Les BRMS

Business Rules Management System

GENITECH Group
GENIGRAPH OPTECH e-Citiz

Présentations

- **Emmanuel Bonnet**
  - **ebonnet (at) genigraph.fr**
  - Responsable Dpt Conseil
  - Consultant, Expert BRMS
  - Formateur IBM/Ilog JRules / JBoss Rules
- **Génigraph**
  - SSII 100% orienté Objet
  - SSII –RM (Règles Métier) !
  - www.genigraph.fr

Les BRMS

EN 4 SLIDES CHRONO ...

SLIDE : 4

Les règles métiers

Les moteurs de règles

Le BRMS

SLIDE : 3

Une règle métier

Si

le conducteur n'a pas eu d'accident depuis 3 ans

Et

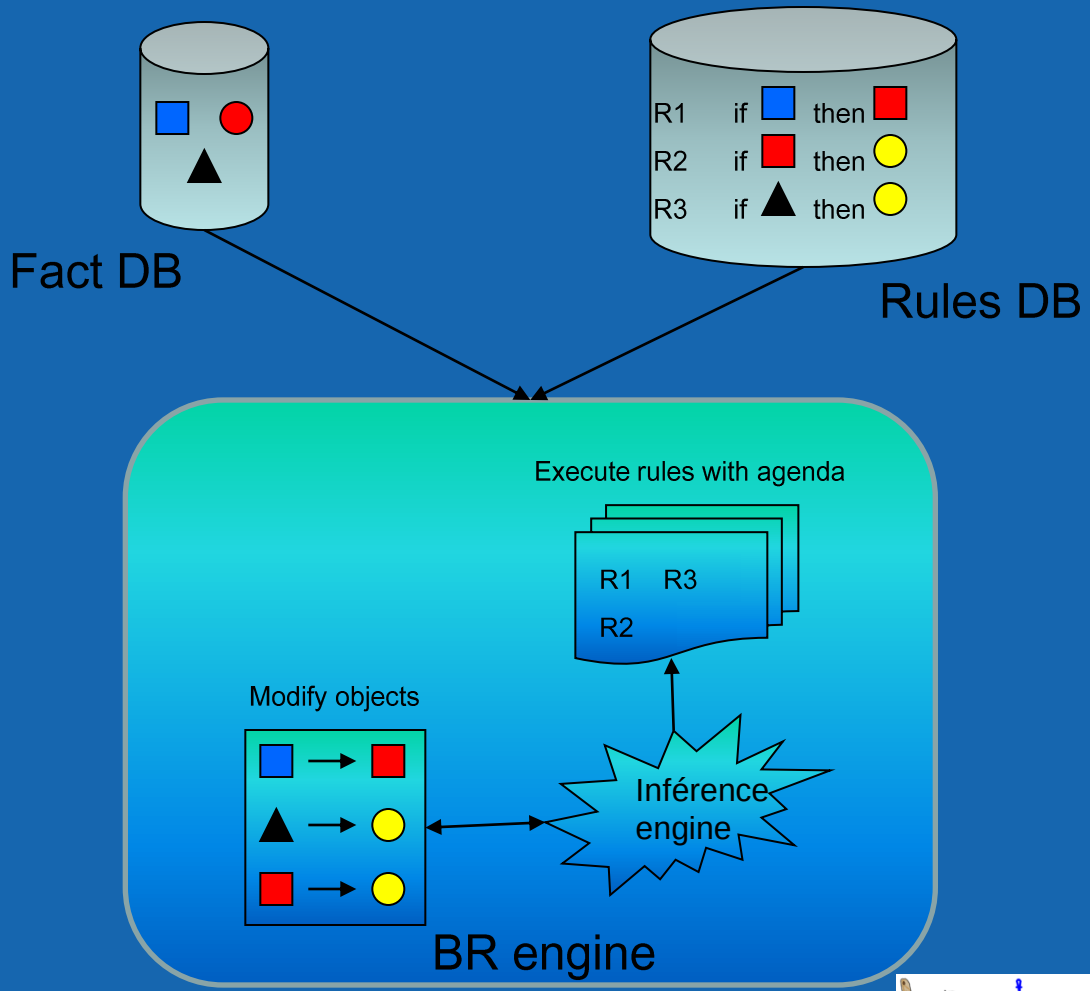
le bonus actuel du conducteur est 0.5

Alors

appliquer au conducteur une remise de 15%

SLIDE : 2

Un moteur d'inférence



SLIDE : 1

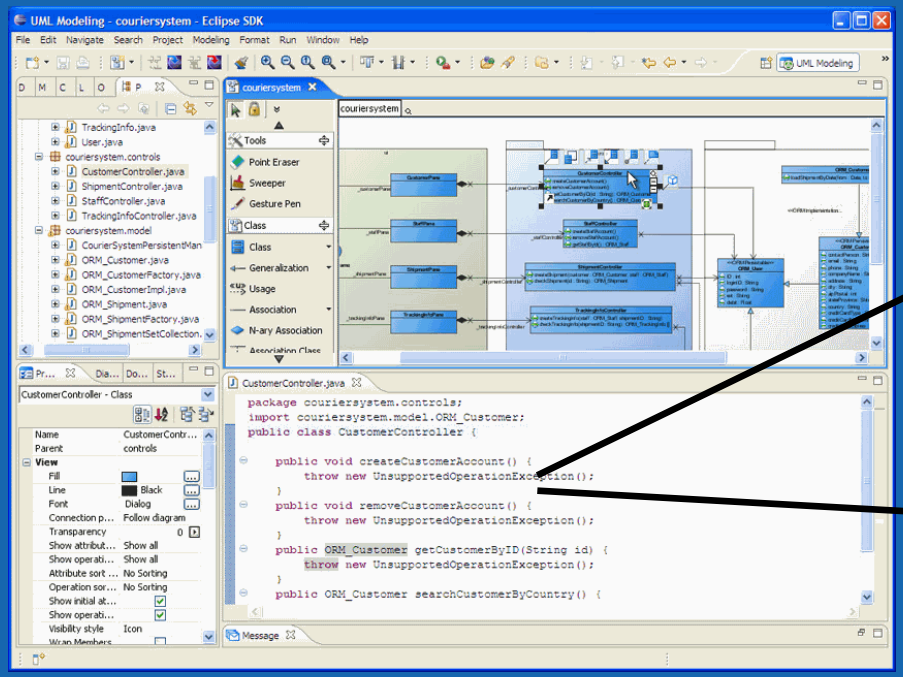
Un BRMS

- Gestion du cycle de vie
  - Règles métier
- Ecriture -> Exécution
  - Versioning
  - Process : Validation, Test

BRMS : what's in it for me ?

- Externaliser
- Expliciter
- Gérer
  - **Des décisions métier**
    - Nichées dans un applicatif

Externaliser ?



