Краеугольный камень – архитектура

> *«Иисус говорит им: неужели вы никогда не читали в Писании: камень, который отвергли строители, тот самый сделался главою угла? Это от Господа, и есть дивно в очах наших?»* ***Евангелие от Матфея, 21,42***

Простота, которая хуже воровства

Первые впечатления при «обкатке» лабораторного практикума по дисциплине «Программная инженерия». Бригада получает в качестве варианта задания разработку системы из серверного и ряда клиентских приложений, например «Диспетчерская служба такси» или «Система распространения и продажи театральных билетов». Более-менее успешно выполняются работы «Видение проекта. Моделирование предметной области», «Моделирование прецедентов», «Разработка требований», «Проектирование графического интерфейса». Типовые ошибки – «грабли», на которые наступают практически все, легко поддаются устранению, т.к. причины их легко объяснимы и понимаются. Пример: в модель предметной области, статическую по своей сути, добавляются компоненты поведения или компоновки (программных или физических компонент) системы. Скажем, наряду с билетом, кассиром, сменой появляется авторизация, сервер или браузер. Но как только дело доходит до разработки архитектуры системы, массовое сознание воспроизводит на свет что-то подобное:

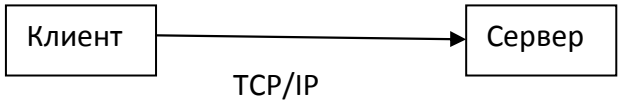

рис.1.1. Архитектурная банальность

Сказать, что это не соответствует действительности, нельзя. Некоторые даже начинают дискуссию на эту тему. С другой стороны, действительность какая-то убогая. Основной ее порок – непродуктивность, ее нельзя развивать дальше, но и нельзя отдать исполнителям, засмеют. Попробуем, для начала, сформулировать, что же следует понимать под архитектурой, прежде чем браться за ее проектирование.

Архитектура как согласованный взгляд на устройство системы

Архитектура, как фундаментальное понятие, нельзя сформулировать одной фразой. Наиболее естественное – перечисление элементов, входящих в ее описание (Philippe Kruchten, *The Rational Unified Process: An Introduction)*. Итак, архитектура – это:

- значимые решения по поводу организации ПС;
- структурные элементы и их интерфейсы, при помощи которых компонуется система;
- поведение - взаимодействие между этими элементами;
- компоновка элементов в иерархию подсистем;
- стиль архитектуры, который направляет эту организацию.

Другое важное свойство – множественность представлений (видов или срезов), каждый из которых имеет отношение к определенной группе участников разработки (концепция 4+1) [12]:

- **функциональное представление –** сценарии, прецеденты, требования, предметная область;
- **логическое представление** – реализация функциональности на системе взаимодействующих классов в логической модели проекта;
- **процедурное представление** – параллелизм, потоки, взаимодействие, масштабируемость, производительность;
- **представление развертывания** – компоновка, топология, коммуникации, администрирование, настройка;
- **представление разработки** – структура программного кода, библиотеки, интерфейсы, абстракции.

В совокупности они должны давать целостное представление об организации системы, чтобы не получилось как в притче о слепых мудрецах и слоне. (каждый из них трогал определенную часть слона и давал свое определение, на что похож слон [11]). Именно создание целостного представления является отличительной особенностью хорошего описания архитектуры.

С другой стороны, излишние подробности в описании каждого из представлений не являются существенными для понимания системы в целом и могут варьироваться в процессе реализации, поэтому для архитектуры уместно требовать **минимальной избыточности** описания. Если собрать все сказанное вместе, то получится примерно так:

> **Архитектура –** минимально избыточное, согласованное описание системы, включая структуру, поведение, компоновку, стили разработки и значимые решения, создающее адекватное представление о всех аспектах ее организации

Архитектурные аспекты проектирования

Любое более-менее общее решение на уровне проектирования является архитектурным. Если исходить из абсолютного предшествования проектирования (дизайна) по отношению к реализации (конструированию), то получается логичная цепочка: архитектура – проектирование - реализация. Хотя на практике часто происходит с точностью до наоборот: реализация «как получится» - осознание проблемы – принятие архитектурного решения – рефакторинг кода (сломать и сделать заново).

