<table>
<thead>
<tr>
<th>Deliverable number:</th>
<th>D5.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deliverable title:</td>
<td>Integrated characters with dialogue and tutorial strategies</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Delivery date:</th>
<th>2011/07/31 (Extension date: 2011/09/30)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submission date:</td>
<td>2011/10/14</td>
</tr>
<tr>
<td>Leading beneficiary:</td>
<td>Augsburg University (UAU)</td>
</tr>
<tr>
<td>Status:</td>
<td>Version 05 (final version)</td>
</tr>
<tr>
<td>Dissemination level:</td>
<td>PU (public)</td>
</tr>
<tr>
<td>Authors:</td>
<td>Michael Wißner, René Bühling, Floris Linnebank, Wouter Beek, Elisabeth André</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project number:</th>
<th>231526</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project acronym:</td>
<td>DynaLearn</td>
</tr>
<tr>
<td>Project title:</td>
<td>DynaLearn - Engaging and informed tools for learning conceptual system knowledge</td>
</tr>
<tr>
<td>Starting date:</td>
<td>February 1st, 2009</td>
</tr>
<tr>
<td>Duration:</td>
<td>36 Months</td>
</tr>
<tr>
<td>Call identifier:</td>
<td>FP7-ICT-2007-3</td>
</tr>
<tr>
<td>Funding scheme:</td>
<td>Collaborative project (STREP)</td>
</tr>
</tbody>
</table>
Abstract

In this deliverable, we introduce three new virtual character support types (Critic, Quiz and Diagnosis) which are linked to different kinds of conceptual knowledge (such as semantic feedback or multiple choice questions) generated within the DynaLearn software. We show how this knowledge is extracted and transformed into character-specific dialogs that are presented by the virtual characters but also encourage the learner to engage and participate. For this process, we rely on the architecture and dialog management system developed in earlier deliverables. Finally, we investigate how different characters and support types can be combined with each other to form integrated dialogs to even further support and engage learners.

Internal reviewers

Bert Bredeweg, Human Computer Studies Laboratory, University of Amsterdam (UvA), The Netherlands

Andreas Zitek, Institute of Hydrobiology and Aquatic Ecosystem Management, University of Natural Resources and Applied Life Sciences (BOKU), Austria

Acknowledgements

The Bayesian models depicted in this paper were created using the GeNIe modeling environment and the core of our User Model implementation is based on jSMILE, a Java wrapper for the SMILE reasoning engine for graphical probabilistic models. Both GeNie and SMILE are developed and contributed to the community by the Decision Systems Laboratory, University of Pittsburgh (http://dsl.sis.pitt.edu).

For the parsing of OWL-based data, we use the OWL API, primarily maintained by the University of Manchester (http://owlapi.sourceforge.net).

The authors would also like to thank partners from UVA and BOKU for their internal review of this deliverable.
Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Modification(s)</th>
<th>Date</th>
<th>Author(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>First Version</td>
<td>2011-08-15</td>
<td>Wißner, Bühling</td>
</tr>
<tr>
<td>02</td>
<td>Version for internal review</td>
<td>2011-09-28</td>
<td>Wißner</td>
</tr>
<tr>
<td>03</td>
<td>Integrated suggestions from BOKU review</td>
<td>2011-09-29</td>
<td>Wißner</td>
</tr>
<tr>
<td>04</td>
<td>Integrated suggestions from UvA review</td>
<td>2011-10-12</td>
<td>Wißner</td>
</tr>
<tr>
<td>05</td>
<td>Final version</td>
<td>2011-10-14</td>
<td>Wißner, André</td>
</tr>
</tbody>
</table>
Contents

