4.
A szoftvergyártás folyamata

Kérdések

- Mi a szoftvergyártás modellje?
- Mi a három alapvető modell és mikor használjuk ezeket?
- Mik a követelménytervezés, a szoftverfejlesztés, a tesztelés és az szoftver-evolúció főbb elemei?
- Mi a „Rational Unified Process” modell?
- Mi a CASE technológia a szerepe a szoftvergyárgyás folyamatában?

Tartalom

1. A szoftvergyártás modelljei
2. Iteratív szoftverfejlesztés
3. A szoftvergyártás lépései
4. A Rational Unified Process
5. A számítógéppel segített szoftverfejlesztés

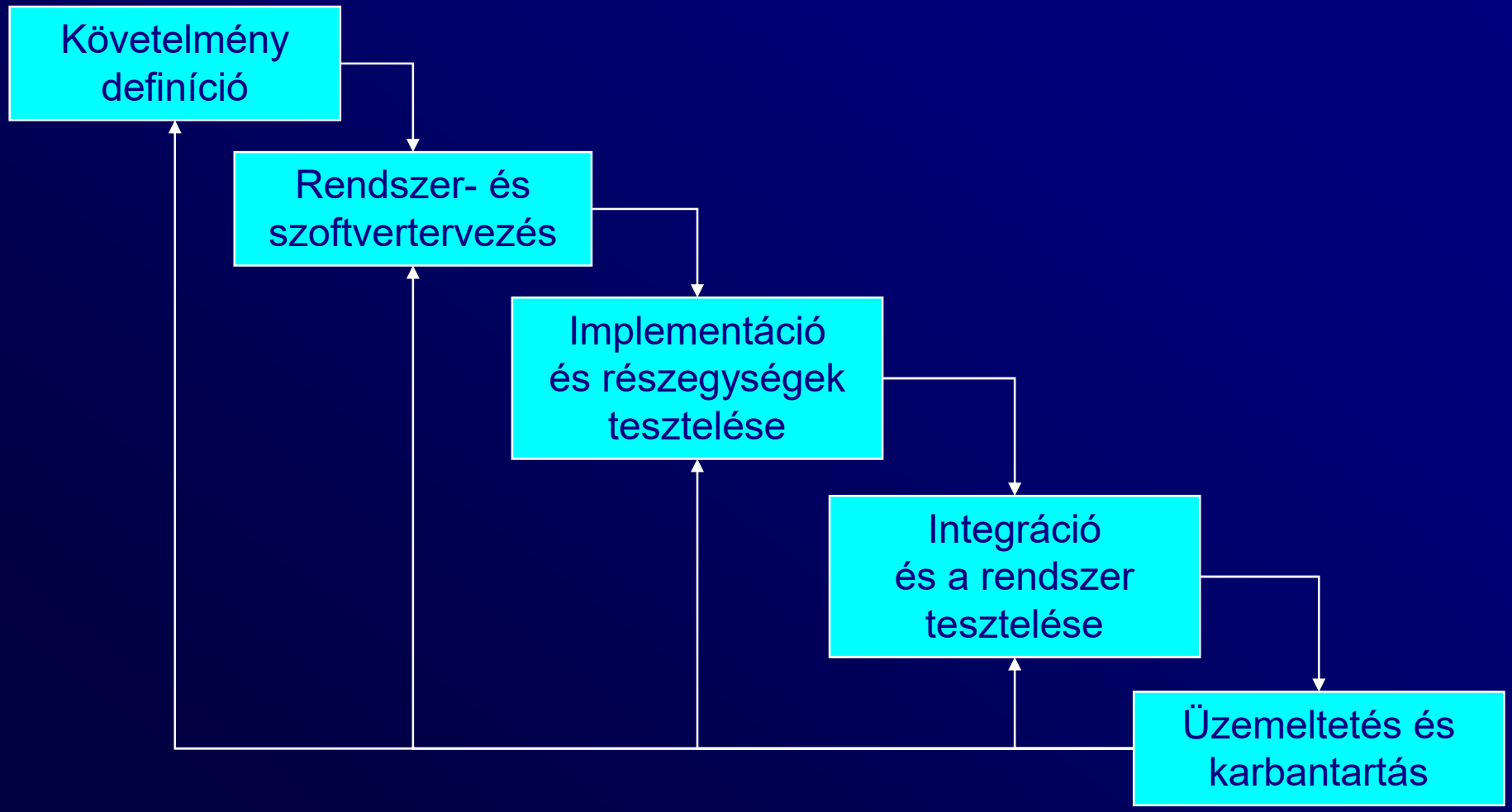
1. Szoftvergyártás

- Tevékenységek olyan strukturált sorozata, amelyek a szoftverek kifejlesztéséhez szükségesek
  - Specifikáció;
  - Tervezés;
  - Ellenőrzés (validáció);
  - Továbbfejlesztés (evolúció).
- A szoftvergyártás absztrakt modellje a gyártási tevékenységet írja le egy adott nézőpontból.

Alapvető szoftvergyártási modellek

- A vízesés (waterfall) modell
  - Élesen elkülönülő specifikációs és fejlesztési fázisok.
- Evolúciós fejlesztési modellek
  - A specifikáció, fejlesztés és validáció átlapolódik.
- Komponens alapú fejlesztés
  - A rendszert kész komponensekből állítjuk össze.
- A fenti modelleknek számos variációja létezik. Pl. formális fejlesztési modell: ez vízesés modellen alapul, ahol a specifikáció formális, ami sok lépésben finomítva elvezet az implementálható tervig.

1.1 A vízesés modell



A vízesés modell fázisai

- Követelményanalízis és – definíció
- Rendszer- és szoftvertervezés
- Implementáció és a részegységek tesztelése
- Részegységek integrálása és a rendszer tesztelés
- Működtetés és karbantartás
- A vízesés modell legfőbb hátrányai:
  - A gyártás megindulás a után nehéz változásokat beépíteni.
  - Egy munkafázisnak be kell fejeződni, mielőtt a következő elkezdődhet.

