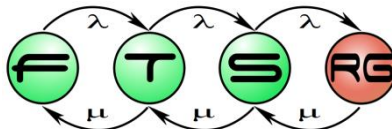


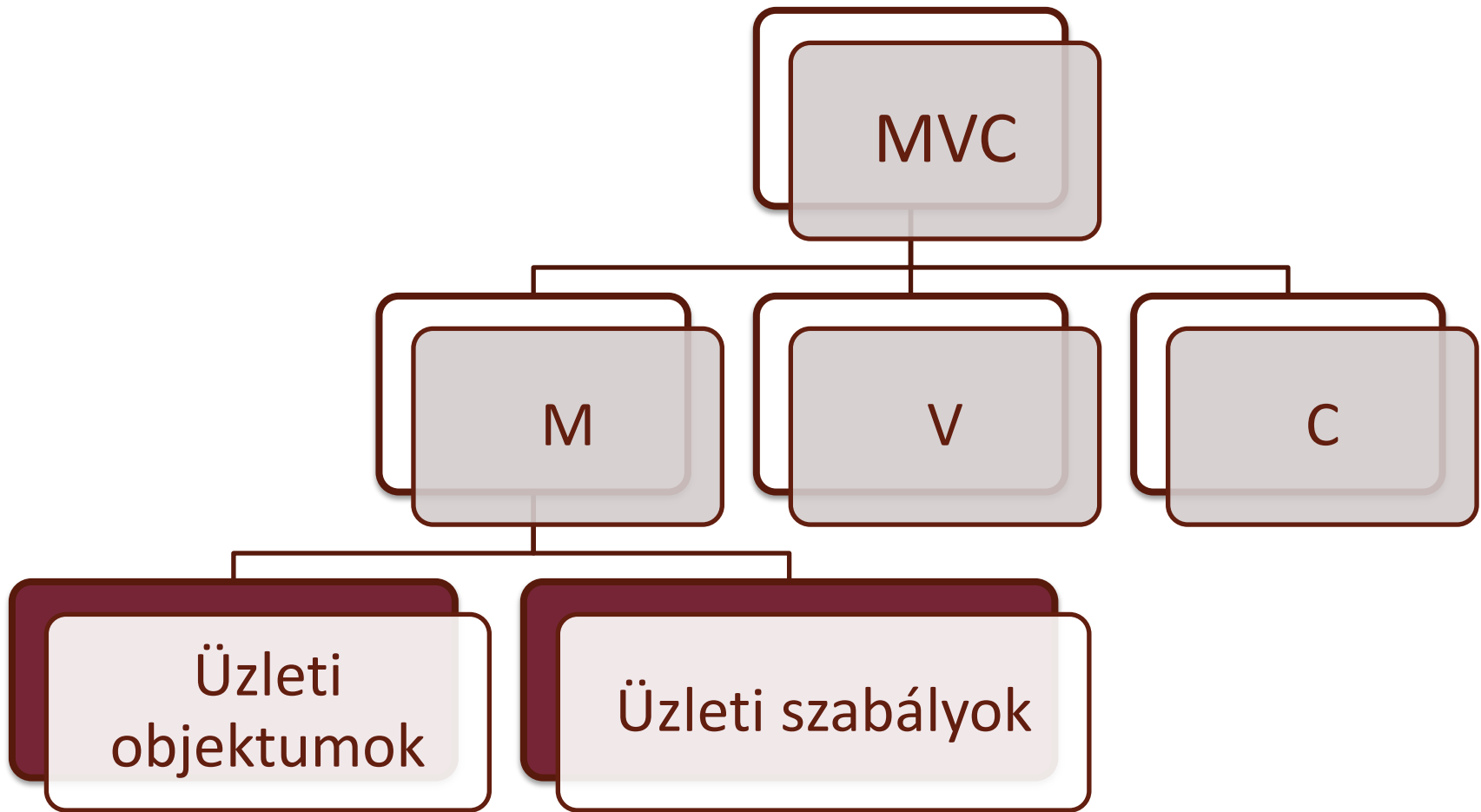
# JBoss Drools laborgyakorlat

Bergmann Gábor



# Ismétlés

- Szabály alapú üzleti logika



# Drools

- JBoss Drools nyílt forrású termékcsalád
  - **Drools Expert** – szabályvégrehajtó motor
  - Drools Guvnor – BRMS
  - ~~Drools Flow~~ jBPM 5 – üzleti folyamat végrehajtó
  - Drools Planner – megoldástér-bejárás
  - Drools Fusion – CEP



