Nagy szoftverinfrastruktúra feletti inkrementális függőségi analízis

Készítette: Csikós Donát

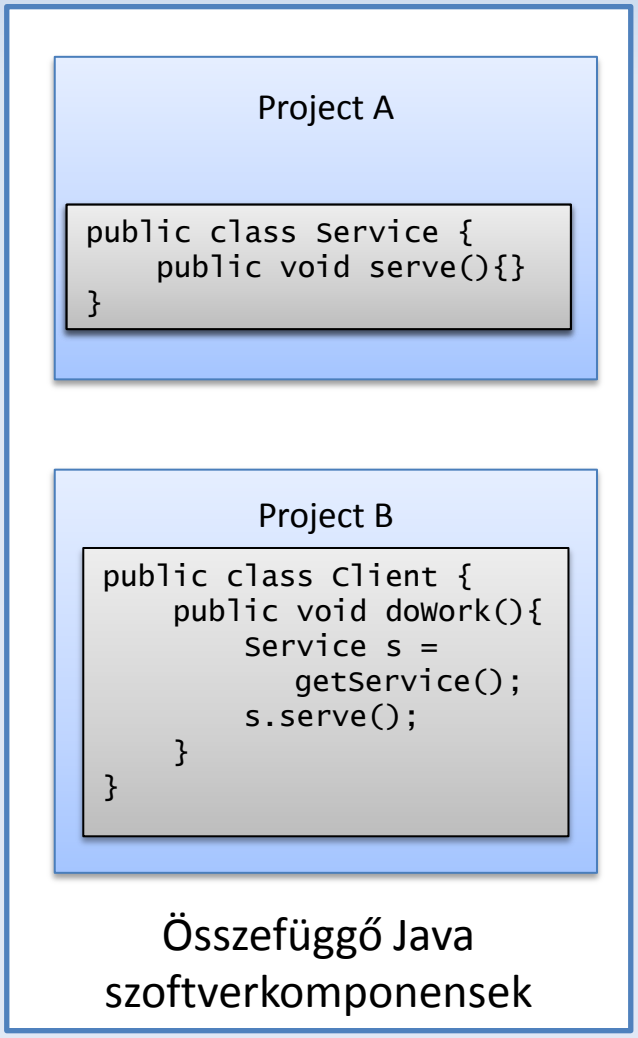
Konzulens: Horváth Ákos, Ráth István

BME MIT

2012.11.13.

Java szoftverek és függőségeik modellezése

Java szoftverek és függőségeik modellezése



Java szoftverek és függőségeik modellezése

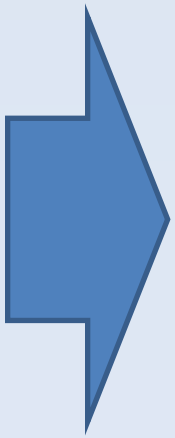
Project A

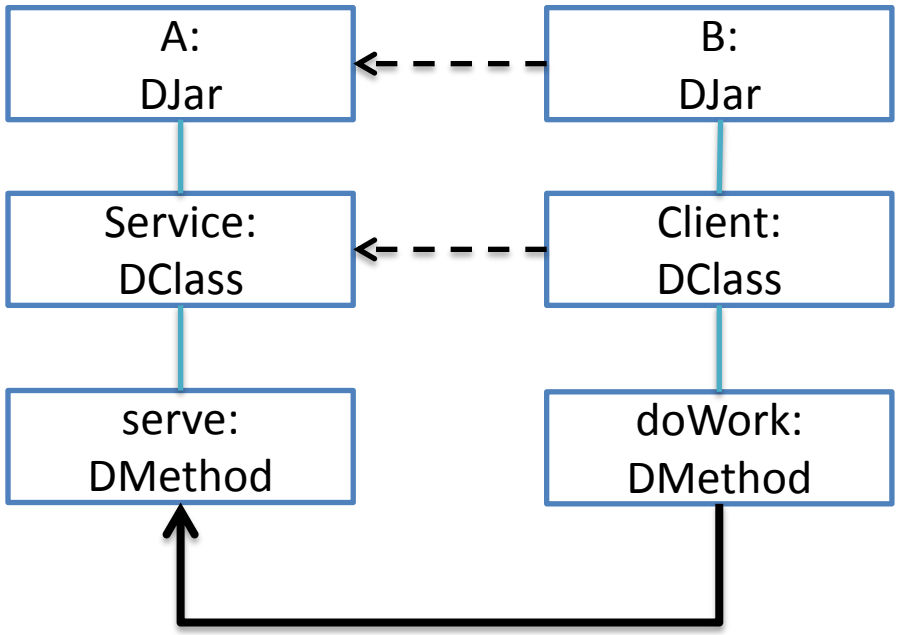
```
public class Service {
    public void serve(){}
}
```

Project B

```
public class Client {
    public void doWork(){
        Service s =
           getService();
        s.serve();
    }
}
```

Összefüggő Java szoftverkomponensek





Java komponensek modell reprezentációja

Java szoftverek és függőségeik modellezése

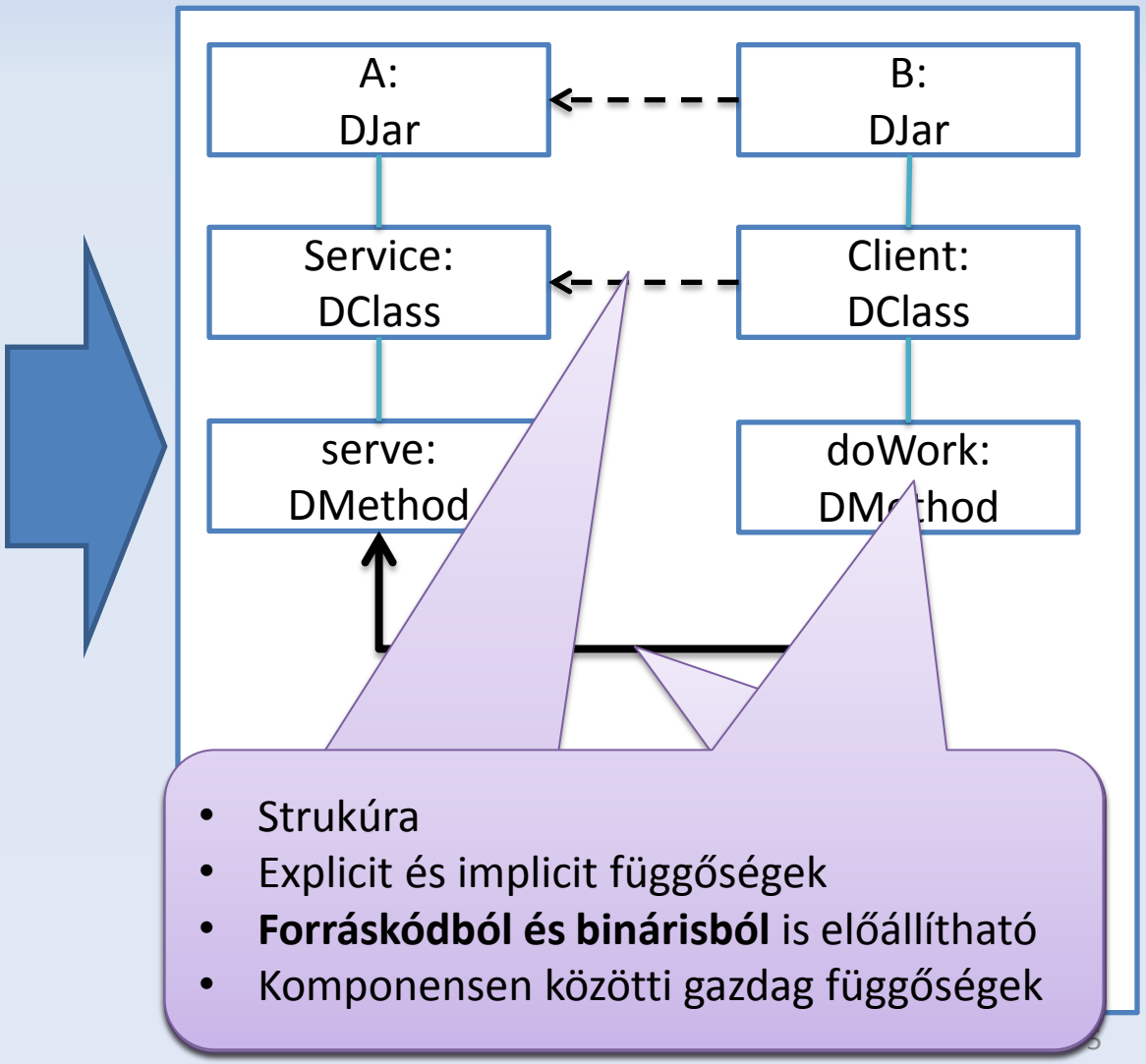
Project A

```
public class Service {
    public void serve(){}
}
```

Project B

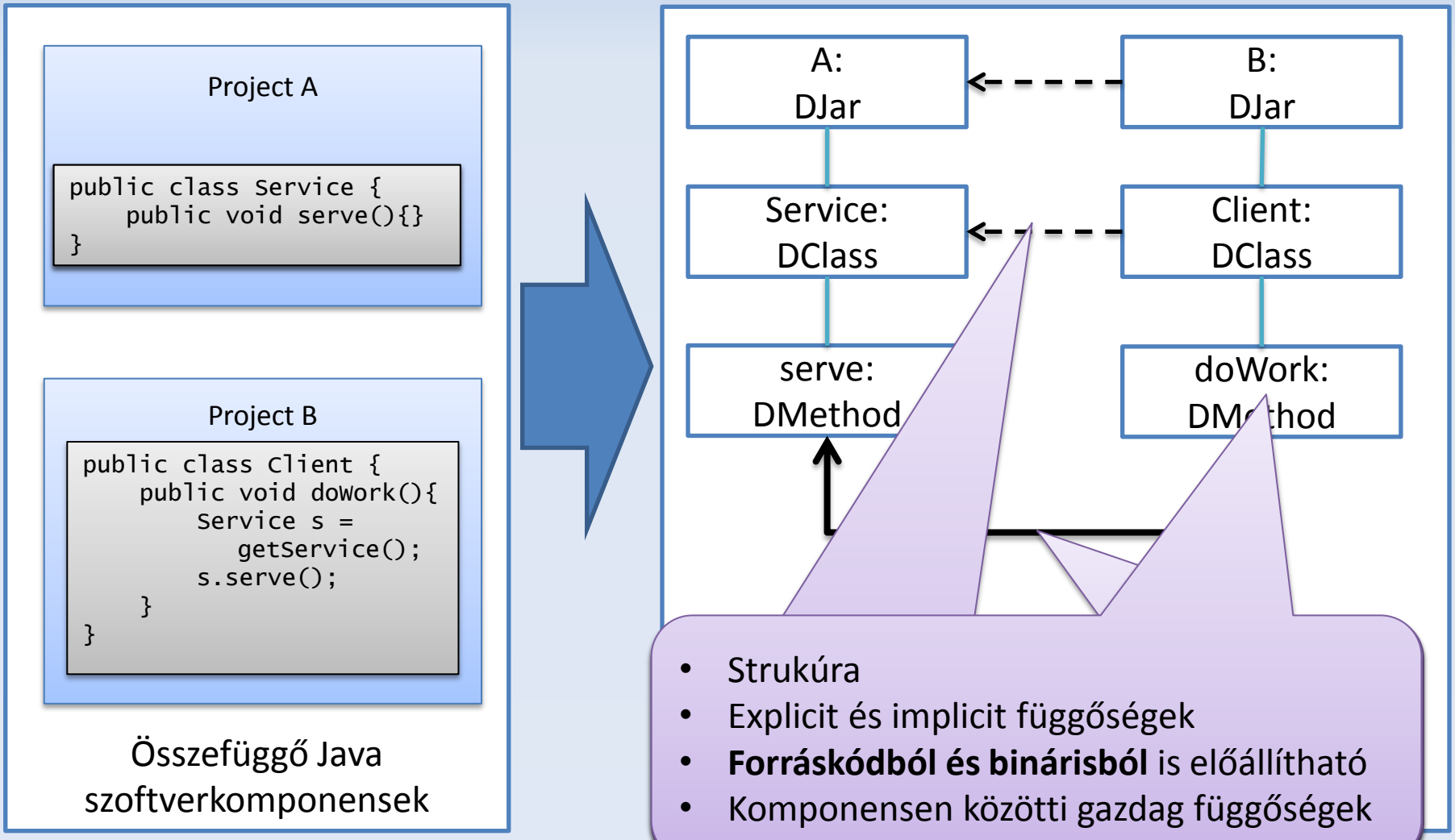
```
public class Client {
    public void doWork(){
        Service s =
           getService();
        s.serve();
    }
}
```

Összefüggő Java szoftverkomponensek

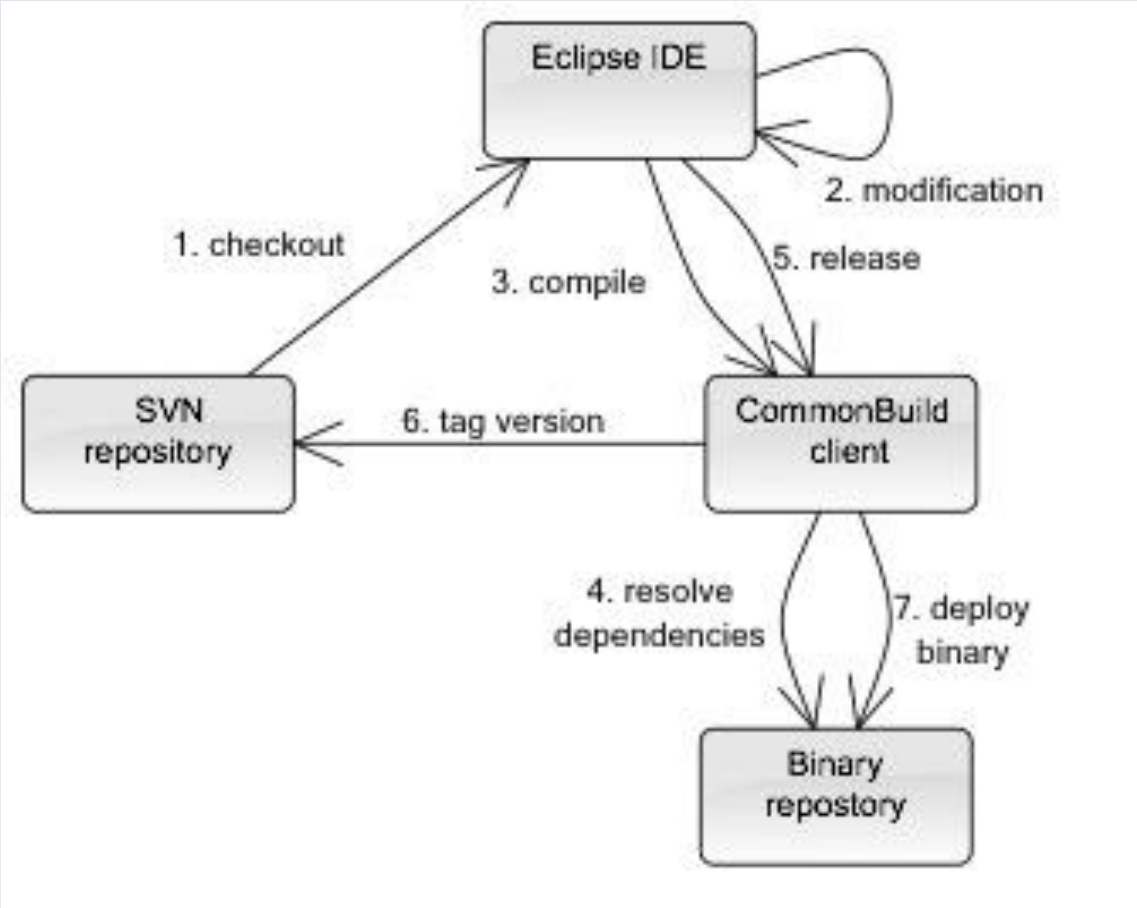


- Strukúra
- Explicit és implicit függőségek
- **Forráskódból és binárisból** is előállítható
- Komponensen közötti gazdag függőségek

