



## **SCY pedagogical agents DV.4**

### *Authors*

*Jörg Kindermann (FhG), Anjo Anjewierden (UT), Lars Bollen (UT), Jan Engler (UDE), Margus Pedaste (UTE), Florian Schulz (FhG), Stefan Weinbrenner (UDE)*

### **Science Created by You (SCY)**

(Project number IST-212814)

Date: 30-10-2011

### **Dissemination level:**

<input checked="" type="checkbox"/>	PU	Public
<input type="checkbox"/>	PP	Restricted to other programme participants (including the Commission Services)
<input type="checkbox"/>	RE	Restricted to a group specified by the consortium (including the Commission Services)
<input type="checkbox"/>	CO	Confidential, only for members of the consortium (including the Commission Services)

**© 2011, SCY consortium**

**Page intentionally left blank**

## **Executive summary**

Agents play an active role in all the central parts of SCY-Lab: they support scaffolding of students working with tools on text production, concept building, experimentation, and also the collaboration between students. Furthermore agents also support other functionalities of SCY-Lab, like the intelligent search, on a level invisible to the user. This report lists all of these contributions, albeit in varying depth, so as to avoid too many duplications from earlier deliverables.

## Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>5</b>
<b>2</b>	<b>Agent scenarios.....</b>	<b>5</b>
2.1	Agent support for building concepts.....	5
2.1.1	Proposing concept map components.....	5
2.1.2	Concept map construction support.....	7
2.1.3	Concept map evaluation.....	10
2.2	Agent support for scientific experiments.....	16
2.3	Collaboration support.....	18
2.3.1	Narrative.....	19
2.3.2	Data mining and pedagogical aspects.....	20
2.4	Agent support for writing hypotheses.....	23
2.5	Table of agents in missions.....	24
<b>3</b>	<b>Agent framework.....</b>	<b>24</b>
3.1	Agent architecture.....	25
3.2	Blackboard design.....	26
3.3	Agent-tool interaction, logging and notifications.....	27
<b>4</b>	<b>Agent technology.....</b>	<b>28</b>
4.1	Overview.....	28
4.2	Collaboration agents.....	28
4.3	Teacher support.....	29
4.4	Agent tool communication.....	30
4.5	Agent support for search.....	31
4.6	Prolog agents.....	33
4.7	Table of all agents.....	33
<b>5</b>	<b>Lessons learned and conclusion.....</b>	<b>35</b>
	<b>References.....</b>	<b>36</b>
<b>6</b>	<b>Appendix: Technical aspects of group formation.....</b>	<b>38</b>

## 1 Introduction

Pedagogical agents have been promoted from the start as one of the central and new concepts in SCY. The agent idea and its realization underwent an evolution during the project lifetime that was guided by experiences in SCY test runs with and without agents involved as well as more isolated analyses and experiments that involved particular parts of SCY-Lab (tools and agents) and also data from outside of the project. Not all of the agent concepts that we developed and described in the previous deliverables proved to be feasible and valuable in the context of SCY-Lab, its missions and tools. This report delivers the full version of those agents that have been identified to be useful and were implemented during this evolution process.

The deliverable is structured in several parts: Section 2 gives an overview of the different agent scenarios that have been developed and integrated into one or several missions. Section 3 explains the agent framework as an integral part of the SCY system design with its server and client parts. Section 4 describes the technical details of agent implementation. There is also an appendix (Section 6) on the technical details underlying the group formation agents, which have not been explained in the previous deliverables.

## 2 Agent scenarios

This section describes agents in the SCY context, using six scenarios. Some of them have already been introduced in previous deliverables. Those scenarios which are new are introduced in detail, including a user story.

### 2.1 Agent support for building concepts

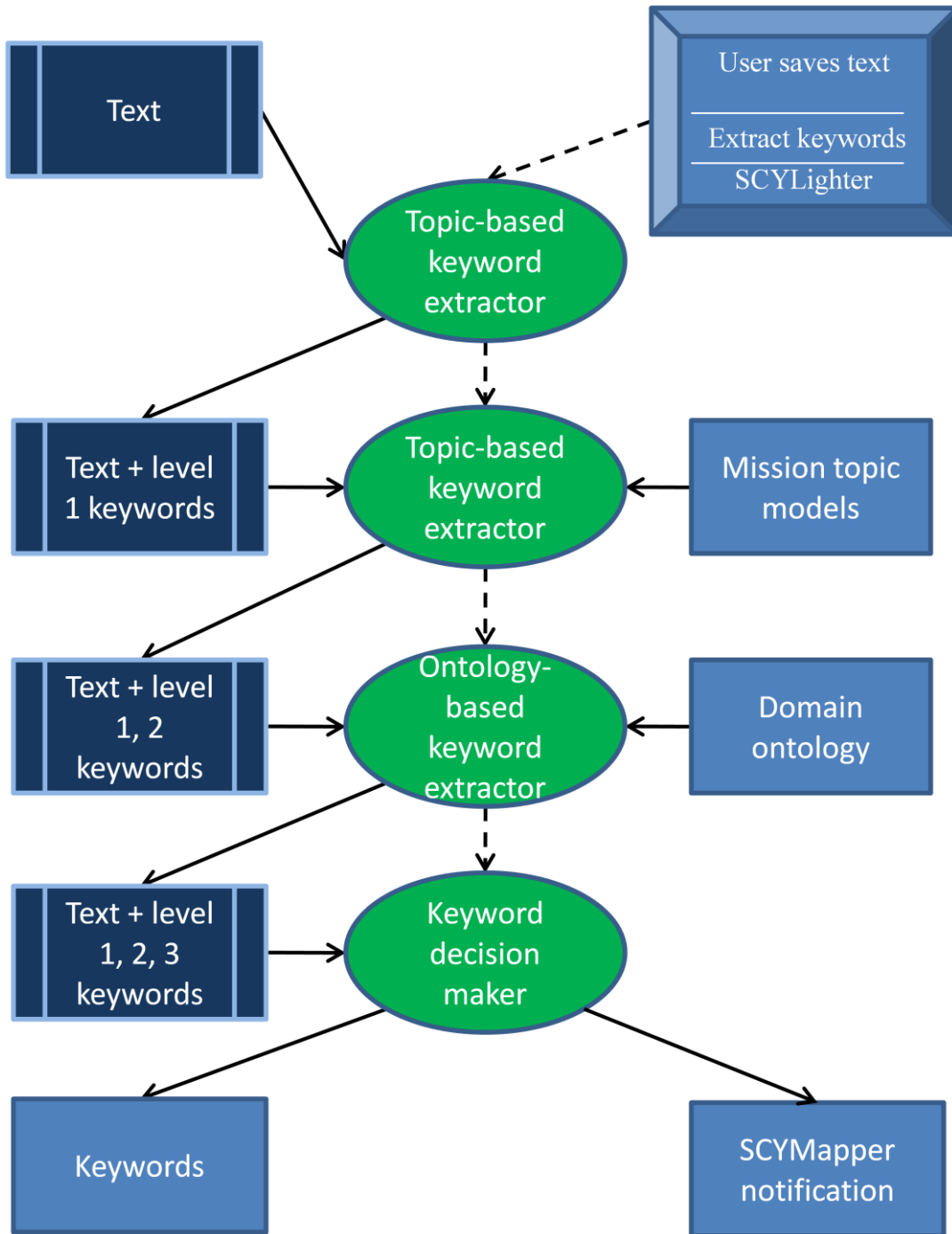
The development of topic-adequate concepts is a central pedagogical principle in SCY, which closely relates to ELO-based learning. In concept maps learners can structure their knowledge about a domain by drawing a graphical representation in which nodes represent concepts and edges relations between the concepts. Students use the SCYMapper tool to represent and store their concepts in ELOs. Agents should therefore support scaffolding of the SCYMapper tool. The three following groups of agents have been created for this purpose.

#### 2.1.1 Proposing concept map components

This scenario was described in deliverable DV.3, section 4.3. The narrative there was closely related to the experimentation scenario, and the user interaction with the agent that proposes keywords for use in a concept map was not easily identifiable. We therefore include a modified narrative here.

*Eric just started working on his first SCY mission, the CO<sub>2</sub>-friendly house. He has seen the challenge and has done the first ideas assignment. He is now part of the expert group on thermal energy and is now exploring a list of resources on thermal energy with his internet browser. The task is to make a concept map with the main concepts. Eric opens the SCYMapper tool and starts working. He finds it difficult to transfer the main ideas in the texts into concepts. Eric marks text sections he finds important, using the SCYLighter tool (which consists of a browser plugin) and stores them in an ELO. The keyword extraction agent notices that Eric opened the SCYMapper tool and also saved text sections from the browser. The agent extracts the relevant keywords from the text ELO and sends the list of keywords to Eric's SCYMapper tool. The tool displays the list, and Eric uses some of the keywords to add concepts and relations to his map.*

The keyword extraction agent is a general agent that is active in several scenarios. The technical details reveal that it is not a single component, but a chain of interacting agents, as is displayed in Figure 1.

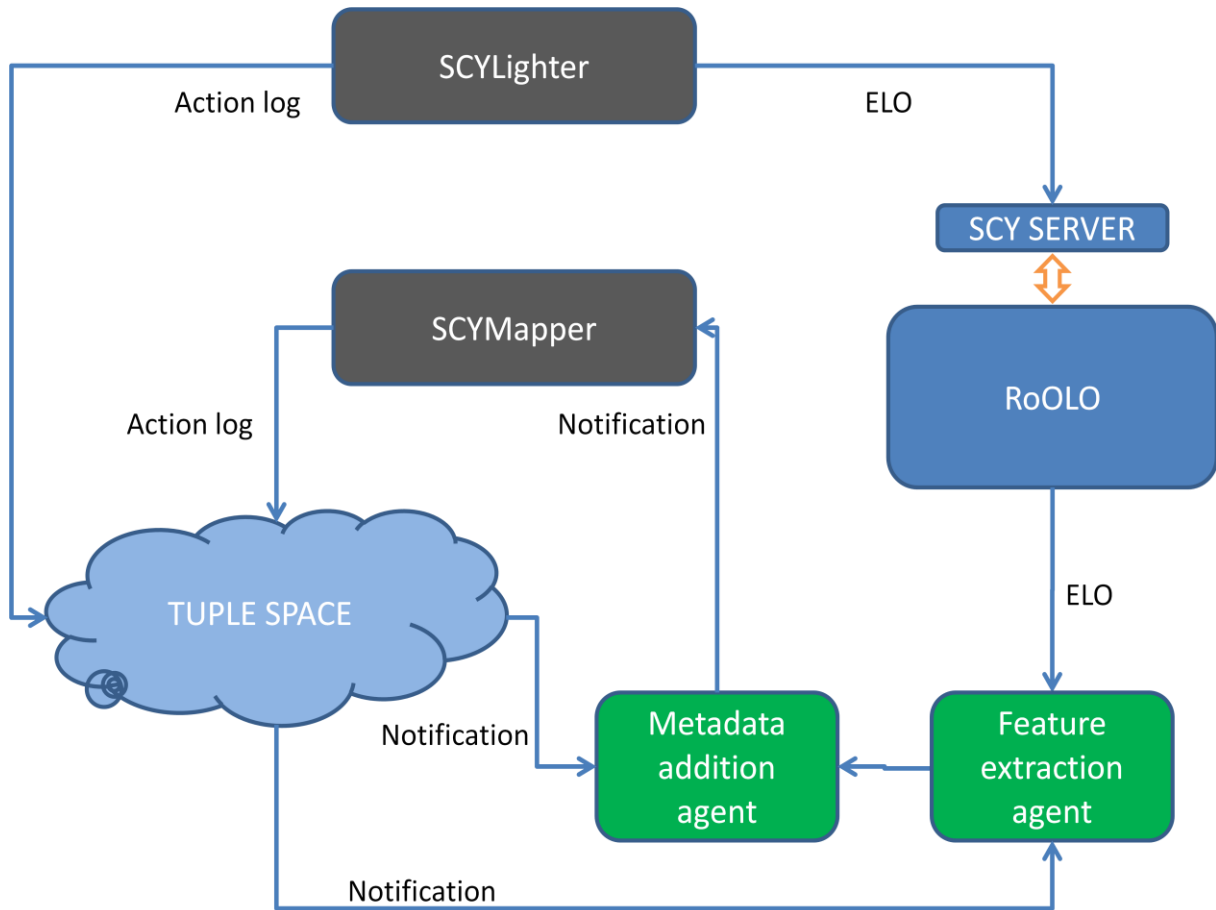


**Figure 1: Agent chain for keyword extraction from ELO content**

In this case it scaffolds the user with terms to name concepts and relations in a concept map. It therefore interacts with the specific tool for concept map design, SCYMapper. It is triggered in a context, where SCYMapper is open and a text ELO with input from a browser has been saved, using the SCYLighter tool.

Figure 2 shows the control flow in this case: The save action of the SCYLighter tool is recorded in the tuple space as an action log while the ELO is sent to the SCY server to be stored in the RoOLO database. When the user also opens the SCYMapper tool, its action log

triggers the feature extraction agent and the metadata addition agent (see the notification arrows). The extracted keywords are transferred from the feature extraction agent to the metadata addition agent. (This is indicated by the unlabeled arrow. Actually the transfer is done by writing and reading the tuple space. This was left out of the diagram for simplicity.) Finally the scaffold terms are sent as a notification to the SCYMapper tool.



**Figure 2: Agent chain for keyword extraction from ELO content**

There is one change with respect to agents proposed in deliverable DV.3. It was decided not to use the peer concept matcher because of the following reasons: the student concept maps that have been produced in the school trial runs show a large variety of structures as well as concept and relation names. Both features lead to a reduced score of matches. The situation could be improved by prescribing the names of concepts and relations that can be used. It was however decided early in the project not to restrict the freedom of concept map design.

