9. Összetevő- és telepítési diagramok. Kódgenerálás és visszafejtés CASE eszközökkel.

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.

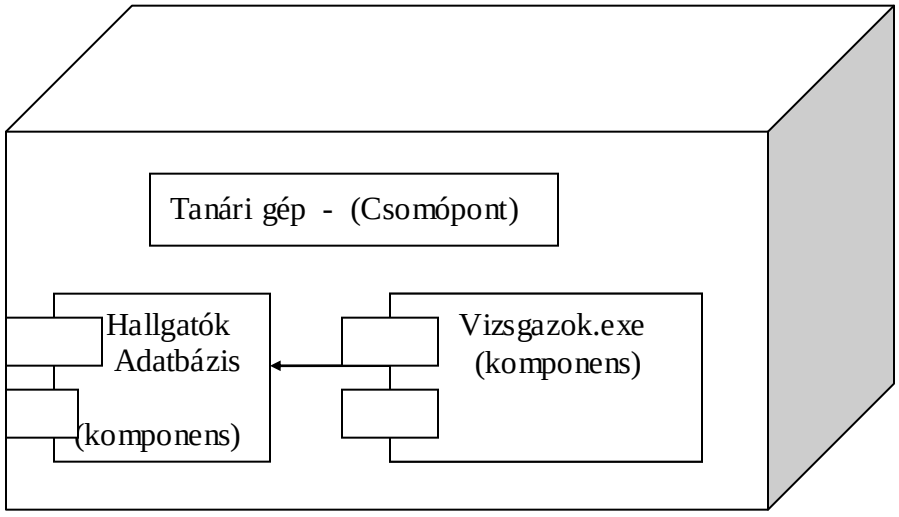
- Szokványos összetevők:
  - Executable végrehajtható program
  - Objektumkönyvtár, statikus vagy dinamikus
  - Adatbázis-tábla
  - Fájl, amely adatokat tartalmaz
  - Dokumentum (szöveges, kép, hanganyag stb.)

A komponensdiagram a probléma megoldására szolgáló rendszer tulajdonságait implementációs szempont szerint fejezi ki. A komponensdiagram alapfogalmai:
- komponens,
- reláció.

A *komponens* a rendszer egy fizikailag létező és kicserélhető része, feltéve, hogy a kicseréléséhez az új komponens csatlakozási felületét a környezettel konform módon valósítjuk meg, azaz a környezethez az új komponens csatlakozási felületét hozzáillesztjük.

A komponensek közötti relációkat két csoportba sorolhatjuk:
- fejlesztés során fennálló reláció
- meghívási reláció

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 például: lokális háló, TCP/IP
Komponens: A csomópontokra komponensek tehetők. Ez jelzi,hogy a komponens a hardver elemen van, és fut.



***CASE**: Computer - Aided Software Engineering,* azaz számítógéppel segített szoftvertervezés.
A szoftverek fejlesztéshez természetesen mindenképpen kell a számítógép, de a rendszertervezésnél, ami a CASE-eszközök területe, nem feltétlenül.

A CASE eszközök célja a szoftverfejlesztés elemzési és tervezési fázisaiban végzett munka hatékonyságának növelése:

- ☐ a rendszer komponenseinek és működésének feltárásával
- ☐ a fejlesztési információk tárolásával és elemzésével
- ☐ a projekt irányításával
- ☐ a fejlesztéshez kapcsolódó információk és teendők adminisztrálásával.

Elvégzi az automatikus ellenőrzéseket, listákat, ábrákat generál, kereszt-hivatkozásokat készít, támogatja a dokumentálást. Természetesen egyetlen fejlesztésnél sem kötelező a CASE-eszköz használata, de a tervezés segít abban, hogy a felhasználó és a fejlesztő kommunikációja egyértelmű legyen. Ideális esetben csak olyan rendszertervet valósítanak meg, amelyet a felhasználó megértett és jóváhagyott. Ez a tervezés persze időigényes dolog, és sok idő manapság már nincs a fejlesztésekre; de a tervezésre fordított idő a későbbiekben sokszorosan megtérül, például ha nem kell az alapoktól újraépíteni a rendszert.

A szoftverfejlesztés teljes életciklusát átfogó, az egyes fázisok munkáját összehangoló technikák, a **CASE** eszközök jelentették, amelyek segítik áthidalni a szervezetek valós problémája és a számítógépes rendszer közötti különbségeket.

Azokat a szoftver-rendszereket, amelyek támogatják, illetve automatizálják a problémadefiniálási, -elemzési, tervezési, modellezési, kivitelezési tevékenységet, elvégzik a tesztelési, rendszerkövetési-, karbantartási és minőségbiztosítási feladatokat, dokumentálják a fejlesztést, valamint irányítják, ellenőrzik és összehangolják a fejlesztő projekt munkáját számítógéppel támogatott fejlesztési eszközöknek, CASE rendszereknek nevezzük.

