Unified Modeling Language

Szoftvertechnológia

Dr. Simon Balázs

BME, IIT

Tartalom

- UML diagramok:
  - Állapotdiagram
  - Időzítődiagram
  - Összetett struktúra diagram
  - Profildiagram
- Az UML diagramok összefoglalása
- Az UML-en túl:
  - Object Constraint Language (OCL)
  - XML Metadata Interchange (XMI)
  - MetaObject Facility (MOF)

Hol tartunk?

Use Case diagram ← A funkcionális követelmények magas szintű leírása: A use case-ek alapvető interakciós sorozatok a rendszer és a felhasználói között

Aktivitásdiagram ← Use case interakciós sorozatok munkafolyamatként ábrázolva

Komponens-diagram ← Áttekintő kép a rendszer architektúrájáról: A komponensek és kapcsolataik

Hova kell telepíteni a komponenseket → Telepítési diagram

Hol tartunk?

Osztály-diagram ← Egy komponens/rendszer struktúrája objektumorientált modellként

Csomag-diagram ← Csomagok és a közöttük lévő függőségek

Objektum-diagram ← A rendszer részletes állapotának ábrázolása egy adott időpillanatban

Hol tartunk?

Szekvencia-diagram	Az üzenetváltások lehetséges sorozata időben rendezve
Kommunikációs-diagram	Az interakciók áttekintő képe (résztvevők és kapcsolataik)
Interakciós áttekintő diagram	Interakciók ábrázolása az Aktivitásdiagramok egy változataként, amely a vezérlés futásáról ad egy áttekintő képet

Most következik:
Hogyan írjuk le egy komponens/osztály belső állapotát?

Hol tartunk?

Most következik:
Hogyan írjuk le egy komponens/osztály belső állapotát?

Strukturális UML diagramok:

Komponens-diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

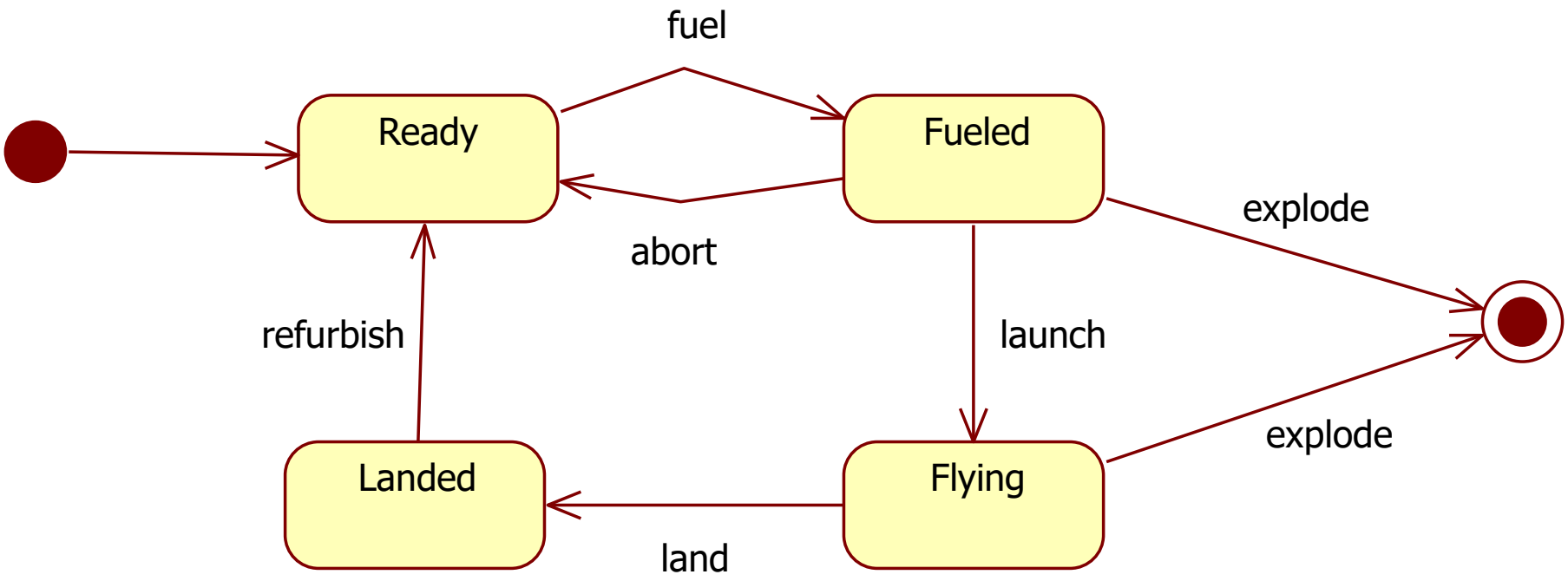
Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakciós áttekintő diagram	

Állapotdiagram (State Machine Diagram)

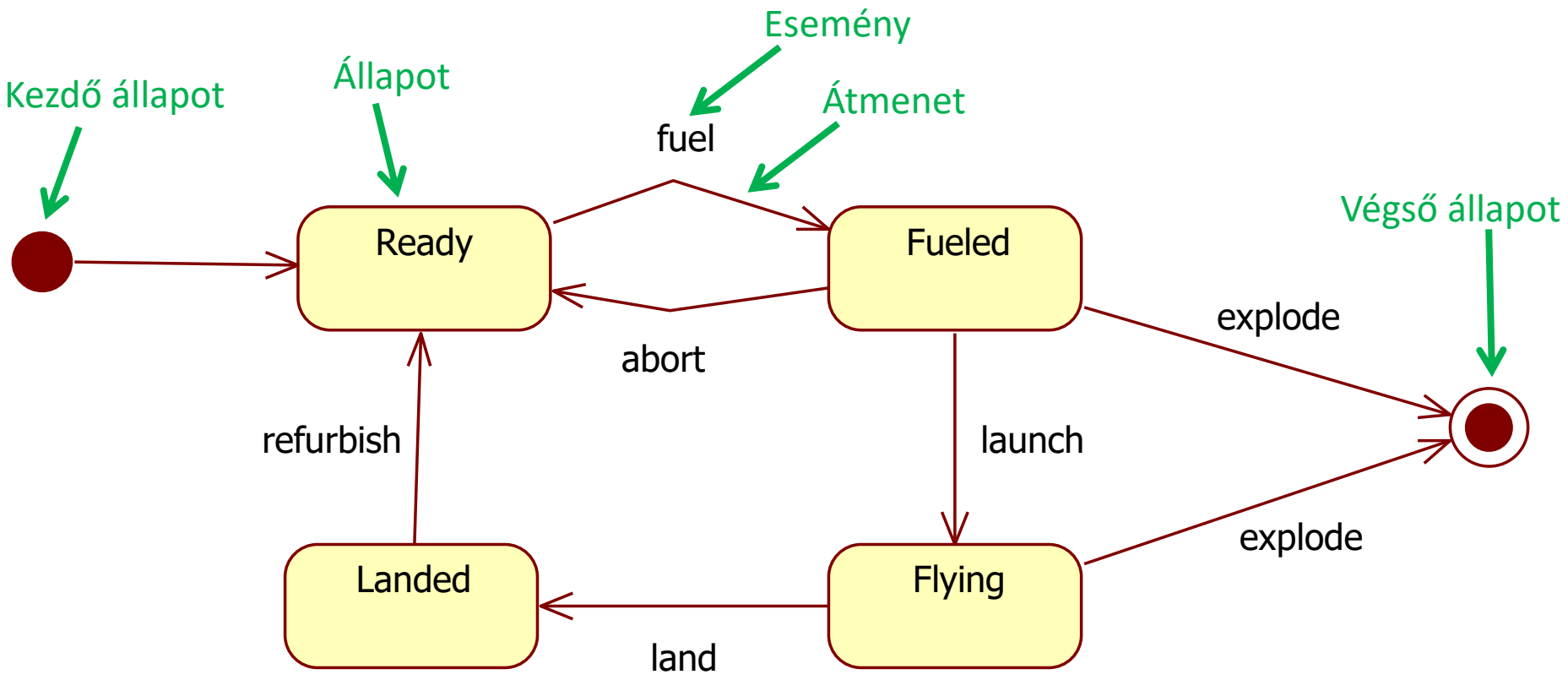
Állapotdiagram

- Diszkrét eseményvezérelt viselkedést ír le véges automaták segítségével formalizálva
- Az állapotok az objektum által tárolt információk különböző kombinációit jelentik
- Az állapotgépek az objektumok lehetséges állapotait ábrázolják, és azt, hogy az objektum hogyan juthat el ezekbe az állapotokba
  - egy objektum állapota akkor változik, ha esemény (függvényhívás) érkezik
- Állapotgépek fajtái:
  - viselkedési állapotgép: egy rendszer részeinek állapotát fejezi ki (pl. osztályok, komponensek viselkedése)
  - protokoll állapotgép: érvényes interakciók sorozatát fejezi ki
- (A következőkben: viselkedési állapotgép. A protokoll állapotgépet ld. a szabványban.)

Állapotdiagram példa: újrahasznosítható rakéta



Állapotdiagram példa: újrahasznosítható rakéta

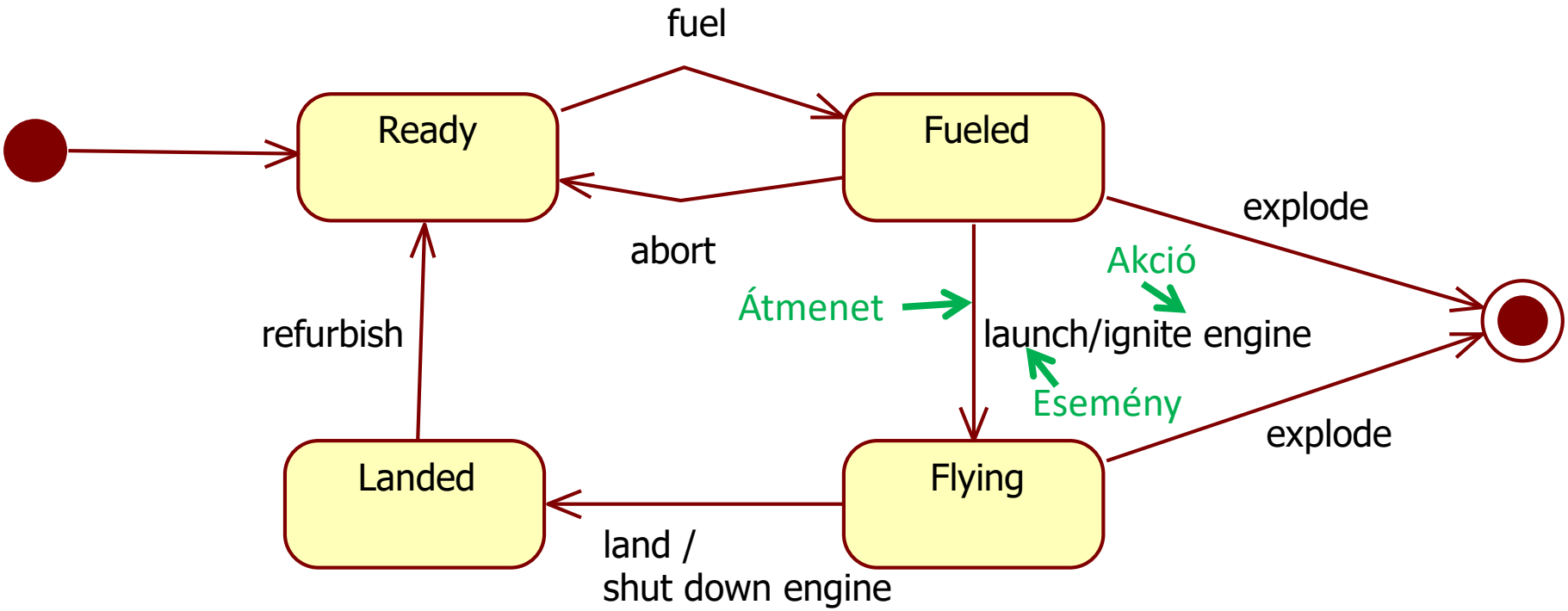


Állapotdiagram

- **Kezdő állapot (initial state)**:
  - a futás a kezdőállapotból kiinduló átmenettel indul
- **Végső állapot (final state)**:
  - speciális állapot, az objektum futásának végét jelzi
- **Állapot (state)**:
  - a futás egy olyan helyzetét reprezentálja, amikor egy invariáns feltétel teljesül
    - tehát az objektum attribútumainak értékei valamilyen feltételt teljesítenek
  - nem feltétlenül teljesen statikus helyzetet jelent
    - az állapotot meghatározó attribútumok értékei változhatnak, feltéve, hogy továbbra is teljesül az invariáns feltétel
- **Átmenet (transition)**:
  - egy lehetséges mozgást ábrázol a forrásállapotból a célállapotba, ha a meghatározott esemény bekövetkezik
  - a nyíl mellé írt címke mutatja az eseményt
- **Esemény (event)**:
  - ha az esemény bekövetkezik, az aktuális állapot átvált annak az átmenetnek célállapotára, amelynek forrásállapota az aktuális állapot, eseménye pedig a bekövetkezett esemény
  - ha több lehetséges átmenet is van, csak egy állapotváltás történik nem-determinisztikusan
  - ha nincs lehetséges átmenet, nem történik állapotváltás
  - egy esemény tipikusan az objektumon hívott metódus (üzenet)

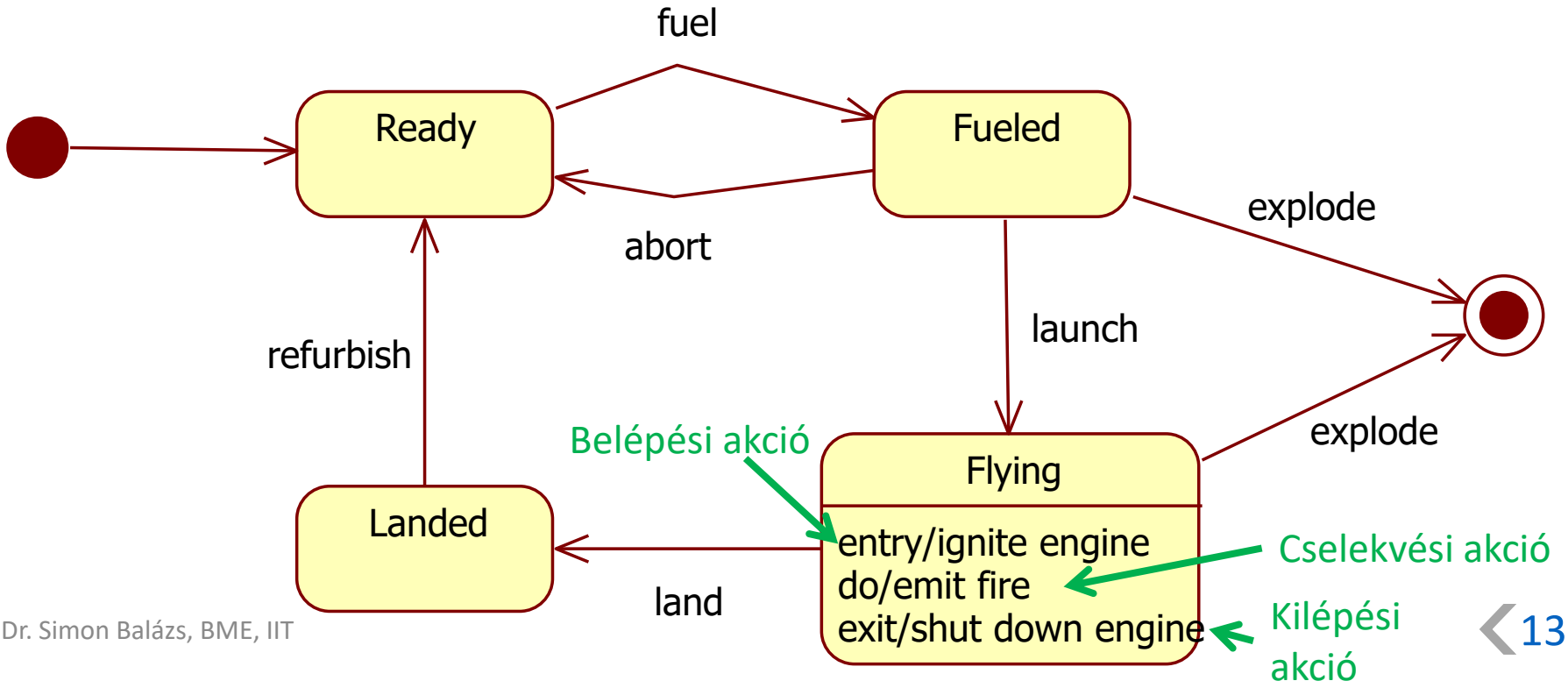
Állapotdiagram: Akciók (actions)

