I2G

INTERGEO

Deliverable N°: D4.4

Platform's Administration Manual

The INTERGEO Consortium

Feb 2009

Version: updated draft September 30, 2010 1:23 P.M.

Main Authors:

Santiago Egido (Maths for More S.L.)
Paul Libbrecht (DFKI GmbH)
Henri Lesourd (Univ. Montpellier 2)

Project co-funded by the European Community under the eContentplus Programme

Project ref.no.	ECP-2006-EDU-410016
Project title	INTERGEO - Interoperable Interactive Geometry for Europe

Deliverable status	updated draft September 30, 2010 1:23 P.M.
Contractual date of delivery	M15 – December 2008
Actual date of delivery	February 28^{th} 2009
Deliverable title	Platform's Administration Manual
Type	Report
Status & version	updated draft September 30, 2010 1:23 P.M.
Number of pages	49
WP contributing to the deliverable	WP4
WP/Task responsible	Paul Libbrecht
Authors	Santiago Egido (Maths for More S.L.), Paul Libbrecht (DFKI GmbH), Henri Lesourd (Univ. Montpellier)
EC Project Officer	Spyridon Pilos
Keywords	web platform, users, resources, maintenance
License	This document is available under the license *Creative Commons Attributions Sharealike Germany 2.5* [?]

Contents

Executive Summary

In order to fulfill the objectives of the Intergeo Project, it is absolutely essential to have a permanent platform where users can interact and share resources, so that communities have enough time to their disposal to appear, grow in a stable environment, and become more productive than just the sum of their members. In order to guarantee the long term sustainability of I2GEO, with limited maintenance resources, an administrator manual is a key element for contingencies to be solved quickly and efficiently. In addition, now that the platform development is approaching its conclusion, this manual describes the website architecture and its modifications with respect to Curriki, so that it will become also a reference documentation for possible future improvements to be easily implemented.

This manual also contains instructions for installation and upgrading, devoting special attention to the differences with Curriki and to the software explicitly developed for I2GEO: CompEd, the search tool, Skills Text Box and the ontology tools. Maintenance and monitoring tasks and tools are described, including among others translation workflows, editorial activities, user support, visitors tracking, and resource management. In addition to I2GEO's architecture, its optimizations and limitations are described.

1 Introduction: the I2GEO Platform, Mission and Choices

The I2GEO platform has been assembled as a web based platform to serve typical users of interactive geometry, in majority practicioners of math education, in order to support the functions of:

- the publication of interactive geometry constructions on the web, with annotations
- automated conversions between the various formats of interactive geometry software
- the possibility to annotate and search resources with a language that crosses the boundaries of the regional curriculum standards

The consortium went through a process of requirements elicitation through user-stories which are concrete scenarios that one wishes to see true when the implementation is complete. The list of them can be found, for partners of the Intergeo project, at `http://intern.inter2geo.eu/node/13`.

From these requirements, and from our exploration of applicable platforms, it was clear that cross-curriculum search was not to be found anywhere as an existing tools and that the particular nature of interactive geometry constructions required a dedicated metadata schema and a dedicated set of services.

The quest for existing platform assessed the maturity (e.g. proven scalability), the ease of customization for the existing and expected team, and the feature coverage (such as the ability to compose supporting materials online) among open-source platforms with an active multi-party development community. Tools such as basic content-management-systems (e.g. OpenCMS, Magnolia), community portals (e.g. LifeRay, Geronimo, Drupal), or learning object repositories (e.g. DLib, LeMill, EducaNext, GNU-edu) were evaluated: none had all of the expected or desirable features.

The tool that came with best results and with a feature-set surpassing our expectations was Curriki, which was chosen as the central storage and editing front-end adjusted with user-interface adaptations for the metadata and for the cross-curriculum search tool. The decision was that of a feature-full system with

little widespread usage but proven scalability; that of a complex system where each adaptation requires the development team to understand the many levels of functions, their programming methods, and their user-interfaces.

This administrator manual describes the I2GEO platform, which is a derivative of Curriki complemented by the cross-curriculum search tool. As much as possible the descriptions cover all the parts simultaneously.

2 Architecture of the I2GEO Platform

I2GEO is made of the following web applications which communicate but are developed separately. All these components are deployed on `http://i2geo.net/`.

Curriki is a web application based on XWiki made for the collaborative creation of learning resources on the web mostly for one single server, `http://www.curriki.org/`. Curriki is distributed under the GNU General Public License [?] by the Global Education Learning Community non-profit company. It is packaged as a java-servlets web application depending on a storage in MySQL. Curriki answers the requests made to the paths starting with `xwiki/`.

All of the features of Curriki are used in I2GEO except some metadata properties (the topic and education levels) and the search tool.

SearchI2G is a web application based on Apache solr, an information retrieval web application. This application is delivered by DFKI GmbH, under the Apache Public License [?] and developed within Intergeo to approach cross-curriculum search. SearchI2G mostly serves as back-end tool for the I2GEO tunings of Curriki. It is packaged as a java servlet web application.

CompEd is a web application based on the AppFuse framework [?]. It aims at the presentation and edition of the topics and competencies, which are the entities used for annotations and search on I2GEO, as explained in [?] and [?]. CompEd is delivered by DFKI GmbH under the Apache Public License [?].

I2GEO **Root** is a simple web application whose role is to store the static parts outside of each of the individual web applications and to redirect requests according to a map so as to honour old URLs and shorter URLs. I2GEO Root is delivered by the Université Montpellier 2 under the Apache Public License [?].

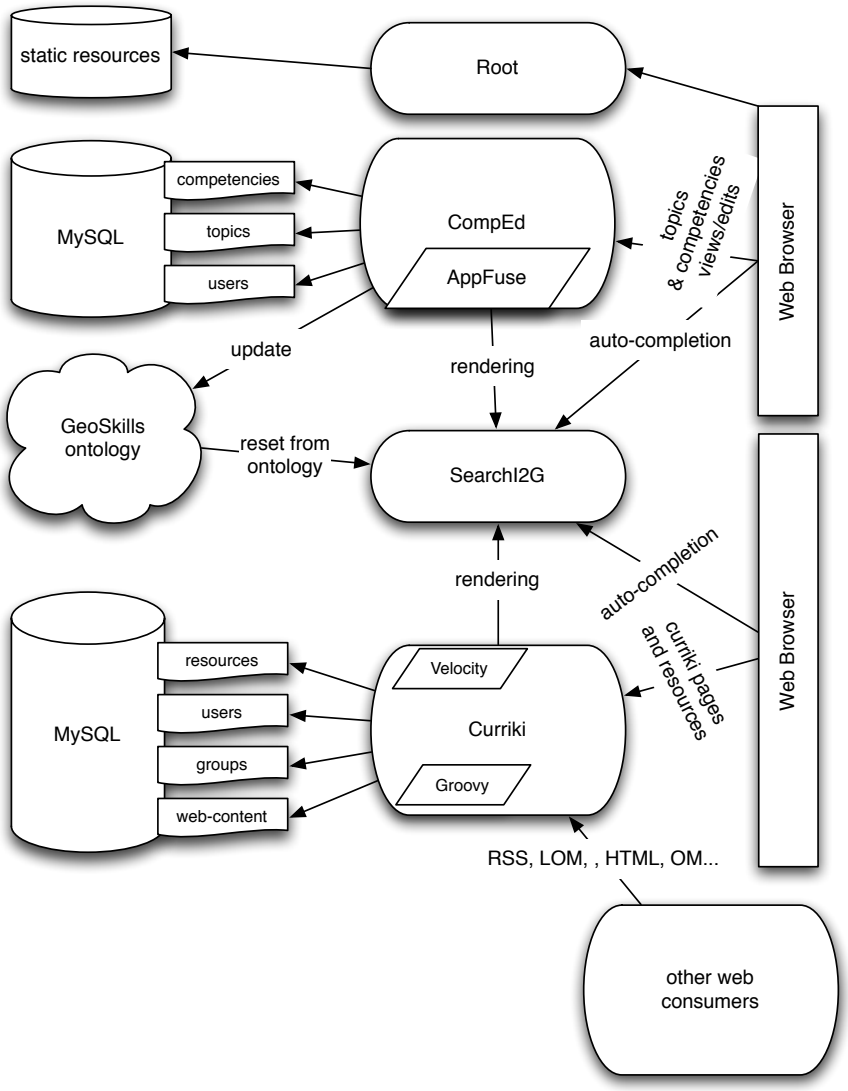

Figure 1: Overall Architecture of the I2GEO Platform

2.1 The Curriki Web-Application

The Curriki system is an XWiki application: it builds on the XWiki architecture by extending it for the purposes of exchanging learning resources and coordinating communities. The extensions are at several levels: user-interface, control, and back-end.

The XWiki model underlying it is that of a java web application with data-objects accessed in memory and stored in SQL databases through the Hibernate persistance layer [?]. The XWiki platform is based on documents which have a title and a body (in many languages). It offers a type-system where objects can be attached to documents and be specified by classes (which are special documents) which describe property names and ranges.

This ability is at the heart of XWiki, and is described well in the XWiki tutorial. It has been used to allow the edition of *traces* in the first steps phase of Intergeo (where traces are simple records relating the title, license, etc).

XWiki documents are, as in any wiki, editable text-pages. XWiki documents are grouped in spaces. Each space has a home page called `WebHome`. Pages delivered by XWiki applications are made by the delivery of side-content, considered to be the inclusion of a few panels, and the delivery of the main body, considered to be the rendering of the text page; each delivery is done along an action. The rendering process is the execution of a script in the Velocity template language [?], combined, possibly, with Groovy fragments [?]. The execution of these rendering scripts is done with a few variables available in their context, such as information about the current-user, access to the document and its attached objects, to the current-action (view, save, inline-edit, edit....), to other documents, or to many other java objects (the core objects are documented in the public API of XWiki [?].