### 2.1.2 Concept map construction support

In several missions the construction of concept maps is a central activity. Therefore, this process is being supported by a group of agents that use the knowledge of the mission ontology to propose the most important but missing nodes and links. This scenario has been described in many details in DIV.2 and there have not been any significant changes since then.

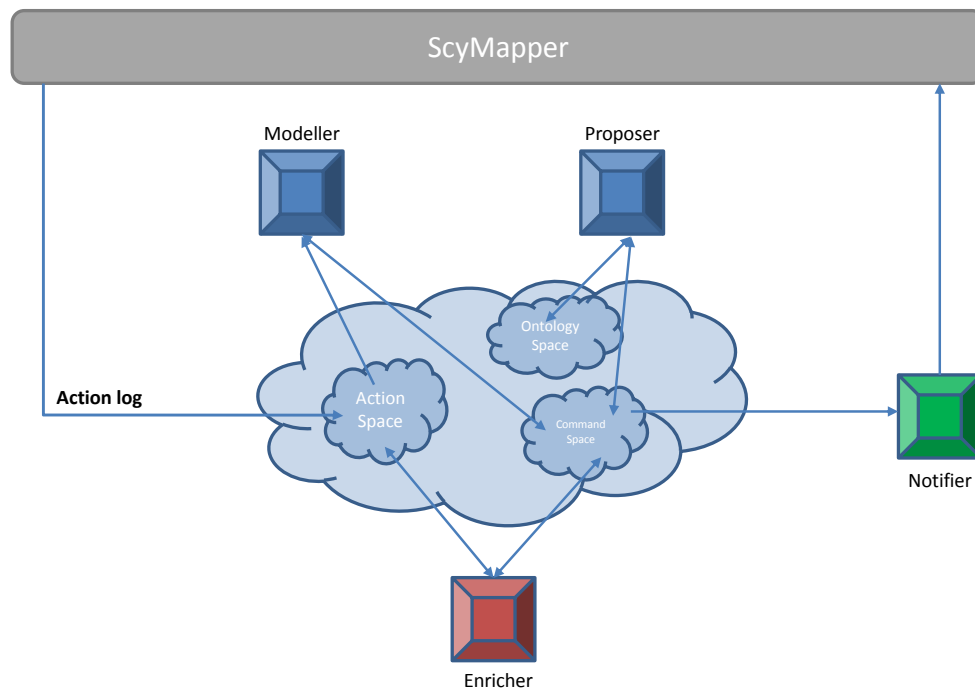
A possible scenario could be the following:

*Alice is a student, who is attending the biology course at high school. Her class is using the SCY software. In the current period, the class is working on the topic of*

*climate change and CO2 emissions. The today's task is to summarise a text about global warming. Therefore, Alice is reading a text, which deals with several topics concerning the impact of global warming as well as environmental politics. During reading the text, she highlights the core concept on the text sheet. After finishing that, she opens the SCYMapper inside the SCY-Lab and start to work. She inserts a concept called "global warming" as the central topic of this text. Then she arranges other important concepts around this core concept and connects these items with each other. After a while she is not sure which of her marked text fragments she should integrate in the (quite big) concept map. In this moment, a help box appears on the right side of her screen showing five concepts that she could integrate. She notices that two of the five concepts are already in her map. She uses a button next to the proposals to indicate that there are synonyms in her concept map. After that they disappear from the proposal list. She creates three new concepts named like the remaining proposals. Then she notices that there are not only concept proposals, but also some information about missing connections. She thinks about the connections and integrates them into her concept map. After some time, Alice thinks that she finished the task, even if there are some other concepts, which are proposed in the help box and she submits her work to the teacher*

The main tasks to realise such a scenario are to recognise if the creation of the concept map is not progressing well enough, to identify concepts and relations that are to be expected from a given text and to compare two concept maps and determine the most relevant missing concepts and relations. Four agents are working between these three spaces as shown in Figure 3:

- The *enricher* is a so-called decision maker agent that only coordinates other agents. In this case it recognises the need for help and triggers other agents. After it has received the responses of the other agents, it writes a notification tuple in the command space.
- The *modeller* is responsible to constantly monitor the user actions in order to tell other agents the current state of the user's concept map.
- The *notificator* just waits for notification tuples to be written in the command space and routes them to the user.
- The *proposer* gets a request from the enricher. Either this request is to estimate the quality of a learner's concept map or to return the next proposals for a learner.



**Figure 3: Architectural overview of the agents and spaces involved in the scenario.**

If the proposer is asked to generate concept or relation proposals, it starts by fetching the current concept map of the user from the modeller. Then, it extracts from the text all relevant concepts. This is done in several steps: First, all ontology terms that occur in the text, are found. Then, all words in the text are stemmed and compared to the stemmed words in the ontology. After that, the same is done for the semi-automatically added keywords. Next, all the ontology concepts that have been found in that way are linked by using the ontology relations. This reference concept map is compared to the learner's concept map and all the missing concepts and relations are determined. In order to propose the most important concepts, for each concept of the reference concept map a relevance is calculated that depends on the graphical centrality of the concept and the reason why it has been added. Ontology terms that appear literally in the text are of course more relevant than keywords that have a common stemmed form with one of the words in the text. The concept and relation proposals that have been determined in this way are sent to the learner by default notifications and SCYMapper interprets them by showing them on the right side in two separate lists (c.f. Figure 4).

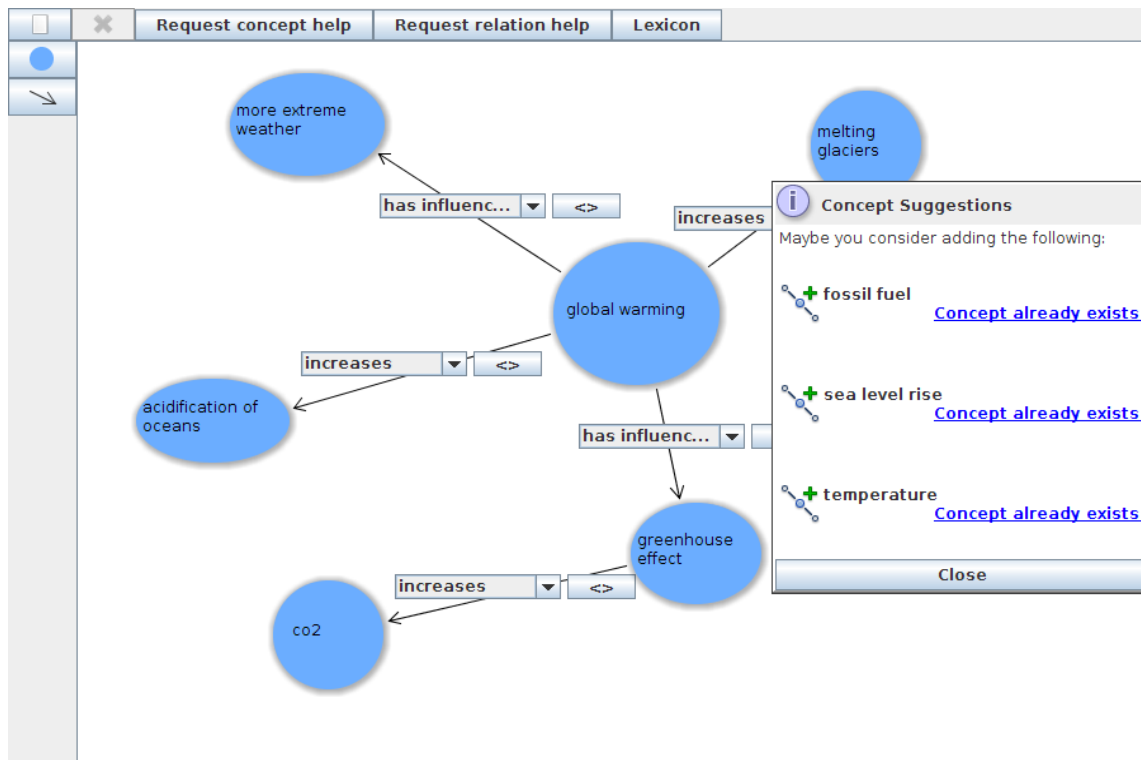


Figure 4: Agent scaffold on concepts in SCYMapper

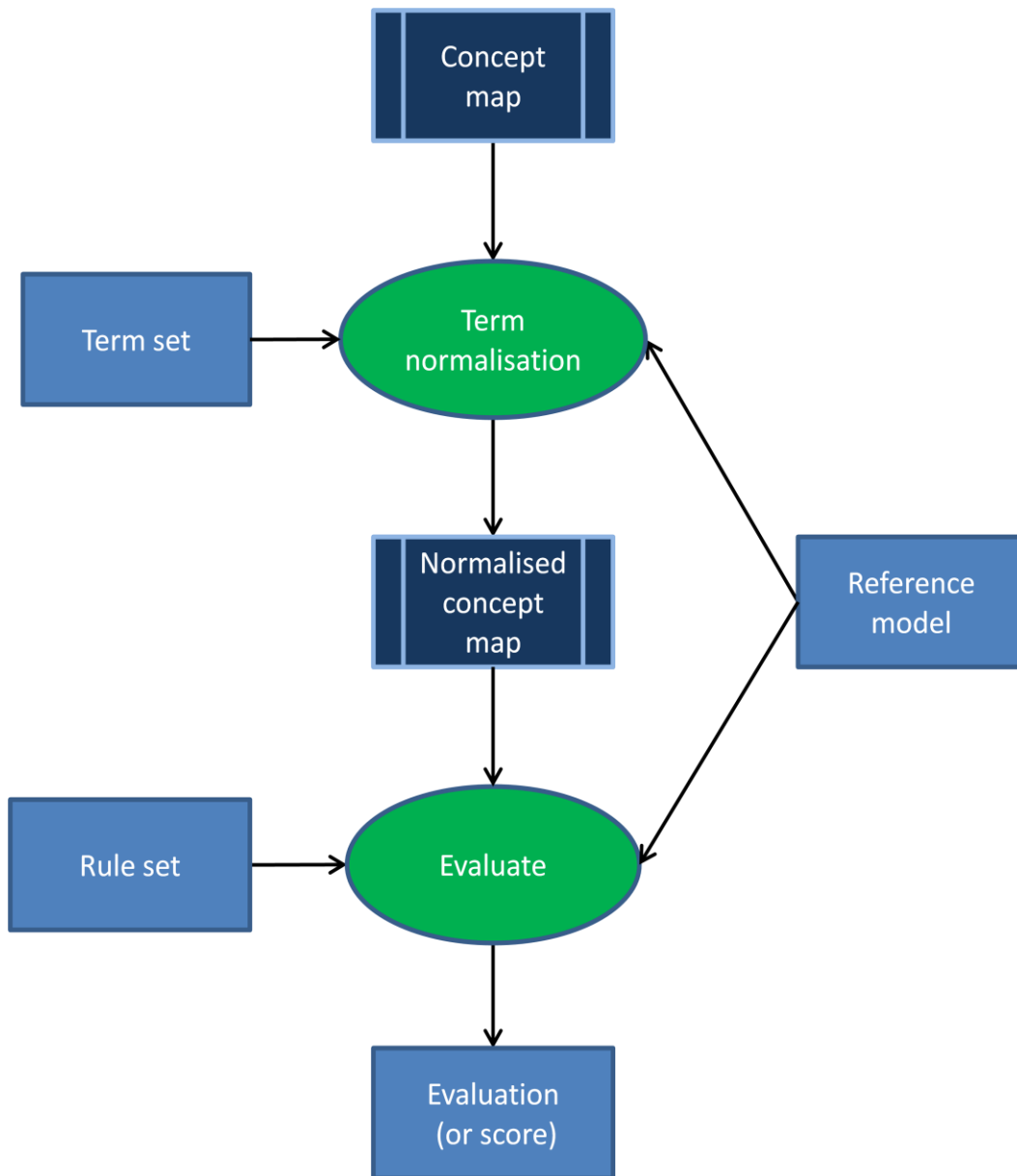
### 2.1.3 Concept map evaluation

The concept map evaluation (CME) agent evaluates concept maps created in SCY missions and uses the result of this evaluation to provide feedback to learners. In this section we describe the CME agent and how it is being used in several SCY missions.

#### 2.1.3.1 Specifying concept maps

In the most general case concept map evaluation requires three specifications: a *term set*, a *reference model* and a *rule set* (see Figure 5). In a term set mission developers define terms learners might use for concepts and relations. Term sets are obviously language dependent. An example of the definition of the term "mobile phase" in the French version of the forensic mission is:

```
<term name="Mobile phase">
  <string>phase mobile</string>
  <string>eluant</string>
</term>
```



**Figure 5: Specifications and steps for concept map evaluation**

A reference model defines the structure of the concept map. A reference model is in most cases an "expert map", but it is also possible to use a concept map of a peer as a reference model. An example of the specification of a reference model is:

```

<reference_model name="forensic" notation="cmap">
  <nodes>
    <node term="Mobile phase"/>
    <node term="Chromatography"/>
    <node term="Rf"/>
    <node term="Solvent"/>
  </nodes>
  <links>
    <link term="is a"/>
    <link term="has a"/>
    <link term="depends on"/>
  </links>
  <edges>
    <edge tail="Rf" head="Mobile phase" link="depends on"/>
    <edge tail="Mobile phase" head="Solvent" link="is a"/>
    <edge tail="Chromatography" head="Mobile phase" link="has
a"/>
  </edges>
  ...
</reference_model>

```

In this partial example from the forensic mission the concepts and relations between "mobile phase" and neighbouring concepts are specified. (The reason for using "node" and "edge" rather than "concept" and "relation" is that the same specifications are also used for other graph-like structures, for instance system dynamics models in SCY Dynamics.) Reference models are normally defined in English. Nodes and links in the reference model refer to the terms in the term set which contain the language specific strings. For instance, if a French learner creates a concept labelled "eluant" in his concept map then it would match the "mobile phase" node in the reference model.