- Az állapotgép az eseményekre akciók végrehajtásával reagálhat
  - pl. egy változó értékének megváltoztatása, I/O művelet, függvény meghívása, másik esemény generálása



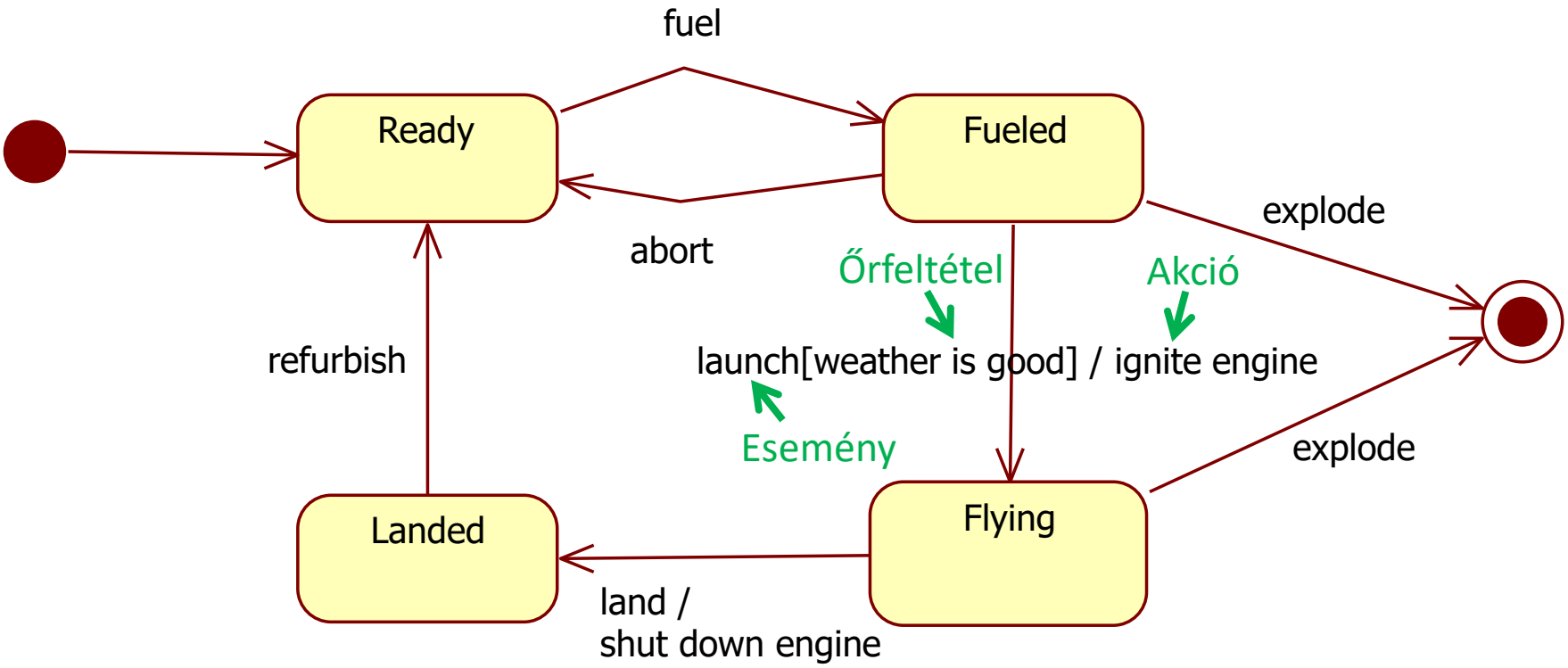
Állapotdiagram: Akciók (actions)

- Állapoton belül is lehet akció:
  - **Belépési akció (entry action)**: akkor fut le, ha az állapotba egy külső állapotátmenettel lépünk be
  - **Kilépési akció (exit action)**: akkor fut le, ha kilépünk az állapotból
  - **Cselekvési akció (do action)**: a végrehajtása akkor kezdődik, miután beléptünk az állapotba (a belépési akció után), és mindaddig fut, amíg véget nem ér, vagy ki nem lépünk az állapotból
- Egy állapotnak több belépési-, kilépési- és cselekvési akciója is lehet



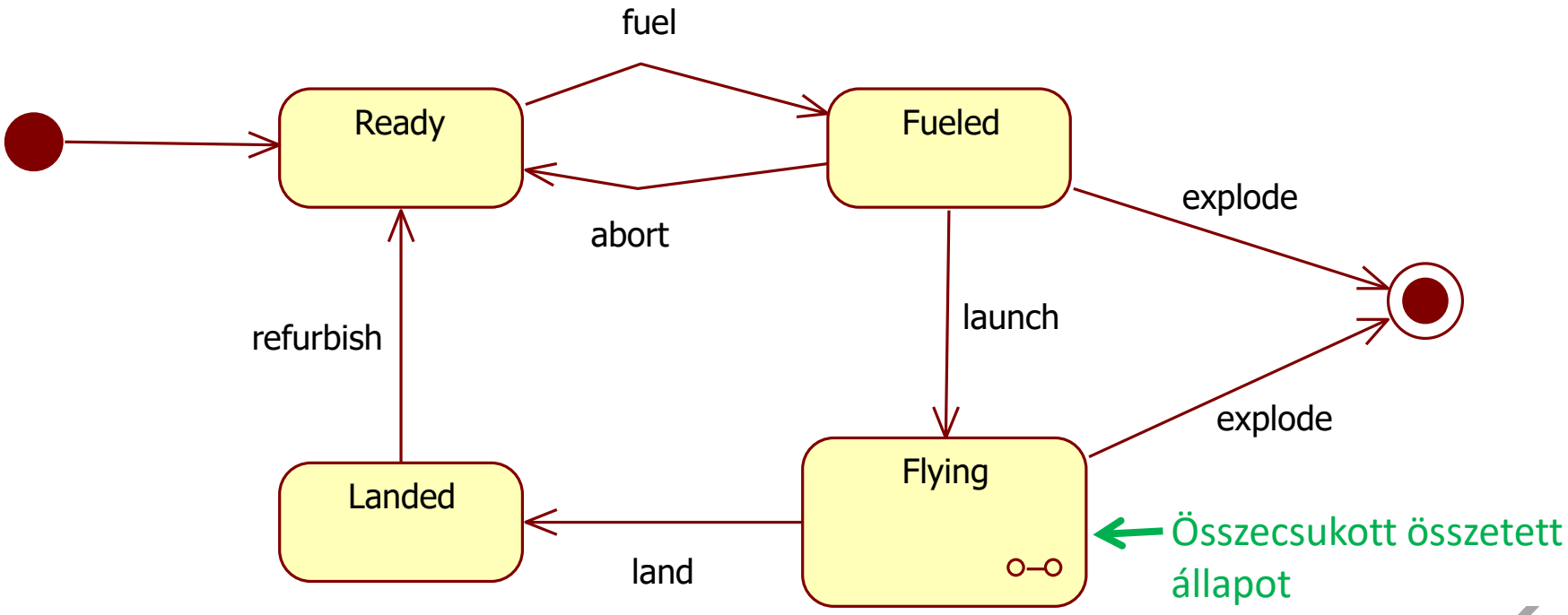
Őrfeltétel (guard condition)

- Az átmenet csak akkor aktív, ha a hozzárendelt őrfeltétel értéke igaz
- Ha egy átmenet nem aktív, akkor nem fut le, még akkor sem, ha a megfelelő esemény érkezik



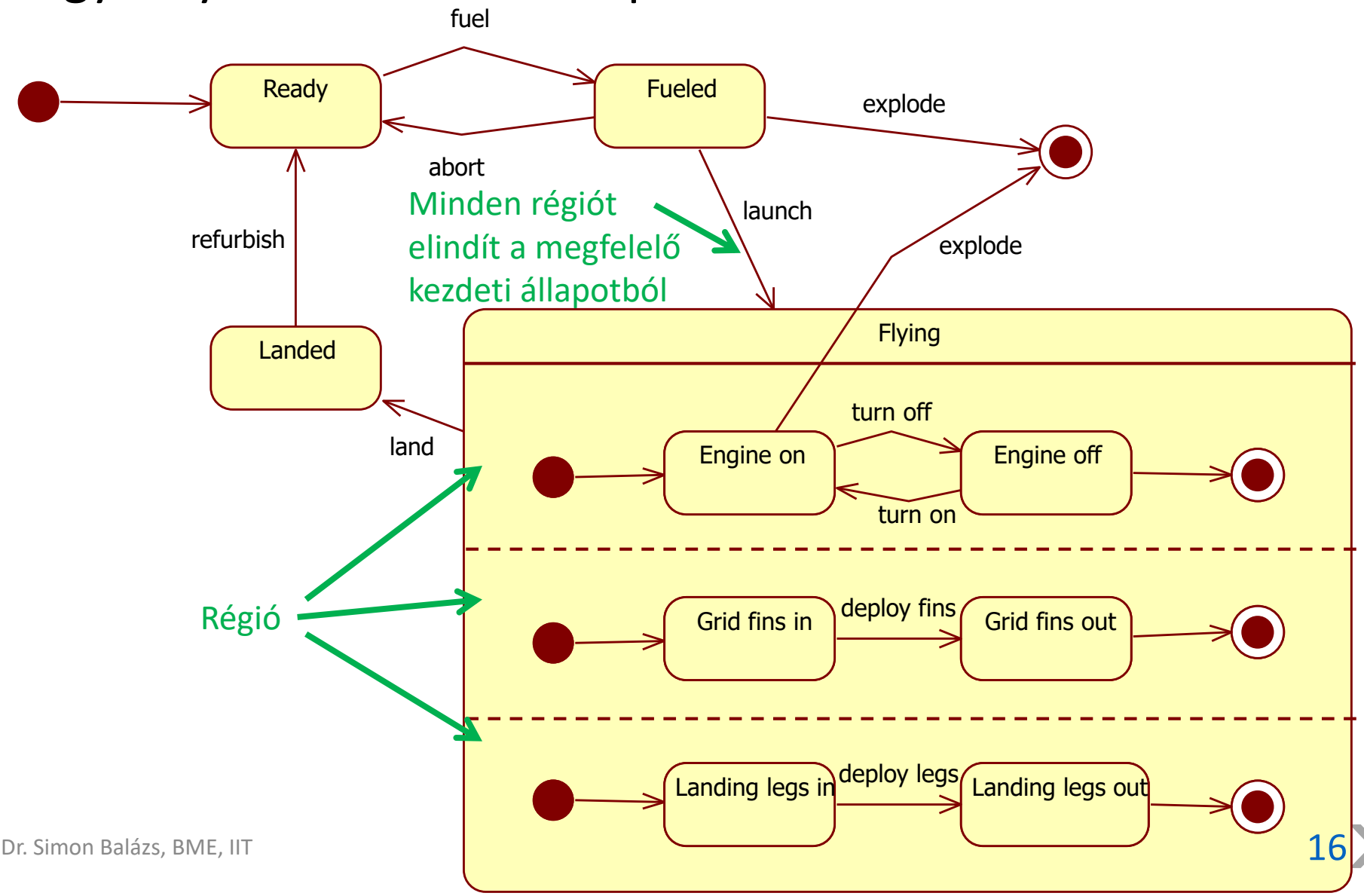
Összetett állapot (composite state)

- Egy egyszerű állapotnak nincsenek belső állapotai vagy átmenetei
- Egy összetett állapotnak van belső struktúrája
  - egy vagy több régió (region)
  - minden egyes régióban állapotok és átmenetek
  - a régiók függetlenek egymástól
- Az olvashatóság kedvéért egy összetett állapot összecsukható egyetlen állapottá



Összetett állapot (composite state)

- Egy kinyitott összetett állapot:

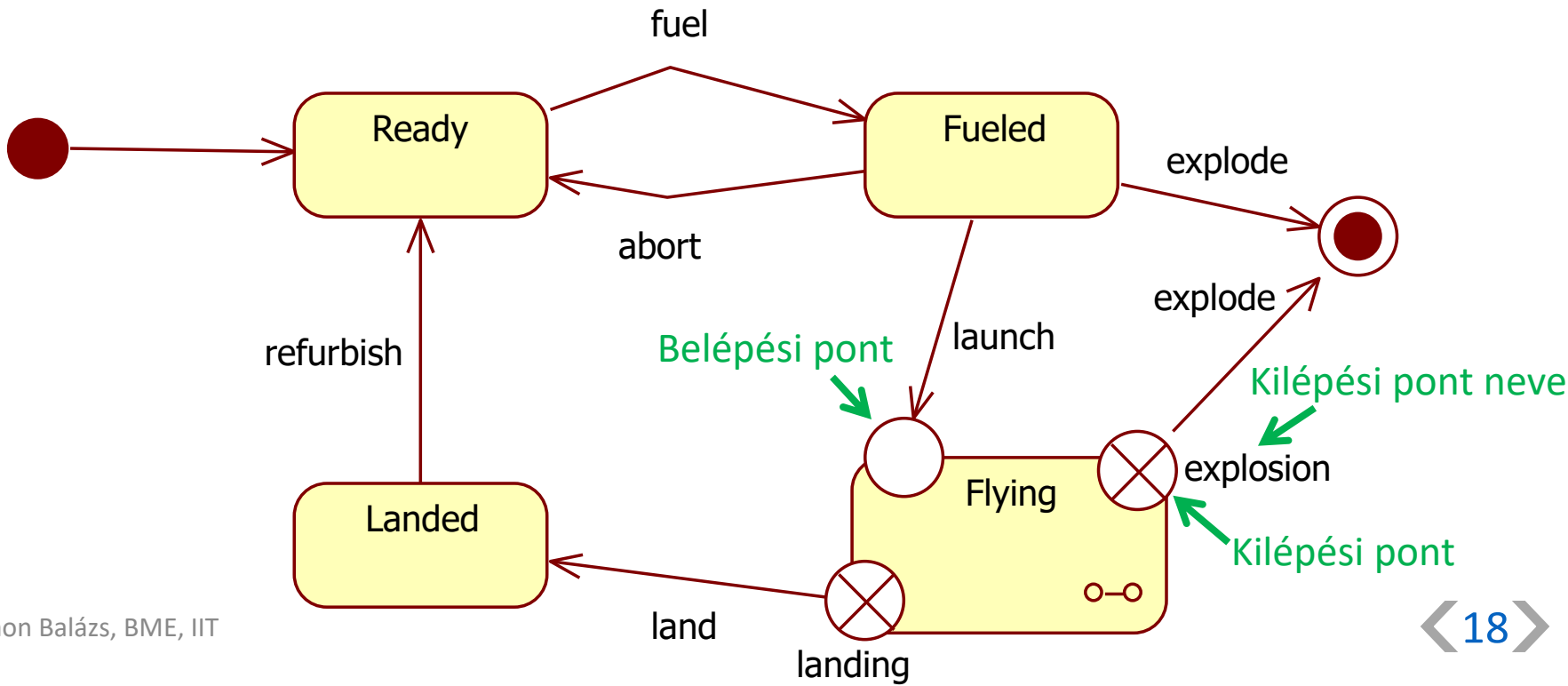


Belépés és kilépés összetett állapotokban

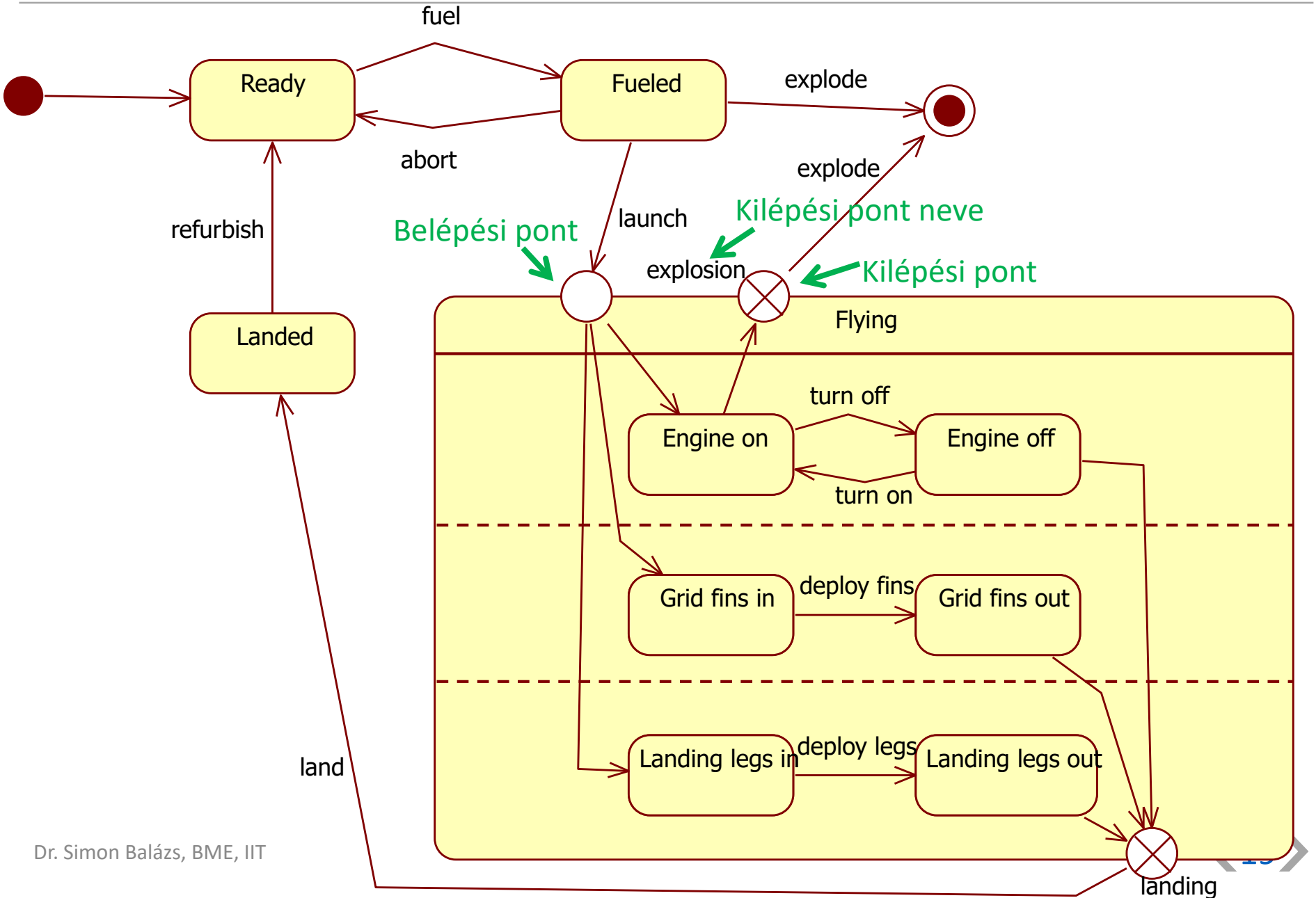
- **Alapértelmezett aktiválás (default activation)**: az átmenet, amely közvetlenül az összetett állapot peremébe megy, minden régiót párhuzamosan elindít a megfelelő kezdeti állapotokból
  - ha nincs kezdeti állapota egy régiónak, a viselkedés nem definiált
- **Explicit aktiválás (explicit activation)**: az átmenet, amely egy összetett állapot belső állapotába fut be, az adott régiót ebből az állapotból indítja, a többi régió alapértelmezett aktiválással indul
- Az összetett állapotból kimenő címkézetlen átmenet csak akkor fut le, ha minden régió eljutott a saját végső állapotába

Belépési és kilépési pontok (entry and exit points)

- A belépési és kilépési pontok az összetett állapot egységbezárását segítik
  - bizonyos helyzetekben hasznos lehet elrejteni egy összetett állapot belsejét, és nem engedni közvetlen átmenetet belülre
  - a belépési és kilépési pontok segítenek a külső átmeneteket összekötni a belső elemekkel



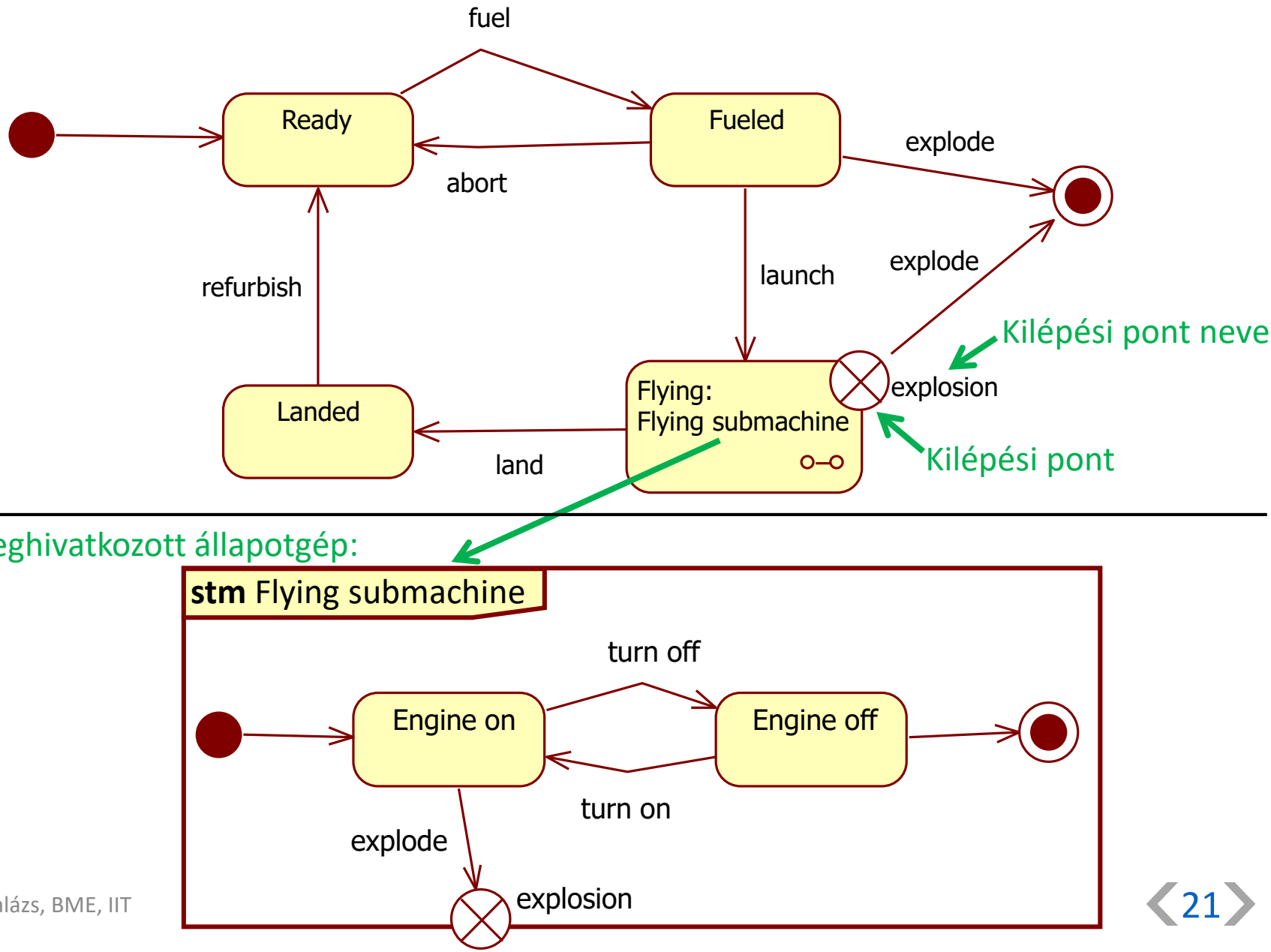
Belépési és kilépési pontok (entry and exit points)



Alállapotgép állapot (submachine state)

- Alállapotgép állapotok segítségével egy állapotgép leírása többször újrahasznosíthatóvá válik
- Hasonló az összetett állapothoz, azonban az alállapotgép állapotok külön-külön példányokat jelentenek a hivatkozott állapotgépből
  - az alállapotgép állapot olyan, mint egy C makró meghívása: mintha bemásolnánk oda a hivatkozott állapotgépet
- A bemenő és kimenő állapotokat az alállapotgép állapothoz kell kötni, de ezek függnek attól a kontextustól, ahol az alállapotgép állapotot használjuk

Alállapotgép állapot (submachine state) példa

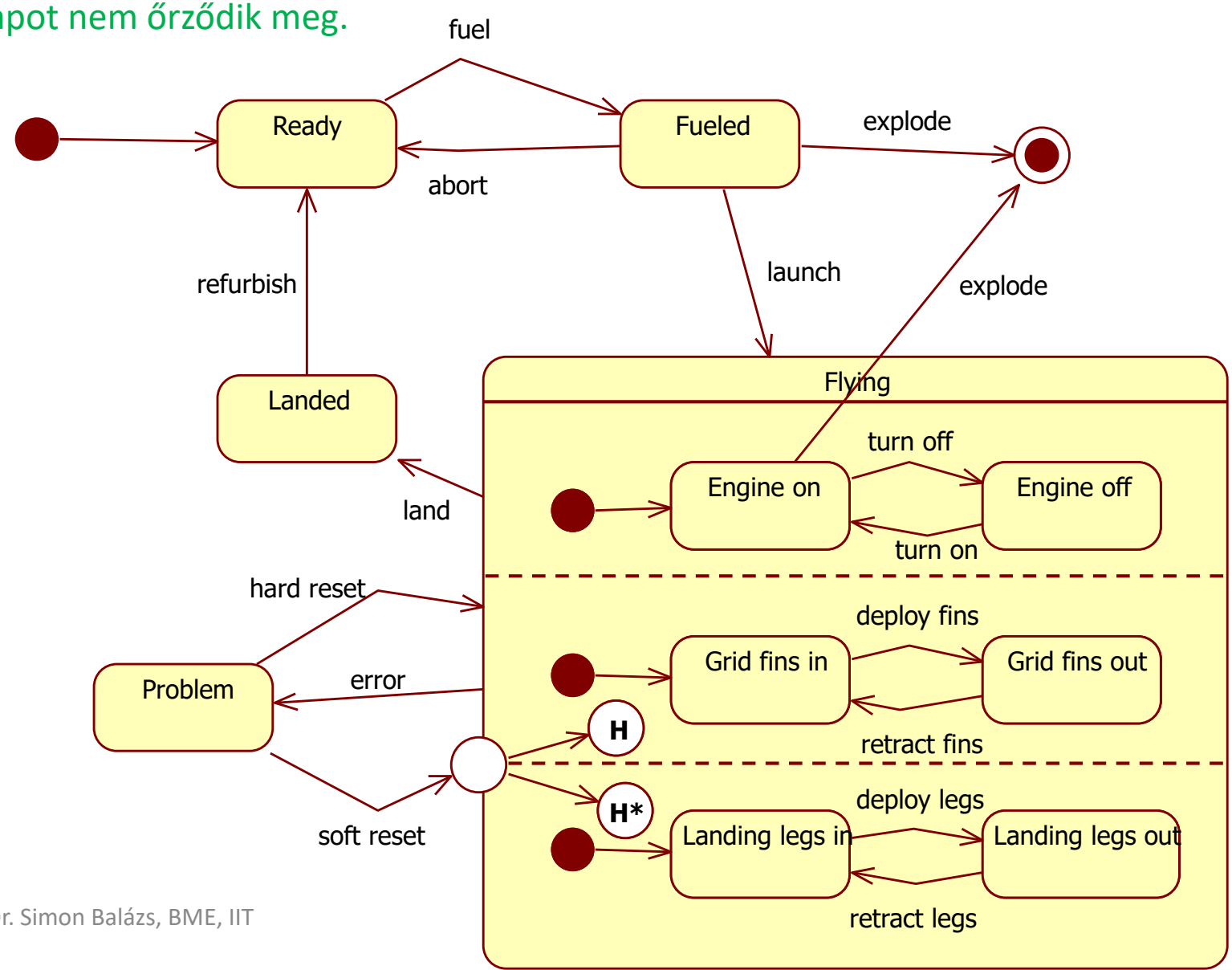


Történet (history)

- A történet állapotok (history states) egy összetett állapot régiói korábbi konfigurációjának visszaállítását segítik, amely akkor volt érvényes, amikor utoljára kiléptünk az összetett állapotból
- Az állapot (konfiguráció) akkor áll vissza, ha az aktív átmenet a történet állapotban ér véget
- Ha nincs korábbi állapotkonfiguráció (vagyis először lépünk be egy összetett állapotba, és ezt egy történet állapoton keresztül tesszük):
  - ha a történet állapotnak van átmenete egy alállapotba, akkor a történet állapot kezdeti állapotként viselkedik
  - egyébként az alapértelmezett aktiválás érvényes
- Kétfajta történet állapot van:
  - **mély történet (deep history)**: visszaállítja az összetett állapot konfigurációját, és minden alállapot konfigurációját rekurzív módon
  - **sekély történet (shallow history)**: csak a legkülső összetett állapot konfigurációját állítja vissza, az alállapotokét nem

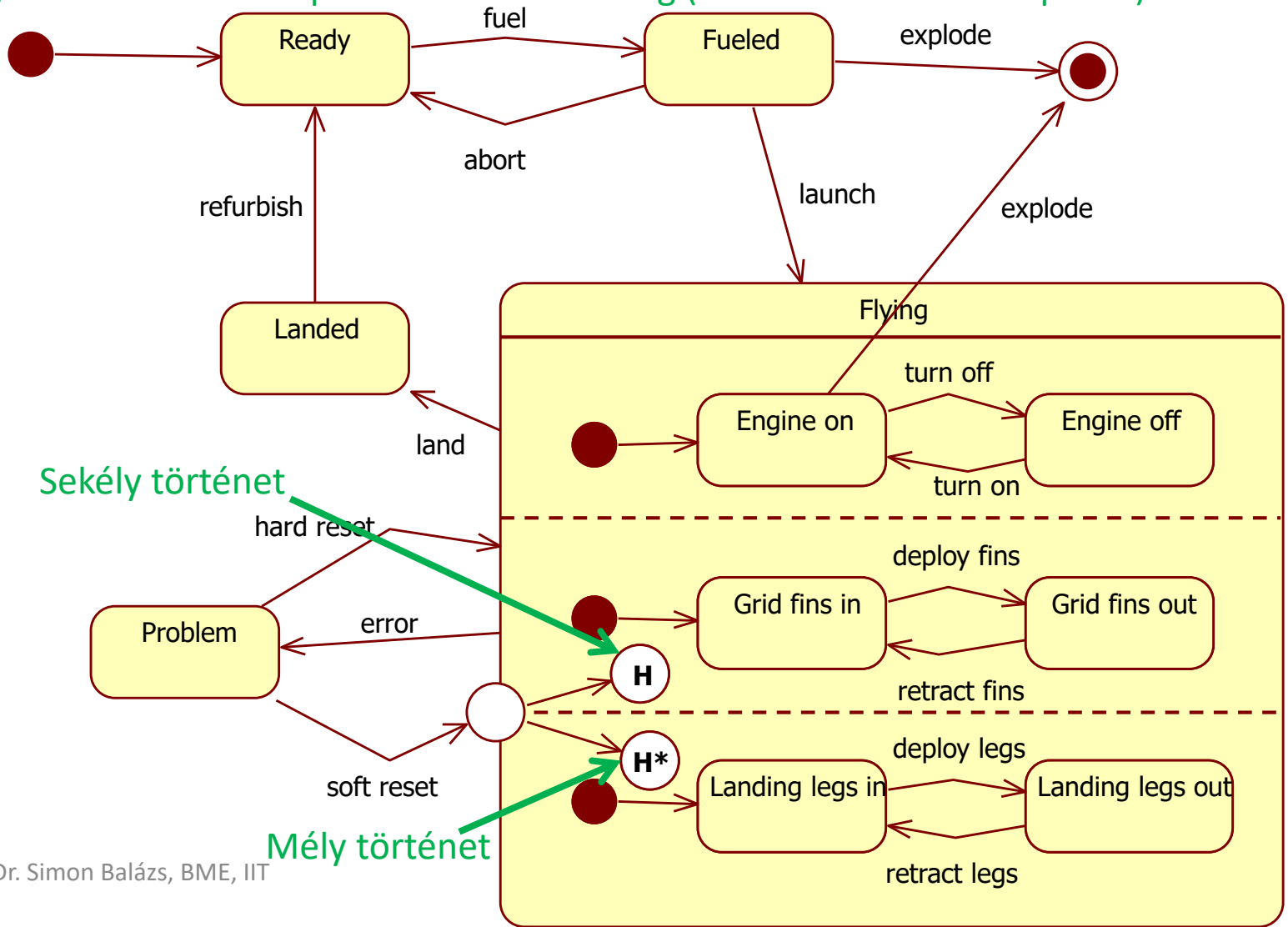
Történet (history) példa

‘hard reset’ esetén: the engine is turned on, grid fins are pulled in, landing legs are pulled in. Állapot nem őrződik meg.



