

Unified Modeling Language

Szoftvertechnológia

Dr. Simon Balázs

BME, IIT

- Modellezés
- Unified Modeling Language (UML)
- UML diagrammok:
 - Use Case diagram
 - Aktivitásdiagram
 - Komponensdiagram
 - Telepítési diagram

Unified Modeling Language (UML)

Modellezés

- Egy modell a valóság egy egyszerűsített változata
- Egy jó modell hangsúlyozza a fontos részleteket és elhanyagolja az irreleváns részleteket
- Példa: Budapest metró térkép

- fontos részletek:
 - állomások nevei
 - állomások sorrendje
 - átszállási pontok
- irreleváns részletek:
 - a város utcái
 - állomások közötti távolság
 - az alagút szerkezeti felépítése

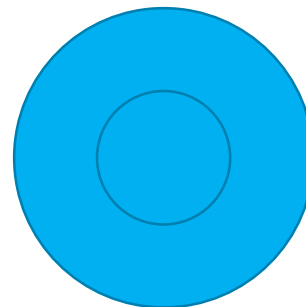
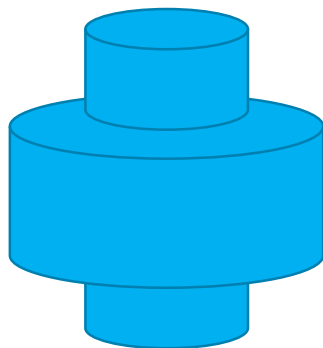


Egyéb modellezési példák

- A Naprendszer modellje:
 - fontos részletek: tömeg, sebesség, bolygók távolsága
 - irreleváns részletek: a bolygókat alkotó atomok
- Egy repülőgép modellje a szélcsatornában:
 - fontos részletek: alak, aerodinamika
 - irreleváns részletek: ülőhelyek száma, felszolgált étel, pilóta
- Egy repülőgép modellje helyfoglaláshoz:
 - fontos részletek: ülőhelyek száma, felszolgált étel
 - irreleváns részletek: alak, aerodinamika, pilóta

Nézetek

- A fontos és az irreleváns részletek függenek attól, hogy:
 - mire használjuk a modellt
 - ki fogja olvasni a modellt
- Minél több információt kell a modellnek lefednie, annál komplexebbé válik
- Néha ugyanaz a modell többféle célra is felhasználható és többféle célközönsége van
 - ilyenkor többféle nézetét is elkészíthetjük ugyanannak a modellnek
 - minden nézet az adott célnak és adott célközönségnek biztosítja a releváns részleteket
- Példa:
 - ugyanaz a modell különböző nézetekben:



- Unified Modeling Language
- Szabvány szoftverrendszerek modellezésére
 - az Object Management Group (OMG) által kiadott szabvány
- Az UML szabvány többféle diagramtípust definiál:
 - mindegyik diagram a rendszer egyfajta nézetét adja
- Az UML szabvány két dolgot ír le:
 - **szintaxis**: hogyan néznek ki az egyes diagramok (nézetek)
 - **szemantika**: mit jelentenek a diagramok egyes elemei
- Mind a szintaxis, mind pedig a szemantika fontos a modell megértéséhez
- A különböző diagramokból kiolvasható szemantikának konzisztensnek kell lennie a diagramok között, különben a modell ellentmondásos és használhatatlan

- Egy UML modellnek két fontos aspektusa van:
 - **Strukturális szemantika (statikus szemantika):** a modellezett rendszer egyes elemeinek jelentését definiálja egy adott időpillanatban
 - **Viselkedési szemantika (dinamikus szemantika):** a modellezett rendszer elemei jelentésének időbeli változását definiálja
- A különböző diagramtípusok a modell különböző aspektusait írják le

UML diagramok típusai

Strukturális UML diagramok:

Component diagram	Deployment diagram	Class diagram	Package diagram
Object diagram	Composite structure diagram	Profile diagram	

Viselkedési UML diagramok:

Use case diagram	Activity diagram	Sequence diagram	Communication diagram
State diagram	Timing diagram	Interaction overview diagram	

UML diagramok típusai – magyarul

Strukturális UML diagramok:

Komponens- diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

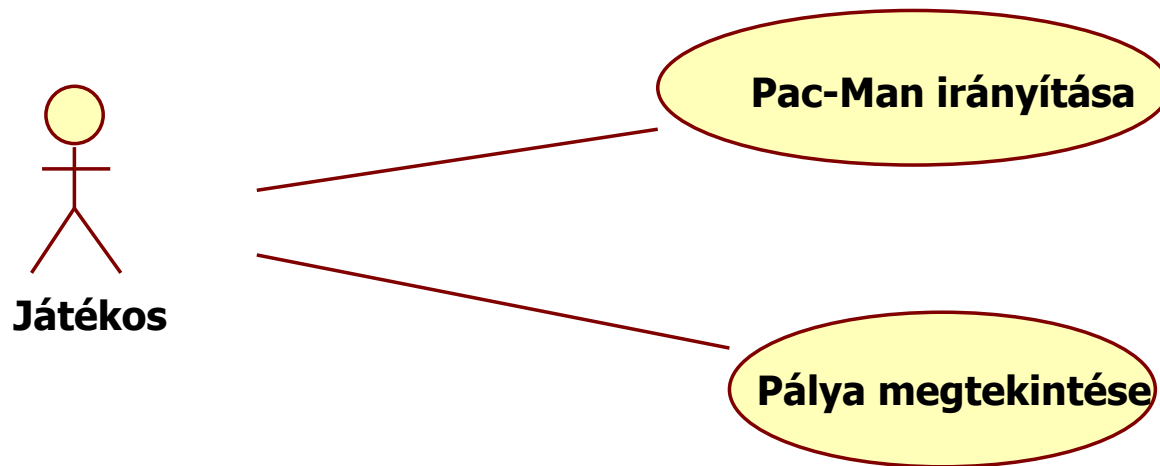
Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakció áttekintő diagram	

Use Case Diagram

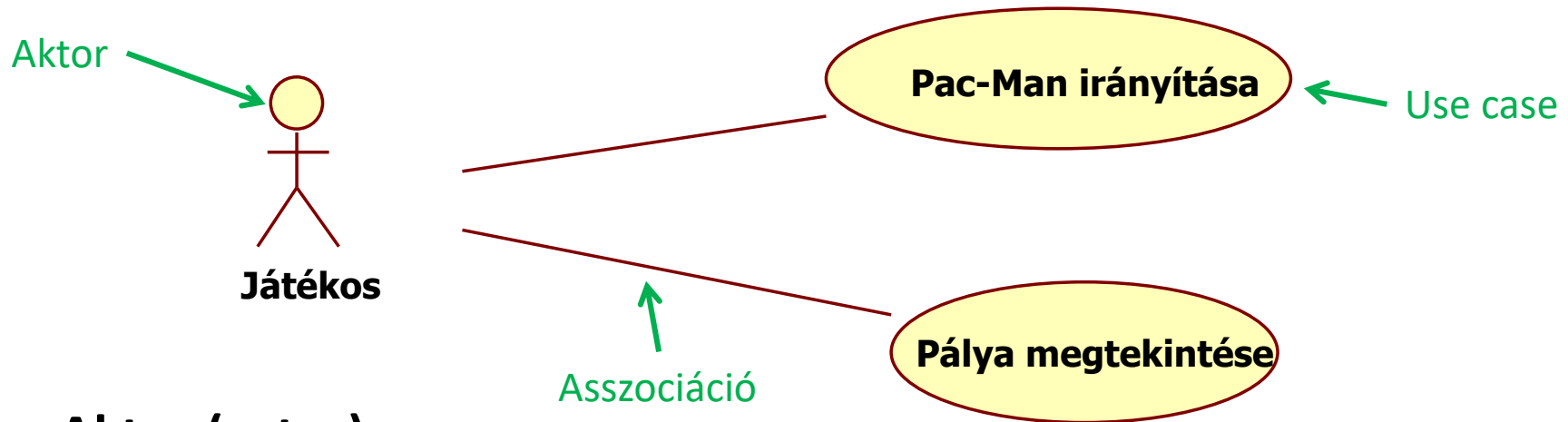
Use Case Diagram



- A rendszer *funkcionális* követelményeit reprezentálja
 - mit kellene a rendszernek csinálnia, hogyan fogják használni a rendszert
 - modellezi a rendszer felhasználóit és a rendszer határait
 - elég egyszerű ahhoz, hogy a megrendelő is megértse
- Példa: Pac-Man játék



Use Case Diagram



■ Aktor (actor):

- egy felhasználó szerepköre, aki a rendszerrel interakcióba lép
- lehet ember vagy külső rendszer is
- (egy fizikai ember többféle szerepkörben is használhatja a rendszert)

■ Használati eset (use case):

- a felhasználó és a rendszer közötti interakciót reprezentálja a felhasználó szemszögéből
- akciók és interakciók sorozata az aktorok és a rendszer között

■ Asszociáció (association):