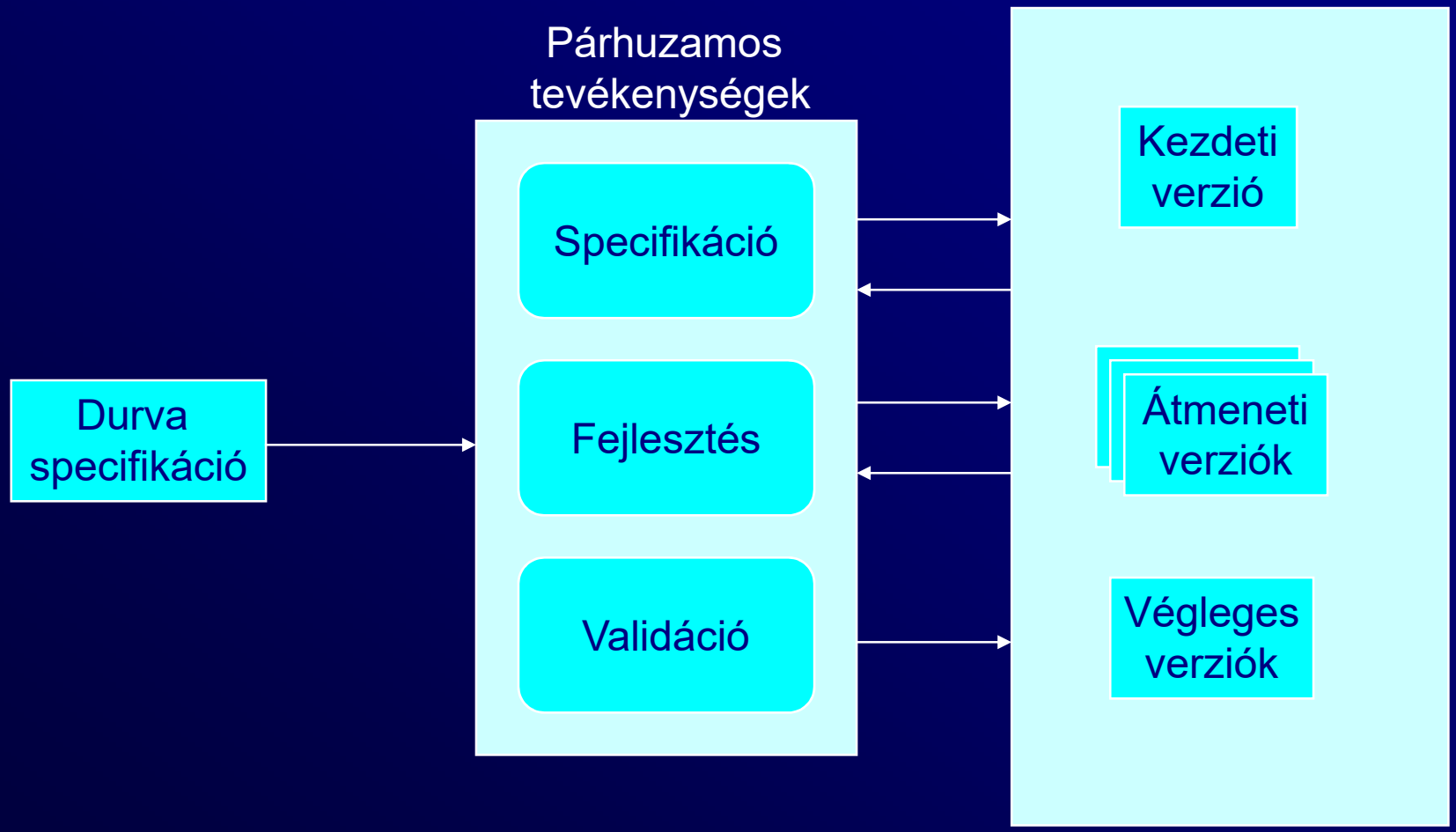
A vízesés modell problémái

- Nehéz a változó megrendelői igényekhez igazodni, mert a projekt nehezen változtatható részegységekből áll.
- Ez a modell akkor hasznos, ha a követelmények jól ismertek és csak nagyon kis változások lehetségesek a fejlesztés során.
- Sajnos csak kevés üzleti rendszernek vannak stabil követelményei.
- A vízesés modellt főleg nagy rendszerek fejlesztése során használják, ahol a fejlesztés több helyszínen történik.

1.2 Evolúciós fejlesztés

- Kísérletező fejlesztés
  - Cél: a megrendelővel együtt egy kezdeti durva specifikációból a végleges rendszert kialakítani. A biztos követelményekből kiindulva a megrendelő igényei szerint újabb funkciókkal bővíthető a rendszer.
- Eldobható prototípus
  - Cél: a homályos követelmények tisztázása. A legkevésbé kiforrott követelményekből indul, hogy tisztázza a valós igényeket.

Evolúciós fejlesztés



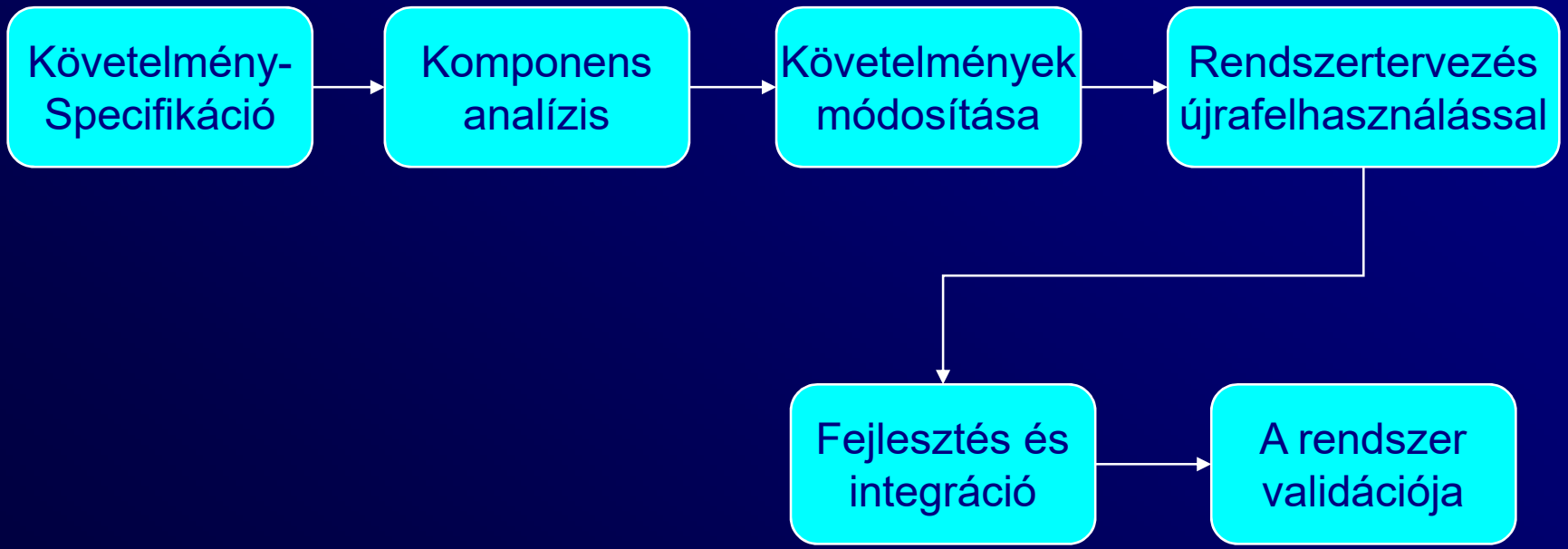
Evolúciós fejlesztés

- Problémák
  - A fejlesztés nem átlátható;
  - A rendszerek gyakran rosszul strukturáltak;
  - Speciális felkészültségre lehet szükség (pl. *rapid prototyping* nyelvek).
- Alkalmazhatóság
  - Kis- és középméretű interaktív rendszerek;
  - Nagy rendszerek részegységei (pl. felhasználói felület);
  - Rövid élettartamú rendszerek.

1.3 Komponens-alapú szoftverfejlesztés

- Szisztematikus újrafelhasználáson alapul. A rendszereket már létező, vagy készen vásárolható (COTS) rendszerekből integráljuk.
- A szoftvergyártás lépései:
  - Komponens analízis;
  - Követelmények módosítása;
  - Rendszertervezés újrafelhasználással;
  - Fejlesztés és integráció.
- Egyre szélesebb körben terjed, ahogy a komponens szabványok fejlődnek.