Curriki leverages this infrastructure by a very large set of velocity macros, which is loaded in memory at start time. Most wiki pages in Curriki are made of objects and their wiki-text is a mere call to these macros. The macros of Curriki are documented in a form close to javadoc [?].

Curriki also provides a few extra java classes which allow finer coding than Velocity or Groovy; they are provided in the form of plugins. Their list can be browsed [?].

Central in this structure is the notion of Curriki asset which corresponds to an educational resource. The Asset XWiki object contains all the generic metadata information that all resources should be annotated with. It has been extended by several properties specific to the Intergeo metadata.

Asset-types are numerous and keep being expanded. In a future release of Curriki and subsequent of I2GEO, special asset-types for each interactive geometry applications will be offered taking care of the individual renderings.

The Curriki asset-types has several sub-types for each of the elementary natures of the resources, notably:

- external assets, aka, hyperlinks
- uploaded files
- collection assets, which contain other assets

The asset-class and sub-classes are not only XWiki classes but also java classes whose methods allow an easy access to the properties both for setting and for writing.[1]

Another set of objects of importance to Curriki is a *group*, which represents a community of users focused on a particular topic. In their space, the groups host communication through a web forum, documentation-oriented resources, and learning materials, all packed within a space with its own history.

The Velocity scripts distributed by Curriki as well as the XWiki rendering infrastructure use these objects in order to generate user-interface language code that the web browser renders. Mostly, the XWiki infrastructure generates HTML pages made of:

- the header-bar containing titles and login-status
- panels for the navigation menu and highlights
- the main body, be it a resource display, a list, reviews or group documents.

The HTML pages embed a large amount of Cascading Style Sheet files and use JavaScript to provide client automation. The style-sheet files are made of XWiki and Curriki style files with a few I2GEO adaptations directly done on files.

The Javascript files are based on a few JS libraries, such as yui, jquery, and ext-js. They also include code generated by the Google Web Toolkit [?] which is particularly used for the Currikulum Editor and for the auto-completion features described below.

The JavaScript tools communicate with the server through a Restlet API [?], allowing for example a file to be uploaded on a temporary document before it is annotated with metadata, or the search results to be read.

2.2 Adaptation Strategies from Curriki to I2GEO

Below, we shall describe the other web applications that constitute the I2GEO platform. This simple co-existance is not enough to perform all the functions of I2GEO, and adaptations to the Curriki UI and object-model are needed.

The adaptation strategy is that of patching an existing source: based on a release of Curriki, the version called `EOU 2` in the present case. The following adaptations were performed:

[1] These objects are obtained from the Curriki plugin using a call such as `xwiki.getPlugin("curriki").fetchAsset("space.name")`.

- the add-a-resource second panel, the Currikulum Editor, the Restlet server interfaces, the asset XWiki classes have been adjusted to replace the input for topic and educational levels by inputs of the SearchI2G web application, using entity designation through auto-completion
- the rendering templates of resources have been adjusted to display these and the quality framework result
- the display of lists of items (such as the RSS and recent resources) has been adjusted to fit I2GEO's
- another tab has been added to the display of resources to link to the quality reviews, the whole quality framework has been injected as a new space into the I2GEO platform
- the translation-bundles, scattered in several places in XWiki have been centralized in the `Translations` space allowing an almost unified management of the translators work
- A LOM-XML view of each resource has been provided
- editorial adaptations have been done and keep being done

All application changes are stored as modified files within the subversion repository of intergeo `http://svn.activemath.org/intergeo/Platform/i2gCurriki`. The quality framework is stored in its own space `http://svn.activemath.org/intergeo/Platform/QualityFramework`.

The editorial changes as well as the translation efforts and configurations are only maintained in the database of I2GEO. They are exported to their XML backup format checked in in `http://svn.activemath.org/intergeo/Platform/i2gCurriki` before upgrading.

See below section 3 for the details on source availability and installation instructions.

2.3 The Intergeo Search Tool

This web application responds at the URL `SearchI2G`, its main function is to maintain an index of both the terms of annotations of resources that live inside the ontology as well as the annotated resources.

In its auto-completion function, the SearchI2G tool is called Skills-text-box, it edits a list of nodes of the GeoSkills ontology [?]. It can be embedded within web pages or called in a pop-up, updating a hidden form field.

In its rendering function, the SearchI2G tool renders a comma separated list of GeoSkills identifiers to HTML with appropriate languages. This function is used in the rendering of resources and lists in the Curriki part of I2GEO.

The maintenance of the index of this part is made by a call to the full-index-creation (which takes approximately 2 minutes as of 01–02–2009) or through the reception of XML documents indicating updates such as the ones the ontology server of CompEd receives (see below).

Both direct and auto-completion rendering present the nodes of GeoSkills with a link to a presentation of them provided by CompEd for topics and competencies and, currently, to the OWLdoc output for educational levels.

The suggestion feature of GeoSkills is built-in in these editors and files new documents on the `Suggestions` space of Curriki.

In its search function, the SearchI2G tool renders a list of resources matching the query. Its index is updated by regular notifications of documents' saves in the Curriki part of I2GEO and the reading of the resource rendering as well as the LOM record.

See below section 3 for the details on source availability and installation instructions.

2.4 The Competency Editor and Ontology Update Server

The CompEd web application is a web application whose role is to edit the topic and competency nodes of GeoSkills. It is described in technical details in [?].

The competency editor offers simple browsing of topics and competencies, most useful for users to navigate the structure of topics and competencies in order to identify by navigation the best term to search and annotate with. A simple feature of SearchI2G's auto-completion code allows this choice to be flowing into the last visited Skills-text-box.

See below section 3 for the details on source availability and installation instructions.

2.5 The I2GEO Root Web-Application

This web application is very simple. It has only two functions:

- redirect requests that are issued to it so as to support short URLs (such as the groups names, e.g. `http://translators.i2geo.net/`) or older URLs (such as those stored before on `http://inter2geo.eu/`). A table of these redirects can be seen on `http://svn.activemath.org/intergeo/Platform/RootWebapp/src/main/webapp/WEB-INF/web.xml`
- static file service for an amount of files is ensured there as well, simply using the Apache and Tomcat infrastructure for file-serving

See below section 3 for the details on source availability and installation instructions.

2.6 Interface of I2GEO to External Web-Services

The web nature of I2GEO is rich. As much as possible, each user action is traceable in the history and each state view is bookmarkable. This feature, although it was not made explicit in the quest for a platform, is of crucial importance for the 'referenceable web', a web that can be talked about and spidered as hinted about in [?]. This feature is often lost in applications of the sort *rich-internet-applications*.

As of today, the following web interoperability possibilities are allowed for external to integrate with I2GEO on the web.

Web Robots web spiders can simply follow links present in web pages and will see all non-private content doing so. For the sake of a faster update pace, an XML *Sitemap* [?] is made available. The Google robot seems to use it every 3 days.

RSS Syndicators The RSS format is a simple XML-based encoding for dated streams of activity. RSS streams of three flavours are available in I2GEO:

- the life of groups is tracked in the group stream. It describes the changes in a group's documents, set of resources, or forum posts.
- a list of all recent resources is displayed as the default RSS feed of this web site
- the list of all changes is available under that same URL to registered administrators

All three kinds can be subscribed to (the administrator one requires one to add the login information) and has been tested with Safari, Apple Mail, NetNewsReader, FireFox, and Google Reader.

Future: Metadata Crawler Both LOM and RDF fragments can be obtained about each resource, for the description of the metadata. Once a suitable way is found, such as GRDDL as followed in [?] or RDFa [?], links from the resource renderings to the records will be presented.

The current team of I2GEO has not yet found a sufficient amount of consumers of such metadata, be it at least to assess the extent of the interoperability.

Future: MetaBlogger The MetaBlogger protocol [?] is a simple protocol used by blogging server to receive blog uploads from desktop applications. This protocol should be used in order for interactive geometry desktop systems to upload the resources directly on I2GEO, with a possible metadata form screen after the post.

3 Installation and Upgrade

3.1 Common Installation Practice

Most of the web applications described below are built from source using the maven 2 build tool [?]. It is common to maintain a single web application container where all the web-applications are running even though they only communicate through web interfaces. On the file system, a directory is made of this web application container, and a building space aside. The building space contains a checkout from the subversion of each project (detailed in the installation instructions); a deployment is then made by the copy of the directory of the web application produced by the build into the web application directory of the container.

