

Szoftvertechnológia

2022 Március 23

5. előadás

Szoftverprototípus készítése



KADA ZSOLT

STRATÉGIAI ÉS FEJLESZTÉSI
IGAZGATÓ

GIRO ZRT.



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Tartalom

1

A PROTOTÍPUSKÉSZÍTÉS SZEREPE

2

A PROTOTÍPUSKÉSZÍTÉS MÓDJAI

3

GYORS PROTOTÍPUSKÉSZÍTÉSI TECHNIKÁK

4

TERVEZÉS ÚJRAFELHASZNÁLÁSSAL



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

A prototípuskészítés szerepe

Prototípus készítése

- A **prototípuskészítés** általában a követelménytervezés része, a követelmények feltárásának és validációjának eszköze.
- A **prototípus** a szoftverrendszer kezdeti verziója, amely alkalmas a rendszer vagy egyes funkciók koncepciójának bemutatására és kipróbálására.
- Korábban úgy tekintették, hogy a prototípus alacsonyabb rendű a kívánt rendszernél.
- Ma a prototípus- és a normál rendszer közti határ fokozatosan elmosódik és sok rendszert az evolúciós modell alapján készítenek.



A prototípuskészítés szerepe

A prototípusok alkalmazása

- A felhasználó nem látja előre, hogyan fogja használni az új rendszert.
- A prototípus elsődleges célja az, hogy segítse a felhasználókat a rendszerkövetelmények megértésében:
 - **A követelmények feltárása:** a prototípussal a felhasználók megtapasztalhatják, hogyan fogja a rendszer a munkájukat támogatni.
 - **A követelmények validálása:** a prototípus felfedheti a félreértéseket, hibákat és hiányosságokat a követelményekben.
- A prototípus csökkenti a követelményekkel kapcsolatos kockázatokat.



A prototípuskészítés szerepe

A prototípuskészítés előnyei

- Segít felismerni a szoftver felhasználója és készítője közti félreértéseket. → **A rendszer jobban illeszkedik a felhasználó igényeihez.**
- Kiderülhet, hogy hiányzik valamely szolgáltatás vagy ellentmondások vannak a szolgáltatások között. → **A rendszer használhatóbb lesz.**
- A szoftverfolyamat elején már egy – legalábbis részben, modellként – működő rendszer áll rendelkezésre. → **Gyorsabban elkészül a rendszer.**
- A prototípus felhasználható a rendszerspecifikáció alapjaként. → **Javul a tervezés minősége.**
- Támogathatja a felhasználók képzését és a rendszertesztet is.
- A fejlesztéshez kevesebb erőforrásra van szükség. → **Költségcsökkentés.**



A prototípuskészítés szerepe

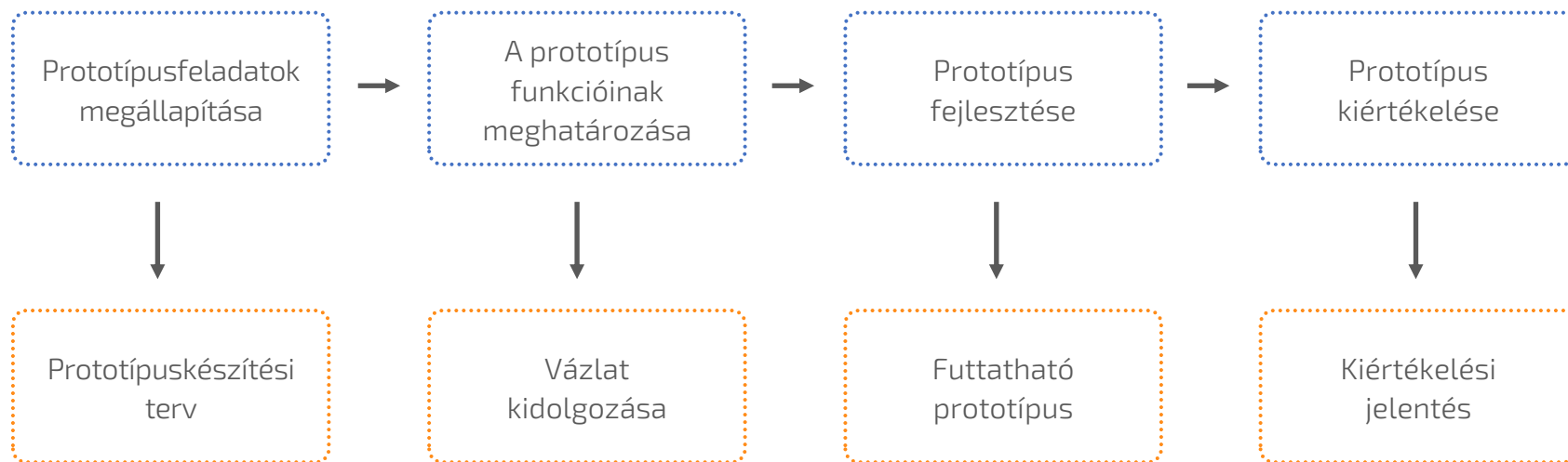
A prototípuskészítés veszélyei

- Eldobható prototípust végleges rendszerként használnak
(*Sérülhet mind a teljesítmény, mind a funkcionalitás, mind a megbízhatóság*)
- A gyors fejlesztésből és az iterációból fakadó hibák:
 - Lassabb válaszidő
 - Bonyolultabb rendszerstruktúra



A prototípuskészítés szerepe

A prototípuskészítés folyamata



Tartalom

1

A PROTOTÍPUSKÉSZÍTÉS SZEREPE

2

A PROTOTÍPUSKÉSZÍTÉS MÓDJAI

3

GYORS PROTOTÍPUSKÉSZÍTÉSI TECHNIKÁK

4

TERVEZÉS ÚJRAFELHASZNÁLÁSSAL



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

A prototípuskészítés módjai

A prototípuskészítés helye a szoftverfolyamatban

- **Evolúciós prototípus készítése:**

- Célja egy működő rendszer átadása a megrendelőnek (*esetleg korlátozott funkcionalitással*).
- A legfontosabb követelmények implementálásával egyszerű rendszer készül, amelyet újabb követelmények feltárásával fokozatosan egészítenek ki új funkciókkal.
- Az **Agilis fejlesztés** alapvető módszere.
- Weblapfejlesztésben és e-business alkalmazásokban is jól használható.

- **Eldobható prototípus készítése:**

- Célja a rendszerkövetelmények feltárása és validálása.
- A nem teljesen megértett követelmények megvalósítása és bemutatása segíti a feltárást.
A követelményspecifikáció elkészülte után nem használható fel.



