Программная инженерия (промышленное программирование) ассоциируется с разработкой больших и сложных программ коллективами разработчиков. Становление и развитие этой области было вызвано рядом проблем:

1) Высокая стоимость программного обеспечения  
2) Сложность его создания  
3) Необходимость управления и прогнозирования процессов разработки

В конце 1960-х, в начале 1970-х произошло событие, которое вошло в историю как первый кризис программирования. Событие состояло в том, что стоимость программного обеспечения стало приближаться к стоимости железа, а динамика роста этой стоимости позволяла прогнозировать, что к середине 90-х годов всё человечество будет заниматься разработкой ПО.

С тех пор программная инженерия получила достаточно бурное развитие. Каждый этап развития связан с появлением или осознанием очередной проблемы и нахождением путей и способов решения этой проблемы.

Сам термин “Software engineering” впервые был озвучен в 68-м году в конференции по науке и технике в Германии.

**Предпосылки и история**Повторное использование кода (модульное программирование)

На первых этапах становления программной инженерии было отмечено, что высокая стоимость программ связано с разработкой одинаковых (или похожих) фрагментов кода в различных программах. Повторное использование при создании новых программ, ранее написанных фрагментов обещало существенное снижение сроков и стоимости разработки.

Главный принцип модульного программирования состоял в выделении таких фрагментов и оформлении их в виде отдельных модулей. Каждый модуль снабжался описанием, в котором устанавливались правила его использования – интерфейс модуля. Интерфейс задавал связи модуля с основной программой – связи по данным, связи по управлению. При этом возможность повторного использования модулей определялась количеством и сложностью этих связей. Наиболее простые оказались модули решения математических задач.

Для многих других типов модулей возможность их повторного использования оказалась проблематичной ввиду сложности их связи с основной программой. Повторное использование модулей со сложными интерфейсами является достаточно актуальной по сей день задачей.

**Рост сложности программ**

Следующий этап возрастания стоимости ПО был связан с переходом от разработки относительно простых программ к разработке программных комплексов. Следует отметить, что этот переход был вызван появлением вычислительной техники 3-го поколения. С переходом на использование интегральных схем производительность компьютеров возросла на порядки, что и создало предпосылки на решение разных сложных задач

К сложным задачам относятся:

• Система управления космическими объектами

• Управление оборонным комплексом

• Автоматизация технологических процессов на производстве

• Автоматизация финансовых учреждений

• И так далее…

Сложность можно оценивать:

1. Большой объём кода

2. Большое количество связей между элементами кода

3. Большое количество разработчиков

4. Большое количество пользователей

5. Длительное время использования

Для таких сложных задач оказалось, что основная часть их стоимости приходится не на создание программ, а на их внедрение и эксплуатацию.

По аналогии с промышленной технологией стали говорить о жизненном цикле программного продукта, как о последовательности определённых этапов:

1. Эта проектирования

2. Этап разработки

3. Этап тестирования

4. Этап сопровождения

Основные принципы и технологии структурного проектирования и кодирования.

1. Нисходящее функциональное проектирование

2. Применение специальных языков проектирования и средств автоматизации использования этих языков

3. Дисциплина проектирования и разработки

Модификация программы

Следующая проблема роста стоимости программ была связана с тем, что изменения требований к программе стали возникать не только на этапе сопровождения, но и на этапе проектирования.

Создание программного продукта превратилось в его перманентное проектирование. Возник вопрос: «Как проектировать и писать программы, чтобы обеспечить возможность внесения изменений в программу, не меняя ранее написанного кода?».

Объектно-ориентированное программирование (ООП)

ДОПИСАТЬ (В ТГ)

Итоги

Программная инженерия (технология промышленного программирования) как направление возникло и формировалось под давлением роста стоимости создаваемого программного обеспечения. Главная цель этой области знаний – сокращение стоимости и сроков разработки программы.

