Université Claude Bernard - Lyon I

Rapport de stage de Master 1 Informatique
Année scolaire 2004-2005

ATELIER D'ANALYSE DE L'ACTIVITÉ DE CONDUITE À PARTIR DE DONNÉES DESCRIPTIVES

Stagiaire :
Jean-Marc Trémeaux

Encadrants :
Alain Mille (LIRIS)
Olivier Georgeon (INRETS)

Laboratoire d'InfoRmatique en Images et Systèmes d'information

Institut National de Recherche sur les Transports et leur Sécurité

Résumé

Des données descriptives de l'activité de conduite ont été receuillies à partir de nombreuses expérimentations. Ces données sources sont structurées sous la forme de successions d'objets de collecte, datés et localisés. Ces séquences sont appelées "traces brutes". Notre objectif est d'utiliser ces traces comme support d'une analyse cognitive du conducteur, c'est à dire une description du modèle mental élaboré par le conducteur pour guider ses comportements. Nous désignons ce modèle mental par "schéma de conduite" ou "Frame".

Nous souhaitons construire un outil pour manipuler et visualiser ces traces. L'objectif est que cet outil soit utilisable par un analyste psychologue pour décrire, tester, valider ou invalider des hypothèses sur les schémas de conduite et leur concordance avec les comportements. Cet outil devra permettre de construire la trace brute conformément à un modèle de collecte, ainsi que de la visualiser sous forme graphique.

Mots-Clés

Modélisation cognitive, Conduite automobile, MUSETTE, Trace, RDF.

Table des matières

1 Contexte théorique

Le travail effectué lors de ce stage s'inscrit dans deux contextes scientifiques. D'une part le LIRIS (Laboratoire InfoRmatique en Image et Systèmes d'information) avec l'acquisition de connaissances "situés" et d'autre part le LESCOT (Laboratoire d'Ergonomie et Sciences Cognitives pour les Transports) avec la modélisation cognitive du conducteur automobile.

1.1 Le LIRIS

1.1.1 Présentation générale

Le LIRIS (Laboratoire InfoRmatique en Image et Systèmes d'information) est né début 2003 à la suite du regroupement de plusieurs laboratoires de recherche lyonnais (LIGIM, LISI, RFV) et d'individualités du domaine des Sciences et Techniques de l'Information et de la Communication. Il est associé au CNRS avec le label UMR 5205. Il a deux thèmes principaux de recherche : l'image numérique et les systèmes d'information, qui sont déclinés suivant quatre axes scientifiques :

- Axe 1 : Données, Documents et Connaissances (D2C) ;
- Axe 2 : Images et vidéos : segmentation et extraction d'information ;
- Axe 3 : Modélisation et réalité augmentée ;
- Axe 4 : Systèmes d'information communicants.

Regroupant environ 150 personnes, dont près de 70 enseignants-chercheurs, le LIRIS a quatre tutelles : l'INSA de Lyon, l'Université Claude Bernard Lyon 1, l'Ecole Centrale de Lyon et l'Université Lumière Lyon 2, et des sites à La Doua, Ecully et Bron.

1.1.2 L'axe Données, Documents et Connaissances

Le défi à relever par les systèmes informatiques omniprésents en tant que supports pour les activités extrêmement variées de la société, est de "faire sens", pour les utilisateurs et les uns pour les autres. Les utilisateurs (ou agents systèmes) peuvent exploiter les services mis à disposition en les interprétant comme connaissances ou comportements intelligents dans le contexte de leur tâche courante.

L'axe D2C travaille à l'élaboration de propositions théoriques et pratiques pour relever ce défi qui pourrait se résumer par la question :

Comment faire émerger des connaissances ou/et des comportements intelligents des systèmes informatiques ?

Il a pour cibles communes la découverte, la gestion, l'exploitation et le partage des connaissances telles qu'elles s'expriment dans ce que nous

proposons d'appeler des "traces documentaires". Une trace documentaire est un conteneur de connaissances qui peut prendre des formes très variées :

- les documents dans leur acception la plus habituelle du terme, en tant qu'artefacts construits, et en particulier les documents numériques,
- les contenus des bases de données en tant que conteneurs de connaissances,
- les traces d'interactions liées à l'usage du système informatique par l'utilisateur,
- les traces d'interactions entre agents informatiques,
- les traces d'usages captées dans les grandes bases de données (permettant de découvrir des profils),
- les codes complexes régissant des phénomènes d'évolution ou de comportement (génomes, séquences d'événements), etc.

1.1.3 Le thème Cognition, Expérience et Agents Situés

La pérennisation du savoir (et du savoir-faire) revêt une importance grandissante dans un nombre croissant d'entreprises. Les outils dans ce domaine peuvent se présenter comme des Aides à la Décision et aussi comme des Aides-mémoire. Dans le premier cas (aides à la décision), on doit clairement se garder de l'utopie consistant à viser des systèmes de "décision automatique" ; il s'agit bien de fournir à l'utilisateur des éléments lui permettant d'avoir une vue plus précise de son problème et des conséquences prévisibles de ses choix. Le second cas inclut aussi bien la mémoire collective que celle d'un individu. Au-delà de l'utilisation par l'ingénieur pour l'aide à la conception par exemple, il convient d'envisager l'utilisation des informations à des fins pédagogiques, par exemple dans le cadre d'EIAH (Environnements Informatiques pour l'Apprentissage Humain).

