



GRAPPLE

D2.3 Version: 1.5

Tool for reasoning about user observations

Document Type	Deliverable
Editor(s):	Fabian Abel, Eelco Herder
Author(s):	Fabian Abel, Eelco Herder, Jan Hidders, Geert-Jan Houben, Daniel Krause, Erwin Leonardi, Kees van der Sluijs
Reviewer(s):	Frédéric Fervaille, Luca Mazzola
Work Package:	WP2
Due Date:	M30
Version:	1.5
Version Date:	September 26th, 2010
Total number of pages:	26

Abstract: This deliverable introduces the Grapple Derivation Rules (GDR) and the GDR Engine that processes these rules. Within the Grapple User Modelling Framework (GUMF), GDR rules are used for reasoning about user (event) data: evidence from multiple statements can be combined into higher-level statements; events can be converted to a different format (e.g. scale conversion); events can be contextualised using external services such as Geonames and DBPedia. The GDR administrator interface provides a user-friendly wizard for creating the rules; more advanced rules can be directly written in GDR syntax.

Keyword list: GUMF, GDR, Grapple Derivation Rules, reasoning, contextualization, annotation, XML, Grapple, user modelling

Summary

This deliverable presents the tools of the Grapple User Modelling Framework (GUMF) that allow for reasoning about user observations. GUMF provides several reasoning mechanisms. The Grapple Derivation Rule (GDR) engine is one of GUMF's most powerful and flexible reasoning components. Given appropriate GDR rules, it generates new knowledge about users by exploiting user data that might even be distributed over different data repositories. This deliverable introduces GDR and its user profile reasoning capabilities. An analysis that reveals how applications can benefit from the user profile reasoning capabilities provided by GUMF in cold-start settings is also presented.

Authors

Person	Email	Partner code
Fabian Abel	abel@l3s.de	LUH
Dominikus Heckmann	heckmann@dfki.de	DFKI
Eelco Herder	herder@l3s.de	LUH
Jan Hidders	a.j.h.hidders@tudelft.nl	TUD
Geert-Jan Houben	g.j.p.m.houben@tudelft.nl	TUD
Daniel Krause	krause@l3s.de	LUH
Erwin Leonardi	e.leonardi@tudelft.nl	TUD
Kees van der Sluijs	k.a.m.sluijs@tue.nl	TUE

Table of Contents

SUMMARY	2
AUTHORS	2
TABLE OF CONTENTS	2
TABLES AND FIGURES.....	3
LIST OF ACRONYMS AND ABBREVIATIONS	4
1 INTRODUCTION.....	5
1.1 Task and Deliverable Description.....	5
1.2 Outline of this Deliverable.....	5
2 RELATED WORK	6
3 USER PROFILE REASONING IN THE GRAPPLE USER MODELING FRAMEWORK.....	7
4 GRAPPLE DERIVATION RULES.....	8

4.1 GDR Definition	9
4.2 GDR XML Syntax	10
4.2.1 GDR Basics	10
4.2.2 gdr:premise.....	11
4.2.3 gdr:consequent.....	11
4.2.4 Operators.....	12
4.3 GDR Engine	12
4.4 Extending GUMF with GDR	14
5 REASONING ABOUT DISTRIBUTED USER PROFILE DATA	16
5.1 GDR User Profile Reasoning Examples	16
5.1.1 Data exchange with simple scale conversion.....	16
5.1.2 Localization by means of reasoning on geographical data	17
5.1.3 Reasoning on user data and semantic enhancement using external data sources.....	18
5.2 Profile Browsing powered by GDR	20
5.3 Inferring User Profiles in cold-start situations	22
CONCLUSIONS AND FUTURE WORK	25
REFERENCES	26

Tables and Figures

List of Figures

Figure 1 User Modeling with GUMF dataspace	7
Figure 3 Grapple statement: Bob's name is Bob Myers.....	8
Figure 5 - The Architecture of GDR Engine.....	13
Figure 7 - GUMF Architecture	15
Figure 9 - GUMF dataspace administration lists GDR rules.	15
Figure 11 - Creating/modifying GDR rules.....	16
Figure 13 - An Example of a GDR Rule in XML Syntax (partial view).....	18
Figure 15 - The Snapshot of A1 and A2 Dataspaces (partial view)	18
Figure 17 - An Example of a GDR Rule in XML Syntax.....	19
Figure 19 - Graph Patterns Across Three Different Data Sources.....	20
Figure 20 - Derived Grapple Statements.....	20
Figure 21 - User profile browsing demo: GDR rules are deactivated.....	20
Figure 22 - User profile browsing demo: GDR rules are activated.....	22
Figure 24 - Average performance of tag prediction with and without utilizing distributed profiles (profile aggregation).	23

Figure 25 - Improving prediction performance (with profile aggregation) by means of Wordnet categorization..... 24

Figure 26 - Tag-based profile prediction performance for the different service constellations..... 24

Figure 27 - Predicting the different facets of tag-based StumbleUpon profiles based on Delicious profiles..... 25

List of Tables

Table 1 - GDR language concepts..... 10

Table 2 - GDR rules for (a) importing and mapping user data from external dataspace and (b) inferring further knowledge by utilizing external Web services..... 22

List of Acronyms and Abbreviations

ALE	Adaptive Learning Environment
FOAF	Friend of a Friend
GDR	Grapple Derivation Rule
GRAPPLE	Generic Responsive Adaptive Personalised Learning Environment
Grapple Statement	Specific statements about a user → a user profile is a set of GRAPPLE statements
GUMF	Grapple User Modelling Framework
GUMO	General User Model Ontology
LMS	Learning Management System
LOM	Learning Object Metadata
OWL	Ontology Web Language
RDF	Resource Description Framework
SCORM	Sharable Content Object Reference Model
SUMO	Suggested Upper Merged Ontology
TEL	Technology-Enhanced Learning
UM	User Model (or sometimes “User Modelling”) a set of Grapple Statements
UMF	User Model Framework
UserML	User Model Markup Language
UserQL	User Model Query Language
WP	Work Package
WPL	WP Leader

1 Introduction

Most, if not all, learning management systems store data about their users in order to keep track of the user's activities. The most basic activity logs register which content has been accessed, which tests have been completed and the scores achieved for these tests. An additional source of information on the user is given by the user profile, which users are typically requested to fill in upon registering.

Exchanging scores or (inferred) knowledge levels from one system to another typically requires conversion. As a basic example, a score might need to be converted from a 1-5 scale to a 1-100 scale. More elaborate conversions may include computing the average of the knowledge level on a concept, as inferred by two different systems. Additionally, user profile data that is available in one system may not be available in the other system. Within the Grapple User Modelling Framework, Grapple Derivation Rules (GDR) are used for these types of conversions.

Grapple Derivation Rules provide a generic framework for reasoning over distributed user data that goes beyond simple conversions and aggregation of user (event) data. By incorporating external data sources GDRs can be used for enriching or contextualising user event data: a statement that "Peter likes Sweden" can be semantically enriched with statements that clarify that Sweden is a country and that Sweden's capital is Stockholm.

This deliverable introduces the Grapple Derivation Rules and how they are integrated in the Grapple User Modelling Framework. Examples on how the GDRs are used are also provided – examples varying from basic conversion to more enhanced semantic enrichment.

1.1 Task and Deliverable Description

Task 2.3 is the development of reasoners for inferring user behaviour and characteristics. Based on the information about user interactions captured by the user event and contextualisation framework (see Deliverable D2.a/b), user events will be interpreted by reasoning about contextual information, the knowledge contained in relevant ontologies and the modelling approach chosen for user modelling. Different reasoning strategies have been explored, among them a rule-based approach and a probability-based approach for reasoning about singular users and an association-based and recommender-based approach for reasoning about collaborative user behaviour.

In D2.2a/b components have been developed that gather information about users (e.g., user interactions observed in Learning Management Systems) and enrich the observed user events with contextual information. This extended service component uses reasoning over domain- and user model information in order to translate low-level user model updates into high-level ones, like the effect of user actions on knowledge about concepts.

1.2 Outline of this Deliverable

In Chapter two a short summary of related work on interoperability has been provided, system mash-ups and data conversion (a more elaborate discussion on related work can be found in the D2.2 deliverables). Chapter three describes the GUMF approach to user modelling: the use of logical dataspace and how these dataspace can be shared; the activation or creation of conversion and reasoning functionality via plug-ins and the concept of semantic enhancement. Chapter four introduces the Grapple Derivation Rules: the syntax, the basic usage, the GDR engine and the integration in GUMF. Chapter five gives various examples on the use of GDR, ranging from simple scale conversion to more advanced semantic enhancement.

2 Related Work

This short chapter provides a short summary of work related to reasoning about user observations. More related work can be found in the Grapple D2.2 deliverables.

Nowadays, numerous Web applications provide adapted and personalised contents and services to their users. To be able to provide such contents and services, these applications explicitly or implicitly collect data about their users and their behaviour. Explicit user data collection approaches rely on asking the user directly for information, for example by using a survey form or by asking the user to give ratings to certain products.

A common problem in LMSs is that the user population and the number of user activities is too small for reliably reasoning about the data. Moreover, user profile information is often limited, as most LMSs do not require students to fill out complete profiles. In the past few years, LMSs have embraced functionality for communication and collaboration and several learning institutions report that they currently use Web 2.0 tools [8].

Mashing up data and tools into one integrated tool has become increasingly popular recently [6, 13, 17]. One work in the mashup related environment that is closely related to our research is presented in [4, 5]. In [4, 5], Gosh et al. present a framework called SUPER (*Semantic User Profile Management Framework*) for capturing and maintaining user profiles using semantic web technology in the retail domain. SUPER aggregates user profile information that spreads over several services/data sources.

Implicit approaches imply the observation of the users' behaviour: Web applications log and monitor the user behaviour in order to construct a user model fitting with the personalisation goals of the application. A key concern in developing such adaptive Web applications is to model the users and their behaviour for achieving the personalisation and adaptation goal of the applications. At the same time, these Web applications are becoming increasingly connected. This creates the interesting challenge of performing user modelling and personalisation across application boundaries. It requires approaches allowing various Web applications to exchange, reuse, interlink and integrate user data. On the one hand, the ability of exchanging, reusing, interlinking and integrating the user models allows applications to enhance and broaden their user models with additional data. In addition, it is particularly essential for a better integration and cooperation between the applications. On the other hand, it helps users to get the content and services that suit their needs and situations and to syndicate these services. As different applications may represent the same information in different ways, using different syntactic and semantic, the Web applications have to ensure interoperability of the user data in order to be able to exchange, reuse and integrate user data. Consequently, addressing the interoperability issue is essential when developing interoperable adaptive Web applications.

