1.

Архитектурные решения в сравнении с «обычными» проектными решениями рассматриваются как более абстрактные, концептуальные и глобальные; они нацелены на успех всей миссии и на наиболее высокоуровневые структуры системы. Детальное проектирование, в свою очередь, определяется как процесс детализации и расширения предварительного проекта (архитектуры) до такой степени, при которой проект полностью готов к реализации.

2.

**1) Репозиторий**

Все совместно используемые подсистемами данные хранятся в центральной базе данных, доступной всем подсистемам. Репозиторий является пассивным элементом, а управление им возложено на подсистемы.

**2) Паттерн Клиент/сервер**

Данные и процессы системы распределены между несколькими процессорами. Паттерн имеет три основных компонента: набор автономных серверов (предоставляют сервисы другим подсистемам), набор подсистем — клиентов (которые вызывают сервисы, предоставляемые серверами) и сеть (служит для доступа клиентов к сервисам). Клиенты должны знать имена серверов и сервисов, в то время как серверам не надо знать имена клиентов и их количество. Клиенты получают доступ к сервисам, предоставляемым серверами посредством удаленного вызова процедур.

**3) Паттерн обьектно-ориентированный**

Система представляется состоящей из совокупности связанных между собой обьектов. Объекты представляют сервисы (методы) другим объектам и создаются во время исполнения программы на основе определения классов обьектов. Объекты скрывают информацию о представлении состояний и, следовательно, ограничивают к ним доступ.

**4) Паттерн Многоуровневая система (Layers) или Абстрактная машина**

В соответствии с паттерном Многоуровневая система структурные элементы системы организуются в отдельные уровни со взаимосвязанными обязанностями таким образом, чтобы на нижнем

3.

**1) Вызов-возврат**

Вызов программных процедур осуществляется сверху вниз, т. е. управление начинается на вершине иерархии процедур и через вызовы передается на нижние уровни иерархии.

**2) Диспетчер**

Один системный компонент назначается диспетчером и управляет запуском и завершением других процессов системы и координирует эти процессы.

**3) Передача сообщений**

В рамках данного паттерна событие представляет собой передачу сообщения всем подсистемам. Любая подсистема, которая обрабатывает данное событие, отвечает на него.

4.

Связность модуля - внутренняя характеристика модуля, характеризующая меру прочности соединения функциональных и информационных объектов внутри одного модуля. Связность модуля характеризует степень его «плотности», степень зависимости его частей и направленности на решение определенной задачи.

**Функционально связный модуль** содержит объекты, предназначенные для решения одной единственной задачи. Примерами функционально связанных модулей являются модули проверки орфографии, вычисления заработной платы сотрудника, вычисления логарифма функции. Связность: хорошая.

В **последовательно связном модуле** его объекты охватывают подзадачи, для которых выходные данные одной из подзадач являются входными для другой (открыть файл – прочитать запись – закрыть файл). Связность: хорошая.

**Информационно связный модуль** содержит объекты, использующие одни и те же входные или выходные данные. Так, по ISBN книги, можно узнать ее название, автора и год издания. Эти три процедуры (определить название, определить автора, определить год издания) связаны между собой тем, что все они работают с одним и тем же информационным объектом – ISBN. Связность: приемлимая.

**Процедурно связный модуль** – это такой модуль, объекты которого включены в различные (возможно, несвязанные) подзадачи, в которых управление переходит от одной подзадачи к следующей (сделать зарядку, принять душ, позавтракать, одеться, отправится на работу). В отличие от последовательно связанного модуля, в котором осуществляется передача данных, в процедурно связанном модуле выполняется передача управления. Связность: приемлемая.

**Модуль с временной связностью** – это такой модуль, в котором объекты модуля привязаны к конкретному промежутку времени. Примером может являться модуль, осуществляющий инициализацию системы. Элементы данного модуля почти не связаны друг с другом за исключением того, что должны выполняться в определенное время. Связность: плохая.

**Модуль с логической связностью** – это такой модуль, объекты которого содействуют решению одной общей подзадачи, для которой эти объекты отобраны во внешнем по отношению к модулю мире. Так, например, альтернативы: поехать на автомобиле, на метро, на автобусе – являются средством достижения цели: добраться в како-то определенное место, из которых нужно выбрать одну. Связность: плохая.

**Модуль со связностью по совпадению** содержит объекты, которые слабо связаны друг с другом (сходить в кино, поужинать, посмотреть телевизор, проверить электронную почту). Связность: плохая.

5.

Сцепление (coupling) - мера взаимозависимости модулей по данным. Сцепление - внешняя характеристика модуля, которую желательно уменьшать.

Типы сцепления:

1) **Полностью независимые модули**. Модули, не вызывающие друг друга и не использующие общих данных, не сцеплены и являются полностью независимыми.

2. **Сцепление по данным**. Модуль А вызывает модуль В. Все входные и выходные параметры вызываемого модуля - простые элементы данных.

3. **Сцепление по образцу**. В этом случае модули ссылаются на одну и ту же глобальную структуру данных.

4. **Сцепление по общей области**. Модули разделяют одну и ту же глобальную структуру данных.

5. **Сцепление по управлению**. Модуль А явно управляет функционированием модуля В с помощью передачи флагов, переключателей или кодов, посылая ему управляющие данные.

6.

**Делегирование** (англ. Delegation) — основной шаблон проектирования, в котором объект внешне выражает некоторое поведение, но в реальности передаёт ответственность за выполнение этого поведения связанному объекту. Шаблон делегирования является фундаментальной абстракцией, на основе которой реализованы другие шаблоны - композиция (также называемая агрегацией), примеси (mixins) и аспекты (aspects).

Плюсы: Возможность изменить поведение конкретного экземпляра объекта вместо создания нового класса путём наследования.

Минусы: Этот шаблон обычно затрудняет оптимизацию по скорости в пользу улучшенной чистоты абстракции.

**Неизменяемый объект** (англ. Immutable object) — в объектно-ориентированном программировании объект, который не может быть изменён после своего создания.

Объект может быть неизменяемым как полностью, так и частично. Например, применение директивы const к какому-либо члену класса в C++ делает объект частично неизменяемым. В некоторых случаях объект считается неизменяемым с точки зрения пользователя класса, даже если изменяются его внутренние поля. Как правило, неизменяемый объект получает все внутренние значения во время инициализации, либо значения устанавливаются в несколько этапов, но до того, как объект будет использован.

**Интерфейс** (англ. interface) — основной шаблон проектирования, являющийся общим методом для структурирования компьютерных программ для того, чтобы их было проще понять. В общем, интерфейс — это класс, который обеспечивает программисту простой или более программно-специфический способ доступа к другим классам.

Интерфейс может содержать набор объектов и обеспечивать простую, высокоуровневую функциональность для программиста (например, Шаблон Фасад); он может обеспечивать более чистый или более специфический способ использования сложных классов («класс-обёртка»); он может использоваться в качестве «клея» между двумя различными API (Шаблон Адаптер); и для многих других целей.

**Model-View-Controller** (**MVC**, «Модель-Представление-Контроллер», «Модель-Вид-Контроллер») — схема разделения данных приложения, и управляющей логики на три отдельных компонента: модель, представление и контроллер — таким образом, что модификация каждого компонента может осуществляться независимо.

**Модель** (Model) предоставляет данные и реагирует на команды контроллера, изменяя своё состояние.

**Представление** (View) отвечает за отображение данных модели пользователю, реагируя на изменения модели.

**Контроллер** (Controller) интерпретирует действия пользователя, оповещая модель о необходимости изменений.