A CASE eszközök feladata, hogy a problémák a korábbinál sokkal mélyebb absztrakciójával logikailag hibátlan, ellenőrzött adat- és feladatleírást hozzanak létre.

A CASE lehetőségek olyan eszközök, amelyek célja a rendszerfejlesztési tevékenység elemzési és –tervezési fázisában végzett munka hatékonyságának növelése

- a rendszer komponenseinek és működésének feltárásával,
- a fejlesztési információk rögzítésével és elemzésével,
- a projekt irányításával, valamint
- a fejlesztéshez kapcsolódó információk és teendők adminisztrálásával.

A számítógéppel támogatott szoftvertervezéshez (Computer-Aided Software Engineering - CASE) használt szoftvereket nevezzük CASE-eszközöknek.
A szoftverfolyamatban a következő tevékenységeket támogatják a CASE eszközök:
Követelményspecifikáció során: grafikus rendszermodellek, üzleti és domain (a modellezni kívánt terület) modellek megtervezése.
Elemzés/tervezés során: adatszótár kezelése, mely a tervben található egyedekről és kapcsolataikról tartalmaz információt; felhasználói interfész generálását egy grafikus interfészleírásból, 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.

Mindegyik fázisban alkalmazható: automatikus dokumentumgenerálás; projektmenedzsment támogatás (ütemezés, határidő 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.
Az eredményességet két tényező korlátozza:
• A szoftvertervezés lényegében tervezői tevékenység, amely kreatív gondolkodást igényel. A létező CASE-eszközök automatizálják a rutintevékenységeket és hasznosítják a mesterséges intelligencia bizonyos technológiáit, de ez utóbbival még nem értek el átütő eredményt.
• A legtöbb szervezetben a szoftvertervezés csoportos tevékenység, és a benne résztvevők rengeteg időt töltenek a csapat más tagjaival való eszmecserével. A CASE-technológia ehhez nem nyújt túl nagy segítséget.

CASE eszközök

Mi a CASE-eszköz?
CASE = **Computer Aided Software Engineering.**
A CASE-eszközök olyan szoftver rendszerek, amelyek a szoftver folyamat tevékenységeit támogatják, automatizálják. általában konkrét módszerekhez kapcsolódnak.
Magas szintu CASE-eszközök:
A szoftverfolyamat kezdeti lépéseit támogatják: elemzés, tervezés, modellezés, rendszer dokumentálás, jelentés-készítés, stb.
Alacsony szintu CASE-eszközök:
A szoftverfejlesztés későbbi tevékenységeit támogatják, mint kódszerkesztés (kódgenerálás!), kódelemzés, nyomkövetés tesztelés, stb.

Egy 1979-es vizsgálat eredménye szerint a megkezdett programok fele elkészül, de sohasem használják, negyede el sem készül, negyede kerül végül használatba, de nagyobbrészben utólagos nagyobb mérvu átalakítás után.
Akkori tapasztalatok szerint a hibajavítás a programozási idö 80 %-át teszi ki.
Hosszú a "**belövési ido**": amíg a rendszert a felhasználó igényeinek megfelelö minöségben muködö állapotba hozzák.

A CASE-eszközök szerepe

Követelményspecifikáció során: grafikus rendszermodellek, üzleti és domain modellek
Elemzés / tervezés során: adatszótár kezelése (a tervben található egyedek és kapcsolataik), felhasználói interfész generálása egy grafikus interfész leírástól a terv ellentmondás mentességgel vizsgálata.
Implementáció során:

- automatikus kódgenerálás (Computer Aided Programming – CAD)
- verziókezelés

Szoftvervalidáció során:

- automatikus teszteset – generálás
- teszt kiértékelés, dokumentálás

Szoftverevolúció során:

- forráskód visszafejtés (reverse engineering),
- régebbi verziójú programoknak automatikus újrafordítása az új verzióra

Minden fázisban:

- automatikus dokumentum generálás
- projektmenedzsment támogatás (ütemezés, határidők figyelése, erőforrás kiosztás, kezelés) 40%
- tervezői tevékenység --> kreatív gondolkodást igényel, csak a rutintevékenységek automatizáltak (MI használata)
- csoportkommunikáció gyenge támogatása

A CASE-eszközök csoportosítása

- funkcionális
- folyamat
- integrációs

Funkcionalitásuk alapján: tervező, szerkesztő, konfigurációkezelő, prototípuskészítő, nyelvi feldolgozó, tesztelő programkezelő, nyomkövető, dokumentációs és újratervezési
Az eszközök által támogatott folyamat alapján
Integrációs szempont szerint:

