Here is the meeting minutes

Indexing :

* Sometimes it may be needed to add a language dynamically to the configuration of the workbench. It seems that if the indexer has indexed a certain number of languages, then it throws an exception if it finds another one in the configuration which has not been indexed.

Action : Investigate on this and in case release a more tolerant indexer ☺

* Suggested list- there is an error in the log file ( i.g.,“too many file” )
	+ Action: investigate on this

Ref: http://202.73.13.50:55234/probe/

* Suggest list:
* Wait a delay (1,5 sec) before both submitting query to the server and returning suggestion list to the user
* Tom’s tests on OWL file: currently tested exports done automatically via Protégé API (that is: various methods available from the API over which we have no control) resulted in corrupted OWL files
	+ Last tentative by Armando: if a proper Jena Model can be returned from the DB Protégé OWL Model (the problem here is that all Protégé KB implementation can return Jena models, but some of them may return Jena models whose implementations are not complete, and Protégé DB is the worst under this aspect), then we’ll write all the triples out. Otherwise, we’ll not be able to conduct any check on file and will need to move to DB testing (see next point)
* Testing of PROTÉGÉ-DB vs AGROVOC-DB:
	+ Human Testing: Lavanya and Gudrun will inspect random concepts from user point of view
	+ Unit testing (this has maximum priority, unless last tentative of previous point returns some usable textual RDF code):
		- Sachit provides any available documentation on conversion strategy (and code for the conversion project)
		- Armando writes the unit test, which will be launched on the current conversion as well as on each other possible conversions we will do
		- The same unit test will be adapted then for testing integrity between PROTÉGÉ-DB (once it is itself tested) and future SKOS version of AGROVOC by using SKOS API\
* Testing Environment:

 One Server is needed for testing purpose.

 Action: MIMOS 2 will set up the testing environment as soon as possible.

* Email from Sachit about exact words

/ TODO this is not really respecting the semantics of search, but it should be discussed with Sachit,

 // there is no need for this, the string from the user is just pingponged to Lucene

 // but i'm not sure if getKeyword does not cut any modifier put by the user

 int searchStrategy;

 String regex = searchObj.getRegex();

 if (regex.equals(SearchParameterObject.EXACT\_MATCH))

 searchStrategy = OntIndexSearcher.EXACT\_STRATEGY;

 else

 searchStrategy = OntIndexSearcher.SIMILARITY\_STRATEGY;

Lets say we get these list of terms after indexing

(African rice, rice, ricewood, riceland, jasmine rice, rice flour)

Case 1:

-----------

Keyword: rice

Filter: exact match

OntoIndex Criteria: OntIndexSearcher.EXACT\_STRATEGY

Code: rice

Desired result: (rice)

Case 2:

-----------

Keyword: rice

Filter: contains

OntoIndex Criteria: OntIndexSearcher.SIMILARITY\_STRATEGY

Code: \*rice\*

Desired result: (African rice, rice, ricewood, riceland, jasmine rice, rice flour)

Case 3:

-----------

Keyword: rice

Filter: starts with

OntoIndex Criteria: OntIndexSearcher.SIMILARITY\_STRATEGY

Code: rice\*

Desired result: (rice, ricewood, riceland, rice flour)

Case 4:

-----------

Keyword: rice

Filter: ends with

OntoIndex Criteria: OntIndexSearcher.SIMILARITY\_STRATEGY

Code: \*rice

Desired result: (African rice, rice, jasmine rice)

Case 5:

-----------

Keyword: rice

Filter: exact word

OntoIndex Criteria: OntIndexSearcher.SIMILARITY\_STRATEGY

Code: ???

Desired result: (African rice, rice, jasmine rice, rice flour)

Could you please tell what should be the code for case 5 and verify if the code mentioned above for other cases are correct.