WP 4 - RE Specifications

Table of Content

1. Authors 2
2. Document Purpose 2
3. Artefacts or Useful Links 2
4. Introduction 3
5. System Architecture 4
6. Adoption of a Common Data Model 4
7. Mahara Mapping to Common Data Model 6
   7.1 Mahara Reduced Model 6
   7.2 Mahara Object Mapping 6
   7.3 Mahara Activity Mapping 12
      7.3.1 Verbs 13
      7.3.2 Mahara event mapping 13
      7.3.3 Access Activity 15
8. Moodle Mapping to Common Data Model 16
   8.1 Moodle reduced model 16
   8.2 Moodle Object Mapping 16
   8.3 Moodle Activity Mapping 19
9. Common API For Data Exchange 22
10. Recommendation Algorithm 24
11. Reference 25

1. Authors

Zhou Lei
Sandy El Helou
Laurent Opprecht
Jacques Monnard
Denis Gillet
Christophe Salzmann
Omar Benkacem
Patrick Roth
Gérald Collaud
François Jimenez

2. Document Purpose

The Recommender System - RS for short - collects user activities in order to provide a personalized and contextualized recommendation of actors (people), assets (resources) or activity spaces.

The current implementation of the RS is integrated with Graasp. As such, for the moment, it only monitors Graasp users and provides recommendation about Graasp objects.

The mid-term purpose is to make of the RE an independent piece of software able to exploit data across different applications, and as a result return a recommended list available to these various applications. The immediate objective is to enable monitoring and recommendation for Moodle and Mahara. For this to happen, **a common data model and an API to exchange the data** has to be developed between the RS and these platforms. This documents describes the requirements for achieving this objective.

3. Artefacts or Useful Links

#	url	Title	Comments
1	http://opensocial-resources.googlecode.com/svn/spec/2.0/Social-Data.xml	opensocial-social-data-specification-2-0	
2	http://graaasp.epfl.ch	Graasp	
3	http://www.opensearch.org/Home	OpenSearch	OpenSearch is a collection of simple formats for the sharing of search results.
4	http://en.wikipedia.org/wiki/Resource_Description_Framework	Resource Description Framework in Wikipedia	Resource Description Framework

5	http://www.w3.org/TR/owl-features/	OWL	
6	http://fm.ea-tel.eu/fm/fmm.php?pwd=81156c-27728	Meeting 4.11.2011	
7	http://dbpedia.org/About	DB Pedia	
8	http://www.w3.org/RDF/Validator/	RDF W3 Validator	
9	http://www.w3.org/TR/rdb2rdf-ucr/	Use Cases and Requirements for Mapping Relational Databases to RDF	
10	http://fr.wikipedia.org/wiki/Dublin_Core	Dublin Core	
11	http://xmlns.com/foaf/spec/	Friend of Friend specification.	Vocabulary used to described people.
12	http://protege.stanford.edu/	Protégé. Application to create ontologies/RDF Schemas/OWL.	

4. Introduction

“Today, more than anytime before, the society is challenged to constantly and actively acquire knowledge in order to stay up-to-date. Moreover, it is confronted with adverse information overload effects such as stress, anxiety, and reduced work efficiency at a personal as well as an organizational level. Personalized recommender systems are instrumental in overcoming the problem of information overload as they help online users find relatively interesting information, services and products. ”[1]
In PLEs, personalized recommender systems have been proved to be very efficient for reducing user efforts on getting useful information. While users benefit from recommender systems on PLE platforms, it seems the old way of recommending resources on a single platform is not sufficient to satisfy user requirements [2]. In order to enable users to get information without switching between different PLE platforms, we propose a federated recommender system that exploits data fetched from several PLE platforms and gives user recommendation based on distributed contents, encouraging by that knowledge sharing and collaboration beyond institutional boundaries.

The aim of this deliverable is to describe the common data model and related API standards to be adopted between different PLE platforms used in the project and the RS.

The UNIGE elearning infrastructure is primarily made of two LMSs – Moodle and Dokeos – an ePortfolio – Mahara – and a streaming platform – Mediaserver. Dokeos is the most widely used LMS accounting for about 90% of all users. UNIGE has migrated Moodle to version 2.0 during summer 2011, and Dokeos to Chamilo (new version of Dokeos) in the near future. Developments will thus target Moodle 2.0 and Chamilo.

With the implementation of this project, users can get recommendation in an easy and convenient way. They can get the recommended resources on different platforms without switching between these platforms. Also we can encourage the exchange of useful public resources beyond platform and institution boundaries.

5. System Architecture

As Figure 1 shows, we design a recommendation engine that collects all the information from different systems and platforms. The information to be collected includes users' action, the object that would be involved in the action and some other attributes concerned with the action.

After the recommendation engine collects all the information, it would parse them and make up (for the first time) or update a graph that contains the relationship between all the users and objects in these systems and platforms according to the different action type.

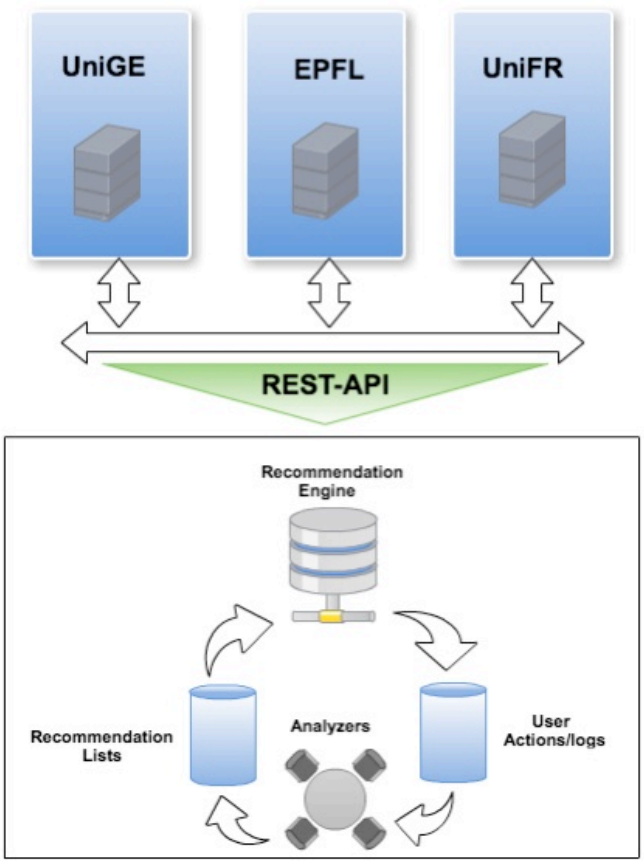

Figure 1 the architecture of the RE

6. Adoption of a Common Data Model

As mentioned before, the recommender system should exploit data belonging to different applications or platforms Considering that data models, naming conventions, and formats vary from user to user and network to network"[3], i becomes crucial to adopt a standard data representation and exchange format across different platforms.