Evolúciós prototípus készítése



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIMEMLT FELSŐOKTATÁSI INTÉZMÉNY
INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

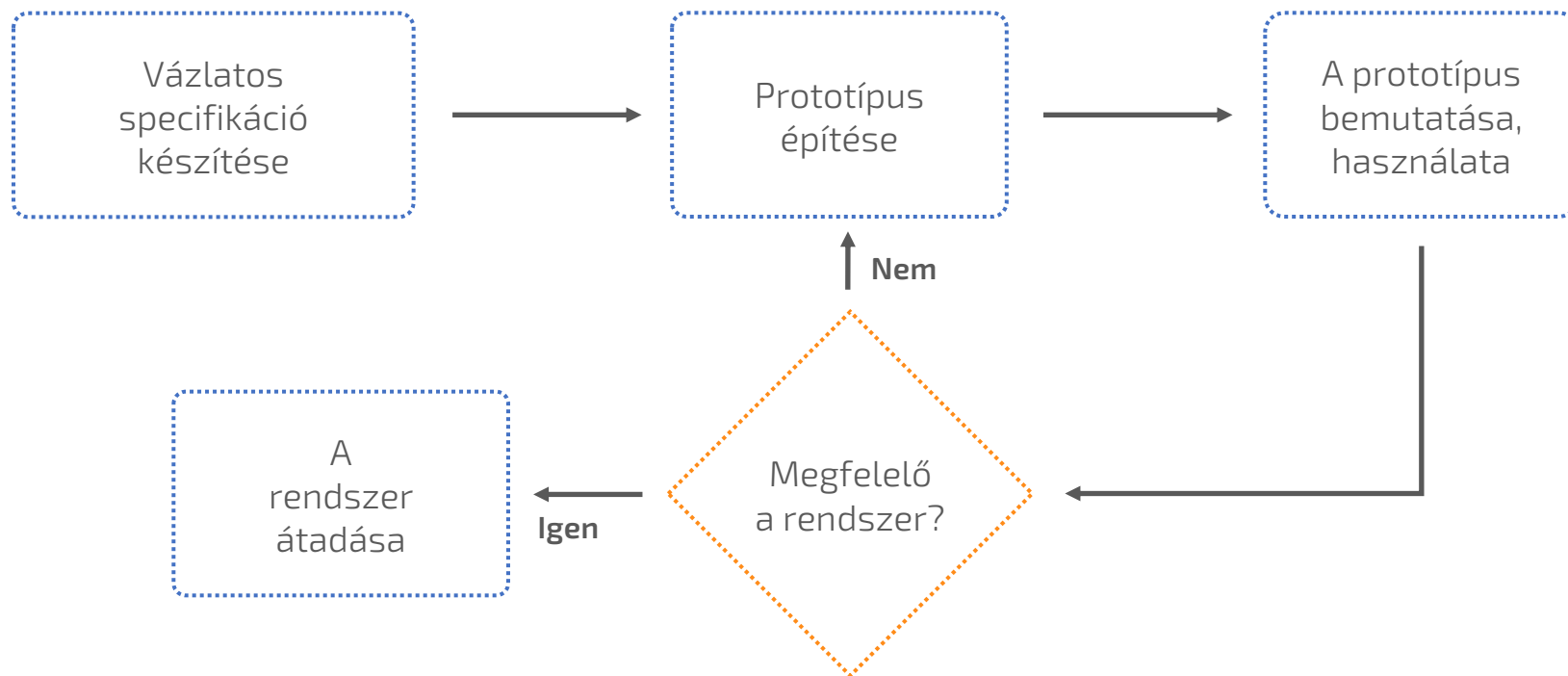
Evolúciós prototípus készítése

- Olyan rendszereknél célszerű alkalmazni, ahol nem készíthető el előre a végleges specifikáció. Ilyenek általában az intenzív felhasználói interfész-használatot igénylő rendszerek.
- Nincs részletes rendszerspecifikáció, sokszor a részletes követelménydokumentum is hiányzik.
- A felhasználó már a rendszer fejlesztése közben jelezheti, hogy milyen irányban kívánja folytatni a fejlesztést.
- A fejlesztéshez gyors, iterálható fejlesztői eszközökre és módszerekre van szükség.
- Mivel nem készül követelményspecifikáció, a validáció is csak a rendszer *(vagyis a prototípus)* bemutatásával történhet.



Evolúciós prototípus készítése

Az evolúciós prototípuskészítés folyamata



Evolúciós prototípus készítése

Az evolúciós prototípuskészítés jellemzői

- A specifikáció, a tervezés és az implementáció részben átlapolható.
- A rendszer **inkrementumok sorozataként** fejlődik (ld. **Agilis módszerek**), és kerül a felhasználóhoz, vagyis a felhasználó kulcsfigurái minden inkrementum tervezésében és értékelésében részt vesznek.
- Gyors fejlesztői eszközök és technikák alkalmazhatók (CASE eszközök, 4GL, folyamatmodellező nyelvek: BPMN-Business Process Modeling Notation, WebServices).
- A felhasználói felületek GUI fejlesztői eszközökkel készíthetők.



Evolúciós prototípus készítése

Az evolúciós prototípuskészítés előnyei



Felgyorsul a rendszerfejlesztés

A gyors fejlesztés, az új rendszer sürgős használatbavétele gyakran fontosabb, mint a követelmények részletes, következetes feltárása vagy a hosszú távú karbantarthatóság.



Növelhető a felhasználó elkötelezettsége

A felhasználók bevonása a rendszerfolyamatba azt eredményezi, hogy a rendszer nagyobb valószínűséggel felel meg az elvárásoknak és a használatba vételkor a felhasználók már ismerik azt és tudják alkalmazni.



Evolúciós prototípus készítése

Az evolúciós prototípuskészítés hátrányai

Vezetési problémák

A hagyományos vezetési módszerek a vízesés modellre alkalmazhatók. Az új technológiák alkalmazásához speciális ismeretekre, esetleg más munkatársakra van szükség.

Karbantartási problémák

A folytonos változások a prototípus szerkezetének sérülését okozhatják, a dokumentáció hiánya és a speciális fejlesztő eszközök a karbantartást veszélyeztetik.

Szerződéskötési problémák

A fix áras szerződéshez előre ismerni kell a rendszer vázlatos követelményeit és tervét. A ráfordításalapú szerződést pedig a megrendelő általában nem fogadja el.



Evolúciós prototípus készítése

A prototípus mint specifikáció

- Egyes követelményeket (*mint pl. a biztonságkritikus funkciókat*) nem lehet a prototípusba beépíteni, így nem fognak szerepelni a specifikációban.
- Egy implementáció nem lehet egy szerződés jogi melléklete, ráadásul menet közben folyton változik.
- A nem-funkcionális követelmények nem tesztelhetők teljes mértékben.



