Программная инженерия: особенности, история становления, основные определения и задачи

# Что такое инженерия и назначение инженера

Инженер – это специалист, осуществляющий инженерную деятельность.

Инженерная деятельность – область технической деятельности, включающая в себя ряд специализированных областей и дисциплин.

Конечной целью инженерной деятельности являются изобретение, разработка, создание, внедрение, обслуживание и/или улучшение техники, материалов или процессов.

В 1712 году была основана первая инженерная школа.

Современный этап развития инженерной деятельности характеризуется системным подходом к решению сложных научно-технических задач, обращением ко всему комплексу социальных, гуманитарных, естественных и технических дисциплин.

Проектирование как особый вид инженерной деятельности формируется в начале 20-го столетия и связано с первоначальной деятельностью чертёжников, необходимостью особого (точного и понятного) графического изображения замысла инженера для передачи исполнителям на производстве.

Различают «внешнее» и «внутреннее» проектирование.

Первое направлено на проработку общей идеи системы, её исследование с помощью теоретических средств, созданных и имеющихся в соответствующей технической науке.

Второе связано с созданием рабочих чертежей (технических и рабочих проектов). Они служат основным документом для изготовления технической системы на производстве.

Проектирование следует отличать от конструирования.

Результат конструкторской деятельности должен быть обязательно материализован в виде опытного образца. С его помощью уточняются расчёты, приводимые в проекте, и конструктивно-технические характеристики проектируемой технической системы.

Характерной чертой развития инженерной деятельности является специализация различных её видов. Это привело к необходимости её теоретического описания в целях обучения и передачи опыта, а также для осуществления автоматизации самого процесса проектирования и конструирования технических систем. В частности, сформировалась концепция CALS (Continuous Acquisition and Life cycle Support), суть которой состоит в применении принципов и технологий информационной поддержки на всех этапах производства изделия.

# Особенности системотехнической деятельности

Главной задачей инженера-системотехника является интеграция инженерной деятельности по отраслям и видам. Однако, сама системотехническая деятельность является неоднородной и включает в себя различные виды инженерных разработок, реализуется различными группами специалистов, занимающихся разработкой отдельных подсистем.

Расчленение сложной технической системы на подсистемы идёт по различным признакам в соответствии со специализацией в технических науках, сложившимися организационными подразделениями, последовательностью этапов работы. Поэтому координация всех аспектов этой деятельности оказывается нетривиальной научной, инженерной и организационной задачей. Для её выполнения требуется группа особых специалистов – координаторов.

# Социальная значимость результата инженерной деятельности

В результате «расслоения» инженерной деятельности, отдельный инженер концентрирует свое внимание лишь на части сложной технической системы, а не на целом. При этом инженер все более и более удаляется от непосредственного потребителя его изделия, конструируя техническую систему отделенной от конкретного человека. Непосредственная связь изготовителя и потребителя, характерная для ремесленной технической деятельности, нарушается.

Это очень хорошо понимал еще в начале ХХ столетия русский инженер-механик и философ-техники П. К. Энгельмейер: "Прошло то время, когда вся деятельность инженера протекала внутри мастерских и требовала от него одних только чистых технических познаний. Начать с того, что уже сами предприятия, расширяясь, требуют от руководителя и организатора, чтобы он был не только техником, но и юристом, и экономистом, и социологом".

В конечном счёте в инженерии сформировалось методология жизненного цикла изделия как совокупности всех существенных этапов «жизни» продукции.



# Сущность и логика становления программной инженерии

Программная инженерия связана с программированием.

На первых порах программисты в индивидуальном порядке решали небольшие научно-технические задачи на конкретной ЭВМ в числовом коде, а затем в автокоде или Ассемблере.

Понемногу задачи становились всё более сложными. Появились библиотеки алгоритмов, сложные языки программирования и методики тестирования. Эти задачи могли быть решены только коллективами программистов с чёткой организацией работы и разделением функций. Даже понимание заказчика и согласование с ним его требований требовало специальных знаний. То есть понадобились специальные системные подходы.