- összekapcsolja az aktort azzal a use case-t, amelyikben részt vesz

Use case leírás példa: Vásárlás

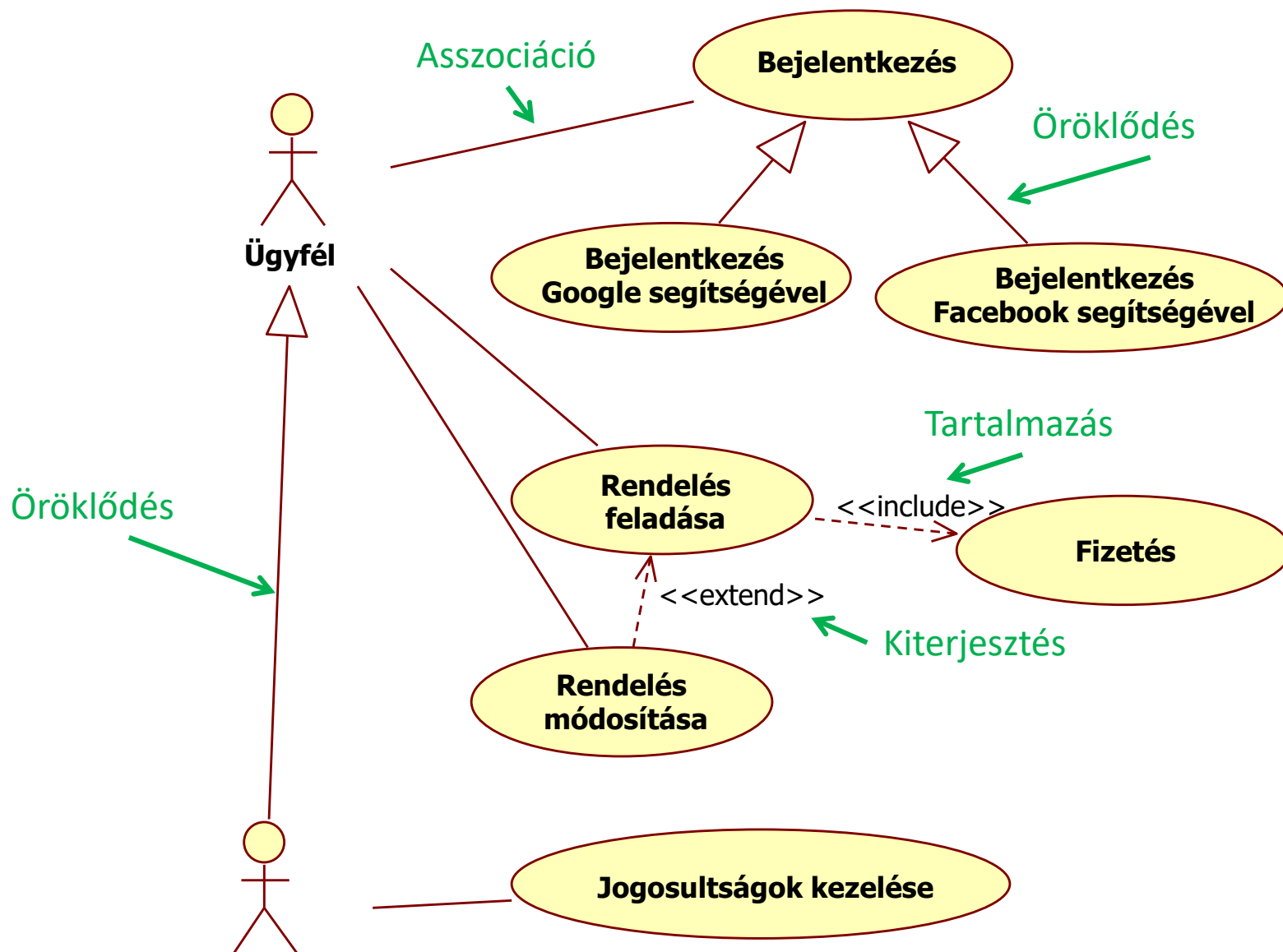


- **Use case:** Árucikk vásárlása
- **Aktorok:** Vásárló, Pénztáros
- **Főforgatókönyv:**
 - 1. A Vásárló kiválasztja az árucikket
 - 2. A Vásárló átadja az árucikket a Pénztárosnak
 - 3. A Pénztáros lehúzza az árucikket a pénztárgépen
 - 4. A Pénztáros elkéri a pénzt a Vásárlótól
 - 5. A Vásárló kifizeti az összeget a Pénztárosnak
 - 6. A Pénztáros visszaadja az árucikket és a blokkot a Vásárlónak
- **Alternatív forgatókönyv 5.A:**
 - 5.A.1. A Vásárlónak nincs elég pénze
 - 5.A.2. A Vásárló üres kézzel távozik

Use Case Diagram: Kapcsolatok

- Aktor és use case között:
 - **Asszociáció (association):**
 - összekapcsolja az aktort azzal a use case-t, amelyikben részt vesz
- Aktorok között:
 - **Öröklődés (generalization):**
 - a leszármazott aktor az őс aktor speciális változata
 - a leszármazott aktor mindazon use case-eket tudja, amiket az őс is
- Use case-ek között:
 - **Öröklődés (generalization):**
 - a leszármazott use case az őс use case speciális változata
 - a leszármazott use case *pontosítja* az őс use case működését
 - **<<extend>>:**
 - a kiterjesztő use case néhány egyéb lépéssel bővíti a kiterjesztett use case lépéseit meghatározott *kiterjesztési pontokon*
 - **<<include>>:**
 - a tartalmazó use case egy ponton „meghívja” a tartalmazott use case-t

Példák kapcsolatokra



Use case leírás sablon (nem része az UML-nek)

- **Cím:** a use case célja
 - tipikusan: ige + főnév
- **Aktorok:** a use case szereplői
- **Főforgatókönyv:** számozott lépések sorozata
 - lépés: <egy egyszerű, az aktor és a rendszer közötti interakciót leíró kijelentés>
- **Alternatív forgatókönyvek:** külön számozott listák, alternatívánként külön lista
 - alternatíva: <egy feltétel, amely a főforgatókönyvtől eltérő interakciót eredményez>
 - pl. a főforgatókönyv 7-es lépésétől elágazó alternatíva száma 7.A.

Use case leírás példa: Vásárlás



- **Use case:** Árucikk vásárlása
- **Aktorok:** Vásárló, Pénztáros
- **Főforgatókönyv:**
 - 1. A Vásárló kiválasztja az árucikket
 - 2. A Vásárló átadja az árucikket a Pénztárosnak
 - 3. A Pénztáros lehúzza az árucikket a pénztárgépen
 - 4. A Pénztáros elkéri a pénzt a Vásárlótól
 - 5. A Vásárló kifizeti az összeget a Pénztárosnak
 - 6. A Pénztáros visszaadja az árucikket és a blokkot a Vásárlónak
- **Alternatív forgatókönyv 5.A:**
 - 5.A.1. A Vásárlónak nincs elég pénze
 - 5.A.2. A Vásárló üres kézzel távozik

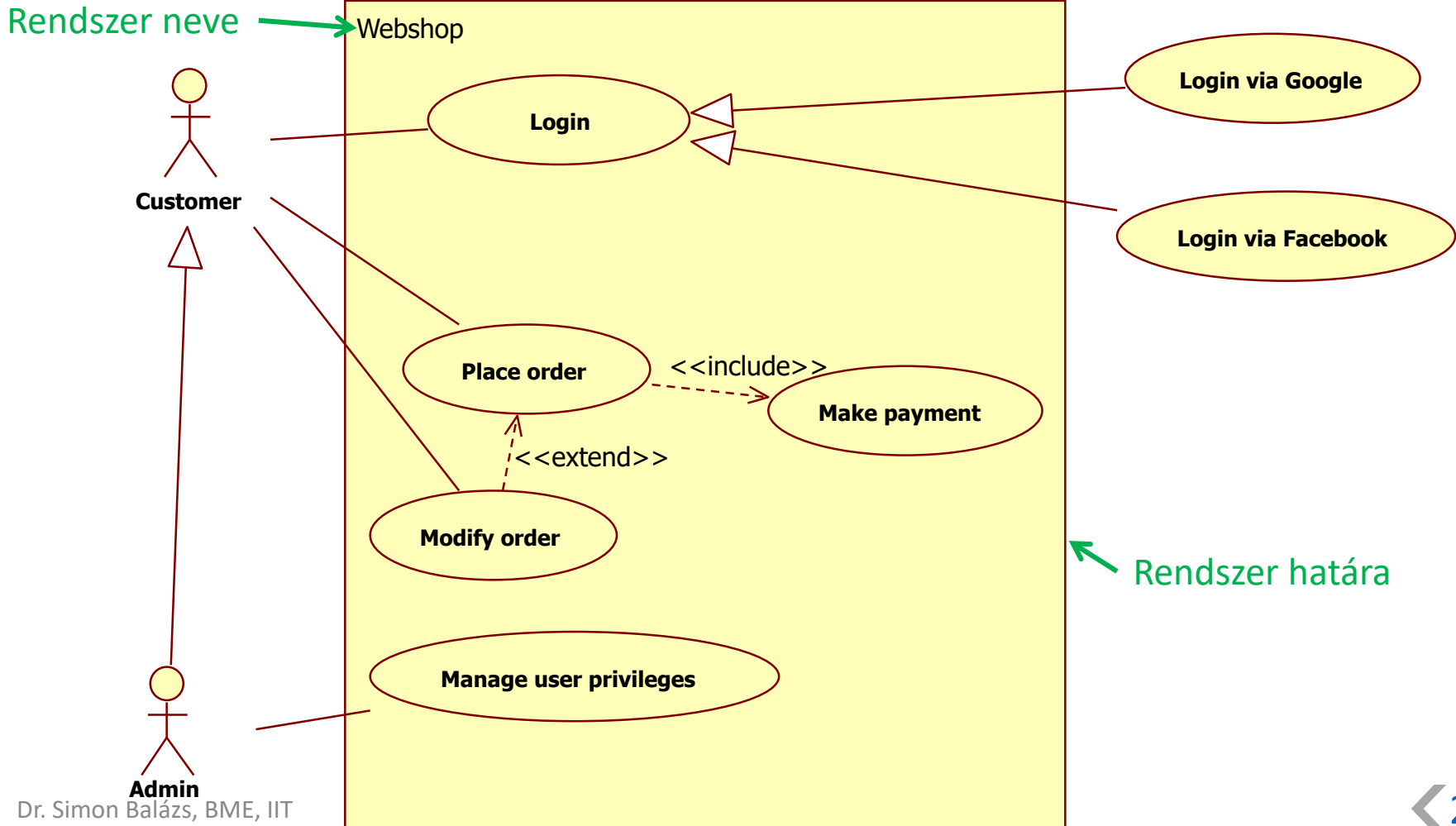
Részletesebb use case leírás

- A use case tartalmának leírására nincs szabvány
- Az előző diákon szereplő minták a leggyakoribb formái a use case-ek leírásának
- De sokkal több mező is lehet egy use case leírásban (Cockburn):
 - Cím, Elsődleges aktor, Cél, Határok, Szint, Megrendelők és érdekeik, Előfeltételek, Utófeltételek (minimális garanciák, sikerességi garanciák), Trigger, Főforgatókönyv, Kiterjesztések, Technológia
- Egyéb lehetséges mezők:
 - Egyedi azonosító, Használat gyakorisága, Kivételes esetek, Lefedett követelmények

Use Case Diagram: Rendszer határa (opcionális)