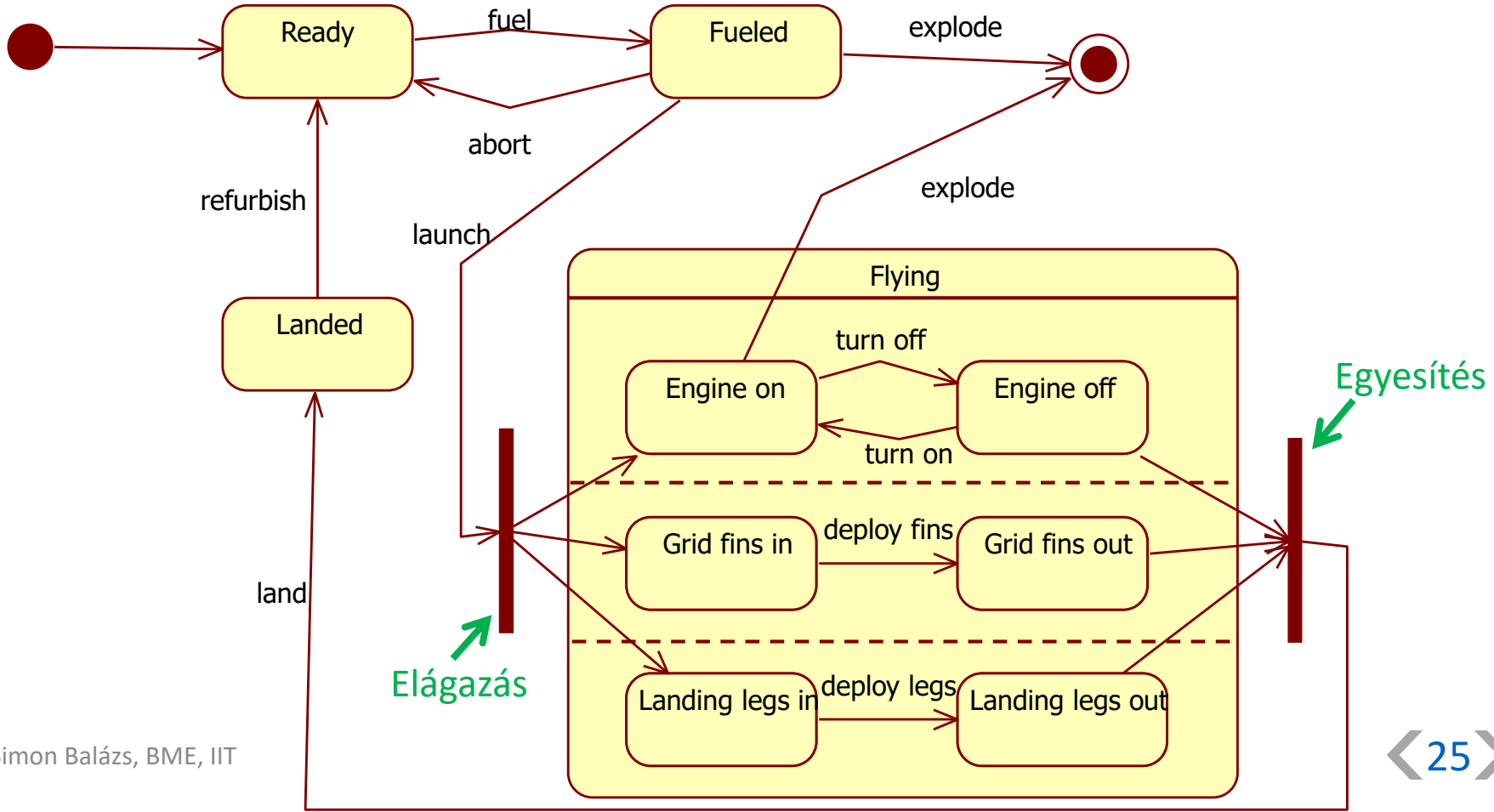
Történet (history) példa

‘soft reset’ esetén: the engine is turned on, grid fins and landing legs preserve their positions.
A lábak belső állapota megőrződik (ha ezek összetett állapotok).
A grid fin-ek belső állapota **nem** őrződik meg (ha ezek összetett állapotok).

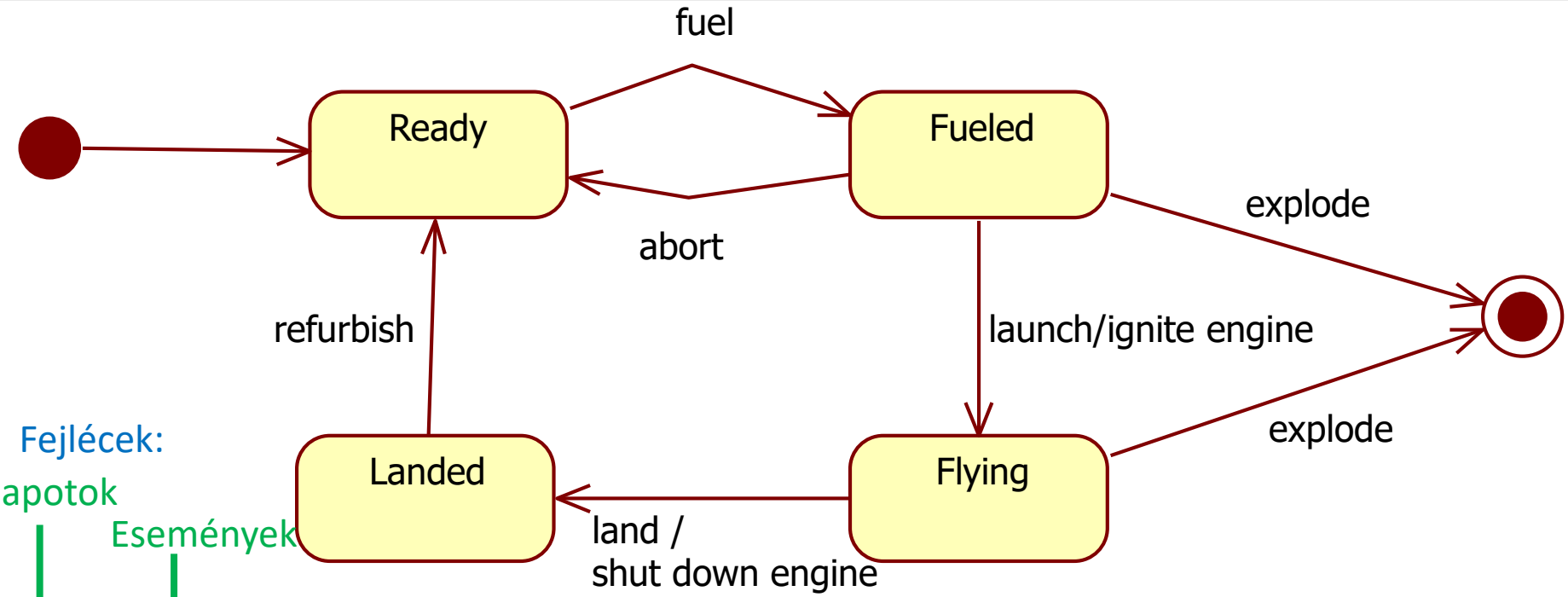


Elágazás és egyesítés (fork and join)

- **Elágazás (fork)**: a beérkező átmenet két vagy több átmenetre bontása, amelyek egy összetett állapot különböző régióiba futnak be
  - a kimenő átmeneteknek nem lehet őrfeltétele vagy eseménye
- **Egyesítés (join)**: több átmenet összevárása, majd egy átmenetként folytatás
  - a bejövő átmeneteknek nem lehet őrfeltétele vagy eseménye



Állapotgép táblázatos formában

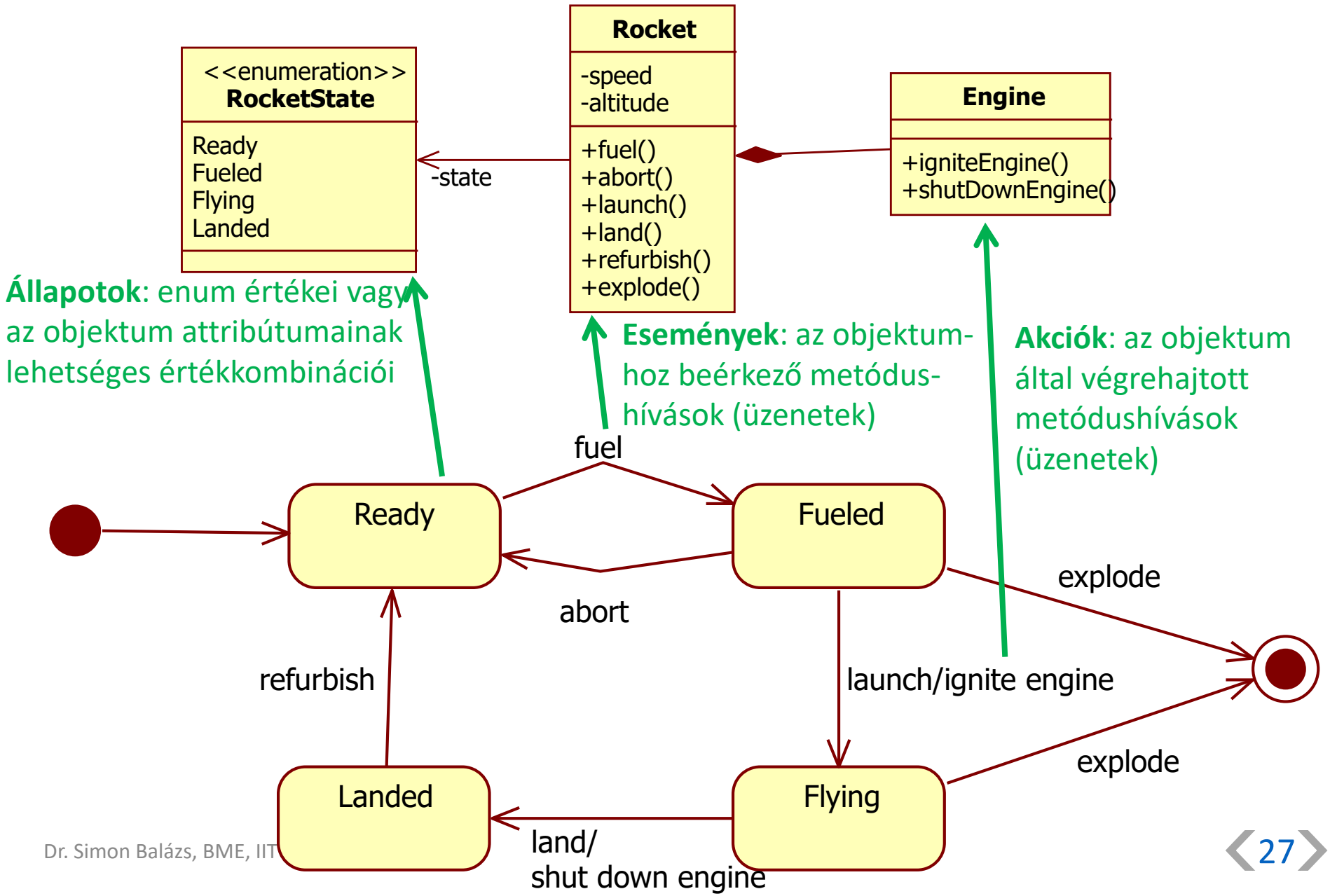


Fejlécek:
Állapotok
Események

	fuel	**abort**	**launch**	**land**	**refurbish**	**explode**
Ready	Fueled/-	-/-	-/-	-/-	-/-	-/-
Fueled	-/-	Ready/-	Flying/ ignite engine	-/-	-/-	-/-
Flying	-/-	-/-	-/-	Landed/ shut down engine	-/-	-/-
Landed	-/-	-/-	-/-	-/-	Ready/-	-/-

Cellák: Következő állapot / Akció

Állapotgép a Rocket osztályra



Hol tartunk?

Strukturális UML diagramok:

Komponens-diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

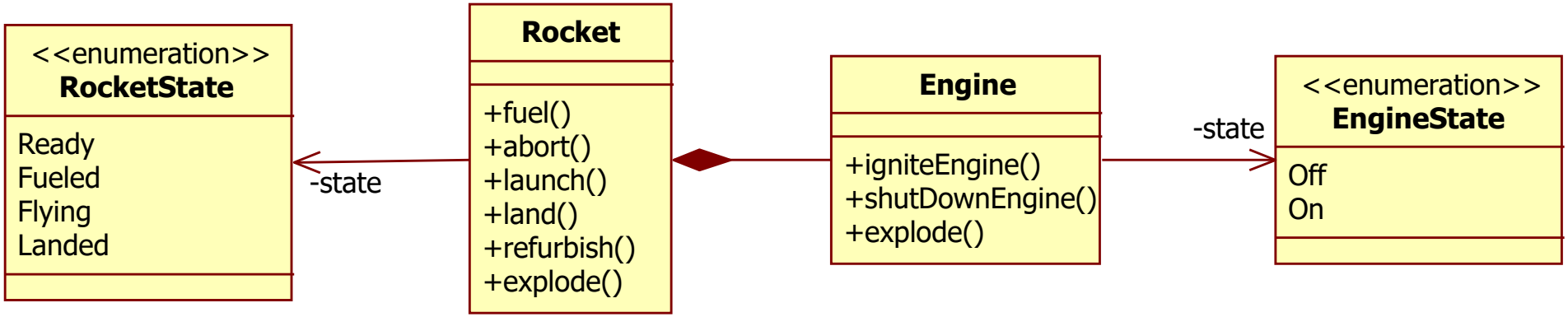
Use case diagram	Aktivitásdiagram	Szekvenciadiagram	Kommunikációs diagram
Állapotdiagram	Időzítődiagram	Interakciós áttekintő diagram	

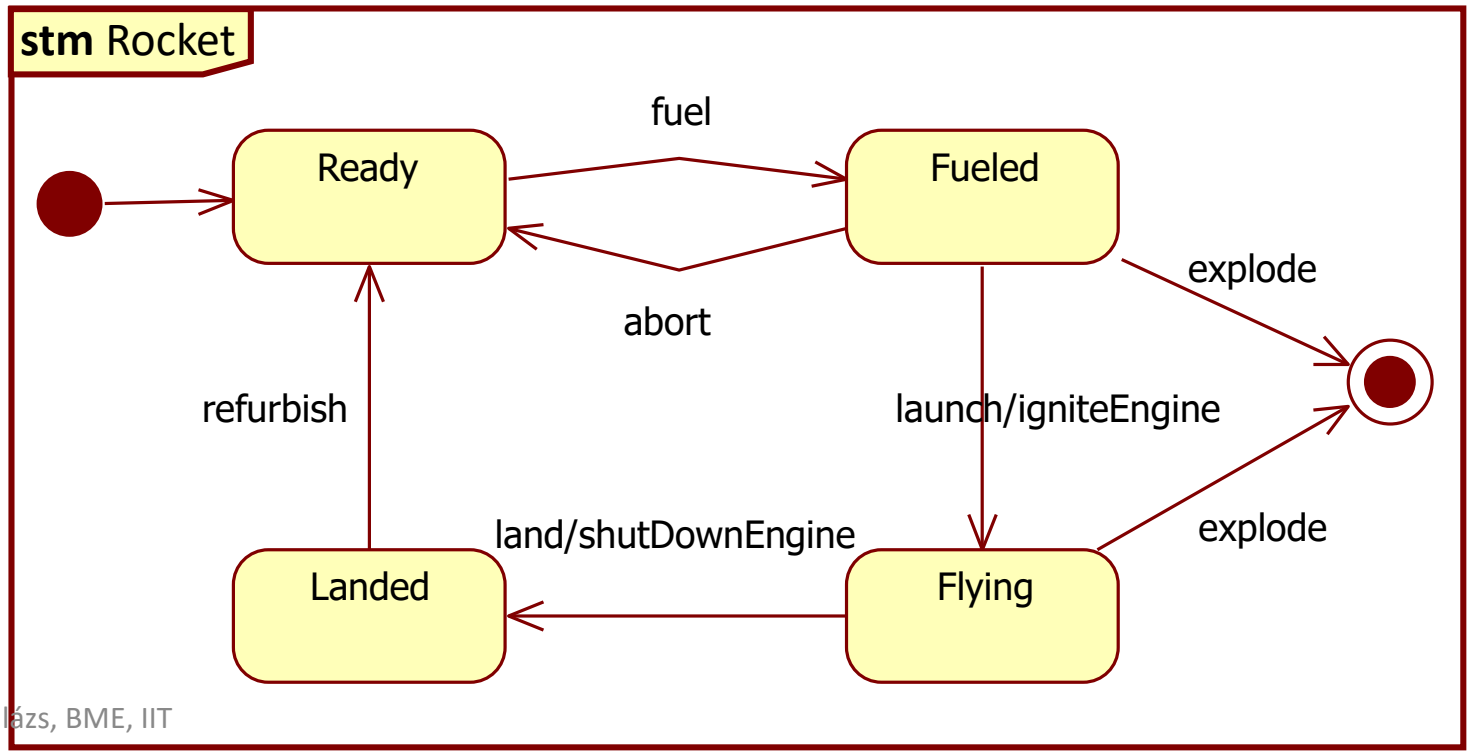
Időzítődiagram (Timing Diagram)

Időzítődiagram (Timing Diagram)

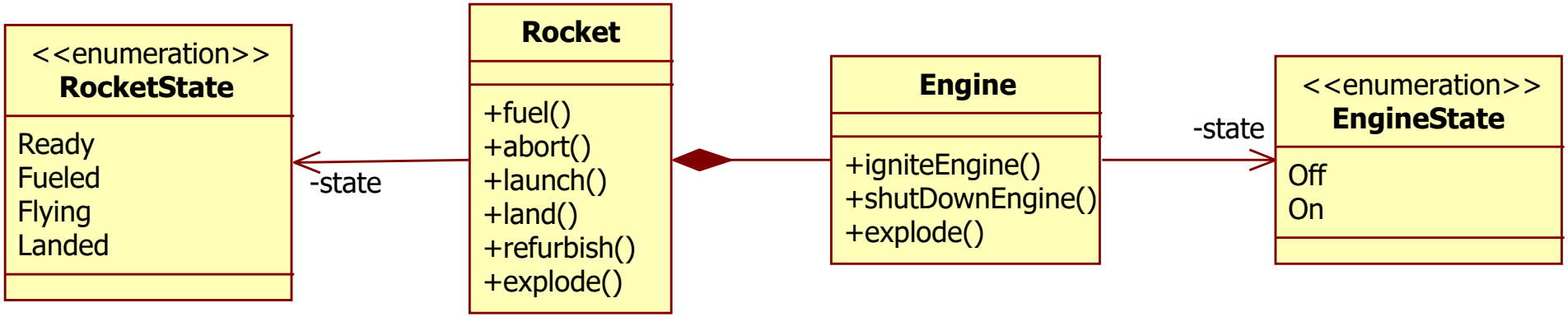
- Az időzítődiagram lifeline-okon belüli és azok közötti állapotváltásokra fókuszálnak egy időtengely mentén
- Az időzítődiagramok önálló classifier-eket és classifier-ek közötti interakciókat ábrázolnak eseményekkel és állapotváltásokkal, követve az időt és az ok-okozati összefüggéseket
- Az időzítődiagram hasonló a szekvenciadiagramhoz, de:
  - az idő balról jobbra telik (nem fentről lefelé)
  - a lifeline-ok állapotát is mutatja
    - az állapot lehet diszkrét vagy folytonos

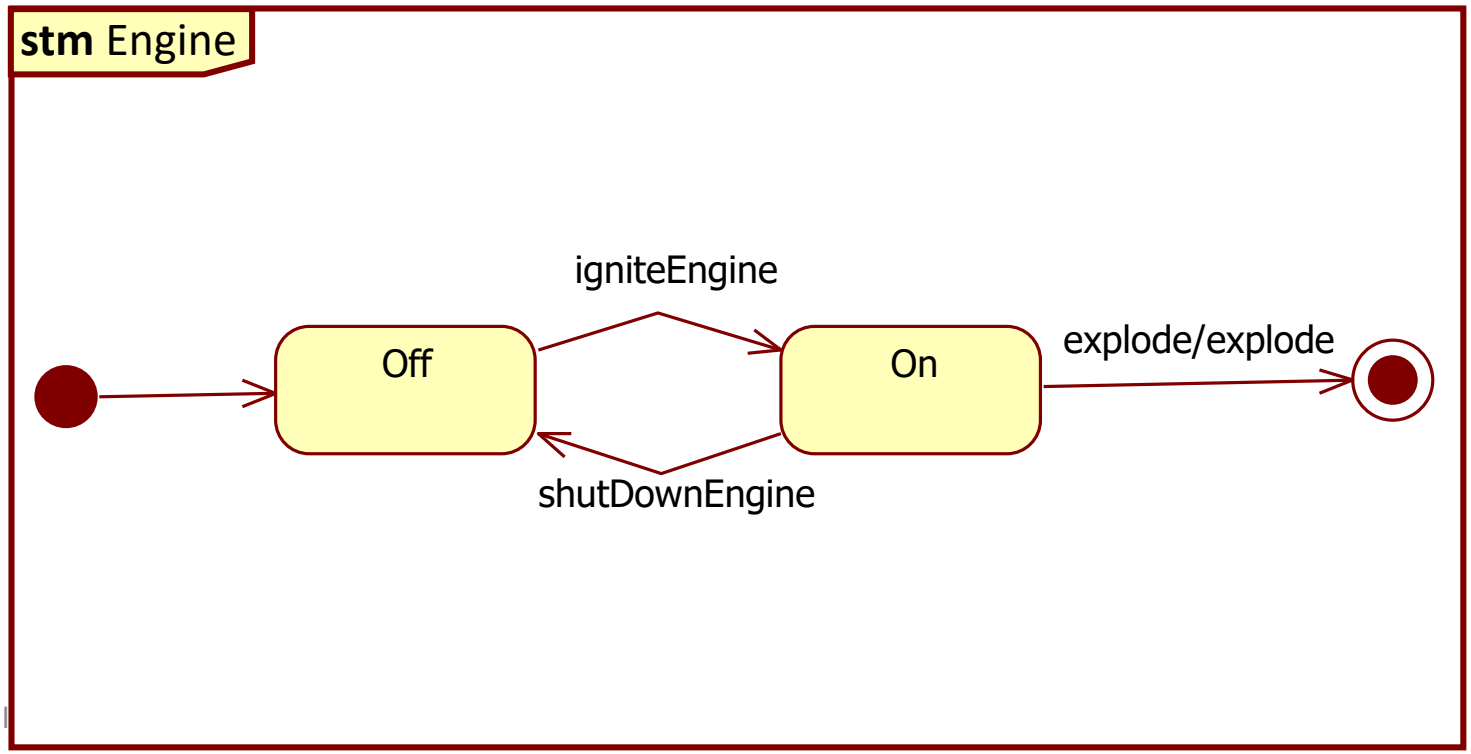
Állapotdiagram a Rocket osztályra



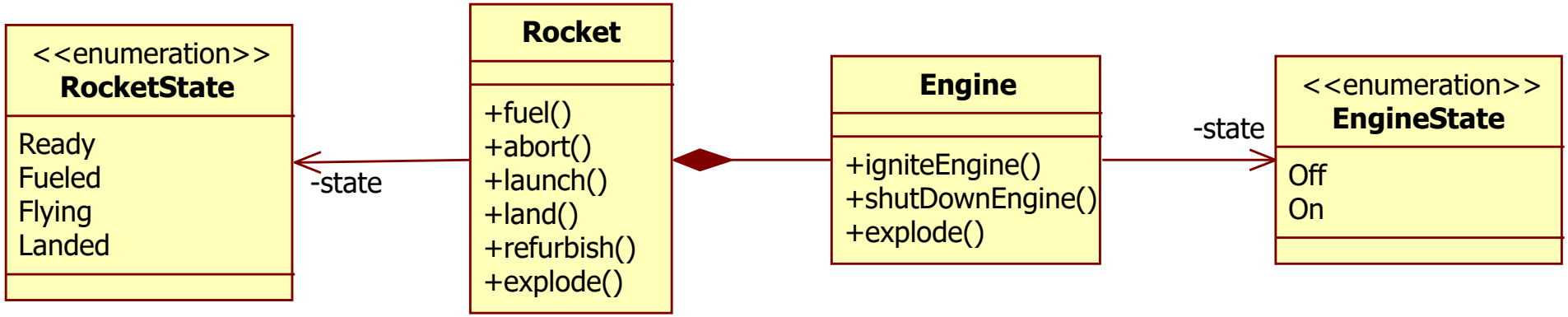


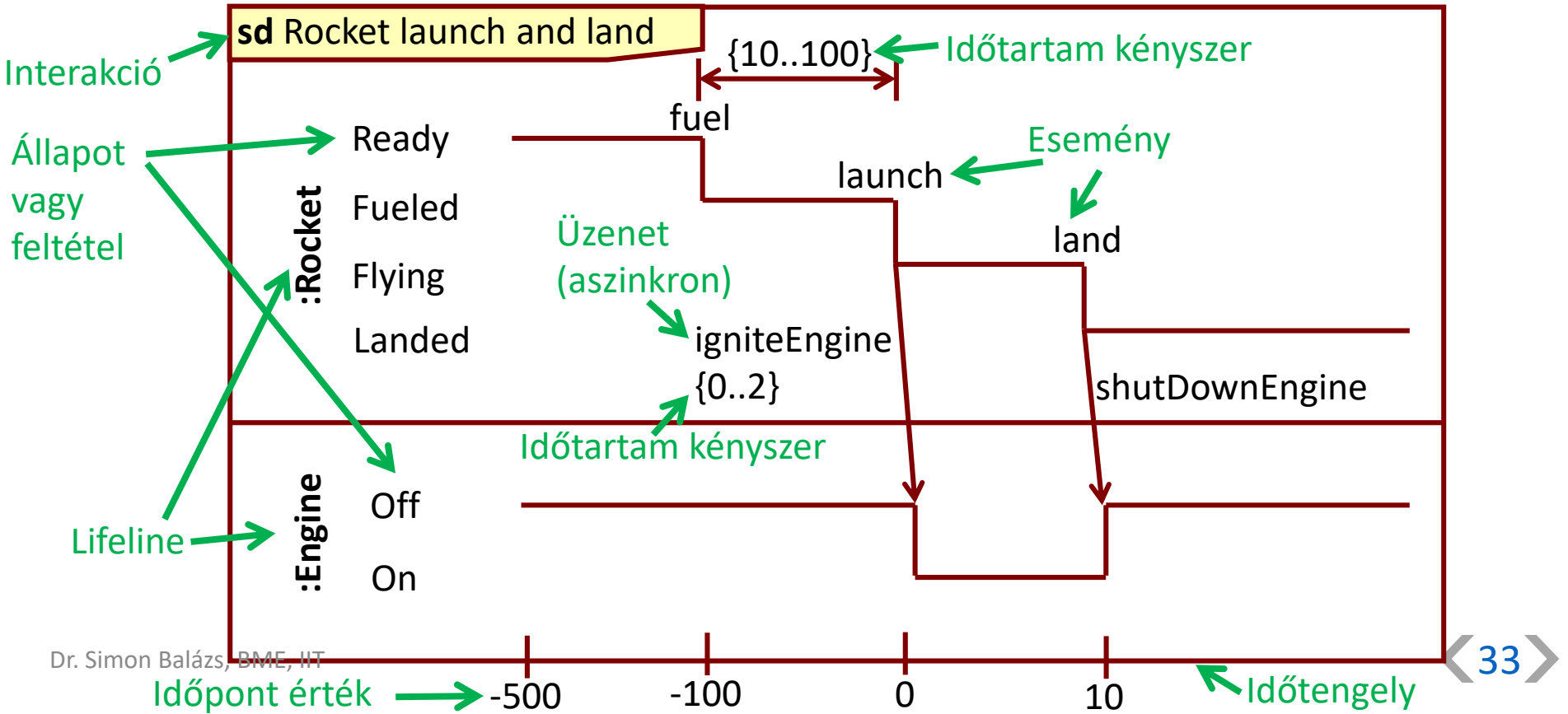
Állapotdiagram az Engine osztályra



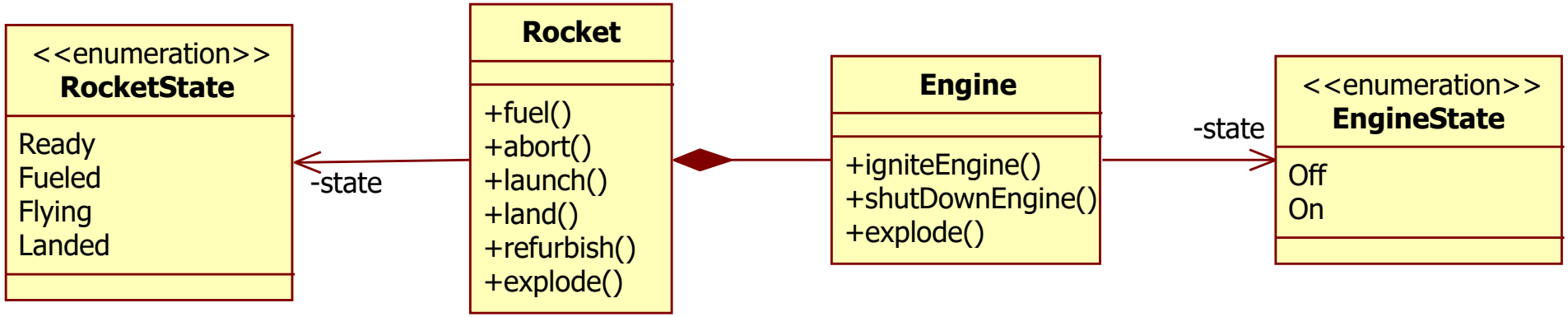


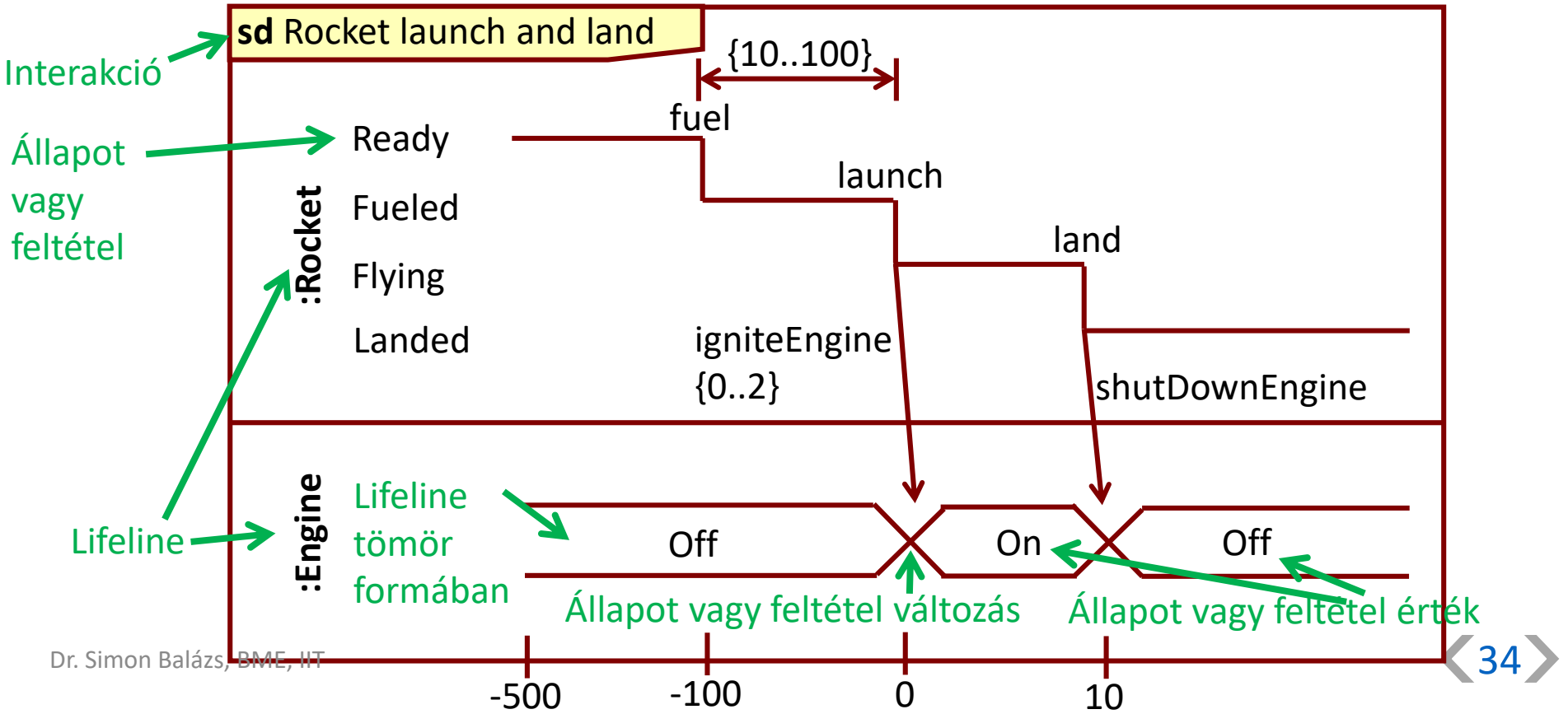
Időzítődiagram példa: rakéta kilövés és leszállás



