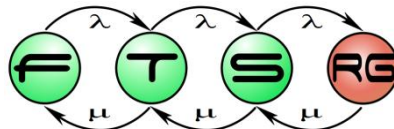


Szabályalapú üzleti logika, eseményfeldolgozás

Gönczy László

Bergmann Gábor

Rendszermodellezés 2013.



Tartalom

- Produkciós rendszerek alapfogalmai
- Üzleti szabályrendszerek
- Komplex események feldolgozása (CEP)
- Esettanulmányok

Produkcións rendszerek alapfogalmai

Szabály alapú működés

- Deklaratíván specifikált viselkedés
 - imperatív utasítássorozat helyett
 - „ha-akkor” szabályokkal
- Hol találkozunk szabály alapú viselkedéssel?
 - Tűzfal konfiguráció / routing tábla
 - MAKEFILE
 - Cron
 - Szakértő rendszerek (expert systems)
 - Diagnosztika, stb...
 - ...

Egy lehetséges kategorizálás

Szabály alapú (rule based) rendszerek

Következtető gépek
(inference engines)

Előre láncoló /
produkciós

Tiszta logikai

Üzleti
szabálymotor

Hátra láncoló

Prolog,
stb.

Tűzfal, stb.

Szabály alapú következtető gépek

- **„Tudásbázis”** (knowledge base)
 - **„Ténybázis”** (fact base) / munkamemória (WM)
 - Változatos felépítés
 - **„Szabálybázis”** (rule base)
 - Szabályok, amelyekkel új tudást lehet kapni
 - „Ha”: feltétel rész, precondition, bal oldal (LHS)
 - „Akkor”: következmény rész, postcondition, jobb oldal (RHS)
- **Végül egy következtető mechanizmus**
 - Előre vagy hátra láncoló
 - Előre láncoló: logikai következtetés vagy üzleti szabályok

Példák

- Szakértői rendszer (pl. orvosi)
 - „Ha egy szerv gyulladt, akkor fájdalmat okozhat”
 - „Ha egy szerv gyulladt és aszpirin van a vérben, csökken a gyulladás”
 - „Fáj a lábam, mi minden okozhatja?”
 - „Ha bevennék aszpirint, mi lenne a következménye?”
- Üzleti szabályok
 - „Ha az ügyfél sokat roamingol, ajánljunk más tarifát”
 - „Ha a járat egyik buszvezetője a többihez képest kiugróan kevés jegyet értékesít, ellenőrizzük”

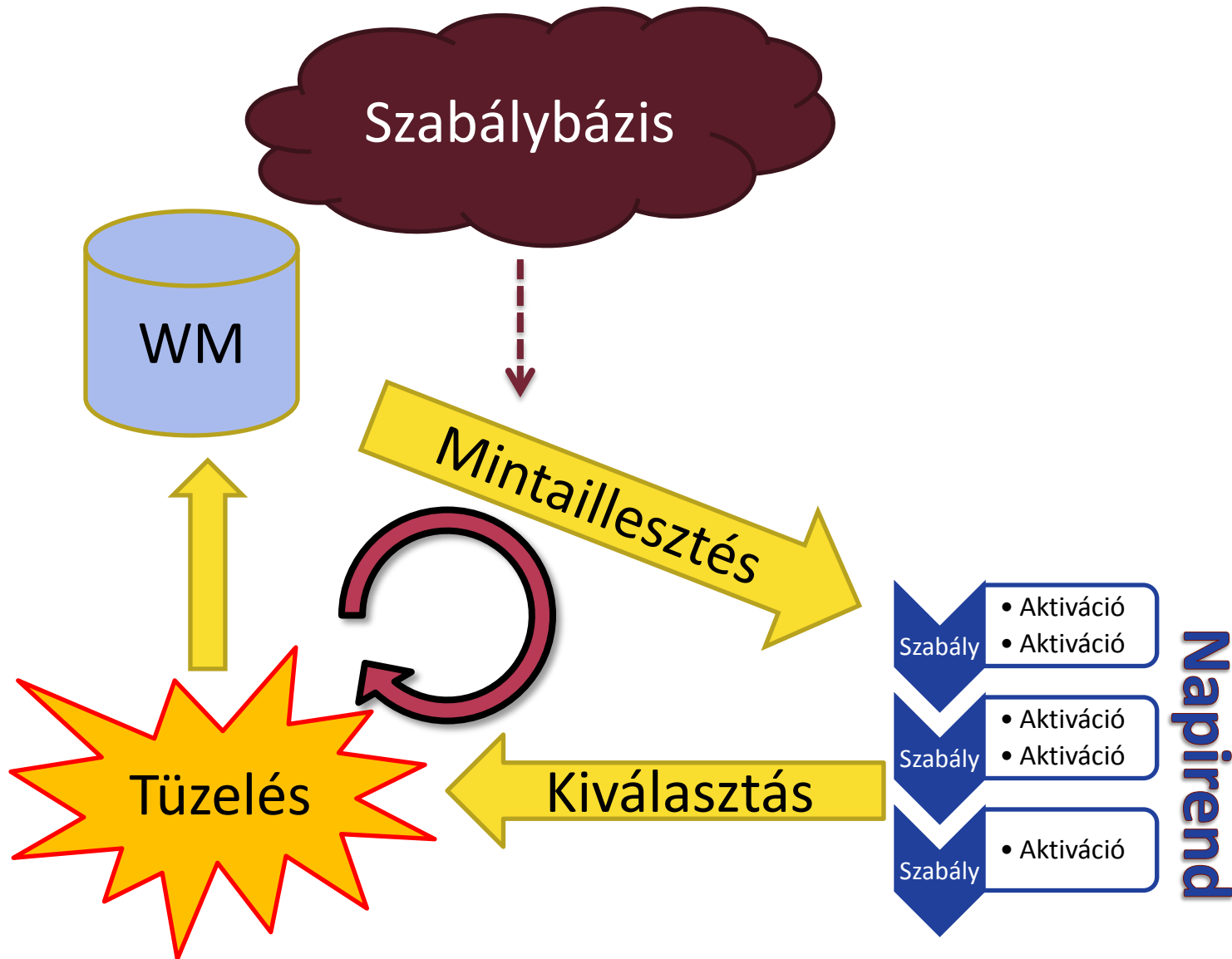
Következtetés

- **Előre** láncoló (induktív/*produkciós*, adatvezérelt)
 - A tényekből újabb tényeket képez (produkciós szabály)
 - Egy következmény teljesítheti egy szabály feltételrészét
 - Analógia: generatív nyelvtan, hatáselemzés
 - Ilyenek például a üzleti szabályrendszerek
 - Logikai következtetés (vs. üzleti szabály)
 - ha a feltétel érvénytelenné válik, a következmény is?
- **Hátra** láncoló (deduktív, igényvezérelt)
 - Egy cél-állítást próbál visszavezetni alaptényekre
 - Analógia: parser, diagnosztika
 - Ilyen például a *Prolog* és számos szakértői rendszer