Abstract ......................................................................................................................... 2
Internal reviewers .......................................................................................................... 2
Acknowledgements ....................................................................................................... 2
Document History ......................................................................................................... 3
Contents ......................................................................................................................... 4
1. Introduction ............................................................................................................... 6
2. New Virtual Character Models .................................................................................. 8
   2.1. Mechanic ............................................................................................................. 8
   2.2. Critic .................................................................................................................... 9
   2.3. Arbitrary character / role combinations .............................................................. 9
3. Quiz .......................................................................................................................... 11
   3.1. Introduction and Purpose .................................................................................. 11
   3.2. Interaction Flow and Examples ......................................................................... 12
   3.3. Knowledge Extraction and Representation ....................................................... 15
4. Critic ......................................................................................................................... 19
   4.1. Introduction and Purpose .................................................................................. 19
   4.2. Interaction Flow and Examples ......................................................................... 19
   4.3. Knowledge Extraction and Representation ....................................................... 21
5. Diagnosis .................................................................................................................. 23
   5.1. Introduction and Purpose .................................................................................. 23
   5.2. Interaction Flow and Examples ......................................................................... 23
   5.3. Knowledge Extraction and Representation ....................................................... 25
6. Integrated Dialogs .................................................................................................... 27
   6.1. Introducing the characters and support types .................................................... 27
   6.2. Getting to know a model ................................................................................... 28
   6.3. Model building support .................................................................................... 28
7. Conclusion & Outlook ............................................................................................... 30
1. Introduction

The DynaLearn project aims to conceive and develop an interactive learning environment which combines current technologies and research from different areas in a way that provides learners and teachers with the optimal tools for a rich educational experience.

The DynaLearn software reflects this through the integration of the following three modules, each providing different benefits to the overall application:

- **Conceptual Modeling (CM):** Offers a graphical editor to build diagrammatic representations to learners to articulate, analyze and communicate ideas, and thereby construct their conceptual knowledge [1]

- **Semantic Technology (ST):** Provides web-based ontology mapping which be used to find and match co-learners working on similar ideas to provide individualized and mutually benefiting learning opportunities [2]

- **Virtual Characters (VC):** A cast of different virtual characters can be called upon to make the interaction with the software engaging and motivating [3]

To further specify the scope of this report, work on the Virtual Characters consists of the following tasks:

- To enable learners to express their ideas on a conceptual model using a virtual character as a presenter that combines verbal and non-verbal means for communication

- To realize various kinds of dialog between virtual characters representing different roles and functions to explain a conceptual model

- To design communicative strategies for multiple agents that engage in a dialog about the model created by their learners

This Deliverable reports on the progress of work on the last two items, namely the results of Task 5.4 “Integration of tutorial planner and animated agents”.

Before we start with the main content of this deliverable, let us first provide a short reminder of the architecture of the VC module from Deliverable 5.2 “Basic Tutorial Tactics for Virtual Agents” [4], as shown in figure 1.

![Figure 1: VC module architecture](image-url)
appropriate content from the CM, arrange it into dialogs between the different characters and create the appropriate scene script that can then be played by the HL.

The IM itself consists of three different components:

- The Dialog Manager governs the overall interaction between the characters and the learner. It also has a dialog history that keeps track of all the characters’ dialogs said and the learner’s responses to them.

- The Verbalizer extracts necessary data from formal representations generated by the CM and transforms them to natural language through the use of scene scripts (dialog templates).

- The User Model keeps track of the learner’s knowledge about the model currently inspected or worked on.

The remainder of this document is structured as follows: We will first introduce to new character models that were designed for the new support types presented in this deliverable (Chapter 2). We will then take a detailed look at each of the three new support types Diagnosis, Quiz and Critic (Chapters 3-5). After that we will describe how all support types can be combined into integrated dialogs between the different characters (Chapter 6) and finally we will end with a conclusion and outlook of future work (Chapter 7).
2. New Virtual Character Models

During implementation of the different use cases and refining the software's interaction capabilities, we created two new virtual characters that cover the functionality of the use cases “Diagnosis” and “Critic”.

2.1. Mechanic

Diagnosis analyzes the contents of the models which the learners constructed and gives suggestions that might help them to find problems.

Taking up the question “Why is it not working?” allegorically the helping virtual character appears like a mechanic as shown in figure 2. He can be expected to take a look into a system or machinery and give a diagnosis of problematic issues. Therefore, he may increase the learner’s motivation by helping to discover which parts to improve.

Figure 2: A mechanic helps to solve problems.

