

GRAPPLE

D6.3a Version: 1.0

Service-oriented modules to resolve conflicting user models

Document Type	Deliverable
Editor(s):	Dominikus Heckmann
Author(s):	Dominikus Heckmann (DFKI) Rafel Math (DFKI), Erwin Leonardi (TUD), Geert-Jan Houben (TUD), Fabian Abel (I3s), Eelco Herder (I3s)
Reviewer(s):	Alexander Nussbaumer (UniGraz), Gordon Brown (TCD)
Work Package:	WP6
Due Date:	31-01-2010
Version:	1.0
Version Date:	14-02-2010
Total number of pages:	24

Abstract: An important task that we have to solve in this work package is the problem of conflicting statements in distributed partial user models. We take a semantic conflict resolution approach to solve semantic conflicts, as well as a meta-data conflict resolution approach. We present an initial design of modules to resolve conflicting user models, consisting of conflict detection strategies and conflict resolution strategies. We base this design on related research areas like classical conflict resolution systems, semantic web reasoning, as well as a discussion on issues about integrating conflict resolution strategies into GUMF.

Keyword list: User Modelling, Conflict Resolution, Integration of partial models

Summary

The problem of conflicting GrappleStatements, given by different LMS in distributed partial models is addressed. Furthermore, the quality of the user model data can vary and make decisions based on user models especially difficult. The task of this deliverable is to design the use of the user models quality ratings from deliverable D6.2) as well as semantic relations between different GrappleStatements to support the conflict resolution, and thus to improve the final learner models.

This deliverable is divided into several chapters: an introduction and a related work chapter, a chapter about the design of the conflict resolution service, a chapter about the new GRAPPLE derivation rule format and finally one about the integration of conflict resolution strategies into GUMF.

Authors

Person	Email	Partner code
Dominikus Heckmann	heckmann@dfki.de	DFKI
Rafael Math	Rafael.Math@dfki.de	DFKI
Erwin Leonardi	e.leonardi@tudelft.nl	TUD
Geert-Jan Houben	g.j.p.m.houben@tudelft.nl	TUD
Fabian Abel	abel@l3s.de	L3S
Eelco Herder	herder@l3s.de	L3S

Table of Contents

SUMMARY	2
AUTHORS	2
TABLE OF CONTENTS	2
TABLES AND FIGURES	3
LIST OF ACRONYMS AND ABBREVIATIONS	3
1 INTRODUCTION	5
2 RELATED WORK	5
2.1 Conflict Resolution in OPS 5	5
2.2 User Model Integration & Ubiquitous User Modelling	6
2.3 Avoiding conflicts by enhancing the Semantics	6
2.4 Rules on the Semantic Web	8
2.4.1 Rule Markup Language	8
2.4.2 Semantic Web Rule Language.....	9
2.4.3 Rule Interchange Format.....	10
2.4.4 REVERSE Rule Markup Language	10
3 DESIGN OF THE CONFLICT RESOLUTION SERVICE	11
3.1 Possible Conflict (Resolution) Categories	11

3.2 Example Scenarios with Conflicting Statements 12

 3.2.1 Scenario A) 12

 3.2.2 Scenario B) 13

3.3 Conflict Resolvers and Conflict Resolution Strategies 13

3.4 Architecture of Smart GrappleStatement Retrieval (→relate to D6.1b)..... 15

4 GRAPPLE DERIVATION RULES FOR CONFLICT RESOLUTION..... 16

 4.1 Format 16

 4.2 Example 17

 4.3 GDR and Conflict Resolution Service 21

5 INTEGRATION OF CONFLICT RESOLUTION INTO GUMF 22

6 CONCLUSION AND OUTLOOK 23

REFERENCES 23

Tables and Figures

List of Figures

Figure 1 - Precision of mapping keywords (tags and categories) to DBpedia URIs. 8

Figure 2 Example visualization of three conflicting GrappleStatements 12

Figure 3 Example statements that form a semantic conflict..... 13

Figure 4: Example of Conflict Resolution Strategies..... 15

Figure 5 Smart GrappleStatement Conflict Resolution Architecture..... 16

Figure 6 Applications A1 and A2 Dataspaces..... 19

Figure 7 - A GDR Rule (Rule 1)..... 19

Figure 8 - A GDR Rule (Rule 2)..... 20

Figure 9 - Architecture of the Grapple User Modeling Framework (GUMF) with Conflict Resolution 22

List of Acronyms and Abbreviations

BLD	Basic Logic Dialect (RIF)
GRAPPLE	Generic Responsive Adaptive Personalized Learning Environment
GALE	GRAPPLE Adaptive Learning Environment
GCC	GRAPPLE Conversion Component
GDR	GRAPPLE Derivation Rule
GrappleBroker	A service to store and retrieve partial user models represented as GrappleStatements
GrappleStatement	A Unit of information about a user together with contextual meta data.
GUMF	GRAPPLE User Model Framework
IMS-LIP	IMS Learner Information Package Specification
LMS	Learning Management System
OPS5	Official Production System 5

OWL	Web Ontology Language
PRD	Production Rules Dialect
R2ML	REVERSE Rule Markup Language
RDF	Resource Description Framework
RIF	Rule Interchange Format
SWRL	Semantic Web Rule Language
TEL	Technology-Enhanced Learning
UM	User Model (or sometimes "User Modelling") a set of GrappleStatements
UMI	User Model Integration
WP	Work Package

1 Introduction

In a distributed user modelling scenario the problem of conflicting statements in distributed partial user models has to be addressed. Furthermore, the quality of the user model data can vary and make decisions based on user models especially difficult. The aim of this deliverable is to use the user models quality ratings from deliverable D6.2) as well as semantic relations between different statements to support the conflict resolution and to improve the final user models.

The deliverable D6.3 consists of a design (D6.3a) and implementation (D6.3b) of extensions to the D6.1 and D6.2 services, to handle conflicting statements in partial user models. The implementation will follow in deliverable (D6.3b).

User Model Integration (UMI) is a reasonably new research area and in the first workshop about that topic¹, Peter Brusilovsky stated *"The integration of adaptive educational systems is changing from an interesting research problem into an important practical task. One of the major challenges that need to be accepted on the way is the development of mechanisms for student model integration."*, see [15].

In Grapple we believe that user/student model integration can be successfully realized with the notion of "conflict resolution" that played an important role in artificial intelligence some decades ago. Interestingly in recent years this issue has been revitalized within the "Semantic Web Reasoning" approach. Unfortunately there is still no "silver bullet" approach to integrate conflict resolution reasoning into the semantic web.

This deliverable is a design deliverable. It is organized into a chapter about related work as well as a chapter about the conflict resolution service in mind. As well as a chapter about Grapple Derivation Rules for conflict resolution and a chapter about the integration of conflict resolution into GUMF.

2 Related Work

Conflict resolution is a range of methods for alleviating or eliminating sources of conflict (according to [http://en.wikipedia.org/wiki/Conflict_resolution], retrieved 11.01.2010). Processes of conflict resolution in real life generally include negotiation, mediation and diplomacy.

In computer science, and specifically the branches of knowledge engineering and artificial intelligence, a conflict resolution inference engine is a computer program that tries to derive answers from a knowledge base to solve the conflict. It is the intelligence used to reason about the information in the knowledge base for the ultimate purpose of formulating new conclusions. Inference engines are considered to be a special case of reasoning engines, which can use more general methods of reasoning.

We start our discussion with a history of conflict resolution in computer science.

2.1 Conflict Resolution in OPS 5

In the historical OPS5 system (see [<http://en.wikipedia.org/wiki/OPS5>] or [13] for instance) a choice of two conflict resolution strategies is presented to the programmer. The LEX strategy orders instantiations on the basis of recency of the time tags attached to their data items. Instantiations with data items having recently matched rules in previous cycles are considered with higher priority. Within this ordering, instantiations are further sorted on the complexity of the conditions in the rule. The other strategy, MEA, puts special emphasis on the recency of working memory elements that match the first condition of the rule. (The latter heuristic is heavily used in means-ends analysis.)

OPS5 is a rule-based or production system computer language, regarded as the first such language to be used in a successful expert system.