Ну а затем информатика «пошла в народ». Создание программ стало требовать системного подхода, и как ответ на это требование сформировалась отрасль знаний под названием «программная инженерия». С самого начала у неё была одна особенность: международный характер информационных обменов обеспечил международный же характер разработок при создании стандартов, областей знаний и методик, составляющих ядро программной инженерии.

Проект ПО – совокупность спецификаций ПО (включающих модели и проектную документацию), обеспечивающих создание ПО в конкретной программно-технической среде).

ПО можно разбить на два класса: «малое» и «большое».

«Малое» (простое) программное обеспечение имеет следующие характеристики:

* Решает одну несложную, четко поставленную задачу;
* Размер исходного кода не превышает нескольких сотен строк;
* Скорость работы программного обеспечения и необходимые ему ресурсы не играют большой роли;
* Ущерб от неправильной работы не имеет большого значения;
* Модернизация программного обеспечения, дополнение его возможностей требуется редко;
* Разрабатывается одним программистом или небольшой группой;
* Подробная документация не требуется, ее может заменить исходный код, который доступен.

«Большое» (сложное) программное обеспечение имеет 2-3 или более характеристик из следующего перечня:

* Решает совокупность взаимосвязанных задач;
* Использование приносит значимую выгоду;
* Удобство его использования играет важную роль;
* Обязательно наличие полной и понятной документации;
* Низкая скорость работы приводит к потерям;
* Сбои, неправильная работа, наносит ощутимый ущерб;
* Программы в составе ПО во время работы взаимодействует с другими программами и программно-аппаратными комплексами;
* Работает на разных платформах;
* Требуется развитие, исправление ошибок, добавление новых возможностей;
* Группа разработчиков состоит из более 5 человек.

Классификация программных проектов по созданию сложного ПО может быть проведена по размеру группы разработчиков и длительности проекта:

* Небольшие проекты – проектная команда менее 10 человек, срок от 3 до 6 месяцев;
* Средние проекты – проектная команда от 20 до 30 человек, протяженность проекта 1-2 года;
* Крупномасштабные проекты – проектная команда от 100 до 300 человек, протяженность проекта 3-5 лет;
* Гигантские проекты – армия разработчиков от 1000 до 2000 человек и более (включая консультантов и соисполнителей), протяженность проекта от 7 до 10 лет.

# Предпосылки возникновения программной инженерии

В конце 60-х – начале 70-х годов прошлого века произошло событие, которое вошло в историю как первый кризис программирования. Событие состояло в том, что стоимость программного обеспечения стала приближаться к стоимости аппаратуры («железа»).

Тогда и заговорили о программной инженерии как о некоторой дисциплине, целью которой является сокращение стоимости программ. К настоящему времени программная инженерия накопила значительный багаж методологий, методик, стандартов и рекомендаций.

Основные причины неудач при разработке ПО:

* Нечеткая и неполная формулировка требований;
* Недостаточное вовлечение пользователей в работу над проектом;
* Отсутствие необходимых ресурсов;
* Неудовлетворительное планирование и отсутствие грамотного управления проектом;
* Частое изменение требований и спецификаций;
* Новизна и несовершенство используемой технологии;
* Недостаточная поддержка со стороны высшего руководства;
* Недостаточно высокая квалификация разработчиков, отсутствие необходимого опыта.

Одним из факторов неудачного создания ПО является плохое планирование, в результате устанавливаются невыполнимые сроки, закладываются недостаточные ресурсы. При этом основной причиной неверного планирования является заблуждение относительно производительности проектировщиков. В большом проекте общая производительность группы разработчиков не равна сумме производительностей отдельных членов группы.

Отметим особенности современных проектов ПО:

* Сложность;
* Отсутствие полных аналогов и высокая доля вновь разрабатываемого ПО;
* Наличие унаследованного ПО и необходимость его интеграции с разрабатываемым ПО;
* Территориально распределенная и неоднородная среда функционирования;
* Большое количество участников проектирования, разобщенность и разнородность отдельных групп разработчиков по уровню квалификации и опыту.

