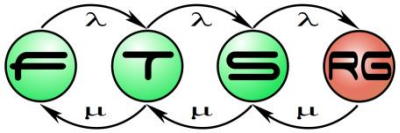
Szabályalapú üzleti logika

Bergmann Gábor
bergmann@mit.bme.hu
Gönczy László anyagainak felhasználásával



Tartalom

- Produkciós rendszerek alapfogalmai
- Üzleti szabályrendszerek
- Drools

Produkciós rendszerek alapfogalmai

Szabály alapú működés

- Deklaratívan specifikált viselkedés
  - imperatív utasítássorozat helyett
  - „ha-akkor” szabályokkal
- Hol találkozunk szabály alapú viselkedéssel?
  - E-mail automatikus szűrőszabályok
  - Tűzfal konfiguráció / routing tábla / cron
  - MAKEFILE
  - Szakértő rendszerek (expert systems)
    - Diagnosztika, stb…
  - …

Egy lehetséges kategorizálás

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, küldjünk rá ellenőrt”

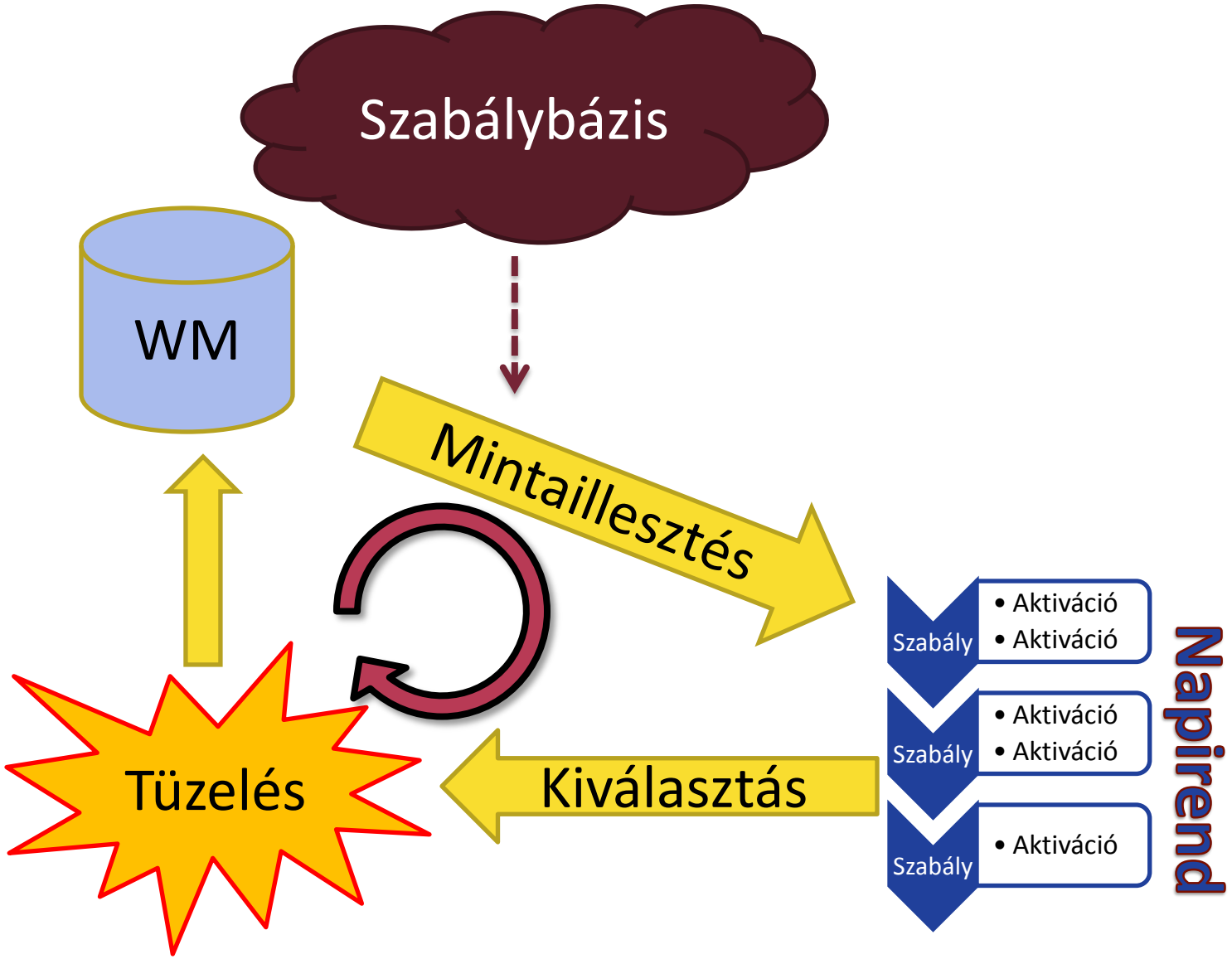
Következtetés

- **Előre** láncoló (deduktív/*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 (abduktív, igényvezérelt)
  - Egy cél-állítást próbál visszavezetni alaptényekre
  - Analógia: parser, diagnosztika
  - Ilyen például a *Prolog* és számos szakértői rendszer