The OPS (which stands for Official Production System) family was developed in the late 1970s. Since it is based on the efficient Rete algorithm it was possible to scale up to larger problems involving hundreds or thousands of rules. OPS5 uses a forward chaining inference engine; programs execute by scanning "working memory elements" with classes and attributes looking for matches with the rules in "production memory". Rules have actions that may modify or remove the matched element, create new ones, perform side effects

¹ User Model Integration Workshop: <http://www.ah2008.org/index.php?section=42>

such as output, and so forth. Execution continues until no more matches can be found. (see <http://en.wikipedia.org/wiki/OPS5>)

For GRAPPLE it is interesting to analyze the conflict resolution strategies. For example the recency-filter could be used in analogy. This means the more recent the information is, the higher the probability it is more beneficial than older information

2.2 User Model Integration & Ubiquitous User Modelling

A key challenge for GRAPPLE's distributed approach to handle user- and learner-models is user model integration. Different systems can represent the same information in different ways, using different syntactic and conceptual structures; therefore a mean of negotiation and clarification of data among systems is required. This is fundamental not only to access information, but also to reuse information. In the GRAPPLE project, we have introduced a harmonized syntactical structure of GrappleStatements (see WP6.1 and WP2.1) . However, this is only the first step towards solving the issue of user model integration.

In a recent workshop about Ubiquitous User Modelling², participants have shown an increased awareness of the need for standards for representing and exchanging user profile data. At the same time we observe that dealing with syntactic and semantic heterogeneity of user models is complicated, especially in an open environment like adaptive hypermedia, adaptive web-based systems and adaptive learning environments. The issue is; how can semantic heterogeneity be handled for ubiquitous user modelling? How can the Semantic Web technologies be employed to cope with such heterogeneity?

These issues are also important to resolve conflicting user models in GRAPPLE, while "Ubiquitous User Modelling" describes ongoing modelling and exploitation of user behaviour with a variety of systems that share their user models. These shared user models can either be used for mutual or for individual adaptation goals. Ubiquitous user modelling differs from generic user modelling by the three additional concepts: *ongoing modelling*, *ongoing sharing* and *ongoing exploitation*. Systems that share their user models will improve the coverage, the level of detail, and the reliability of the integrated user models and thus allow better functions of adaptation. Ubiquitous user modelling implies new challenges of scalability, scrutability and privacy that strongly relate to the situation in GRAPPLE.

2.3 Avoiding conflicts by enhancing the Semantics

In GRAPPLE, users are modelled by means of Grapple statements (see Deliverable 2.1), which consist of *subject-predicate-object* structures enriched with metadata such as *date* of creation, *level* denoting to which degree the statements seems to be true, or the *creator* of the statement. Some conflicts may arise because the semantics of a given statement are not clearly defined. While the subject usually refers to a user and the predicate to some domain vocabulary that is shared among the involved systems and therewith semantically well defined, the object can sometimes be unclear. For example, the meaning of the object value might be ambiguous as it is for the following Grapple statement (formatted in RDF/XML).

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns="http://www.grapple-project.org/grapple-core/#"
  xmlns:gc="http://www.grapple-project.org/grapple-core/#"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xml:base="http://www.grapple-project.org/grapple-core/">

  <gc:Statement rdf:ID="example-app-mary-2010-01-10-165252631562">

    <!-- main part -->
    <gc:user rdf:resource="http://grapple-project.org/user/marry"/>
    <gc:predicate rdf:resource="http://xmlns.com/foaf/0.1/interest"/>
    <gc:object rdf:datatype="&xsd:anyURI">
      Java
    </gc:object>
    <gc:level rdf:datatype="&xsd;double">0.9</gc:level>
    <gc:origin>
```

² Workshop about Ubiquitous User Modeling, supported by GRAPPLE: <http://www.u2m.org/ubiquum2009/>

```
[user: mary, key: interest, value: Java]
</gc:origin>

<!-- meta part -->
<gc:creator rdf:resource="http://example.org/clientapplicationXY#"/>
<gc:created rdf:datatype="&xsd;dateTime">
  2010-01-10T14:20:30+01:00
</gc:created>
</gc:Statement>

</rdf:RDF>
```

The Grapple statement above states that Mary (<http://grapple-project.org/user/marry>) is interested (<http://xmlns.com/foaf/0.1/interest>) in "Java". However, it is not exactly clear whether the term Java refers to the Indonesian island or to the programming language. Another problem that can cause conflicts is given by different words (synonyms) that actually refer to the same concepts. For example, there could be the following two Grapple statements.

1. (<http://grapple-project.org/user/marry>, <http://xmlns.com/foaf/0.1/interest>, "J2EE")
2. (<http://grapple-project.org/user/marry>, <http://xmlns.com/foaf/0.1/interest>, "Java 2 Enterprise Edition")

Both statements basically say that Marry is interested in the enterprise edition of the Java programming language. However, the first statement uses the abbreviation while in the second statement the full name of the edition is applied.

Solutions to such problems that are inspired by Semantic Web paradigms try to map such ambiguous or synonymic keywords to URIs that clearly describe the meaning of the words. There exist already systems such as Faviki³ that foster the usage of URIs instead of keywords. Further, there exist approaches that automatically map keywords to meaningful URIs. For example, Marchetti et al. present an approach, which exploits WordNet⁴ and Wikipedia URLs to detect and define the meaning of words in collaborative tagging systems [27]. In [28] Passant et al. present a framework that can be applied to attach URIs to tag assignments that clearly describe the meaning of the corresponding tag.

We investigated the feasibility of mapping keywords to DBpedia URIs. DBpedia [26] can be described as the Semantic Web version of Wikipedia. DBpedia makes a huge portion of Wikipedia articles available in RDF format and enriches each article with further metadata, e.g. articles are classified using the Yago ontology [29]. In an evaluation we compared two different lightweight approaches for mapping words to DBpedia URIs.

- **DBpedia Lookup:** The naive lookup strategy invokes the DBpedia lookup service with the keyword that should be mapped to a URI as a search query. DBpedia ranks the returned URIs similarly to PageRank [30] and our naive mapping strategy simply assigns the top-ranked URI to the keyword in order to define its meaning.
- **DBpedia Lookup + Feedback:** The advanced mapping strategy is able to consider feedback while selecting an appropriate DBpedia URI. Whenever a keyword occurs, for which already a correctly validated DBpedia URI exists in the repository, then that URI is selected. Otherwise the strategy reverts back to the naive DBpedia Lookup.

We evaluated both strategies on the data corpus of the TagMe! system⁵ that enables users to annotate resources with freely chosen keywords and further enables them to categorize resources into categories that are suggested by the system but can also be extended by the end-users.

³ <http://faviki.com>

⁴ <http://wordnet.princeton.edu/>

⁵ <http://tagme.groupme.org/>

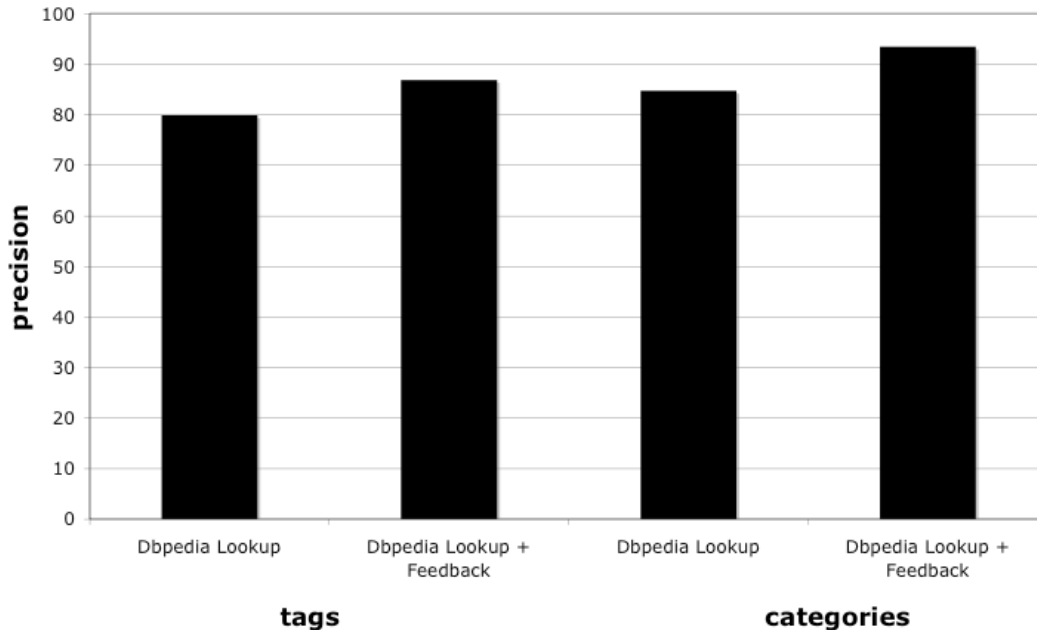


Figure 1 - Precision of mapping keywords (tags and categories) to DBpedia URIs.

Figure 1 shows the results of our evaluation [0] with respect to precision (= number of correctly mapped words / number of mapped words). The mappings of the naive approach result in a precision of 79.92% for mapping tags to DBpedia URIs and 84.94% for mapping categories considering only those keywords where a DBpedia URI that describes the meaning properly exists (e.g., keywords such as "me" or "to-read" were not fed into our algorithms). The consideration of feedback improves the precisions of the naive DBpedia Lookup clearly to 86.85% and 93.77% respectively, which corresponds to an improvement of 8.7% and 10.4%. As the mapping accuracy for categories is higher than the one for tags, it seems that the identification of meaningful URIs for categories, which are suggested by the system, is easier than for totally freely chosen tags. In summary, the results of the DBpedia mapping are very encouraging. The mapping strategies themselves can be enhanced by also considering the context of the tag/category that should be mapped. For example, for mapping a tag assignment one could select the DBpedia URI, which best fits to the DBpedia URIs of the categories that are related to the context of the tag assignment (e.g., that are already assigned to the same resource). The DBpedia mapping reduces the number of distinct tags and categories within dataset by 14.1% and 20.9% respectively, which promises a positive impact on the recall when performing keyword search. The strategies therewith successfully identify synonyms that are a potential source of conflict.

2.4 Rules on the Semantic Web

In this section, we briefly review the rule language and rule interchange language on the Semantic Web since rules will play the major role in the conflict resolution process.

2.4.1 Rule Markup Language

The Rule Markup Language (RuleML) [19] defined by the Rule Markup Initiative is a markup language developed to express both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks. The Rule Markup Initiative develops a modular RuleML specification and transformations from and to other rule standards/systems. Moreover, it coordinates the development of tools to elicit, maintain, and execute RuleML rules. RuleML itself covers the entire rule spectrum, including *derivation rules*, *transformation rules* and *reaction rules*. It can therefore specify queries and inferences in Web ontologies, mappings between Web ontologies, and dynamic Web behaviours of workflows, services, and agents [16].

2.4.2 Semantic Web Rule Language

Semantic Web Rule Language (SWRL) [16] is a proposal for a Semantic Web rules-language that is based on a combination of the OWL DL and OWL Lite sublanguages of the OWL Web Ontology Language [17,18] with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language [19].

Rules are of the form of an implication between an antecedent (body) and consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. In the human readable syntax, a rule has the form:

$$\textit{antecedent} \implies \textit{consequent}$$

where both antecedent and consequent are conjunctions of atoms written $a_1 \sqcap \dots \sqcap a_n$. Variables are indicated using the standard convention of prefixing them with a question mark (e.g., ?x). Using this syntax, a rule asserting that the composition of *hasParent* and *hasBrother* properties implies the *hasUncle* property would be written:

$$\textit{hasParent}(?x,?y) \sqcap \textit{hasBrother}(?y,?z) \implies \textit{hasUncle}(?x,?z)$$

This rule says if *x* has parent *y* and *y* has brother *z*, then *x* has uncle *z*.

The XML Concrete Syntax of SWRL is a combination of the OWL Web Ontology Language XML Presentation Syntax [20] with the RuleML XML syntax [19]. The above rule in XML concrete syntax is shown below.

```
<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="hasParent">
      <ruleml:var>x</ruleml:var>
      <ruleml:var>y</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="hasBrother">
      <ruleml:var>y</ruleml:var>
      <ruleml:var>z</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:individualPropertyAtom swrlx:property="hasUncle">
      <ruleml:var>x</ruleml:var>
      <ruleml:var>z</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_head>
</ruleml:imp>
```

A SWRL rule can also be in RDF. The RDF concrete syntax for the above rule is as follows.

```
<swrl:Variable rdf:ID="x"/>
<swrl:Variable rdf:ID="y"/>
```

```

<swrl:Variable rdf:ID="z"/>
<ruleml:Imp>
  <ruleml:body rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;hasParent"/>
      <swrl:argument1 rdf:resource="#x" />
      <swrl:argument2 rdf:resource="#y" />
    </swrl:IndividualPropertyAtom>
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;hasBrother"/>
      <swrl:argument1 rdf:resource="#y" />
      <swrl:argument2 rdf:resource="#z" />
    </swrl:IndividualPropertyAtom>
  </ruleml:body>
  <ruleml:head rdf:parseType="Collection">
    <swrl:IndividualPropertyAtom>
      <swrl:propertyPredicate rdf:resource="&eg;hasUncle"/>
      <swrl:argument1 rdf:resource="#x" />
      <swrl:argument2 rdf:resource="#z" />
    </swrl:IndividualPropertyAtom>
  </ruleml:head>
</ruleml:Imp>

```

2.4.3 Rule Interchange Format

The Rule Interchange Format (RIF) [24] is a standard in development within the W3C Semantic Web Activity. When completed, it will be a component of the semantic web, along with (principally) RDF [25] and OWL [17,18]. Although originally envisioned by many as a "rules layer" for the semantic web, in reality the design of RIF is based on the observation that there are many "rules languages" in existence, and what is needed is to exchange rules between them [24]. There are two categories of dialects in RIF: *logic-based dialects* and dialects for *rules with actions*. Generally, logic-based dialects include languages that employ some kind of logic, such as first-order logic (often restricted to Horn logic) or non-first-order logics underlying the various logic programming languages. The rules-with-actions dialects include production rule systems as well as reactive (or event-condition-action) rules. The standard RIF dialects are Core, BLD and PRD. These dialects depend on an extensive list of data types with built-in functions and predicates on those data types. The Core dialect comprises a common subset of most rule engines. The Basic Logic Dialect (BLD) adds features to the Core dialect that are not directly available such as: logic functions, equality in the then-part and named arguments. The Production Rules Dialect (PRD) adds the notion of forward-chaining rules.

2.4.4 REVERSE Rule Markup Language

REVERSE Rule Markup Language (R2ML) [21] is developed by the REVERSE Working Group I1 [22] for the purpose of rules interchange between different systems and tools. It is a comprehensive and user-friendly XML rule format that allows: interchanging rules between different systems and tools, enriching ontologies by rules, connecting your rule system with R2ML-based tools for visualization, verbalization, verification and validation. It integrates the Object Constraint Language (OCL) [23], the Semantic Web Rule Language (SWRL) [16], and the Rule Markup Language (RuleML) [19]. Integrity rules, derivation rules,

production rules and reaction rules are supported by R2ML. Integrity rules, also known as (integrity) constraints, consist of a constraint assertion, which is a sentence in a logical language such as first-order predicate logic or OCL. R2ML derivation rules have “conditions” and “conclusions”. Production rules have “conditions”, “post-conditions” and a “produced action”.

3 Design of the Conflict Resolution Service

This conflict resolution service is based heavily on work done in [4] and [9] and is applied to the new situation in GRAPPLE of distributed, generic, personalized learning environments.

In widely spread distributed user modelling and adaptive system scenarios as given in GRAPPLE it can't be expected that all systems use the same standards for representing user models and context information. However, as a common basis and to narrow down the research area, we assume that all systems that are involved in the information exchange are able to use the framework of GrappleStatements: either as their main representation language or at least as an additional transforming feature. To support this assumption, for example, the IMS-LIP Learner Information Package (as used by the LMSs to exchange user data) has recently been translated into GrappleStatements. Even though we assume that all involved user-adaptive and context aware systems are supposed to communicate with the same *grammatical* framework of GrappleStatements and GrappleQueries, there is still a number of interesting points of possible ambiguity left to be solved.

3.1 Possible Conflict (Resolution) Categories

As every LMS is allowed to enter statements into repositories, some information might be contradictory. Conflicts among GrappleStatements, such as a contradiction caused by different opinions of different creators or changed values over time, are categorized as follows (adapted from [4]):

1. REPRESENTATIONAL & SYNTACTICAL LEVEL:
Each system can choose between a variety of possible representations to express the same information which leads to the so-called *variation mapping*. The statements can for instance differ in the use of the statement attributes like subject, predicate, object, level, creator, created etc.. Clear *modelling guidelines* are necessary.
2. SEMANTICAL LEVEL:
The systems are not forced to use the same vocabulary, that is to say the same ontology, to represent the meaning of the concepts, which leads to user model integration problem number one: *ontology merging* and *semantic web integration*. (Here, we have to analyze and integrate the outcomes from deliverable D5.2, the GCC = Conversion components between GRAPPLE and LMSs)
3. OBSERVATION AND INFERENCE LEVEL:
Several sensors can see same things differently and claim to be right, measurement errors can occur, systems may have preferred information sources (Here, the rating and reputation system of deliverable D6.2 can be utilised)
4. TEMPORAL AND SPATIAL LEVEL:
Information can be out of date or out of spatial range, a degree of expiry can hold, thus reasoning on temporal and spatial meta data becomes necessary (Temporal reasoning can be built into GUMF directly since we have the creation time stored through the Dublin Core attributed “created”)
5. PRIVACY AND TRUST LEVEL:
Information can be hidden, incomplete, secret or falsified on purpose. A system of trustworthiness could be applied (Here, also the rating and reputation system of deliverable D6.2 can improve the conflict resolution)

All these aspects are content- and domain independent and can be relevant for user related information as well as context related information. In the next section several conflicts are analyzed and categorized and possible solutions are presented. In relation to the GrappleQuery retrieval task, the two concepts of *precision* (*How many retrieved statements are really relevant?*) and *recall* (*Is all relevant information retrieved?*) are also taken into consideration. This section is partly based on the research described in [12] and on ideas about meta rules and conflict resolution in OPS5, see for example [13].

3.2 Example Scenarios with Conflicting Statements

3.2.1 Scenario A)