Evolúciós prototípus készítése

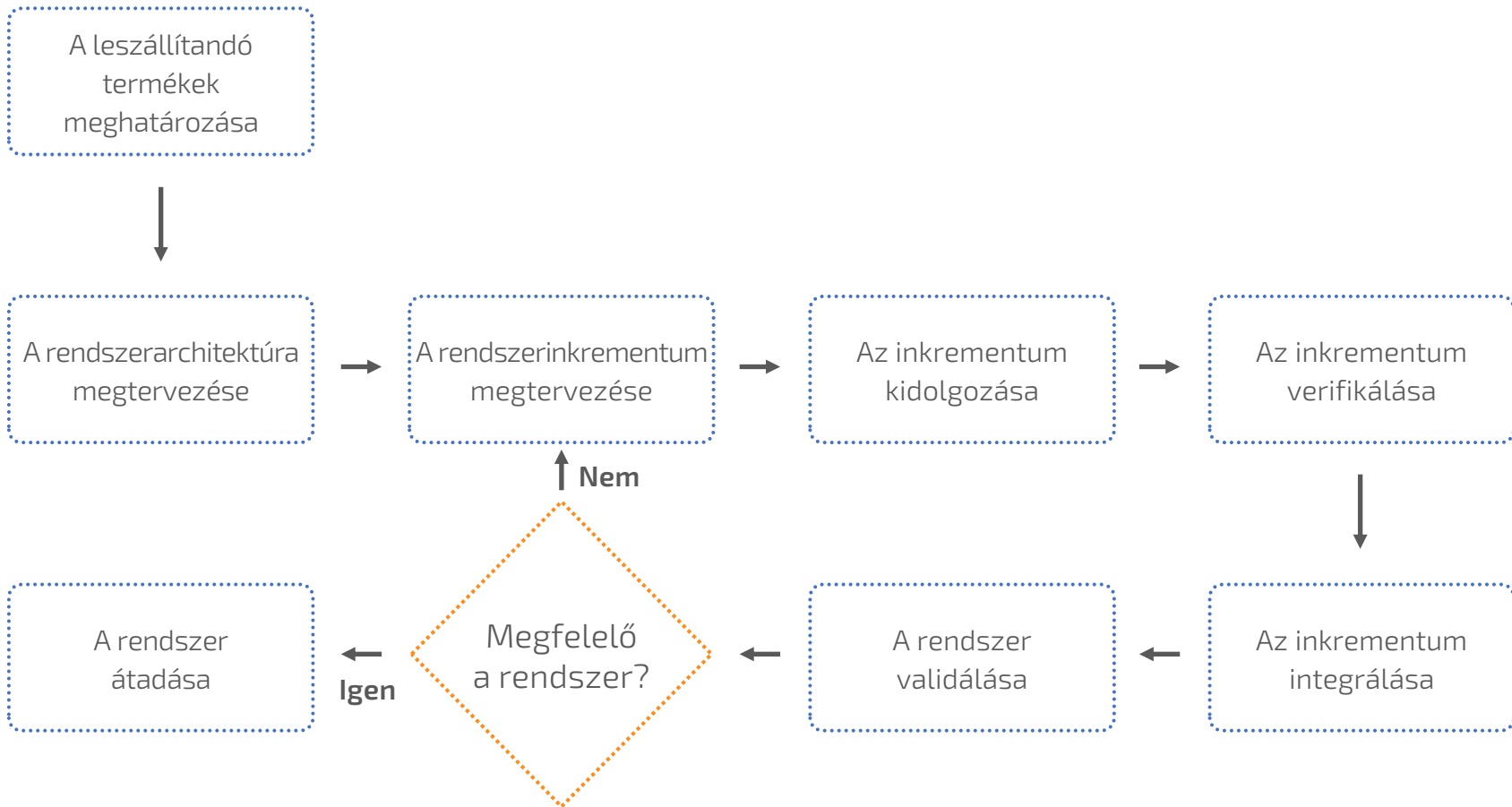
Inkrementális fejlesztés

- Az átfogó architektúra és rendszerkörnyezet megteremtése után a rendszer inkrementumokban készül és jut el a felhasználókhoz.
- Minden inkrementumhoz készülhet specifikáció és dokumentáció.
- A leszállított inkrementumokat a felhasználók tanulmányozhatják, így azok prototípusként használhatók.
- Az inkrementális fejlesztés tartalmazza a prototípuskészítés sok előnyét, miközben a folyamat jobban vezethető és a rendszer struktúrája is kézbentartható.
- Az agilis módszerek (*agile methods*) alapja.



Evolúciós prototípus készítése

Az inkrementális fejlesztési folyamat



Eldobható prototípus készítése



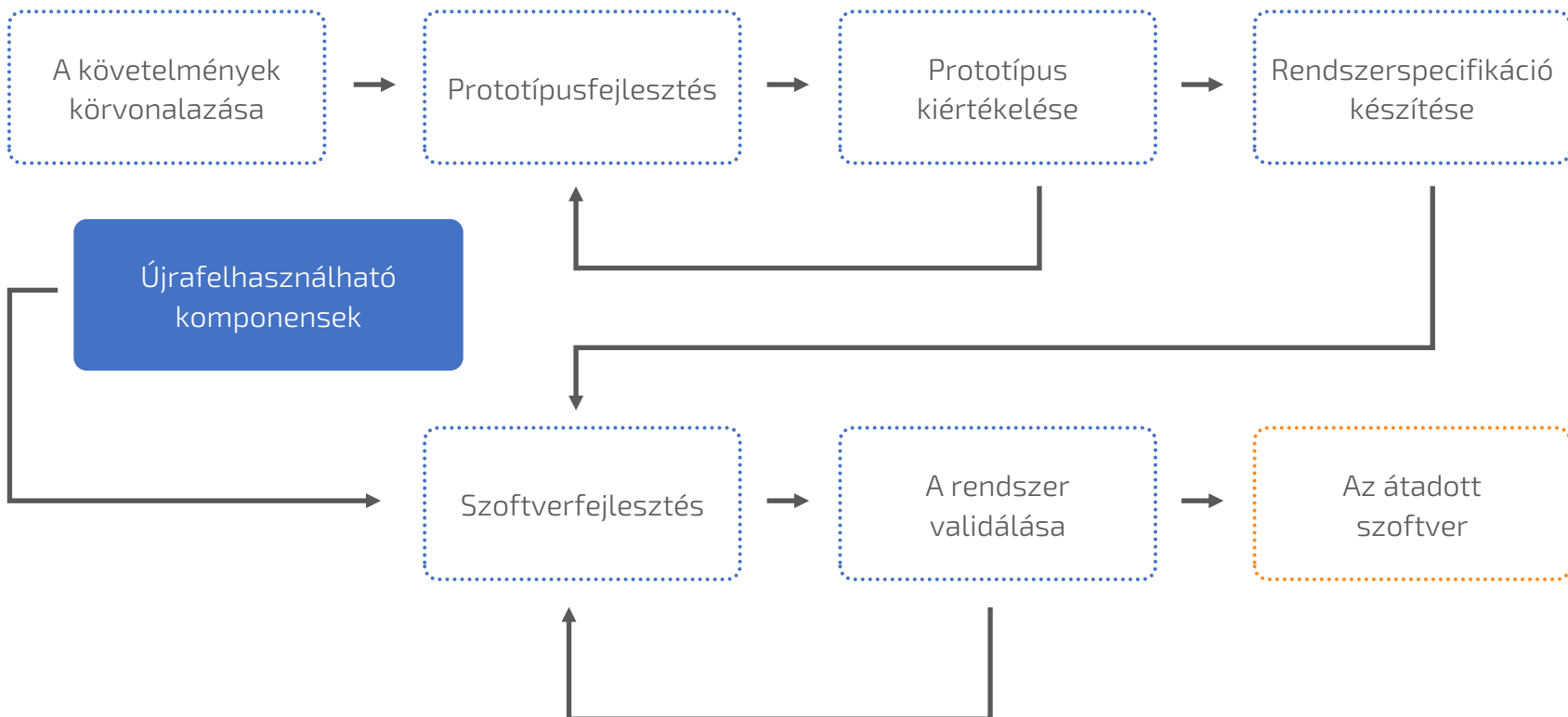
Eldobható prototípus készítése

- Célja a követelményspecifikációból fakadó kockázat csökkentése.
- A prototípust egy kezdeti specifikáció alapján készítik, validálásra átadják a felhasználónak, majd eldobják.
- Az eldobható prototípus nem tekinthető végleges rendszernek, mert:
 - A kivételek, hibák kezelése általában hiányzik.
 - Több rendszertulajdonság kimaradhat a prototípusból.
 - Nem készül specifikáció a hosszú távú karbantartásra.
 - A prototípus még nem a megfelelő struktúra szerint épül.



