WEBES ALKALMAZÁSOK TERVEZÉSE, FEJLESZTÉSÉNEK MENETE

Tarcsi Ádám

OKJ vizsga: 1188-06 Web-alkalmazás tervezés

- Nemzeti Munkaügyi Hivatal, Szakképzési és Felnőttképzési Igazgatóság: www.nive.hu
- Szakmai és vizsgakövetelmények: https://www.nive.hu/index.php?tip=szvk&option=com_jumi&view=application&fileid=7&kulcsszo=web&keres=Keres
- Szóbeli vizsgafeladatok:
- https://www.nive.hu/index.php?tip=1&option=com_jumi&view=application&fileid=11&kulcsszo=web-alkalmaz%C3%A1s+tervez%C3%A9s&kereses=Keres%C3%A9s
- Korábbi írásbeli feladatok: http://www.forrai.hu/hu/iskola/page/101020-1188-06-web-alkalmazas-tervezes

Tervezés


I HIRED ALL OF YOU BECAUSE THE PROJECT WILL TAKE 300 MAN DAYS TO COMPLETE.
THERE ARE 300 OF YOU, SO I WANT YOU TO FINISH BY FIVE O'CLOCK AND CLEAN OUT YOUR DESKS. YOU'RE ALL FIRED.
IF IT TAKES MORE THAN ONE MEETING TO MANAGE A PROJECT, I LOSE INTEREST.
www.dilbert.com
scottadams@aol.com
9-3-07
©2007 Scott Adams, Inc./Dist. by UFS, Inc.

Bevezetés

- Web site vs. Web alkalmazás
- Webtechnológia (Web engineering) vs. szoftvertechnológia
- Webes architektúra
- KKV vs. nagyvállalati környezet

Szoftverfejlesztés tevékenységei

- Elvárások elemzése és specifikáció
- Tervezés
- Implementálás
- Kipróbálás, validálás
- Szoftverevolúció: karbantartás, fejlesztés

Kiegészítő tevékenységek

- Projekt menedzsment
- Verzió kezelés / verzió követés
- Erőforrás menedzsment
- Minőségbiztosítás
- Terméktámogatás
- Projekt értékelés, fejlesztési folyamat továbbfejlesztése

Feladatkörök

- Megrendelő
- Szervezői, tervezői feladatok: rendszerszervezés, szoftver architect, projektvezetés, marketing, stb.
- Web-fejlesztés: kilens, szerver oldalon
- Web-design
- Adatbázis: adminisztráció, fejlesztés
- Tesztelés
- Üzemeltetés

Szoftverfolyamat-modellezés

Szoftverfolyamat modellek

- Vízesés
- Boehm féle spirál
- Gyors prototípus
- Inkrementális (evolúciós)
- Újrafelhasználás orientált (komponens alapú)
- V
- OMT (Object Modelling Technique)
- RUP (Rational Unified Process)
- Agilis módszerek: SCRUM, Extreme Programming (XP), Lean, stb.

Vízesés modell

Vízesés modell

- **Követelmények felmérése**: igények, elvárások meghatározása, összefoglalása. Jelen állapot (helyzetfelmérés), probléma, elérendő cél definiálása.
- **Rendszerjavaslat**: Alternatívák, szükséges erőforrások, költségek megválaszolása, alapvető lépések a projektterv összeállításához. A rendszerjavaslat az első olyan dokumentum, amelyet a megrendelő megkap, melyből az eddig végzett munkát megítélheti, a fejlesztés perspektíváiról képet alkothat.
- **Rendszerspecifikáció**: rendszertervezőnek szól. Input-output típusok, fájlok definiálása, nagyvonalú rendszerterv (hardver és szoftveres), adatstruktúra, interfész-definíció. Döntések, azok bemutatása (pl.: vásárolt v. fejlesztett részek), stb.
- **Logikai és fizikai tervek**: szoftver és adatbázis. A lépések konkrét definiálása. Megvalósítási terv (idő, erőforrások, ember, pénzügyi források, hogyan érjük el a célokat) és rendszerterv elkészítése. Architektúra, hálózati topológia, funkcióspec., navigációs és oldal desing-ek, adatterv - DB diagram, osztálydiagrammok.

Vízesés modell

- Implementáció = megvalósítás
- Szoftverdalidáció = tesztelés
- Rendszerátadás (élesbe helyezés → online)
- Üzemeltetés, karbantartás → visszamutat a korábbi állapotokra.

Logikai és fizikai rendszerterv

- Logikai rendszerterv: a felmerült probléma megoldására kidolgozott működési-, szervezeti-, adat- és folyamatmodell, mely többféle eszközkörnyezetben megvalósítható módon, logikai szinten van megfogalmazva.
- Fizikai rendszerverv: egy logikai rendszerterv alapján több fizikai is készíthető más-más hardver/szoftver környezetre is tervezhető, megvalósítható. Konkrét eszközbázisra, adott környezetre épül.

Logikai tervezés

- A rendszer működési logikájának tervezése
- Folyamatok (funkciók) tervezése
- Adattervezés
- Felhasználói interfészek tervezése

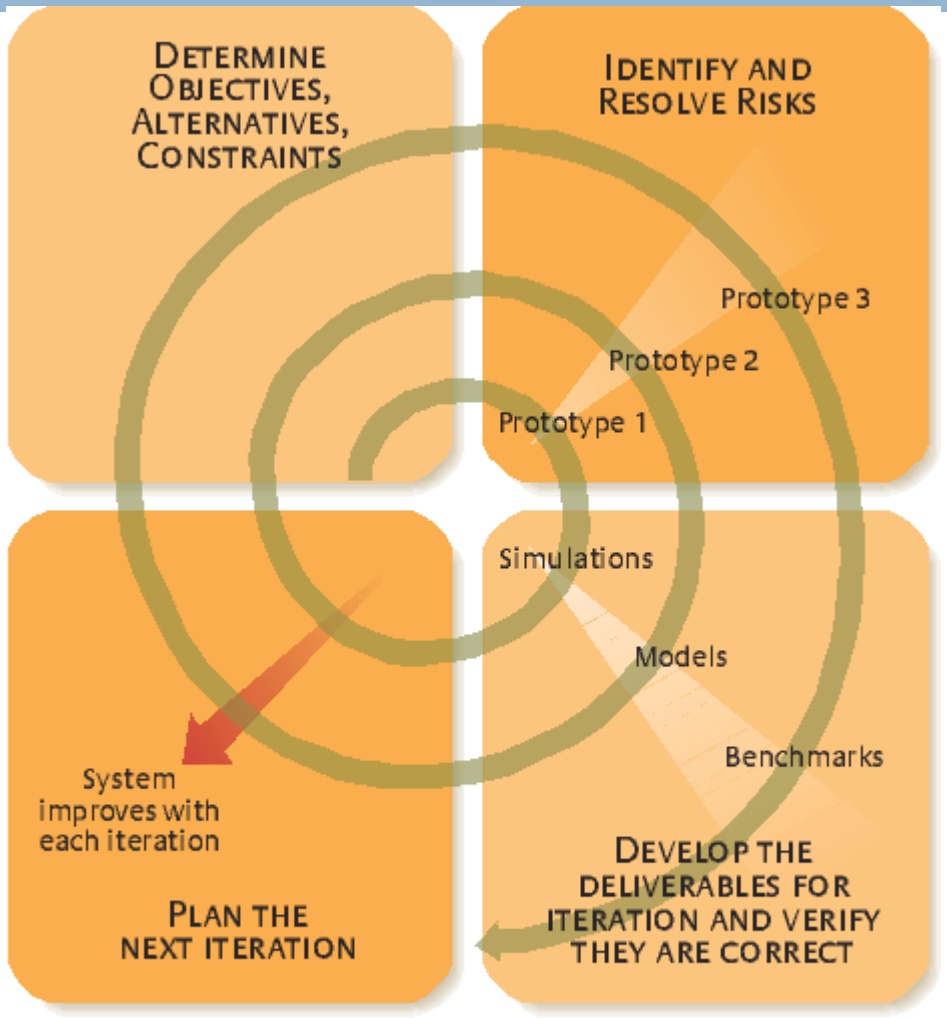
Fizikai tervezés

- Adatterv
- Rendszerspecifikációk (fejlesztési, futtatási környezet)
- Szoftverarchitektúra (rétegek)
- A rendszer működésének elve
- Programspecifikációk – funkciótervek
- I/O tervek, rendszer interfészek
- Biztonsági terv

Vízesés modell

- A következő fázis addig nem indulhat el, amíg az előző be nem fejeződött. Ez a modell akkor működik jól, ha a követelmények teljesen ismertek.
- **Előny**: Jól menedzselhető és ellenőrizhető. Minden fázisban jól definiált feladatok. Minden fázis jól dokumentálható. Előre jól definiálható követelmények esetén jól alkalmazható.
- **Hátrány**: Nagyon sok probléma csak az utolsó fázisban derül ki, így a javítás nagyon költséges. Korán kell jelentős döntéseket hozni, ez hibás döntésekhez vezethet. Nehéz a rendszert a fejlesztés közben változó követelményekhez igazítani. Sok dokumentációs munkát igényel.

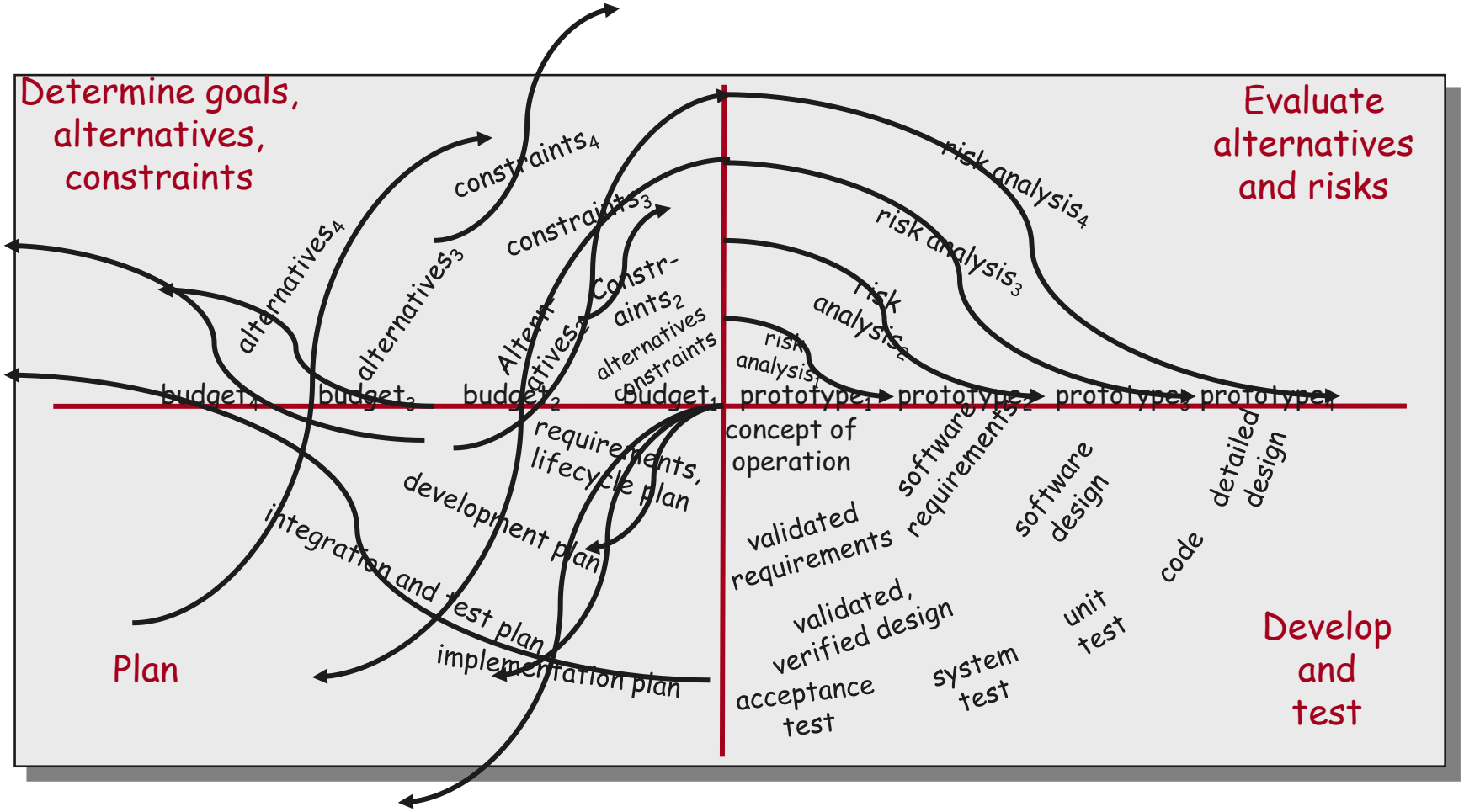
Spirál modell



Adapted from B. Boehm, "A Spiral Model of Software Development and Enhancement," ACM SIGSOFT Engineering Notes 11, no. 4 (1986): 25.

- megvalósíthatóság
- a rendszer követelményeinek meghatározása
- rendszertervezés,
- stb.

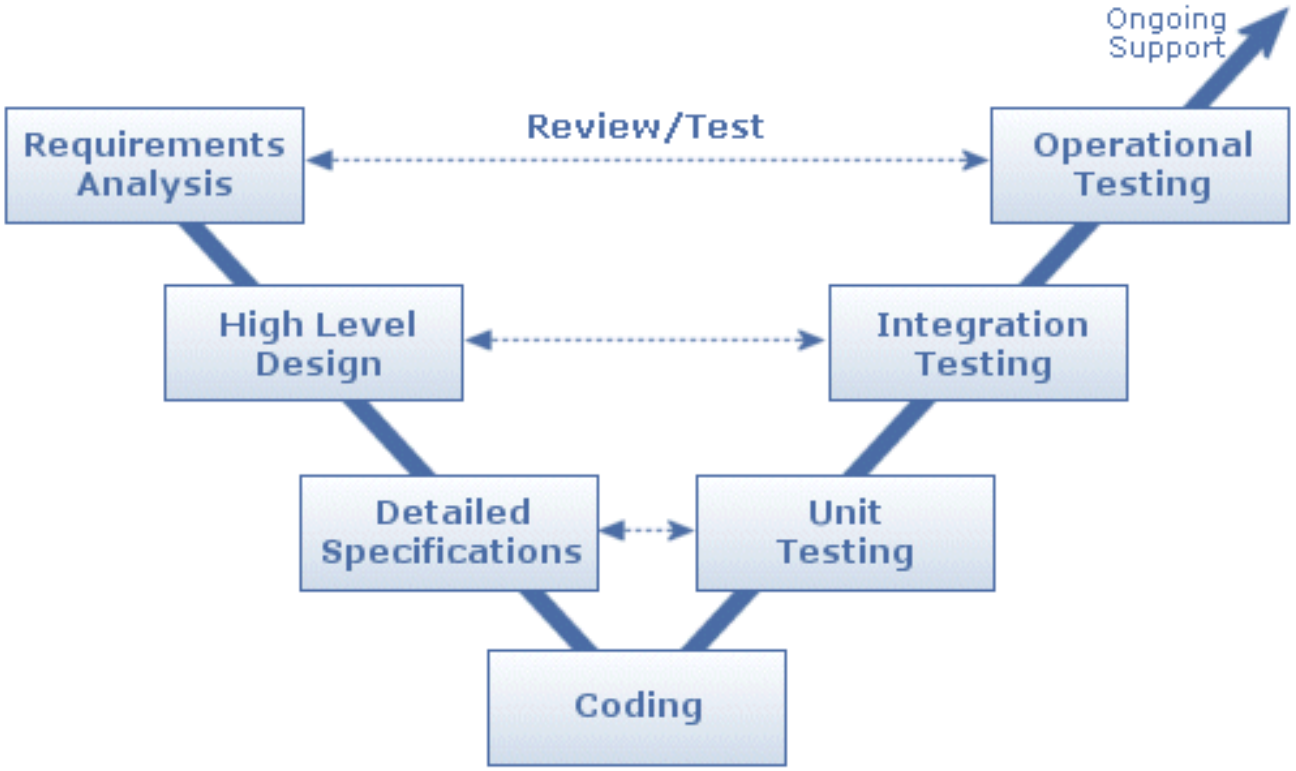
Spirál modell



Spirál modell

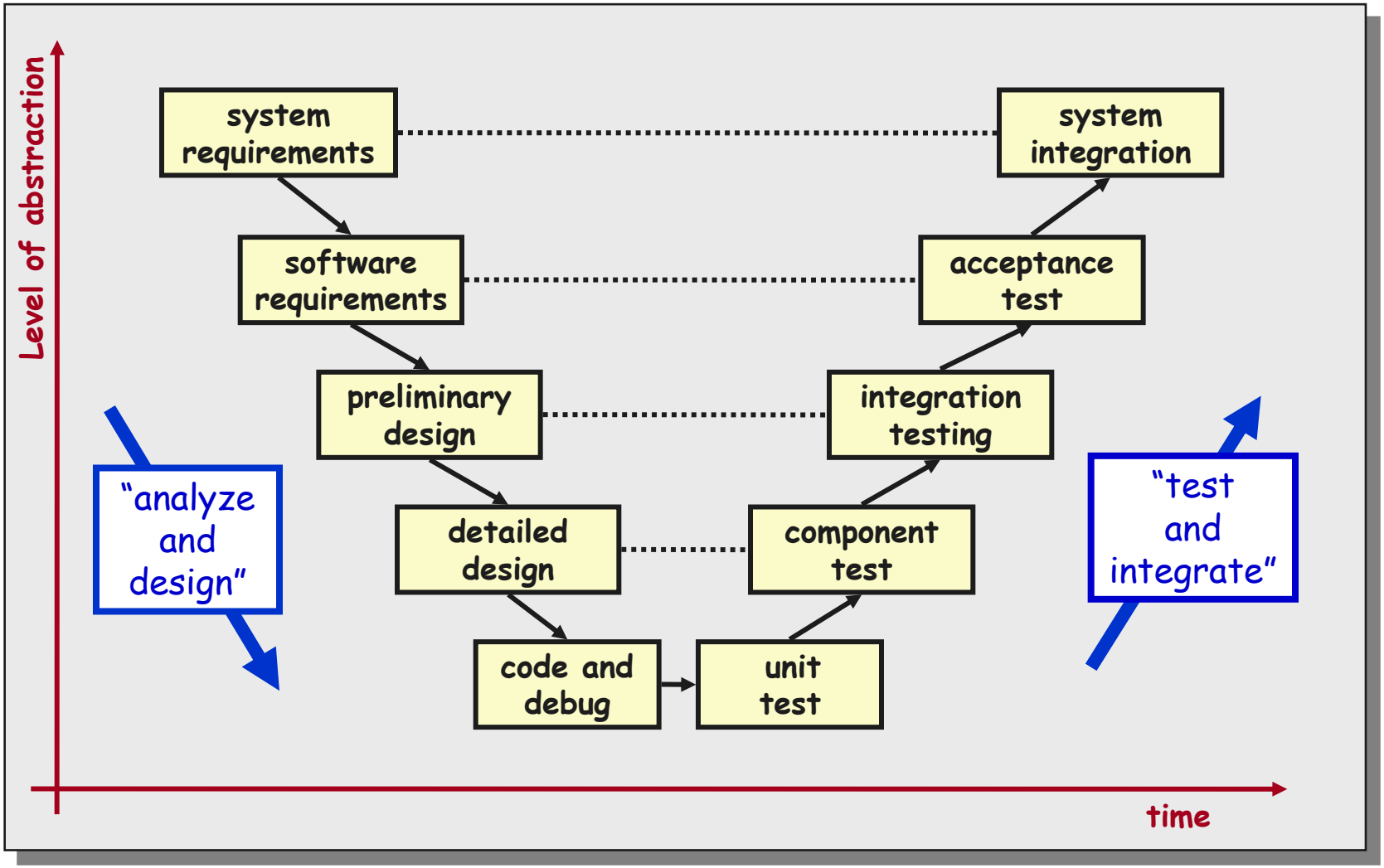
- **Előny**: a kockázati tényezőkkel explicite számol. A spirális modellben nincsenek rögzített fázisok, és felölelhet más folyamatmodelleket is (vízesés, evolúciós, stb.).
- **Hátrányai**: a modell alkalmazása bonyolult, munkaigényes feladat; a párhuzamos foglalkoztatás csak a 3. szektorban lehetséges.

V modell



Forrás: http://softwareandme.wordpress.com/2009/10/20/software-development-life-cycle/sdlc_v_model

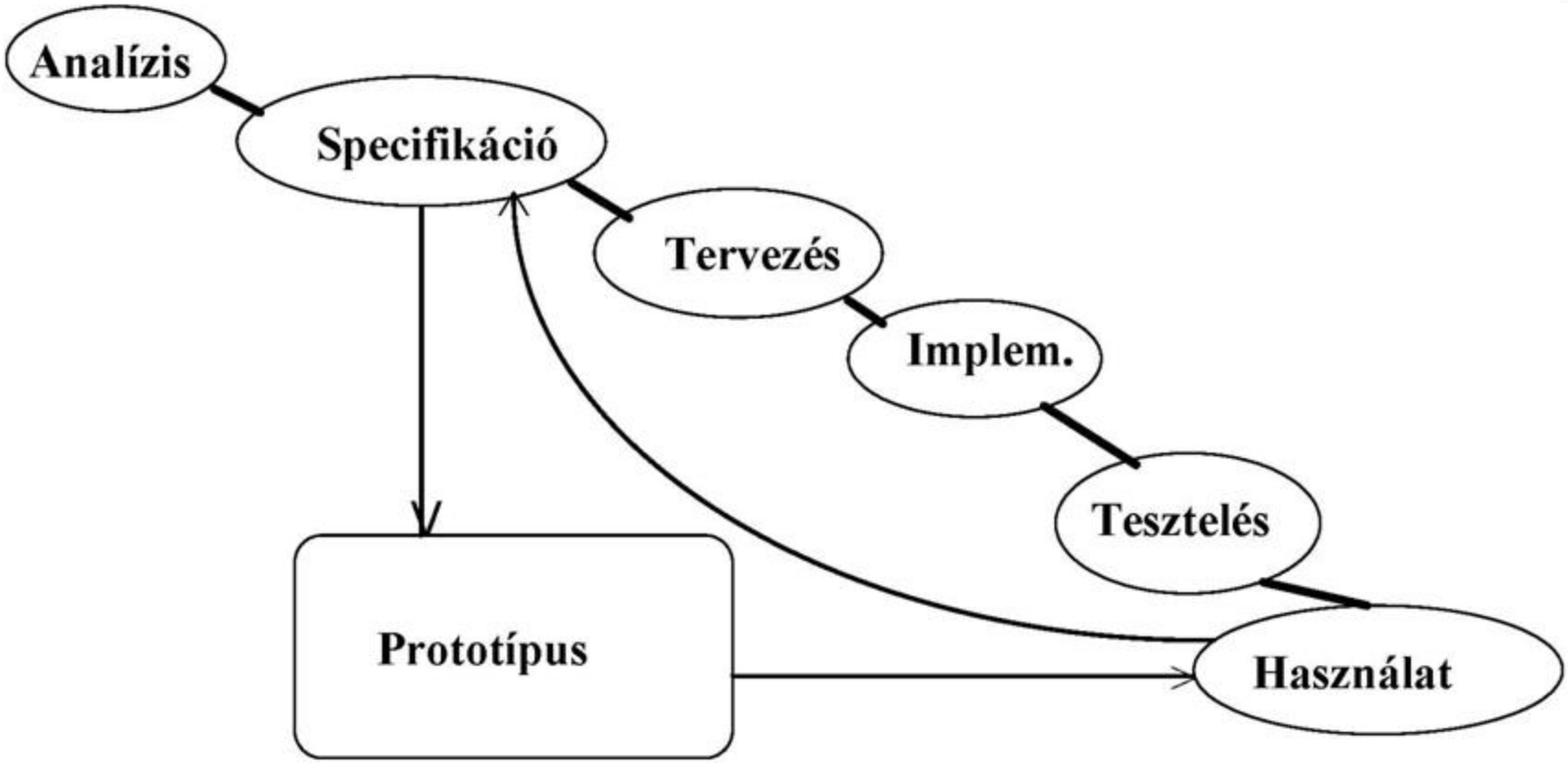
V modell


Level of abstraction
system requirements
software requirements
preliminary design
detailed design
code and debug
unit test
component test
integration testing
acceptance test
system integration
"analyze and design"
"test and integrate"
time

V modell

- Egy módosított vízesés modell.
- Megkülönbözteti a fejlesztésen belül a konstrukciós és a tesztelési fázisokat.
- Definiálja a tesztelés szintjeit.
- Szemlélteti, hogy a tesztelési munka végigköveti a teljes fejlesztési folyamatot.
- Összefüggést tételez fel az egyes konstrukciós fázisok és az egyes tesztelési szintek között.