In order to run the installations described below, one needs the following prior installations:

- a Java Development Kit version 1.6 or above. For several platforms, this can be downloaded from `http://java.sun.com/`.
- the Tomcat web server: one needs to have Java installed first. Then, one only needs to unpack the binary distribution of Tomcat (one can get it from `http://tomcat.apache.org`). Once it is done, the script `bin/startup.sh` starts the server. You should then see the initial Tomcat page at the URL `http://localhost:8080` in your browser ;
- the Maven project builder (version 2): very easy install, same as with Tomcat above, i.e., it depends on Java, and one only needs to download the binary distribution from `http://maven.apache.org/`, unpack it, then the command `bin/mvn` should work out of the box ;
- the MySql database (version 5 at least): here, we will sketch MySql's install procedure, by means of using the version provided at `http://www.washington.edu/computing/web/publishing/mysql-install.html` as an example. First, download the tarball:

  ```
  /home/henri>wget http://www.washington.edu/computing/web/
  publishing/mysql-standard-5.0.27-linux-i686.tar.gz
  ```

  ```
  /home/henri>untargz mysql-standard-5.0.27-linux-i686.tar.gz
  ```

```
/home/henri>mv mysql-standard-5.0.27-linux-i686 mysql ; cd
mysql
/home/henri/mysql>_
```

Then, run the install script:

```
/home/henri/mysql>./scripts/mysql_install_db
...
/home/henri/mysql>_
```

Next, one must edit the `.my.cnf` file, which has to be located in the homedir of the user owning the database (i.e., in the '~' directory, namely `/home/henri` in our current example). Basically, the `.my.cnf` configuration file contains information about where the file socket MySql needs to communicate with its clients should be located, and where the MySql software and its data files (i.e.: the main database) should be located. Given what we showed above, a possible `.my.cnf` could be:

```
[mysqld]
port=3306
socket=/tmp/mysql.sock
basedir=/home/henri/mysql
datadir=/home/henri/mysql/data
[client]
port=3306
socket=/tmp/mysql.sock
```

Finally, one should launch mysqld and initialize the root password:

```
/home/henri/mysql>./bin/mysqld &
/home/henri/mysql>./bin/mysqladmin -u root password
'passRoot'
...
/home/henri/mysql>_
```

If everything went correctly, you should now be able to enter in an interactive session and run some commands, e.g.:

```
/home/henri/mysql>./bin/mysqld --max_allowed_packet=32M &
/home/henri/mysql>./bin/mysql -u root -p
Enter password:  ...
Welcome to the MySQL monitor.
```

```
Commands end with ; or \g.
Your MySQL connection id is 1 to server version:
5.0.27-standard
Type 'help;' or 'h' for help.  Type 'c' to clear the buffer.
mysql> use mysql;
mysql> create table People (Nom VarChar(255),Prenom
VarChar(255));
mysql> Insert into People Values ('Lesourd','Henri');
mysql> Insert into People Values ('Libbrecht','Paul');
mysql> select * from People;
+------+----+
| Nom       | Prenom |
+------+----+
| Lesourd   | Henri  |
| Libbrecht | Paul   |
+------+----+
2 rows in set (0.00 sec)
mysql> quit;
/home/henri/mysql>_
```

For the case of the current installation at DFKI, a SuSE Linux server, the `platform2` directory contains the servlet container, the Tomcat server version 6. 0.18 from the Apache foundation (`http://tomcat.apache.org/`). The `platform2` directory also contains a `buildingSpace` directory with `curriki-dev`, `SearchI2G`, `comped`, `comped-maven-plugin`, `RootWebApp` as described below. Our documentation only provides code snippets for the Unix shells.

Monitoring is then done through the inspection of the log files. Mostly the single log file of the servlet container is used thus far. In the current installation it is the file `conf/catalina.out`.

Before installing, you might want to talk with the translations coordinator and make sure you are going to compile with the last version of the translation files, see page 41.

3.2 Installing Curriki

In this section, we describe the lower-level part (i.e.: installing the files, configuring the database and the container) of Curriki's installation procedure. The next section describes the remaining parts which have to be performed manually, along with the XWiki's import/export mechanism.

To devise the install procedure below, we used information from the following sources:

`http://curriki.xwiki.org/xwiki/bin/view/Main/I2GcurrikiBuild`

`http://curriki.xwiki.org/xwiki/bin/view/Main/InstallationInstructions`

`http://dev.xwiki.org/xwiki/bin/view/Community/Building#HInstallingMaven`

`http://platform.xwiki.org/xwiki/bin/view/AdminGuide/InstallationMySQL`

We had to debug and to verify many things, for this information is error prone and often incomplete. But we had to start from there, for it was (mostly) the only documentation available. We also had a look in mailing lists, to find workarounds for some bugs in the software components, which are (as much as possible) described below (cf. section **3.4.**).

To perform the task, we provide an automated script. You should first download the directory containing the installation script (i.e.: `http://svn.activemath.org/intergeo/Platform/install/`) somewhere on your server. To make things clear, let's say we want to perform the install in the directory `/home/henri/Intergeo`. Then we will proceed as follows:

```
/home/henri>mkdir Intergeo ; cd Intergeo

/home/henri/Intergeo>svn co http://svn.activemath.org/intergeo/Platform/install

/home/henri/Intergeo>cd install

/home/henri/Intergeo/install>ls

configFiles install

/home/henri/Intergeo/install>_
```

The `install` script should be edited, to provide the following informations:

- The directory where the install takes place ;
- Tomcat's home directory ;
- MySql's home directory ;
- The password of MySql's root account ;

If we suppose that Tomcat and MySql home directories are `/opt/Tomcat/` and `/opt/MySql`, respectively, and that MySql's root's password is '`passRoot`', then the first lines of the install script (located at `/home/henri/Intergeo/install/install`) should become:

```
#!/bin/bash

TOMCAT_HOME=/opt/TomCat

export TOMCAT_HOME

INSTALL_DIR=/home/henri/Intergeo/install

export INSTALL_DIR # Where you did unpack i2geo's install dir

MYSQL_HOME=/opt/MySql

export MYSQL_HOME PASS=passRoot # MySql's root password
```

Before starting the script, we need to have an instance of MySql running, with the appropriate value of its parameter `max_allowed_packet` (this is necessary because there are some cases where XWiki uses very big packets). Let's start (or restart it):

```
/home/henri/Intergeo/install>cd /opt/MySql

/opt/MySql>bin/mysqld --max_allowed_packet=32M &

...
```

We can now run the install script:

```
/home/opt>cd ~/henri/Intergeo/install

/home/henri/Intergeo/install>./install

...
```

Once it is finished, everything should (hopefully) have been automagically compiled and configured. To see if it really worked, let's first restart Tomcat:

```
/home/henri/Intergeo/install>cd /opt/Tomcat

/opt/Tomcat>bin/shutdown.sh ; bin/startup.sh

...
```

If we point a browser to the URL `http://localhost:8080/xwiki`, we should now be able to see Intergeo's homepage (it's okay if some parts are missing or if some documents can't be found: we will solve this problem in the second part of the install).

3.3 Steps to perform manually

3.3.1 Importing some files

Once you successfuly installed the wiki, you should import a bunch of Curriki-related data in your system, by means of the XWiki's import mechanism (the import and export mechanisms in XWiki are described in more detail below).

To do this, go to:

`http://localhost:8080/xwiki/bin/import/Admin/Import` (Note: to be on `/import` and not on `/admin` (like it is described in some places on the Internet) is important, for there is a bug in XWiki preventing success when you run the import from `/admin/Admin/Import`. Thus the correct URL is the one above).

Then in « Add an attachment », choose the file:

`wiki/target/curriki-wiki.xar`

as an upload file. Once you click « Attach this file », the page should take some time (not more than one or two minutes, otherwise there is a problem, cf. points (2) and (4) in **1.1.c.** above), and then reappear, with the file you just uploaded in the list of « Available files to import ». If you click on it, the list of its contained elements should appear under « Available documents to import ». Then click the button « Import » to finally perform the import operation.

If everything goes well, you should then see the following error message:

Error: You are not allowed to view this document or perform this action.

To correct this problem, log in in the XWiki as user '`Admin`', using the password: '`admin`', and then go to:

`http://localhost:8080/xwiki/bin/view/Admin/GiveProgRights`, and click « confirm ».

Finally, you also need to import the files containing the (multilingual) messages which are used by the Intergeo software. The related XAR file should be located at:

`http://svn.activemath.org/intergeo/Platform/Translations/` (Note: that's currently how we do it. In the future, we plan to automate this step completely).

Once the translation files have been imported, they must be added to the list « Internationalization Document Bundles », located at the page under the

link « Programming » on `/xwiki/bin/admin/XWiki/XWikiPreferences` (you only need to do one declaration per multilingual page, rather than one declaration per language. E.g. given a multilingual page having the localizations `Translations.XWiki`, `Translations.XWiki:de`, `Translations.XWiki:es` and `Translations.XWiki:fr`, you only need to add « `Translations.XWiki` » to the list).

Congratulations! You installed the basic Intergeo platform successfully!

3.3.2 About XWiki's import/export mechanism

One important problem in wiki systems (at least, this is how things are in XWiki) is their use of an opaque system (i.e., a database) to store the pages and, more generally, various other kinds of data (e.g., scripts). That's why mechanisms for *exporting* data outside of the wiki (e.g. for backup) and for *importing* in the wiki (e.g. for installing) are necessary.

3.3.2.a The XAR file format

In XWiki, the XAR file format is used for the purpose of exporting and importing data. As a matter of fact, a XAR file is a JAR file, containing a bunch of XML files (one for each page contained in the XAR), plus a `package.xml` file containing the list of all the files contained in the XAR.

Thus it follows that to create a XAR file (and provided the appropriate `package.xml` file has been created), a simple '`jar cf`' command is sufficient. For example, given the two XML files `A` and `B`, which store the content of the two pages A and B on the XWiki, we only need to put these two files in a directory (e.g., '`tmp`'), plus the related `package.xml` file:

```
/home/henri/tmp>ls

A B package.xml

/home/henri/tmp>_
```

Once the appropriate '`jar cf`' command has been run:

```
/home/henri/tmp>jar cf ab.xar package.xml A B

/home/henri/tmp>_
```

we obtain a XAR file suitable for importing the pages A and B in another XWiki:

```
/home/henri/tmp>ls *xar
```

```
ab.xar

/home/henri/tmp>_
```

It should be noted that the structure in the directory 'tmp' above mimics the structure of XWiki's naming, i.e., the pages are supposed to appear under /xwiki/bin/view/ on the XWiki server where they will be imported (same rule applies for directories, e.g. the file F1 located in the directory tmp/Dir1 would represent the URI /xwiki/bin/view/Dir1/F1 on the server).