To make it easier for mission developers the agent can generate a reference model and the corresponding initial term set from a concept map ELO created with SCY Mapper.

A final type of specification is a rule set. Rule sets are used if the evaluation is not a simple comparison between a learner map and reference model. The rule set below defines when the concept map of a learner in the ECO mission has a sufficient number of concepts for the topic of photosynthesis. The rule specifies that at least five of the given concepts must be present in the learner's map.

```

<rule name="B2 - photosynthesis" result_type="bool">
  <true at_least="5">
    <node term="CO2" status="present"/>
    <node term="Water" status="present"/>
    <node term="O2" status="present"/>
    <node term="glucose" status="present"/>
    <node term="energy" status="present"/>
    <node term="light" status="present"/>
    <node term="photosynthesis" status="present"/>
  </true>
</rule>

```

### 2.1.3.2 Evaluating concept maps

The most critical step in concept map evaluation is determining whether a label a learner uses matches a string in the term set. In Figure 5 this step is called term normalisation. The CME agent uses an algorithm based on edit distance to compare a learner label to all strings in the term set and then selects the best match. The algorithm takes care of misspellings (e.g. "pahse" rather than "phase") and by default disregards word order (e.g. "phase mobile" versus "mobile phase").

After term normalisation, the reference models, and optionally the rule set, are used for the evaluation. Finding matching nodes and edges in the learner map and reference model is generally a straightforward procedure.

In the rule set the status of a match can be expressed as one of "present" (in both learner and reference), "missing" (in reference, not in learner), or "redundant" (in learner, not in reference). The status can be specified for specific nodes, but also for all nodes. For example, the rule that counts all "missing" nodes is:

```

<rule name="number of missing concepts"
result_type="numerical">
  <node status="missing"/>
</rule>

```

This rule is used for feedback to the learner, for instance with a feedback message like "You are missing X concepts in your concept map".

In the forensic mission the following rule for listing all redundant concepts is used to base feedback on:

```

<rule name="redundant concepts" result_type="enumeration">
  <node status="redundant"/>
</rule>

```

### 2.1.3.3 Scenarios in SCY missions

The CME agent is used in three SCY missions. In these missions the concept mapping task and the type of feedback to learners are different for each.

For all missions it was decided to keep the communication between the concept mapping tool (SCYMapper) and the agent as simple as possible. Evaluation is triggered when the agent sees an "ELO saved" action from the learner. The agent then looks up which ELO is saved, loads the corresponding specifications (term set etc.), performs the evaluation, composes a feedback message and finally sends the message as a notification (see Section 3.3) to SCYMapper. The learner can then click on a blinking icon to read the message.

#### Energy fact sheet in Pizza mission

The energy fact sheet is a concept map in which all concepts are given in advance. The concept mapping task is to drag and drop concepts that are not yet part of the map to the correct location (see Figure 6).

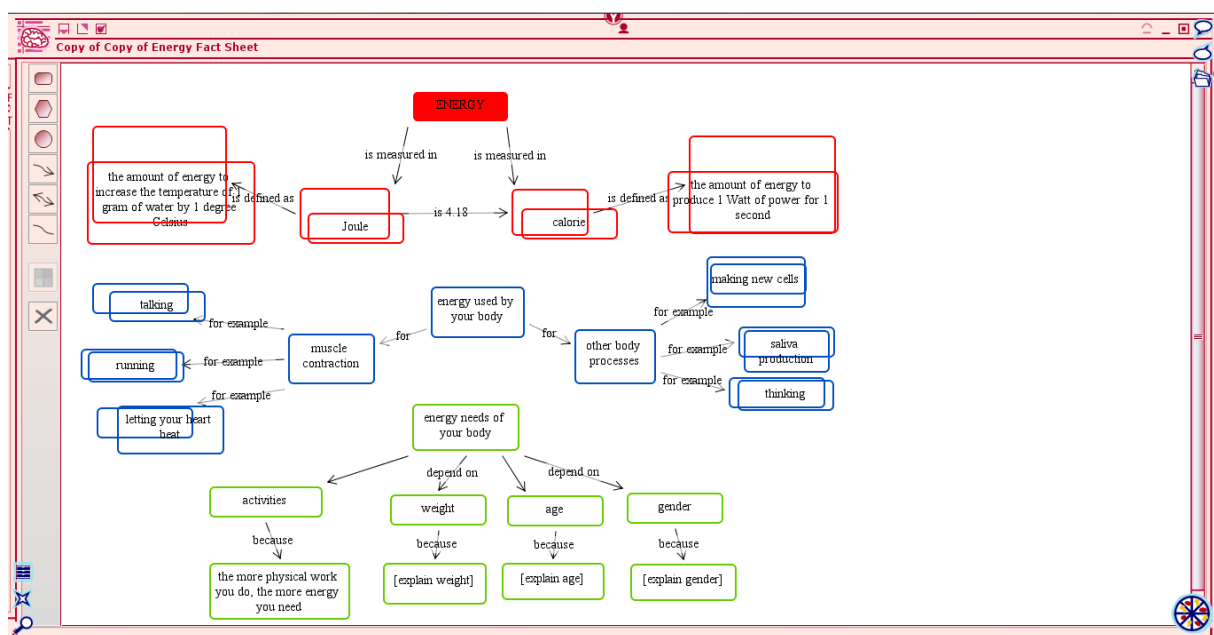


Figure 6: Energy Fact Sheet of the pizza mission

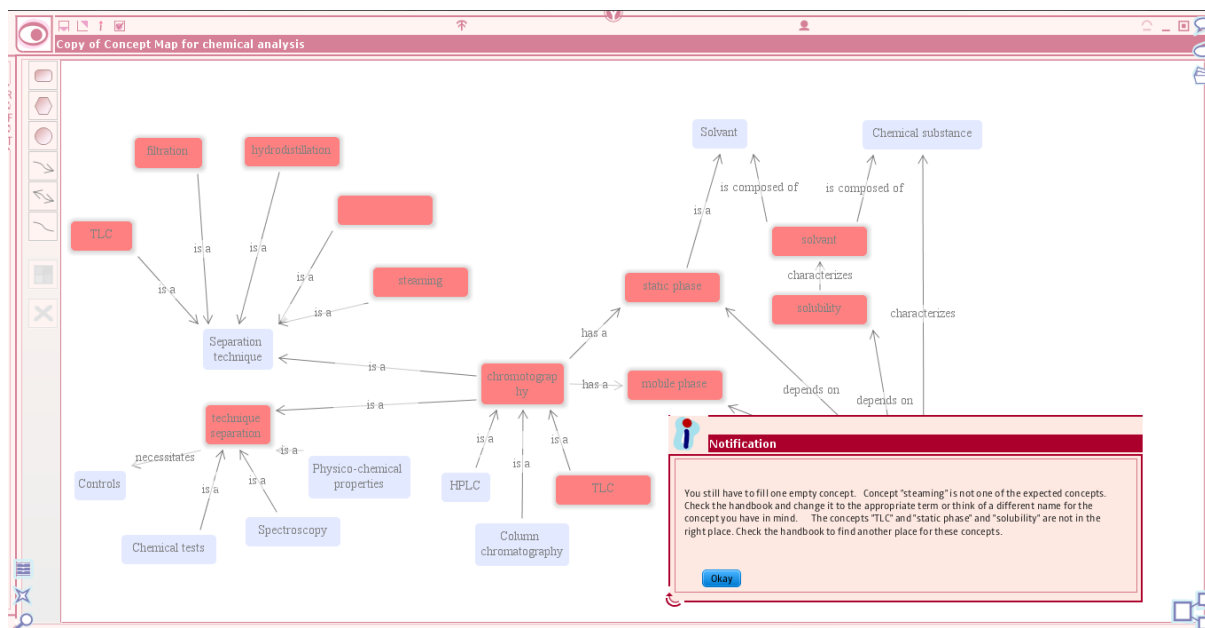
The feedback is simply the number of concepts the learner dropped on the correct location. To be able to do this the agent needs to know which concepts are droppable and which are not. This is specified in the reference model with the "drop" attribute:

```
<node term="calorie" drop="true"/>
<node term="thinking" drop="true"/>
<node term="running" drop="true"/>
```

The scenario is that evaluation starts when the learner has dropped all droppable concepts. Until that happens he gets the message: "There are still X concepts that need to be dragged to an empty spot". After dropping all concepts, the message becomes: "You have X concepts correct and Y are wrong". When a learner has completed the map he is congratulated with: "You have placed all concepts correctly. Congratulations!".

## Chemical concept map in forensic mission

The map in the forensic mission is about concepts related to chemical processes in forensic science. As with the energy fact sheet the complete structure is given and some concept labels are blank. The task of the learner is to fill in these blanks with the appropriate concept.



**Figure 7: Chemical concept map of the forensic mission**

The scenario is that a learner gets feedback about three aspects of the evaluation. Firstly, as with the energy fact sheet the learner is informed about the number of blanks that still need to be filled. Secondly, if the agent does not recognise a concept label (a "redundant" concept) the learner is informed about this. And, finally, the learner is informed about concepts that are not in the correct location in the map. This can result in lengthy feedback (the French version; please see Figure 7 for the English version):

Il vous manque 1 concept.

Le concept "fred flintstone" ne fait pas partie de la liste des concepts attendus. Afin de le remplacer par un autre concept, relisez le manuel de laboratoire ou bien cherchez un synonyme.

Les concepts "eluant" et "identification technique" ne sont pas bien placés. Relisez le manuel de laboratoire afin de leur trouver d'autres places.

The concept labels in the feedback messages are those used by the learners, for example "eluant" rather than the reference model node "mobile phase".

In this scenario the agent needs to know which concepts are initially blank and have to be filled in by the learner:

```
<node term="Mobile phase" fill_in="true"/>
```

Determining whether a concept is dislocated is trivial when the concept map is completely filled with the correct concepts. Then it suffices to check all edges from and to the concept are the same in the learner and reference map. If the concept map is partially filled, or when some labels are not understood this approach no longer works and it can happen that a concept which is in the correct location is marked as dislocated.

The agent implements the following definition of correctly located. First, the learner map is scanned for edges in which both the head and tail concept have the status "present". Second, for each concept it is checked whether these edges are all in the reference model. If this is the case, the concept is correctly located although some of the ingoing or outgoing concepts might not be filled in or correct. We have not been able to prove that this algorithm is correct.

### **Topic concept map in ECO mission**

In the ECO mission learners have to follow the inquiry cycle for four topics in the ecology domain: primary production, photosynthesis, trophic levels and pH. Learners first follow the inquiry cycle and then update their concept map with the concepts they encountered in that cycle. The agent therefore has to know which inquiry cycle (topic) they come from and select the appropriate rule from the rule set to perform the evaluation. For example when a learner saves a hypothesis related to the photosynthesis topic this is recorded and when the concept map is later saved it is evaluated with the photosynthesis rule.

The concept mapping task in the ECO mission is open-ended: learners start with an empty map. It is therefore possible that some learners cannot complete the map on their own. The CME agent helps by presenting these learners with a better concept map of a class mate. The feedback initially is: "That is a good start but you still miss *X* concepts." Next, either of "The concept map has improved. *X* concepts are still missing" or "The previous concept map had more correct concepts." or "Your concept map has not improved. If you want me to look for a good concept map from a class mate then click the save button again". In the latter case, the agent looks for a better concept map from a class mate and presents it to the learner. If no better concept map can be found then the message is "I did not find any class mate with a better concept map.". When the concept map for a topic is complete, the learner is congratulated. In Estonian this reads: "Palju õnne!".

#### *2.1.3.4 Summary*

The concept map evaluation agent has been included in three SCY missions. In each of these missions the concept mapping task is slightly different. In the pizza mission learners have to drag and drop predefined concepts to the correct location in a concept map. In the forensic mission learners have to fill in blank concepts about chemical processes. And in the ECO mission learners incrementally fill a concept map after they learn about the ecology domain in four inquiry cycles.

The CME agent can cater for a variety of different scenario's by separating language issues (mission can specify the labels of concepts and relations in a term set), the structure (the reference model defines how concepts are related to each other) and the type of evaluation to be performed (in rules).

## **2.2 Agent support for scientific experiments**

Allowing learners to do experiments "hands-on" is another central concept of SCY. Doing "real" experiments in the classroom or outside is one aspect. This context is not accessible directly to automatic scaffolding by agents. Another aspect; however is to experiment in a simulated context. Here the situation of the learner can be analysed to some degree and scaffolds can be generated automatically.

There is a whole group of agents that aims at detecting whether a learner uses the SCYSim simulator to prove a hypothesis in a systematic fashion or not. If these agents detect an unsystematic behaviour, e.g., that a learner changes many variables at the same time and therefore cannot judge the influence of a single variable on the output, a corresponding

message is displayed. The complete scenario including user story and technical description of the agents was already mentioned in DV.3, section 4.4 and there have not been any significant changes since then.

