**1 глава - понятие программной инженерии**

***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 60-х/начале 70-х годов произошло событие, которое вошло в историю как ***первый кризис программирования***. Событие состояло в том, что стоимость программного обеспечения стала 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 1968 году на 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 связана с разработкой одинаковых и похожих фрагментов кода в различных программах. Повторное использование при создании новых программ ранее написанных фрагментов кода 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 и связи по управлению. При этом возможность повторного использования модулей определялась количеством и сложностью этих связей. Наиболее простыми 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 и по сей день задачей.

## Рост сложности программ.

Следующий этап возрастания стоимости ПО был связан с переходом от разработки относительно простых программ к разработке сложных 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 для решения различных 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 принципе и технологии структурного проектирования и кодирования.

1. Нисходящее функциональное проектирование
2. Применение специальных языков проектирования и средств автоматизация 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.

## Продолжение кризиса программирования

Несмотря на то, что программная инженерия достигла определённых успехов, **перманентный кризис программирования продолжается**. Связано это с тем, что на рубеже 80-х – 90-х годов началась **информационно-техническая революция**, вызванная взрывным ростом использования информационных средств.

* **Успешный проект** – вовремя и в рамках бюджета был выполнен весь намеченный фронт работы.
* **Проблемный проект** – нарушение сроков, перерасход бюджета или реализация не полноценная
* **Провальный проект** – большой перерасход бюджета

## Понятие программной инженерии

На сегодняшний день нет единого определения понятия «***программная инженерия*».**

Существует несколько трактовок:

1. Установление и использование обоснованных инженерных принципов для экономного получения ПО, которое надёжно и работает на реальных машинах
2. Та форма инженерии, которая применяет принципы информатики ***(computer science)*** и математики для рентабельного ращения проблем через ПО
3. Применении систематического, дисциплинированного и измеряемого подхода к разработке, использованию и сопровождению.
4. Дисциплина, целью которой является создание качественного ПО, создание которого завершается вовремя, не превышает выделенных бюджетных средств и удовлетворяет выдвигаемым требованиям.

## Программное обеспечение и программный продукт

**Программное обеспечение** – это набор компьютерных программ, процедур и связанной с ними документации 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, но также и вся связанная с ними документация и данные, необходимые для корректной работы программы. В зависимости от того, для кого разрабатываются программные продукты, продукты бывают двух типов:

1. Коробочные
2. Заказные

**Программная инженерия** – это инженерная дисциплина, которая связана со всеми аспектами производства ПО, от начальной стадии создания и спецификации до интеграции и поддержки.

## Инженерная дисциплина

**Инженеры** – это те специалисты, которые выполняют практическую работу и добиваются практических результатов. Для решения задачи инженеры применяют теории, методы и средства. В этом случае инженер ищет метод или средство для решения задачи, применяет его и несёт ответственность за результат. Набор таких инженерных методов и способов, теоретически, возможно, не обоснованных, но получивших неоднократное подтверждение на практике, играют большую роль. В программной инженерии они получили название **лучшие практик** (***best practics***).

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

**Программная инженерия** 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 программная инженерия занимается **практическими проблемами** создания ПО. Информатика составляет **теоретические основы** программной инженерии, и инженер по программному обеспечению **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 **реальным** большим системам.

**Информатика** - это **не единственный** теоретический фундамент программной инженерии, **это ещё** управление финансами, организация работ в коллективе, взаимодействие с заказчиком и так далее.