The file package.xml must contain an entry per file, otherwise the files having no entries are ignored by the import mechanism. Thus in our example above, our package.xml could look like:

```
<?xml version="1.0" encoding="utf-8"?>

<package>

<infos>...</infos>

<files>

<file defaultAction="-1" language="">A</file>

<file defaultAction="-1" language="">B</file>

<file defaultAction="-1" language="">Dir1.F1</file>

</files>

</package>
```

(Note: please observe the notation 'Dir.F1' above, which would lead to a page located at 'Dir/F1' on the XWiki. As far as we understood it, it's not possible to add more levels, e.g., Dir1/Dir2/F1 is not possible).

3.3.2.b The xwikidoc XML dialect

Finally, the content of one file is an XML-based format describing the page, i.e., its content plus the metadata. For example, here is one of our translation pages, called 'Translations.XWiki':

```
<?xml version="1.0" encoding="utf-8"?>

<xwikidoc>

<web>Translations</web>

<name>XWiki</name>

<language>fr</language>
```

```
...
<creator>XWiki.segido</creator>
<author>XWiki.segido</author>
...
<content>1 &lt;h1&gt;Translations, Francais&lt;/h1&gt;
&lt;pre&gt;
#side panel
panel.navigation.home= HOME
panel.navigation.home.rollover= I2geo - Géometrie Interactive
...
</content>
</xwikidoc>
```

As can be seen, although it's not extremely hard, there is still some work to do to create such XML files from their content only. Usually, such files are obtained by means of exporting data out of an XWiki. The obtained XAR file can be unpacked by means of a '`jar xf`' command, and its `package.xml` can be edited, to create a smaller XAR from it.

3.3.2.c Exporting data out of an XWiki

The export function available from the URI `/xwiki/bin/admin/XWiki/XWikiPreferences` of any XWiki install can be used to export the whole content of the XWiki. Although useful, this function is not very precise (not only does it exports mostly everything, but there are some pages it doesn't takes into account).

For performing our own exports, we used the more configurable Groovy script available at:

`http://svn.xwiki.org/svnroot/xwiki/curriki/tags/curriki-1.2.0/wiki/src/main/resources/Admin/ExportPageList`

This script takes an XWiki page containing the exact list of the pages to be exported, and creates the XAR file accordingly.

We will not discuss this topic of XWiki's export in more detail. But we had to mention it, for it clearly appears to us that without using a (perhaps customized) version of the `ExportPageList` script pointed out above (or a similar script), no serious backup policy could be implemented: thus it's really important to know that it exists.

3.4 About the install script (aka: « if it doesn't work... »)

The Intergeo platform is developed on top of a software stack which, although robust and implementing standard technologies, is in many respects not very well polished (i.e.: the components depend on many things, and their interfaces are not always well defined). Furthermore, part of the software is still under development while we are writing this documentation.

This is why in the list below, we summarize a bunch of important things we observed while developing the installation procedure (which are very useful to know if it fails at some point):

1. In the various `pom.xml` files (especially: `curriki/pom.xml`, `curriki/web/pom.xml`, `curriki/gwt/pom.xml`), there are version values with '`SNAPSHOT`' inside: it means that we rely on beta versions to perform the build. Given how Maven operates (i.e.: by means of automatically downloading the packages from remote repositories), we can be sure that once the stable versions will appear (in such a case, the '`SNAPSHOT`'/beta versions disappear from the servers), the Maven build will break.

   There is no solution to this problem, except banning the practice of using beta versions in the builds. It turns out that the developers of the software components which wrote the `pom.xml` files we depend on do not honor this. Currently, we removed as much as possible of these '`SNAPSHOT`' dependencies, but some remain, because the stable versions do not exist. Once these stable versions will be available, the related dependencies to their beta snapshots should be corrected (removing the '`SNAPSHOT`' substring is sufficient, Maven does the rest) ;

2. The Java-based implementation of XWiki is extremely dependent on the initial memory configuration of the JVM, which should be configured with the options '`-Xms512m -Xmx2048m`' when starting Tomcat (see `CATALINA_OPTS` and `JAVA_OPTS` in `bin/catalina.sh`).

   This is automatically done by the install, but in case one observes extremely long delays while loading a page (i.e. several minutes), it can be related to memory exhaustion in Java. To know what's going on, the files `logs/catalina.out` and `logs/localhost*.log` are very useful, they contain all Java error and log messages ;

3. The XWiki does not always start, it can be that you only get a blank page, or that it starts but that later some operation (e.g. importing a XAR file) repeatedly fails. In such circumstances, we observed that restarting Tomcat can help. We also met some situations where it was not sufficient, while rebooting the machine solved the problem ;

4. When importing XAR files, it seems that the Lucene indexer comes into play. If for some reason it cannot finish his job, the database can become

corrupted. This is a very vicious kind of error, because such corruption problems do not always appear explicitely (i.e.: you still can boot the wiki, everything looks fine, while in fact, one portion of the database is corrupted, preventing you from performing some operation, e.g., finishing an interrupted XAR import, or removing the corrupted XAR).

In such circumstances, we found that the only reliable way to get rid of the corrupted data is to erase everything and redo the whole install from scratch, *including deleting the MySql files where the database is physically located* (i.e., delete everything in `mysql/data`, then redo the «`scripts/mysql_install_db`», plus «`bin/mysqladmin -u root password 'passRoot'`», then run the whole `install` script once again).

There are also other ways how the database can become corrupted. In any case, to our knowledge, the receipt that works is the one we just sketched.

Another related problem is that in current versions of MySql, the '`drop user`' command does not always work properly. Thus if you simply drop the user '`xwiki`' and create a new one having the same name (e.g., because you forgot the password...), it won't work (i.e.: the '`drop user`' command can fail: please have a look in the `install` script to see how it's done ; of course in such a case, you also have to redo the related `grant` command) ;

Finally, here is a list of some important config files you could have to edit and/or to watch in order to tune and/or to supervise your newly installed system (the install works well for running an instance of XWiki/Curriki at `http://localhost:8080`. But for things like, e.g. changing the hostname, changing your database's password, etc., you have to edit the config files):

i. the `~/.my.cnf`, which is MySql's main config file ;

ii. the `~/m2/settings.xml`, which is Maven 2's main config file ; this file is set by the `install` script, and you need it this way to be able to compile Curriki ;

iii. the `xwiki.cfg` file, located in `web/target/curriki-web-1.0-SNAPSHOT/WEB-INF/xwiki.cfg`. You need to edit this file to configure XWiki, e.g., setting the hostname, configuring the indexing, and various other things ;

iv. the `hibernate.cfg.xml` file, located in `web/target/curriki-web-1.0-SNAPSHOT/WEB-INF/hibernate.cfg.xml`. This file describes your database, especially the username used to connect to the database, and the password. It coheres with the related part in the `install` script, where we create a database called '`xwiki`', and a user '`xwiki@localhost`', having the password '`xwiki1234`'.

It should be noted that in spite of the notation, this MySql user *does not* corresponds to a user on the server (i.e.: there's no need to create the

same user with the same password on your server, it has absolutely no effect) ;

v. the file `bin/catalina.sh` in your Tomcat installation: this is the script responsible for starting the container, and especially, for tuning the JVM's memory size ;

vi. the log files `logs/catalina.out` and `logs/localhost*.log` in your Tomcat installation are very useful, they contain all Java error and log messages ;

3.5 Installing other Web-Applications

As is the case for Curriki, the other web applications of I2GEO are all built using the maven 2 system. The following simple mechanism thus applies to build and install them.

3.5.1 Installing the Root web application

Within the build directory, checkout:

```
svn co http://svn.activemath.org/intergeo/Platform/RootWebapp/
```

It may be worth reviewing `RootWebapp/src/main/webapp/WEB-INF/web.xml`.

Then, from the `RootWebapp` directory that has just been created:

```
mvn package
```

which creates a directory called `root` which is the web application. That application should be installed at the right place to answer to queries to the paths without a web application (in Tomcat, this is done by the usage of the name ROOT and leaving it in the `webapps` directory, thus we do:

```
rsync -avc target/root/ ../../tomcat/webapps/ROOT/ --dry-run
```

to first see the files affected; then, once confirmed:

```
rsync -avc target/root/ ../../tomcat/webapps/ROOT/
```

A directory `static` should then also be created as a separate webapp:

```
cd ..
svn co http://svn.activemath.org/intergeo/Platform/static/
```

and copied to the appropriate directory

```
rsync -av static/ ../tomcat/webapps/static/
```

More documentation about the root web application of I2GEO can be obtained from `http://svn.activemath.org/intergeo/Platform/RootWebapp/README.txt`.

3.5.2 Installing CompEd and onto-Update server

CompEd is made of three modules, the CompEd web application itself, the CompEd maven plugin, and the ontoUpdate server. Before installing, you might want to check with the translators coordinator what the status of the CompEd bundle is, see page 41.

CompEd Maven Plugin Check out comped-maven-plugin and first install it (in the local maven repository):

```
svn co http://svn.activemath.org/intergeo/Platform
        /comped-maven-plugin/
cd comped-maven-plugin
mvn install
```

OntoUpdate Server Checkout the `ontoUpdate` server:
`svn co http://svn.activemath.org/intergeo/Platform/ontoserver/`

build it (the testing part of `ontoserver`):

```
cd ontoserver
cd updateCoder
mvn install
cd ../web
mvn package
```

deploy it:
`rsync -av target/ontoUpdate/ ../../../tomcat/webapps/ontoUpdate/ --dry-run`
to first see the files affected; then, once confirmed:
`rsync -av target/ontoUpdate/ ../../../tomcat/webapps/ontoUpdate/`

Within the application server directory, a directory called data will be written to. For safety we create that directory with a subversion snapshot which we expose on the web:

```
cd ../../../tomcat
cd webapps
svn co http://svn.activemath.org/intergeo/ontologies/
cd ..
ln -s webapps/ontologies data
```

CompEd Web Application Checkout the CompEd web application:

```
svn co http://svn.activemath.org/intergeo/Platform/comped/
cd comped
```

Then edit the settings contained in the file `pom.xml`: in particular the database connection properties of the profile `mysql-prod` need to be adjusted based on a database you allocate for CompEd to work.