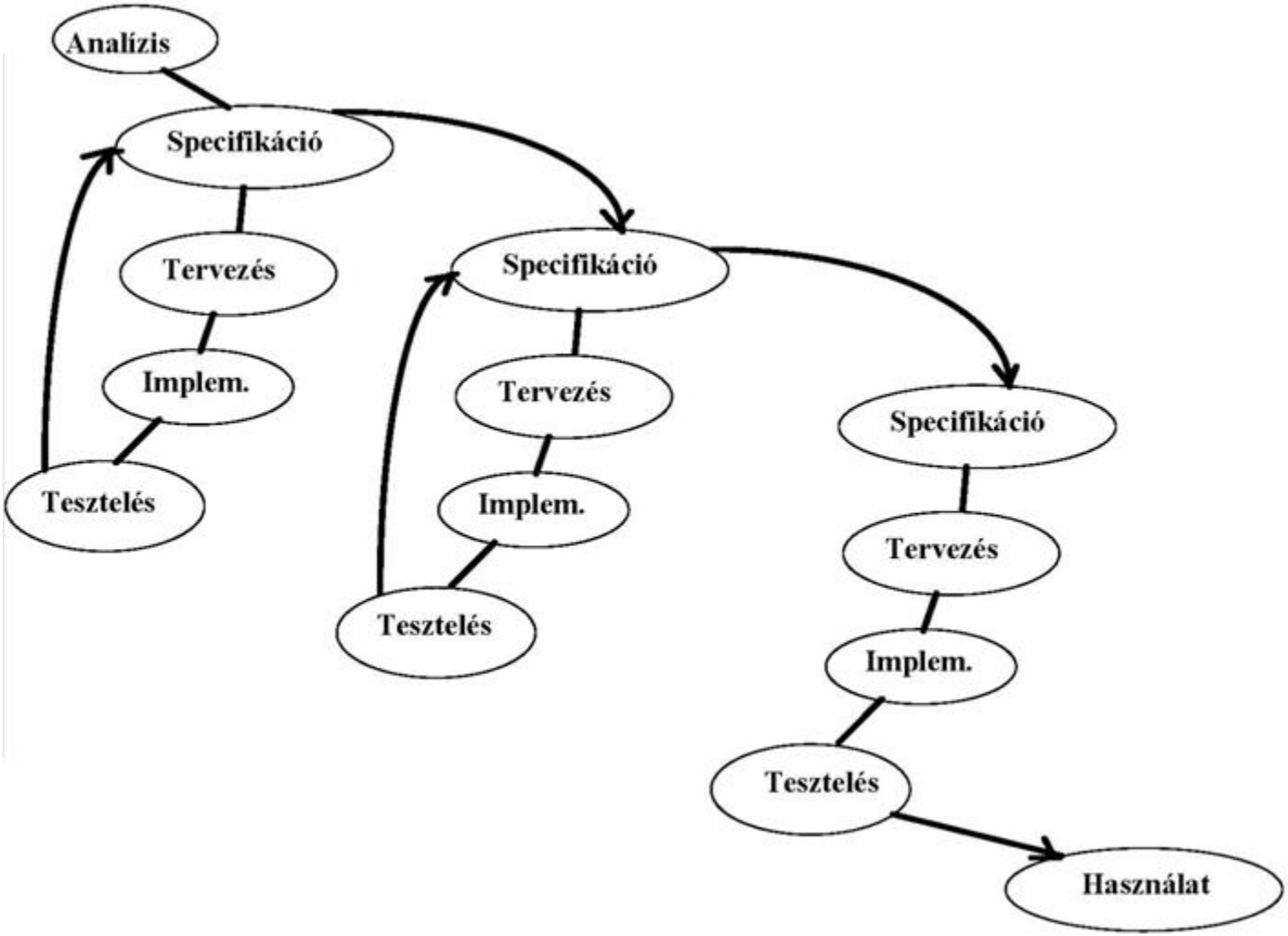
Gyors prototípus modell



Gyors prototípus modell

- Segíti a fejlesztő és a felhasználó kommunikációját.
- Főleg kisebb csoportoknál vált be.
- A teljes fejlesztési folyamatot általában nem fedi le, de jól alkalmazható kiegészítő módszerként.

Inkrementális (evolúciós)



Evolúciós modell

- Ki kell fejleszteni egy kezdeti implementációt (prototípust), azt a felhasználókkal véleményeztetni, majd sok-sok verzión át addig finomítani, amíg megfelelő nem lesz. **Iterációs** modellnek is nevezik. Objektum orientált fejlesztésben gyakran használják.
- Ez a modell a felhasználó kívánságait jobban kielégítő programot eredményez. A kis (<100.000 programsor) és közepes (<=500.000 programsor) rendszerek fejlesztéséhez ideális.
- **Hátrányai**: a folyamat nem látható; a rendszerek gyakran szegényesen strukturáltak; a gyors fejlesztés rendszerint a dokumentáltság rovására megy.

Újrafelhasználás orientált fejlesztés (komponens alapú)

- Komponenselemzés
- Követelménymódosítás
- Rendszertervezés újrafelhasználással
- Fejlesztés és integráció

Komponens alapú modell

- **Előnye:** lecsökkenti a kifejlesztendő részek számát, így csökkenti a költségeket és a kockázatot. Ez általában a kész rendszer gyorsabb leszállításhoz vezet.
- **Hátrányai:** akövetelményeknél hozott kompromisszumok elkerülhetetlenek, és ez olyan rendszerhez vezethet, ami nem felel meg a felhasználó valódi kívánságának.

Egyéb modellek, módszertanok

- Agilis
- XP – eXtreme Programming
- SCRUM
- Lean
- MDA – Model Driven Architecture
- MDD- Model Driven Design
- TDD – Test Driven Design
- BDD – Business Driven Design
- …

Tervezési eszközök, módszertanok

CASE eszközök

- Computer-Aided Software Engineering
- **Követelményspecifikáció:** grafikus rendszermodellek, üzleti és domain
- **Elemzés/tervezés során:** adatszótár kezelése, mely a tervben található egyedekről és kapcsolataikról tartalmaz információt; felhasználói interfész generálását egy grafikus interfész-leírásból, melyet a felhasználóval együtt készíthetünk el.; a terv ellentmondás mentesség vizsgálata
- **Implementáció során:** automatikus kódgenerálás (Computer Aided Programming - CAP);verziókezelés
- **Szoftvervalidáció során:** automatikus teszt-eset generálás, teszt-kiértékelés, -dokumentálás
- **Szoftverevolúció során:** forráskód visszafejtés (reverse engineering); régebbi verziójú programnyelvek automatikus újrafordítása újabb verzióba.

CASE eszközök

- Automatikus dokumentumgenerálás;
- Projektmenedzsment támogatás (ütemezés, határidők figyelése, erőforrás-tervezés, költség- és kapacitásszámítás, stb.)
- A CASE-eszközök korai pártolói azt jósolták, hogy a szoftverek minőségében és a termelékenységben nagyságrendi javulást okoznak ezek az eszközök, de valójában csak 40% körüli a javulás.

UML

UML

- Unified Modeling Language
- Egységes modellező nyelv
- 2.1.2 (ISO/IEC 19505) http://www.uml.org
- Object Management Group
- Eric J. Naiburg, Robert A. Maksimchuk: UML földi halandóknak. Kiskapu Kiadó, Budapest, 2006.

UML

- Dokumentálható
  - A szoftverrel szemben támasztott követelmények
  - A szoftver felépítése
  - A szoftver működése
- Grafikus elemek
- Nem programozási nyelv
- Nem módszertan
- „Csak” segédeszköz

Diagram típusok

Szerkezeti diagramok:

- **Osztálydiagram (class)**
- **Objektumdiagram (object)**
- **Csomagdiagram (package)**
- **Összetevő diagram (component)**
- **Összetett szerkezet diagram (composite stucture)**
- **Kialakítás diagram (deployment)**

Viselkedési diagramok:

- **Tevékenység diagram (activity)**
- **Használati eset vagy feladat diagram (use-case)**
- **Állapotautomata vagy állapotgép diagram (state machine)**
- **Kölcsönhatási diagramok:**
  - **Sorrend diagram (sequence)**
  - **Kommunikációs diagram (communication)**
  - **Időzítés diagram (timing)**
  - **Kölcsönhatás áttekintő diagram (interaction overview)**

Diagram típusok

- **A szerkezeti (statikus) és a viselkedési (dinamikus) diagramok.** A szerkezeti diagramok nem törődnek az időbeli változással, hanem a modellezett rendszer állapotát egy adott időpillanatban mutatják be.
- **Viselkedési diagramok** folyamatában, változásában mutatják ugyanazt a modellezett rendszert.

Diagram típusok

- **Osztálydiagram:** az UML modellezésben leggyakrabban használt diagramfajta. A rendszerben található állandó elemeket, azok szerkezetét és egymás közötti logikai kapcsolatát jeleníti meg.
- **Objektumdiagram:** a rendszer egy adott időpontban érvényes pillanatképét határozza meg. Az osztálydiagramból származtatjuk.
- **Csomagdiagram:** a csomagok olyan modellelemek, amelyek más modellelemek csoportosítására szolgálnak, és ezeket valamint a köztük lévő kapcsolatokat ábrázolja ez a fajta diagram.

Diagram típusok

- **Összetevő diagram:** az összetevő vagy komponens a rendszer fizikailag létező és lecserélhető része, feltéve, hogy az új komponens csatlakozási felülete (interfésze) megegyezik a régivel. (Mint a LEGO-kockák.) Ez a diagram főleg implementációs kérdések eldöntését segíti. A megvalósításnak és a rendszeren belüli elemek együttműködésének megfelelően mutatja be a rendszert.
- **Összetett szerkezeti diagram:** A modellelemek belső szerkezetét mutatja.

Diagram típusok

- **Kialakítás diagram:** A fizikai (kész) rendszer futásidejű felépítését mutatja. Tartalmazza a hardver és a szoftverelemeket is.
- **Tevékenységdiagram:** A rendszeren belüli tevékenységek folyamatát jeleníti meg. Általában üzleti folyamatok leírására használjuk.
- **Használati eset/feladat diagram:** A rendszer viselkedését írja le, úgy, ahogy az egy külső szemlélő szemszögéből látszik.
- **Állapotautomata diagram**: Az objektumok állapotát és az állapotok közötti átmeneteket mutatja, valamint azt, hogy az átmenetek milyen esemény hatására következnek be.

Diagram típusok

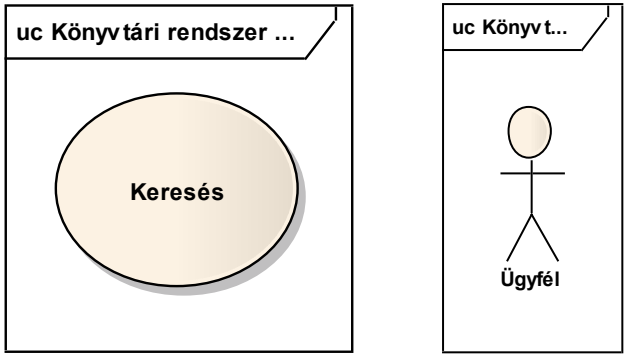
- **Kommunikációs diagram:** Az objektumok hogyan működnek együtt a feladat megoldása során, hogyan hatnak egymásra.
- **Sorrenddiagram:** Az objektumok közötti üzenetváltás időbeli sorrendjét mutatja.
- **Időzítés diagram:** A kölcsönhatásban álló elemek részletes időinformációit és állapotváltozásait vagy állapotinformációit írja le.
- **Kölcsönhatás áttekintő diagram:** Magas szintű diagram, amely a kölcsönhatás-sorozatok közötti vezérlési folyamatról ad áttekintést.

Használati eset (use case) diagram

- Leggyakrabban a követelményelemzés és a specifikáció során alkalmazzák
- A rendszer viselkedését írja le, ahogyan az egy külső szemlélő szemszögéből látszik

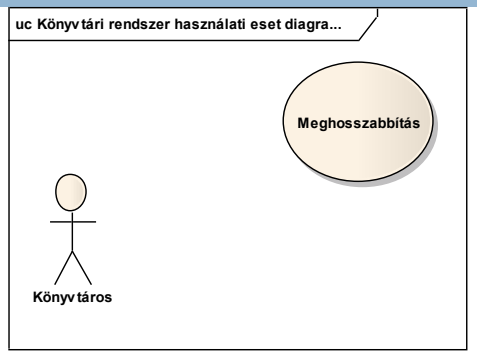
Összetevői

- Használati eset
- Szereplő
- Rendszerhatár

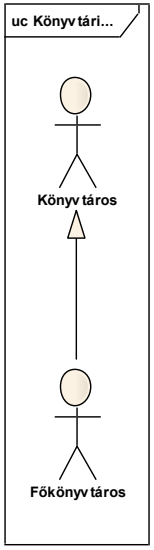


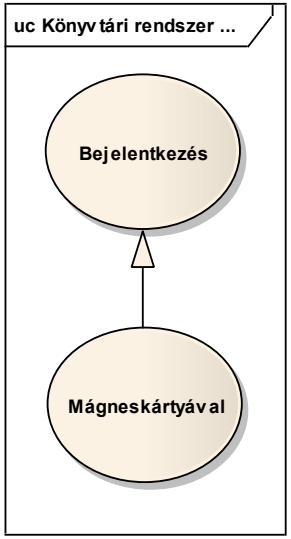
Kapcsolatok

- Asszociáció



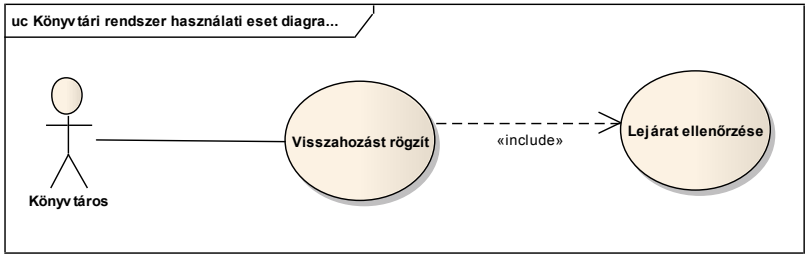
- Általánosítás



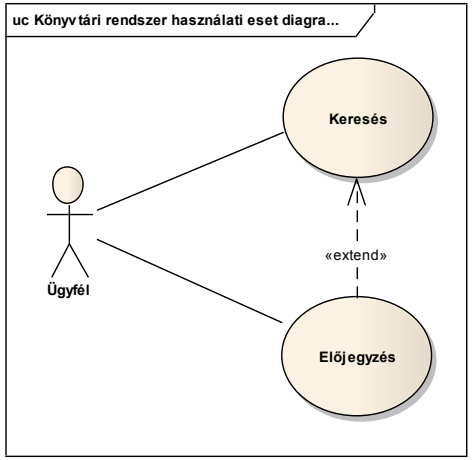


Kapcsolatok

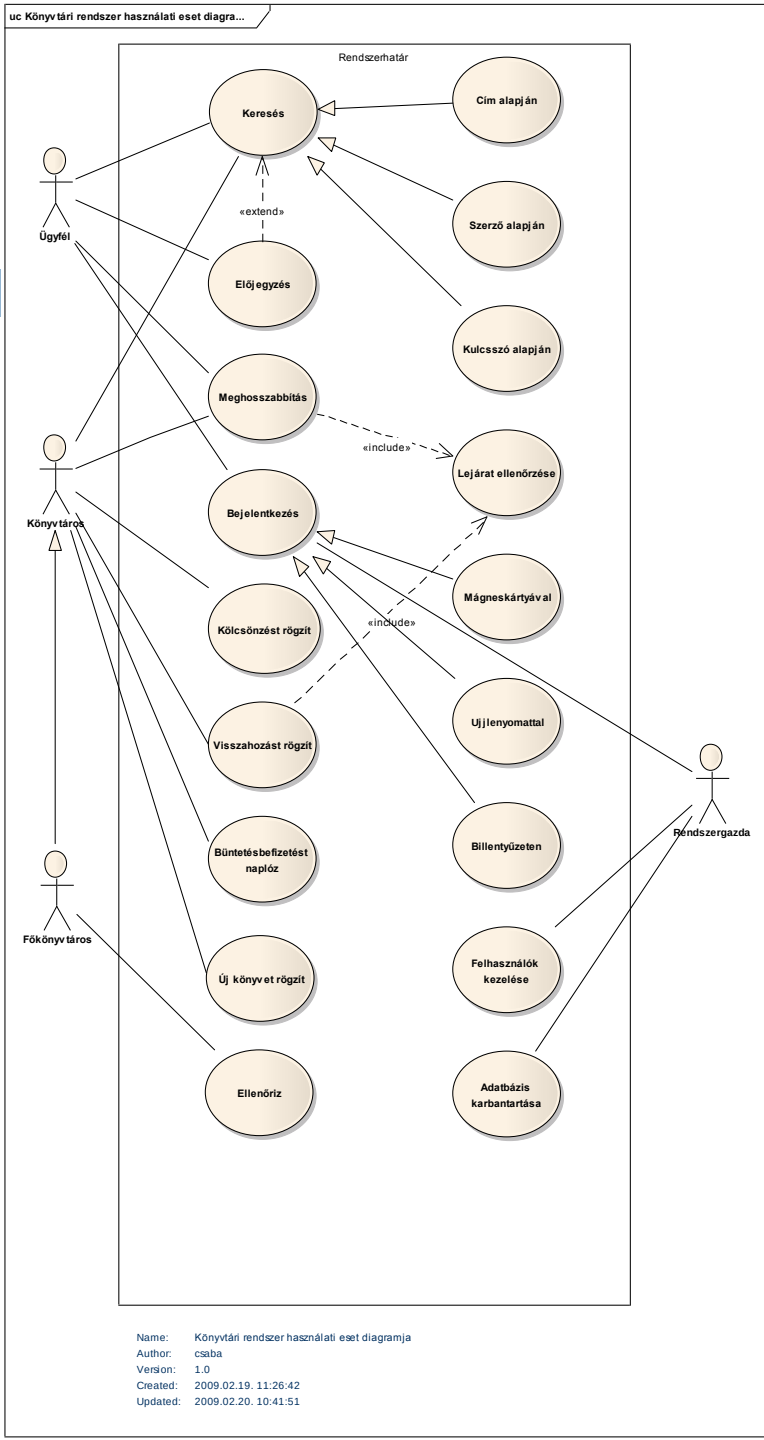
- <<include>>



- <<extend>>



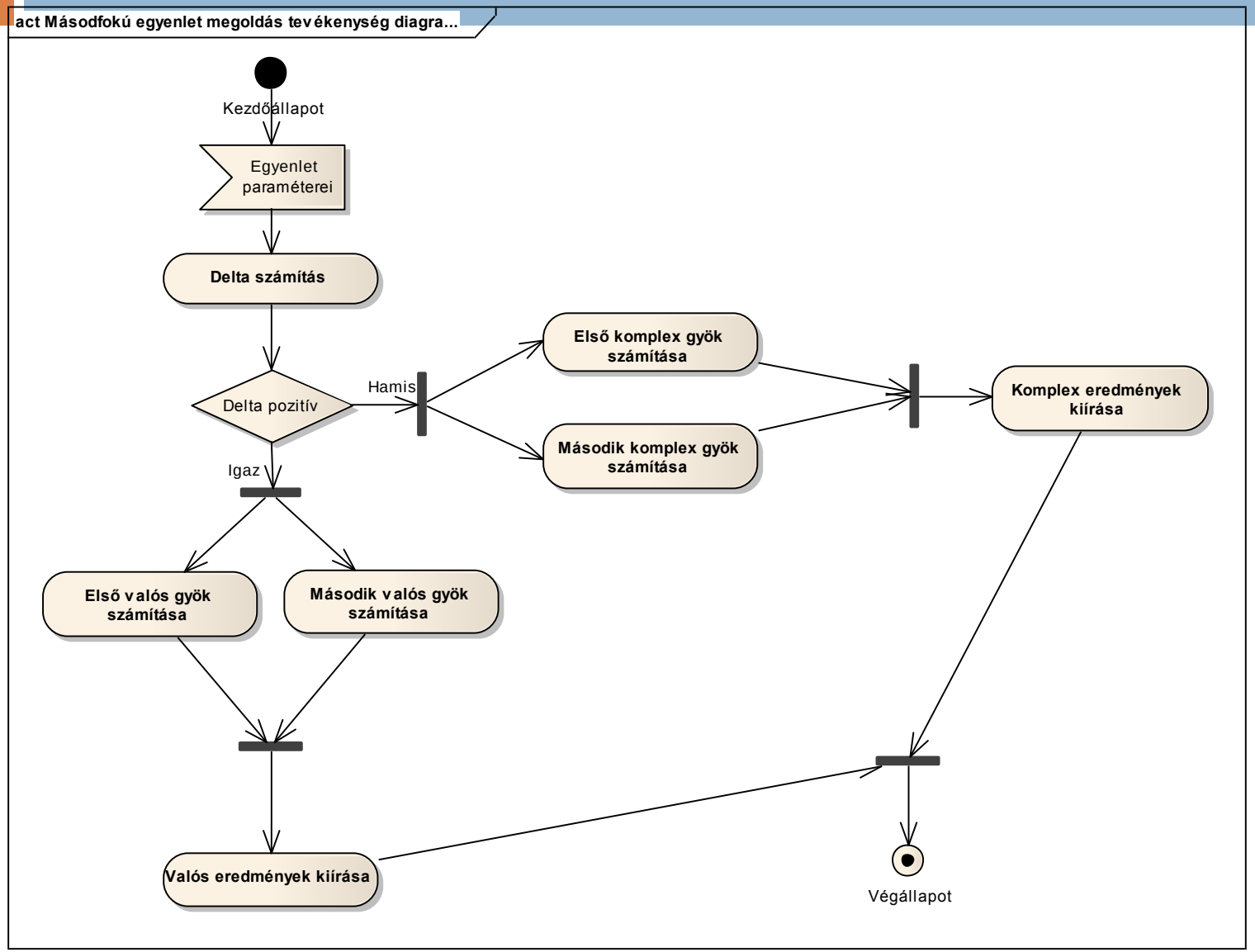
Példa


uc Könyvtári rendszer használati eset diagra...
Rendszerhatár
Keresés
Cím alapján
Szerző alapján
Kulcsszó alapján
«extend»
Előjegyzés
Meghosszabbítás
«include»
Lejárat ellenőrzése
Bejelentkezés
Mágneskártyával
Kölcsönzést rögzít
«include»
Ujjlenyomattal
Visszahozást rögzít
Billentyűzeten
Büntetésbefizetést naplóz
Új könyvet rögzít
Felhasználók kezelése
Ellenőriz
Adatbázis karbantartása
Ügyfél
Könyvtáros
Főkönyvtáros
Rendszergazda
Name: Könyvtári rendszer használati eset diagramja
Author: csaba
Version: 1.0
Created: 2009.02.19. 11:26:42
Updated: 2009.02.20. 10:41:51

