

- Логическое представление – удовлетворение функциональным требованиям;

- представление процесса – проблемы параллелизма;
- физическое представление – проблемы распределения;
- представление разработки, показывающее, как процесс разбит на блоки реализации с явным представлением зависимостей между блоками.



Различные авторы выделили ряд основных архитектурных стилей. Перечислим их.

- Общие конструкции, например, слои, трубы и фильтры, классная доска;
- распределенные системы, например, клиент-сервер, трехуровневая, брокерская;
- интерактивные системы;
- адаптируемые системы, например, микроядро;
- прочие.

Архитектурное проектирование – это творческий процесс. В процессе проектирования разработчики программного обеспечения должны принять ряд фундаментальных решений, которые существенно влияют на программное обеспечение и процесс разработки. Полезно думать о процессе архитектурного проектирования с точки зрения принятия решений, а не с точки зрения деятельности. Часто влияние на атрибуты качества и компромиссы между конкурирующими атрибутами качества являются основой для проектных решений.

Один из подходов к обеспечению повторного использования проектов и компонентов программного обеспечения заключается в разработке семейств программ, также известных как линейки программных продуктов. Это может

быть сделано путем выявления общих черт среди членов таких семейств и путем разработки многоразовых и настраиваемых компонентов с учетом вариативности среди членов семейств. В объектно-ориентированном программировании ключевым связанным понятием является понятие фреймворка. Это частично завершенная программная система, которая может быть расширена путем создания экземпляров определенных расширений, например, подключаемых модулей.

ПРИНЦИПЫ ПРОЕКТИРОВАНИЯ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Обучаемость

Дружественность

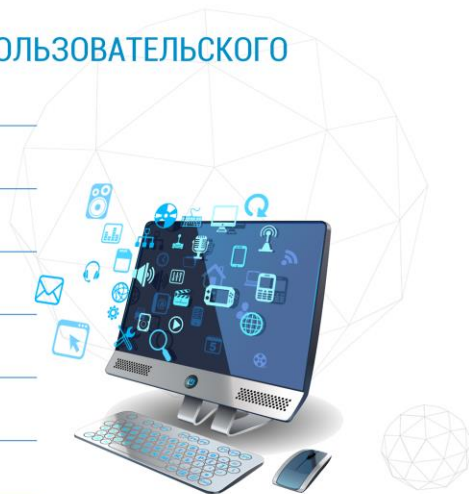
Последовательность

Отсутствие сюрпризов

Восстанавливаемость

Руководство пользователя

Многопользовательность



Проектирование пользовательского интерфейса – важная часть процесса разработки программного обеспечения. Проектирование пользовательского интерфейса должно гарантировать, что взаимодействие между человеком и машиной обеспечивает эффективную работу и управление машиной. Чтобы программное обеспечение полностью раскрыло свой потенциал, пользовательский интерфейс должен быть спроектирован таким образом, чтобы соответствовать навыкам, опыту и ожиданиям предполагаемых пользователей.

Общие принципы проектирования пользовательского интерфейса

- Обучаемость. Программное обеспечение должно быть простым в освоении, чтобы пользователь мог быстро начать работать с ним.
- Дружественность. Интерфейс должен использовать термины и концепции, взятые из опыта людей, которые будут использовать программное обеспечение.
- Последовательность. Интерфейс должен быть согласованным, чтобы сопоставимые операции активировались одинаково.
- Минимальный сюрприз. Поведение программного обеспечения не должно удивлять пользователей.
- Восстанавливаемость. Интерфейс должен обеспечивать механизмы, позволяющие пользователям восстанавливаться после ошибок.

- Руководство пользователя. Интерфейс должен давать значимую обратную связь при возникновении ошибок и предоставлять пользователям контекстную помощь.
- Разнообразие пользователей. Интерфейс должен обеспечивать соответствующие механизмы взаимодействия для разных типов пользователей и для пользователей с разными возможностями. Например, слепые, слабовидящие, глухие, дальтоники и т. д.

СТИЛИ ВЗАИМОДЕЙСТВИЯ С ПОЛЬЗОВАТЕЛЕМ



Вопрос-ответ

Прямая манипуляция

Выбор меню

Заполнение формы

Командный язык

Естественный язык



Проектирование пользовательского интерфейса должно решить две ключевые проблемы.

- Как пользователь должен взаимодействовать с программой?
- Как следует представить пользователю информацию из программного обеспечения?