Each application has a different object model. This is the case for both Mahara and Moodle but it is more generally the case for any application. To be able to exchange data with the RE, front-end platforms should communicate using a common vocabulary. This is achieved by mapping the static object model of each application to a common data model. The reference model chosen is the 3A interaction model adopted in Graasp and used to communicate with the RE [4]. The model is based on the following few constructs:

- Actors: now refers to users or people. Initially, the concept of actors integrated also the notion of Applications (and more generally any entity capable of initiating an event). Later, a clear separation between human actors and applications was adopted.
- Assets: refers to different kinds of resources (files, presentations, videos, wikis) created and shared among actors.
- (Activity) Spaces: refers to containers where one or more actors share assets and apps

The model has been chosen for its simplicity making it easier to map complex types to a few entities.

Graasp reduced model
Diagram shows the most important object in Graasp as well as their relationships.

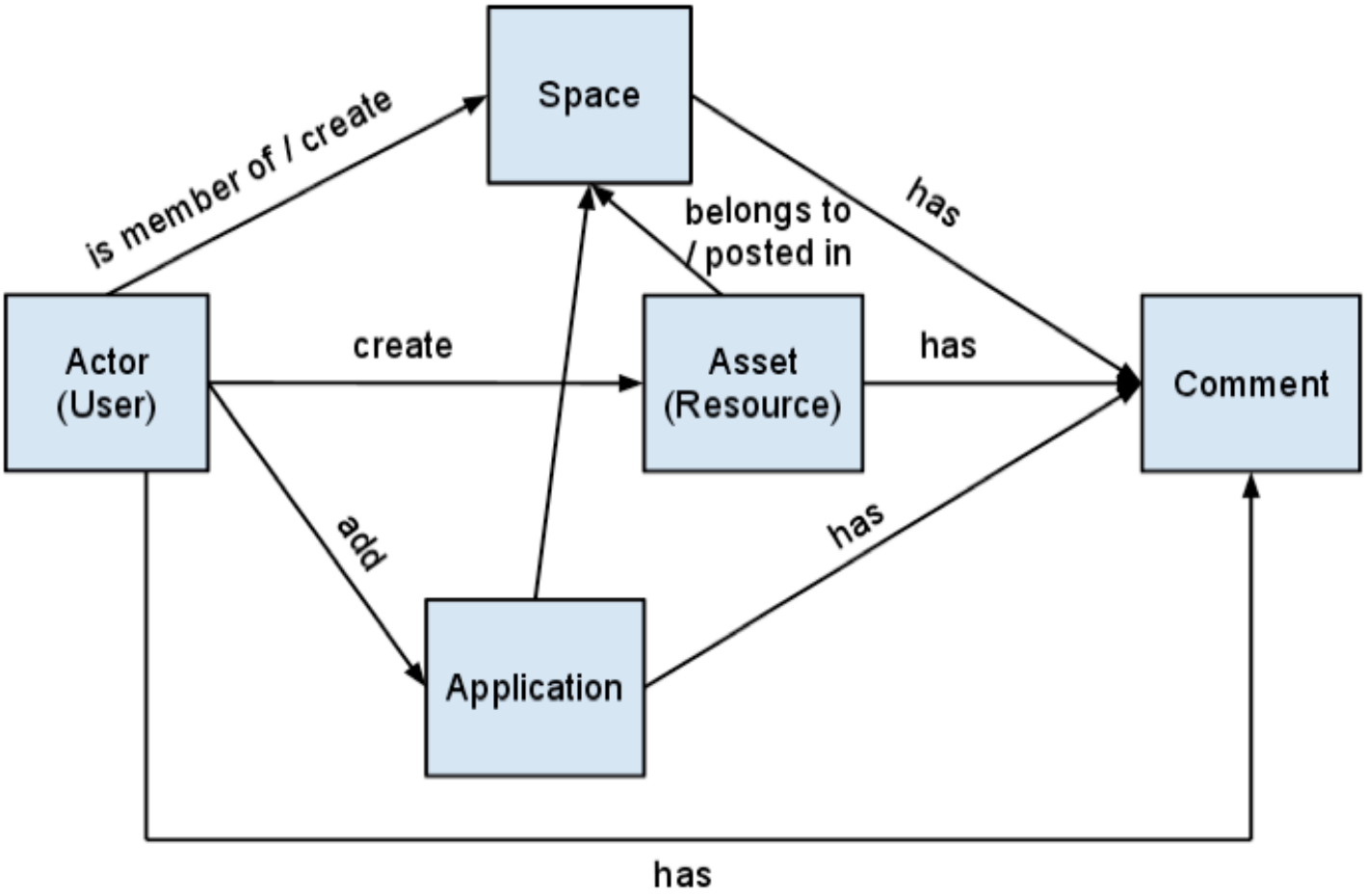

Figure 2 Graasp reduced model

The following sections describe the detailed mapping from Mahara and Moodle to the 3A interaction model used ir Graasp.

7. Mahara Mapping to Common Data Model

7.1 Mahara Reduced Model

Below is a simplified diagram showing the most important object types in Mahara as well as their relationships.