Újrafelhasználás-alapú fejlesztés



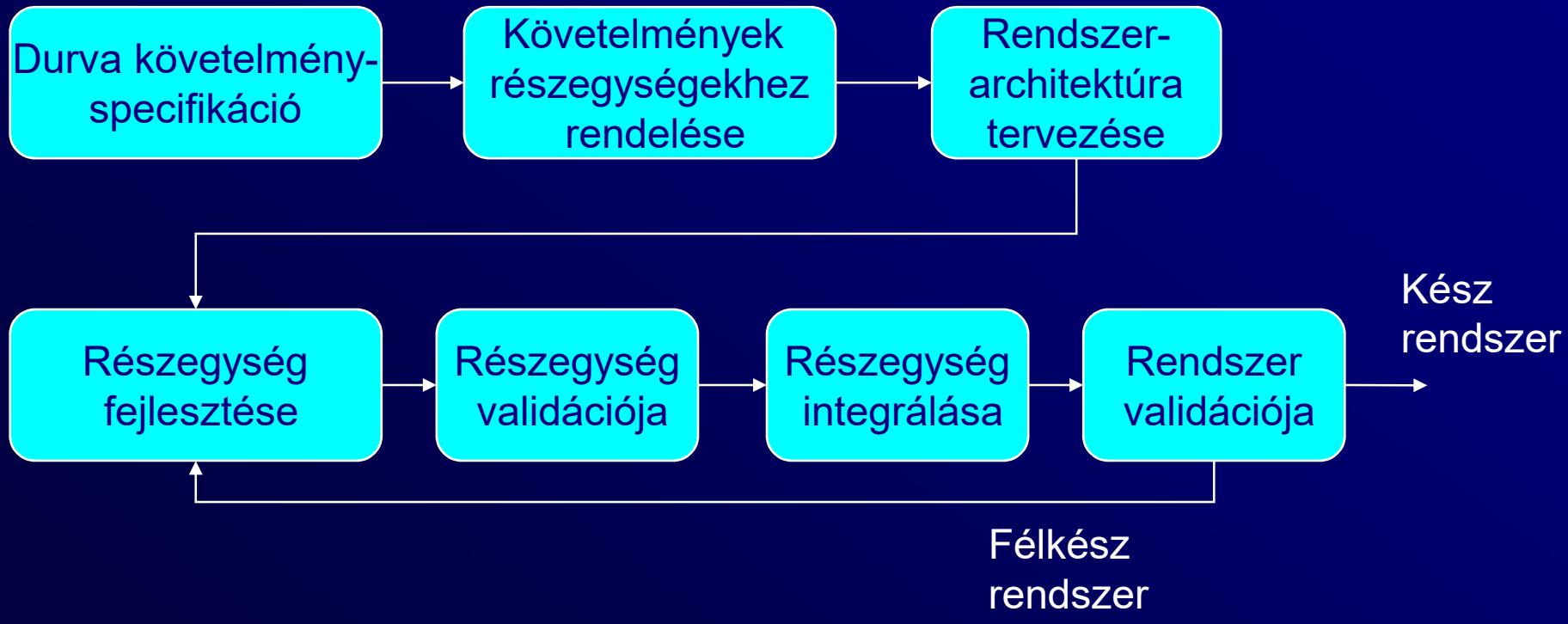
2. Iteratív szoftverfejlesztés

- A rendszerkövetelmények MINDEN projekt során változnak, így az iteratív megközelítés (korábban elvégzett munkafázisok átdolgozása) minden nagyobb rendszer fejlesztésének része.
- Az iteratív megközelítés valamennyi alapvető módszerhez alkalmazható.
- Két kapcsolódó megközelítés:
  - Inkrementális teljesítés
  - Spirális fejlesztés

2.1 Inkrementális teljesítés

- A rendszert nem egy részletben szállítjuk, hanem a fejlesztés és átadás részekre van bontva. Minden újabb átadott részegység a rendszer újabb funkcionalitását valósítja meg.
- A felhasználó igényeknek megfelelő prioritási sorrendben szállítunk, a legfontosabb funkciókkal kezdve.
- Amint egy részegység fejlesztése elkezdődött, annak követelményeit „befagyasztjuk”. Későbbi részegységek követelményei még változhatnak.

Inkrementális teljesítés



Az inkrementális teljesítés előnyei

- Minden átadás során működő részegységeket helyezünk üzembe. A rendszer korábban kezdheti meg (rész)működését.
- Korábbi komponensek prototípusként működnek, így a későbbi részegységek követelménytervezésében ezek is segítenek.
- Kisebb a projekt teljes csődjének esélye.
- A legfontosabb szolgáltatásokat tesztelik a legtovább.

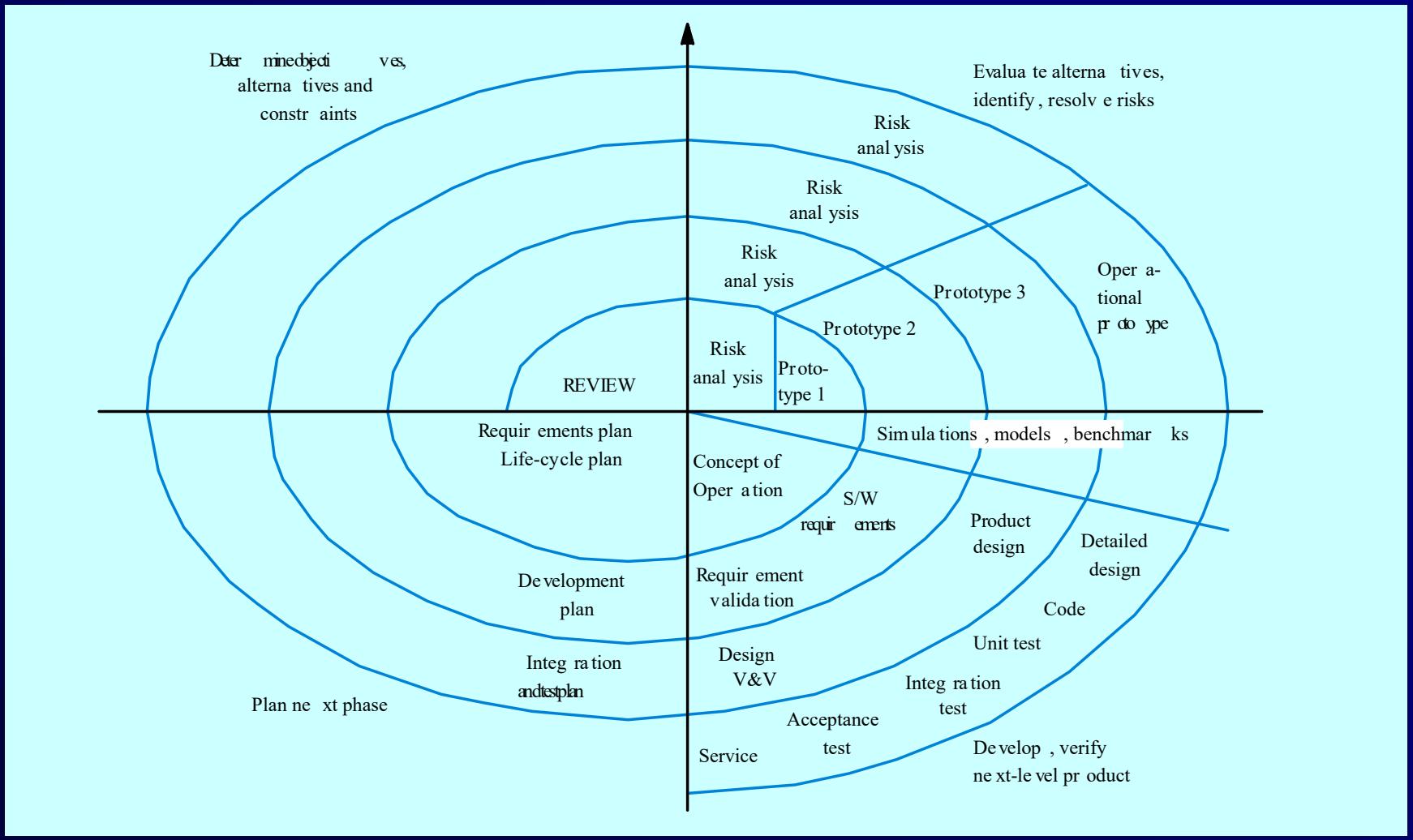
Extrém programozás

- Itt a fejlesztés és átadás nagyon kis funkcionális részegységenként történik.
- Állandó kódjavítás, a felhasználó aktív részvétele a fejlesztésben, valamint „páros programozás” jellemzik.

2.2 Spirális fejlesztés

- A gyártási folyamat sokkal inkább egy spirállal jellemezhető, mint tevékenységek (visszalépéses) sorozataként.
- A spirál minden hurka a gyártási folyamat egy fázisát jelképezi.
- Nincsenek fix hurkok (pl. specifikáció, vagy tervezés). A hurkokat az igényeknek megfelelően alakítjuk ki.
- A kockázatkezelés explicit módon megjelenik a gyártási folyamatban.

A szoftvergyártás spirális modellje



A spirális modell szektorai

- Célkitűzések megállapítása
  - Az adott fázis céljainak megállapítása.
- Kockázatbecslés és -csökkentés
  - A kockázati tényezők felmérése, valamint a legfőbb kockázati faktorok várható hatásának csökkentése.
- Fejlesztés és validáció
  - Az általános módszerek közül bármely kiválasztása.
- Tervezés
  - A projekt áttekintése és a spirál következő fázisának megtervezése.