- <http://www.jboss.org/drools/documentation>

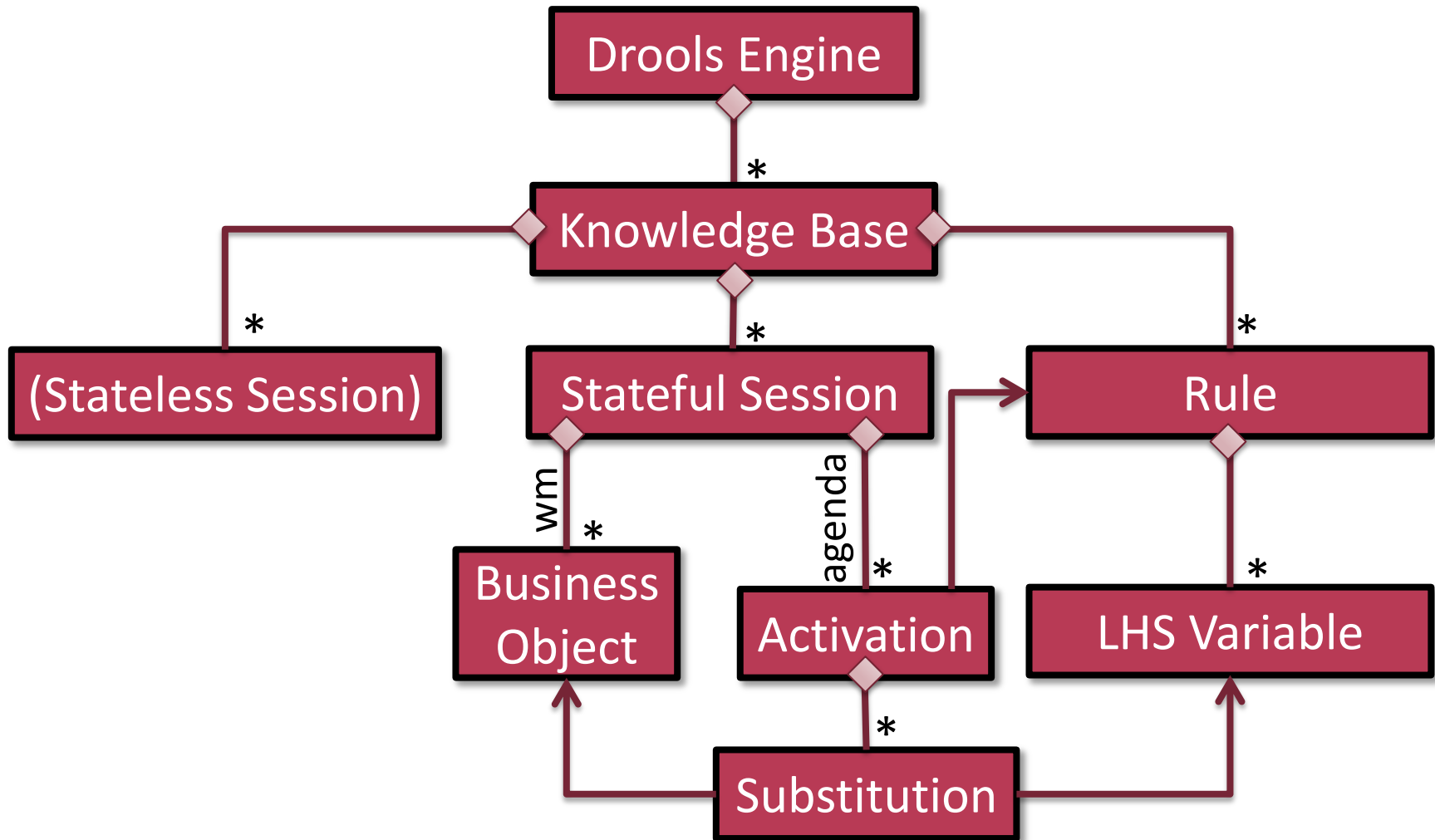
# Drools Expert alapok

- Szabályvégrehajtó motor (Java, beágyazható)
- Eclipse alapú fejlesztőkörnyezet
  - Szabály szerkesztő
  - Debug támogatás
- Szabályok bevitele
  - Kódolás (Java, mvel) → **DRL formátum**
  - Egyszerű GUI-n (Guvnor, BRL)
  - Sablon alapján (rule template)
    - „Természetes nyelvű” szövegből (DSL)
    - Döntési táblából (Excel)

# Drools Expert alapok

- Adottak tények / üzleti objektumok (POJO!)
  - „Ténybázis” / **munkamemória** (working memory, **WM**)
- Adottak produkciós szabályok („Szabálybázis”)
  - „When” / Left hand side (**LHS**) / Condition
    - Kielégítő változóbehelyettesítés (tuple) → **aktiváció**
  - „Then” / Right hand side (**RHS**) / Action
- Szabályok tüzelése, amíg van tüzelhető
  - Aktivációk → **agenda** (conflict set)
  - **Inkrementális mintaillesztés** (RETE)
  - Aktiváció kiválasztása (konfliktusfeloldás) → tüzelés

