Лекция №5. ПАТТЕРНЫ И ФРЕЙМВОРКИ В АРХИТЕКТУРЕ ИС

Учебные вопросы:

1. Паттерны
2. Антипаттерны
3. Фреймворки
4. Примеры фреймворков

Вопрос №1

Паттерны (***шаблоны***) впервые появились в середине 1990-х гг. как составная часть объектно-ориентированного подхода и рассматривались как набор объектов, организованных определенным образом для решения конкретного класса задач.

Основное отличие паттернов от компонентов состоит в том, что **компонент является модулем**, который после настройки можно включить в состав системы, т.е. его можно рассматривать как готовый к употреблению строительный блок, а **паттерн — это только заготовка**, которую еще надо обработать, т. е. добавить код, определяющий функциональность.

Концептуальные паттерны — это паттерны, функционирование которых описывается в терминах предметной области. Такие паттерны относятся к приложению в целом или крупным подсистемам ИС.

Паттерны проектирования — это паттерны, для описания которых используются термины, относящиеся к разработке программных систем, такие как объект, класс, модуль. Паттерны проектирования описывают решение общих проблем в конкретном контексте.

Программные паттерны — это паттерны, для описания которых используются такие относительно низкоуровневые понятия как деревья, списки и т. п.

Архитектурный паттерн (architectural patterns) описывает структуру программной системы и определяет состав подсистем, их основные функции и допустимые способы компоновки подсистем. Архитектурные паттерны называют также архитектурными стилями.

Системные паттерны (system patterns) представляют собой приложение на верхнем (системном) уровне. Системные паттерны можно рассматривать как паттерны, использование которых позволяет получить улучшенные архитектурные решения.

Структурные паттерны с одинаковой эффективностью применяются как для разделения, так и для объединения элементов приложения.
Структурные шаблоны могут быть использованы для решения различных задач. Например, **шаблон *Адаптер*** может обеспечить возможность двум несовместимым системам обмениваться информацией, тогда как **шаблон *Фасад*** позволяет отобразить упрощенный пользовательский интерфейс, не удаляя ненужных конкретному пользователю элементов управления.

Поведенческие паттерны (behavioral patterns) применяются для передачи управления в системе.

1. Цепочка ответственности (Chain of Responsibility)
2. Команда (Command)
3. Интерпретатор(Interpreter)
4. Итератор (Iterator)
5. Медиатор (Mediator)
6. Моментальный «снимок» (Memento)

Производящие паттерны (creational patterns) предназначены для создания объектов в системе.

Паттерны параллельного программирования ориентированы на обеспечение корректного взаимодействия асинхронно протекающих процессов и ориентированы на решение двух основных задач: *совместное использование ресурсов* и *управление доступом к ресурса*

паттерны — это классы проверенных практикой проектных решений, использование которых приводит к положительным результатам.

Вопрос №2

Антипаттерны (antipatterns), также известные как ловушки (pitfalls) — это классы наиболее часто внедряемых ***плохих решений*** проблем.

Они изучаются как категория, в случае, когда их хотят избежать в будущем, и некоторые отдельные случаи их могут быть распознаны при изучении неработающих систем

Частью хорошей практики программирования является избегание антипаттернов.

Антипаттерны в управлении разработкой ПО и их свойства

- **«Дым и зеркала» (Smoke and mirrors)** Демонстрация того, как будут выглядеть ненаписанные функции. Название происходит от двух излюбленных способов, которыми фокусники скрывают свои секреты
- **«Раздувание» ПО (Software bloat)** Разрешение последующим версиям системы требовать все больше и больше ресурсов
- **«Функции для галочки»** Превращение программы в конгломерат плохо реализованных и не связанных между собой функций (как правило, для того чтобы заявить в рекламе, что функция есть)

