# A tesztelés szükségessége

- Veszélyek megjelenése
  - Hardver téren az integráltsági fok növekedése
  - Szoftver téren a milliós nagyságrendben lévő utasításszám
  - A rendszerek teljesítményének növekedésével együtt járt a bonyolultság lényeges növekedése is.
  - Megnőtt az emberi eredetű tervezési, javítási és üzemeltetési hibák előfordulásának valószínűsége.

## TESZTELÉSI STRATÉGIÁK

### Hardver és szoftver a biztonság szempontjából

- A két szféra közül a szoftver az, amely a legtöbb gondot, problémát okozza.
- Hardver
  - több technológiai tapasztalat
  - szabványos, kipróbált építőelemek
  - az elterjedtségből adódóan az esetleges tervezési hibák könnyebben napfényre kerülnek
- Szoftver
  - egyedi, speciális fejlesztések

**TESZTELÉSI STRATÉGIÁK**

# Szoftverfejlesztési modellek

- Több fejlesztési fázis
- Életciklus modellek
  - vízesés modell
  - spirál modell
  - V-modell
- A különböző modellek a fejlesztési projektek egyes fontosabb aspektusainak hangsúlyozásában térnek el egymástól.
- Mindegyikük egy új minőségbiztosítási-fejlesztési paradigma sajátságait hordozza magában.

**TESZTELÉSI STRATÉGIÁK**

# A vízesés modell főbb fázisai

- Rendszerkövetelmények megadása.
- Szoftver-követelmények megadása.
- Előzetes programtervek elkészítése, elemzése.
- Programtervezés.
- Programkódolás.
- Tesztelés.
- Működtetés.

**TESZTELÉSI STRATÉGIÁK**

# A spirál modell főbb fázisai

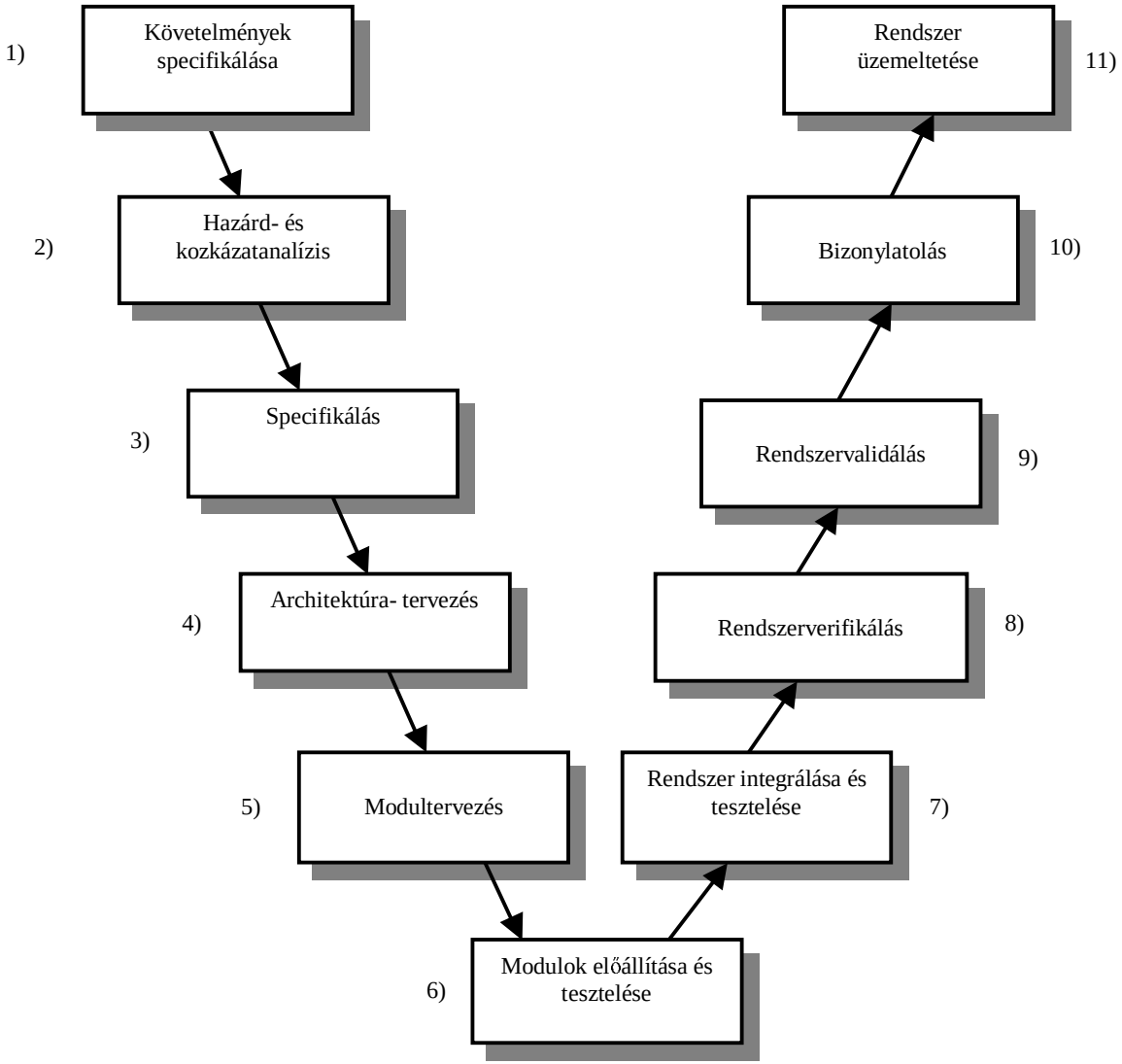
- A működési koncepció kialakítása.
- Követelmények tervezése.
- Életciklus tervezése.
- Kockázat-analízis.
- Szoftver-követelmények meghatározása.
- Szoftver-termék tervezése.
- A tervezés verifikálása és validálása.
- Integrálás és tesztelés tervezése.
- Részletes tervezés.
- Kódolás.
- Egységek (modulok) tesztelése.
- Összeintegrálás és tesztelés.
- Elfogadási tesztelés.
- Üzembe helyezés.

# A V-modell

- A biztonságkritikus számítógéprendszerek esetében terjedt el.
- A modell jól kifejezi a tervezési folyamat fentről lefelé történő haladását (top-down megközelítés), míg a tesztelési folyamat lentről felfelé halad (bottom-up megközelítés).
- A gyakorlatban a különböző fázisok nem szigorúan a megadott sorrendben hajtódnak végre.

## TESZTELÉSI STRATÉGIÁK

# A V-modell



**TESZTELÉSI STRATÉGIÁK**

# V-modell életciklus állomások

- Követelmények specifikálása
  - Funkcionális követelmények dokumentációja
- Hazárdok és kockázatok elemzése
- A teljes rendszer-specifikáció
  - A funkcionális követelmények valamint a biztonsági követelmények együttese alkotja.
- Architekturális tervezés:
  - A teljes informatikai rendszer architektúrája hardverből és szoftverből tevődik össze.
  - HW-SW co-design
- A szoftver modulokra bontása

**TESZTELÉSI STRATÉGIÁK**

# V-modell életciklus állomások

- A modulok elkészítése és tesztelése
  - A tesztelési folyamatok előzetesen megtervezendők
- A szoftver-modulok összeintegrálása
  - Inkrementális tesztelés
  - „big bang” tesztelés
- A teljes rendszer verifikálása
  - rendszer megfelel-e a specifikációjának

**TESZTELÉSI STRATÉGIÁK**

# V-modell életciklus állomások

- A teljes rendszer validálása
  - rendszer megfelel-e a felhasználói követelményeknek
  - biztonságigazolás
- Bizonylatolás (certification)
  - a hatósági előírások és szabványok szerinti megfelelés eldöntése
- Üzembe helyezés, üzemeltetés, karbantartás, elavulás, üzemeltetés megszüntetése

# Hazárdok és kockázatok elemzése

- Hazárd: Olyan helyzet, amely egy valóságos vagy lehetséges veszélyt jelent az emberek vagy a környezet számára.
- Kockázat: Egy hazárdhoz kapcsolódó események és azok következményei, valószínűségi alapon. Azt fejezi ki, hogy milyen valószínűségű veszélyes következményei lehetnek egy hazárdnak.

# Hazárdanalízis

- Hibamódok és hatásuk analízise
  - Ebben az eljárásban komponens-hibákat tételezünk fel külön-külön, és azt vizsgáljuk, hogy a hibáknak milyen hatásuk lehet a rendszer biztonságára.
- Eseményfa-analízis (event-tree analysis)
  - Események hatásának elemzése a végső kimenetelekig különböző szinteken.
- Hibafa-analízis (fault-tree analysis)
  - Vészhelyzetek visszavezetése a kiindulási okokig boole-gráf segítségével.

