anabasis

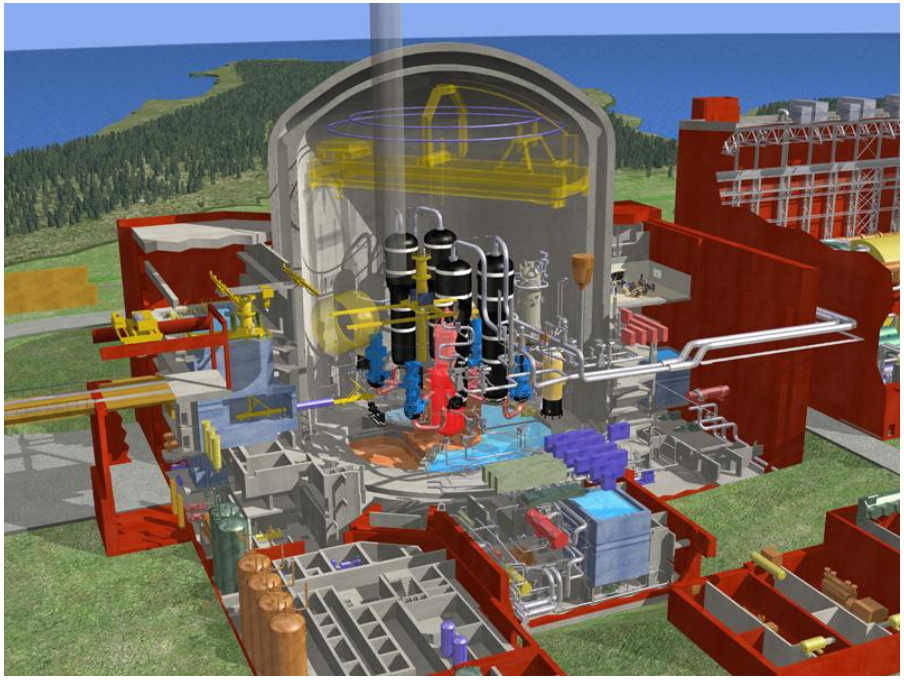
Ontologies et règles en logique

Applications industrielles pilotées par la sémantique métier

Michel Vanden Bossche mvandenbossche@anabasis-assets.com
Clément Sipieter sipieter@anabasis-assets.com

Nancy – 12 février 2020

Comment réaliser de l'ingénierie complexe ?



Science et ingénierie

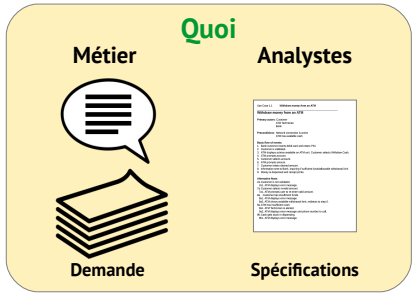
Théorie	Expression mathématique	Calcul numérique
Mécanique Hamiltonienne	Équations aux dérivées partielles	Méthode des éléments finis
Flux neutronique et section efficace	Fonction de Bessel, Laplacien	Méthode de projection incrémentale
...	...	...
Vocabulaire, concepts, relations	Expressions formelles	Algorithmes

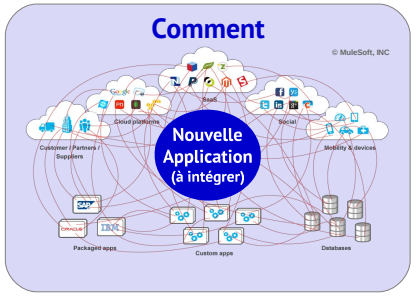
Comment développe-t-on un logiciel ? (1/2)

- **Trois types de logiciels** (Meir M. Lehman, 1980)

S-Programs	spécifiables (formellement)	Mettre un satellite en orbite
P-Programs	spécifiables à approximer	Jeu d'échec
E-Programs	non spécifiables (formellement)	SI « humains »

- **Comment construit-on* les E-Programs** (95+ % du logiciel) **?**





Essentiellement, « à la main », de manière pragmatique :

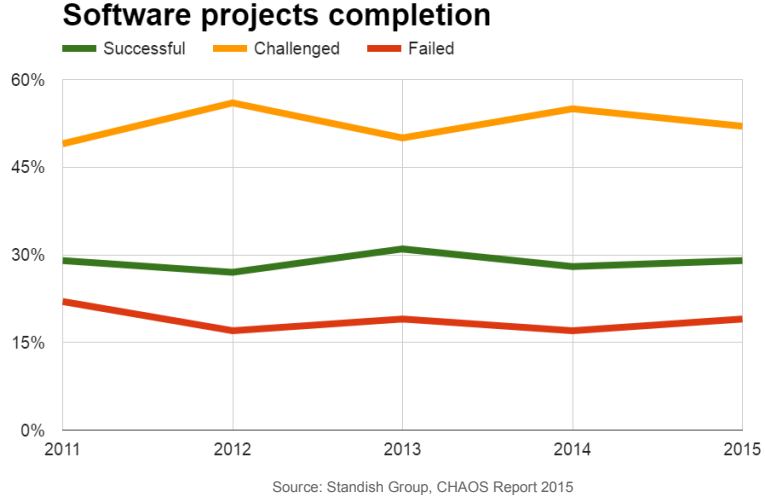
- les spécifications (MOA) sont informelles (Word, Excel, PPT, Visio)
- la construction (MOE) est majoritairement de la programmation
- Il n'y a ni « théorie », ni « expression mathématique » (l'algorithmique reste)

* En entreprise ou chez un éditeur de logiciels "COTS"

Comment développe-t-on un logiciel ? (2/2)

- **La réussite des projets informatiques reste un défi**
  - Louvois, ONP, SIHREN...
  - DHL NFE, Lidl eLWIS...
  - Danish Debt Management Agency

Malgré une profusion des technologies et de méthologies, le **taux de succès** des projets informatiques (30%) n'évolue pas favorablement.

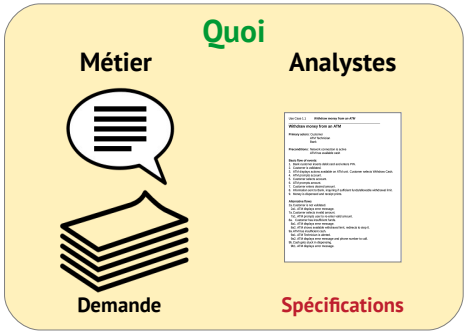


Source: Standish Group, CHAOS Report 2015

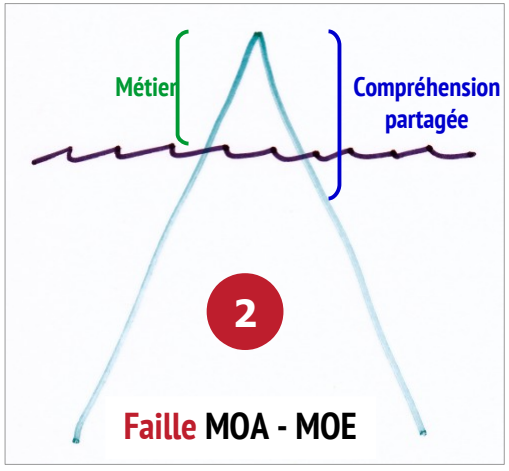
Comme le **logiciel** est l'élément **critique** de la **transformation numérique**, il est impératif de changer la manière de le concevoir et de le développer.

1. Pourquoi des ontologies ?

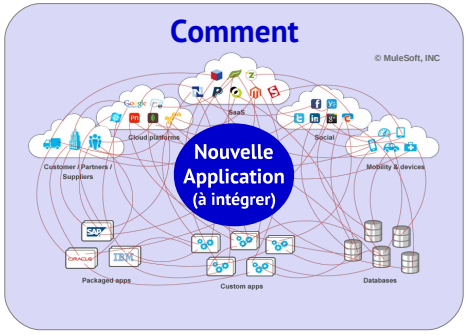
Pourquoi n° 1 – Spécification métier (MOA)



1



Source: Angela Wick



3

1. Un document (Word, Excel, PPT) n'est pas testable, donc pas validable
2. Tout n'est pas exprimé (soit : paraît évident, pas détaillé, pas identifié…)
3. Le seul livrable exécutable est une boîte noire pour les métiers, lié à une technologie particulière (spécifique [Java…], progiciel [SAP, HR Access…])

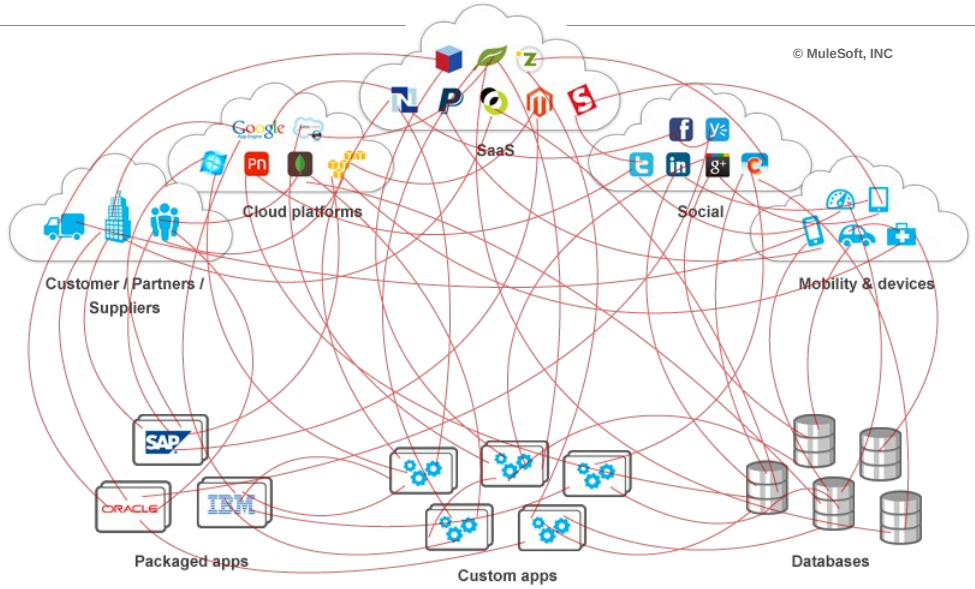
The hardest part of the software task is arriving at a ***complete and consistent specification,*** *and much of the essence of building a program is in fact the* ***debugging of the specification.***

Frederick P. Brooks Jr.