Здесь можно навскидку перечислить аспекты проектирования, в которых важны предварительные архитектурные решения, а также соответствующие им артефакты проекта:

- структура системы – подсистемы, слои, интерфейсы, проектные классы;
- поведение системы – диаграммы устойчивости, взаимодействия, состояний;
- структура кода - подсистемы, пакеты, абстракции, интерфейсы, библиотеки, зависимости;
- паттерны проектирования (конструирования);
- анализ требований и функционала – модели классов анализа (на основе модели предметной области), бизнес-объекты;
- проектирование данных – структура БД, форматы данных, конвертирование, миграция, протоколы;
- пользовательский интерфейс;

- параллелизм – синхронизация, взаимодействие, «узкие места»;
- обработка событий (связность по управлению);
- доступ к данным – связность, синхронизация изменений, целостность;
- масштабирование – оценка производительности, технологические решения;
- устойчивость – нейтрализация и логирование ошибок, восстановление;
- управление конфигурациями – отслеживание ошибок, обновление версий, бета-тестирование;
- безопасность – идентификация, шифрование, защита;
- развертывание – размещение системы по физическим узлам;
- среда разработки – языки программирования, IDE, фреймворки, контроль версий.

Для того, чтобы окончательно утвердиться в тезисе об архитектуре как «каркасе», на котором разворачивается процесс проектирования, процитируем содержание документа «Свод знаний по программной инженерии (SWEBOK)[13]. Проектирование программного обеспечения (Software Design)»:

1.4 Техники применения – ключевые идеи и концепции: абстрагирование, связность и соединение, модульность и декомпозиция, инкапсуляция, разделение интерфейса и реализации

2. Ключевые вопросы

2.1 Параллелизм

2.2 Контроль и обработка событий

2.3 Распределение компонентов

2.4 Обработка ошибок и исключительных ситуаций и обеспечение отказоустойчивости

2.5 Взаимодействие и представление (MVC)

2.6 Сохраняемость данных (доступность «долгоживущих» данных)

3. Структура и архитектура программного обеспечения

3.1 Архитектурные структуры и точки зрения (структурная, поведенческая, логическая, физическая, реализации кода)

3.2 Архитектурные стили

3.3 Шаблоны проектирования

3.4 Семейства программ и фреймворков (повторное использование архитектурных решений, линии продуктов)

5. Нотации проектирования

5.1 Структурные описания, статический взгляд

5.2 Поведенческие описания, динамический взгляд

Архитектурная пропасть между функционалом и исполнением

Функциональное проектирование в целом (сюда входит бизнес- и системная аналитика, моделирование предметной области, разработка требований) до некоторой степени висит в воздухе, т.к. отвечает на вопрос «что сделать», но не отвечает «каким образом». Более того, она не гарантирует, что требуемый функционал получится реализовать эффективно и гармонично.

В то же время прямолинейная реализация функционала исполнителями, не основанная на проработанной архитектуре (вроде бы известно «что сделать» и «как»), приводит к созданию плохо структурированного и неуправляемого кода (антипаттерн «комок грязи»).

Аналитика и исполнение - это разные берега бурного потока разработки, которые соединяются спасительным мостиком архитектуры. Для целей архитектурного проектирования в команде разработчиков вводится роль системного архитектора. По своему «социальному происхождению» системный архитектор – выходец из среды разработчиков, который имеет богатый опыт применения конкретных архитектурных решений, знает внутренние процессы и механизмы их работы, и протчая, и протчая и протчая.

Теперь понятно, почему так неудачно складывается процесс архитектурного проектирования в цикле лабораторных работ: отсутствие опыта превращает этот процесс в схоластику в большей степени, чем все остальные.