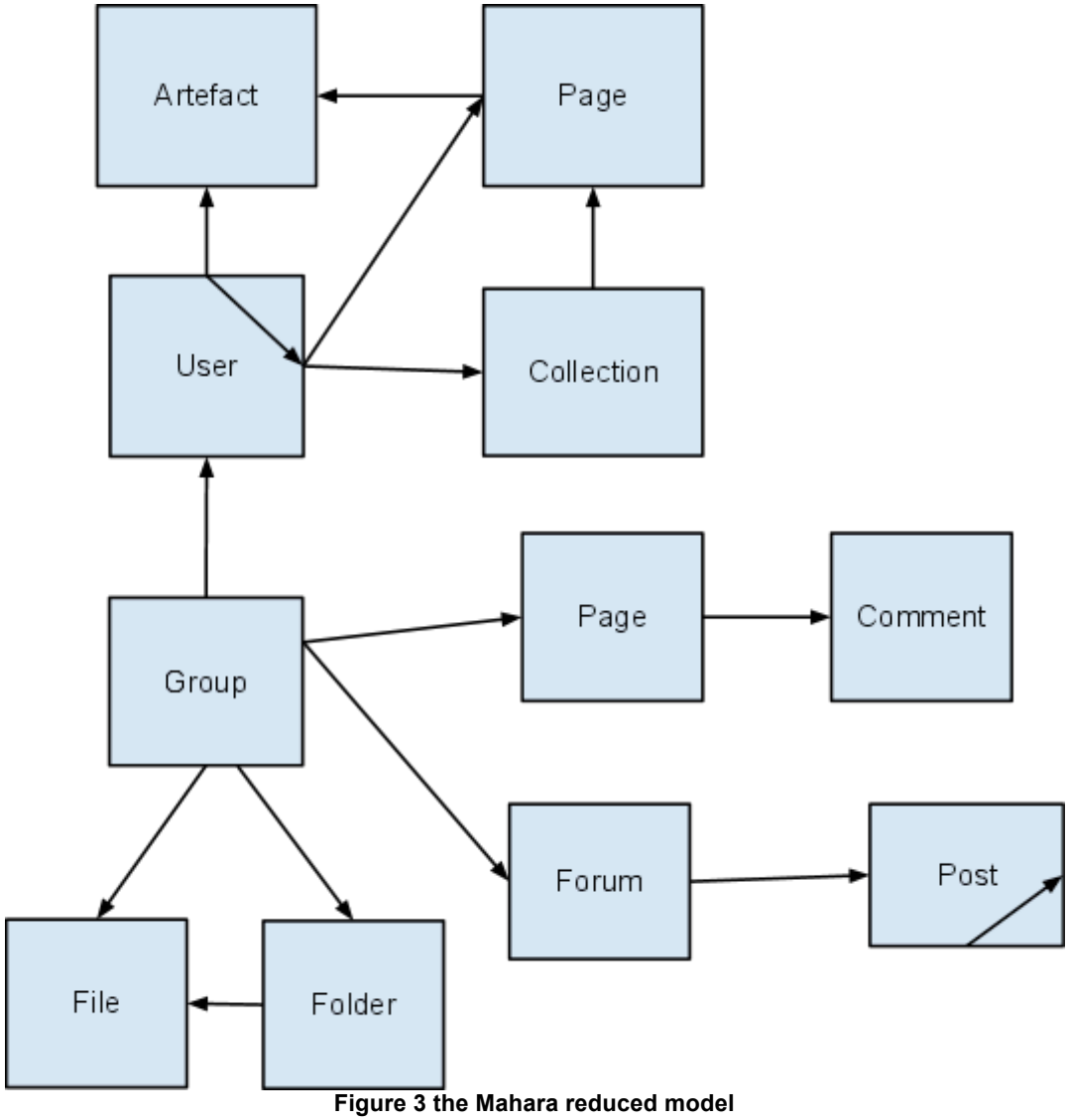

Figure 3 the Mahara reduced model

7.2 Mahara Object Mapping

All Mahara objects share a set of common properties. Those properties are:

id: Object's global unique id. This is the application id. It must be globally unique. Required.
pid: Object's public id. For actors it is the Shibboleth Unique Id. Required for actors.

dataset: Dataset uri. I.e. the url of the provider: www.dokeos.unige.ch, www.moodle.unifr.ch, etc. This is somewhat implied by the query url yet this provides an opportunity to clearly identify the root source - for example to limit the search to a specific dataset. Required.
name: Name of the object. Optional.
description: Description for the object. Can contain HTML. Optional.
url: Link to access the object. May not be unique to an object as some may be accessed through the same page. Required.
datastream: Direct url to the data stream. While the url field point to the UI where the user should navigate the stream field point to the resource's data. For a file this may be the direct download link. Maybe used to parse content and extract tags for recommendation. Should not be displayed to the end user through the recommendation interface as content may be protected. If provided the url should be accessible by the recommendation engine. Optional.
data[1]**:** Free text describing the object. Possibly in html format. Maybe used to extract keywords for recommendation. Should not be displayed by the Re to the end user as content may be protected. Optional.
thumbnail: Link pointing to an thumbnail image. Optional.
created: Created time. Optional. Format ISO 8601, 970-01-01T00:00:00+00:00.
modified: Modified time. Optional. Format ISO 8601, 970-01-01T00:00:00+00:00.
tags: List of tag. Optional.
category: Object's category. String with forward slashes as separators. I.e. Science/Mathematics/Numerical Analysis. Optional.
type: The RE object's type. One of Actor, Asset, Space. Required.
kind: The object application's type: course, forum, view, file, etc. Useful to change the recommendation display and allow user to better identify an asset. Application specific. Optional. Note that a possibly better scheme could be to add a schema and derive the application's type from one of the AAA type. Yet this would only be possible for some transmission protocols - rdf, xml.

The default format for dates is ISO 8601, i.e. 970-01-01T00:00:00+00:00.

The following table describes the mapping between Mahara and RE objects.

object in Mahara	**object in RE**	**Comments**	**Data**
User	Actor	Users are people registered with Mahara. Users can have friends, create views and collections. Users can register to groups. Users can create content/artefacts of different types: files, folders, blogs, etc. Yet this content is always private and can only be shared through views.	**id**: User id. The application's id. **pid**: User public id. I.e. the Shibboleth Unique Id. Not the application's id. Required. **dataset:** application uri **name**: Display name **description**: Short introduction about the person **url**: Link to the home/profile page **thumbnail:** Link to a thumbnail image **type=**actor: RE object's type **atype=**actor **Relationships** **Actors**: User's friends

[1] Considering to the amount of information transferred to the RE, maybe we should give some keywords instead of transferring the whole file. The solution is to be decided.

			Assets: None? Assets belonging to the user. **Spaces:** Collections that are shared. Groups the users participates in. Views that are shared.
View	Space	Mahara makes a clear distinction between content - artefacts - and presentations - views. Artefacts are used to create content. Views are used to display content. A view is a web page made of blocks. Different type of blocks can be added to a view. Some of the blocks are used to display artefacts - files, blogs, etc. - others are used to directly enter content - html, url, etc. Views by default are private to their owners: a user, a group, an institution or the Mahara site. Views can be shared with others by granting access rights. Different access rights exist. A view can be shared with a group, a user, through a secured url, etc.	**id**: the view id. Application's unique id. **dataset:** application uri **name**: Name of the view **description**: View's description **tags**: list of tags **url**: Link to access the view **type**=space: RE object's type **atype**=view **Relationships** **Actors**: View owner **Assets:** View's artefacts. I.e. artefacts displayed in this view. **Spaces:** None
View Comment	Comment	View's comments are messages posted on views. They can be private or public. They form a flat list. That is is it is not possible to reply to a comment. Comments have a body but do not have a title. Files can be attached to comments.	**id**: the commentid. Application's unique id. **dataset:** application uri **name**: Title of the comment. That is the string "comment" since view's comments do not have a title. **description**: Comment **url**: Link to access the view being commented **type**=comment: RE object's type **atype**=comment **Relationships** **Actors**: Author of the post **Assets:** None. **Spaces:** None.
Collection	Space	A collection is an ordered list of views sharing the same access rights. A collection has only one owner - a user. A collection is not displayed directly. Instead views belonging to the collection are displayed with an optional navigation menu.	**id**: the collection id. Application's unique id. **dataset:** application uri **name**: Name of the collection **description**: Collection's description **url**: Link to the first view/page in the collection **type**=space: RE object's type **atype**=collection