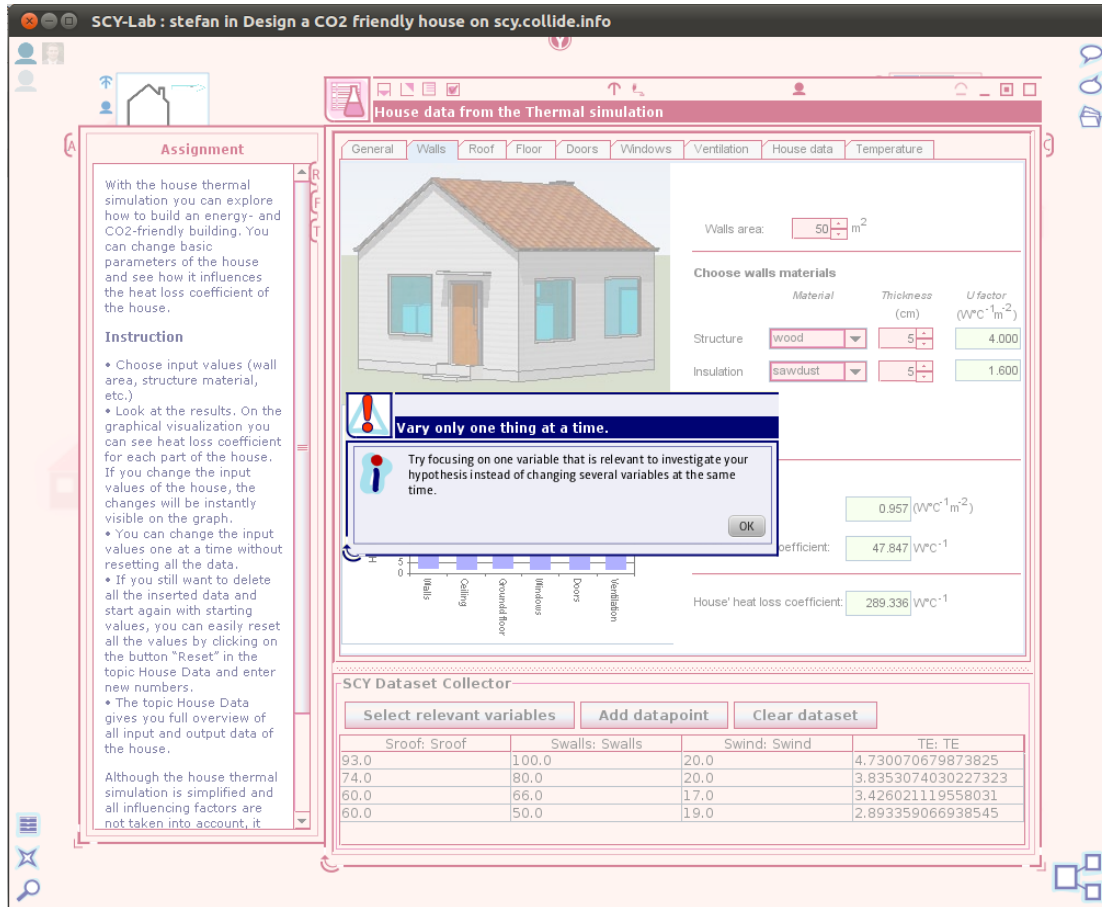


Figure 8: Dialog showing the VOTAT message

A possible scenario could be the following:

Once Eric has read the resources and finished the concept map successfully, he is reminded of the overall goal: to develop a CO2-friendly house. After the student has been reviewing reading material that explains the factors that influence heat loss (types of walls materials commonly used, U-Factor etc.), and after taking a look at the simulation factors, the student tries to use SCYSim to test the hypothesis: "If thickness of the insulation material is increased, then the result variables will decrease." First, he plays around with the variables that can be changed in SCYSim. Even though, he has a concrete hypothesis, which he wants to investigate, he is not sure which variables need to be changed. After a while, a dialog appears with the following message: "Instead of changing several variables at the same time, try focusing on one variable that is relevant to investigate your hypothesis" (c.f. Figure 8). The student reviews again his hypothesis and realises that several variables are relevant. The student runs the experiment by changing the material and by changing the thickness of the material. After a while a dialog appears again saying: "In the beginning, start by only changing one variable at the same time." The student focuses on thickness of material and starts changing it across several experiments. After a few experiments the next dialog says: "If choosing a third value for a variable, then choose an equal increment as between first and second values." The student takes the new information into account and now systematically changes thickness with equal increments. The student receives the

following message from the next dialog: "Based on the hypothesis provided before, you seem to be running appropriate experiments."

To realize this scenario, four agents are involved in the interpretation and processing of the user's actions:

- The *VOTAT Monitor* is the first agent. VOTAT stands for Vary One Thing At a Time and describes an important heuristic related to hypothesis testing (Klahr & Dunbar, 1988). The heuristic suggests that a variable that is not relevant for the specific hypothesis should be held constant (e.g., hold the same value as in the previous experiment). Alternatively change only one variable at a time. The task of the VOTAT agent is to detect, which and how many variables have been modified within a specific time frame.
- The *Canonical Monitor* detects whether the learner chooses simple, canonical manipulations or not. Specifically, the agent counts if the learner uses equal increment to change the variable. The monitor should provide information to another agent that offers advice on how to change variables, while testing a hypothesis.
- The *Tool Experience agent* detects the tool experience of learners working with SCYSim. It offers information about how familiar the user is with a specific tool. Basically, this is solved by looking into the logs to determine how much time the user spent using that tool.
- The *SCYSim Behaviour Classifier* takes up and interprets information from the Tool Experience agent, VOTAT Monitor, and Canonical Monitor, and decides on pre-defined rules whether to show scaffolding notifications or not.

The information flow with respect to the above mentioned scenario is sketched below (Figure 9).

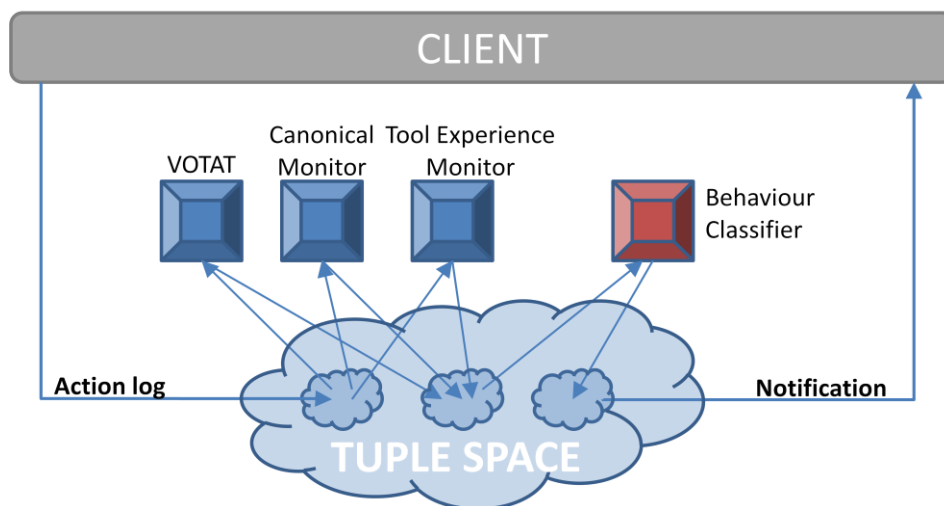


Figure 9: Agent use case detect systematic behaviour

### 2.3 Collaboration support

Another central SCY concept is that of collaborating learners. Collaboration has been implemented in a flexible way in SCY-Lab, enabling user-triggered collaboration on a particular ELO as well as collaborating groups of learners.

Collaboration between students and working together in groups is a central pedagogical idea of SCY. This is in detail described in deliverable DI.1. The formation of student groups and the interaction of students are neatly shown in particular in figure 4 in section 4.2.3 of DI.1 with the CO<sub>2</sub>-friendly house design mission as an example. More technical aspects are discussed in section 1.3 of deliverable DI.2.

In general there are three types of mechanisms of group formation foreseen in SCY-Lab: teachers may give directives to form collaboration groups of students, students themselves may invite other students to collaborate on a particular ELO, and pedagogical agents may initiate the formation of a working group. Student-triggered collaboration is supported by agents, but in a technical sense only. There are no pedagogical aspects that require a pedagogical agent, and therefore this way of collaborating is not described here in detail. In this section we describe agent support to form working groups without direct teacher interaction.

### 2.3.1 Narrative

A possible scenario could be the following:

*A class of 10 students starts working on the SCY ECO mission. Their teacher Omar helps them to log into SCY-Lab and asks the students to compose individually a concept map about ecosystems. After that the teacher guides students to the first learning topic – The role of light in ecosystems. The students individually write down their problems, research questions and hypotheses. Next, they enter to the second LAS – the Inference space. Jack moves there and a pop-up window appears. There it is said that he has to wait until enough peers are available to form working groups of students. Groups are needed to jointly apply limited material and equipment but it also has a positive effect while the peers support each other in a small learning group.*

*Jack completed the first assignments – the concept map, problem formulation, research question, and hypotheses – quickly and got a high quality score. After a while, Juliet, Helen, Maria, Jens, Carlos, and Joost enter the second LAS. Marcus, Jan, and Tanja are still working on their hypotheses. All the six newcomers get the same notification as Jack did. Teacher Omar set the limit of students needed for group formation to 80% of the total class size in preparing the pedagogical plan of the mission with the authoring environment. Now there are only 7 students out of 10 in the inferences' space. One more is needed before groups can be formed and the learning process can continue.*

*Finally, Jan also enters the inferences' space and the group formation agent starts the grouping process. The teacher also set the range of the members in one group. It should be set according to the class size and, in this case, the equipment available. This time Omar decided to form up to 3 groups of at least 3 students so that students with different learning outcomes will be mixed and the students with higher level outcomes support the others in learning.*

*The designers of the ECO mission selected the strategy for grouping students. There are several options: students with similar or different qualities of ELOs could be assigned to one group. There is also a possibility for randomised assignment where the ELO features are not taken into account.*

*Thus, the group formation criteria for this class of students are:*

1. *The student group size is from 3 to 4*
2. *The group formation is based on the quality of student concept maps*
3. *Students with different qualities will be assigned to one group*

*After Jan was entering the inferences' space each student gets a notification message. For Jack it says, now you are assigned to a group and you can continue working; your group members are Juliet and Joost. Jack had a high quality concept map. Juliet completed it quickly but it was poor. Two more groups were formed. The third group had only 2 members which was not enough to continue work. Therefore, Carlos and Helen have to wait for other students. When Tanja enters the LAS she can be assigned to the third group and they can continue*

*All three groups start working with SCYEd (Experimental Design Tool) for planning their experimental procedure. During this process also Marcus enters the inferences' space as the last student. The group formation agent decides that he should be the member of their group. Thus, one more buddy will be available in the list of buddies of the group members and a notification will be sent to all group members. They will finish with their experimental procedure and continue practical activities in the same group. When the inferences are completed, all students enter the third LAS – the problem solution space. There they should continue their work individually and their list of buddies of the former group is not available anymore.*

### **2.3.2 Data mining and pedagogical aspects**

While the (pedagogical) motivations of group formation in the cases of teacher directives and student-initiated collaboration may remain implicit, they have to be made explicit in the case of pedagogical agents initiating collaboration. Generally speaking an agent must rely on manifest properties of students that are accessible with SCY-Lab technology. Based on these properties, the choice can only cover three cases: grouping of students with *similar* properties, students with *dissimilar* properties, and student *selection at random*. All three strategies may be justified pedagogically in a particular learning context, and the choice has to be made by the teacher via the authoring tool.

Since SCY-Lab features ELO-centric learning, student properties have to be derived from student-produced ELOs. The example used here is the construction of concept maps and hypothesis texts. The group formation agent extracts features from these two types of ELOs and uses them to compare students.

From the hypothesis ELOs the agent extracts a sentence-keyword histogram, i.e. the number of sentences in the hypothesis without expected keywords, with one expected keyword, etc.

From the concept map ELOs the agent extracts the basic features number of concepts, number of relations, number of concept names, and the graph edit distance (please see deliverable DV.3, section 2.3.1) of the student concept map as compared to a reference solution that is provided as part of the mission design. Tests with concept maps from the school trials in France, Norway, Estonia, and the Netherlands showed that these general features are sufficient to distinguish clusters of concept maps. The resulting clusters showed examples of different degree of elaboration upon manual inspection of the clusters. We abstained from the derivation of a numerical measure for impartial evaluation of clusterings, because this seems nearly impossible. We also abstained from an inclusion of more specialised features of concept maps into the cluster algorithm, because both the structural and nominal variants of

the student maps showed very large variations that would be difficult to compare. Our solution is thus a heuristic.

The teacher may decide which type of ELOs to use in the group formation process. The features of one or both types of ELOs are then represented as numerical vectors and used in the clustering algorithm that is the basis of the group formation process. The technical details are explained in the appendix (Section 6).

The results of group formation must be communicated to the students. Technically this is solved by sending every student in a group a notification message (see Section 4.4) which is displayed in a message dialog in SCY-Lab and placing each member in a group in the buddy list and introduce them as friends if they were not collaborating before. The actual collaboration via the SCY-Lab tools must be initiated manually by the students. This situation is displayed in Figure 10. The SCY-Lab desktops of two users are displayed. The students entered a LAS in which working groups are required. They are notified that they are foreseen to join the same group.

