A Flexible Rule-Based Method for Interlinking, Integrating, and Enriching User Data

Erwin Leonardi[1], Fabian Abel[2], Dominikus Heckmann[3], Eelco Herder[2], Jan Hidders[1], Geert-Jan Houben[1]

[1] Delft University of Technology, PO Box 5031, 2600 GA Delft, the Netherlands
{e.leonardi, a.j.h.hidders, g.j.p.m.houben}@tudelft.nl
[2] L3S Research Center, Appelstrasse 9a, 30167 Hannover, Germany
{abel, herder}@l3s.de
[3] German Research Center for Artificial Intelligence, Saarbrucken, Germany
heckmann@dfki.de

Abstract. Many Web applications provide personalized and adapted services and contents to their users. As these Web applications are becoming increasingly connected, a new interesting challenge in their engineering is to allow the Web applications to exchange, reuse, integrate, interlink, and enrich their data and user models, hence, to allow for user modeling and personalization across application boundaries. In this paper, we present the Grapple User Modeling Framework (GUMF) that facilitates the brokerage of user profile information and user model representations. We show how the existing GUMF is extended with a new method that is based on configurable derivation rules that guide a new knowledge deduction process. Using our method, it is possible not only to integrate data from GUMF dataspaces, but also to incorporate and reuse RDF data published as Linked Data on the Web. Therefore, we introduce the so-called Grapple Derivation Rule (GDR) language as well as the corresponding GDR Engine. Further, we showcase the extended GUMF in the context of a concrete project in the e-learning domain.

Keywords: user modeling, user data integration, personalization, semantic enrichment, knowledge derivation

1 Introduction

Nowadays, numerous Web applications provide adapted and personalized 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 behavior. Explicit user data collection approaches rely on asking the user directly, for example, by using a survey form or by asking the user to give ratings to certain products. Implicit approaches imply the observation of the users' behavior: Web applications log and monitor the user behavior in order to construct a user model fitting with the personalization goals of the application. So, a key concern in developing such adaptive Web applications is to model the users and their behavior for achieving the personalization 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 modeling and personalization 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.

In essence, there are two ways to ensure interoperability between two applications and their user models: the *shared format* approach [5,20,22] and the *conversion* approach [6]. The shared format approach involves a lingua franca, an agreement between all parties on a common representation and semantic. An alternative approach, which is more flexible, involves conversion between the different applications' user models. Conversion allows for flexible and extensible user models, and for applications to join into a platform. Moreover, in contrast to a shared format approach, conversion is suitable for "open-world user modeling", which is not restricted to one specific set of systems [6].

Furthermore, we observe that there is a growing effort known as Linking Open Data[1] to make data interlinked and openly accessible on the Web by following the principles of Linked Data [7]. This effort opens opportunities to unlock a huge potential of data, including the user data. By *reusing* this interlinked data (such as DBpedia[2] and GeoNames[3]), various relationships between data can now be derived and discovered, and thus make data more meaningful and richer. Note that this data is published as RDF and accessible through a SPARQL endpoint. Nevertheless, the distributed nature of the RDF data sources creates a new interesting problem, that is, the problem of integrating RDF data from multiple distributed data sources. There are two possible solutions for this problem: *data centralization* and *query federation* [8,9,10]. The first approach provides a query service over a collection of data copied from different sources on the Web, while the second approach executes queries only on selected datasets that are part of the collection. This observation leads us to investigate how this distributed interlinked data can be *reused* and be beneficial for the purpose of exchanging, integrating, and enriching user data in the interoperable adaptive Web applications.

In this paper, we present the Grapple User Modeling Framework (GUMF) that facilitates the brokerage of user profile information and user model representations. We show how the existing GUMF [11] is extended with a new flexible rule-based method that enhances the reasoning capability of GUMF by allowing the applications to specify a "*recipe*" that guides the new knowledge deduction process in the

[1] http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
[2] http://dbpedia.org/
[3] http://geonames.org/

distributed setting using a rule language called Grapple Derivation Rule language (GDR). GDR extends GUMF with the flexibility for applications to flexibly define configurations that guide the user data integration and enrichment processes. Also, with GDR the applications are able not only to integrate data from GUMF dataspaces, but also to incorporate and reuse linked data published on the Web. Without GDR performing such processes are more complex and may not be efficient. To validate this, the implementation of the GUMF extended with GDR is applied in the GRAPPLE project[4] for user data in the e-learning domain.