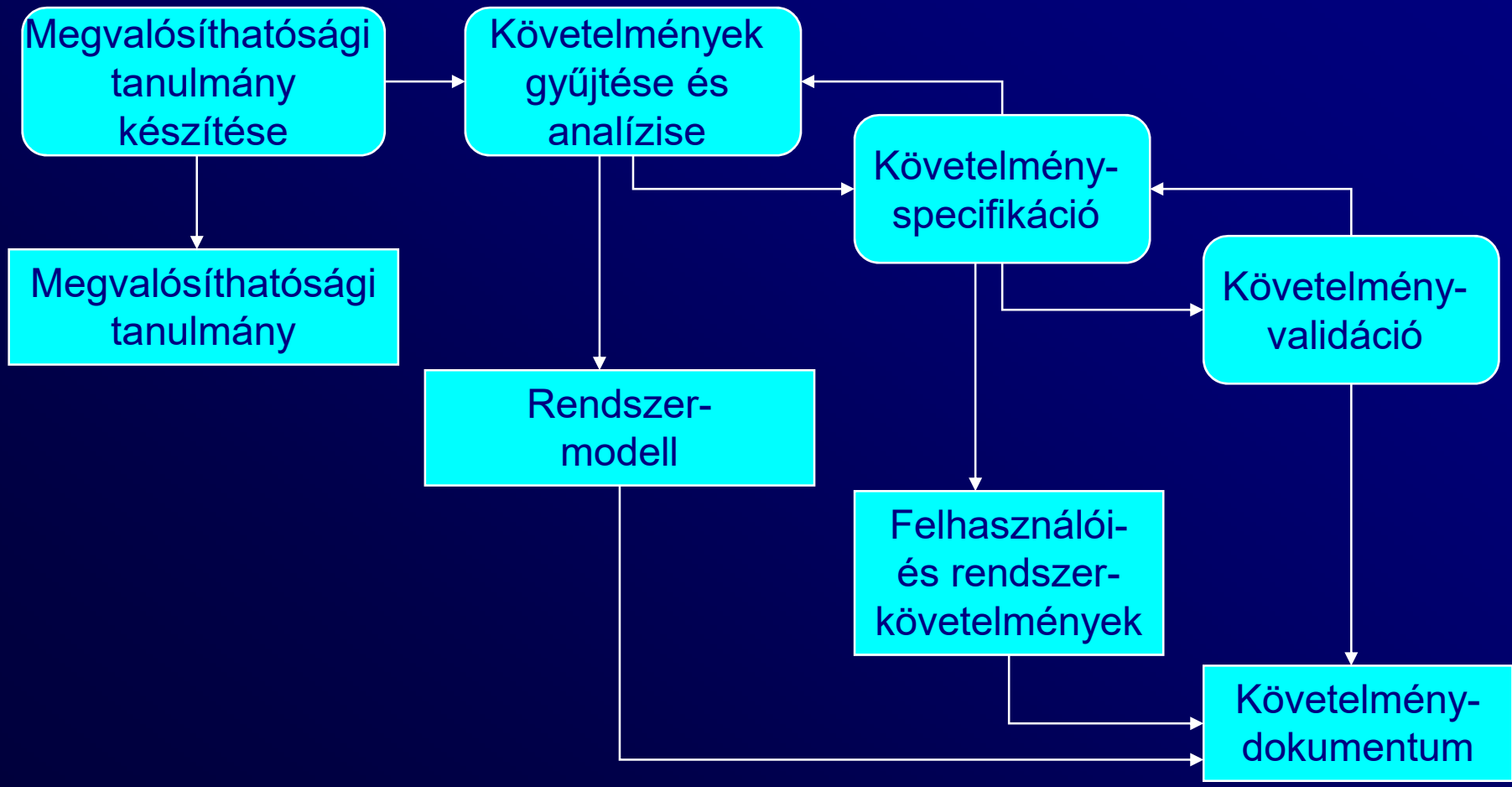
3. A szoftvergyártás lépései

- Szoftver specifikáció;
- Szoftver tervezés és implementáció;
- Szoftver ellenőrzés (validáció);
- Szoftver továbbfejlesztés (evolúció).

3.1 Szoftver specifikáció

- Választ keresünk a következő kérdésekre: milyen szolgáltatásokat várunk el a rendszertől, milyen kötöttségeket és kényszereket kell figyelembe venni a fejlesztés és üzemeltetés során.
- Követelménytervezési lépései:
  - Megvalósíthatósági tanulmány;
  - Követelmények gyűjtése és analízise;
  - Követelmény specifikáció;
  - Követelmény validáció.

A követelménytervezési eljárás



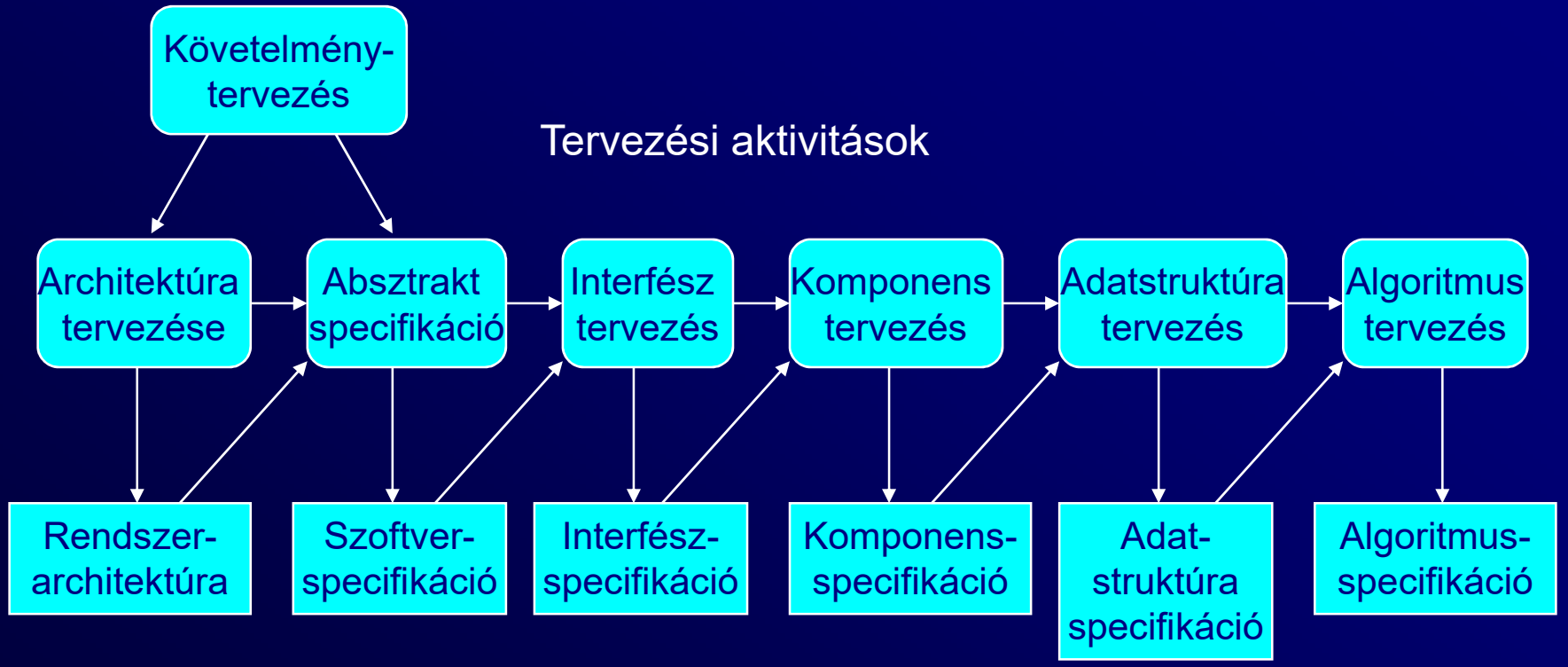
3.2 Szoftvertervezés és implementáció

- Az az eljárás, amelynek során a specifikáció egy futtatható rendszerré alakul át.
- Szoftver tervezés
  - Olyan szoftver struktúra tervezése, amely megvalósítja a specifikációt;
- Implementáció
  - A tervezett struktúrának végrehajtható kóddá alakítása;
- A tervezés és implementálás lépései egymással szorosan összefüggnek és átlapolódhatnak.