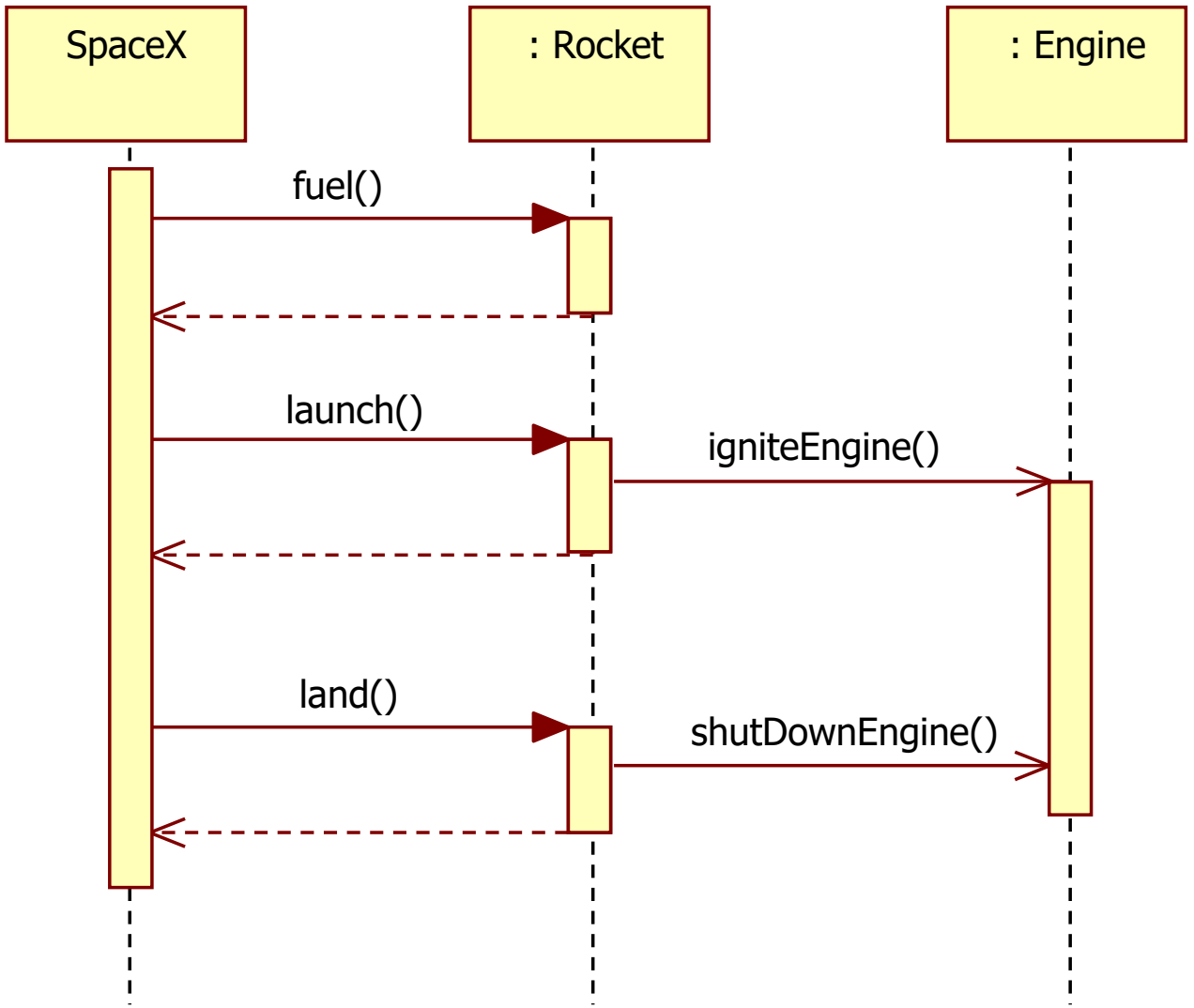


Időzítődiagram példa: rakéta kilövés és leszállás

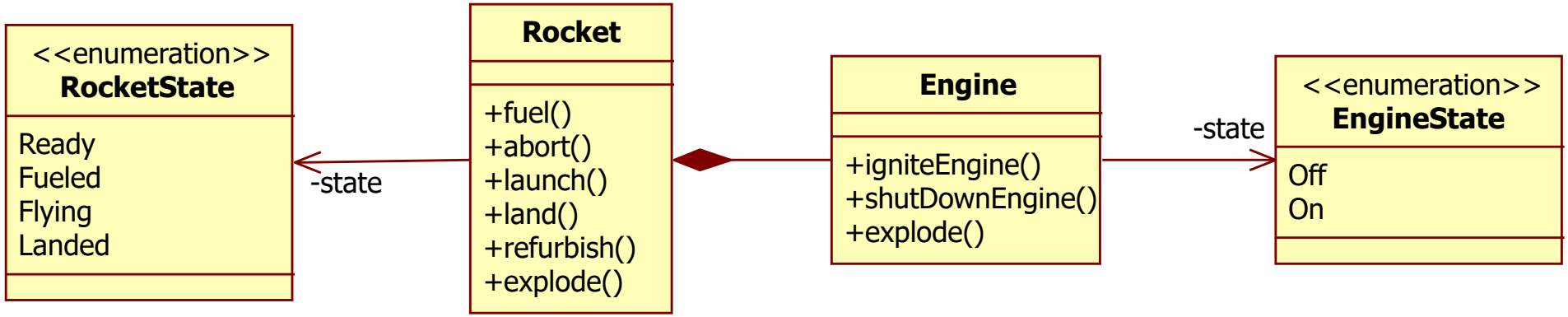


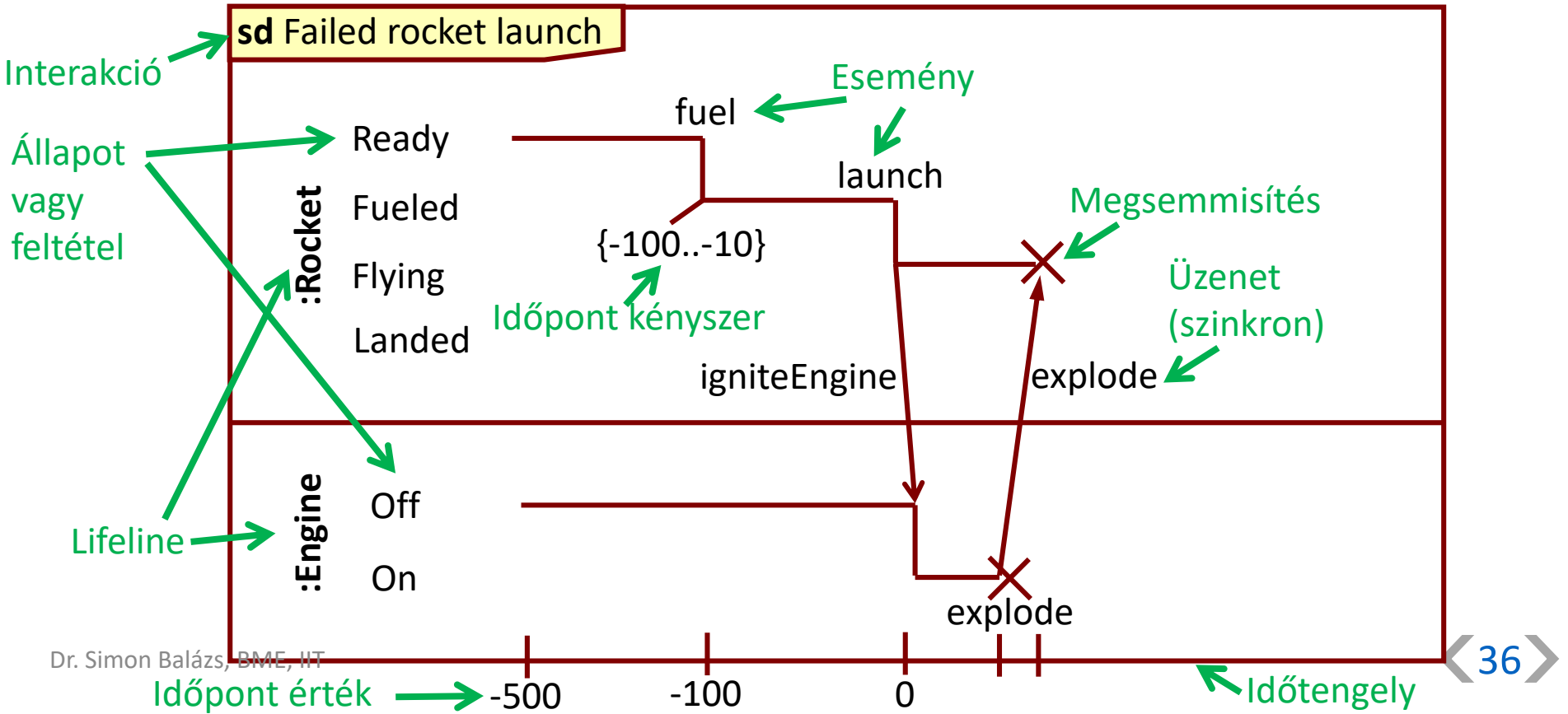


Szekvenciadiagram példa: rakéta kilövés és leszállás

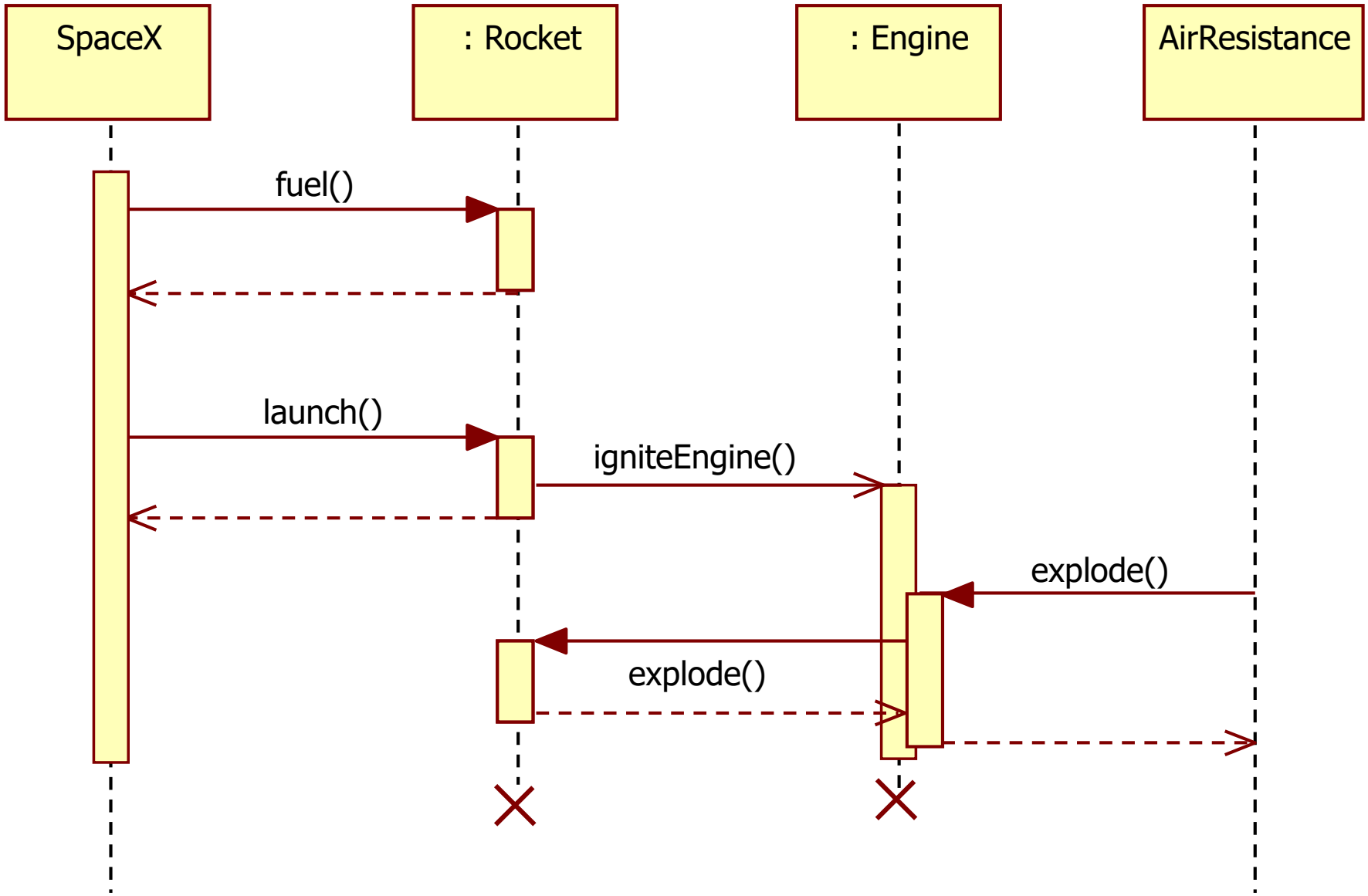


Időzítődiagram példa: sikertelen rakéta kilövés





Szekvenciadiagram példa: sikertelen rakéta kilövés



Hol tartunk?

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	Interakciós áttekintő diagram	

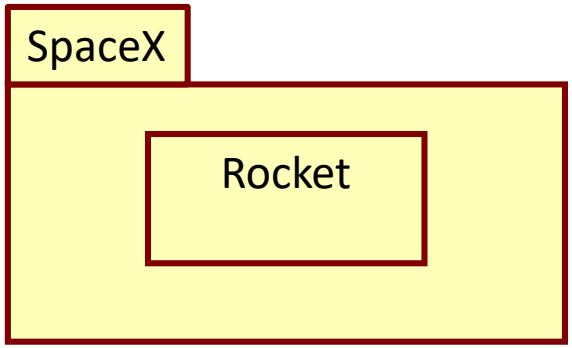
Összetett struktúradiagram (Composite Structure Diagram)

Összetett struktúradiagram (Composite Structure Diagram)

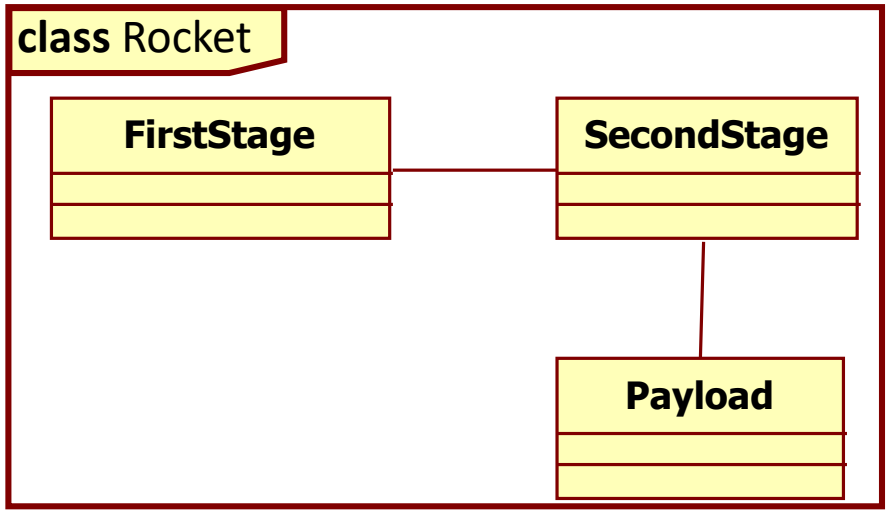
- Egy struktúrával rendelkező classifier (pl. osztály, komponens) belső szerkezetét mutatja
- Általában egy classifier belső szerkezetét nem mutatjuk egy komponensdiagramon vagy osztálydiagramon
  - ilyenkor hasznos az összetett struktúradiagram

Összetett struktúradiagram példa

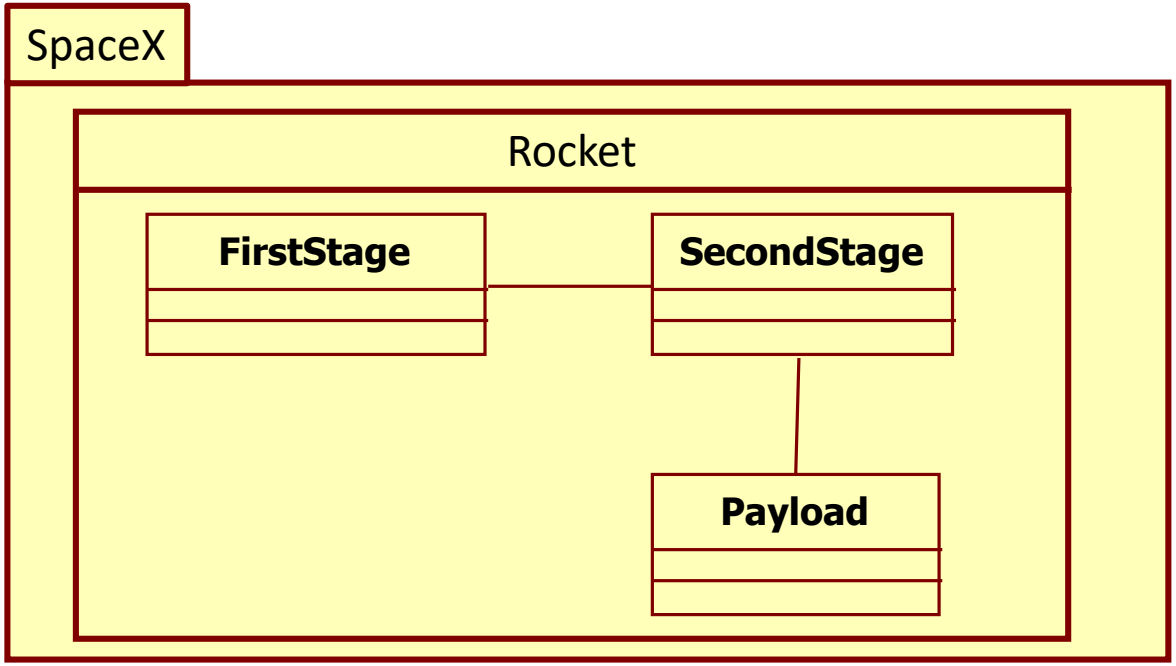
Osztálydiagram, ahol a Rocket osztály belső szerkezete nem látszik:



Összetett struktúradiagram a Rocket osztályra:



Osztálydiagram, ahol a Rocket osztály belső szerkezetét külön compartment mutatja:



Hol tartunk?