Imagine GALE should recommend the next learning object Mary should choose in her mathematics course and GALE applies the following rule:

*if the learner has high knowledge level in mathematics, GALE start with X
but if the learner's knowledge level in low, GALE recommends Y.*

GALE'S knowledge consists of the three following, conflicting statements about Mary's knowledge in mathematics: LMS C claims that Mary is a good, LMS B claims that Mary is medium while Mary herself claims that she is a bad in mathematics.

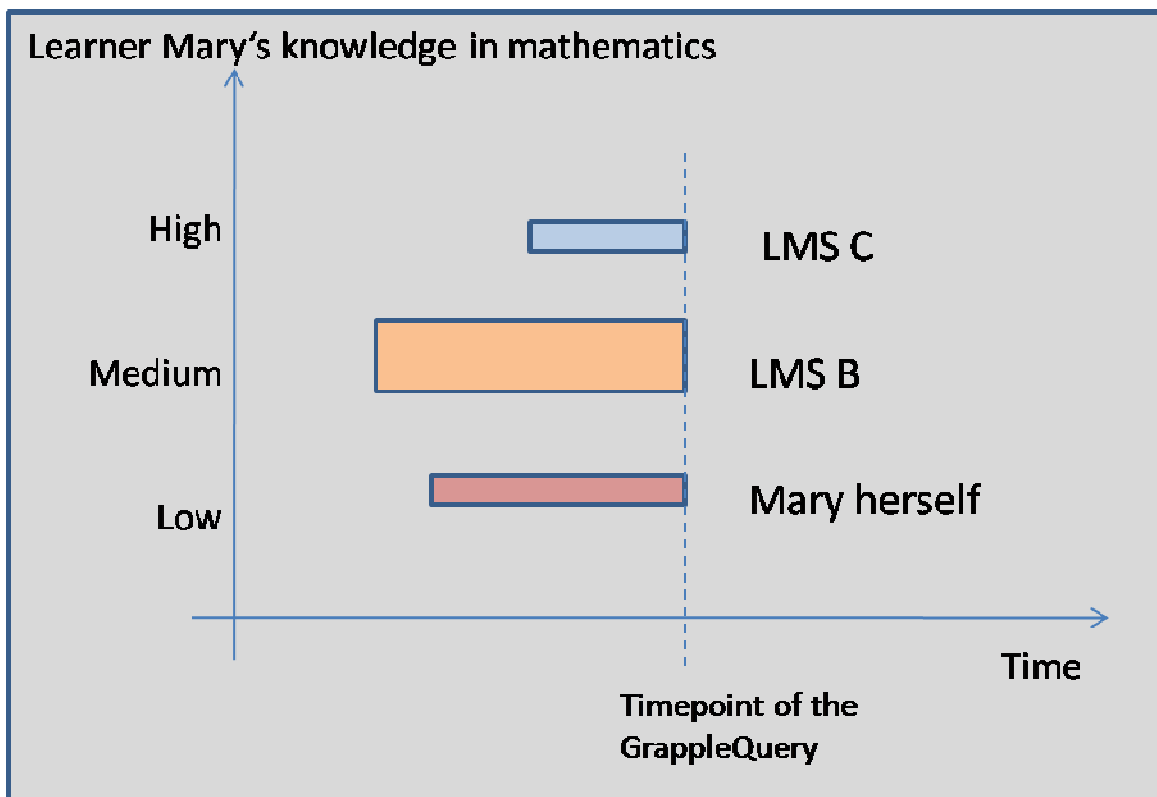


Figure 2 Example visualization of three conflicting GrappleStatements

Figure 1 visualizes these three statements as rectangles that also indicate additional meta data dimensions: the time-axis shows the created values which reveals that LMS B's statement is the oldest one, while LMS C's statement is the most recent one. The height of the rectangle relates to the confidence value: the confidence rating of LMS B's statement is the highest while the user's statement has the lowest confidence. So, how should GALE decide?

How do we resolve such conflicts? Naive approaches could ask: should we return the latest entry? Should we return a random element? Should we return the one with the highest confidence value or should we prefer the learner's own entry? Conflicting statements turned out to be a complex and abstract problem but

with no generic solution that is valid for all situations. However, in our approach we support the means to express the intended conflict resolution methods as described in section 3.3. Conflicts on the semantic level are presented in the following example.

3.2.2 Scenario B)

Let us assume that a GRAPPLE mobile learning environment, used with a mobile device, uses the following energy saving and contrast optimizing adaptation rule if its device's battery is low:

*if the surrounding brightness is low then set the display brightness to its minimum,
if the surrounding brightness is medium or high then set the display brightness to medium.*

Since the mobile device has no light sensor by itself, it needs to retrieve and infer information about the *surrounding brightness* from the user model and context service. The current position of the mobile device should be *Room124* and the light situation that is given by the semantically conflicting statements:

- 1) subject=*Room124* predicate=*brightness* range=*lowHigh* object=*low*
- 2) subject=*Saarbrücken* predicate=*weather* object=*sunny with no clouds*
- 3) subject=*Light124.1* predicate=*switched* range=*onOff* object=*on*

Figure 3 Example statements that form a semantic conflict

The most direct statement 1) about the brightness in *Room124* claims that it is dark. However, statement 2) claims that the sun is shining. Since the device is inside a building which could hinder the sun to lighten the surroundings the situation is not clear. Furthermore, statement 3) claims that the lights in this room are switched on. This conflict is difficult to detect since ontological reasoning, spatial reasoning and qualitative reasoning might be necessary to handle this problem satisfactorily. Here, only a complete knowledge base and general semantic web reasoning can help.

3.3 Conflict Resolvers and Conflict Resolution Strategies

Conflict Resolvers are special kind of filters that control the *conflict resolution process*. An ordered list of these resolvers defines the *conflict resolution strategy*. They are modelled in the *GrappleQuery.strategy* attribute. These resolvers are needed if the *match process* and *filter process* leave several conflicting statements as possible answers. Three kinds of conflict resolvers can be identified:

- The *most(n)*-resolvers that use meta data for their decision,
- The *add*-resolvers that add expired or replaced statements to the conflict sets, and
- The *return*-resolvers that don't use any data for their selection.

mostRecent(n)

Especially where sensors send new statements on a frequent basis, values tend to change quicker than they expire. This leads to conflicting non-expired statements. The *mostRecent(n)* resolver returns the *n* newest non-expired statements, where *n* is a natural number between 1 and the number of remaining statements.

mostNamed(n)

If there are many statements that claim A and only a few claim B or something else, than *n* of the "most named" statements are returned. Of course it is not certain that the majority necessarily tells the truth but it could be a reasonable rule of thumb for some cases.

mostSpecific(n)

If the range or the object of a statement is more specific than in others, the *n*-"most specific" statements are returned by this resolver. For example if: predicate=*hasKnowledge*, object=*SolarSystem* and first range=*yesNo* while the second range=*Novice-Occasional-Professional-Expert-Grandmaster*, the statement with the second range contains a more specific information. Another specificity range ordering is for example: *yesNo < lowMediumHigh < 0%-100%*

mostPersonal(n)

If the creator of the statement is the same as the statement's subject (a self-reflecting statement), this statement is preferred by the *mostPersonal(n)* resolver. Furthermore, if an *is-friend-of relation* is defined, statements by friends could be preferred to statements by others. However, this resolver bears the problem, that users might not be their best judge. However due to privacy arguments, in some situations the user's own statements that are given (on purpose) should be preferred.

addExpired

Per default the already expired statements are filtered out. However, if one wants to take them into consideration, the *addExpired*-resolver adds these statements to the conflict sets.

addReplaced

Statements that are marked with the replaced-flag by other statements are also per default filtered out and not considered in the situation retrieval process. The *addReplaced*-resolver brings these statements back into the process.

addPrivate

Statements that do not pass the privacy settings are always filtered out. However, for development, testing and administrative reasons experimental private statements may also be recognized with the *addPrivate*-resolver.

returnAll

If the remaining conflict set should not be resolved any further by the integrated mechanism, the resolver *returnAll* returns all remaining statements that can then be resolved by an external conflict resolution method, resolved by introspection or left unresolved since our approach also allows conflicting extensions in parallel.

returnNone

If still a conflict occurs that could not be resolved until the *returnNone* resolver is applied, no statement is returned. This is a very safe way not to say something wrong.⁶

returnRandom(n)

If after applying several filters still no unique value is found but a unique answer is expected, a random pick will be offered by this resolver. This credulous behaviour is selected by the requestor and therefore acceptable.

returnDialog

If no unique value is found, an alternative conflict resolution strategy could be *clarification by dialog*⁷. In some cases an appropriate human-computer dialog will be initiated in this case, (triggered by GUMF via the LMS)

These conflict resolver rules are based on common sense heuristics. An important issue to keep in mind is the problem that resolvers and strategies imply uncertainty. To contribute to this, the confidence value of the resulting statement is appropriately changed; furthermore, the conflict situation is added to the evidence attribute. Further ideas like calculating the *average* value or the *maximum* value of the statement's object can be covered by the function attribute that allows the application of mathematical functions to the value of either one statement or a set of statements. The *function evaluation* forms the last part of the *control procedure*.