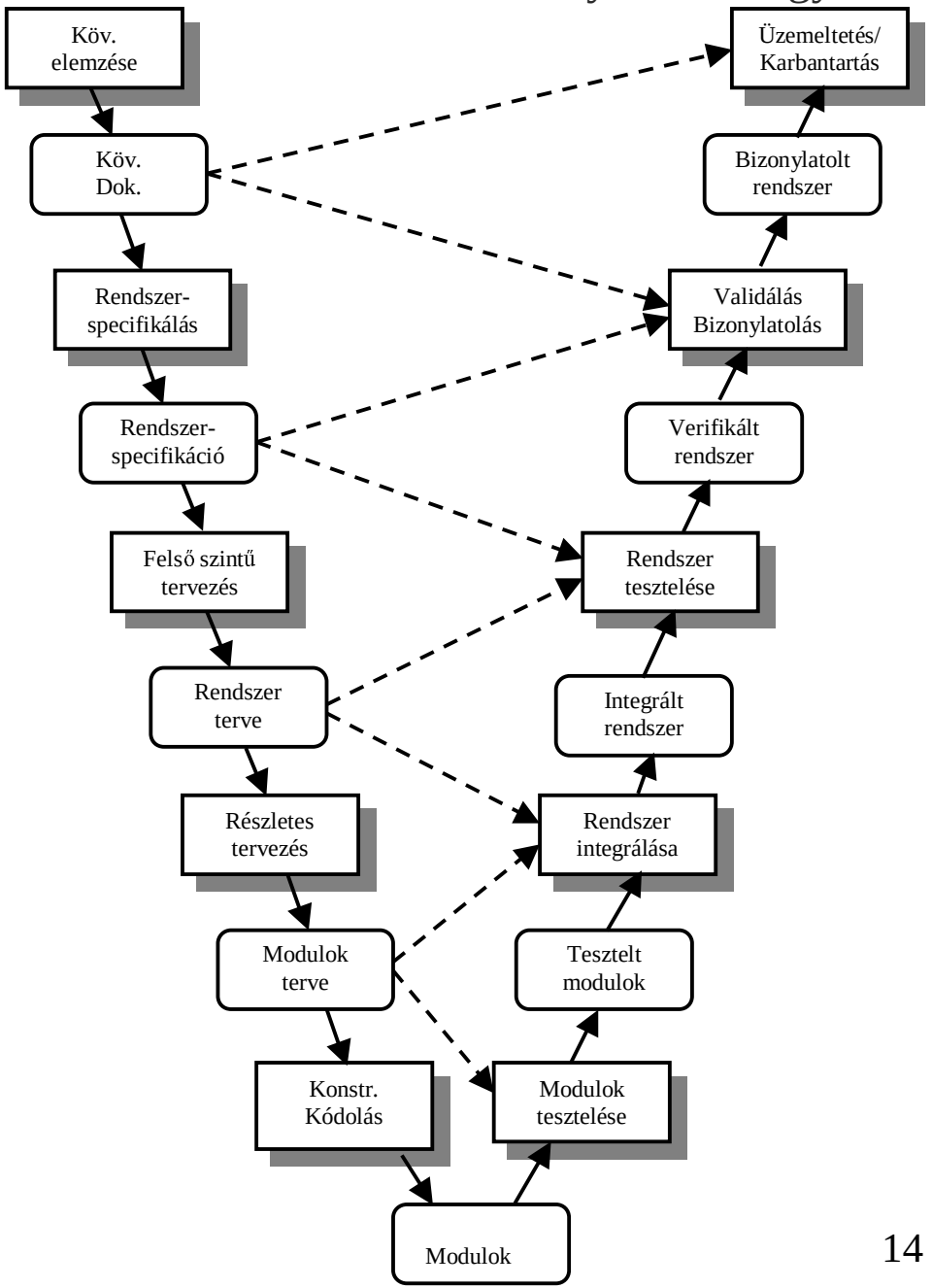
**TESZTELÉSI STRATÉGIÁK**

# A szoftver-modulok összeintegrálása

- Inkrementális tesztelés
  - Integrálás egyenkénti bővítéssel és teszteléssel
  - Minden egyes bővítés után külön teszteljük az addig összeállt komplexumot
- „Big bang” tesztelés
  - Az összes modult egyszerre rakjuk össze, és a teljes komplexumra hajtjuk végre a tesztelést

## TESZTELÉSI STRATÉGIÁK

# V-modell tevékenységek



**TESZTELÉSI STRATÉGIÁK**

# Verifikáció és validáció

- Verifikáció:
  - Az a folyamat, amelyben meghatározzuk, hogy a vizsgált informatikai rendszer szoftvere teljesíti-e mindazokat a követelményeket, amelyeket egy előző fázisban specifikáltak a szoftver fejlesztési vagy előállítási folyamatában
- Validáció:
  - A teljes szoftver vizsgálata és kiértékelése azzal a céllal, hogy meghatározzuk, minden szempontból megfelel-e a felhasználói követelményeknek.
- Egy biztonságkritikus rendszernél a következőket kell igazolni:
  - A funkcionális megfelelőséget.
  - A teljesítményben való megfelelőséget.
  - A biztonsági követelmények teljesülését.

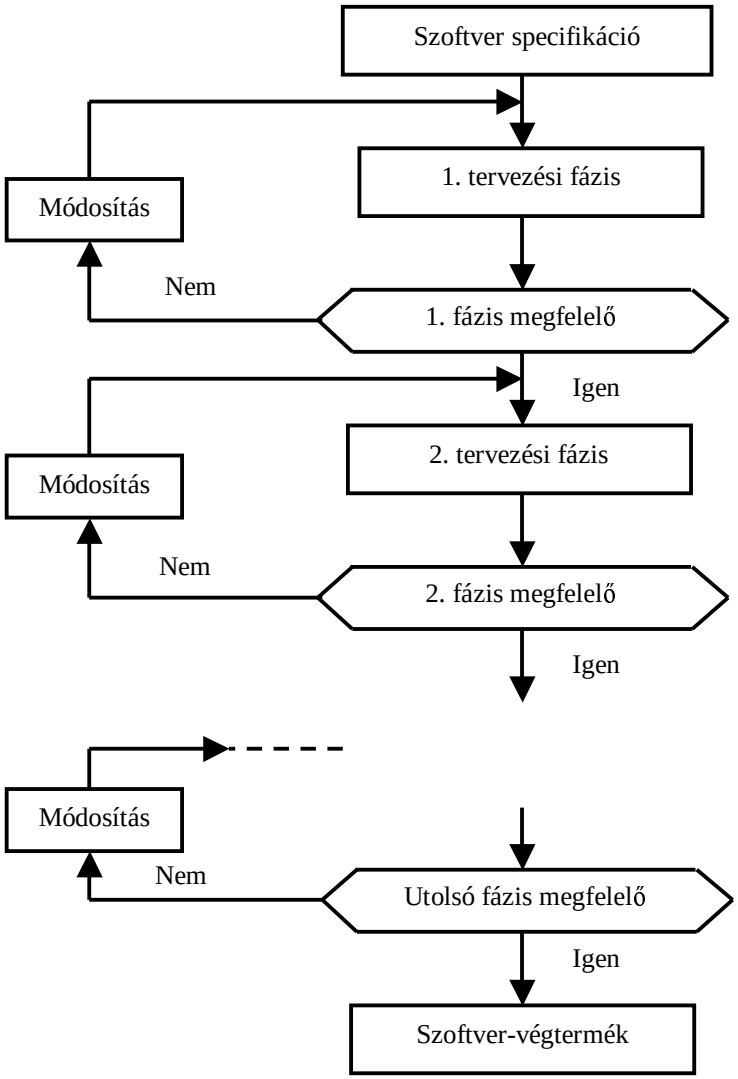
**TESZTELÉSI STRATÉGIÁK**

# A fejlesztési folyamat vizsgálata

- A fejlesztés egyes stádiumaihoz egymástól eltérő megvalósítási reprezentációk tartoznak.
- A helyes végrehajtás deklarálása azt kívánja meg, hogy bizonyítsuk ezen reprezentációk közötti egyértelmű összhang teljesülését.
- A bizonyítási folyamat elvégzése úgy logikus, hogy az egymást követő fejlesztési fázisok közötti összhang, ekvivalencia igazolását végezzük el lépésenként.