The rest of this paper is organized as follows. Section 2 discusses the related work. We briefly introduce GUMF in Section 3. In Section 4, the Grapple Derivation Rule language (GDR) and the GDR Engine are presented. We also elaborate how GUMF is extended with GDR. Section 5 showcases the extended GUMF in the e-learning domain in the context of a concrete project. Finally, Section 6 concludes our discussion.

2 Related Work

In the user modeling research field, a host of approaches have been delivered to address the user model interoperability problem. There are basically two approaches: the *shared format* approach and the *conversion* approach. In the first approach, a common language for a unified user profile (*a lingua franca*) is needed. Examples of this approach are the General User Model Ontology (GUMO) [20] and Composite Capability/Preference Profiles (CC/PP)[5]. This approach is easily exchangeable and interpretable as there is no syntactic and semantic heterogeneity issue to be addressed [20]. 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* [21]. The conversion approach is more flexible and suitable for open and dynamic environments [6]. In this approach, a technique has to be developed for converting a user model of one application to another application. It should deal with the problem of syntactic and semantic heterogeneity. The 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. It is also possible that the mappings are incomplete because required information in one model is not available in the other model.

Furthermore, the Grapple Derivation Rule language builds upon existing rule languages such as the Rule Markup Language (RuleML) [18] defined by the Rule Markup Initiative. RuleML 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. RuleML itself covers the entire rule spectrum, from *derivation rules* to *transformation rules* to *reaction rules*, and thus can specify queries and inferences in Web ontologies, mappings between Web

[4] GRAPPLE is the acronym for an EU FP7 STREP Project denoting "Generic Responsive Adaptive Personalized Learning Environment" – http://www.grapple-project.org/

[5] http://www.w3.org/Mobile/CCPP/

ontologies, and dynamic Web behaviors of workflows, services, and agents. The Semantic Web Rule Language (SWRL) [15] 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 [16,17] with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language [18]. 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". The observation that there are currently many "rules languages" in existence in the web community lead to the Rule Interchange Format (RIF) which is a standard in development within the W3C Semantic Web Activity [19]. GDR is different from the existing rule languages at least for the following reasons. Firstly, it provides definitions of premise and consequent at the level of Grapple statements that constitute the lingua franca when interacting with GUMF. Secondly, it allows the integration of knowledge using multiple distributed data sources published as Linked Data on the Web.

To deal with the distributed nature of data sources published on the Web as RDF data, recently, there has been much research on the subject of integrating different RDF graphs into a single RDF graph and the related problem of querying distributed RDF data sources that were integrated into a single virtual RDF data source. Langegger et al. present in [10,12] the *SemWIQ* system that has a mediator-wrapper architecture and allows the integrated data to be queried with a subset of SPARQL and implements and optimizes these queries by translating them to an algebra called ARQ2. The notion of *networked graphs* is introduced by Schenk et al. in [13] where they discuss the problem of integrating different RDF graphs by defining SPARQL-based integration rules between them. The problem of optimizing a query that queries different external RDF data sources is discussed by Zemanek et al. in [14] which concentrates on minimizing communication cost by using semi-joins. The same problem is addressed by Hartig et al. in [9] which focuses on the subproblems of efficiently finding the data sources related to the query during query execution and efficiently executing the queries by using an iterator-based pipeline approach in its query evaluation plans. Finally, the *DARQ* system, described by Quilitz et al. in [8] allows the integration of distributed RDF data sources into a single virtual RDF data source by specifying which data is to be found in which external data source. It uses query-rewriting and cost-based query optimization to obtain efficient distributed query evaluation plans.

3 GUMF

The Grapple User Modeling Framework (GUMF) [11] enables systems to benefit from the multi-faceted user data traces that are distributed across different Web systems. GUMF provides generic user modeling functionality that is adaptable to the requirements of the individual systems that utilize it: it aggregates, contextualizes and models user data so that systems can easily incorporate the data without having to solve interoperability issues such as schema mapping. Further, GUMF together with its plug-ins feature reasoning capabilities for deducing new information about users

from their profile and activity data. In the context of the afore mentioned GRAPPLE project, GUMF is applied to provide user modeling functionality across e-learning application boundaries and thus it connects learning management systems such as Moodle, AHA!, and CLIX. In the remainder of this section we present the architecture and components of GUMF in more detail.

