UML (Unified Modeling Language)

UML nélkül

- **Kommunikációs szakadék van a megrendelő és a fejlesztők között.**
    - Agilis módszereket be kell vonni.
    - Prototípusokat kell fejleszteni és azokat véleményeztetni.
    - Meg kell találni a közös nyelvet.
- **Kommunikációs szakadék van fejlesztő és fejlesztő között**
    - Tapasztalat, tudásszint béli különbségek lehetnek.

Alapelvek

- Grafikus leírónyelv, ami segít vizualizálni, specifikálni, tervezni és dokumentálni.
- **Kinek jó?**
    - Megrendelő egy folyamatábrát könnyen tud értelmezni.
    - Fejlesztő könnyebben megérti, hogy a másik fejlesztő rendszere hogy működik.
    - Vizuális ábrázolás jobb megértést biztosít.
    - Dokumentáció és így alapos lesz általa.

UML használata

- UML egy szigorú modellező nyelv.
- **Modellező eszközök:**
    - Microsoft Visio
    - Visual Paradigm for UML
    - Rational Rose

Célok

- Egyszerűsíti a bonyolult struktúrákat.
- Kommunikációs eszközként szolgál.
- Automatizálja a szoftverek előállítását és folyamatokat.
- Segíti a szerkezeti problémák megoldását.
- Javítja a munka minőségét.

Diagramok bemutatása (kategóriák: Deployment / Behavioral / Structural / Implementation).

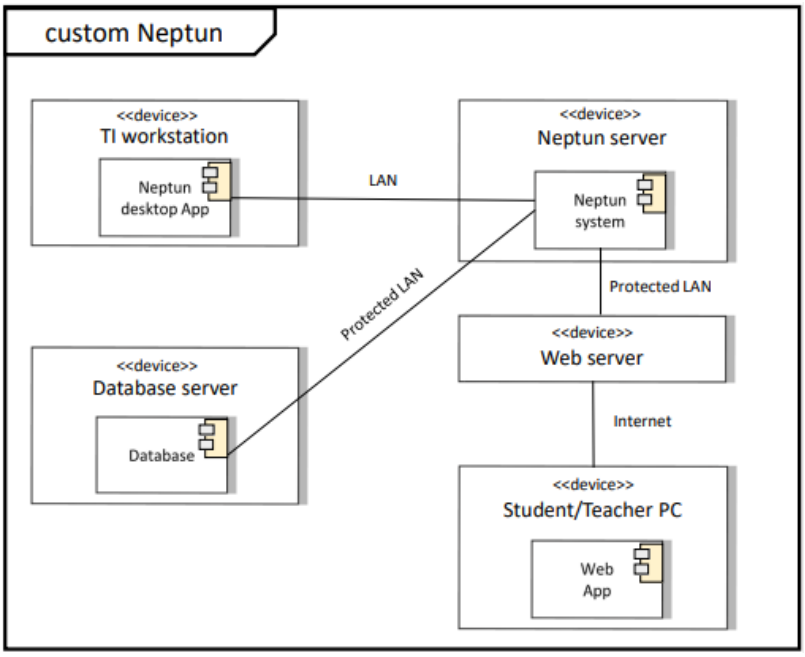
Structural

Structural – Composit Structure Diagram

- A Composite Structure diagram egy UML diagram, ami megmutatja a szoftverrendszer belső szerkezetét.
- Tartalmazza az osztályokat, interfészeket, csomagokat és azok kapcsolatait.
- Segít a felhasználónak látni, hogy mi van egy objektumon belül és hogyan illeszkednek össze a különböző tulajdonságok.
- A részek (**parts**) a strukturált osztályozó (**classifier**) által tulajdonolt egy vagy több példányt jelentik.
- A csatlakozók (**connectors**) a részek közötti kommunikációt jelölik.
- A portok (**ports**) az osztályozó példány és annak környezete közötti interakciós pontokat jelölik.
- A kollaboráció (**collaboration**) az osztályozók közötti interakciók leírására szolgál.