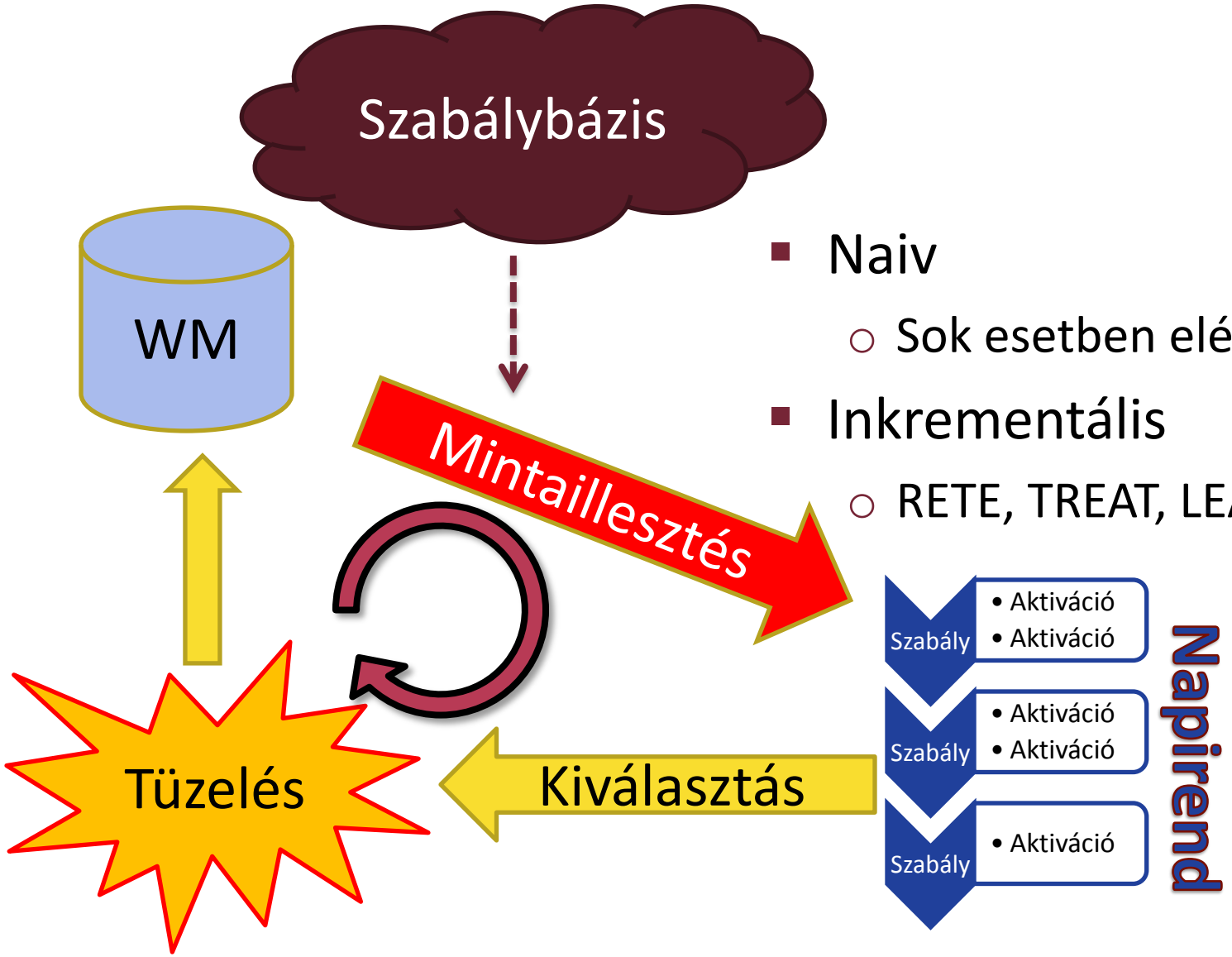
Produkciós rendszer fogalomtár

- 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

Tipikus produkciós rendszer