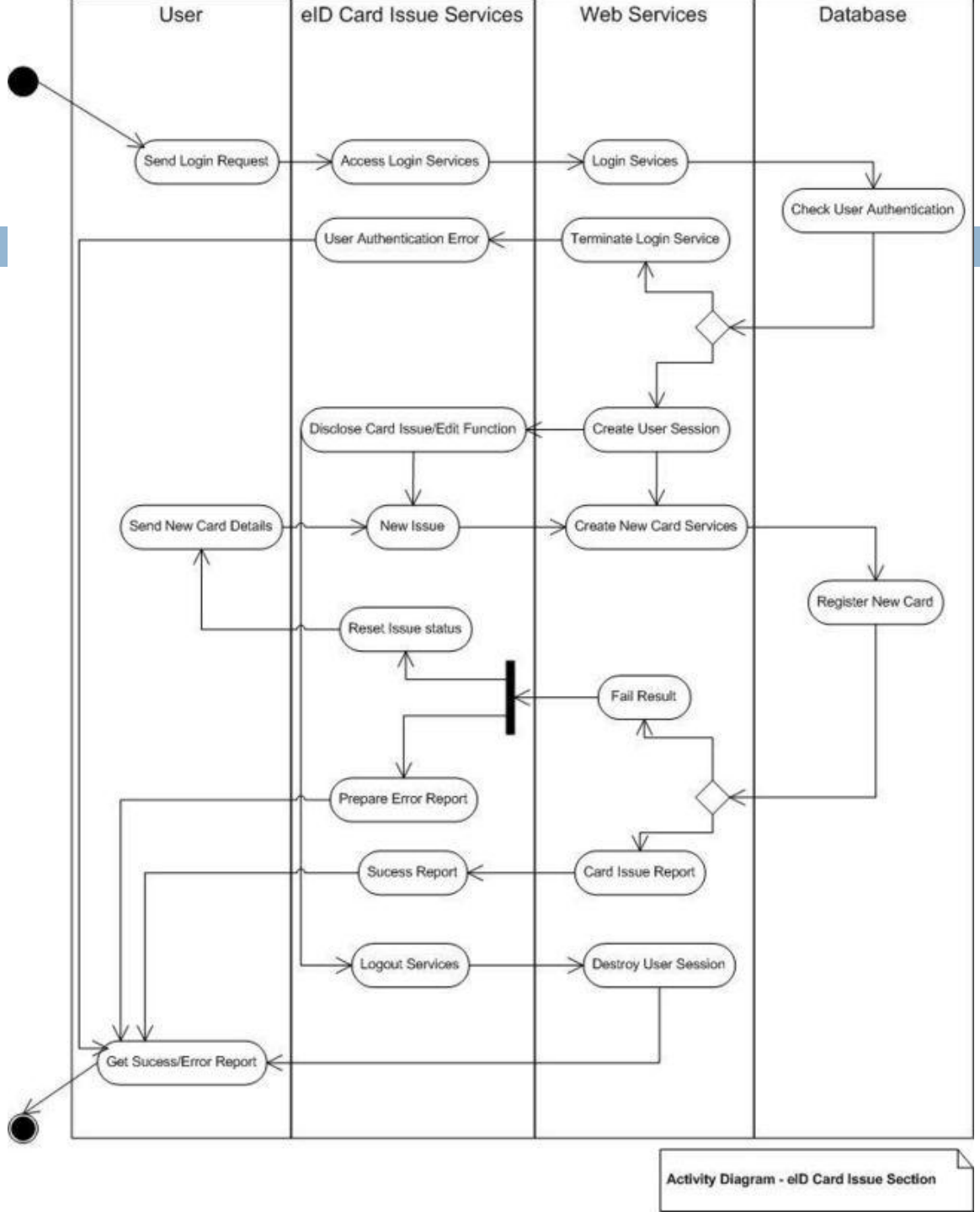
Tevékenység (activity) diagram

- A probléma megoldásának a lépéseit szemlélteti, a párhuzamosan zajló vezérlési folyamatokkal együtt
- Hasznos az üzleti vagy munkafolyamatok modellezésére, használati esetek vagy konkrét algoritmusok lefutásának leírására

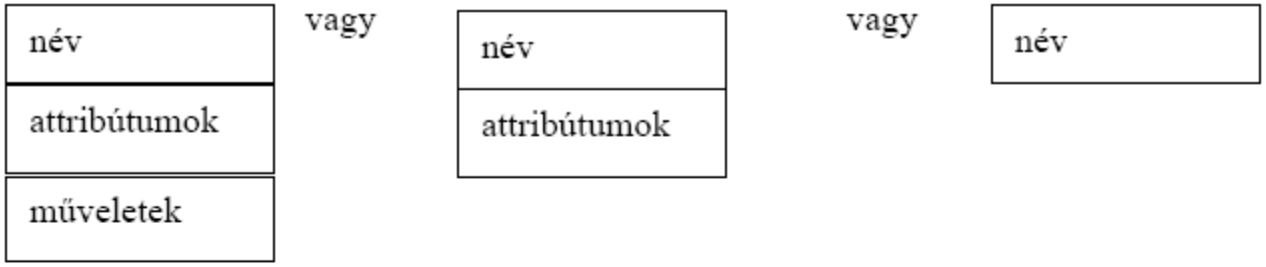
Másodfokú egyenlet megoldása



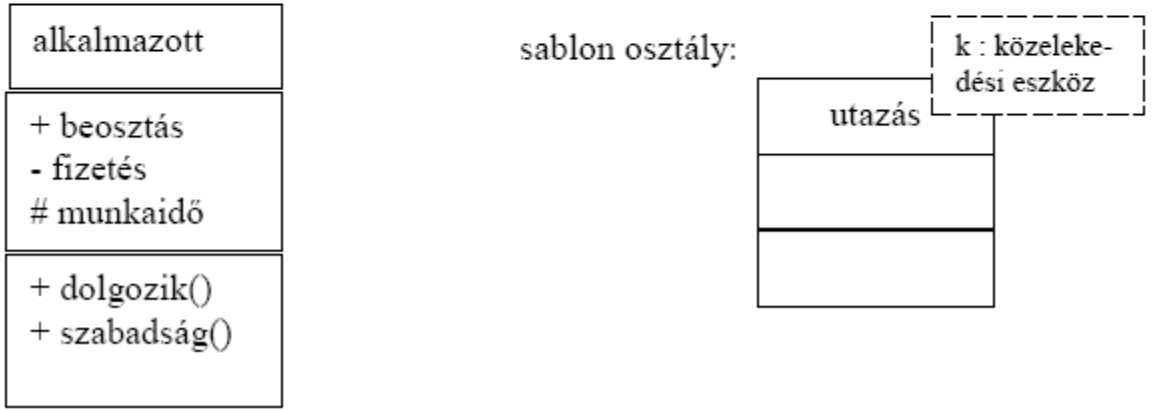
Activity diagram



Osztálydiagram



Az attribútumok és műveletek láthatóságai: + public, # protected, - private, ~ package. Pl.:

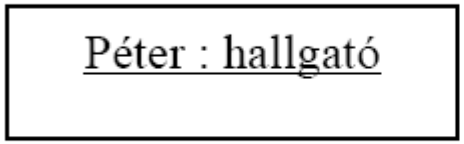


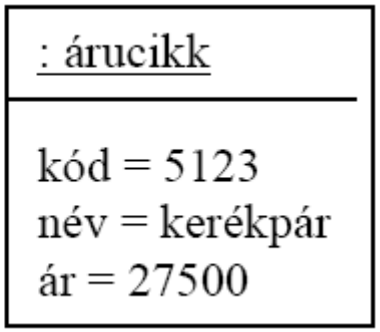
Az osztályok közötti kapcsolatok

- Asszociáció/társítás (association)
- Aggregáció/rész-egész kapcsolat (aggregation)
- Általánosítás (generalization)
- Függőség (dependency)
- Megvalósítás (realization)

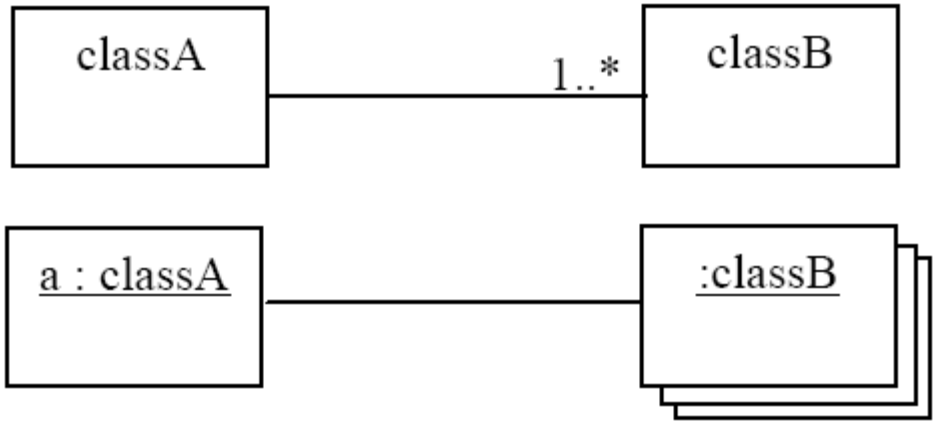
Objektum diagram

Az objektum rajzjele:

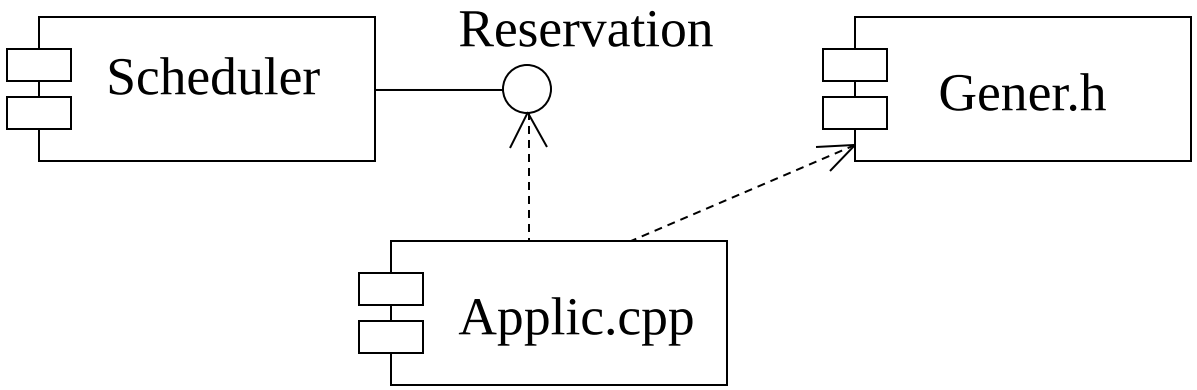




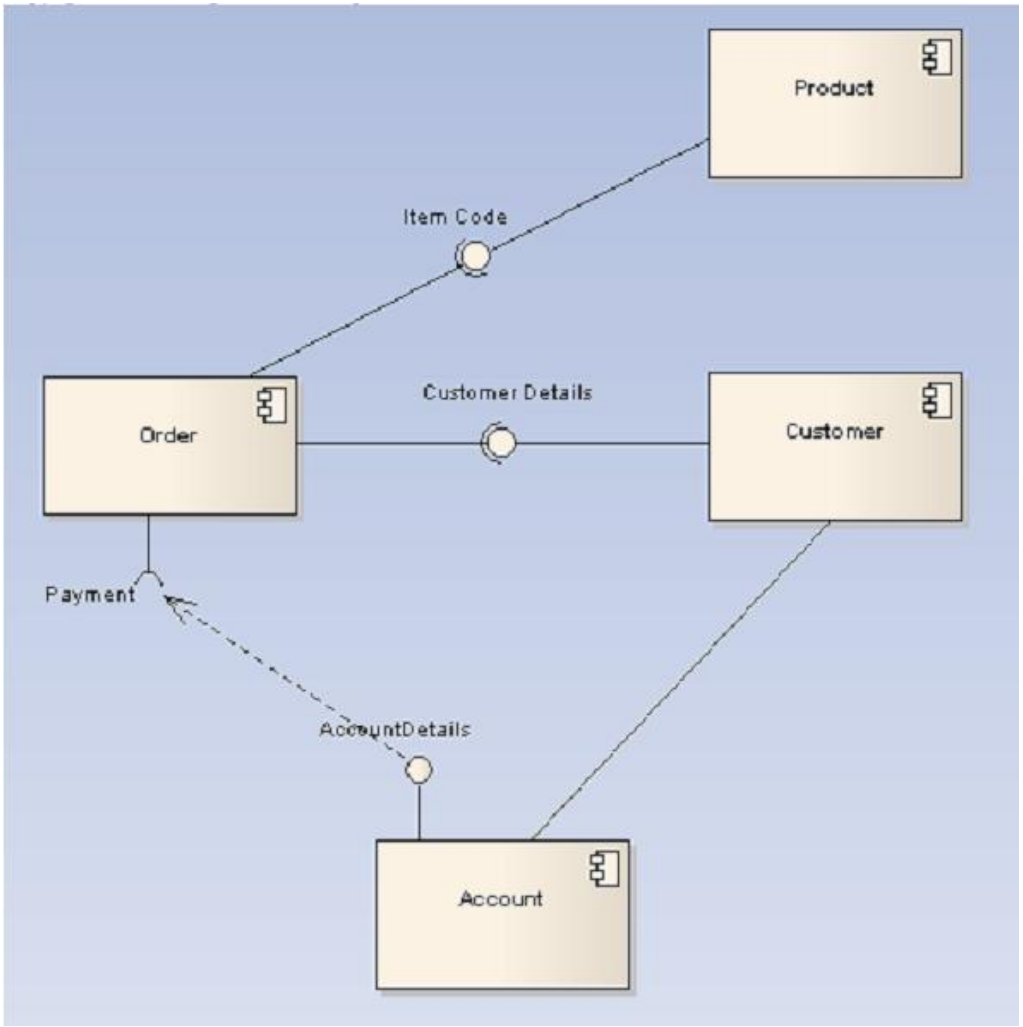
Az osztály- és az objektumdiagram kapcsolata:



Komponens diagram

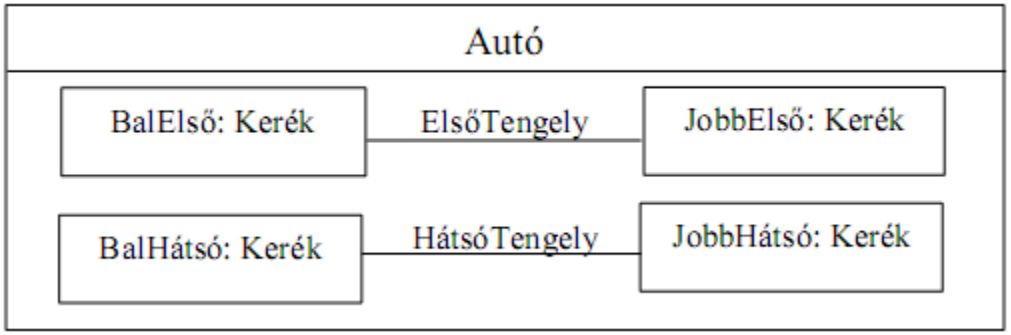


Komponens diagram

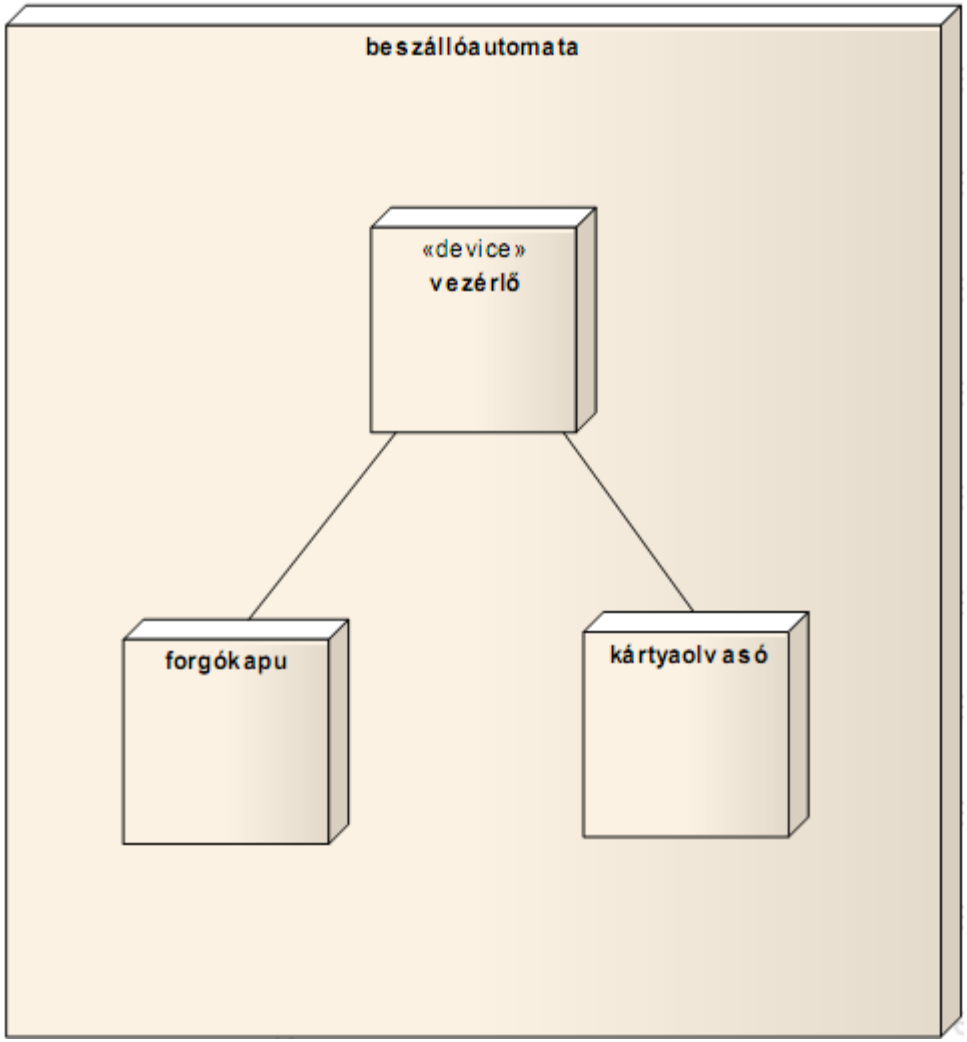


Összetett szerkezeti diagram

- Strukturált osztály: az osztály belső szerkezetét is megmutatja.



Kialakítási diagram



Kialakítási diagram

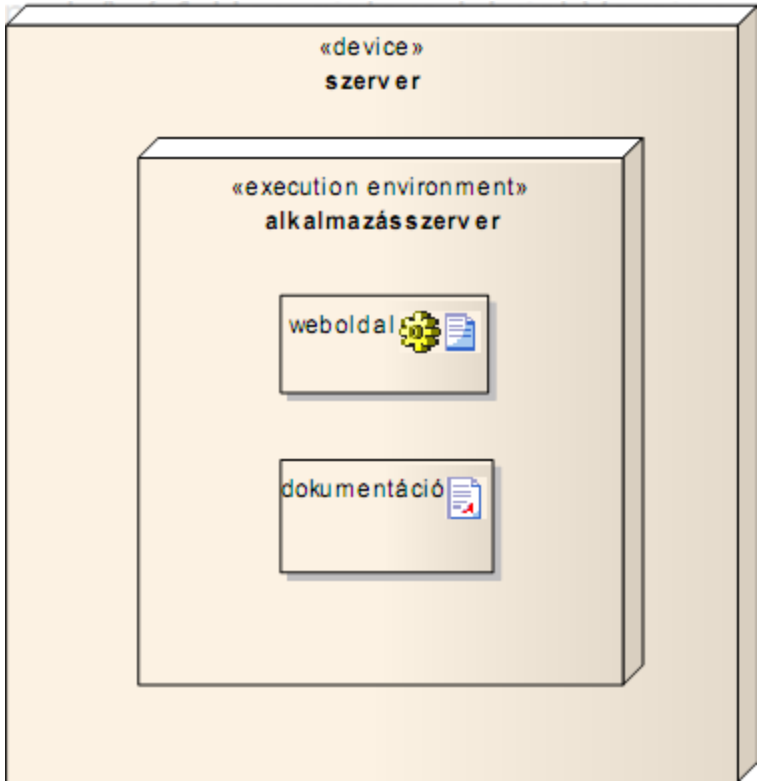


Diagram típusok

Szerkezeti diagramok:

- **Osztálydiagram (class)**
- **Objektumdiagram (object)**
- **Csomagdiagram (package)**
- **Összetevő diagram (component)**
- **Összetett szerkezet diagram (composite stucture)**
- **Kialakítás diagram (deployment)**

Viselkedési diagramok:

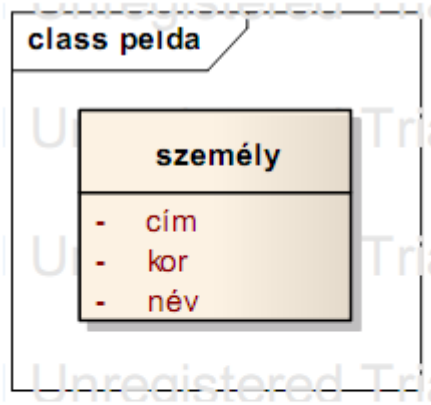
- **Tevékenység diagram (activity)**
- **Használati eset vagy feladat diagram (use-case)**
- **Állapotgép diagram (state machine)**
- **Kölcsönhatási diagramok:**
  - **Sorrend diagram (sequence)**
  - **Kommunikációs diagram (communication)**
  - **Időzítés diagram (timing)**
  - **Kölcsönhatás áttekintő diagram (interaction overview)**

Állapotgép diagram

- Az osztályok objektumainak, a használati eseteknek és a protokolloknak a dinamikus viselkedését mutatja, vagy a dialógusok lefutásának leírására is alkalmas
- Állapot: az objektum állapotát az attribútumai konkrét értékeinek n-esével jellemezzük.