Eldobható prototípus készítése

Az eldobható prototípuskészítés folyamata



Eldobható prototípus készítése

A prototípus átadása

- A vezetők gyakran nyomást gyakorolnak a fejlesztőkre, hogy egy működő *eldobható prototípust végleges rendszerként* adjanak át.
- Ez nagyon veszélyes, mert:
 - Az eldobható prototípus nem alakítható úgy, hogy a nem-funkcionális követelményeknek (*teljesítmény, megbízhatóság, skálázhatóság, stb.*) eleget tegyen.
 - A prototípus rendszerint dokumentálatlan marad, mert a cél a gyors elkészítés és bemutatás.
 - A változtatások miatt a rendszer struktúrája általában romlik a fejlesztés során.
 - A prototípus készítésekor az általános szervezeti szabványokat nem tartják be (*minőségbiztosítás, technológiai fegyelem, projektdokumentálás*).



Tartalom

1

A PROTOTÍPUSKÉSZÍTÉS SZEREPE

2

A PROTOTÍPUSKÉSZÍTÉS MÓDJAI

3

GYORS PROTOTÍPUSKÉSZÍTÉSI TECHNIKÁK

4

TERVEZÉS ÚJRAFELHASZNÁLÁSSAL



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Gyors prototípuskészítési technikák

- A gyors prototípuskészítéshez az alábbi technikák alkalmazhatók:
 - Fejlesztés dinamikus, magas szintű nyelven,
 - Adatbázis-programozás,
 - Grafikus modellek és fejlesztő eszközök,
 - Komponensek és alkalmazások összeépítése.
- A gyakorlatban ezeket együttesen alkalmazzák.
- A legtöbb prototípuskészítő eszköz tartalmazza a *vizuális programozás támogatását*, ahol grafikus szimbólumok reprezentálják a függvényeket, adatokat, feldolgozószkripteket, akár összetett üzleti funkciókat. Az eszköz a rendszer vizuális reprezentációjából generálja a végrehajtható programot.



Magas szintű nyelvek

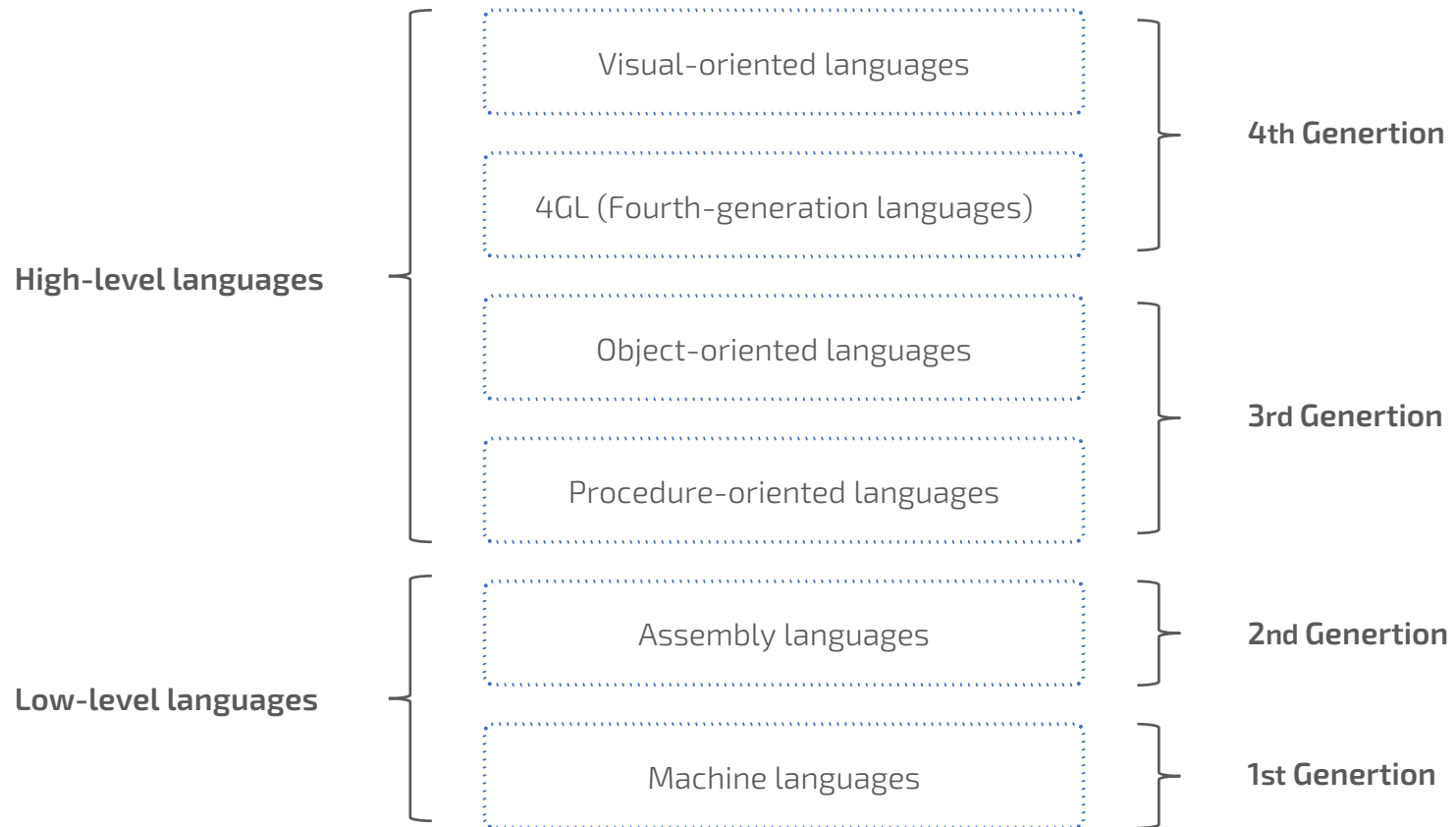


PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY
INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR



Magas szintű nyelvek

A programozási nyelvek fejlődése



Magas szintű nyelvek

Fejlesztés magas szintű nyelven

- Olyan nyelvek, amelyek hatékony futási idejű adatkezelő eszközöket tartalmaznak.
- Korábban a nagy rendszerek fejlesztéséhez nem használtak ilyeneket, mert nagy teljesítményű futtató rendszereket igényelnek, ami növeli a tárigényt, csökkenti a futási teljesítményt.
- Némely nyelv olyan integrált támogató környezetet tartalmaz, amely felhasználható a gyors prototípuskészítéshez.
- A magas szintű nyelvek többsége fejlett felhasználói interfész-fejlesztő képességekkel rendelkezik.