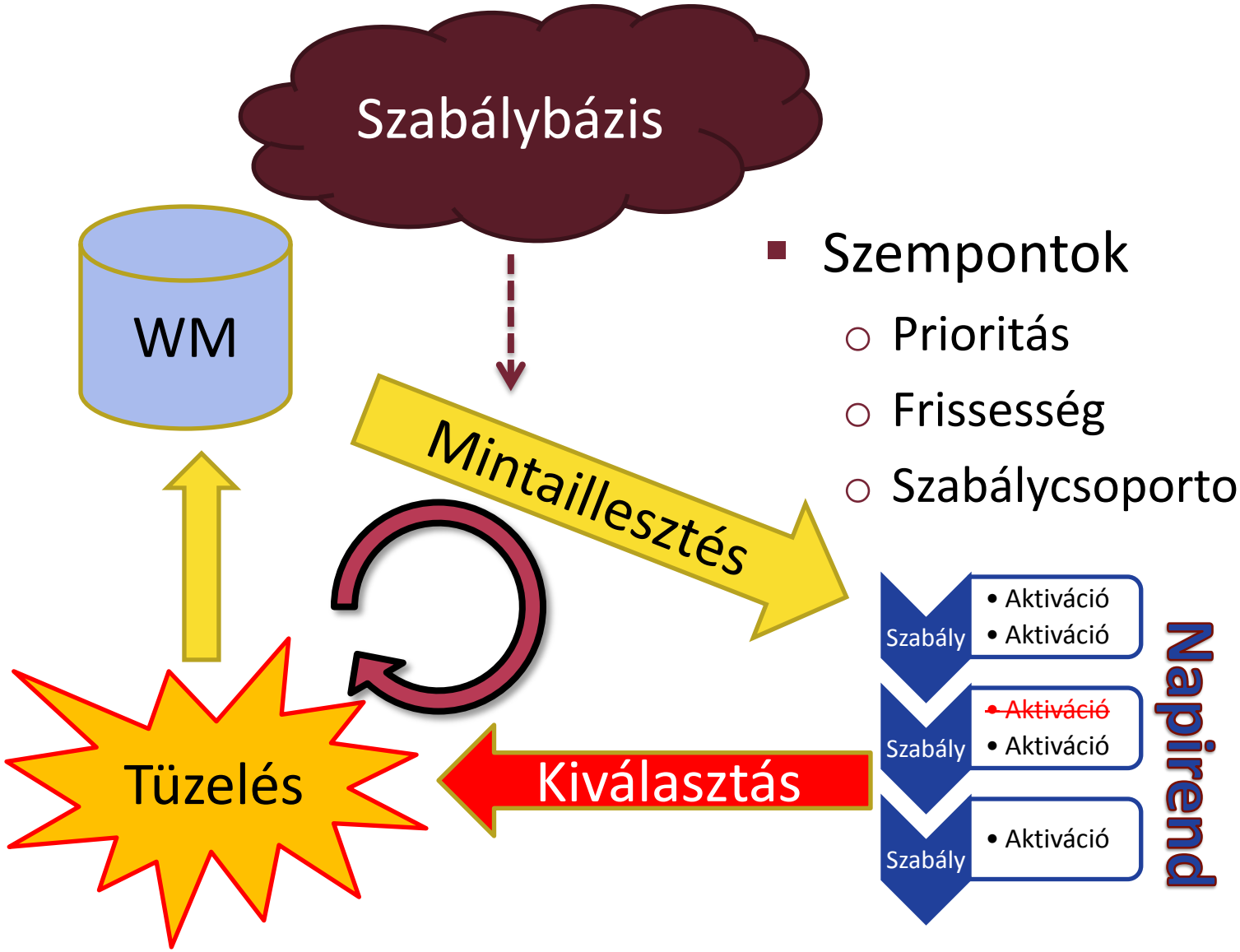


Tipikus produkciós rendszer



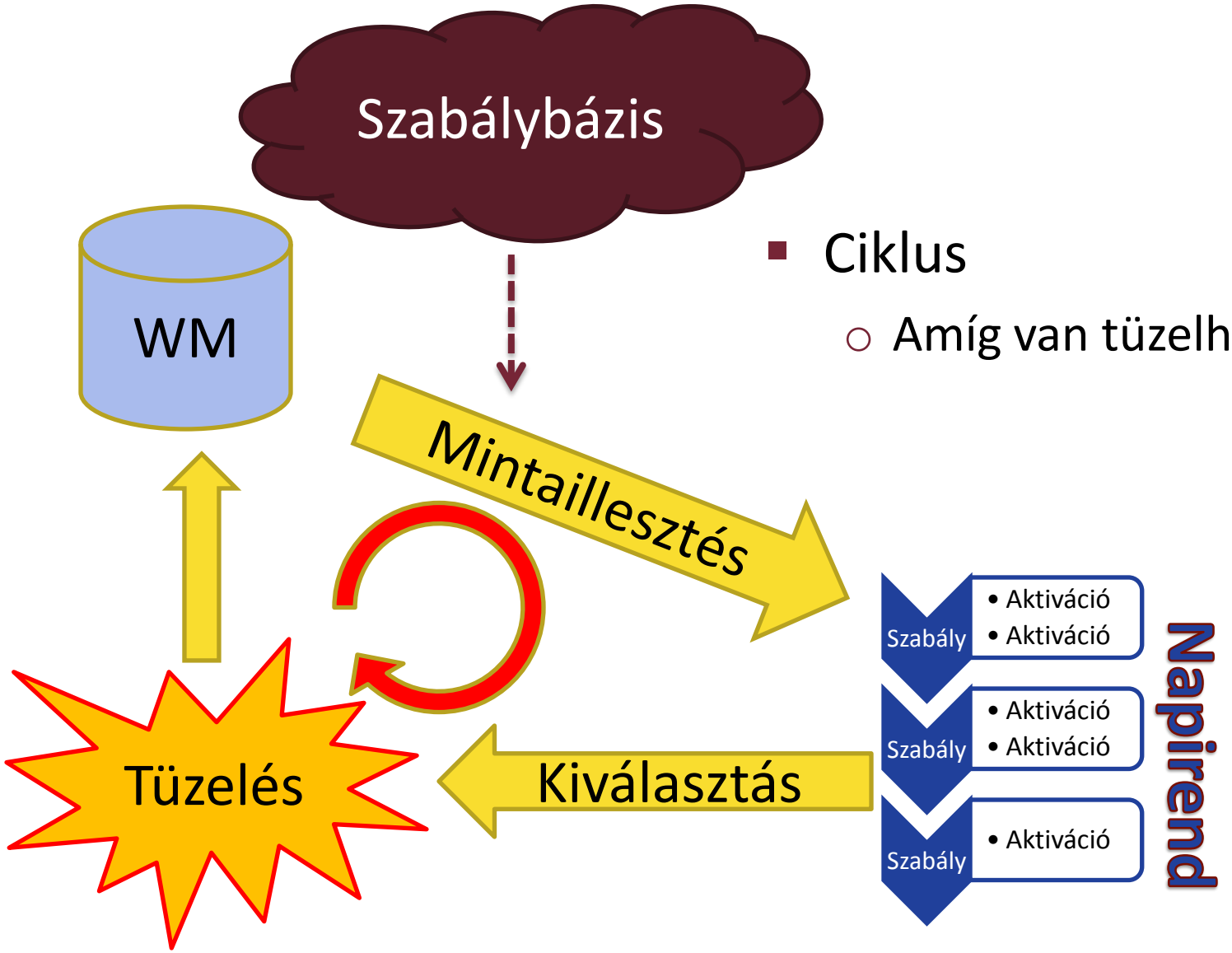
- Naiv
  - Sok esetben elég
- Inkrementális
  - RETE, TREAT, LEAPS, stb.

