**Software Engineering Notes**

*Created: 2024-09-15*

*Modified: 2024-09-30*

**REFERENCES**

* Introduction to Software Engineering course at the University of Toronto

**PRODUCT MANAGEMENT**

* About solving customer’s problems by deciding on what development team should build.
* A successful software project needs a balance of tech, UX, and business (domain).
* ***Users***: people who use the product you build.
* ***Customers***: people who pay for the product you build.

**Determining What Software to Build**

1. Define high-level product goals. Answer who, what, why focusing on problem and users not product.
2. Understand what users need using personas, e.g., user stories: As <role>, I want <action/desire>, so that <benefit>.
   1. ***Acceptance Criteria***: conditions a product must satisfy to be accepted by user in story.
3. Product design: build a minimal prototype, UI, how it works, check with stakeholders (users, customers).

**Lean Project Management**

* Build a prototype that is minimal, MVP (minimum viable product)
* [TODO] Lean startup: <https://theleanstartup.com/principles>

**DATA MODELLING**

**Data Modelling**

* ***Conceptual Data Model*** (CDM): a business-level artifact that defines persistent entities, attributes of entities, relationships between entities.
* ***Logical Data Model*** (LDM): platform independent, normalized (no duplicates) data model made up of tables of 1-1 mappings between CDM entities and LDM tables.
* Database normalization:
  + ***1NF***: each attribute have atomic data and each table has a PK (unique id).
  + ***2NF***: every non-key attribute (column) must depend on all columns of the PK.
  + ***3NF***: attributes determined only by PK.
  + Reference: <https://youtu.be/upS2HlUj1gI?t=137>
  + TODO: <https://learn.microsoft.com/en-us/office/troubleshoot/access/database-normalization-description>

**Serialization and Persistence**

* ***Serialize***: convert in-memory objects into data to be written or streamed as a string or bytes array, (e.g., Python pickle, JSON, XML).
* ***Desterilize***: convert serialize data back into in-memory objects.

**Types of NoSQL DBs**

* Document, key-value, wide-column, graph.
* TODO:

<https://learn.microsoft.com/en-us/dotnet/architecture/cloud-native/relational-vs-nosql-data>

<https://www.mongodb.com/resources/basics/databases/types>

**Managing Data**

* ***Data Access Object*** (DAO): design pattern(??) that abstracts details of underlying data store.
  + TODO: <https://www.oracle.com/java/technologies/dataaccessobject.html>
  + <https://sourcemaking.com/design_patterns/decorator>
* ***Object Relational Management*** (ORM): data manipulated in object graph and stored in tables.

**SOFTWARE ARCHITECTURE**

[TODO]

[Getting the Basics - Software Architecture Introduction (part 1) (youtube.com)](https://www.youtube.com/watch?v=8UlLgOf20Ho&t=68s)

Behaviour ○ Structure

[The Software Architecture Handbook (freecodecamp.org)](https://www.freecodecamp.org/news/an-introduction-to-software-architecture-patterns/)

[10 Common Software Architectural Patterns in a nutshell | by Vijini Mallawaarachchi | Towards Data Science](https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013)

○ Model View Controller ○ Client-Server ○ Layered

Design patterns

Catalog of Design Patterns (Specifically look up "Mediator" and "Chain of commands")

[10 Design Patterns Explained in 10 Minutes (youtube.com)](https://www.youtube.com/watch?v=tv-_1er1mWI)

Value of architecture increases over time, slow initially

[Software Architecture Guide (martinfowler.com)](https://www.martinfowler.com/architecture/)