Разработка ПО имеет следующие специфические особенности:

* Неформальный характер требований к ПО и формализованный основной объект разработки – программы;
* Творческий характер разработки;
* Дуализм ПО (с одной стороны, ПО - статический объект – совокупность текстов, с другой стороны, – динамический, поскольку при эксплуатации порождаются процессы обработки данных);
* При своем использовании (эксплуатации) ПО не расходуется и не изнашивается, но морально устаревает;
* «Неощутимость», «воздушность», «квазинематериальность» ПО, что подталкивает к безответственному переделыванию, поскольку легко стереть и переписать, чего не сделаешь при проектировании зданий и аппаратуры.

Ответом на кризис в разработке ПО стало создание программной инженерии (software engineering) как специальной дисциплины или области знаний.

Инженерия ПО (software engineering) – совокупность инженерных методов и средств создания ПО.

Фундаментальная идея программной инженерии: проектирование ПО является формальным процессом, который можно изучать и совершенствовать.

Основными целями программной инженерии являются:

* Системы должны создаваться в короткие сроки и соответствовать требованиям заказчика на момент внедрения.
* Качество ПО должно быть высоким.
* Разработка ПО должна быть осуществлена в рамках выделенного бюджета.
* Системы должны работать на оборудовании заказчика, а также взаимодействовать с имеющимся ПО.
* Системы должны быть легко сопровождаемыми и масштабируемыми.

# Этапы развития программной инженерии

## Эволюция технологий создания программного обеспечения

Основные этапы становления технологии:

* 70-е и 80-е годы XX века– систематизация и стандартизация процессов создания ПО (структурный подход);
* 90-е годы- начало 21-го века – переход к сборочному, индустриальному способу создания ПО (объектно-ориентированный подход);
* с середины 90-х годов до настоящего времени – развитие компонентного подхода и сетевых технологий, создание CASE-технологий проектирования ПО

В рамках этих этапов развитие программной инженерии происходило и происходит по многим направлениям (нитям), решая проблемы, возникающие при разработке программного обеспечения в связи с развитием вычислительной техники и возникающими новыми и всё более усложняющими задачами.

В настоящее время выделяют несколько методологий (парадигм/основных подходов/моделей) в программировании:

* Императивное программирование (императивная парадигма)
* Декларативное программирование (декларативная парадигма)
* Функциональное программирование
* Логическое программирование
* Структурное программирование
* Модульное программирование
* Функциональное программирование
* Логическое программирование
* Объектно-ориентированное программирование
* Программирование, основанное на классах
* Программирование, основанное на прототипах
* Субъектно-ориентированное программирование

Императивное программирование – это такой стиль написания исходного кода компьютерной программы, для которого характерно следующее:

* В исходном коде программы записываются инструкции (команды);
* Инструкции выполняются последовательно;
* При выполнении инструкции данные, полученные при выполнении предыдущих инструкций, могут читаться из памяти;
* Данные, полученные при выполнении инструкций, могут записываться в память

Основные черты императивных языков:

* использование именованных переменных;
* использование оператора присваивания;
* использование составных выражений;
* использование подпрограмм;
* и др.

Рассмотрим развитие технологий программирования в рамках императивной парадигмы и ряд фундаментальных проблем разработки программ и найденных базовых методов их решения.

1. Первый этап – «стихийное» программирование (от появления первых вычислительных машин до середины 60-х годов XX в). Первые программы имели простейшую структуру. Они состояли из собственно программы на машинном языке и обрабатываемых ею данных. Сложность программ в машинных кодах ограничивалась способностью программиста одновременно мысленно отслеживать последовательность выполняемых операций и местонахождение данных при программировании.

Появление ассемблеров позволило вместо двоичных или 16-ричных кодов использовать символические имена данных и мнемоники кодов операций. В результате программы стали более «читаемыми».

Создание языков программирования высокого уровня существенно упростило программирование вычислений, снизив уровень детализации операций, что позволило увеличить сложность программ.

Появление в языках средств, которые могли оперировать подпрограммами , позволило создать огромные библиотеки расчетных и служебных подпрограмм, которые можно было вызвать из разрабатываемой программы.

Слабым местом такой архитектуры было то, что при увеличении количества подпрограмм возрастала вероятность искажения части глобальных данных какой-либо подпрограммой. Чтобы сократить количество таких ошибок, было предложено в подпрограммах размещать локальные данные.