Java szoftverek és függőségeik modellezése

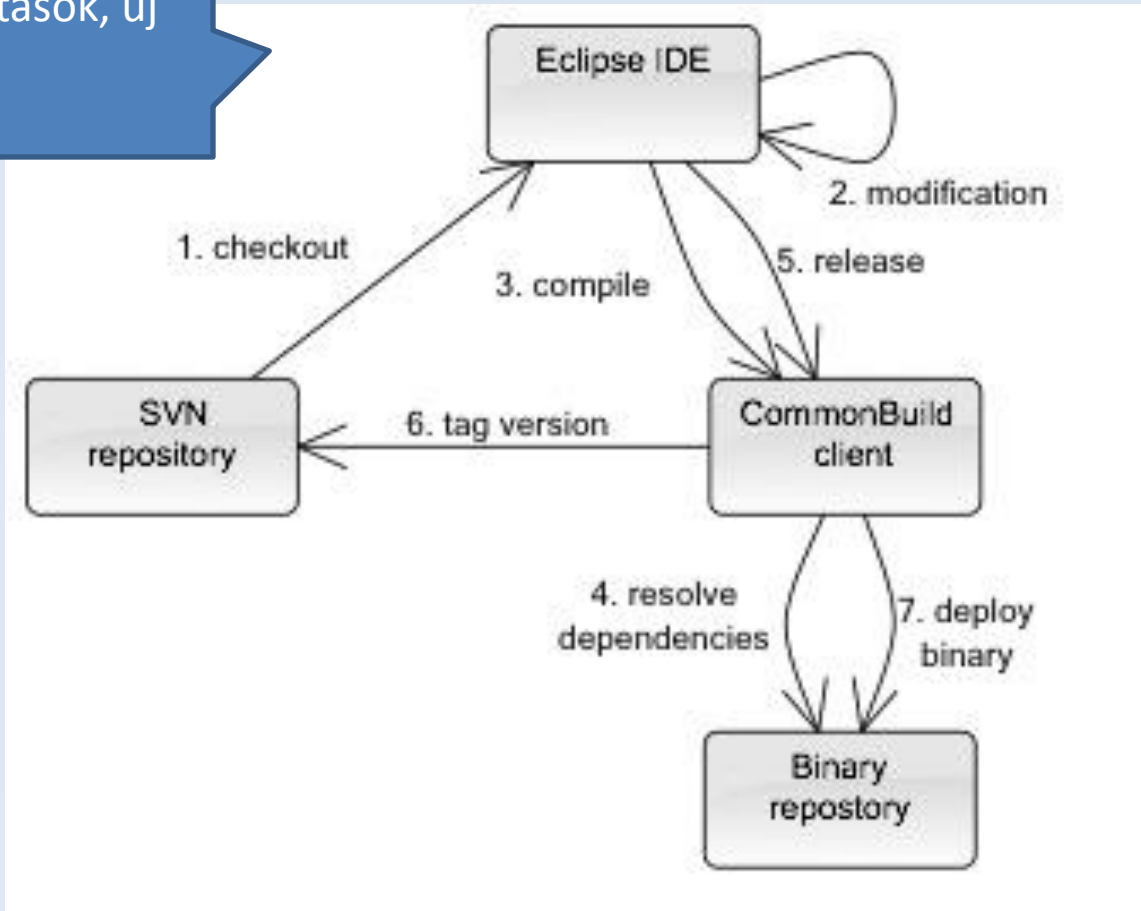


Függőségmenedzsment a gyakorlatban



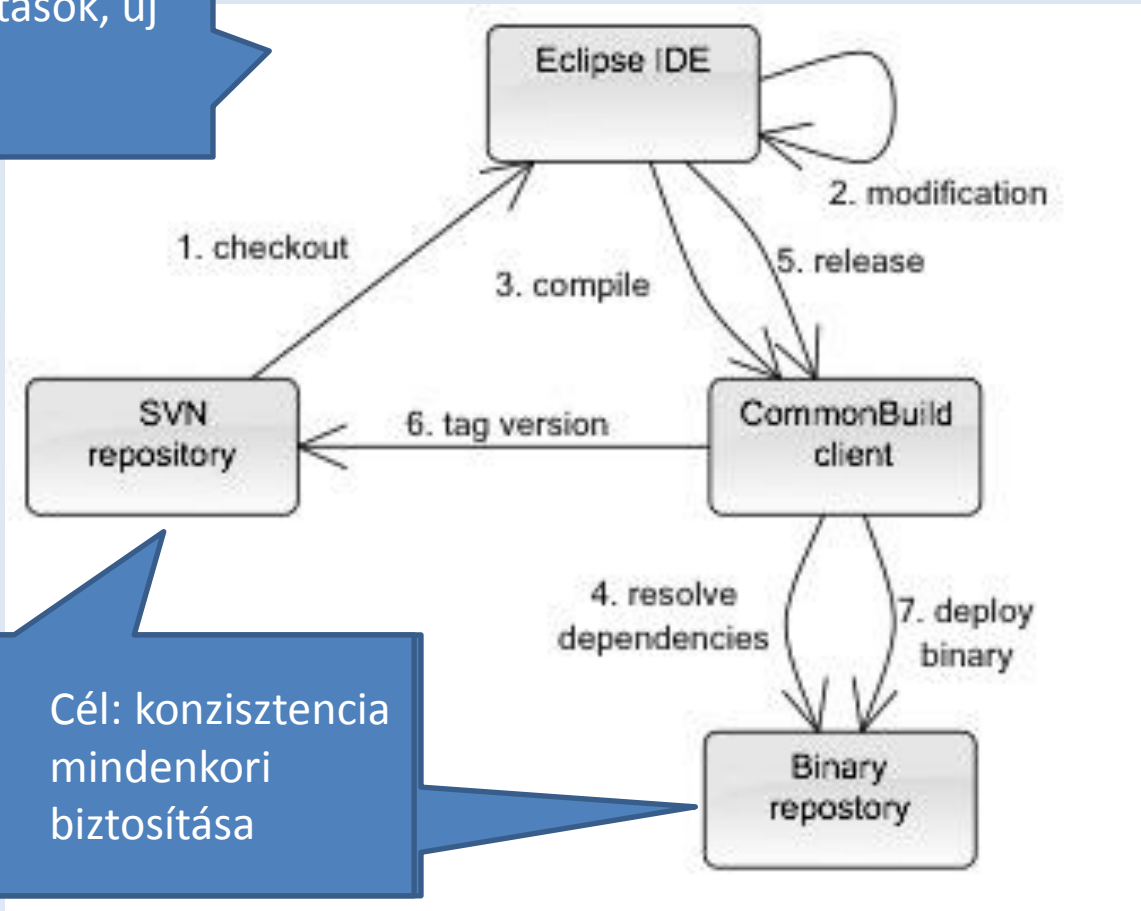
Függőségmenedzsment a gyakorlatban

Szoftver életciklus:
gyakori hibajavítások, új funkciók



Függőségmenedzsment a gyakorlatban

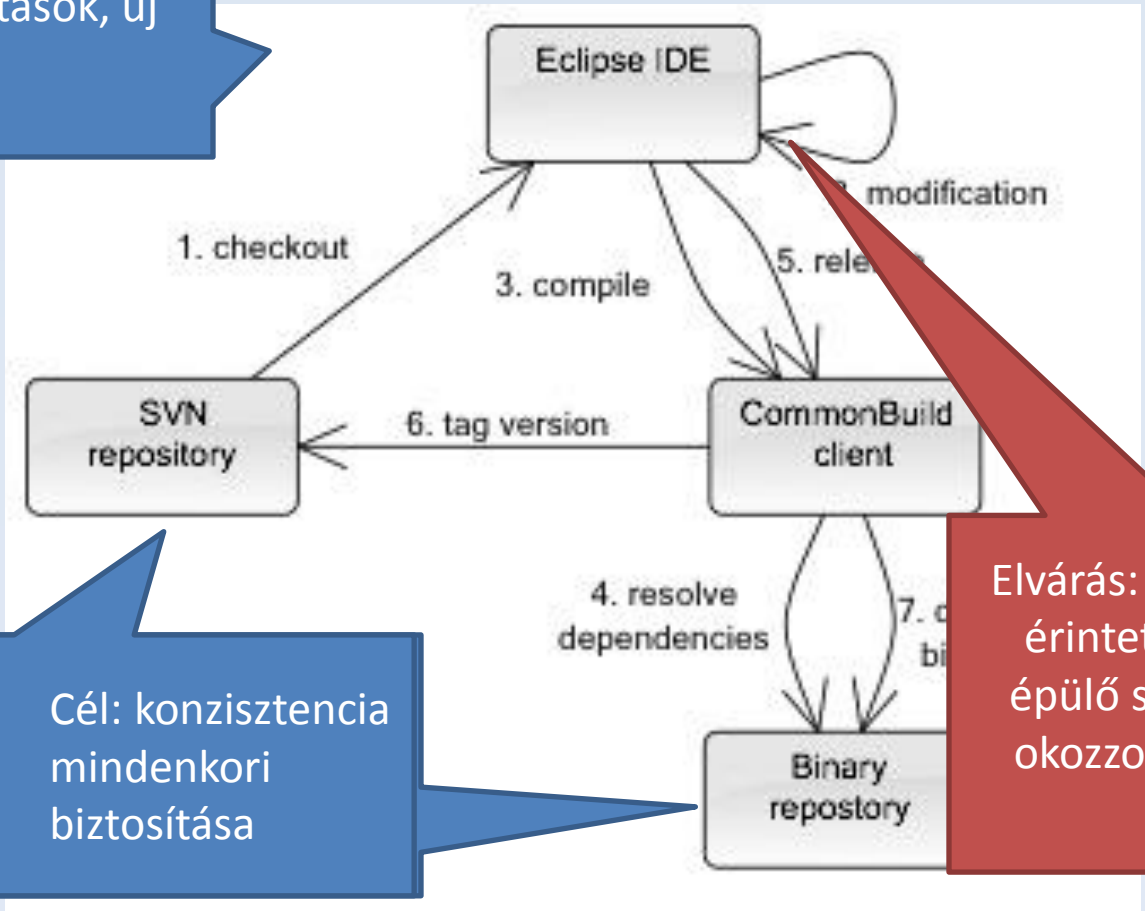
Szoftver életciklus: **gyakori** hibajavítások, új funkciók



Cél: konzisztencia mindenkori biztosítása

Függőségmenedzsment a gyakorlatban

Szoftver életciklus: **gyakori** hibajavítások, új funkciók



Cél: konzisztencia mindenkori biztosítása

Elvárás: egy módosítás az érintett komponensre épülő szoftverekben ne okozzon hibát (**smooth upgrade**)

Függőségmenedzsment a gyakorlatban

Szoftver életciklus:
gyakori hibajav[illegible] új funkciók

Szükséges:
(statikus) függőségi viszonyok ismerete a **teljes** szoftverinfrastruktúrán (komponensek, verziók)

- Kiszámítható a változtatások potenciális hatása
- → *Mit változtathatunk meg és hogyan*

Cél: konzisztencia mindenkori biztosítása

[illegible]egy módosítás az [illegible]t komponensre épülő szoftverekben ne okozzon hibát (**smooth upgrade**)

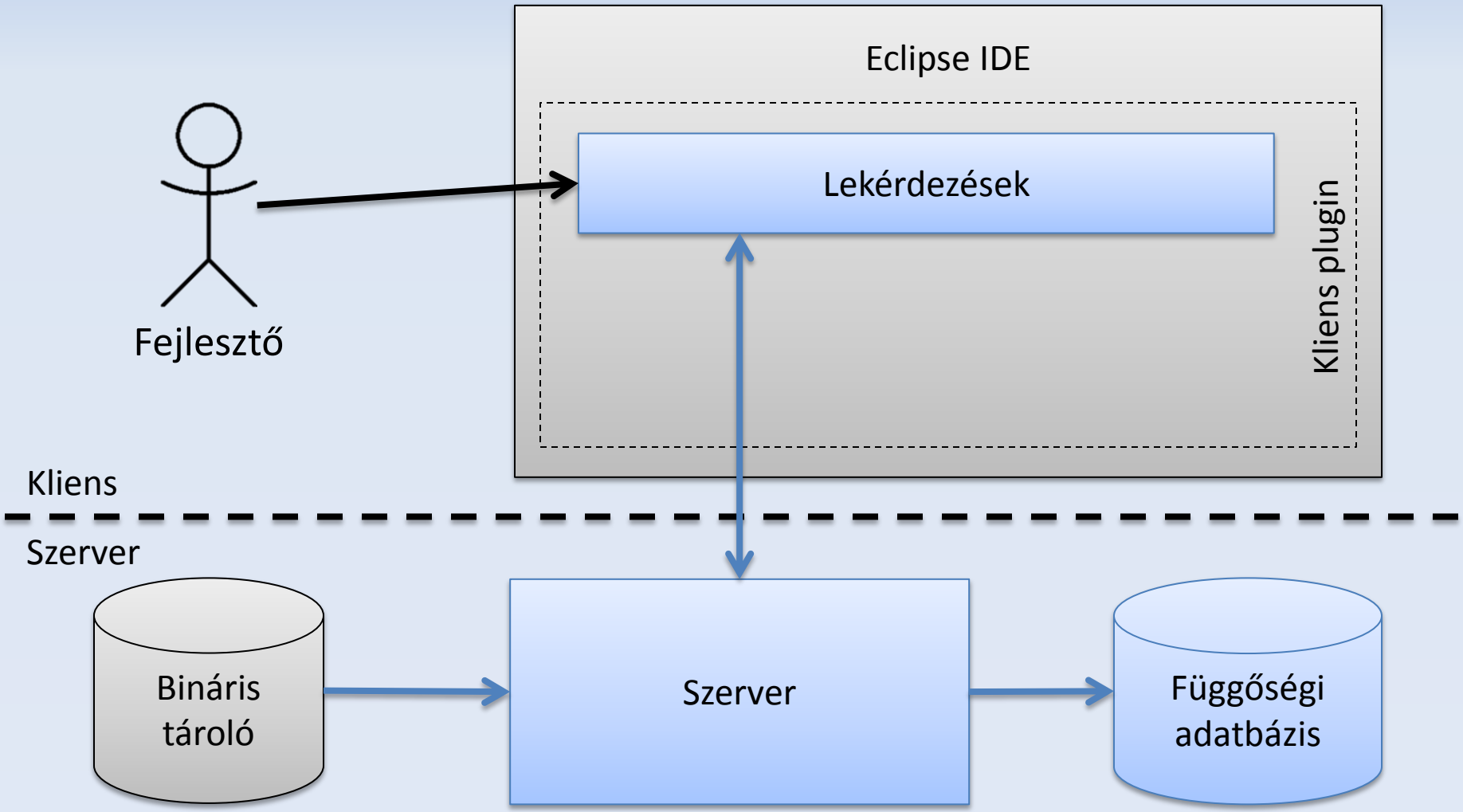
INKREMENTÁLIS, HIBRID FÜGGŐSÉGI ANALÍZIS

Architektúra

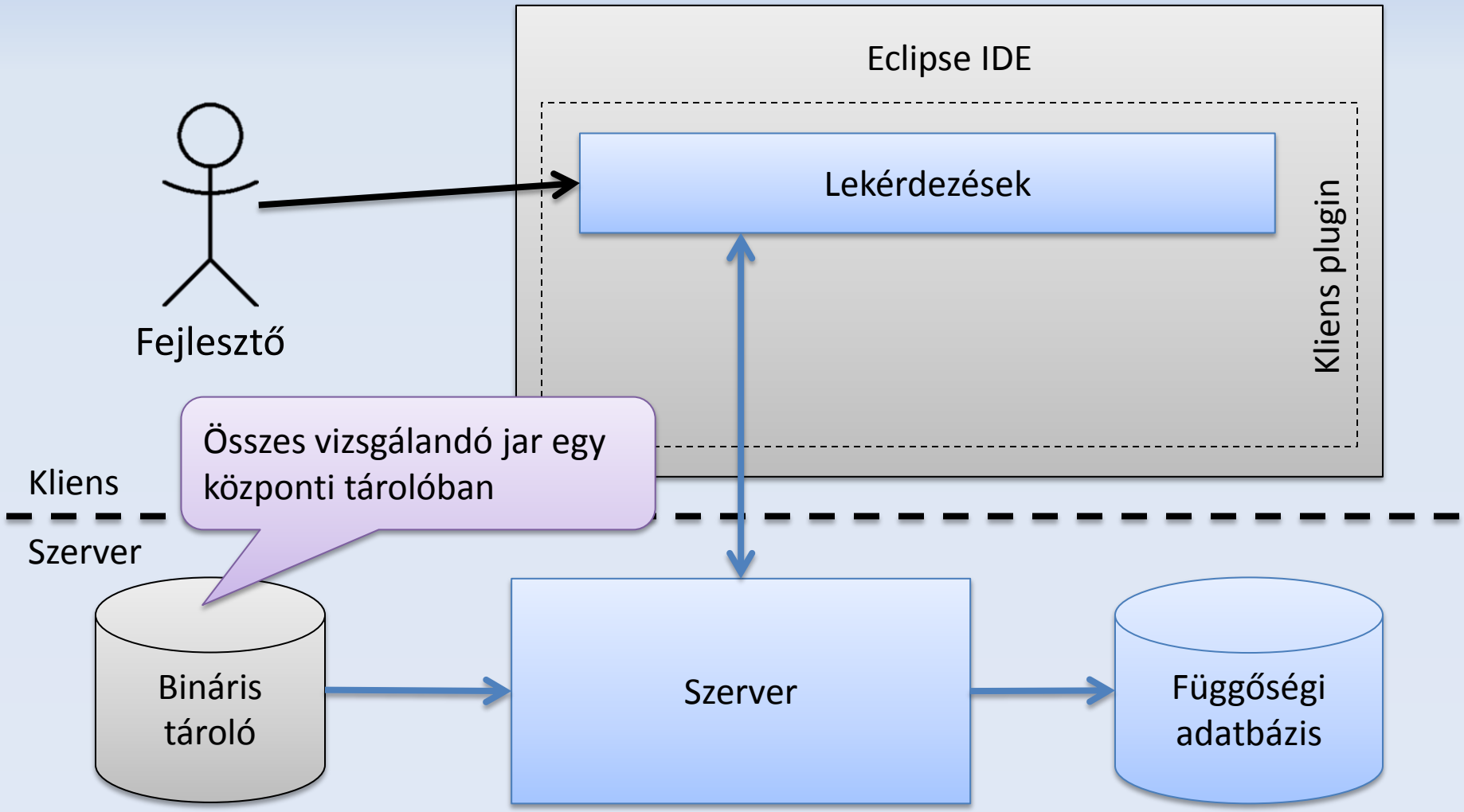
- Függőségi információk megjelenítése (fejlesztői munkaállomások):
  - Gyors lekérdezés a függőségi modellen
  - Eclipse keretrendszerbe integrálva

- Függőségi analízis (build rendszer):
  - Gyors függőségi modellépítés a Java binárisokból
  - Függőségi modell karbantartása új verziók esetén

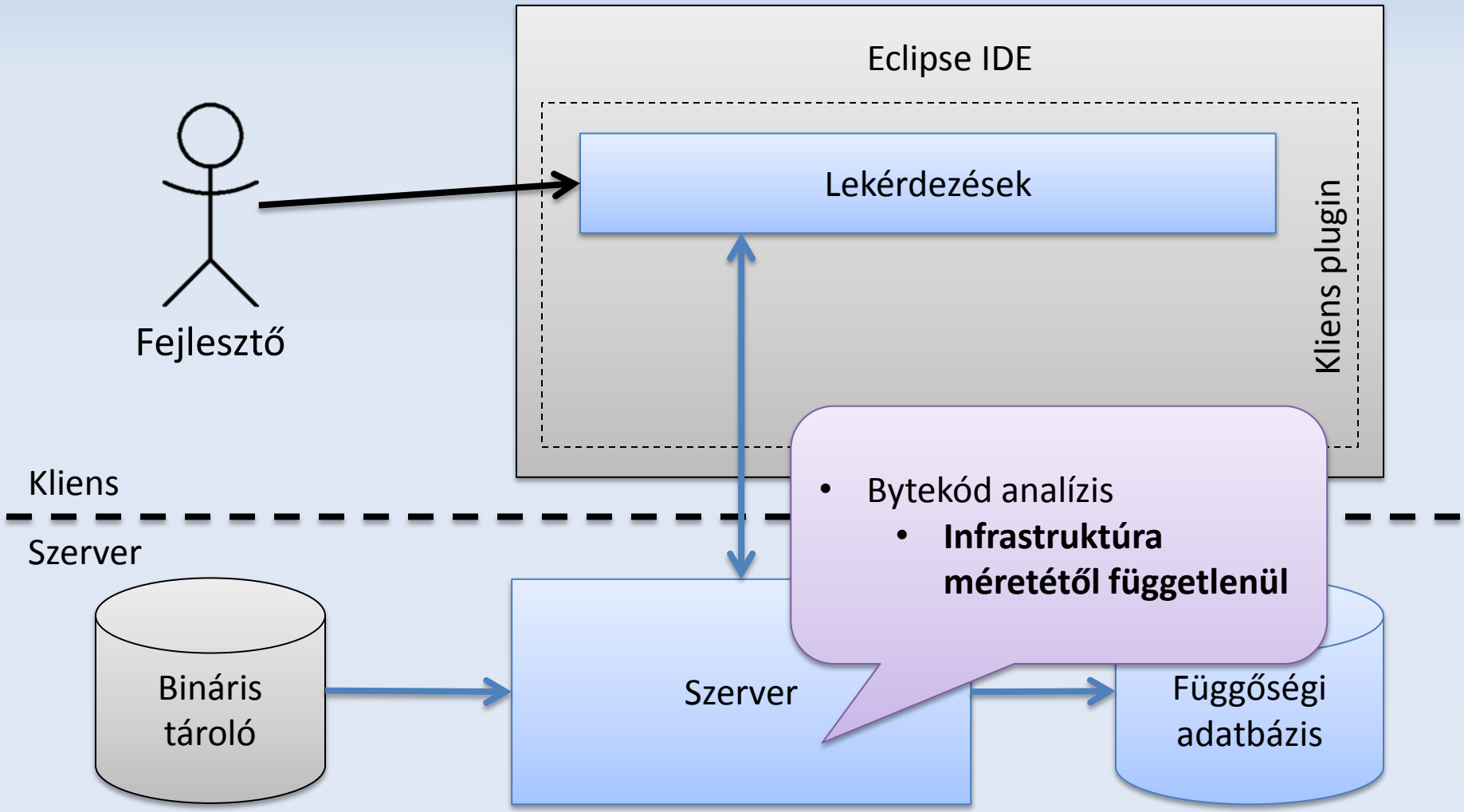
Architektúra



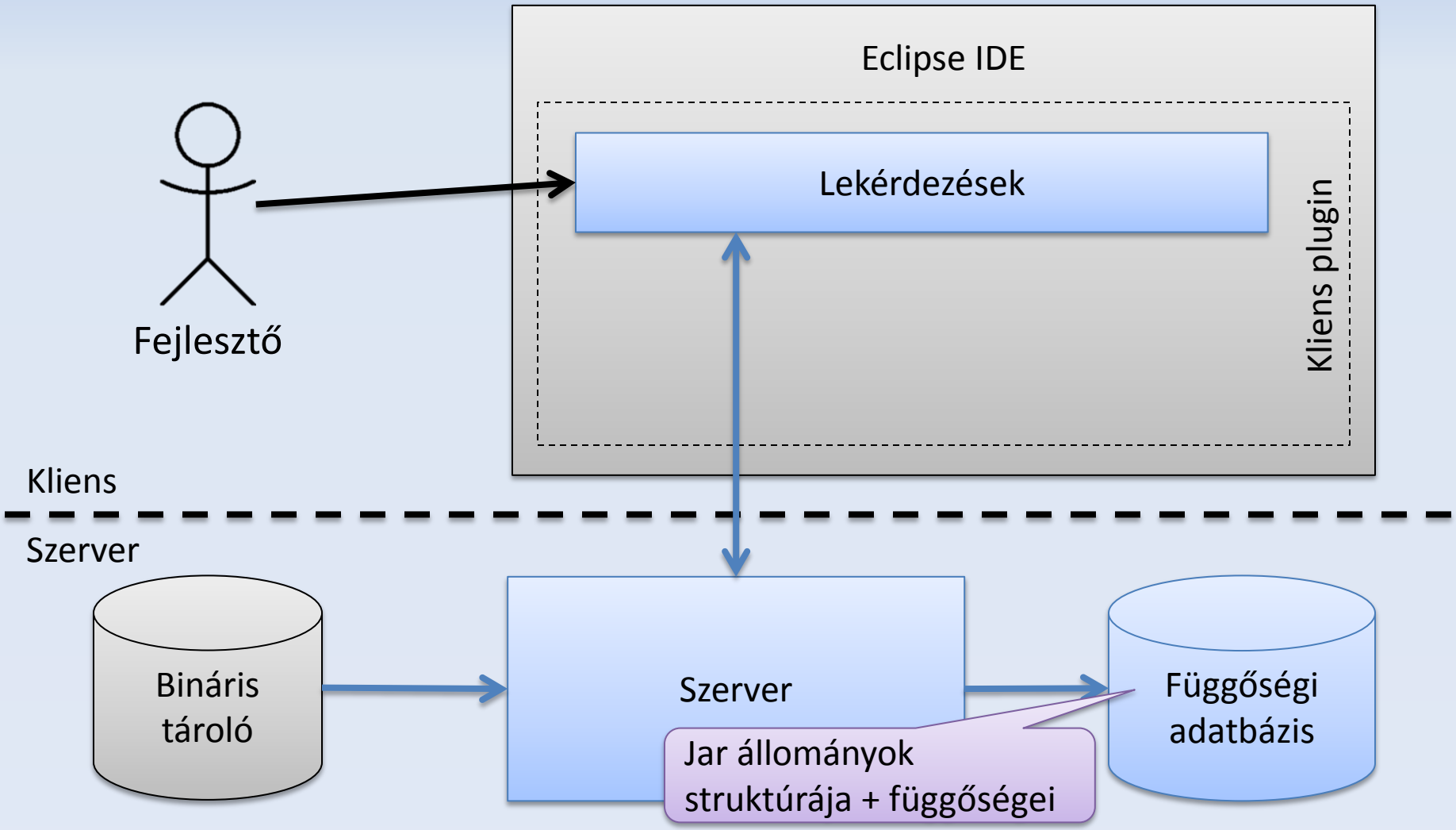
Architektúra



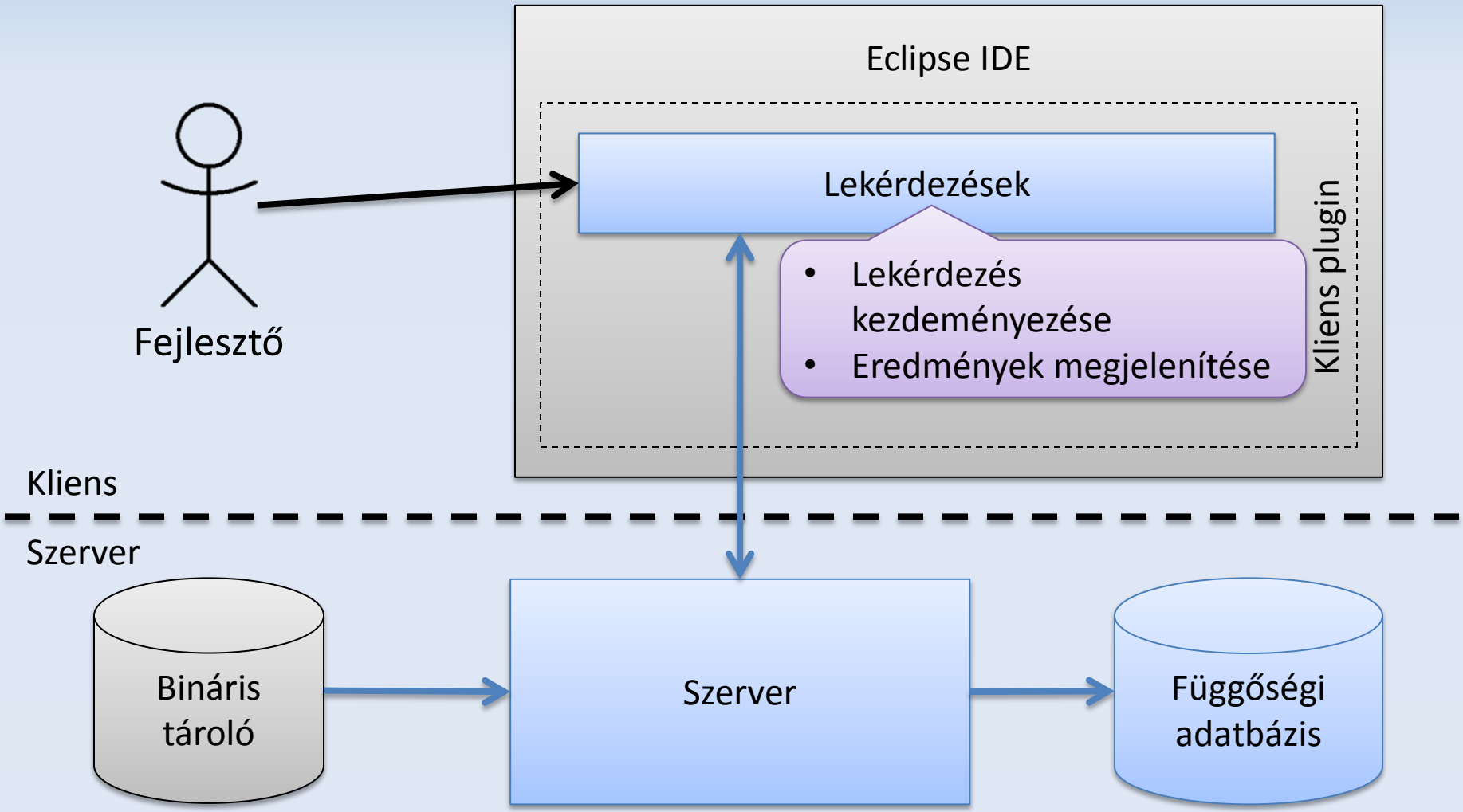
Architektúra



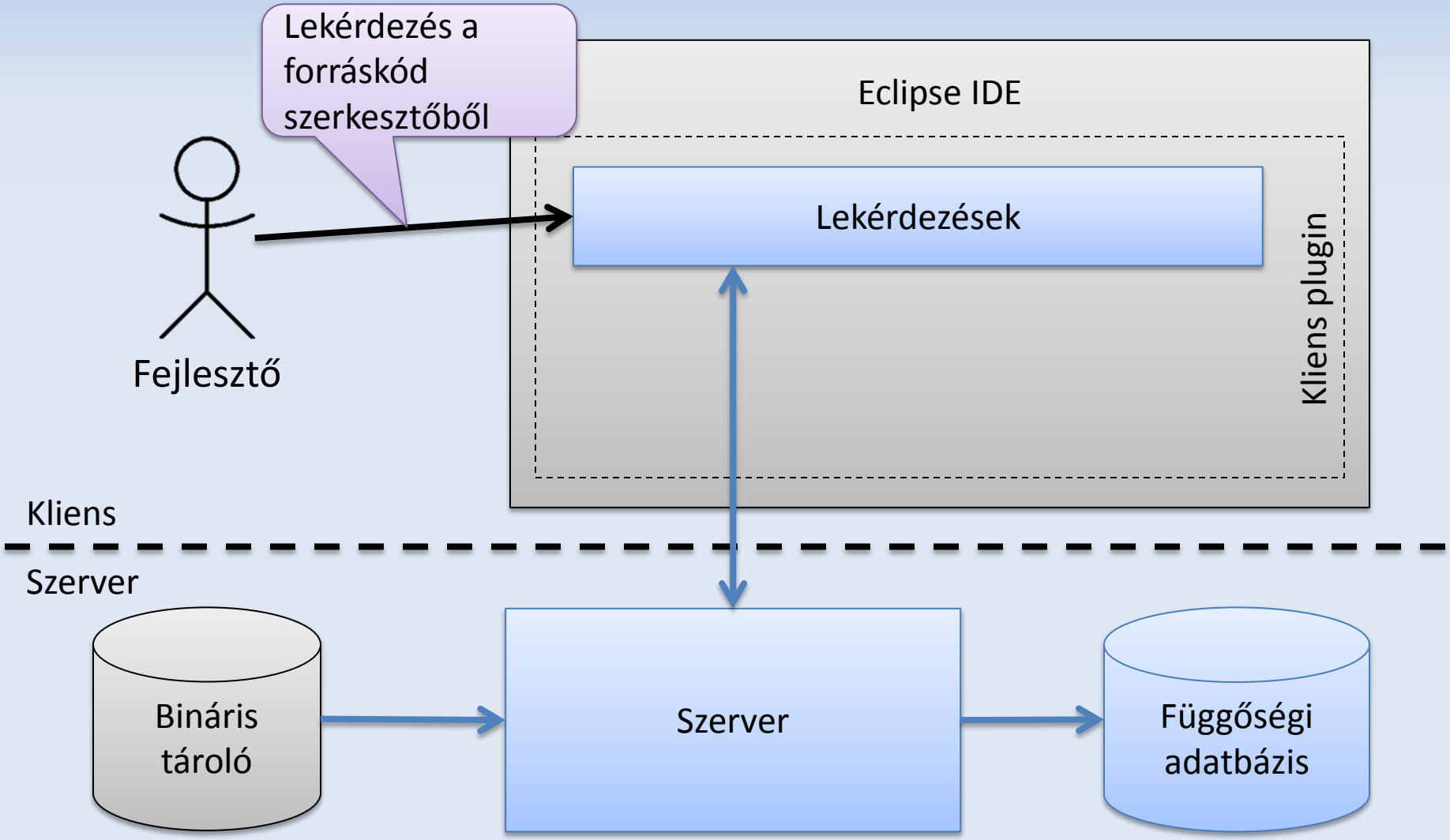
Architektúra



Architektúra



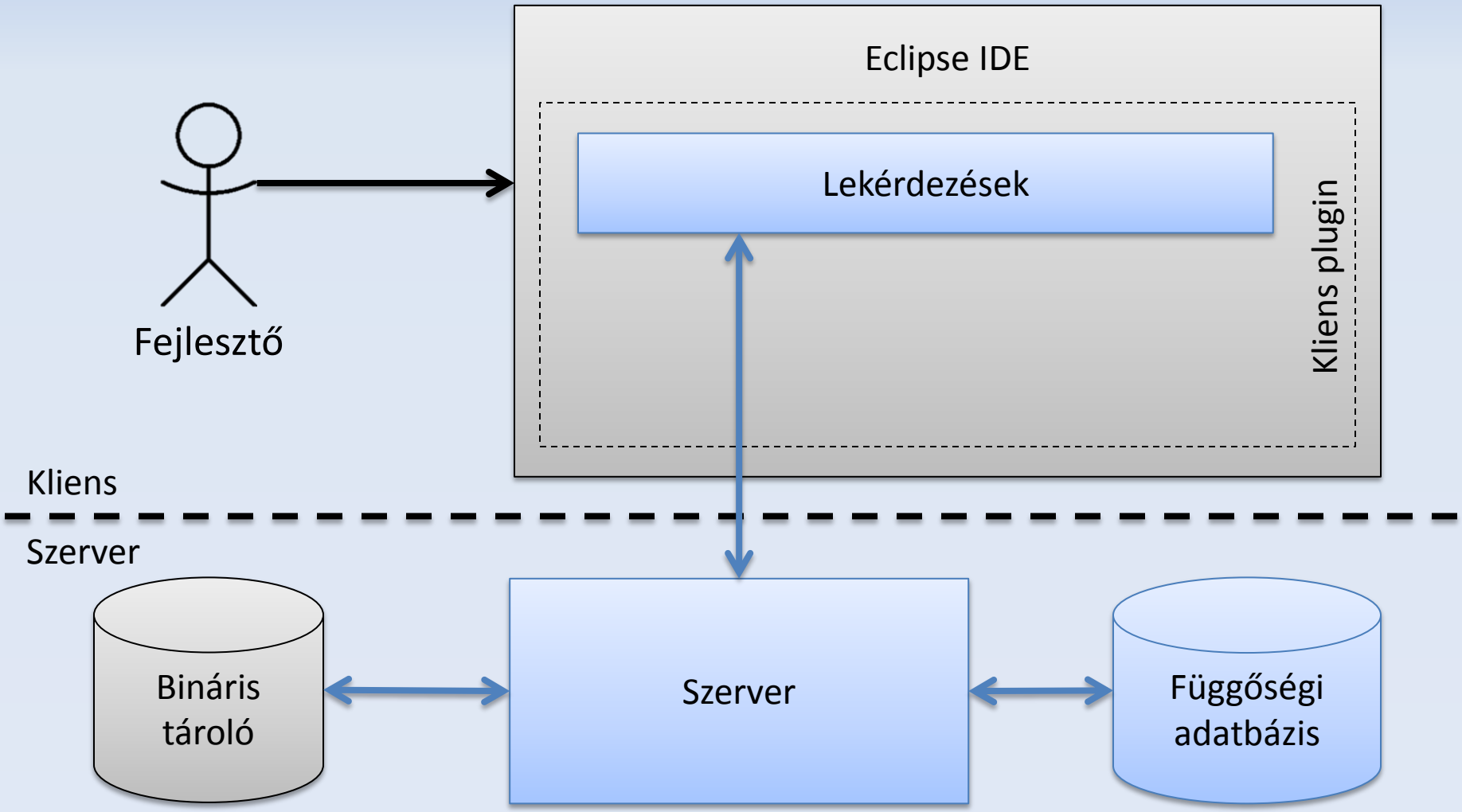
Architektúra



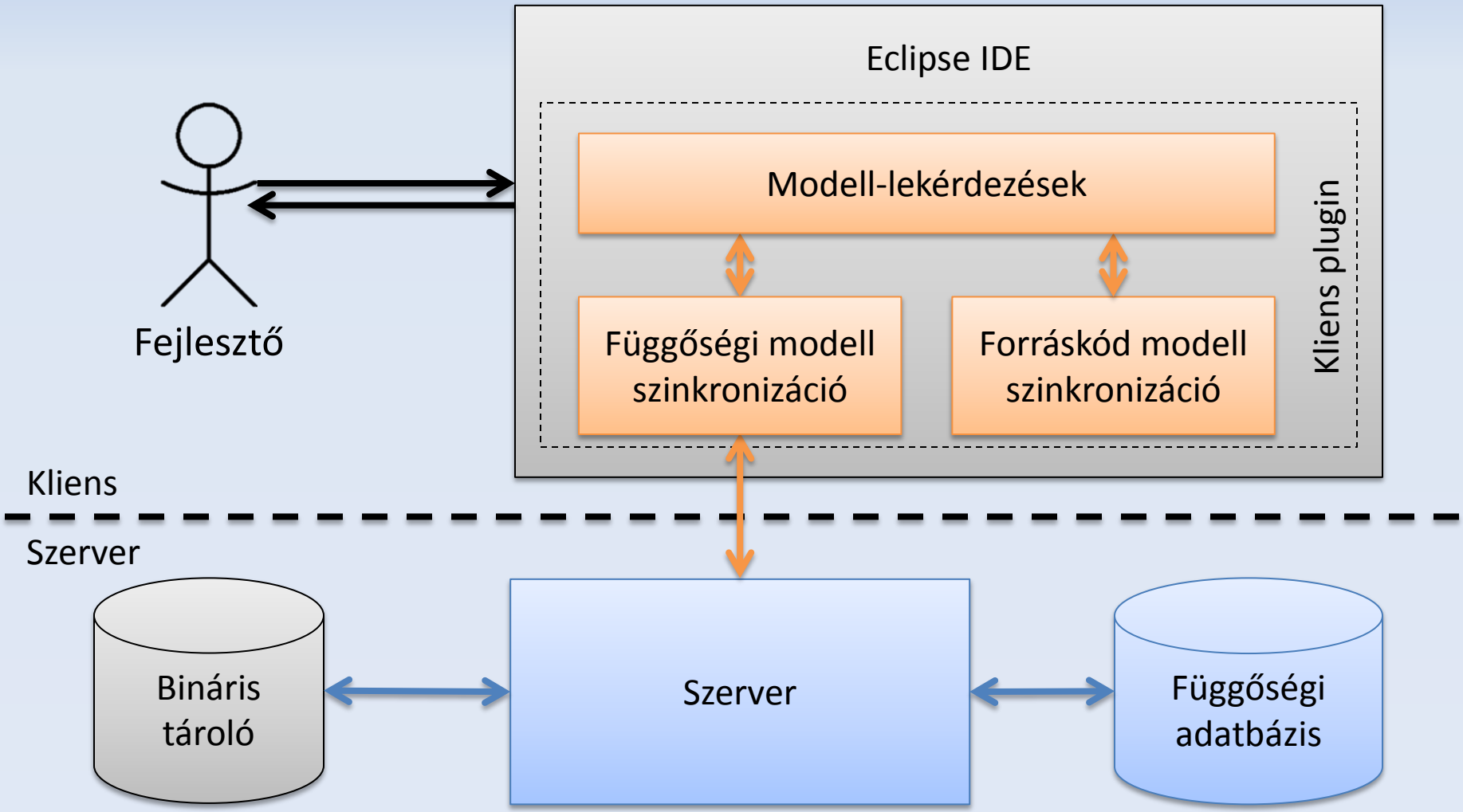
Hibrid függőségi analízis

- Lokális forráskód-projektek felhasználása
  - *Mi változott meg a fejlesztő lokális projektjeiben?*
  - *Milyen hatással van a változás a ráépülő projektekre?*
- Javasolt módszer
  - Forráskód és függőségi adatbázis összekapcsolása (**hibrid analízis**)
  - **Inkrementális lekérdezések** az **összes** elem függőségeire →
    Azonnali visszacsatolás a forráskód szerkesztése közben

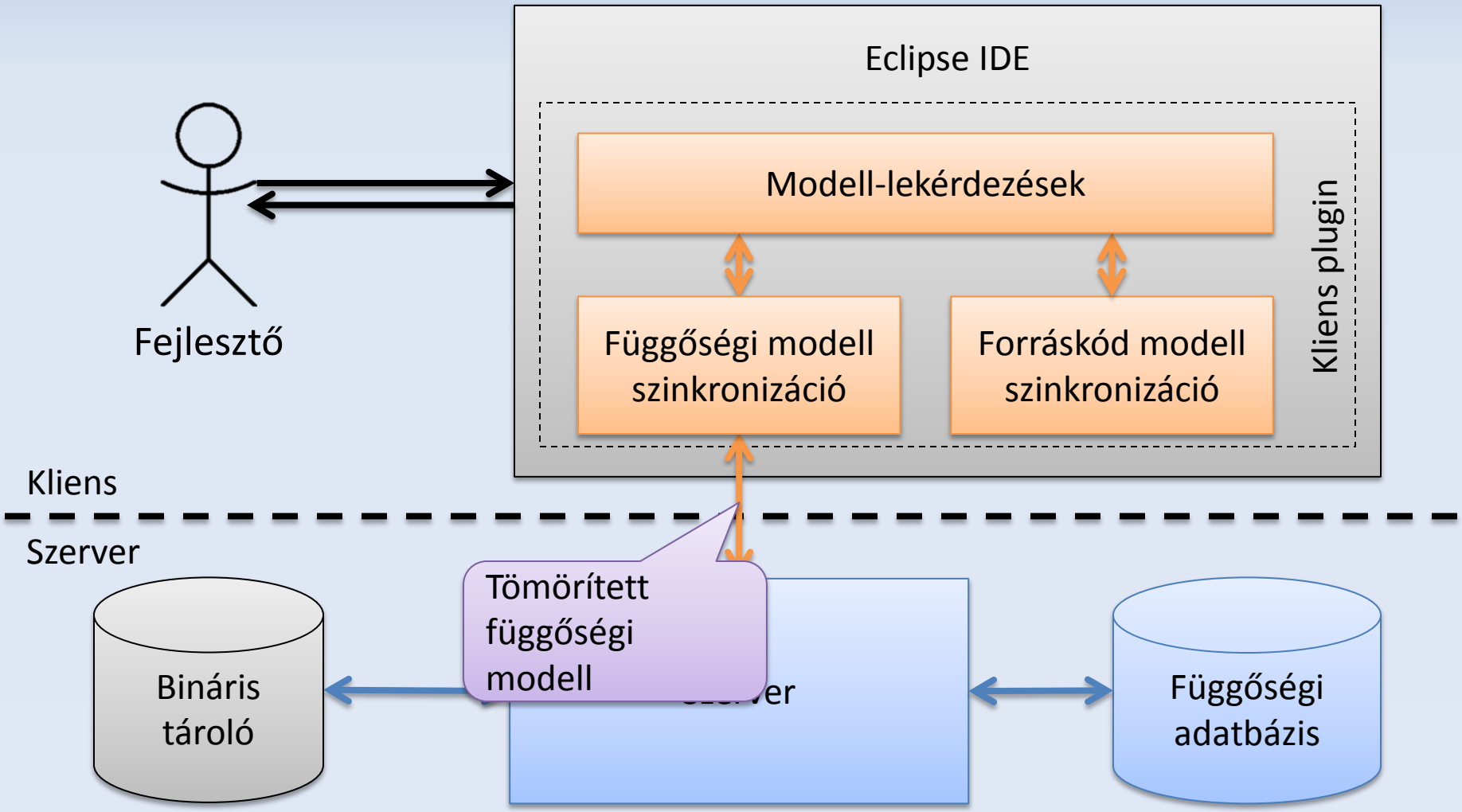
Kiterjesztett architektúra



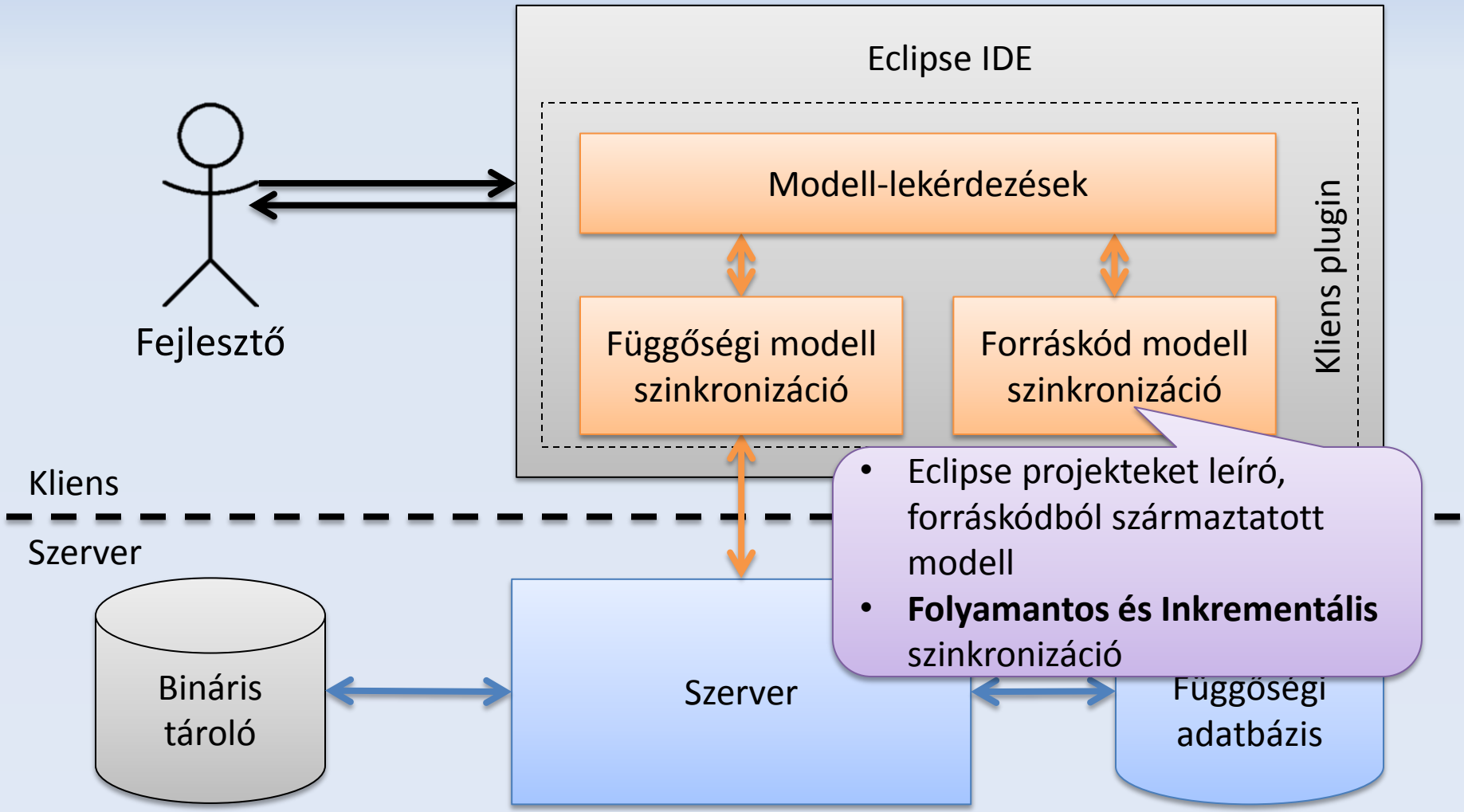
Kiterjesztett architektúra



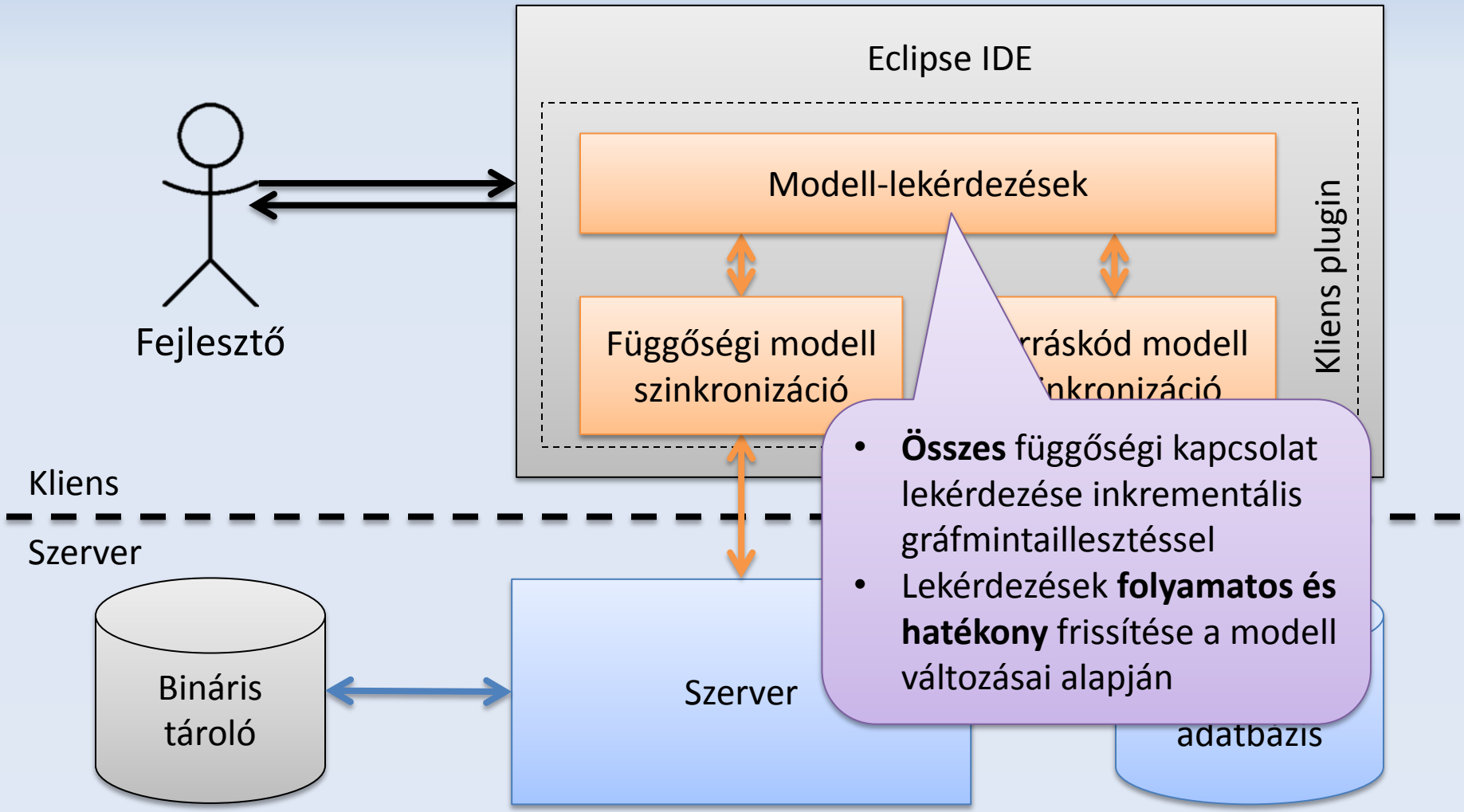
Kiterjesztett architektúra



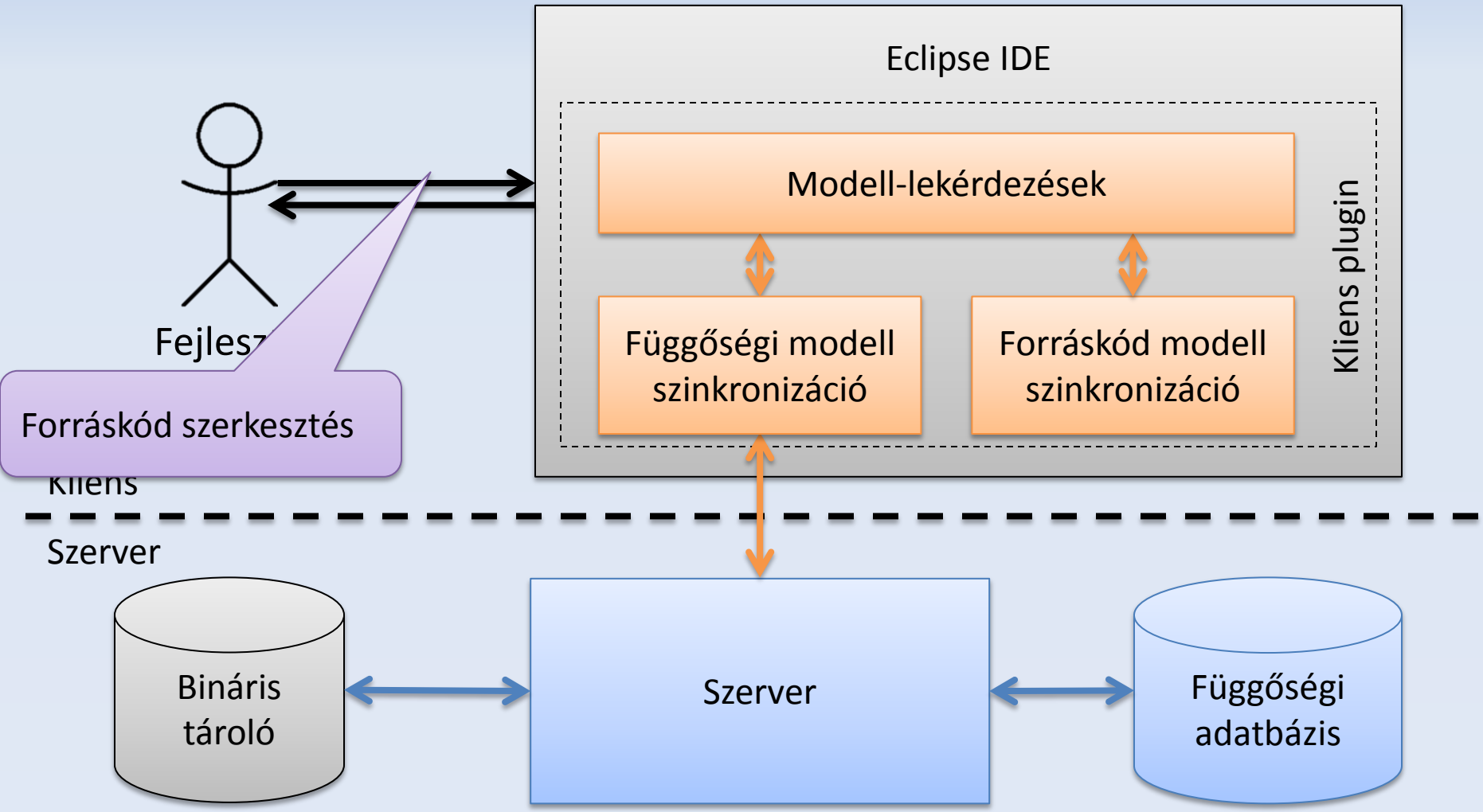
Kiterjesztett architektúra



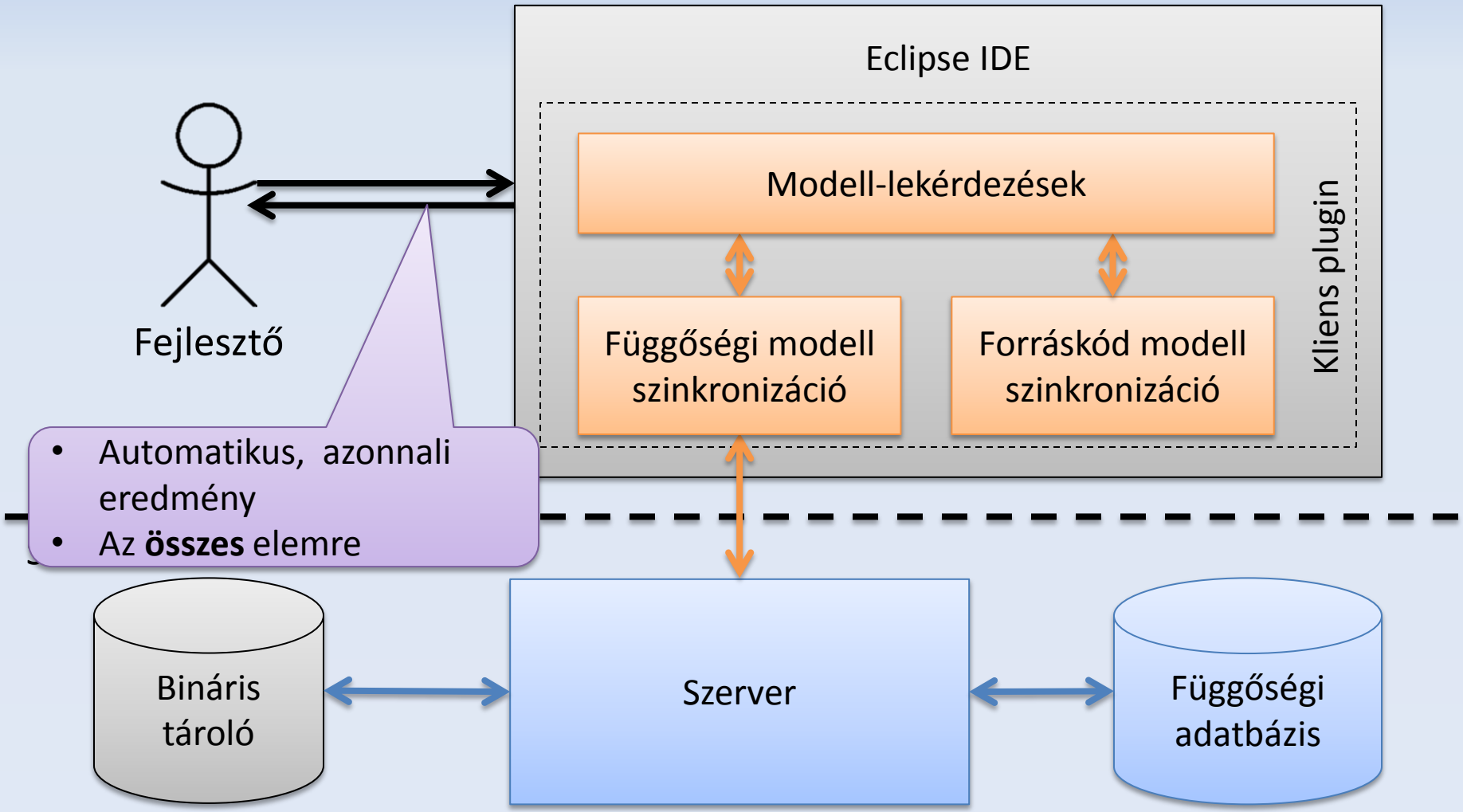
Kiterjesztett architektúra



Kiterjesztett architektúra

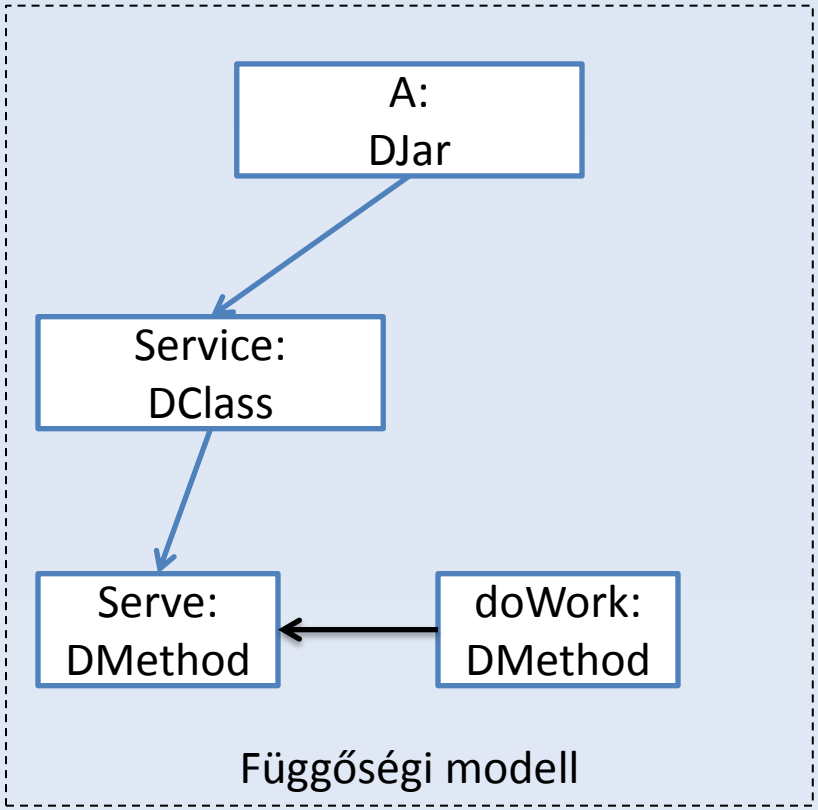


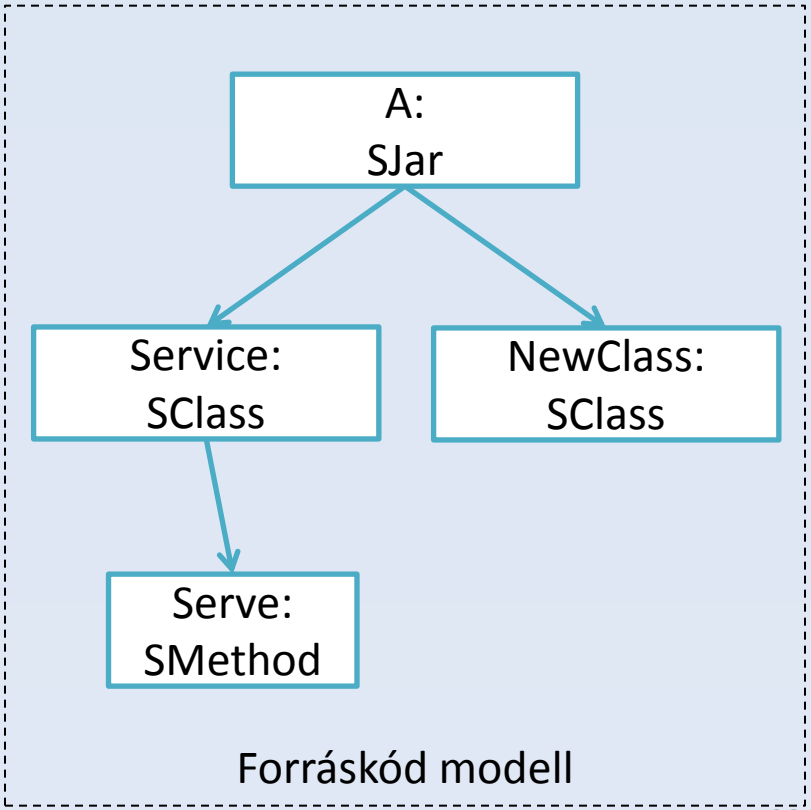
Kiterjesztett architektúra



Inkrementális lekérdezések gráfminták alapján

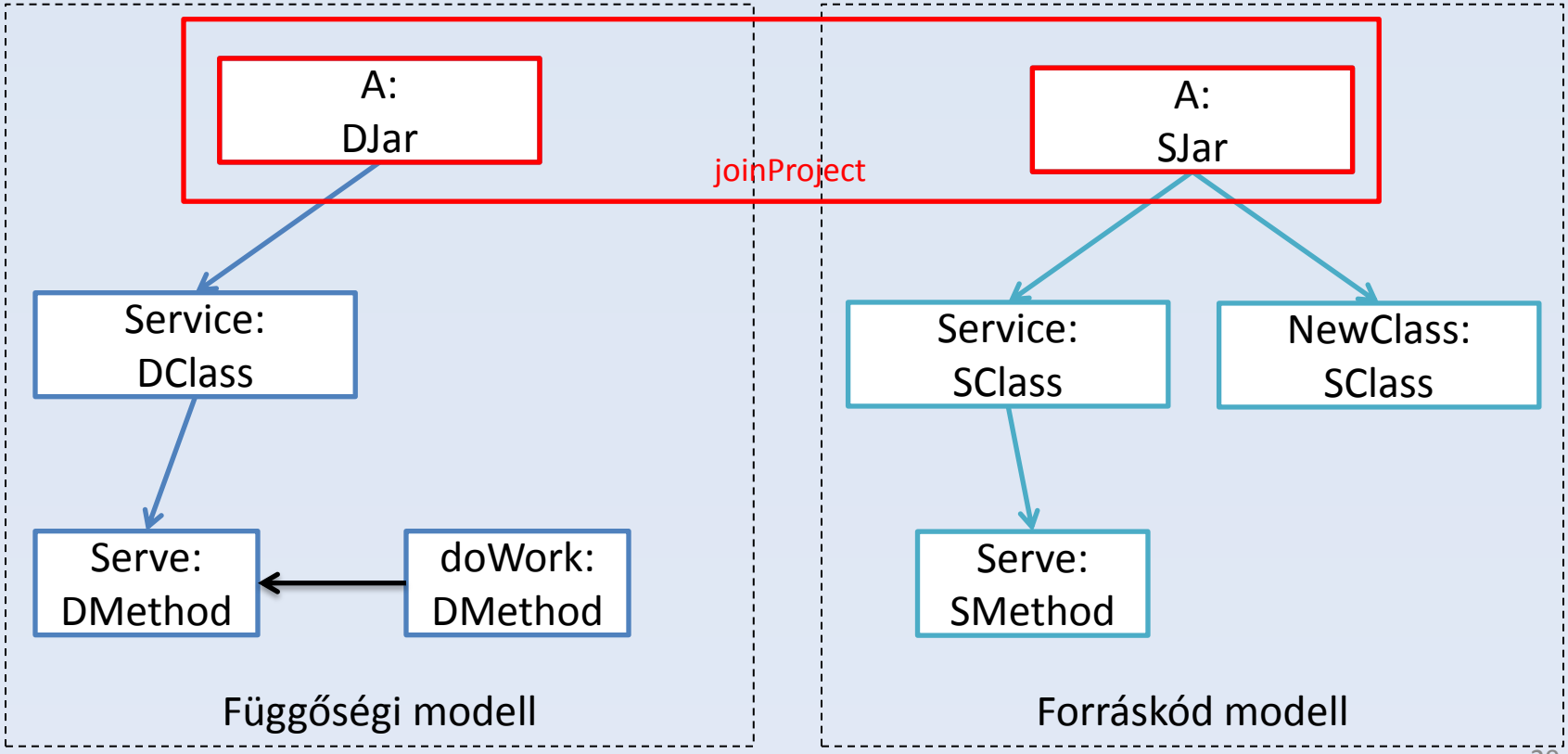
- EMF-IncQuery deklaratív modell-lekérdezések
- A függőségi- és forráskód modellek logikai összekapcsolásával
- Inkrementális kiértékelés = eredmény + eredmény változásai





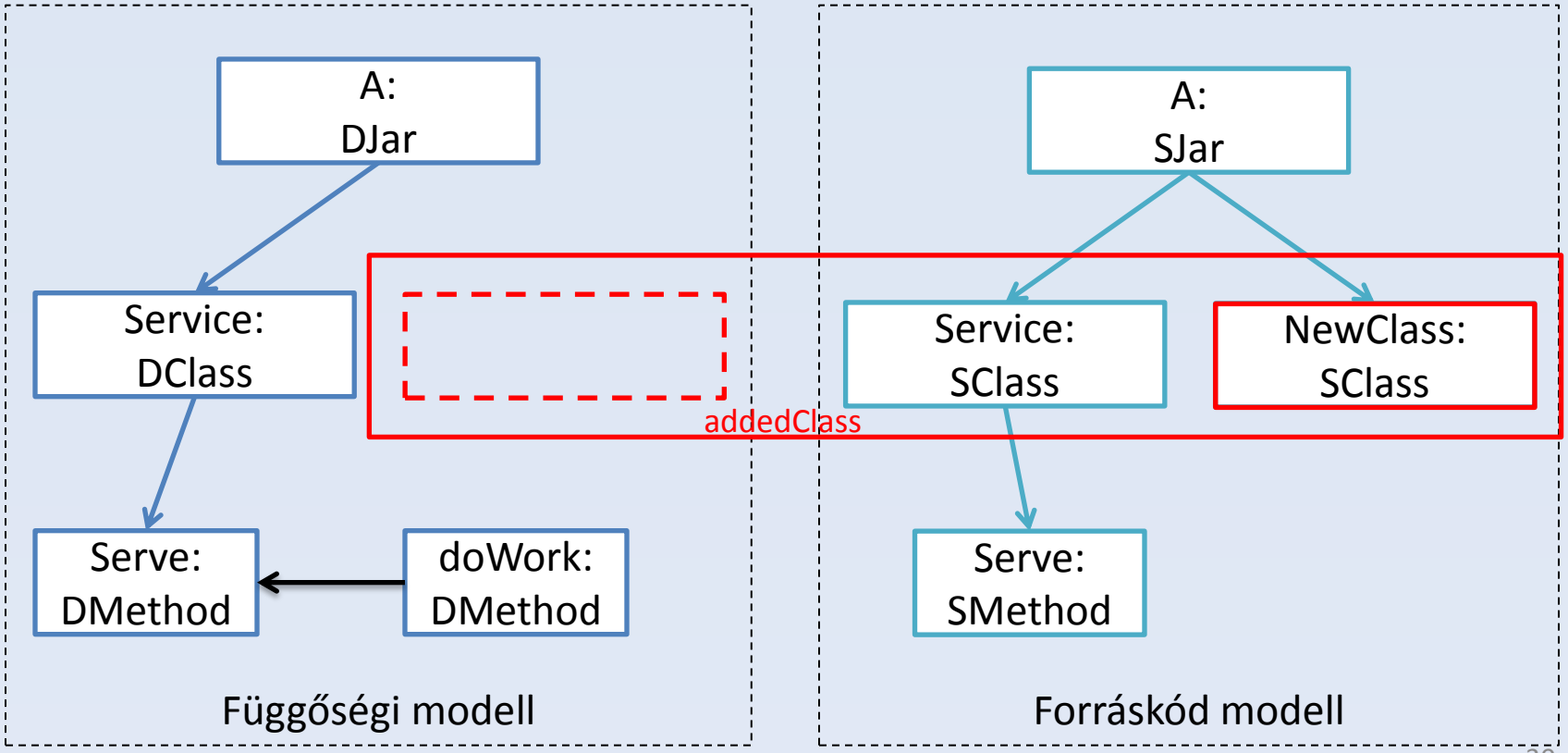
Inkrementális lekérdezések gráfminták alapján

- EMF-IncQuery deklaratív modell-lekérdezések
- A függőségi- és forráskód modellek logikai összekapcsolásával
- Inkrementális kiértékelés = eredmény + eredmény változásai



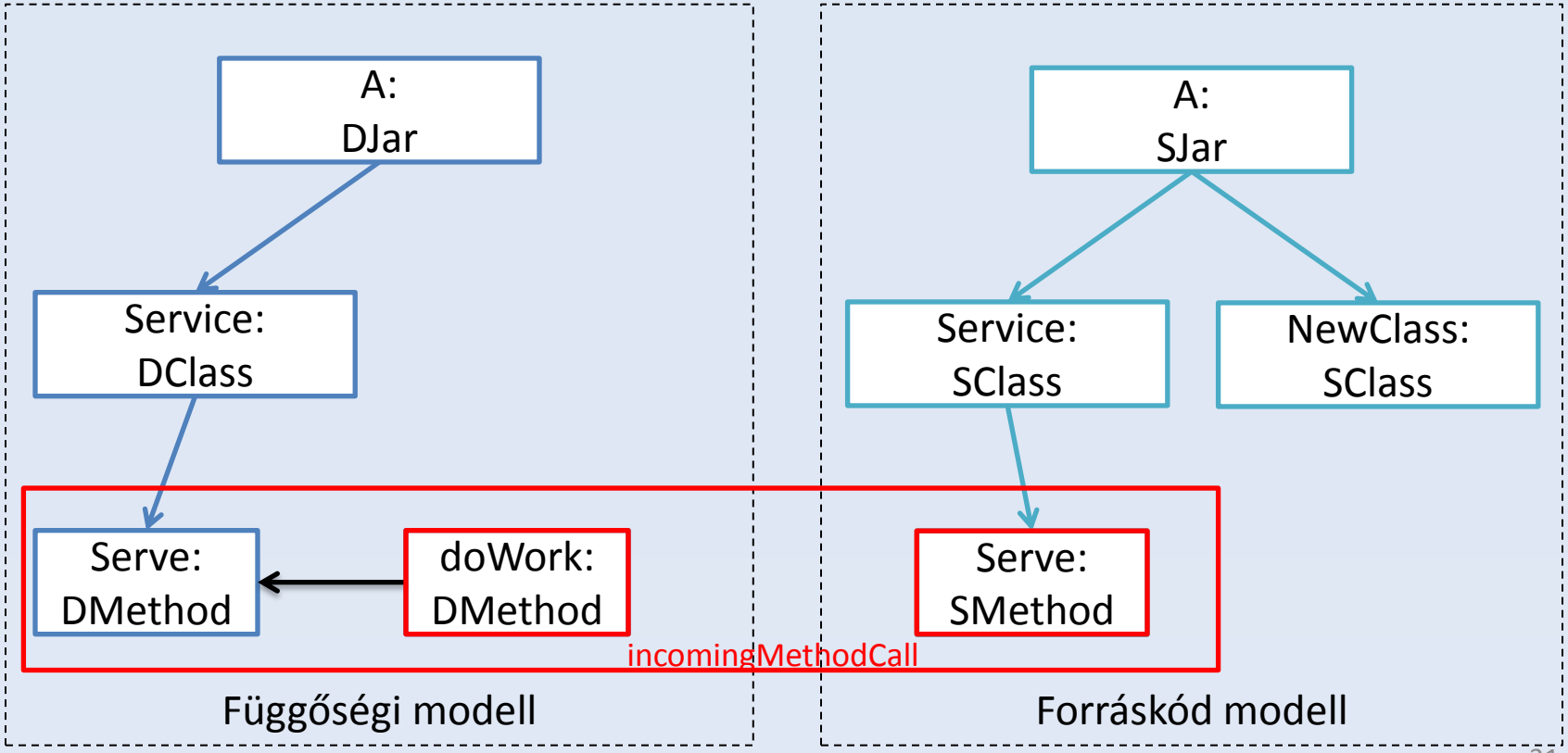
Inkrementális lekérdezések gráfminták alapján

- EMF-IncQuery deklaratív modell-lekérdezések
- A függőségi- és forráskód modellek logikai összekapcsolásával
- Inkrementális kiértékelés = eredmény + eredmény változásai



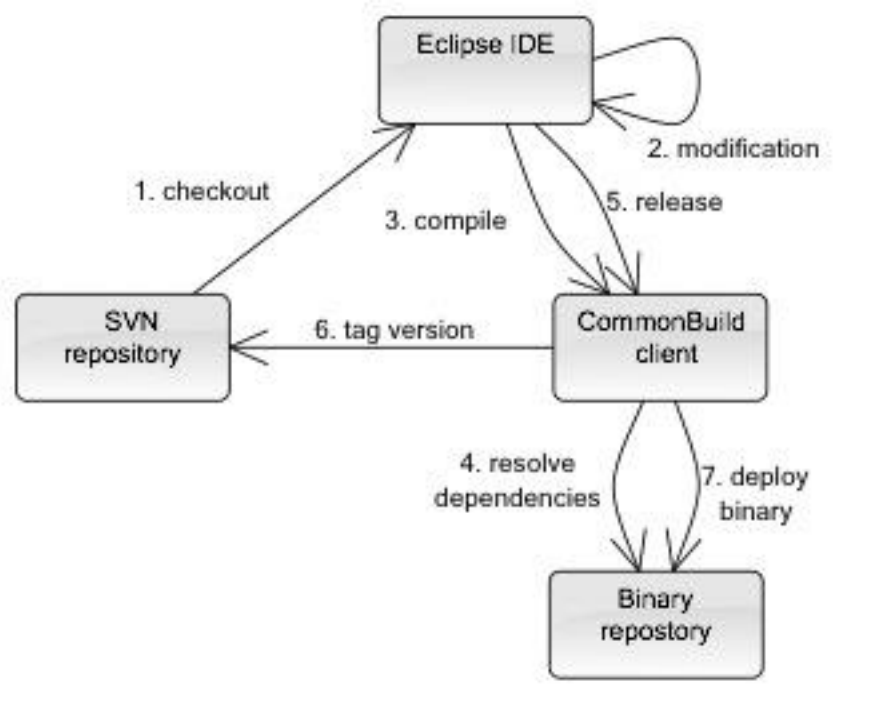
Inkrementális lekérdezések gráfminták alapján

