Модульная архитектура ПО

JSP & Servlets
14 уровень, 5 лекция

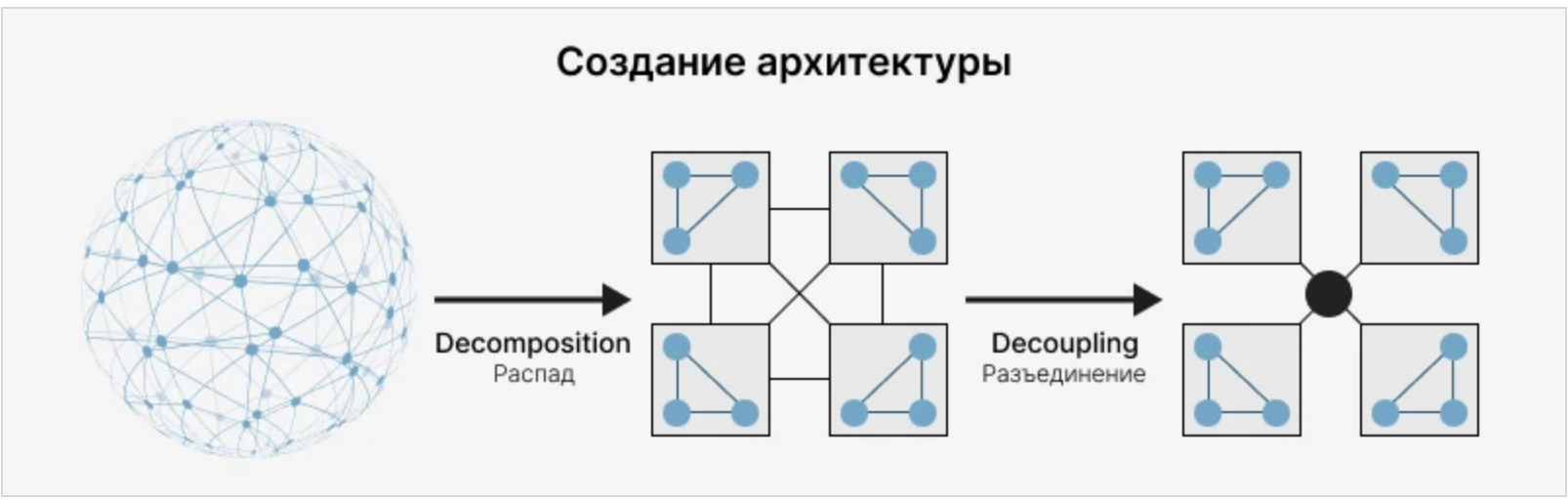
ОТКРЫТА

6.1 Декомпозиция

Несмотря на разнообразие критериев, все же главной при разработке больших систем считается задача снижения сложности системы. Для снижения сложности ничего, кроме деления на части, пока не придумано.

Иногда для простоты это называют принципом “разделяй и властвуй”, но, с точки зрения архитектора ПО, речь идет об **иерархической декомпозиции**.

Сложная система должна строиться из небольшого количества более простых подсистем, каждая из которых, в свою очередь, строится из частей меньшего размера и так до тех пор, пока самые небольшие части не будут достаточно просты для непосредственного понимания и создания.



Отличная новость состоит в том, что данное решение является не только единственно известным, но и универсальным. Помимо снижения сложности, оно одновременно обеспечивает **гибкость системы**, дает хорошие возможности для **масштабирования**, а также позволяет повышать устойчивость за счет дублирования критически важных частей.

Соответственно, когда речь идет о построении архитектуры программы, создании ее структуры, под этим подразумевается декомпозиция программы на подсистемы, сервисы, слои, подпрограммы и функциональные модули и организация их взаимодействия друг с другом и внешним миром.

И самое ценное тут вот что: чем более независимы подсистемы, тем безопаснее сосредоточиться на разработке каждой из них в отдельности в конкретный момент времени и при этом не заботиться обо всех остальных частях.

6.2 Преимущества модульной архитектуры

Использование принципа иерархической декомпозиции позволяет избавиться от хаоса в тысячах классов твоего кода. Помнишь, что твой код разбивается по пакетам (package) и подпакетам? Это и есть одно из выражений иерархический декомпозиции.

Твоя программа из кучи классов превращается в набор библиотек и модулей, взаимодействующих друг с другом по хорошо определенным и простым правилам. Это в свою очередь позволяет контролировать ее сложность, а также дает возможность получить все те преимущества, которые обычно соотносятся с понятием хорошая архитектура.

Вот самые основные из них:

- **Масштабируемость** (Scalability) – возможность расширять систему и увеличивать ее производительность за счет добавления новых модулей.
- **Ремонтопригодность** (Maintainability) – изменение одного модуля не требует изменения других модулей.
- **Заменимость модулей** (Swappability) – модуль легко заменить на другой.

- **Возможность тестирования** (Unit Testing) – модуль можно отсоединить от всех остальных и протестировать/починить.
- **Переиспользование** (Reusability) – модуль может быть переиспользован в других программах и другом окружении.
- **Сопровождаемость** (Maintenance) – разбитую на модули программу легче понимать и сопровождать.

Можно сказать, что в разбиении сложной проблемы на простые фрагменты и заключается цель всех методик проектирования. А термином “архитектура” в большинстве случаев просто обозначают результат такого деления плюс "некие конструктивные решения, которые после их принятия с трудом поддаются изменению" (Мартин Фаулер «Архитектура корпоративных программных приложений»).

Поэтому большинство определений в той или иной форме сводятся к следующему:

"Архитектура идентифицирует главные компоненты системы и способы их взаимодействия. Также это выбор таких решений, которые интерпретируются как основополагающие и не подлежащие изменению в будущем".

"Архитектура — это организация системы, воплощенная в ее компонентах, их отношениях между собой и с окружением. Система — это набор компонентов, объединенных для выполнения определенной функции."

Таким образом, **хорошая архитектура — это, прежде всего, модульная/блочная архитектура**. Чтобы получить хорошую архитектуру, надо знать, как правильно делать декомпозицию системы. А значит, необходимо понимать, какая декомпозиция считается “правильной” и каким образом ее лучше проводить.

‹ Предыдущая лекция

Следующая лекция ›

JavaRush — это интерактивный онлайн-курс по изучению Java-программирования с нуля. Он содержит 1200 практических задач с проверкой решения в один клик, необходимый минимум теории по основам Java и мотивирующие фишки, которые помогут пройти курс до конца: игры, опросы, интересные проекты и статьи об эффективном обучении и карьере Java-девелопера.