## Кейс (Case)

## Свойства хорошей программы

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

1. Сопровождаемость (*maintain ability*) – это значит, что программа должна быть написана с расчётом на **дальнейшее развитие**. Это **критическое свойство** системы, т.к. изменения ПО **неизбежны** в следствии изменения 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, **простота** его изменения и **добавление** новых функций.
2. Надёжность (*dependability*) – это понятие, включающее в себя:
   1. Отказоустойчивость – это возможность **восстановления** программы и данных в случае **сбоя в работе**.
   2. Безопасность – сбои в работе программы **не должны** приводить к опасным последствиям. **Защищённость** от случайных и непреднамеренных **внешних воздействий**.
   3. Эффективность – программное обеспечение **не должно** впустую тратить **системные ресурсы** (память, процессорное время, каналы связи и так далее).
   4. Удобство использования (*usability*) – возможность **комфортного взаимодействия** с программой **именно тем типом** пользователей, на которых **рассчитано приложение**.
3. Реализация **нефункциональных** требований требует **больше** затрат, чем **функциональных.**

Сопровождаемость требует значительных усилий по поддержанию соответствия проекта исходному коду и применения специальных методов создания модифируемых программ. Надежность дополнительных средств восстановления данных после сбоя. Эффективность поиска специальных архитектурных решений и оптимизации кода. А удобство – это проектирования не интуитивно понятного интерфейса, а профессионального.

## Профессиональные и этические требования

Профессиональные обязательства:

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

## Стандарты программной инженерии

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

## Какие бывают 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 жизненного цикла состоит во взаимосвязанности всех этих действий. Жизненный цикл промышленного изделия определяется как последовательность этапов, состоящих из технологических процессов. К этим этапам относятся:

1. Проектирование
2. Изготовление образца
3. Организация производства
4. Серийное производство
5. Эксплуатация
6. Ремонт
7. Вывод из эксплуатации

Разработка стандартов жизненного цикла и их практическое применение сталкиваются с рядом проблем:

1. Внедрение стандартов требует вложения значительных средств.
2. Неясность, все ли требуемые процессы нужно выполнять и в какой мере.
3. Различные типы ПО (информационные системы, системы реального времени, бизнес-системы).
4. Высокая динамика отрасли и устаревание стандартов.
5. Терминологическая неоднозначность различных государственных и корпоративных стандартов.
6. Во многих случаях применение стандартов было вызвано только требованием заказчика.

## Модель жизненного цикла программного продукта

Известны некоторые типовые модели жизненного цикла программного обеспечения, которые проявили себя в определённых условиях, имеют определённые преимущества, недостатки и условия применимости. Эти типовые модели устанавливают некоторые принципы организации модели жизненного цикла ПО. К числу основных моделей жизненного цикла 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. Выработка требований
3. Проектирование
4. Реализация
5. Интеграция
6. Эксплуатация
7. Сопровождение

Основными принципами каскадной модели являются:

1. Строго-последовательное выполнение фаз.
   1. Каждая последующая фаза начинается после полного выполнения предыдущей фазы.
   2. Каждая фаза имеет входные и выходные данные.
   3. Каждая фаза полностью документируется.
   4. Переход от одной фазы к другой осуществляется посредством обзора с участием заказчика.
2. Основа модели, сформулированные требования к которой не должны изменяться.
3. Критерии качества результата – соответствие продукта установленным требованиям.

Каскадная модель имеет следующие преимущества:

1. Проста и понятна заказчиком, т.к. часто используется другими организациями.
2. Простота и удобство в применении.
   1. Процесс разработки выполняется поэтапно.
   2. Её структурой может руководствоваться даже слабо подготовленный или неопытный персонал.
   3. Она способствует осуществлению строгого контроля менеджмента проекта.
3. Каждую стадию могут выполнять независимые команды.
4. Позволяет достаточно точно планировать сроки 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-компонент, на которой обычно выявляется большая часть ошибок, выполняется в конце разработки, что сильно увеличивает стоимость решения ошибок.
4. Запаздывание с получением результата – если в процессе выполнения проекта требования изменились, то получится устаревший результат.
5. Недостатки каскадной модели особо остро проявляются в случае, когда трудно или невозможно сформулировать требования.

Каскадная модель впервые чётко сформулирована в 1970 году неким учёным Ройсом. На начальном этапе оно сыграло чёткую роль как метод разработки ПО.

Подходит для следующих типов задач: