8. RUP tervezési fázisa. Az elemzés eredményeinek átvétele a tervezésben

A tervezésnek a készítendő rendszerről teljes mélységben részletes képe kell adnia, a kiválasztott programozási nyelvekre és technológiákra vonatkozóan is, így meg kell adnia az implementáláshoz szükséges dokumentumokat, annak ütemezéséhez a rendszer kezelhető részekre kell bontania és meg kell határoznia a különböző részek közötti interfészeket.

A tervezés nem egy egy lépcsős folyam, általában többszörös iterációból és visszalépésekből áll.
A tervezés első lépése az **architektúrális tervezés**, mellyel meghatározzuk és dokumentáljuk a rendszert felépítő alrendszereket és a köztük lévő kapcsolatokat.
Ezen keretek tartalmát a **használati esetek tervezési** szintű megvalósításával töltjük fel, majd az osztályokra fókuszálva összegyűjtjük az azokkal kapcsolatos **implementációs követelményeket.**

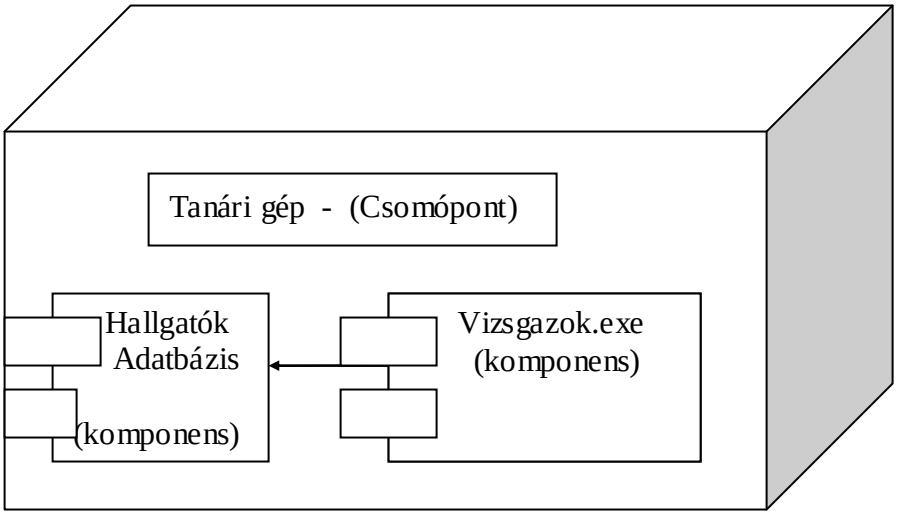
A tervezés eredménye a **tervezési modell**, amely a használati esetek fizikai szintű megvalósítását írja le.
A tervezés másik eredménye a **telepítési modell**, amely a rendszer hardver elemei és a közöttük lévő fizikai viszonyokat és a hardver és szoftver elemek közti kapcsolatot írja le.
(A rendszer futás közbeni topológiáját adja ábrázolja.)

Összetevő diagram (Komponens diagram)
A rendszert alkotó fizikai összetevőket (szoftverelemeket) és az azok közti kapcsolatokat ábrázolja.
Az összetevő diagramon megadható a logikai nézet osztályainak forrásösszetevőkhöz való hozzárendelése, valamint a forráskódok hozzárendelése futtatható összetevőkhöz. A

- Szokványos összetevők:
  - Executable végrehajtható program
  - Objektumkönyvtár, statikus vagy dinamikus
  - Adatbázis-tábla
  - Fájl, amely adatokat tartalmaz
  - Dokumentum (szöveges, kép, hanganyag stb.)

Telepítési diagram:
A telepítési diagram a telepítési modell diagramja.
Itt adjuk meg a rendszer topológiáját.
A telepítési diagram segítségével a rendszer szoftver elemeit a hardver elemekhez rendeljük.
Megadjuk,hogy mely összetevők(szoftver elemek) milyen számítógépen futnak.

Elemei:
Csomópont: egy hardver elem, processzor, számítógép vagy egyéb eszköz.
A csomópont sztereótpusa például: pc, pc-kliens, pc-szerver
Kapcsolat: A csomópontok közötti kapcsolat lehet társítási, öröklési, tartalmazási, stb.
A kapcsolat sztereo típusa lehet pédául: lokális háló, TCP/IP
Komponens: A csomópontokra komponensek tehetők. Ez jelzi,hogy a komponens a hardver elemen van, és fut.



Adatbázis tervezés:
A tervezés során perzisztens osztályokat határozunk meg.
A perzisztens osztályok tárolására megfelelő adatbázis szerkezetet határozunk meg.
A megfelelő teljesítmény eléréséhez megfelelő mechanizmusok és stratégiák kidolgozása adatok tárolására és visszakeresésére.
A perzisztens tervezési osztályokat leképezzük adatmodellre.
A leképezést elősegítő fél automatikus eszköz: Java Hibernate.

+++

8. A RUP tervezési fázisa. Az elemzés eredményeinek átvétele a tervezésbe. Adatbázis- és valósidejű tervezés.

A tervezés során az elemzés által meghatározott keretek töltjük fel tartalommal és formáljuk a rendszert teljes alakká, amely teljesíti a korábban meghatározott összes funkcionális és nem-funkcionális követelményeket.

A Unified Process a tervezés célját a következő főbb pontokban foglalja össze:

- A tervezésnek a készítendő rendszerről teljes mélységében részletes képet kell adnia, a kiválasztott programozási nyelvekre, operációs rendszerre, technológiákra vonatkozóan is.
- Össze kell állítania az implementációhoz szükséges dokumentumokat.
- Az implementáció ütemezéséhez a rendszert kezelhető részekre kell bontania.
- Meg kell határoznia a különböző részek közötti interfészeket.
- Adott jelölési rendszert felhasználva a tervezési döntéseket egzaktul meg kell adnia.
- Meg kell rajzolni a rendszer implementációjának a vázát.

Az elemzéssel összehasonlítva, a tervezés nem a fogalmi modell kialakítására törekszik, hanem a rendszernek az annál konkrétabb fizikai modelljét határozza meg, amely az implementáció „alaprajza” *(blueprint)* lesz. A tervezés már konkrét platformokra és technológiákra épít. Az „alaprajznak” már formálisnak és pontosnak kell lenni, hogy az alapján az implementáló programozók további bizonytalanság nélkül felépíthessék a teljes rendszert.

A tervezés munkafolyamatának a legfontosabb eredménye a *tervezési modell (design modell),* ami egy olyan objektum modell, amely a használati esetek fizikai megvalósítási módját írja le, figyelembe véve az összes

funkcionális és nem-funkcionális követelményt, valamint a megvalósítási környezetet. A tervezési modell a megvalósított rendszer áttekintő terveként szolgál.
A tervezési modellben alapvető szerepűek a *tervezési osztályok (design class)*, amely az implementációs osztályok közvetlen absztrakciója.

A tervezés másik eredménye a telepítési modell *(deployment model)*, amely a rendszer részeinek az ún. csomópontokon (pl. számítógépeken) való fizikai elhelyezkedését ábrázolja.
A telepítési modellben:

- minden csomópont valamely számítógépet, vagy hasonló jellegű hardware elemet jelöl,
- a csomópontok közötti kapcsolatok a közöttük lévő kommunikáció eszközére utalnak, pl. Internet, intranet, soros vonal.