**TESZTELÉSI STRATÉGIÁK**

# A fejlesztési fázisok kapcsolata



**TESZTELÉSI STRATÉGIÁK**

# ”V & V” eljárás

- A verifikáció és a validáció a fejlesztési folyamat két összefüggő velejárója, a két eljárás mindig együtt kezelendő.
- A verifikáció végső soron arra ad választ, hogy **a termék előállítása helyesen történt-e**.
- A validáció arra ad választ, hogy **a helyes terméket állítottuk-e elő.**

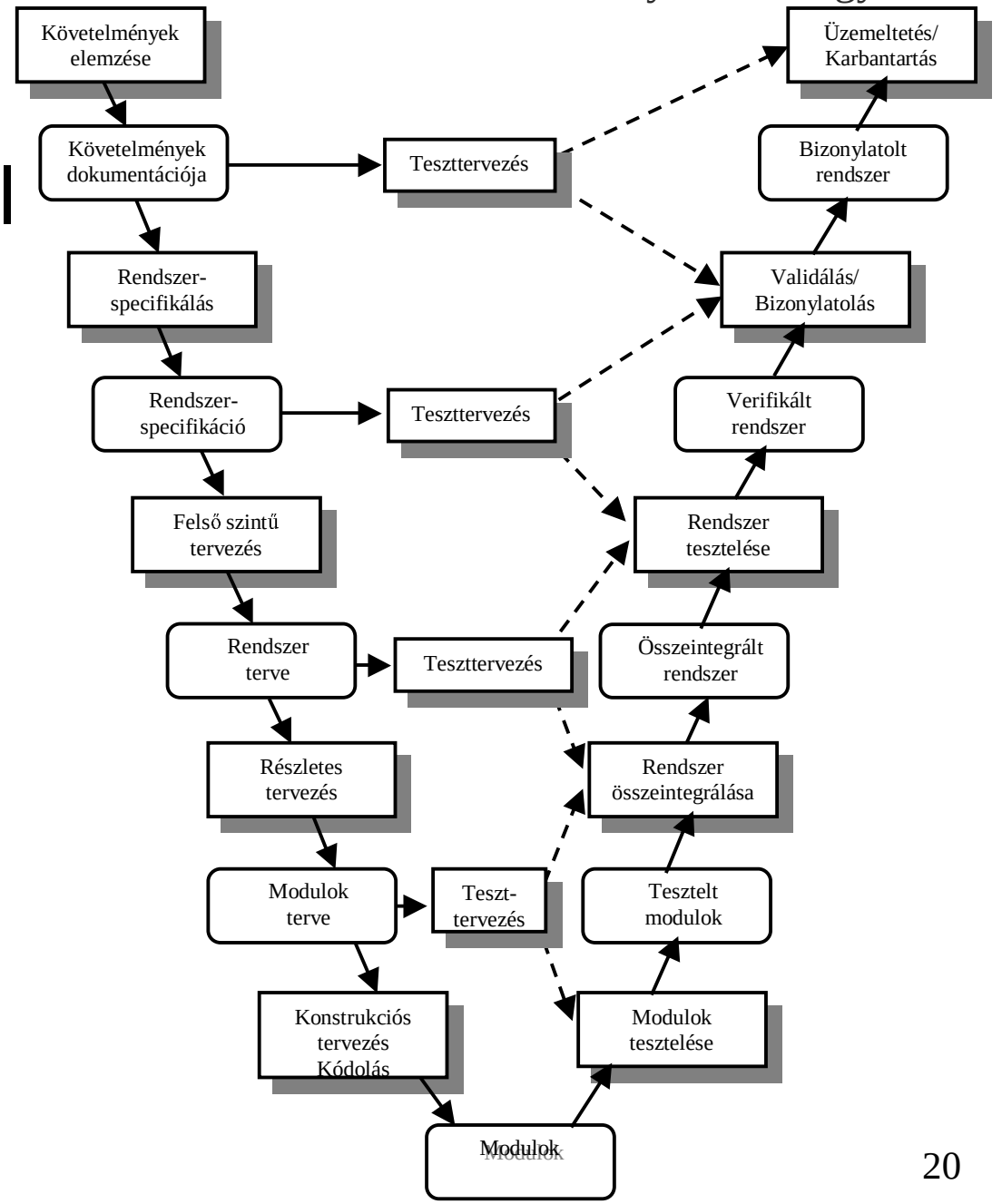
**TESZTELÉSI STRATÉGIÁK**

## A verifikáció és a validáció fázisainak tervezése

- Az előre való tervezés lényeges mind a költségek becslése, mind pedig minimalizálása szempontjából.
- A verifikálás és a validálás egyaránt a tesztelésen alapul, a tesztek megtervezése igen fontos része lesz a fejlesztési folyamatnak.
- A validációs terv befolyásolni tudja magának a projektnek a további menetét.
- A validációs szempontoknak a figyelembe vétele a tervezés egy korai szakaszában jelentős megtakarításhoz vezethet a későbbi szakaszok ráfordításában.

## TESZTELÉSI STRATÉGIÁK

# V-modell teszttervezéssel



**TESZTELÉSI STRATÉGIÁK**

# IEC 1508 követelménylista

- Annak leírása, hogy a validálás mikor hajtandó végre.
- Annak megadása, hogy ki hajtja végre a validálást.
- A rendszerműködés figyelembe veendő módjainak megadása, az alábbiak szerint:
  - A használat előkészítése
  - Indítás
  - Automatikus üzemeltetés
  - Manuális üzemeltetés
  - Félautomatikus üzemeltetés
  - Állandósult állapotbeli működés
  - Újraállítás (resetelés)
  - Teljes leállítás
  - Karbantartás
  - Előrelátható abnormális feltételek kezelése

**TESZTELÉSI STRATÉGIÁK**

# IEC 1508 követelménylista

- A biztonságra kiható rendszerek megadása, és azon külső kockázatcsökkentő eszközök megadása, amelyeket validálni kell mindegyik működési módban.
- A validálás technikai stratégiája.
- Intézkedések, technikák, eljárások, amelyek arra használandók, hogy mindegyik biztonsági funkcióról el legyen döntve, hogy az megfelel-e az általános biztonsági követelményeknek.
- Hivatkozások az általános biztonsági követelmények dokumentumaira.

**TESZTELÉSI STRATÉGIÁK**

# IEC 1508 követelménylista

- A szükséges környezet, amelyben a validálás tevékenységet el kell végezni.
- Az elfogadási/elutasítási kritériumok.
- Irányelvek és szabályok a validációs eredmények kiértékelésére vonatkozóan.

# Egységek, modulok tesztelése

- A moduláris fejlesztés lehetővé teszi a moduláris vagy egységenkénti tesztelést.
- Egységtesztelés
  - jórészt white-box jellegű strukturális tesztelés
  - tesztesetek kiegészítése a specifikáció tesztelésére szolgáló black-boksz tesztesetekkel
  - az egység- vagy modultesztek jelentik a szoftverfejlesztés közbeni legalacsonyabb szintű tesztelést

**TESZTELÉSI STRATÉGIÁK**

# Egységek, modulok tesztelése

- A tesztesetek nem csak egyszer használatosak a fejlesztés során a hibamentes kódolás ellenőrzésére, hanem mindannyiszor végre kell hajtani őket, ahányszor a szoftveren változtatást végzünk vagy a modult egy új, másik környezetben használjuk fel (regressziós tesztelés).
- A modultesztek során teszteljük az adott modul és a modult a rendszerhez csatoló interfész funkcionális működését.

# Egységek, modulok tesztelése

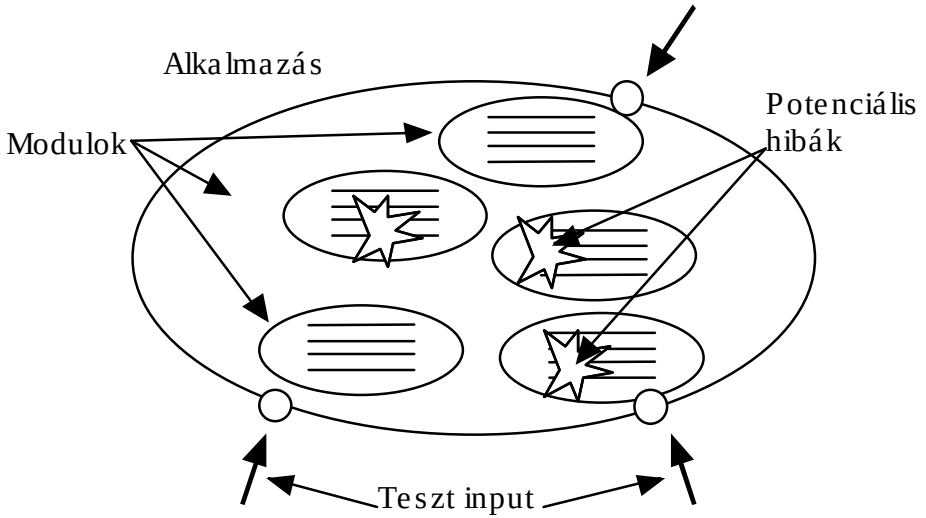
- A modul tesztelése egy tesztkörnyezetben történik
- A modul a tesztkörnyezettel kombinálva alkot egy működő rendszert.
  - modulok egymástól teljesen függetlenül kerülnek tesztelésre (izolációs tesztelés)
  - a tesztelendő modul már korábban tesztelt modulokkal kombinálva kerül tesztelésre (inkrementális tesztelés)
- A tesztkörnyezetek egy tesztvégrehajtó egységből és egy kiegészítő részből állnak.