			Relationships **Actors**: Collection's owner **Assets:** None **Spaces:** The list of views belonging to the collection
Group	Space	A group in Mahara is collaboration space. Different types of groups exist. The group's type governs how people can join a group. A group can have views, forums and files/folders. The group's home page is itself made of a view. It can be edited using the same mechanism as any other views. "Normal" views have a type set to 'portfolio'.	**id**: the group id. Application's unique id. **dataset:** application uri **name**: Name of the group **description**: Group's description **url**: Link to the group homepage. **type**=space: RE object's type **atype**=group **Relationships** **Actors**: People participating in the group. Both administrators and participants. **Assets:** First level files. I.e. files belonging to the group and having no parent folder. **Spaces:** Forums of the group, portfolio's views of the group (i.e. excluding the group homepage), first level folders (i.e. folders belonging to the group and having no parents). Items marked as deleted are not displayed.
Group file	Asset	A group file is an artefact having the appropriate type (image, file or archive) and belonging to a group. A group file can have a folder as a parent. If the parent is null they are first level files. A group file can be accessed through the group file navigation page. A group file can be displayed in different views - both group and user views.	**id**: the file artefact id. Application's unique id. **dataset:** application uri **name**: Name of the file artefact **description**: File artefact's description **url**: Link to the group's page displaying the file. I.e. we link to the page displaying the file and not directly to the downloading link. **type**=asset: RE object's type **atype**=file **Relationships** **Actors**: Owner of the file **Assets:** None **Spaces:** Space in which it is posted
Group folder	Space	A group folder is an artefact having a type of 'folder' and belonging to a group. A group folder can have a folder as a parent. If the parent is null it is a first level folder. A group folder can be accessed through the group file navigation page. A group folder can be displayed in	**id**: the folder artefact id. Application's unique id. **dataset:** application uri **name**: Name of the folder artefact **description**: Folder artefact's description **url**: Link to the group's page displaying the folder's content. **type**=space: RE object's type **atype**=folder

		other views - both group and user views.	**Relationships** **Actors**: Owner of the file **Assets:** List of files belonging directly to this folder. I.e. we do not list files showing in children folders. **Spaces:** List of folders belonging directly to this folder. I.e. we do not listfolders showing in children folders.
Forum	Space	A forum in Mahara is a type of interaction. I.e. it is not an artefact. A forum is made of forum's topics. Each topic is a list of posts.	**id**: the forum id. Application's unique id. **dataset:** application uri **name**: Name of the forum **description**: Forum's description **url**: Link to the group's page displaying the forum's content. **type**=space: RE object's type **atype**=forum **Relationships** **Actors**: None **Assets:** Active forum topics belonging to this forum. I.e. we do not list topics marked as deleted. **Spaces:** None.
Forum topic	Comment	A forum topic is a discussion topic. It exists as a direct child of the forum and contains a list of posts/comments organized in a hierarchical manner. Posts could be listed as assets of the topic yet sending them would have the side effect of making them public. Bypassing the security features of the Group. The reason being that posts are only made of a title and a description. I.e. the post's meta-data are the post's data. If we share the post's description then we share everything. If we don't share the description we share nothing and it is useless. One option to overcome this limitation may be to send posts to the RE for recommendation processing but not as object for recommendation. This could be achieved either by marking	**id**: the forum topic id. Application's unique id. **dataset:** application uri **name**: Title of the forum topic **description**: Forum's topic's subject. **url**: Link to the group's page displaying the forum's topic content. I.e. the discussion. **created**: Created time. **type**=comment: RE object's type **atype**=topic **Relationships** **Actors**: Actor who created the topic **Assets:** None. **Spaces:** None. **Comments:** First level comments. That is those which are direct children of the forum topic excluding grand children.

		the object as informational only or by using a different type of object. Note that the same remark applies to comments in general.	
Forum post	Comment	A forum post is a message posted either as answer to the main topic or to to another message.	**id**: the forum post id. Application's unique id. **dataset:** application uri **name**: Title of the forum post. Will be empty in most cases as few people cares to fill-in another subject. **description**: Forum post's message. **url**: Link to the group's page displaying the forum's post content. I.e. the discussion. **created**: Created time. **type**=comment: RE object's type **atype**=post **Relationships** **Actors**: Actor who created the topic **Assets:** None. **Spaces:** None. **Comments:** First level comments. That is those which are direct children of the forum topic excluding grand children.
Site	Space	The application can have site level views and files/folders. Files/folders are only accessible to administrators and through json/block requests. Since they are not accessible through a standard interface they are not listed as "assets" of the site. They will be accessible though through views when they are added. Application level views are listed when at lest one access right has been granted. Those having no access right are not accessible and therefore not listed.	**id**: the site id. Application's unique id. **dataset:** application uri **name**: Name of the site **description**: Description for the site **url**: Link to Mahara's home page. **type**=space: RE object's type **atype**=site **Relationships** **Actors**: Active users. I.e. excluding those marked as either deleted or suspended. **Assets:** None. **Spaces:** Groups. Institutions
Institution	Space	Mahara can host several institutions. Users belongs to one institution only. Each institution can have a different presentation. This feature is optional. Each institution can have views and files/folders. The remarks made for "Site" above	**id**: the institution id. Application's unique id. **dataset:** application uri **name**: Name of the institution **description**: Description of the institution **url**: Link to Mahara's home page. **type**=space: RE object's type **atype**=institution

		applies to institutions as well. Note that this feature is not used by the University of Geneva as only one institution c applies.	**Relationships** **Actors**: Active users belonging to the institution. I.e. excluding those marked as either deleted or suspended. **Assets:** None. **Spaces:** Institution's level views being shared in some ways and belonging to the institution.
Notification	N/A	Notifications are messages sent to a user by different part of the system. Notifications are sent to a specific user and can be transmitted by one of the notification plugins: email, internal, etc. Different types of notifications exists: group, institution messages, view access messages, message from another user, etc. The user can select which type of notifications must be sent by which type of notification plugin. internal, email, etc.All notifications - even those sent by email - can are accessible in the user account page. Plugins can add new types of notifications. Note: notifications are private. As such they are not relevant to be served as recommendations yet they may contain information useful to make recommendations.	**id**: the notification id. Application's unique id. **dataset:** application uri **name**: Subject of the message **description**: Content of the message **url**: Link to Mahara's home page. **type**=?: RE object's type **atype**=notification **Relationships** **Actors**: To and From if they exists **Assets:** None. **Spaces:** None.
Blog and Blog entries	N/A	Blog and blog entries are artefact. As such they are not directly accessible but must be shared through a view. See View Artefacts above.	N/A
Tags	N/A	Two different kind of tags exists: one for view and another one for artefacts such as blogs. Artefact's tags are themselves artefacts whereas views tags are a separate list of words.	N/A

