1. Какие два важных аспекта имеет понятие программной инженерии?
2. Как совокупность инженерных методов, которые применяемых на протяжении жизненного цикла программных средств.
3. Как инженерная дисциплина, охватывающая все аспекты жизненного цикла ПС от начальной стадии разработки требований до завершения использования ПС.

Программная инженерия – это совокупность инженерных методов. Инженеры решают конкретные практические задачи. Если соответствующие методы и подходы существуют, они адаптируют их к решению конкретной проблемы, стремясь достичь максимальной эффективности и качества ПС. Если подходящего метода не существует (проблема неразрешима в рамках существующих теорий) инженер методом проб и ошибок подбирает приемлемое решение для данной конкретной ситуации, применяет его и несет ответственность за результат. Набор таких инженерных методов, теоретически пока не обоснованных, но получивших неоднократное подтверждение на практике, имеет большое практическое значение. В программной инженерии они получили название лучших практик (best practices).

1. Приведите примеры сложных и простых программных систем, разница в подходах к их созданию (кратко).

Конечно, не все программные системы сложны. Существует множество программ, которые задумываются, разрабатываются, сопровождаются и используются одним и тем же человеком. Обычно это начинающий программист или профессионал, работающий изолированно. Это не означает, что такие системы плохо сделаны или, что их создатели имеют низкую квалификацию. Но такие системы, как правило, имеют очень ограниченную область применения и короткое время жизни. Обычно их лучше заменить новыми, чем пытаться повторно использовать, переделывать или расширять. Разработка подобных программ скорее утомительна, чем сложна.

Мы будем рассматривать процесс разработки сложных программных комплексов. Они применяются для решения самых разных задач, таких, например, как системы с обратной связью, которые управляют или сами управляются событиями физического мира, и для которых ресурсы времени и памяти ограничены; задачи поддержания целостности информации объемом в сотни тысяч записей при параллельном доступе к ней с обновлениями и запросами; системы управления и контроля за реальными процессами (например, диспетчеризация воздушного или железнодорожного транспорта). Системы подобного типа обычно имеют длительное время жизни, и большое количество пользователей оказывается в зависимости от их нормального функционирования. Среди промышленных программ также встречаются среды разработки, которые упрощают создание приложений в конкретных областях и программы, которые имитируют определенные стороны человеческого интеллекта

1. Перечислите основные причины сложности программного обеспечения.
2. сложностью реальной предметной области, из которой исходит заказ на разработку;

пользователи с трудом могут объяснить в форме, понятной разработчикам, что на самом деле нужно сделать. Бывают случаи, когда пользователь лишь смутно представляет, что ему нужно от будущей программной системы

1. трудностью управления процессом разработки;

Чем больше разработчиков, тем сложнее связи между ними и тем сложнее координация, особенно если участники работ географически удалены друг от друга, что типично в случае очень больших проектов. Таким образом, при коллективном выполнении проекта главной задачей руководства является поддержание единства и целостности разработки

1. необходимостью обеспечения достаточной гибкости программы;

Программирование обладает предельной гибкостью, и разработчик может сам обеспечить себя всеми необходимыми элементами, относящимися к любому уровню абстракции. Такая гибкость чрезвычайно соблазнительна. Она заставляет разработчика создавать своими силами все базовые строительные блоки будущей конструкции, из которых составляются элементы более высоких уровней абстракции. В отличие от строительной индустрии, где существуют единые стандарты на многие конструктивные элементы и качество материалов, в программной индустрии таких стандартов почти нет. Поэтому программные разработки остаются очень трудоемким делом.

1. неудовлетворительными способами описания поведения больших дискретных систем.

внутри большой прикладной программы могут существовать сотни и даже тысячи переменных и несколько потоков управления. Полный набор этих переменных, их текущих значений, текущего адреса и стека вызова для каждого процесса описывает состояние прикладной программы в каждый момент времени. Так как исполнение нашей программы осуществляется на цифровом компьютере, мы имеем систему с дискретными состояниями.

1. Наряду с приведенными причинами сложности, работу со сложными системами усугубляют ограниченные физические возможности человека, которые связаны с объемом его краткосрочной памяти. Психологические 19 исследования показали, что количество структурных единиц, за которыми **одновременно может следить человек, равно 7±2.**
2. Перечислите пять признаков сложной системы.
3. Сложные системы часто являются иерархическими и состоят из взаимозависимых подсистем, которые в свою очередь также могут быть разделены на подсистемы. Это связано, скорее всего, с особенностью нашего восприятия - мы можем понять лишь те системы, которые имеют иерархическую структуру.
4. 2. Выбор того, какие компоненты в данной системе считаются элементарными, относительно произволен и зависит от исследователя. Низший уровень для одного наблюдателя может оказаться достаточно высоким для другого. Одна и та же роль элемента в системе может быть важной для одного и незначительной для другого исследователя.
5. 3. Внутрикомпонентная связь в системе обычно сильнее, чем связь между компонентами. Это обстоятельство позволяет отделять «высокочастотные» взаимодействия внутри компонентов от «низкочастотной» динамики взаимодействия между компонентами. Это дает возможность изолированно изучать каждую часть.
6. 4. Иерархические системы обычно состоят из немногих типов подсистем, по-разному скомбинированных и организованных. Подобно тому, как в растении много разных клеток, но тип элементарного элемента один – клетка.
7. 5. Любая работающая сложная система является результатом развития работавшей ранее более простой системы. Сложная система, спроектированная «с нуля», никогда не заработает. Следует начинать с работающей простой системы.
8. Объясните следующие понятия: «декомпозиция», «абстракция», «иерархия» (виды иерархий).

Декомпозиция - Программную систему можно разделить на подсистемы и совершенствовать независимо, при этом в уме не нужно держать информацию обо всей системе сразу. Различают два типа декомпозиции - **алгоритмическая и объектно ориентированная**. При **алгоритмической декомпозиции** основное внимание концентрируется на порядке происходящих событий (процессах). **Объектно-ориентированная декомпозиция** придает основное значение объектам, которые в процессе взаимодействия друг с другом определяют поведение системы.

Абстракция – выделение существенные характеристики некоторого объекта, отличающего его от всех других объектов, четко определяя его концептуальные границы с точки зрения наблюдателя.

Иерархия «обобщение-специализация» (другое название «общее и частное» или «is-a»), в которой подкласс представляет собой специализированный частный случай своего суперкласса. Эта иерархия по-другому еще называется структура классов. Например, роза – это цветок, цветок – это растение.

Второй тип иерархии – это иерархия «быть частью» (другое название «целое-часть» или «part-of»), когда про элемент говорят, что он часть другого. По-другому такая иерархия называется структура объектов. Например, лепестки - часть цветов.

1. Объясните понятие «каноническая форма системы».

Понятия структура классов и структура объектов, объединенные с пятью признаками сложных систем, называются канонической формой 21 сложной системы (рис. 2.2). Те программные системы, в которых хорошо продумана каноническая форма, являются наиболее успешными.