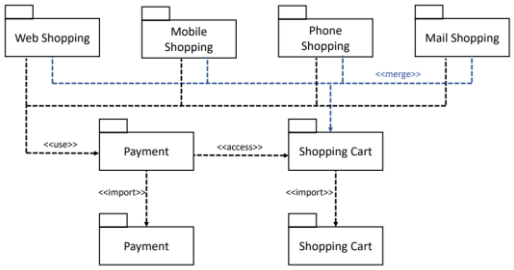
Structural – Deployment Diagram

- A rendszer futásának elemeit bemutató diagram.
- **A működtető elemek lehetnek**
    - Számítógépek
    - Hálózati csomópontok
    - Egyéb környezetek (VM, konténer)
- **Akár a fejlesztési fázis első diagramja is lehet**
    - Ha a környezet már készen van (új szoftvert kell írni meglévő környezetre)
    - **Új rendszernél a részletes tervezéskor használjuk.**



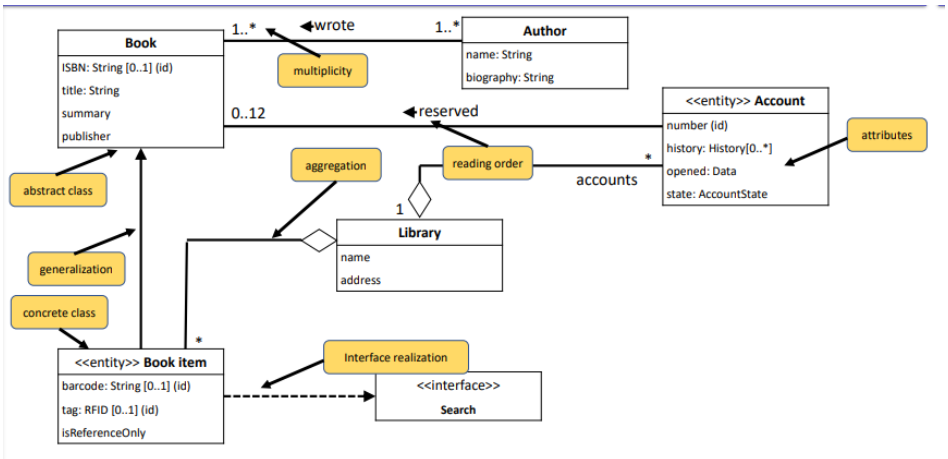
Structural – Package Diagram

- **Package diagrammon ábrázoljuk a különböző DLL-eket, rétegeket és a köztük való összevonásokat, projekt függőségeket.**
- Felépítés, tartalmazás, függőség (Amiket csomagolni lehet pl. egy DLL-be)
- **Kapcsolatok**
    - **package**: Maga a Class Library (DLL)
    - **package merge**: <<merge>>
    - **usage dependency**: <<use>>
        - Függőség, mert lehet, hogy egy másik package is használhatja
    - **private import:** <<access>>
        - Projekt referenciaként használ egy másik package-t.
        - Szigorúbb kapcsolat
    - **public import:** <<import>>
        - Lazább kapcsolat



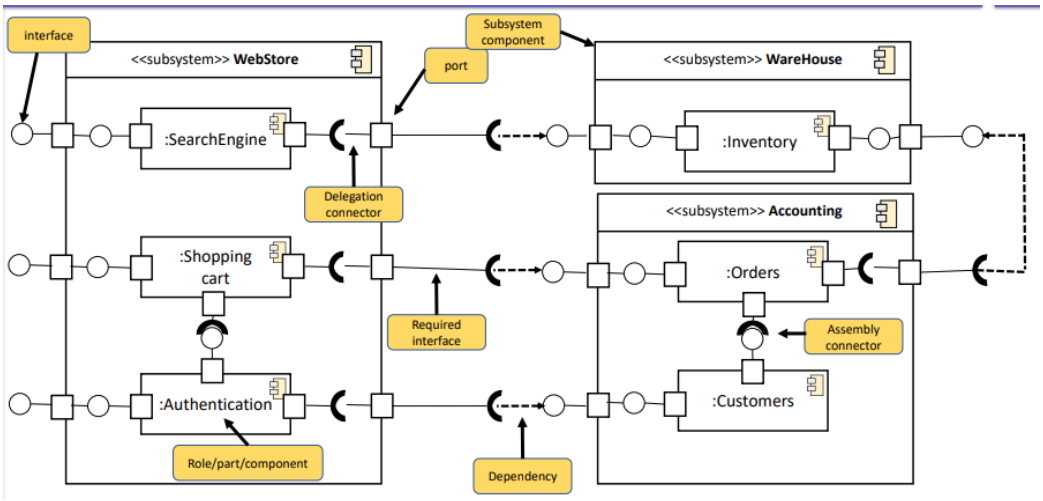
Structural - Class Diagram

- Osztályok vagy konkrét objektumok ábrázolása
    - Adattagok, metódusok
- Korai tervezéskor még nem készül el, ilyenkor a Package diagram interfészei elegek.
- **Multiplicitás**
- **Aggregáció**
- **Reading order**
- **Generalization**
- **Concrete class**
- **Interface realization**
- **Attributes**



