# II. Архитектура на облачна система за управление на поръчките от клиенти

В този раздел са представени основни елементи, които да положат фундамент на софтуерно решение (прототип). Първа подточка ще разгледа общи принципи на софтуерната архитектура. След това втора и трета ще дадат детайлна характеристика на концептуалните и функционални модели на подсистемите.

## 2.1 Същност, цел и обхват на софтуерната архитектура

В общ смисъл, софтуерната архитектура е структурирано решение, което може да оптимизира общи атрибути на качеството като: висока производителност, сигурност, контрол, управляемост, мащабируемост, достъпност. (Ali Babar et al., 2009)

Основни изисквания към системата са:

* Потребителски – начинът, по който крайните потребители взаимодействат със системата;
* Бизнес изисквания – по-евтино, по-бързо, по-добре от конкурентите;
* Изисквания към ИТ системата (инфраструктурата);

Кларк и Уокър (2001) въвеждат концепцията за композиционни модели като начин за разработка. Структурни елементи и интерфейси, съставящи системата са:

* Обекти – градивни елементи от ниско ниво;
* Комуникационните канали в архитектурата;
* Картографиране структурата на данните от високо ниво;
* Архитектурни стилове (Model-View-Controller, Ориентираната към микроуслуги и други), които ръководят композицията.

Освен това дизайнът трябва да обхваща икономически, технологични ограничения и естетически проблеми.

Целта е да се структурира документ от високо ниво, който да не навлиза в подробности за изпълнението. Сложността на операциите трябва да е сведена до минимум, като да обръща внимание на всички изисквания. Приложенията трябва да се съвместими с всички случаи на употреба и бизнес сценарии. Критичните точки трябва да се идентифицират и проучат. Добра практика при проектиране на софтуерна архитектура е да бъде ориентирана към модулност, приемайки лесно новите промени.

Принципи на проектиране, отвърдили се във времето като неоходимост за архитектурата са:

* Разделяне на грижите (Separation of Concerns) - всеки обект и модул трябва да бъде в своя собствена грижа и контекст;
* Капсулиране (Encapsulation)
* Инверсия на зависимостта (Dependency Inversion
* Изрични компоненти (Explicit Components
* Единична отговорност (Single Responsibility) - елементите в трябва да имат една единствена цел.
* Не се повтаряйте (Don’t Repeat Yourself ) –важно е да няма компоненти с една и съща цел;
* Устойчивост и невежество относно инфраструктурата (Persistence & Infrastructure Ignorance)
* Презентация Невежество (Presentation Ignorance
* Ограничени контексти (Bounded Contexts
* Тестваемост (Testability

Унифицираният език за моделиране (Unified Modeling Language) e графичен език за визуализиране, специфициране, конструиране и документиране на елементите на една софтуерна система.

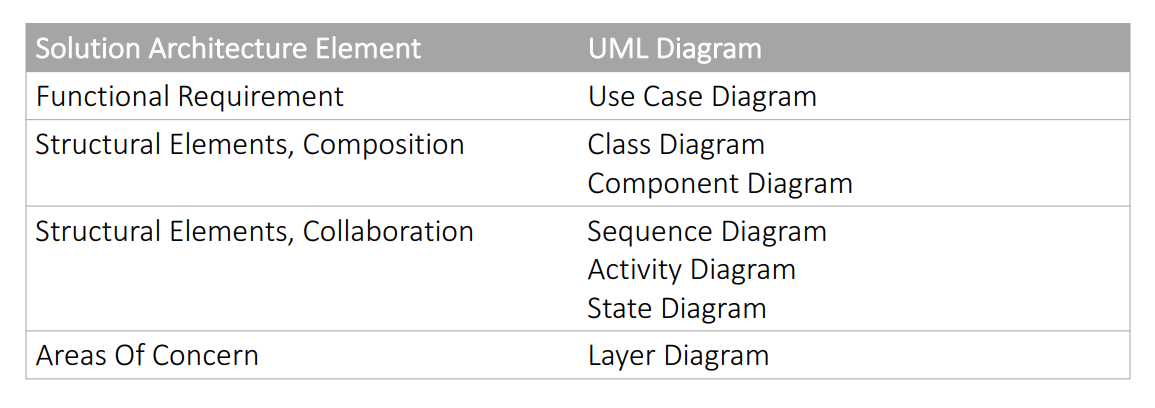
Основни атрибути на UML:

* + Визуално – лесно се вижда представянето на архитектурата;
  + Абстрактно – стои далеч от детайлите на изпълнението;
  + Описателен – показва пълното представяне;
  + Стандарт – UML е отвърден световен стандарт;
  + Поддържа генериране на код – определени секции могат да бъдат конвертирани в код;

Типове UML модели:

* + Бизнес модел – нетехнически, детайлен, приемайки системата като черна кутия.
  + ИТ модел – разделя се на статични (структурни), които изобразяват как се съчетават различните елементи, и динамични (Поведенчески), които се фокусират върху взаимоотношенията.

UML диаграми в архитектурата:



Стратегии за проектиране на UML:

* + UML като скица –предназначен към общи насоки;
  + UML като план - много подробен, може да се пишете код въз основа на диаграмите;
  + UML като валидиране – валидиране на изпълнението спрямо диаграматал;

# Литература:

1. <https://www.sciencedirect.com/topics/computer-science/structured-solution> (Anil Kumar Thurimella, T. Maruthi Padmaja, in Economics-Driven Software Architecture, 2014)