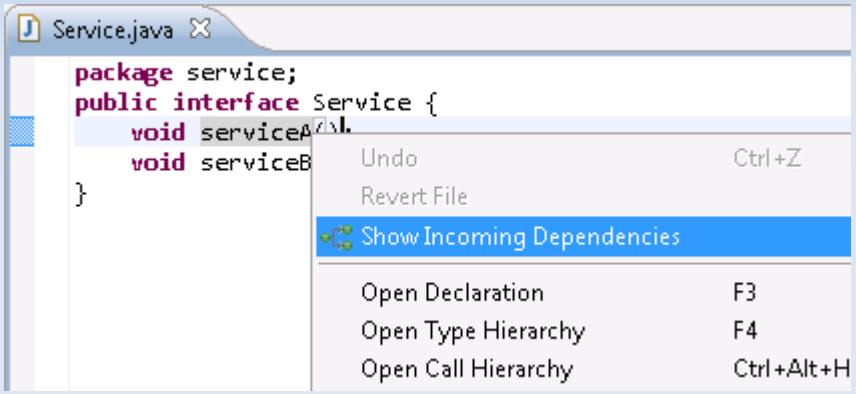
- EMF-IncQuery deklaratív modell-lekérdezések
- A függőségi- és forráskód modellek logikai összekapcsolásával
- Inkrementális kiértékelés = eredmény + eredmény változásai

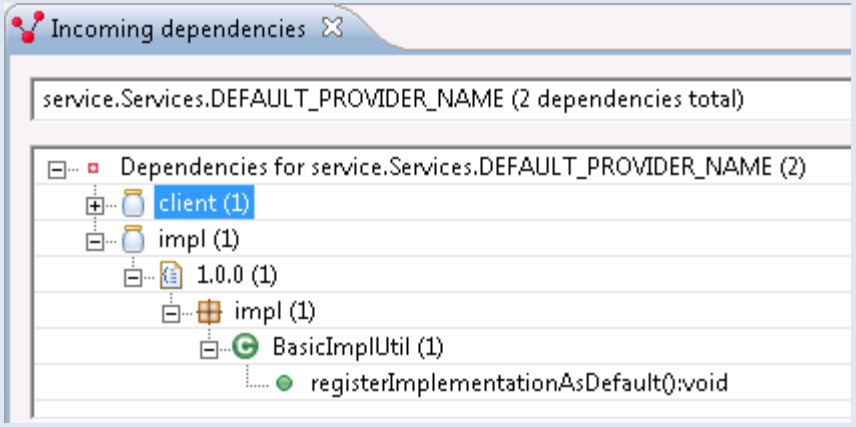


Integráció a fejlesztői keretrendszerbe

- Fejlesztési folyamat

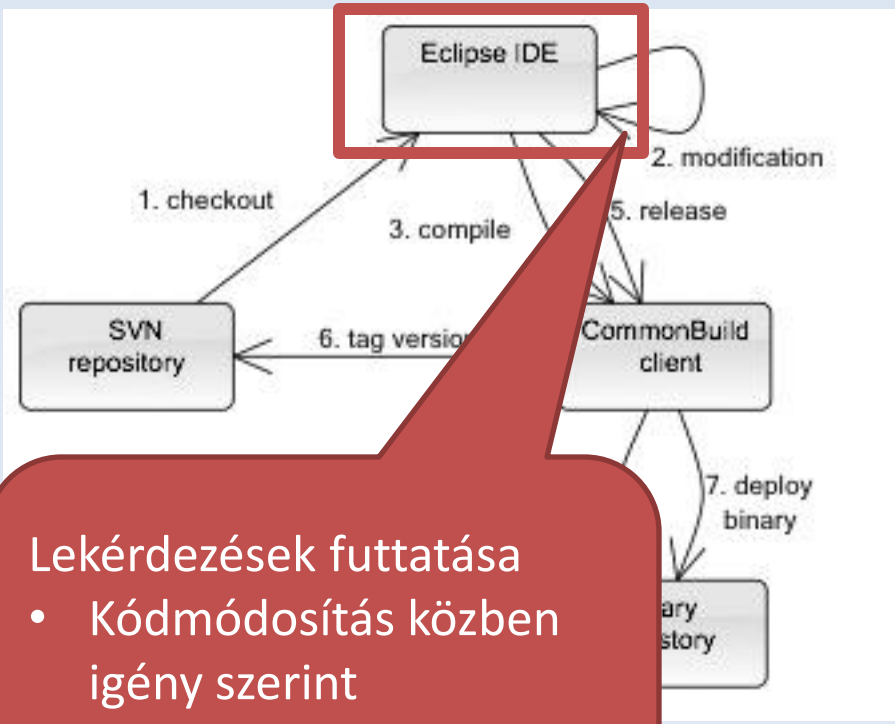






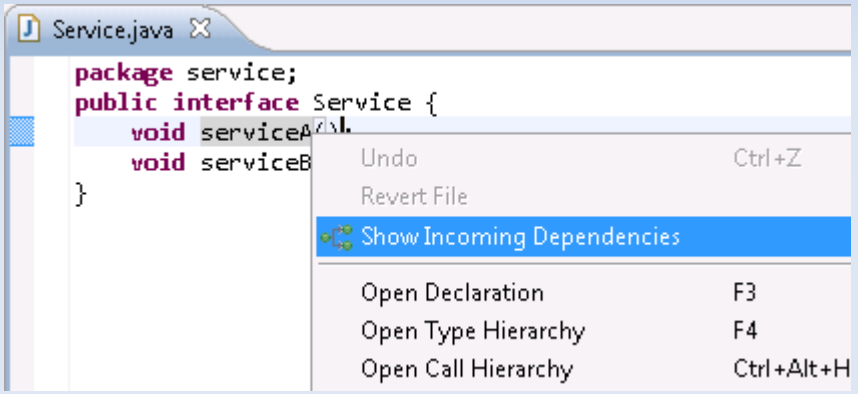
Integráció a fejlesztői keretrendszerbe

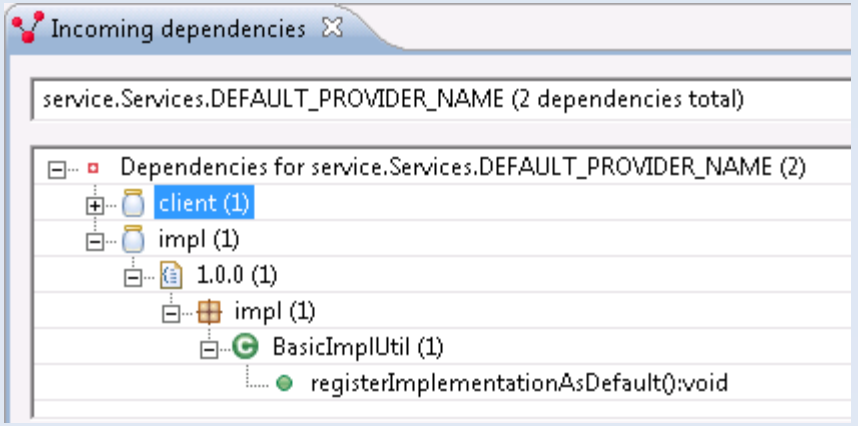
- Fejlesztési folyamat



Lekérdezések futtatása

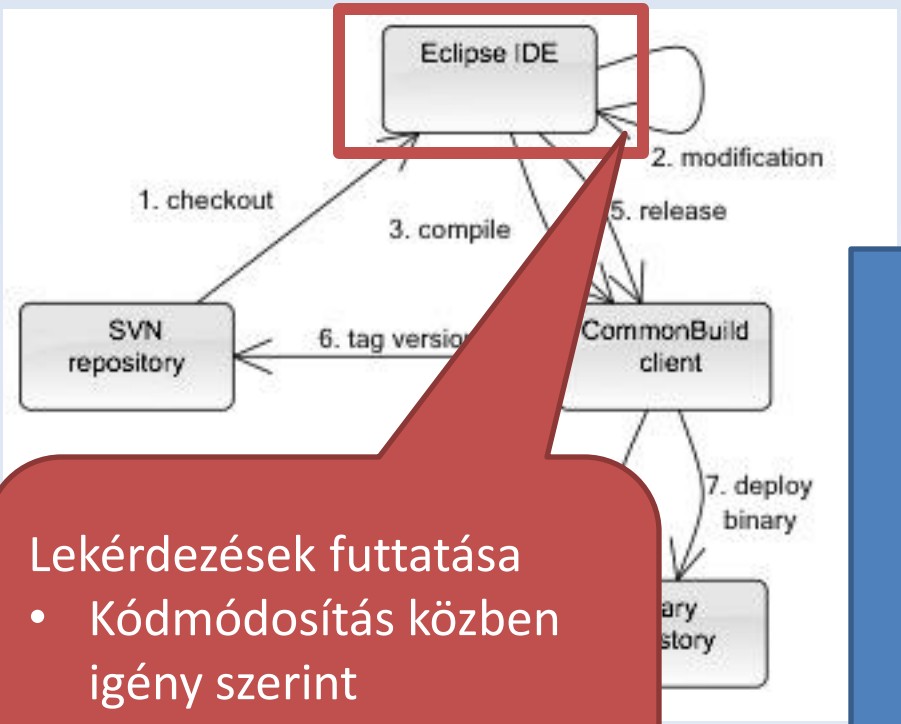
- Kódmódosítás közben igény szerint
- Release előtt ellenőrzési céllal



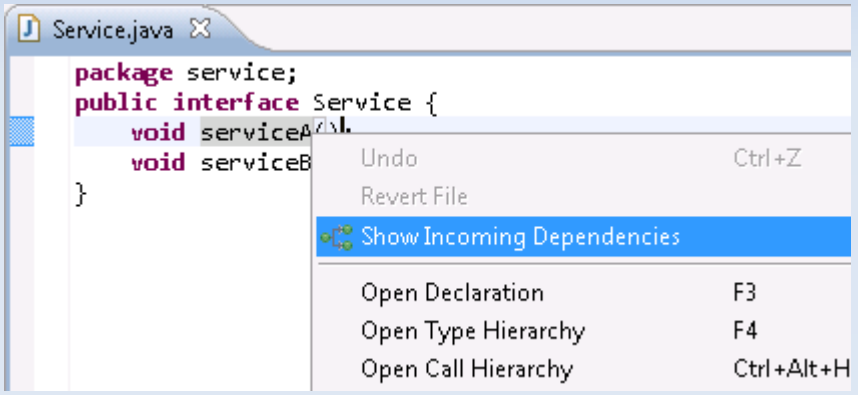


Integráció a fejlesztői keretrendszerbe

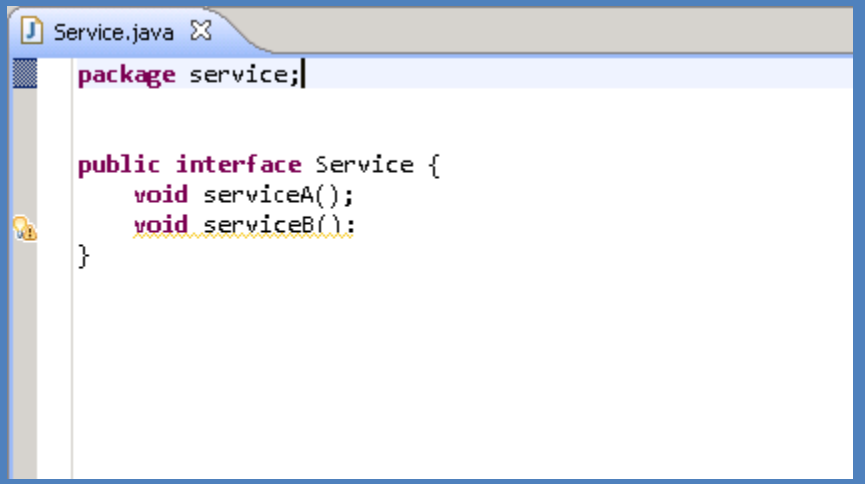
- Fejlesztési folyamat



Lekérdezések futtatása
- Kódmódosítás közben igény szerint
- Release előtt ellenőrzési céllal



Kiterjesztési lehetőség:

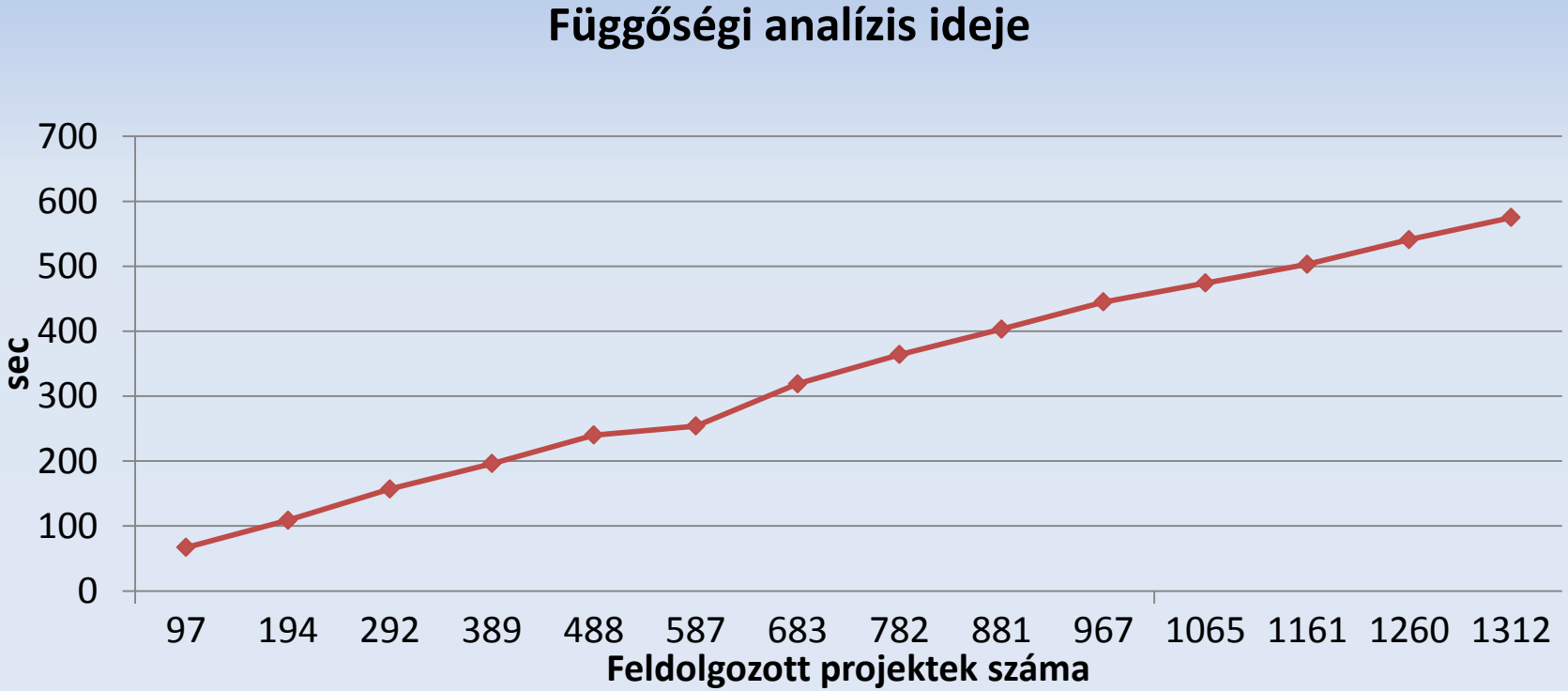


A RENDSZER TELJESÍTMÉNYE

Hatékonyság mérése – miért?