Il s'agit donc de mobiliser les possibilités des systèmes informatiques pour développer des assistants "intelligents" des tâches médiatisées. Les interactions homme-homme médiatisées par les systèmes tout comme les interactions agents-agents forment le fond commun de l'axe thématique. Les systèmes informatiques eux-mêmes ne peuvent en effet plus être considérés comme des entités fermées intégrant des mécanismes automatiques pour des tâches spécifiques, mais constituent des systèmes ouverts, s'appuyant explicitement sur la prise en compte des diverses interactions qu'ils entretiennent avec un environnement complexe et changeant : il s'agira alors de voir à quel point il est possible de voir ces systèmes comme des "écosystèmes artificiels". Cette facette de la problématique est en cours de construction dans l'axe thématique.

La prise en compte des interactions comme fondement de l'émergence du sens implique la collaboration avec les spécialistes des sciences cognitives. Une première application d'une telle collaboration concerne la prise

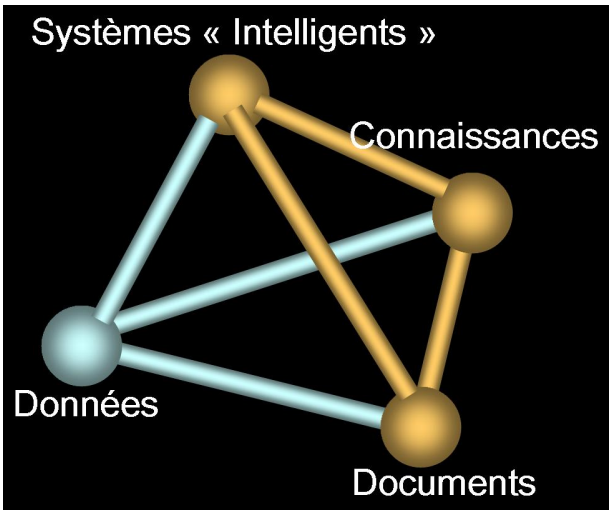

FIG. 1 – Positionnement du thème CEXAS dans l'axe D2C.

en compte des aspirations des interlocuteurs d'un système (aussi bien ceux qui l'interrogent, ceux qu'il contribue à former, que ceux qui l'alimentent en connaissances). Une seconde approche consiste à étudier en parallèle (chez les humains et dans les logiciels), les mécanismes d'évocation, tels que ceux qui permettent de reconnaître dans un problème nouveau des éléments de similitude avec un problème déjà résolu en vue d'adapter sa (ré)solution.

1.2 MUSETTE

1.2.1 Principe

MUSETTE [5] (Modeling USEs and Tasks for Tracing Experience) est un modèle général pour capter l'expérience d'usage d'une ressource. Il a pour objectif de réutiliser cette expérience en contexte pour assister l'usager dans son interaction avec cette ressource.

Il se base sur la capture d'une trace d'utilisation destinée à être analysée à l'aide de signatures de tâches expliquées. Ces signatures de tâches expliquées permettent d'une part d'identifier des épisodes dans la trace et d'autre part de leur rattacher des explications susceptibles d'aider l'utilisateur.

MUSETTE a pour l'instant exclusivement été appliqué à l'assistance à l'utilisation de systèmes informatiques. Dans ce contexte, son architecture générale est donnée par la figure 2.

Un utilisateur qui n'est pas directement observable par le système agit sur lui pour la réalisation d'une tâche, ce qui entraîne des modifications dans le système.

Un agent observateur observe ces changements selon un modèle d'observation et génère une trace primitive conforme à un modèle d'utilisation.

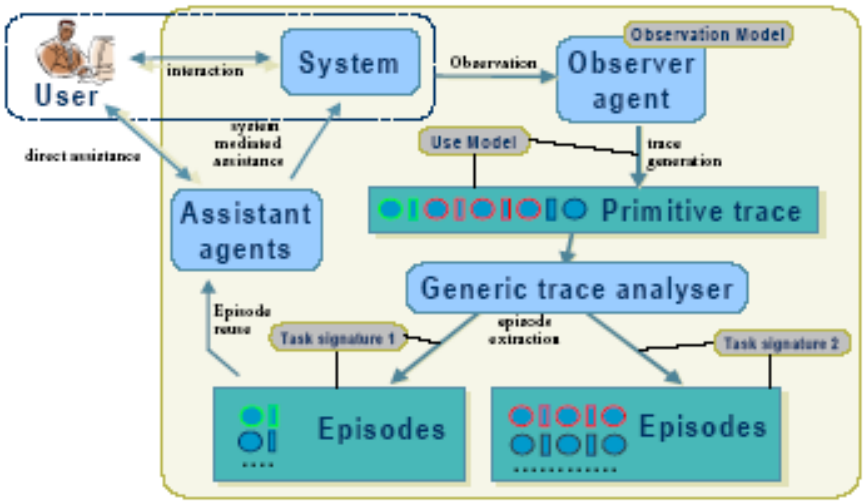

FIG. 2 – Le modèle MUSETTE

Ensuite, un analyseur de traces génériques extrait des épisodes signifiants de cette trace primitive en se basant sur des signatures de tâches expliquées.

Ces épisodes peuvent être réutilisés par un agent assistant qui peut assister l'utilisateur, soit en tant qu'agent clairement distinct du système, soit en modifiant directement le système.

1.2.2 Modèle d'utilisation et modèle d'observation

On appelle modèle d'utilisation la définition des constituants de la trace primitive.

Il est constitué d'objets d'intérêt (OI) qui peuvent appartenir à l'une des trois catégories : entité, événement ou relation. Les entités sont des objets présents à l'utilisateur pendant son interaction avec le système, les événements sont des faits qui surviennent pendant cette interaction. Les relations sont de type binaire et peuvent relier toute entité ou événement à tout autre.

Le modèle d'utilisation d'un système va donc définir quels sont les entités, événements et relations observables pendant l'utilisation du système. Il dépend largement de l'usage qu'on voudra faire ultérieurement des traces.