Антипаттерны в разработке ПО

- **Неопределенная точка зрения** (Ambiguous viewpoint) Представление модели без спецификации ее точки рассмотрения
- **«Большой комок грязи»** (Big ball of mud) Система с нераспознаваемой структурой
- **«Раздувание интерфейса»** (Interface bloat) Изготовление интерфейса очень мощным и очень трудным для осуществления

Антипаттерны в объектно-ориентированном проектировании

- **«Божественный» объект** (God object) Концентрация слишком большого количества функций в одной части системы (классе)
- **«Полтергейст»** (Poltergeist) Объекты, чье единственное предназначение — передавать информацию другим объектам

Антипаттерны в области программирования

- **Ненужная сложность** (Accidental complexity)Внесение ненужной сложности в решение
- **«Действие на расстоянии»** (Action at a distance) Неожиданное взаимодействие между широко разделенными частями системы
- **«Накопить и запустить»** (Accumulate and fire) Установка параметров подпрограмм в наборе глобальных переменных

Методологические антипаттерны

Организационные антипатгерны

Вопрос №3

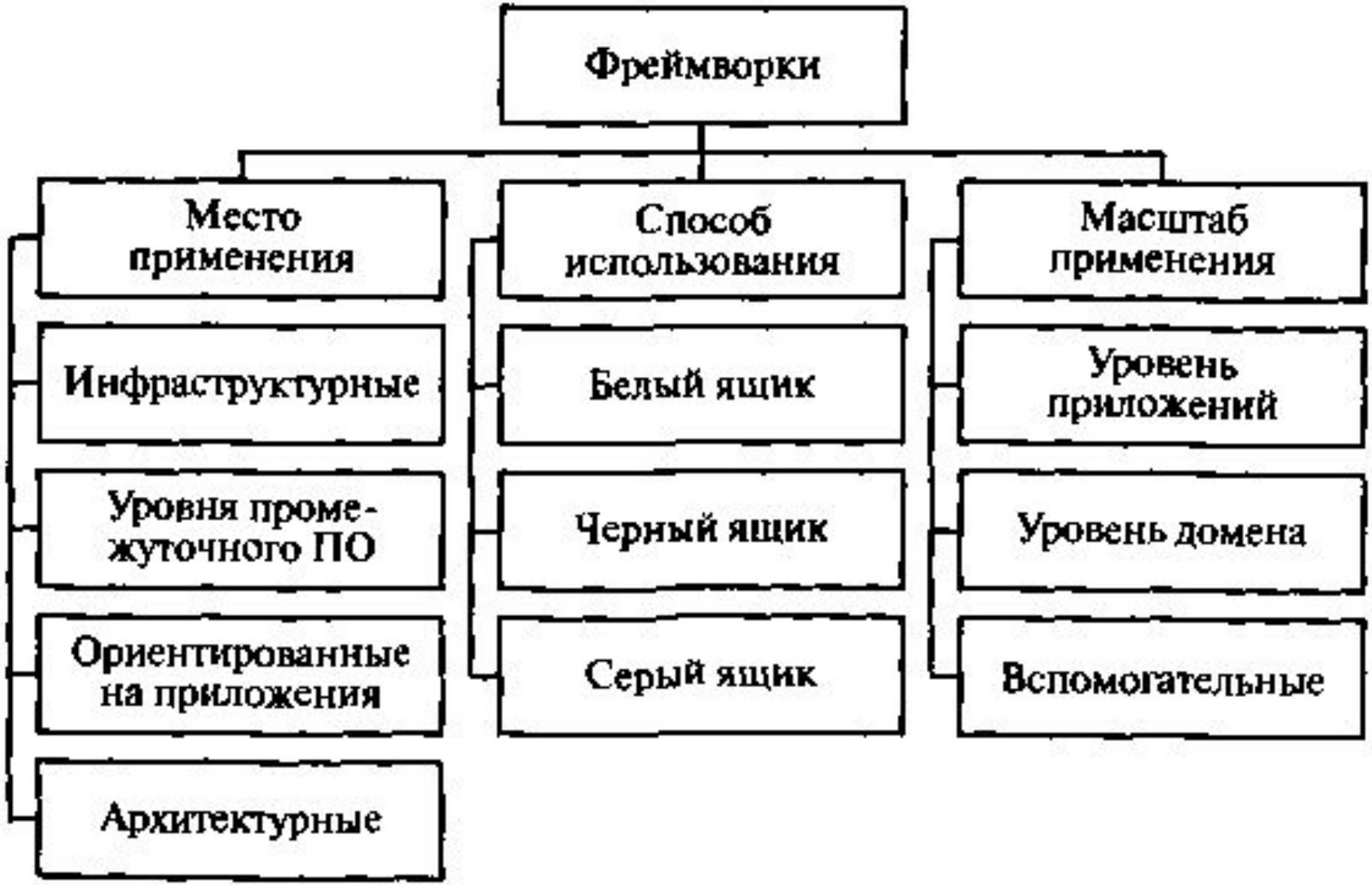
Фреймворк определяют, как набор типовых решений, методик проектирования и классов, которые могут быть использованы при решении множества сходных задач.