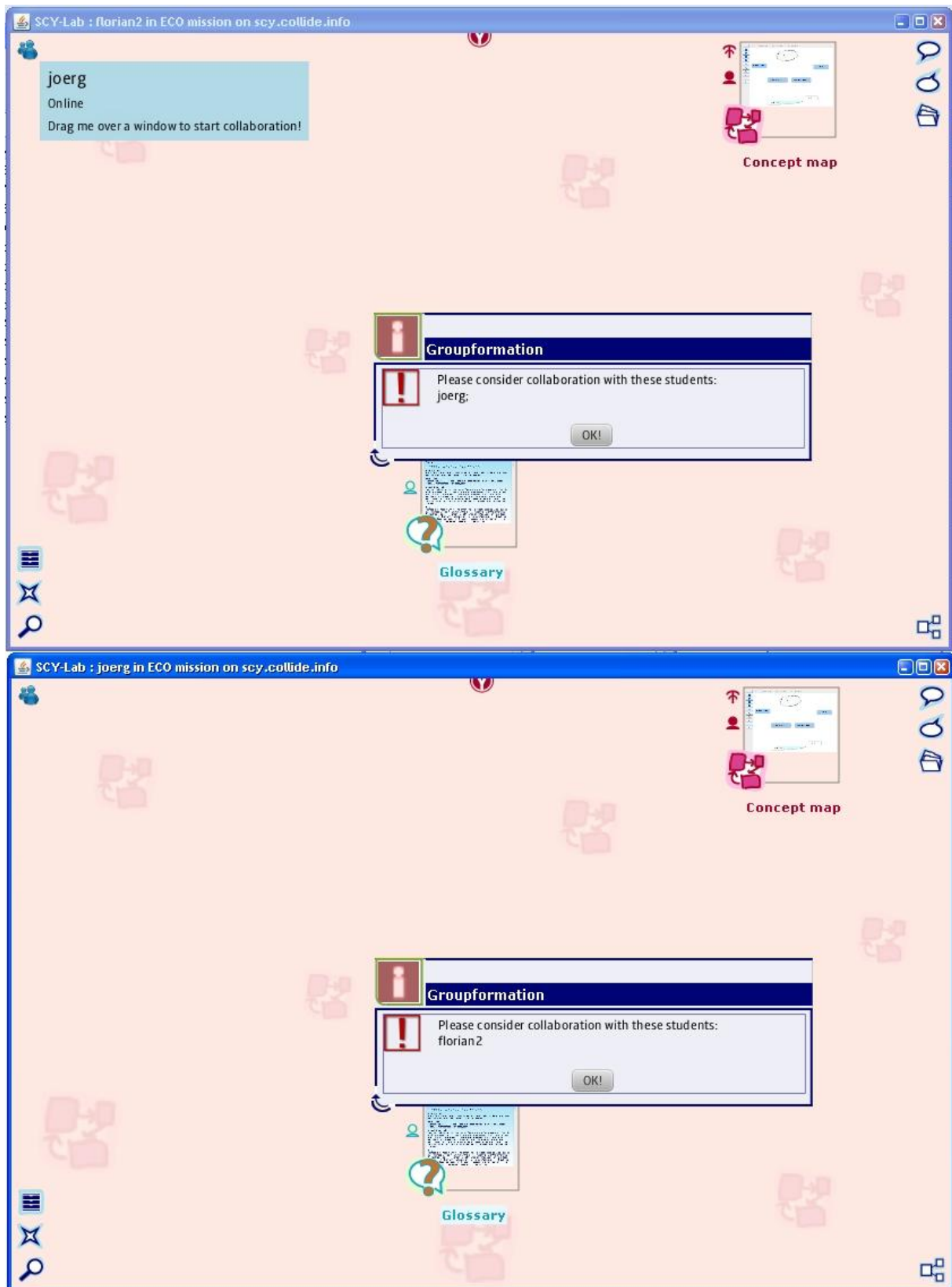


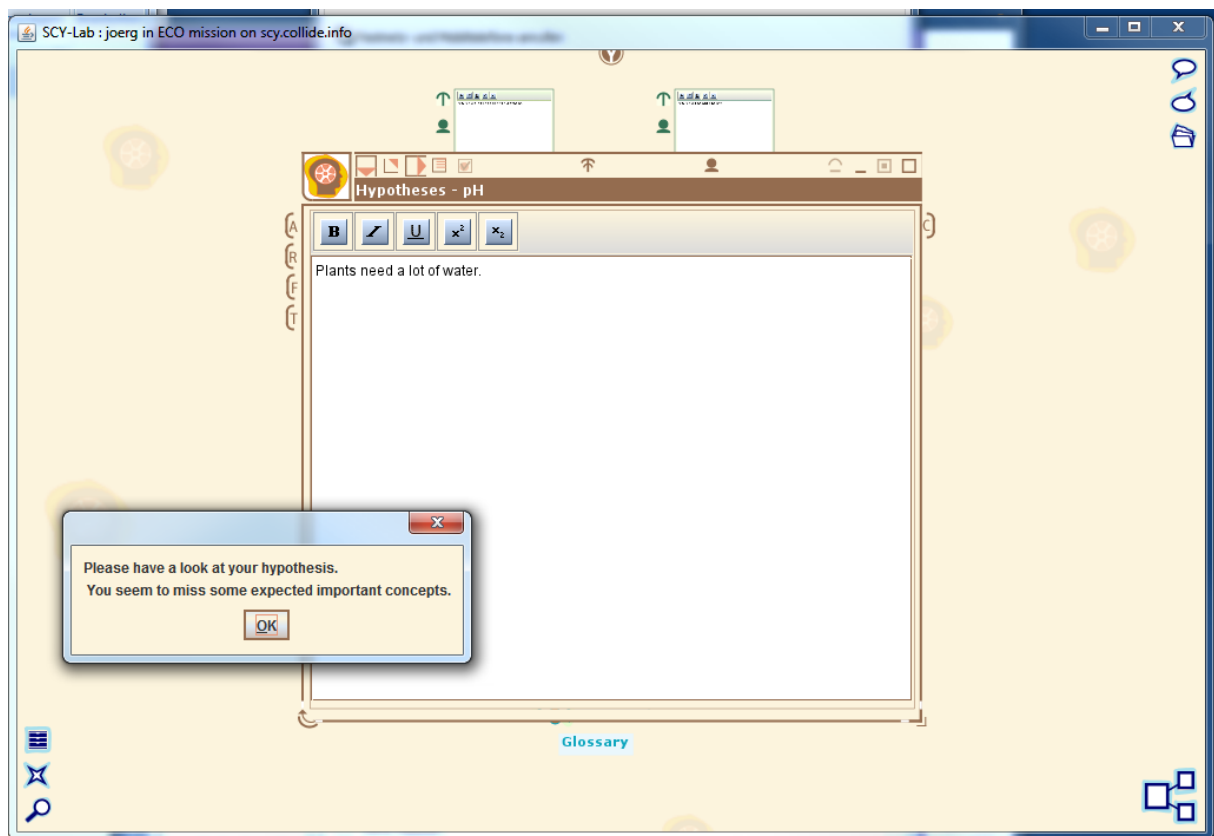
Figure 10: Two user desktops with group formation notification

## 2.4 Agent support for writing hypotheses

Writing texts is an activity mainly required to produce reports and summaries of final results of mission work, and, in a more specialized context, hypotheses that describe the expected outcome of scientific experiments. The latter case is the only one that is automatically evaluated qualitatively in SCY-Lab.

A possible scenario could be the following:

*A group of students continues working with SCYEd (Experimental Design Tool) for planning their experimental procedure in the Eco mission. After planning their experiment they now have to write down their hypothesis about the experiment outcome. The teacher set the scaffolding level to “high”. So on the first “save” action of their SCYEd tool they receive the message “Please have a look at your hypothesis. You seem to miss some expected important concepts.” They continue to discuss the experiment, and extend their hypothesis description, using more technical terms than before. Now there is no further scaffolding after they hit the “save” button.*



**Figure 11: Scaffolding a weak hypothesis text**

The hypothesis evaluation scenario was originally part of the scenario on hypothesis driven experimentation. This scenario is described in deliverable DV.3, section 4.4. The student there experiments with the house simulator of mission 1 and tries to come up with reasonable hypotheses about the effects that a variation of independent variables like wall thickness has on the dependent variables like room temperature. This is a rather closed setting for hypothesis evaluation, because the independent and dependent variables are known, and there is also a reference model that relates them. It is therefore possible to define a code-book for evaluation. When other SCY missions became operational, it soon became obvious that hypothesis evaluation was needed in more open contexts as well. It was therefore decided to separate hypothesis evaluation from the experimentation scenario. The agent was simplified

and the code-book approach was dropped. The syntactic analysis of student hypothesis in the originally planned form was feasible only for English (a proof of concept is given in Daxenberger & Kindermann (2011)), because automatic syntactic parsing of sentences was not available for the other languages used in the missions. Therefore the syntactic analysis was simplified too. The agent now works with a modified version of the Keyword-based analyser (see section 4.4.3.3 of DV.3). It analyses the distribution of mission-relevant keywords over the sentences of the hypothesis text and gives a feedback to the student that indicates whether the text should deal with more of the expected keywords, or it should relate the keywords better to one another. If no flaws are found in the text, no feedback is given.

## 2.5 Table of agents in missions

Table 1 shows which agent type is active in which mission. The systematic behaviour agent is specifically used in mission 1 (the “CO2 mission”), because this mission features a sophisticated experimental simulation environment. It also features a LAS with a free inquiry task that is well suited for the concept map supporting agents. Group formation currently is only required in mission 2 (the “Eco mission”), but in a very central place: groups are formed in four LASs, based on both concept maps and hypothesis texts. Mission 3 (the “Pizza mission” aims at younger students. It is therefore more restricted and features a specialized version of concept map evaluation called energy fact sheet evaluation. The agents that support concept map evaluation and hypothesis evaluation and the agents supporting smart search are more general and they are thus active in several or all missions.

**Table 1: agents active in missions**

Tool / agent	CO2	Eco	Pizza	DNA
SCYSim / systematic behaviour	X			
SCYMapper / keyword proposal	X			
SCYMapper / concept & relation proposal	X			
SCYMapper / concept map evaluation		X		X
SCYEd & SCYText / hypothesis evaluation	X	X		X
SCYEd & SCYText / group formation		X		
SCYMapper / EnergyFactSheet evaluation			X	
Search supporting agents	X	X	X	X

## 3 Agent framework

In general, the pedagogical agent framework did not undergo major changes, but it rather grew and slightly changed step by step in the course of the project. Almost all aspects of the concept that has already been described in DVI.1, section 2.3, and later in DIV.3, section 3, are still correct. However, due to the experiences made in the SCY project, the system naturally evolved and therefore this section focuses on these changes to previous deliverables.

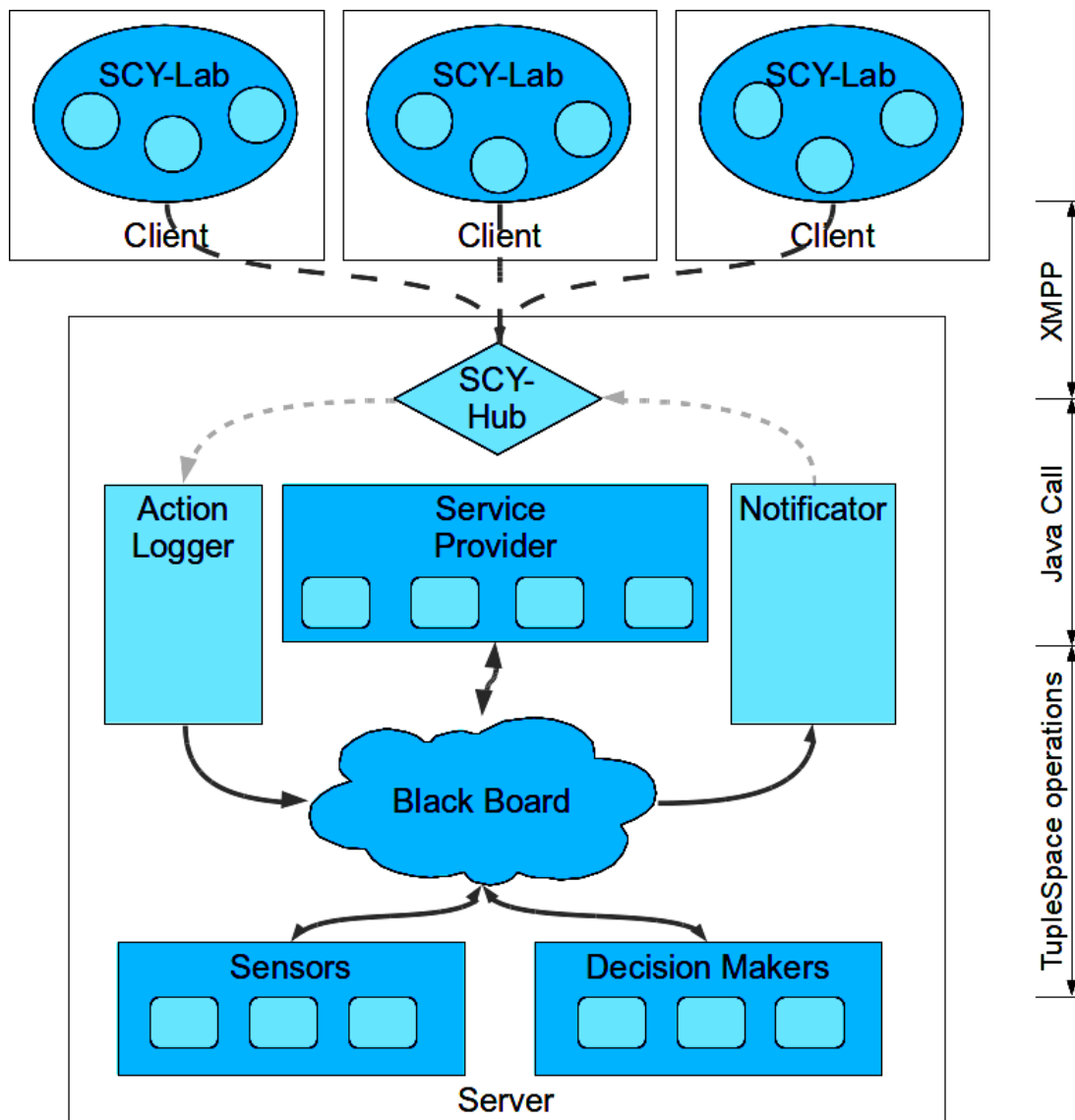


Figure 12: The final architecture of the pedagogical agent framework.