Logique technique

Décisions métiers

BRMS : Externaliser

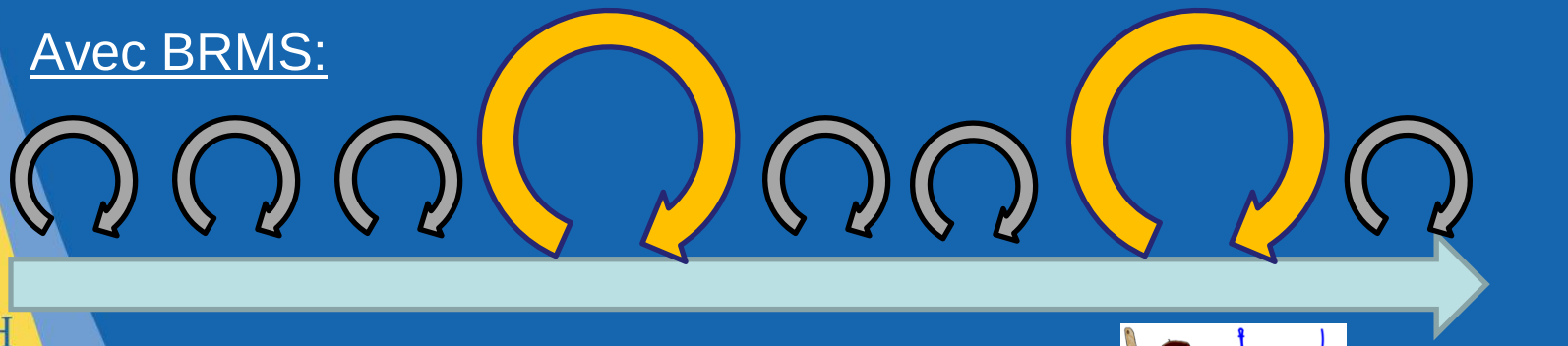
- Logique métier extérieure à l'application
  - Modifiable
  - indépendant du code applicatif
  - Cycles courts
- Existe-t-il un besoin ?

Cycle de vie des règles

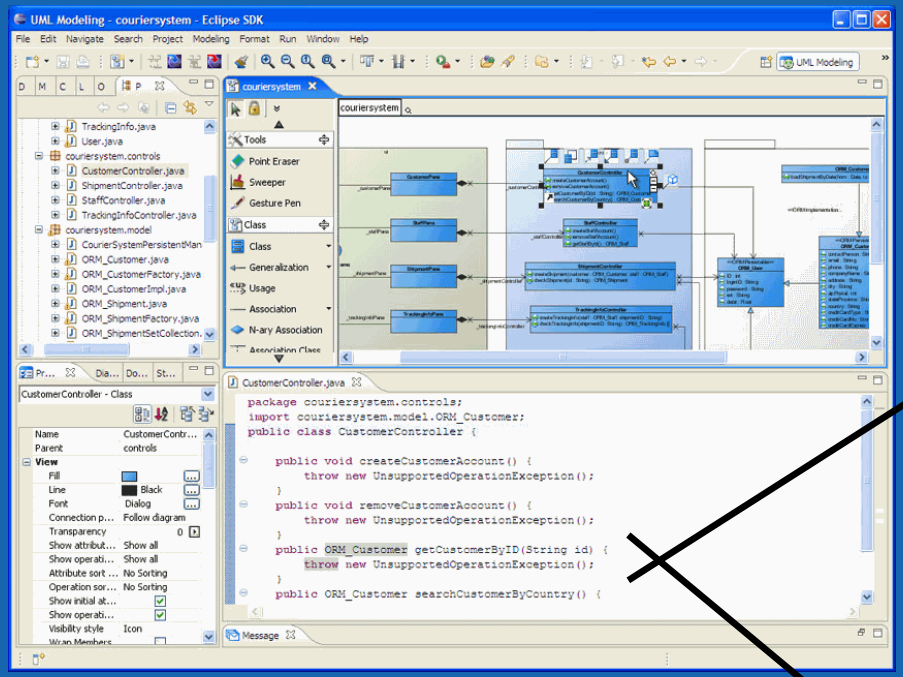
- Le cycle de vie des règles change plus souvent que celui de l'application

Sans BRMS:

Avec BRMS:



Expliciter ?




```
public RentalAgreement(Customer customer,
        Branch pickupBranch, Date pickupDate,
        Branch returnBranch, Date returnDate,
        CarGroup requestedCarGroup, String[] coverages) {
    this.customer = customer;
    this.pickupBranch = pickupBranch;
    this.pickupDate = pickupDate;
    this.returnBranch = returnBranch;
    this.returnDate = returnDate;
    this.requestedCarGroup = requestedCarGroup;
    this.coverages = coverages;
    this.offers = new HashMap<String, Offer>();
}

private Customer customer;
public Customer getCustomer() {
    return this.customer;
}
public void setCustomer(Customer cust){
    customer= cust;
}


private Map<String, Offer> offers;
public Collection<Offer> getOffers() {
    return offers.values();
}
public void addOffer(String name) {
    Offer offer = new Offer(name);
    offers.put(name, offer);
}
public Offer getOffer(String name) {
    Offer offer = (Offer)offers.get(name);
    return offer;
}
public Offer getBestOffer() {
    Collection<Offer> offers = getOffers();
    if (offers.isEmpty()) return null;

    Offer[] offersArray = (Offer[]) offers.toArray(new Offer[offers.size()]);
    Arrays.sort(offersArray);
    return offersArray[offersArray.length-1];
}
```

Si

le conducteur n'a pas eu d'accident depuis 3 ans

Et

le bonus actuel du conducteur est 0.5

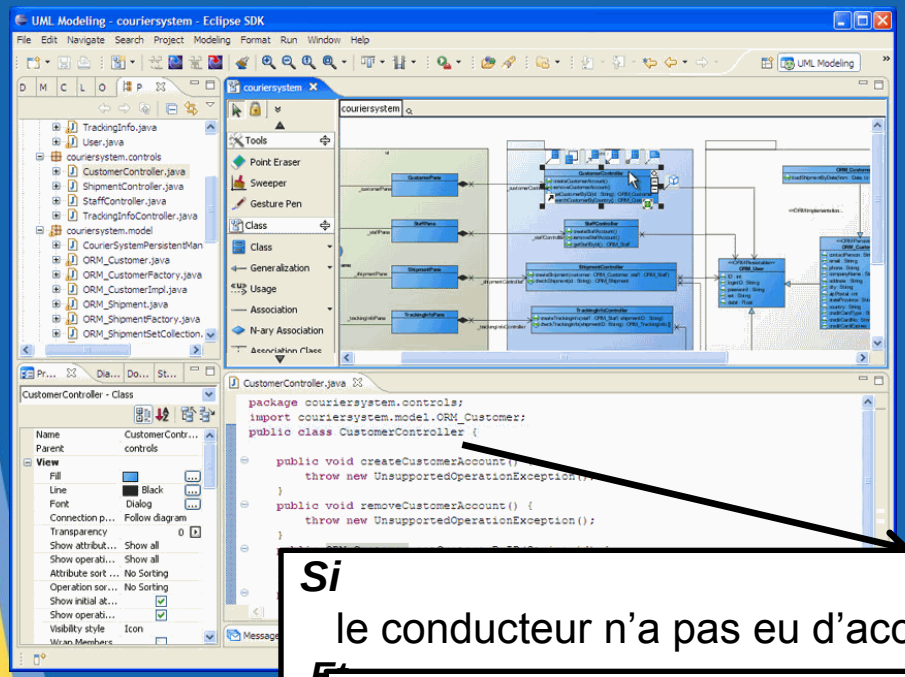
Alors

appliquer au conducteur une remise de 15%

BRMS : Expliciter

- Compréhensible
  - Métier lisible / visible
- Modifiable
  - Pas besoin des informaticiens
- Traçable
  - On peut « relire » la séquence d'une décision

Gérer ?