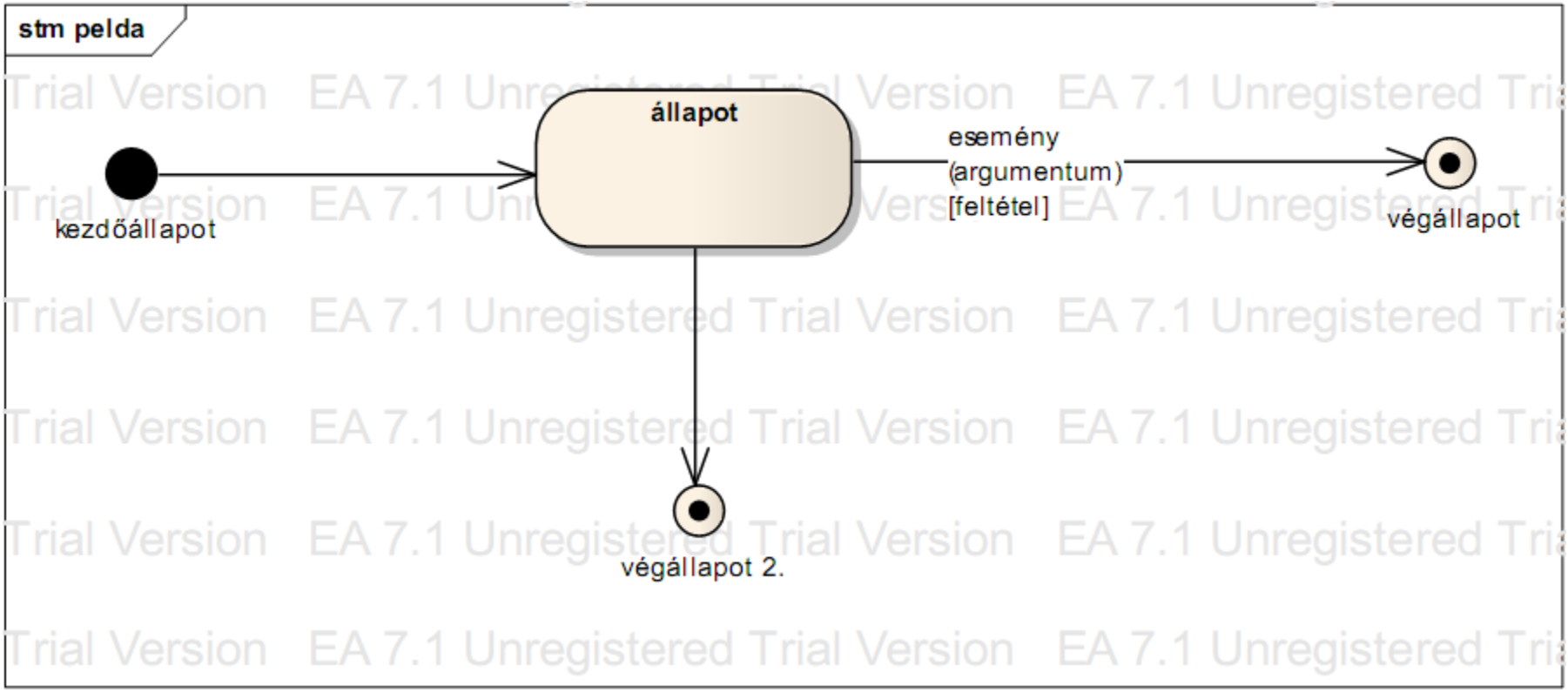
Állapotátmenet:

- két állapot közötti kapcsolat, amely kifejezi, hogy egy adott állapotban lévő objektum egy
- esemény vagy valamely feltétel bekövetkezésének hatására milyen másik állapotba kerül.

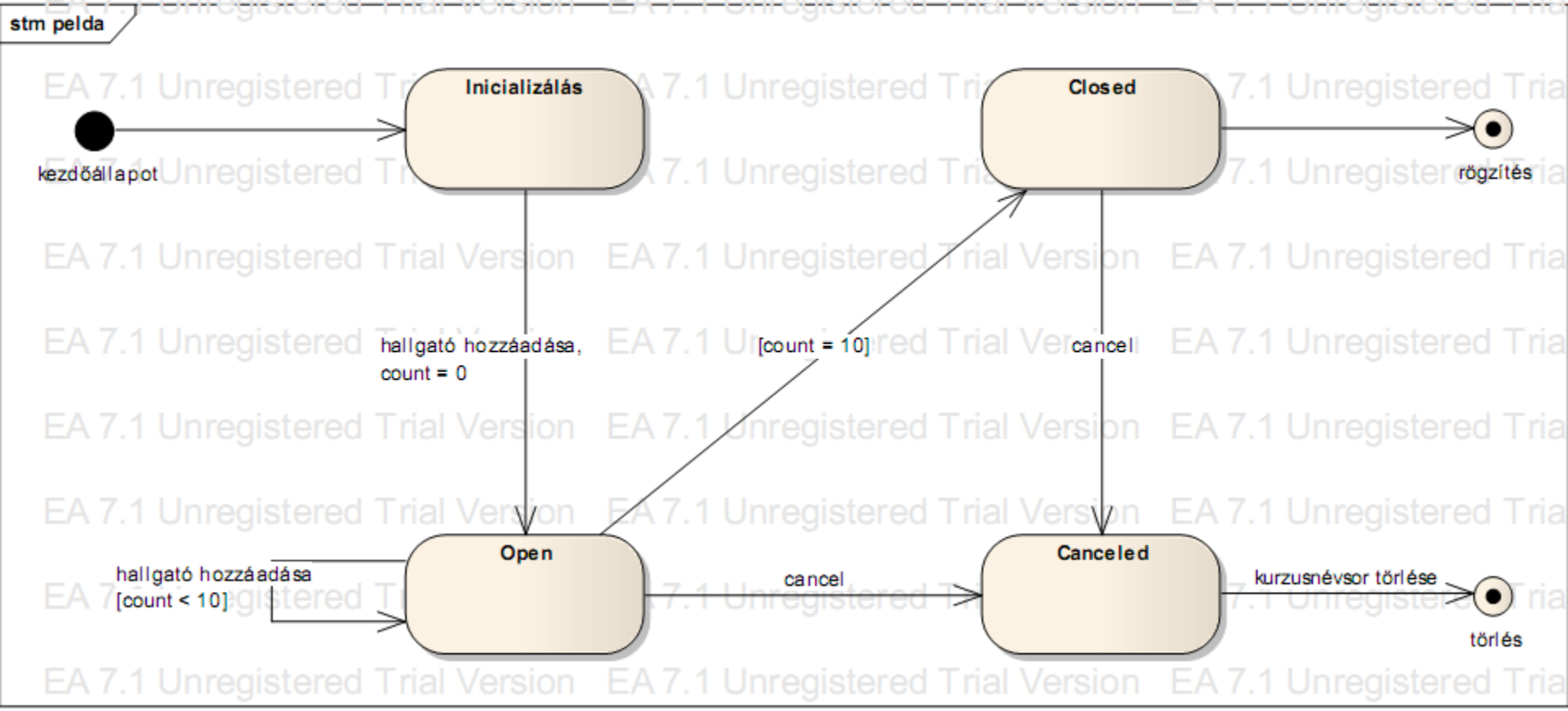


Szabó Zsolt
Budapest, Fő utca 2.
25

Állapotgép diagram



Állapotgép - tanulmányi rendszer - tantárgyfelvétel



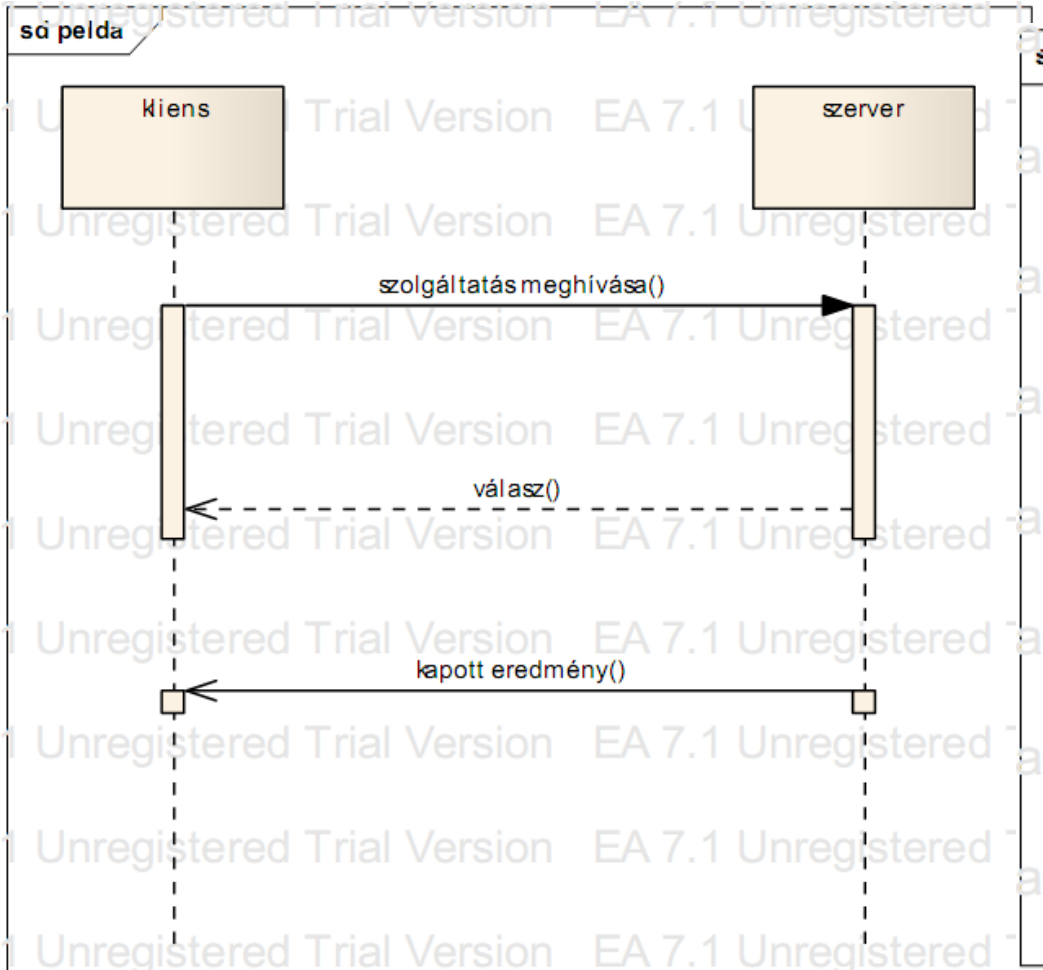
Kölcsönhatási diagramok

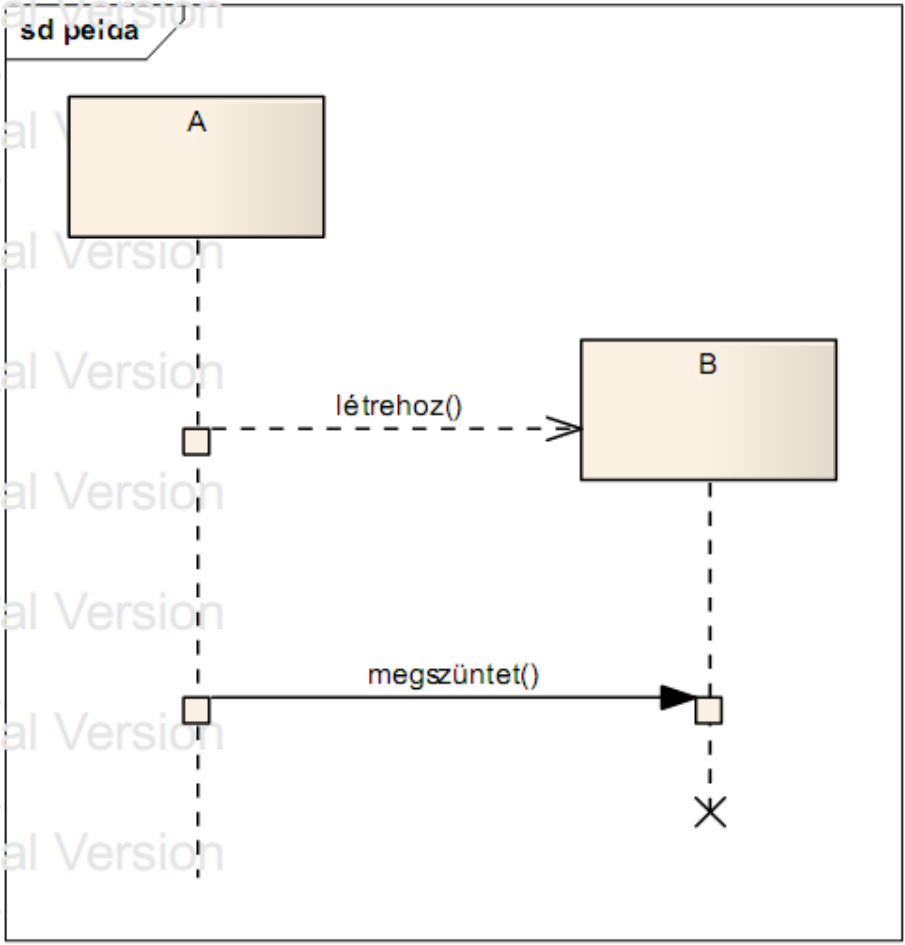
- Sorrend diagram – kevés résztvevő sok üzenettel
- Kommunikációs diagram – sok résztvevő kevés üzenettel
- Időzítés diagram – kevés résztvevő, komplex időbeli egymásra hatás

Sorrend diagram

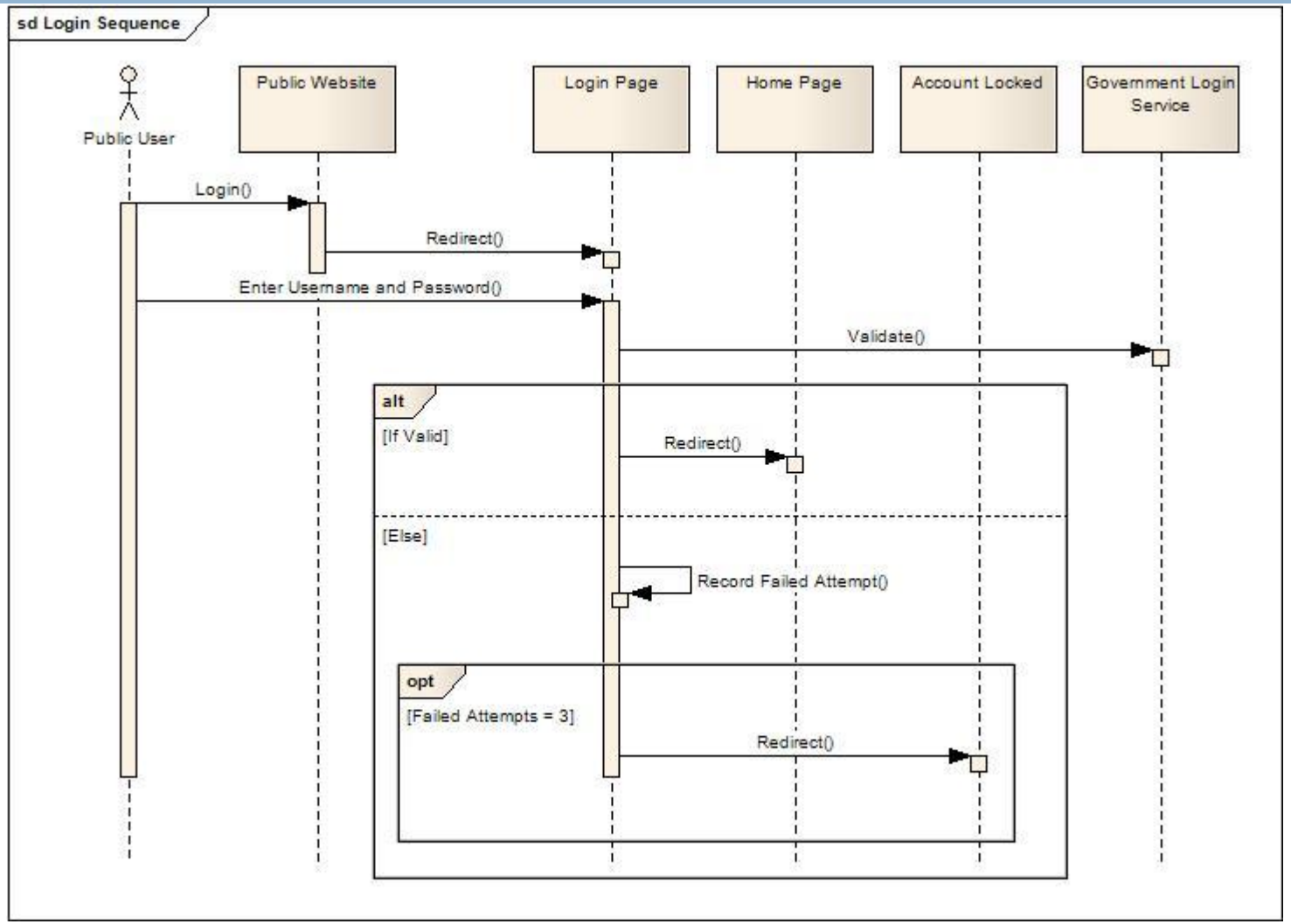
- Üzenetváltásokat ábrázol, amelyek több, kölcsönhatásban lévő partner között zajlanak le
- A partnerek lehetnek osztályok, aktorok, komponensek, csatlakozók és csomópontok.
- Akkor érdemes használni, **ha kevés résztvevő** (partner) van, de azok **sok üzenetet** küldenek.
- Az életvonalon látható üres téglalapot aktivációs résznek nevezzük, ilyenkor csinálhat valamit az adott szereplő.

Sorrend (szekvencia) diagram



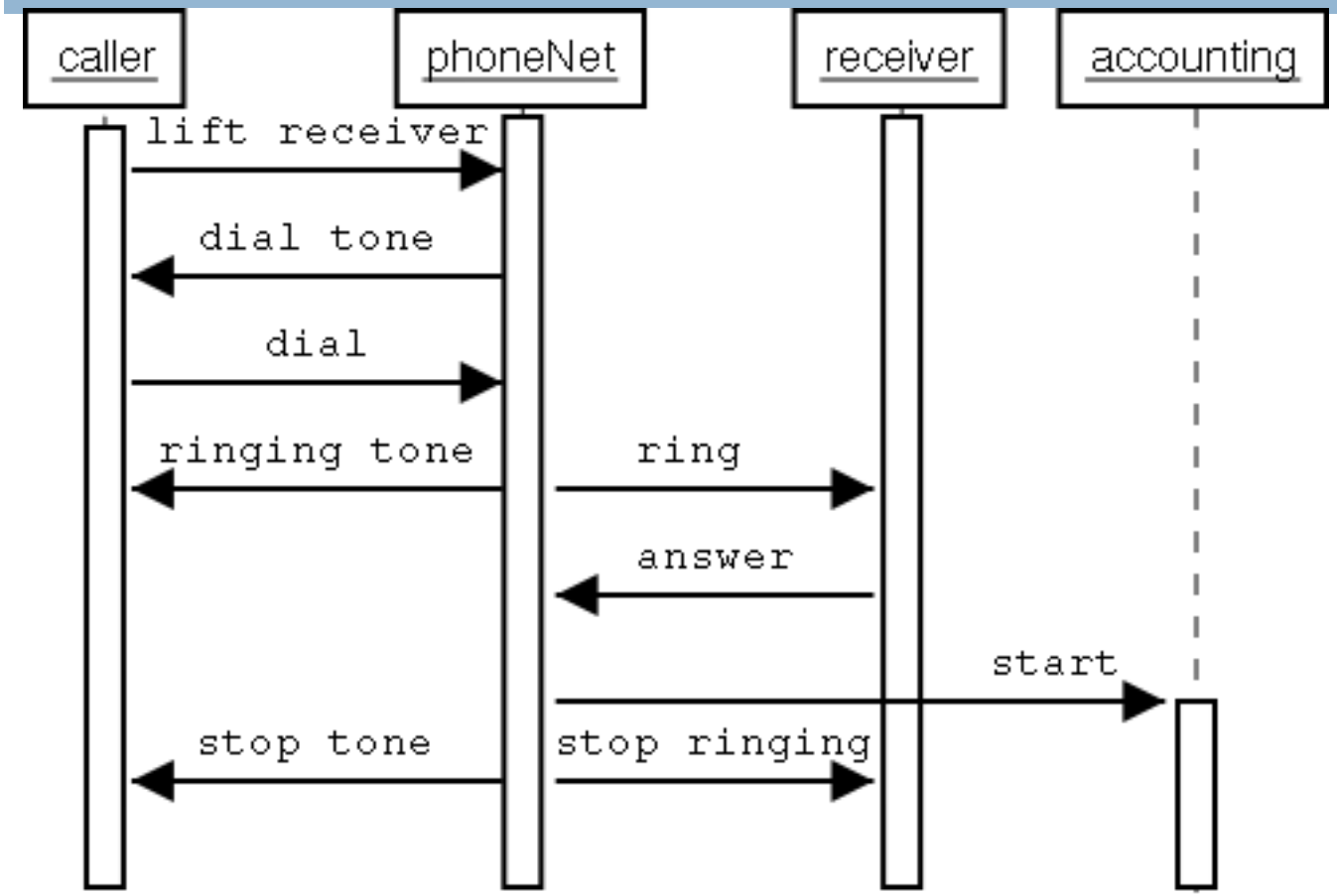


Sorrend (szekvencia) diagram



http://www.modernanalyst.com/Resources/Articles/tabid/115/articleType/ArticleView/articleId/353/Enterprise-Architect-for-Business-Analysts.aspx

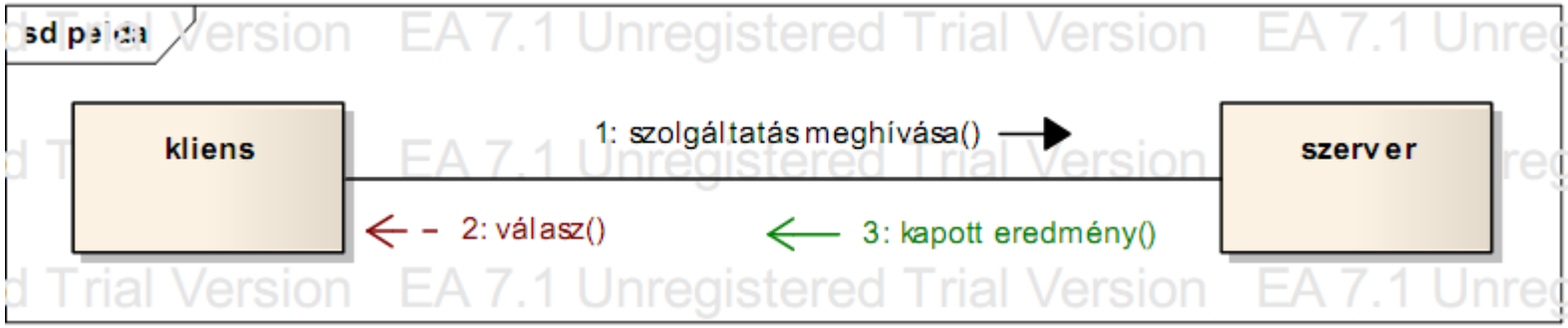
Sorrend (szekvencia) diagram



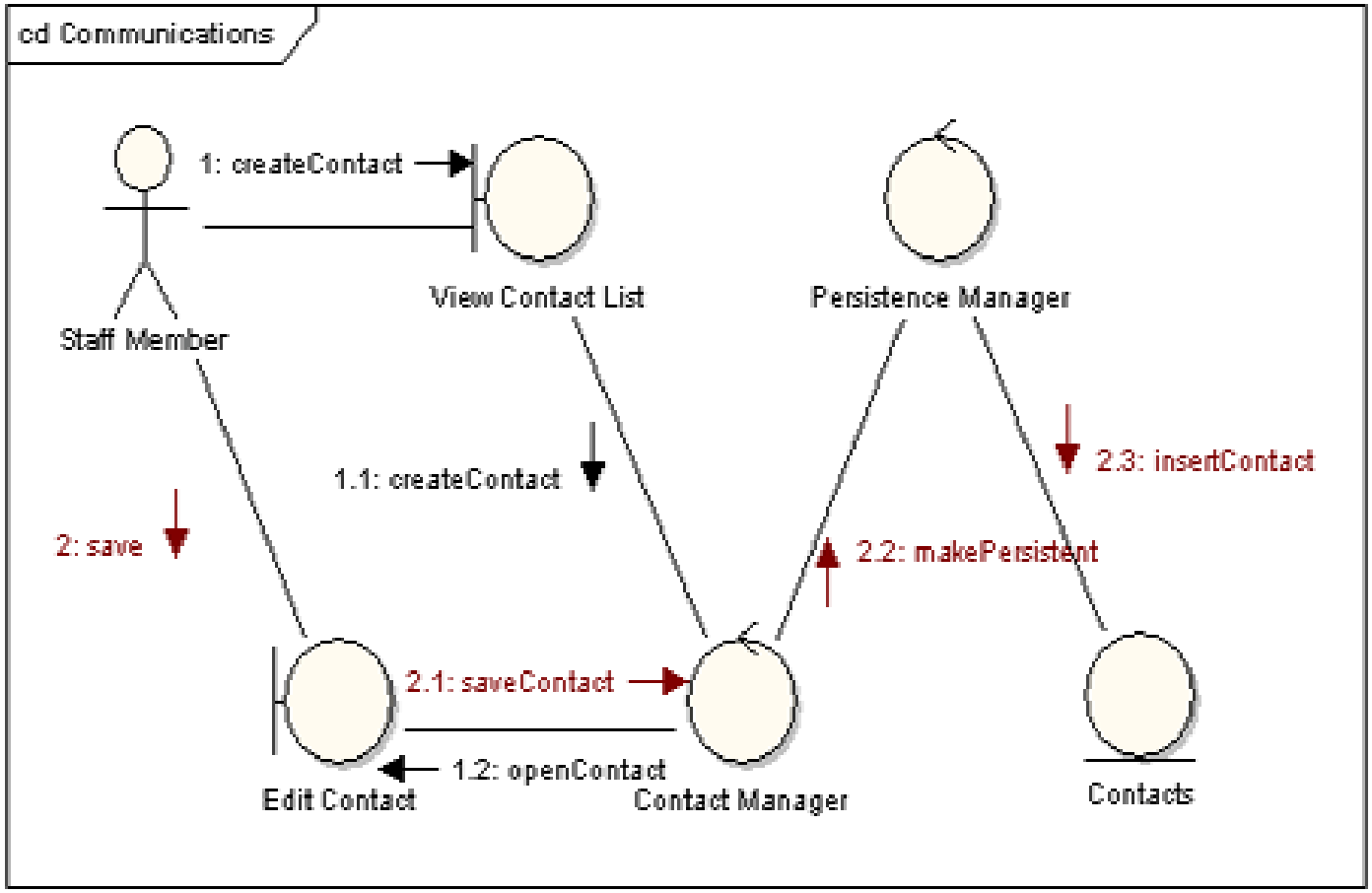
http://www2.pms.ifi.lmu.de/publikationen/diplomarbeiten/Sacha.Berger/illustrations/UML/SequenceDiagram-initiatingPhoneCall.pdf.png

Kommunikációs diagram

- ha sok résztvevő van és azok viszonylag kevés üzenetet küldenek egymásnak.