Tipikus produkciós rendszer



- Szempontok
  - Prioritás
  - Frissesség
  - Szabálycsoportok

Tipikus produkciós rendszer

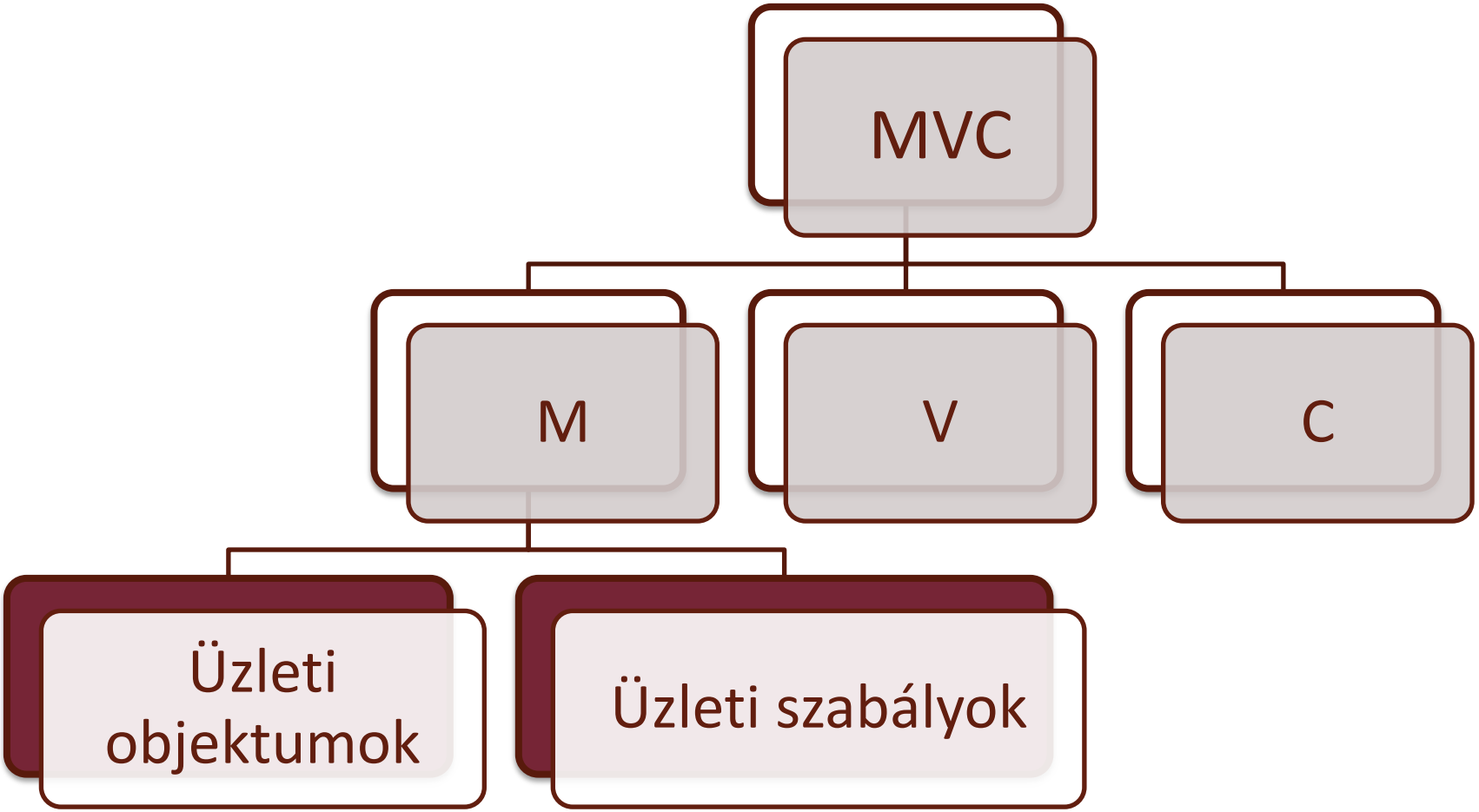


- Ciklus
  - Amíg van tüzelhető szabály

Üzleti szabályrendszerek

Business Rule Systems

- Szabály alapú üzleti logika



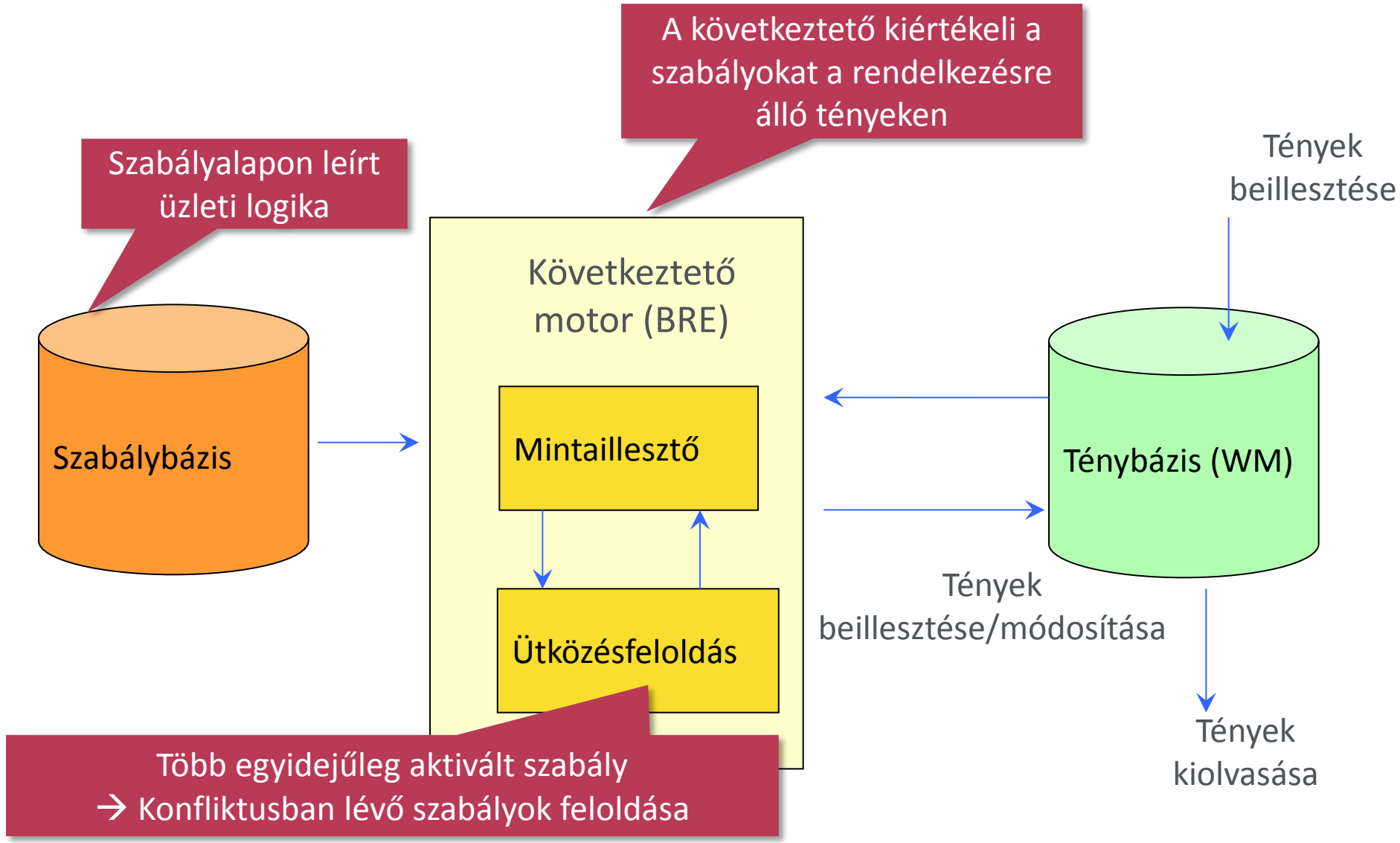
Üzleti szabályok

- **Üzleti logika** „kiszervezésére” végrehajtató modell
- **Üzleti objektumok**at 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

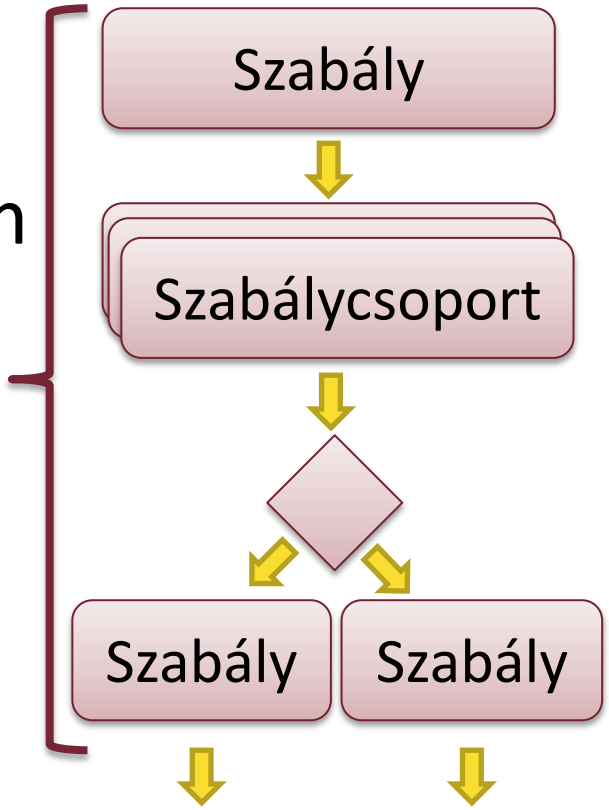
- Üzleti szabályok produkciós rendszer szemszögből
  - „Tények” → üzleti objektumok
  - Kvázi produkciós szabályok, de RHS tetszőleges **akció**
  - Nem (feltétlen) logikai következtetés
    - Érvénytelenné váló feltétel, akció hatása mégis megmarad
    - Egy aktiváció többször is tüzelhet (pl. addig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