To address the user profile interoperability there are basically two approaches: the *shared format* approach and the *conversion* approach. In the shared format approach, a common language for a unified user profile (*a lingua franca*) is needed. Applications have to follow the unified format [15]. Examples of this approach are the General User Model Ontology (GUMO) [7] and Composite Capability/ Preference Profiles (CC/PP)¹. This approach is easily exchangeable and interpretable as there is no syntactic and semantic heterogeneity issue to be addressed [7]. However, this approach is not suitable for open and dynamic environments, such as the Web as it is impractical and in many cases impossible to enforce Web applications to follow the *lingua franca* [11] because new applications might have individual needs and require new vocabulary concepts to describe application-specific characteristics. The conversion approach is more flexible and suitable for open and dynamic environments [3]. In this approach, a technique has to be developed for converting a user profile of one application to another application. The developed technique should deal with the problem of syntactic and semantic heterogeneity. Observe that potential drawbacks of this approach are that it is possible that some information is lost during the conversion process and that it is possible that models are simply incompatible. This means that there is no suitable mapping for these models. It is also possible that the mappings are incomplete because required information in one model is not available in the other model. For example, with the advent of social tagging, RDF vocabularies such as the Tag Ontology² have been developed, which can hardly be mapped to former vocabularies such as DCMI metadata³ without losing information, because concepts such as tag assignments (user-tag-resource relations) can only

¹ <http://www.w3.org/Mobile/CCPP/>

² <http://www.holygoat.co.uk/projects/tags/>

³ <http://dublincore.org/documents/dcmi-terms/>

approximately be modelled with DCMI terms (e.g., with dc:subject one loses the information which user assigned the tag to the resource).

User Profile Reasoning in the Grapple User Modelling Framework

User modelling functionality of GUMF is embedded into so-called dataspace. A dataspace logically bundles data that might be stored on different servers and that is possibly maintained by other applications. The user modelling components that are applied to enrich data stored by client applications have been implemented as depicted in Figure 1 [1]. Client C1 stores information about a user in a dataspace and more precisely in the repository associated with the dataspace. C1 might for example report that a new user registered to the system. Information about the user is internally modelled by means of Grapple statements, for example, C1 stores that a new user whose name is "Bob Myers" registered to C1. Figure 2 shows the corresponding statement in RDF/XML syntax.

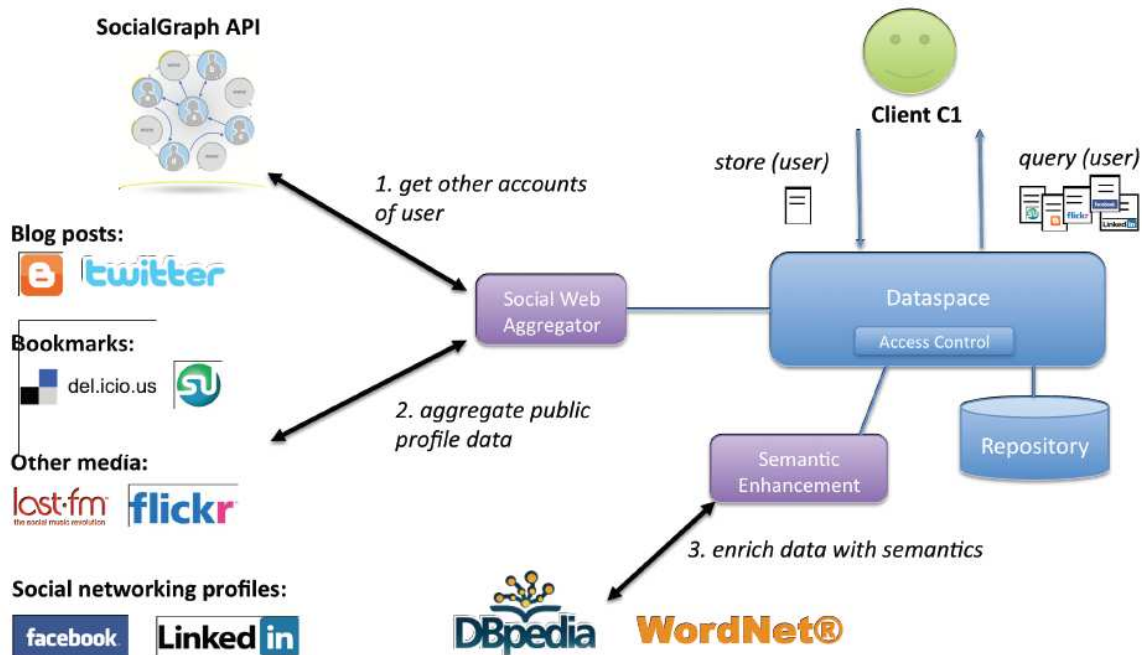


Figure 1 User Modelling with GUMF dataspace

Grapple statements are subject-predicate-object bindings enriched with metadata. Below there is an XML-formatted example of a Grapple statement. Grapple statements not only describe the actual statement, i.e. Bob's (*gc:subject* = *http://bob.myopenid.com*) name (*gc:predicate* = *http://xmlns.com/foaf/0.1/name*) is "Bob Myers" (= *gc:object*), but also additional details such as the creator of the statement (*gc:creator*), the time when the statement was created (*gc:created*) or the degree to which the statement holds for the subject (*gc:level*)⁴. Storing a Grapple statement might trigger some plug-ins embodied into the dataspace. In Figure 1, the *Social Web Aggregator* [1] obtains other accounts the user has via the *Social Graph API*⁵. Given these mappings, the plug-in gathers – if available – public profile data about the user from the corresponding platforms: tag-based profiles from Delicious, StumbleUpon, Last.fm, and Flickr, social network profiles from LinkedIn and Facebook, and blog posts from Twitter and Blogspot. The aggregated profile data is then enriched with semantic annotations (*Semantic Enhancement* in Figure 1). In particular, the elements of the tag-based profiles are mapped to DBpedia URIs that specify the semantic meaning of the tags and WordNet⁶ categories are applied to cluster the profile. Hence, based on the rather basic Grapple statement, as listed in Figure 2, GUMF gathers the distributed profile traces of the user so that the client can exploit a rich profile the next time it is querying the dataspace (cf. Figure 1).

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:gc="http://www.grapple-project.org/grapple-core/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

```

⁴ Note that *gc:creator* and *gc:created* are sub-properties of *dc:creator* and *dc:created* as defined by DCMI (see also Deliverable D2.1).

⁵ <http://socialgraph.apis.google.com>

⁶ <http://wordnet.princeton.edu>

```

xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <gc:Statement rdf:about="http://grapple-project.org/2010-01-28-526341">
    <gc:subject rdf:resource="http://bob.myopenid.com"/>
    <gc:predicate rdf:resource="http://xmlns.com/foaf/0.1/name"/>
    <gc:object>Bob Myers</gc:object>
    <gc:level rdf:datatype="xsd:double">1.0</gc:level>
    <gc:created rdf:datatype="xsd:dateTime">
      2010-01-28T00:09:20.621+02:00
    </gc:created>
    <gc:creator rdf:resource="http://grapple-project.org/client/1"/>
  </gc:Statement>
</rdf:RDF>

```

Figure 2 Grapple statement: Bob's name is Bob Myers.

Dataspaces go beyond the notion of namespaces as they explicitly denote a set of things (e.g. data, reasoning rules, data aggregation plug-ins, schema mapping rules), on which an operation – such as a query, store or reasoning operation – should be performed. In more detail, such dataspace represent the part of GUMF that a certain client application is managing and is responsible for, i.e., its own workspace. The Administrator of a GUMF client application can configure dataspace and plug-ins via the *GUMF Admin Interface*. Activating or deactivating plug-ins and adjusting plug-ins and reasoning rules directly influence the behaviour of dataspace. Inspired by Web 2.0 practices, a key principle of GUMF is that dataspace can be shared across different client applications. Therefore, clients can subscribe to other dataspace, as long as the administrator of the dataspace approves them. When subscribed to a dataspace, the client is allowed to query it. However, it might still not be allowed to access all statements that are made available via the dataspace, as fine-grained access control functionality can be embedded in the dataspace as well.

The components that are plugged into dataspace come in different flavours: Some plug-ins are black-box components, whose functionality is hard-coded and can be adjusted by passing parameters (e.g., data aggregators for LinkedIn, Flickr, etc. expect the username for whom the data should be aggregated, but the core functionality of these components cannot be changed) while others are rule-based and are thus highly flexible. In the next section, the GDR language is introduced that extends and enhances the reasoning capability of GUMF and enables developers and administrators to create such flexible, rule-based dataspace plug-ins that are capable of integrating user data from multiple Grapple dataspace and data published as Linked Data on the Web.

3 Grapple Derivation Rules

The Grapple User Modelling Framework (GUMF) facilitates the brokerage of user profile information and user model representations. The Grapple Derivation Rule language (GDR) and the corresponding GDR engine extend GUMF with a new flexible rule-based method that enhances the reasoning capability of GUMF. GDR rules allow the applications to specify a "recipe" that guides how new knowledge should be deduced from possibly distributed data sources.

GDR is a rule language that operates on Grapple statements. A GDR rule allows authors to specify the kind of Grapple statements that can be deduced when a given premise (specified via Grapple statement) holds. Therefore, GDR can be applied to solve the following tasks.

- User profile reasoning: deduction of Grapple statements describing a user.
- Conflict resolution: GDR rules can specify conditions for the selection of Grapple statements.
- Profile mapping: schema conversion such as mapping of predicates (e.g., *gale:knowledge* ⇒ *external:knowledge*).

GDR extends GUMF with the flexibility for applications to define configurations that guide the user data integration and enrichment processes. GDR enables applications not only to integrate data from GUMF dataspace, but also to incorporate and reuse linked data published on the Web.

This section, elaborates in detail on the Grapple Derivation Rule language (GDR) that enables GUMF to provide a flexible way of defining plug-ins by allowing the applications to specify a "recipe" for integrating and enriching user data. The GDR Engine that processes a GDR rule and derives new Grapple statements is also discussed. Finally, the extension of GUMF with GDR is presented.

3.1 GDR Definition

this section provides a concise, formal definition of the Grapple Derivation Rules. The definitions provided here will be used in the example-based explanation of the GDR basics in the next section.

In human readable syntax, a GDR rule has the form: $a \Rightarrow c$, where:

- a and c are the respectively the antecedent and the consequent of the rule
- a is a conjunction of premises written $p1 \wedge \dots \wedge pn$

A set of GDR rules is interpreted as OR-connected so that there is no need to allow for conjunctions in the premises. The **premises** of a GDR rule are classified into two types: (1) dataspace premises and (2) external source premises.

1. A *dataspace premise* describes conditions over a Grapple dataspace as of a pattern-based Grapple Query.
2. An *external source premise* specifies conditions as triple patterns over an external data source accessible through a SPARQL endpoint.

The **consequent** describes the Grapple statements that will be derived if all the premises are hold. Basically, it specifies the subject, predicate and object properties of the Grapple statements and optionally additional fields such as *level* properties.

A GDR rule also has extra information such as name, description and creator. **Variables** are indicated using the standard convention of prefixing them with a question mark (e.g., $?x$). The GDR rule is formally defined as follows.

Definition 1: Dataspace Premise. A dataspace premise d is a 2-tuple (ds, f) , where ds is the Grapple dataspace identifier, and f is a partial function that maps a finite set of Grapple statement properties to variables and constants. A set of dataspace premises is defined as D .

Definition 2: External Source Premise. An external source premise e is a 4-tuple $(uri, endpoint, namedGraph, T)$, where:

- uri is the informal identifier of the dataset
- $endpoint$ is the URI of SPARQL endpoint of the data source where the dataset is stored
- $namedGraph$ is the named graph that is used to store the dataset in the data source
- T is a basic graph pattern with at least one Grapple statement pattern such that these fields of the statement, which can have arbitrary values, are set with variables (e.g., $gc:subject = ?X$) while others are set with real values (e.g., $gc:predicate = http://xmlns.com/foaf/0.1/interest$).

A set of external source premises is defined as E .

Definition 3: Consequent. A consequent c is a 2-tuple (ds, T) , where T is a graph pattern that specifies what kind of Grapple statements are generated if the premise of the GDR rule holds and ds specifies the dataspace where these statements will be made available.

Definition 4: A GDR Rule. A GDR rule r is a 3-tuple (M, A, c) , with:

- M is a set of additional information of r , such as the name, description, and creator of the rule,
- A is the antecedent of the rule, which is a conjunction of premises written $p1 \wedge \dots \wedge pn$, where $pi \in (D \cup E)$ and $n > 0$,
- c is the consequent of the rule such that all variables in c appear in at least one premise of A .

Disjunctions (OR) are not permitted in the premise of a GDR rule. However, they can be modelled by constructing a set of GDR rules. Each GDR rule is individually evaluated by the GDR engine, i.e. GDR rules are OR-connected.

3.2 GDR XML Syntax

XML namespace of GDR XML syntax:

- The **namespace** of GDR is: `http://www.grapple-project.org/grapple-derivation-rule/`
- Standard **namespace abbreviation**: `gdr`

3.2.1 GDR Basics

In XML syntax, a GDR rule is described by means of the `gdr:rule` element which embodies the premise(s) (`gdr:premise`) and consequence (`gdr:consequent`) of the GDR rule. The basic structure of an XML-formatted GDR rule is as follows.

```
<gdr:rule
  xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
  ...>
  <gdr:premise ...>
    ...
  </gdr:premise>
  <gdr:consequent ...>
    ...
  </gdr:consequent>
</gdr:rule>
```

The above elements `gdr:rule`, `gdr:premise`, and `gdr:consequent` can/must have the following attributes.

element	attribute	value range	Mandatory/optional	description
gdr:rule	name	string	mandatory	meaningful name or title of the rule
	description	string	optional	description of the rule, i.e. short explanation for users that describes the purpose of the rule
	creator	anyURI	optional	some URI that identifies the creator of the GDR rule
	id	<i>not restricted</i>	optional	GDR rules might have an identifier to facilitates maintenance of GDR rules, e.g. in GUMF, where GDR rules are associated with dataspace, the <code>id</code> of a GDR rule is used by GUMF to identify the rule and thereby allow for maintenance of rules (change/remove/activate/de-activate rules).
gdr:premise	dataspace	integer or anyURI	mandatory for <i>dataspace premises</i> (see Def. 1)	the ID or URI of the dataspace for which the premise should be evaluated
	source	anyURI	mandatory for <i>external source premises</i> (see Def. 2)	the URI of the source for which the premise should be evaluated
gdr:consequent	dataspace	integer or anyURI	mandatory	the ID or URI of the dataspace in which the consequent of the rule will be valid if all premises hold

Table 1 - GDR language concepts.

3.2.2 gdr:premise

GDR supports so-called *dataspace premises* (see Def. 1) and *external source premises* (see Def. 2). A dataspace premise describes a Grapple statement pattern that will be applied to a particular GUMF dataspace. To specify the Grapple statement pattern the Grapple Core format (cf. grapple-core.owl) is used. For example:

```
<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#"
  id="1"
  name="Name of the Rule"
  description="Short description"
  creator="http://example.org/creator">

  <gdr:premise dataspace="1">
    <gc:subject>userXY</gc:subject>
    <gc:predicate rdf:resource="http://example.org/predicate"/>
    <gc:object>?object</gc:object>
    <gc:level>?level</gc:level>
  </gdr:premise>

  <gdr:consequent ...>
    ...
  </gdr:consequent>
</gdr:rule>
```

Variables start with a question mark (e.g., *?object* and *?level*). Things that do not start with a question mark are interpreted as values (e.g., *userXY* and *http://example.org/predicate*). The above dataspace premise matches all Grapple statements in the dataspace, which is identified via its ID (here: "1"), where the user (*gc:subject*) is equal to *userXY* and the predicate is equal to *http://example.org/predicate* while the object (*gc:object*) and level (*gc:level*) can have arbitrary values. The graph pattern of the premise of the above GDR rule could be translated into a SPARQL query as follows:

```
PREFIX gc:    <http://www.grapple-project.org/grapple-core/>
CONSTRUCT ...
WHERE {
  ?X  rdf:type          gc:Statement .
  ?X  gc:subject        ?user . FILTER regex(?user, "userXY") .
  ?X  gc:predicate      <http://example.org/predicate> .
  ?X  gc:object         ?object .
  ?X  gc:level          ?level
}
```

Clearly the SPARQL query is far more complex than the GDR premise pattern as the Grapple statement structure has to be translated into RDF.

3.2.3 gdr:consequent

The consequent of a GDR rule specifies the Grapple statements that can be deduced when all premises are true, using a Grapple statement pattern. For example:

```
<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#"
  ...>

  <gdr:premise dataspace="1">
    ...
  </gdr:premise>

  <gdr:consequent dataspace="1">
    <gc:subject>userXY</gc:subject>
```

```

<gc:predicate rdf:resource="http://example.org/otherPredicate"/>
<gc:object>?object</gc:object>
<gc:level>?level</gc:level>
</gdr:consequent>

</gdr:rule>

```

The above example says that Grapple statements such as (*userXY*, *http://example.org/otherPredicate*, *?object*, *?level*) can be deduced when the premise(s) hold. The variables will then be replaced by the corresponding instantiations that are obtained when the premise of the GDR rule is matched.

3.2.4 Operators

GDR currently supports the following operators.

operator	Operation applicable to	description
op:multiply(?level, X)	<i>?level</i> * <i>X</i> gc:level	multiplies the given <i>?level</i> with <i>X</i>
op:divide(?level, X)	<i>?level</i> / <i>X</i> gc:level	divides the given <i>?level</i> by <i>X</i>
op:add(?level, X)	<i>?level</i> + <i>X</i> gc:level	adds <i>X</i> to the given <i>?level</i>
op:subtract(?level, X)	<i>?level</i> - <i>X</i> gc:level	subtracts <i>X</i> from the given <i>?level</i>

For example, the following GDR rule instructs the multiply operator to increase the knowledge level of a student for the concept "gale://.SolarSystem" if the student completes a quiz (with ID 118316) that tests the student's knowledge in this field. The rule simply multiplies the current knowledge level for these concepts by 1.5, i.e. new knowledge level of SolarSystem is equal to the previous knowledge level multiplied by 1.5:

```

<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="Quiz Level to
External Knowledge (CLIX)" description="Maps the quiz/test result of a specific
CLIX score to external knowledge"
creator="http://pcwin530.win.tue.nl:8080/grapple-umf/client/1">
  <gdr:premise dataspace="1">
    <gc:subject>?user</gc:subject>
    <gc:predicate rdf:resource="http://www.grapple-project.org/ims-
lip/completedTest"/>
    <gc:object>118316</gc:object>
    <gc:level>?level</gc:level>
  </gdr:premise>
  <gdr:consequent dataspace="1">
    <gc:subject>?user</gc:subject>
    <gc:predicate rdf:resource="http://gale.tue.nl/predicate/knowledge"/>
    <gc:object>gale://gale.tue.nl/cam/DavidTestCAM/SolarSystem</gc:object>
    <gc:level>op:multiply(?level,1.5)</gc:level>
  </gdr:consequent>
</gdr:rule>

```

3.3 GDR Engine

The GDR Engine is responsible to derive new knowledge based on a "recipe" defined in a GDR rule that possibly effects the integration of data from different data sources. Figure 3 depicts its architecture and interactions with other GUMF modules.

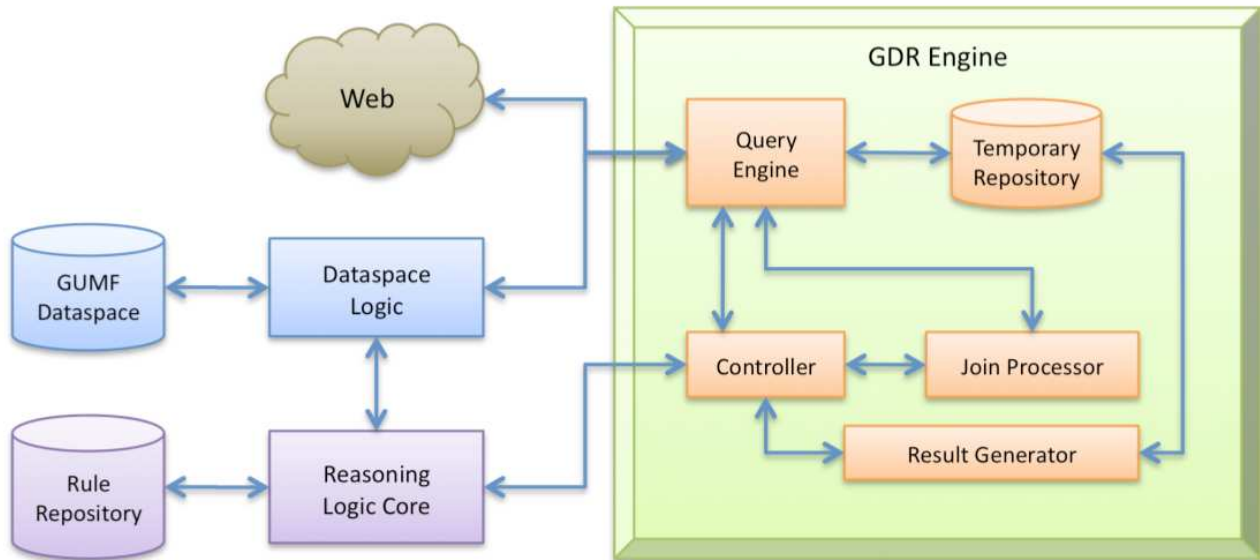


Figure 3 - The Architecture of GDR Engine

The GDR Engine consists of five components:

1. **Query Engine (QE):** The QE inside the GDR Engine performs the following tasks: 1) by sending query requests to the GUMF Dataspace Logic, it fetches data from GUMF dataspaces; 2) it queries external data sources through SPARQL endpoints; 3) it reads and writes data that is temporarily maintained in the TR.
2. **Temporary Repository (TR):** The TR component is an RDF repository that is used to store the RDF triples of intermediate results (e.g. join results).
3. **Join Processor (JP):** The JP component is responsible for performing join operations. This component interacts with the QE whenever it wants to retrieve data from the TR and the external data sources as well as to put data into the TR.
4. **Result Generator (RG):** The RG analyses the premises and consequent of the rule, generates a SPARQL query that will be issued against the GDR's temporary repository and constructs a set of Grapple statements to be returned to the GUMF Reasoning Logic Core as the result.
5. **Controller:** the Controller manages the whole process that takes place inside the GDR Engine. It receives requests from the GUMF Reasoning Logic Core. It also utilises the QE to fetch data and maintain intermediate data temporarily and the JP component to perform join operations. It exploits the RG to generate a set of newly derived Grapple statements.

When a client application issues a pattern-based query q to GUMF (e.g., `../user/anna/predicate/knowledge` will return all statements that specify the knowledge the user 'anna' has with respect to some concept), the Dataspace Logic forwards query q to the Reasoning Logic. The Reasoning Logic Core module then checks if there are any GDR rules relevant for q . For each relevant rule r , the Reasoning Logic Core sends a request to the GDR Engine to process rule r . The GDR Engine then performs the following steps.

1. The GDR Engine first evaluates all the dataspace premises of r and maintains the result of each dataspace premise evaluation in the TR by utilising the QE.
2. Based on the Grapple Query patterns specified in the dataspace premises, the QE sends requests to the GUMF Dataspace Logic to fetch data from dataspaces.
 - If no dataspace premise evaluations are applicable, the GDR Engine stops processing r and returns null, meaning that rule r derives no result. The intuition behind this is as following: the empty result of a dataspace premise d means that there is no Grapple statement that satisfies the pattern defined in premise d . Consequently, premise d does not hold, and thus rule r does not hold.

- In the case that all dataspace premise evaluations return results, the GDR Engine continues processing r .
3. Next, the GDR Engine exploits the JP to join the dataspace premises using the results stored in the TR.
- If two dataspace premises $d1$ and $d2$ share the same variables, i.e. if the graph patterns of the corresponding premises overlap, they can be joined. The join results are also temporarily stored in the TR.
 - If two premises can be joined, but the join result is empty, the GDR Engine stops processing r and returns null.

Note that in the current implementation, the GDR Engine joins the dataspace premises based on the appearance order in r . Optimising the join order is an interesting and non-trivial research problem. However, since the focus of this paper is to present a configurable method for integrating and enriching user data, the join optimisation issue will be investigated in the future.

4. The next step is to process the external premises. The GDR Engine also processes the external source premises according to the appearance order in r . Given an external premise e , the JP checks if e can be joined with previously processed premises (both dataspace and external source premises).
- If e can be joined, the QE is exploited to fetch the data of previously processed premises stored in the TR. It will construct SPARQL queries based on the specified triple patterns and fetched data and send these queries to the SPARQL endpoint of e . The results of these queries are maintained by the TR.
 - If e can be joined with another external source premise e_j that is not processed yet, the JP will process e_j first before processing e . This process stops if all possible joins between premises are performed.
 - If there is an external source premise e_k that cannot be joined with other premises, the JP requests the QE to construct a SPARQL query based on just the specified triple patterns. In constructing the query, the QE takes into account whether or not premise e_k and the consequent of r share at least one variable. If they do not share any variables, the QE rewrites the query to an ASK form to test whether or not it has a solution. The intuition is that if a premise cannot be joined with other premises and the data from this premise will not be used in the final result, it should be checked whether this premise returns any results. The constructed query is sent to the SPARQL endpoint of e_k and the result is stored in the TR.
5. After all premises are processed, the Controller sends request to the RG that generates a set of new Grapple statements. The RG analyses rule r and generates a SPARQL query that is executed against the temporary data stored in the TR. The result of this SPARQL query is then modelled as Grapple statement and sent to the Controller, which subsequently sends it to the Reasoning Logic Core.

3.4 Extending GUMF with GDR

The GDR Engine has been implemented in Java. For the Temporary Repository component, the implementation has been based on the open-source RDF framework Sesame⁷. Sesame offers a good level abstraction on connecting to and querying of RDF data, similar to JDBC.

The GDR Engine is integrated into GUMF as a module inside the Reasoning Logic as depicted in Figure 4.

⁷ <http://www.openrdf.org/>

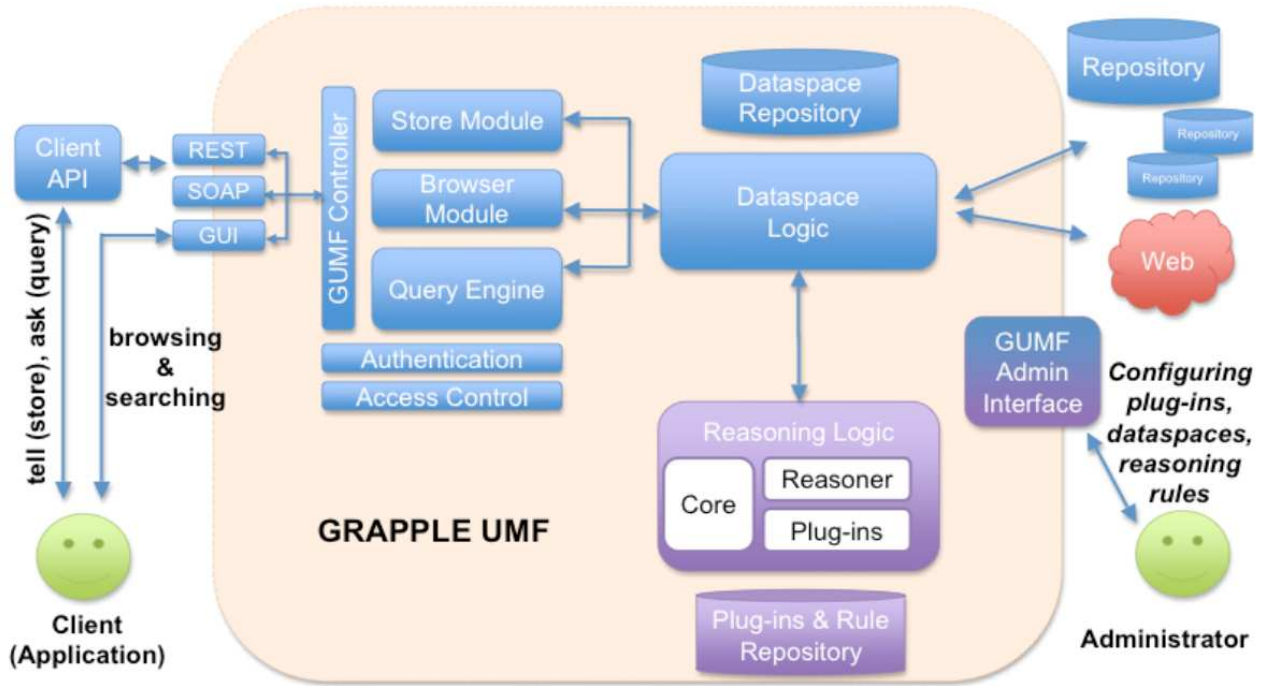


Figure 4 - GUMF Architecture

The integration of GDR required the extension of several components in GUMF. For example, a feature in the Reasoning Logic Core component has been added to detect which GDR rules in the dataspace are relevant for a Grapple query sent by GUMF client. This can be done by analysing the consequent of the rules. The Reasoning Logic Core has to be able to communicate with the GDR Engine. The GUMF administrator page was extended to show the list of specified GDR and a hyperlink to the GDR rule creation page (Figure 5).

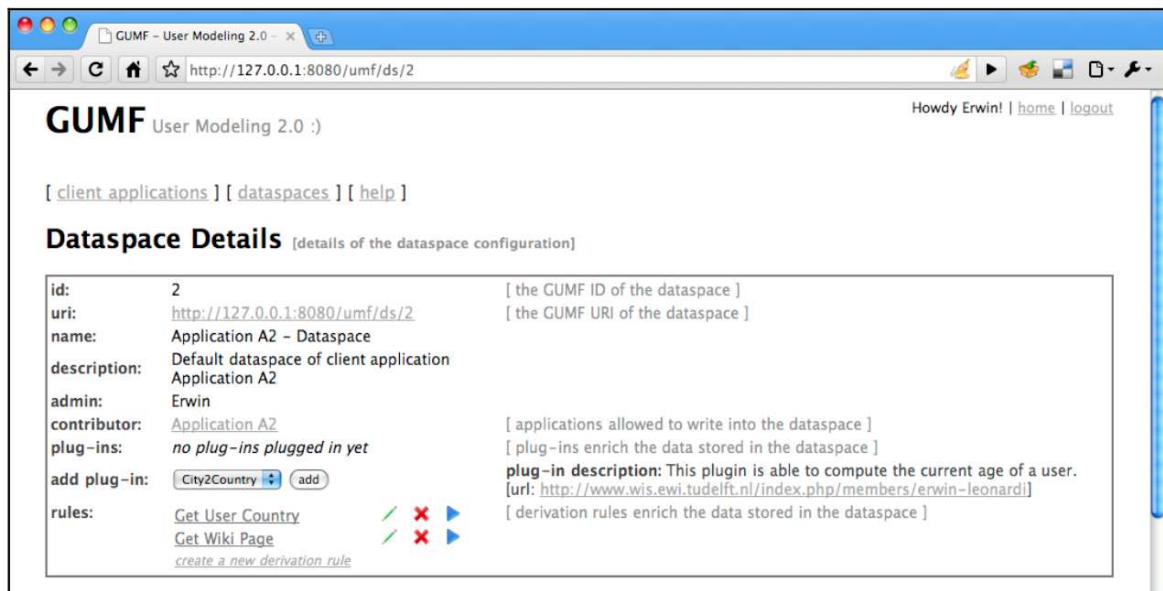


Figure 5 - GUMF dataspace administration lists GDR rules.

There are two ways of creating a GDR rule: friendly mode and expert mode. In the friendly mode, shown in Figure 6, the dataspace administrator specifies the rule by filling in the provided form fields.

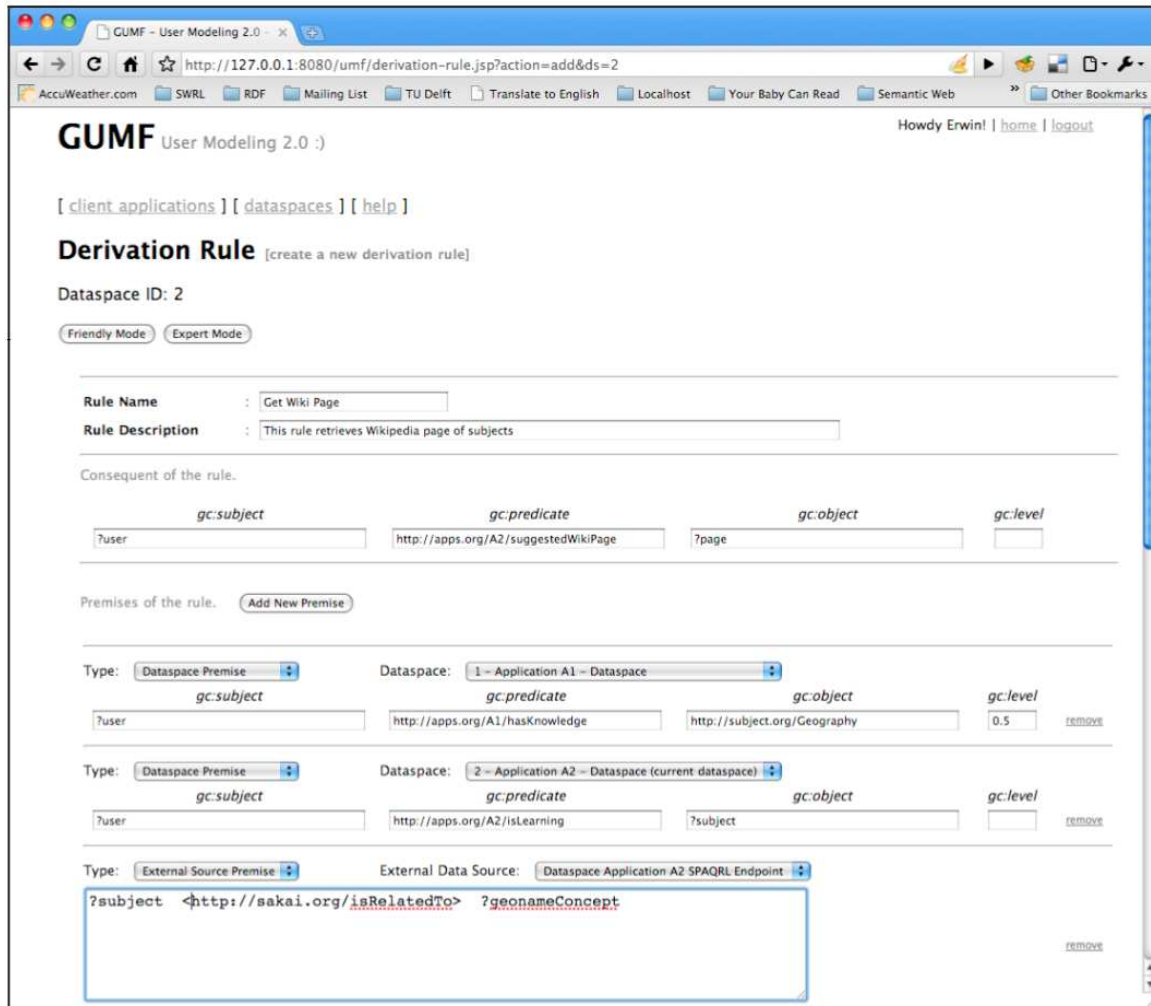


Figure 6 - Creating/modifying GDR rules

In the expert mode, the administrator has to type the rule in XML format.

4 Reasoning about distributed user profile data

This chapter showcases the user profile capabilities of GUMF by means of examples.

4.1 GDR User Profile Reasoning Examples

In the subsequent sections different usage examples of Grapple Derivation Rules are presented. In the first example, a quiz result is mapped to a knowledge level – together with a simple scale conversion. In the second example, reasoning about user profile data and geographical concepts is used for localisation. In the third example, concepts that a user is sufficiently familiar with, are linked to articles that provide additional background information on these concepts.

Despite the differences in 'intelligence', all examples share the same principle: if a combination of statements can be found that share a given pattern (the premise), the contents of these patterns is used for deriving a new statement (the consequent).

4.1.1 Data exchange with simple scale conversion

Below is an example of a simple GDR rule that basically maps the predicate and level attribute of Grapple statements: Grapple statements that match the pattern (*?user, ..completedTest, solar-system-quiz, ?level*) are mapped to statements like (*?user, ..knowledge, ..SolarSystem, ?level * 10*).