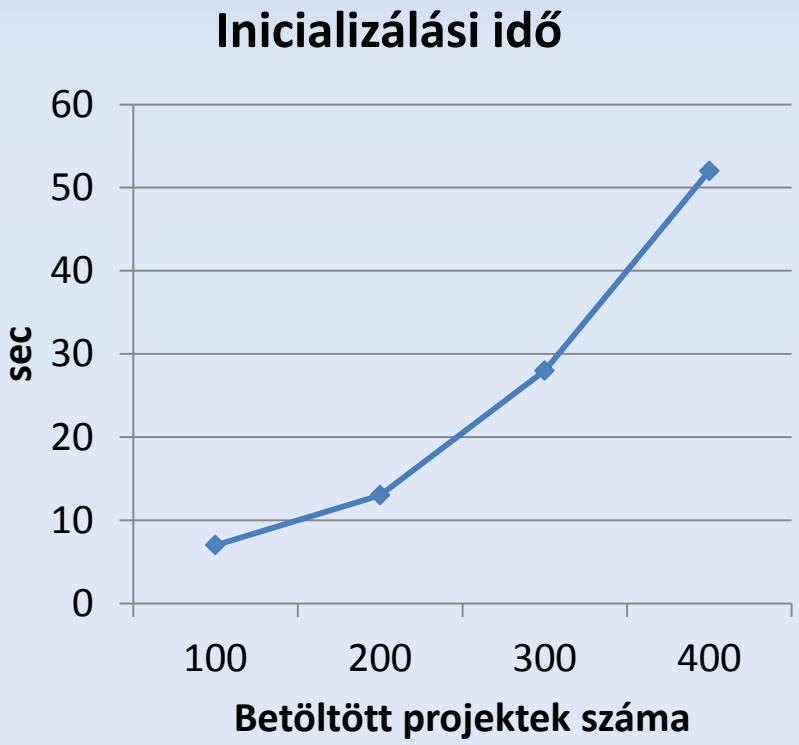
- Az eszköz működését a CERN Controls Systems fejlesztőivel együttműködve, éles üzemben értékeltük ki
- Célok
  - Build szerver:
    - Bináris függőségi analízis gyors legyen (1300+ JAR)
    - Függőségek lekérdezése gyors legyen
  - Fejlesztői környezet:
    - Azonnali függőségi analízis visszacsatolás a forráskód módosításával a teljes szoftverinfrastruktúrára! (= tipikus lokális munkakörnyezet – 5-10 projekt – és a szoftverinfrastruktúra – 100+ projekt – **együttesén** működjön, erőforráskorlátos fejlesztői gépeken is)

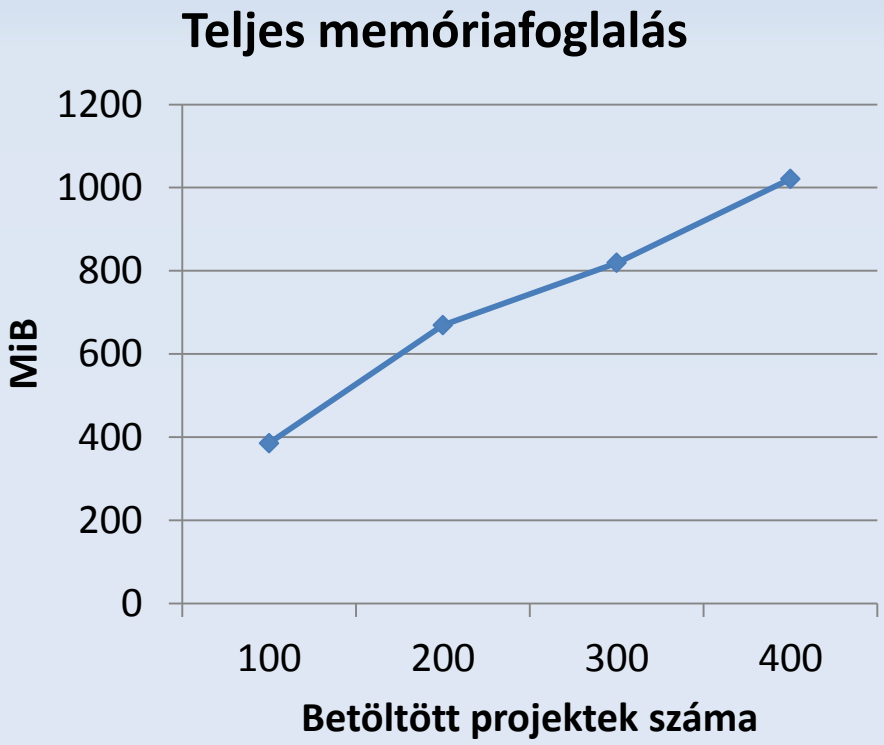
Függőségi analízis sebessége



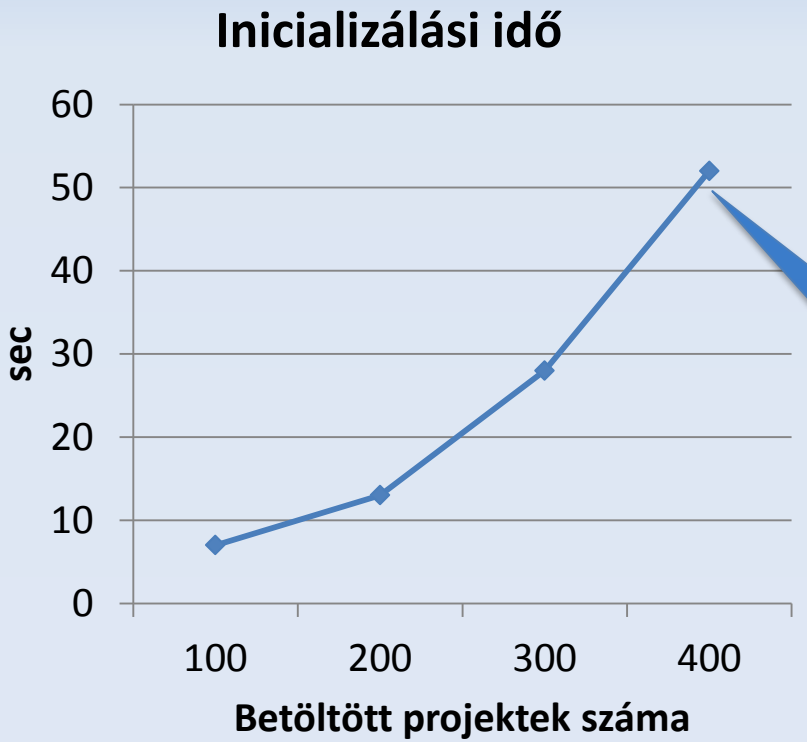
- Teljes függőségi analízis:
  - Kb. 10 perc **(vö: kb. 1 napos teljes build**)
  - Függőségi viszonyok felderítése és karbantartása: ~0,5sec/projekt **(vö: néhány perc / projekt build idő)**
- Függőségi lekérdezés ideje egyetlen elemre: ~200ms

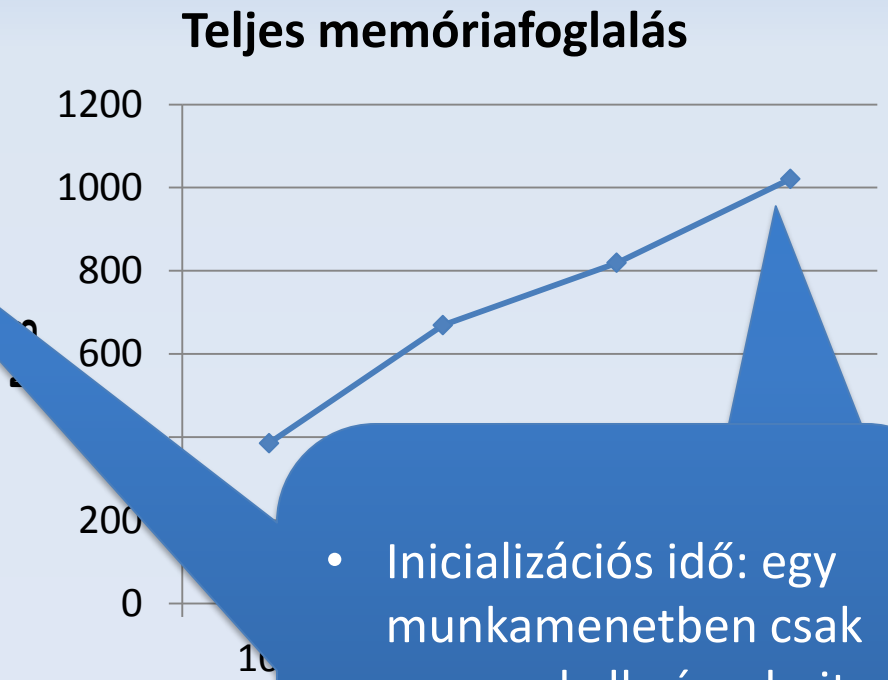
Modell-lekérdezések teljesítménye





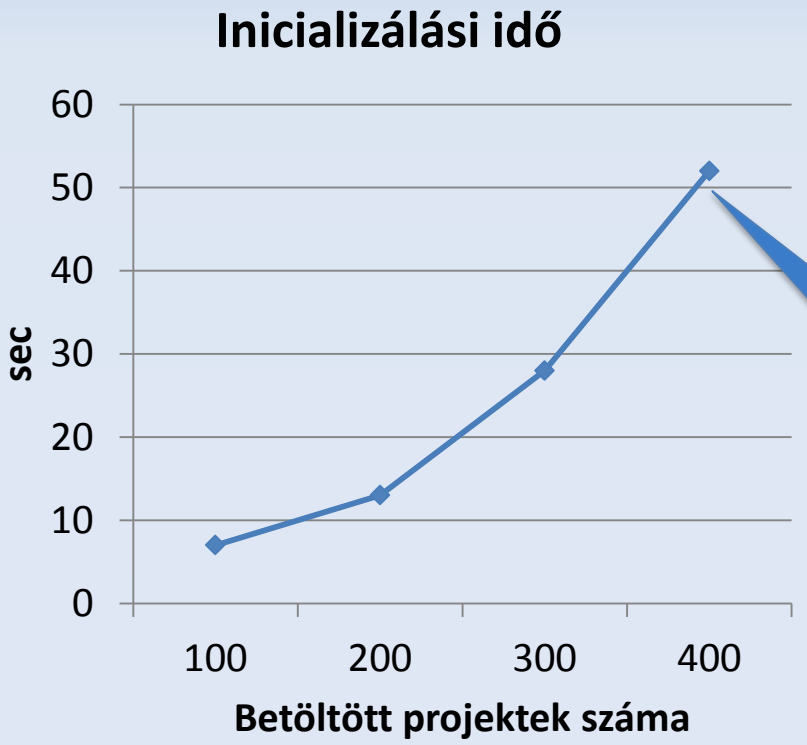
Modell-lekérdezések teljesítménye

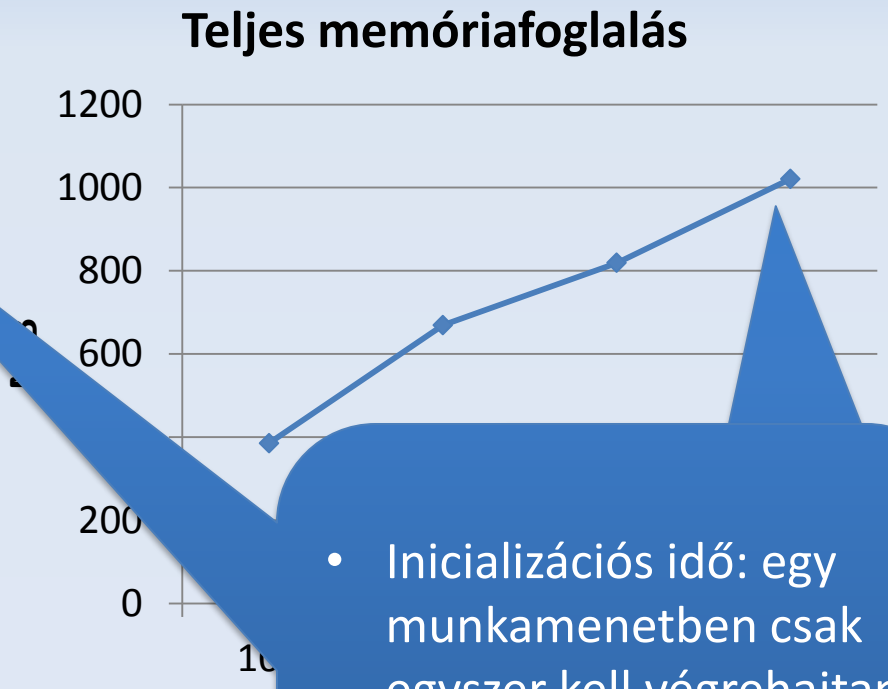




- Inicializációs idő: egy munkamenetben csak egyszer kell végrehajtani
- Memóriafoglalás: éles infrastruktúra elfér 1GB RAM-ban

Modell-lekérdezések teljesítménye





- Inicializációs idő: egy munkamenetben csak egyszer kell végrehajtani
- Memóriafoglalás: éles

- Kiértékelés egy változtatás esetén: ~1ms az **összes** elemre és kapcsolatra → azonnali visszacsatolás

EREDMÉNYEK ÖSSZEFOGLALÁSA

Eredmények

- Új módszer nagyméretű szoftver infrastruktúra hibrid, inkrementális függőségi analízisére
  - Forráskód és bináris függőségi modellek összekapcsolása alapján
  - Inkrementális gráfmintaillesztéssel
- Megvalósított keretrendszer
  - Nagy mennyiségű bináris komponens hatékony függőségi analízise
  - Inkrementális modell-forráskód szinkronizáció
  - A rendszer teljesítőképességét igazoló mérések
- A rendszer jelenleg éles használatban van: CERN Controls Systems
  - ~1300 Java projekt, projektenként átlag 15 aktív verzió / projekt, átlagosan összesen 10 release / nap
  - Virtualizált fejlesztői munkaállomások (2GB RAM)

VÁLASZOK A BÍRÁLÓK KÉRDÉSEIRE

C/C++ kiterjesztés

- Forráskód kezelése:
  Eclipse CDT AST modell alapján
- Függőségi analízis:
  - Forráskód esetben fordítóprogrammal
    (a build közben)
  - Bináris esetben fordítófüggő, nyitott kérdések
    - Statikusan linkelt binárisok esetén kódinstrumentációval („debug szimbólumok” alapján)
    - Dinamikusan linkelt esetben hivatkozott könyvtárak szimbólumai alapján
    - Bináris kód belső struktúrájának felderítése kérdéses

Álpozitív függőségek

- Modelltömörítés (fejlesztőkörnyezetbeli memóriakorlát miatt)
  - Szerver oldalról érkező elemek kvalifikált neve elveszik → álpozitív függőségek jelenhetnek meg

  = függőségi analízis felülbecslést végez (létező függőséget nem hagyunk figyelmen kívül)
- Gyakorlatban ez nem probléma:
  - Ilyen esetekben a lekérdezés mindig végrehajtható a szerver oldalon is, ahol precíz eredményeket kapunk
  - A felhasználó ennek tudatában van
- A kliens oldali memóriakorlát feloldása esetén a tömörítés elhagyható

Szervezeti kérdések, követelmény-menedzsment eszközök

- CERN szervezeti sajátosságok
  - Cél: üzembiztos működés (konzervatív technológiai fejlődés)
  - Nagy méretű, elosztott, dinamikusan változó szervezet
    - Munkatársak gyakran változnak
    - Csapatok a „függőségeik mentén” kommunikálnak
    - Kis méretű magasszintű irányító csoport a globális célok meghatározásához
- A fentiekből következik
  - Összetett követelménykezelő rendszerek használata túl nagy „overheaddel” járna
  - CERN szoftverekre a „Unix szemlélet” jellemző: *do one thing but do it well*