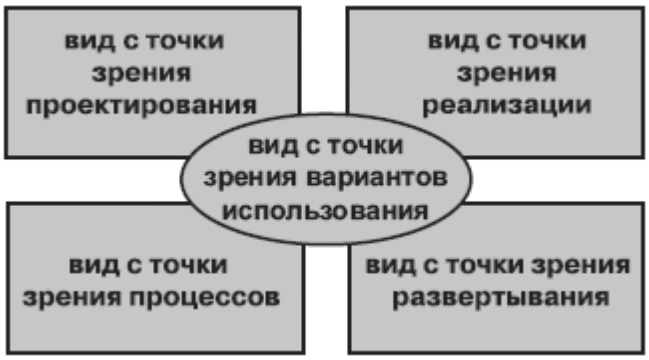
Архитектура программных систем

Архитектура (одно из определений) – это набор существенных решений относительно:

- организации программной системы;
- выбора структурных элементов, составляющих систему, и их интерфейсов;
- поведения этих элементов, определенного в их кооперациях;
- объединения этих структурных и поведенческих элементов в более крупные подсистемы;
- архитектурного стиля, определяющего организацию системы: статические и динамические элементы и их интерфейсы, кооперацию и композицию.

Архитектура программного обеспечения касается не только его структуры и поведения, но также пользовательских свойств, функциональности, производительности, гибкости, возможности повторного использования, понятности, экономических и технологических ограничений и компромиссов, а также эстетических вопросов. Как показано на рисунке, архитектура программной системы может быть наилучшим образом описана с помощью пяти взаимосвязанных представлений. Каждое представление – проекция организации и структуры системы, сосредоточенная на определенном ее аспекте.



Вид «4+1»

Представление вариантов использования системы охватывает варианты использования, описывающие поведение системы с точки зрения конечных пользователей, аналитиков и тестировщиков. Их взгляд в действительности специфицирует не истинную организацию программной системы, а лишь некие движущие силы, формирующие системную архитектуру. В языке UML статические аспекты этого представления передаются диаграммами вариантов использования, а динамические его аспекты – диаграммами взаимодействий, состояний и деятельности.

Представление системы с точки зрения проектирования охватывает классы, интерфейсы и кооперации, формирующие словарь проблемы и ее решение. Это представление в основном поддерживает функциональные требования к системе, то есть сервис, который она должна

предоставлять конечным пользователям. Статические аспекты этого представления в UML сосредоточены в диаграммах классов и объектов, а динамические передаются диаграммами взаимодействий, состояний и деятельности. Диаграмма внутренней структуры класса, в частности, также полезна.

Представление взаимодействия (с точки зрения процессов) системы показывает поток управления, проходящий через разные ее части, включая возможные механизмы параллелизма и синхронизации. Это представление касается производительности, масштабируемости и пропускной способности системы. Статические и динамические аспекты этого представления в UML представлены в некоторых видах диаграмм, используемых в представлении дизайна, но сфокусированы на активных классах, управляющих системой, и передаваемых между ними сообщениях.

Представление реализации системы охватывает артефакты, используемые для сборки и физической реализации системы. Это представление в первую очередь относится к управлению конфигурацией версий системы, состоящей из независимых (в определенной степени) файлов и компонентов, которые могут быть собраны различными способами для формирования работающей системы. Оно также связано с отображением из логических классов и компонентов в физические артефакты. В UML статические аспекты этого представления отражены в диаграммах артефактов, динамические аспекты – в диаграммах взаимодействия, состояний и деятельности.

Представление развертывания системы охватывает узлы, образующие топологию оборудования, на котором работает система. Это представление в основном связано с поставкой, распределением и установкой частей, составляющих физическую систему. Его статические аспекты в UML описываются диаграммами размещения, а динамические – диаграммами взаимодействий, состояний и деятельности.

Каждое из этих пяти представлений может быть достаточным для различных заинтересованных лиц, имеющих отношение к разработке и эксплуатации системы, позволяя им сосредоточиться только на тех аспектах архитектуры, которые непосредственно их касаются. Однако все эти представления также взаимодействуют друг с другом. Узлы из представления размещения содержат компоненты из представления реализации, которые, в свою очередь, представляют собой физическую реализацию классов, интерфейсов, кооперации и активных классов из представлений дизайна и взаимодействия.