```
<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#"

id="1"
name="Quiz Level to External Knowledge (CLIX)"
description="Maps the quiz/test result of a specific CLIX score to external
knowledge"
creator="http://pcwin530.win.tue.nl:8080/grapple-umf/client/1">

  <gdr:premise dataspace="1">
    <gc:subject>?user</gc:subject>
    <gc:predicate rdf:resource="http://www.grapple-project.org/ims-
lip/completedTest"/>
    <gc:object>solar-system-quiz-id</gc:object>
    <gc:level>?level</gc:level>
  </gdr:premise>
  <gdr:consequent dataspace="1">
    <gc:subject>?user</gc:subject>
    <gc:predicate rdf:resource="http://gale.tue.nl/predicate/knowledge"/>
    <gc:object>gale://gale.tue.nl/cam/DavidTestCAM/SolarSystem</gc:object>
    <gc:level>op:multiply(?level,10)</gc:level>
  </gdr:consequent>
</gdr:rule>

```

4.1.2 Localisation by means of reasoning on geographical data

Consider a simple example of using GDR to enrich user data. Suppose an adaptive application wants to select the language in which it presents pages to its user, based on the country the user lives in. Unfortunately, the application only has information about the city the user lives in her user profile. To address this, the administrator of the application defines a GDR rule as depicted in Figure 7 over the application's own data space and background knowledge. This rule specifies that if the name of the city where a user lives is *cityName* and the city is located in a country with name *countryName*, then the name of the country where the user lives is *countryName*. The relation between city and country can be discovered by exploiting knowledge available in an external data source; GeoNames, which can also translate between different languages. It can for example, either use "Hannover" or "Hanover" as *cityName* to query for the capital of Lower Saxony in Germany. The rule not only reasons over the user data that the application has but also extends and enriches it with knowledge from external data sources. It also provides a view-based query mechanism to the application, as the user's country information is dynamically computed and available from another external data source. Using the presented language, applications can easily integrate user data from other applications they have access to, or add external data sources, by modifying the rules without changing the applications. This way it provides flexibility to the administrators of adaptive applications.

```

01 <gdr:rule xmlns:gc="http://www.grapple-project.org/grapple-core/"
02     xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
03     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
04     name="Get User's Country" ... >
05 <gdr:premise dataspace="http://www.grapple-project.org/gumf/ds/1">
06   <gc:subject>?user</gc:subject>
07   <gc:predicate rdf:resource="http://profile.org/city" />
08   <gc:object>?cityName</gc:object>
09 </gdr:premise>
10 <gdr:premise sourceURI="http://geonames.org/" ... >
11   <gdr:pattern>?city geo:name ?cityName</gdr:pattern>
12   <gdr:pattern>?city geo:featureClass geo:P</gdr:pattern>
13   <gdr:pattern>?city geo:inCountry ?countryURI</gdr:pattern>
14   <gdr:pattern>?country geo:featureClass geo:A</gdr:pattern>
15   <gdr:pattern>?country geo:inCountry ?countryURI</gdr:pattern>
16   <gdr:pattern>?country geo:name ?countryName</gdr:pattern>
17 </gdr:premise>
18 <gdr:consequent dataspace="http://www.grapple-project.org/gumf/ds/1">
19   <gc:subject>?user</gc:subject>
20   <gc:predicate rdf:resource="http://profile.org/country" />
21   <gc:object>?countryName</gc:object>
22 </gdr:consequent>
23 </gdr:rule>

```

Figure 7 - An Example of a GDR Rule in XML Syntax (partial view).

4.1.3 Reasoning on user data and semantic enhancement using external data sources

Suppose there are two adaptive e-learning applications; A1 and A2 that use GUMF. A1 is a Moodle-based application that is used for a basic Geography course, and A2 is an AHA!-based application that is used for an Urban Geography course. Figure 8(a) depicts a set of Grapple statements in the A1 dataspace. Figure 8(b) depicts a set of Grapple statements in the A2 dataspace. A set of triples derived by a semantic enhancement plug-ins that relates data in the dataspace to the GeoNames concepts is shown in Figure 8(c).

gc:subject	gc:predicate	gc:object	gc:level
user:anna	profile:origin	"Dubai"	1.0
user:anna	A1:hasKnowledge	subject:Geography	0.8
user:bob	profile:origin	"Delft"	1.0
user:cindy	profile:origin	"Johannesburg"	1.0
user:cindy	A1:hasKnowledge	subject:Geography	0.6
user:donald	profile:origin	"Beijing"	1.0
user:donald	A1:hasKnowledge	subject:Geography	0.4

(a) Grapple Statements in A₁ Dataspace

gc:subject	gc:predicate	gc:object	gc:level
user:anna	A2:isLearning	subject:Malaysia	0.0
user:bob	A2:isLearning	subject:Japan	0.0
user:cindy	A2:isLearning	subject:Delft	0.0
user:donald	A2:isLearning	subject:Bangkok	0.0
user:frans	A2:isLearning	subject:Japan	0.0

(b) Grapple Statements in A₂ Dataspace

subject	predicate	object
subject:Malaysia	A2:isRelatedTo	<http://sws.geonames.org/1733045/>
subject:Japan	A2:isRelatedTo	<http://sws.geonames.org/993960/>
subject:Delft	A2:isRelatedTo	<http://sws.geonames.org/2757345/>
subject:Bangkok	A2:isRelatedTo	<http://sws.geonames.org/2757345/>

(c) Derived Triples in A₂ Dataspace

Figure 8 - Snapshot of A1 and A2 Dataspaces (partial view)

The creator of A₂ would like to suggest Wikipedia pages about the subject the students are currently taking to enhance their knowledge about the subject if they have good basic knowledge about Geography. She knows that application A₁ provides the basic Geography course, and therefore chooses to reuse data from A₁. She

applies for a dataspace subscription to A_1 and the creator of A_1 approves this subscription request. A_2 is able to query data in the A_1 dataspace. The creator of A_2 defines a GDR rule named "Get Wiki Page" as shown in Figure 9 that can be used to integrate data from four distributed data source to get the URLs of Wikipedia pages.