Tipikus üzleti szabálymotor működése

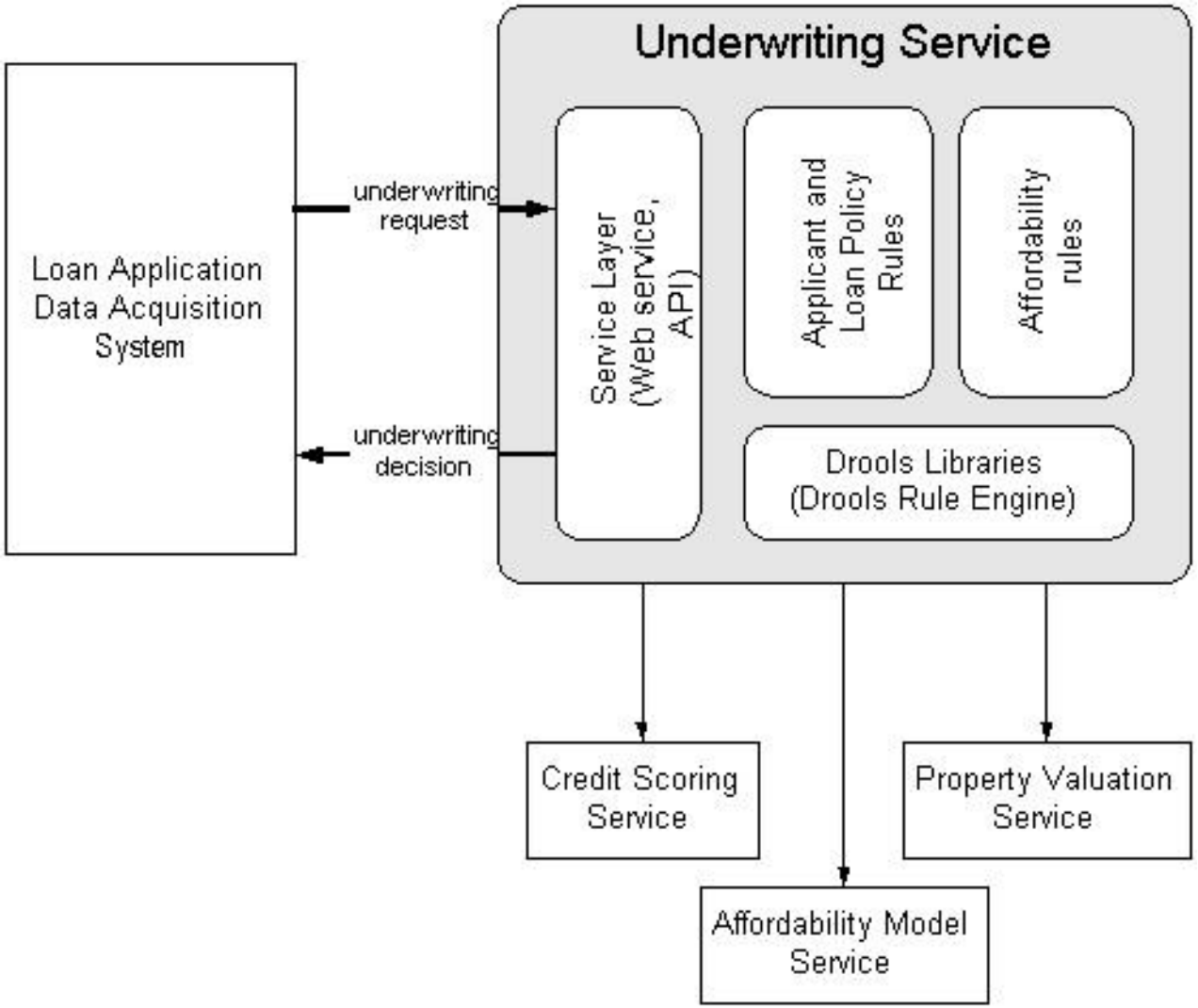


BRE vezérlése

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



Példa alkalmazás



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

Self-service portálok

- Szabálybázis: jogrendszer / üzletkötési feltételek
- Felhasználó kérdez
  - Jogosult vagyok-e EVÁra?
  - Mekkora K+F adókedvezményt kapok a kutatóhelyre?
  - Mekkora lesz a kötelező biztosításom?
- Bekérendő adatok meghatározása
  - Naiv: minden ismert ténytípust, attribútumot bekér
  - A kérdésből hátrafele láncolással leszűkíthető (pl. OPA)
- Adatok bekérése
- Következtetés levonása, eredményjelzés

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, Oracle OPA*, stb.

BRMS – szolgáltatások

- **Szabálytár**
  - Kereshető, automatizáltan módosítható
  - Verziózás
- **Végrehajtó** könyvtár (BRE) → végrehajtó szerver
- **Tool support**
  - IDE, webes felület
  - Template lehetőség, döntési tábla
  - Magasabb granularitású szabályok
  - Tesztelési támogatás: hatásanalízis / unit test
  - 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

Felhasználási területek - példák

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

JBoss Drools

Drools

- JBoss Drools nyílt forrású termékcsalád
  - Drools Expert – szabályvégrehajtó motor (BRE)
  - Drools Guvnor – szabálytár (BRMS)
  - ~~Drools Flow~~ jBPM 5 – üzleti folyamat végrehajtó
  - Drools Planner – megoldástér-bejárás
  - Drools Fusion – komplexesemény-feldolgozó

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

Drools Expert alapok

- Szabályvégrehajtó motor (Java, beágyazható)
  - WM: POJO üzleti objektumok
- 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)

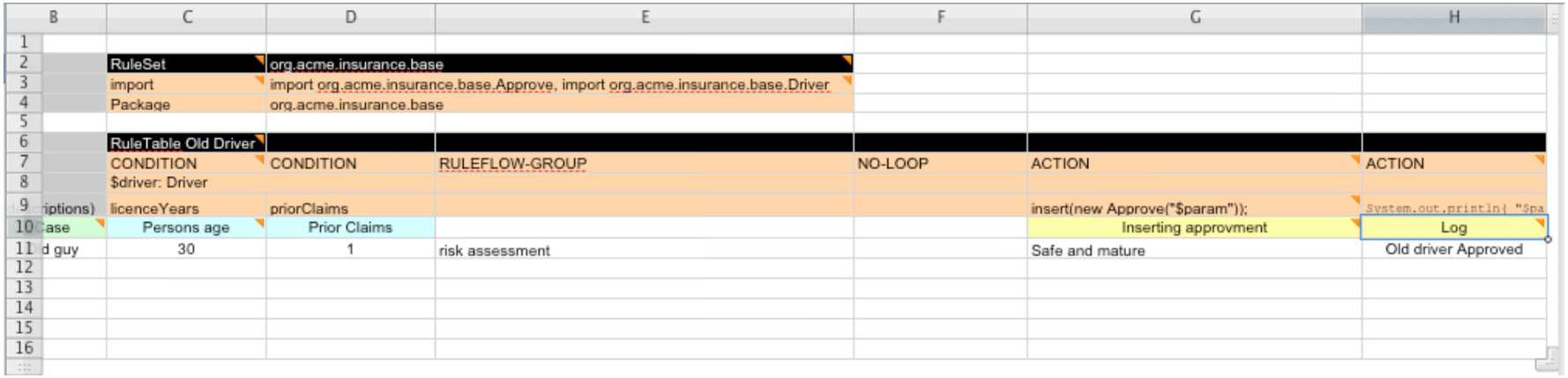
Egyszerű Drools szabályok

```
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
```