The properties in the `prod` and `prod-repopulate` profiles should also be adjusted to contain the paths:

- `comped.ontologyURL`: the path from which the ontology is loaded at repopulation (per default the svn path)
- `comped.sync.destinationURL`: the path to which to send the XML documents describing the updates (this is the URL of the ontoServer that has been installed above)
- `comped.sync.responseURL`: the URL to which the ontoUpdate server should send the response
- `comped.searchI2G.server`: the other URL to send the XML documents describing the updates
- `comped.searchI2G.browserPath`: the host-relative path to the javascript of the skills-text-box routines

Then build the table digest of the ontology:
`mvn comped:gs2comped`

and populate the database:
`mvn -Pprod-clean-repopulate,mysql-prod clean package`

Then deploy the web application:

`rsync -av target/comped-1.0-SNAPSHOT/ ../../tomcat/webapps/comped/ --dry-run`
to first see the files affected; then, once confirmed:
`rsync -av target/comped-1.0-SNAPSHOT/ ../../tomcat/webapps/comped/`

3.5.3 The Intergeo Search Tool

The Intergeo Search tool is a simple web application built from several maven modules.

Checkout and build:

```
cd ../buildingSpace
svn co http://svn.activemath.org/intergeo/Platform/SearchI2G/
cd SearchI2G
mvn install
```

(currently, Linux, Windows, and MacOSX are supported, depending on the management of temporary files `mvn install` may need to be replaced by `mvn clean install`)

Then deploy the web application:

```
rsync -av web/target/i2geo-search-web-1.0-SNAPSHOT \
  ../tomcat/webapps/SearchI2G/ --dry-run
```

to first see the files affected; then, once confirmed:

```
rsync -av web/target/i2geo-search-web-1.0-SNAPSHOT \
  ../tomcat/webapps/SearchI2G/
```

3.5.4 The Search Tool's XWiki Plugin

Having done this, we need to move around jar files for the class static members to be shared between web-applications allowing macros to be called. This is done by moving an amount of jar files to the shared web-application-container directory. We start within the tomcat directory and consider its shared jar directory called `lib`.

```
  cp ../buildingSpace/SearchI2G/api/target/i2geoAPI-1.0-SNAPSHOT.jar lib/
  cp ../buildingSpace/SearchI2G/xwikicomponent/target/\
searchi2g-xwiki-component-1.0-SNAPSHOT.jar lib/
  cp ~/.m2/repository/com/google/gwt/gwt-servlet/1.5.3/gwt-servlet-1.5.3.jar lib/
```

Although this should be automatically done, you may wish to verify that none of the jars that we have moved actually is within the `webapps` directory.

Finally, the plugin can be activated by adding, in the `webapps/xwiki/WEB-INF/xwiki.cfg` the class-name `net.i2geo.xwiki.SearchI2GXWikiPlugin` after a comma after `CurrikiActivityStreamPlugin`.

4 Editorial Activity on I2GEO

The editorial activity on I2GEO consists of all the edition tasks for *static* pages as well as the management of users. Both these topics are presented in this section.

4.1 XWiki Syntax

XWiki pages are written using a particular syntax, which is an extension of HTML. Velocity and Groovy scripts can also be present. As an introduction, these may be most often used codes:

i. Use of text modes: **bold**, `__underline__`, `~~italic~~`, `--strike--`, `<tt>monospace</tt>`, `<sup>superscript</sup>`, `<sub>subscript</sub>`.

ii. The beginning of a list item is a newline marked with a single * or -.

iii. Paragraphs are separated by two blank lines.

iv. Headers are constructed by just starting a line with numbers separated with points, as in "1.1 Level 2 title", which would be equivalent to HTML "`<h2>`1.1 Level 2 title`</h2>`". To insert a header of level 2 and let XWiki decide which numbering it should have, just use `==`; the number of equal signs indicates the level of the header.

v. A link is denoted by "[pageName]"; a link with a label, "[label>pageName]" where pageName can be either space.page or page.

vi. A table starts and ends with "{table}"; cells start with "`|`", title cells with "`|=`", and rows end with a newline.

vii. A velocity macro is called using the "#macroname(param1, param2, ..., paramN)" syntax. A list of available Curriki velocity macros can be seen at `http://curriki.xwiki.org/xwiki/bin/view/Design/CurrikiVelocityAPIs`; you don't need programming privileges to use them!

viii. XWiki macros are called using the "{{macroname param1="value1" ... paramN="valueN" }}" syntax. They are useful to create particular HTML constructs. A useful macro is {pre} which avoids the execution of the XWiki syntax for the included part.

ix. Through Velocity (and Groovy), data from Java objects can be read and methods be called. In velocity one uses `$varName` to refer to a variable, while in Groovy the simple variable name is used; for example, to obtain the full name of the user. The properties and function names available can be seen in [?].

x. A Velocity script can be anywhere in the text, it starts with the # sign, for example "#set ($var = "whatever")". See [?] for a complete language reference.

xi. A Groovy script between marks "<% def var = "whatever" %>".. See [?] for more details about this language.

There are many more options; see, for instance, `http://platform.xwiki.org/xwiki/bin/view/Main/XWikiSyntax`.

Keep in mind that you can always use plain HTML, so that you don't really need to know this syntax to create new pages. For instance, if some day you have to insert a link somewhere, you could just resort to the classical `<a href="http://.../">text</a>`.

4.2 Pages, translations, spaces and their management

In XWiki, files are grouped logically in spaces for better handling.

Each file can be translated into any of the languages. **Localization** is the **translation** of a whole page, including images and links, as opposed to translation of just a text. Localization of pages has particular subtleties. For instance, in order for texts to fit well into a page, sometimes they must have the right length. Some entities which require customized formats are dates, times, currencies, titles (Mr., Ms.) and so on. Localizing entire documents fully respects all these features, but runs a higher risk of becoming deprecated.

To avoid this, best practice indicates that pages that require an amount of programming be written without language and that pieces of language dependent text are fetched from a *bundle*, a dictionary of phrases. A bundle is a set of property files, one file for each language; we will talk more about property files in the next section.

Bundles belong in spaces. Bundle files belong in bundles or directly in spaces.

I2GEO **bundles setup and differences with Curriki**: In Curriki, bundles are distributed among many spaces, related to the application

development, and they are accessible through the administration page. In I2GEO, however, for the ease of translation, most of the bundles have been moved to the `Translations` space, and they can be accessed at `http://i2geo.net/xwiki/bin/view/TranslatableFiles/`. The bundle CurrikiGWTTranslations, which is used in the Currikulum editor, is loaded in a particular way, and so, for the moment, it will have to be copied by hand over to the old bundle accessed through the administration page (this will be solved; see jira issue 273 at `http://jira.activemath.org/browse/IG-273`). Applications' bundles for CompEd and the Search tool are also loaded in their own particular ways at compile time, and so the content of the files accessed at the Translatable Files page will have to be copied over by hand before recompilation to the files in the directories `intergeo/Platform/comped/src/main/resources/ApplicationResources_xx.properties` or `intergeo/Platform/SearchI2G/api/src/main/java/net/i2geo/api`.

How to create a page: use a browser to go to the address you wish it to have (for instance, `http://i2geo.net/xwiki/bin/view/MyPage/Here`; make sure you are using English; click on the edit button with the pencil icon on the top, left corner of the page; write something; save.

How to edit, rename, delete pages: use a browser to go to the page and use the buttons on the toolbar on the top of the page; this is only available if you are in the administration group. You have also options to see the history of a page, print it, and manage its attachments.

How to translate a page: visit it, edit it, change the language using the combo box in the page header, modify it, and save. **Warning: make sure you are using the right language before saving**, or you will modify the wrong file.

How to edit a bundle file: view the file, then click on the edit button with the pencil icon on the top, left corner of the page. It is not advisable to use a browser to go directly to its edition URL, because it's easy to make a mistake, create a useless file, and then forget about it.

How to create a bundle file: edit any file in the bundle, change the language, and save. What you will save is a copy of the file with the default language for the bundle, normally English.

How to create a bundle: edit an existing bundle file, change its BundleName to the name you desire, and save. **Warning: make sure you are editing the file in English when you save**; otherwise, the default language of the bundle would not be English, and this would be undesirable for translators, who would then not start with a copy of the English file.

How to see which bundles get loaded: go to `/xwiki/bin/admin/XWiki/XWikiPreferences?editor=globaladmin&section=Programming&skin=toucan`.

Space management: go to `http://i2geo.net/xwiki/bin/admin/XWiki/XWikiPreferences?editor=globaladmin&section=Programming&space=`

`XWiki&skin=toucan`, and select the space in the "Programming" combo box. You can see a list of spaces at `ListAllSpacesinCurrikiminusGroups`, and you can see a list of pages in a space at `http://i2geo.net/xwiki/bin/view/Admin/SpacePages`.

4.3 Panel edition

In XWiki, panels are blocks of HTML code that are included in many pages. Panels include the left menu (which is called Navigation panel) and other window areas such as the "Sponsors" rectangle or the "Table of Contents" section that appears when viewing an asset.

If a panel has to be modified, go the Administration page and click on "Panels" in the "XWiki Preferences" section; there you can find options to create new panels, and edit or customize existing ones.