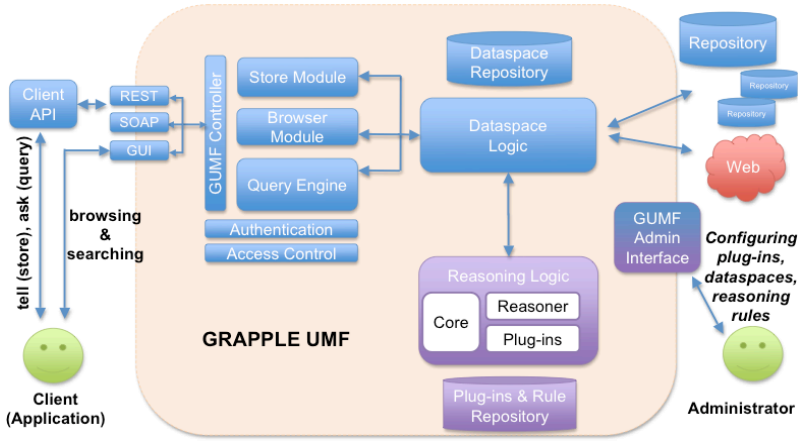

Fig. 1 GUMF Architecture

3.1 Architecture and Building Blocks

GUMF can be considered as an intelligent storage and reasoning engine that provides uniform access to distributed heterogeneous user data. **Fig. 1** shows its architecture. The blue elements at the top provide the essential, generic functionality of the framework; the purple components at the bottom provide generic as well as domain-specific plug-in and reasoning functionality.

Client applications can access GUMF either via a RESTful or SOAP-based API. Further, there is a Java Client API that facilitates development of GUMF client applications. Client applications mainly approach GUMF to store user information (handled by the *Store Module*) or to query for information (handled by *Query Engine*). By default, user profile information is modeled by means of Grapple statements (see below) that constitute the lingua franca when interacting with GUMF. Grapple statements are basically reified RDF statements about a user, enriched with DCMI metadata[6] for describing provenance details. The current GUMF implementation supports SPARQL [4] and SeRQL [2] queries as well as a pattern-based query language – *Grapple Query* language – that exploits the Grapple statement structure to specify what kind of statements should be returned.

Queries are executed on so-called dataspaces (*Dataspace Logic*) that logically bundle data that is possibly distributed across different sources on the Web, as well as offer reasoning functionality provided by different reasoners and plug-ins of the

[6] http://dublincore.org/documents/dcmi-terms/

Reasoning Logic. Dataspaces thus 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 dataspaces represent the part of GUMF that a certain client application is managing and responsible for, i.e., its own workspace. The Administrator of a GUMF client application can configure dataspaces and plug-ins via the *GUMF Admin Interface* (see **Fig. 1**). Activating or deactivating plug-ins and adjusting plug-ins and reasoning rules directly influence the behavior of dataspaces. Inspired by Web 2.0 practices, a key principle of GUMF is that dataspaces can be shared across different client applications. Therefore, clients can subscribe to other dataspaces, 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 dataspaces as well.

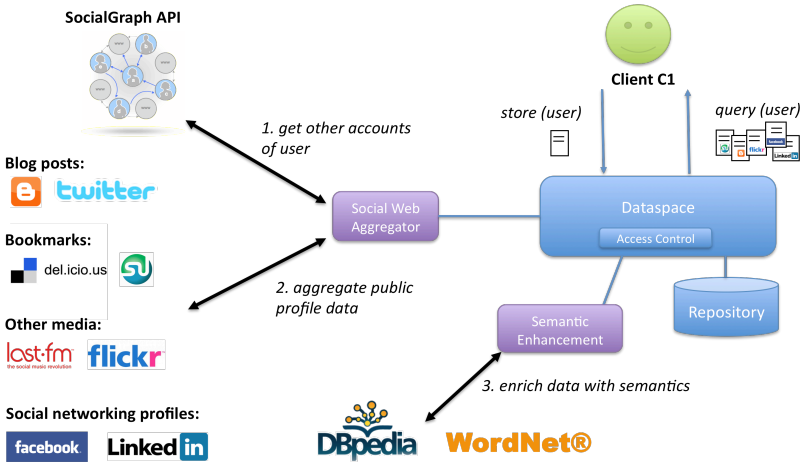

Fig. 2 User Modeling with GUMF Dataspaces

3.2 User Modeling with Intelligent Dataspaces

User modeling functionality of GUMF is embedded into dataspaces. In [1] we implemented the user modeling components that are applied to enrich data stored by client applications as depicted in **Fig. 2**. 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 modeled by means of Grapple statements, for example, C1 stores that a new user whose name is "Bob Myers" registered to C1. **Fig. 3** shows the corresponding statement in RDF/XML syntax.