Программная инженерия прошла несколько этапов развития, в процессе которых были сформулированы фундаментальные принципы и методы разработки программных продуктов. Основной принцип программной инженерии состоит в том, что программы создаются в результате выполнения нескольких взаимосвязанных этапов (анализ требований, проектирование, разработка, тестирование), составляющих жизненный цикл программного продукта.

**Программное обеспечение** определяется как набор компьютерных программ, процедур и связанных с ними документаций и данных. Взгляд на ПО как только на программу, находящуюся в компьютере, слишком узок. Дело в том, что продаётся не только программа, но и документация. Поэтому ПО иногда называют программным продуктом.

Программный продукт – это не только программы, а также вся связанная с ними документация и данные, необходимые для корректной работы программы. В зависимости от того, для кого разрабатываются программные продукты, бывают двух типов:

* Коробочные продукты
* Заказные продукты

Программная инженерия – инженерная дисциплина, которая связана со всеми аспектами производства ПО от начальных стадий создания спецификации до интеграции и поддержки

**Инженерная дисциплина**

**Инженеры** – те специалисты, которые выполняют практическую работу и добиваются практических результатов

Для решения задачи инженеры применяют теории, методы и средства. В этом случае инженер ищет средства или метод для решения задачи, применяет его и несёт ответственность за результат

Набор таких инженерных методов и способов, теоретически возможно необоснованных, но получивших неоднократное подтверждение на практике играет большую роль. В ПИ они получили название лучших практик

Инженеры работают в условиях ограниченных ресурсов (временных, финансовых). Продукт должен быть создан в установленные сроки в рамках выделенных средств, оборудования и людей

ПИ занимается не только техническими вопросами производства ПО, но и управлением программными проектами, вопросы планирования, финансирования, надёжности, управления коллективом. Программные инженеры применяют систематичные и организованные подходы к работе для достижения максимальной эффективности и качества ПО, их задача состоит в адаптации существующих методов и подходов к решению своей конкретной проблемы

Отличие информатики от программной инженерии:

Информатика (computer science) занимается теорией и методами вычислительных и программных систем, в то время как ПИ занимается практическими проблемами создания ПО. Информатика составляет теоретические основы ПИ и инженер по ПО должен знать информатику. Программные инженеры зачастую используют приёмы, которые применимы только в конкретных условиях и не могут быть обобщены, а теории информатики не всегда могут быть применены к большим системам.

Информатика – не единственный теоретический фундамент ПИ, это ещё управление финансами, организация работ в коллективе, взаимодействие с заказчиком и т.д.

Отличие ПИ от других инженерий

**24.10.24**

1 - почему доля провальных проектов в ПИ так велика в сравнении с другими инженериями?

2 - Можно ли в ПИ применить опыт других инженерий?

Жизненный цикл продукта любой инженерии в упрощённом виде включает фазы: проектирование создание образца, испытание, производство и эксплуатация

Компьютерная программа – нематериальный объект. Отсюда следует следующее отличие: фаза производства состоит в копировании образца на другие носители, стоимость фазы мала. Если кодирование считать элементом проектирования, то отсутствует также и фаза создания образца.

1. Стоимость программы – в основном стоимость её проектирования
2. Стоимость проектирования коробочных продуктов размазывается по копиям
3. Стоимость заказных продуктов остаётся высокой

Второе существенное отличие состоит в том, что программа – искусственный объект. Для программы нет объективных законов.

Программная инженерия – молодая дисциплина, опыт которой составляет несколько десятков лет. ПИ иногда сравнивают с ранней строительной инженерией. Несмотря на молодой возраст ПИ уже накопила определенный опыт, который позволяет делать удачные проекты. Этот опыт выражен в основных принципах.

**Из чего складывается стоимость ПО**

Структура стоимости ПО существенно зависит от типа ПО, применяемых методов его разработки. Многие отмечают высокую долю стоимости этапа сопровождения (60% и более). Этап сопровождения включает выполнение двух видов работ: внесение изменений в программу, исправление ошибок в программе (несоответствие первоначальным требованиям).