AMOA Métier

MOE Technique

Pourquoi n° 2 – Intégration

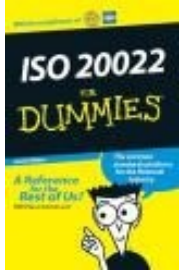


Complexité et entropie

- les mêmes données sont échangées avec des **syntaxes différentes**
- Leur **qualité** est **limitée** (si tout va bien) à un **silo**, pas à l'ensemble intégré
- la **sémantique** (le sens métier) de ces données n'est ni explicite, ni claire

The semantic barrier

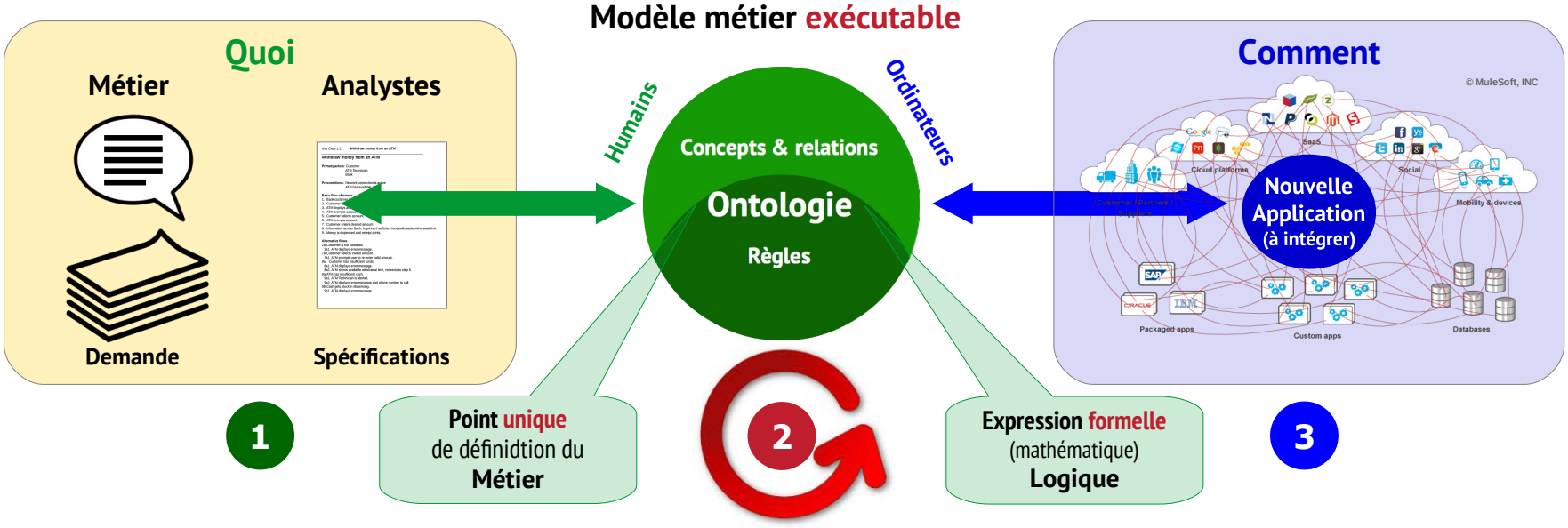
Once the syntax is out of the way, another barrier appears: the semantic barrier. Specialists in different domains or countries have developed their own jargon or vocabularies. Different words might refer to the same concept, or worse, the same word could have different meanings.



Messages financiers

2. Comment ?

Approche centrée-ontologie



1. Le **métier** et les **analystes** (MOA) expriment les **besoins** et les **spécifications** de manière **informelle** (sous forme de documents) ; ils fournissent aussi des **données de test**.

2. Des **ontologues** (AMOA) modélisent **100%** du problème métier en créant une **ontologie** qui décrit, **avec le langage du métier** (1) les **concepts** et leurs **relations** et (2) les **règles** métier. Cette ontologie est **exécutable** : elle est **testée**, **expliquée** et **améliorée itérativement**.

3. Une **API générée automatiquement** offre aux **informaticiens** (MOE) un accès **informatique** à l'ontologie. Grâce à cette API, ils **ajoutent** le **code technique** nécessaire (BD, IHM, intégration technique) pour obtenir une **application opérationnelle** (sur l'infrastructure) **pilotée par** l'**ontologie métier** (qui en forme le cœur), avec l'architecture technique souhaitée

Qu'est-ce qu'une **Ontologie** ?

- C'est une **spécification formelle** dans le **langage du métier**

Exemple d'une phrase d'un document de spécification RH (Word, Excel) :

> Un enfant est rattaché à un temps partiel si le temps de travail permet le rattachement et si le lien entre l'enfant et l'agent permet de prendre un temps partiel et si l'enfant a moins de 3 ans lors de la prise du temps partiel.

Cette proposition peut être exprimée en **logique formelle** (ensembliste) :

Concepts	personne, agent, enfant, temps de travail...
Attributs	nom, date (de naissance, de début, de fin)...
Relations	entre enfant et agent (qui sont des personnes)...
Règle	Si un agent est lié à un enfant, et cet agent a un temps de travail, et le temps de travail a une date de début et cet enfant a une date de naissance et la date de naissance est moins de trois ans avant cette date de début et le temps de travail permet un temps partiel pour un enfant alors L'enfant est rattaché au temps de travail de cet agent

C'est le **langage du métier**, **exécutable** (par raisonnement logique)

Ontologie avec **règles**

Cette **règle** n'est **pas** un **programme** (séquence d'instructions), mais une **proposition logique** qui définit ce qui est **vrai** du point de vue **métier** (**sémantique**), dans le langage du métier. Elle est **auto-documentée**.

Objectif : disposer d'un langage métier exécutable

Métier

Informatique

Un enfant est rattaché à un temps partiel si le temps de travail permet le rattachement et si le lien entre l'enfant et l'agent permet de prendre un temps partiel et si l'enfant a moins de 3 ans lors de la prise du temps partiel.

Suppose un langage exécutable proche de ceci

Humains

Manuel

Automatique

Machine

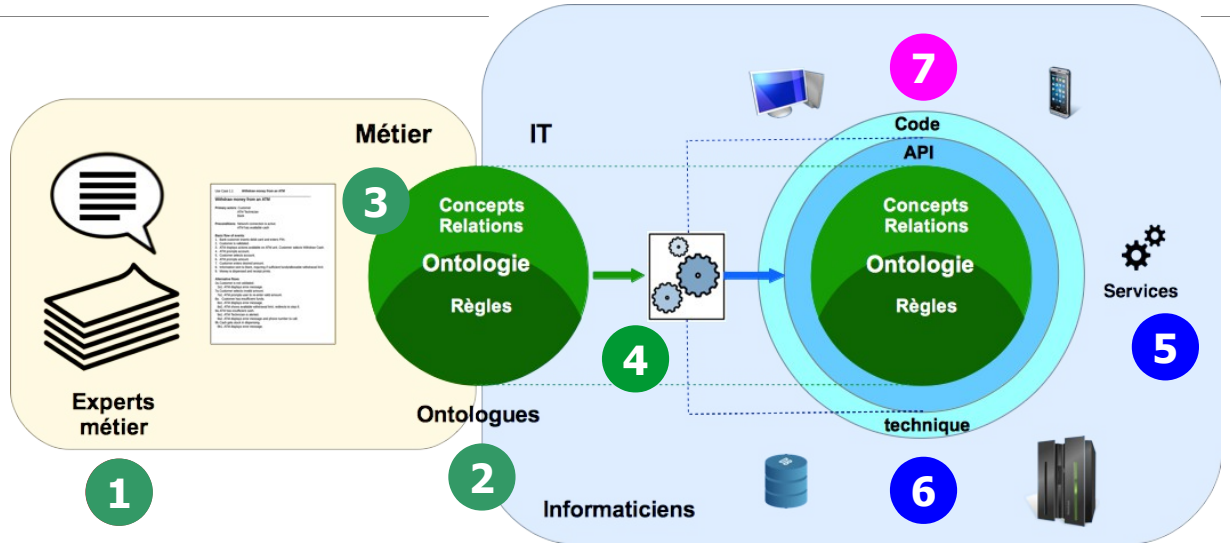
```
SELECT a.nudoss, f.sitcod, f.dateff,rtrim(g.preenf,' ')||'|'||to_char(g.naienf,'YYYY/MM/DD')||' 00:00:00||'
  from hr.zy00 a,  hr.zypo f, hr.zye5 g, PIPER.PIPER_AGENT2 pa
  where
  --
  --          cas 1 - congé parental CPAR01, CPAR02, CPPARE  da
  --
    a.nudoss = f.nudoss and f.sitcod in ('CPAR01','CPAR02','CP  RE')
  and a.nudoss = g.nudoss and g.typpar in ('2N','2L','2 ','02','2D','2F','2R','2T','2U','2Z')
  and (g.naienf <= f.dateff and g.naienf+1095 >= f.dateff)
  and a.nudoss = pa.nudoss
```

Programme

Tout ce qui est **métier** dans ce code **disparaît**

Boîte noire (pour les métiers) ! Conforme ? Correct ? Effets de bord ?