Figure 4 shows several examples of conflict resolution strategies. In general, a strategy can be defined by every combination of resolvers, but not all make sense.

⁶ This rule could be compared with *sceptical inheritance* in non-monotonic reasoning: *I don't know!*

⁷ The idea of *clarification by dialog* was recommended by Vania Dimitrova.

- 1.) strategy=returnAll
- 2.) strategy=mostRecent(1)
- 3.) strategy=mostRecent(4) / returnRandom(2)
- 4.) strategy=mostNamed(5) / mostSpecific(1)
- 5.) strategy=mostPersonal(3) | function=average

Figure 4: Example of Conflict Resolution Strategies

The first conflict resolution strategy is the empty one, which means that no conflict resolution will be applied and all statements that pass the match and filter process will be returned. Note that if the query.strategy attribute is left empty, a default strategy will be applied but not necessarily the *returnAll* one. The second conflict resolution strategy is adequate for statements that are frequently renewed by only one sensor, thus simply the last entry is returned. The third strategy is slightly more complex: the four most recent statements are handed over and checked for the three most confident ones. However, only two statements of these three are returned by random selection. The fifth and last conflict resolution strategy first selects the most personal ones and then the most confident ones and finally applies the average query.function to calculate the average value of the two remaining statement's object. This strategy could be interesting for user model dimensions that do not change over time like *personality traits*.

If we revisit the prefacing example of figure 2, (Mary's knowledge in mathematics) and apply the strategies of figure 4, the second and the third strategy would return *high*, the fourth strategy would return *medium* while the fifth strategy would return *low*. However, the first strategy returns all three values and therefore is no help in this situation. These completely different results show the power and the challenge of the conflict resolution strategies. However they also indicate that not all conflicting statements can be satisfactorily solved by resolution strategies. As different strategies lead to different statements and resulting values, the choice of the "right" conflict resolution strategy isn't that obvious.

Different classes of problems need different conflict resolution strategies. The open question is if there is any correlation of the best strategy in certain circumstances. We expect that already the user model dimension equivalence classes correlate with the best strategies. The conflict resolution topics discussed so far left the semantics of the statement's main part out of consideration. However, we need to integrate both: conflict resolution with semantic analysis.

In the prefacing example the conflict was easy to detect since all three statements used the same subject, the same auxiliary, the same predicate and even the same range (*low-medium-high*). If one statement used instead the range *poor-average-good-excellent* a so-called SEMANTICRANGEMAPPING would have to be executed, where for example *poor* is mapped to *low*, *average* and *good* are mapped to *medium*, and *excellent* is mapped to *high*. A more complex problem is how can we detect conflicts even if the auxiliary and predicate in the conflicting statements (or in the query and the statements) differ even though the same ontology is used? For example, system

B could use predicate=*hasKnowledge* and object=*chess* while LMS C uses predicate=*hasAbility* and object=*boardGames*. In such a case, we already need SEMANTICPROPERTYMAPPING. See [14] for a discussion about the semantic conflict resolution ontology. A closely related problem arises if statements refer to different ontologies. In such a case we additionally need ONTOLOGYMAPPING in order to be able to detect and solve these conflicts.

For the challenge of detecting possible conflicts we use a technique which is also used in production systems: we classify statements into so-called *conflict sets*. Mathematically speaking these statements are classified into equivalence classes.

To summarize, we have demonstrated that the complex task of detecting conflicting statements in a repository can be analyzed with a fine-grained definition of conflict sets that represent syntactical as well as semantic conflicts. Ontological reasoning is mostly integrated by using semantic mapping functionality. However, further research (like the experimental analysis in GRAPPLE of the conflict detection classes) needs to be carried out at this point.

3.4 Architecture of Smart GrappleStatement Retrieval (→relate to D6.1b)

The architectural diagram in Figure 5 shows the semantic conflict resolution part of the SmartSituation retrieval process.

The numbered ovals indicate the reading direction and represents:

- (1) (no icon) the request in GrappleQuery that has to be parsed first.
- (2) (no icon) the distributed retrieval of GRAPPLESTATEMENTS.
- (3) (no icon) the input to the conflict resolution process.
- (4) the three syntactical procedures VARIATIONMAPPING, REMOVEEXPIRED and REMOVEREPLACED.
- (5) the semantic procedure SEMANTICMAPPING that is based on knowledge in ontologies (like GUMO, SUMO/MILO) and the knowledge base WordNet.
- (6) the detection of conflicts and the construction of $(S^*, A^*, P^*, R^*)_{\text{nonExpired, nonReplaced}}$ conflict sets as defined in [4].
- (7) the post-processing of ranking, format, naming and function that control the output format.
- (8) (no icon) the resulting Grapple statements that are reported to the requestor.

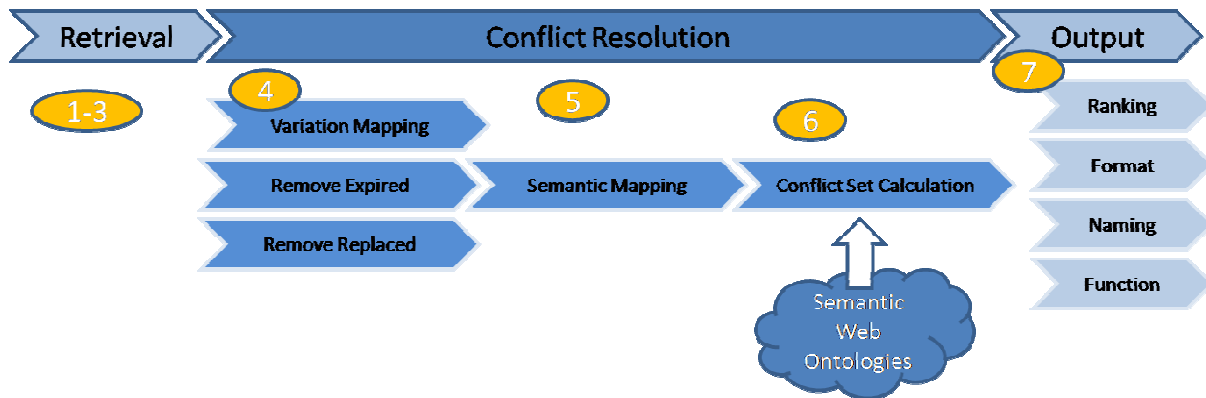


Figure 5 Smart GrappleStatement Conflict Resolution Architecture

4 Grapple Derivation Rules for Conflict Resolution

Grapple Derivation Rule (GDR) is an XML-based rule language used by Grapple User Modeling Framework (GUMF) to reason over and enrich existing Grapple statements about users. It supports reasoning over multiple GUMF dataspace and using external knowledge bases (e.g. DBpedia⁸, GeoNames⁹, DBLP¹⁰, WordNet¹¹, etc.). The external knowledge base plays an important role as it allows GUMF to reason over Grapple statements that have different level of semantics, and to translate low-level concepts in the user model into high-level ones. Note that Grapple Derivation Rule will be discussed in details in deliverables D2.3. In this chapter we briefly introduce GDRs and point out how they can be used for conflict resolution in GRAPPLE.

4.1 Format

In human readable syntax, a GDR rule has the form:

$$\text{antecedent} \implies \text{consequent}$$

where antecedent are conjunctions of premises/conditions written $p_1 \square \dots \square p_n$ and consequent is specified in term of Grapple statements. Premises are classified into two different types:

⁸ <http://dbpedia.org/>

⁹ <http://geonames.org/>

¹⁰ <http://www.informatik.uni-trier.de/~ley/db/> and <http://dblp.l3s.de/d2r/>

¹¹ <http://wordnet.princeton.edu/>

1. *Dataspace premises*, which specify conditions in term of Grapple statement for user data in GUMF dataspace,
2. *External premises*, which specify conditions in triple patterns for external knowledge bases accessible through SPARQL endpoints.

Variables are indicated using the standard convention of prefixing them with a question mark (e.g., ?x). In XML format, a GDR rule has the form:

```
<gdr:rule xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
  xmlns:gc="http://www.grapple-project.org/grapple-core/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  name="rule name"
  creator="uri of the client application which creates the rule"
  description="a short description about the rule">

  <!-- dataspace premise definition -->
  <gdr:premise dataspace="dataspace id">
    <gc:subject>subject</gc:subject>
    <gc:predicate rdf:resource="predicate" />
    <gc:object>object</gc:object>
    <gc:level>level</gc:level>
  </gdr:premise>

  <!-- external source premise definition -->
  <gdr:premise source="external source uri">
    <gdr:pattern>a triple pattern</gdr:pattern>
    <gdr:pattern>a triple pattern</gdr:pattern>
    ... ..
  </gdr:premise>

  <!-- consequent definition -->
  <gdr:consequent dataspace="dataspace id in which the rule is defined">
    <gc:subject>another subject</gc:subject>
    <gc:predicate rdf:resource="another predicate" />
    <gc:object>another object</gc:object>
    [<gc:level>another level</gc:level>]
  </gdr:consequent>
</gdr:rule>
```

4.2 Example

Suppose a Moodle-based client application A1 and Sakai-based client applications A2 are using GUMF. Application A2 subscribes to application A1 dataspace, and thus allows application A2 to query application A1 dataspace. Figure 5 depicts the dataspace of application A1 and application A2 (partial view only).

Example 1: Suppose application A2 wants to derive new Grapple statements describing *profile:origin* property of its users using data in application A1 dataspace. Unlike application A1 that maintains the origin city of the users, application A2 decides to keep the origin country of the users. For this purpose, application A2 defines a GDR rule as depicted in Figure 5.

gc:subject	gc:predicate	gc:object	gc:level
user:anna	profile:origin	"Dubai"	1.0
user:anna	moodle:hasKnowledge	subject:Geography	0.8
user:bob	profile:origin	"Delft"	1.0
user:cindy	profile:origin	"Johannesburg"	1.0
user:cindy	moodle:hasKnowledge	subject:Geography	0.6
user:donald	profile:origin	"Beijing"	1.0
user:donald	moodle:hasKnowledge	subject:Geography	0.4

(a) Grapple Statements in Application A1 Dataspace

gc:subject	gc:predicate	gc:object	gc:level
user:anna	sakai:isLearning	subject:Malaysia	0.0
user:bob	sakai:isLearning	subject:Japan	0.0
user:cindy	sakai:isLearning	subject:Delft	0.0
user:donald	sakai:isLearning	subject:Bangkok	0.0
user:frans	sakai:isLearning	subject:Japan	0.0

(b) Grapple Statements in Application A2 Dataspace

subject	predicate	object
subject:Malaysia	sakai:isRelatedTo	<http://sws.geonames.org/1733045/>
subject:Japan	sakai:isRelatedTo	<http://sws.geonames.org/993960/>
subject:Delft	sakai:isRelatedTo	<http://sws.geonames.org/2757345/>

subject:Bangkok	sakai:isRelatedTo	<http://sws.geonames.org/2757345/>
-----------------	-------------------	------------------------------------

(c) Additional Knowledge in Application A2 Dataspace, derived by another Plug-in

Figure 6 Applications A1 and A2 Dataspaces

```

01 <gdr:rule xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
02   xmlns:gc="http://www.grapple-project.org/grapple-core/"
03   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04   name="Get User Country"
05   creator="http://www.grapple-project.org/umf/client/2"
06   description='This rule derives the countries of the users">
07   <gdr:premise dataspace="2">
08     <gc:subject>?user</gc:subject>
09   </gdr:premise>
10   <gdr:premise dataspace="1">
11     <gc:subject>?user</gc:subject>
12     <gc:predicate rdf:resource="http://profile.org/origin" />
13     <gc:object>?cityName</gc:object>
14   </gdr:premise>
15   <gdr:premise source="http://geonames.org/">
16     <gdr:pattern>?city &lt;http://www.geonames.org/ontology#name&gt; ?cityName</gdr:pattern>
17     <gdr:pattern>?city &lt;http://www.geonames.org/ontology#featureClass&gt;
18       &lt;http://www.geonames.org/ontology#P&gt;</gdr:pattern>
19     <gdr:pattern>?city &lt;http://www.geonames.org/ontology#inCountry&gt; ?countryURI</gdr:pattern>
20     <gdr:pattern>?country &lt;http://www.geonames.org/ontology#featureClass&gt;
21       &lt;http://www.geonames.org/ontology#A&gt;</gdr:pattern>
22     <gdr:pattern>?country &lt;http://www.geonames.org/ontology#featureCode&gt;
23       &lt;http://www.geonames.org/ontology#A.PCL&gt;</gdr:pattern>
24     <gdr:pattern>?country &lt;http://www.geonames.org/ontology#inCountry&gt; ?countryURI</gdr:pattern>
25     <gdr:pattern>?country &lt;http://www.geonames.org/ontology#name&gt; ?countryName</gdr:pattern>
26   </gdr:premise>
27   <gdr:consequent dataspace="2">
28     <gc:subject>?user</gc:subject>
29     <gc:predicate rdf:resource="http://profile.org/origin" />
30     <gc:object>?countryName</gc:object>
31   </gdr:consequent>
32 </gdr:rule>

```

Figure 7 - A GDR Rule (Rule 1)

Lines 07-09 are used to get all the users in application A2 dataspace (id=2). The second premise (Lines 10-14) is used to get the profile:origin property of the users stored in application A1 dataspace (id=1). An external knowledge base that maintains geographical information, namely GeoNames, is used to derive the country from a city. Lines 15-23 define conditions in triple patterns that are used to retrieve the name of the country from GeoNames given the name of a city. Finally, the consequent derives new Grapple statements describing *profile:origin* property of the users for application A2.

Note that the details about how a GDR rule is processed by the Grapple Derivation Rule Engine (GDR Engine) will also be presented in deliverable D2.3.

Example 2: Suppose application A2 wants to suggest to its users Wikipedia pages about the subjects the users are currently taking (described using property *sakai:isLearning*) the users learned *subject:Geography* using application A1 and the level is more than 0.5. Rule 2 depicted in Figure 7 can be used to get the appropriate Wikipedia¹² pages for the users.

Rule 2 contains two dataspaces premises (Lines 07-12 and 13-17) and three external source premises (Lines 18-20, 21-23, and 24-26). Note that the external source premise defined in Lines 18-20 is used to query application A2 dataspaces via dataspaces SPARQL endpoint. Lines 07-12 is used to get the users who have knowledge level about Geography more than 0.5. The premise in Lines 13-17 retrieves the subjects that are currently being taken by the users. Then, using the additional knowledge in application A2 dataspaces, the subjects are related to the concepts in GeoNames (Lines 19-20). The external source premise in Lines 21-23 retrieves the DBpedia concepts that are related to GeoNames concepts using *owl:sameAs* property. The Wikipedia pages described by *foaf:page* property then can be found in DBpedia knowledge base (Lines 24-26). Finally, the consequent derives new Grapple statements describing *sakai:suggestedWikiPage* property of the users for application A2.