```

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" creator="http://localhost:8080/umf/client/2"
05     description="This rule retrieves Wikipedia page of subjects">
06     <gdr:premise dataspace="1">
07         <gc:subject?user</gc:subject>
08         <gc:predicate rdf:resource="http://apps.org/A1/hasKnowledge" />
09         <gc:object>http://subject.org/Geography</gc:object>
10         <gc:level>0.5</gc:level>
11     </gdr:premise>
12     <gdr:premise dataspace="2">
13         <gc:subject?user</gc:subject>
14         <gc:predicate rdf:resource="http://apps.org/A2/isLearning" />
15         <gc:object?subject</gc:object>
16     </gdr:premise>
17     <gdr:premise uri="http://localhost:8080/umf/ds/2" namedGraph=""
18         endpoint="http://localhost:8080/umf/rest/sparql/ds/2?client=2&token=123">
19         <gdr:pattern?subject
20             &lt;http://sakai.org/isRelatedTo&gt; ?geonameConcept</gdr:pattern>
21     </gdr:premise>
22     <gdr:premise uri="http://geonames.org/" namedGraph="http://geonames.org"
23         endpoint="http://localhost:8890/sparql">
24         <gdr:pattern?geonameConcept
25             &lt;http://www.w3.org/2002/07/owl#sameAs&gt; ?dbpediaConcept</gdr:pattern>
26     </gdr:premise>
27     <gdr:premise uri="http://dbpedia.org/" namedGraph="http://dbpedia.org"
28         endpoint="http://dbpedia.org/sparql" >
29         <gdr:pattern?dbpediaConcept
30             &lt;http://xmlns.com/foaf/0.1/page&gt; ?page</gdr:pattern>
31     </gdr:premise>
32     <gdr:consequent dataspace="2">
33         <gc:subject?user</gc:subject>
34         <gc:predicate rdf:resource="http://apps.org/A2/suggestedWikiPage" />
35         <gc:object?page</gc:object>
36     </gdr:consequent>
37 </gdr:rule>

```

Figure 9 - An Example of a GDR Rule in XML Syntax.

There are two dataspace premises and three external source premises defined in the GDR rule. The first dataspace premise (Lines 06 – 11) is used to determine the students who have passed the Geography subject using application A_1 with at least a 50% score, i.e. it would detect user:anna and user:cindy. The second dataspace premise (Lines 12 – 16) retrieves a set of Grapple statements whose *gc:predicate* is *http://apps.org/A2/isLearning*. These dataspace premises are joined, and the result of the join is as following.

user	subject
user:anna	subject:Malaysia
user:cindy	subject:Delft

Using this result, the external source premises are processed. For example, the bindings of variable *subject* that is one of the variables in the first external premise (Lines 17 – 19) are available. The values of the bindings of variable *subject* and the triple pattern specified in this premise are used to construct SPARQL queries that will be sent to the SPARQL endpoint of the premise. For the first external source premise, the following SPARQL query is constructed.

```

SELECT ?geonameConcept ?subject
WHERE {
  { ?subject <http://sakai.org/isRelatedTo> ?geonameConcept .
    FILTER (?subject = <http://subject.org/Delft> ) . }
  UNION
  { ?subject <http://sakai.org/isRelatedTo> ?geonameConcept .
    FILTER (?subject = <http://subject.org/Malaysia> ) . }
}

```

The result of this query is stored in the Temporary Repository for further processes.

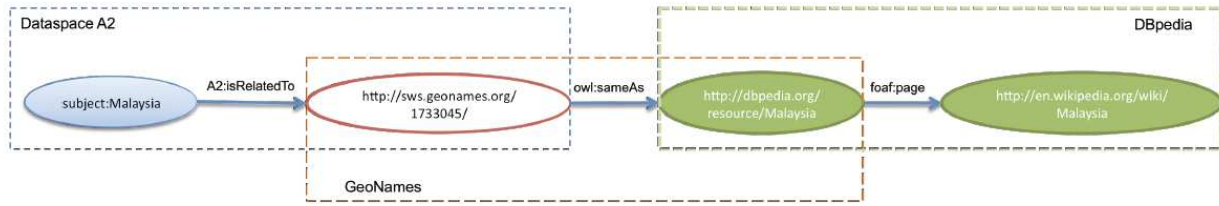


Figure 10 - Graph Patterns Across Three Different Data Sources.

gc:subject	gc:predicate	gc:object
user:anna	A2:suggestedWikiPage	<http://en.wikipedia.org/wiki/Malaysia>
user:cindy	A2:suggestedWikiPage	<http://en.wikipedia.org/wiki/Delft>

Figure 11 - Derived Grapple Statements.

Basically, the external source premises specify the graph patterns across three different data sources (namely, dataspace A_2 , GeoNames, and DBpedia) that must be matched in order to get the Wikipedia pages. For example, Figure 10 depicts the path from resource *subject:Malaysia* to resource *http://en.wikipedia.org/wiki/Malaysia*. The GDR rule in Figure 9 derives two Grapple statements as depicted in Figure 11.


4.2 Profile Browsing powered by GDR

To demonstrate the GDR capabilities a small GUMF client application has been developed that allows end-users to navigate through distributed profiles. GDR example rules similar to the previous section have been defined to highlight the benefits of GDR.

GUMF & GDR Interlinking, Integrating, and Enriching User Data


Select user:

fabian's profile



attribute	value	data
name	Fabian Abel	../predicate/name/
websites	http://www.l3s.de/~abel/	../predicate/homepage/ (target dataspace) and ../predicate/website/ (source dataspace)
cities	Hannover	../predicate/city/
interests	no interests available	../predicate/hobbies/ (target dataspace) and ../predicate/interest/ (source dataspace)

Countries related to fabian



Countries deduced from cities:

Figure 12 - User profile browsing demo: GDR rules are deactivated.

Figure 12 shows the front-end of the demo application⁸. It lists the profile attributes of the given user (*fabian*), but only the attributes that are available in the dataspace of the profile browsing demo application:

```
<rdf:Description rdf:about="http://www.grapple-project.org/umf/12263581">
  <rdf:type rdf:resource="http://www.grapple-project.org/grapple-core/Statement"/>
  <gc:subject rdf:resource="http://www.fabianabel.de/foaf.rdf#fabian"/>
  <gc:predicate rdf:resource="http://profile.org/city"/>
  <gc:object>Hannover</gc:object>
  <gc:level rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.7</gc:level>
  <gc:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
    2010-01-01T09:00:00.000+02:00</gc:created>
  <gc:creator>Application 1</gc:creator>
</rdf:Description>

<rdf:Description rdf:about="http://www.grapple-project.org/umf/1212672">
  <rdf:type rdf:resource="http://www.grapple-project.org/grapple-core/Statement"/>
  <gc:subject rdf:resource="http://www.fabianabel.de/foaf.rdf#fabian"/>
  <gc:predicate rdf:resource="http://profile.org/homepage"/>
  <gc:object rdf:resource="http://www.13s.de/@abel"/>
  <gc:level rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.7</gc:level>
  <gc:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
    2010-01-01T09:00:00.000+02:00</gc:created>
  <gc:creator>Application 1</gc:creator>
</rdf:Description>

<rdf:Description rdf:about="http://www.grapple-project.org/umf/2673893">
  <rdf:type rdf:resource="http://www.grapple-project.org/grapple-core/Statement"/>
  <gc:subject rdf:resource="http://www.fabianabel.de/foaf.rdf#fabian"/>
  <gc:predicate rdf:resource="http://profile.org/name"/>
  <gc:object>Fabian Abel</gc:object>
  <gc:level rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.7</gc:level>
  <gc:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
    2010-01-01T09:00:00.000+02:00</gc:created>
  <gc:creator>Application 1</gc:creator>
</rdf:Description>
```

In this example just the name, location and homepage of the user are listed while other profile information such as the personal interest are missing. The Google Map that should highlight the countries related to the user is empty, because the profile browsing application is not able to deduce that “Hannover”, the city (<http://profile.org/city>) associated with the given user. (<http://www.fabianabel.de/foaf.rdf#fabian>) is located in Germany. Appropriate GDR rules enable the profile browser to (a) obtain missing profile attributes from other sources and (b) prepare the available profile data so that it can be processed by the application and the Google Map respectively.

(a) Importing hobbies from another dataspace (including attribute mapping):

```
<gdr:rule xmlns:gc="http://www.grapple-project.org/grapple-core/"
  xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  name="Import Hobbies from Application 3"
  creator="http://127.0.0.1:8080/umf/client/1"
  id = "3"
  description="Import Hobbies from Application 3 - Slide 7">
  <gdr:premise dataspace="3">
    <gc:subject?user</gc:subject>
    <gc:predicate rdf:resource="http://application3.com/interest" />
    <gc:object?hobby</gc:object>
  </gdr:premise>
  <gdr:consequent dataspace="1">
    <gc:subject?user</gc:subject>
    <gc:predicate rdf:resource="http://profile.org/hobbies" />
    <gc:object?hobby</gc:object>
  </gdr:consequent>
</gdr:rule>
```

(b) Inferring countries from given cities:

```
<gdr:rule xmlns:gc="http://www.grapple-project.org/grapple-core/"
  xmlns:gdr="http://www.grapple-project.org/grapple-derivation-rule/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  name="Deduce Country with conflict resolution"
  creator="http://application1.com/"
  description="This rule derives the countries of the users - Slide 6">
  <gdr:premise dataspace="37">
    <gc:subject?user</gc:subject>
    <gc:predicate rdf:resource="http://profile.org/city" />
    <gc:object?cityName</gc:object>
    <gc:created?time</gc:created>
    <gdr:filter>MAX(?time) FOR ?cityName LIMIT 1</gdr:filter>
  </gdr:premise>
  <gdr:premise sourceURI="http://geonames.org/"
    endpoint="http://wisserver.st.ewi.tudelft.nl:8892/sparql"
    namedGraph="http://geonames.org">
```

⁸ <http://pcwin530.win.tue.nl:8080/grapple-gdr-demo/>

```

<gdr:pattern>
  ?city &lt;http://www.geonames.org/ontology#name&gt; ?cityName </gdr:pattern>
<gdr:pattern>
  ?city &lt;http://www.geonames.org/ontology#featureClass&gt;
    &lt;http://www.geonames.org/ontology#P&gt; </gdr:pattern>
  <gdr:pattern>
    ?city &lt;http://www.geonames.org/ontology#inCountry&gt; ?countryURI
  </gdr:pattern>
  <gdr:pattern>
    ?country &lt;http://www.geonames.org/ontology#featureClass&gt;
      &lt;http://www.geonames.org/ontology#A&gt; </gdr:pattern>
  <gdr:pattern>
    ?country &lt;http://www.geonames.org/ontology#featureCode&gt;
      &lt;http://www.geonames.org/ontology#A.PCLI&gt; </gdr:pattern>
  <gdr:pattern>?country &lt;http://www.geonames.org/ontology#inCountry&gt; ?countryURI
</gdr:pattern>
  <gdr:pattern>?country &lt;http://www.geonames.org/ontology#name&gt; ?countryName
</gdr:pattern>
</gdr:premise>
  <gdr:consequent dataspace="37">
    <gc:subject>?user</gc:subject>
    <gc:predicate rdf:resource="http://profile.org/country" />
    <gc:object>?countryName</gc:object>
  </gdr:consequent>
</gdr:rule>

```


Table 2 - GDR rules for (a) importing and mapping user data from external dataspace and (b) inferring additional knowledge by utilising external Web services.

Table 2 lists the corresponding GDR rules. Table 2(a) shows the GDR rule that queries the dataspace of another application (dataspace 3) for interests (<http://application3.com/interest>) and makes these interests available as “hobbies” (<http://profile.org/hobbies>) in the dataspace of the profile browser (dataspace 1). For inferring the location of a given city, a SPARQL query is sent to the GeoNames Web service (see Table 2(b), cf. Section 4.1.3). Activating these rules via the GUMF administration interface allows the profile browser to also display profile information (hobbies as well as some profile picture) from the other sources and allows for using the Google Maps feature (see Figure 13).

GUMF & GDR Interlinking, Integrating, and Enriching User Data


Select user:

fabian's profile



attribute	value	data
name	Fabian Abel	../predicate/name/
websites	http://www.i3s.de/~abel/	../predicate/homepage/ (target dataspace) and ../predicate/website/ (source dataspace)
cities	Hannover	../predicate/city/
interests	Semantic Web Personalization User Modeling Social Media Folksonomies Personalized Search	../predicate/hobbies/ (target dataspace) and ../predicate/interest/ (source dataspace)

Countries related to fabian



Countries deduced from cities: United States Federal Republic of Germany

Figure 13 - User profile browsing demo: GDR rules are activated.

4.3 Inferring User Profiles in cold-start situations

To further illustrate the benefits of user modelling and user profile reasoning with GUMF, it has been analysed how the distributed user modelling capabilities in combination with the contextualisation features (see Deliverable 2.2b) allow for the computation of recommendations particularly in situations where the so-

called *cold-start problem* is given. This problem occurs for example, whenever new users register to a system. The learning management systems connected to the Grapple infrastructure share profile information via GUMF so that even the LMSs that are infrequently used or do not have a large base of users have access to rich profile data (which was possibly produced in another LMS). However, still the consideration of external, distributed profile data aggregated by GUMF (cf. Deliverable 2.2b) might be essential and/or beneficial. This sub-section investigates how GUMF can infer profile information for a specific application by exploiting distributed profile data generated in other applications.

The focus is in particular on tag-based profiles, which are sets of weighted tags where the weight specifies the user's interest into the corresponding tag. The profile deduction task has been formalised as a prediction problem:

Tag prediction task. *Given a set of tags that occur in the tag-based profile of user u in system A , the task of the tag prediction strategy is to predict whether those tags will also occur in u 's profile in system B .*

The performance is measured using *precision* (= correctly classified as overlapping tags / classified as overlapping tags), which is the fraction of correctly classified tags, *recall* (= correctly classified as overlapping tags / overlapping tags), which is the fraction of relevant tags that have been identified and *f-measure* (= harmonic mean of precision and recall). The intention is not to find the best prediction algorithm, but to examine the impact of features extracted from GUMF profile aggregation. That is why a *Naive Bayes classifier* has been applied that is fed with different features. The benchmark tag prediction strategy (without profile aggregation) bases its decision on a single feature: (F1) overall usage frequency of t in system B . In contrast, the strategy that makes use of profile aggregation (i.e., distributed profile data) also applies (F2) u 's usage frequency of t in system A and (F3) size of u 's profile in system A .

For the analysis a dataset was used with 139 users who linked their Flickr, StumbleUpon and Delicious accounts. For these users, 78412 tag assignments that were performed on the 200 latest images (Flickr) or bookmarks (Delicious and StumbleUpon) have been crawled. Overall, users tagged more actively in Delicious than in the other systems: over 75% of the tagging activities originate from Delicious, 16.3% from StumbleUpon and 5% from Flickr. The usage frequency of the distinct tags shows a typical power-law distribution in all three systems, as well as in the aggregated set of tag assignments: while some tags are used very often, the majority of tags is rarely used or even just once.

On average, each user provided 564.12 tag assignments across the different systems. The user activity distribution corresponds to a Gaussian distribution: 26.6% of the users have less than 200 tag assignments, 10.1% have more than 1000 and 63.3% have between 200 and 1000 tag assignments. Interestingly, people who actively tagged in one system do not necessarily perform many tag assignments in another system. For example, none of the top 5% taggers in Flickr or StumbleUpon are also among the top 10% taggers in Delicious. This observation of unbalanced tagging behaviour across different systems again reveals possible advantages of profile aggregation for current tagging systems: given a sparse tag-based user profile, the consideration of profiles produced in other systems might be used to tackle sparsity problems.

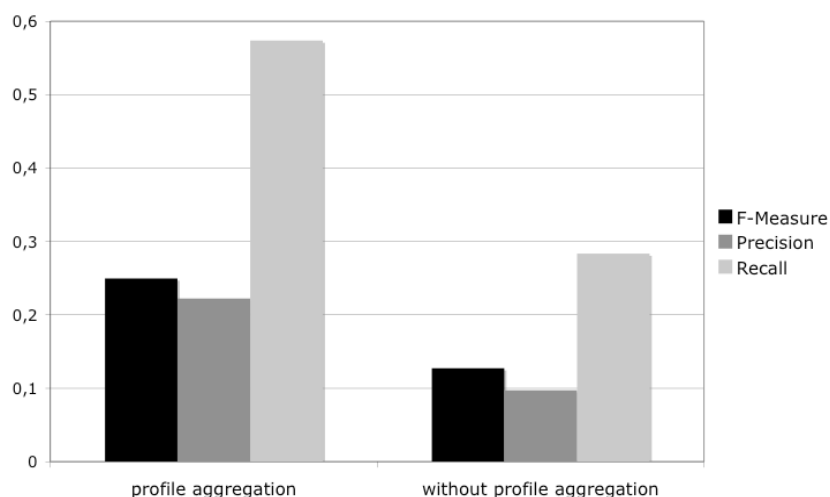


Figure 14 - Average performance of tag prediction with and without utilising distributed profiles (profile aggregation).

Figure 14 compares the average performance of both tag prediction strategies. For each of the 139 users and each service combination (Flickr → Delicious, Delicious → Flickr, StumbleUpon → Delicious, etc.) the strategies had to tackle the prediction task specified above. The benefits of the profile aggregation features