Типовое распределение стоимости между основными этапами:

1. 15% - спецификация – формулировка требований и условий разработки
2. 25% - проектирование – разработка и верификация проекта
3. 20% - разработка – кодирование и тестирование
4. 40% - интеграция и тестирование – объединение и сборочное тестирование продукта

Для коробочного ПО характерна более высокая доля тестирования, но за счёт сокращения доли спецификации.

Распределение стоимости заказного ПО зависит от его сложности. При сложном ПО возрастает доля интеграции и тестирования. Сокращение доли проектирования и разработки достигается за счёт применения опробованных проектных решений и повторного использования элементов.

**Методы программной инженерии**

Методы программной инженерии - структурный подход к созданию ПО, который способствует производству высококачественного продукта и эффективном в экономическом смысле способом.

Метод программной индустрии основан на идее создания моделей ПО, поэтапном преобразовании этих моделей в программу. На этапе спецификации создаётся модель – описание требований, которые далее преобразуются в модель проекта ПО.

Методы должны включать в себя следующие компоненты:

1. Описание моделей системы и нотация
2. Правила и ограничения
3. Рекомендации
4. Руководство по применению метода

Нет идеальных методов. Все они применимы только для тех или иных случаев. Выбор метода – задача специалиста по ПИ

Что такое case?

**07.11.24**

**Свойства хорошей программы**

Хорошая программа должна делать то, что ожидает от неё заказчик, т.е. удовлетворять требования заказчика. Такие требования называют функциональными. Кроме функциональных требований существует ряд общих характеристик (нефункциональные требования):

* Сопровождаемость – означает, что программа должна быть написана с расчётом на дальнейшее развитие. Это критическое свойство системы, т.к. изменение ПО неизбежно вследствие системы бизнеса. Сопровождение программы часто выполняют не те люди, которые её разрабатывают. Сопровождаемость включает такие элементы, как наличие и понятность проектной документации. Соответствие проектной документации исходному коду, понятность исходного кода и простота изменения исходного кода и добавление новых функций
* Надёжность:
  + Отказоустойчивость – возможность восстановления программы и данных в случае сбоя в работе.
  + Безопасность – сбои в работе в программе не должны приводить к опасным последствиям. Защищённость от случайных и преднамеренных внешних воздействий.
  + Эффективность – программное обеспечение не должно впустую тратить системные ресурсы.
  + Удобство использования – именно тем типом пользователей, на которых рассчитано приложение

Реализация нефункциональных требований требует больше затрат, чем функциональных.

Сопровождаемость требует значительных усилий по поддержанию соответствия проекта исходному коду и применение специальных методов создания модифицируемых программ. Надёжность – дополнительные средства восстановления системы после сбоя. Эффективность поиска специальных архитектурных решений и оптимизации кода. А удобство – проектирование не интуиктивно понятного интерфейса, а профессионального

**Профессиональные и этические требования**

Профессиональные обязательства:

1. Конфиденциальность – программные специалисты должны уважать и соблюдать конфиденциальность в отношении работодателей или заказчиков.
2. Компетентность – программный специалист не должен завышать свой истинный уровень компетенции.
3. Защита интеллектуальной собственности – специалист должен соблюдать законодательство и принципы защиты интеллектуальной собственности при использовании чужой интеллектуальной собственности.
4. Злоупотребление компьютером

**Стандарты программной инженерии**

Для заключения контракта заказчик должен быть уверен, что разработчик справится и не завалит проект. В мировой практике промышленного производства гарантии успеха являются стандарты производства продуктов и услуг и сертификация производителей на соответствие этим стандартам. Процесс стандартизации и сертификации давно вошёл в программную инженерию (это основа промышленного производства программных продуктов).

Какие бывают стандарты?