4.4 XWiki groups and their management

An XWiki group is **not** a group where users talk, as a Curriki group is, but rather an administrative tool that allows some users to have some access privileges (viewing, edition, erasure, etc) to some resources. Examples of XWiki groups are administrators (have access to almost everything) and translators (have read and write access to bundles). When it is desired that a user have some access privileges, he is added to the corresponding group. Normal users cannot create XWiki groups or even view them.

To manage XWiki groups, go to the Administration page, and click on "Groups" in the "User Administration" section. Click on the group you want to manage, and then on the "Edit" button with the pencil icon on the top, left corner of the page. To **add a new user**, enter its username in the "Add new user" textbox. To **remove a user**, click on the red cross next to its name. To **create a new XWiki group**, enter its name in the "Add new group" textbox.

To modify the list of resources that a group can access, go to `http://i2geo.net/xwiki/bin/admin/Translations/WebPreferences?editor=spaceadmin&section=Rights`, select the space in the "Rights" combo box, and the group in the grid below (you might prefer typing the group name in the text box). By ticking a checkbox in the grid, you grant the column access to the row group in all resources of the selected space as depicted in figure 2.

4.5 Curriki groups

A Curriki group is a tool for users to communicate. Users can create Curriki groups. Belonging to a Curriki group only gives access to the group's resources.

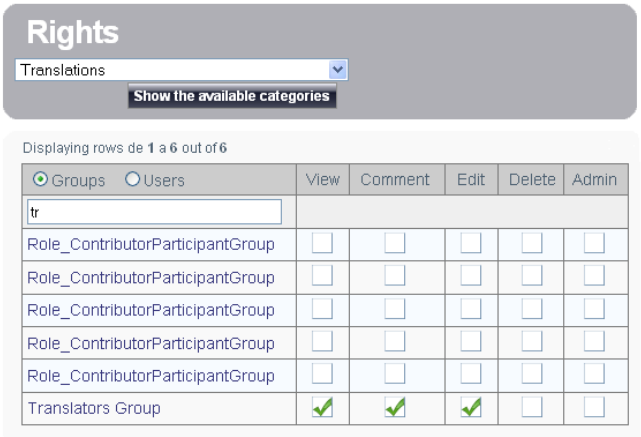

Figure 2: Editing Rights for a space

For more information on Curriki groups, see the I2GEO Basic User's Manual, available at `http://i2geo.net/xwiki/bin/view/Coll_curriki/Tutorials`.

Normal users cannot delete Curriki groups, so occasionally you may receive a request to do it; in this case, go to `http://i2geo.net/xwiki/bin/view/Admin/GroupDelete`. Beware, this cannot be undone!

4.6 Activating, deactivating, and deleting users

Users can be activated, deactivated and deleted at `http://i2geo.net/xwiki/bin/view/Admin/MemberDelete`. Deleting a user should always be a last-ditch recourse, because it cannot be undone and can raise complications; it is preferable to deactivate his/her account.

Manual activation is sometimes necessary because the activation mail never reached the user. It is done by the edition of the user's XWiki-object.

4.7 Managing CompEd users

Being a registered user at CompEd is separate from being a registered user at I2GEO. This a known limitation which we hope users will accept.

Normal users in CompEd have no modification rights. Users can however be activated for the role of *translators* who simply translates the names or to the role *curriculum encoder* who create and maintain all the topics and competencies. The list of users and their roles can be managed in the Administration section of the administrator user.

5 Monitoring the website

5.1 Monitoring the server

The Administration page, `http://i2geo.net/xwiki/bin/view/Admin/`, contains several easy to use tools to check the state of the server and perform some maintenance operations:

1. Check Server Status: `http://i2geo.net/xwiki/bin/view/Admin/RequestsStatus`.
2. See Memory Status: `http://i2geo.net/xwiki/bin/view/Admin/MemoryStatus`, with options to perform garbage collection and flush cache at garbage collection.
3. See Requests status: `http://i2geo.net/xwiki/bin/view/Admin/RequestsStatus`.

In addition, you can subscribe to an RSS feed at `http://i2geo.net/xwiki/bin/view/Main/WebRss?xpage=rdf`, and to the blog RSS feed at `http://i2geo.net/xwiki/bin/view/Main/BlogRss?xpage=rdf`.

Being an administrator and issuing this request will show you all the changed pages, while the normal RSS-reader-oriented feed does not. This provides an efficient way to track all changes done by users such as the introduction of an editorial change, the introduction of a quality review, or even of a temporary file before the metadata input. Some browsers support the direct rendering of RSS streams as web pages (as of today, Firefox and Safari if configured to be the RSS reader).

Section 7.2, on Manual Maintenance, describes other operations related to monitoring the server which lead to maintenance actions.

5.2 Helping users

Administrators must often check the users mailing list to see whether there is a user asking for help within reasonable limits. (Ignore homework help demands,

spam, etc.) This list can be accessed at `http://i2geo.net/xwiki/bin/view/Main/UsersMailingList`.

Among other things, users can send bug reports to this list. They could also create jira issues, so that the administrators should check occasionally whether there is a new issue at `http://jira.activemath.org/browse/IG`. Jira documentation can be found at `http://www.atlassian.com/software/jira/docs/latest/`.

In helping a user, it might be necessary to find his/her account; use the user search tool at `http://i2geo.net/xwiki/bin/view/Admin/UserSearch`. For some questions, the easiest answer will be to refer her/him to the Basic User's Manual, which is located at `http://i2geo.net/xwiki/bin/download/Coll_segido/I2geoBasicUserManual/BasicUserManual.pdf`.

User accounts can be activated or deactivated at `http://i2geo.net/xwiki/bin/view/Admin/MemberDelete`.

5.3 Tracking (Google Analytics)

In order to analyze the behavior of users, an account for I2GEO has been set up at Google Analytics (GA). It is a free account, which means it can track only up to five million visits per month, and we can configure only four goals (defined below).

GA works by receiving notifications sent each time an I2GEO page is downloaded. All pages contain a script which sends a message to GA indicating their URL and an anonymous user code; thus, GA can track what the users are doing. This data is analyzed nightly, and so it is not possible to see the effects of changes on the web immediately . One problem with this approach is that some pages on I2GEO do many things, and so it is not possible to know what the user is doing just by looking at the URL; a good example is the Currikulum Editor, which lets the user view, edit or comment on a resource within the same page. (Note: this may change. At some point, changes in I2GEO's Currikulum editor may be introduced so as to provide GA with the necessary information).

GA allows us to find out which resource is visited most, how much time users spend annotating, which websites are sending us visitors, which country they are from, and what users are searching for, by storing what they enter in the search boxes. By defining goals such as resource duplication, we could measure re-utilization of software. A sample report is at 3.

In GA lingo, a **goal** is an operation detected when a user has gone through a pre-determined sequence of URL's that have to be traversed in a fixed order. For instance, if the same user consecutively visits an ad page, a sales page, and an order confirmation page, the website has achieved a goal (or a **conversion**; a visitor has been converted into a buyer). An **objective** is the place where we would like a visitor to go.

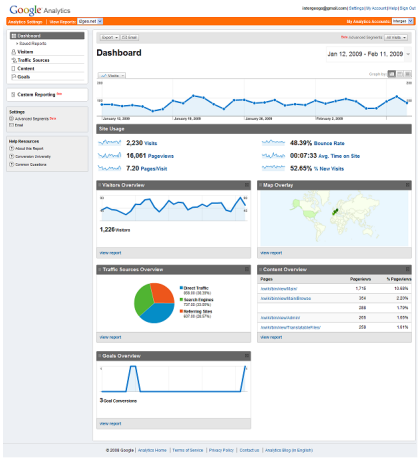

Figure 3: A Google Analytics report for i2geo

In order to **access the** I2GEO**'s GA account**, go to `http://www.google.com/analytics/`, click on "Access Analytics", and enter "IntergeoGA@gmail.com" as our gmail address; the password is the same as that of the superadmin account. Our account code is "UA-6685035-1".

To configure a goal, you should first decide which URLs will define it. Click on "Analytics settings" (top, left corner), then in "Edit" within the accounts table (in our case there is only one row). In the section "Conversion Goals and Funnel" there are four goals, labeled G1 to G4. Decide which goal you are interested in, and click on its "Edit" link. **Math type** indicates how URL's are to be processed. The **goal URL** is the address that, when visited, will indicate that the goal has been achieved. The **goal value** is used to "weigh" the relative importance of different goals; it can possibly be ignored. The **obligatory** checkbox indicates whether a user has to go necessarily through the first step in order to complete the goal; this would be useful if, for instance, we wanted to split "buyers" into "people who came to buy" and "people who bought after seeing the ad". Each step has a name and a URL; the goal URL is "the final step", although it is not formally a step. A goal can have no steps, but it must have a goal URL. Finally, click on "Save Changes" when you are done.

By clicking on "funnel Visualization", a visual representation of the paths that users have followed is displayed as in figure 4.

For more information on GA, see `http://209.85.139.110/analytics/tour/index_en-US.html`.