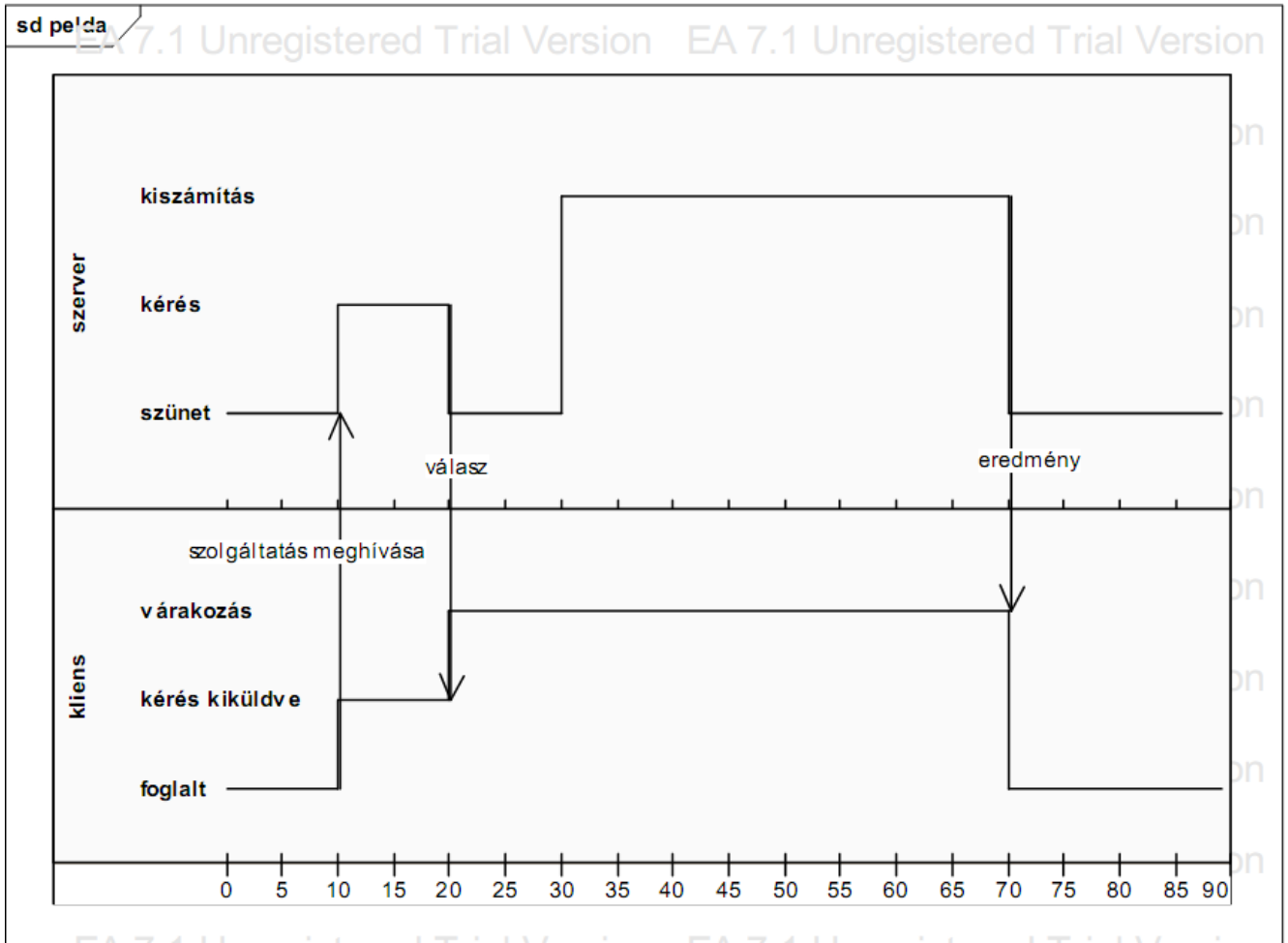
Kommunikációs diagram



http://afruj.wordpress.com/2008/07/28/the-unified-modeling-language-uml-part-8/

Időzítés diagram

- kevés résztvevő, komplex időbeli egymásra hatás



Tesztelés

Hibák csoportosítása

- Specifikációs hibák
- Programozási hibák:
  - Hibás funkcióteljesítés.
  - Hiányzó funkciók.
  - Adatkezelési hibák az adatbázis elérése során.
  - Kezdési és befejezési hibák.
  - Hibák a felhasználói interfészben.
  - Határértékek alá vagy fölé kerülés.
  - Kódolási hiba.
  - Algoritmikus hiba.
  - Inicializálási hiba.
  - A vezérlési folyamat hibája.
  - Adatátviteli hiba.
  - Input-output hiba.

Szoftverek ellenőrzése és elemzése

Technikák

- Statikus elemzés
  - Szoftver átvizsgálás
  - Automatikus elemzés
- Dinamikus tesztelés
  - Hiányosság tesztelés
  - Stressz tesztelés
  - Komponens tesztelés
    - egységteszt (unit test): egységek, komponensek önálló, más komponensektől független tesztelése
    - modul teszt: egymástól függő egységek, modulok tesztelése
  - Rendszer (integrációs) tesztelés

Szoftver átvizsgálás

- Legalább 4 fős csapat: szerző, olvasó, tesztelő, moderátor
- Átvizsgálás ↔ Szerző módosít
- Tipikus hibák: adath., vezérlési h., I/O h., interfész h., tárkezelési h., kivételkezelési h.
- A program hibáinak több mint 60%-a felderíthető ily módon

Automatikus elemzés

- Szoftver, ami a forrásszöveget vizsgálja (nem futtat)
- Nem használt kódrészlet, inicializálatlan változók
- Tipikus hibák: adath., vezérlési h., I/O h., interfész h., tárkezelési h.

Hiányosság tesztelés

Célok

- A specifikáció és a megvalósított program közötti eltérések felderítése
- A rendszer helytelen működésének előidézése

Irányelvek

- Válasszunk olyan bemeneteket, amelyekkel az összes lehetséges hibaüzenetet előidézhetjük
- Próbáljuk meg elérni a bemeneti pufferek túlcsordulását
- Ugyanaz a bemenet ugyanazt a kimenetet adja mindig?
- Kényszerítsük ki az érvénytelen kimenetet

Módszerek

- Fekete doboz tesztelés
  - A rendszer/komponens belseje nem ismert
  - A bemenetre adott válaszok tanulmányozása
- Fehér doboz tesztelés
  - Ismert a szerkezet
  - Kis egységekre alkalmazzák
  - Minden független végrehajtási utat kipróbálnak (útvonal teszt)

Fehér doboz tesztelés

Útvonalak felderítése

- Egyszerű esetekben folyamatgráf felrajzolása kézzel
- Bonyolult esetekben dinamikus programelemzővel (DPE)
- DPE: a fordítóval együttműködő szoftver, számolja, hogy az egyes utasítások hányszor kerültek végrehajtásra – mi maradt ki → futási profil

Stressz tesztelés

Cél

- A teljesítmény és a megbízhatóság tesztelése
- A tervezett terhelés mellett képes legyen dolgozni a rendszer
- Olyan hiányosságok is kiderüljenek, amelyek nem jelennek meg normál körülmények között

Eljárás

- Addig növeljük a terhelést, amíg a rendszer teljesítménye elfogadhatatlanná nem válik

Verfikációs, validációs teszt

- Verifikációban az egyes fázisok közötti összhang fejlesztése a feladat, megfelel-e a specifikációnak.
- Validáció a végső rendszer működésének vizsgálata, hogy jó e felhasználói szempontból.

Alfa- és béta tesztelés

- Alfa tesztelés: átvételi tesztelés a megrendelővel
- Béta tesztelés: korlátozott végfelhasználói (potenciális felhasználók által, a normál használat során végzett) teszt

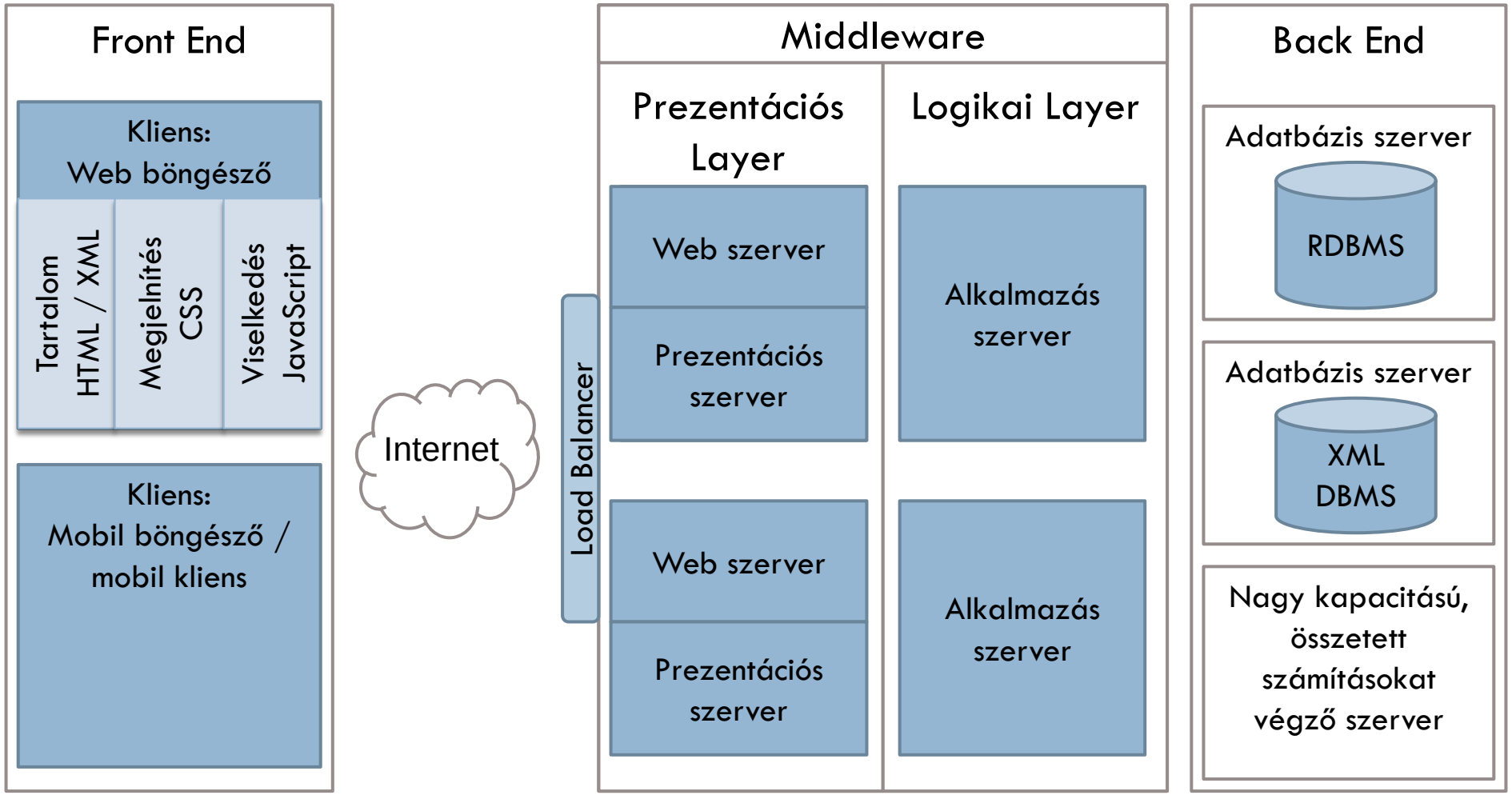
Szoftverminőség

Szoftverminőség

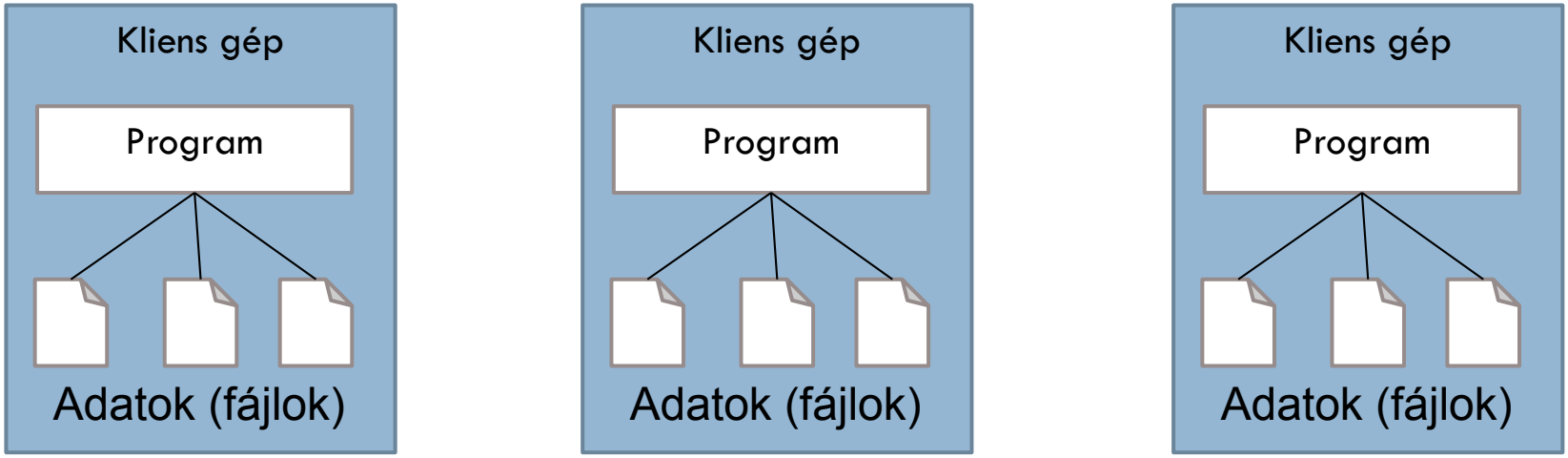
- Működőképesség: A funkciók azok, amelyek eleget tesznek a meghatározott vagy következtetett szükségleteknek. Célnak való megfelelősség, pontosság, megfelelés, biztonság.
- Megbízhatóság: hogy a szoftver a működőképességét adott feltételek mellett, adott időtartamon keresztül fenntartja. Hibatűrő képesség, helyreállíthatóság.
- Használhatóság: érthetőség, megtanulhatóság, működtethetőség
- Hatékonyság: válasz- és végrehajtási idők, erőforrás-kihasználás.
- Karbantarthatóság, rugalmasság
- Hordozhatóság, újrafelhasználhatóság, együttműködési képesség

Webes architektúrák

Web-es architektúra



Egygépes (standalone) alkalmazások



- A program teljes egészében a munkaállomáson fut.
- Az adatok ugyanitt tárolódnak.
- Egyszerre csak egy felhasználó használhatja.
- Semmilyen hálózati kapcsolat nincs, a különálló programok közti adatszinkronizáció meglehetősen nehézkes.

Egyszerű kliens-szerver alkalmazások 1.

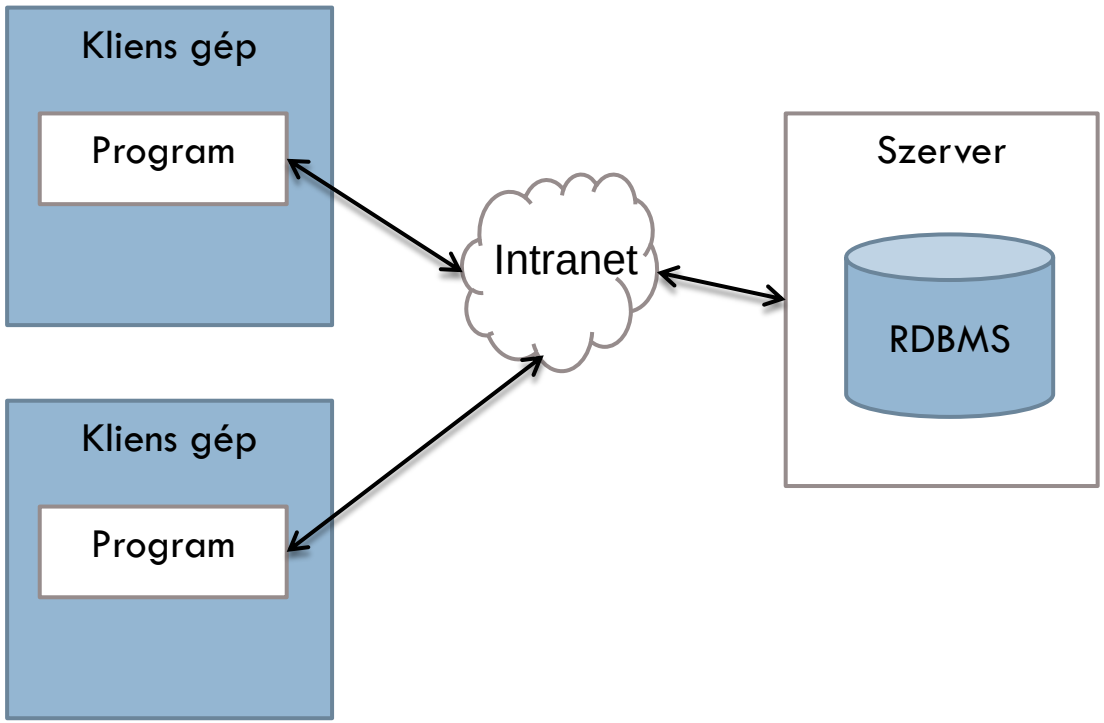
Egy vagy több szerver gép erőforrásait (jellemzően adatait) megosztja a kliensek között. Jobb esetben on-line.

Az alkalmazás egy része (adatbázis-kezelő rsz.) a szerven fut.

Az alkalmazás logikát implementáló rész a kliens gépeken fut. „***vastag kliens rendszerek***”

Egy adatbázist többféle kliens program is használhat.

Egyszerre több konkurrens felhasználó használhatja.



Egyszerű kliens-szerver alkalmazások 2.

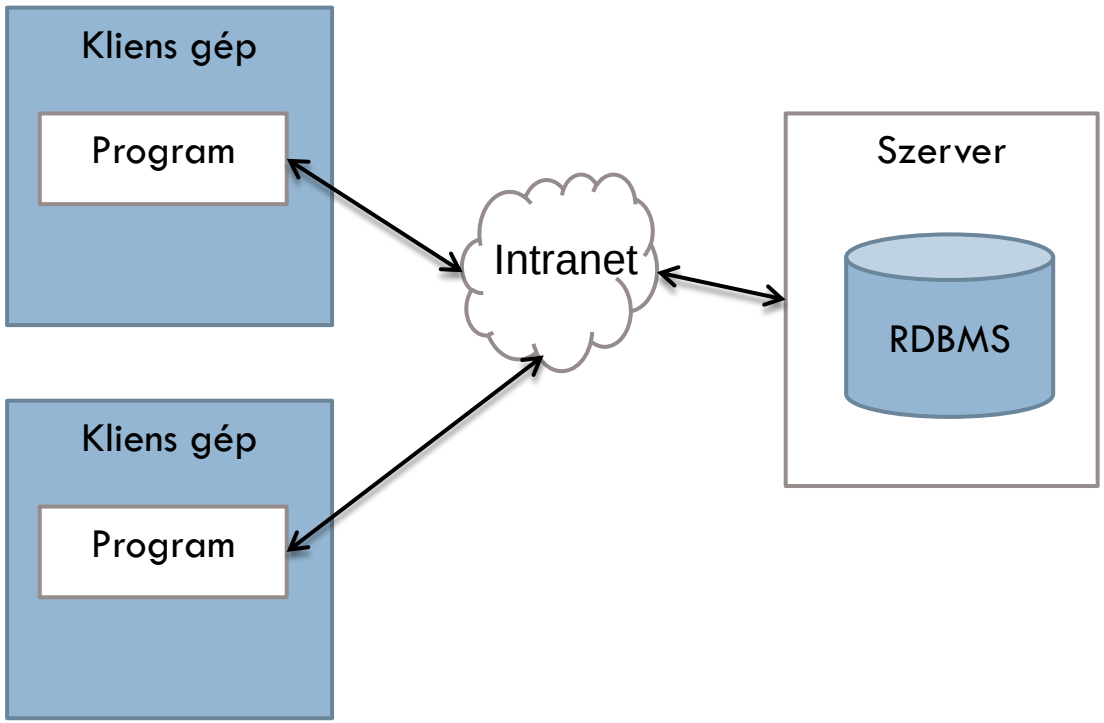
Jellemzően intranet-es alkalmazásoknál használatos.

Terheli a kliens gép erőforrásait.

Gyakran mindenféle driver-ek telepítését igényli a kliens gépeken

Verziófrissítés alkalmával az összes kliens-en frissíteni kell a programot.

