**ПЛАН ЗАНЯТИЯ**

**Дисциплина:** МДК.01.04 Системное программирование

**Преподаватель:** Галузин А.Б.

**Курс:** 4

**Группа:** П-40

**Специальность:** 09.02.07 Информационные системы и программирование

**Дата:** 4.09.24

**Время проведения:** 12.50-13.25, 1 пара

**Тема:** Архитектура системных программ

**Цель занятия:**

**дидактическая:** изучить архитектуру системных программ

**развивающая**: развивать абстрактное мышление, логику

**Вид занятия** лекция

**Литература**

Головнин О. К. Введение в системное программирование и основы жизненного цикла системных программ: учебное пособие – Самара: Изд-во Самарского университета, 2021, стр. 28-30.

**Интернет-ресурсы:**

<https://nuancesprog.ru/p/12019/?ysclid=mf4i2sonta618683708>

**ЗАДАНИЕ**: законспектировать лекцию с учетом контрольных вопросов.

**КОНСПЕКТ ЛЕКЦИИ**

План

1. Зачем нужна архитектура ПО
2. Что же такое архитектурная модель ПО
3. Классификация архитектурных моделей ПО
4. Организация межпрограммных связей

**1. Зачем нужна архитектура ПО**

Первые разработчики создавали программное обеспечение без архитектуры. Сначала это казалось удобным: никаких издержек, связанных с планированием, и ускоренное прототипирование (разработка макета). Но по мере усложнения ПО теряло гибкость и управляемость, а каждое новое изменение обходилось все дороже. Это мешало развивать проект за границы, определенные изначально. Такая система получила название Большой комок грязи (Big Ball of Mud).

За годы развития ПО разработчикам удалось придумать надежные подходы, чтобы устранить недостатки проектирования без архитектуры.

Необходимость архитектуры обоснована сложностью программного обеспечения. Продуманная архитектура облегчает разработку и дальнейшее развитие ПО. Она служит базисом, каркасом создаваемой системы, интегрируя отдельные компоненты и создавая высокоуровневую модель системы.

**2.** **Что же такое**  **архитектурная модель ПО**

**Архитектурная модель ПО** – принципиальная организация ПО, воплощенная в его элементах, их взаимоотношениях друг с другом и со средой, а также принципы, направляющие проектирование и эволюцию ПО.

Архитектура включает в себя:

− выбор структурных элементов и их интерфейсов, с помощью которых обеспечивается их взаимодействие и совместное функционирование;

− соединение выбранных элементов структуры и поведения во всё более крупные системы;

− архитектурный стиль (software architecture style) − набор принципов, используемых в архитектуре, направляет всю организацию – все элементы, их интерфейсы, их сотрудничество и их соединение.

Короткое описание основных архитектурных стилей приведено в таблице 1.

|  |  |  |
| --- | --- | --- |
| Таблица 1. Основные архитектурные стили | | |
| **Архитектурный стиль** | **Описание** |  |
| Клиент-серверная модель | Разделение системы на два приложения – клиент и сервер. При работе клиент посылает запросы на обслуживание серверу |  |
| Компонентная архитектура | Деление системы на компоненты, которые могут быть повторно использованы и не зависят друг от друга. Каждый компонент снабжается известным интерфейсом для коммуникаций |  |
| Многоуровневая архитектура | Разделение функций приложения на группы (уровни), которые организованы в виде стекового набора |  |
|  |
|  |
| Шина сообщений | Система, которая может посылать и передавать информа-  ционные сообщения в определённом формате по общему  коммуникационному каналу. Благодаря этому организу-  ется взаимодействие систем без указаний конкретных по-  лучателей сообщений |  |
|  |
|  |
|  |
|  |
| Многозвенная архитектура | Разделение функций подобно многоуровневой архитек-  туре, но группировка происходит не только на логиче-  ском, а и на физическом уровне – отдельным группам со-  ответствует отдельный компьютер (сервер, кластер) |  |
|  |
|  |
|  |
|  |
| Объектно-ориентированная архитектура | Представление системы в виде набора взаимодействую-  щих объектов |  |
|  |
| Выделенное представление | Выделение в системе отдельных групп функций для взаи-  модействия с пользователями и обработки данных |  |
|  |
|  |
| Архитектура, ориентированная на сервисы | Каждый компонент системы представлен в виде незави-  симого сервиса, предоставляющего свои функции по  стандартному протоколу |  |
|  |
|  |
|  |
|  |

Важно понимать, что стили не исключают совместное применение, особенно при проектировании сложных систем. По сути, архитектурные стили допускают группировку согласно направлению решаемых ими задач.

Архитектурный вид состоит из двух основных компонентов: структурных элементов и отношений между ними. При этом схематически архитектура может быть представлена в логическом или физическом виде.

Если структурные элементы представляют собой концептуальные (логические) компоненты ПО, то такая модель программной архитектуры будет соответствовать логической. Физическая же архитектура будет являться реализацией логической структуры, когда ее элементами станут физически существующие компоненты, такие как программы, базы и файлы данных, интерфейсы и т. д.

**3. Классификация архитектурных моделей ПО**

Существует множество способов классификации и описания архитектурных моделей ПО. Рассмотрим основные из них, где классификация производится по количеству звеньев (уровней):

− одноуровневая – все компоненты программы размещены на одном компьютере и на нем же выполняются;

− двухуровневая архитектура предлагает наличие двух компонентов: клиента и сервера (к которому подключен клиент).

− трехуровневая архитектура предполагает наличие трех компонентов: клиента, сервера приложений (к которому подключен клиент) и сервера баз данных (с которым работает сервер приложений).

**4. Организация межпрограммных связей**

Организация межпрограммных связей возможна многими способами. Связи могут быть установлены:

− на уровне библиотек (для программ, функционирующих на одном устройстве);

− на уровне сервисов, осуществляющих обмен данными по различным протоколам (для клиент-серверных приложений, для интеграции серверов);

− на уровне данных (совместный доступ к данным);

− через различные технологии интеграции (ESB, CORBA, ActiveX и др.).

Разработка системного ПО может быть значительно облегчена за счет использования **библиотек подпрограмм**, инкапсулирующих (инкапсуляция (от лат. in capsule — в оболочке) — это заключение данных и функциональности в оболочку) функции и визуальные компоненты для повторного использования.

Преимущества такого подхода:

* не требуется включать в программу код часто используемых функций, что существенно сокращает объем кода;
* различные программы, выполняемые в некоторой ОС, могут пользоваться кодом одной и той же библиотеки;
* изменения и улучшения функций библиотек не требуют перекомпиляции программы.

В современных средах программирования для поставки и интеграции библиотек используются диспетчеры пакетов, предназначенные для управления библиотеками подпрограмм и установления зависимостей между ними. Примеры диспетчеров пакетов: NuGet, Maven.

**Вопросы для самоконтроля**

1) Зачем нужна архитектура ПО?

2) Дайте определение архитектурной модели ПО. Что она включает?

2) Какие классы архитектурных моделей ПО выделяют при классификации по количеству звеньев?

3) Какими способами возможна организация межпрограммных связей?

4) Для чего применяются библиотеки подпрограмм?

5) Что такое прототипирование?