# Drools Expert architektúra



# Szabály felépítés

- Fejléc: név
  - + opcionális beállítások (pl. **salience**)
- LHS: deklaratív lekérdezés
  - **Objektumminták**, attribútum-korlátozásokkal
    - Az objektum és az attribútumai megköthetőek **változóban**
    - Pl. `$szilva: Szilva(suly<8, $id: id!=$masik_id)`
  - **eval(...)** → további kényszerek a változókra
  - Kvantorok: **not**, **exists**, **forall** (ld. később)
- RHS: Java parancsok, felhasználva a változókat
  - WM manipulációra speciális támogatás (ld. köv. dia)

Az aktivációban ezek lesznek behelyettesítve

# Working Memory kezelés

- Inkrementális mintaillesztés → változásokra frissül
  - WM változtatás csak a Drools API-n keresztül
  - Különösen: objektumok attribútumainak módosítása!
- Session API: WM manipuláció
  - `insert(obj)` → `FactHandle`
  - `retract(fHandle)`, `update(fHandle, newObj)`
- RHS támogatás
  - **insert**(obj) és **retract**(obj)
  - **update**(obj) vagy **modify**(obj) {setX(), ...}
  - **insertLogical**(obj) → automatikus visszavonás





# Drools Expert hibakeresési támogatás

- **Audit View – logelemzés**
  - Végrehajtott szabálypéldányok
  - Elvégzett WM módosítások
  - Megjelenő / eltűnő aktivációk
  - Kell: Logger nyitás + lezárás, Audit nézet rákapcsolás
- **Debug mode: Working Memory / Agenda View**
  - Töréspontnál WM tartalom / aktivációk
  - session-re rá kell mutatni, hogy működjenek
  - Debug as Drools Application → RHS töréspont is lehet

The logo for Drools, featuring a stylized head profile on the left composed of red, blue, and orange lines, followed by the word "Drools" in a bold, dark blue, distressed font.

# Kvantorok

- **Kielégíthetlenség**
  - **not** feltétel
  - A kvantált elemekkel sehogy se tehető igazzá
- **Egzisztenciális kielégíthetőség**
  - **exists** feltétel
  - Akárhány megoldása is van, csak egy aktivációt szül
- **Univerzális állítás**
  - **forall** kire `mi_lesz_igaz`
  - első feltétel kielégíthető → továbbiak is kielégíthetőek
- **Aggregáció** (összeg, darabszám, átlag, stb.)
- **Feltételek:** zárójelen belül **and**, **or** használható

# Attribútum-korlátozások

- Attribútum: ami getterrel látható (JavaBean)
  - + **this**
- Korlátozás: összehasonlítás egy kifejezéssel
  - Drools-változók is használhatóak
  - Java összehasonlítás (pl.  $\geq$ ,  $==$ ) + kiegészítések
- Drools-specifikus összehasonlító operátorok
  - Kollekciónak elemvizsgálata
    - **memberOf**, **not memberOf**, **contains**, **not contains**
  - Regexp illesztés stb.
- Zárójelezés, **&&**, **||**

# Komplex szabály

```
rule "Cukroz"
```

```
  when
```

```
    $szilva: Szilva ()
```

```
    ## a pulton van a szilva
```

```
    $pult: KonyhaPult (tartalom contains $szilva)
```

```
    ## ures a szilva
```

```
    not Object (this memberOf $szilva.tartalom)
```

```
  then
```

```
    KockaCukor $cukor = new KockaCukor();
```

```
    insert($cukor);
```

```
    modify($szilva) {
```

```
      berak($cukor);
```

```
    }
```

```
end
```

# Tüzelés vezérlése

- Több szabály aktiválódott → prioritásos döntés
  - Nagyobb prioritású előbb tüzel; alapesetben 0
  - Fejlécbe **salience** opció, pl. **salience 100**
- Végtelen ciklus elkerülése  
(RHS lefutás után engedélyezve maradó szabály)
  1. Aktiváció automatikusan törlődik tüzelés előtt 😊
    - egy **update()** vagy **modify()** után újra megtalálja ☹️
  2. Fejlécbe **no-loop** opció: utólagos törlés 😊
    - másik szabályból **update()** után mégis megtalálja ☹️
    - de **ruleflow-group** esetén **lock-on-active** jó 😊
  3. Valódi megoldás: aktiváció megszüntetése 😊

# Dokumentáció

- <http://www.jboss.org/drools/documentation>



# Köszönetnyilvánítás

Az oktatási anyag elkészítése az Európai Unió és Magyarország támogatásával a TÁMOP 4.2.4.A/1-11-1-2012-0001 azonosító számú „Nemzeti Kiválóság Program – Hazai hallgatói, illetve kutatói személyi támogatást biztosító rendszer kidolgozása és működtetése országos program” című kiemelt projekt keretei között valósult meg.