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



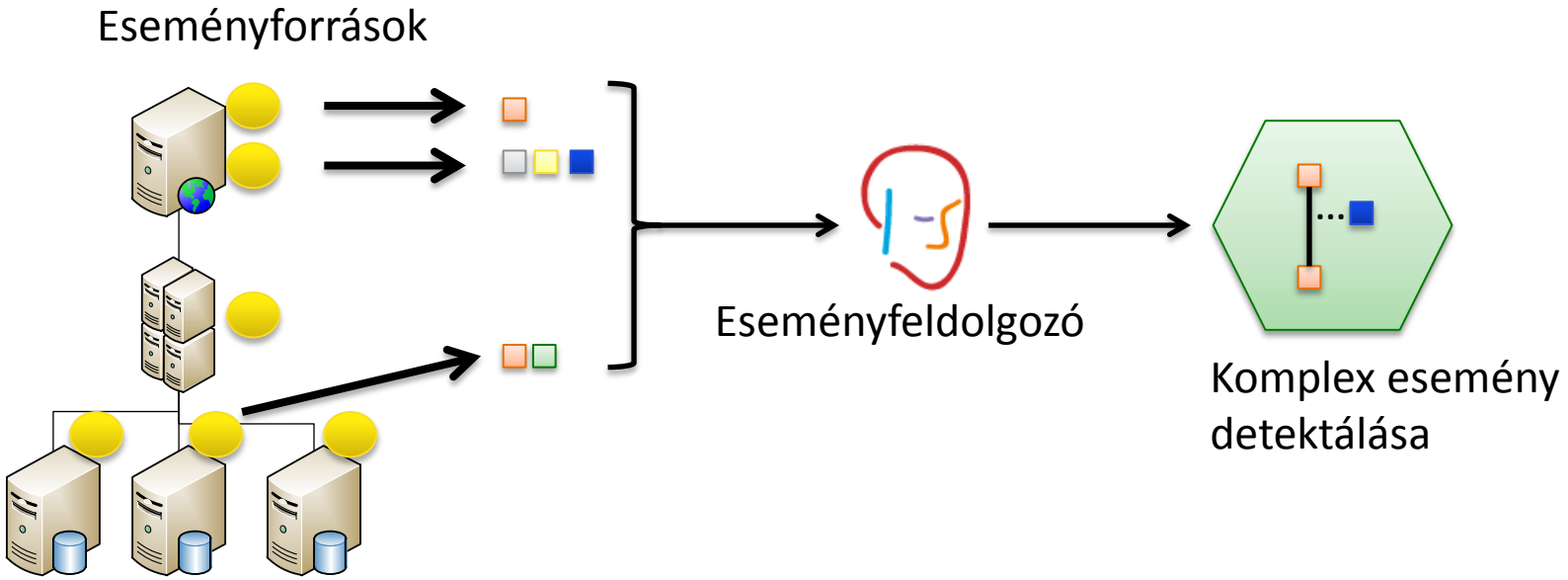
	B	C	D	E	F	G	H
1							
2		RuleSet	org.acme.insurance.base				
3		import	import org.acme.insurance.base.Approve, import org.acme.insurance.base.Driver				
4		Package	org.acme.insurance.base				
5							
6		RuleTable Old Driver					
7		CONDITION	CONDITION	RULEFLOW-GROUP	NO-LOOP	ACTION	ACTION
8		$driver: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

További Drools modulok

- jBPM 5 workflow: Business Rule Task
- Drools Server
  - Drools Expert példány Java webalkalmazásban
  - Spring / WS / … integráció
- Guvnor: a Drools BRMS rendszere
  - Szerveroldali modul: Java webalkalmazás
  - Szabályok, döntési táblák, stb. rendszerezett tárolása
  - Verziókezelés, hozzáférésvédelem
  - Kezelés webes/Eclipse kliensből
  - Szabálybetöltés, futtatás Expert által (fájl helyett)

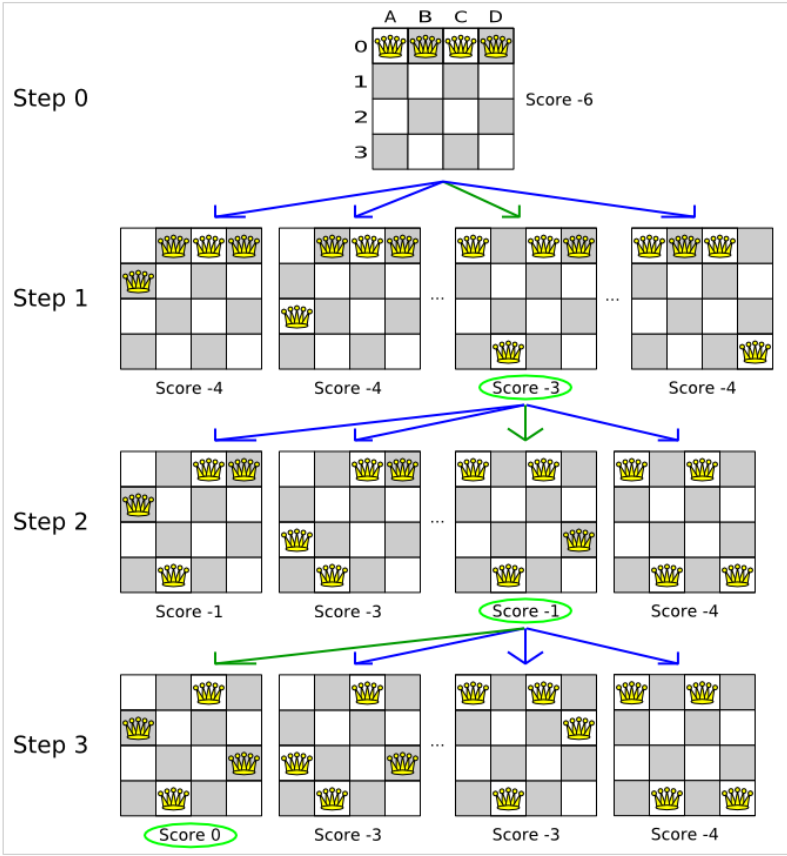
További Drools modulok

- Drools Fusion: **CEP** (Complex Event Processing)
  - Forrásokból heterogén események érkeznek
  - Kiegészített szabálynyelv: időablak, eseményviszonyok
    - Pl. „ha az elmúlt 3 percben túl sok a sikertelen tranzakció…”
  - Események automatikus megőrzése, amíg releváns