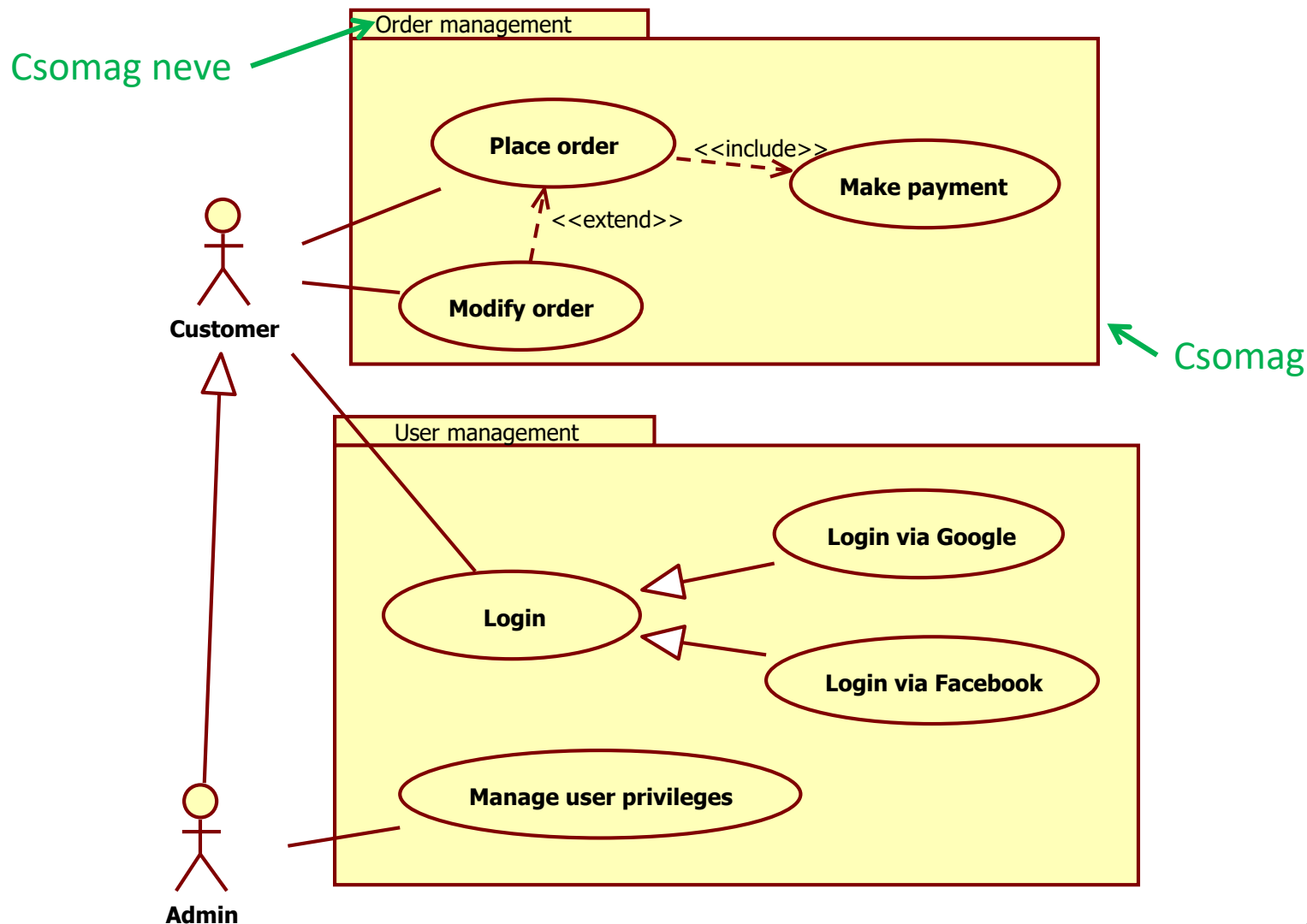
■ Rendszer határa (System boundary box):

- keret, amely a rendszer határát jelzi
- a kereten kívüli use case-ek nem részei a rendszernek



Use Case Diagram: Csomag (opcionális)

- **Csomag (package):** célja különböző elemek csoportosítása



UML diagramok típusai

Strukturális UML diagramok:

Komponens- diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildíagram	

Viselkedési UML diagramok:

Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakció áttekintő diagram	

Aktivitásdiagram (Activity Diagram)

Aktivitásdiagram



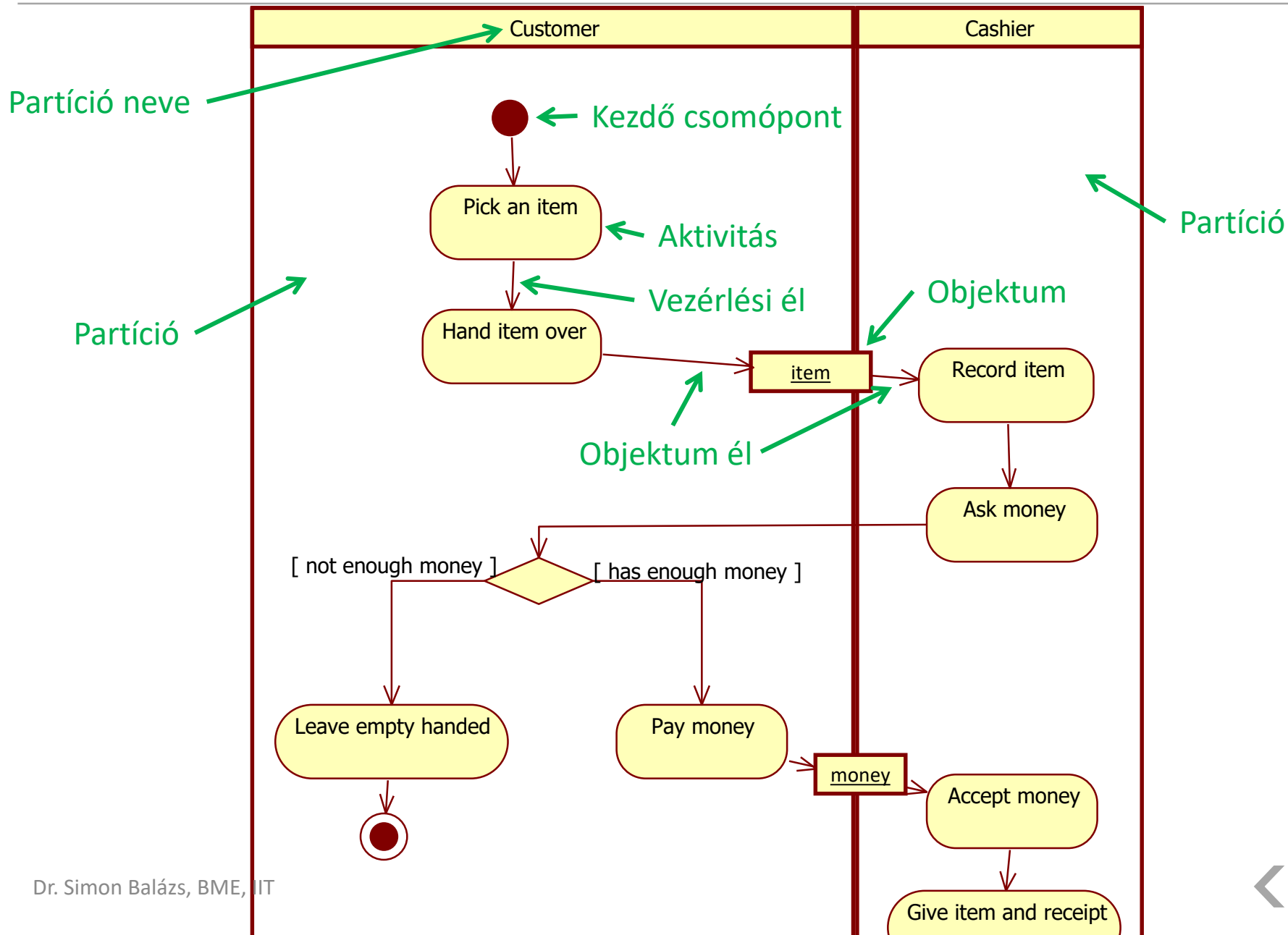
- Az aktivitásdiagramok munkafolyamatok lépéseit ábrázolják grafikusan
 - tevékenységek, döntések, ismétlések, párhuzamos működés
- Tipikusan a következők modellezésére használjuk őket:
 - egy adott use case leírása
 - egy a felhasználók és a rendszer közötti üzleti folyamat vagy munkafolyamat
 - egy algoritmus lépései
- Az aktivitásdiagramok a *funkcionális* követelmények munkafolyamatként történő formális leírását adják
- Az aktivitásdiagramok többszintűek lehetnek
 - pl. először egy magas absztrakciós szintű folyamat, majd később egy alacsonyabb absztrakciós szintű folyamat a lépések finomításával

Use case leírás példa emlékeztető: Vásárlás

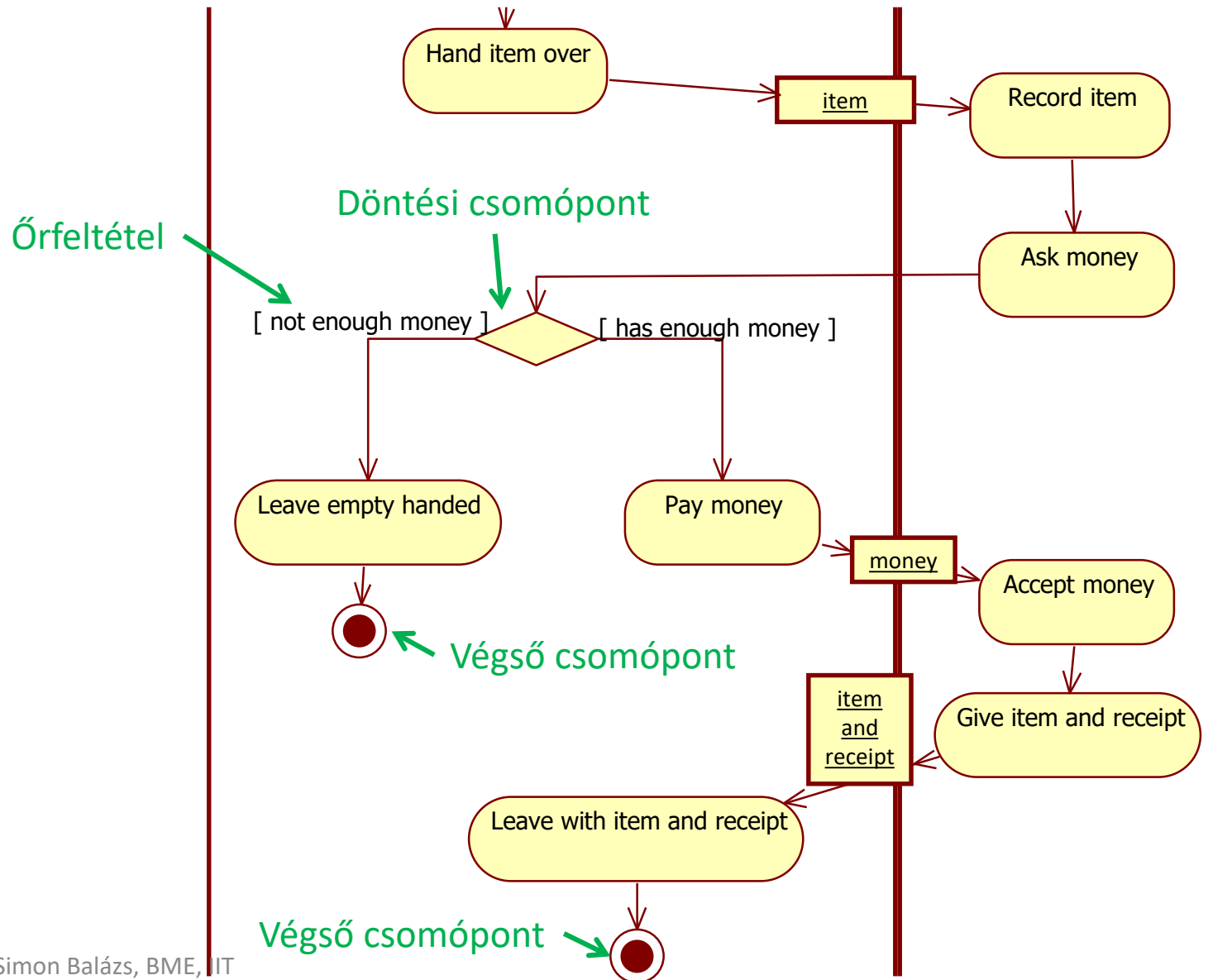


- **Use case:** Árucikk vásárlása
- **Aktorok:** Vásárló, Pénztáros
- **Főforgatókönyv:**
 - 1. A Vásárló kiválasztja az árucikket
 - 2. A Vásárló átadja az árucikket a Pénztárosnak
 - 3. A Pénztáros lehúzza az árucikket a pénztárgépen
 - 4. A Pénztáros elkéri a pénzt a Vásárlótól
 - 5. A Vásárló kifizeti az összeget a Pénztárosnak
 - 6. A Pénztáros visszaadja az árucikket és a blokkot a Vásárlónak
- **Alternatív forgatókönyv 5.A:**
 - 5.A.1. A Vásárlónak nincs elég pénze
 - 5.A.2. A Vásárló üres kézzel távozik

Aktivitásdiagram példa: Vásárlás use case I.



Aktivitásdiagram példa: Vásárlás use case II.



Aktivitásdiagram elemei

- **Aktivitás (activity):**
 - az aktivitásdiagram által leírt viselkedés egyes lépéseit modellezi
- **Kezdő csomópont (initial node):**
 - egy aktivitás futtatásának kezdeteként szolgál
 - nincsenek bejövő élei
 - egy aktivitásnak több kezdő csomópontja is lehet: ilyenkor több futás indul párhuzamosan
- **Vezérlési él (control flow):**
 - irányított kapcsolat két aktivitás között (forrás és cél)
 - a futás folyamatát jelöli: a cél aktivitás akkor indul, amikor a forrás aktivitás befejeződött
- **Végső csomópont (final node):**
 - a végső csomópontban a futás befejeződik
- **Döntés (decision):**
 - a kimenő vezérlési élek közül választ az őrfeltételeik alapján
 - legfeljebb egy kimenő él kerül végrehajtásra
- **Őrfeltétel (guard condition):**
 - egy vezérlési élhez tartozó feltétel
 - a vezérlési él csak akkor válhat aktívvá, ha az őrfeltétel igaz

Aktivitásdiagram elemei

■ **Partíció (partition):**

- egy felhasználó (szerepkör) vagy a rendszer egy része egy adott dimenzió mentén
- jelölése: egy sáv
- a sávban lévő aktivitásokat az adott felhasználó (szerepkör) vagy részrendszer hajtja végre
- a partíciók hierarchikusak is lehetnek: alpartíciók

■ **Partíció neve (partition name):**

- az adott partíció által reprezentált felhasználó (szerepkör) vagy részrendszer neve
- a sáv fejlécére van írva

■ **Objektum (object):**

- aktivitások között átadható adatot reprezentál

■ **Objektum él (object flow):**

- adatokat visz át aktivitások között

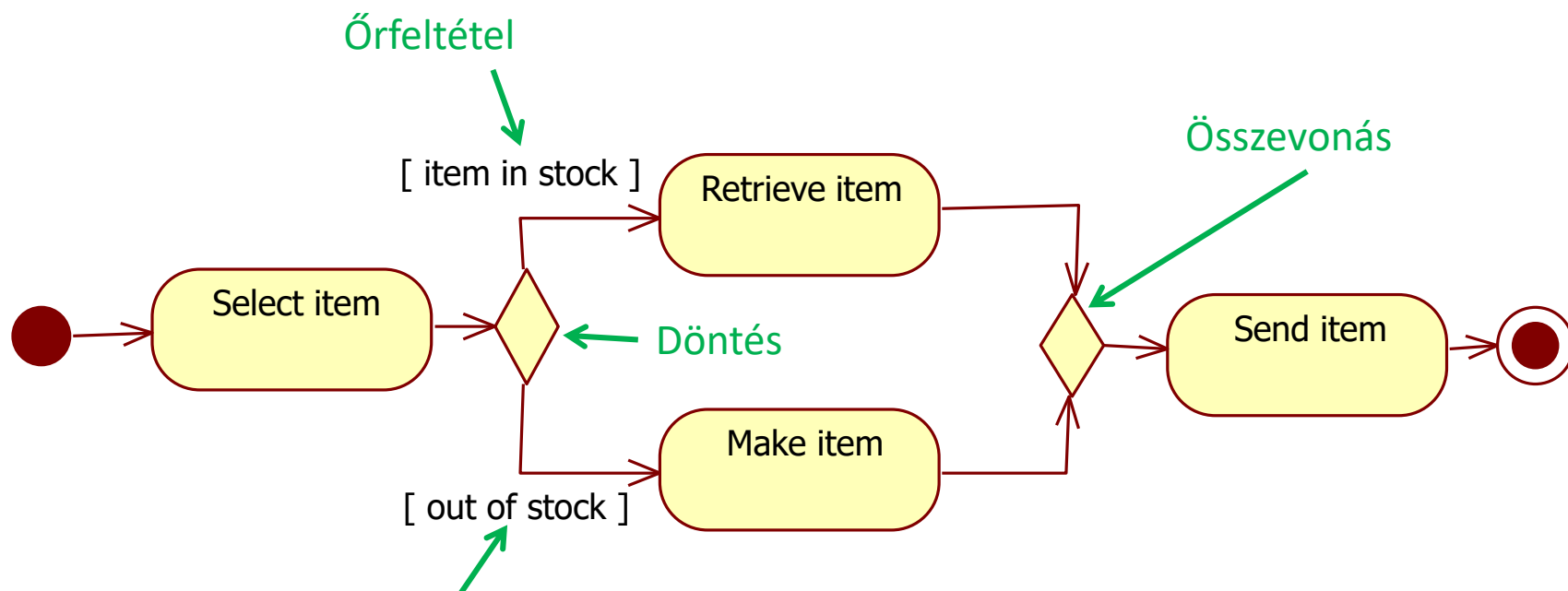
Aktivitásdiagram: döntés és összevonás

■ Döntés (decision):

- a kimenő vezérlési élek közül választ az őrfeltételeik alapján
- legfeljebb egy kimenő él kerül végrehajtásra

■ Összevonás (merge):

- több vezérlési ág összevonása szinkronizáció nélkül
- ha több beérkező ág is aktív, az összevonás aktivitás többször is lefut



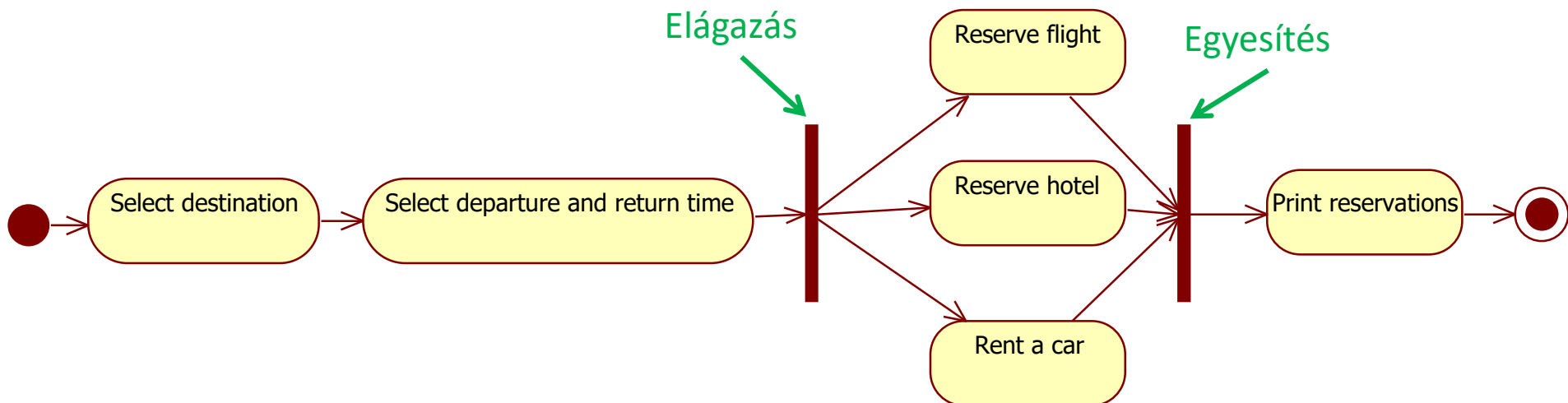
Aktivitásdiagram: elágazás és egyesítés

■ Elágazás (fork):

- a vezérlés több párhuzamosan futó ágra bomlik

■ Egyesítés (join):

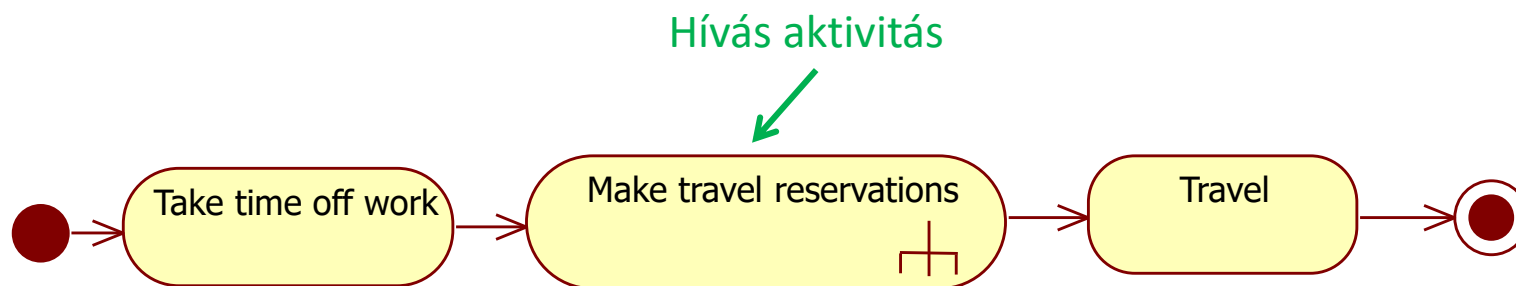
- szinkronizálja (bevárja) az összes beérkező ágat
- alapértelmezetten az összes beérkező ágnak be kell fejeződnie, hogy a vezérlés továbbléphessen (ez a viselkedés felülbíráható egy egyéni egyesítő kifejezéssel)



Aktivitásdiagram: hívás aktivitás

■ Hívás aktivitás (call activity):

- a hívás aktivitás meghív egy másik aktivitást
- a meghívott aktivitás viselkedését egy másik aktivitásdiagram is leírhatja
- azaz: aktivitásdiagramok egymásba ágyazhatók
- az aktivitásdiagramok így különböző szintű részletezettséget mutathatnak

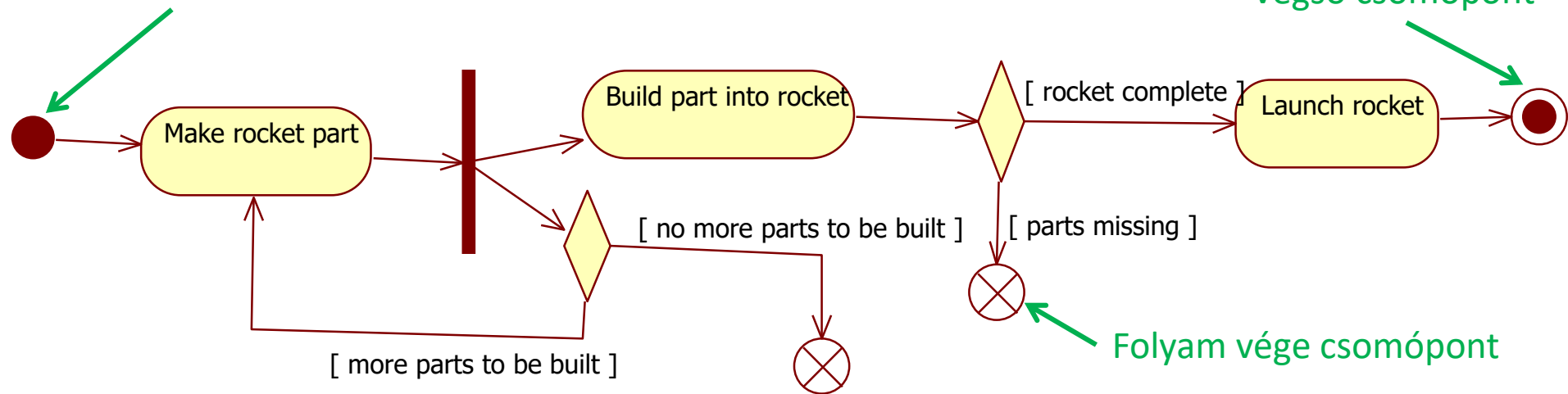


Aktivitásdiagram: végső csomópont

- Kétfajta végső csomópont létezik:
 - **Végső csomópont aktivitás (activity final node):** befejezi az aktivitásdiagram által leírt viselkedést, minden vezérlési ág befejeződik
 - **Folyam vége csomópont (flow final node):** csak az adott vezérlési ág fejeződik be, a többi vezérlési ág fut tovább

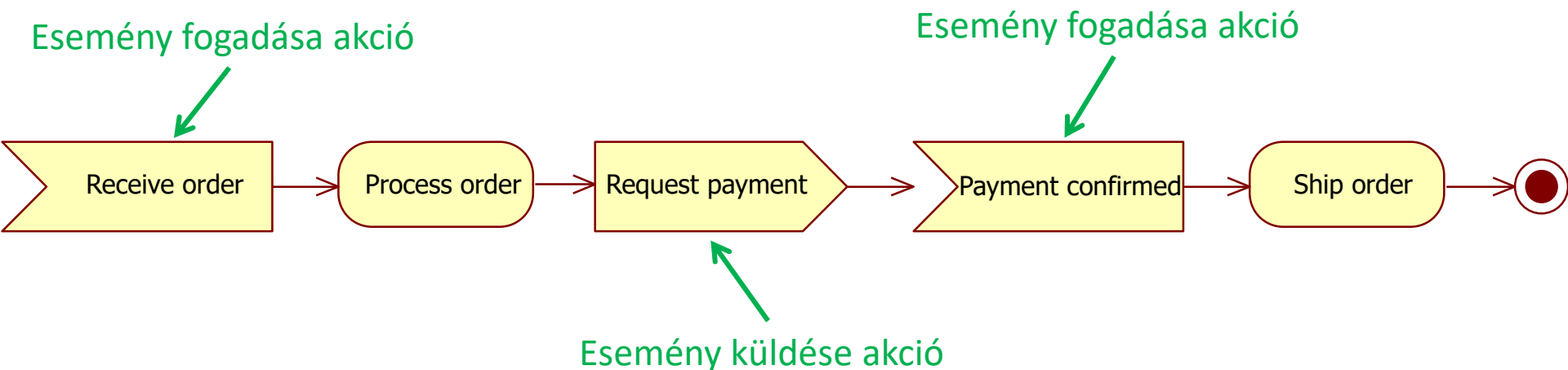
Kezdő csomópont

Végső csomópont



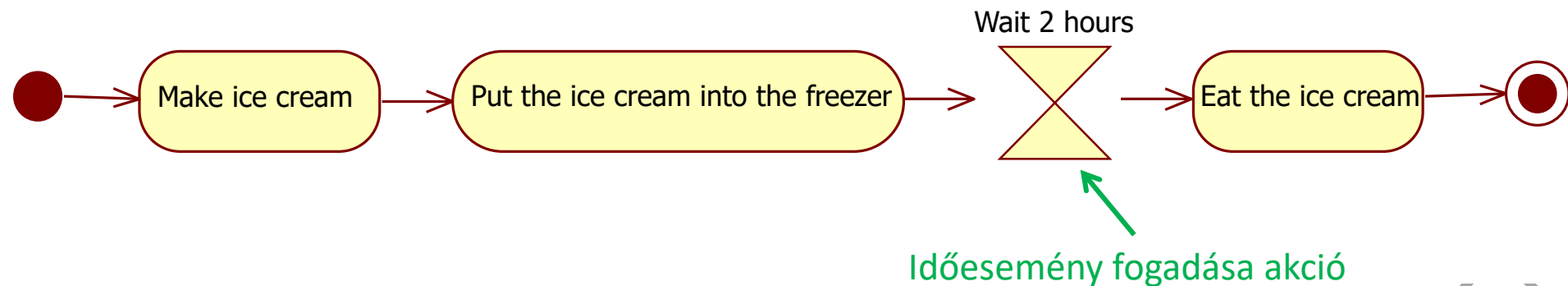
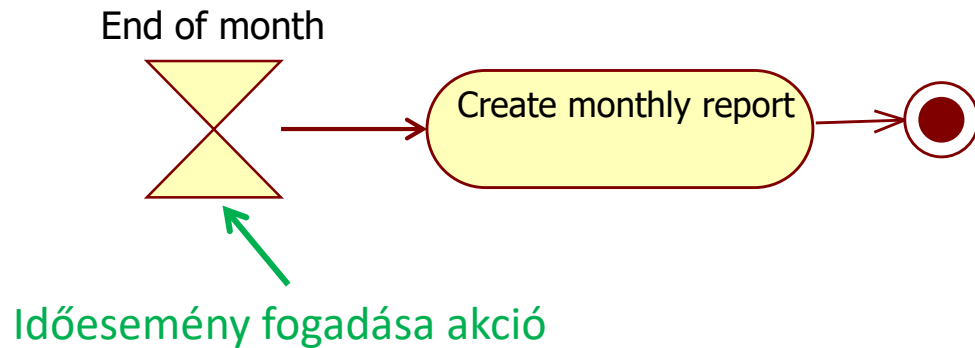
Aktivitásdiagram: jelzések

- Aktivitásokat események (jelzések) is elindíthatnak, nem csak a kezdeti csomópont
- Eseményeket az aktivitásdiagram elején és közepén is lehet fogadni
- A jelzések kezelésével kapcsolatos akciók:
 - **Esemény fogadása akció (accept event action):** megvárja, amíg az esemény (jelzés) bekövetkezik, majd elindítja a vezérlés futását
 - **Esemény küldése akció (send signal action):** jelzést küld, amely egy esemény fogadása akcióval elkapható



Aktivitásdiagram: időzíti események

- **Időesemény fogadása akció (accept time event action):**
elindítja a vezérlést egy adott időpontban
- Példák:



UML diagramok típusai

Strukturális UML diagramok:

Komponens- diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildíagram	

Viselkedési UML diagramok:

Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakció áttekintő diagram	

Komponentsdiagram (Component Diagram)

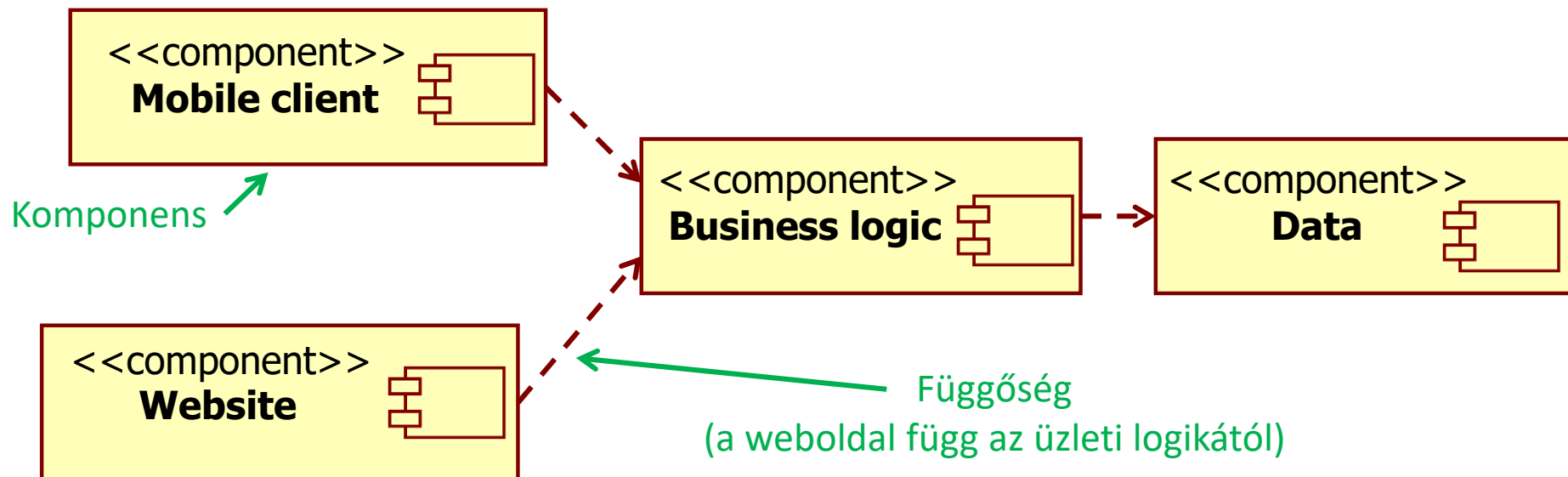
Komponensdiagram



- Egy szoftverkomponens a kompozíció alapegysége, szerződészerűen specifikált interfészekkel rendelkezik és csak explicit módon függ környezetétől. Egy szoftverkomponens önállóan telepíthető és harmadik fél által kompozíciós egységként felhasználható. (Szyperski)
- A komponensdiagram egy rendszer komponenseit és a közöttük lévő kapcsolatokat mutatja
- A komponensdiagram tipikusan tervezés közben készül, de finomítható telepítés és futtatás során

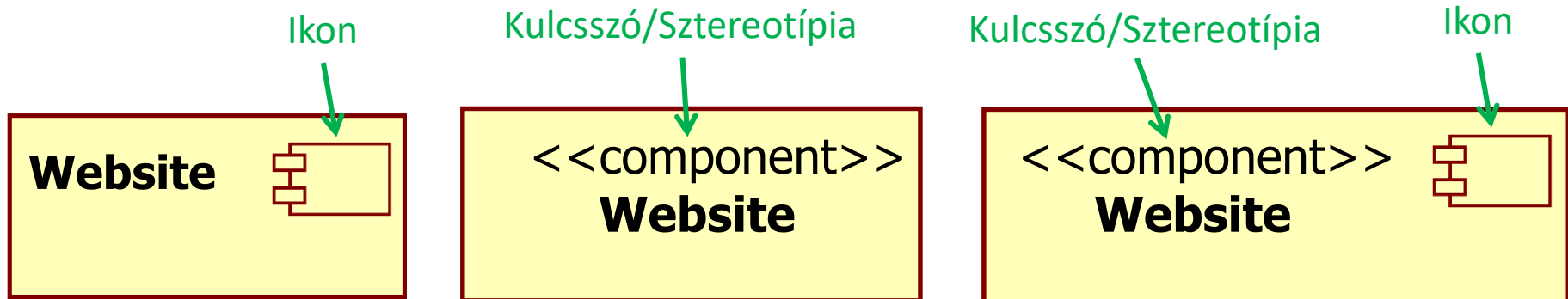
Komponensdiagram példa: Mobil+webes alkalmazás

- Az áttekintő komponensdiagram a komponenseket (component) és a közöttük lévő függőségeket mutatja
- **Függőség (dependency):** “A” függ “B”-től azt jelenti, hogy ha “B” változik, akkor a változás kihathat “A”-ra is
- A függőség fajtája nincs jelölve
- A függőséget jelentő nyíl hiánya egyben a függőség hiányát is jelzi



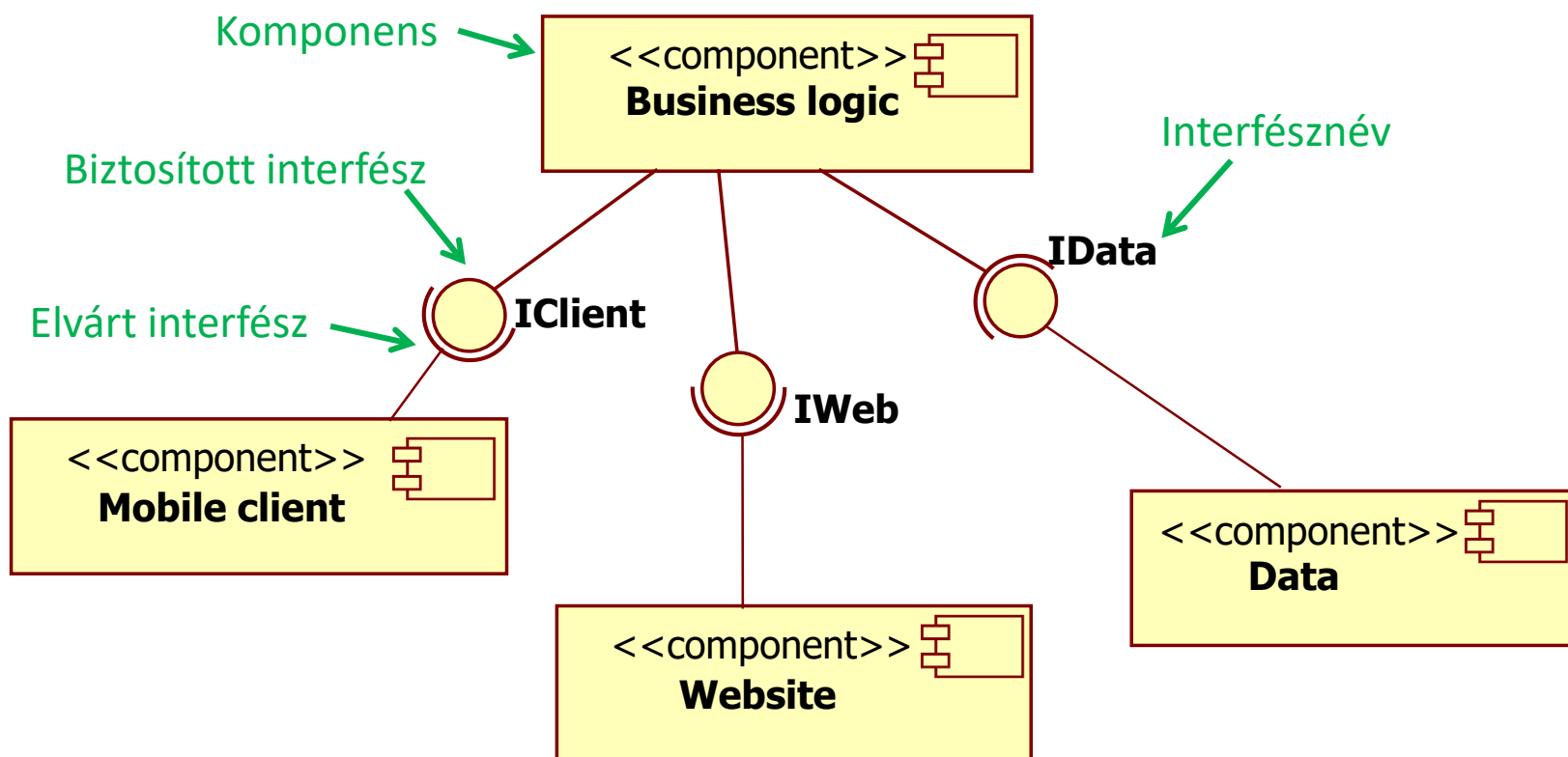
Komponens jelölése

- Három lehetséges jelölési mód (ekvivalensek egymással):
 - téglalap ikonnal
 - téglalap *component* kulcsszóval/sztereotípiával (keyword/stereotype)
 - téglalap ikonnal és *component* kulcsszóval/sztereotípiával



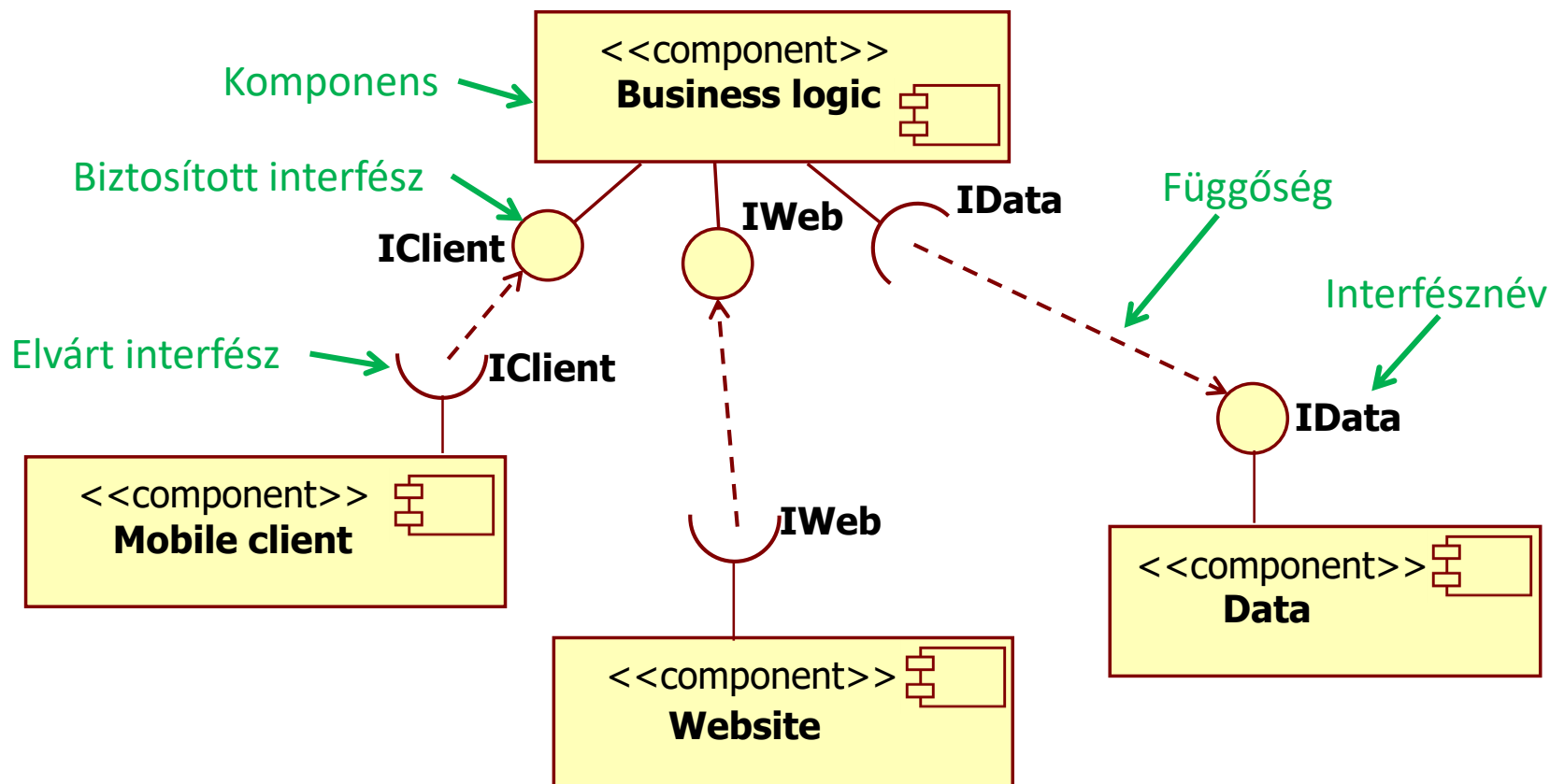
Komponensdiagram példa: Mobil+webes alkalmazás

- Egy részletesebb komponensdiagramon ábrázolhatók a komponensek a biztosított (provided) és elvárt (expected) interfészeikkel, valamint a közöttük lévő kapcsolatokkal



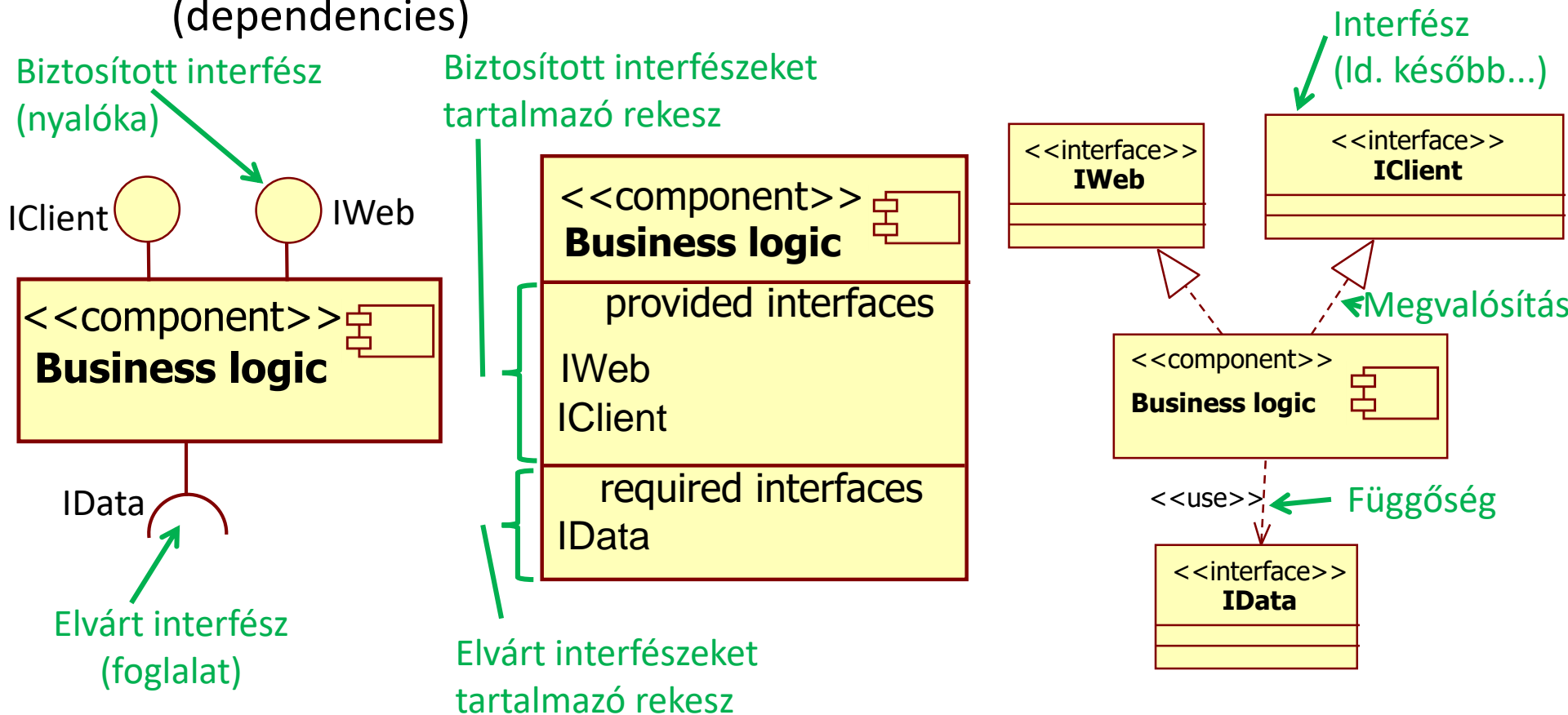
Komponensdiagram példa: Mobil+webes alkalmazás

- A biztosított és elvárt interfészek függőség (dependency) segítségével is összekapcsolhatók



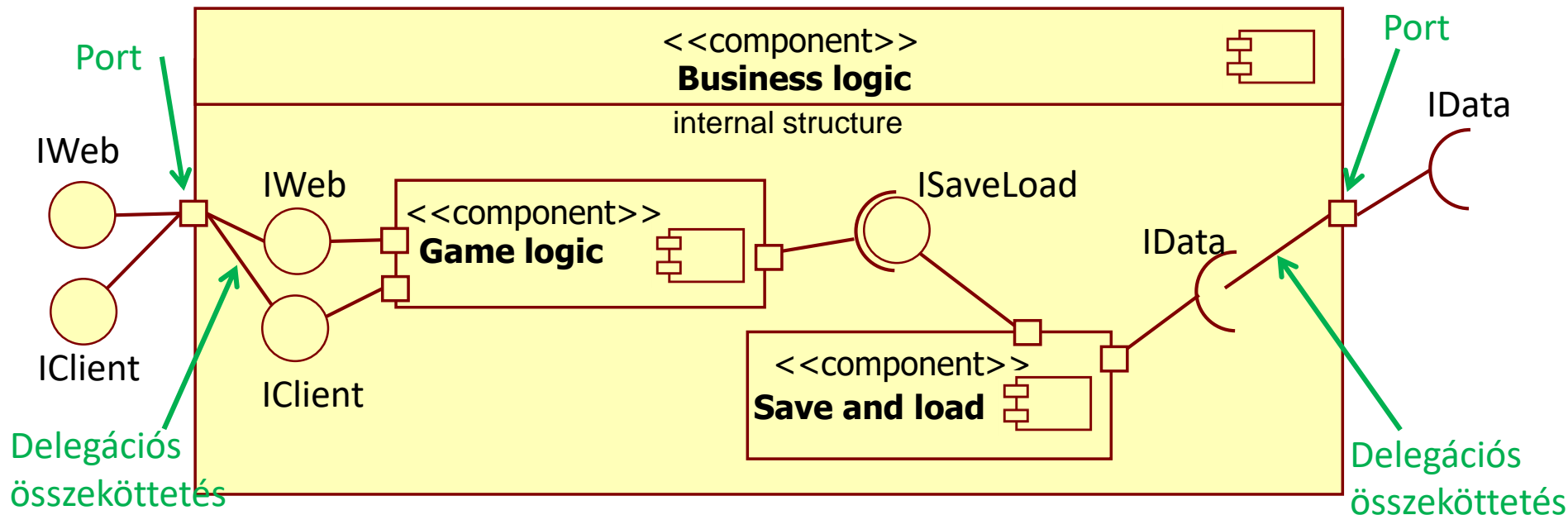
Biztosított és elvárt interfészek

- Három lehetséges jelölés (ekvivalensek egymással):
 - komponens nyalókákkal (lollipops) és foglalatokkal (sockets)
 - komponens rekeszekkel (compartments)
 - komponens megvalósításokkal (realizations) és függőségekkel (dependencies)



Összetett komponens (composite component)

- Az összetett komponenst más komponensek implementálják
- A külsőleg biztosított és elvárt interfészek portokon keresztül vannak kivezetve
- Delegációs összeköttetések (delegation connectors) kapcsolják össze a külsőleg biztosított és elvárt interfészeket és a belső komponenseket, amelyek megvalósítják vagy elvárják azokat



UML diagramok típusai

Strukturális UML diagramok:

Komponens- diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildíagram	

Viselkedési UML diagramok:

Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakció áttekintő diagram	

Telepítési diagram (Deployment Diagram)

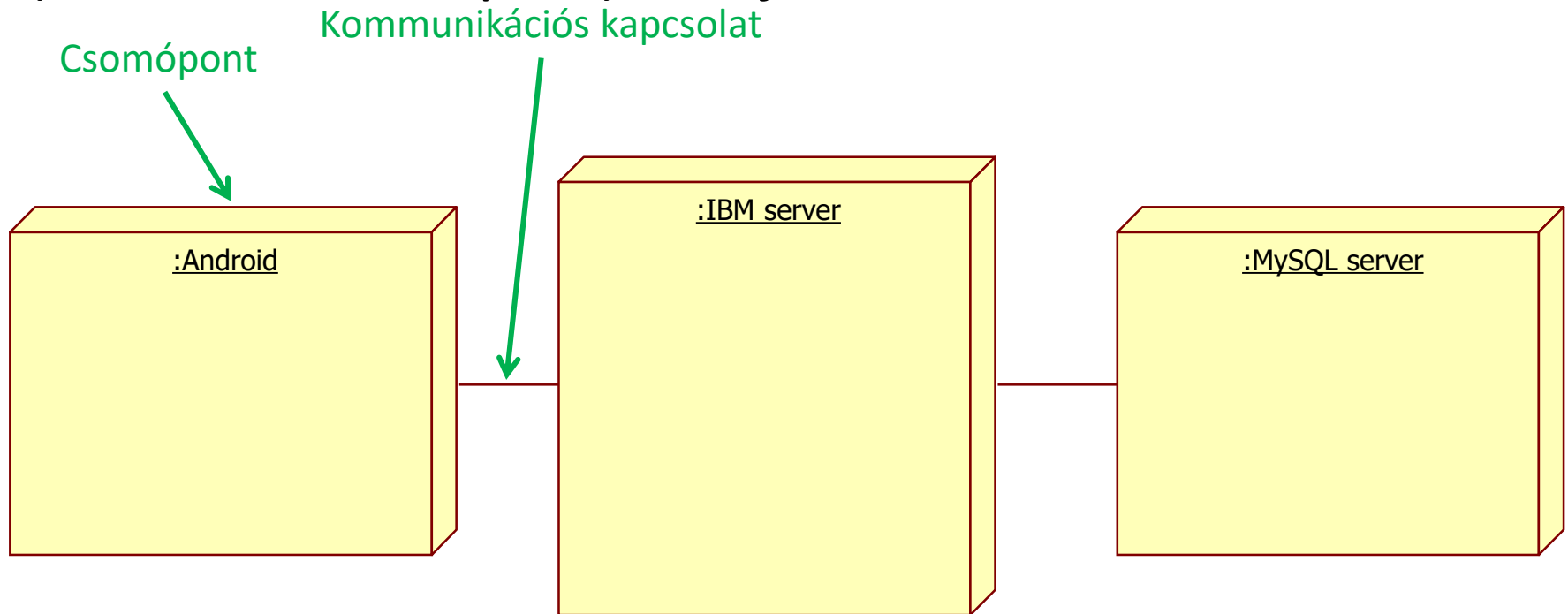
Telepítési diagram



- A telepítési diagram a rendszer futási architektúráját és szoftver artifact-ek rendszerelemekhez való rendelését ábrázolja
- A telepítési diagram segítségével a rendszer hardveres és szoftveres topológiája is ábrázolható
- A telepítési diagramot tipikusan átadáskor használjuk a telepítési architektúra megtervezésére