Проектирование пользовательского интерфейса должно включать взаимодействие с пользователем и представление информации. При проектировании пользовательского интерфейса следует учитывать компромисс между наиболее подходящими стилями взаимодействия и представления программного обеспечения. А также между предысторией и опытом пользователей программного обеспечения и доступными устройствами. Взаимодействие с пользователем включает выдачу команд и предоставление связанных данных программному обеспечению. Стили взаимодействия с пользователем можно разделить на следующие основные стили.

- Вопрос-ответ. Взаимодействие по существу ограничивается обменом одним вопросом-ответом между пользователем и программным обеспечением. Пользователь задает вопрос программе, и программа возвращает ответ на вопрос.
- Прямая манипуляция. Пользователи взаимодействуют с объектами на экране

компьютера. Прямое манипулирование часто включает указывающее устройство, которое манипулирует объектом и вызывает действия, определяющие, что нужно делать с этим объектом.

- Выбор меню. Пользователь выбирает команду из списка команд меню.
- Заполнение формы. Пользователь заполняет поля формы. Иногда поля включают меню, и в этом случае в форме есть кнопки действий, с помощью которых пользователь может инициировать действие.
- Командный язык. Пользователь выдает команду и предоставляет соответствующие параметры, чтобы указать программе, что делать.
- Естественный язык. Пользователь выдает команду на естественном языке. То есть естественный язык – это интерфейс к командному языку, который анализируется и транслируется в программные команды.

РЕКОМЕНДАЦИИ ПРЕДСТАВЛЕНИЯ ИНФОРМАЦИИ

Подход MVC - эффективный способ показать представление информации  представляемой информации

- обратная связь
- абстрактные визуализации

Рекомендации:

- ограничение используемых цветов
- изменение цвета
- цветовая кодировка
- последовательное цветовое кодирование
- использование не только цвета



Представление информации может быть текстовым или графическим. Хороший дизайн позволяет отделить представление информации от самой информации. Подход MVC – модель – представление – контроллер. Это эффективный способ отделить представление информации от представляемой информации. Обратная связь может быть предоставлена путем повторного ввода данных пользователя во время завершения обработки. Абстрактные визуализации можно использовать, когда необходимо представить большие объемы информации. В соответствии со стилем представления информации дизайнеры также могут использовать цвет для улучшения интерфейса. Есть несколько важных рекомендаций.

- Ограничьте количество используемых цветов.
- Используйте изменение цвета, чтобы показать изменение статуса программного обеспечения.
- Используйте цветовую кодировку для поддержки задачи пользователя.
- Используйте цветовое кодирование продуманно и последовательно.
- Используйте цвета, чтобы облегчить доступ для людей с дальтонизмом или дефицитом цвета. Например, используйте изменение насыщенности и яркости цвета, старайтесь избегать сочетаний синего и красного.
- Не полагайтесь только на цвет, чтобы передать важную информацию

пользователям с разными возможностями. Например, слепота, плохое зрение, дальтонизм и т. д.

ПРОЕКТИРОВАНИЕ ПОЛЬЗОВАТЕЛЬСКОГО ИНТЕРФЕЙСА

Анализ пользователей	анализ задач пользователей, рабочей среды, другого программного обеспечения и принципов взаимодействия
Создание прототипов программного обеспечения	взаимодействие с пользователями для создания дружелюбного интерфейса
Оценка интерфейса	оценка эргономики

Проектирование пользовательского интерфейса – это итеративный процесс. Прототипы интерфейса часто используются для определения функций, организации и внешнего вида пользовательского интерфейса программного обеспечения. Этот процесс включает три основных действия.

- Анализ пользователей. На этом этапе дизайнер анализирует задачи пользователей, рабочую среду, другое программное обеспечение и то, как пользователи взаимодействуют с другими людьми.
- Создание прототипов программного обеспечения. Разработка прототипа программного обеспечения помогает пользователям направлять эволюцию интерфейса.
- Оценка интерфейса. Дизайнеры могут наблюдать за опытом пользователей с развивающимся интерфейсом.

Разработчики пользовательского интерфейса могут использовать метафоры и концептуальные модели для создания сопоставлений между программным обеспечением и некоторой системой ссылок. Данная система известна пользователям в реальном мире. Это может помочь пользователям с большей готовностью изучить и использовать интерфейс. Например, операцию «удалить файл» можно превратить в метафору с помощью значка корзины. При разработке пользовательского интерфейса инженеры-программисты должны

быть осторожны, чтобы не использовать более одной метафоры для каждой концепции. Метафоры также представляют собой потенциальные проблемы в отношении интернационализации, поскольку не все метафоры имеют смысл или применяются одинаково во всех культурах.