A tervezés lépései

- Architektúra tervezése
- Absztrakt specifikáció
- Interfészek tervezése
- Komponensek tervezése
- Adatstruktúrák tervezése
- Algoritmusok tervezése

A szoftvertervezés folyamata



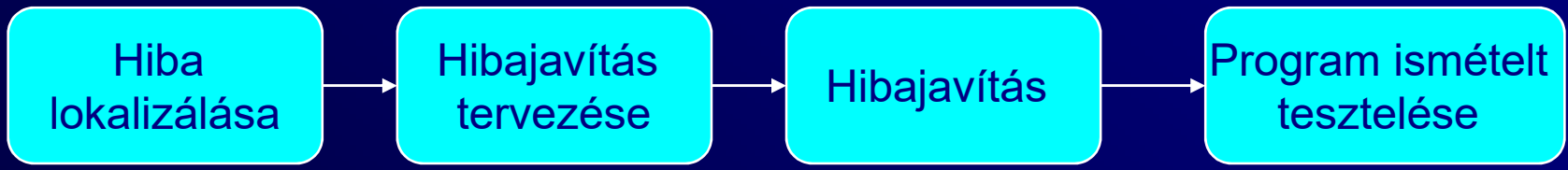
Strukturált módszerek

- Szisztematikus szoftvertervezési módszerek
  - Automatikus kódgenerálás
- A terv dokumentálása rendszerint grafikus modellek segítségével történik.
- Lehetséges modellek:
  - Objektum modell;
  - Szekvenciális modell;
  - Állapot-átmeneti modell;
  - Strukturális modell;
  - Adatfolyam modell.

Programozás és hibakeresés

- A terv programmá alakítása, valamint a hibák eltávolítása.
- A programozás egyéni tevékenység – nincs rá általános módszer.
- A programozók tesztelést végeznek, hogy a programhibák kiderüljenek, majd ezeket kijavítják (hibakeresés, debuggolás).

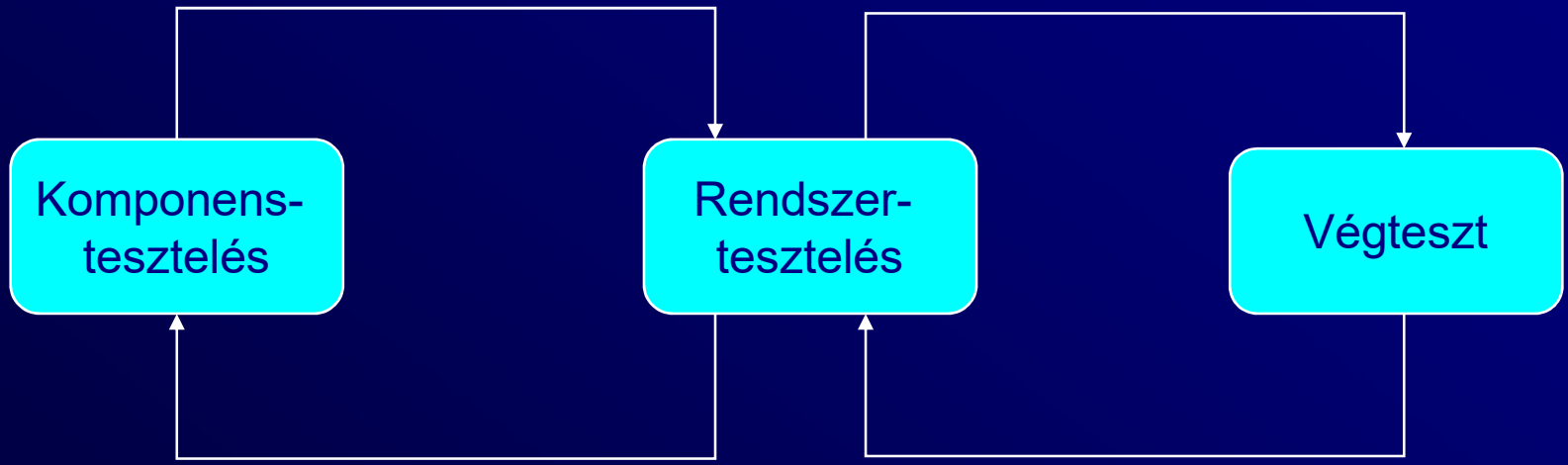
A hibakeresés folyamata



3.3 Szoftver validáció

- A verifikáció és validáció (V & V) célja annak bizonyítása, hogy a rendszer teljesíti a specifikációban foglaltakat és a felhasználó igényeinek megfelelően működik.
- Elemei: Ellenőrzés, felülvizsgálat és rendszertesztelés.
- Rendszertesztelés: a rendszer futtatása olyan tesztadatokkal, amely a specifikáció szerint a valós működés során előfordulhat.

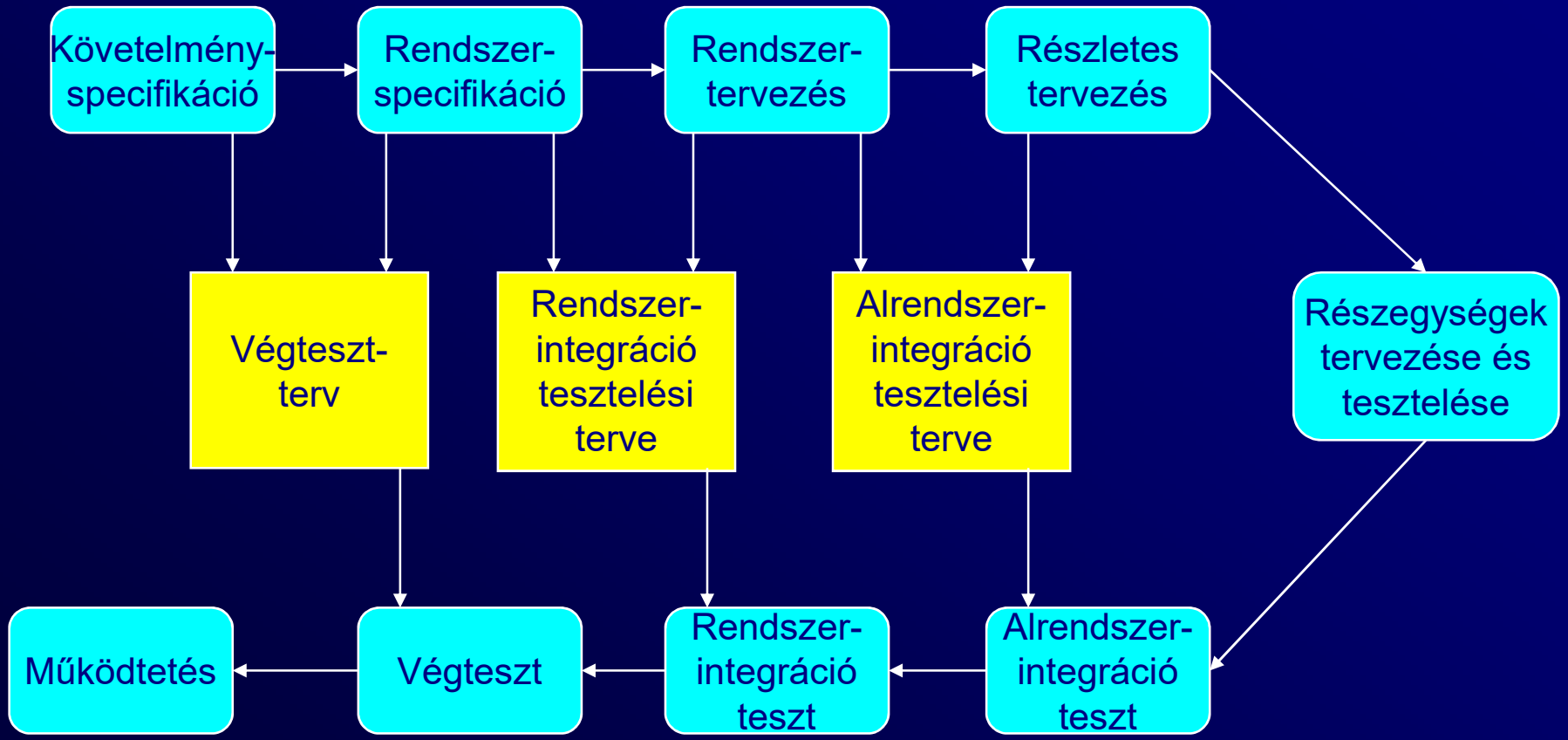
A tesztelési eljárás



A tesztelés lépései

- Komponens- és részegység-tesztelés
  - A különálló komponenseket egymástól függetlenül teszteljük;
  - A komponensek lehetnek: függvények, objektumok, vagy ezek összetartozó csoportjai.
- Rendszertesztelés
  - A rendszer egészének tesztelése. Különösen fontos az eredő tulajdonságok ellenőrzése.
- Végteszt (átadási teszt)
  - A megrendelő által szolgáltatott valós adatokkal annak ellenőrzése, hogy a megrendelő igényeit valóban kielégíti.