Structure – Component Diagram

- Alrendszereket tartalmaz, azon belül milyen osztályok vannak és ezeknek milyen kapcsolataik vannak.
- Abban különbözik a Package diagramtól, hogy külső szolgáltatásokat is jelölhetünk benne.
- **Subsystem component:** Alrendszer
- **Component:** Komponens, részek
- **Interface:** körökkel ábrázoljuk
- **Port:** Négyzettel ábrázoljuk, várunk paramétert konstruktoron interfésszel
- **Delegation connector:** Kapcsolat
- **Required interface:** Vár interfészt
- **Assembly connector:** Alrendszerekben komponensek közötti kapcsolat
- **Dependency:** Függőség, Dependency Injection



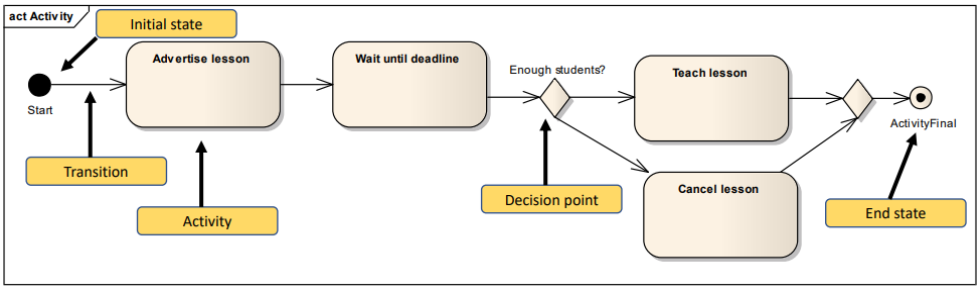
Behavioral

Behavioral – Use Case Diagram

- **Célja:** Megrendelővel való egyeztetés, hogy pontosan milyen szerepköröket, funkciókat képzelt el, és ezeket a funkciókat melyik szerepkörrel lehet igénybe venni.
- **Aktor és használati eset közötti megfeleltetés.**
- **Van öröklődés:**
  - Aktorok között
  - Használati esetek között
- **Tartalmazás/Kibővítés**
  - Használati esetek között

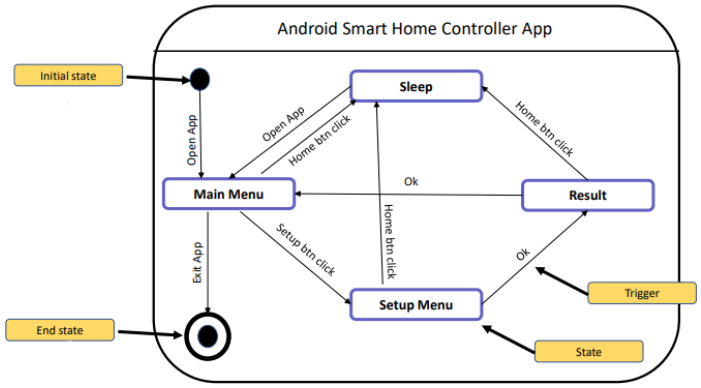
Behavioral – Activity Diagram

- Use-case utáni következő lépés.
- **Célja**:
  - Rendszer folyamat lerajzolása.
  - Leírja az egyik tevékenységtől a másikig tartó sorrendet.
  - Leírja a rendszer párhuzamos, elágazó és egyidejű folyamatát.
- A lépésenkénti tevékenységek és műveletek munkafolyamatainak grafikus ábrázolása.
- **Használati eseteknél a cél:**
  - Belső folyamatok ábrázolása
  - Egymás után következőségek ábrázolása
- 1 használati eset = 1 Activity diagram
- Egy diagrammon belül más use case-ek is előjöhetnek.
- **Initial state, End state:** kezdő és vég állapot
- **Transition:** Átmenet
- **Activity:** Aktivitás
- **Decision point:** Döntési pontok



Behavioral – State Machine Diagram

- Rendszer/Objektum állapotainak egymás után következőségét ábrázolja.
- Irányított gráf, aminek csomópontjai a logikai állapotok, amik élei a köztük lévő átmenetek.
- A végrehajtható műveletek az állapotokhoz és az átmenetekhez is tartozhatnak.
- **Probléma, hogy egy idő után átláthatatlan lesz a diagram.**
    - State transition
- **Initial, end state:** kezdő és vég állapot
- **State:** az adott állapot
- **Trigger:** átviszi egyik állapotból a másikba