### 3.1 Agent architecture

As shown in Figure 12, the final architecture of the pedagogical agent framework is very similar to the one presented in DV.3. The core is represented by the blackboard that acts as a shared memory and message exchange service. Several agents are “floating” around this blackboard that can be divided into three classes: The sensors that react on incoming events from the system (mostly user actions), the service providers that react on other agent’s queries and often encapsulate a certain service (like the ontology or RoOLO), and finally the decision makers that use this sensory data and the service providers in order to decide whether or not to trigger an action towards the user.

The communication with the clients is routed via the SCY-Hub (c.f. DVI.6, section 2.2) and the incoming (respectively outgoing) component is called action logger (respectively the notificator). In Figure 12, the action logger and notificator are drawn in different colour than the other agents and this represents the different technical realizations. While all sensors, decision makers and service providers are technically either a Java instance of the SCY agent interface or a prolog agent, the action logger and notificator are actually subcomponents of the

SCY-Hub. However, we still consider these two components conceptually agents since they communicate with the blackboard.

The previously planned “hotline” has been removed from this diagram. Its purpose was to allow tools or SCY-Lab itself to communicate directly with the agents. However, we did not find a reasonable use case that could not be covered by a similar message flow using the action logger and notifiicator.

Besides many other changes, a particular new sensor agent instance was developed, called the session agent. This agent listens to the stream of action logs and reconstructs some more basic, mission and domain independent parameters of the state that are quite often used as input by other agents. This conceptually important transformation from an action-based stream of single events to a state-based representation of users that are currently logged in results in a set of tuples that are stored in the “session space” that contains information such as the names of users logged in, their language, the mission they chose, the tools and ELOs they have opened, etc. This information is used by several agents to deduce necessary information. For example the keyword agents need a language specific model to determine the important keywords from a text. With the information which user uses which language in the session space the agent can determine the right model to load and extract the keywords. Another example would be to identify the users that are present in a mission or a LAS. This information is used in two different ways. First the authoring tools (c.f.DVII.3) use the information to graphically display the location of the students in the mission. Secondly the information is used in the group formation agent to determine how many students are executing a mission and depending on the number of present students to decide whether enough students are present in a LAS so that group formation is feasible.

### 3.2 Blackboard design

The decision to use a blackboard approach for the agent framework of SCY was a good one. Especially the explorative nature of the research project SCY fitted well with the flexible approach of a blackboard that due to its loose coupling allowed for local changes and modifications without affecting the system on a global level. Another advantage of a multi-agent system is that it encourages the developers to reuse code by processing one agent’s output by several agents. This is especially used in the agent that extracts keywords to aid the student in constructing a concept map and the agent that assesses hypotheses. Both have to extract keywords on the basis of a text the student copied respectively entered into a tool. The difference is that the agent that extracts keywords for the concept maps uses several algorithms to extract keywords and merges the results of these algorithms to determine the keywords to suggest. On the other hand the hypothesis assessing agent uses only a subset of these algorithms, as experiments have shown that the most basic algorithms dilute the result. As a result we implemented the different algorithms as individual agents that get a text and the language to use and apply the specific algorithm to extract keywords. Because these agents are independent of any other agent and just need the text and the used language to work, they could easily be reused in the context of the hypothesis agent that asks only a subset of the present keyword extracting agents to determine the keywords in the hypothesis text.

Another aspect of the development in a TupleSpaces environment is the role of the blackboard. Because of its nature, the blackboard can be used as a message passing platform (for volatile and maybe directed data) on the one hand, and as a storing facility (for persistency) on the other hand. In many cases, the data inside the tuples were somewhere in between these two extremes. However, we came to the conclusion that it is advisable to keep in mind that storing data on the blackboard is only useful, if it is possible for another component to reuse this data. The more specific this data is and thus the more specific the

format is, the more it is probable that it will never be used by a second agent. The already mentioned language models used for the keyword extraction are such a case. The different algorithms use a specific model only usable in the context of this algorithm. These models need to be language and mission specific, i.e. for every language and mission we have a slightly different model. The models are built during the mission design and will never change and become rather big. This was a reason why we do not store them in the TupleSpaces (ADB), as described in DVI.1. First of all, the models would not be reused by other agents and due to the size of the model the loading was slow. We therefore chose to save the models as files so they can be as easily accessed as via the TupleSpaces. A binding folder structure allows us to have different models for each language and mission.

### 3.3 Agent-tool interaction, logging and notifications

Coming from earlier deliverables (e.g. DV.1, DV.3, DVI.I), the action logging facilities have been designed to communicate in XML format; but quite early in the project, it was decided that a single action log entry should be stored as a *tuple* in a specific SQLSpaces server. Reasons for that can be found in DVI.3. A tuple contains the same information as an according XML representation, but provides easy-to-use means for filtering, searching and parsing actions in various programming languages. Agents, who are listening for certain user activities are implemented against an SQLSpaces server interface. This approach has proven to be stable, scalable and flexible enough to transport a variety of action log information and is able to implement various agents. However, the previously mentioned XML representation of actions are still used for export and exchange purposes.

As an example, an action log would look like the following in XML and tuple format. It originates from the SCYSimulator tool when changing a variable value in the simulation (see Figure 13 and Figure 14).

```
<action id="5b4123c3-095c-4bd6-895e-89a503984d35"
  user="lars"
  type="value_changed"
  timemillis="1315812235271">
  <context>
    <tool>simulator</tool>
    <mission>n/a</mission>
    <session>n/a</session>
    <eloURI>scy://unsaved_elo_95a8d935-f1b6-4280-ba54-
b69db173a022</eloURI>
  </context>
  <attributes>
    <property name="newValue"><![CDATA[1.0]]></property>
    <property name="name"><![CDATA[i10]]></property>
    <property name="oldValue"><![CDATA[0.0]]></property>
  </attributes>
</action>
```

Figure 13: Action log excerpt in XML format

```

("action":String,
"5b4123c3-095c-4bd6-895e-89a503984d35":String,
"1315812235271":long,
"value_changed":String,
"lars":String,
" simulator ":String,
"n/a":String,
"n/a":String,
"scy://unsaved_elo_95a8d935-f1b6-4280-ba54-69db173a022":String
"newValue=1.0":String,
"oldValue=0.0":String,
"name=i10":String)

```

**Figure 14: Action log excerpt in tuple format**

Consequently, the notification mechanism, which typically starts with an agent sending out a notification, makes use of SQLSpaces as well. Here, a notification tuple is created and send via XMPP to the according SCY-Lab client, where it is dispatched to the processing component (see section 4.4 for details). The lessons learned from the action log have helped to implement the communication in the reverse direction.

## 4 Agent technology

### 4.1 Overview

This section describes the important bits and pieces of agents "under the hood" - like agent configuration support and shortly references components that have been described in detail in earlier deliverables.

### 4.2 Collaboration agents

The collaboration in SCY-Lab is technically implemented by using XMPP messages, to be more precisely by using a multi-user chat (MUC). All participating instances, regardless of them being two instances of the same tool running in two different SCY-Labs or two different running in the same SCY-Lab, are clients in a MUC and write and receive all actions as participants of this MUC. The Openfire server then passes on these actions to all clients, which use them to update their internal state accordingly.

The process of initiating this infrastructure can be subdivided into the steps of creating the MUC and telling all clients to join the MUC. Since there is no direct communication between clients in the SCY architecture, the latter step has to be done via the server. Although, the first step could also be done on the client-side, we decided to do it centrally on the server side and to turn it into an agent. That way, it is possible to use the notification facilities we already use for scaffolding in order to pass the MUC id to the collaboration participants, among others. This collaboration agent can then also be used by other agents like the group formation agent to initiate collaboration between learners.

Another feature of the collaboration agent is that it can handle an invitation procedure before the actual initiation of the collaboration. This procedure is started when a learner drops another user's icon to a tool. This causes SCY-Lab to log this collaboration request and this action triggers the collaboration agent, which notifies the SCY-Lab of the invited user. Then, a message asking whether the other user is willing to collaborate is displayed on the invited user's screen. The answer to this question is again logged and the agent either initiates the

collaboration in case of a positive answer or informs the first SCY-Lab about the negative response.

### 4.3 Teacher support

To support the teacher in monitoring the students and adapting the mission to the level of the students we created two agents and made parameters of the agents changeable from the outside. Preliminary versions of these agents were developed early in the project and are described in Khayat et al. (2011).

The developed agents are first the Session Agent that monitors the whereabouts of the students in a mission and other information such as which tools he uses at the moment and what language. This information can be read from the SCY Author Runtime view and will be displayed in a visualization of the mission. So the teacher sees in which LAS which students are working.

We also implemented support for the teacher to change the scaffolding level of SCY-Lab on the fly. This has effects on the behaviour of the tools as well as of the agents. A slider exists in the teacher tools that changes the scaffolding levels. The slider supports settings from off to medium to high scaffolding. Each setting lets the agents behave differently. For example the setting *off* turns some agents off. A setting of *medium* will let the agent have a low frequency of interfering respectively helping. So it is possible that only more serious scaffolds are reported back to the student and minor messages or modifications are ignored. A setting of *high* will report even minor scaffolds or will provide assistance earlier. For example the agent that supports a student by proposing keywords as concepts for a concept map can directly provide these on a setting of high interference or wait until it detects that the student has not found any concepts in a certain period of time. The same effect is adapted by the systematic behaviour agents. The hypothesis evaluation is either done on every “save” action (high scaffolding), or only when the student presses the “I am finished” button. This allows adapting the agent behaviour to the level of the student. Please see Table 2 for an overview of scaffolding level effects on the agents.

**Table 2: Effects of scaffolding levels on agents**

agent	Scaffolding levels		
	off	medium	high
Systematic behaviour	No notification	Increase reaction time	Reduce reaction time
Keyword proposal	No notification	Increase reaction time	Reduce reaction time
Concept & relation proposal	No notification		
Concept map evaluation	No notification		
Hypothesis evaluation	No notification	Triggered on finish	Triggered on save
Group formation	No notification		
EnergyFactSheet evaluation	No notification		
Search supporting agents	Always active		

#### 4.4 Agent tool communication

The agents occasionally need to communicate with the tools on the client side to inform students of the results e.g. they were assigned to a group or the keywords found for aiding the creation of concept maps. Respectively the tools need to convey information to the agents via the action logs. But agents can communicate only through the TupleSpaces, whereas SCY-Lab is running client side and has no connection to the TupleSpaces. Therefore we needed to implement two special purpose agents that differ in our definition from other agents. They communicate not only via the TupleSpaces but use the SCY-Hub (see Figure 12) to translate between the XMPP format and the TupleSpaces format.