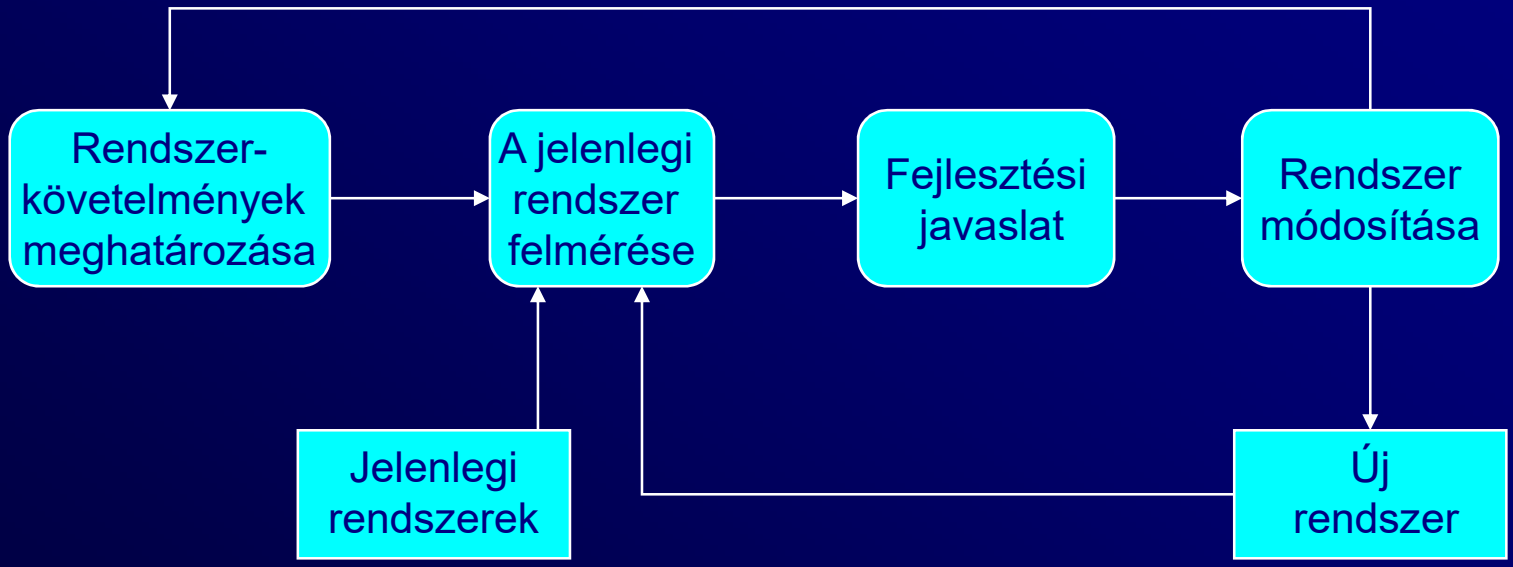
A tesztelés fázisai



3.4 A szoftver evolúciója

- A szoftver eredendően rugalmas és változtatható.
- Ahogy a változó üzleti-gazdasági körülmények miatt a követelmények változnak, a kiszolgáló szoftvernek is változnia és fejlődnie kell.
- Bár a fejlesztés és karbantartás között régebben éles határvonal húzódott, ez egyre kevésbé releváns, hiszen egyre kevesebb a teljesen új rendszer (evolúció).

Rendszerek evolúciója



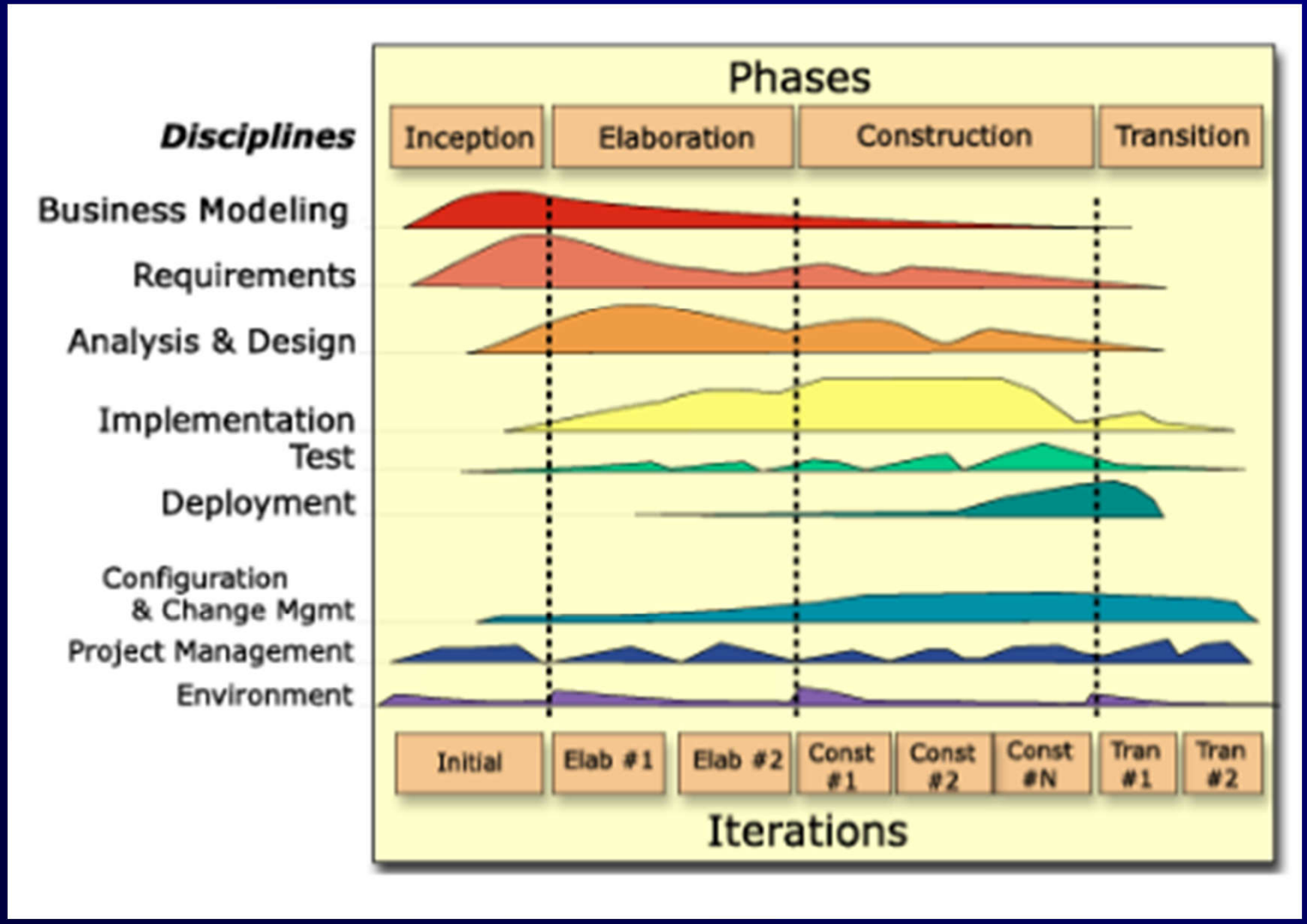
4. A Rational Unified Process

- Korszerű tervezési módszer, amely az UML, és a hozzá kapcsolódó eljárásokból jött létre.
- Definiálja: ki, mit, hogyan
- Általában három nézetet használunk:
  - Dinamikus nézet: a fejlesztési ciklus fázisait az idő és a tartalom függvényében mutatja;
  - Statikus nézet: A gyártási folyamat tevékenységeit mutatja;
  - Gyakorlati nézet: Jól bevált gyakorlati útmutató.