Processus triplement agile



Métier
1. Exprime les besoins, fournit les cas de tests, vérifie l'ontologie

Ontologues
2. Construisent l'ontologie : (1) concepts et relations, (2) règles

Métier+Ontologues
3. Testent l'ontologie, comprennent le pourquoi des résultats, modifient

Ontologues
4. Génèrent une API : l'ontologie est « consommable » par les informaticiens

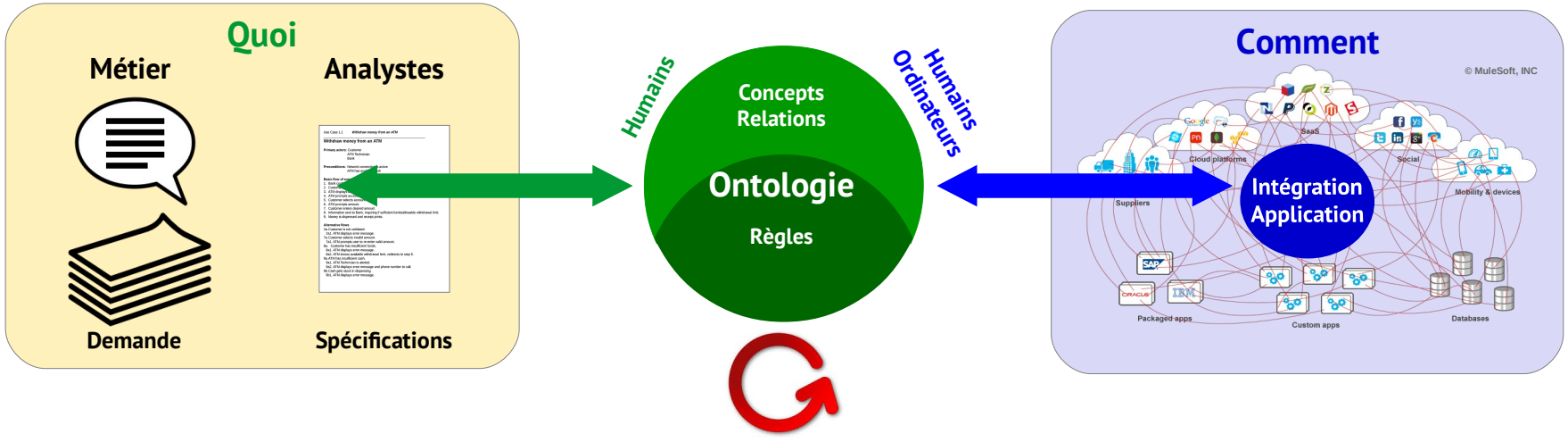
Informaticiens
5. Définissent l'architecture
6. Développent les aspects techniques (IHM, DB, intergiciel, intégration…)

Ergonomes
7. Conçoivent et développent l'ergonomie de l'IHM (par itération)

3. Quoi ?

Outillage - ODASE

ODASE est une plate-forme et un ensemble d'outils, de développement, de débogage métier, d'exécution et d'intégration de **SI centrés ontologies**



Ontologie

Modèle d'un domaine de connaissances

Théorie
Logique
Standards

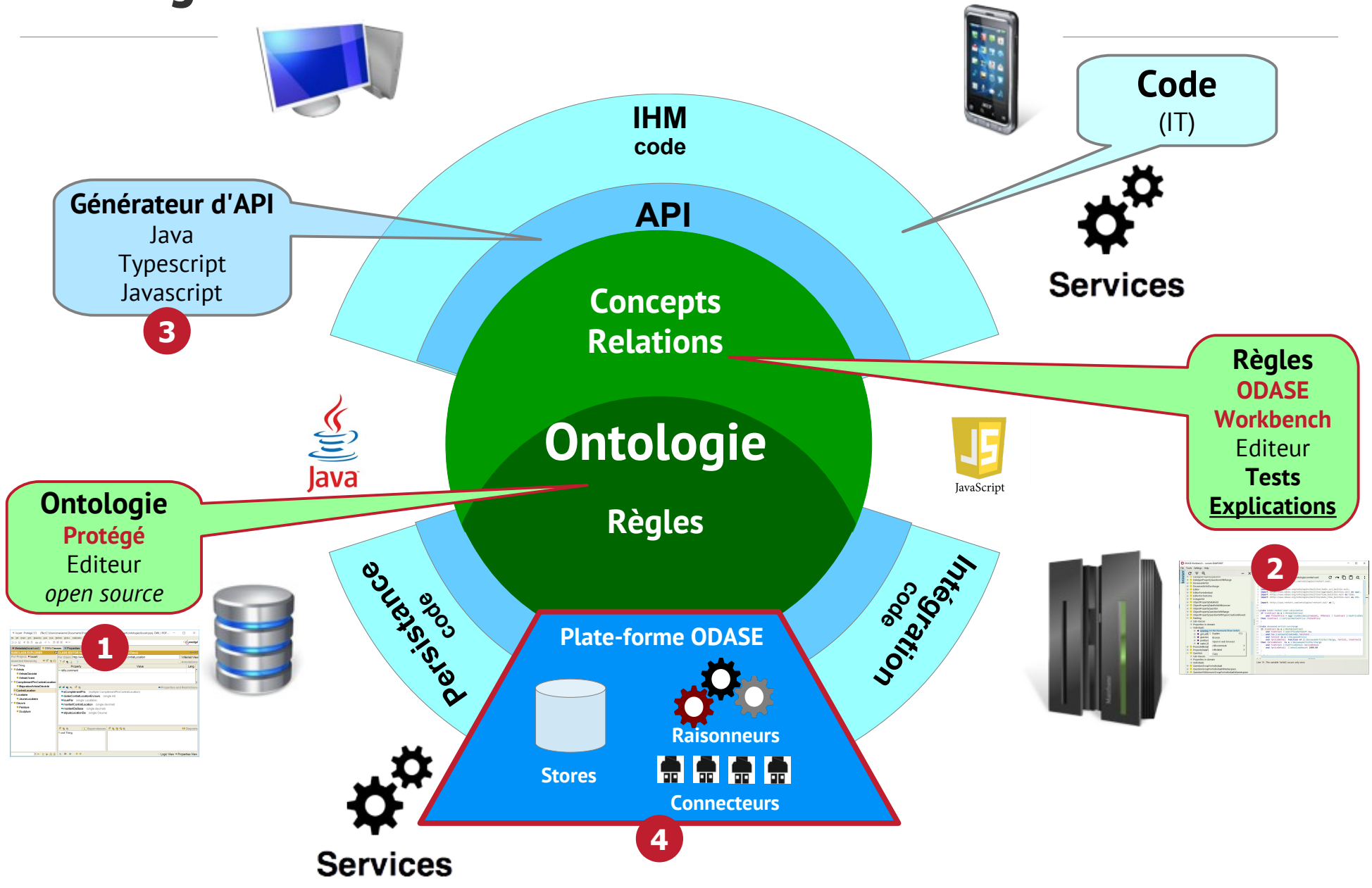
- Vocabulaire: concepts, relations, attributs
- Règles et axiomes
- Logique ensembliste, inférence (raisonnement)
- Explications (débogeur métier)
- Standards: OWL, RDF, RIF/SWRL (W3C)