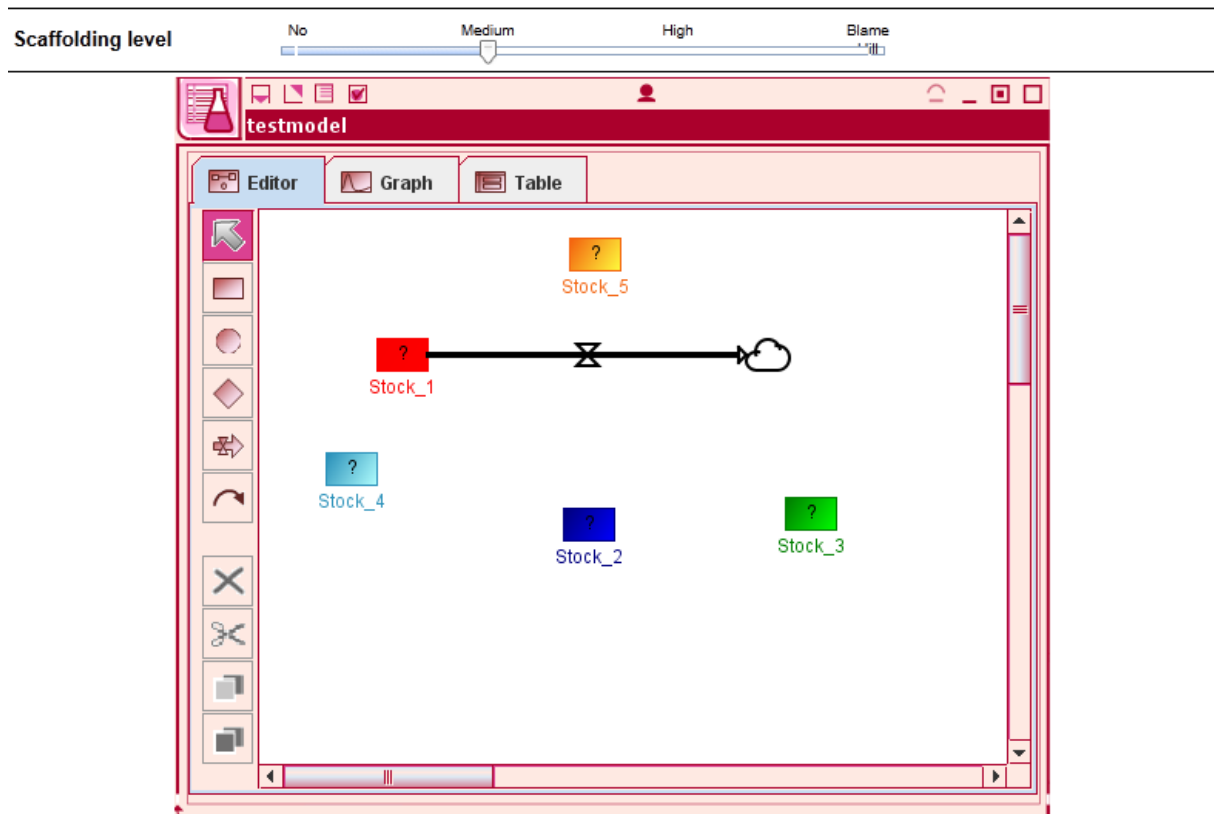
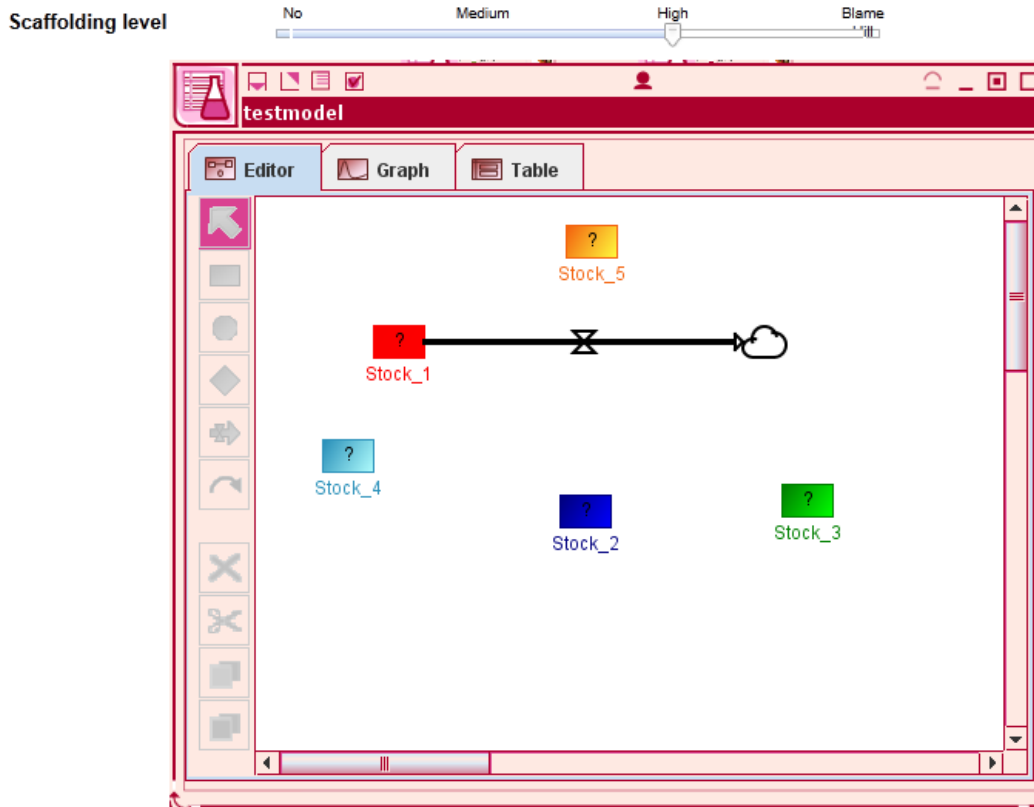


Figure 15: Medium scaffolding level visible effect on the SCYDynamics tool

The first special purpose agent is the *ActionLogger*. It is connected to the SCY-Hub on the one side and to the TupleSpaces on the other. Its purpose is to translate the incoming XMPP formatted action logs into tuples that can be processed by the agents. (See DII.1) On the other side resides the *Notificator*. It is as well connected to the SCY-Hub and the TupleSpaces but its purpose is the other way round. It collects all tuples that match the notification tuple format and translates them to XMPP messages that can be sent to SCY-Lab where another component routes them to the correct tool that is able to display this notification. For the tuple formats see DV.3 Section 3.2. Figure 15 and Figure 16 show examples of varying scaffolding levels which are communicated to the SCYMapper tool by the ScaffoldingNotificator agent.



**Figure 16: High scaffolding visible effect on the SCYDynamics tool**

The only thing that changed is that the notification tuple now must contain a field called EloURI that holds the URI of the affected ELO. This ELO URI is then used on the SCY-Lab side to route the notification to the tool that displays this ELO. Every tool is automatically registered as a receiver of notifications. Nonetheless it is still possible to manually register components that react on specific notifications. The routing will first try to find a tool that currently displays the ELO specified in EloURI. Whenever no tool is found that displays this ELO or the ELO URI is not important to this type of notification i.e. is set to “n/a”, the notification is routed to manually registered components. So it is possible to handle notifications that are meant to be displayed independent of any specific tool.

#### 4.5 Agent support for search

Agents not only support students working with SCY-Lab tools directly, but they also support another important internal component of SCY-Lab, the search engine. Searching is based on indexing ELOs. This in turn is based on ELO meta-data which are written into ELOs on save by feature extraction agents. These are the same agents that also support hypothesis evaluation and concept map construction. This has already been described in detail in deliverable DIV.3

As described in DIV.5, the SCY repository (RoOLO - Repository of Open Learning Objects) is the central component for storing, searching and retrieving ELOs and their metadata. From an agent's perspective, it is crucial to get access to this core functionality. SCY's agent framework (presented in Section 3) is designed to support multiple programming languages to implement agents, so that the simple (Java-centric) approach to just pass references to RoOLO objects to an agents is not possible. A lightweight and flexible way of providing access to RoOLO for all kinds of agents is the "RoOLO Accessor Agent (RAA)". The RAA has direct access to the repository and it is, like all agents in the SCY world, connected to the Tuplespace. It supports a tuple-based communication protocol to get invoked by all kinds of

other agents facilitating the Tuplespace to communicate with the other agents. A simple query for retrieving an ELO from RoOLO looks like this tuple:

```
<UniqueID>, "roolo-agent", "elo", <URI>
```

The RAA will return another tuple that starts with the unique ID and a String "roolo-response" and a third field containing the retrieved ELO as an XML string. As ELOs can be quite large, we introduced a function to retrieve only the metadata of an ELO. This function can be invoked by using the same query tuple as before, but instead of "elo" the querying agent puts metadata in the third field of this tuple. As already mentioned in DIV.5, we implemented an agent that could help users in a search activity. This agent recognises that the user is searching for ELOs of a special kind (like "give me all ELOs created by the user "Stefan" and with the keyword "CO2") and retrieves the same list of ELOs that is presented to the user. Using heuristics, this agent is able to reformulate the query so that it will retrieve a better set of results (c.f. section 2.6). As this agent must be able to perform search queries on an agent level, the RAA is equipped with the functionality to search for ELOs based on metadata entries (e.g. author or keywords) and return the search results to the querying agent. An invocation of this functionality looks like that: <UniqueID>, "roolo-agent", "search", "Stefan CO2". Since all metadata in the Lucene-based index is aggregated in one "contents" field, it is not needed to specify the keys of the specific metadata entries to search for.

The response to that query is again a tuple that starts with the unique ID and the string "roolo-response". The following field contains an XML element with all search results encoded as elements below the root element "SearchResults". If the agent is interested in just the number of search results, it can use the string "hit-count" instead of "search". The result will be an integer value in the third field instead of the "real" results.

The presented methods to access RoOLO using the RAA cover all relevant usages of the SCY repository as far as read-only access is enough. During the implementation of several agents in the SCY project, it turned out that some agents additionally needed write access to RoOLO. After some discussions we decided to allow agents to write in the metadata section of an ELO, as this will not necessarily create a new version of an ELO (for details see DIV.5). Unfortunately, there is no generic way of allowing agents to add arbitrary metadata entries or values as the key-value mechanism of ELOs is typed with non-primitive data types and, as mentioned above, the agent framework consists of agents that are developed in programming language other than Java, (e.g. Prolog), which are not able to instantiate the (Java-based) container for the values. Therefore, we decided to add specific methods to the RAA to allow non-Java agents to write values inside the metadata. An example for this access is the Prolog-based *Concept Map Evaluation Agent* (see Section 2.1.3). This agent evaluates a concept map that was created by a student by comparing the student's solution to an expert map that has been created beforehand. You can find more detailed information about this agent in Section 2.1.3. After this evaluation is finished, the agent passes the results of the evaluation together with additional information about the evaluation method and other information to the RAA by sending a tuple with the signature:

```
<UniqueID :String>, "roolo-agent", "cmmetadata",  
<URI :String>, <Method :String>, <TermSet :String>,  
<ReferenceModel :String>, <RuleSet :String>
```

The URI is referring to the ELO that should be enriched by this evaluation. The response to this "order" is the metadata of the ELO with the added evaluation section in the metadata.

This kind of specific write access to RoOLO can be easily extended to other use cases by adapting the RAA.

It is foreseen, that Java-based agents can add "directly" metadata keys and values by passing a serialised string to the RAA. Yet, there was no need for this kind of access, so this functionality is not implemented so far.

#### 4.6 Prolog agents

In the SCY project the TupleSpace serves as a communication channel between tools, agents and other services in SCY-Lab. Most of SCY-Lab is written in Java, some agents are written in Prolog (e.g., the concept map evaluation agent and the VOTAT agent). Developing a Prolog agent which connects to the TupleSpace is straightforward with the application programmer's interface for Prolog. Below is a fragment of code that creates a tuple asking \*roolo-agent\* to insert evaluation mark-up in a concept map ELO.

```
evaluate_elo_markup(ELO, Value, Method, TermSet, RefMod,
RuleSet) :-
    tspl:uid(Id),
    tspl_actual_field(string, Id, F0),
    tspl_actual_field(string, 'roolo-agent', F1),
    tspl_actual_field(string, 'cmmetadata', F2),
    tspl_actual_field(string, ELO, F3),
    term_to_atom(Value, AtomValue),
    tspl_actual_field(string, AtomValue, F4),
    tspl_actual_field(string, Method, F5),
    tspl_actual_field(string, TermSet, F6),
    tspl_actual_field(string, RefMod, F7),
    tspl_actual_field(string, RuleSet, F8),
    tspl_tuple([F0,F1,F2,F3,F4,F5,F6,F7,F8], Tuple),
    cme_tuple_space(command, TS, _),
    tspl_write(TS, Tuple).
```

All Prolog predicates starting with `tspl_` are part of the Prolog API for the TupleSpace. For example, `tspl_write/2` writes the tuple created on the previous lines into the TupleSpace and this tuple is then, hopefully, picked up by the Roolo agent which is written in Java.

Our experience is that developing agents using the Prolog TupleSpace API is very convenient.

#### 4.7 Table of all agents

**Table 3** shows an overview of all types of agents that have been defined and implemented. Please note that each agent entry in the table may technically be implemented in a number of agent components or processes that work together.

**Table 3: SCY agents**

<b>Agent category</b>	<b>Description</b>	<b>Interacts with</b>
Agent supervisor agent	Registers activated agents	Agents
Authoring agent	Support of teacher fine tuning of scaffolding levels	Teacher runtime view
Collaboration agent	Initiate and support student collaboration on ELOs	SCY-Lab tools that create ELOs
Concept map evaluation agent	Checks a concept map against a reference map	SCYMapper
Concept and relation proposer agent	Proposes concept map elements based on a text ELO and mission ontology	SCYMapper & SCYText
Group formation agent	Support of formation of student working groups	SCY-Lab user buddy management, mission map
Hypothesis evaluation agent	Evaluate student hypothesis texts	SCYEd
ELO metadata extraction agents	Extract metadata from various types of ELOs, support of smart indexing and search	RoOLO
ELO open monitor agent	Monitor tool activity	SCY-Lab tools
Portfolio monitor agent	Manage portfolio-related user communication	SCY portfolio
ELO save monitor agent	Monitors whether ELOs are saved to RoOLO	Tools & RoOLO
Keyword proposer agent	Proposes keywords from a text as names in a concept map	SCYLighter & SCYMapper
Ontology agent	Query the mission ontology	Mission ontology
RoOLO accessor agent	Retrieves ELOs from RoOLO	RoOLO
Search enriching agent	Suggests alternative search terms, if user's search provides too few or too many results.	RoOLO accessor agent
Sensor agent	Evaluate user behaviour	SCYSim
Session monitor agent	Tracks location of users in mission map	Mission map & action logs
Votat agent	Monitors changes of parameter values in SCYSim	SCYSim
Workflow controller agents	Monitors learners with respect to time used in LAS	Mission map & action logs

## 5 Lessons learned and conclusion

After more than three years of development it is time to recapitulate what can be learned when comparing the concepts and ideas that guided the project from the start with the achievements. We do this in a number of small sections that expand on various aspects.

**Pedagogical aspects:** This deliverable is due before the final SCY test runs that will be done in October and November. Most of the agents have not yet been used in test runs extensively and therefore it is difficult to say anything about the impact on learners and learning. What is encouraging is that the agent that evaluates system dynamics models has been used extensively in a SCY-related Ph.D. project. In experiments in which this agent gave tacit feedback about the quality of the models, learners who obtained feedback produced much better models than learners who did not obtain feedback from the agent. The pedagogical context is described in several papers (Mulder et al., 2009; Mulder et al., 2011; Mulder et al., submitted).

**Data mining aspects:** A central basic technology used in the pedagogical agents is data mining. This has been subject of the development since the start of the project (see DV.1). Data mining requires a large amount of relevant data that has strong relations with the field of analysis. In the case of SCY the data has both to be mission-specific as well as student specific. We need explanatory texts that describe the scientific mission background for example to derive semantic models that allow analysing student texts or concept maps. We also need examples of student-produced ELOs to be able to estimate the complexity of ELOs that will be produced when running the SCY missions. This of course is a situation that is near to a vicious circle, because it results in the requirement of data (ELOs) that can only be produced using the tools we want to build based on the data mining results. Therefore we had to work with approximations and to use student data from outside of the SCY project as a start. Later we were able to refine our development using the first results from the SCY test runs. It turned out that the first design approximations of data mining tools performed equally well on the new data obtained from SCY-Lab test runs.