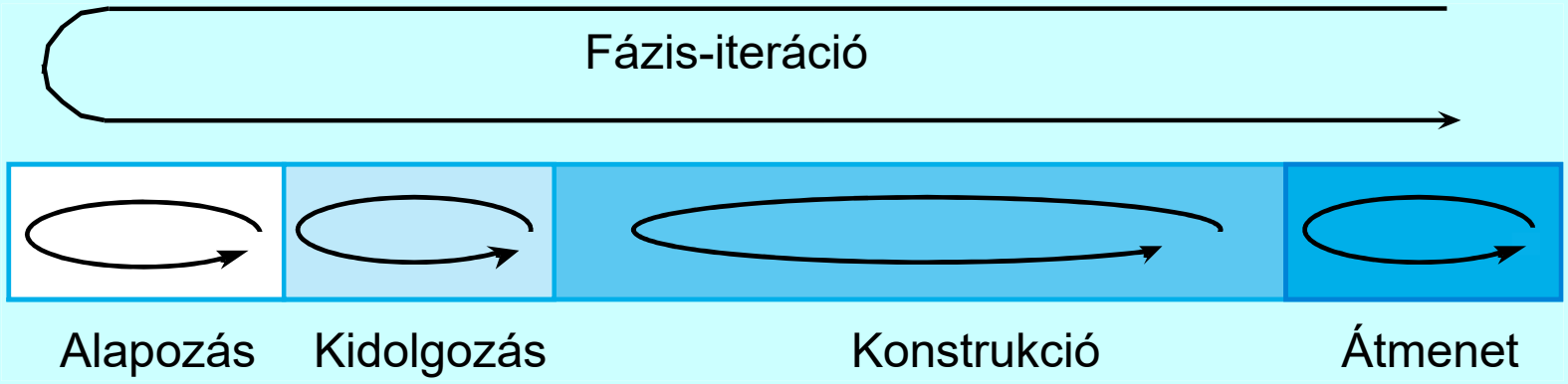
Ki, mit, hogyan

- Ki
  - gyakorlatok, kompetenciák, felelősségek
- Mit
  - dokumentumok, modellek, kód
- Hogyan
  - munkaegységek, feladatok, folyamatok

4.1 Dinamikus nézet



A RUP fázismodellje



A RUP fázisai

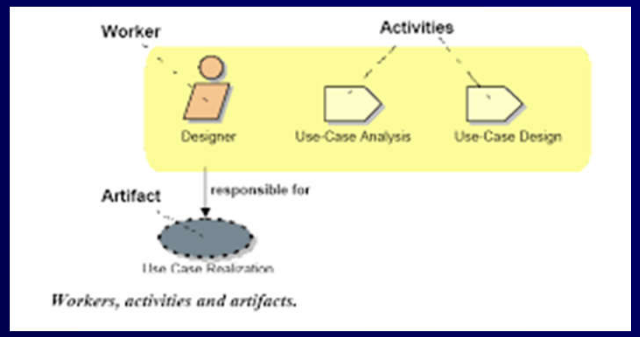
- Alapozás
  - A rendszer számára egy üzleti modell megalkotása.
- Kidolgozás
  - A problématér megértése és a rendszer-architektúra kidolgozása.
- Konstrukció
  - Rendszertervezés, programozás és tesztelés.
- Átmenet
  - A rendszer telepítése a működési környezetbe.

4.2 Statikus nézet

- A fejlesztési folyamat alatti tevékenységekre fókuszál: munkafolyamatok
- 6 alap munkafolyamat
  - üzleti modellezés, követelmények, analízis és tervezés, implementáció, tesztelés, telepítés
- 3 támogató munkafolyamat
  - konfiguráció és változás menedzsment, projekt menedzsment, környezet

Munkafolyamatok

- A tevékenységek, termékek, szerepek (munkások) értelmes sorrendjét definiálja
- Bármely munkafolyamat bármelyik fázisban aktív lehet
- A munkafolyamat leírás UML modellek köré van szervezve
  - use case model, sequence model, object model, stb.



Workers, activities and artifacts.

Munkafolyamok

Munkafolyam	Leírás
Üzleti modellezés	Az üzletmenet esettanulmányokkal (*use case*) való modellezése.
Követelmények	A rendszerrel kapcsolatba lépő *aktorok* azonosítása. Esettanulmányok kidolgozása a követelmények modellezésére.
Analízis és tervezés	Tervezési modell kidolgozása és dokumentálása architektúrális-, komponens-, objektum-, valamint szekvenciális modellek segítségével
Implementáció	Rendszerkomponensek implementálása és alrendszerekké alakítása. A tervezési modellekből automatikus kódgenerálás segíti ennek a folyamatnak a gyorsítását.
Tesztelés	A tesztelés iteratív eljárás amely az implementációval együtt hajtódik végre. A rendszertesztelés az implementáció befejeztével kezdődik.
Telepítés	Egy *release* készül, amelyet a felhasználóknak terjesztve ott installálnak.
Konfiguráció- és változás-menedzsment	Ez a kiegészítő munkafolyam menedzseli a rendszeren végrehajtott változtatásokat.
Projekt-menedzsment	Ez a kiegészítő munkafolyam menedzseli a rendszerfejlesztést.
Környezet	Ez a munkafolyam a fejlesztő csapat megfelelő szoftvereszközökkel való ellátásával foglalkozik.

4.3 Gyakorlati útmutatók

- Ajánlott szoftvermérnöki technikák
- Hibák minimálisra csökkentése
- Hatékonyság maximálisra növelése

6 gyakorlati útmutató

- Iteratív szoftverfejlesztés
- Követelmény-menedzsment
- Komponens-alapú architektúrák használata
- Szoftver vizuális modellezése
- Szoftver minőség verifikálása
- Szoftver változások kontrollja

5. Számítógéppel segített szoftverfejlesztés

- CASE (Computer-aided software engineering) olyan szoftver, amely a szoftverfejlesztés és evolúció folyamatát segíti.
- Tevékenységek automatizálása
  - Grafikus szerkesztők rendszermodellek fejlesztésére;
  - Adatkönyvtár tervezési entitások menedzselésére;
  - Grafikus felhasználó felület szerkesztő;
  - Debuggerek hibakereséshez;
  - Automatikus transzlátorok új programverziók generálásához.

CASE technológia

- A CASE technológia bevezetése jelentős fejlődési lépés volt a szoftvergyártásban, de elmarad az egykor prognosztizált nagyságrendi fejlődéstől...
  - A szoftverfejlesztés kreatív gondolkodást igényel – nem lehet automatizálni;
  - A szoftverfejlesztés nagy projektek esetén csapatmunka, így sok idő fordítódik a csoportok közötti interakcióra. A CASE technológia nem támogatja ezt.