Si
le conducteur n'a pas eu d'accident depuis 3 ans

Si
le conducteur n'a pas eu d'accident depuis **6 ans**

Si
le conducteur n'a pas eu d'accident depuis **10 ans**
Et
le bonus actuel du conducteur est **0.8**
Alors
appliquer au conducteur une remise de 15%

BRMS :Gérer

- Décisions Métier
  - Stockée / Référencées
- Organisation des connaissances
  - Searchable !!
  - Exécutable !!!

Projets : the Usual Suspects

- Le métier change souvent
  - « encore un nouveau requirement ! »
- La connaissance est très pointue
  - «SVP, pas d'informaticien !»
- Des décisions doivent être tracées
  - « Pourquoi cette décision ? »
- Initialiser des données …..

Tout ce que vous voulez savoir sur

Les règles métiers

Les moteurs de règles

Les BRMS

sans jamais avoir osé le demander …

GENITECH Group
GENIGRAPH OPTECH e-Citiz

Paris JUG

Tout ce que vous voulez savoir sur …

Les règles métiers

sans oser le demander

Une règle, c'est quoi?

If

…

and/or

…

Then

…

(***Else***

…

)

Le métier, c'est quoi?

La rédaction de spécifications ?

La présence à des réunions ?

La gestion des prestataires ?

« Le savoir du client »

Les règles métier, c'est quoi

Du code ?
Une implémentation technique ?
Des Design patterns ?
Une application informatique ?

« Le savoir du client »

Sous la forme

IF
THEN

Exemple

Si

le conducteur n'a pas eu d'accident depuis 3 ans

Et

le bonus actuel du conducteur est 0.5

Alors

appliquer au conducteur une remise de 15%

Exemple

Langage usuel

Grammaire

Si

le conducteur n'a pas eu d'accident depuis 3 ans

Et

le bonus actuel du conducteur est 0.5

Alors

appliquer au conducteur une remise de 15%

Exemple

Si

le conducteur n'a pas eu d'accident depuis 3 ans

Et

le bonus actuel du conducteur est 0.5

Alors

appliquer au conducteur une remise de 15%

Attribut

Concept

Test

Concept

Traitement

Mapping

Grammaire / Langage usuel

+

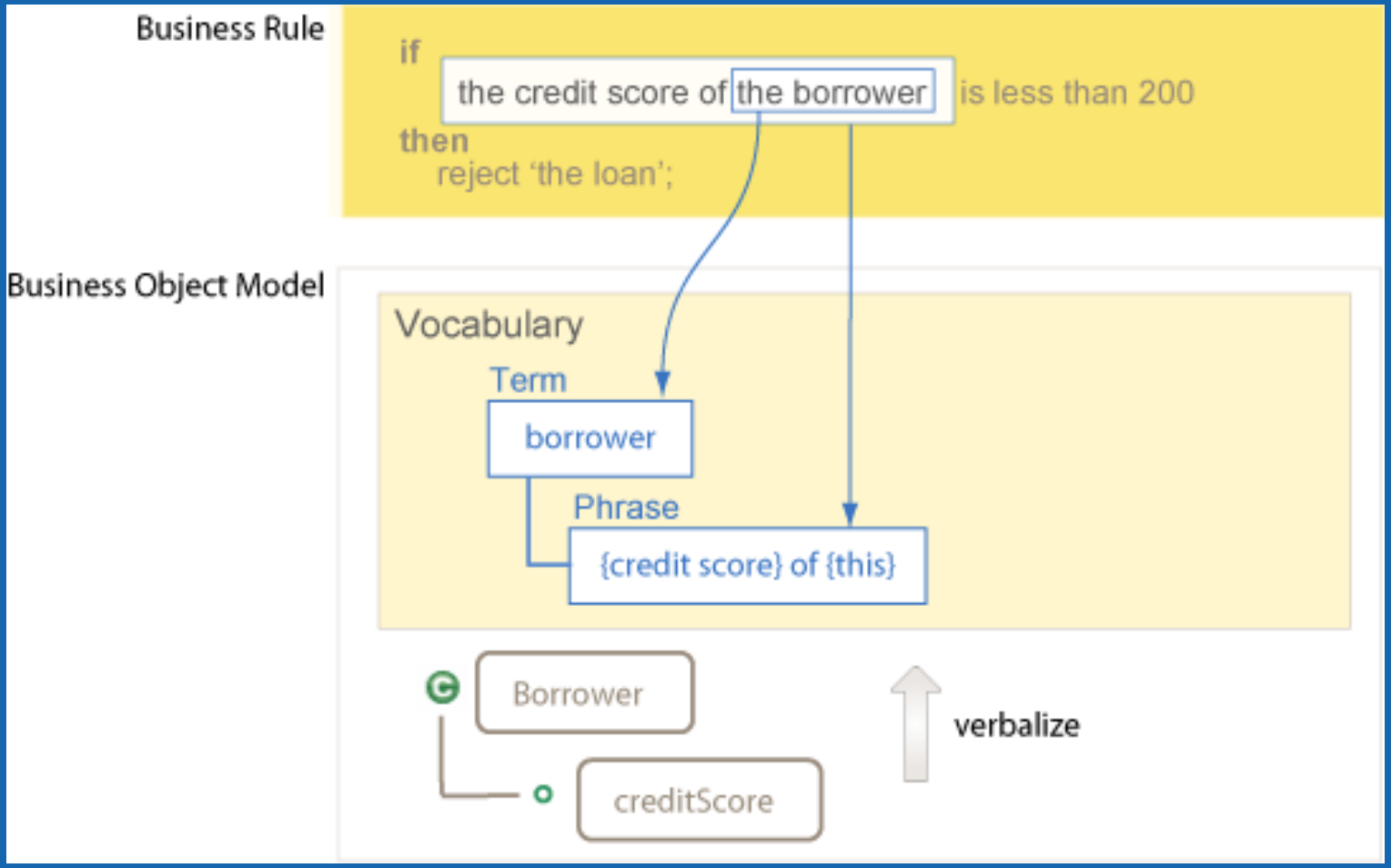
Concept / Attributs / Traitements

=

Mapping entre langages

naturel ← →orienté objet

JRules Mapping



Règle métier : résumé

- Le savoir du client
- En langage naturel
  - Structuré
    - Formalisme : IF / THEN
    - Grammaire
- Traduction informatique
  - Mapping vers langages objets
  - Sorte de « Compilation »

Sous le capot ? (JBoss Rules)

```
rule "grant customer"
when
    the customer with no accident in 3 year
    and
    the customer bonus is 0.5
then
    grant a 15% discount to the customer
```

```
rule "grant customer"
    when
        $customer : Customer(lastAccident.date < UtilDate.findDateInPast(3) && bonus == 0.5);
    then
        $customer.grantDiscount(15);
```

Le Langage naturel

- Parlons en ….
- Les utilisateurs qui lisent des règles !!!!
  - Ils peuvent écrire aussi ?
- N’est ce pas un mythe ?
  - » Java = grammaire + vocabulaire
  - » Sécurité, simplicité, lisibilité

Tout ce que vous voulez savoir sur …

Les moteurs de règles

sans oser le demander

Moteur de règle

- Exécute
  - les **règles** en regards de **faits**
- Optimise

  (Volume)
- Garantit la cohérence

  (Séquence / Contexte)

Moteur d'inférence

L'algo 'Red is dead'

☐ *Il vaut mieux exécuter une fois une règle*

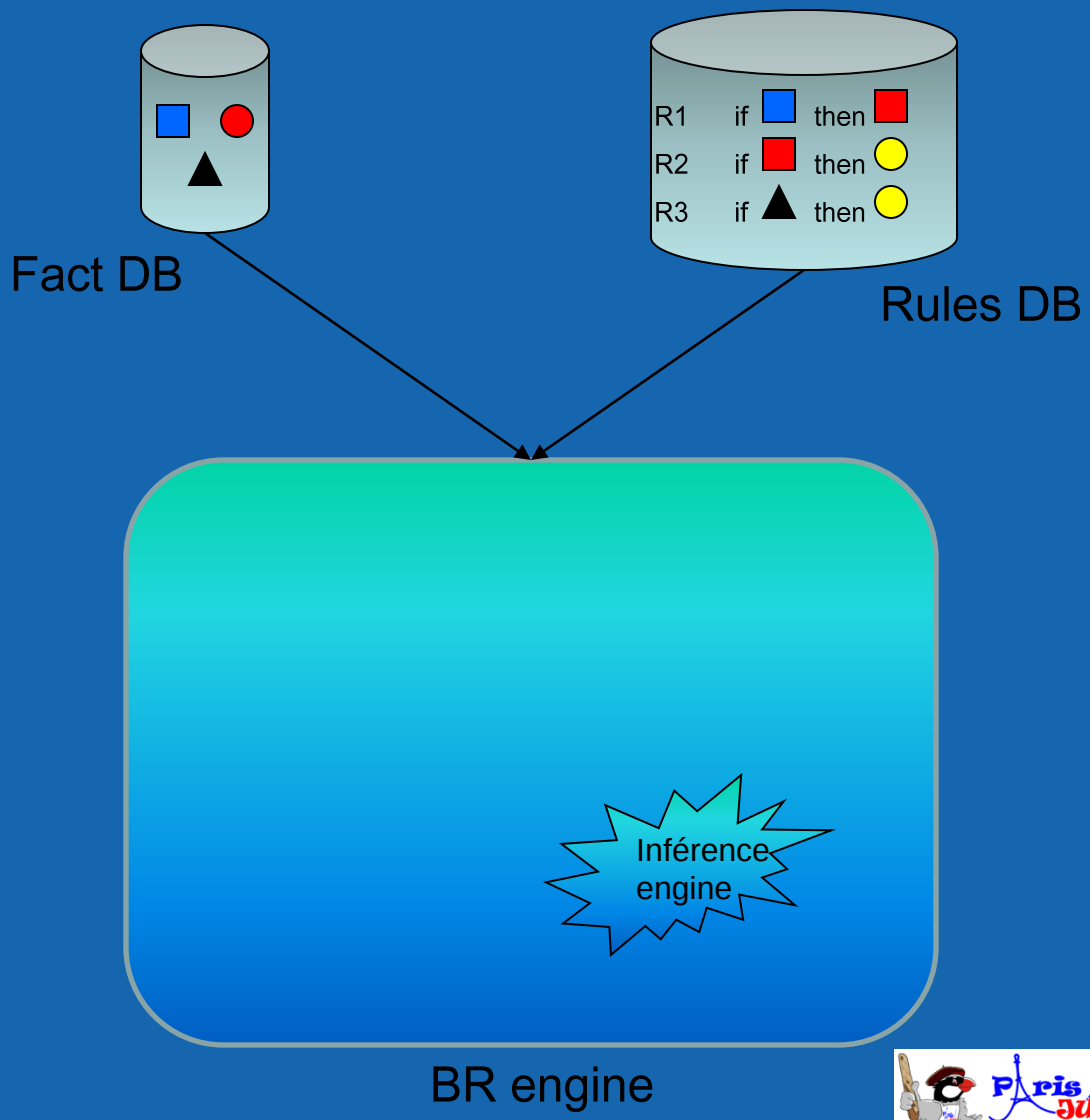
☐ *Il vaut mieux exécuter mille fois une règle*

☒ *Il vaut mieux exécuter une fois mille règles*

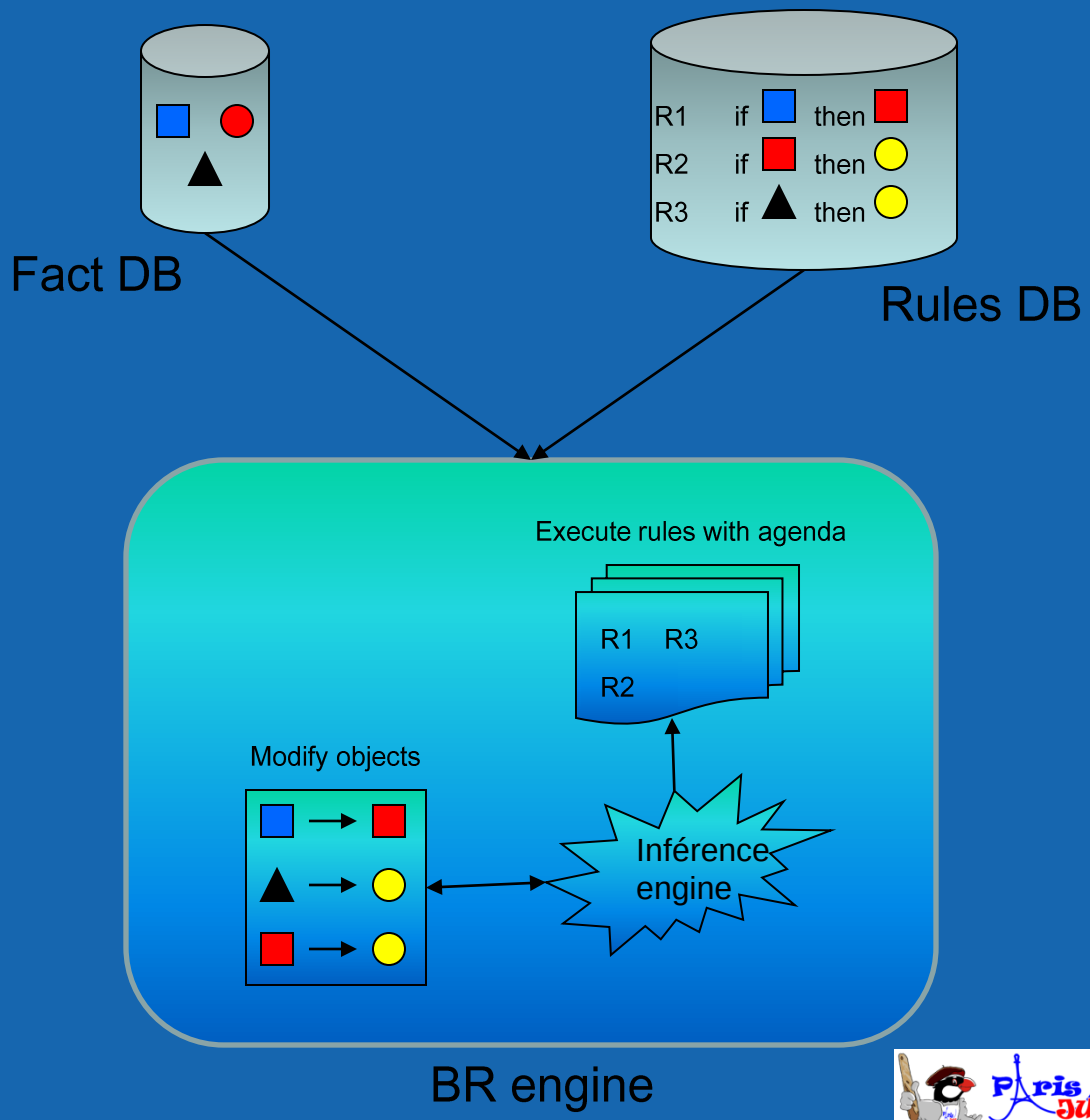
☒ *Il vaut mieux exécuter mille fois mille règles*

« Attention, ca va couper «

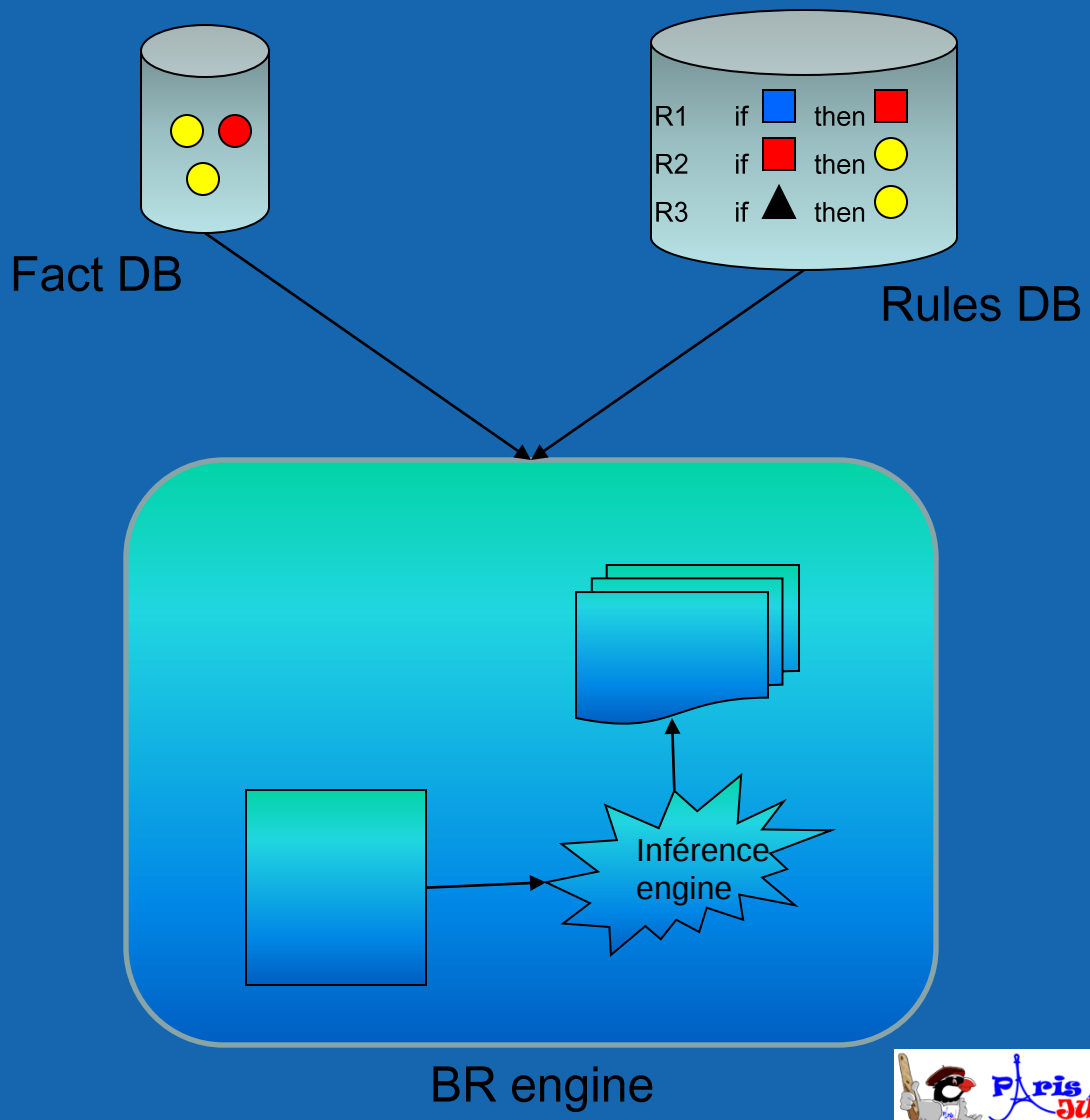
La structure (1/3)



La structure (2/3)



La structure (3/3)

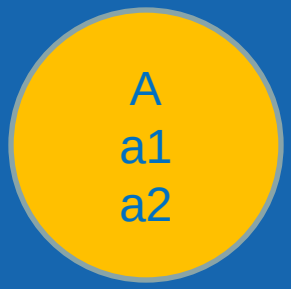


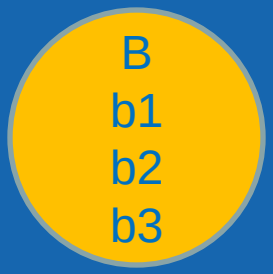
RETE

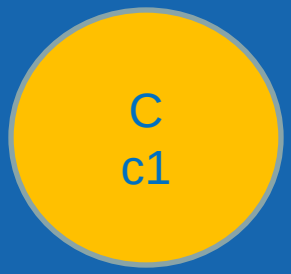
- Algorithme RETE (Charles Forgy – 1980)
- Rapide
- Gère
  - Chaînage avant
  - Un grand nombre de règles
  - Un gros volume de données
  - La validité d'un résultat au cours de l'exécution (règles inter-dépendantes)

Principe du RETE

- Cohérence garantie
  - réévaluation constante
- Principe de « pattern matching »
- Données vérifiées « contre » les conditions des règles
  - Liste d'instances de règles candidates
  - Réévaluation après chaque éxecution d'une règle







IF

B.b1 = 2

C.c1 = B.b2

Then

R2 (B,C).