Появление средств поддержки подпрограмм позволило осуществлять разработку программного обеспечения нескольким программистам параллельно.

1. Второй этап – структурный подход к программированию (60-70-е годы XX в.). В основе структурного подхода лежит декомпозиция (разбиение на части) сложных систем с целью последующей реализации в виде отдельных небольших (до 40-50 операторов) подпрограмм.

Структурное программирование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 1970-х годах Э. Дейкстрой и другими.

В соответствии с данной методологией любая программа строится из трёх базовых управляющих структур: последовательность, ветвление, цикл; кроме того, используются подпрограммы.

Методология структурного программирования появилась как следствие возрастания сложности решаемых на компьютерах задач, и соответственно, усложнения программного обеспечения.

Цель структурного программирования — повысить производительность труда программистов, в том числе при разработке больших и сложных программных комплексов, сократить число ошибок, упростить отладку, модификацию и сопровождение программного обеспечения.

К числу таких сложных программ относятся: системы управления космическими объектами, управления оборонным комплексом, автоматизации крупного финансового учреждения и т.д.

Сложность таких комплексов оценивалась следующими показателями:

* Большой объем кода (миллионы строк)
* Большое количество связей между элементами кода
* Большое количество разработчиков (сотни человек)
* Большое количество пользователей (сотни и тысячи)
* Длительное время использования

Основные принципы и идеи структурного программирования изложены ниже:

* Алгоритм и программа должны разрабатываться поэтапно, по шагам. На начальном этапе проектирования сложная задача (проект) должна разбиваться на более простые части (блоки), каждая из которых должна разрабатываться независимо друг от друга.
* Логика любой программы должна опираться на минимальное количество следующих достаточно простых базовых управляющих структур: ветвление; повторение; следование.

В то время, когда этому принципу уделяли большое внимание, было доказано, что эти три структуры обладают функциональной полнотой. Это означает, что схема любого алгоритма (программы) может быть представлена с использованием только этих конструкций.

* Каждая из БАС должна иметь один вход и один выход.
* Должна соблюдаться строгая дисциплина планирования и документирования, поддержка соответствие кода проектной документации.

1. Модульное программирование.

Модульное программирование предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные, в отдельно компилируемые модули (библиотеки подпрограмм), например, модуль графических ресурсов, модуль подпрограмм вывода на принтер и т.п.

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, или насколько эти связи удалось согласовывать с организацией данных и управления основной программы..

При таком подходе разработка программного обеспечения несколькими программистами значительно упрощается. Каждый разрабатывает свои модули независимо, обеспечивая взаимодействие через специально оговоренные межмодульные интерфейсы. Созданные модули в дальнейшем без изменений можно использовать в других разработках, что также повысило производительность труда программистов.

Узким местом модульного программирования является то, что ошибка в интерфейсе при вызове подпрограммы выявляется только при выполнении программы (из-за раздельной компиляции модулей обнаружить эти ошибки раньше невозможно).

При увеличении размера программы обычно возрастает сложность межмодульных интерфейсов, и с некоторого момента предусмотреть взаимовлияние отдельных частей программы становится практически невозможно. Для разработки программного обеспечения большого объема было предложено использовать объектный (объектно-ориентированный) подход. Родоначальниками этого подхода были Буч(Booch) и Рамбо (Rumbaugh)

Теоретические разработки и внедрение этого подхода составляет сущность *третьего этапа* развития технологий проектирования, активное развитие которого приходится на период с середины 80-ых до конца 90-ых годов 20-го столетия.

Суть подхода состоит в том, что вводится понятие класса как развитие понятия модуля с определенными свойствами и поведением, характеризующими обязанностями класса. Каждый класс может порождать объекты – экземпляры данного класса.

При этом работают основные принципы (парадигмы) ООП:

* Инкапсуляция – объединение в классе данных (свойств) и методов (процедур обработки).
* Наследование – возможность вывода нового класса из старого с частичным изменением свойств и методов.
* Полиморфизм – определение свойств и методов объекта по контексту

Основным достоинством объектно-ориентированного программирования по сравнению с модульным является «более естественная» декомпозиция программного обеспечения, которая существенно облегчает его разработку.