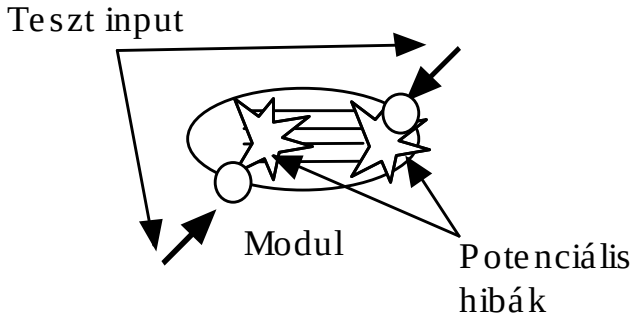
# A modulltesztelés szükségessége

- Az egységtesztek elvégzése a hibák nagy részét feltárja
- Az integrációs, vagyis az alkalmazást egészében tekintő tesztek nem alkalmasak a hibák jórészének felderítésére.
  - A nagyobb kód integrációk sokkal komplexebbek.

**TESZTELÉSI STRATÉGIÁK**

# A modulltesztelés szükségessége



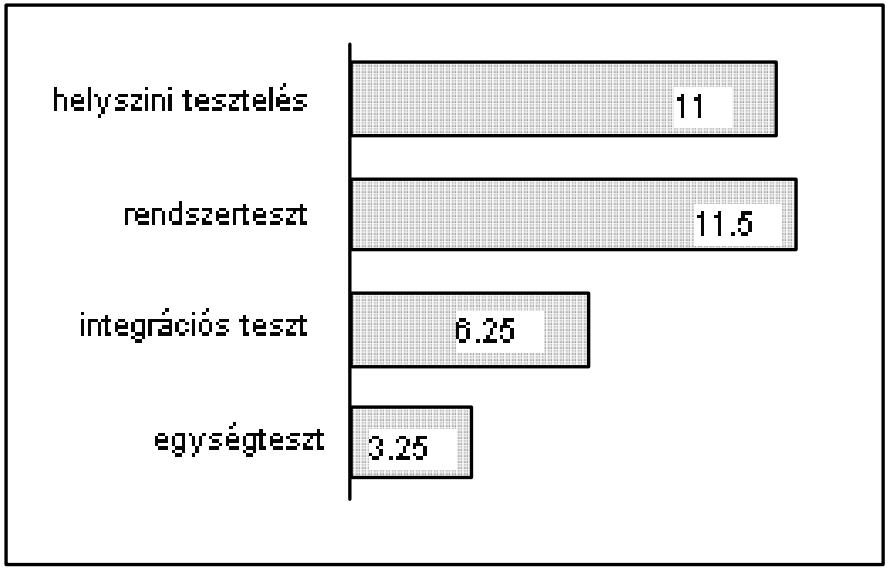

Az egységszintű hibák felderítése integrációs tesztelés közben nehéz

Az egységtesztelés jobb lefedettséget eredményez

**TESZTELÉSI STRATÉGIÁK**

# A modulltesztelés szükségessége

- Minél később kerül felderítésre egy hiba, annál magasabb a javítási költsége.
- Az egységtesztek létrehozása, karbantartása sokkal könnyebb és kényelmesebb, mint a későbbi fázisokban végrehajtott teszteké.

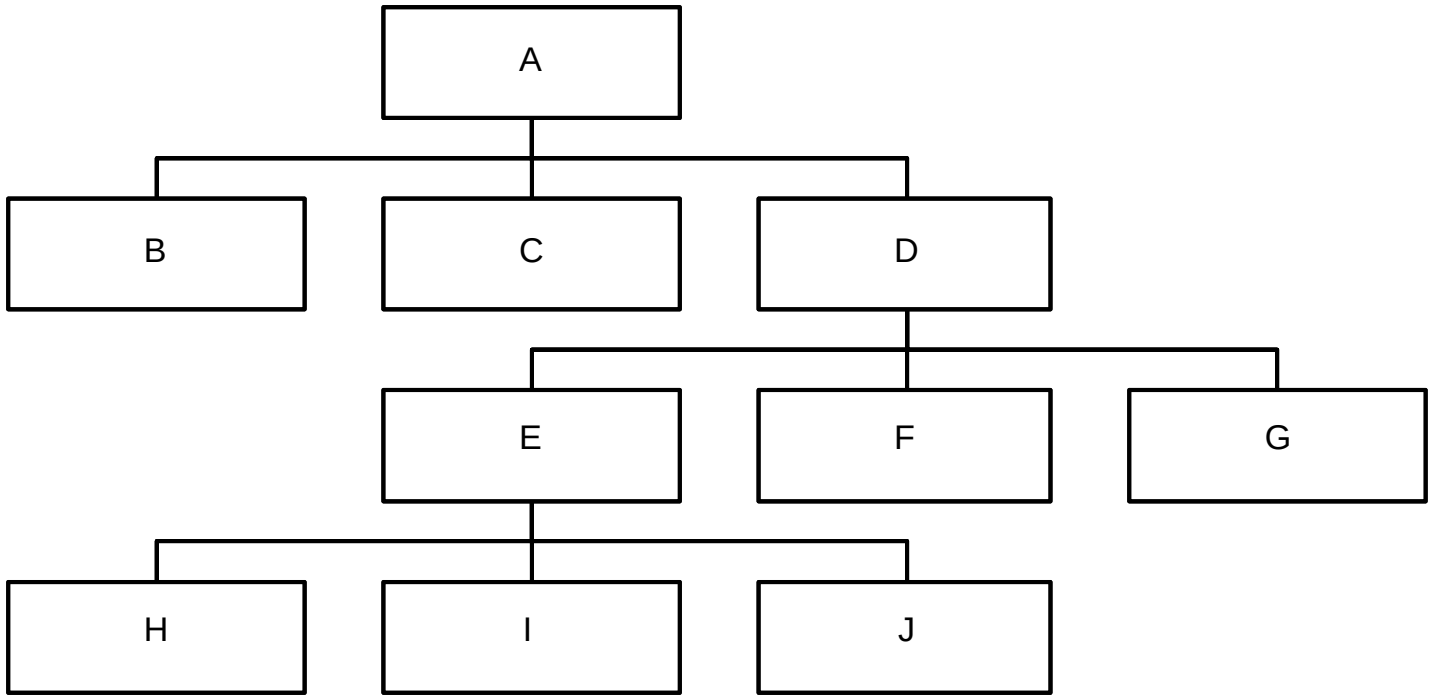



Különböző típusú tesztek egymáshoz viszonyított relatív időigénye

# Az egységtesztelés szervezése

- Három stratégia
  - Izolációs tesztelés
  - Top-down tesztelés
  - Bottom-up tesztelés

# Példa modul hierarchia



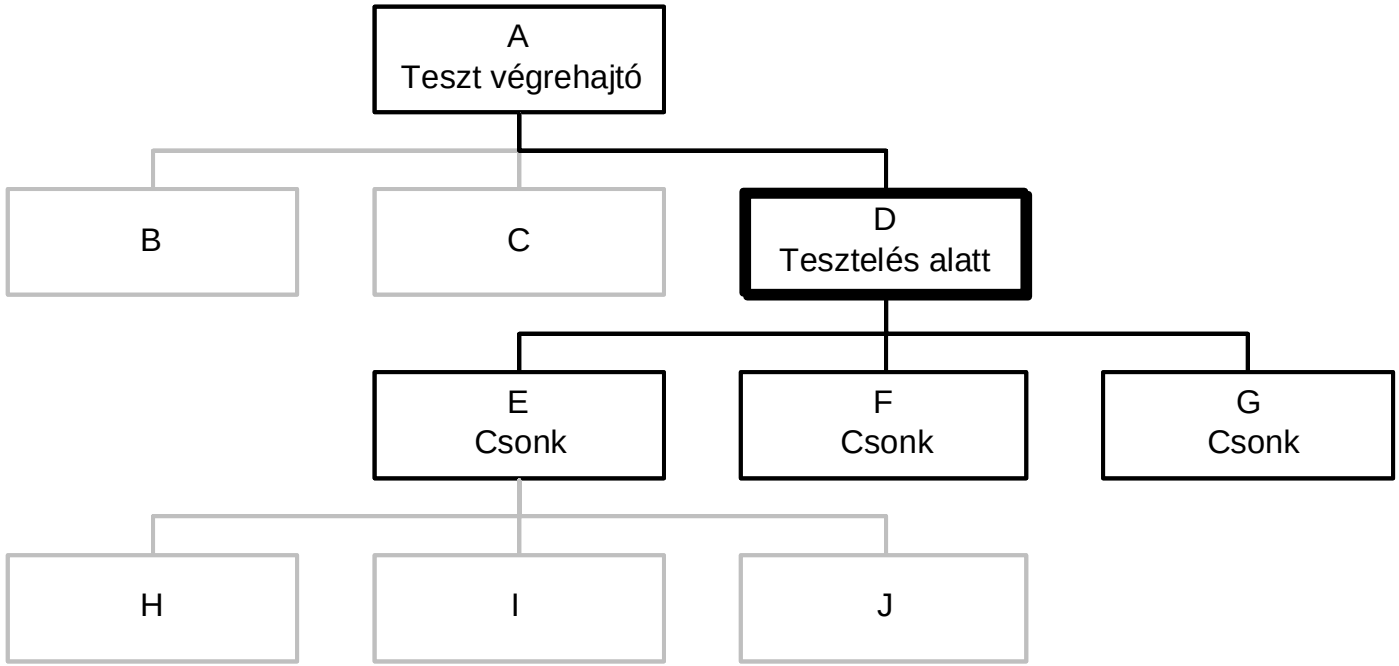
**TESZTELÉSI STRATÉGIÁK**

# Izolációs tesztelés

- Az izolációs tesztelés során a tesztelés alatt álló modul az általa hívott és az őt hívó moduloktól elszigetelten kerül tesztelésre.
- Az egységek tetszőleges sorrendben tesztelhetők, mivel egyik egység tesztelése sem kívánja meg más modulok teszteltségét.
- Minden egyes modulhoz szükség van egy tesztvégrehajtó komponensre és az összes általa hívott egységeket helyettesítő csonkokra.