Architecture informatique

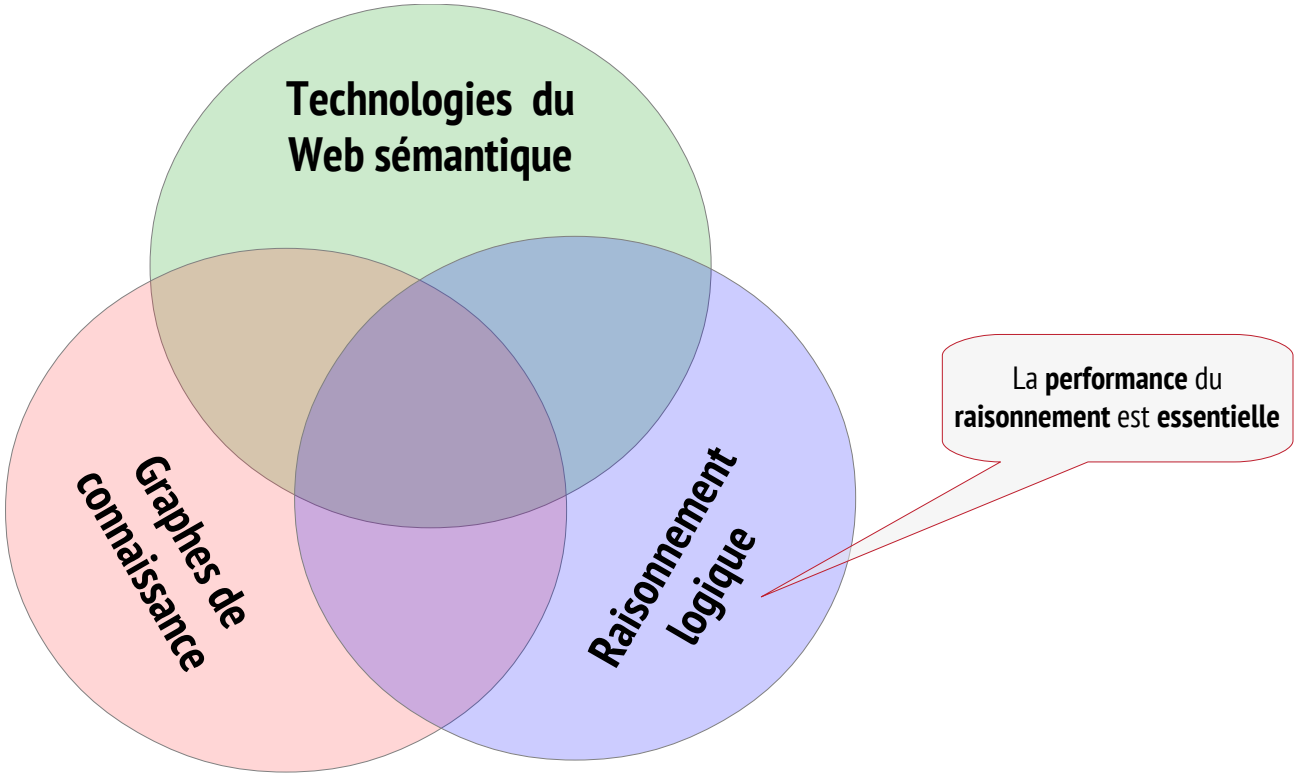
Contructions
Outils du marché

- Génération d'API *type-safe* (Java, TypeScript)
- Services REST, sérialisation JSON-LD
- Infrastructure x86 standard
- Windows ou Linux, physique ou cloud
- Connexions DB (SQL, NoSQL), IHM (Angular...)
- Intégration Excel, COBOL, SAP, XML...

Outillage – ODASE : Plate-forme et outils



Technologies mises en œuvre (1/2)



- Standards du **Web sémantique** définis par le W3C : OWL, RDF, RIF/SWRL...
- Autres standards du **W3C** : HTTP, URI, HTML, XML...
- Autres standards **techniques** : Java, JavaScript/TypeScript, REST, HATEOAS...

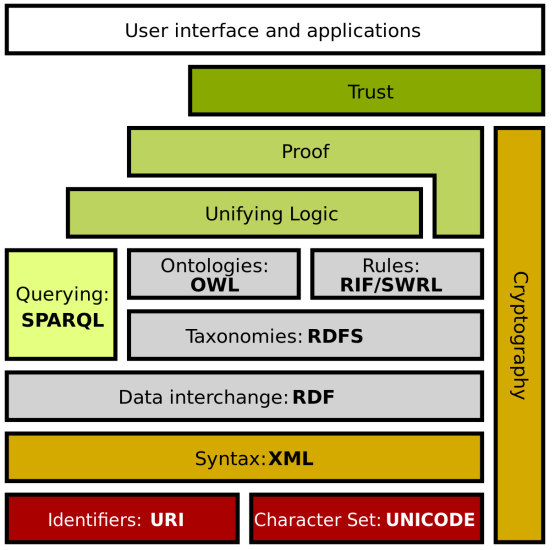
Technologies mises en œuvre (2/2)

Technologies du Web sémantique

OWL
- **Web Ontology Language**
- Concepts & relations en logique ensembliste

RDF
- **Resource Description Framework**
- Triples: sujet – prédicat – objet

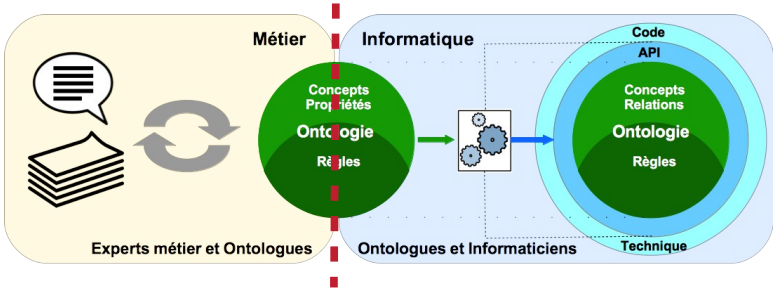
SWRL
- **Semantic Web Rule Language**
- Règles métier déclaratives, en logique



4. Bénéfices

Bénéfices (1/4)

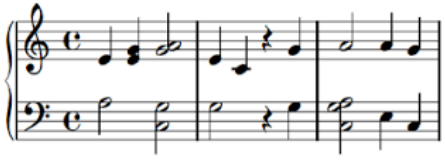
- **Séparation complète et claire**
  entre métier et informatique



- **Connaissance métier partagée**
  par tous les intervenants et les systèmes

- **Pérennité**

  Exemple : une partition qui peut être jouée
  - sur tout instrument
  - sur un synthétiseur avec interface MIDI
  - par un robot (qui aurait appris à jouer)

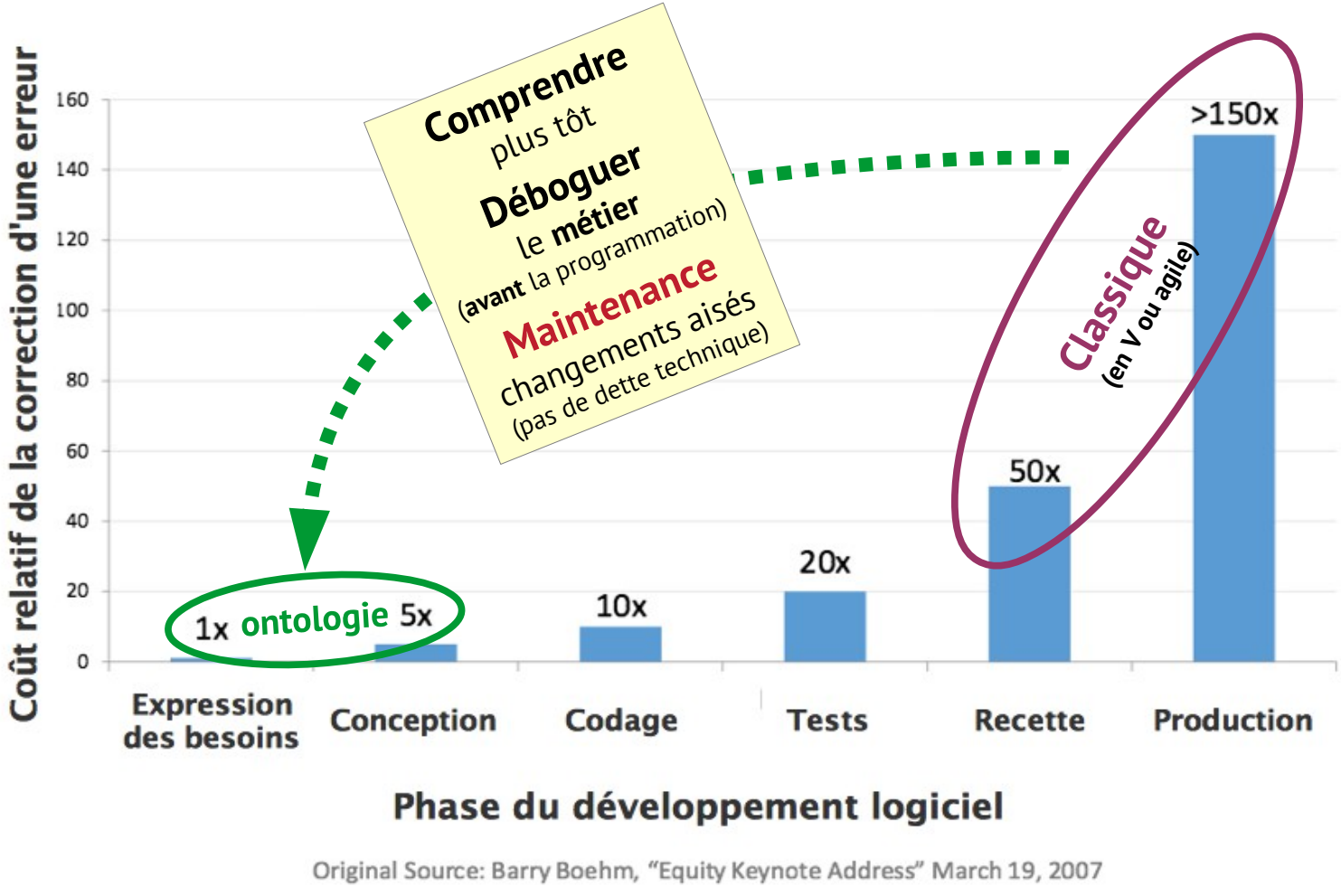


Le langage du métier

- **Productivité des développeurs**

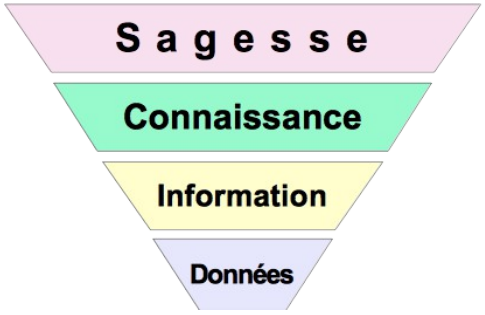
  20 à 50 fois moins de lignes de code
  - le cœur métier est dans l'ontologie
  - il a été « débuggé » pendant la conception
  - le code informatique n'est que technique : il est facile à débugger pour les informaticiens
  - la productivité augmente considérablement avec l'effet de co-construction Métier–IT
  - les développeurs « full stack », difficiles à trouver sur le marché, sont moins sollicités

Bénéfices (2/4)



Très grand impact sur la construction et la maintenance

Bénéfices (3/4)



Il faut consulter un médecin si X a moins de 4 ans
Une température > 38 °C est un signe d'infection
Température corporelle en °C de la personne X
38,5

Aujourd'hui : approche « centrée applications »

Connaissance et sagesse résultent de l'écriture de programmes spécifiques qui sont **centrés processus** (entrée → processus → sortie).