Grapple statements are subject-predicate-object bindings enriched with metadata. They 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*)[7]. Storing a Grapple statement might trigger some plug-ins embodied into the dataspace. In **Fig. 2**, the *Social Web Aggregator* [1] obtains other accounts the user has via the Social Graph API[8]. 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 **Fig. 2**). In particular, the elements of the tag-based profiles [3] are mapped to DBpedia URIs that specify the semantic meaning of the tags and WordNet[9] categories are applied to cluster the profile [1]. Hence, based on the rather basic Grapple statement, which is listed in **Fig. 3**, 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. **Fig. 2**).

```
<?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#"
         xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
<gc:Statement rdf:about="http://grapple-project.org/2010-01-28-526341">
  <gc:subject redf: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>
```

Fig. 3 Grapple statement: Bob's name is Bob Myers.

The components that are plugged into dataspaces come in different flavors: Some plug-ins are black-box components while others are rule-based and are thus highly flexible. In [1], an example of black-box plug-ins is presented and in [11] a rule-based plug-in that is limited to integrate data *only* within a single Grapple dataspace is discussed. In the next section, we introduce the GDR language 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 dataspaces and data published as Linked Data on the Web.

[7] Note that gc:creator and gc:created are sub-properties of dc:creator and dc:created as defined by DCMI.

[8] http://socialgraph.apis.google.com

[9] http://wordnet.princeton.edu

4 GDR

In this section, we elaborate in details 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. We also discuss the GDR Engine that processes a GDR rule and derives new Grapple statements. Finally, we present how GUMF is extended with GDR.

4.1 GDR Definition

In the human readable syntax, a GDR rule has the form: $a \Rightarrow c$, where a and c are the antecedent and consequent of the rule, respectively, where a is a conjunction of premises written $p_1 \wedge ... \wedge p_n$. The premises of a GDR rule are classified into two types: *dataspace premises* and *external source premises*. A dataspace premise describes conditions over a Grapple dataspace in the form of a pattern-based Grapple Query. An external source premise specifies conditions in the form of 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. It specifies the subject, predicate, and object properties of the Grapple statements, and optionally the 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 following.

Definition 1. **[Dataspace Premise]** A dataspace premise d is a 2-tuple (ds, f), where ds is the Grapple dataspace identifier, and f is 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, and T is a basic graph pattern with at least one triple pattern. A set of external source premises is defined as E.

Definition 3. **[Consequent]** A consequent c is a dataspace premise (ds, f), where f is defined for at least *gc:subject*, *gc:predicate*, and *gc:object*, and at most also *gc:level*.

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 $p_1 \wedge ... \wedge p_n$, where $p_i \in (D \cup E)$ and $n > 0$,

- *c* is the consequent of the rule such that all variables in *c* appear in at least one premise of *A*.

In Section 4.3, we present the XML serialization format of GDR by an example. The next section introduces the engine that interprets and enforces given GDR rules.

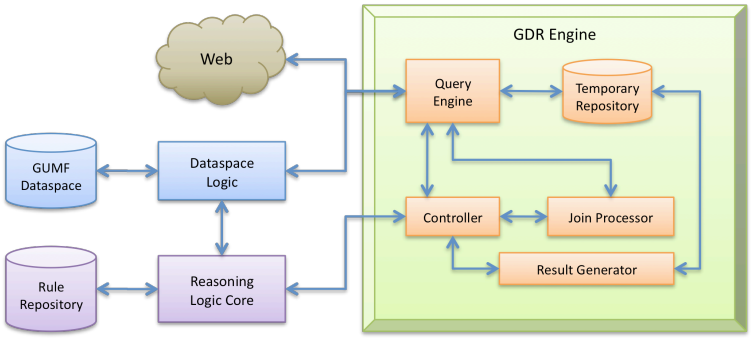

Fig. 4. The Architecture of GDR Engine

4.2 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. **Fig. 4** depicts its architecture and interactions with other GUMF modules. The GDR Engine consists of five components: the *Controller*, the *Query Engine* (*QE*), the *Join Processor* (*JP*), the *Result Generator* (*RG*), and the *Temporary Repository* (*TR*).