**21.11.24**

**Стандарты**

1. Корпоративные
2. Отраслевые
3. Государственные
4. Международные

**Жизненный цикл программного продукта**

**Начало стандартизации жизненного цикла продукта**

Методологическую основу любой инженерии составляет понятие жизненного цикла (ЖЦ) изделия как совокупности всех действий, которые надо выполнить на протяжении всей жизни изделия

Смысл ЖЦ состоит во взаимосвязанности всех этих действий. ЖЦ промышленного изделия определяется как последовательность этапов, состоящих из технологических процессов. К этим этапам относятся:

* проектирование
* изготовление образца
* организация производства серийное производство
* эксплуатация
* ремонт
* вывод из эксплуатации

При разработке стандартов ЖЦ и их практическом применении сталкиваются с рядом проблем:

* Внедрение стандартов требует вложение значительных средств
* Неясность. Все ли требуемые процессы нужно выполнять и в какой мере
* Различные типы ПО (информационные системы, системы реального времени, бизнес-системы)
* Высокая динамика отрасли и устаревание стандартов
* Терминологическая неоднозначность различных государственных и корпоративных стандартов
* Во многих случаях применение стандартов было вызвано только требованием заказчика, хотя на практике часто тормозило выполнение проектов

**Модель жизненного цикла программного продукта**

Известны некоторые типовые модели ЖЦ ПО, которые проявили себя в определенных условиях, имеют определённые имущества, недостатки и условия применимости. Эти типовые модели устанавливают некоторые принципы организации модели ЖЦ ПО. К числу основных моделей ЦЖ ПО следует отнести каскадную и спиральную модели. На практике часто используют итерационную, V-образную, инкрементную и модель быстрого прототипирования

**Каскадная (водопадная) модель:**

1. Исследование
2. Требования
3. Проектирование
4. Реализация
5. Интеграция
6. Эксплуатация
7. Сопровождение

**Основными принципами каскадной модели являются:**

1. Строго последовательное выполнение фаз
2. Каждая последующая фаза начинается лишь тогда, когда, когда завершилась предыдущая
3. Каждая фаза имеет определённые критерии входа и выхода
4. Каждая фаза строго документируется
5. Переход от одной фазы к другой осуществляется по средствам обзора с участием заказчика
6. Основа модели – сформулированные требования, которые меняться не должны
7. Критерии качества результата – соответствие продукта установленным требованиям

**Каскадная модель имеет следующие преимущества:**

1. Проста и понятна заказчикам, т.к. часто используется другими организациями
2. Простота и удобство применения
   1. Процесс разработки выполняется поэтапно
   2. Её структурой может пользоваться даже слабо подготовленный или неопытный персонал
   3. Она способствует выполнению строгого контроля менеджмента проекта
3. Каждую стадию могут выполнять независимые команды
4. Позволяет достаточно точно планировать строки и затраты

При использовании каскадной модели для неподходящего проекта могут проявляться следующие недостатки:

1. Попытка вернуться на одну или две фазы назад
2. Приведёт к значительному увеличению затрат
3. Интеграция компонентов, на которых выявляется большинство ошибок, выполняется в конце разработки
4. Запаздывание с получением результатов – если в процессе выполнения требования изменились, то получится устаревший результат

Недостатки каскадной модели особо остро проявляются в случае, когда трудно, а бывает и невозможно, сформулировать требования

Каскадная модель впервые чётко сформулирована в 1970 году Ройсом. На начальном периоде она сыграла ведущую роль как метод регулярной разработки сложного ПО

**05.12.24**

**Подходит для следующих задач:**

1. Требования и их реализация максимально чётко определены и понятны:
   * Научно-вычислительный тип задач
   * Операционные системы и компиляторы
   * Системы управления объектами
2. Повторная разработка типового продукта
3. Выпуск новой версии уже существующего проекта
4. Каскадная модель находит применение как элемент других типов