Chaque application a tendance à **recréer les données** dont elle a besoin, ce qui engendre **redondance**, **complexité**, **coûts**, **rigidité, erreurs**...

Ontologie : approche « centrée connaissances & données »

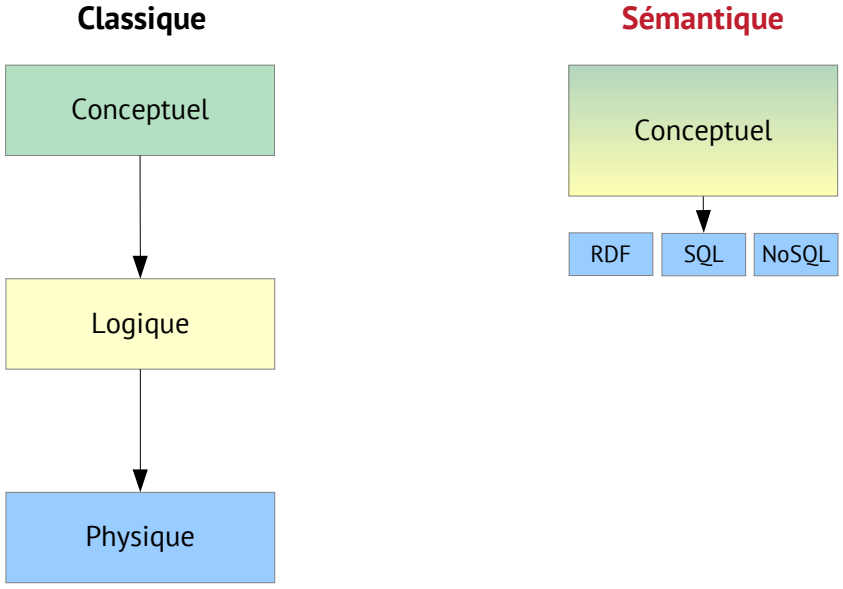
C'est une approche où le **métier** est défini en un **point unique** (ontologie avec des règles) **hors de tout silo**. Les **données** ont une sémantique claire et sont exploitées et gouvernées de manière **unifiée**.

La **qualité des données** est assurée par sa **sémantique**.

Bénéfices (4/4)

- **Le modèle conceptuel est directement opérationnel**

  Contrairement aux systèmes d'information classiques, il n'y a qu'**un seul modèle**, le modèle conceptuel. Les données opérationnelles sont gérées sous forme de graphes (ou « triplets » RDF : sujet – prédicat – objet). Ces triplets peuvent être persistés et gérés tels quels dans un *store* RDF ou dans des *stores* SQL ou NoSQL (comme les BD Graphes), moyennant une intégration (par exemple R2RML : https://www.w3.org/TR/r2rml/), ce qui facilite aussi l'**agilité technique** (modernisation) en plus de l'agilité métier.



5. Règles : SWORD

De SWRL vers SWORD

- **W3C**
  - **SWRL** est une proposition de standard (2004 : https://www.w3.org/Submission/SWRL/)
    - Utilisé pour aller au-delà de ce qui peut être exprimé en OWL
    - Clauses de Horn dont les atomes sont des classes et des propriétés OWL
  - W3C a développé **RIF**
    - Rule Interchange Format
    - Contient des « dialectes »

      RIF-PRD Production Rule Dialect (non-déclaratif)

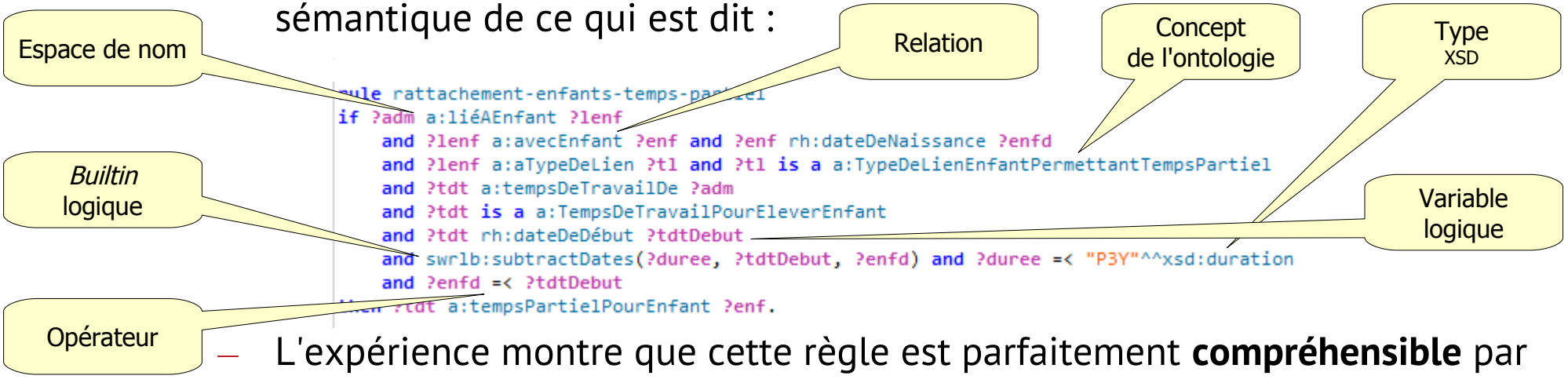
      RIF-BLD Basic Logic Dialect (déclaratif), équivalent à SWRL
- **Contraintes du monde réel**
  - SWRL et RIF-BLD sont insuffisants dans le monde « réel »
  - C'est la raison du développement dans ODASE de **SWORD**
    - Syntaxe textuelle plus conviviale que celle de Protégé
    - Extensions indispensables

SWORD – Syntaxe textuelle de SWRL

SWRL
« abstrait »

Si un agent est lié à un enfant,
et cet agent a un temps de travail,
et le temps de travail a une date de début
et cet enfant a une date de naissance
et la date de naissance est moins de trois ans avant cette date de début
et le temps de travail permet un temps partiel pour un enfant
Alors
le temps de travail est rattaché à l'enfant

- Cette règle est facilement compréhensible par un humain, mais elle doit être compréhensible par un ordinateur pour être exécutable (et testable).
- Il faut pour cela ajouter quelques éléments de syntaxe, sans changer la sémantique de ce qui est dit :



- L'expérience montre que cette règle est parfaitement **compréhensible** par un sachant métier puisque ce qui est important, à part les **si ... et ... alors**, sont les concepts et propriétés exprimés dans le **langage métier**.

SWORD – Implémentation et extensions (1/4)

- **Implémentation**
  - Atteindre une performance comparable à celle d'un codage classique
  - *Query-based*, pas *Classification-based*
  - Pas Tableau
    - Top-down SLDNF
    - Rule rewriting
    - Or-parallelism
- **Extensions indispensables**
  - Existentiels
  - Agrégats
  - Négation
- **Discuté dans la proposition SWRL 2** http://wiki.ruleml.org/index.php/SWRL
  - Propose *Existentials* et *NAF* mais pas les agrégats

SWORD – Existentiels (2/4)

- **Function of**
  - Individus inferés
  - Il existe un individu « fonction de » (pour tout) …
  - Builtin particulier pour (dé-)construire une URI à partir d'objets
    - URIs, litéraux données
- **Exemple**
  - Tout couple mère, père ayant au moins un enfant est une famille traditionnelle avec enfant(s)

```
if ?m mèreDe ?e
    and ?p pèreDe ?e
    and ?famille function of[?m, ?p]
then ?famille is a FamilleTraditionnelleAvecEnfant
    and ?m estMembreDe ?famille
    and ?p estMembreDe ?famille
    and ?e estMembreDe ?famille.
```

SWORD – Agrégats (3/4)

- **Agrégats**
  - Min, max, product, count...
- **Exemple**
  - Le prix de location et la somme des détails par œuvres plus 10€ de frais de dossier

```
if ?c is a l:ContratLocation
    and ?details = aggr:sumDecimals{?valeur, ?detail |
            ?detail is a l:DétailPrixParOeuvre
            and ?detail l:détailDuContrat ?c
            and ?detail l:prixPourOeuvre ?valeur
        }
    and ?prix = ?details + 10.0
then ?c l:prixLocation ?prix.
```