В самом широком смысле под фреймворком понимают общепринятые архитектурно-структурные решения и (или) подходы к проектированию ИС.

Применительно к программным системам фреймворк, или каркас, представляет собой набор классов или структур, которые описывают решение некоторого класса задач.

Фреймворк можно рассматривать как реализацию системы паттернов проектирования.

Фреймворк представляет собой скелетное решение достаточно крупной задачи и обычно включает в себя большое количество как паттернов, так и компонентов.



Классификация фреймворков

Архитектурные фреймворки это«совокупность соглашений, принципов и практик, используемых для описаний архитектур и принятых применительно к некоторому предметному домену и (или) в сообществе специалистов (заинтересованных лиц)»

Фреймворки уровня промежуточного ПО используются для интеграции распределенных приложений и компонентов.

Фреймворки, ориентированные на приложения, используются, в первую очередь, для поддержания процесса разработки систем, ориентированных на конечного пользователя и принадлежащих некоторому конкретному предметному домену.

Использование ***инфраструктурных фреймворков*** упрощает разработку инфрастуктурных элементов, таких как, например операционные системы. Обычно такие фреймворки используются внутри организации и не поступают в продажу.

Фреймворки, используемые по принципу белого ящика, называют также архитектурными фреймворками (architecture-driving framework). Фреймворк, работающий по принципу белого ящика, определяется через интерфейсы объектов, которые разработчик может добавлять в систему.

Фреймворки, используемые по принципу черного ящика, называют также фреймворками, управляемыми данными. В качестве основных механизмов формирования приложения выступают композиция компонентов и параметризация.

Фреймворки уровня приложения (application frameworks) обеспечивают полный набор функций, которые реализуются типовыми приложениями. **Обычно** сюда входят GUI, базы данных, документация. Примером таких фреймворков могут быть MFC (Microsoft Foundation Classes), которые служат для создания приложений, ориентированных на работу в среде MS Windows.

Фреймворки уровня домена (Domain frameworks) используются для создания приложений, относящихся к определенному предметному домену.

Вопрос №4

Фреймворк Захмана

В основе данного фреймворка лежит классификация (таксономия) артефактов. функционирование организации можно описать в терминах ответа на шесть простых вопросов: что, как, где, кто, когда, почему:

- используемые данные (что?);
- процессы и функции (как?);
- места выполнения процессов (где?);
- организации и персоналии (кто?);
- управляющие события (когда?);
- цели и ограничения, определяющие работу системы (почему?).

Фреймворк Захмана

Ответы на эти вопросы можно давать с использованием различных понятий, т.е. с разной степенью детализации. При этом выделяется шесть уровней:

- уровень контекста;
- уровень бизнес-описаний;
- системный уровень;
- технологический уровень;
- технический уровень;
- уровень реальной системы.