- Eszköz (Tool): egy folyamatlépést támogatnak
- Eszközkészletek (Toolkits): néhány folyamatlépést támogatnak
- Környezetek (I-CASE Integrated Workbench): a szoftverfolyamat minden lépését támogatják, eszközök teljesen integráltak

Néhány konkrét CASE-eszköz:
iUML; Objectory; Paradigma Plus,Rationel Suite (IBM); ARIS;Forte ADE; Cool termékcsalád

Kódgenerálás és architektúra tervezés helye a szoftvergyárban

Természetes elvárás, hogy egy **CASE** eszközben gondosan elkészített terveinket minél intenzívebben használhassuk fel a szoftver fejlesztés későbbi folyamatai során. Az implementáció a kész tervek alapján nagy mértékben leegyszerűsödik, és jelentősen csökken az esélye annak, hogy sok befektetett energia után derül ki: rossz irányba indultunk el.

Mindemellett a terv-kód kapcsolat automatizálható is. Egész egyszerűen készíthetőek olyan szkriptek, amelyek a tervből kinyerik a deklarációs információkat, azaz létrehozzák az osztályokat, attribútumokat és metódusokat. Így készen kapjuk a program vázát, generáljuk a forráskód 4-5%-át. Nem érdemes azonban itt megállni, több irányban is növelhető a hatékonyság:

- Először is felhasználhatjuk a terv dinamikus részeit, például UML esetében a kommunikációs és állapot diagramokat. Így az intenzív kommunikációt folytató objektumok (tipikusan vezérlő) metódusainak vázaként előállítható a ki és bemenő üzenetek, illetve események. Az állapotgép-szerű működéssel rendelkező objektumok kódja pedig teljes egészében generálható. Ezzel újabb 8-10% nyerhető.
- Amennyiben a **CASE** eszköz testreszabhatósága lehetővé teszi, célnyelv- és projektspecifikus információkat is bevihetünk a tervbe, sőt ezeket vizuálisan meg is jeleníthetjük (plusz 3-5%).

Létezik egy magasabb szintje is a kódgenerálásnak, ami az ismétlődő részletek többszöri alkalmazásán alapul, ez pedig a minta alapú **kódgenerálás**. Alapelve, hogy a többször előforduló részletből készülő forráskódot egyszer kell elkészíteni, később ezt tetszőleges alkalommal mintaként felhasználhatjuk. Ilyen tervezési részlet lehet az adatbázis primitívektől kezdve, az elosztott kommunikációs szabványon át (pl. CORBA) a hiba és kivételkezelésig bármi. Ezzel a módszerrel megszabadulhatunk a programozás "favágó" részétől, lényegében a terv szintjére emelhetjük az újrafelhasználhatóságot. Még tovább léphetünk, ha tervezési és programfejlesztési mintákat használunk, ezt nevezzük architektúra központú fejlesztésnek (Architecture Centric Development). Mintákat készíthetünk egészen bonyolult, de sokszor alkalmazható részletekből, mint például egy bevált cache architektúra vagy teszt felület generátor. Ettől kezdve különválik az alkalmazás üzleti modelljének tervezése a minél használhatóbb architektúra minták tervezésétől, az összekapcsolódási pont a **kódgenerálás** pillanata. A generálás bemenetei tehát az üzleti

modell és a tervezési minták, kimenete a forráskód fájlok, melyek az implementációkor hasznosíthatók. Az így elérhető kódgenerálási szint alkalmazástól függően 30-70%.

A felsorolt technikák jelentős hatással vannak a szoftver fejlesztés hatékonyságára hiszen:

- A terv kompaktabbá és áttekinthetőbbé válik, mentes lesz implementációs részletektől.
- A karbantartás és fejlesztés egyszerűsödik, mert az architekturális változások nem érintik a tervet.
- Kényelmesen és magától értetődően alkalmazható a forward engineering elv.
- A program írásának ideje lerövidül.
- A kódírásnál csökken a hibázás lehetősége.
- A projekten belüli programozási konvenciók automatikusan teljesülnek.
- Egyszerűen megoldható a több nyelvre való generálás.
- A fejlesztés jól szegmentálható, mert a tervezők az üzleti modellre koncentrálhatnak, a programozók meg mentesülnek a modell teljes körű megismerése alól. Mindez a párhuzamos munkát is segíti.

Fontos szem előtt tartani, hogy mindezen technikák alkalmazásához a **CASE** eszköznek támogatnia kell a generátor saját igényünkre való alakítását. A minta alapú **kódgenerálás** hatékonyságának előnyei pedig csak akkor érvényesülnek, ha a generálás nem szkript- hanem template (minta) alapú.