SWORD – Négation (4/4)

- **Negation As Failure** (NAF)
  - Permet une fermeture locale
- **Exemple**
  - Tant qu'un artiste n'est pas reconnu comme décédé, alors il est considéré comme vivant

```
if ?a is a l:Artiste
    and NAF(?a is a l:ArtisteDécédé)
then ?a is a l:ArtisteVivant.
```

6. Démonstration pédagogique

Démonstration LocArt

- **Quoi**

  Démonstration à vocation pédagogique : la location d'oeuvres d'art

- **Contexte de la démonstration**

  Un atelier 1 a défini les exigences du métier

  Au cours d'un atelier 2, les ontologues/informaticiens

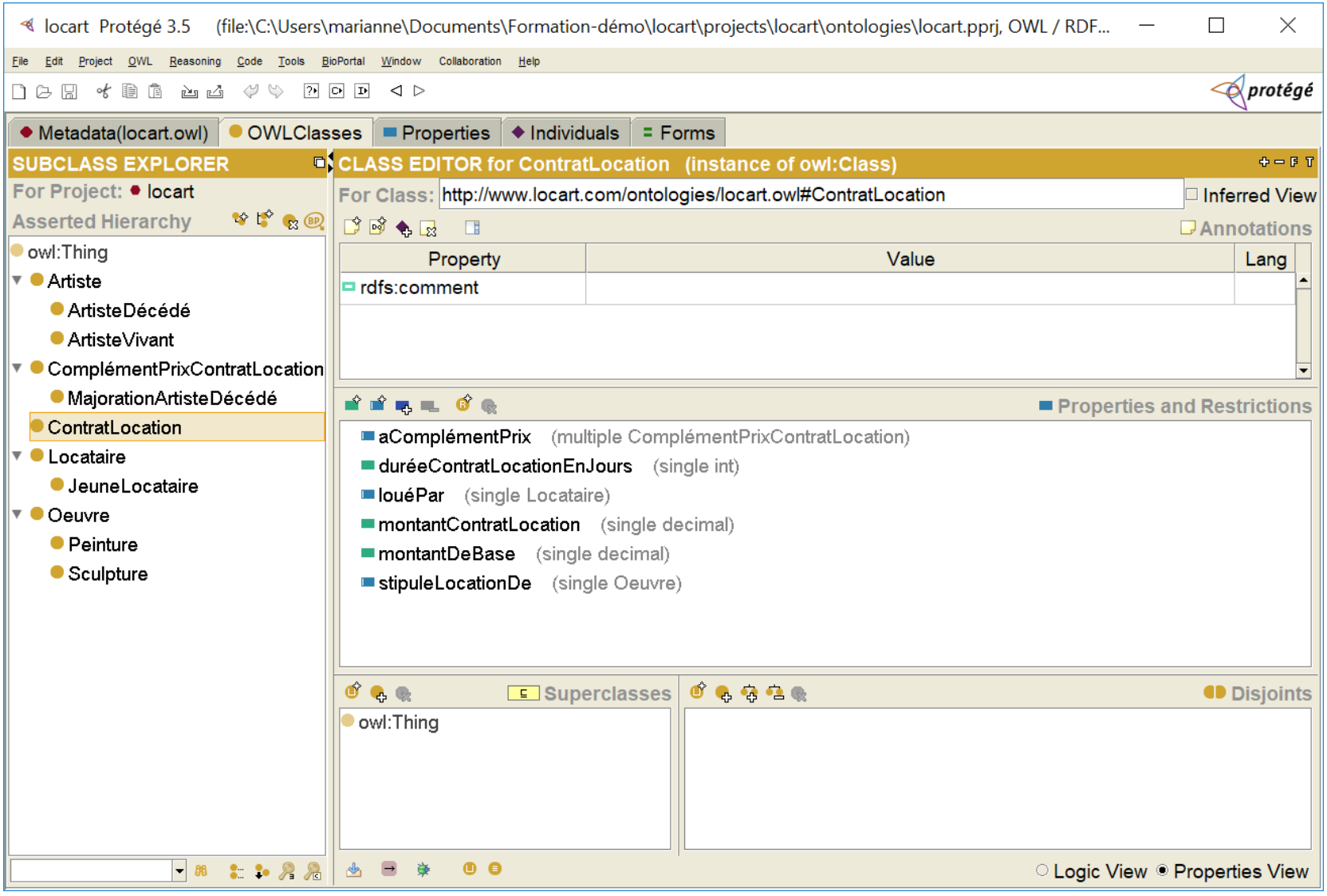
  - montrent l'ontologie (le modèle) et l'application v1
  - prennent connaissance des nouvelles exigences du métier
  - montrent l'ontologie et l'application v2

- **Technologie**

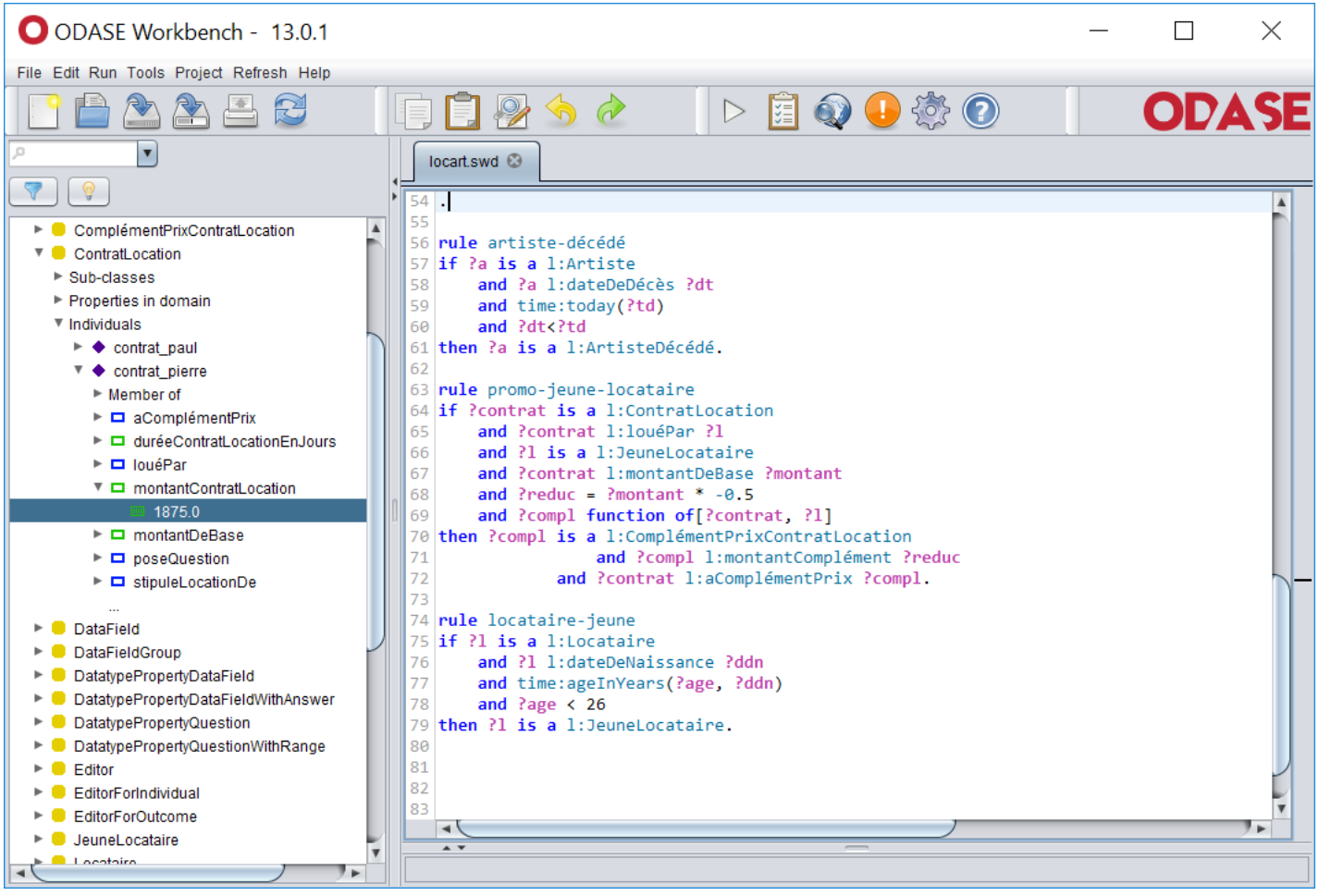
  Aucun composant particulier :

  - Client — Web avec Angular 6 (pourrait être Vue, React...)
  - Architecture — REST avec JSON-LD (*Linked Data*)
  - Serveur — Windows 10 (ou Linux), Java, Jetty, Jersey...