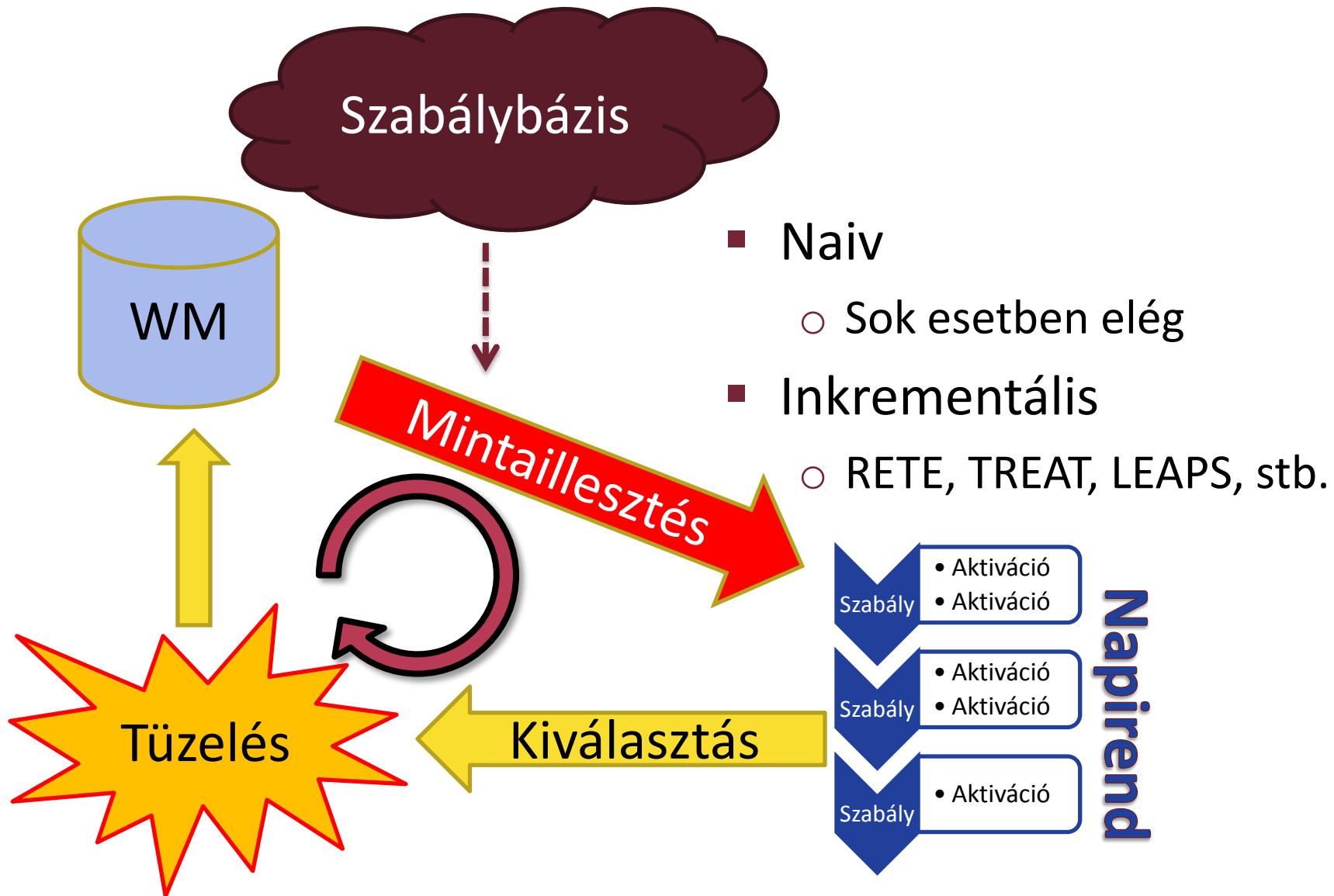
Produkción rendszer fogalomtár

- Munkamemória (working memory, **WM**)
 - Folyamatosan változó ténybázis
- **Aktivált** (activated, triggered) produkciós szabály
 - Minden feltétele ki van elégítve, tüzelhet
- **Aktiváció**
 - Szabály LHS egy konkrét kielégítő behelyettesítése
 - „n-es” (tuple), minden lekötetlen változóhoz egy érték
- **Tüzelés** (firing)
 - Szabály konkrét végrehajtása egy adott aktivációra
- **Napirend** (agenda, conflict set)
 - Összes (tüzelésre váró) aktiváció