АНАЛИЗ И ОЦЕНКА КАЧЕСТВА ПРОЕКТИРОВАНИЯ ПО

Анализ проектирования программного обеспечения:

- неформальные и формализованные методы

Статический анализ:

- формальный или полужформальный статический анализ

Моделирование и прототипирование:

- динамические методы оценки проектирования



Различные атрибуты способствуют качеству разработки программного обеспечения, включая различные способности. К таким способностям относятся: модифицируемость, переносимость, тестируемость, удобство использования, правильность, надежность. Существует интересное различие между атрибутами качества, различимыми во время выполнения. Например, производительность, безопасность, доступность, функциональность, удобство использования. А также есть различие между теми, которые не различимы во время выполнения. Например, модифицируемость, переносимость, возможность повторного использования, тестируемость. А также теми, которые связаны с архитектурой. Например, концептуальная целостность, правильность, полнота. Различные инструменты и методы могут помочь в анализе и оценке качества разработки программного обеспечения.

- Анализ проектирования программного обеспечения: неформальные и формализованные методы определения качества артефактов проектирования. Анализ проектирования программного обеспечения также может оценить безопасность. Можно просмотреть вспомогательные средства для установки, эксплуатации и использования.
- Статический анализ: формальный или полужформальный статический невыполнимый анализ, который можно использовать для оценки проекта.

Анализ уязвимости проектирования может быть выполнен, если безопасность вызывает беспокойство. В формальном анализе проекта используются математические модели, которые позволяют разработчикам прогнозировать поведение и проверять производительность программного обеспечения вместо того, чтобы полностью полагаться на тестирование. Формальный анализ проекта может использоваться для обнаружения остаточных ошибок спецификации и проектирования.

- Моделирование и прототипирование: динамические методы оценки проектирования. Например, моделирование производительности или технико-экономические прототипы.

МЕРЫ ОЦЕНКИ АСПЕКТОВ ПРОЕКТИРОВАНИЯ ПО

- функциональная декомпозиция
- структурное проектирование

Функционально-ориентированные проектные меры:

Объектно-ориентированные меры проектирования:

- диаграммы классов
- оценка свойств внутреннего содержимого класса



Меры могут использоваться для оценки или количественной оценки различных аспектов проектирования программного обеспечения; например, размер, структура или качество. Большинство предложенных мер зависит от подхода, использованного при разработке проектного решения. Эти меры подразделяются на две большие категории.

- Функционально-ориентированные структурированные проектные меры. Это меры, полученные путем анализа функциональной декомпозиции. Они обычно представлены с использованием структурной диаграммы, называемой иерархической диаграммой, на которой могут быть вычислены различные меры.
- Объектно-ориентированные меры проектирования. Эта структура проекта обычно представлена в виде диаграммы классов, на которой могут быть вычислены различные меры. Также могут быть вычислены показатели свойств внутреннего содержимого каждого класса.

Выделенные меры позволяют качественно оценить проектное решение, дать конструктивные предложения на вносимые изменения. Любые меры в первую очередь направлены на то, чтобы довести решение до качественного продукта, удовлетворяющего требованиям конечного пользователя.

ОБОЗНАЧЕНИЯ РАЗРАБОТКИ ПО

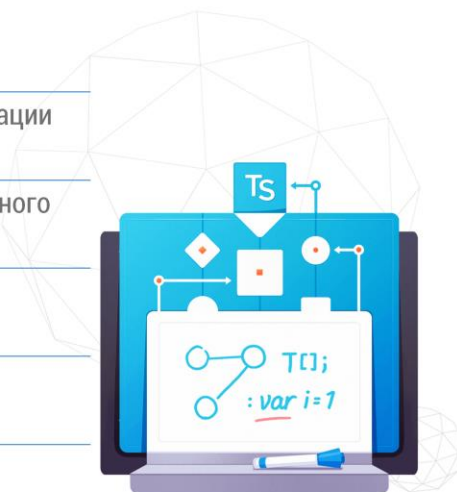
для описания структурной организации проекта

для описания поведения программного обеспечения

для описания архитектуры ПО

для представления проекта

для применения методов проектирования



Существует множество обозначений для представления артефактов проектирования программного обеспечения.

Некоторые используются для описания структурной организации проекта, другие – для описания поведения программного обеспечения. Некоторые обозначения используются в основном при архитектурном проектировании, а другие – в основном при детальном проектировании, хотя некоторые обозначения могут использоваться для обеих целей. Кроме того, некоторые обозначения используются в основном в контексте конкретных методов проектирования.

Обратите внимание, что при разработке программного обеспечения часто используются несколько обозначений. Здесь они подразделяются на нотации для описания структурного статического представления по сравнению с поведенческим динамическим представлением.