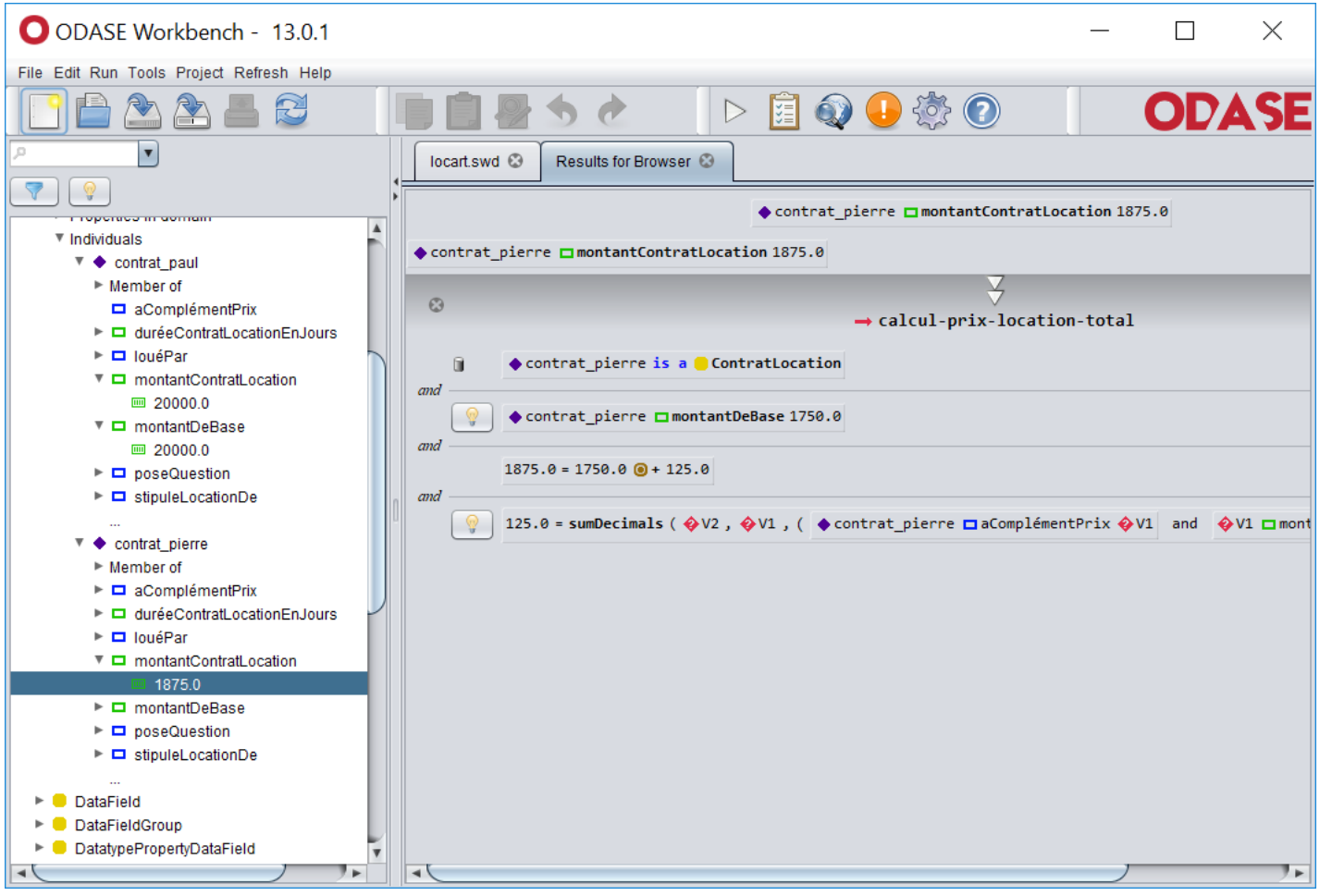
Concepts et propriétés (Protégé)



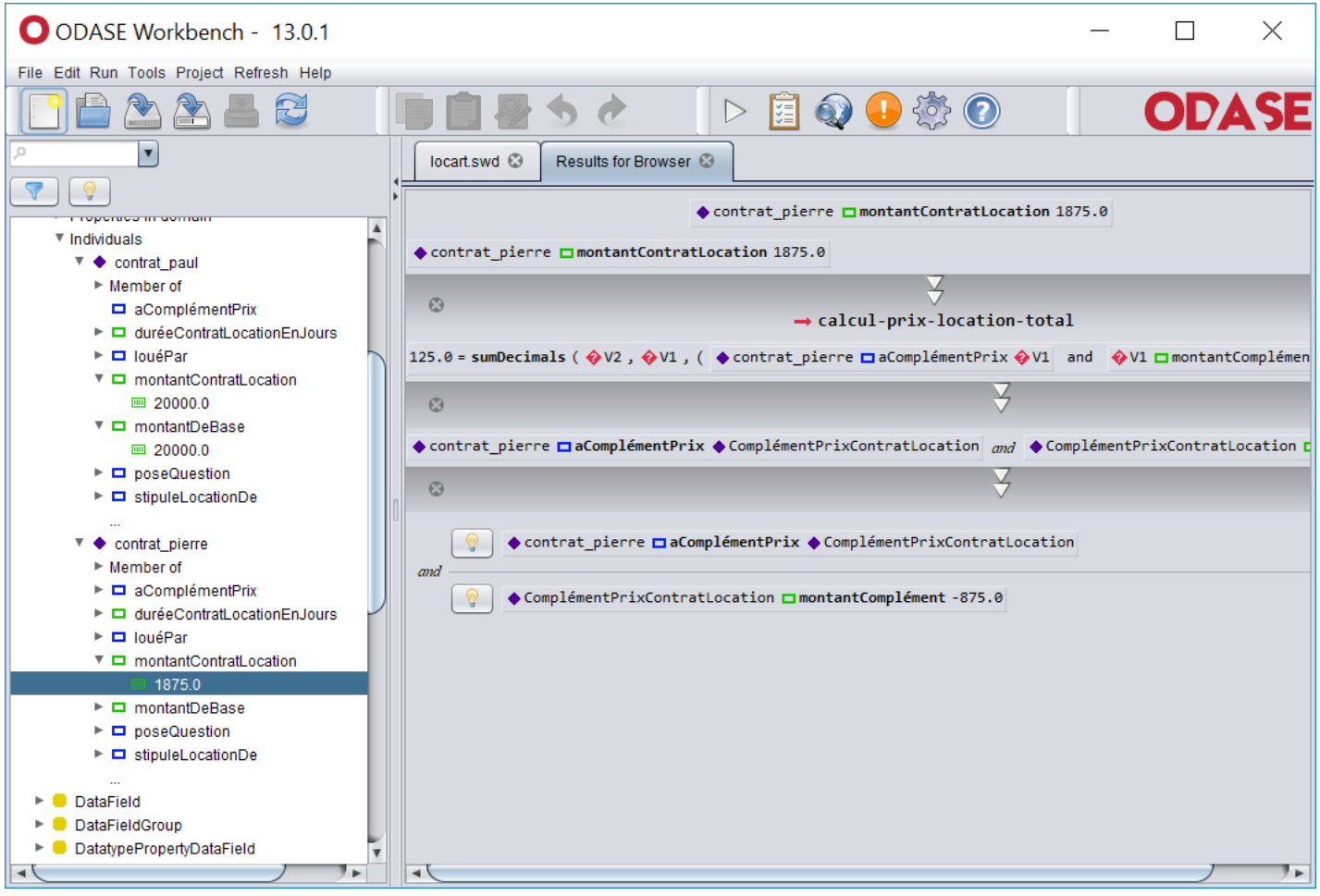
Règles (ODASE Workbench)



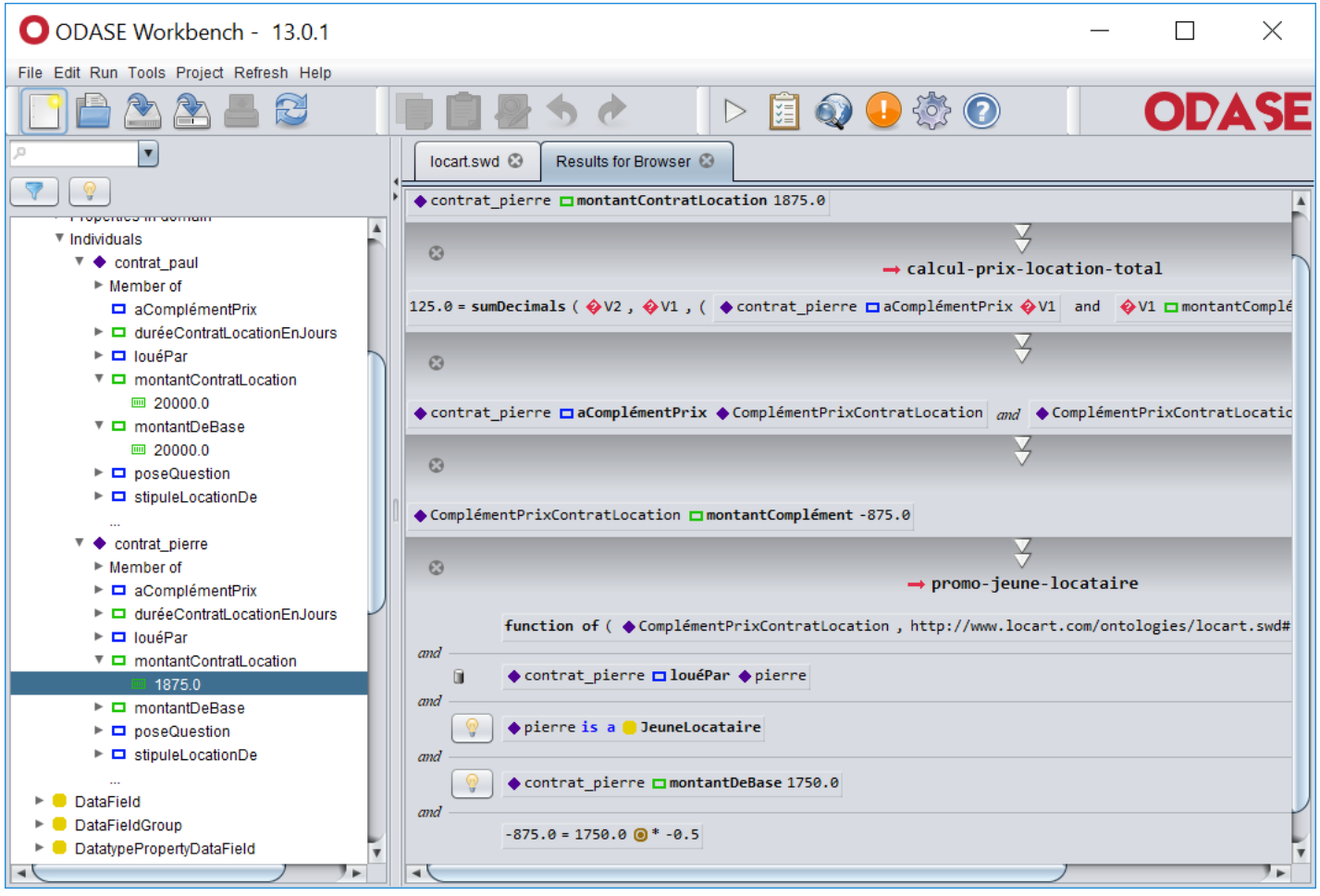
Explication #1 (ODASE Workbench)