További Drools modulok

- Drools Planner: optimalizáció
  - Alapállapot + változók → keresési tér
  - Drools szabállyal
    - Kényszerek
    - Célok
    - Akár a lehetséges lépések is
  - Kritériumok szerinti optimális megoldás megkeresése



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”

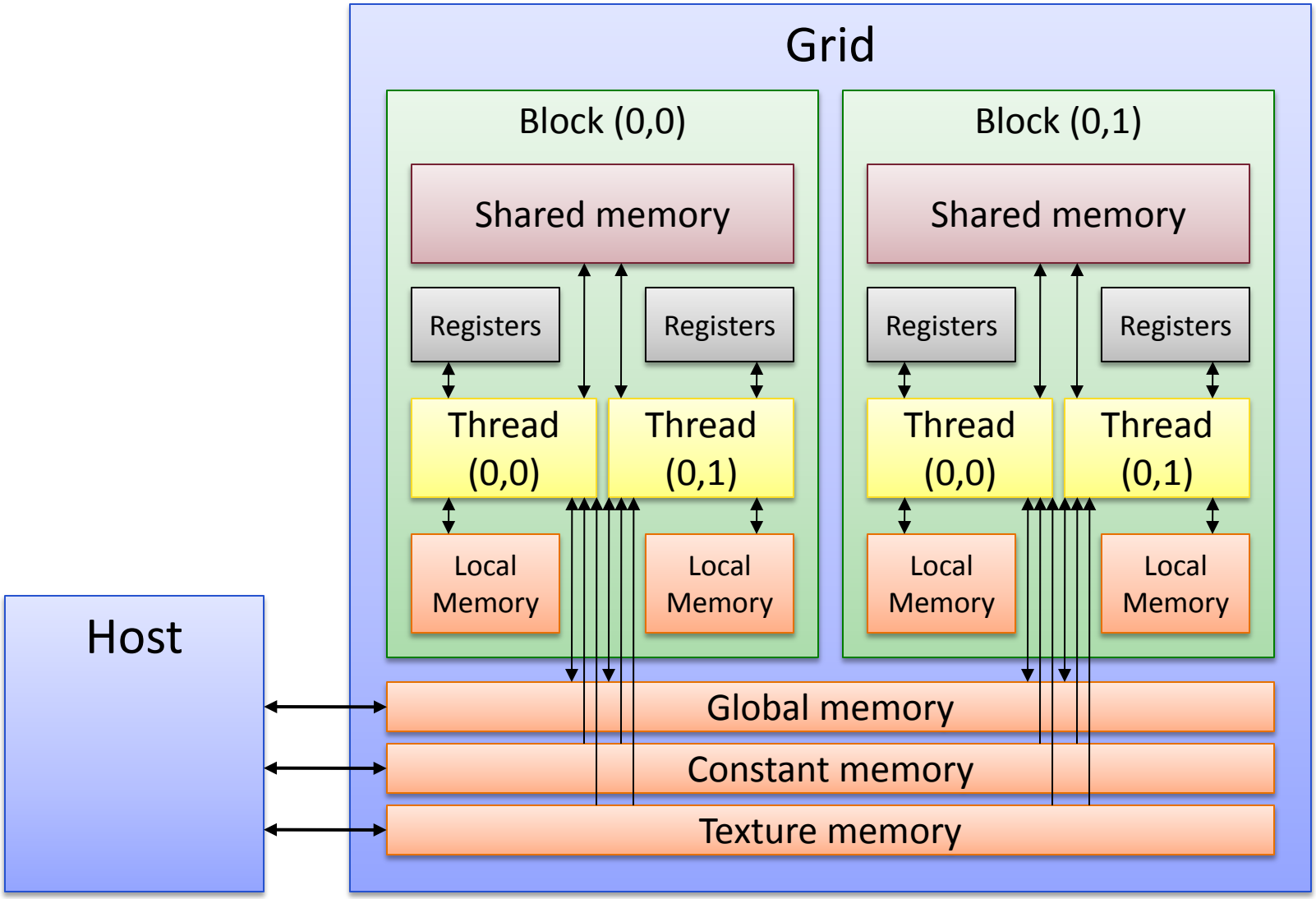
A feladat

- 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 feldogozást
  - 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