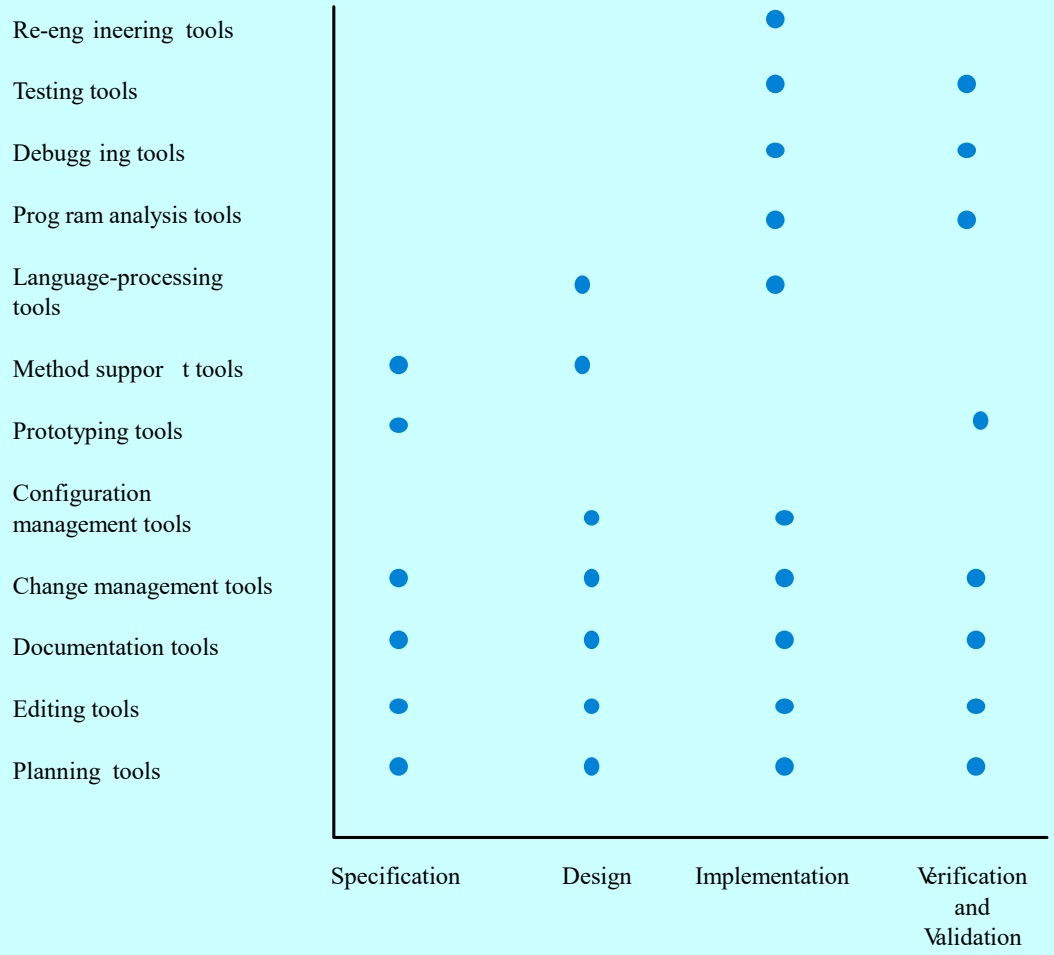
CASE rendszerek osztályozása

- Az osztályozás segít megérteni a különféle CASE eszközök felhasználási lehetőségeit a szoftvergyártás során.
- Funkcionális nézet
  - Az eszközöket funkciójuk szerint osztályozzuk.
- Aktivitás nézet
  - Az eszközöket a folyamatban általuk támogatott tevékenységek szerint osztályozzuk.
- Integrációs nézet
  - Az osztályozás alapja, hogy az eszközök hogyan vannak nagyobb egységekbe szervezve.

Eszközök funkcionális osztályozása

Tool type	Examples
Planning tools	PERT tools, estimation tools, spreadsheets
Editing tools	Text editors, diagram editors, word processors
Change management tools	Requirements traceability tools, change control systems
Configuration management tools	Version management systems, system building tools
Prototyping tools	Very high-level languages, user interface generators
Method-support tools	Design editors, data dictionaries, code generators
Language-processing tools	Compilers, interpreters
Program analysis tools	Cross reference generators, static analysers, dynamic analysers
Testing tools	Test data generators, file comparators
Debugging tools	Interactive debugging systems
Documentation tools	Page layout programs, image editors
Re-engineering tools	Cross-reference systems, program re-structuring systems

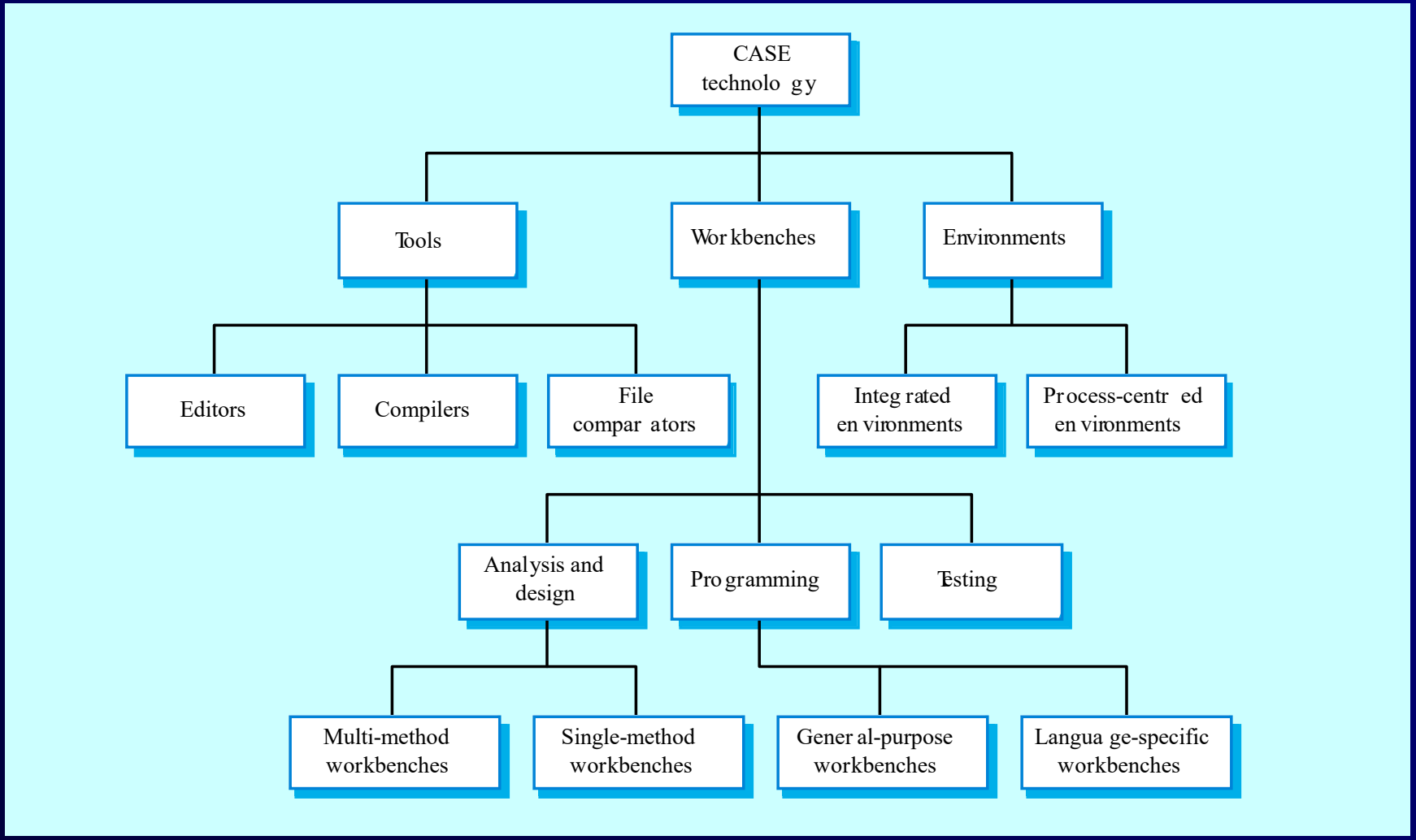
Eszközök aktivitás-alapú osztályozása



CASE eszközök integrációja

- Eszköz (tool)
  - Elemi műveletek támogatása szolgál (pl. konzisztencia-ellenőrzés, szövegszerkesztés, stb.)
- Munkapad (workbench)
  - Egy gyártási fázist támogat (pl. specifikáció, tervezés). Általában néhány integrált eszközt tartalmaz.
- Környezet (environment)
  - Az egész szoftvergyártási folyamat minden lényeges elemét tartalmazza. Általában számos integrált munkapadot tartalmaz.

Eszközök, munkapadok, környezetek



Összefoglalás

- A szoftvergyártás során szoftver rendszerek előállítása / kifejlesztése történik.
- A szoftvergyártás modelljei ezen eljárások absztrakt reprezentációja.
- A legfőbb tevékenységek a specifikáció, tervezés, implementáció, validáció és evolúció.
- Az általános szoftvergyártási modellek a gyártási folyamat szerkezetét írják le. Pl.: vízesés modell, evolúciós fejlesztés és komponens-alapú szoftverfejlesztés.
- Az iteratív modellek a szoftvergyártás folyamatát aktivitások körfolyamataként írják le.

Összefoglalás

- A követelménytervezés a szoftver specifikáció megalkotásának folyamata.
- A tervezési és implementációs folyamatok a specifikációt végrehajtható programmá transzformálják.
- A validáció során ellenőrizzük, hogy a rendszer teljesíti-e a specifikációt és a felhasználó igényei szerint működik-e.
- Az evolúció a rendszer üzembe helyezés utáni módosítása.
- A Rational Unified Process olyan szoftvergyártási modell, amely elválasztja a tevékenységeket a folyamat fázisaitól.
- A CASE technológiák a szoftvergyártás folyamatát támogatják.