A **decompiler**-ek (vagy **kódvisszafejtő programok**) olyan programok, melyek a fordítóprogramokkal ellentétes műveletet hajtanak végre. Azaz az alacsony absztakciós szintű (gépi kódú) futtatható programokat fejtik vissza (amelyek számítógépek által értelmezhető formában vannak) magasabb absztrakciós szintű kóddá, (amelyet emberek által olvasható formában jelenítenek meg).

CASE: Computer Aided Software Engineering.
Azokat szoftvereket nevezzük így, amelyek támogatják a rendszerfejlesztés teljes folyamatát a problémadefiniálástól a kivitelezésen át a bevezetésig, üzembe helyezésig és elkészítik a rendszer-dokumentációt.
A CASE lehetőségek olyan eszközök, amelyek célja a rendszerfejlesztési tevékenység elemzési és tervezési fázisaiban végzett munka hatékonyságának növelése:
- a rendszer komponenseinek és működésének feltárásával
- a fejlesztési információk rögzítésével és elemzésével
- a projekt irányításával, valamint
- a fejlesztéshez kapcsolódó információk és teendők adminisztrálásával.

Kialakulásuk a 80-as évekre tehető, az alkalmazások bonyolultsága, az ember-gép kommunikáció tervezésének előtérbe kerülése miatt a tervezői dokumentációk elkészítése, a projektben résztvevők munkájának összehangolása nagy adminisztrációs feladatot jelentett, és magától értődően ezen feladatokra is számítógépes támogatást kezdtek használni: pl. folyamatábrák készítése valamilyen grafikus editorral, vagy képernyőtervek készítése valamilyen fejlett adatbázis-kezelővel, vagy 4GL eszköz használata programfejlesztéshez.
A 4GL (negyedik generációs fejlesztő) eszközök fő funkciói: programtervezés támogatása, kódgenerálás, output-generálás.
Ezeket a lehetőségeket azután egyetlen programrendszerbe ötvözték, ilyen módon a részek közötti összefüggések is leírhatóvá és ellenőrizhetővé váltak.
Egy ilyen programcsomag szerkezetében is őrzi ezeket a részfeladatokat, így a következő három nagy modulra bontható (strukturált módszertan támogatása esetén)
- ábraszerkesztő modul
- adatszótár-kezelő modul
- alkalmazás generátor.

Az ábraszerkesztő modul önállóan is használható, és a programcsomag által támogatott módszertan előírásainak megfelelő ábrák megszerkesztésére alkalmas.
Általában készül hierarchikus rendszerábra, egyed-kapcsolati adatmodell, struktúra ábrák a programszerkezetekhez, és felhasználói interfész-tervek.
Az adatszótár olyan adatbázis, amely a rendszerfejlesztési munka adatait és azok összefüggéseit tartalmazza. Több eszközben Repository-nak nevezik.
Az adatszótár táblái, az ábrák, ábraelemek, felhasználói követelmények, stb. adatait tartalmazza.

Az ábraszerkesztő outputjait, vagyis a kész ábrák adatait- neveket, kódokat, kapcsolatokat - automatikusan tárolja, és lehetőséget biztosít az ábrák és ábraelemek leírásainak kiegészítésére. Például E-K ábra egyedtípusaihoz attribútumok megadása, egyed-előfordulások számának becslése, stb.
Továbbá olyan jegyzékeket, táblázatokat készíthetünk, amelyet ábraszerkesztővel nem tudtunk elkészíteni, de a rendszerfejlesztés dokumentációjának tartalmaznia kell, például funkció-meghatározás, egyed-esemény mátrix, stb.
Elemzési feladatokhoz az adatszótár-kezelővel keresztreferencia táblázatokat készíthetünk: egyed/attribútum, attribútum/output, stb.
Az adatszótár kezelő támogatja a relációs adatelemzést, ellenőrzi az ábrák kapcsolatait és teljességét.
Az adatszótár adattartalmából elkészíthető a rendszerdokumentáció, automatikusan elkészíthető az adatbázis definíció egy választott adatbázis-kezelő nyelvhez.
Alkalmazás generátor, olyan 4GL fejlesztőeszköz, amely az adatszótár adattartalmára alapozva támogatja a kivitelezési munkát: programterveket, képernyőterveket szerkeszthetünk a segítségével, eredményképpen program dokumentációt készít és generálja a tervezett rendszer programjait.
Ez a modul képes a feladata megfordítására is, azaz működő program szerkezetét képes megfejteni, és arról dokumentációt készíteni. Ez a **reverse engineering**, vagy másképpen kód vizualizáció.

reverse engineering és kódgenerálás funkciói révén UML modelleket és forráskódot állítanak elő, valamint a kész modellből logikai és fizikai adatmodellt generálnak.