Magas szintű nyelvek

3.-4. generációs nyelvek

- 3GL
 - C, C++, C#, Java
 - egy 3GL utasítás kb. 5-10 assembly utasítás
- 3GL és 4GL között
 - Python, Ruby
- 4GL
 - ORACLE ADF, LabVIEW, ORACLE Reports, SQL
 - Számos 3GL nyelvhez van kiegészítő *library* 4GL szintű funkcionalitással
 - Egy 4GL utasítás kb. 30-40 assembly utasítás



Magas szintű nyelvek

4GL jellemzői

- Tartalmaz:
 - GUI tervezés
 - Vizuális fejlesztőkörnyezet
 - A felhasználó is megértheti, sőt programozhat is
 - A programkészítés termelékenységét növeli (van ellenkező tapasztalat is)
- De:
 - Több erőforrást igényel
 - Nehezen menedzselhető
 - Szegényes a tervezési módszertani támogatás
 - Több száz 4GL nyelv létezik, nem mindegyik jó mindenre
- James Martin: prototípuskészítésre, iteratív tervezésre, adatkezelésre használható.



Magas szintű nyelvek

4GL fejlesztői környezetek

- A nagy adatbázisok és alkalmazási rendszerek szállítói kifejlesztették 4GL fejlesztői környezeteket is.
- Ezek grafikus segítséget nyújtanak:
 - nemcsak az adatbázis tervezéséhez, kezeléséhez
 - az alkalmazás felhasználói felületeinek tervezéséhez,
 - report készítéséhez, stb.
- Továbbá támogatják:
 - az üzleti folyamatok tervezését,
 - az alkalmazásintegrációt
(régir rendszerekkel, partnerek rendszereivel)
 - a vezetéstámogató eszközöket
(Business Intelligence Beans)



Tartalom

1

A PROTOTÍPUSKÉSZÍTÉS SZEREPE

2

A PROTOTÍPUSKÉSZÍTÉS MÓDJAI

3

GYORS PROTOTÍPUSKÉSZÍTÉSI TECHNIKÁK

4

TERVEZÉS ÚJRAFELHASZNÁLÁSSAL



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Tervezés újrafelhasználással

A szoftver újrafelhasználása

- A legtöbb mérnöki tervezési tevékenység komponensek újrafelhasználásán alapul. A terveket más rendszerekben már kipróbált, szabványos, kisebb-nagyobb komponens újrafelhasználására alapozzák *(csavaroktól a hajtóművekig)*.
- A szoftverfejlesztés hagyományosan az eredeti fejlesztésen alapul, de a minőség javítása, a költségek és a fejlesztési idő csökkentése érdekében mindinkább előtérbe kerül a szoftver komponensek újrafelhasználása.
- Ehhez olyan tervezési módszereket kell alkalmazni, amely a szisztematikus újrafelhasználáson alapul.



Tervezés újrafelhasználással

Újrafelhasználáson alapuló szoftverfejlesztés

- **Alkalmazási rendszerek újrafelhasználása**

- Teljes alkalmazási rendszerek újrafelhasználása:
- Beépítve más rendszerekbe, vagy
- Speciális felhasználói igényeket kiszolgáló alkalmazáscsaládok kifejlesztése.

- **Komponensek újrafelhasználása**

- Különböző méretű (*objektum – alrendszer*) komponensek beépítése új rendszerekbe.
(*pl. driverek, interfészmodulok, stb.*)

- **Függvények újrafelhasználása**

- Egyszerű, jól definiált tevékenységet végző komponensek újrafelhasználása.
(*pl. szabványos könyvtárak*)



Tervezés újrafelhasználással

Az újrafelhasználás előnyei

- ✓ **Javuló megbízhatóság**
A komponenseket már több működő rendszerben kipróbálták.
- ✓ **Alacsonyabb projektkockázat**
A komponensek ára és adaptálási költsége pontosabban tervezhető.
- ✓ **A szaktudás jobb kihasználása**
A speciális szaktudás a komponensben testesül meg, nem szükséges minden projekthez külön alkalmazni.
- ✓ **Szabványosság**
A szabványoknak való megfelelést a komponensek garantálják (*interfészek, kommunikációs és GUI szabványok*)
- ✓ **Gyorsabb fejlesztés**
Egy rendszer kifejlesztése gyorsabb, ha kevesebb eredeti fejlesztést igényel.

Tervezés újrafelhasználással

Az újrafelhasználás hátrányai



Növekvő karbantartási költségek

A komponens forráskódja és tervezési dokumentációja hiányában növekszik a karbantartás költsége.



Az eszköztámogatás hiánya

A CASE eszközök gyakran nem támogatják az újrafelhasználást.



A „nem mi találtuk ki” jelenség

Egy teljes rendszer kidolgozása nagyobb szakmai kihívás.



A komponenskönyvtárak karbantartása

Sokba kerül a komponenskönyvtárak feltöltése és folyamatos karbantartása.



Az újrafelhasználható komponensek megtalálása és adaptálása

Még nem fejlődtek ki a komponensek megtalálását és adaptálását segítő általános technikák.

Tervezés újrafelhasználással

Kritikus követelmények

- Meg kell találni a megfelelő újrafelhasználható komponenseket. Ehhez katalógusokra és nyilvántartásokra, alkalmas keresőmechanizmusokra van szükség.
- Az újrafelhasználónak bíznia kell abban, hogy a komponens a leírtaknak megfelelően és megbízhatóan működik.
- A komponenseknek olyan dokumentációval kell rendelkezniük, amely érthető, teljes, aktuális és hivatkozik a korábbi felhasználásokra is (*referenciák*).



Újrafelhasználás programgenerátorral



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY
INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Újrafelhasználás programgenerátorral

- A generátoralapú újrafelhasználás akkor lehetséges, ha egy programgenerátor tartalmazza egy szakterület alapvető ismeretanyagát. *(pl. adatfeldolgozás)*
- Az ilyen programgenerátorok tartalmazzák a szabványos algoritmusokat és függvényeket, és ezek paraméterezését követően a generátor automatikusan előállítja a programot.
- A szakterületre kidolgozott nyelven vagy újabban grafikus eszközökkel lehet elkészíteni a rendszer modelljét.

(Ebben az esetben elsősorban a szakterületi tudás újrafelhasználásáról van szó.)