Le modèle d'observation doit ensuite être défini comme un ensemble de règles et de moyens permettant d'extraire du système les objets définis par le modèle d'utilisation. Il doit faire l'objet d'une programmation spécifique liée à chaque système observé.

Une fois que les modèles d'utilisation et d'observation sont définis il reste à générer les traces sous la forme d'une succession d'états et de transitions (cf. figure 3).

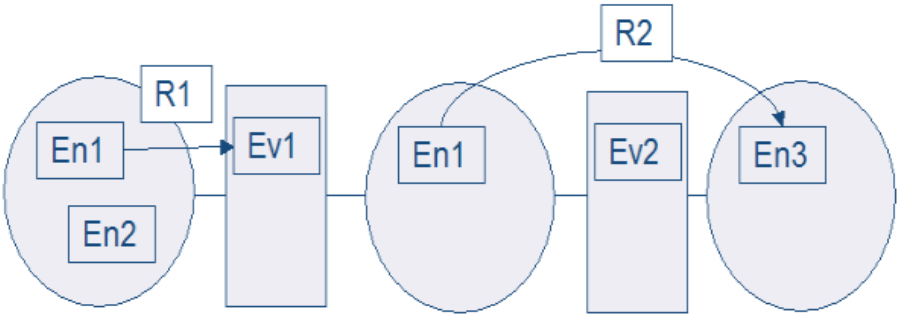

FIG. 3 – Formalisme de la trace primitive

Les entités sont rattachées aux états et les événements sont rattachés aux transitions. Les états et les transitions permettent de regrouper les entités et les événements sous forme d'unités temporelles, ils n'ont pas de signification causale en eux-mêmes.

Cette structuration de la trace va permettre d'identifier des épisodes consistant à passer d'un état initial à un état final en passant par différents états intermédiaires.

1.2.3 Signatures de tâches et épisodes

On appelle épisode d'utilisation toute partie de la trace qui correspond à une expérience d'exécution d'une tâche spécifique et qui peut être réutilisée dans une situation similaire.

Les épisodes sont localisés dans la trace primitive grâce à :

- des motifs dans le graphe formé par les objets d'intérêt ;
- des contraintes sur la disposition de ces objets d'intérêt dans la trace (co-occurrence, distance et position par rapport aux états et transitions) ;
- des contraintes liées à la définition des objets d'intérêt (valeurs des attributs, contraintes explicites).

Une fois que ces caractéristiques ont été identifiées, elles peuvent être considérées comme une signature de tâche. Cette signature de tâche peut être enrichie d'annotations en langage humain ou d'indications précisant son exploitation, elle constitue alors une signature de tâche expliquée ou Extasi.

Les épisodes ne sont pas limités à la partie permettant de reconnaître la signature de tâche. Une fois la tâche identifiée on peut affiner l'interprétation et mieux comprendre le rôle des différents objets d'intérêt. L'épisode est enrichi par les explications apportées par l'Extasi.

1.3 Le LESCOT

Le LESCOT (Laboratoire d'Ergonomie et Sciences Cognitives pour les Transports) est une unité de recherche de l'INRETS (Institut National de Recherche sur les Transports et leur Sécurité).

L'INRETS est un EPST (Établissement Public à caractère Scientifique et Technique) de 550 personnes qui a pour mission la réalisation et la valorisation de recherches sur les transports, au plan technique, économique ou social.

Les missions du LESCOT visent plus particulièrement à l'amélioration de la qualité d'usage des transports pour les professionnels ou les particuliers.

Elles se déclinent selon deux grandes thématiques de recherche :

- Systèmes embarqués et sécurité de conduite ;
- Situation de handicap, vieillissement et mobilité.

Le LESCOT regroupe une vingtaine de personnes sur le site de l'INRETS de Bron ; il dispose de véhicules instrumentés et d'un simulateur pour l'étude de la conduite automobile.

Pour disposer d'une base théorique permettant de mieux répondre à ses objectifs, le LESCOT a développé un modèle cognitif informatisé du conducteur automobile : COSMODRIVE [10] (COgnitive Simulation MOdel of the DRIvEr).

2 Représentation des connaissances

2.1 RDF

RDF [2] (Resource Description Framework) est un langage pour représenter des informations sur des ressources et pour permettre un traitement automatique de celles-ci. Une des syntaxes de ce langage est RDF/XML. Il s'agit d'un dialecte XML développé par le consortium W3C.

RDF est basé sur l'idée d'identifier les concepts et les objets par des URIs (Uniform Resource Identifiers), et de décrire les ressources en termes de propriétés et de valeurs de propriétés. Cela permet de représenter un document comme un graphe orienté étiqueté.

Un document structuré en RDF est un ensemble de triplets. Un triplet RDF est une association :

$$(Sujet, Predicat, Objet)$$

La figure 4 fournit un exemple d'un graphe RDF décrivant les propriétés d'une ressource nommée *Collect_Object_1*. Cette ressource possède 4 propriétés : *type*, *next_co*, *date*, *steering* qui ont respectivement pour valeur les ressources *Left_Turn*, *Collect_Object_2* et les littéraux 24 et 30.