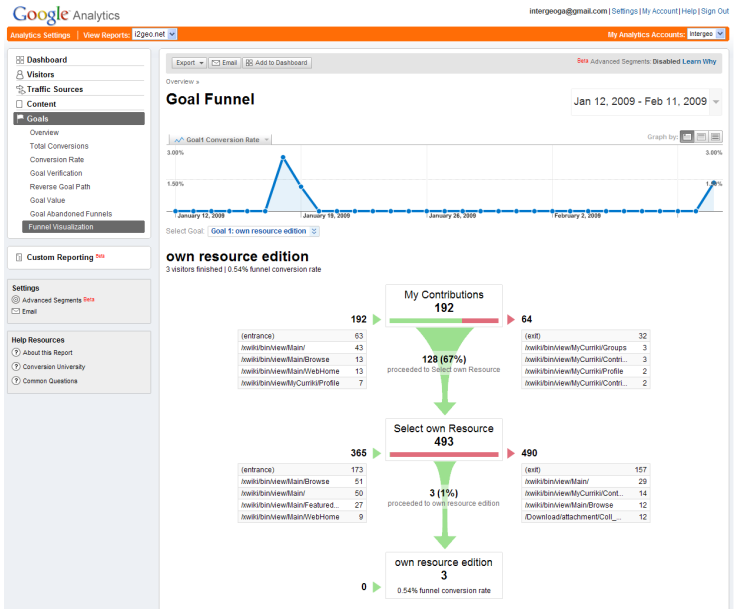

Figure 4: Funnel visualization.

6 Translations Workflows

Since translating a website is quite a time consuming task, and it is desirable that I2GEO be translated to many languages because of its project nature, during the development of the platform it was decided to have a translations coordinator to organize all the tasks related to translations, and to minimize the time translators have to spend in organizing their work. These tasks can be split in two groups: tasks for the translations coordinator, and tasks for the translators.

It is quite possible that the website administrator and the translations coordinator will be different persons.

6.1 Translations coordinator tasks

The translations coordinator has to:

- **Provide general support to translators**; send them the Translator's Manual and maybe the Basic User Manual too, answer questions about character encoding, tell them URL's where things are, etc.
- **Save translators' time** by organizing their tasks. Working with property files is time consuming and prone to committing little errors; before telling translators to do something, do it yourself and see what complications can arise, so that only one persons wastes time. Be precise when describing the tasks to do; mention the files and the properties involved explicitly.
- **Manage the translators group** (`http://i2geo.net/xwiki/bin/view/Group_Translators/`). This is a Curriki group, and it has to be checked occasionally for new messages.
- **Add new languages**. This involves:
  1. Go to `http://i2geo.net/xwiki/bin/admin/XWiki/XWikiPreferences?editor=globaladmin&section=General&skin=toucan`, add the new language to the list, and save. Wait one night for the change to take effect.
  2. Add the new language the in the Global Class Translations bundle, at least in English (the other languages will have to have its translations updated).
  3. In order to add the new language to the skills-text-box, add the new language in SkillsTextBox.gwt.xml as supported language, and then send a mail to the wp4 mail list so that skills-text-box is recompiled.
  4. In order to add the new language to the Currikulum Builder, edit XWiki.AssetClass by going to `http://i2geo.net/xwiki/bin/edit/CurrikiCode/AssetClass?editor=class`, clicking on "language", modifying "Values", and then on "Save".
  5. Go to `http://i2geo.net/xwiki/bin/edit/XWiki/CurrikiSpaceClass?editor=class`, click on the "language" tag, and replace "values" with the result in `http://i2geo.net/xwiki/bin/view/XWiki/SupportedLanguages?xpage=plain&len=long`, replacing spaces for pipe characters ("|").
  6. Add the two language codes in `http://i2geo.net/xwiki/bin/view/Translations/Languages`. Note that his file is not translatable, since it contains the languages as they will appear in the language selection box, and so translators will not edit this file - the translator coordinator has to do it. You will have to ask each translator how his/her language is written in his/her language to include it in this file.

- **Add new translators to the XWiki translator group**. This is the group that allows translators to edit bundles without being administrators. Note that translators must belong to both groups; they are responsible for subscribing to the Curriki Translators Group, but an administrator has to add them to the XWiki Translators group. Go to the Administration page, in the "User administration" section click on "Groups", then on "Translators Group". In order to modify the list, click on the "Edit" button on the top, left corner.

- **Add a new banner**. The image with the "Interoperable Interactive Geometry for Europe" text seen on top of the page header has to be translated; this requires, for each new language, the creation of a new image and an upload of it to the directory `/xwiki/skins/curriki8/`. There are two related properties in the Registration bundle:

  ```
  ## translatable slogan
  header.slogan.png=/xwiki/skins/curriki8/i2g_slogan_en.png
  header.slogan.alt=Interoperable interactive geometry for Europe
  ```

- **Perform Curriki updates** (the parts that affect translation files). If the Curriki files are updated, use the Translation Update Tool to automatically introduce the changes in the I2GEO files; new entries will be marked with a "-TRANSLATE-ME" string. See `http://www.curriki.org/xwiki/bin/view/Messages_Group_CurrikiTranslations/TranslationUpdateToolDescription` for details. The normal procedure would consist in the translations coordinator updating all the translations (this will require some amount of time), sending the files with new entries to the translators, and checking that they upload the revised files.

- **Keep a backup of translations**. Even though translators are likely to keep copies of their work, and the website administrator is responsible for making backups regularly (see section 7.1 on Automated Maintenance), it may be a good idea that the translations coordinator keep a copy of all the translation files. Remember that some translators may be difficult to contact. A note of caution: translators may work on the I2GEO files at any time, so you should always think of your copies of the translation files as obsolete. For the same reason, if you add a new property to the files, tell the translators by leaving a message for the group and sending them an email, or else you risk them overwriting your changes. The language files are stored at directory `Platform/i2gCurriki/wiki/src/main/pages/Translations`; they can be refreshed by running the script refresh-from-i2geo.sh.

- **Coordinate the work of developers and translators**.

  As developers introduce new improvements, new texts will have to be translated. You are to coordinate both teams so that everybody can do their work efficiently and finish in time.

New properties. Occasionally it will be necessary to add some new properties to the translation files; this could be caused by a Curriki developer or an I2GEO developer. You have to add this property in the English files. Quite possibly the developer will tell you in which file and in which place within the file it should go; else, you will have to decide this. (The property will work no matter where you place it, but it should be in an easy to find place). Consider adding some comment describing the context in which this property appears, so that it is clear whether it should be translated in singular or plural, masculine or feminine, etc. Then send a message to the Curriki Translators Group notifying the translators about the new properties; this message should be very clear as to the file, location, property names, and comments, so that only one person (you) has to spend time figuring out these details.

CompEd and the Skills Text Box Editor. The CompEd bundle is loaded by CompEd in a particular way: during its building (or compilation) it reads the files in the directory `src/main/resources/ApplicationResources_xx.properties`. This means that the files that the translators can edit are not the ones that CompEd reads; and, moreover, that changing the files in that directory won't modify the texts shown by CompEd until the whole program is rebuilt again. When CompEd is going to be rebuilt, the person in charge of this task should notify you in time, so that you can tell some translators to finish off that file, and then you should "copy the files by hand" from the Translations space to that directory. The same happens with the Skills Text Box Editor, whose bundle is SKBi18n, and the files are in `intergeo/Platform/SearchI2G/api/src/main/java/net/i2geo/api`.

CurrikiGWTTranslations. The Currikulum Builder also loads its property files in a particular way; concretely, it searches for them in the bundle CurrikiGWTTranslations within the XWiki space, but the translators edit the CurrikiGWTTranslations bundle within the Translations space, and so their changes will not be seen immediately. When a modification is done by a translator, you should copy the corresponding file from the Translations space (accessible through the Translatable Files page) to the XWiki space (accessible through the Administration page, link GWT). Unlike for CompEd, no recompilation is necessary, and so the changes will be visible immediately after copying the file. This problem will be solved at some time; see jira issue 273 at `http://jira.activemath.org/browse/IG-273`).

Some of these tasks will require that the translations coordinator be an administrator of I2GEO.

If some problem arises related to the translation files, remember to read section 4.2. The fact that the bundle structure of I2GEO is different to that of Curriki may cause complications.

There is a tool, available only to administrators, which will find out which file properties have not been translated. In the Administration page, search for the "Compare Tool" next to the list of files to be translated, and click on it. Select the file, "English" on the left, another language on the right, and click on "Compare".

6.2 Translators tasks

Translators have to:

- **Read the Manual for Translators**, which can be found at `http://i2geo.net/xwiki/bin/download/Coll_segido/IntergeoTranslatorsManual/IntergeoTranslatorsManual.pdf`.
- **Subscribe to the Translators group** and check its messages occasionally. (Here it is meant the Curriki translators group; translators may not be aware that there exists an XWiki translators group.)
- **Translate the files** at `http://i2geo.net/xwiki/bin/view/TranslatableFiles/`.

7 Maintenance

The maintenance of I2GEO consists of many tasks (as is common with most web systems), some of which are automated and some of which require careful monitoring and/or optimization, based on the actual usage.

7.1 Automated Maintenance

Automated maintenance works by maintenance scripts invoked through cron-jobs. They can be found in the subversion and should be tuned for the local installation, in particular because they have to contain passwords:

1. `nightly.sh` is run every night, performing the XWiki backups.
2. `refresher-1hours.sh` is run every hour. It requests the regular CompEd ontology reset.

These scripts are monitored by their output, logged per email at every execution. They can be obtained from:
`svn co http://svn.activemath.org/intergeo/Platform/maintenance/`

I2GEO performs backup scripts at two levels:

1. At the database level. A backup is performed every night within a large SQL dump. Dumps are kept every last day of the week, once every week of the last month, and every month of the last year. As of today, the last dump of the Curriki database measured (compressed) 141,033,984 bytes, while the CompEd database measured 99,710 bytes.
2. At theThe XWiki level. The XWiki environment offers an export capability where the pages and all their histories are saved within XML documents. This export separates application files and user files: as of today the last exported measured (non-compressed) 129,184 bytes for the application files and 658,932 bytes for the user files. Such a backup is performed every night, running the nightly script and is concluded by a restart of the servlet container and a re-indexing of SkillsTextBox.