Újrafelhasználás programgenerátorral

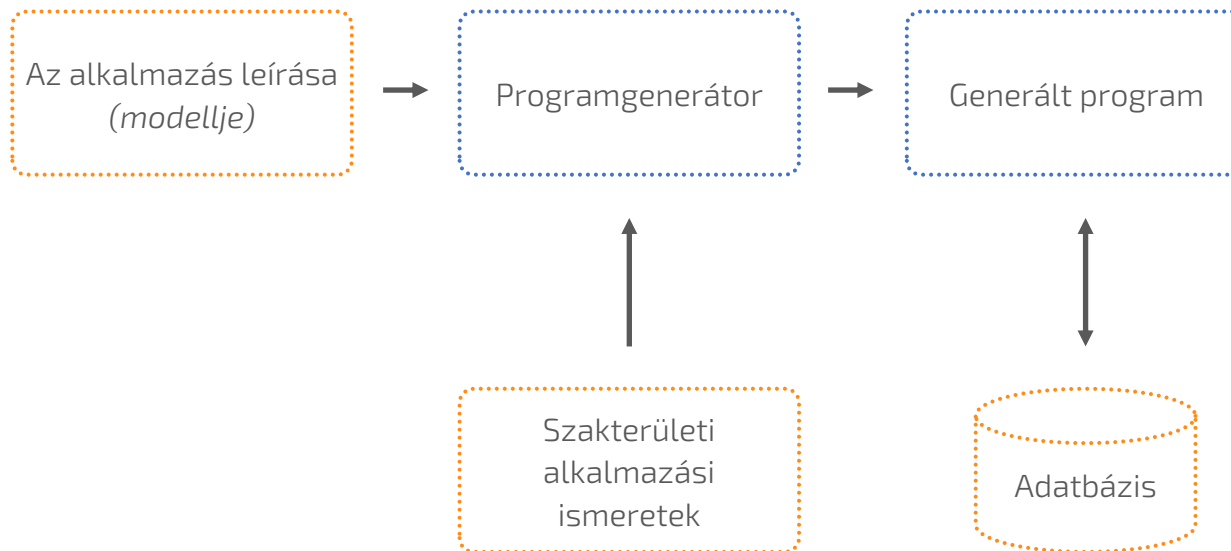
A programgenerátorok típusai

- A generátorok fajtái:
 - **Alkalmazásgenerátorok** - üzleti adatfeldolgozó rendszerek készítésére.
 - **Szintaktikus elemzők** - a programozási nyelvek értelmezésére.
 - **CASE eszközökben lévő kódgenerátorok** - egy szoftvertervből a tervezett rendszer implementációját állítják elő.
- A generátoralapú újrafelhasználás igen költséghatékony, de viszonylag kevés szakterülethez léteznek ilyen rendszerek (*üzleti adatfeldolgozás, eBusiness, stb.*)
- Ezzel a módszerrel könnyebben állíthatók elő az alkalmazások, mint a komponensalapú módszerrel.



Újrafelhasználás programgenerátorral

A generátor alapú újrafelhasználás folyamata



Komponensalapú fejlesztés



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Komponensalapú fejlesztés

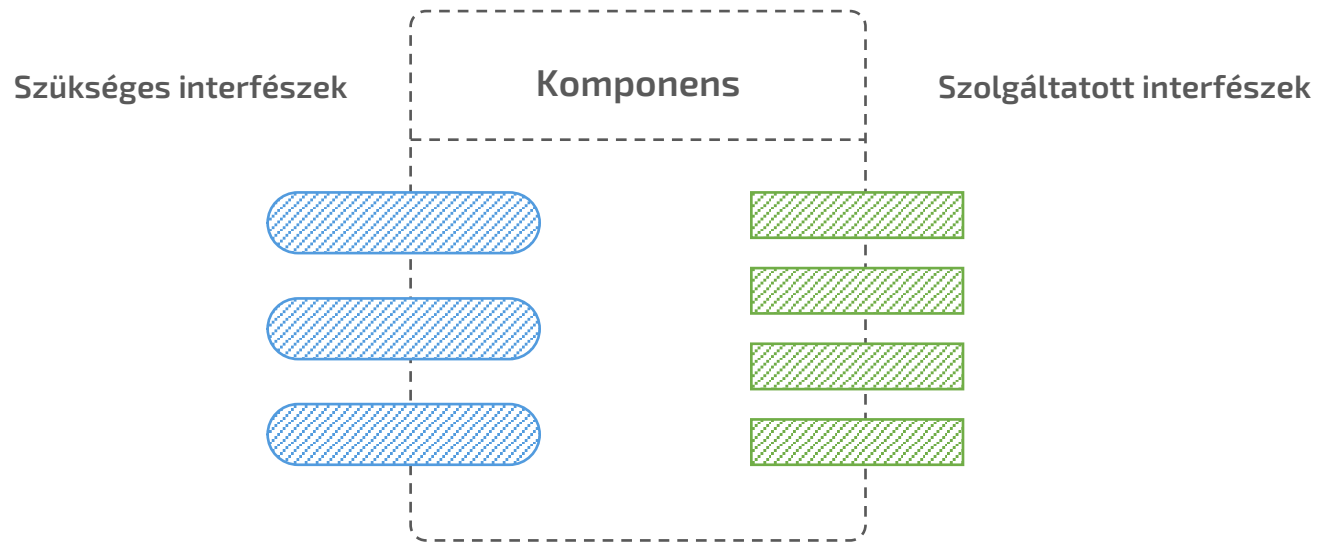
- A komponensalapú szoftverfejlesztés (*CBSE – Component Based Software Engineering*) az újrafelhasználáson alapul.
- Kialakulásának oka az, hogy az objektumorientált fejlesztés nem igazán támogatja az újrafelhasználást, mert:
 - Az egyedi objektumosztályok túl részletesek és specifikusak és csak a folyamat késői fázisában kapcsolódnak az alkalmazáshoz.
 - Nem alakult ki olyan piac, ahol az egyes szakterületek objektumosztályaihoz lehetne hozzájutni.
- A komponensek az objektumosztályoknál sokkal absztraktabbak és különálló szolgáltatásoknak tekinthetők.



Komponensalapú fejlesztés

A komponensek interfészei

- **Szolgáltatott interfészek** – a komponens által szolgáltatott interfészek.
- **Szükséges interfészek** – azok az interfészek, amelyeket a komponenst használó rendszernek vagy környezetének kell biztosítania.