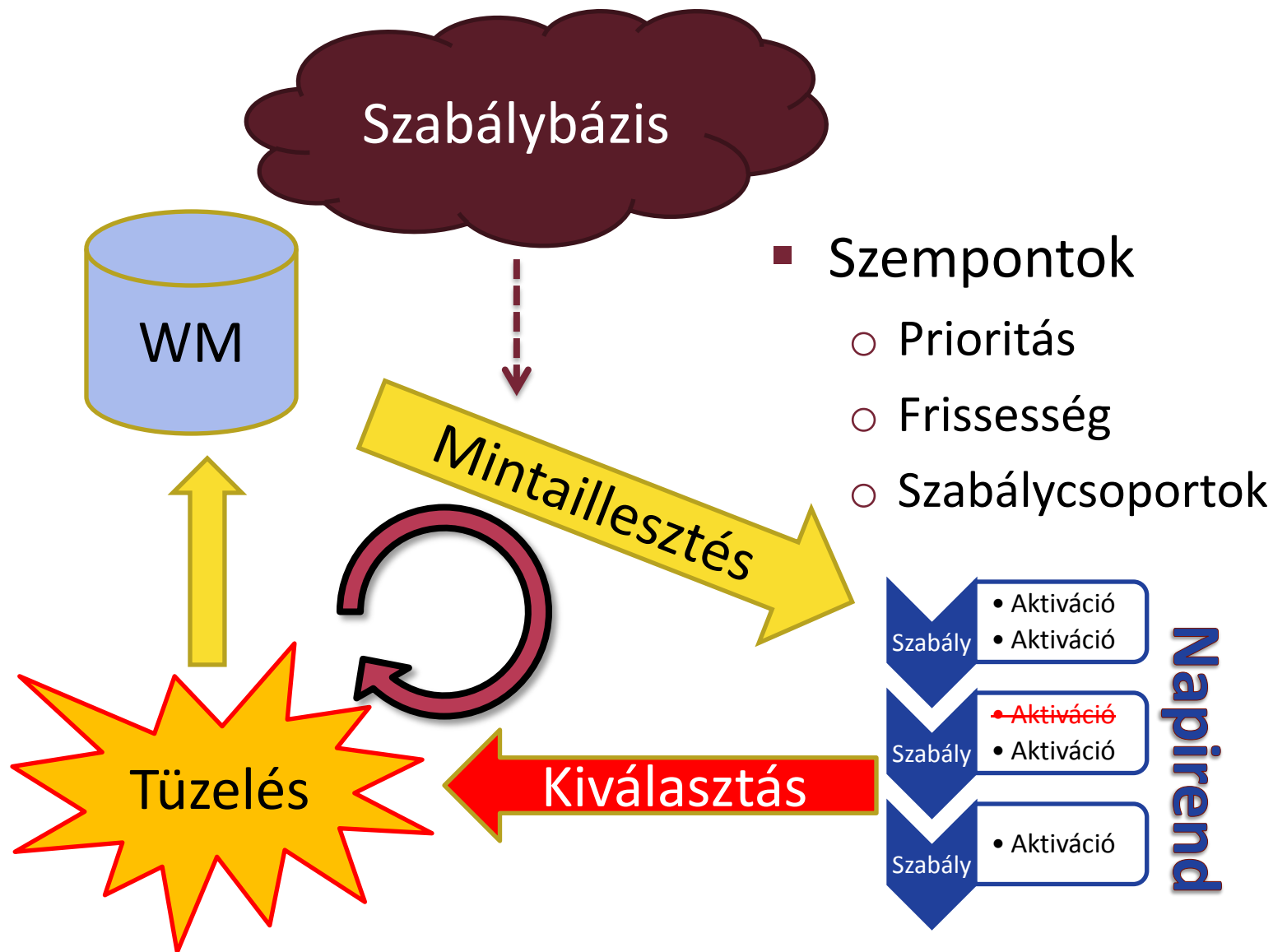
Egyszerű produkciós rendszer



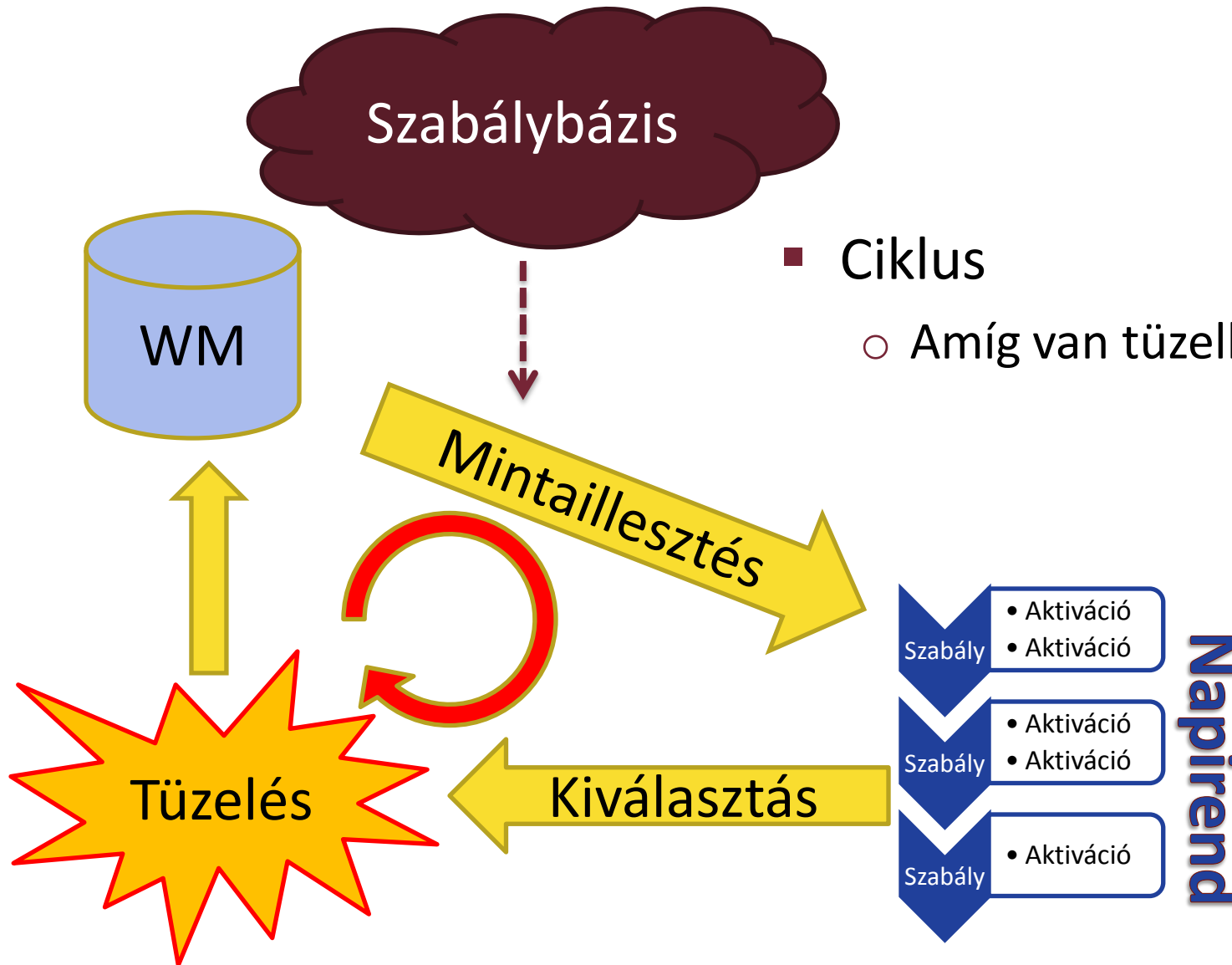
Egyszerű produkciós rendszer



Egyszerű produkciós rendszer



Egyszerű produkciós rendszer



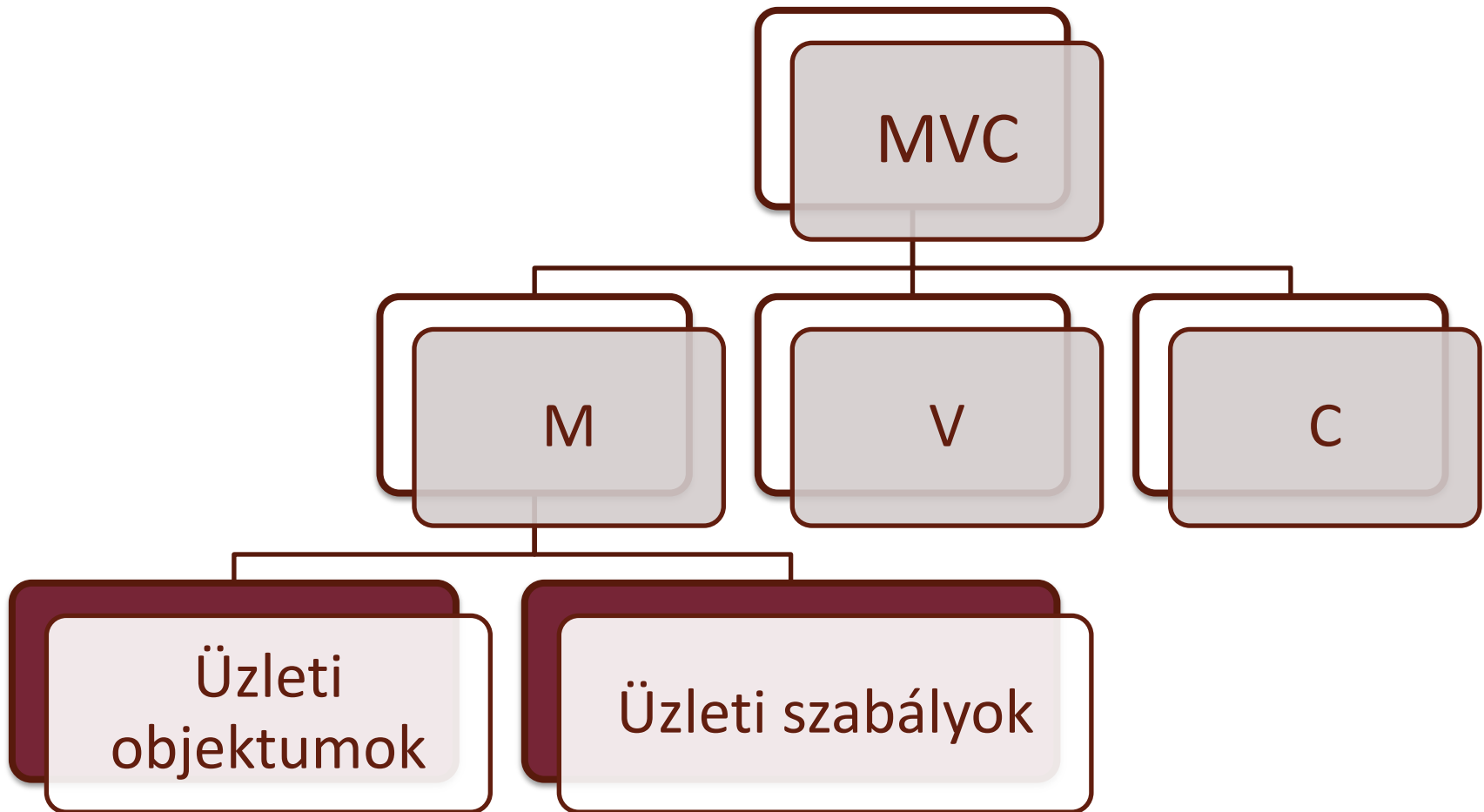
- Ciklus

- Amíg van tüzelhető szabály

Üzleti szabályrendszerek

Business Rule Systems

- Szabály alapú üzleti logika



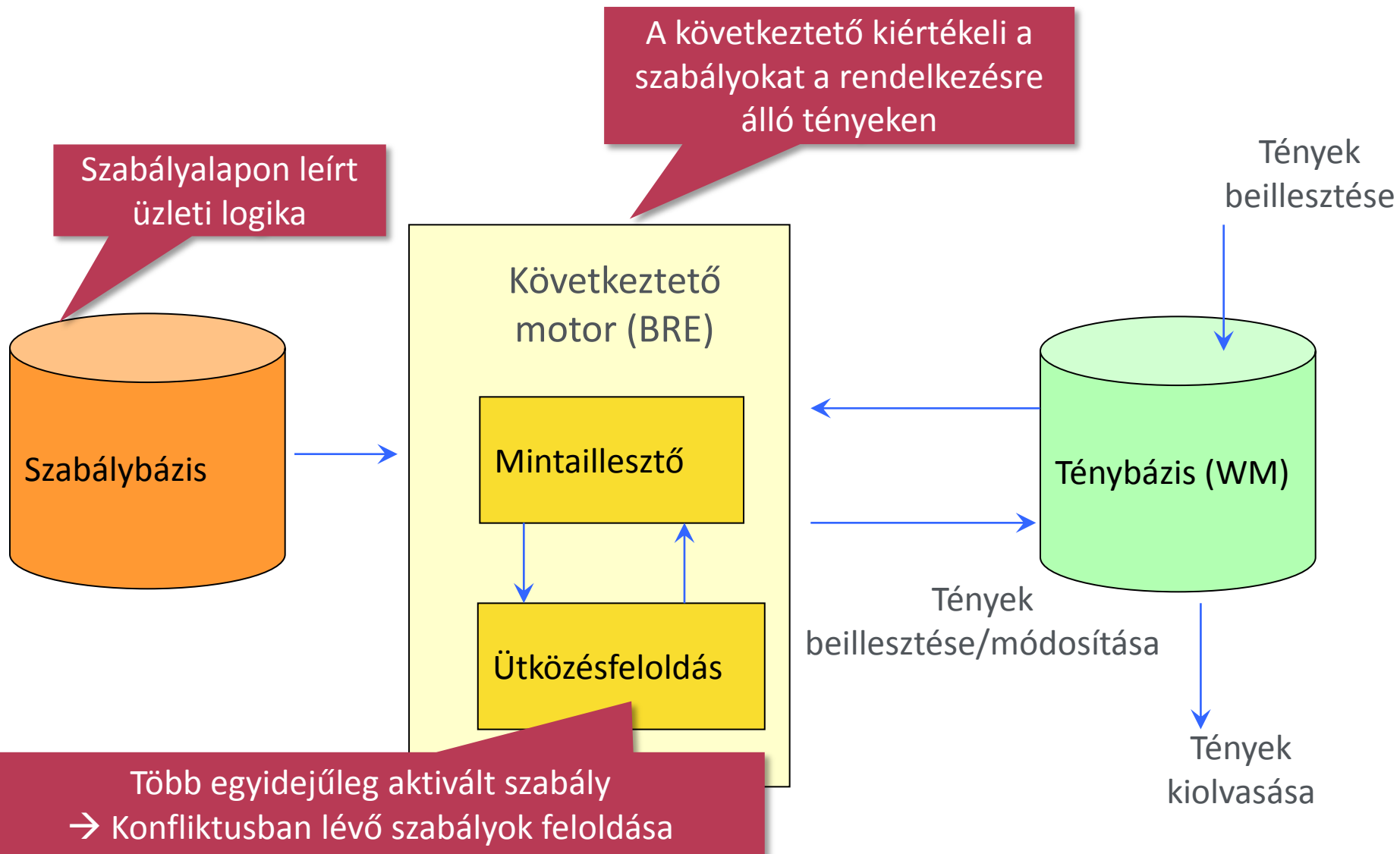
Üzleti szabályok

- **Üzleti logika** „kiszervezésére” végrehajtható modell
- **Üzleti objektumokat** figyelhet, manipulálhat
- Felépítése: ha → akkor
 - „ha az ügyfél 30 év alatti, emeljük 35%-al az ajánlatot”
 - „ha az ügyfél egyenlege 500Ft alá csökkent, értesítsük”
 - „ha más ügyfél korábban bejelentkezett már azonos lakcímre, nem adunk kedvezményt”