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	Interakciós áttekintő diagram	

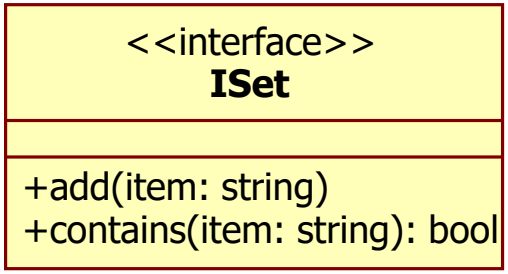
Profildiagram
(Profile Diagram)

Profildiagram (Profile Diagram)

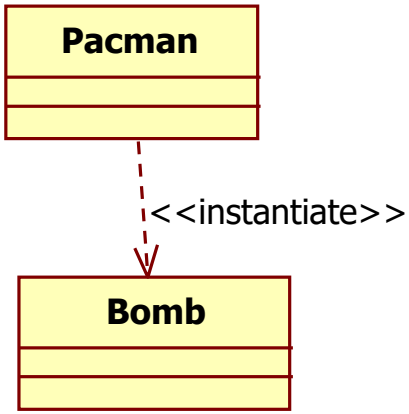
- A profildiagram segítségével saját sztereotípiákat/kulcsszavakat (stereotype/keyword) definiálhatunk
- A sztereotípiák modellelemekhez csatolhatók
- A sztereotípiák módosítják az adott modellelem jelentését
  - az egyénileg definiált sztereotípiák jelentését nem az UML szabvány határozza meg, hanem mi
  - az egyéni sztereotípiák értelmezése és feldolgozása a mi feladatunk, amikor az UML diagramokból programkódot készítünk
- A sztereotípiák kiváló bővítési lehetőséget biztosítanak az UML-ben

Szabványos sztereotípiák

- Vannak sztereotípiák, amelyeket az UML szabvány definiál
- Példák:
  - **<<interface>>**
    - egy osztályhoz csatolható
    - azt jelzi, hogy ez többé már nem osztály, hanem egy interfész



  - **<<instantiate>>**
    - egy függőséghez csatolható
    - pontosítja a függőség jelentését: azt jelzi, hogy a kliens példányokat készít a szerverből

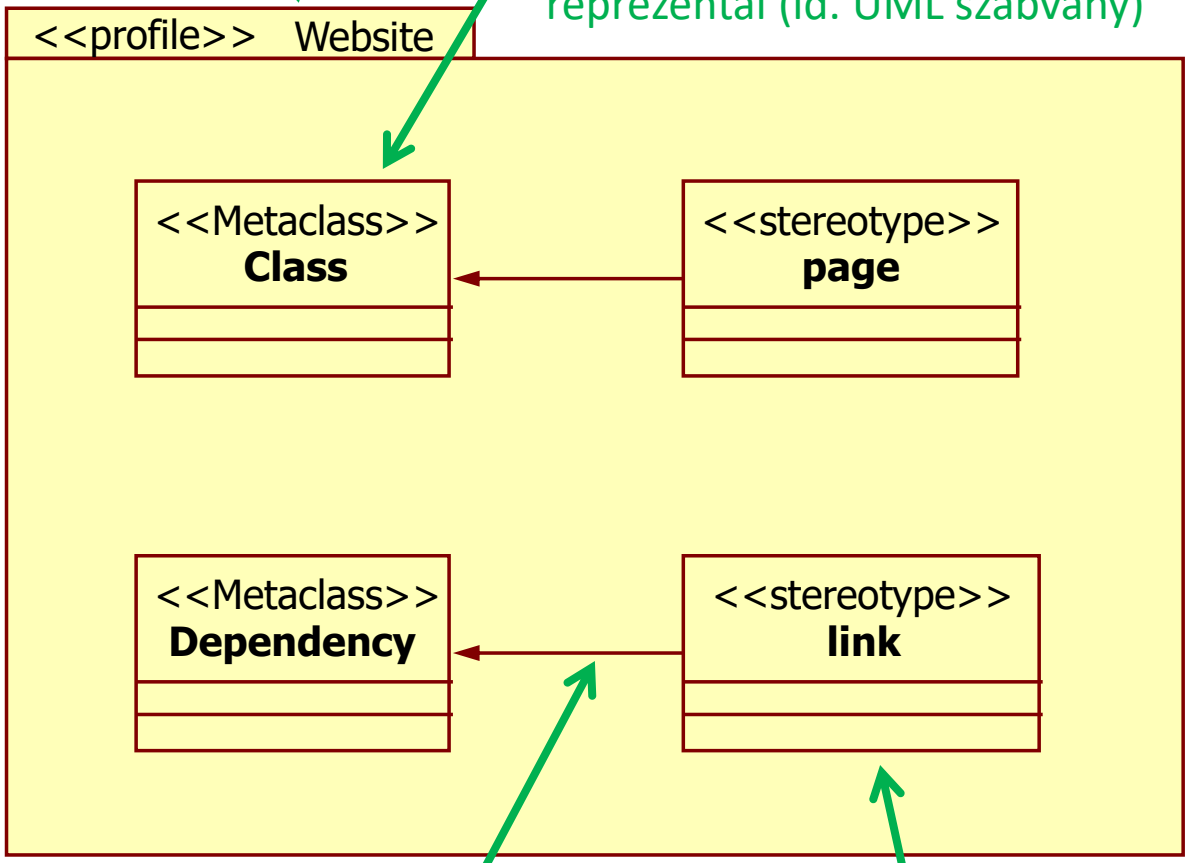


- De készíthetünk saját sztereotípiákat is a profildiagramok segítségével

Profildiagram példa: Weboldalak

Egy <<profile>> sztereotípiával ellátott csomag egy UML profilt definiál

Egy <<MetaClass>> sztereotípiával ellátott osztály egy UML szabványban definiált modellelemet reprezentál (ld. UML szabvány)

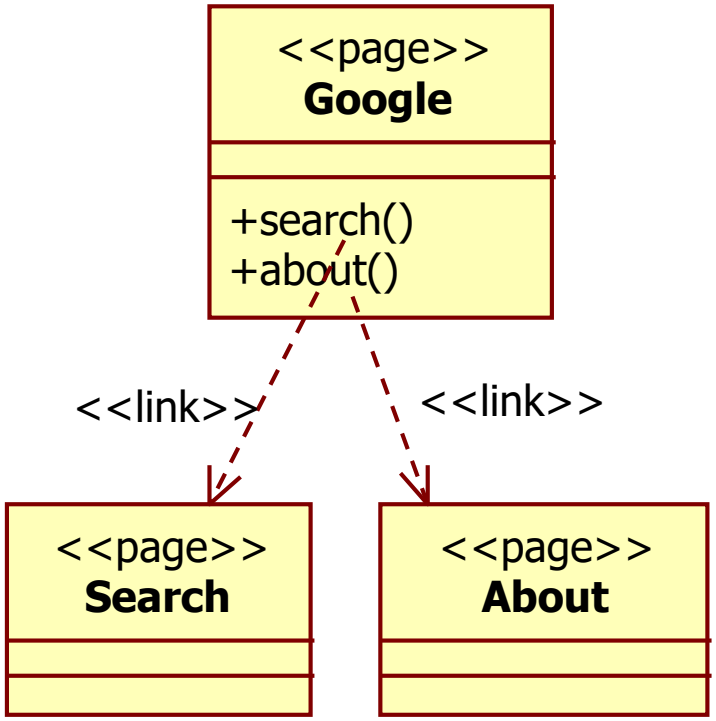


Ez a nyíl azt jelzi, hogy az adott sztereotípia hozzárendelhető az adott modellelemhez

Egy <<stereotype>> sztereotípiával ellátott osztály egy egyéni sztereotípiát definiál

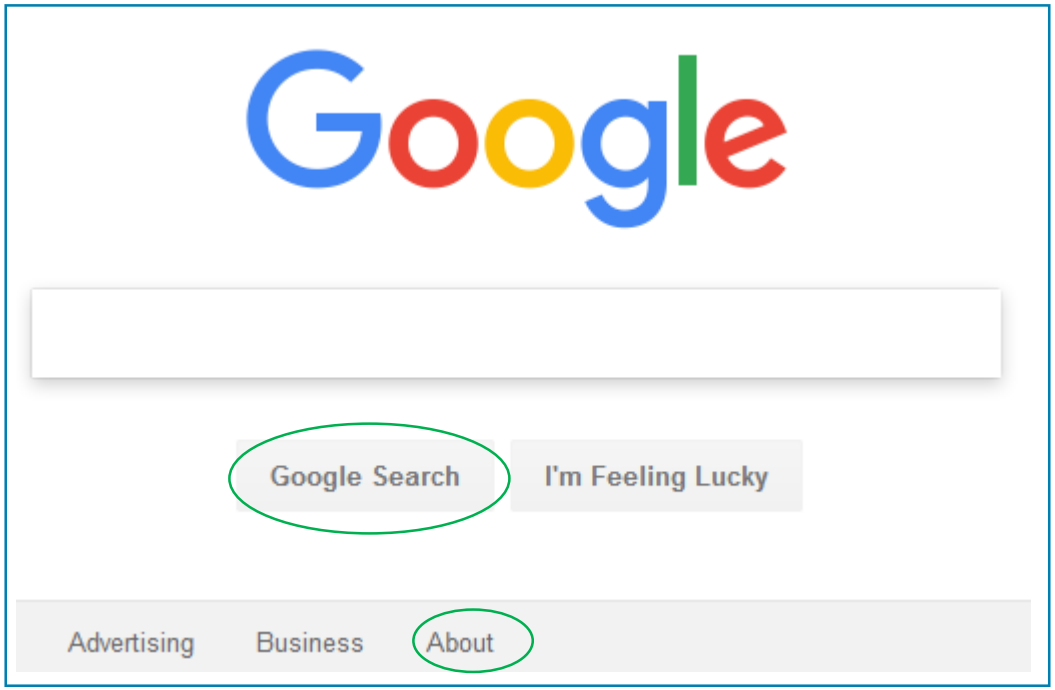
Osztálydiagram példa: Weboldal

Egy osztálydiagram, amely az általunk definiált egyéni sztereotípiákat használja:



A diagram értelmezése rajtunk múlik, mi definiáljuk a sztereotípiák jelentését

Például a diagram jelentheti azt, hogy minden <<page>> sztereotípiával rendelkező osztály egy weboldal, és minden <<link>> -kel rendelkező függőség egy link a következő oldalra:



A sztereotípiák nagyon jó bővítési lehetőséget adnak az UML-hez.
Egész diagramok vagy modellelemek jelentését átdefiniálhatjuk!

Hol tartunk?

Strukturális UML diagramok:

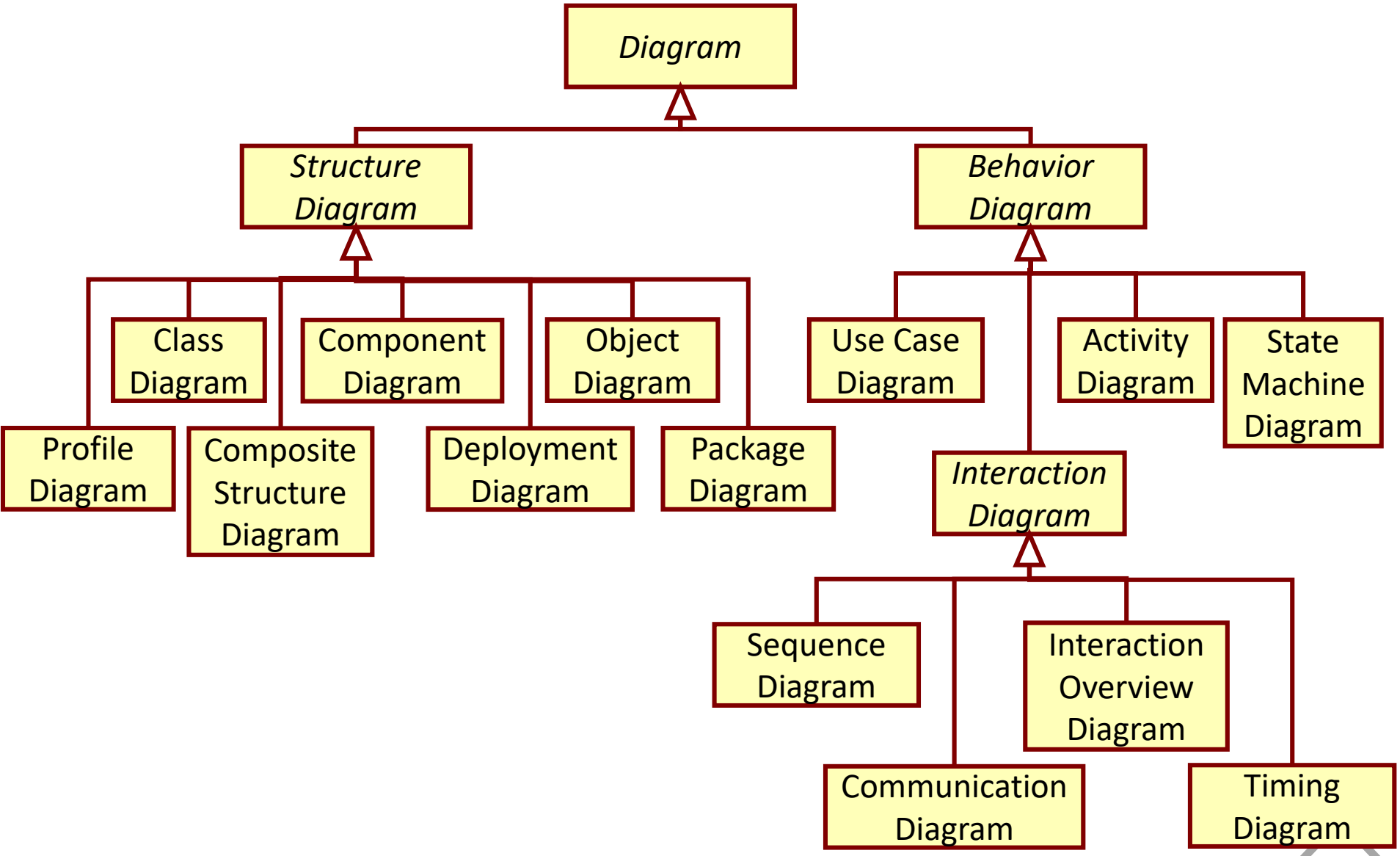
Komponens-diagram	Telepítési diagram	Osztálydiagram	Csomagdiagram
Objektumdiagram	Összetett struktúradiagram	Profildiagram	

Viselkedési UML diagramok:

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

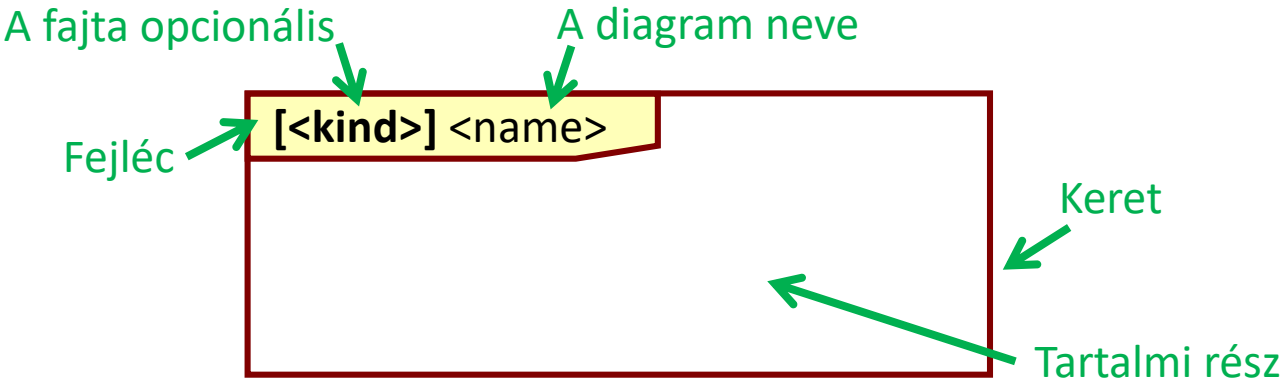
Az UML diagramok összefoglalása

UML diagramok



Keret (frame)

- Minden diagramnak van egy tartalmi része
- Opcionálisan egy diagramnak lehet kerete és fejléce is:



- A keret egy téglalap:
  - elsődlegesen akkor használjuk, ha a diagram által reprezentált elem szélén is lehetnek modellelemek, pl.
    - osztályok és komponensek esetén: portok
    - állapotgépek esetén: belépési és kilépési pontok
    - szekvenciadiagram: kapuk
- Ha nincs rá szükség, a keret elhagyható, és a tervezőeszköz rajzterületének széle alkotja a keretet
  - ha nincs keret, akkor nincs fejléc sem

Diagramok fajtái

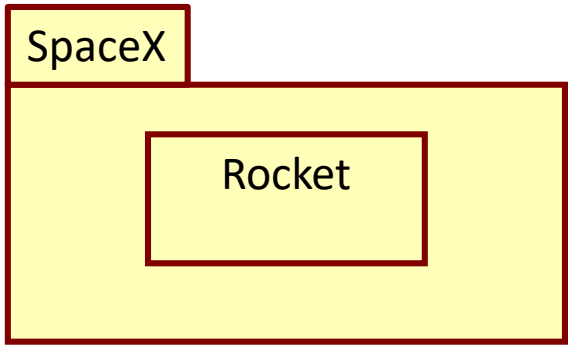
- A fajta az alábbiak egyike:

Fajta	Rövidítés
activity	**act**
class	
component	**cmp**
deployment	**dep**
interaction	**sd**
package	**pkg**
state machine	**stm**
use case	**uc**

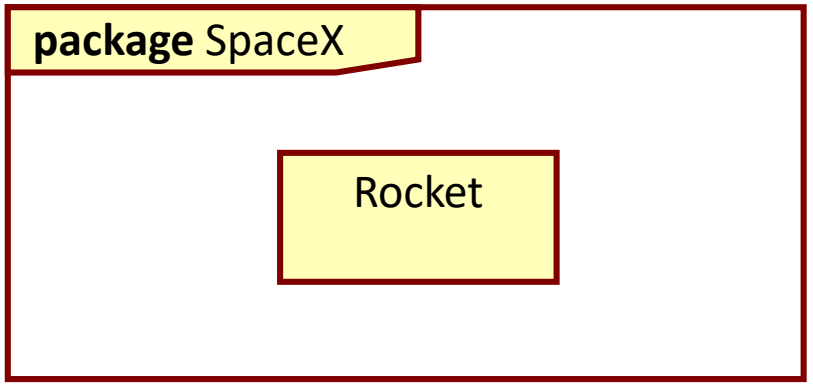
- A fajta hosszú változata helyett a rövidítés is használható

Keret példa

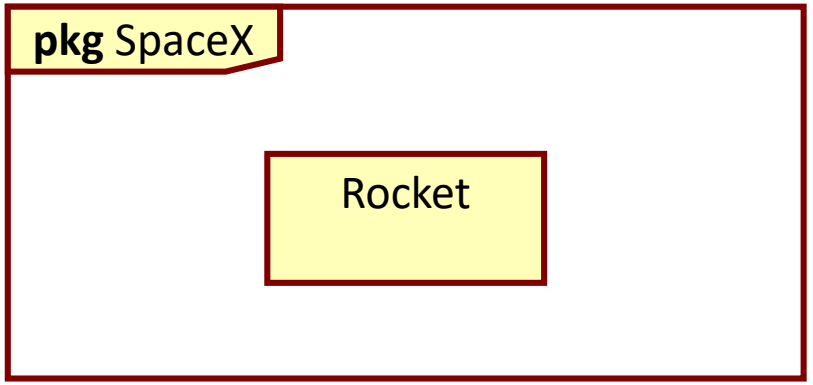
A SpaceX csomag egy nagyobb osztály-diagram részeként ábrázolva:



A SpaceX csomag és tartalma keretként ábrázolva:

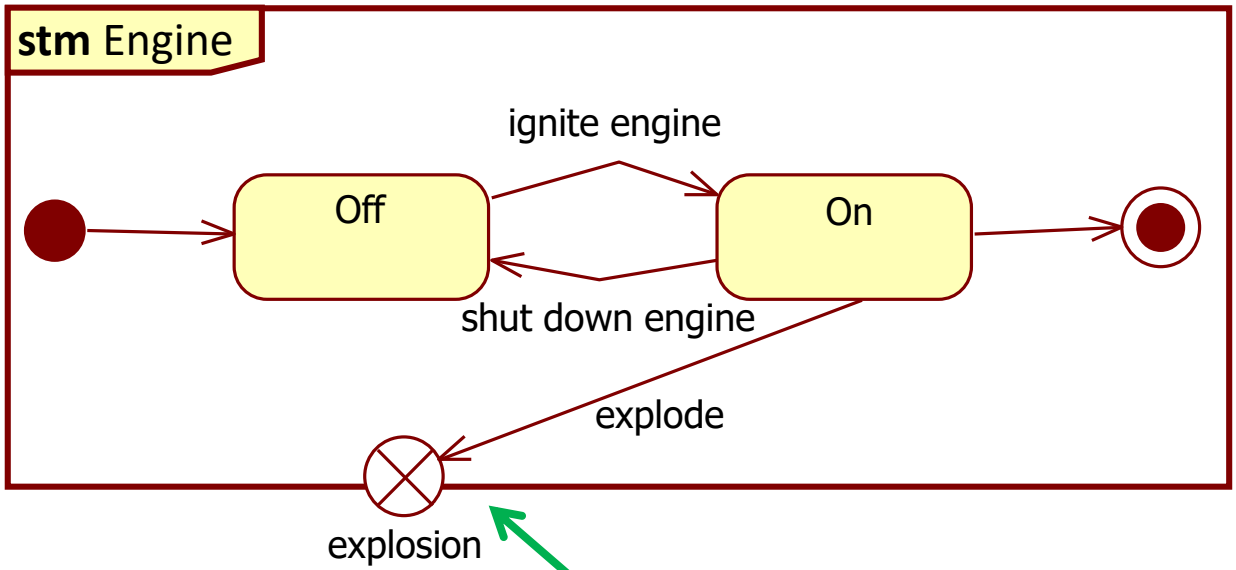


A SpaceX csomag és tartalma keretként ábrázolva:



Keret példa

Állapotdiagram az Engine osztályhoz:



A keret szükséges a kilépési pont miatt

Az UML-en túl

Az UML-en túl

- Az UML-t az Object Management Group (OMG) definiálta
- Az OMG egyéb UML-hez köthető szabványokat is definiál:
  - **Object Constraint Language (OCL)**
    - szöveges szkriptnyelv, amely segítségével metódusok viselkedése, azok előfeltételei és utófeltételei, valamint osztályok invariáns tulajdonságai is leírhatók
  - **XML Metadata Interchange (XMI)**
    - modellezőeszközök között diagramok cseréjére szolgál
  - **MetaObject Facility (MOF)**
    - az UML szabvány definiálására használt modellező nyelv
    - a MOF az UML egy részhalmaza: egy egyszerűsített osztálydiagram
    - a MOF más modellező nyelvek leírására is használható
    - a MOF saját magát is le tudja írni: a MOF a MOF segítségével van definiálva

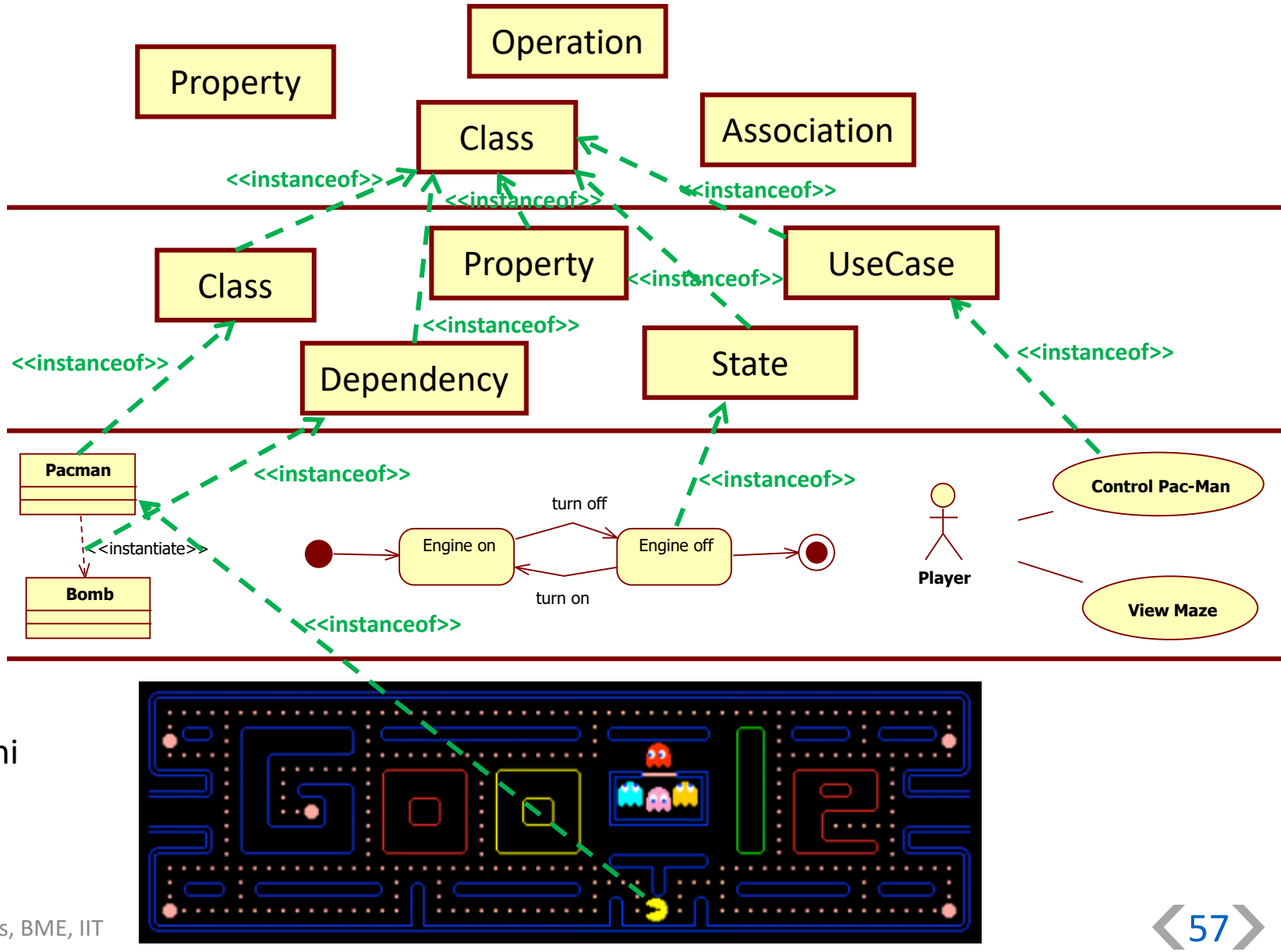
Meta szintek

MOF

UML szabvány

Általunk készített UML modellek

Futás közbeni objektumok

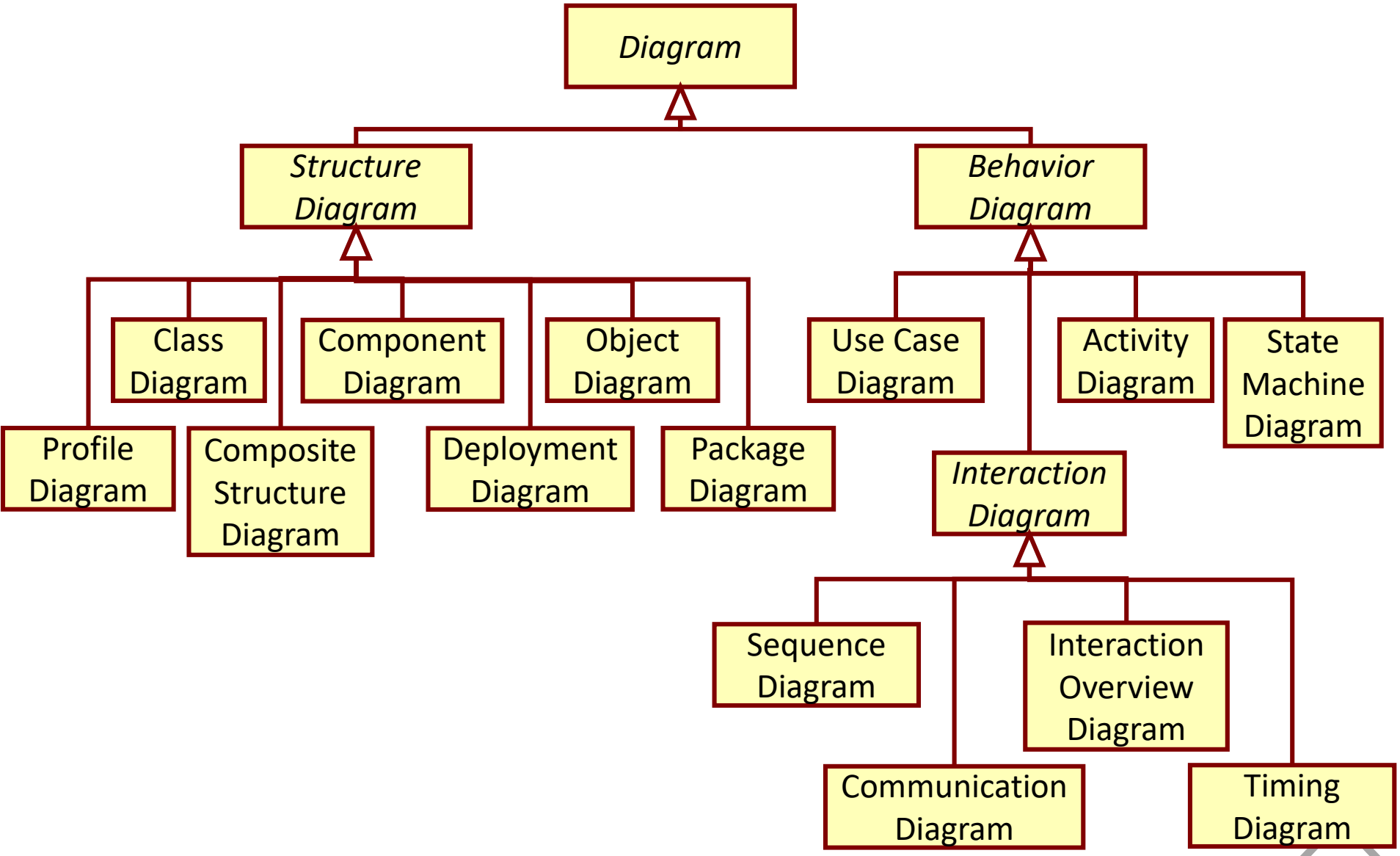


Összefoglalás

Összefoglalás

- UML diagramok:
  - Állapotdiagram
  - Időzítődiagram
  - Összetett struktúra diagram
  - Profildiagram
- Az UML diagramok összefoglalása
- Az UML-en túl:
  - Object Constraint Language (OCL)
  - XML Metadata Interchange (XMI)
  - MetaObject Facility (MOF)

UML diagramok



Hol tartunk?

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	Interakciós áttekintő diagram	