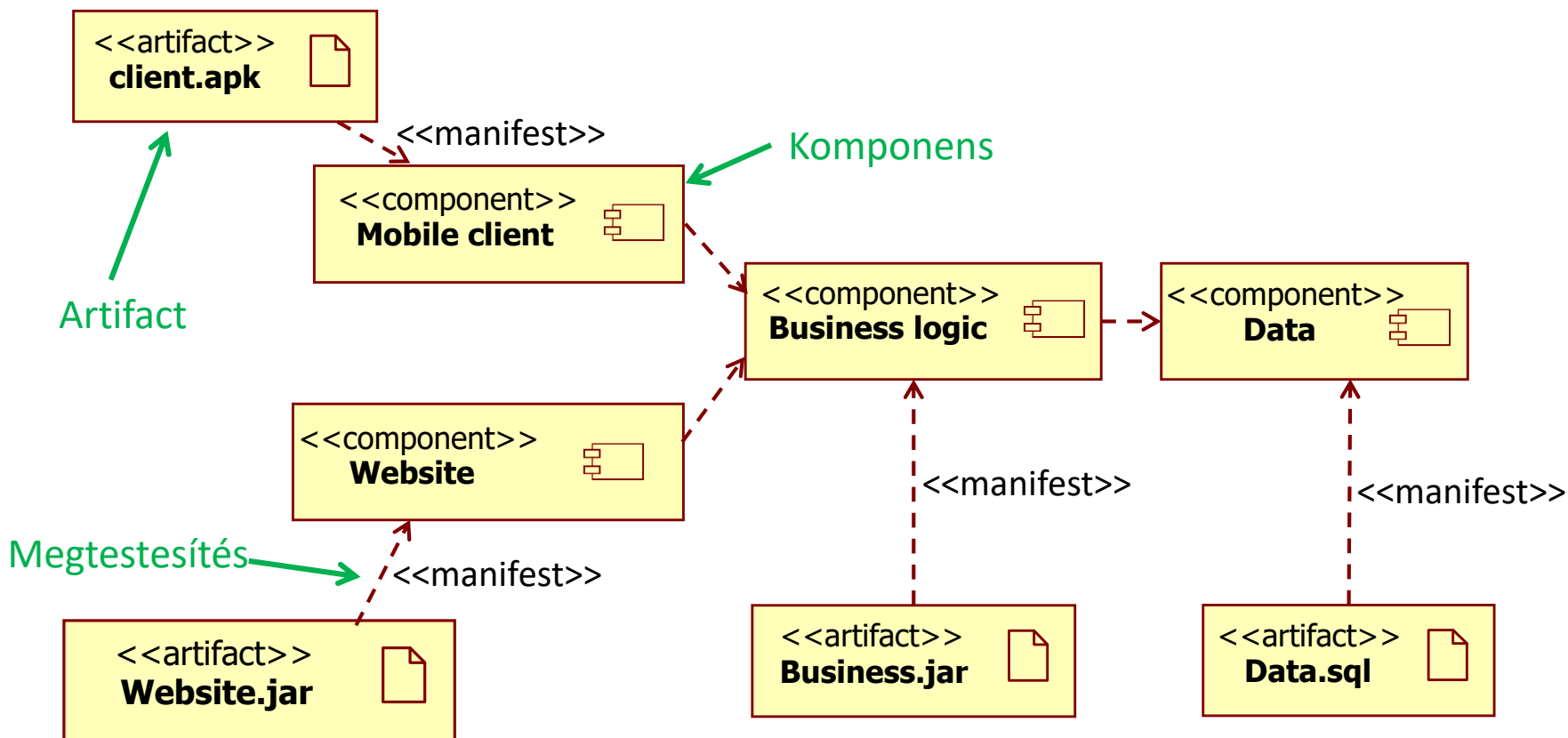
Telepítési példa: Mobil+webes alkalmazás

- A rendszer architektúráját csomópontokkal (nodes) és a közöttük lévő kommunikációs kapcsolatokkal (communication path) írhatjuk le



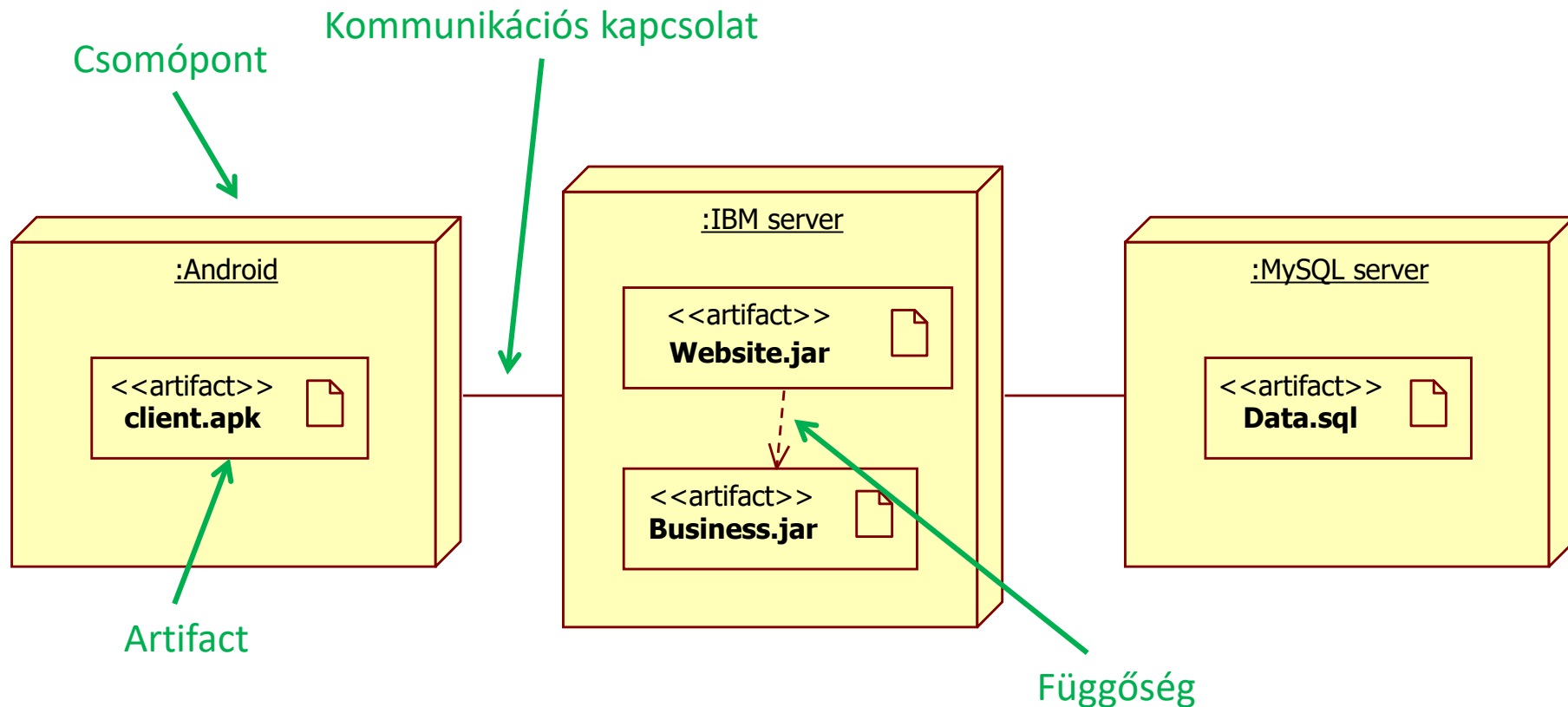
Artifact példa: Mobil+webes alkalmazás

- Egy artifact olyan dolgot reprezentál, amelyet vagy a szoftverfejlesztési folyamat vagy pedig a rendszer működése állít elő vagy használ fel
 - pl. futtatható fájl, szkript, adatbázis tábla, dokumentum
- Egy artifact egy komponens megtestesítését is jelentheti



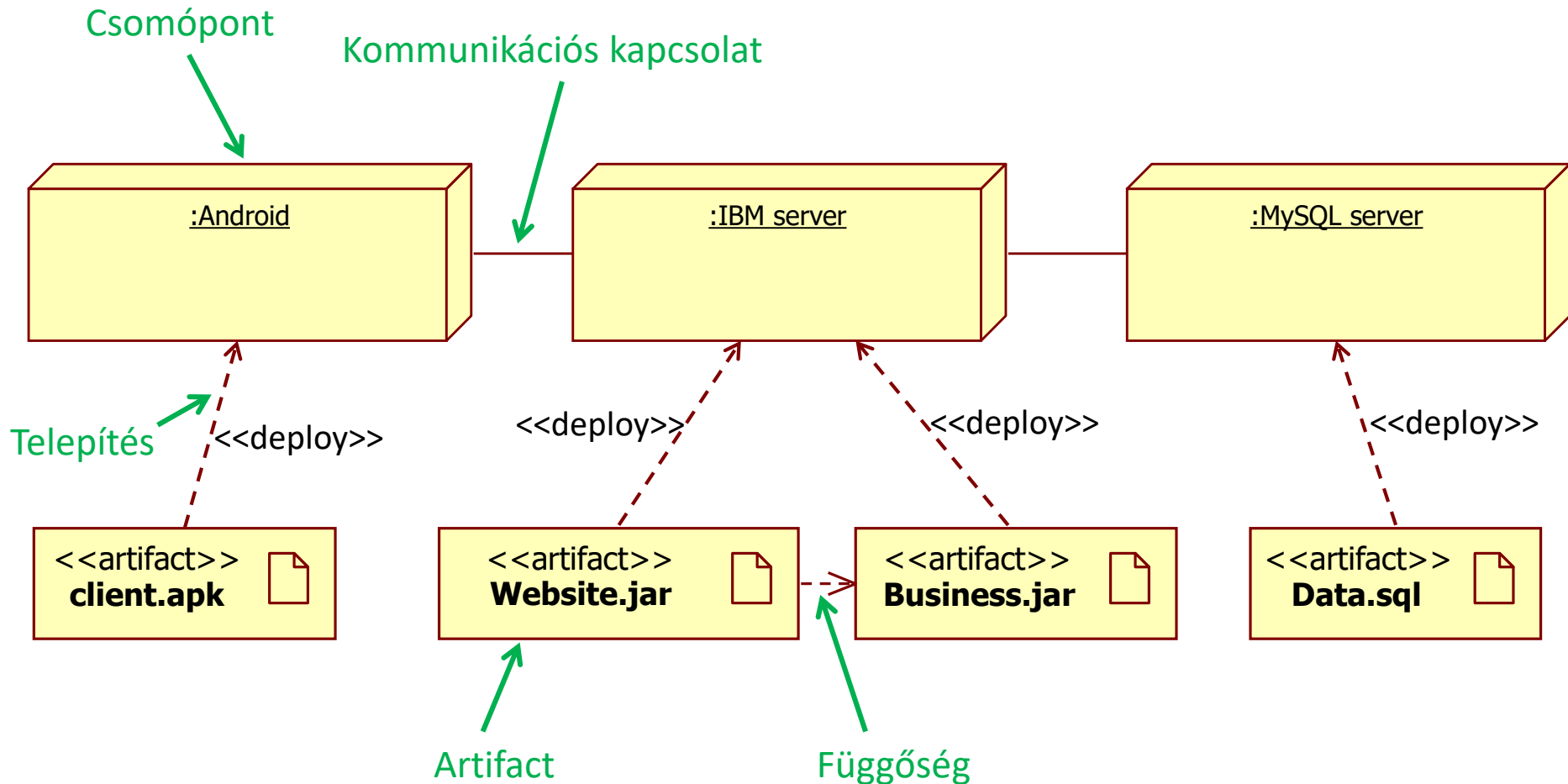
Telepítési példa: Mobil+webes alkalmazás

- Artifact-eket csomópontokra lehet telepíteni
- Függőségek artifact-ek között is lehetnek



Telepítési példa: Mobil+webes alkalmazás

- A telepítendő artifact-okat deploy kulcsszóval/sztereotípiával ellátott függőség segítségével is lehet csomóponthoz rendelni
 - (ez a diagram ekvivalens az előző dián szereplővel)



UML diagramok típusai

Strukturális UML diagramok:

Komponens- diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakció áttekintő diagram	

Összefoglalás

- Modellezés
- Unified Modeling Language (UML)
- UML diagrammok:
 - Use Case diagram
 - Aktivitásdiagram
 - Komponensdiagram
 - Telepítési diagram

UML diagramok típusai

Strukturális UML diagramok:

Komponens- diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildíagram	

Viselkedési UML diagramok:

Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakció áttekintő diagram	