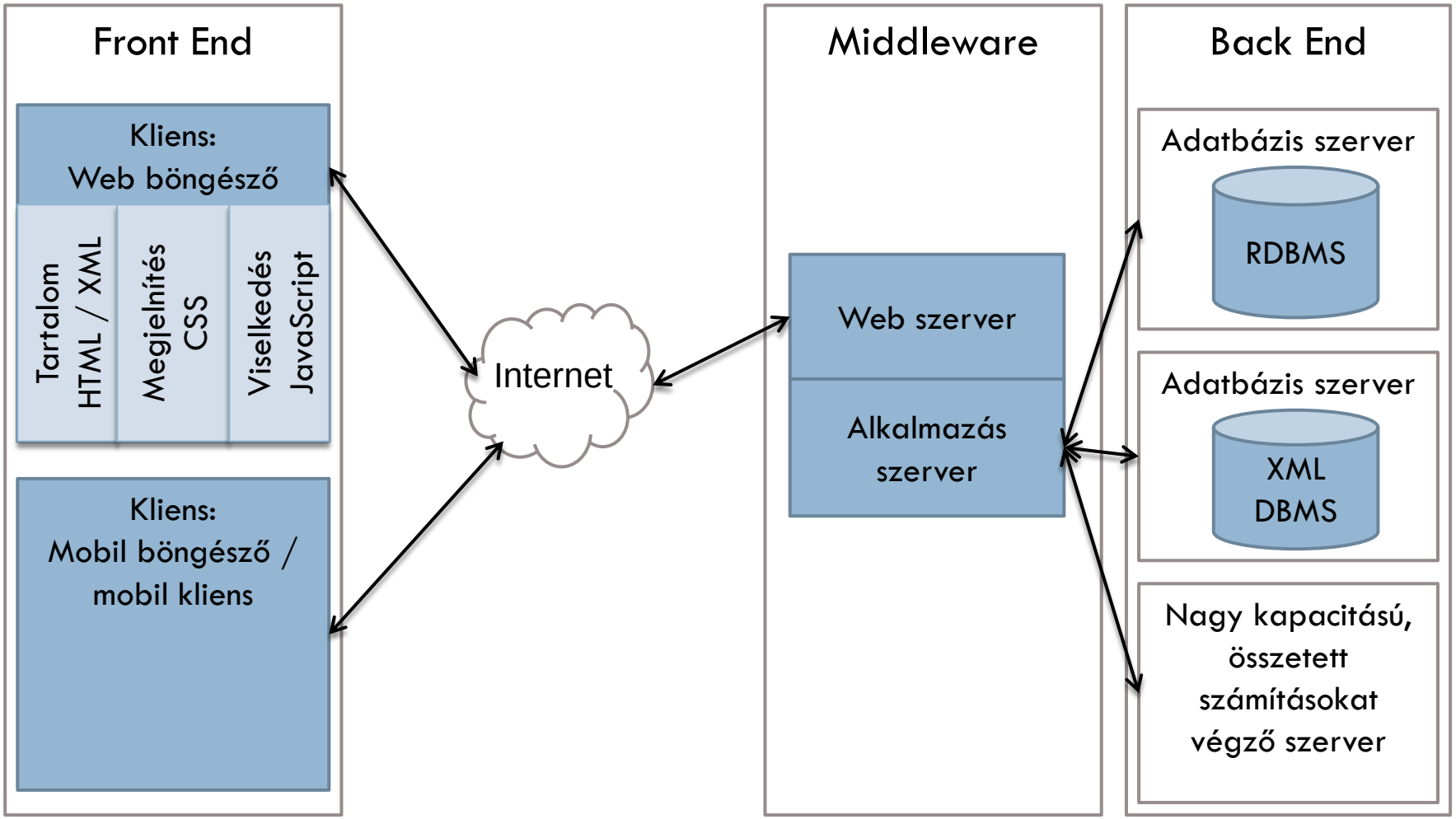
A RAD (Rapid Application Development) sok eszközzel támogatott, számos jó vizuális fejlesztőkörnyezet: gyorsan „összekattint-gathatunk” és leprogramozhatunk komoly alkalmazásokat.



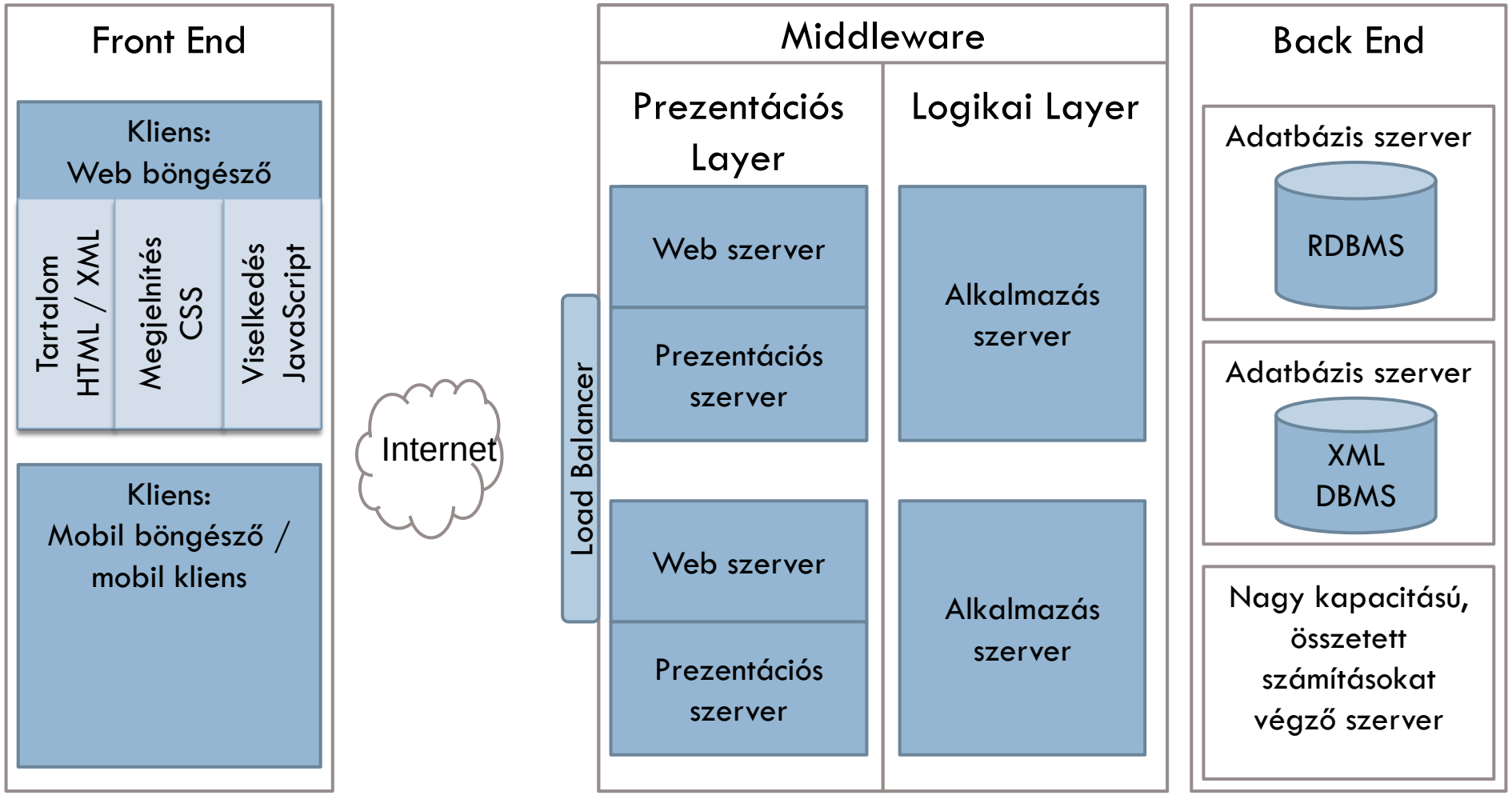
Többrétegű (multitier) hálózati alkalmazások

- Minimálisan három réteg létezik:
  - Front End = kliens oldali felhasználói réteg (általában egy WEB böngészőben)
  - Middleware = szerver oldali prezentációs és logikai réteg (általában egy WEB szerveren beágyazott script-ekben összeolvasztva a megjelenítés és az egyszerűbb logika)
  - Back End = hátsó szerver oldali nagykapacitású tároló (adatbázis szerver) vagy számoló réteg

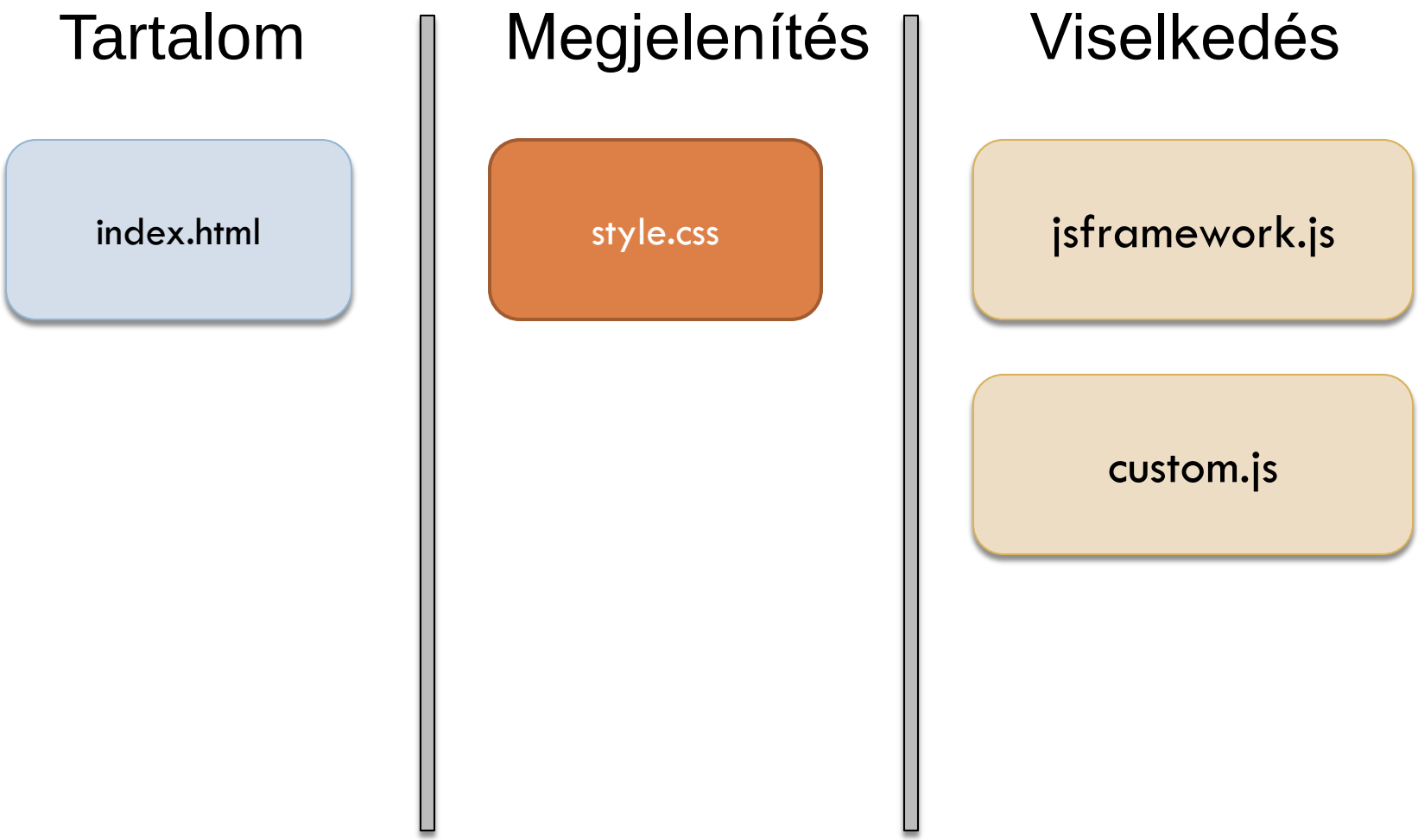
Háromrétegű architektúra



Többrétegű architektúra



A kliens oldal



Többrétegű architektúra jellemzői

- Load balancing, terhelésmegosztás.
- Tervezést támogató környezetek: Java J2EE, .Net.
- Architektúra felosztás-összevonás logikai szinten.
- A rendszer logikai architektúrája (tervezés, programozás) független a számítógépes megvalósítástól, hálózattól.
- A logikai réteg tovább osztható.
- Nagyon sok konkurens felhasználó kiszolgálására optimalizálva

Többrétegű architektúra jellemzői

- Kliens gép: böngésző, a logika – többnyire – a szerveren található – vékony kliens architektúra
- Minimális logika a klienseken: a beviteli adatok validálására, a lapok speciális megjelenítésére (pl. JavaScript).
- A szerveren elkülönül az adattárolás, a logika és a prezentáció → eltérő szerepkörök
- Az egyes szintek önmagukban is tesztelhetőek.
- A rendszer egyes komponensei több célra vagy újra felhasználhatók.

Többrétegű architektúra jellemzői

- A vékony kliensek miatt nagyon gyenge kliens gépek is elegendők.
- A technológia platformfüggetlen.
- A kliensekre nem kell drivert telepíteni.
- A verziófrissítés csak a szervert érinti, a klienseket nem.
- Sajnos egyelőre elég kevés eszköz támogatja a RAD-ot (Rapid Application Development), a környezet kevés segítséget nyújt a programozónak a megoldási lehetőségek kiválasztásában
  - → „Házi szabványok”, saját keretrendszerek készülnek.
- Nehezebb tesztelni

Szükséges ismeretek

- Hálózati, szerver és kliens oldali megoldások (TCP/IP és HTTP protokoll és működése)
- WEB-es prezentációs megoldások (HTML nyelv, CSS), web-grafika
- WEB szerverek, böngészők, kliens oldali WEB programozás alapjai (pl. JavaScript)
- Adatbázis-kezelés (a relációs modell, adatmodellezés, SQL)
- Rendszerek közti adatkommunikáció „önleíró” dokumentum nyelven = XML (XML felépítése, használata, kapcsolódó technológiák érintőlegesen: DTD, XSD, XSL ill. XSLT)
- XML alapú adatbázisok (XML adattárolás alapjai, lekérdező nyelvek: XPath, XQuery)
- Multimédiás adatbázisok (nagy méretű multimédiás anyagok tárolása adatbázisokban, visszakeresés, hatékonyság)
- Programozási módszertan
- WEB programozás (módszerek, beágyazott script-nyelvek, PHP, .NET (ASP.NET), Java Web)
- Vállalati környezetre tervezett webes fejlesztői környezetek (pl.: .Net, Java EE) Multimédiás WEB programozás – bináris tartalmak (stream-ek, header, letöltés, feltöltés)
- Multimédiás WEB programozás – vektorgrafikus és programozott tartalmak (SVG, Flash)
- Informatikai biztonság (adatvédelem, kommunikációs vonalak védelme, védelem illetéktelen behatolásokkal szemben, meghibásodások elleni védelem)
- Üzemeltetétési, rendszergazdai ismeretek
- Többrétegű (összetett) webes alkalmazások fejlesztése, projektmenedzsment, rendszerszervezési ismeretek
- Web-gazdaságtan, web marketing,...

Web-site vs. Web-alkalmazás

	Web-site	Web-alkalmazás
Statikus / dinamikus	Statikus vagy statikus-szerű	Dinamikus
Adatbázis	Nem szükséges / nem tipikus	Jellemző
Hagyományos (asztali) alkalmazás	Nem implementálható asztali alkalmazásként	Rendelkezik asztali alkalmazásokhoz hasonló funkcionalitásokkal
Authorizáció	Nem jellemző	Jellemző
Bookmarking / search engine	Tipikus, jellemző	Nem működik. Keresőmotorok számára irreleváns, feldolgozhatatlan
Szerver-oldali logika	Nem jellemző	Mindig
Kliens-oldali logika	Nem jellemző, de előfordulhat	Jellemzően
Példa	Híroldalak, (Wikipedia)	Google Docs

Keretrendszer vs. Tartalomkezelő (CMS)

	Web-es fejlesztés célja			
Programozói készségek, érettségi szint az adott környezetben		Tisztán tartalom megosztás	Tartalom-megosztás kevés fejlesztéssel	Szofisztikált funkciók, a tartalmi szempontok nem fontosak
	Kezdő	CMS	CMS, de fejlesztés nem ajánlott	Projekt nem ajánlott
	Haladó	CMS	CMS / Framework	Framework
	Profi	CMS	CMS / Framework	Framework

Web site tervezés

Web Site tervezés

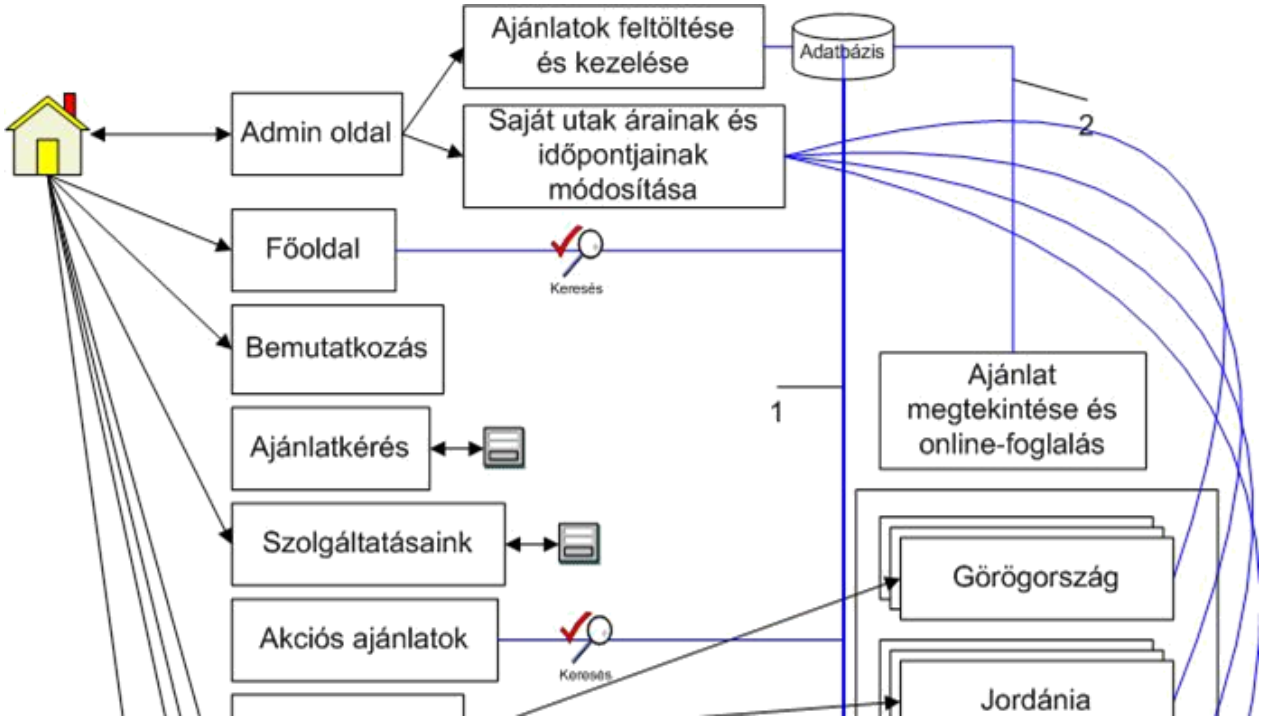
1. Információ gyűjtés
2. Tervezés
3. Tartalom és design
4. Fejlesztés
5. Tesztelés, minőségi ellenőrzés / Üzembehelyezés
6. Karbantartás

1. Információgyűjtés

- Igényfelmérés:
  - több lépcsőben,
  - funkcionális igények felmérésével
  - marketing- és stratégiai célok meghatározása
- Előzetes árajánlat
- Domain név és tárhely (www.domain.hu)

2. Tervezés – I.

- Anyagbeszerzés – I.
- Adat, funkcionális, navigációs terv készítése



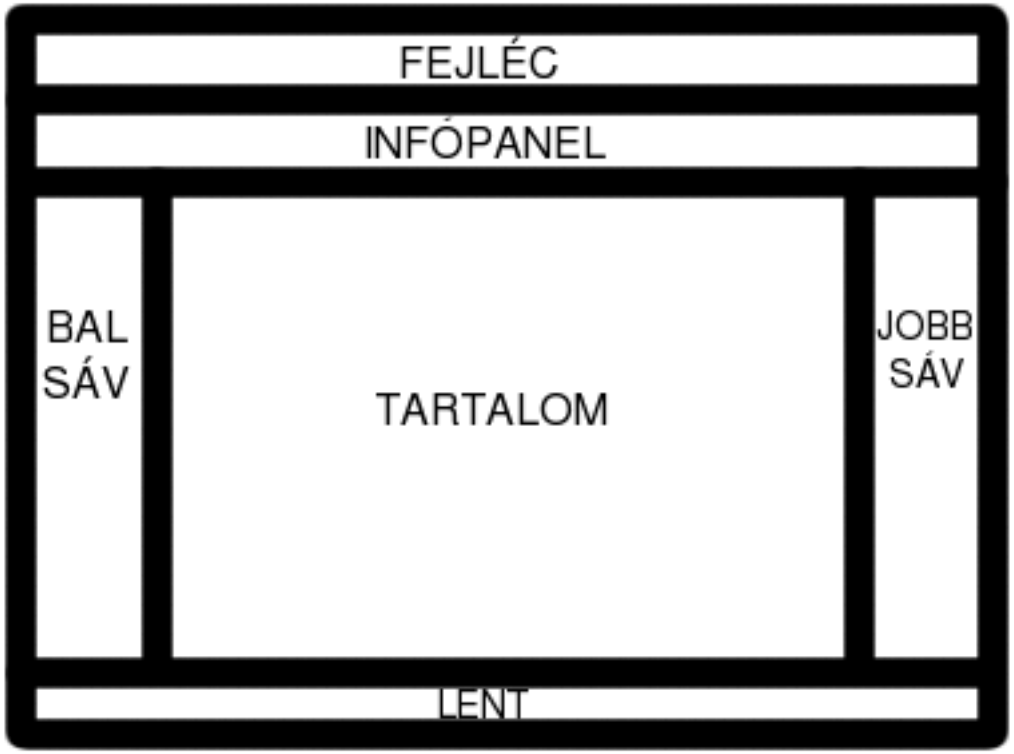
2. Tervezés – II.