**Спиральная модель жизненного цикла ПО** (посмотреть дома)

**Итерационная модель** (посмотреть дома)

**Инкрементальная модель** (посмотреть дома)

**Управление программным проектом**

**Характеристики проекта**

1. **Цель проекта** – наличие чётко выраженного конечного результата, определяемого в терминах затрат, качества и времени реализации
2. **Уникальность** – проект – разовое начинание, которое не будет повторяться
3. **Ограниченность во времени** – проект имеет начало и конец
4. **Ограниченность ресурсов** – ресурсы, которые мы выделяем на выполнение проекта
5. **Сложность** – для достижения целей проекта необходимо решить множество задач
6. **Неопределённость** – возможность достижения цели в указанные сроки с выделенными ресурсами заранее не гарантировано
7. **Предсказуемость** – по мере реализации проекта меняется потребность в тех или иных ресурсах

**Проект –** достаточно сложный вид деятельности, которым сложно управлять в силу его уникальности и ограниченности во времени

Пирамида (время, деньги, качество)

**Ролевая модель команды:**

1. **Менеджер проекта –** главное действующее лицо, обладающее знаниями и навыками, необходимыми для успешного управлением проекта
2. **Проектировщик –** функция проектирования архитектуры высокого уровня и контроля её выполнения (анализ требований, разработка архитектуры, участие в планировании проекта, контроль выполнения проекта, участие в отборе кадров)
3. **Разработчик –** роль, ответственная за непосредственное создание конечного продукта (помимо программирования в функции разработчика входит: контроль архитектурных и технических спецификаций продукта, подбор технологических инструментов из стандартов, диагностика и разрешение всех технических проблем, подбор инструментов разработки, метрик и стандартов, контроль за документацией, тестированием и другими этапами, мониторинг состояния продукта)
4. **Тестировщик –** роль, ответственная за удовлетворение требований к продукту (составление плана тестирования, контроль выполнения плана, разработка теста, автоматизация тестирования, выбор инструментов, метрик стандартов для тестирования, организация бета-тестирования
5. **Инженер по качеству –** 3 уровня качества:
   * **Качество конечного продукта –** обеспечивается тестированием
   * **Качество процесса разработки**
   * **Качество организации**

В некоторых случаях функции инженера по качеству возлагаются на *тестировщика*

Отличие от тестировщика:

* + Составление плана качества, когда план тестирования
  + Описание процесса – все процессы должны быть оцифрованы, описание процессов являются их формализацией (метрики и т.д.)
  + Оценка процессов – включают регистрацию хода выполнения процессов и оценку метрик
  + Улучшение процессов – переопределение процессов, автоматизация части работ, обучение персонала

1. **Технический писатель** – разработчик документации (разработка плана документирования, выбор и разработка стандартов и шаблонов подготовки документов, выбор средств автоматизации документирования, разработка документации, организация тестирования документации
2. **Технолог разработки** ПО – обеспечивает выполнение: поддержка модели жизненного цикла ПО, создание и сопровождение среды сборки продукта, создание и сопровождение процедуры установки так, чтобы каждая сборка устанавливалась автоматически с учётом версии и конфигурации, управление исходными текстами

**Управление продуктом (Product Management)**

**Цель -** удовлетворённые заказчики

**Компетенции** – бизнес-отдача, интересы заказчика, планирование продукта, маркетинг

**Функции –** выступает в роли представителя заказчика, формирует общее видение и рамки проекта. Организует работу с требованиями заказчика, развивает сферы применения в бизнесе, формирует ожидания заказчика, определяет компромиссы, возможности, время, ресурсы. Организует маркетинг, пиар, разрабатывает, поддерживает, исполняет план документации

**Управление программой (Program Management)**

**Цель –** достижение результата в рамках проектных ограничений

**Компетенции –** управление проектом, выработка архитектурных решений, контроль производственного процесса, административные службы

**Функции –** управляет процессом разработки с целью получения готового продукта в отведённые сроки, формирует спецификацию продукта и разрабатывает его архитектуру. Регулирует взаимоотношения и коммуникацию внутри проектной группы, следит за временным графиком проекта и готовит отчётность о его состоянии. Проводит в жизнь важные компромиссные решения. Разрабатывает, поддерживает и исполняет сводный план и календарный график проекта. Организует управление рисками

**Разработка (development)**