Munkafolyamat:

1. Architekturális tervezés

Az architekturális tervezést a *csomópontok és hálózati konfigurációk meghatározásával* kezdjük. A hálózati konfigurációt általában a három-rétegű modell *(three-tier)* alapján szervezzük meg, ahol az első réteg a felhasználói felületeket tartalmazó kliensprogramot jelenti, a középső réteg az üzleti logikát megvalósító szerver-program, a nagymennyiségű adatok tárolása pedig a harmadik réteg adatbázis-kezelőjével történik.
A következő lépés az *alrendszerek és interfészeik azonosítása*. Az alrendszerekkel a tervezési modellt kezelhető egységekre bontjuk fel.
A felbontás történhet a tervezés kezdetén, vagy munka közben, a rendszer bővülésével. A rendszert általában több csomagot is magába foglaló rétegekre *(layer)* bontjuk, ahol is egy réteg csomagjai csak egymásra vagy az alsóbb rétegekre hivatkoznak. A diagramokon célszer_ ábrázolnunk az alrendszerek közötti függőségeket *(dependency)*, mivel így könnyebb a későbbi változtatások tervezése. Végül a függőségek alapján meghatározzuk, hogy az alrendszerek közötti kapcsolat milyen interfészeken keresztül valósul meg.
Az *architekturálisan lényeges tervezési osztályok azonosításának* tevékenységét gyakran a tervezési munka kezdetén hajtjuk végre. A kiindulópontunk az architekturálisan lényeges *elemzési* osztályok, melyek általában tervezési osztályokként is megjelennek.

2. Használati eset tervezése

A használati eset tervező a tervezés során egyenként megvizsgál egy vagy több használati esetet és leírja azok tervezési szintű megvalósítását. Először *azonosítjuk a résztvevő tervezési osztályokat*.
A használati eset megvalósításához leírjuk az objektum-interakciókat. A kiindulópontjaink a tervezési osztályok, valamint a használati eset elemzési szintű megvalósítása. Az interakció ábrázolására a Unified Process a szekvencia-diagramokat ajánlja, mely konkrét aktorokat, tervezési objektumokat és a közöttük lezajló üzenetváltásokat tartalmazza.

3. Osztály tervezése

Az osztály tervezésekor kiemelünk és *körvonalazunk egy-egy tervezési osztályt*, megadva annak az alapvető kereteit.

- Határ-osztályok *(boundary class)* tervezése az alkalmazott felhasználói felület, illetve más illesztési felület technológiájára épül. A technológia kiválasztása így az osztály alapszerkezetét is meghatározza.
- Tárolt („perzisztens”) információkat reprezentáló entitás-osztályok *(entity class)* alapszerkezetét a tárolási módszer határozza meg, így igényelheti például a relációs adatbázis táblájából történő betöltést, illetve mentést.
- A vezérlő osztályok *(control class)* alapszerkezetét a vezérlés jellege határozza meg, így általában figyelembe kell venni az osztottsággal, hatékonysággal, esetleg a szinkronizációval kapcsolatos problémákat. A vezérlő osztályok gyakran adott tranzakciót kísérnek végig, így kialakításuk a használt tranzakciós technológiát követi.

Az osztály *műveleteinek azonosításakor* a metódusok szignatúráját (nevét, paramétereit) és leírását a használt programozási nyelv terminológiájával és az ott érvényes szintaktikai szabályoknak megfelelően írjuk le.

Az attribútumok azonosításakor ugyancsak a használt programozási nyelv szintaktikáját alkalmazzuk. Az attribútumok az osztály jellegzetességei, melyek a legtöbb esetben a műveletek végrehajtásához szükségesek.

Az attribútumok mellett *azonosítanunk* kell az *asszociációs és aggregációs viszonyokat*. Az objektum-interakciók gyakran megkövetelik a megfelelő osztályok közötti asszociációkat.
Az attribútumok, műveletek és asszociációk segítségével már könnyen *azonosíthatjuk az* osztályok közötti *általánosítás-pontosítás viszonyokat*, illetve meghatározhatjuk a már felvett osztályok alapján azok általános változatait, amelyekbe a közös, ismétlődő jellegzetességeket kiemelhetjük. A jelölésnek a programozási nyelven elérhető konkrét technikára kell utalni, mely objektumorientált nyelv esetén általában az öröklés *(inheritance)*.