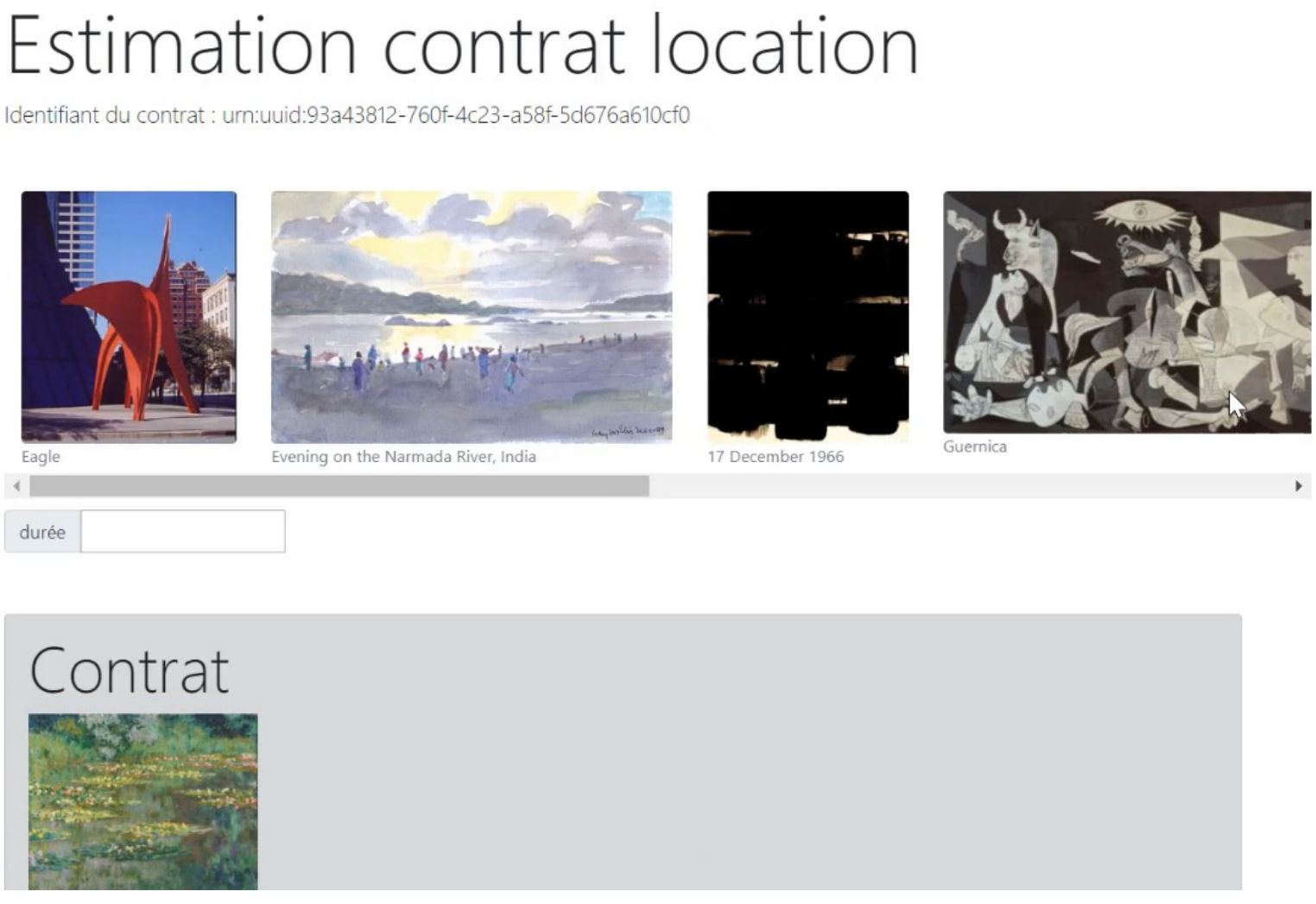
Explication #2 (ODASE Workbench)



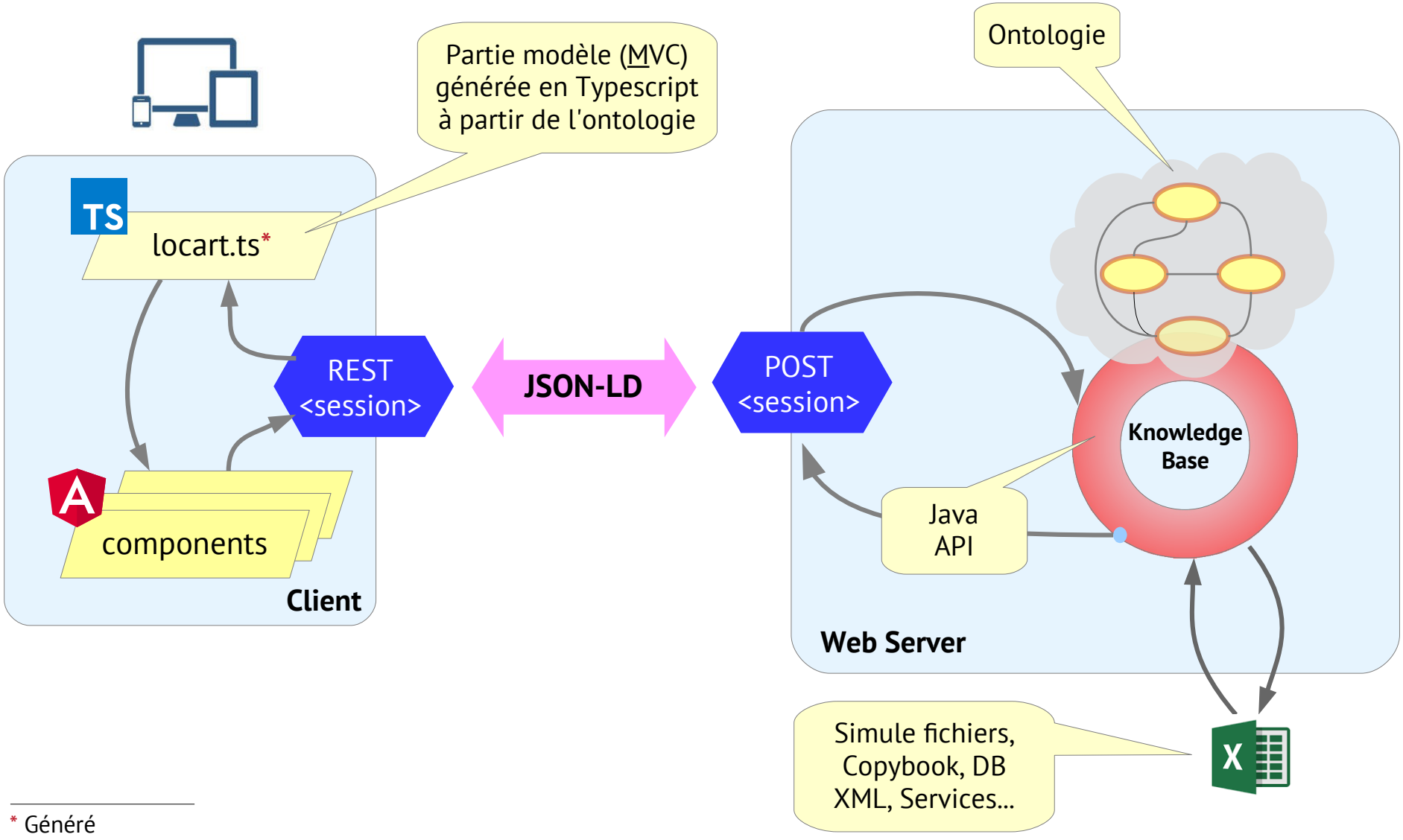
Explication #3 (ODASE Workbench)



Application Web (exemple Angular) **pilotée par l'ontologie**



Architecture technique (typique Web, avec Angular, mais pourrait être différente)



* Généré