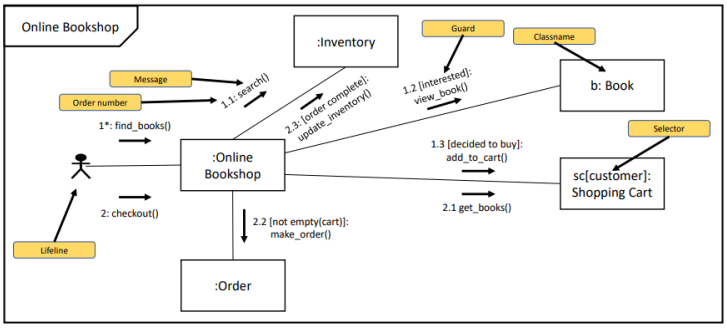


Behavioral – Interaction Diagram

- Egy folyamat résztvevői (aktorok vagy modulok vagy rétegek) közötti kommunikációs folyamatokat ábrázoljuk
- Három különböző diagramot különböztet meg:
    - **Communication diagram:** résztvevők és sorrend
    - **Timing diagram:** időzítési információk, megkötések és állapotok is
    - **Sequence diagram:** ciklusok, feltételek és élettartamok is.
- Mindhárom típus ugyanarra a célra való csak más mélységben mutatja be a kommunikációs folyamatokat.

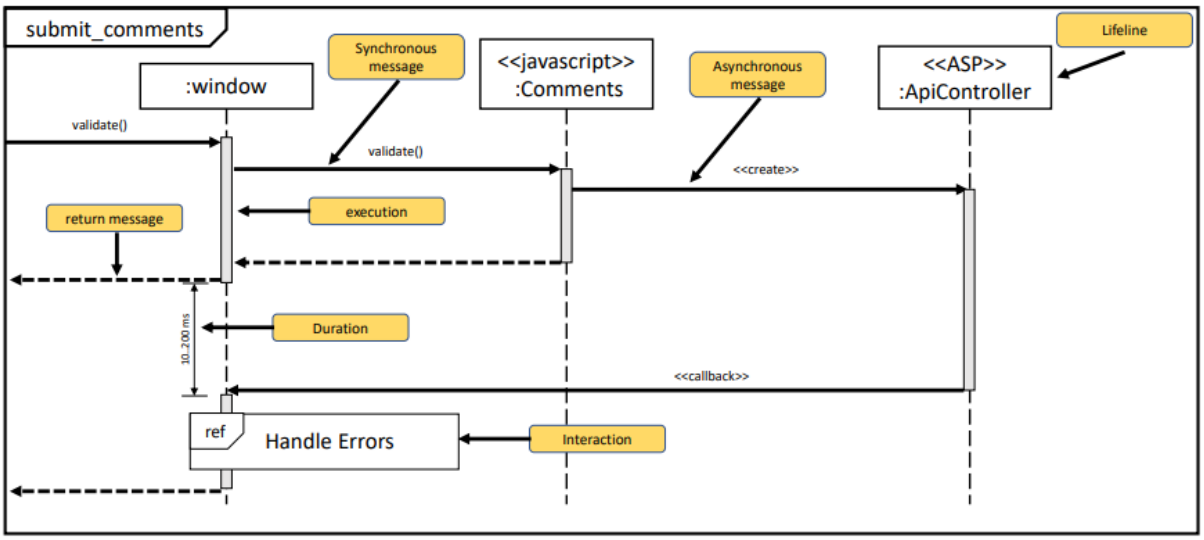
Behavioral – Communication Diagram

- **Lifeline:** Felhasználó hogyan tud az osztályokkal kommunikációba kerülni.
- **Message:** Vmilyen metódus végrehajtása
- **Order number:** Sorrendet írja le
- **Guard:** „Őrszem”, csak akkor hajtódjon végre ha xy
- **Classname:** Osztály neve
- **Selector:** Paramétereket adhatunk át, jelzésre jó



Behavioral – Sequence Diagram

- Validációk, majd azoknak válasza, de validáció közben végrehajtódhat egy másik folyamat.
- **Lifeline:** Pl egy ApiController
- **Synchronous message:** Visszaad üzenetet
- **Asynchronous message:** Nem ad vissza üzenetet
- **return message:** Vissza ad egy értéket vagy üzenetet



Behavioral – Timing Diagram

- **Lifeline:** „Életvonal”
- **Timeline:** Idővonal", ami az lifeline-on belül mutatja, hogy az ő timeline-ja az a különböző állapotok között hogyan változik.
- **Synchronous message:** Kérést küldünk az egyik lifeline-ból a másikba egy kérést, és az időzítések miatt nincsenek **asynchronous message**-k.
- **Reply:** Válasz
- **State change:** Függőleges vonalak az állapot változások.
- **Event/stimulus:** Címke, ami vmilyen triggert jelöl.