- Technológiai és megrendelői döntések
  - Statikus vs. dinamikus (PHP, .NET, Java, stb.) oldal
  - Adatbázis vs. fájl tárolás
  - Ki tartja karban az oldalakat: megrendelő, készítő vagy rendszergazda, stb.
  - Saját oldal (sablon) vs. keretrendszer vs. tartalomkezelő rendszer
  - Tárhely-szükséglet tervezés
- Árajánlat

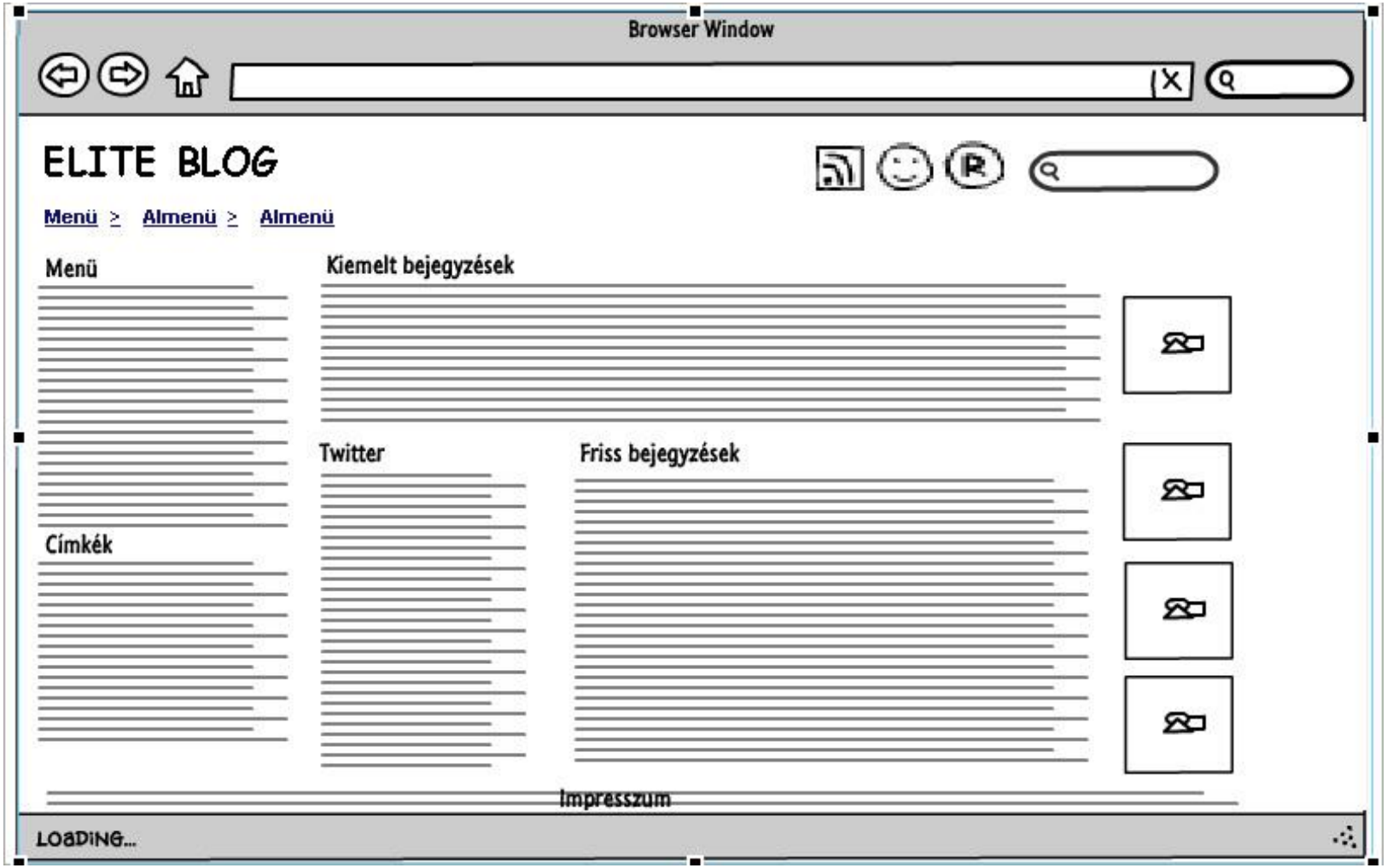
3. Tartalom és design

- Marketing-terv készítése
- Arculat-terv, logótervek készítése
- Feladatok meghatározása
- Sablon készítése
- Döntés a design-ról

Presentation Model



Presentation Model - Mockup



Web-site tervezés - sablon

4. Fejlesztés

- Anyagbeszerzés – II.
- További oldalak elkészítése (sablon)
- Fejlesztés
  - Szerver oldali kód
  - Kliens oldali kód

5. Tesztelés, értékelés

- Tesztelés
- Mérések értékelése
- Javítások, amennyiben szükségesek
- Üzembe helyezés
- Karbantartási terv
  - Jótállás
  - Karbantartás
  - Support

6. Karbantartás

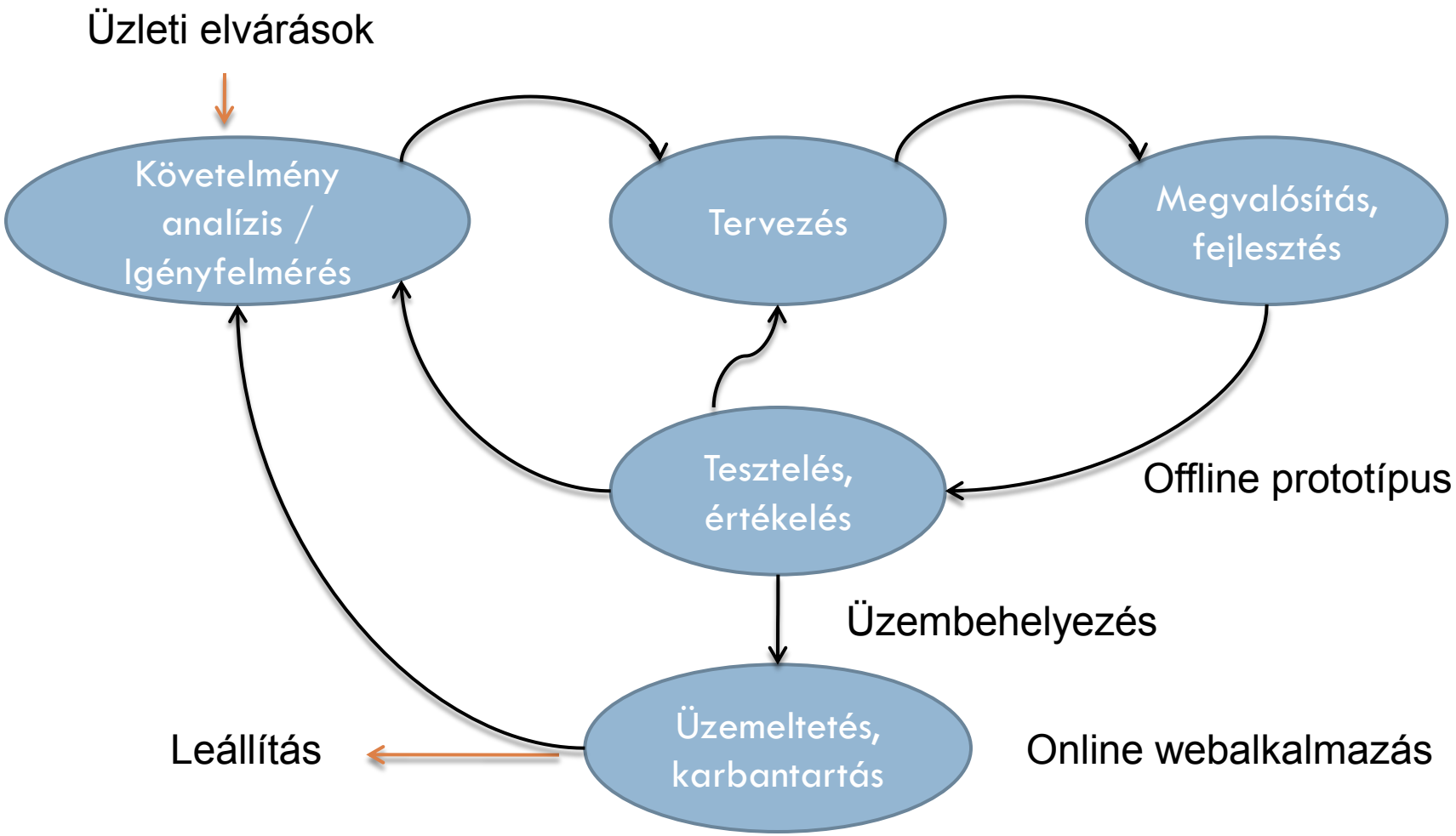
- Javítások
- Üzemeltetési feladatok

Web alkalmazás tervezése

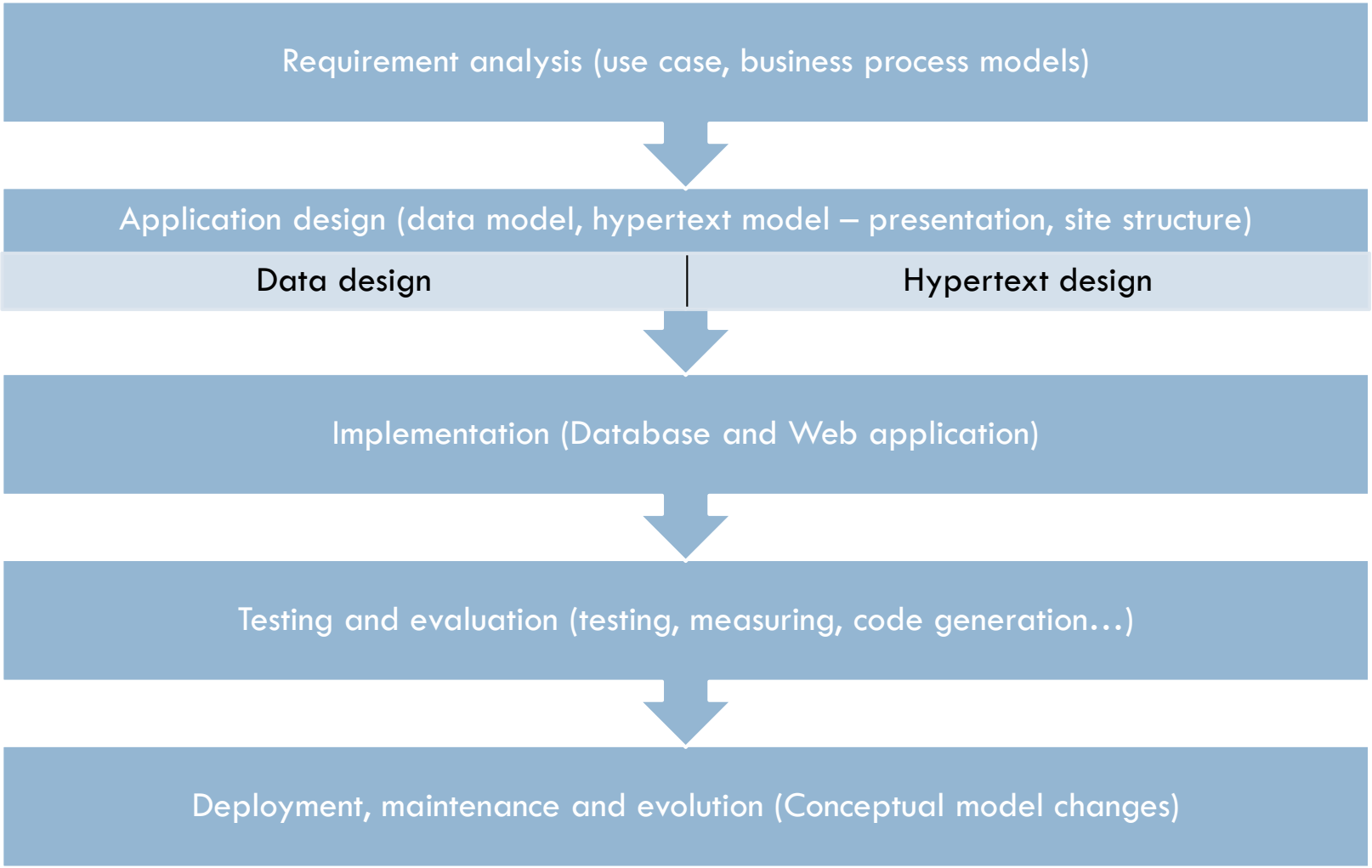
Példa: albumkezelő web-alkalmazás

- Célunk egy olyan webalkalmazás készítése, amely lehetővé teszi fényképalbumok készítését, megtekintését, publikálását.

Modern Web Alkalmazások evolúciós modellje



WebML Development Processes



Követelmény-analízis

- Pontosan milyen oldalak lesznek?
- Milyen adatok jelenjenek meg az oldalakon?
- Hogyan nézzenek ki ezek az oldalak? (sablon)
- Milyen összefüggésben vannak ezek az oldalak? (oldaltérkép)
- Hogyan azonosítom a felhasználókat, hogyan különböztetem meg, hogy kinek milyen albumai vannak? (azonosítás, autentikáció)
- Általában: milyen műveleteket és oldalakat érhetnek el az azonosítatlan és azonosított felhasználók? (szerepkörök, autorizálás)
- Mik az egyes oldalak adatigényei? (modell felépítése, körvonalazása)
- Milyen struktúrában, hogyan tároljuk az adatokat? (adatbázis)
- Milyen eszközök támogatottak? (asztali böngésző, mobil kliensek)

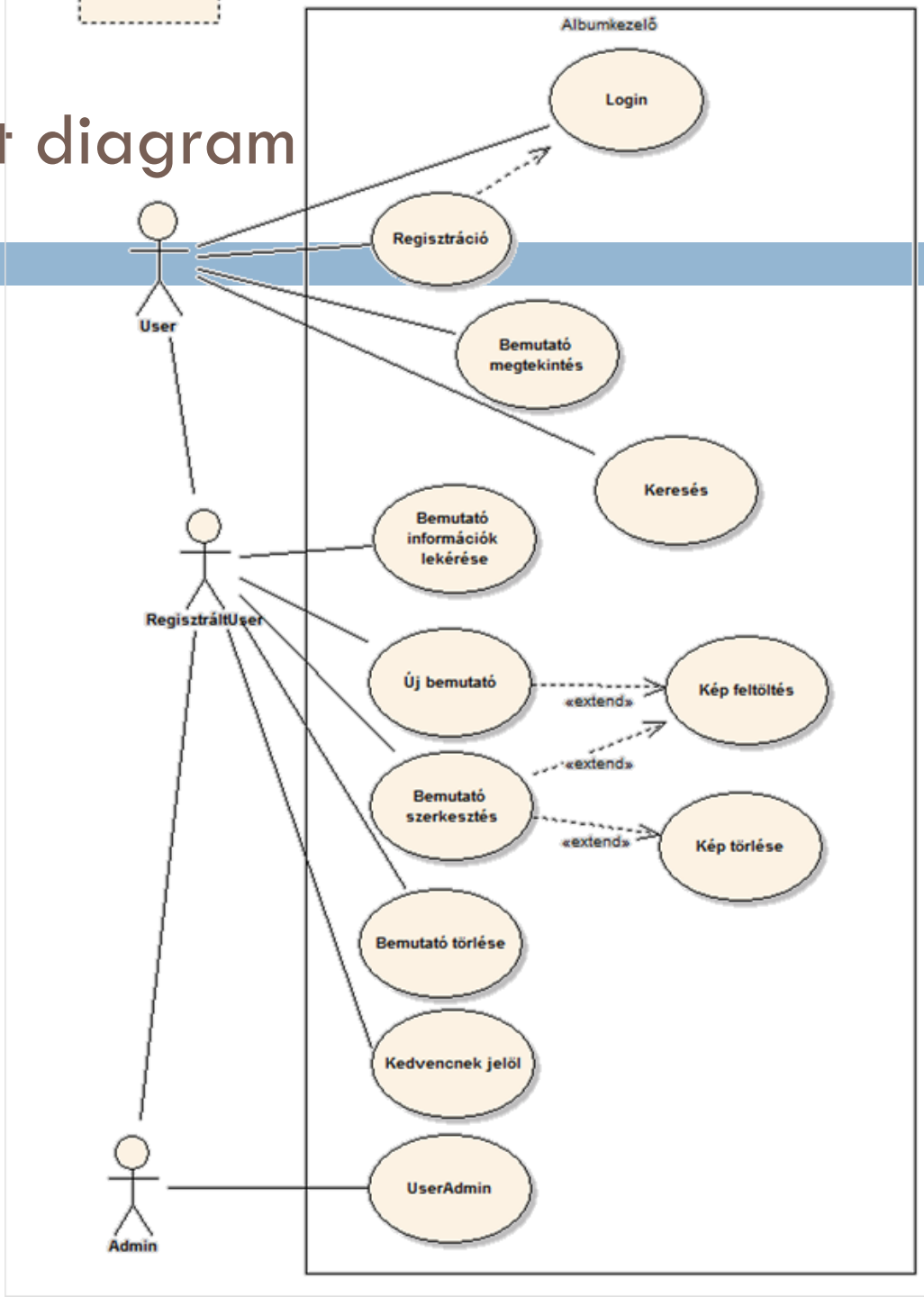
Tervezési szempontok

- szerepkörök
- oldalvázlatok készítése
- oldaltérkép (site struktúra)
- oldalfunkciók adatokkal
- adatbázis tervezése
- modell előkészítése
- designtervek készítése, dinamikus tartalmak jelölése

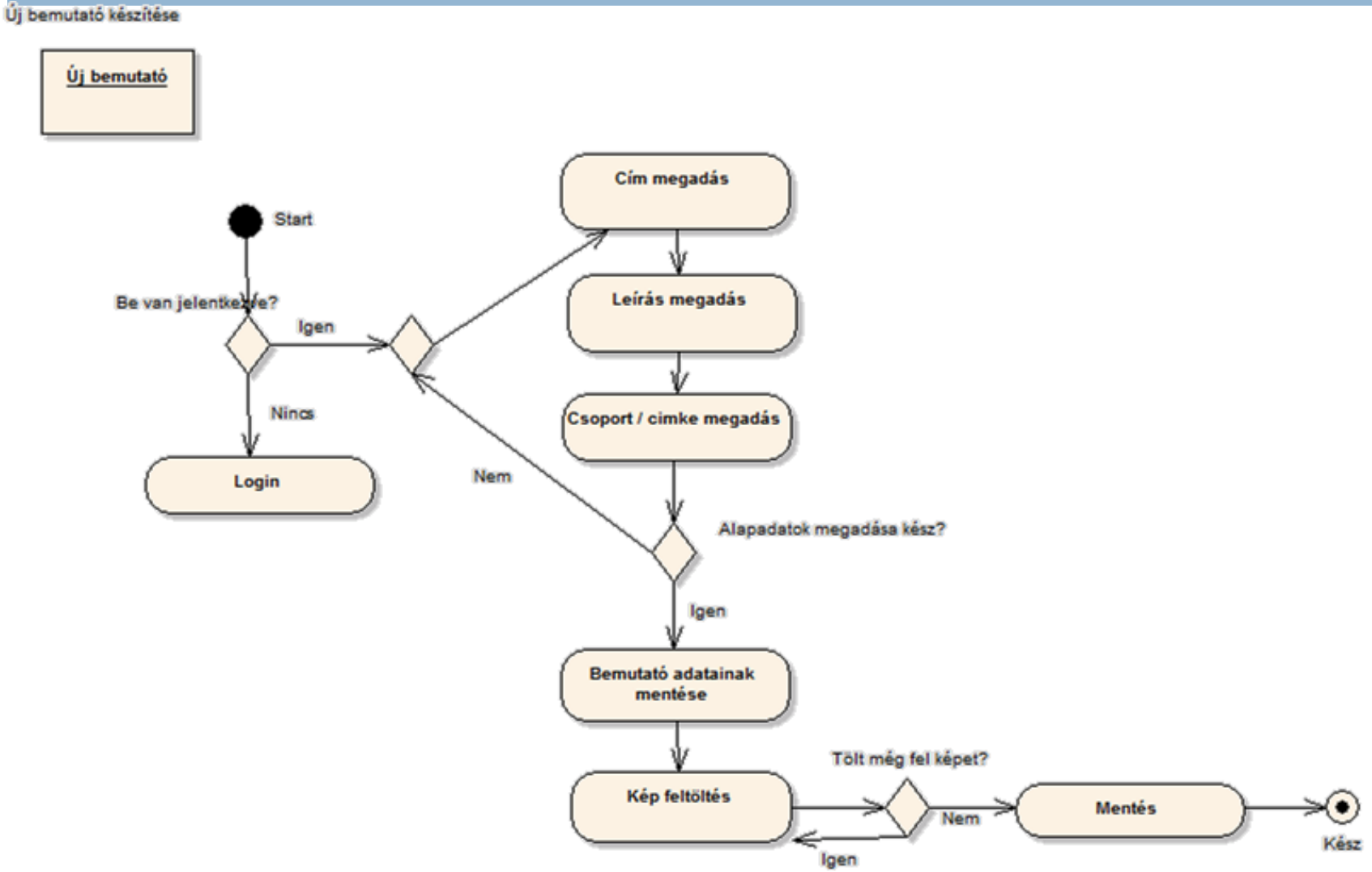
Oldalak és funkciók tervezése

- Főoldal
  - Hivatkozás a bejelentkezésre és a regisztrálásra
  - 10 legnépszerűbb bemutató listája
    - Album adatai: indexkép, címe, leírása, hányszor tekintették meg
    - Funkciók: egy bemutatóra kattintva betöltődik a bemutató
  - Egy véletlenszerűen kiválasztott album vetítése
- Bemutató megtekintése
  - Bemutató adatai...

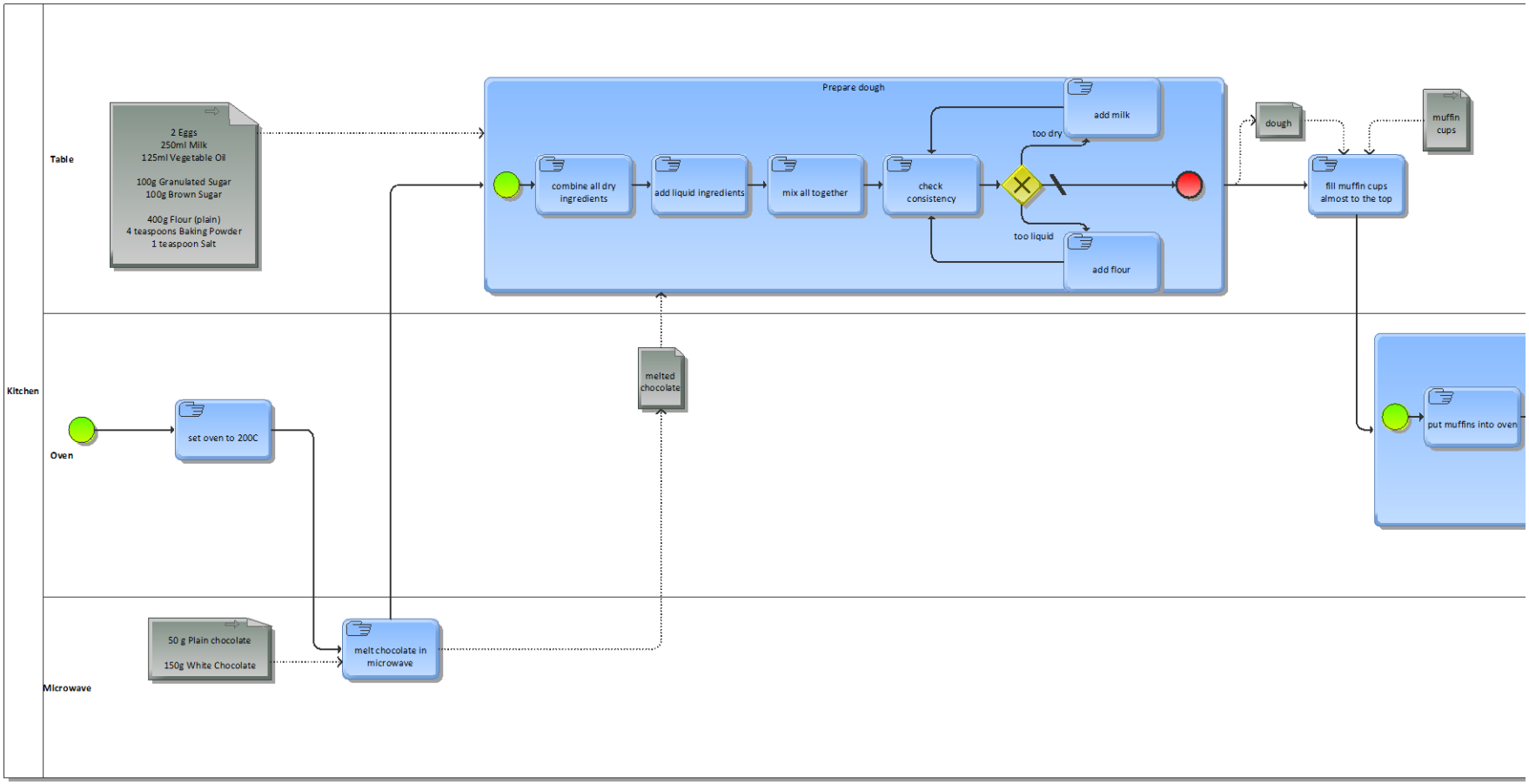
Use case – használati eset diagram



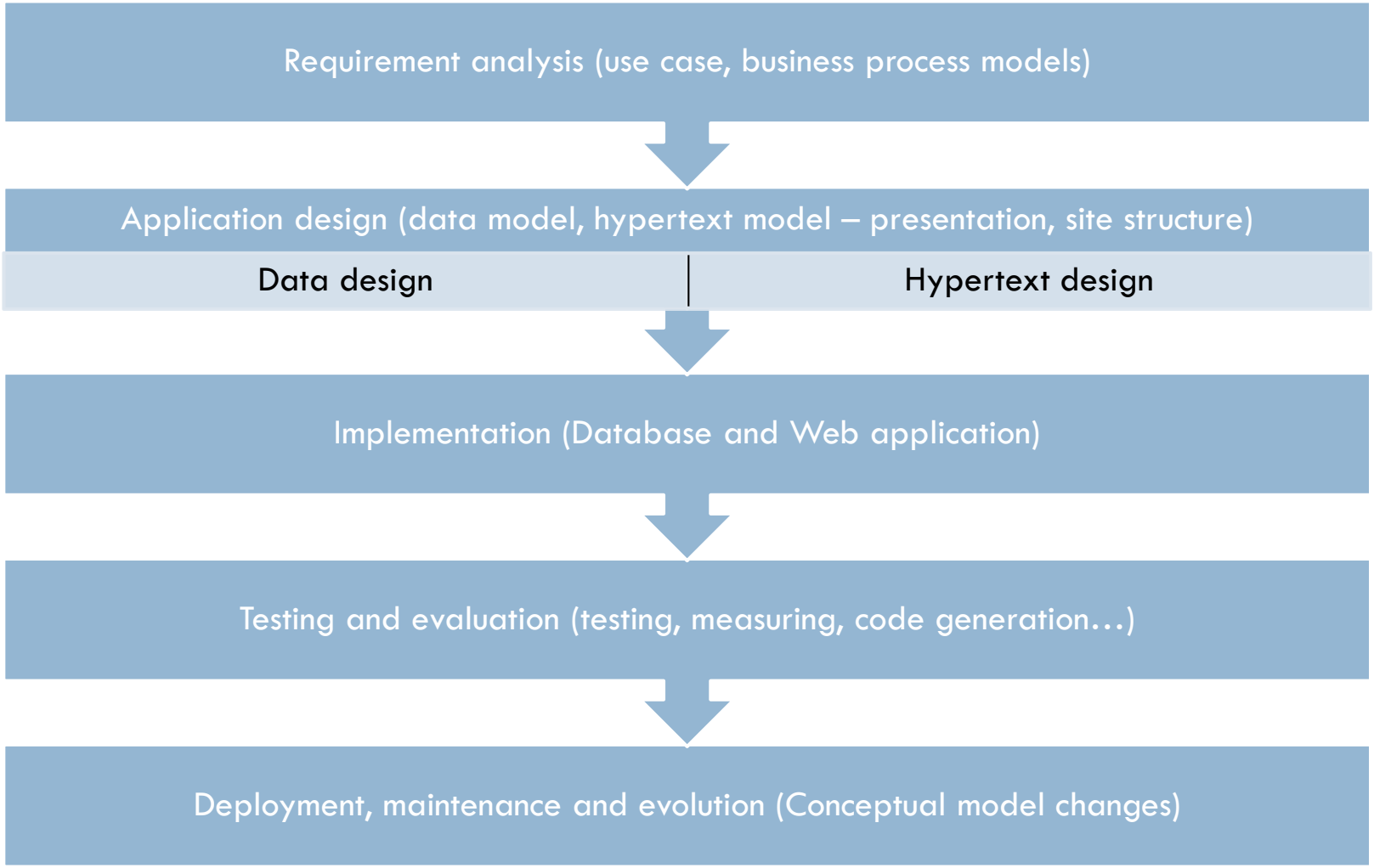
Activity diagram



Business Process Model (BPM)