 - „ha a hallgatónak legalább húsz lezárt féléve van, nem szerzett aláírást diplomatervezésből és nem kapott köztársasági elnöki engedélyt, akkor megszüntetendő a jogviszonya, feltéve hogy ötéves képzésre jár és az ezt előíró jogszabály hatályba lépése óta kezdte tanulmányait”

Üzleti szabálymotor

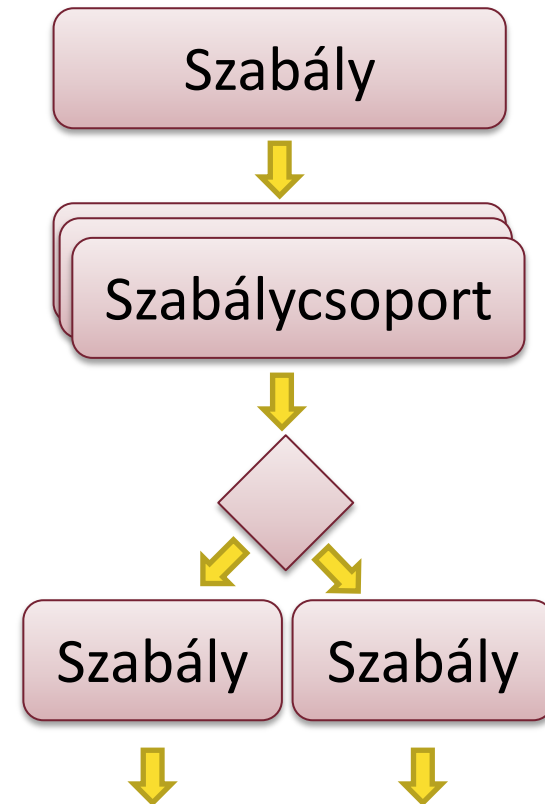
- Üzleti szabályok produkciós rendszer szemszögből
 - „Tények” → üzleti objektumok
 - Kvázi produkciós szabályok, de RHS tetszőleges **akció**
 - Nem (feltétlen) logikai következtetés
 - Érvénytelenné váló feltétel, akció hatása mégis megmarad
 - Egy aktiváció többször is tüzelhet (pl. addig jár a korszó...)
- Üzleti szabálymotor (Business Rules Engine, **BRE**)
 - Üzleti szabályokat végrehajtó szoftver
 - Produkciós rendszer, a matematikai háttértől elvonatkoztatva, programozási platformként
 - Kapcsolat a külvilággal: WM, vagy akciók

Üzleti szabálmotor működése



BRE vezérlése

- Alapértelmezett: tüzelési ciklus
 - Amíg van még tüzelhető szabály
 - Vagy STOP szabályig
- Komplex rendszer: vezérlési folyamat
 - Pl. jBPM workflow
 - Kiválthatja a bemutatott ciklust
- Eseményvezéreltség is elképzelhető
 - „Alvó” szabályok
 - Külön utasítás nélkül



Egyszerű Drools szabályok

```
rule "We have an honest Politician"
```

```
  salience 10
```

```
  when
```

```
    exists( Politician( honest == true ) )
```

```
  then
```

```
    insertLogical( new Hope() );
```

```
end
```

```
rule "Hope Lives"
```

```
  salience 10
```

```
  when
```

```
    exists( Hope() )
```

```
  then
```

```
    System.out.println("Hurrah!!!  
    Democracy Lives");
```

```
end
```

```
rule "Hope is Dead"
```

```
  when
```

```
    not( Hope() )
```

```
  then
```

```
    System.out.println( "We are all  
    Doomed!!! Democracy is Dead" );
```

```
end
```

```
rule "Corrupt the Honest"
```

```
  when
```