Le même graphe peut aussi être écrit selon la notation triplets comme un ensemble de 4 déclarations :

```
<Collect_Object_1> <type>     <Left_Turn>
<Collect_Object_1> <next_co>  <Collect_Object_2>
<Collect_Object_1> <date>     "24"
<Collect_Object_1> <steering> "30"
```

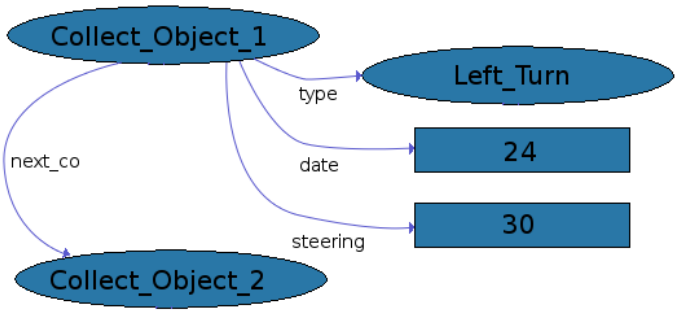

FIG. 4 – Un graphe RDF décrivant la ressource Collect_Object_1

2.2 RDFS

RDF fournit un langage pour exprimer des déclarations à propos de ressources, en utilisant des propriétés et des valeurs. Cependant, il nous faut aussi définir le vocabulaire (l'ensemble des termes) que nous souhaitons utiliser dans ces déclarations, le structurer en une hiérarchie de classes et spécifier quelles propriétés s'appliquent à quelles classes.

Par exemple, une personne désireuse de décrire l'activité de conduite pourrait définir des termes tels que *virage à droite*, ou encore *angle du volant*.

RDFS [3] (RDF Vocabulary Description Language : RDF Schema) est un langage permettant de décrire de telles *classes* et *propriétés*, et de spécifier quelles classes et propriétés doivent être utilisées ensemble.

L'ensemble de déclaration suivant définit deux classes : *Eye_Glance* et *Mirror_Glance*, ainsi qu'une classe *Eye_Center_Mirror_Glance* dérivée des deux précédentes par héritage multiple, et enfin une propriété *duration* pouvant s'appliquer aux ressources de type *Eye_Center_Mirror_Glance*.

```
<Eye_Glance>                <type>       <Class>
<Mirror_Glance>             <type>       <Class>
<Eye_Center_Mirror_Glance> <type>       <Class>
<Eye_Center_Mirror_Glance> <subClassOf> <Eye_Glance>
<Eye_Center_Mirror_Glance> <subClassOf> <Mirror_Glance>
<duration>                  <type>       <Property>
<duration>                  <domain>     <Eye_Center_Mirror_Glance>
```

On constate qu'RDFS est essentiellement un système de typage. Il existe des langages plus expressifs pour représenter des ontologies, tels que OWL [4] (Web Ontology Language).

2.3 Protégé

Le logiciel Protégé [1] est un editeur d'ontologies et un cadre de développement pour l'ingénierie des connaissances. Nous nous servirons de Protégé de deux façons :

- définition d'un *"Modèle de Collecte"* (*cf.* figure 5). Ce modèle de collecte est un vocabulaire permettant de décrire la représentation descriptive de l'activité de conduite, appelée *"Trace Brute"*.
- cadre de développement d'une application, appelée MTAB (Musette Tab), permettant de construire et visualiser cette trace brute.

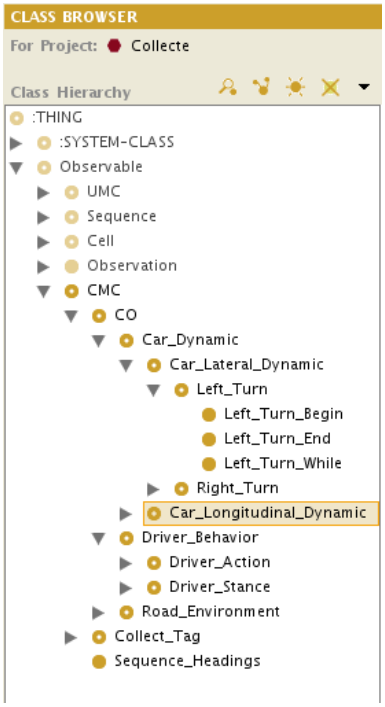

FIG. 5 – Définition de l'ontologie dans Protégé.

2.4 Modèle de Collecte

Il apparaît utile de définir un standard pour la trace brute. Pour cela, on peut la définir comme une simple succession chronologie d'objets de collecte, enrichie par des descripteurs symboliques.

Cette trace brute est conforme à un modèle de collecte (ontologie des concepts et relations présents dans la trace brute). Elle est lisible par un analyste qui construit la trace primitive.

Le modèle de collecte est défini dans le logiciel Protégé par la classe *CMC* (Collect Model Class) et ses sous-classes.

2.4.1 Objet de Collecte

Les objets de collecte sont des données provenant du dépouillement vidéo, et d'un traitement numérique des signaux des capteurs : détections de seuils, points d'inflexion, entropie...

Ils sont définis comme un type abstrait *CO* dérivant de la classe *CMC*, et se répartissent en trois catégories :

- comportements du conducteur : mouvements, stratégies visuelles, points remarquables des courbes d'action sur les commandes...

– dynamique du véhicule : points remarquables des courbes de vitesse et de volant...
– description de la situation : objets de l'environnement

On appelera *"Données Sources"* un graphe ne contenant que des objets de collecte, et conforme au modèle de collecte.

2.4.2 Marqueur

Un *marqueur* est un descripteur symbolique défini dans le modèle de collecte, et inféré dans la trace brute par application de règles formelles. Les marqueurs sont des instances du type *Collect_Tag*, lui-même dérivé de la classe *CMC*.

On appelera *"Trace Brute"* un graphe contenant des objets de collecte et des marqueurs, et conforme au modèle de collecte. Une trace brute est produite à partir d'une donnée source par un processus itératif d'ajout de marqueurs.

3 Production des Données Sources

Les données collectées de l'expérimentation de conduite sont fournies en format CSV (Comma Separated Values). Il est donc nécessaire d'effectuer un pré-traitement afin d'exprimer ces données sources dans le langage RDF et de les mettre en conformité avec le modèle de collecte. Ce traitement est schématisé figure 6 et détaillée dans les paragraphes suivants.