4. Alrendszer tervezése

Az alrendszer tervezésekor *ellenőriznünk* kell az *alrendszer függőségeit*, hogy az alrendszer a lehető legkevésbé függjön a többi részrendszertől.

A tervezés további feladata az *alrendszer interfészeinek ellenőrzése*. Az interfészek által definiált műveleteknek teljesítenie kell az összes olyan, az alrendszerre vonatkozó követelményt, melyet a különböző használati eset megvalósítások során játszott szerepe megkövetel.

A tervezés során az *alrendszerek tartalmát* is *ellenőrizni* kell, hogy az interfészekben meghatározott műveletek megvalósítása a követelményeknek megfelelően hajtódik-e végre.

A tervezésnek a készítendő rendszerről teljes mélységben részletes képe kell adnia, a kiválasztott programozási nyelvekre és technológiákra vonatkozóan is, így meg kell adnia az implementáláshoz szükséges dokumentumokat, annak ütemezéséhez a rendszer kezelhető részekre kell bontania és meg kell határoznia a különböző részek közötti interfészeket.

A tervezés nem egy egy lépcsős folyam, általában többszörös iterációból és visszalépésekből áll.
A tervezés első lépése az **architektúrális tervezés**, mellyel meghatározzuk és dokumentáljuk a rendszert felépítő alrendszereket és a köztük lévő kapcsolatokat.
Ezen keretek tartalmát a **használati esetek tervezési** szintű megvalósításával töltjük fel, majd az osztályokra fókuszálva összegyűjtjük az azokkal kapcsolatos **implementációs követelményeket.**

A tervezés eredménye a **tervezési modell**, amely a használati esetek fizikai szintű megvalósítását írja le.
A tervezés másik eredménye a **telepítési modell**, amely a rendszer hardver elemei és a közöttük lévő fizikai viszonyokat és a hardver és szoftver elemek közti kapcsolatot írja le.
(A rendszer futás közbeni topológiáját adja ábrázolja.)

Adatbázis tervezés:
A tervezés során perzisztens osztályokat határozunk meg.
A perzisztens osztályok tárolására megfelelő adatbázis szerkezetet határozunk meg.
A megfelelő teljesítmény eléréséhez megfelelő mechanizmusok és stratégiák kidolgozása adatok tárolására és visszakeresésére.
A perzisztens tervezési osztályokat leképezzük adatmodellre.
A leképezést elősegítő fél automatikus eszköz: Java Hibernate.

A **valósidejű rendszer** definíciója: a rendszer helyes működése egyrészt a rendszer által adott eredményektől, másrészt attól az időtől függ, amennyi idő alatt ez az eredmény előáll.
Valósidejű egy rendszer, ha a specifikációjában valamilyen időbeli viselkedésre vonatkozó előírás szerepel a külső, valós időhöz kötötten.

A valós idejű rendszerek tervezésének lépései:
– Meg kell határozni a rendszer által feldolgozandó ingereket és az azokra adott válaszokat.
– Minden inger-válasz párhoz meg kell határozni az időzítési megszorításokat.
– Választani kell a rendszer számára egy végrehajtási platformot, vagyis hardvert és egy valós idejű operációs rendszert (futtatórendszert) kell választani.
– Az inger és a válasz feldolgozását konkurens folyamatokba kell győjteni. Az ingerek és válaszok minden osztályához egy folyamatot rendelünk.
– Minden ingerre és válaszra meg kell tervezni a szükséges számításokat végző algoritmusokat.
Ezt gyakran a tervezés korai szakaszában kell megtenni, hogy kiderüljön a feldolgozási műveletek időigénye.
– Ki kell alakítani az ütemezési rendszert, amely biztosítja, hogy a folyamatok időben elindulnak és határidőre befejeződnek.
– A rendszert integrálni kell egy valós idejű futtatórendszer vezérlése alatt.