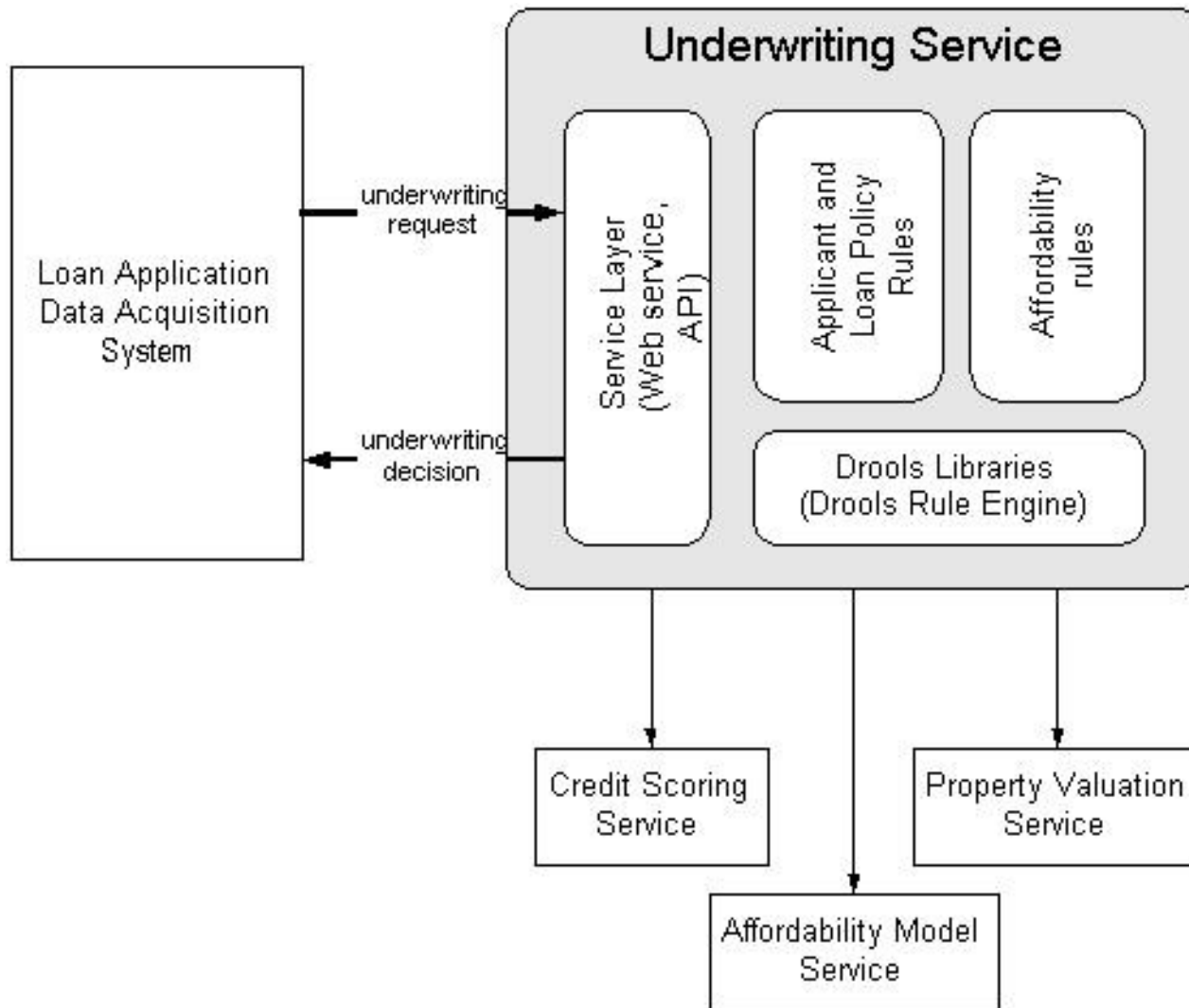
```
    politician : Politician( honest == true )  
    exists( Hope() )
```

```
  then
```

```
    System.out.println( "I'm an evil  
    corporation and I have corrupted " +  
    politician.getName() );  
    modify( politician ) {  
      setHonest( false )  
    }
```

```
end
```

Példa alkalmazás



<http://onjava.com/onjava/2007/01/17/building-enterprise-services-with-drools-rule-engine.html>

BRMS

- BRMS = **Business Rule Management System**
- BRE + kapcsolódó szolgáltatások
- Számos termék
 - *G2, JBoss Rules (**Drools**), IBM ILOG (J)Rules, Blaze Advisor, MS BRE, TIBCO iProcess, OPA stb.*

Microsoft[®]

ILOG
Changing the rules of business

Fair Isaac

gensym

Drools

TIBCO[®]
The Power of Now[®]

■ Szabálytár

- Kereshető, automatizáltan módosítható
- Verziózás

■ Végrehajtó könyvtár (BRE) → végrehajtó szerver

■ Tool support

- IDE, webes felület
- Template lehetőség, döntési tábla
- Magasabb granularitású szabályok
- Tesztelési támogatás, gyors próba
- Üzleti szótár építése meglévő adatokból

Szabály alapú üzleti logika előnyei

- Dedikált szabálytár → karbantarthatóság
 - Üzleti logika könnyebben módosítható
 - Pont ez **változhat** leggyakrabban: új rendeletek, stb.
- **Redundancia elkerülése**
 - Ugyanaz az üzleti logika sok modulban megjelenhet
- Jó esetben az **üzleti döntéshozók** is tudják olvasni
 - Sőt, akár írni is: természetes nyelvi verbalizáció, spreadsheet alapú szabálygenerálás
- Hatékony végrehajtás (inkrementális mintaillesztés)
- Cserélhető körülötte az architektúra
- Eszköztámogatás

Szabály alapú üzleti logika hátrányai

- Sorrendiség körülményesebb
 - V.ö. imperatív programnyelvekkel
 - Megoldás: integráció workflow motorral (ld. Drools) ?
- Univerzális absztrakciós nehézségek
 - Túl elvont nyelv → bizonyos feladatokra alkalmatlan
 - Nem elég elvont → nem is egyszerűbb, mint a Java
 - „Szivárgás” (law of leaky abstractions)
- Alkalmazási tapasztalatok nem mindig pozitívak
- Human factor
 - „... néha nem árt megkerülni a szabályt”

Felhasználási területek - példák

- Biztosítók, bankok
 - Kalkulációk kiemelése
 - Szabályok következetes kikényszerítése
 - Ügyek elbírálásának támogatása
- E-Kormányzat
 - Regisztráció kiértékelése
 - Adó, járulékszámítás
- Logisztika
 - Szállítmányozási döntések támogatása

Példa szabálydefinícióra: Drools döntési tábla

■ Döntési tábla – forrás: spreadsheet

○ Sok hasonló szabály

- „ha <30 éves és legalább 2 éve ügyfél, kapjon 25%-ot”
- „ha 31-49 éves és legalább 3 éve ügyfél, kapjon 17%-ot”
- ...

○ Eltérő paraméterek (feltételek, akció részei)

- Akár kifejezés, pl. >30

○ Üzleti döntéshozó által meghatározandó

B	C	D	E	F	G	H
1						
2	RuleSet	org.acme.insurance.base				
3	import	import org.acme.insurance.base.Approve, import org.acme.insurance.base.Driver				
4	Package	org.acme.insurance.base				
5						
6	RuleTable Old Driver					
7	CONDITION	RULEFLOW-GROUP	NO-LOOP	ACTION	ACTION	
8	\$driver: Driver					
9	licenceYears	priorClaims		insert(new Approve("\$param"));	System.out.println("Spa	
10	Persons age	Prior Claims		Inserting approval	Log	
11	30	1	risk assessment	Safe and mature	Old driver Approved	
12						
13						
14						
15						
16						

Esettanulmány: szabálykiértékelés párhuzamosítása GPU felett

Központi Szabálybázis Felhő kutatás-fejlesztési projekt

(GOP-1.1.1-11-2012-0216, Profitexpert Kft.)