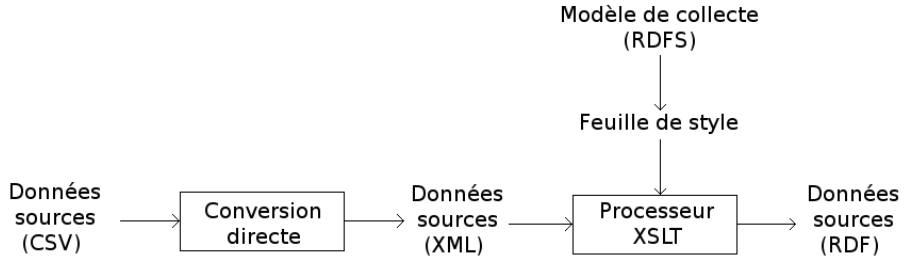

FIG. 6 – Production des données sources RDF.

3.1 Production des données sources XML

Les données sources sont fournies sous forme CSV, comme une séquence d'objets de collecte ayant chacun les propriétés *date*, *distance parcourue*, *vitesse*, *angle du volant*, *type*, ainsi que des paramètres facultatifs.

Voici un exemple de données sources CSV (seuls les 3 premiers objets de collecte sont décrits) :

```
Temps,Dist,Vit,Vol,Type,Par1,Val1,Par2,Val2,Par3,Val3,Par4,Val4
 0.04, 0.39,   35,    0,Min_Deceleration,Decel,-0.21,
 1.00, 9.60,   34,    8,R_Tete_Gauche,Duree, 0.36,Long, 3.78,
 1.72,16.38,   34,  -26,Debut_Accelerateur,Derivee,16.00,
```

Un premier traitement est effectué pour traduire ces données du format CSV en arbre XML, plus facilement exploitable par des outils génériques. La traduction est neutre et est effectuée par un programme nommé *CSV2XML*.

Voici un exemple d'exécution :

```
java CSV2XML donnees.csv donnees.xml
```

3.2 Production des données sources RDF/XML

La seconde étape est de produire des données sources au format RDF, et plus précisement conformes à sa syntaxe RDF/XML.

Ce traitement est effectué, après définition du modèle de collecte, par une transformation XSLT. La feuille de style XSLT, nommée *collecte.xsl* prend en compte le modèle de collecte nommé *collecte.rdfs*.

Ce modèle de collecte comprend la correspondance entre la valeur de la colonne *"Type"* du document CSV et le nom de la classe dans le modèle de collecte. Le graphe RDF produit est ainsi conforme au modèle de collecte.

Voici un exemple d'exécution :

```
java -jar saxon8.jar donnees.xml collecte.xsl > donnees.rdf
```

4 Production de la Trace Brute

La trace brute est produite à partir des données sources par un processus itératif d'ajout de *"marqueurs"* (cf. figure 7). Ces marqueurs, définis dans le modèle de collecte comme des spécialisations de la classe *Collect_Tag*, décrivent l'activité de conduite à un niveau plus abstrait.

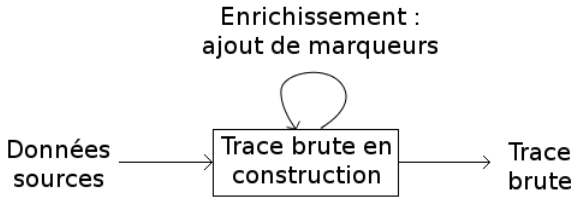

FIG. 7 – Cycle de production de la Trace Brute.

4.1 Jena

Jena [8] est un cadre Java pour construire des applications du Web Sémantique. Il fournit un environnement de programmation pour RDF, RDFS et OWL, et inclut un moteur d'inférence à base de règles.

Un *modèle Jena* est une structure de données permettant de représenter un graphe RDF. L'application MTAB contient 4 modèles Jena (cf. figure 8) :

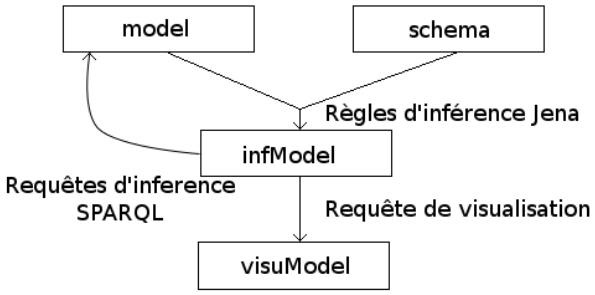

FIG. 8 – Structures de données Jena de l'application MTAB.

- *model* correspond à la trace brute en cours de construction ;
- *schema* correspond au modèle de collecte, qui est lui-même un graphe RDF ;
- *infModel* est un graphe inféré à partir de *model* et de *schema*, contenant des propriétés supplémentaires telles que la fermeture transitive de la relation *"sorte de"*. Les requêtes d'inférences et de visualisation sont effectuées sur ce modèle ;

– *visuModel* correspond à l'affichage à l'écran de la trace brute.

4.2 Inférence de types

Le langage RDF vient avec un ensemble de règles formelles décrites dans le document *RDF Semantics* [11].

Ces règles nous permettent notamment d'exprimer la transitivité de la relation *"sorte de"* (*rdfs :subClassOf*). Afin d'appliquer cette transitivité, nous utilisons le moteur d'inférence de Jena avec les règles de production suivantes :

```
rdfs8:  (?a rdfs:subClassOf ?b), (?b rdfs:subClassOf ?c)
           -> (?a rdfs:subClassOf ?c)
rdfs9:  (?x rdfs:subClassOf ?y) ->
           [ (?a rdf:type ?y) <- (?a rdf:type ?x)]
```

Ces règles, appliquées aux modèles Jena *model* et *schema*, produisent un troisième modèle *infModel* qui résulte de l'union des deux premiers modèles ainsi que du calcul de la fermeture transitive de la relation *rdfs :subClassOf*.