The Controller manages the whole process happening inside the GDR Engine. It receives requests from the GUMF Reasoning Logic Core. It also utilizes 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. 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*. The *TR* component is an RDF repository that is used to store the RDF triples of intermediate results (e.g. join results). The *JP* component is responsible in 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*. The *RG* analyzes 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.

When a client application issues a pattern-based query *q* to GUMF, 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 first evaluates all the dataspace premises of *r* and maintains the result of each dataspace premise evaluation in the *TR* by utilizing the *QE*. 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 there is at least one dataspace premise evaluation that returns no result, then 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*.

Next, the GDR Engine exploits the *JP* to join the dataspace premises using the results stored in the *TR*. If two dataspace premises d_1 and d_2 share the same variables, then 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, then 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*. Optimizing 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 optimization issue will be investigated in the future.

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, then the *QE* is exploited to fetch the data of previously processed premises stored in the *TR*, to construct SPARQL queries based on the specified triple patterns and fetched data, and to send these queries to the SPARQL endpoint of *e*. The results of these queries are maintained by the TR. Note that the results can be used in processing other external source premises. 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 constructs a SPARQL query only based on 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, then 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, then we only need to check whether this premise returns any results. The constructed query then is sent to the SPARQL endpoint of e_k, and the result is stored in the *TR*.

After all premises are processed, then the Controller sends request to the *RG* that generates a set of new Grapple statements. The *RG* analyzes 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 modeled as Grapple statement and sent to the Controller, which subsequently sends it to the Reasoning Logic Core.

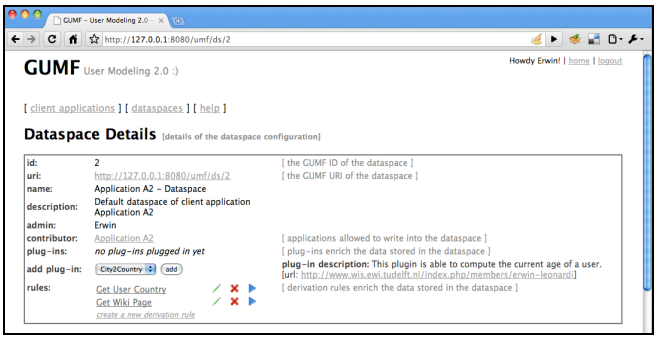

Fig. 5. GUMF Administrator Page

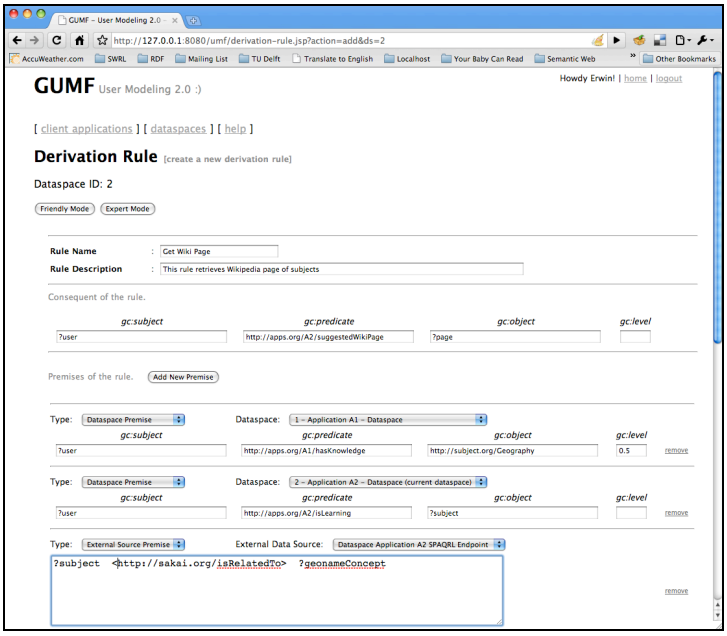

Fig. 6. GDR Rule Creation Page: Friendly Mode

4.3 Extending GUMF with GDR

We implemented the GDR Engine in Java. For the Temporary Repository component, we choose to base our implementation on the open-source RDF framework Sesame[10]. 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. For this purpose, there are several components in GUMF that have

[10] http://www.openrdf.org/

to be extended. The "Plug-in & Rule Repository" of GUMF is extended to be able to store the GDR rules. We added a feature in the Reasoning Logic Core component to detect which GDR rules in the dataspace are relevant for a Grapple query sent by GUMF client. This can be done by analyzing the consequent of the rules. The Reasoning Logic Core has to be able to communicate with the GDR Engine. Furthermore, the GUMF administrator page is extended such that it shows the list of specified GDR and a hyperlink to the GDR rule creation page (**Fig. 5**). There are two ways of creating a GDR rule: *friendly mode* and *expert mode*. In the friendly mode (**Fig. 6**) the dataspace administrator specifies the rule by filling up provided form fields. In the expert mode, the administrator has to type the rule in XML format.