„Párhuzamos feldolgozásra alkalmas szabálybázis motor kutatása”

A feladat

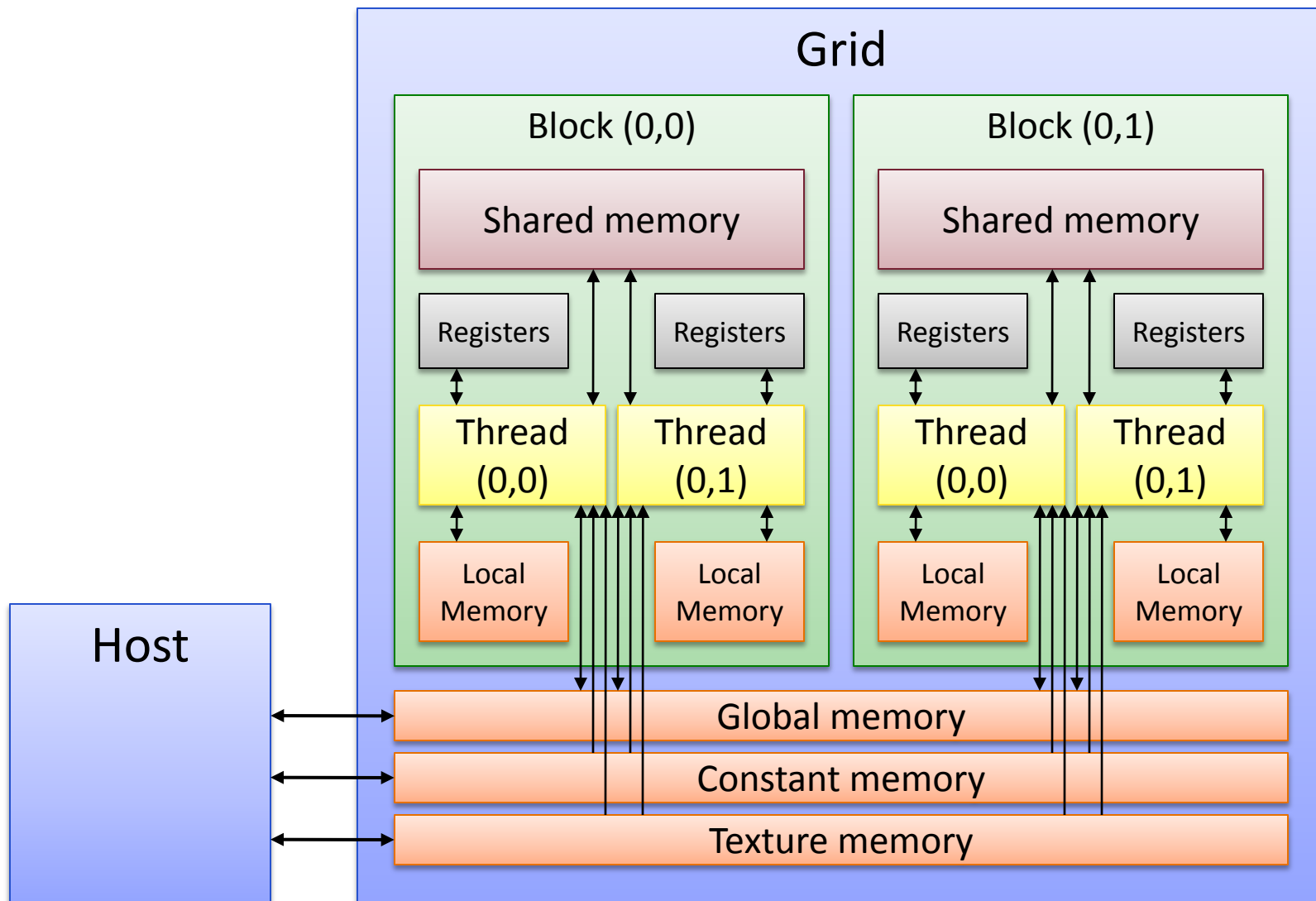
- Adott szabályvégrehajtó eszköz (Oracle Policy Automation) működését párhuzamosítani
 - GPU felett (CUDA)
 - OPA input/output feldolgozásával
 - OPA-val egyező eredményt adva
- ... lényegében újraírni
 - Szabályok feldolgozását (sorrendiség)
 - Input feldolgozást
 - kiértékelést
- Modellalapú tervezés vs szabálykiértékelés vs párhuzamosítás vs teljesítménymérés....

CUDA röviden

- NVIDIA által kifejlesztett platform
 - Sokmagos grafikus kártyák
 - Saját programozói interfész
- Általános célú kód végrehajtása a grafikus processzoron
 - Nyelvi kiterjesztések használata (C,C++,Fortran, ...)
 - Ezeket támogató speciális fordító
- Jelentős teljesítménynövekedés, ahol
 - Párhuzamos végrehajtást jól ki lehet használni
 - Adat-hozzáférésre és végrehajtásra nézve is jól darabolható