Enfin, une dernière règle nous permet d'identifier plus aisément une relation directe :

```
r1: (?r rdf:type ?t) -> (?r jrv:directType ?t)
```

4.3 SPARQL

SPARQL [9] est un langage d'accès aux documents RDF, en cours de standardisation par le DAWG (RDF Data Access Working Group) du W3C.

Il fonctionne sur un principe de recherche exacte de motifs dans un graphe. Le résultat d'une recherche peut être utilisé pour générer de nouveaux triplets RDF.

4.4 Requêtes d'inférence

Les règles formelles permettant de générer des marqueurs dans la trace brute sont exprimées dans le langage SPARQL. Une requête SPARQL est effectuée sur le modèle Jena *infModel* et permet donc d'exprimer des conditions tenant compte de la transitivité de la relation *"sorte de"*. Elle génère éventuellement de nouveaux triplets, qui sont alors ajoutés au modèle Jena *model* (cf. figure 8). Ces triplets sont désignés dans la requête SPARQL par des noeuds blancs.

La figure 9 illustre une requête d'inférence. Cette requête sélectionne deux objets de collecte successifs de type *Car_Speed* dont l'écart de vitesse est inférieur à un seuil fixé. Si le résultat de la recherche est positif,

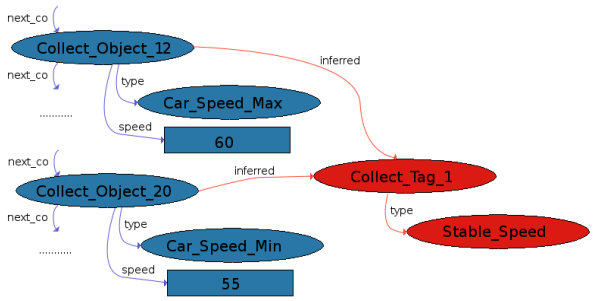

FIG. 9 – Inference d'une nouvelle étiquette.

un nouveau marqueur est généré dans la trace brute. Celui-ci est de type *Stable_Speed* , et est lié aux deux objets de collecte selectionnés par la relation *inferred*.

Cette requête est exprimée dans le langage SPARQL comme suit :

```
PREFIX xs: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX kb: <http://protege.stanford.edu/kb#>

CONSTRUCT {
        ?r1 kb:inferred _:a .
        ?r3 kb:inferred _:a .
        _:a a kb:Stable_Speed }
WHERE {
        ?r1 a kb:Car_Speed ;
            kb:date ?d1 ;
            kb:speed ?v1 .
        ?r3 a kb:Car_Speed ;
            kb:date ?d3 ;
            kb:speed ?v3 .
        FILTER xs:double(?d1) < xs:double(?d3) .
        FILTER xs:double(?v3) - xs:double(?v1) < 4 .
        FILTER xs:double(?v1) - xs:double(?v3) < 4 .
        OPTIONAL {
                ?r2 a kb:Car_Speed ;
                    kb:date ?d2 .
                FILTER xs:double(?d1) < xs:double(?d2) .
                FILTER xs:double(?d2) < xs:double(?d3)
        } .
        FILTER !bound(?r2)
}
```

Les deux objets de collecte sont désignés par les variables ?r1 et ?r3. La requête fait usage de la négation par l'échec afin de s'assurer que les deux objets se suivent avec aucun objet intermédiaire de type *Car_Speed*. Nous faisons l'hypothèse d'un éventuel objet ?r2 situé chronologiquement entre ?r1 et ?r3. Si ?r2 existe, alors la solution est rejetée.

5 Visualisation de la trace brute

L'application MTAB fournit une visualisation graphique de la trace brute. Celle-ci est paramétrable par l'utilisateur, grace à l'emploi de requêtes SPARQL permettant de sélectionner les types d'objets et les propriétés voulues.

5.1 Requêtes de visualisation

Les requêtes de visualisation, de manière analogue aux requêtes d'inférence, s'expriment dans le langage SPARQL et s'effectuent sur le modèle Jena *infModel*, et permettent donc de sélectionner les objets de collecte et les marqueurs en fonction de leur types abstraits.

La propriété *libelle* du modèle Jena *infModel* permet de spécifier le contenu textuel de l'ellipse.

Voici une requête de visualisation permettant de sélectionner toutes les instances de CMC, toutes leurs propriétés excepté les types, et de leur affecter leur type direct comme libellé :

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX kb: <http://protege.stanford.edu/kb#>
PREFIX jrv: <http://jena.hpl.hp.com/2003/JenaReasoner#>

CONSTRUCT {
        ?r ?p ?o .
        ?rlib jrv:libelle ?libelle
}
WHERE {
        ?r rdf:type kb:CMC ;
                ?p ?o .
        FILTER !(?p = rdf:type) .
        FILTER !(?p = jrv:directType) .