IF

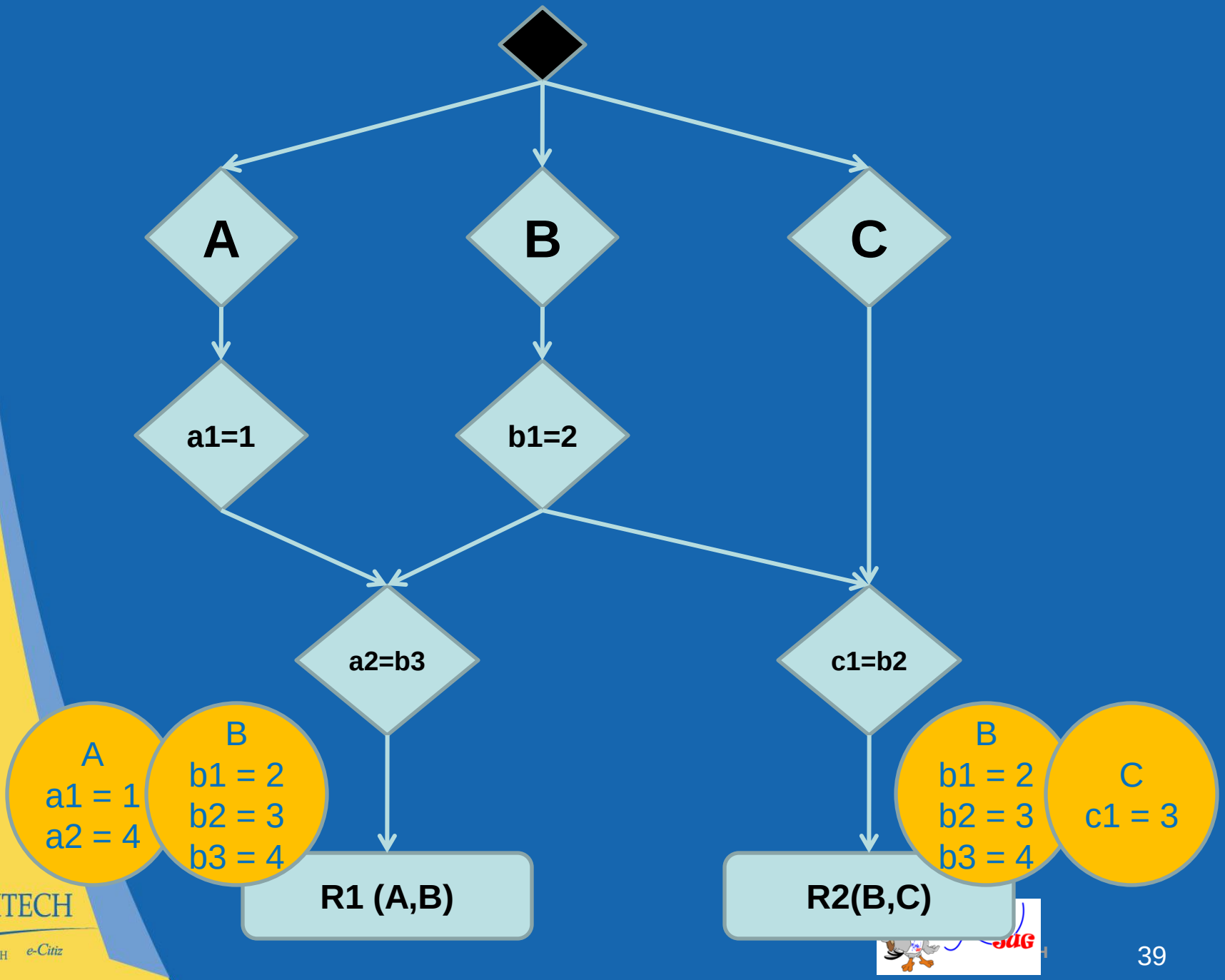
A.a1 = 1

B.b2 = 2

A.a2 = B.b3

Then

R1 (A,B).

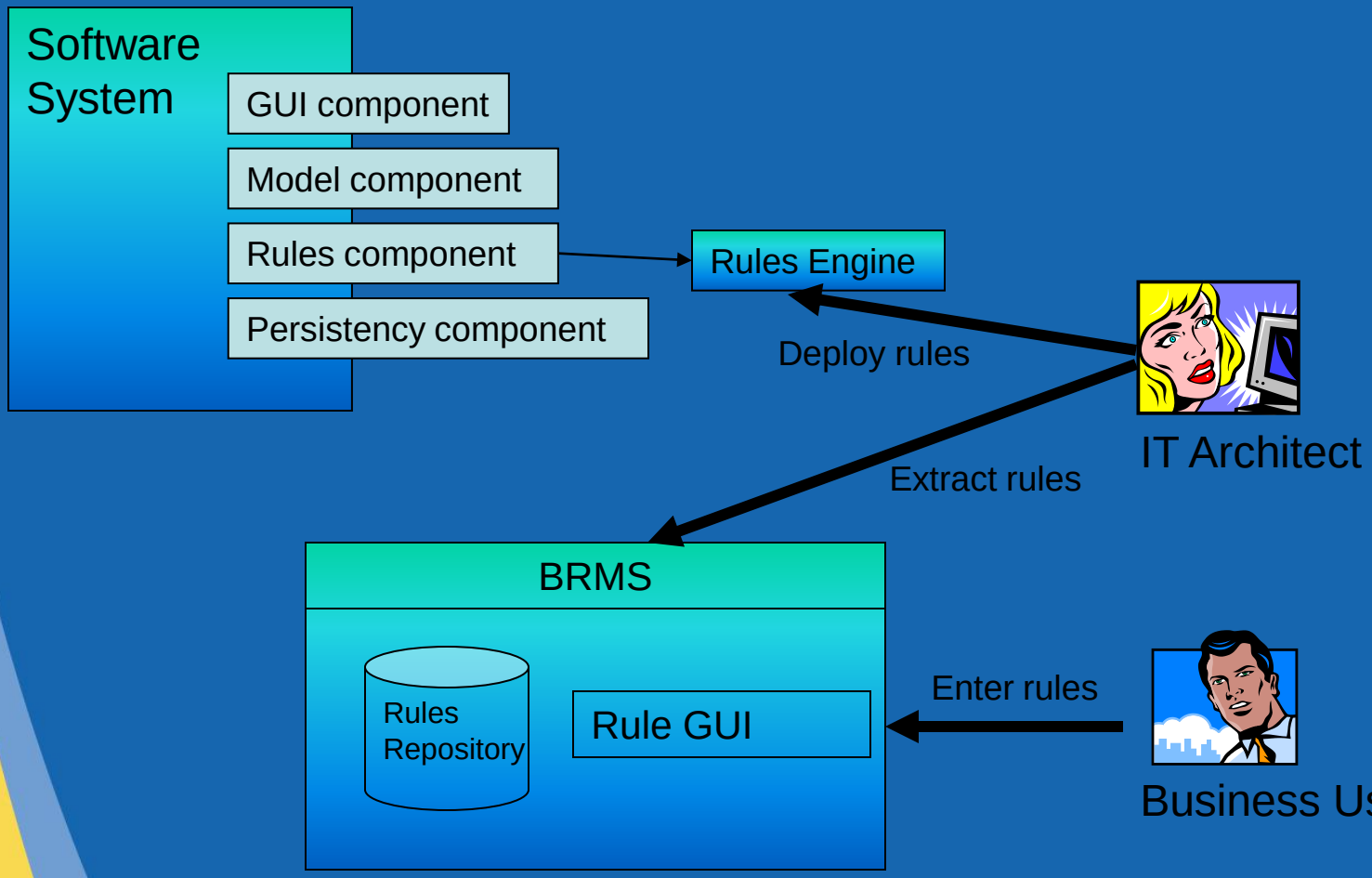

A
B
C
a1=1
b1=2
a2=b3
c1=b2
A
a1 = 1
a2 = 4
B
b1 = 2
b2 = 3
b3 = 4
R1 (A,B)
B
b1 = 2
b2 = 3
b3 = 4
C
c1 = 3
R2(B,C)