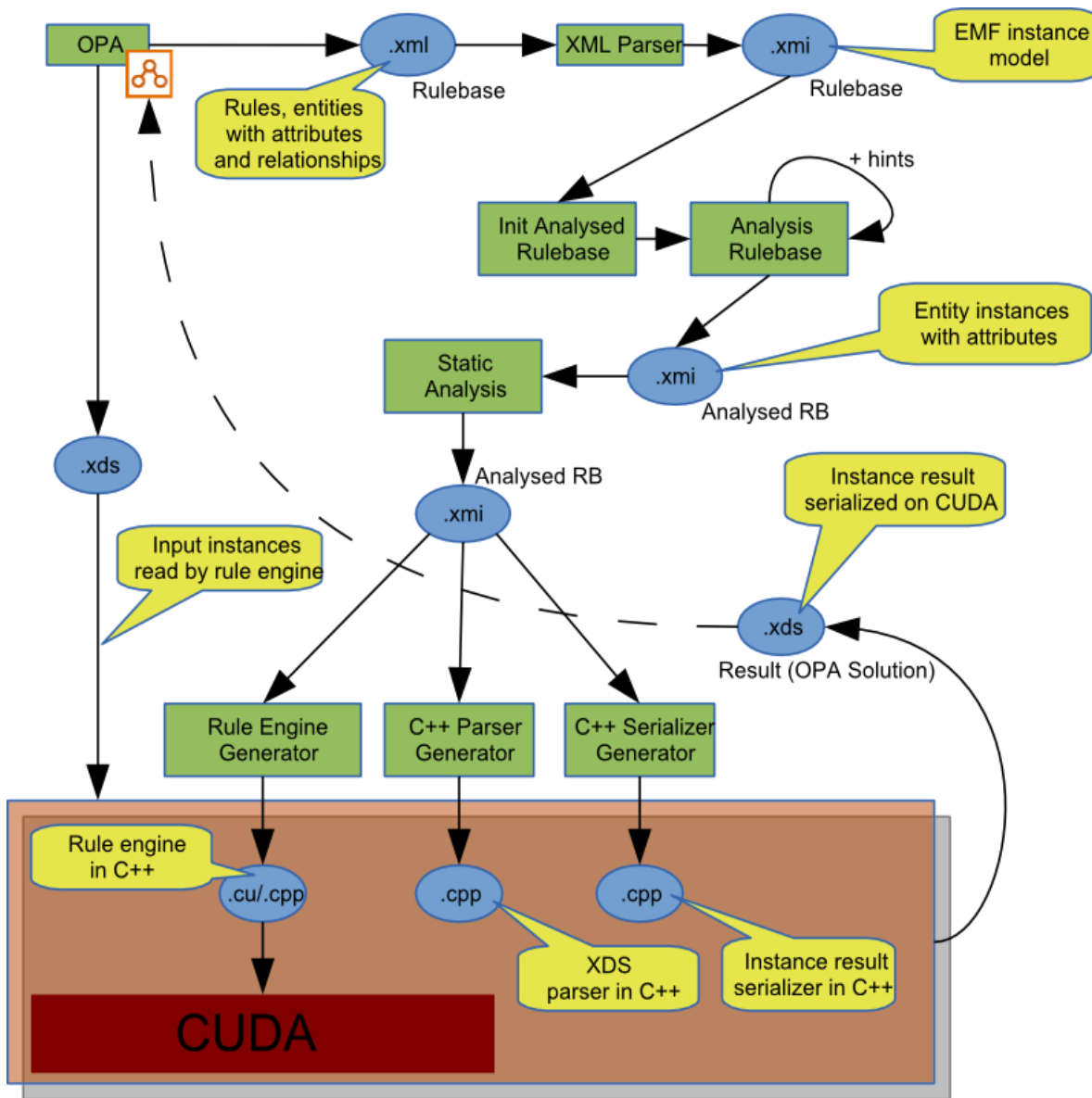
Architektúra - memória



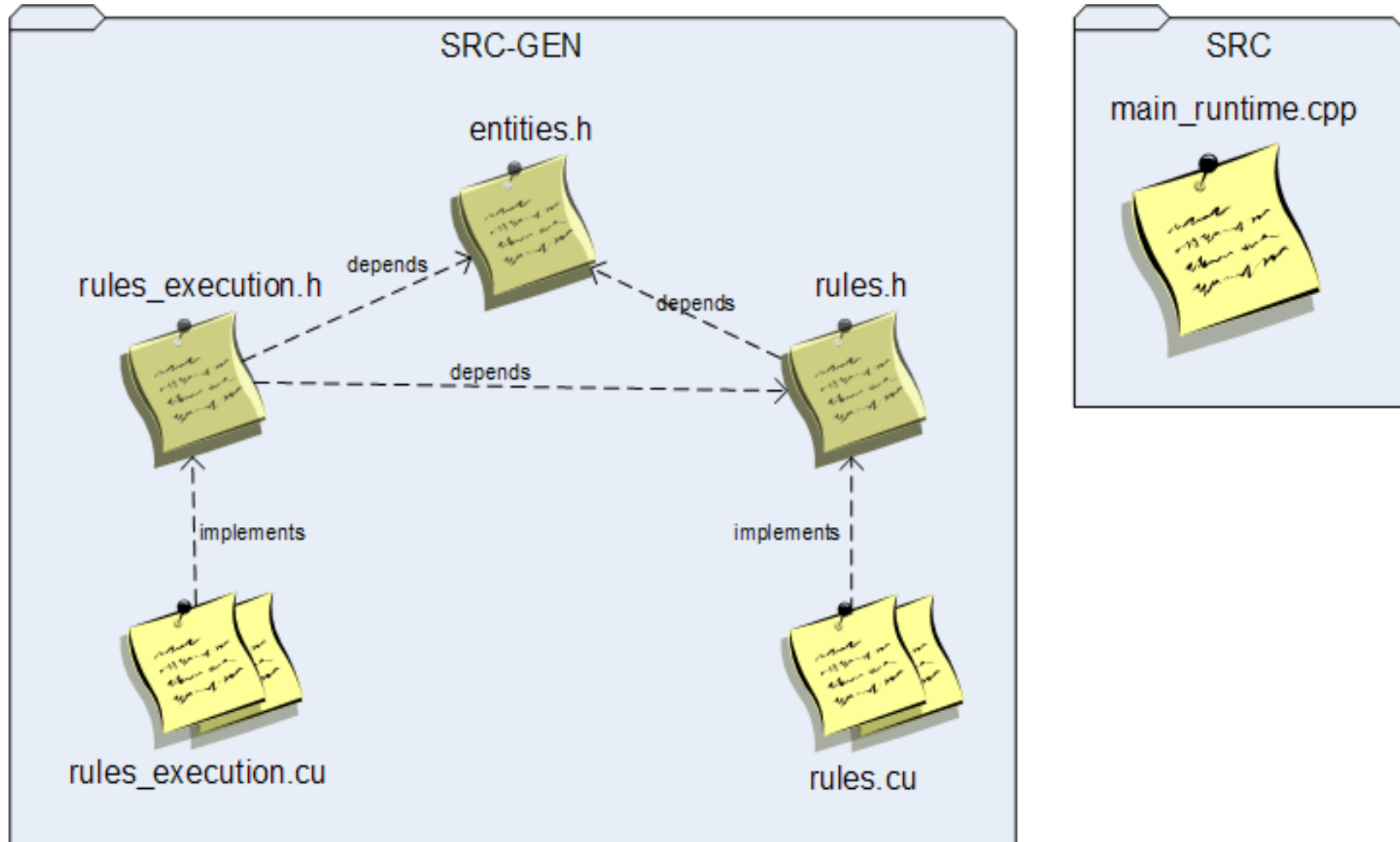
Szabálymotor funkcionalitás

- Java alapú kódgenerálás
- Szabálybázis és entitásmodell független
- Szabályok (attribútumok) alapján szétosztás CPU/GPU közt
 - Memóriaműveletek számának csökkentése
- Szabályok szakértői „jelölése”
- Átlátszó működés (OPA input/output)

A prototípus szabálmotor működése



CUDA kód felépítése



Felhasznált technológiák

- Eclipse (Java alapú, ingyenes)
 - EMF (adattárolás)
 - EMF IncQuery (mintaillesztés statikus analízishez)
 - XTend (kódgenerálás)
 - CDT (C++ fejlesztés)
 - NSight Eclipse Edition (CUDA)
- MinGW (C/C++ fordító)

A prototípus értékelése

- Tervezési időben többletköltség
 - Méretek: Mintapélda: 440.000 sor generált kód csak a szabály
 - Futási idő: C++ build időigényes
- Tesztelhetőségre
 - Tesztgenerátor (szabálybázishoz entitások)
 - OPA/CUDA kimenet összehasonlítás
- Futási időben megtakarítást hozhat
- Transzparens működés