**TESZTELÉSI STRATÉGIÁK**

# Izolációs modul tesztelés



**TESZTELÉSI STRATÉGIÁK**

# Az izolációs tesztelés előnyei

- Az izolációs tesztelés biztosítja a legkönnyebb utat a megfelelő strukturális tesztlefedettség eléréséhez.
  - A lefedettség elérési nehézsége nem függ az adott egység hierarchiabeli helyétől
- A tesztvégrehajtó egységek egyszerűbbek.
- Tetszőleges számú egység tesztelhető párhuzamosan
- Az egy egységen végrehajtott módosítások csak az adott egység tesztjének a változtatását vonják maguk

TESZTELÉSI STRATÉGIÁK

# Az izolációs tesztelés hátrányai

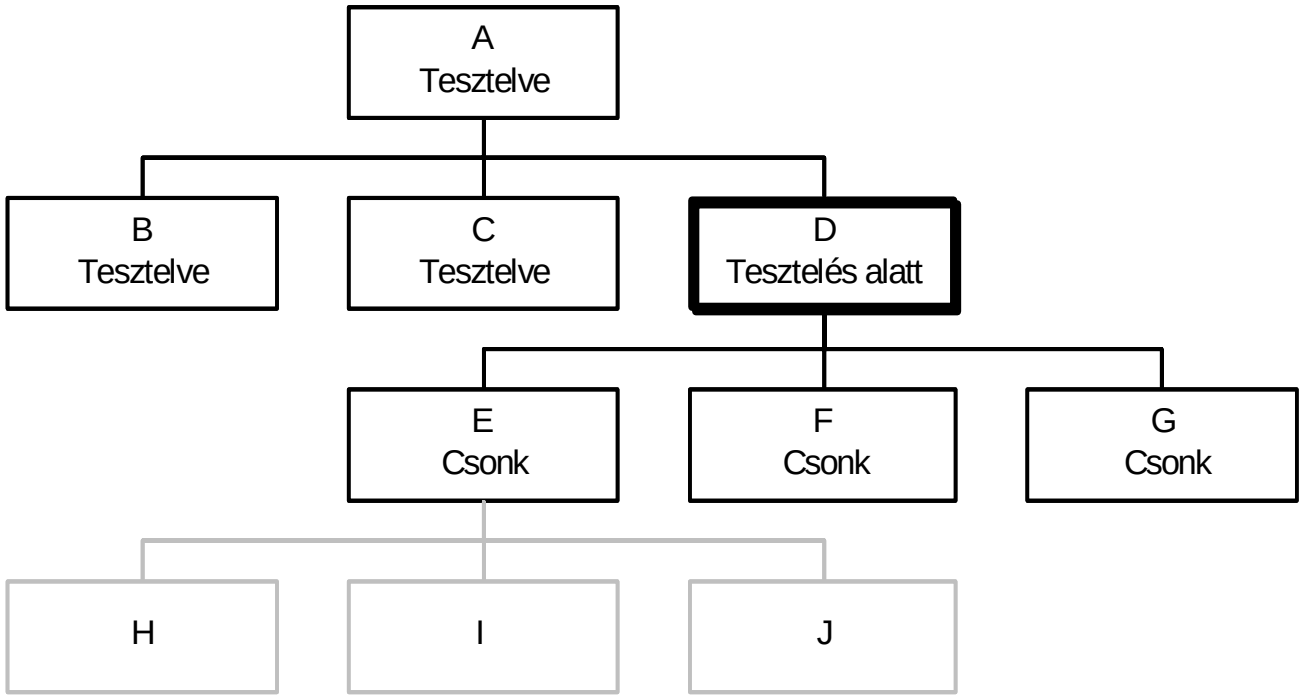
- Nem biztosítja az egységek korai integrációját.
- Az izolációs tesztelési megközelítés megköveteli strukturális tervezési információk rendelkezésre állását és tesztvégrehajtó modulok és csonkok írását.
  - ezt az egységhierarchia minden szintjén el kell végezni

# Top-down tesztelés

- Top-down egységteszteléskor az egyes modulok az őket hívó modulokból kerülnek tesztelésre
- Először a hierarchia csúcsán álló modul kerül tesztelésre, az összes hívott modult csonkok helyettesítik.
- Ez a folyamat addig ismétlődik, míg a legalsó szintű egységek is tesztelésre kerülnek.

**TESZTELÉSI STRATÉGIÁK**

# Top-down modul tesztelés



**TESZTELÉSI STRATÉGIÁK**

# A top-down tesztelés előnyei

- A top-down egységtesztelés az egységek korai, szoftver integrációs fázis előtti integrációját biztosítja.
- Mivel a modulok részletes tervezése top-down jellegű és a top-down egységtesztelés a teszteket így a modulok tervezési sorrendjében hajtja végre
- Top-down egységteszteléssel azonosítható az alsóbb szintű egységekben megvalósított redundáns funkcionalitás, mert ez nem fedhető le tesztekkel.

**TESZTELÉSI STRATÉGIÁK**

# A top-down tesztelés hátrányai

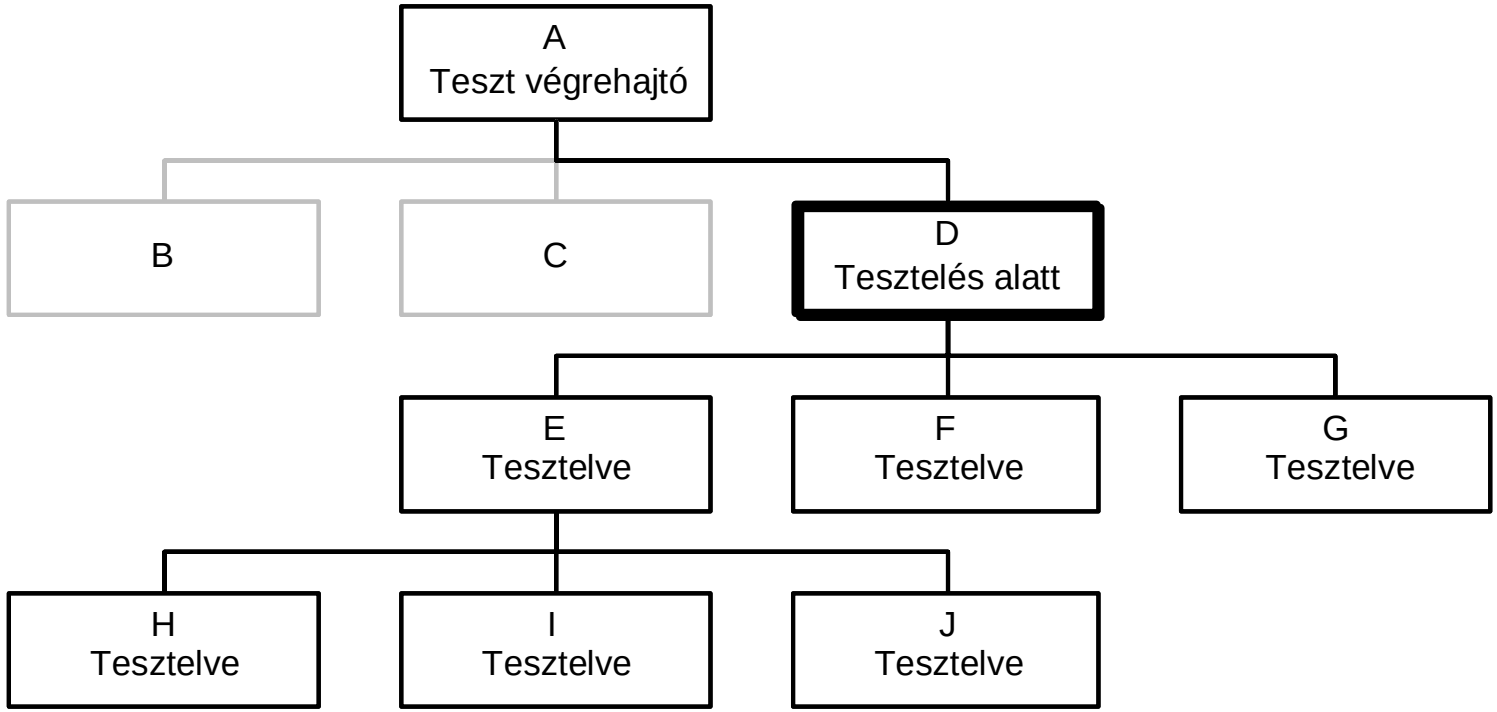
- A top-down tesztelést a csonkokkal lehet irányítani, a tesztesetek több csonkot is használhatnak egyszerre.
- Minden egyes tesztelt egységgel a tesztelés egyre komplikáltabb lesz.
- Egy egységen végrehajtott módosítás gyakran befolyásolja a hierarchiában azonos vagy az alacsonyabb szinten lévő egységek tesztelését.
- Az egységek tesztelésének sorrendjét az egységhierarchia határozza meg

# Bottom-up tesztelés

- Bottom-up egységteszteléskor a modulok az őket hívó moduloktól elszigetelten kerülnek tesztelésre, de a valódi hívott modulokat használják saját hívásaikhoz.
- Először a legalsó szinten lévő modulokat kell tesztelni, majd ezeket kell használni a felsőbb szintű egységek tesztjeinél, vagyis már tesztelt hívott egységekkel kerül sor a tesztelésre.