7.3 Mahara Activity Mapping

Activities are time event providing information about a change to the RE. Activities are short lived. They describe that something happened at a specific time. The consequences of the activity may last - for example an object's update - but the activity itself is instantaneous. Activities can provides information about user's navigation - that is logging - notify a change of application's state - Create Read Update or Delete action - or be triggered by a user - for example evaluation of a resource.

Activities are closely linked to the RE Object Model as many of them represent a change of the RE model - for example a new actor was created. Together they form a common vocabulary used to communicate time events by the applications. As such it is necessary to translate applications' events to RE activities.

Default activity structure:

provider: Sender, i.e. the url of the application or an agreed upon identifier. Required.
published: Time when the activity occurred. Format ISO 8601, 970-01-01T00:00:00+00:00. Required.
actor: User who performed the activity. Format Object. {id, name, ...} See Mahara User Object Mapping. Required.
verb: Action that was performed. See list of actions below. Required.
object: Object of the verb. Format: object, one of Actor, Asset or Space. Required.
target: Target of the verb. Format: object, one of Actor, Asset or Space. Optional.

7.3.1 Verbs

Verb	Description	Example	Comments
create	Object created	User created	Create reflects the creation of an object better than add.
update	Object updated	User updated	
delete	Object deleted	User deleted	
add	Object added to another object	User added to space	Add better reflects addition of an object than "join". User join group would be fine but Asset joined space does not sound so well.
remove	Object removed from another object	User removed from space	
like	User like object	User like a an asset	Not used for the moment
view	User viewed object	User viewed an asset	Using view instead of like for viewing pages allow us to use keep like for further use.

7.3.2 Mahara event mapping

Events are hooks in the code that can be intercepted by other part of application. Typically plugins. Events are predefined by Mahara and cannot be extended.

Event in Mahara	Activity Verb	Activity Data	Comments
activateuser	N/A		Only create, delete and undelete actions are transmitted for users. Mapping other types of actions may trigger many “delete” and “undelete” user actions.
addfriend	add	actor = logged in user object = friend user AAA model target = user AAA model	Friends are transmitted as an addition of an actor to an actor without using a space.
addfriendrequest	N/A		
blockinstancecommit	update	actor = logged in user object = view AAA model	A block instance commit is transmitted as a View update activity
creategroup	create	actor = logged in user object = group AAA model	There is no delete group event in Mahara
createuser	create	actor = logged in user object = user AAA model	
deactivateuser	N/A		
deleteartefact	N/A		User artefacts are essentially private so we don’t transmit deletion for them. Could be done though.
deleteartefacts	N/A		
deleteuser	delete	actor = logged in user object = user AAA model	
deleteview	delete	actor = logged in user object = view AAA model	Handle only portfolio’s views.
expireuser	N/A		
removefriend	remove	actor = logged in user object = friend user AAA model target = user AAA model	

removefriendrequest	N/A		
saveview	created==modified => create created<> modified=> update	actor = logged in user object = view	Handle only portfolio's views.
suspenduser	N/A		
undeleteuser	create	actor = logged in user object = user AAA model	Unlikely to happen but possible.
unexpireuser	N/A		
unsuspenduser	N/A		
updateuser	update	actor = logged in user object = user AAA model	The update action is a bit unclear as some actions - update password for example - do not trigger an update action event. On the other hand changing institution does trigger event.
userjoinsgroup	join	actor = logged in user object = user target = group	There is no userleavegroup event.

7.3.3 Access Activity

Access activity is generated by users' navigating to specific pages. Url access can be monitored by plugins and notified as an activity.

The following table describing which navigation to a url will trigger which activity.

Url	**Activity Verb**	**Activity Data**	**Comments**
interaction/forum/view.php	view	actor = logged in user object = forum AAA model	
interaction/forum/topic.php	view	actor = logged in user object = forum topic AAA model	
group/view.php interaction/forum/index.php? view/groupviews.php artefact/file/groupfiles.php group/members.php	view	actor = logged in user object = group AAA model	First level access to an interaction or group url is considered access to a group.
artefact/file/groupfiles.php?folder=xxx	view	actor = logged in user	Files can only be seen

		object = group folder AAA model	through a folder.
user/view.php	view	actor = logged in user object = user AAA model	User profile
view/view.php	view	actor = logged in user object = view AAA model	View access. Could be user, group or institution view.

8. Moodle Mapping to Common Data Model

8.1 Moodle reduced model

Note that the model below includes only a subset of all elements that exist in Moodle (i.e. Moodle provides many differen "activities", such as discussion forums, assignments, quizzes, chats rooms, etc). In the first phase, only the mos important ones (from a recommendation point of view) will be used by the RE. At a later stage, and depending or experiences made, additional object types may be added to this model.