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 A_1 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 A_2 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 Tripples in A_2 Dataspace

Fig. 7 The Snapshot of A_1 and A_2 Dataspaces (partial view)

5 Use Case

In this section, we showcase the extended GUMF in the e-learning domain in the context of the GRAPPLE project. GDR applied in GUMF allows for distributed user modeling across e-learning systems. Suppose there are two adaptive e-learning applications, namely, A1 and A2 that use GUMF. A1 that is a Moodle-based application is used for a basic Geography course, and A2 that is an AHA!-based application is used for an Urban Geography course. **Fig. 7**(a) depicts a set of Grapple statements in the A1 dataspace. **Fig. 7**(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 **Fig. 7**(c).

The creator of A_2 would like to suggest Wikipedia pages about the subject that the students are currently taking for enhancing their knowledge about the subject if they have good basic knowledge about Geography. She knows that application A_1 provides the basic Geography course, and thus chooses to reuse data from A_1. She applies for a dataspace subscription to A_1 and the creator of A_1 approves this subscription request. Thus, A_2 is able to query data in the A_1 dataspace. Moreover, the creator of A_2 defines a GDR rule named "Get Wiki Page" as shown in **Fig. 8** 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=""
            endpoint="http://localhost:8080/umf/rest/sparql/ds/2?client=2&amp;token=123">
18         <gdr:pattern>?subject
               &lt;http://sakai.org/isRelatedTo&gt; ?geonameConcept</gdr:pattern>
19     </gdr:premise>
20     <gdr:premise uri="http://geonames.org/" namedGraph="http://geonames.org"
           endpoint="http://localhost:8890/sparql">
21         <gdr:pattern>?geonameConcept
               &lt;http://www.w3.org/2002/07/owl#sameAs&gt; ?dbpediaConcept</gdr:pattern>
22     </gdr:premise>
23     <gdr:premise uri="http://dbpedia.org/" namedGraph="http://dbpedia.org"
            endpoint="http://dbpedia.org/sparql" >
24         <gdr:pattern>?dbpediaConcept
               &lt;http://xmlns.com/foaf/0.1/page&gt; ?page</gdr:pattern>
25     </gdr:premise>
26     <gdr:consequent dataspace="2">
27         <gc:subject>?user</gc:subject>
28         <gc:predicate rdf:resource="http://apps.org/A2/suggestedWikiPage" />
29         <gc:object>?page</gc:object>
30     </gdr:consequent>
31 </gdr:rule>
```

Fig. 8 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. The second one (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 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. Hence, 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.

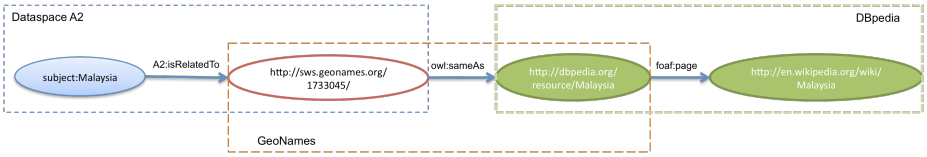

Fig. 9 Graph Patterns Across Three Different Data Sources

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>

Fig. 10 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, **Fig. 9** depicts the path from resource *subject:Malaysia* to resource *http://en.wikipedia.org/wiki/Malaysia*. The GDR rule in **Fig. 8** derives two Grapple statements as depicted in **Fig. 10**.

6 Conclusion

In this paper, we have extended the Grapple User Modeling Framework (GUMF) with the Grapple Derivation Rule language (GDR), and thus the reasoning capability of GUMF is extended and enhanced by allowing Web applications to exchange, reuse, integrate, and enrich the user data using not only data in Grapple dataspaces, but also openly accessible data published on the Web as Linked Data in a flexible and configurable way. We have implemented and integrated our method into the GUMF and applied it in an e-learning setting where different e-learning systems (such as Moodle, AHA!, and CLIX) are connected. Our method successfully supports the integration and enrichment of user data as demonstrated by a representative use case.