**TESZTELÉSI STRATÉGIÁK**

# Bottom-up modul tesztelés



**TESZTELÉSI STRATÉGIÁK**

# A bottom-up tesztelés előnyei

- Biztosítja az egységek korai, szoftverintegrációs fázis előtti integrációját
- A teszteseteket teljes egészében a tesztvégrehajtó egységek határozzák meg, nincs szükség csonkokra.
- az egységhierarchia alján álló modulok tesztelését relatív egyszerűvé teszi.
- Ez a tesztelési stratégia jól használható objektumok tesztelésére.

**TESZTELÉSI STRATÉGIÁK**

# A bottom-up tesztelés hátrányai

- Ahogy a tesztelés felfelé halad az egységhierarchiában úgy válik a bottom-up egységtesztelés mind bonyolultabbá és ezért egyre nehezebben kifejleszthetővé és karbantarthatóvá.
- Egy egység megváltoztatása gyakran hatással van a hierarchiában felette lévő egységekre.

# Egységtesztelési stratégiák összehasonlítása

- Az izolációs egységtesztelés a legjobb választás egységtesztelésre.
- A top-down stratégia nem jó választás egységtesztelésre, hanem inkább a már tesztelt egységek integrálásának megfelelő eszköze.
- A bottom-up megközelítés jó egységtesztelési stratégia lehet különösen, ha objektumokat és egység újra felhasználást tekintünk.
- Azonban a bottom-up megközelítés inkább a funkcionális, mint a strukturális tesztelés irányába mutat.

**TESZTELÉSI STRATÉGIÁK**

# Integrációs- és rendszertesztelés

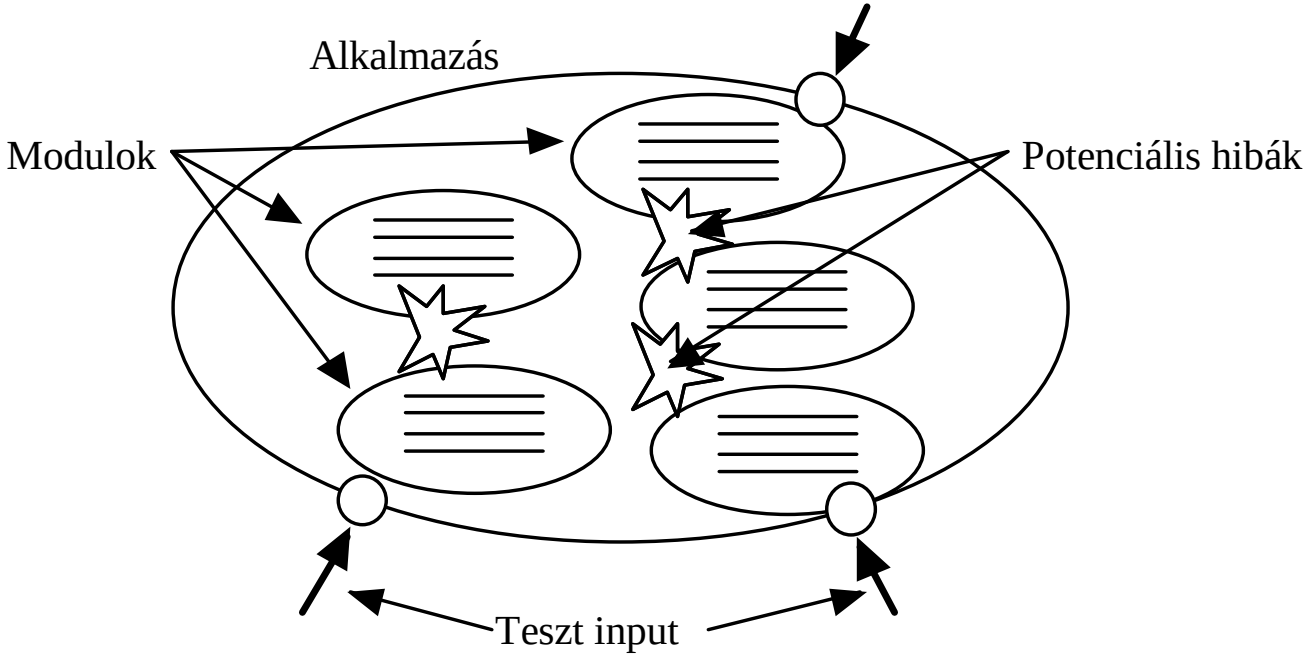
- A modul szintű tesztelést követően magasabb szinteken kell a tesztelést tovább folytatni.
- Legalacsonyabb szinten idetartoznak a modulok rendszerbeli együttműködését (integrációját) ellenőrző integrációs tesztek.
- A következő, ún. funkcióteszt szinten már tisztán black-box jellegű teszteket végzünk.
- A következő szint a rendszertesztek szintje.

**TESZTELÉSI STRATÉGIÁK**

# Integrációs tesztelés

- Az integrációs tesztek a szoftver életciklus integrációs fázisában kerülnek végrehajtásra.
- A kooperáló modulokból felépített rendszerek állapotát az egyes modulok közötti interakciók is meghatározzák. Elképzelhető, hogy egy rendszer annak ellenére hibásan működik, hogy az összes alkotó komponense egyenként korrekt.
- Az egész rendszer szekvenciális működését moduljainak a kapcsolata is befolyásolja.

# Integrációs tesztelés



# Rendszertesztelés

- Ez a tesztelési fázis a megfogalmazott céloknak a követelmények meghatározásakor specifikációvá történő alakításánál elkövetett hibák feltárását szolgálja.
- A rendszerteszteket lehetőleg a fejlesztőktől független és a végfelhasználók helyzetét ismerő szervezetnek kell végeznie.

**TESZTELÉSI STRATÉGIÁK**

# Rendszertesztelés

- Szolgáltatás tesztelés
  - A rendszer nyújtja-e azokat a szolgáltatásokat melyeket céljai között megjelöltek.
  - Nem a szoftverspecifikációra épül.
- Mennyiségi tesztelés
  - A szoftver működését nagy mennyiségű adattal teszteljük a kapacitáskorlátok ellenőrzésére.
  - A mennyiségi tesztelés viszonylag erőforrás igényes.

**TESZTELÉSI STRATÉGIÁK**

# Rendszertesztelés

- Löket terheléses tesztelés (Stressz-tesztelés)
  - Nagy mennyiségű adat rövid idő alatti feldolgozásáról van szó.
  - A rendszer valós helyzetekbeli robosztusságának ellenőrzésére végeznek olyan stressztesztеket is, melyek a valóságban soha elő nem forduló feltételeket jelentenek.
- Használhatósági tesztelés
  - A használat közbeni minőség azt jelenti, hogy egy termék egy meghatározott felhasználó által, egy meghatározott felhasználási körben használva, meghatározott célok hatékony és produktív elérésére, mennyire kielégítő és mennyire vezet megelégedésre.
  - A tesztelés során a felhasználó teljesítményére vonatkozó kvantitatív (idők, pontosság, elvégzett feladatok száma) és a megelégedettségre, véleményre vonatkozó kvalitatív (kérdőívek, megjegyzések) adatokat gyűjtenek.

**TESZTELÉSI STRATÉGIÁK**

# Rendszertesztelés

- Biztonsági tesztelés
  - A biztonsági tesztelés célja, hogy felfedje az adatbiztonsággal és adatvédelemmel kapcsolatos hibákat.
  - Tesztesetek létrehozásához előnyős lehet a hasonló rendszerek már felderített biztonsági hiányosságainak ismerete.
- Teljesítménytesztelés
  - A programok egy részénél a teljesítmény vagy a hatékonyság specifikálva van különböző terheléseknél és konfigurációkra meghatározott válaszidők és feldolgozási sebességek formájában.
  - A tesztesetek célja annak a demonstrálása, hogy a rendszer nem képes a megadott teljesítménycélok kielégítésére.