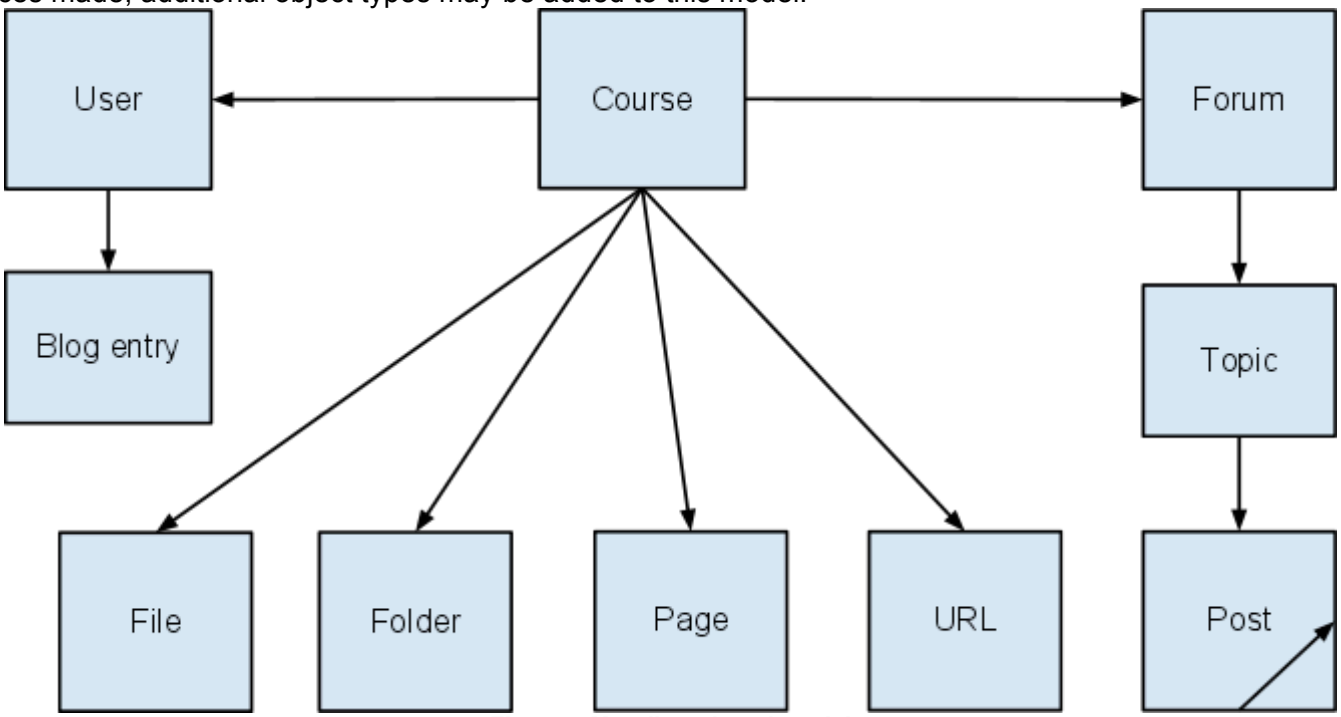

Figure 4 Moodle reduced model

8.2 Moodle Object Mapping

Objects structure
id: Object's global unique id. Required.
pid: Object's public id. For actors it is the Shibboleth Unique Id. Required for actors.
dataset: Dataset uri. I.e. the url of the provider: www.dokeos.unige.ch, www.moodle.unifr.ch, etc. Required.
name: Name of the object. Required.
description: Description for the object. Can contain HTML. Optional.
url: Link to access the object.
thumbnail: Link pointing to an thumbnail image. Optional.

created: Created time. Optional. Format ISO 8601, 970-01-01T00:00:00+00:00.
modified: Modified time. Optional. Format ISO 8601, 970-01-01T00:00:00+00:00.
tags: List of tags. Optional.
type: The RE object's type. One of Actor, Asset, Space. Required.

Object in Moodle	Object in RE	Comments	Data
User	Actor	Users are people registered in Moodle. Users can be enrolled in courses (as teacher, student, etc.). Depending on their specific role(s) in a course, users can create various types of content : files, urls, folders, etc.	**id**: User id **pid**: User public id. I.e. the Shibboleth Unique Id. Not the application's id. Required. **dataset:** application uri **name**: Full name **description**: Short introduction about the person **url**: Link to the profile page **thumbnail:** Link to a thumbnail image **tags** : a user can add tags to his/her profile **type**:actor: RE object's type **Relationships** **Spaces:** Courses user is enrolled in.
Course	Space	A course contains a collection of "static" content items (pages, files, folders, urls) and activities (discussion forums, quizzes, chat rooms, etc.).	**id** : course id **dataset:** application uri **name** : course full name **description** : a short summary of the course **url** : link to the course home page **type** : space **Relationships** **Assets** : all pages, files, folders, urls in the course **Spaces** : all forums in the course **Actors** : users enrolled in course
Page	Asset	A page is created in a course in HTML format.	**id** : page id **dataset:** application uri **name** : course full name **description** : a short description of the page **url** : a link to the page **created** : creation time **type** : asset
File	Asset	A file is a link to a file stored either directly in Moodle or in an external repository	**id** : resource id **dataset:** application uri **name** : resource name **description** : a short description of the resource **url** : a link to the resource

			created : creation time **type** : asset
Folder	Asset	A Moodle folder mimics a folder in a standard file system. It may contain files and other folders. In the context of the PLE, it is considered an atomic asset (i.e. no sub-assets for contained files and folders).	**id** : folder id **dataset:** application uri **name** : folder name **description** : a short description of the folder **url** : a link to the folder **created** : creation time **type** : asset
URL	Asset	A url is a link to an external web site	**id** : URL id **dataset:** application uri **name** : URL name **description** : a short description of the URL **url** : a link to the URL **created** : creation time **type** : asset
Forum	Space	A forum consists of a list of discussion topics.	**id** : forum id **dataset:** application uri **name** : forum name **description** : a short description of the forum **url** : a link to the forum **type** : space **Relationships** **Assets:** Forums topics in the forum
Forum topic	Comment	The starting message of a discussion topic in a forum. Answers to the topic (and answers to answers) are forums posts.	**id** : topic id **dataset:** application uri **name** : topic title **description** : comment content **url** : a link to the topic **created** : creation time **type** : comment **Relationships** **Actors**: Actor who created the topic **Assets:** None. **Spaces:** None. **Comments:** First level comments. That is those which are direct answers to the forum topic.
Forum post	Comment	A forum post is a message posted either as answer to the	**id** : post id **dataset:** application uri

		main topic or to to another post.	**name** : post title **description** : comment content **url** : a link to the post **created** : creation time **type** : comment **Relationships** **Actors**: Actor who created the topic **Assets:** None. **Spaces:** None. **Comments:** posts which are direct answers to this post.
Blog entry		Moodle allows users to create blog entries, even though there is no concept of a blog in Moodle. A blog entry is basically some text with associated tags. Blog entries can be referenced in different locations in Moodle (e.g. in a course). They are often searched/accessed through the associated tags.	**id** : blog entry id **dataset:** application uri **name** : title blog entry **description** : content of blog entry **url** : a link to the blog entry **tags** : tags associated with the entry **created** : creation time **type** : asset **Relationships** **Actor** : creator of the blog entry **Assets:** None **Spaces :** None **Comments** : None

Other types of objects may be added at a later stage (e.g. other Moodle activities such as wikis, chats, etc.).

8.3 Moodle Activity Mapping

In Moodle, there are two independent sources of data that could be used to feed the RE :

1. **Moodle log table**

   Moodle logs (some) user actions in a table inside its database. This includes for instance a user viewing a resource or posting a message in a discussion forum. However, many actions are not logged, e.g. all those related to groups (adding a user to a group). Logging additional actions would necessitate modifications of Moodle core code, which is not really the best idea (same problem as in Mahara above). Also, Moodle's documentation does not include a list of all actions that are logged. To find all these actions, one has to look in Moodle source code for calls to function "add_to_log" (which is actually called 236 times !).