7.2 Manual Maintenance

Several other maintenance operations can be realized when the need arises — for example when suspicious behaviour is observed or when the quality of the data should be enhanced. We list the possible actions.

Caches Refresh Each directory of the `static` web application (as created in the Root web application installation) may be the cache of some other piece of content. Up to now, this is the case for `BigListTraces.html` (which should be pulled from `i2geo.net/xwiki/bin/view/XWiki/BigListTraces`) and of all files in `JSTrans`; a script `refresh.sh` refetches these.

JSTrans upgrade After changing a phrase that is used in the javascript code (including GWT code such as skills-text-box, including the "add a resource" sequence of dialogues), please also invoke the JSTrans Upgrade and empty your cache so as to see the changes appearing. Just visit `http://i2geo.net/xwiki/bin/view/Admin/UpgradeJSTrans` to upgrade JSTrans.

Clean-Asset-Temp The *asset-temporary storage* is used at each creation step of resources, e.g. to hold the content of a file upload that has no correct name yet because it hasn't yet been tagged with metadata. This space `AssetTemp` sometimes contains *left-overs*, which can be cleaned thanks to the web functions at: `http://i2geo.net/xwiki/bin/view/XWiki/clean+assetTemp`.

Take CompEd Offline In some situations, it is undesirable that CompEd users continue working. One of them is when work on the ontology with a full ontology editor is being planned. A call to `http://localhost:53080/comped/setReadOnly.html` can then be made, while `http://localhost:53080/comped/unSetReadOnly.html` or a server restart reverts the status. In the example above `localhost:53080` is the host part of the URL, which indicates the direct connection to the servlet container.

File Check The File Check system is used to screen new additions for educational appropriateness; it allows removing spam, offensive content, viruses, and material that doesn't work, such as broken links or empty templates. When new files are uploaded, or existing content is modified, they are published immediately, but they are included in a list of files to be checked. The interface to this tool is at `http://i2geo.net/xwiki/bin/view/FileCheck/`; since the list shown may grow quite large, it is possible to ignore the collections of some users, see `http://i2geo.net/xwiki/bin/view/FileCheck/ExcludeList?bc=`. Files do not have to be approved or removed immediately; they can be marked for special review because of technical reasons (software which might be

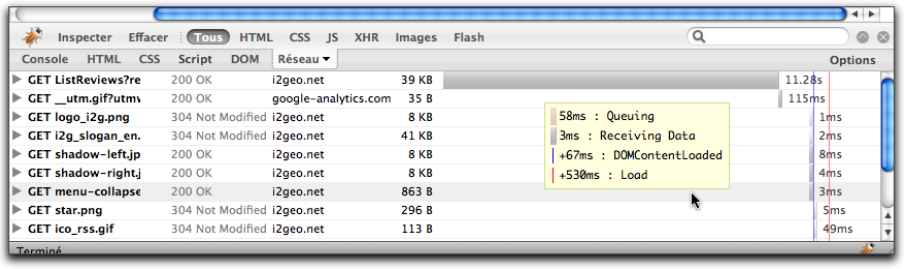

Figure 5: Measuring delivery performance on `http://i2geo.net/xwiki/bin/view/QF/ListReviews?resource=Coll_cdording.Sommedesanglesduntriangle`, the list of reviews of the currently featured resource.

malicious), because they are written in a language which cannot be screened out automatically and so a native speaker is needed to confirm its educative value, or because the material is suspected to have copyright problems. Only resources not reviewed, or reviewed with a status of "OK", are published. Resources with other status are not accessible to the general public; they could be erased or approved. A more detailed description of the File Check system is available at `http://www.curriki.org/xwiki/bin/view/Coll_curriki/FileCheck?bc=;Coll_curriki.EOU3HelpCollection;Coll_curriki.AbouttheResourceRepository`.

Section 5.1, about monitoring the server, describes some operations which may be useful in getting information on the server status.

7.3 Optimization Strategies

The I2GEO platform may be considerably optimized by a few measures, all of which have been postponed to a near-realistic set-up which allows to measure the effective performance bottlenecks and potential optimization spaces. The tools used to measure the network load for a browser are provided with developer tools for the browser. On Safari, the web inspector is available, on Firefox the Firebug extension; both provide a graphical display of the time taken by each request. A few optimizations based on this have been done recently.

An example measure is provided in figure 5. It displays the load-time following the request of the list-of-reviews of the current featured resource. This list shows that the total delivery is almost entirely impacted by the page delivery while other resources take no time (but may take so if network performance is lower).

Another measure can be taken by imitating the actions of users with the browser setup of a typical user and measuring the response time.

Cache management is the crucial aspect of any web server: it enables the server not only to avoid the retransmission of some of the files, but it also allows

browser-internal representations to be preserved: a script's compilation or a picture scaling, both in memory, to be preserved. To this end, most of the resources that each I2GEO page references are *static* in the sense that they are not permanently changing; the servlet-container recognizes this and negotiates the proper headers with browsers. However, the rendering of XWiki pages is generally not cached, since they change a lot. This is acceptable, since they are mostly less than 50 kilobytes in size.

One elementary optimization strategy has been to convert some of the resources from scripts or style-files to become static resources which become subjects of manual maintenance as in section 7.2. That includes the javascript translation files. This has cut the download of each page by several seconds. These cached pages are in the `static` web application, and need to be refreshed every time the translations are changed. To this end, the script `recreate.sh` needs to be invoked.

Most of the optimization strategies are a posteriori and the I2GEO platform still has room for more optimization. Among others, using `context.setCacheDuration(60)` to most pages that are not fully dynamic so as to indicate that the result of the XWiki rendering should be cached for 60 seconds. Several other strategies are explained at [?].

Finally, simplification of the user interface library is a potentially important optimization approach. It shall be possible after finalizing the user interface (e.g. by dropping the user interface tree library, called YUI, used in Curriki's topics and levels annotations).

7.4 The news system

In order to publish a piece of news in the front page of i2geo, you have to write it in a new entry of your blog, add the word "news" to its list of keywords, and publish it; this will indicate that this blog entry is a piece of news of general interest that should be shown to everybody.

If you want to publish a piece of news that will interest only Spanish Intergeo members, do not add "news" but "news_es", and write it in Spanish; that is, add the 2 letter code of the language to "news_". News that are of interest only to users from the United Kingdom must have the keyword "news_en".

If you want to remove or modify a piece of news you wrote, just delete or edit its blog entry.

Each i2geo visitor is shown the 5 most recent news that are either of general interest or written in his/her chosen language.

News are not usually translated; if needed, they could be written in several languages, and only one translation would be shown to each visitor.

8 Limitations and Difficulties in the Current System

The choice of Curriki as base platform was made because only minimal adaptations were needed and it is a base-platform backed up by a live open-source development community. Unfortunately, this has also been the choice of a base platform that has rarely been deployed in other instances. Although the Curriki team did help us with a large amount of issues, starting with the installation instructions, our experience into re-using this software and all its features has been plagued with failures and blind searches.

A few axes of difficulty are enumerated below:

- Curriki, as almost any other web server application, uses a very large amount of dependencies, whose management, although automated by Maven, is fragile. Curriki could have solved this by relying on releases only, but this policy was not taken: many snapshot releases contain dependencies which often disappear from the network of Maven repositories.

- Installation instructions might be more precise and more complete. For example, it took us several tries to realize that MySQL was needed; our first attempt was done with PostgreSQL, which made the groups feature fail.

- A global documentation of Curriki would be useful. This work is partially done at `http://curriki.xwiki.org/`, where design documents of the data model objects and documentation on the velocity scripts are available. A few more documents and a tighter integration are desirable (for javadoc plugins for example).

- The multi-layered structure of XWiki makes it uneasy to decide which is the best language for adaptations. The Groovy language seems the best for encoding case dependent behaviours, but it fails at accessing the very rich library of velocity user-interface macros of Curriki. Conversely, case-dependent behaviour in Velocity is error prone and fragile. No real solution has been found, except our learning and the writing of our own library code to help in rendering.

- Curriki has the ability to encode many mime-types and their dedicated behaviours; to this date, we have failed at activating this facility. The same holds for an action on the server-side API rendering of the interactive geometry constructions (in HTML code to use it or in pictures). Curriki has announced that this will be fully revised in their next release.

Another difficulty of our approach is related to the patching approach: most of the changes we have made on Curriki cannot be properly isolated in the modules planned for adaptations; instead, we have had to do these directly in the core. For example, code for the back-end asset-model, the GWT-based Currikulum Editor, and the add-path ExtJS code needed to be changed. As a result, the `i2gCurriki` directory is made of patched files which need a thorough revision at every new update. The last upgrade of I2GEO to Curriki `eou2` took about two weeks of intensive work.

Because of these limitations we follow a very slow-paced upgrade mechanism: only release-points are incorporated; the next one may happen some time during spring 2009. All issues we encounter are tracked on the jira tracker `http://jira.activemath.org/browse/IG` which is visible to the users (who can see if an issue progresses) and to the Curriki and XWiki communities.

Because of these difficulties, deployment is done in two stages: a first server called "draft", accessible at `http://draft.i2geo.net/xwiki`, is constructed with the attempts at adaptations to the newer platform. This server can then be taken for a test-drive by many users, who are requested to report on the jira bug tracking system.

Finally, another conservative measure is to keep any strong development outside of Curriki itself, and then integrate it. This includes the rendering infrastructure for interactive geometry files, the search tool and ontology editing processes.

9 Acknowledgements

We are deeply grateful to the Curriki team for the continuous help and support they have provided us.

We have also indebted to the open source community in general and, more in particular, to the Apache Foundation and XWiki. We have benefited from a free Google Analytics account.

10 Bibliography