**TESZTELÉSI STRATÉGIÁK**

# Rendszertesztelés

- Konfigurációtesztelés
  - Bizonyos programoknak eltérő operációs környezetekben is működniük kell. Ezek a különböző környezetek jelenthetnek eltérő hardver konfigurációkat (adott esetben teljesen különböző architektúrákat), különböző operációs rendszereket ill. különböző szoftver installációkat.
  - A konfigurációtesztelés sok esetben különböző konfigurációkban elvégzendő teljesítménytesztelést jelent.
  - Ellenőrizni kell a kompatibilitási célok elérését vagy a konverziós eljárásokat

**TESZTELÉSI STRATÉGIÁK**

# Rendszertesztelés

- Megbízhatósági tesztelés
  - Általában igen nehéz a teszteléshez rendelkezésre álló idő alatt a rendkívül alacsony meghibásodási ráta követelményeknek megfelelő rendszerek tesztelése.
  - Bizonyos megbízhatósági paraméterek érvényességére matematikai modellek alapján adhatunk becsléseket.
- Dokumentációtesztelés
  - A dokumentáció ellenőrzése pontosság és világos érthetőség szempontjából.

# Validációs tesztelés

- A validációs folyamatokban egyrészt a rendszer funkcionális téren elvárt működésének ellenőrzésére van szükség, másrészt pedig az ezen túlmenő, további felhasználói elvárások teljesülését is igazolni kell.
- Szükség van arra, hogy a rendszer olyan tulajdonságait, működési módjait is megvizsgáljuk, amelyeket a tervezési specifikáció nem definiált.

**TESZTELÉSI STRATÉGIÁK**

# Valósidejű rendszerek időzítési szempontjai

- Az igazolási feladat megoldása két részből áll:
  - A rendszerkövetelményekből meg kell határozni az időzítési korlátozásokat.
  - Meg kell mutatni, hogy ezek a korlátozások ki vannak elégítve.
- A hardver ilyen irányú vizsgálata kidolgozott, dinamikus teszteléssel, vagy szimulációs ellenőrzéssel.
- A szoftver időbeli sajátságainak vizsgálata már általában sokkal bonyolultabb, a viselkedés összetettsége következtében.

# Környezeti szimuláció

- A biztonságkritikus rendszerek esetében gyakran nincs arra lehetőség, hogy a rendszert teljes egészében a valós működési körülmények között vessük vizsgálat alá.
- Az ilyen szituációkban az elterjedten alkalmazott megoldás a rendszer valós környezetének a szimulációja.

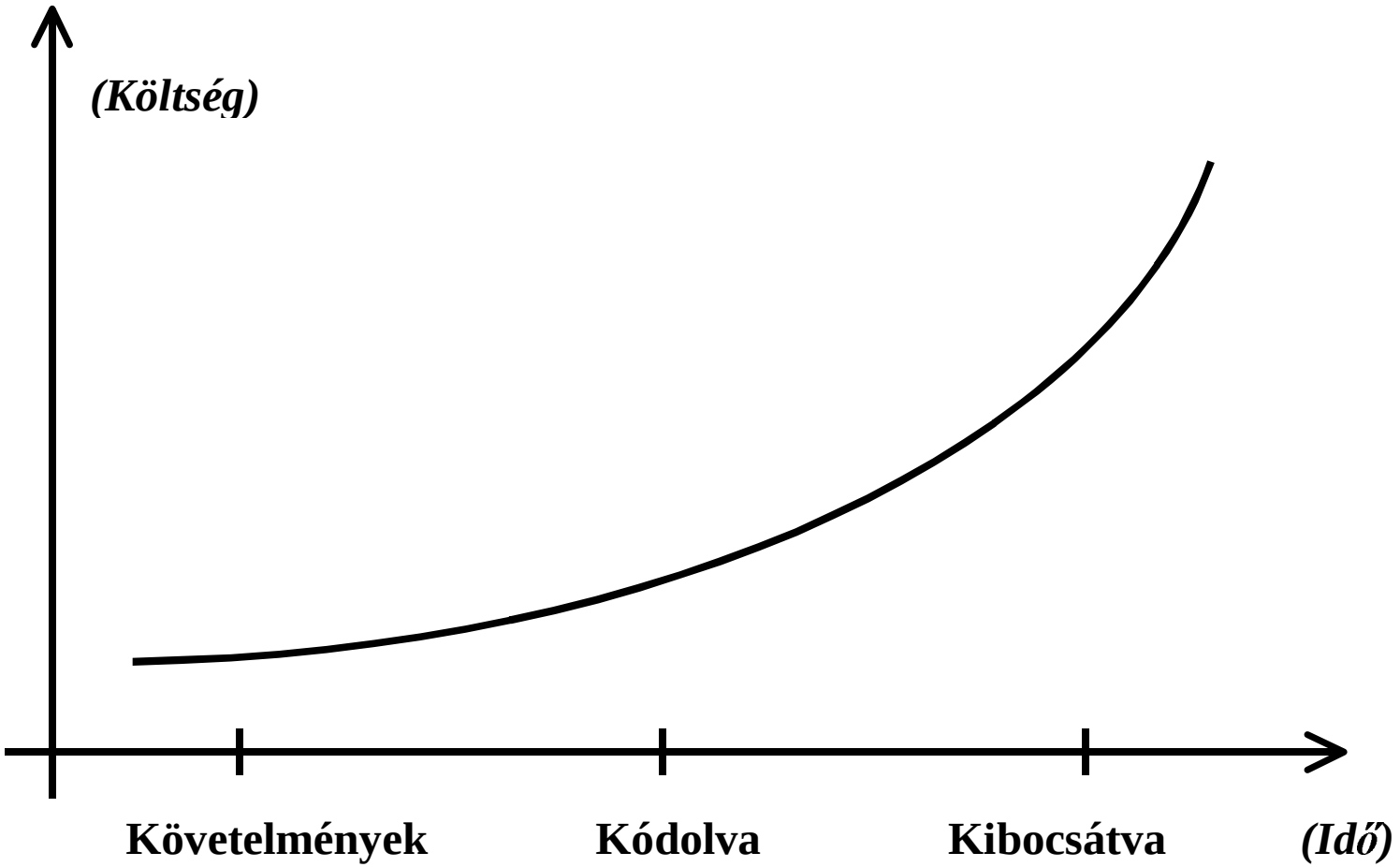
# Környezeti szimuláció

- A szimulátorok minőségének meghatározását több tényező befolyásolja, melyek közül az alábbiakban sorolunk fel néhányat:
  - A szimulált változók: Mely környezeti változók vannak figyelembe véve, és melyek vannak elhanyagolva.
  - Az alkalmazott modellek pontossága: Azoknak a modelleknek a pontossága, amelyek a különböző környezeti hatásokat reprezentálják.
  - A számítások pontossága: Az egyes változók kiszámításánál alkalmazott felbontási fok és pontosság.
  - Az időzítések figyelembevétele: A számítási sebesség hatása a szimulációra.

# A tesztelés költsége

- Magától értetődik, hogy minden modulokból felépülő rendszernél növekedni fog a hibakeresés és hibajavítás költsége, ha arra az építési folyamat egy későbbi szakaszában kerül sor, ahhoz képest, hogy egy korábbi szakaszban történt volna meg ez.

# A tesztelési költségek változása



# Tesztelési ráfordítások

- Tesztköltségi tényező, – test cost factor –TCF
  - Tesztköltségi tényező =
    (Tesztelés költsége) / (Tesztelés költsége + Tervezés költsége)
  - A gyakorlati tapasztalatok szerint a tesztköltségi tényező tipikusan a 0,25 – 0,75 közötti sávban mozog.

# Bizonylatolás

- Annak igazolása, hogy a termék minden tekintetben megfelel a hatósági előírásoknak, szabványoknak, követelményeknek.
- A fejlesztőnek meg kell mutatnia, hogy minden fontosabb hazárd azonosítva lett, és gondoskodás történt az elkerülésükről, továbbá hogy a rendszer kellő védelmet kapott az illetéktelen hozzáférés és helytelen kezelés ellen.
- Az üzemeltetéshez előírt bizonylat kiadásához szükség van annak igazolására is, hogy az adott rendszer biztonságos. Az ehhez végrehajtandó folyamatot nevezzük **biztonságigazolásnak**.

**TESZTELÉSI STRATÉGIÁK**

# Biztonságigazolási dokumentumok

- A biztonságkritikus rendszer leírása.
- A biztonsági tervezésben részt vevő személyzet kompetenciájának igazolása.
- A biztonsági követelmények specifikációja.
- A hazárdok és kockázatok analízisének eredményei.
- A kockázatok csökkentésére alkalmazott módszerek részletezése.
- A tervezési elemzések eredménye, amely azt mutatja meg, hogy a rendszer tervezése minden szempontból elősegíti a biztonsági célok elérését.
- A verifikációs és validációs stratégia.
- Az összes verifikációs és validációs tevékenység eredményei.
- A biztonsági vizsgálatok eredményei.
- Azon események feljegyzése, amelyek a rendszer üzemeltetése során léptek fel.
- A rendszeren végrehajtott változtatások feljegyzése, és annak bizonyítása, hogy a rendszer továbbra is megőrizte biztonságosságát.