Структурные диаграммы отображают статическую структуру элементов, показывая сами элементы – классы, объекты, пакеты или модули, физические узлы, компоненты и интерфейсы. Они также показывают отношения между этими элементами.

Диаграммы поведения отображают динамическое поведение элементов в системе. Они показывают, как система ведет себя и взаимодействует с собой и

другими объектами: пользователями, другими системами. Они показывают, как данные перемещаются в системе, как объекты взаимодействуют друг с другом, как течение времени влияет на систему и так далее.

АСПЕКТЫ ПРОЕКТА ПО

Языки описания архитектуры
Диаграммы классов и объектов
Диаграммы компонентов
Карточки ответственности класса
Диаграммы развертывания
Диаграммы сущностей-отношений
Языки описания интерфейсов
Структурные диаграммы



Следующие обозначения представляют структурные аспекты проекта программного обеспечения, то есть они используются для описания основных компонентов и их взаимосвязей.

- Языки описания архитектуры: текстовые, часто формальные языки, используемые для описания архитектуры программного обеспечения в терминах компонентов и соединителей.
- Диаграммы классов и объектов: используются для представления набора классов и объектов и их взаимосвязей.
- Диаграммы компонентов: используются для представления набора компонентов – физическая и заменяемая части системы, которая соответствует и обеспечивает реализацию набора интерфейсов и их взаимосвязи.
- Карточки соучастников ответственности класса: используются для обозначения имен компонентов / классов, их обязанностей и имен их взаимодействующих компонентов.
- Диаграммы развертывания: используются для представления набора физических узлов и их взаимосвязей и, таким образом, для моделирования физических аспектов программного обеспечения.
- Диаграммы сущностей – отношений: используются для представления

концептуальных моделей данных, хранящихся в информационных репозиториях.

- Языки описания интерфейсов: языки, подобные программированию, используемые для определения интерфейсов программных компонентов.
- Структурные диаграммы: используются для описания структуры вызовов программ.

ОПИСАНИЯ ПОВЕДЕНИЯ

Диаграммы действий

Диаграммы связи

Диаграммы потоков данных

Таблицы решений и диаграммы

Блок-схемы

Диаграммы последовательности

Диаграммы перехода состояний

Формальные языки спецификации

Псевдокод и языки разработки программ



Следующие обозначения и языки используются для описания динамического поведения программных систем и компонентов. Описания поведения могут включать обоснование проектного решения, например, как проект будет соответствовать требованиям безопасности.

Диаграммы действий: используются для отображения потока управления от действия к действию. Могут использоваться для представления одновременных действий.

Диаграммы связи: используются для отображения взаимодействий, происходящих между группой объектов; акцент делается на объекты, их ссылки и сообщения, которыми они обмениваются по этим ссылкам.

Диаграммы потоков данных DFD : используются для отображения потока данных между элементами. Схема потока данных предоставляет описание, основанное на моделировании потока информации вокруг сети операционных элементов, при этом каждый элемент использует или изменяет информацию, поступающую в этот элемент.

Таблицы решений и диаграммы: используются для представления сложных комбинаций условий и действий.

Блок-схемы: используются для представления соответствующих действий, которые необходимо выполнить.

Диаграммы последовательности: используются для отображения взаимодействий между группой объектов с акцентом на временном порядке сообщений, передаваемых между объектами.

Диаграммы перехода состояний и диаграммы состояний: используются для отображения потока управления от состояния к состоянию и того, как поведение компонента изменяется в зависимости от его текущего состояния в конечном автомате.

Формальные языки спецификации: текстовые языки, использующие базовые математические понятия для строгого и абстрактного определения интерфейсов и поведения программных компонентов, часто в терминах предварительных и постусловий.

Псевдокод и языки разработки программ PDL : языки, подобные структурированному программированию, используемые для описания, как правило, на стадии детального проектирования поведения процедуры или метода.

СТРАТЕГИИ И МЕТОДЫ ПРОЕКТИРОВАНИЯ ПО

Функционально-ориентированный подход

- метод проектирования → определение основных функций
- программного обеспечения → разработка функций
- уточнение функций

Объектно-ориентированный подход

- метод проектирования → на основе объектов

Существуют различные общие стратегии, помогающие направлять процесс проектирования. В отличие от общих стратегий, методы более конкретны в том смысле, что они обычно предоставляют набор обозначений. Эти обозначения будут использоваться с методом, описанием процесса. Сам процесс будет использоваться при следовании методу. Здесь же будет обозначен набор руководящих указаний по использованию метода. Такие методы полезны в качестве общей основы для команд разработчиков программного обеспечения. Функционально-ориентированный структурированный подход к проектированию

Это один из классических методов проектирования программного обеспечения, при котором декомпозиция сосредотачивается на выявлении основных функций программного обеспечения. Затем на их разработке и уточнении в иерархической нисходящей манере. Структурированный дизайн обычно используется после структурированного анализа, таким образом создавая среди прочего диаграммы потоков данных и связанные описания процессов. Исследователи предложили различные стратегии и эвристики для преобразования DFD в архитектуру программного обеспечения, обычно представляемую как структурная диаграмма.