1. Современная технология программирования — компонентный подход, который предполагает построение программного обеспечения из отдельных компонентов — физически отдельно существующих частей программного обеспечения, которые взаимодействуют между собой через стандартизованные двоичные интерфейсы.

В отличие от обычных объектов объекты-компоненты можно собрать в динамически вызываемые библиотеки или исполняемые файлы, распространять в двоичном виде (без исходных текстов) и использовать в любом языке программирования, поддерживающем соответствующую технологию.

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

## Становление CASE-технологий

Одновременно с развитием компонентного подхода развивались и внедрялись в практику так называемые CASE (Computer Aided Software Engineering) технологии проектирования программного обеспечения информационных систем, что оказалось новой ветвью в технологии промышленной разработки и реализации сложных и значительных по объему систем программного обеспечения.

Термин CASE используется в настоящее время в весьма широком смысле. Первоначальное значение термина CASE, ограниченное вопросами автоматизации разработки только лишь программного обеспечения (ПО), в настоящее время приобрело новый смысл, охватывающий процесс разработки сложных ИС в целом.

Теперь под термином CASE-средства понимаются программные средства, поддерживающие процессы:

* Создание и сопровождение ИС, включая анализ и формулировку требований;
* Проектирование прикладного ПО (приложений) и баз данных;
* Генерацию кода;
* Тестирование;
* Документирование;
* Обеспечение качества;
* Конфигурационное управление;
* Управление проектом;
* Другие процессы.

CASE-средства вместе с системным ПО и техническими средствами образуют полную среду разработки ИС.

## Стандартизация и глобализация в сфере программной инженерии

Повсеместное внедрение информационных технологий и систем, вычислительной и телекоммуникационной техники в сферы управления экономикой, научные исследования, производство, а также появление множества компаний — производителей компьютеров и разработчиков программного обеспечения часто рождало проблемы:

* Программное обеспечение, без проблем работающее на одном компьютере, не работает на другом;
* Системные блоки одного вычислительного устройства не стыкуются с аппаратной частью аналогичного;
* ИС компании не обрабатывает данные заказчика или клиента, подготовленные ими на собственном оборудовании;
* При загрузке страницы с помощью «чужого» браузера вместо текста и иллюстраций на экране возникает бессмысленный набор символов.

Эта проблема, реально затронувшая многие сферы бизнеса, получила название проблемы совместимости вычислительных, информационных и телекоммуникационных устройств.

Эти проблемы привели к необходимости объединения конкретных вычислительных устройств и формированию единого информационного пространства (Unified Information Area — UIA).

Такое пространство можно определить, как совокупность баз данных, хранилищ знаний, систем управления ими, информационно-коммуникационных систем и сетей, методологий и технологий их разработки, ведения и использования на основе единых принципов и общих правил, обеспечивающих информационное взаимодействие для удовлетворения потребностей пользователей.

# Заключение

Программная инженерия как некоторое направление возникло и формировалось под давлением роста стоимости создаваемого программного обеспечения. Главная цель этой области знаний - сокращение стоимости и сроков разработки программ.

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, внедрение, сопровождение), составляющих жизненный цикл программного продукта.

Программная инженерия, как и любая другая, занимается не только техническими вопросами производства ПО (специфицирование требований, проектирование, кодирование, …), но и управлением программными проектами, включая вопросы планирования, финансирования, управления коллективом и т.д. Кроме того, задачей программной инженерии является разработка средств, методов и теорий для поддержки процесса производства ПО.

Фундаментальными методами проектирования и разработки являются модульное, структурное и объектно-ориентированное проектирование и программирование.

Несмотря на то, что программная инженерия достигла определенных успехов, перманентный кризис программирования продолжается. Связано это с взрывным ростом использования информационных средств: персональный компьютер, локальные и глобальные вычислительные сети, мобильная связь, электронная почту, семантический Интернет, интеллектуальные технологии, роботизация, Интернет вещей, обеспечение компьютерной безопасности и т.д.