        ?rlib rdf:type kb:CMC ;
                jrv:directType ?libelle
}
```

5.2 Visualisation intéractive

La visualisation se fait grâce à la bibliothèque ZVTM [7]. Celle-ci permet de représenter des graphes complexes et de naviguer suivant la métaphore de la caméra (translation, zoom).

L'application fait également appel au logiciel Graphviz [6] afin d'effectuer le rendu du graphe et le placement des formes géométriques.

6 Conclusion

L'objectif de ce stage était de réaliser un atelier pour l'étude de la trace. Cette étude s'inscrivait dans le contexte du modèle Musette.

Un travail a été réalisé sur la représentation, la production et la visualisation de la trace brute. Cependant, la même infrastructure pourrait être utilisée pour la production de la trace primitive, puis l'identification des schémas de conduite.

Concernant l'application développée, des développements pourraient se poursuivre dans plusieurs directions.

Tout d'abord, il apparait nécessaire d'intégrer la chaîne de production des données sources RDF dans l'extension MTAB pour faciliter l'utilisation de celle-ci. En aval, l'application pourrait intégrer la production de la trace primitive et l'étude des schémas de conduite.

Deuxièmement, il est possible d'améliorer la représentation graphique de la trace brute pour plus de lisibilité, par exemple par l'association d'icones à certains types d'objets de collecte.

De plus, il faudrait tester le mécanisme d'inférence des marqueurs avec de nouvelles requêtes d'inférence afin d'en déterminer les limites. Le travail du DAWG et plus particulièrement le développement des langages de manipulation RDF sont à suivre sur ce point.

Enfin, l'utilisation de l'application MTAB devrait permettre en retour d'enrichir le modèle de collecte afin d'apporter une meilleure description de l'activité de conduite.

Références

[1] N. F. Noy, M. Sintek, S. Decker, M. Crubezy, R. W. Fergerson, M. A. Musen. Creating Semantic Web Contents with Protege-2000. IEEE Intelligent Systems 16(2) :60-71, 2001.

[2] F. Manola, E. Miller, B. McBride. RDF Primer. W3C Candidate Recommendation, 10 Février 2004. Disponible à l'adresse http ://www.w3.org/TR/rdf-primer/.

[3] D. Brickley, R.V. Guha, B. McBride. RDF Vocabulary Description Language 1.0 : RDF Schema. W3C Candidate Recommendation, 10 Février 2004. Disponible à l'adresse http ://www.w3.org/TR/rdf-schema/.

[4] M.K. Smith, C. Welty, and D.L. McGuinness. OWL Web Ontology Language Guide. W3C Candidate Recommendation, 18 Août 2003. Disponible à l'adresse http ://www.w3.org/TR/owl-guide/.

[5] P-A Champin, Y. Prié, A. Mille. Musette : a framework for Knowledge Capture From Experience. 2004.

[6] E.R. Gansner, S.C. North. An open graph visualization system and its applications to software engineering. Software - Practice and Experience, 00(S1), 1-5 1999.

[7] E. Pietriga. Environnements et langages de programmation visuels pour le traitement de documents structurés. (Ch. 5) Thèse de Doctorat, Grenoble, INPG, 15 novembre 2002 Disponible à l'adresse http ://www.inria.fr/rrrt/tu-0769.html.

[8] B. McBride. Jena : Implementing the RDF Model and Syntax Specification Semantic Web Workshop, WWW2001 Disponible à l'adresse http ://www.hpl.hp.com/personal/bwm/papers/20001221-paper/.

[9] E. Prud'hommeaux, A. Seaborne. SPARQL Query Language for RDF. W3C Working Draft, 19 avril 2005. Disponible à l'adresse http ://www.w3.org/TR/rdf-sparql-query/.

[10] T. Bellet, H. Tattegrain-Veste. COSMODRIVE : un modèle de simulation cognitive du conducteur automobile. In spérandio, J.C., Wolf, M. Formalismes de modélisation pour l'analyse du travail et l'ergonomie. P77-110. 2003

[11] P. Hayes, B. McBride. RDF Semantics. W3C Candidate Recommendation, 10 Février 2004. Disponible à l'adresse http ://www.w3.org/TR/rdf-mt/.

A Manuel utilisateur de l'extension MTAB

Le texte suivant décrit la procédure l'installation de l'extension MTAB pour Protégé, ainsi que ses instructions d'utilisation.

A.1 Installation

MTAB est une extension (plug-in) pour Protégé 3.0 permettant de construire et de visualiser une trace brute.

Il est fourni sous forme d'une archive *mtab-1.0-binary.zip* dont le contenu est à décompresser et à copier dans le répertoire *plugins* de Protégé. Après installation, le répertoire *plugins* de Protégé doit contenir un sous-répertoire nommé *fr.inrets.musette*, contenant lui-même les fichiers .jar de l'application.