Tout ce que vous voulez savoir sur …

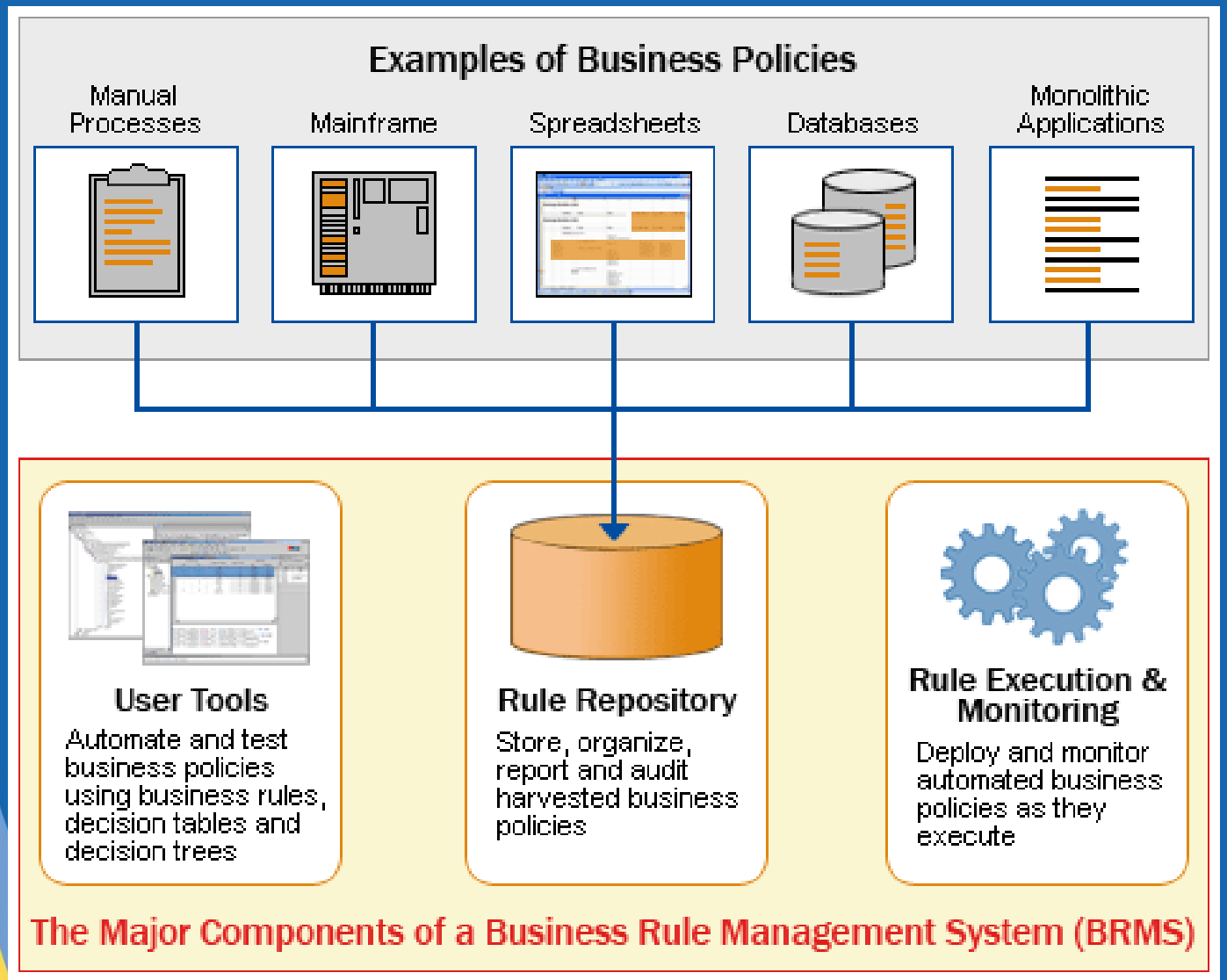
Les BRMS

sans oser le demander

L'architecture

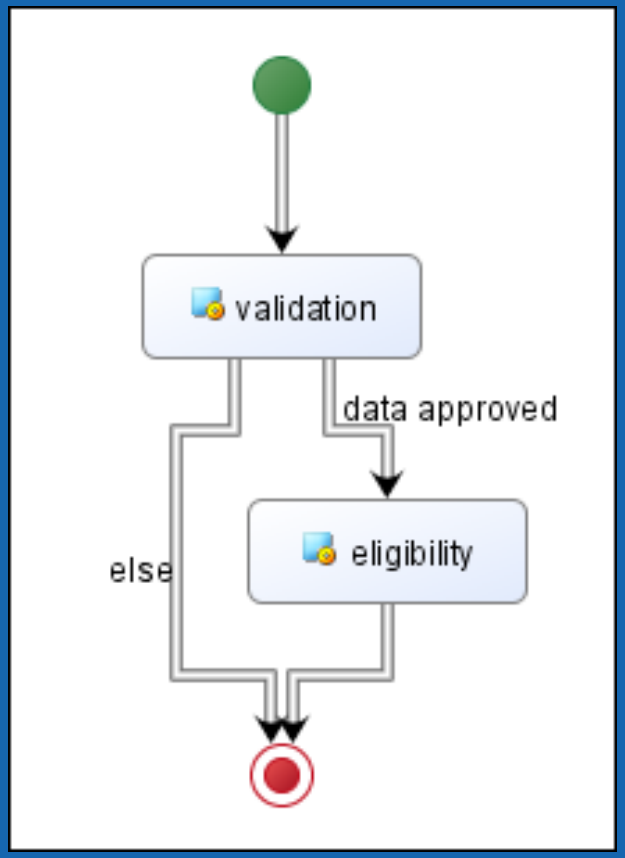


Les composants



Ruleflow

- Un workflow pour règles



Différents types de règles

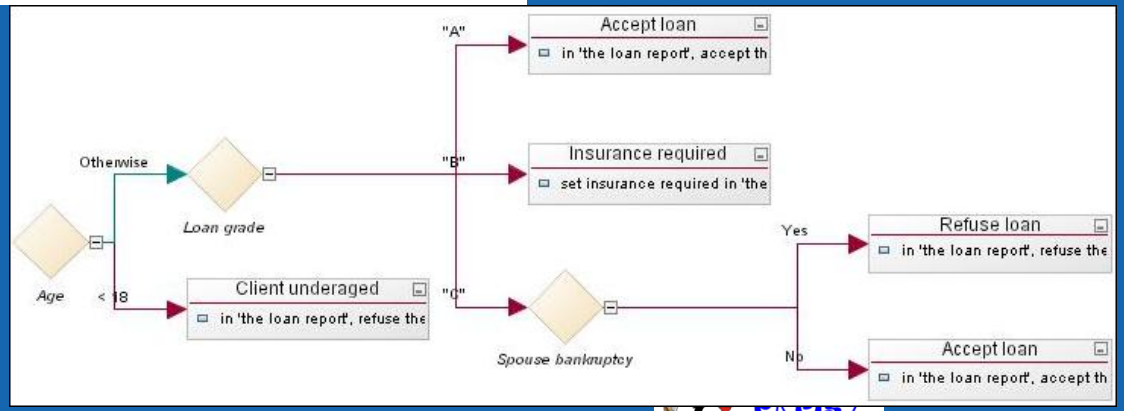
variable test

definitions
```
definitions
  set 'loyal customer' to a customer
    where the category of this customer is Gold;
```
conditions
```
if
  the age of 'loyal customer' is more than 60
```
actions
```
then
  give a 10% discount to 'loyal customer';
else
  give a 5% discount to 'loyal customer';
```