As a continuation of our work, we plan to extend GDR specification such that we can derive not only Grapple statements, but also RDF graphs. We also plan to improve the join heuristic of GDR Engine as currently the join process only follows the order of appearance in the rule. We also would like to evaluate the GDR Engine in terms of performance and, especially, scalability to explore the limit of our approach as Semantic Web reasoning applications typically run into scalability issues.

Acknowledgments. This work was partially supported by the European 7th Framework Program project GRAPPLE ("Generic Responsive Adaptive Personalized Learning Environment"): http://www.grapple-project.org.

References

1. Abel, F., Henze, N., Herder, E., Krause, D.: Interweaving Public Profile Data on the Web, Technical Report, L3S Research Center, Hannover, Germany, 2010.
2. Broeskstra, J., Kampman, A.: SeRQL: A Second Generation RDF Query Language. SWAD-Europe Workshop on Semantic Web Storage and Retrieval, Vrije Universiteit, Amsterdam, Netherlands, 2003.
3. Firan, C.S., Nejdl, W., Paiu, R.: The Benefit of Using Tag-based Profiles. In: Proc. of LA-WEB 2007, Washington, DC, USA, IEEE Computer Society, 2007.
4. Prud'hommeaux, E., Seaborne, A.: SPARQL Query Language for RDF. W3C Recommendation, January 2008. Available at: http://www.w3.org/TR/rdf-sparql-query/
5. Stewart, C., Celik, I., Cristea, A., Ashman, H.: Interoperability between aeh user models. In Proc. of APS 2006, 2006.
6. Aroyo, L., Dolog, P., Houben, G., Kravcik, M., Naeve, A., Nilsson, M., Wild, F.: Interoperability in personalized adaptive learning. J. Educational Technology &Society 9 (2) (2006) 4–18.
7. Berners-Lee, Tim: Design Issues: Linked Data. (2006) http://www.w3.org/DesignIssues/LinkedData.html.
8. Quilitz, B., Leser, U.: Querying Distributed RDF Data Sources with SPARQL. In Proc. of ESWC 2008, 2008.
9. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL Queries over the Web of Linked Data. In Proc of ISWC 2009, 2009.
10. Langegger, A., Wöß, W., Blöchl, M.: A semantic web middleware for virtual data integration on the web. In Proc of ESWC 2008, 2008.
11. Abel, F., Heckmann, D., Herder, E., Hidders, J., Houben, G.-J., Krause, D., Leonardi, E., van der Sluijs, K.: A Framework for Flexible User Profile Mashups. In the Proc. of the APWEB 2.0 2009 Workshop in conjunction UMAP 2009, 2009.
12. Langegger, A.: Virtual data integration on the web: novel methods for accessing heterogeneous and distributed data with rich semantics. In Proc. of iiWAS'08, 2008.
13. Schenk, S., Staab, S.: Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web. In Proc. of WWW '08, 2008.
14. Zemanek, J., Schenk, S., Svatek, V.: Optimizing sparql queriesover disparate rdf data sources through distributed semi-joins. In ISWC 2008 Poster and Demo Session Proceedings. CEUR-WS, 2008.
15. Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. http://www.w3.org/Submission/SWRL/.
16. McGuinness, D. L., van Harmelen, F. (eds.): OWL Web Ontology Language Overview, W3C Recommendation, Feb, 2004. http://www.w3.org/TR/owl-features/.
17. W3C OWL Working Group (eds.): OWL 2 Web Ontology Language Document Overview, W3C Recommendation, Oct, 2009. http://www.w3.org/TR/owl2-overview/.
18. Rule Markup Language Initiative. Rule Markup Language (RuleML). http://ruleml.org/
19. Kifer, M.: Rule Interchange Format: The Framework. In the Proc. of RR 2008, 2008.
20. Heckmann, D., Schwartz, T., Brandherm, B., Schmitz, M., von Wilamowitz-Moellendorff, M.: GUMO - The General User Model Ontology. In Proc. of UM 2005, 2005.
21. Kuflik, T.: Semantically-Enhanced User Models Mediation: Research Agenda. In Proc. of UbiqUM'2008 Workshop at IUI 2008, Gran Canaria, Spain, 2008.
22. Finkel, J. R., Grenager, T., Manning, C.: Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. In Proc. of ACL 2005, 2005.