L'extension MTAB nécessite également l'installation du logiciel externe Graphviz. Celui-ci peut être téléchargé à l'adresse *http ://www.graphviz.org/*.

A.2 Utilisation

A.2.1 Activation

Après le lancement de Protégé et le chargement d'un projet, l'extension MTAB doit être activée à partir du panneau de configuration accessible par le menu *Project -> Configure -> Tab Widgets*, puis en cochant la case *MTab*.

A.2.2 Configuration

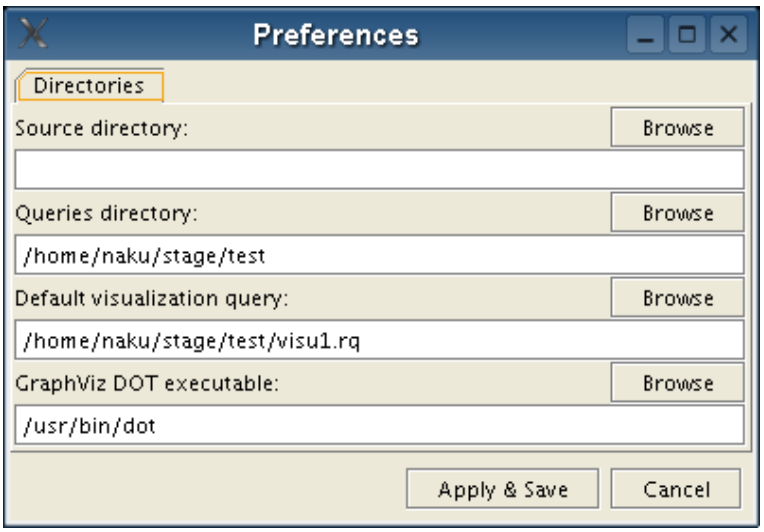

FIG. 10 – Configuration de l'extension MTAB

Le panneau de configuration de l'extension MTAB, représenté sur la figure 10, comporte 4 boîtes de saisie :

- *Source directory* : répertoire dans lequel sont entreposées les traces sources. Ce paramètre n'est pas pris en compte dans la version actuelle de MTAB ;
- *Queries directory* : répertoire dans lequel sont entreposées les requêtes d'inférence ;
- *Default visualization query* : requête de visualisation executée par défaut. Il s'agit d'une requête sélectionnant tous les objets de collecte et tous les marqueurs ;
- *GraphViz DOT executable* : emplacement de l'executable DOT de Graphviz. Ce paramètre est nécessaire afin d'exécuter une requête de visualisation. La valeur de ce champ est généralement */usr/bin/dot* sous les systèmes UNIX et *C :\Program Files\ATT\Graphviz\bin\dot.exe* sous les systèmes Windows.

Le bouton *Apply & Save* permet de valider les choix effectués et de les sauvegarder pour une future exécution de Protégé.

A.2.3 Génération des données sources

Actuellement, il n'est pas possible de générer les données sources RDF (c'est-à-dire d'exécuter la chaîne de transformation CSV -> XML -> RDF) directement à partir de l'extension MTAB. Il est nécessaire de générer les données sources RDF extérieurement à MTAB, puis de remplacer le fichier *nom_du_projet.rdf* par les données sources RDF et enfin de redémarrer Protégé.

A.2.4 Panneau de contrôle

La partie gauche de l'onglet MTAB, représentée à la figure 11, constitue le panneau de contrôle. Celui-ci se divise en 3 parties :

- partie supérieure :
  - 1 : affichage du panneau de configuration ;
  - 2 : exporter la trace brute, avec éventuellement les nouveaux triplets générés par des requêtes d'inférence, vers Protégé ;
  - 3 : importer les instances de protégé dans MTAB et perdre les modifications éventuelles ;
- panneau "INFERENCE" :
  - 4 : sélectionner une requête d'inférence ;
  - 5 : exécuter une requête d'inférence et génére de nouveaux objets dans la trace brute ;
- panneau "VISUALIZATION" :
  - 6 : sélectionner une requête de visualisation ;

FIG. 11 – Panneau de contrôle de l'extension MTAB

– 7 : exécuter une requête de visualisation mettre à jour la vue de la trace brute.

A.2.5 Interface de visualisation et de navigation

La partie droite de l'onglet MTAB, représentée à la figure 12, constitue la visualisation de la trace brute.

Les instances d'objets du modèle de collecte sont représentées sous forme d'un ruban vertical, orienté de haut en bas suivant la relation d'ordre chronologique *next_co*. Les propriétés autres que *next_co* et *inferred* sont regroupées pour chaque objet de collecte dans un cadre à droite de celui-ci, avec leurs valeurs associées.

Les objets de collectes sont représentés dans des ellipses dont la couleur de fond dépend du type de l'objet :

– jaune : instance de la classe *Car_Dynamic* ;
– bleu : instance de la classe *Driver_Behavior* ;
– rouge : instance de la classe *Collect_Tag*.

La figure 13 représente les commandes de navigation dans la vue de la trace brute. Voici leur description, de gauche à droite :

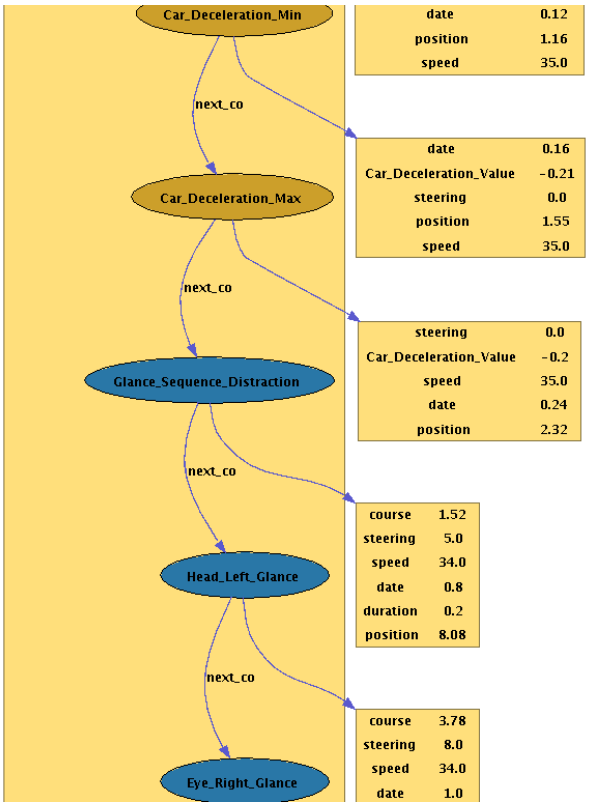

FIG. 12 – Visualisation de la trace brute

- les flèches permettent d'effectuer une translation dans la vue ;
- au centre, un bouton permet d'avoir une vue globale de la trace brute ;
- les deux loupes permettent d'effectuer un zoom arrière et un zoom avant.

Il existe des raccourcis clavier afin de se déplacer dans la vue de la trace brute :

- les flèches du pavé numérique permettent d'effectuer une translation ;
- les touches "PAGE UP" et "PAGE DOWN" effectuent respectivement un zoom arrière et un zoom avant ;
- la touche "HOME" permet d'avoir une vue globale de la trace brute.

FIG. 13 – Commandes de navigations de l'extension MTAB