Condition Columns | Action Columns

Column Header / Rows = Rules

	Grade	Amount of loan Min	Amount of loan Max	Insurance required	Insurance rate
0	A	< 100,000		false	
1		100,000	300,000	true	0.001
2		300,000	600,000	true	0.003
3		⩾ 600,000		true	0.005
4	B	< 100,000		false	
5		100,000	300,000	true	0.002
6		300,000	600,000	true	0.005
7		⩾ 600,000		true	0.008



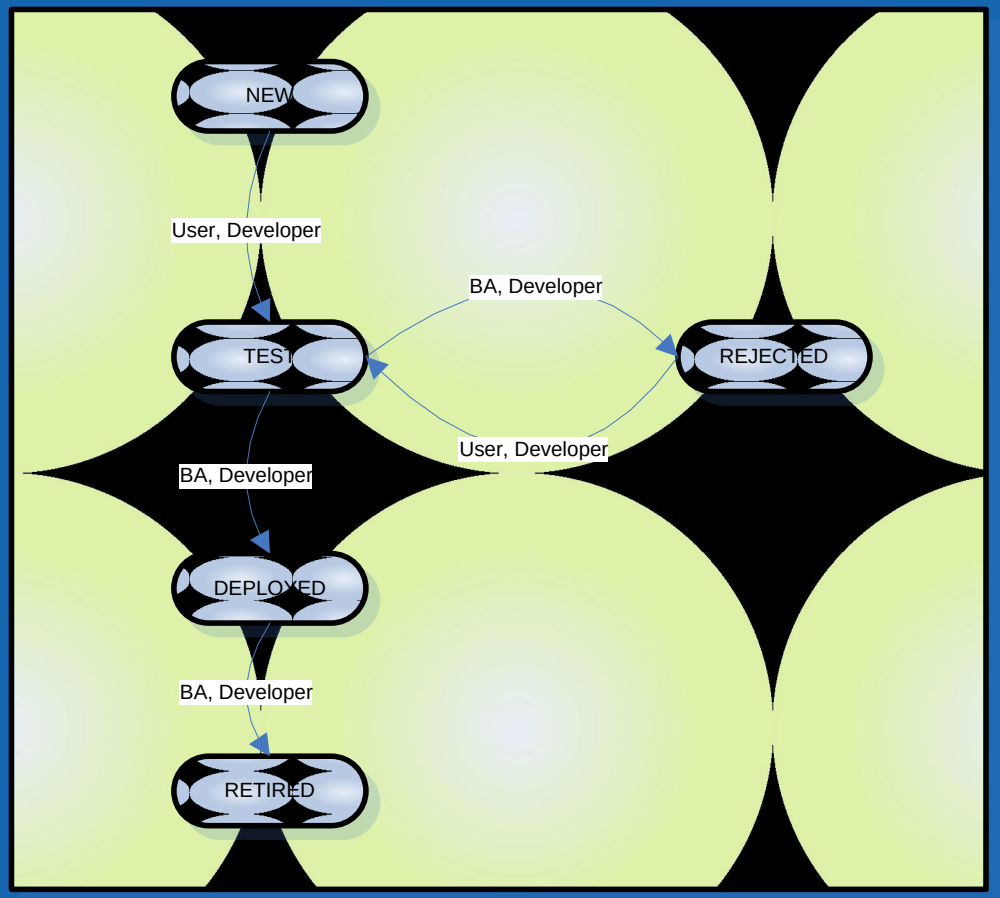
BRMS : intégration

- Intégration Java / .NET
  - Souvent simplissime
- Frameworks J2EE
  - Déploiement à chaud
  - Clustering / montée en charge

BRMS : outils

- Gestion de configuration
  - Edition
  - Exécution
- Processus de déploiement
  - Gestion des roles/droits
- Agilité
  - Tests automatiques

Exemple de Process



Les BRMS

retours d'experiences

Les défis

- Identifier les décisions à externaliser
- Extraire / Organiser le métier
- Trouver le bon process
  - Qui fait quoi ? BA, IT
  - Comment on déploie ?

Les faux problèmes

- Les performances
  - Rarement un problème !!!!
- La mise en production rapide
  - Attention !
- L'implication des utilisateurs
  - Toujours un plus

Les anti patterns

- Solution = Marteau
  - problèmes = clous
- Cycle de vie court
  - Bypassons les lourdeurs
    - Gestion de conf, tests, validation ne sont pas a éliminer …
- Utilisateur, langage naturel
  - C'est pas de l'informatique
    - Allégeons les process pour la prod (test) … mais testons

Impact sur le projet

- Ajout d'un nouvel axe de développement
  - Axe règle
  - Tâches/Rôles supplémentaires spécifiques aux règles
    - Utilisateurs ….
  - QA différente
    - Tests unitaires ne sont pas des tests d'intégration
- Méthodologie liée aux règles
  - Analyse/Capture
  - Cycles itératifs
  - Tests unitaires
  - Travail main dans la main IT/BA
  - Délégation progressive de responsabilité de IT vers BA

BRMS du marché

- IBM / ILOG-JRules
- FICO Blaze Advisor
- Drools / RedHat JBoss Rules
- Corticon BRMS
- Pegasystems PegaRULES

  etc…

- Non-BRMS
  - Jess

Les leaders
IBM / ILOG JRules

- Avantages:
  - Historique ➔ Outil complet et mature
  - Beaucoup d'outils pour TOUT gérer
  - Excellente intégration, surtout en J2EE (RES)
  - Excellentes possibilités en langage naturel
  - Excellents outils pour la délégation du pouvoir aux BAs / Policy Managers
  - Moteur: intègre plusieurs algorithmes d'exécution
- Inconvénients:
  - Ticket d'entrée ..
  - Outil complet ➔ Phase d'apprentissage plus longue

Les leaders
Drools / RedHat JBoss Rules

- Avantages:
  - Drools : Open source
  - Wiki, mailing list
  - Bons outils de gouvernance des règles
  - Simple à intégrer
- Inconvénients:
  - Edition en langage naturel moins puissante
  - Outils de reporting limités
  - Pas de framework d'intégration J2EE
  - Tables de décision limitées