are significant. The profile aggregation strategy performs – with respect to the f-measure – 96.1% better than the strategy that does not benefit from profile aggregation (correspondingly, the improvement of precision and recall is explicit). It is also important to notice that the average percentage of overlapping tags is less than 4%. A random strategy that simply guesses whether tag t will overlap or not (probability of 0.5), would fail with a precision lower than 2%. On average, the profile aggregation strategy can detect 57.4% of the tags in system A that will also be part of the tag-based profile in system B.

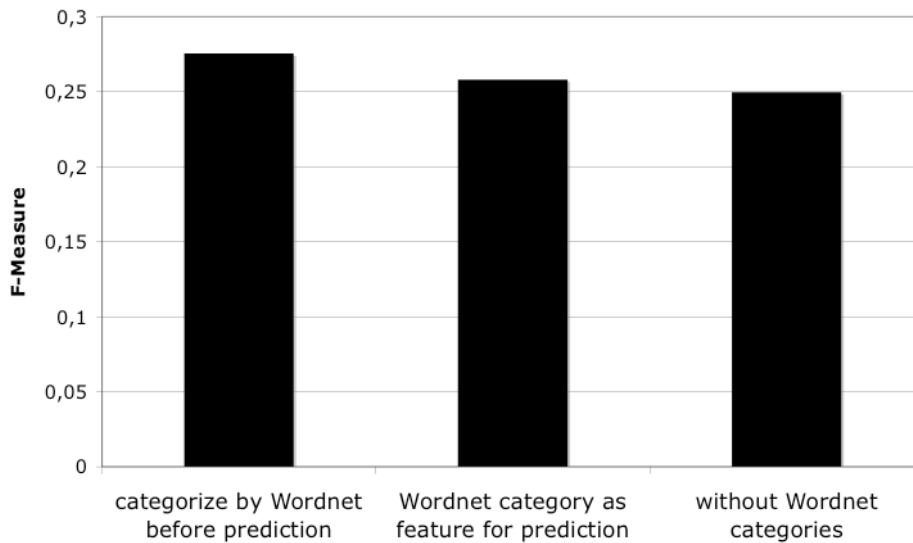


Figure 15 - Improving prediction performance (with profile aggregation) by means of Wordnet categorisation.

The performance can be improved additionally by clustering the tag-based profiles according to Wordnet categories that is possible with GUMF as well (cf. Deliverable 2.2b). Figure 15 shows that the consideration of Wordnet features – (F4) Wordnet category of tag t and (F5) relative size of corresponding Wordnet category cluster in u 's profile – leads to a small improvement from 0.25 to 0.26 regarding the f-measure. However, if tag predictions are done for each Wordnet cluster of the profiles separately, the improvement is considerably higher as the f-measure increases from 0.25 to 0.28.

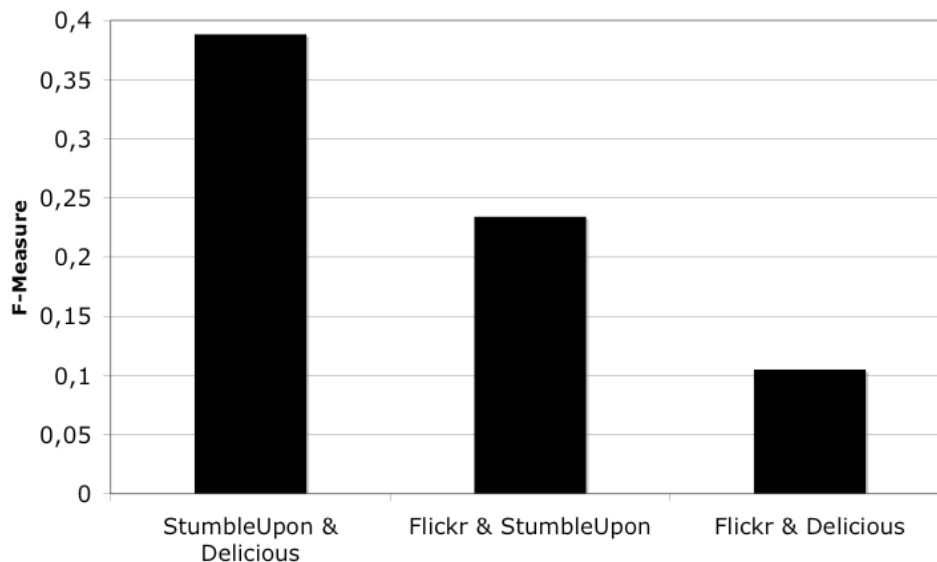


Figure 16 - Tag-based profile prediction performance for the different service constellations.

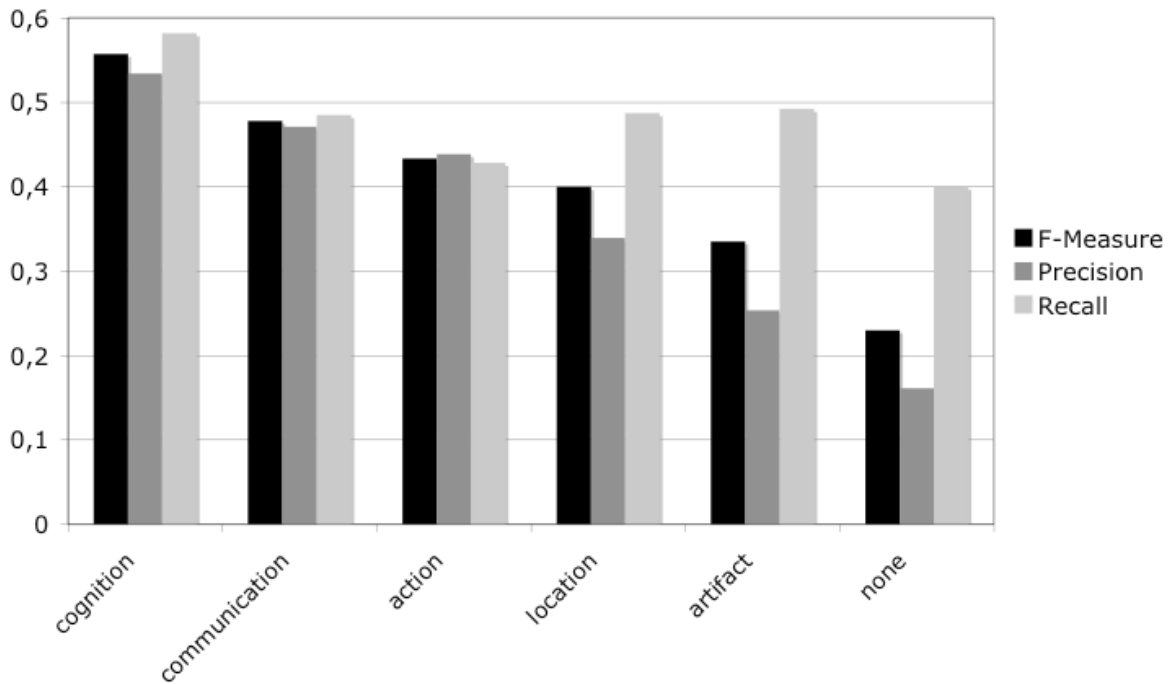


Figure 17 - Predicting the different facets of tag-based StumbleUpon profiles based on Delicious profiles.

Figure 16 shows the tag prediction performance (using features F1-5) focusing on specific service combinations. While tag predictions for Flickr/Delicious based on tag-based profiles from Delicious/Flickr perform quite weak, the predictions between Flickr and StumbleUpon show a much better performance (f-measure: 0.23). For the two bookmarking services, StumbleUpon and Delicious, which also have the highest average overlap for the individual tag-based profiles (i.e. u's Delicious and StumbleUpon profile are usually more similar than u's Flickr and StumbleUpon profile), tag prediction works best with f-measure of 0.39 and precision of 0.36. Figure 17 illustrates for what kind of tags prediction works best between Delicious and StumbleUpon. For tags that cannot be assigned to a Wordnet category (*none*), the precision is just 16% while recall of 40% might still be acceptable. However, given tags that can be mapped to Wordnet categories, the performance is up to 0.57 regarding f-measures. Given *cognition* tags (e.g., search, ranking) of a particular user *u*, the profile aggregation strategy, which applies the features F1-5, can predict the cognition tags *u* will use in StumbleUpon with a precision of nearly 60%: even if a user has not performed any tagging activity in StumbleUpon, one could recommend 10 cognition tags out of which 6 are relevant for user *u*.

In summary, the analysis of GUMF with respect to predicting application-specific profiles by means of user data generated in some other system shows that the consideration of profile data from other sources can be applied to solve cold start problems. In particular, it has been showed that the profile aggregation strategy for predicting tag-based profiles significantly outperforms the benchmark that does not incorporate profile features from other sources. Hence, systems that require certain user profile data benefit from GUMF as follows: (i) GUMF provides access to distributed profile data, (ii) GUMF enriches such profile data (e.g. with Wordnet metadata), and (iii) GUMF is able to deduce/predict these profile attributes that are required by the requesting system.

5 Conclusions and Future Work

This deliverable has extended the Grapple User Modelling Framework (GUMF) with the Grapple Derivation Rule language (GDR), and the reasoning capability of GUMF is extended and enhanced by allowing Web applications to exchange, reuse, integrate, and enrich the user data. This is achieved using the statements in Grapple dataspace but also openly accessible data published on the Web as Linked Data in a flexible and configurable way. This method has been implemented and integrated into the GUMF and applied in an e-learning setting where different e-learning systems (such as Moodle, AHA!, and CLIX) are connected. This method successfully supports the integration and enrichment of user data as demonstrated by a number of representative use cases. An analysis is presented that revealed that GUMF allows for the deduction of user profiles also in cold-start settings where systems usually suffer from incomplete/sparse user profiles.

As a continuation of this work, it is proposed to extend GDR specification so that not only Grapple statements can be derived but also RDF graphs. The plan is to improve the join heuristic of GDR Engine as currently the join process only follows the order of appearance in the rule. It would also be good to evaluate the GDR Engine in terms of performance and especially, scalability to explore the limit of this approach as Semantic Web reasoning applications typically run into scalability issues.

References

1. Abel, F., Henze, N., Herder, E., Krause, D.: Interweaving Public Profile Data on the Web. Proc. User Modeling, Adaptation and Personalization (UMAP) 2010, Hawaii (2010)
2. Abel, F., De Coi, J. L., Henze, N., Kösling, A., Krause, D., and Olmedilla, D.: Enabling Advanced and Context-Dependent Access Control in RDF Stores. In Proceedings of 6th International Semantic Web Conference (ISWC), Busan, Korea, pages 1-14, Springer (2007)
3. L. Aroyo, P. Dolog, G.-J. Houben, M. Kravcik, A. Naeve, M. Nilsson, and F. Wild. Interoperability in Personalized Adaptive Learning. Educational Technology & Society, 9(2):14-18, 2006.
4. R. Ghosh, and M. Dekhil. Mashups for Semantic User Profiles. In Proc. of the 17th International Conference on World Wide Web (WWW 2008), Beijing, China, Apr, 2008.
5. R. Ghosh, and M. Dekhil. I, me and my phone: identity and personalization using mobile devices, HP Labs Technical Report HPL-2007-184, Nov, 2007.
6. Google Mashup Editor. <http://editor.googlemashups.com/>
7. D. Heckmann, T. Schwartz, B. Brandherm, M. Schmitz, and M. von Wilamowitz-Moellendorff. GUMO - The General User Model Ontology. In Proc. of 10th International Conference on User Modeling (UM 2005), Edinburgh, UK, Jul, 2005.
8. Herder, E. and Marenzi, I. RTST Trend Report: lead theme Connecting Learners. Deliverable D1.2 of the Stellar Network of Excellence. March, 2010.
9. 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/>.
10. Kifer, M.: Rule Interchange Format: The Framework. In the Proc. of RR 2008, 2008.
11. T. Kuflik. Semantically-Enhanced User Models Mediation: Research Agenda. In Proc. Of 5th International Workshop on Ubiquitous User Modeling (UbiqUM'2008), workshop at IUI 2008, Gran Canaria, Spain, Jan, 2008.
12. McGuinness, D. L., van Harmelen, F. (eds.): OWL Web Ontology Language Overview, W3C Recommendation, Feb, 2004. <http://www.w3.org/TR/owl-features/>.
13. Microsoft Popfly. <http://www.popfly.com/>
14. Rule Markup Language Initiative. Rule Markup Language (RuleML). <http://ruleml.org>
15. C. Stewart, A. Cristea, I. Celik, and H. Ashman. Interoperability between AEH user models. In Proc. of the Joint International Workshop on Adaptivity, Personalization & the Semantic Web, workshop at Hypertext 2006, Odense, Denmark, Aug, 2006.
16. W3C OWL Working Group (eds.): OWL 2 Web Ontology Language Document Overview, W3C Recommendation, Oct, 2009. <http://www.w3.org/TR/owl2-overview/>.
17. Yahoo! Pipes. <http://pipes.yahoo.com/pipes/>