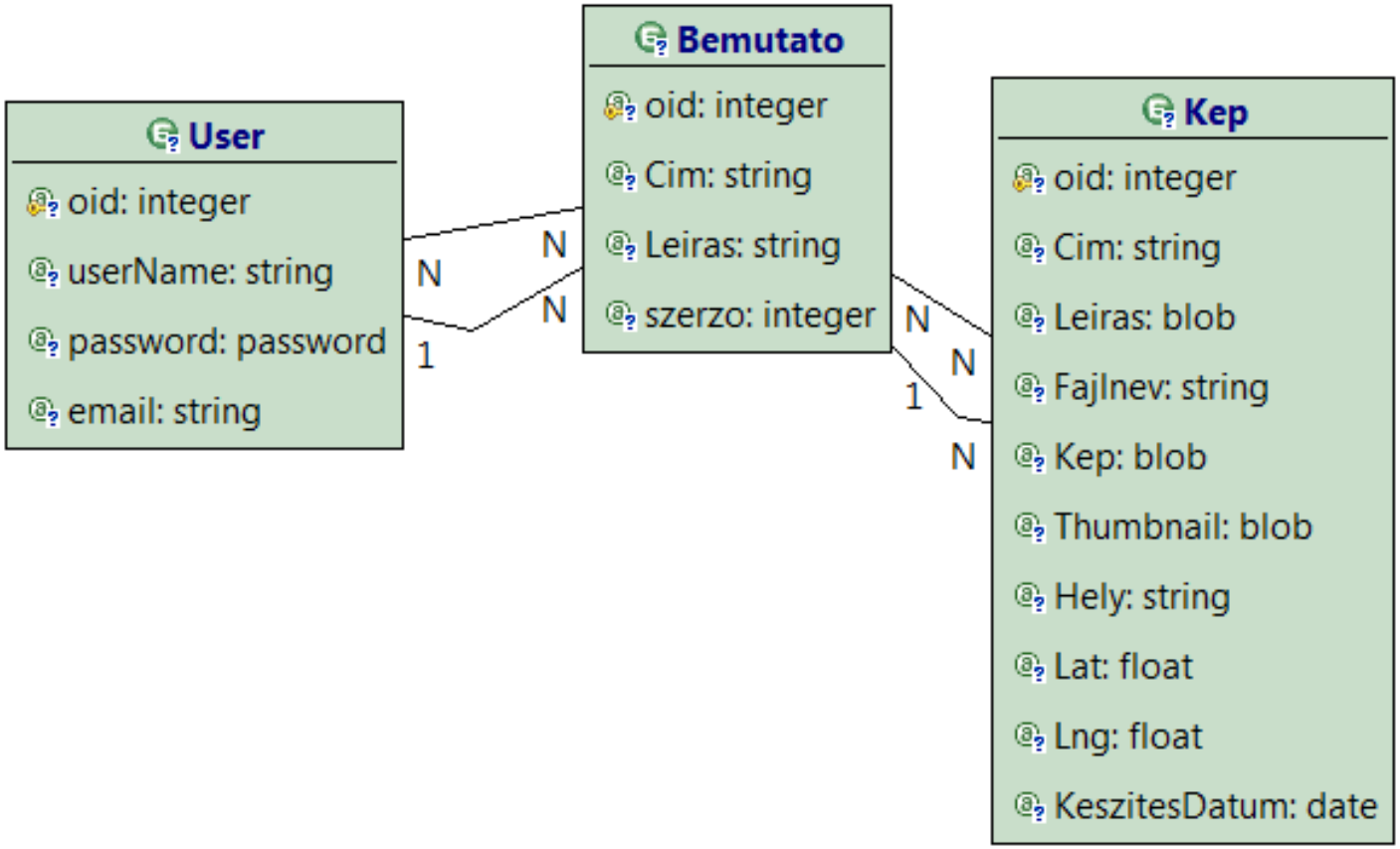
WebML Development Processes



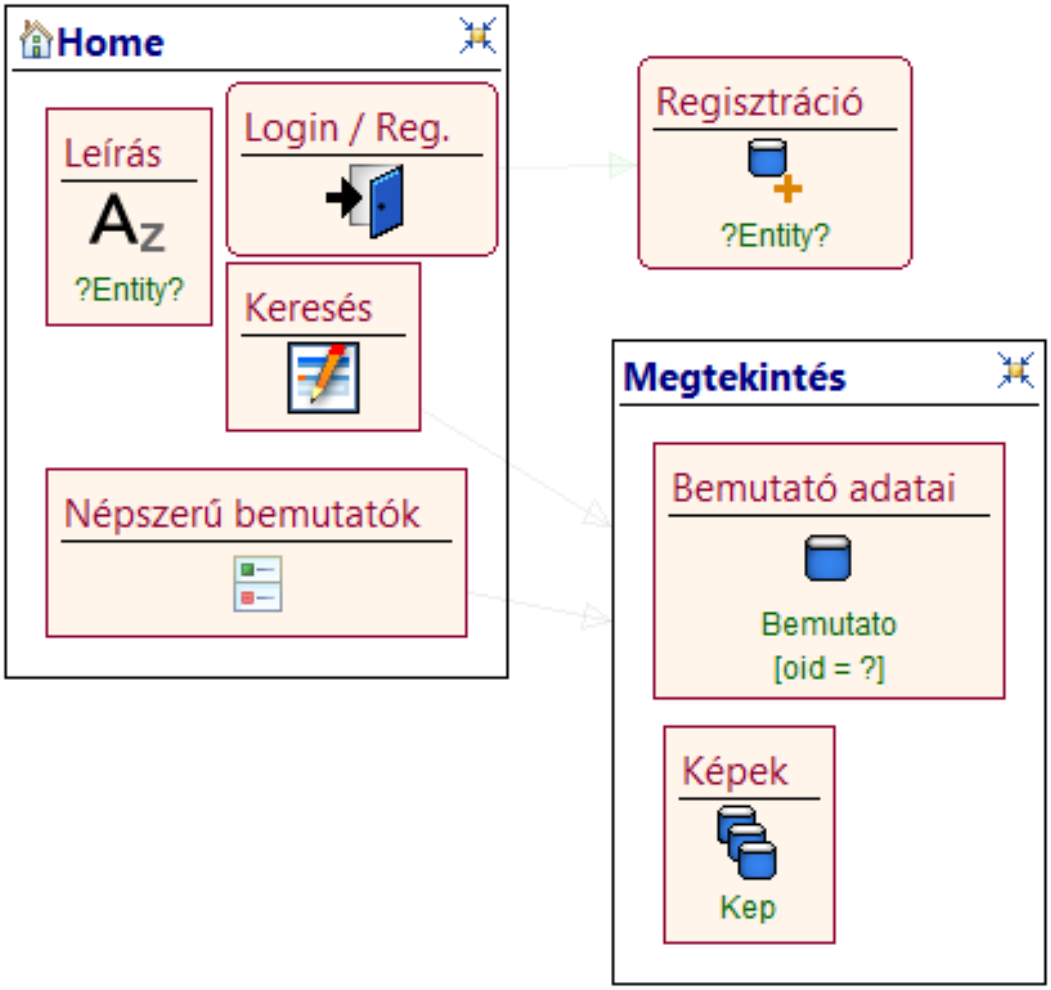
Adatbázis tervezés

1. **Cél meghatározás, a feladat:** Meghatározzuk a **tárolandó adatok körét,** az adatbázis **használatának módját,** az **elvégzendő részfeladatokat.**
2. **Logikai (koncepcionális) adatmodell készítése**
3. **Fizikai adatmodell készítése**
4. **Táblák meghatározása:** Az összegyűjtött információkat témakörökre**, táblákra bontjuk** (***normalizálás***). Kerülni kell a többszörös adatbevitelt, de minden szükséges adatot tárolni kell.
5. **A táblák mezőinek meghatározása, funkcionális függőségek megállapítása**
6. **Kapcsolatok felállítása a táblák között**
7. **Teszt változat elkészítése, a terv finomítása**
8. **Üzembehelyezés**
9. **Karbantartás**

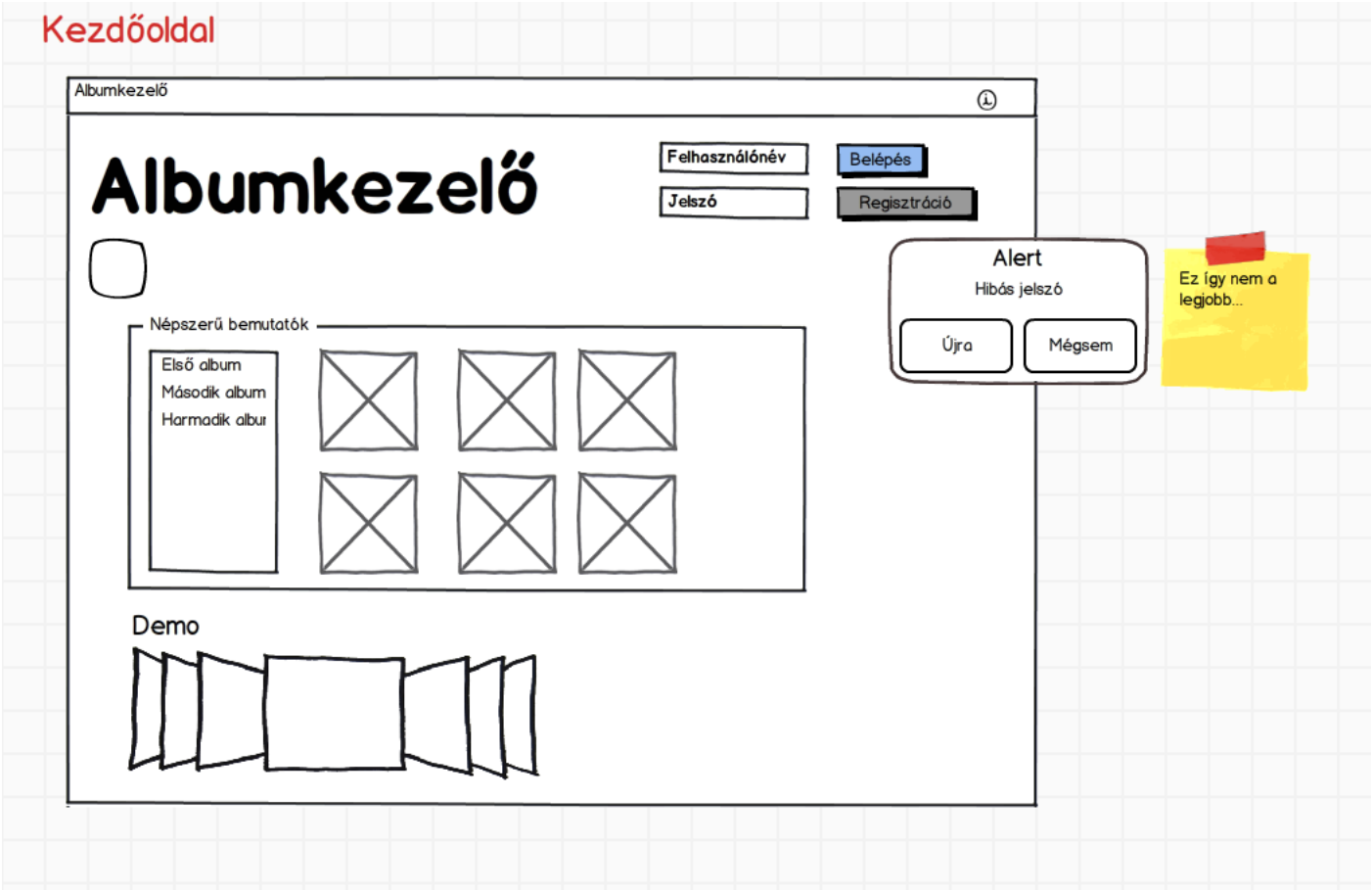
Data model


User
oid: integer
userName: string
password: password
email: string
Bemutato
oid: integer
Cim: string
Leiras: string
szerzo: integer
Kep
oid: integer
Cim: string
Leiras: blob
Fajlnev: string
Kep: blob
Thumbnail: blob
Hely: string
Lat: float
Lng: float
KeszitesDatum: date
N
N
1
N
N
N
1
N

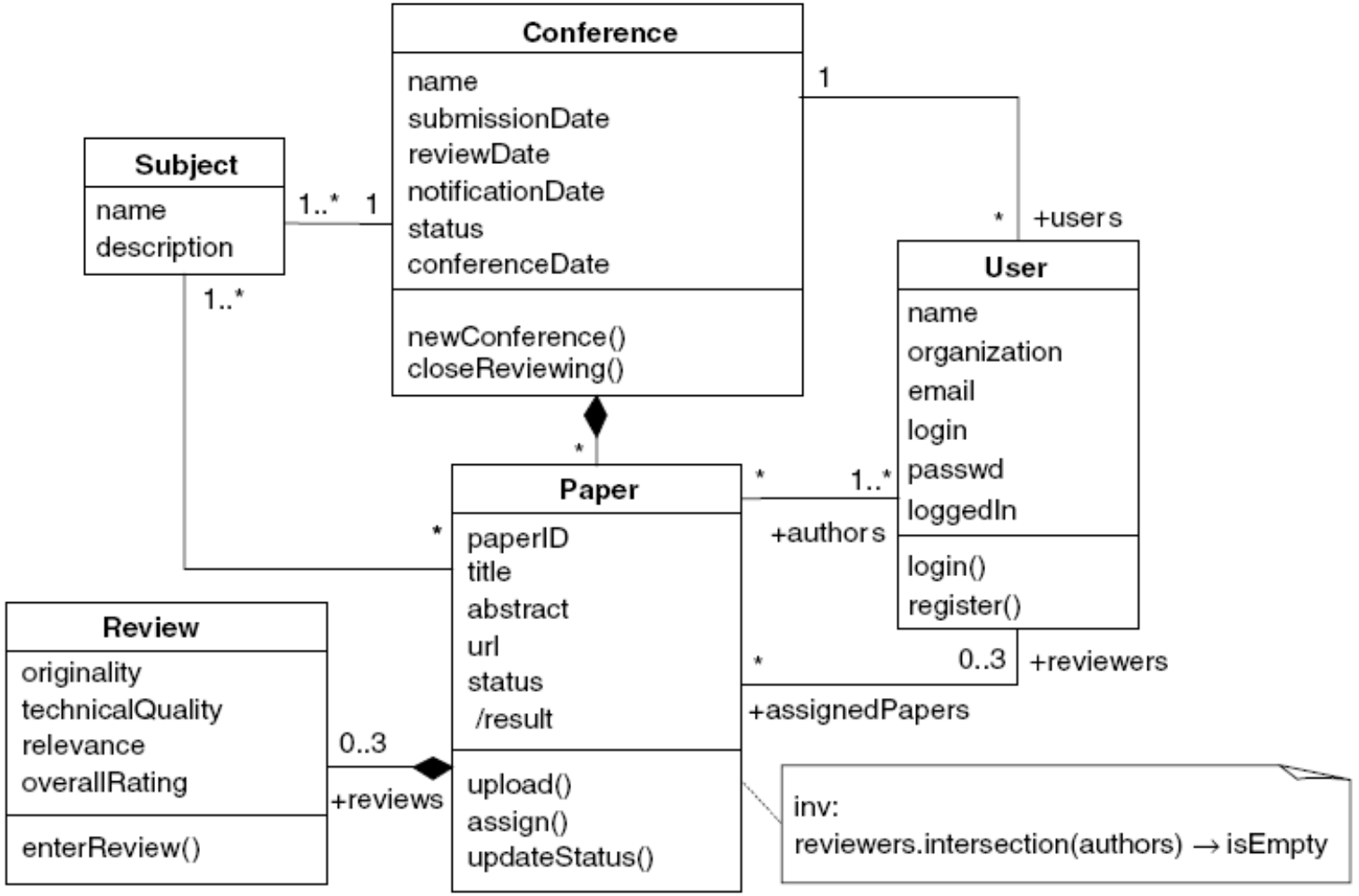
Site structure –hypertext model



Presentation: Mock-up, majd design



Osztálydiagram



Fejlesztői - megrendelői „evolúció”

Fejlesztői evolúció 1. szint – Kezdeti

- Fejlesztő oldaláról
  - Legtöbb web programozó, HTML-t "írók".
  - Statikus web lapok
  - WYSIWG szerkesztők, szövegszerkesztők
  - Nem használnak mintákat, sablonokat
  - Nem használnak fejlesztést segítő eszközöket
  - Tesztelés hiányzik vagy kezdetleges
  - Nem jellemző a program logika
  - Kis csapat, kezdetleges oldalak
  - Nincsenek elkülönült szerepek
- Megrendelő szempontjából
  - Elsődleges cél a jelenlét az Interneten
  - Kevés, ritkán változó tartalom
  - Csak egy ún. elektronikus prospektus oldalt várnak el
  - Kevés visszatérő látogató (ha van egyáltalán)
  - Nincs web-es stratégia, vagy cél

Fejlesztői evolúció 2. szint – Ismételhető

- Fejlesztői oldalról
  - A hagyományos web programozást segítő tananyagok, könyvek segítségével ezt a szintet lehet elérni
  - Tapasztalat útján, sok megrendelést követően juthat el ide a cég → a fejlesztő cég mérete növekedik
  - Elkezdenek újrafelhasználható komponenseket, sablonokat használni
  - Dinamikus weblapok megjelenése, kezdetleges program logikával. → A növekvő megrendelési igények miatt is
  - Típus hiba: kísérletező fejlesztő, a legújabb technológiákat használja a "szép oldalakért", de a funkcionalitás rovására.
  - Megjelenhetnek a tartalomkezelő rendszerek (CMS), de gyakran még tervezetlenül
- Megrendelő szempontjából
  - A megrendelő is fejlődik, a tartalom frissessége is számít már.
  - A megrendelő szeretné a tartalmat maga alakítani

Fejlesztői evolúció 3. szint - Meghatározott

- Megrendelő szempontjából
  - A marketing stratégia és a web stratégia összetalálkozik
  - → Konkretizálódnak az elvárások
  - Vevőkkel, partnerekkel is elektronikusan akarják tartani a kapcsolatot
  - Intranet oldalak megjelenése
- Fejlesztő oldaláról
  - E-kereskedelmi, ügyfélszolgálati szolgáltatások megjelenése
  - Keretrendszer, CMS használat már bevett szokás
  - Profi fejlesztő csapat szükséges
  - Használnak már fejlesztő, tervező eszközöket.
  - Biztonsági elvárások is megjelennek
  - Folyamatos fejlesztői képzések
  - Adatbázisok használata
  - A fejlesztői szerepek szétválnak: programozó, adatbázis és (web) server adminisztrátor, designer

Fejlesztői evolúció 4. szint – Menedzselt

- Megrendelő szempontjából
  - Tartalomkezelő rendszer használata szükséges
  - Belső portál az alkalmazottaknak és a partnereknek
  - Profi belső üzemeltetői csapat is kellhet (nem minden esetben!)
- Fejlesztői oldalról
  - Web service – Szolgáltatás Orientált Architektúra
  - Architektúra tervezés
  - Web 2.0 alkalmazások megjelenése
  - Projektmenedzsment a középpontba kerül
  - Munkafolyamat-kezelés
  - Célok, eredmények mérése, értékelése
  - További szintek: tervező, rendszerszervező, tesztelő
  - Állandó együttműködés a megrendelővel
  - Menedzselhető fejlesztő rendszerek: J2EE, .NET.

Fejlesztői evolúció 5. szint – Optimalizáló

- Megrendelő oldaláról
  - ERP funkciók az Intraneten,
  - Az elkülönült IT rendszerek összeköttetése,
  - A piac gyors, rugalmas reakciókat vár el
- Fejlesztő oldaláról
  - A szervezet fejlődik, alkalmazkodik, tanul!
  - A fejlesztői folyamatok folytonos változása a legfontosabb.
  - Produktivitás, hatékonyság, a megrendelői elvárások kerülnek a középpontba.
  - Hiba megelőző, elemző módszerek a fejlesztésben.
  - A termék a lehető legjobb minőségben készül el, határidőre!

Irodalom

- Gerti Kappel, Birgit Pröll, Siegfried Reich, Werner Retschitzgger: **Web Engineering**, John Wiley &Sons, 2006.
- Sven Casteleyn, Florian Daniel, Peter Dolog, Maristella Matera: **Engineering Web Applications**, Springer, 2009
- Horváth Győző, Tarcsi Ádám, Menyhárt László: **Többrétegű webes alkalmazások fejlesztése**, ELTE, 2012
- Tarcsi Ádám: **Quality measurement of Business WEB Applications**, Serdica Journal of Computing, 2008. pp. 31-44.
- Dr. Johanyák Zsolt Csaba: **Szoftvertechnológia előadás**, KF, 2010