Komponensalapú fejlesztés

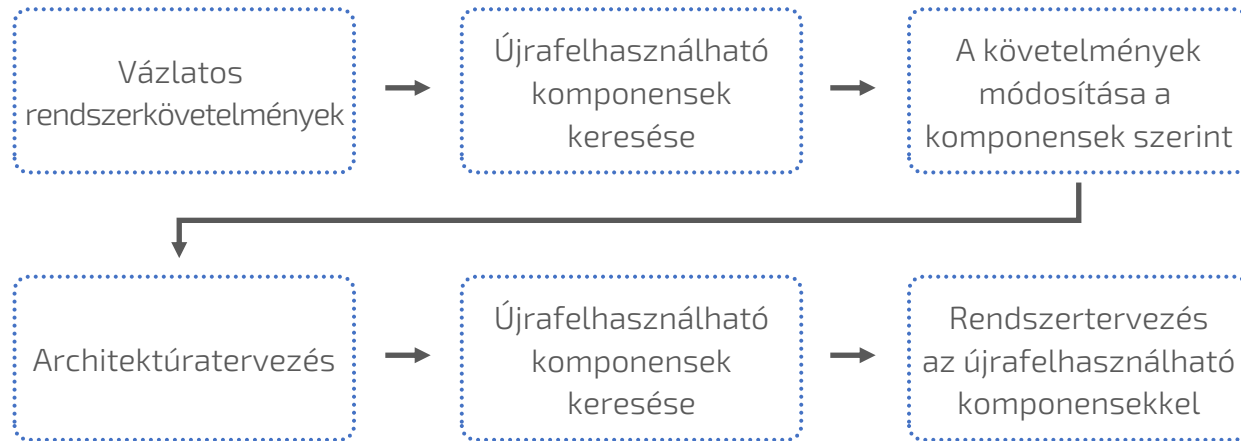
A komponensalapú fejlesztés folyamata

- A komponensalapú fejlesztés beilleszthető a szabályos szoftverfolyamatba, ha beépítjük abba az újrafelhasználással kapcsolatos tevékenységeket:
- Komponensek specifikálása,
- Komponensek megtalálása,
- A tervek *(esetleg a követelmények)* módosítása a meglett komponensek tulajdonságainak megfelelően.
- Ez az **alkalmazkodó újrafelhasználás**



Komponensalapú fejlesztés

Fejlesztés komponensek újrafelhasználásával



- A rendszer megvalósítása történhet prototípuskészítéssel vagy inkrementális módon.
- A legtöbb programozási nyelvben hivatkozhatunk könyvtárban tárolt komponensekre
- Leggyakrabban script nyelvet használnak a komponensek integrálására.



Komponensalapú fejlesztés

Hátrányok, nehézségek

- A komponensek inkompatibilitása miatt a költség- és időmegtakarítás a vártnál kevesebb lehet *(integrációs munkák)*.
- Nehézséget okozhat a komponensek megtalálása *(nincsenek szabványok a komponensek tulajdonságainak leírására, hiányoznak az egységes komponenskönyvtárak)*.
- A követelmények változását követő evolúció lehetetlen, ha a komponensek nem cserélhetők.
- A karbantartás nehezebb.
- Mindezek ellenére *(a fejlesztési idő és a szoftverek élettartamának csökkenése)* sokszor megéri az újrafelhasználható komponensek alkalmazása.



Komponensalapú fejlesztés

Alkalmazás-keretrendszerek



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Komponensalapú fejlesztés

Alkalmazás-keretrendszerek

- A keretrendszer absztrakt és konkrét osztályok gyűjteményéből és a köztük lévő interfészekből álló alrendszerterv.
- A keretrendszerek úgy implementálhatók, hogy a terv részeit komponensek hozzáadásával egészítjük ki.
- Általában viszonylag nagy, újrafelhasználható egységek, de nem önálló alkalmazások.
- Az alkalmazások több keretrendszer integrálásával hozhatók létre.



Komponensalapú fejlesztés

Alkalmazás-keretrendszerek

A KERETRENDSZEREK CSOPORTOSÍTÁSA

- **A rendszer infrastruktúrájának keretrendszerei**
 - A rendszer infrastrukturális alapjainak (*kommunikáció, felhasználói felületek, titkosítás, stb.*) fejlesztését támogatják.
- **Köztes, integrációs keretrendszerek**
 - komponensek közti kommunikációt és információcserét támogató szabványok és osztályok. (*Ilyen például a CORBA, JavaBean, JMI, COM, DCOM, .NET, stb.*)
- **Vállalati alkalmazások keretrendszerei**
 - Az egyes speciális szakterületi alkalmazások fejlesztését támogatják. A szakterületi tudást tartalmazzák (*pl. pénzügy, telekommunikáció*).



Komponensalapú fejlesztés

Alkalmazás-keretrendszer

A KERETRENDSZEREK KIBŐVÍTÉSE

- A keretrendszer általános struktúra, amely a konkrét alkalmazás létrehozásakor konkrét osztályokkal kibővíthető.
- A keretrendszer kibővítése az alábbiakat jelenti:
 - A keretrendszer absztrakt osztályainak kiegészítése konkrét osztályokkal.
 - Műveletek hozzáadása, amelyek meghívhatók a keretrendszer által kezelt események bekövetkezésekor.
- A keretrendszerek hátránya a bonyolultság. Sok időt igényel az effektív használatukhoz szükséges megismerésük.



Komponensalapú fejlesztés „Polcról levehető” termékek



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Komponensalapú fejlesztés

„Polcról levehető” termékek

- A „polcról levehető”, COTS – Commercial Off-The-Shelf rendszerek általában komplett alkalmazási rendszerek, amelyek API-val rendelkeznek.
- Legtöbbször rendszerszoftvertermékek az egyszerű komponenseknél nagyobb funkcionalitással.
- Nagy rendszerek építésekor gyakran használt stratégia a COTS termékek integrálása, különösen a gyors fejlesztést kívánó eCommerce, eBusiness rendszerek körében. A fejlesztési idő nagyságrendekkel csökkenthető.




Komponensalapú fejlesztés

„Polcról levehető” termékek

COTS INTEGRÁCIÓS NEHÉZSÉGEK

- A funkcionalitás és a teljesítmény nem tartható kézben:
 - A COTS rendszerek sokszor kevésbé effektívek, mint azt a reklámokban ígérik.
- A COTS rendszerek együttműködése bizonytalan
 - A különböző COTS rendszerek eltérő feltételezésekkel készültek (*pl. sorkezelés*), ezért az integráció nehéz lehet.
- Az evolúció ellenőrizhetetlen
 - A szállító és nem a felhasználó határozza meg.
- A COTS termékek támogatása
 - A szállító által biztosított támogatás gyakran nem terjed ki a rendszer teljes élettartamára.





Komponensalapú fejlesztés Újrafelhasználásra fejlesztett komponensek



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

Komponensalapú fejlesztés

Újrafelhasználásra fejlesztett komponensek

KOMPONENSEK FEJLESZTÉSE ÚJRAFELHASZNÁLÁSRA

- Az újrafelhasználható komponensek meglévő komponensek általánosításával hozhatók létre. Ehhez nagy tapasztalat kell.
- A komponens újrafelhasználhatóságának jellemzői:
 - Stabil szakterületi absztrakciókra támaszkodik.
 - El kell rejtenie az állapotait és műveleteket kell biztosítani a az állapotához való hozzáférésre.
 - Amennyire lehetséges, függetlennek és önállóan kell lennie.
 - A hibakezelést interfészekon keresztül kell megoldania.
- Az újrafelhasználhatóság és a használhatóság ellentmondása:
 - Minél általánosabb interfésszel rendelkezik, annál inkább újrafelhasználható, de annál bonyolultabb, vagyis kevésbé használható.



Komponensalapú fejlesztés

Újrafelhasználásra fejlesztett komponensek

ÚJRAFELHASZNÁLHATÓ KOMPONENSEK FEJLESZTÉSE

- Az újrafelhasználható komponensek fejlesztési költségei többszörösen meghaladják az egyszerű, specifikus komponensek költségeit. Ezt egyetlen projekt költségeiből nem lehet fedezni, ezért kialakultak olyan szoftverfejlesztő vállalatok, amelyek specializálódtak az újrafelhasználható komponensek fejlesztésére.
- Az általános komponensek kevésbé effektívek, több erőforrást használnak és végrehajtási idejük hosszabb, mint a specifikus komponenseké.