Appearing as a mechanic, this virtual character is also suitable to show up in situations where the user needs technical help. For example the DynaLearn installation software contains multiple pages in the graphical user interface where technical advice is given. These hints which are a kind of additional information or system dependent suggestion are therefore also presented by the mechanic character. Figure 3 shows a screen of the installer where users have to install software depending on their respective system configuration. The installer reads the actual system configuration and generates a suggestion which is presented by the character.

Figure 3: Due to his mechanical relation the character also appears in technical tasks like installer advisory.
2.2. Critic

As shown in Deliverable 5.1 “Models and basic animation for characters” [5], there are multiple ways of providing feedback. The Critic support type could be performed by the Teacher character which already takes the role of giving basic help and orientation.

But, as we also pointed out in D5.1, there is a clear distinction between the support types of basic help and the semantic feedback: The first is answering learner’s questions in a constructive and positive way. The latter, however, is more formal and critical as it is based on a reference model, making the process of feedback somewhat similar to grading of the learner’s work. We thought that this difference in the support types should also be reflected in the characters that present them.

Therefore, we designed a new character, as shown in figure 4. The Critic now turns into an additional role on the side of the student hamsters. It has a rougher and naughty visual appearance compared to the other student characters which is meant to reflect the critical mentality.

2.3. Arbitrary character / role combinations

Since DynaLearn 0.9.0 the Virtual Characters can be assigned to different tasks, i.e. each of the different support types can be used with any of the characters, not just the assignment we envisioned. This is done by changing the dynalearn.conf configuration file (found in the DynaLearn directory).

```
<Characters>
  <BasicHelp>teacher</BasicHelp>
  <Quiz>quizmaster</Quiz>
  <Diagnosis>mechanic</Diagnosis>
  <Feedback>critic</Feedback>
</Characters>
```

The names used in his configuration are mapped to the different characters as shown in figure 5.

```
girl  boy  critic  teacher  quizmaster  mechanic  textonly
```

Figure 4: A student character available critical

Figure 5: The names for the Virtual Characters, as used in the DynaLearn configuration file.
“textonly” means that there will be no character at all for the respective role. Instead, only speech bubbles appear.

As an example, if you wanted the Basic Help role to be performed by the girl and the Diagnosis by no character at all, the corresponding section in your dynalearn.conf file should look like this:

```
<Characters>
  <BasicHelp>girl</BasicHelp>
  <Quiz>quizmaster</Quiz>
  <Diagnosis{textonly</Diagnosis>
  <Feedback>critic</Feedback>
</Characters>
```
3. Quiz

3.1. Introduction and Purpose

With the Quiz support type, we build upon work done in D3.3 “Question generation and answering” [6], or more precisely: the generation of multiple choice questions from a model built by the learner. Similar to the Teachable Agent mode, the default character to present these questions is the quizmaster. But different from the quiz challenge in the TA mode, learners now have to answer the questions themselves.

The process of putting together the questions and answers from their formal representations, presenting them with the character and collecting the learner’s answer is not too different from the other interactions we described previously. However, the more interesting point in this support type is the decision of which questions to ask in the first place: As described in D3.3, each question generated by the component has certain characteristics, such as a question type, a difficulty and one or more ingredients in the model that the question is about. Instead of using entirely random variations of these characteristics as input, we choose them so they are in line with the two purposes we want the Quiz to fulfill: On the one hand, it should be entertaining for the learner while on the other, it should further learners’ understanding of their models.

We achieve part of the former of these purposes through the avoidance of repetition, using our dialog history (which, in this case, simply keeps a formal representation of each question asked): First, each unique question is asked a maximum of two times and after its first occurrence it is only used again if there are no other questions available on a certain model ingredient. Second, the questions are selected in a way which will ensure that their types are as evenly distributed as possible. In other words, when given the choice between two otherwise equally suited questions, the algorithm will select the one whose type was asked less often in the current session. Again, this information is readily available in the dialog history.