Объектно-ориентированный подход к проектированию

Было предложено множество методов проектирования программного обеспечения на основе объектов. Эта область эволюционировала от раннего объектно-ориентированного проектирования середины 1980-х годов, в котором наследование и полиморфизм играют ключевую роль. И завершилась областью проектирования на основе компонентов, где метаянформация может быть определена и доступна. Корни объектно-ориентированного проектирования уходят в концепцию абстракции данных. Проектирование, основанное на ответственности, было предложено в качестве альтернативного подхода к объектно-ориентированному проектированию.

СТРАТЕГИИ И МЕТОДЫ ПРОЕКТИРОВАНИЯ ПО

Подход проектирования → ориентирован на структуру данных

- акцент на структуре данных, а не на функциях

Компонентно-ориентированный подход

- решает вопросы предоставления, разработки и интеграции компонентов

Другие подходы:

- итеративные и адаптивные методы
- аспектно-ориентированное проектирование
- сервис-ориентированная архитектура



Подход проектирования, ориентированный на структуру данных
Проектирование, ориентированное на структуру данных, начинается со структур данных, которыми управляет программа, а не с функции, которую она выполняет. Инженер-программист сначала описывает структуры входных и выходных данных, а затем разрабатывает структуру управления программой на основе этих диаграмм структур данных. Были предложены различные эвристики для особых случаев – например, когда есть несоответствие между входными и выходными структурами.

Компонентно-ориентированный подход к проектированию

Программный компонент – это независимая единица, имеющая четко определенные интерфейсы и зависимости, которые можно составлять и развертывать независимо. Компонентное проектирование решает вопросы, связанные с предоставлением, разработкой и интеграцией таких компонентов для улучшения повторного использования. Повторно используемые и готовые программные компоненты должны соответствовать тем же требованиям безопасности, что и новое программное обеспечение. Доверительное управление – это проблема проектирования; компоненты, созданные как имеющие определенную степень надежности, не должны зависеть от менее надежных компонентов или служб.

Существуют и другие интересные подходы.

- Итеративные и адаптивные методы реализуют приращения программного обеспечения и снижают внимание к строгим требованиям и проектированию программного обеспечения.
- Аспектно-ориентированное проектирование – это метод, с помощью которого программное обеспечение конструируется с использованием аспектов для реализации сквозных задач и расширений. Все это выявляется в процессе требований к программному обеспечению.
- Сервис-ориентированная архитектура – это способ создания распределенного программного обеспечения с использованием веб-сервисов, выполняемых на распределенных компьютерах. Программные системы часто создаются с использованием служб от разных поставщиков, поскольку стандартные протоколы, такие как HTTP , HTTPS , SOAP , были разработаны для поддержки связи служб и обмена служебной информацией.

ИНСТРУМЕНТЫ ДЛЯ РАЗРАБОТКИ ПО

Инструменты проектирования - это «инструменты моделей для реализации программного обеспечения»

Действия инструментов проектирования:

преобразовать модель требований в проект

обеспечить представление функций и их интерфейсов в ПО

уточнить и разбить модели для представления

Рекомендовать методику оценки качества ПО



Инструменты проектирования программного обеспечения могут использоваться для поддержки создания артефактов проектирования программного обеспечения в процессе его разработки. Они могут частично или полностью поддерживать следующие действия:

- преобразовать модель требований в проектное представление;
- для обеспечения поддержки представления функциональных компонентов и их интерфейсов;
- реализовать эвристическое уточнение и разбиение;
- предоставить рекомендации по оценке качества.

Инструменты проектирования – это в основном инструменты построения программного обеспечения на основе моделей. Это помогает проверять модели системы или программного обеспечения. Например, инструмент может проверять согласованность объектов данных в базе данных и находить несоответствия и дефекты. Инструменты проектирования также можно использовать для проверки моделей состояния или объектных моделей. Инструменты проектирования обычно используются разработчиками и могут помочь в разработке программного обеспечения. Они позволяют находить и идентифицировать дефекты на ранней стадии, когда их проще и дешевле исправить.