```

01 <gdr:rule xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
02   xmlns:gc="http://www.grapple-project.org/grapple-core/"
03   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04   name="Get Wiki Page"
05   creator="http://www.grapple-project.org/umf/client/2"
06   description="This rule retrieves Wikipedia page of subjects">
07   <gdr:premise dataspaces="1">
08     <gc:subject>?user</gc:subject>
09     <gc:predicate rdf:resource="http://moodle.org/hasKnowledge" />
10     <gc:object>http://subject.org/Geography</gc:object>
11     <gc:level>0.5</gc:level>
12   </gdr:premise>
13   <gdr:premise dataspaces="2">
14     <gc:subject>?user</gc:subject>
15     <gc:predicate rdf:resource="http://sakai.org/isLearning" />
16     <gc:object>?subject</gc:object>
17   </gdr:premise>
18   <gdr:premise source="http://www.grapple-project.org/umf/rest/sparql/ds/2?client=2&token=1234567890">
19     <gdr:pattern>?subject &lt;http://sakai.org/isRelatedTo&gt; ?geonameConcept</gdr:pattern>
20   </gdr:premise>
21   <gdr:premise source="http://geonames.org/">
22     <gdr:pattern>?geonameConcept &lt;http://www.w3.org/2002/07/owl#sameAs&gt;
23       ?dbpediaCountry</gdr:pattern>
24   </gdr:premise>
25   <gdr:premise source="http://dbpedia.org/">
26     <gdr:pattern>?dbpediaCountry &lt;http://xmlns.com/foaf/0.1/page&gt; ?page</gdr:pattern>
27   </gdr:premise>
28   <gdr:consequent dataspaces="2">
29     <gc:subject>?user</gc:subject>
30     <gc:predicate rdf:resource="http://sakai.org/suggestedWikiPage" />
31     <gc:object>?page</gc:object>
32   </gdr:consequent>
33 </gdr:rule>

```

Figure 8 - A GDR Rule (Rule 2)

¹² <http://www.wikipedia.org/>

4.3 GDR and Conflict Resolution Service

One of the important tasks of the Conflict Resolution Service is to detect conflicting statements in repository. The conflicts can be syntactical conflicts as well as semantic conflicts. The Grapple Derivation Rule (GDR) can be used to perform ontological reasoning using external knowledge bases.

Example 1: Application A has a statement about user John as follows.

```
subject=john    predicate=city    object=Delft
```

Another application B also has a statement about John:

```
subject=john    predicate=country    object=Germany
```

These statements are semantically conflicting to each other. The GDR can be used to assert, for example, Delft is a city in the Netherlands by using GeoNames. The consequences if such contradictory statements occur are not straight forward, as discussed in chapter 3.

Example 2: Application A has a statement about user Hans as follows.

```
subject=hans    predicate=city    object=Delft
```

Another application B also has a statement about Hans:

```
subject=hans    predicate=town    object=Delft
```

These statements are not semantically conflicting to each other. A GDR rule can be defined to check that city and town are synonym in WordNet.

Example 3: Application A has a statement about user Mary as follows.

```
subject=mary    predicate=likes    object=chess
```

Another application B also has a statement about John:

```
subject=mary    predicate=likes    object=boardGames
```

These statements can be semantically not conflicting. Using DBpedia and OpenCyc¹³, a GDR rule is able to determine that chess is a kind of board games (by exploring *rdf:type* property).

¹³ <http://www.opencyc.org/cyc/opencyc>

Note that the above examples have exemplified the usage of GDR in Conflict Resolution Service. However, there could be some necessary pre- and post-processing when using GDR. Furthermore, at this point, further research still has to be done (for example, to find out to what extent GDR can be used and to do the experimental analysis).

5 Integration of Conflict Resolution into GUMF

Proposed strategies for conflict resolution will be integrated into the Grapple User Modeling Framework (GUMF) that is introduced in Deliverable 6.1 a/b and depicted in

Figure 9. The different conflict resolution strategies are attached to different components of the framework. The strategies mentioned in Section 3.3 have to be integrated into the “Query Engine” module so that client applications are enabled to specify conflict resolution strategies for each individual query. By contrast, Grapple Derivation Rules (GDR), introduced in Section 4, are part of the GUMF Reasoning Logic.

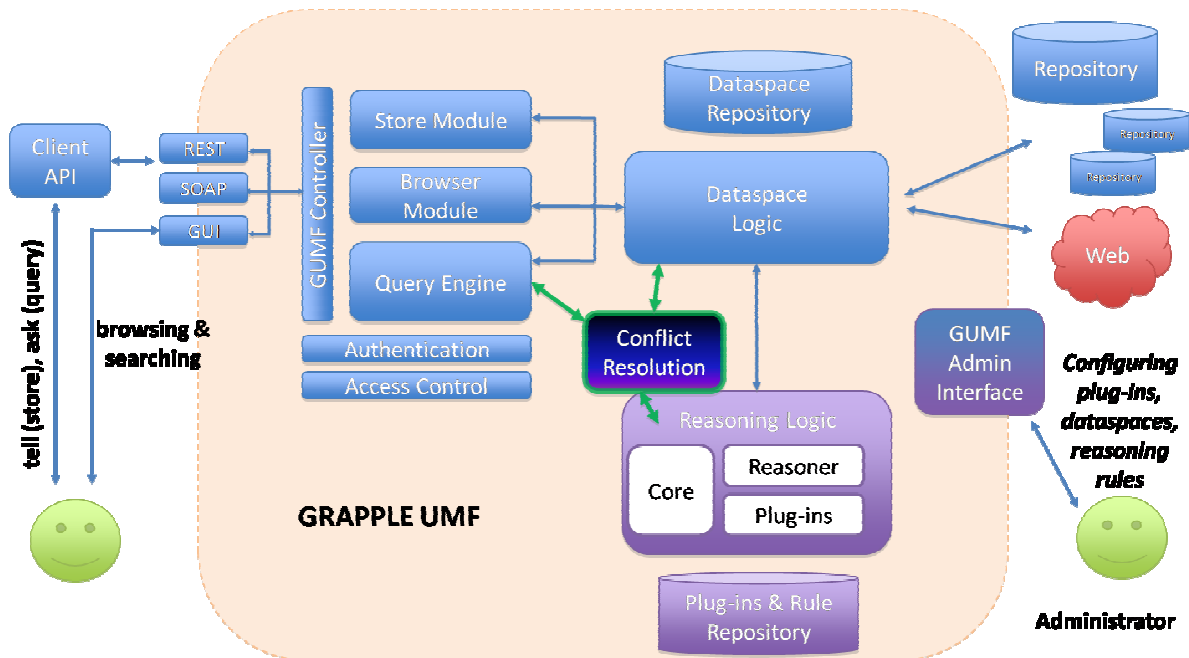


Figure 9 - Architecture of the Grapple User Modeling Framework (GUMF) with Conflict Resolution

Derivation rules can be defined for individual dataspace. Given a query that is sent to a particular dataspace, the Dataspace Logic ensures that the activated plug-ins and reasoners will be triggered. Conflict resolutions formulated in GDR will be interpreted by the corresponding GDR reasoner, which can also query external knowledge sources to deduce whether some conclusion can be drawn or not. The Grapple statements that match the query (including statements that are deduced by the reasoning logic) are finally ordered and filtered by the Query Engine according to conflict resolution strategies that are directly specified as part of the query (cf. Grapple Query, Deliverable 6.1b).

The service orientation of our approach can be seen on the left hand side in Figure 9, the Web Service interfaces REST and SOAP integrate into the GUMF controller. It handles the Query Engine module that runs as a service. New is the conflict resolution service that stands in between this query engine, the dataspace logic and the reasoning logic. All components are developed as individual modules.

6 Conclusion and Outlook

We have presented an initial design of modules to resolve conflicting user models, consisting of conflict detection strategies and conflict resolution strategies. We base this design on a chapter on related research areas like classical conflict resolution systems, semantic web reasoning, as well as a discussion on issues about integrating conflict resolution strategies into GUMF.

The implementation of the designed concepts has already been started and the first results need to be applied to the overall GRAPPLE architecture into the GUMF framework.

Implicitly, our strategy assumes that the statements are enriched with meta data like the creation time of the unit of information.

Conflicting statements turned out to be a complex and abstract problem but with no generic solution that is valid for all situations. In the upcoming b) deliverable, we have to invest time to implement and integrate the approach as a service into GRAPPLE's User Model Framework GUMF, but also to experiment with the application of different conflict resolution strategies in different situations.

The concrete next steps will be the implementation of the conflict detecting filters as well as the implementation of the conflict resolving strategies. Furthermore, the application of GRAPPLE derivation rules as main conflict resolution intelligence has to be initiated for the follow-up deliverable D6.2b.

References

1. Berners-Lee, Tim ; Hendler, James ; Lassila, Ora: The Semantic Web. In: Scientific American 284 (2001), Mai, Nr. 5, pp. 34–43.
2. Wolfgang Wahlster: Kap. 1 In: Bullinger, Hans-Jörg (Hrsg.): Trendbarometer Technik: Visionäre Produkte, Neue Werkstoffe, Fabriken der Zukunft. München, Wien : Hanser, 2004, pp. 62–63. – http://www.dfki.de/~wahlster/Publications/Semantisches_Web.pdf
3. Fensel, Dieter ; Wahlster, Wolfgang ; Lieberman, Henry; Hendler, James: Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. MIT Press, Februar 2003. – ISBN 0262062321
4. Heckmann, Dominikus: Ubiquitous User Modeling. Berlin : Akademische Verlagsgesellschaft Aka GmbH, 2006. – ISBN 3898382974
5. Recktenwald, Pascal: Ein Web 2.0 Bewertungssystem für UbisWorld, Bachelor Thesis, Saarland University. 2007
6. O'Reilly, Tim: What Is Web 2.0. September 2005. – <http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>
7. Schwarzkopf, Eric; Heckmann, Dominik; Dengler, Dietmar; Kröner, Alexander: Mining the Structure of Tag Spaces for User Modeling. In: Complete On-Line Proceedings of the Workshop on Data Mining for User Modeling at the 11th International Conference on User Modeling. Corfu, Griechenland, 2007, pp. 63–75
8. Wahlster, Wolfgang ; Dengel, Andreas: Web 3.0: Convergence of Web 2.0 and the Semantic Web. In: Telekom Technology Radar II, June 2006, pp. 1–23
9. Heckmann, Dominikus; Christian Blass: Context Integration for Ubiquitous User Modeling: Solving Semantic Conflicts with WordNet and GUMO, 5th International Workshop on Ubiquitous User Modeling, UbiqUM'2008, Gran Canaria, Spain, 2008
10. Heckmann, Dominikus, Eric Schwarzkopf, Junichiro Mori, Dietmar Dengler, Alexander Kröner: The User Model and Context Ontology GUMO revisited for future Web 2.0 Extensions, *Contexts and Ontologies: Representation and Reasoning (2007)*, pp.37-46

11. Heckmann, Dominikus: Situation Modeling and Smart Context Retrieval with Semantic Web Technology and Conflict Resolution, *T.R. Roth-Berghofer, S. Schulz, and D.B. Leake (Eds.): MRC 2005*, LNAI 3946, pp. 34–47, Springer-Verlag Berlin Heidelberg, 2006
 12. Blass, C.: Conflict resolution strategies for ubiquitous user modeling. Student report, Department of Computer Science, Saarland University, Saarbrücken, Germany, 2004
 13. Brownston, L., Farrell, R., Kant, E., and Martin, N.: Programming Expert Systems in OPS5: An Introduction to Rule-based Programming. Addison-Wesley, Reading, Massachusetts. 1985
 14. Ram, S. and Park, J.: Semantic Conflict Resolution Ontology (SCROL): An ontology for detecting and resolving data and schema-level semantic conflicts. *IEEE Transactions on Knowledge and Data Engineering*, 16(2):189–202. 2004
 15. Peter Brusilovsky, Sergey Sosnovsky, Michael Yudelson, Amruth Kumar, Sharon Hsiao: User Model Integration in a Distributed Adaptive E-Learning System. Proceedings of the 6th International Workshop on Ubiquitous User Modeling with the focus on: User Model Integration (UMI 2008), Hannover, Germany, 2008
 16. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosz, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. <http://www.w3.org/Submission/SWRL/>
 17. McGuinness, D. L., van Harmelen, F. (eds.): OWL Web Ontology Language Overview, W3C Recommendation, February, 2004. <http://www.w3.org/TR/owl-features/>
 18. W3C OWL Working Group (eds.): OWL 2 Web Ontology Language Document Overview, W3C Recommendation, October, 2009. <http://www.w3.org/TR/owl2-overview/>
 19. Rule Markup Language Initiative. Rule Markup Language (RuleML). <http://ruleml.org/>
 20. Hori, M., Euzenat, J., Patel-Schneider, P. F.: OWL Web Ontology Language XML Presentation Syntax, W3C Note, June, 2003. <http://www.w3.org/TR/owl-xmlsyntax/>
 21. Wagner, G., Giurca, A., Lukichev, S.: R2ML: A General Approach for Marking up Rules, Dagstuhl Seminar Proceedings 05371, in Bry, F., Fages, F., Marchiori, M., Ohlbach, H. (Eds.) Principles and Practices of Semantic Web Reasoning, 2005.
 22. Reasoning on the Web with Rules and Semantics (REVERSE). <http://www.reverse.net/>
 23. Object Constraint Language (OCL), v2.0, <http://www.omg.org/docs/ptc/03-10-14.pdf>
 24. Kifer, M.: Rule Interchange Format: The Framework. In the Proceedings of the Second International Conference on Web Reasoning and Rule Systems (RR 2008), Karlsruhe, Germany, October - November, 2008.
 25. RDF Working Group: Resource Description Framework (RDF). <http://www.w3.org/RDF/>
 26. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A Nucleus for a Web of Open Data. In et al., A., ed.: The Semantic Web, 6th International Semantic Web Conference (ISWC), 2nd Asian Semantic Web Conference (ASWC), pages 715–728. November, 2007.
 27. Marchetti, A., Tesconi, M., Ronzano, F., Rosella, M., Minutoli, S.: SemKey: A Semantic Collaborative Tagging System. In: Workshop on Tagging and Metadata for Social Information Organization at WWW '07, Banff, Canada, May 8-12, 2007.
 28. Passant, A., Laublet, P.: Meaning Of A Tag: A collaborative approach to bridge the gap between tagging and Linked Data. In: Proceedings of the WWW 2008 Workshop Linked Data on the Web (LDOW2008), Beijing, China. April, 2008.
 29. Suchanek, F. M., Kasneci, G., and Weikum, G.: Yago: A Core of Semantic Knowledge. In Proceedings of 16th international conference on World Wide Web (WWW '07). Banff, Canada, May 8-12, 2007.
 30. Page, L., Brin, S., Motwani, R., Winograd, T.: The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.
- Abel, F., Kawase, R., Krause, D., Siehdnel, P., and Ullmann, N.: The Art of Multi-faceted Tagging – Interweaving spatial annotations, categories, meaningful URIs and tags. In Proceedings of 6th International Conference on Web Information Systems and Technologies (WEBIST), Valencia, Spain, April 2010. (to appear)