As for the furthering of learners’ understanding, we employ two different mechanisms. One is a slowly increasing difficulty of the asked questions. As described in D3.3, there are the following difficulty types:

- Report: Question indicates place of answer
- Search singular: Answer is to be found near the references in the question
- Search plural: Answer must be found somewhere
- Explain singular: Learner must reason with uniform information to answer
- Explain plural: Learner must reason with pluriform information

With each new quiz session, we start with “Report”, the easiest of the five difficulty levels and after a set of three questions we adjust the level if necessary: If only one or even zero of these three questions was answered correctly, the difficulty is decreased. At two correct answers, it stays the same and for three it is increased.

Finally and most importantly, we try to ask those questions that will increase a learner’s knowledge about the model, i.e. we ask questions about those concepts which the learner apparently has not understood so far. This is achieved through the use of a Bayesian Network in which we track the learner’s knowledge progress, based on the ideas of the user model described in Deliverable 5.2.
3.2. Interaction Flow and Examples

For this example, let us assume that a learner has built a simple model and ran a simulation on it (as depicted in figure 6) and decided to start a Quiz.

The quizmaster then appears, greets the learner and asks the first question. Since this is the first question in the session every model ingredient would be eligible (as they are all assumed unknown) as the focus. The same holds for the question type, none of them has been asked more than others. The difficulty is set to the lowest level, "Report". Figure 7 shows the quizmaster asking that question, where the learner would look to find it, the answering of the question and the quizmaster’s response.

As we can see, the first question is about the ingredient “shade”. The learner takes a look at the value history for the respective state, finds the correct value and is able to answer it correctly. This is then entered into the user model, so future questions can take the increased knowledge of “shade” into account. Since only one question has been asked so far, the difficulty stays the same. Continuing with the example, figure 8 shows the next interaction cycle.
Here, we can observe the following things: The second question is not about the simulation but instead about the model itself, and the quizmaster points this out to the learner, i.e. he tells the learner which of the two different screens the information can be found in. However, the learner answers the question wrong, confusing the concepts of a proportionality with that of an influence. The quizmaster remarks on that, indicates the right answer and the fact that the “P+” is now less known is entered into the user model. The effect of this can be seen when we take a look at the third question, depicted in figure 9.

As we can see, this question is about “size”. Because the previous question was answered wrong, and knowledge of P+ influences the knowledge of both “size” and “shade”, their knowledge also decreased. So at this point, the knowledge of “growth” is still at its initial value, “shade” had both a knowledge increase and decrease and “size” only had a decrease. Therefore “size” is selected as the ingredient for the focus of the next question. We can also see that the question type is the same that was used in the first question. Although this goes against the idea of selecting a variety of question types, the heuristic had to be dropped in this case because there were no other questions available for “size” at the desired difficulty level. Speaking of the difficulty level, this will be reconsidered after the learner answers this third question.
The quiz can end in one of two ways: Either a certain threshold (cf. next section) is reached for the overall knowledge level of the entire model (meaning that the learner understands it well enough) or there are no more questions that can be asked. In the latter case, the overall model knowledge is then put into a verbal rating. Figure 10 shows three examples of the quizmaster announcing the ending of the quiz.

![Figure 10: Three different endings to the quiz](image)

Finally, to conclude this example, let us take a look at the “learning chart” that is generated for the beginning of this quiz session (figure 11). It shows the changes in the knowledge level from question to question for each ingredient in the model as well as the model itself.

![Figure 11: The learning chart for the example quiz session](image)

The plots show the changes in knowledge described above: The first answer increased the knowledge of “shade”, as well as its associated entity “tree” and the entire model. The second answer decreased the
knowledge of the proportionality, both associated quantities, their entity and the model. Also, as described above, “size” is the lowest of the three quantities when the focus for the third question is determined.

3.3. Knowledge Extraction and Representation

When the learner starts a new session of the Quiz, the first things that happens is the construction of the user model. This process is described in D5.2 but we will reiterate the most important steps here. Basically, the user model contains a node for each model ingredient, but these nodes are organized in different layers. First there is the so-called “generic layer” which contains a node for each ingredient definition in the model. Next is the “specific layer” which is made up of the nodes corresponding to the instances of these definitions, but only those which are actually active in the current simulation. Corresponding elements in both layers are then linked, so multiple instances of the same definition all add changes in their knowledge levels to the knowledge of the definition. Finally, answer collectors and question nodes are added to the “bottom” of the network (where new evidence will be entered) and a single node, representing the knowledge of the entire model, is added to the “top”.