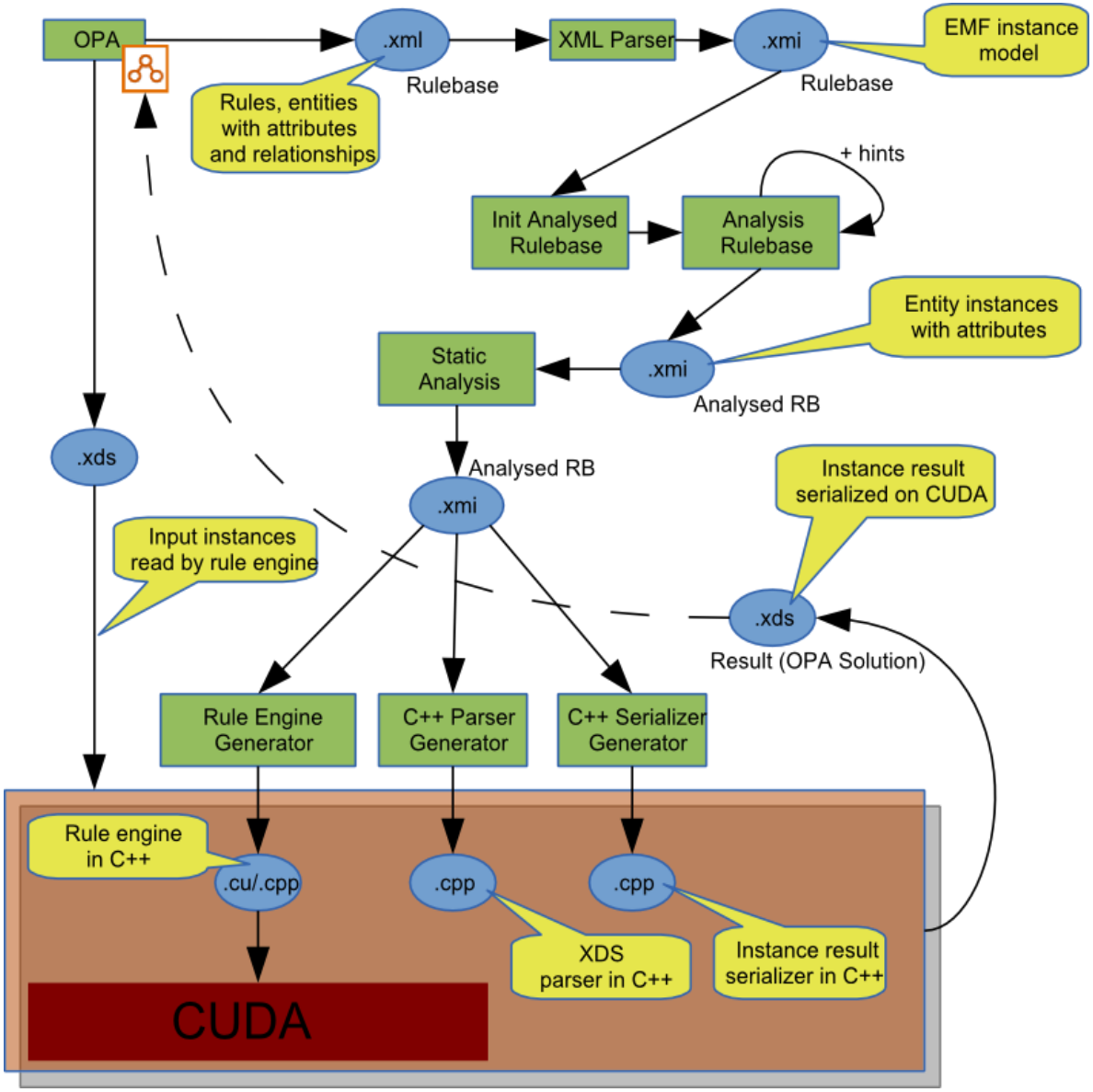
Architektúra - memória



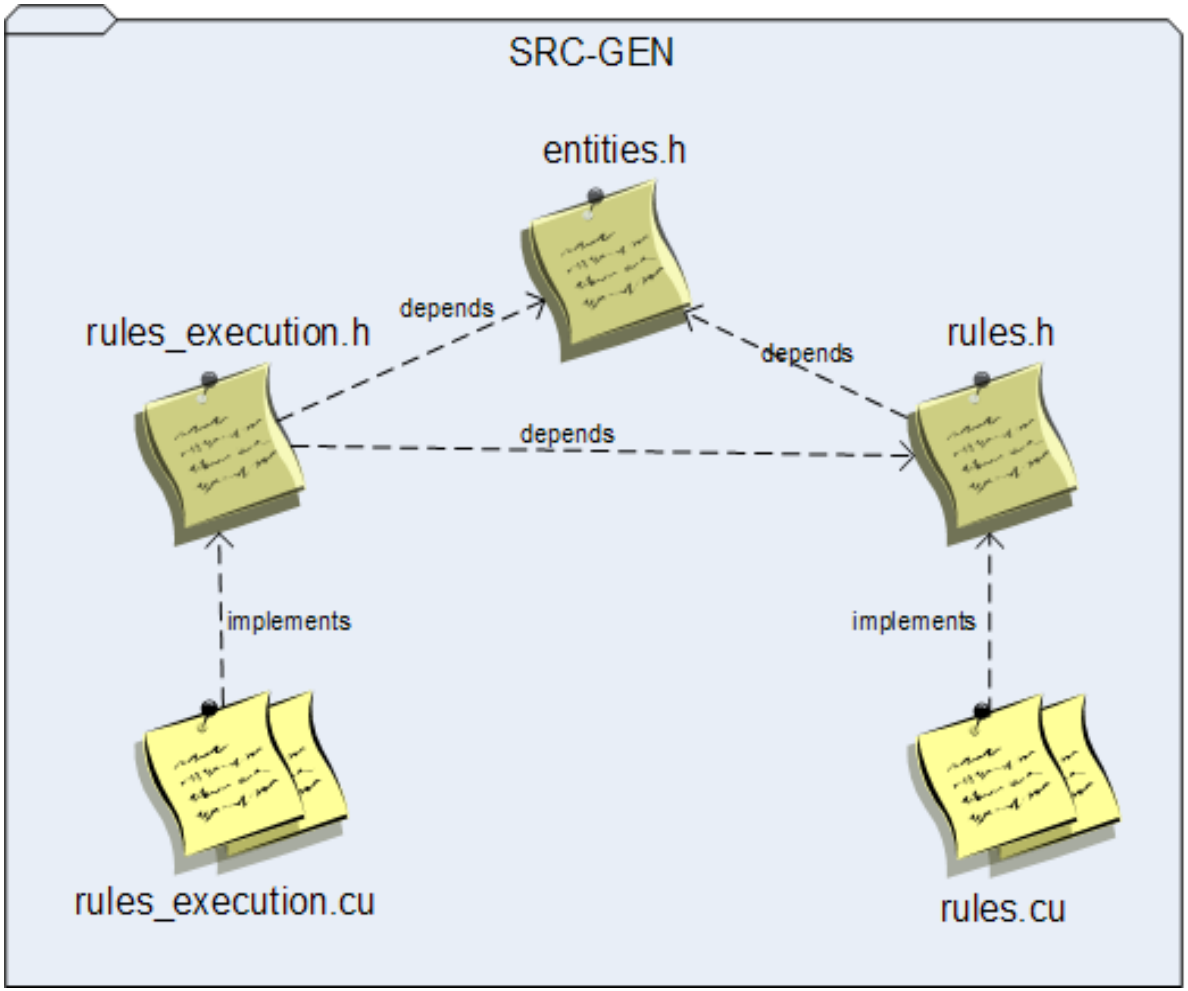
Szabálymotor funkcionalitás

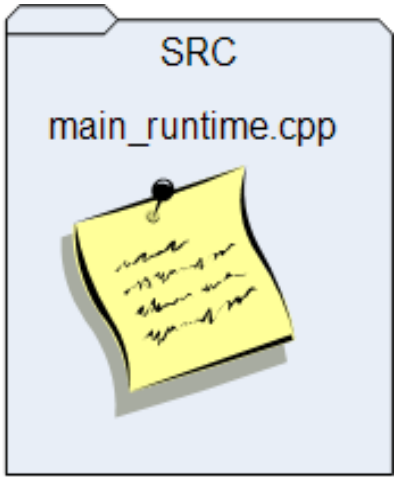
- Java alapú kódgenerálás
- Szabálybázis és entitásmodell független
- Szabályok (attribútumok) alapján szétosztás CPU/GPU közt
  - Memóriaműveletek számának csökkentése
- Szabályok szakértői „jelölése”
- Átlátszó működés (OPA input/output)

A prototípus szabálymotor 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