2. **Moodle events**

   Moodle generates *events* when something "interesting" happens that is worth alerting the system about Programmers can then add their own event handlers for any of these events. But again, the list of predefined events (i.e. those that can be handled) doesn't include everything (see http://docs.moodle.org/dev/Events_API#Events_which_exist). In fact, there is no event for many of the actions tha would be interesting for the RE. For instance, there is no event when a user is viewing a resource. Note also tha events are not logged, so if we wanted to use Moodle events as a source of information for the RE, then we would have to create our own log of events, which is not a trivial task (unless events are sent in real time to the RE, but this is not really feasible).

In the first stage, given the above, the log table (with the limitations mentioned) seems to be the best source o information for feeding the RE. If we later find that we need to use other actions for recommendations, we may either adc additional calls to add_to_log in Moodle's code (to continue using a single source of information) or use events via even handlers.

Moodle's log table contains the following fields that can be used for the mapping :
- time : time when action occurred
- userid : id of user performing action
- course : id of course where action happened
- module : type of object, e.g. course, folder, user, resource
- cmid : unique id of object across all objects in all courses (course-module id)
- action : type of action performed (view, add, update)
- url
- info : additional information on action

Default activity structure

(based on ActivityEntry from OpenSocial Data Specification 2.0)

actor: user who performed the activity. Required
object: object of the verb. Format: object, one of Actor, Asset or Space. Required.
provider: sender, i.e. the url of the application or an agreed upon identifier. Required (not repeated in table below, as it is always the same).
published: time when the activity occurred. Format ISO 8601, 970-01-01T00:00:00+00:00. Required.
target: target of the verb. Format: object, one of Actor, Asset or Space. Optional.
verb : action that the activity describes. Required. Same list of actions as in Mahara (created, update, delete, add, remove, like, view).

The table below lists the actions from Moodle log that will be made available to the RE (partial list, should be extended over time).

Action in Moodle	**Verb**	**ActivityObject**	**Comment**
User enrols in a course	add	actor : user object : user published : time of action target : course verb : add	
User unenrols from a course	remove	actor : user object : user published : time of action target : course verb : remove	
User views a page	view	actor : user object : page published : time of action verb : view	

User views a file	view	actor : user object :file published : time of action verb : view	
User views a folder	view	actor : user object : folder published : time of action verb : view	
User views a URL	view	actor : user object : URL published : time of action verb : view	
User creates a forum topic	add	actor : user object : forum topic published : time of action verb : add target : forum	
User views a forum topic	view	actor : user object : forum topic published : time of action verb : view	
User posts an answer to a forum topic	add	actor : user object : post published : time of action target : forum topic verb : add	
User posts an answer to a forum post	add	actor : user object : post published : time of action target : parent post verb : add	
User views a forum post	view	actor : user object : forum post published : time of action verb : view	
User creates a blog entry	add	actor : user object : blog entry published : time of action verb : add	
User views a blog entry	view	actor : user object : blog entry published : time of action	

		verb : view	

9. Common API for Data Exchange

The different PLE platforms will make their data available through a common REST API [5], so that the RS can fetch the different assets (or resources), spaces, and people and their interactions, and construct/update the common recommendation graph. With respect to implementation issues, it is still open whether to use RDF to exchange data o using OpenSocial API, with a json or xml format.
Since we exchange data across different platforms, it's important to ensure that every actor or asset has a globally unique identification so that we can recognize and fetch them. For the assets, their id would be the respective application plus the system id. And for actors, the transmitted id should be a globally user id – Shibboleth[6] id - so that the same actor in different application is recognized as being the same person.

Examples of the two alternatives are provided below. Once the data is fetched and the recommendation algorithm is run the RS will return a recommended list of heterogenous items using REST web services and of course following the same common format with which data was initially sent. OpenSocial is based on a Google initiative to ease the exchange o data across different social media applications [2][3][7][8]. It has become popular in many platforms. These socia networks, known as OpenSocial containers, allow OpenSocial Gadgets to access information stored within the socia platform [7]. In the OpenSocial specifications, verbs, objects, object type and etc. are defined. With this standard, we car transfer the information in different platforms into a unified one. Although OpenSocial is effective in realization, it is no perfect yet. As we've analyzed the business in Mahara, we found that the definition in OpenSocial is not enough. Using the original OpenSocial can not exactly describe some actions in Mahara. So if we were to adopt it, we might need to extend it and add some new definitions, which can describe the business better.

Example 1
Get the RDF description of a user.

Request
Get /ac8b4ba95ccc0b817f4858d57dc848af-1

Response

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:dcterms="http://purl.org/dc/terms/"
        xmlns:aaa="http://ple.unige.ch/re/1#"
        xml:base="http://localhost/mahara_laurent/artefact/graaasp/service/data/"
        >
  <rdf:Description rdf:about="ac8b4ba95ccc0b817f4858d57dc848af-1">
    <aaa:id>admin</aaa:id>
    <aaa:name>Admin User</aaa:name>
    <aaa:url>http://localhost/mahara_laurent/user/view.php?id=1</aaa:url>
    <aaa:type>actor</aaa:type>
```