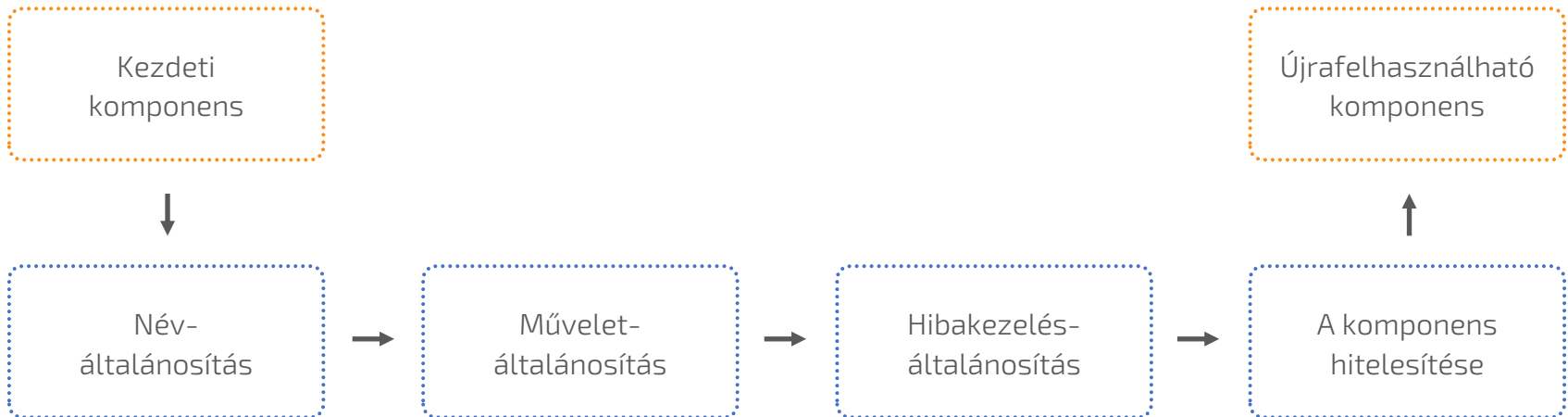


Komponensalapú fejlesztés

Újrafelhasználásra fejlesztett komponensek

SPECIFIKUS KOMPONENS KITERJESZTÉSE

Egy specifikus komponens újrafelhasználhatóvá tétele az alábbi folyamattal végezhető:



Alkalmazáscsaládok



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR

```
139         title={
140             target={
141                 href={trackUrl(url)}
142             }
143         }
144         Instagram
145     </a>
146 </li>
147 </ul>
148 </div>
149 };
150 }
151
152 renderWhatsAppLinks() {
153     return (
154         <div className={styles.whatsappLinks}>
155             <ul className={styles.whatsappLinksList}>
156                 {this.renderWhatsAppLink('Facebook')}
157                 {this.renderWhatsAppLink('Twitter')}
158                 {this.renderWhatsAppLink('LinkedIn')}
159                 {this.renderWhatsAppLink('YouTube')}
160                 {this.renderWhatsAppLink('Instagram')}
161                 {this.renderWhatsAppLink('TikTok')}
162                 {this.renderWhatsAppLink('Pinterest')}
163                 {this.renderWhatsAppLink('Snapchat')}
164                 {this.renderWhatsAppLink('WhatsApp')}
165             </ul>
166         </div>
167     );
168 }
169
170 renderWhatsAppItem(title, url) {
171     return (
172         <li className={styles.whatsappLinkItem}>
173             <a
174                 href={trackUrl(url)}
175                 target="_blank"
176                 rel="noopener noreferrer"
177             >
178                 {title}
179             </a>
180         </li>
181     );
182 }
183
184 renderFooterSub() {
185     return (
186         <div className={styles.footerSub}>
187             <Link to="/" title="Home - Unsplash">
188                 <img alt="Home icon" type="image" className={styles.footerSubLogo} />
189             </Link>
190             <span className={styles.footerSubSlogan}>
191                 Unsplash
192             </span>
193         </div>
194     );
195 }
196
197 render() {
198     return (
199         <div className={styles.footerGlobal}>
200             <div className="container">
201                 {this.renderFooterMain()}
202                 {this.renderFooterSub()}
203             </div>
204         </div>
205     );
206 }
207
208 }
```

Alkalmazáscsaládok

- Az alkalmazáscsalád az alkalmazási rendszerek olyan termékcsaládja vagy termékvonulata, amelyek egy közös, szakterület-specifikus architektúrára épülnek („közös *mag*”).
- Az alkalmazáscsaládnak ezt a közös magját minden esetben újra felhasználják, amikor egy új alkalmazást fejlesztenek ki.
- Mindegyik specifikus alkalmazás különbözik a többitől, miután a közös mag más komponensekkel egészül ki.



Alkalmazáscsaládok

Az alkalmazáscsaládok specializációja

- **Platformspecializáció**

- Az alkalmazás egyes verziói különböző platformokra készülnek
(pl. *Windows, Solaris, Linux, ...*), de a funkcionalitás azonos.

- **Konfigurációs specializáció**

- Az alkalmazás egyes verziói különböző perifériákkal képesek együttműködni.
(*A perifériákat kezelő komponensek különböznek.*)

- **Funkcionális specializáció**

- Az alkalmazás egyes verziói eltérő funkcionális követelmények kielégítésére készülnek.
(*A funkcionális komponensek különböznek.*)



Köszönöm a figyelmet!

Az előadásról



Az előadás a Pázmány Péter Katolikus Egyetem Információs Technológiai és Bionikai Karán meghirdetett A szoftvertechnológia alapjai című tárgy tananyagát mutatja be.

Kada Zsolt a GIRO Zrt. vezérigazgató-helyettese.

Mérnöki képesítéseit a Torinói Műszaki Egyetemen és a Pázmány Péter Katolikus Egyetemen szerezte. Pályafutását Torinóban kutató fejlesztőként kezdte a Telecom Italia és a Politecnico di Torino közös projektjein. A pénzügyi szférában dolgozott mind banki (Erste Bank), mind beszállítói oldalon (IND). A közigazgatásban a Közigazgatási és Elektronikus Közszolgáltatások Központi Hivatalának (KEKKH) IT fejlesztési főosztályát vezette.



Kapcsolódó források



- **Vető István, A szoftvertechnológia alapjai diasor**
- **Ian Sommerville, Szoftverrendszerek fejlesztése**
 - 17. fejezet, Gyors szoftverfejlesztés
 - 18. fejezet, Szoftver-újrafelhasználás



PÁZMÁNY PÉTER KATOLIKUS EGYETEM - KIEMELT FELSŐOKTATÁSI INTÉZMÉNY

INFORMÁCIÓS TECHNOLÓGIAI ÉS BIONIKAI KAR - KUTATÓ KAR