**TESZTELÉSI STRATÉGIÁK**

# A formális módszerek szerepe

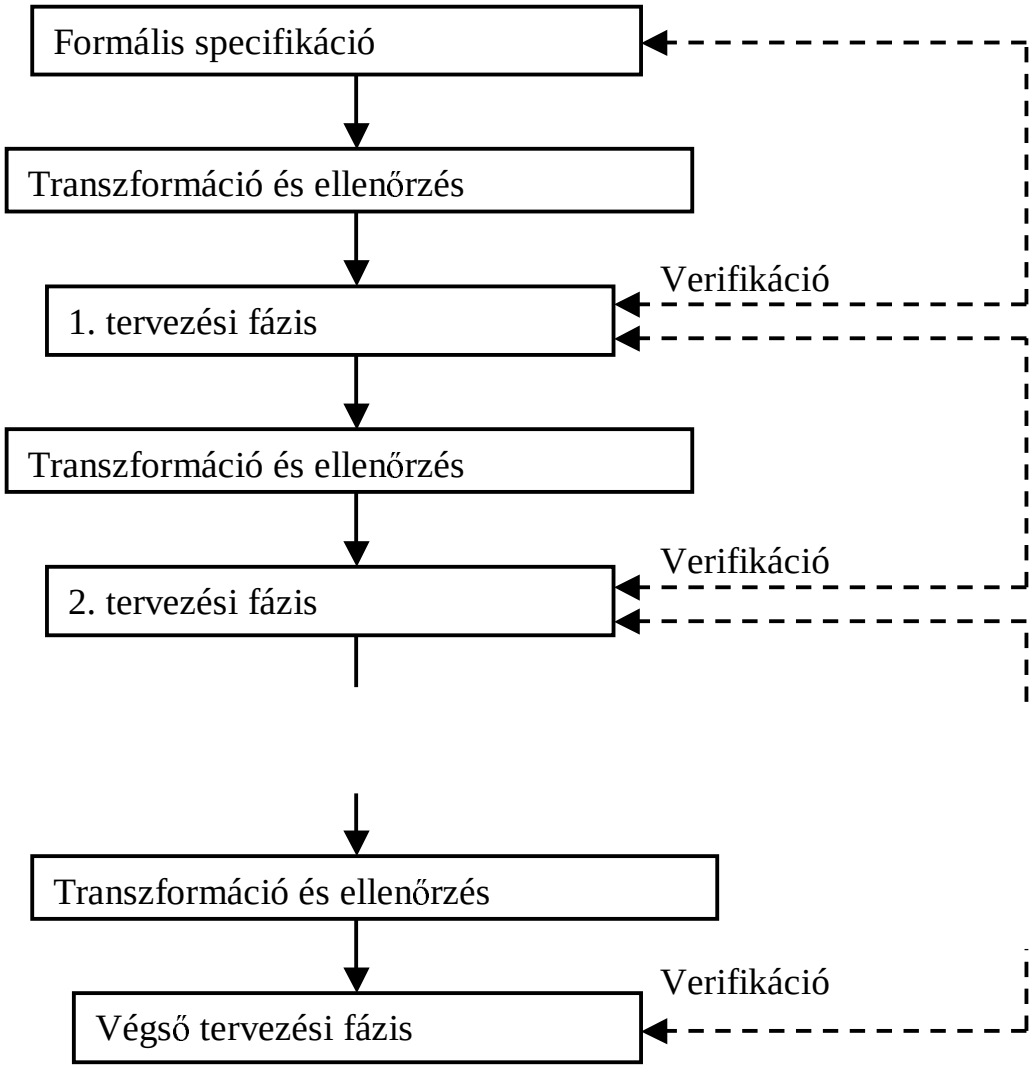
- A formális módszerek matematikai alapokon álló eljárások, amelyeket az informatikai rendszerek
  - specifikálására,
  - tervezésére,
  - analízisére.
- Egyértelmű, pontos és ellentmondásmentes specifikáció megteremtése.
  - CASE-eszközök
  - formalizált módszerek

# Formális módszerek

- Formális nyelveken alapulnak
- Automatikus ellenőrzés a tervezési hibák, ellentmondások kimutatására
- A specifikáció és a végtermék közötti fejlesztési út több fázisból áll.
- Az egyes fázisok közötti átmenetet több esetben automatikus transzformáció tudja biztosítani.
- Az átalakítás, transzformáció során szükség van annak bizonyítására is, hogy az új reprezentáció ekvivalens az őt megelőzővel.

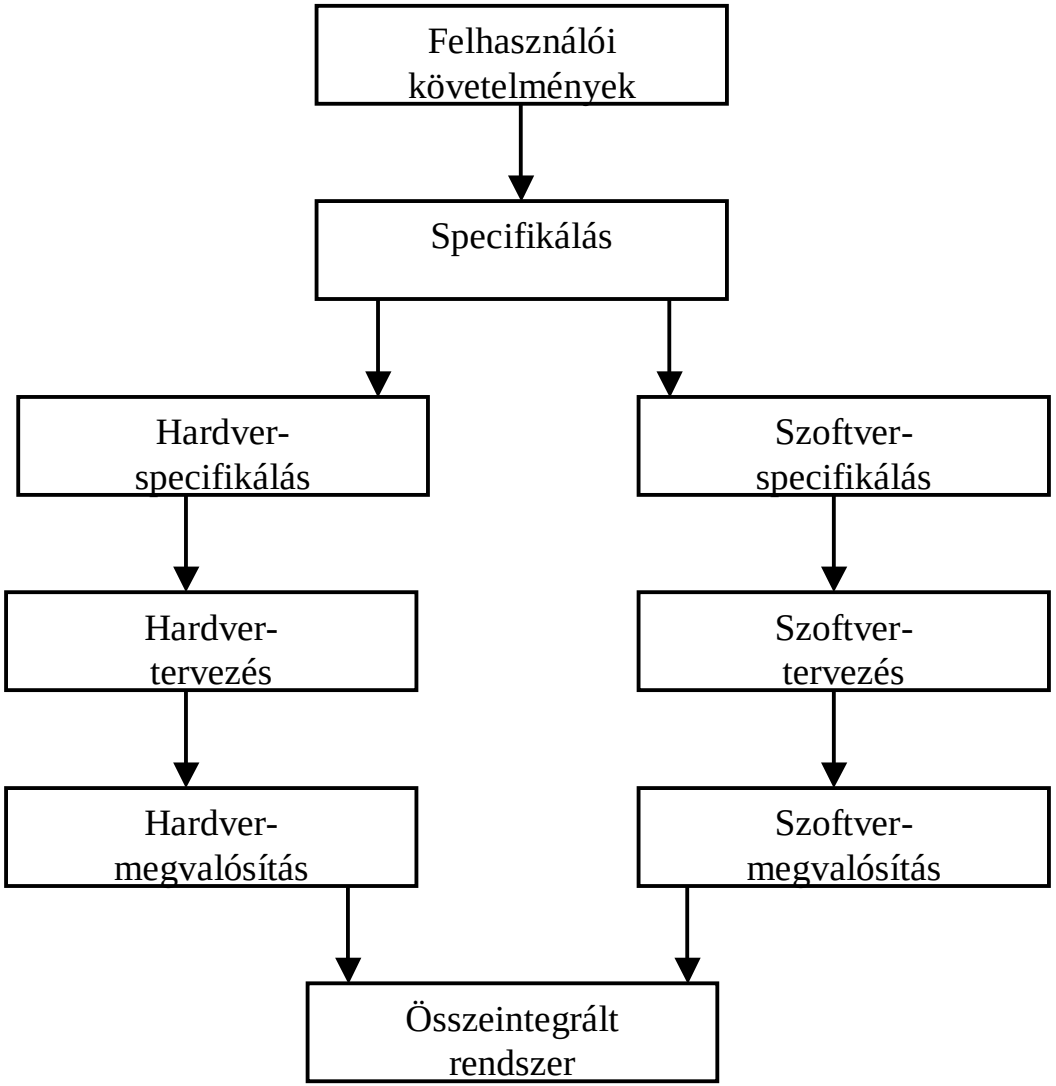
**TESZTELÉSI STRATÉGIÁK**

# A formális módszerek alkalmazása



## TESZTELÉSI STRATÉGIÁK

A hardver és szoftver tervezési és előállítási folyamata



**TESZTELÉSI STRATÉGIÁK**

# A HW&SW tervezési és előállítási folyamata

- A hardverben például nagy jelentőségre tett szert a VHDL (Hardware Description Language).
  - magas szintű nyelv, amely alkalmas a hardver logikai és időbeli működési módjainak definiálására.
- Több jól bevált „silicon compiler” létezik már.
- A szoftverben terjedőben van az UML (Unified Modeling Language, - egységes modellező nyelv) felhasználása.