```
    <aaa:spaces rdf:resource="cd7135ae6d956a595aeb55dc4fcd10f0-6" />
    <rdf:type rdf:resource="http://ple.unige.ch/re/1#actor"/>
  </rdf:Description>
</rdf:RDF>
```

Example 2
Get the profile of the given user in json format

Request
/people/{guid}/@self

Response
{"id" : "profileID", "displayName" : "profileName", "attachments" : [{"id" : "personID", "displayName" : "personName", "aboutMe" : "the introduction of the person", "objectType" : "person"}] }

Rest

url pattern	Http Verb	Comments
/	Get	The site/application's data
/{id}	Get	Descriptor for object {id}. Note that the expected id is the application unique id and not the public id. For example users have two types of ids: the application id and the global/public id - shibboleth global id, etc. Querying this interface with a Shibboleth id is not supported.
/{id}?format={html|json|xml|rdf}	Get	All data from the {id} object in the specified format: json, xml, html, RDF.

Security issues:

As the web services expose potentially private, data access must be limited to the RE only. It is important to note the access rules are handled at the application level rather than at the RE level especially when it comes to closed items (items whose metadata is public and exchanged with RS but content is restricted). The proposed scheme is to use standard account/passwords passed as parameters over HTTPS. Additional security schemes may apply. For example, IP based security. Since the activities are recorded according to the same standard, all the result would be unified and could be analyzed by the RE.

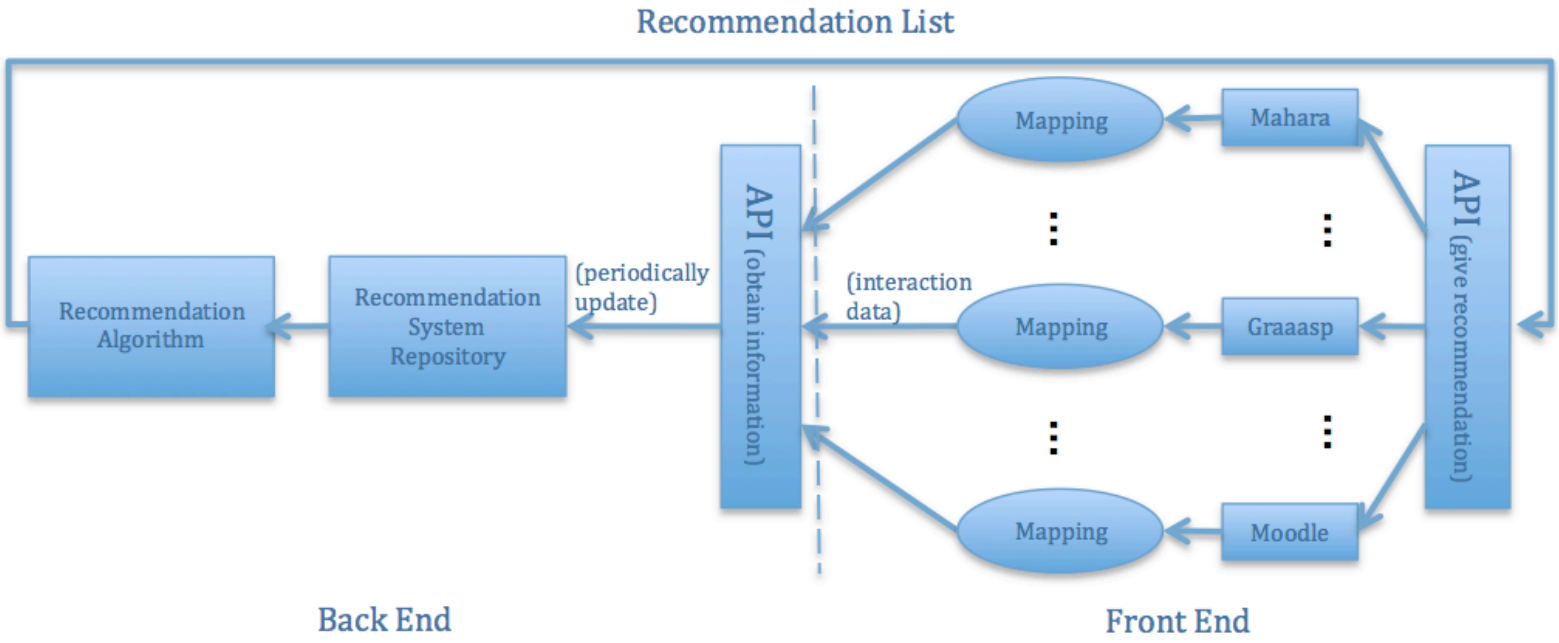

Figure 5 Data flow of the whole system

10. Recommendation Algorithm

This section summarizes the recommendation algorithm, which will be adopted.

A recommendation graph is constructed taking as nodes all public and closed actors, activity spaces, applications, and assets, and as assets their inter-relations (e.g. actor x is a member of space y). Then, the 3A ranking algorithm is run on the graph in order to return a recommended list of relevant items for each target user.

"The 3A ranking algorithm is influenced by the original pagerank algorithm that was developed by Page and Brin for ranking hypertext documents for Google. The pagerank algorithm is based on the idea that if the owner of page j links to a page i, he/she is implicitly indicating that page I is important. It follows that the more incoming links page I has, the more it is considered as globally important because many pages are linking to it. In addition, if authors of "authoritative" pages link in their turn to other pages, then they also confer importance to the latter. A random jump parameter, defining the probability of randomly falling on a page, and the damping factor, representing the probability to follow page links instead of jumping on a random page, are introduced to the algorithm.

Unlike the graph of hypertext documents of the original pagerank algorithm, the social graph of the 3A model involves heterogeneous nodes related by different types of edges that are not necessarily equally important. In such a multi-relational graph, when the surfer falls on a node, he or she can choose to follow different types of relations. The probability to fall on interesting nodes depends upon the probability that the adopted way will lead to them.

Finally, according to the target user and the context, all the nodes will be computed and ranked by the importance. And after the extraction of the items that the target user already have, a list of recommended items will be displayed to the user"[1].

11. Reference

[1] Sandy El Helou. The 3A Interaction Model and Relation-Based Recommender System: Adopting Social Media Paradigms in Designing Personal Learning Environments. École polytechnique fédérale de lausanne thése N°4829(2010)

[2] Wenjun Wu, Thomas Uram, Michael Wilde, Mark Hereld and Michael E. Papka. Accelerating science gateway development with Web 2.0 and Swift. Proceedings of the 2010 TeraGrid Conference, Article No. 23.

[3] Lynne Grewe and Sushmita Pandey. Quantization of Social Data for Friend Advertisement Recommendation System ADVANCES IN PARALLEL DISTRIBUTED COMPUTING, Communications in Computer and Information Science 2011,Vol 203, Part 1, 596-614.

[4] El Helou, S., Li, N. and Gillet, D. (2010) The 3A interaction model: towards bridging the gap between formal and informal learning. Proceedings of The Third International Conferences on Advances in Computer-Human Interactions

[5] RESTful Web services: The basics, http://www.ibm.com/developerworks/webservices/library/ws-restful/

[6] Shibboleth, http://shibboleth.internet2.edu/about.html

[7] Alexandros Dais, Mara Nikolaidou, and Dimosthenis Anagnostopoulos. OpenSocialGov: A Web 2.0 Environment for Governmental E-Service Delivery. ELECTRONIC GOVERNMENT AND THE INFORMATION SYSTEMS PERSPECTIVE, Lecture Notes in Computer Science, 2011, Vol. 6866/2011, 173-183

[8] Martin Friedrich, Martin Wolpers, Ruimin Shen, Carsten Ullrich, Ralf Klamma, Dominik Renzel, Anja Richert, Bodo von der Heiden. Early Results of Experiments with Responsive Open Learning Environments. Journal of Universal Computer Science, vol. 17, no.3 (2011), 451-471