Another aspect is the complexity of data mining technology to be used. In several cases it turned out that the originally planned technology was too complex to be adapted to SCY-Lab requirements, given the limited amount of data available early in the project. Here the substitution of the original technology through simpler components turned out to be advantageous.

**Mission design and agents:** Missions are, in the context of the SCY project, designed according to the scenarios developed in the project. Each mission can be applied according to a number of pedagogical scenarios which could differ from each other by the number of agents supporting the learning process and their level of guidance given to the learners. In order to integrate the needs of educational designers and agent developers there have been applied two main methods of communication. Firstly, mission developers were represented in the group of agent developers and in all their meetings. Particularly, the developers of the ECO mission participated in discussions and mainly in developing the group formation agents. Secondly, some separate meetings and online discussions between mission and agent developers were organized. As a result of the communication, a table of tools and agents used in particular missions was developed. It helped to ensure that all agents are applied in a reasonable way and all missions are supported by agents where it is possible and pedagogically relevant. This also led the mission designers to a detailed analysis of how to fine tune the agents according to the pedagogical needs in the context of every mission.

**Agent architecture aspects:** The agent architecture based on a blackboard and message passing concept was introduced right at the beginning and remained stable for the whole duration of the software design and development phase of SCY. It is one of the central features of the system design that supports flexibility and interaction between components. A side-effect is the re-use of agents in varying contexts that require a specific functionality that a particular agent provides. An important restriction that arose during development is the blackboard usage. The blackboard should only store data that is re-used by another agent or component.

It is furthermore notable that the agent architecture is not only used to interact with students through the SCY-Lab tools, but also to provide internal services like the creation of ELO meta-data for the search engine.

**Agent-tool interaction aspects:** Concerning the agent-tool interaction, the "lessons learned" include the advantages and challenges of loosely-coupled architectures and software communication. The communication from tools to agents is handled by the action logging framework, which collects the information about an user action on the client-side and sends it asynchronously via XMPP to the SCY server, where it is persistently stored in an SQLSpace. The client does not wait for successful transmission and does not care about the results or consequences of his messages. The same is true for notifications, which are sent from the agents to the clients, respectively to the tools. This mechanism makes implementation of tools and agents easy, robust, scalable, and flexible with respect to allowing for several programming languages.

Loosely-coupled architectures however present challenges to testing and debugging. Especially in the case of smart agents, that may need some seconds to collect data (from the action log), analyze and report results (via the notification mechanism), it is hard to tell when or if an agent will show visible results. Also, minor differences on protocol level can easily result in miscommunication between agents and tools, which cannot be discovered at compile time.

**Conclusion:** It can be stated that agents play an active role in all the central parts of SCY-Lab: they support scaffolding of students working with tools on text production, concept building, experimentation, and also the collaboration between students. Furthermore agents also support other functionalities of SCY-Lab, like the intelligent search, on a level invisible to the user. They have proven to be a valuable and stabilizing support to the SCY approach of ELO-centred experimentation and learning. The modular agent concept also supports easy adaptation and re-use of agents in the various missions that have been developed in the SCY project.

## References

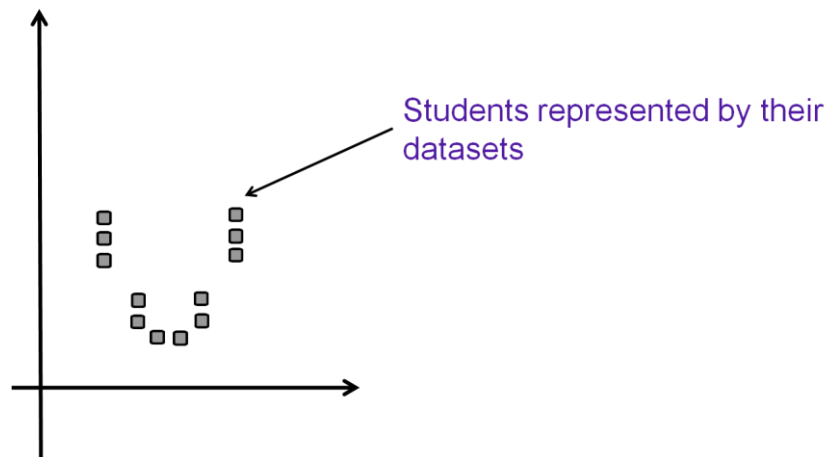
- Anjewierden, A., Chen, W., Wichmann, A. & van Borkulo, S. P. (2009). Process and product analysis to initiate peer-to-peer collaboration on system dynamics modelling. In J. Bourdeau, R. Mizoguchi, & S. Isotani. (Eds.), *Workshop on Intelligent and Innovative Support for Collaborative Learning Activities*, pp. 1–7, Rhodes, Greece. CSCL 2009.
- Daxenberger, J. & Kindermann, J. (2011): Using syntactic snippets to improve speech act recognition. Submitted to 12th International Conference on Parsing Technologies, October 5-7, 2011, Dublin
- Khayat, N., Mock, M. & Kindermann, J. (2011): Towards Automatic Behavior Analysis of Learners in a Technology-Enhanced Learning Environment. In: Proceedings of the 3rd

- International Conference on Computer Supported Education, Volume 1, Noordwijkerhout, The Netherlands, May 6 - 8, 2011, pp. 197 – 192
- Klahr, D., & Dunbar, K. (1988). Dual space search during scientific reasoning. *Cognitive Science: A Multidisciplinary Journal*, 12, 1-48.
- Mulder, Y.D., Lazonder, A.W. & de Jong, T. (2009): Finding out how they find it out: An empirical analysis of inquiry learners' need for support. In: *International Journal of Science Education*, Vol. 32, 2009, pp. 2033 - 2053
- Mulder, Y.D., Lazonder, A.W. & de Jong, T. (2011): Validating and extending the effects of model progression in simulation-based inquiry learning. Submitted to Journal of Computer Assisted Learning
- Mulder, Y.D., Lazonder, A.W. & de Jong, T. (2011): Comparing two types of model progression in an inquiry learning environment with modelling facilities. In: Learning and Instruction, 2011, in press
- Weinbrenner, S., Engler, J. & Hoppe, U. (2011). Ontology-supported Scaffolding of Concept Maps. Paper presented at the AIED conference. Auckland, New Zealand
- Weinbrenner, S., Engler, J., Hoppe, H.U. (2011). Ontology-supported scaffolding of concept maps. The 15th International Conference on Artificial Intelligence in Education, 2011.
- Weinbrenner, S., Engler, J., Wichmann, A., Hoppe, H.U., (2011). Towards an ontology-supported evaluation of concept maps. Submitted to The 15th International Conference on Artificial Intelligence in Education, 2011
- Witten, I.H. & Frank, E. (2005) Data Mining: Practical Machine Learning Tools and Techniques: 2nd Edition. San Francisco: Morgan Kaufmann.

## 6 Appendix: Technical aspects of group formation

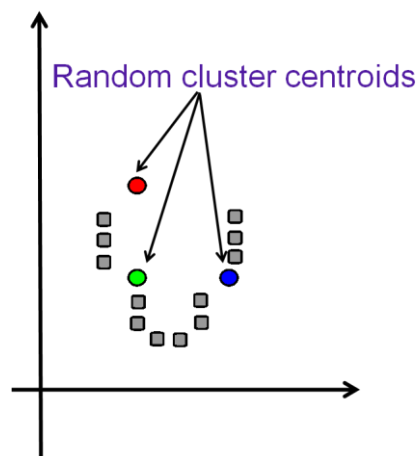
In a first step, the students are assigned to *clusters* by the k-means algorithm (Witten & Frank (2005)). The algorithm works as follows:

Students are represented by  $n$  numerical features that are extracted from their ELOs. For a concept map ELO, these are the number of nodes of the map, the number of relations, number of concept names, and the graph edit distance – resulting in a four dimensional vector for each student. Consequently each student can be represented by a location in an  $n$ -dimensional (4 in this case) coordinate system.



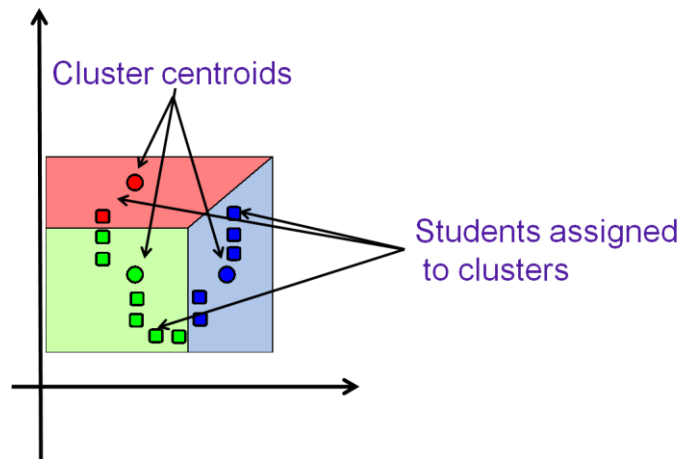
**Figure 17: Students are represented by numerical features in a coordinate system**

This is sketched in Figure 17. Please note that the figure only displays a two-dimensional coordinate system.



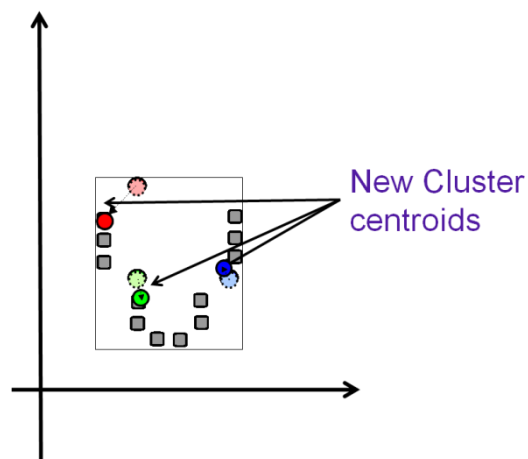
**Figure 18: Cluster centroids are added in random positions**

In a next step, the number of clusters has to be determined deliberately, and an according number of *cluster centroids* (again  $n$ -dimensional vectors). In SCY-Lab the number of clusters is the same as the number of student groups to be formed. In Figure 18 there are three clusters to be formed.



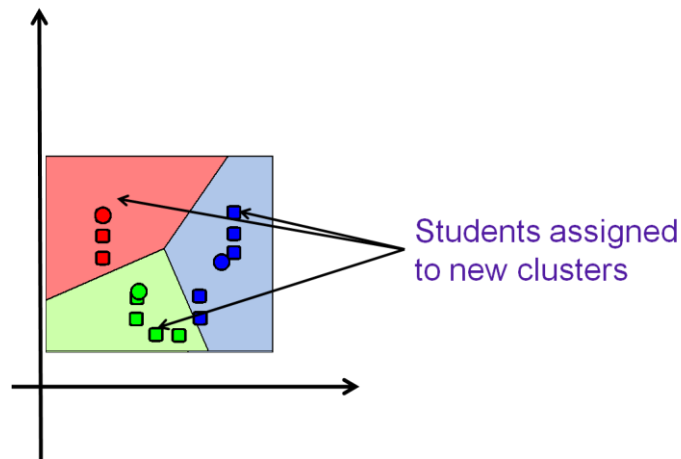
**Figure 19: Students are assigned to nearest centroids**

Now the student locations are assigned to those centroids which are nearest to the locations with respect to the common Euclidean distance. Figure 19 shows the result. Clearly the distribution of students in the clusters is rather unequal, with the red cluster containing only one student vector. Therefore the location of the centroids is now optimized as displayed in Figure 20. The centroid locations are moved by a small amount in the direction of the centre of the cluster as defined by the student vectors of this cluster.



**Figure 20: New centroids are computed**

In the next step the student vectors are re-assigned to the new clusters as shown in Figure 21. The adjustment of the cluster centroids and the re-assignment are repeated until the centroids do not change, or, alternatively for a maximum number of iterations.



**Figure 21: Students are re-assigned**

Then Groups of students are formed by selecting students from the clusters according to one of the three selection strategies: if groups of similar students are formed, the students of one group are selected from the same or a nearby cluster; if groups of dissimilar students are formed, students are selected from different clusters. For the random strategy we do not need to perform the clustering step at all.

Technically we had to solve various problems and challenges. One major difficulty was to foresee and predict how students move between LASs. In general group formation only makes sense if enough students are present in the LAS where the groups should be formed. So we decided to form groups only when a certain fraction of all students are present. This fraction can be set via a parameter for each LAS individually. A reasonable default value would be  $2/3$  of all students in a mission.

Nevertheless there are still students that will enter the LAS after groups have already been formed. They then need to be assigned to the existing groups. The assignment of late students to already existing groups must be done in accordance with the selected group formation strategy. So if similar students would be grouped the new student must be placed in a group with matching students as well. If the selected strategy was to have dissimilar students in the groups the new student can actually be placed in any group as the groups should already contain a variety of students of different degrees of similarity. The same criteria apply to the random group formation strategy. In these cases we choose to place the new student in the smallest group for ease of implementation.

Once the groups are formed they will stay the same for all the LASs. So if students enter a LAS where they already were assigned to a group he or she either has to wait until all other former members of his or her group are present or he or she is the last joining member of the former group and the group is present as a whole.

Another challenge was: What happens with the group if a student leaves the LAS. We decided to remove the student from the group in the buddy list so only the members of the group present in a LAS are considered to be collaborating in a group. As soon as the student enters the LAS again he or she will be placed in the same group as described earlier.