To clarify the process of building the user model, let us take a look at how this would happen in the case of the Tree & Shade model used in the above example. First, the “model data”, i.e. the generic ingredients is requested from the CM and sent as an OWL structure. From this, the generic layer can be built as shown in figure 12.

![Figure 12: The generic layer](image)

As we can see, one node is added for each of the ingredients in the model. The exact process of extracting these generic ingredients from the OWL representation of the model and building the layer is as follows:

1. Find all ExpressionFragments and ModelFragments as well as their subclasses
2. For each of these fragments, find the elements that were added to them as consequences or conditions
3. For each of these elements, if it is an Entity, Quantity or Inter-Quantity relation, it is a generic ingredient so add a node for it

Next, the set of specific ingredients is requested, the “simulation data”. From this, the specific layer can be constructed, as seen in figure 13.
In this layer, the structure of the model itself is better represented than in the generic layer: Knowledge of the quantities feeds back to their governing entity, inter-quantity dependencies influence both associated quantities, and knowledge of quantities is further split into knowledge of the magnitude and the derivative. Again, the exact process of extracting this data from the OWL representation of the simulation and building the layer is as follows:

1. Find all Entities
2. For each of these entities, add a node and find all its Quantities
3. For each of these quantities, add a node, connect it to the entity’s node
4. For each of these quantities, add a node for its Magnitude and Derivative, connect them to the quantity
5. For each of these quantities, finds all its Inter-Quantity Dependencies
6. For each of these inter-quantity dependencies, add a node and connect it to both its associated quantities

Next, answer collector and question nodes are added, as depicted in figure 14.
At this point we need to introduce the concept of a "knowledge node", which is every node at the "edge" of the network, i.e. inter-quantity dependencies, magnitudes and derivatives. Each of these knowledge nodes receives an answer collector (orange) which in turn is linked to a question node (light blue). The question nodes are used to enter the actual evidence into the network, while the answer collectors will be used (though not in this example) to propagate the knowledge generated trough evidence in the question nodes to other knowledge nodes that are an instance of the same ingredient definition. A more detailed explanation of the use of answer collectors in the user model can be found in deliverable 5.2. Now, the specific nodes are linked to their generic counterparts. This is done by inspecting the `isInstanceOf` relation of each specific ingredient and finding the respective element among the generic ingredients.

Finally, the "overall model knowledge" node can be added to the network. This "overall model knowledge" or simply "model" node represents the threshold mentioned earlier. If its "Known" value is high enough, the quizmaster ends the quiz. Figure 15 shows the finished user model after the final construction step.

![Figure 15: The finished user model](image)

After the user model is now finished, the quiz can begin. Each quiz lasts for a number of interaction rounds until an ending condition is met as described above. Each interaction round consists of the following steps:

1. Determine least known ingredient(s) from the user model
2. Generate and send question request for that ingredient
3. Select a question from the received list
4. Have character ask the question and collect learner’s answer
5. Update user model based on learner’s answer
6. Check for quiz ending condition

Let us now take a closer look at each of these steps.

**Determine least known ingredient(s) from the user model** From the user model one or more ingredients are selected based on the following algorithm:

   If there is only one knowledge node of lowest probability: return it;

If there is only one knowledge node of lowest probability: return it;
If there is only one quantity of lowest probability: return it;
If there is only one entity: return the quantity node(s) with the lowest probability;
Return the entity / entities with the lowest probability;

Generate and send question request for the selected ingredient(s) An OWL structure is generated that represents a question request for the selected ingredient(s). The request also specifies the desired answer method difficulty. It is then sent to the CM.

Select a question from the received list Based on the request, a certain amount of questions is sent back. From this list, one question is selected based on the following algorithm:

If there is a question that was asked once or never and whose type is among the least asked types: return it;
If there is a question that was asked once or never return it;
Disable the restriction to a particular answer method; Generate and send a new request; return;
Select another concept; Generate and send a new request; return;
End the quiz;

Have character ask the question and collect learner’s answer Based on the type of the selected question, an appropriate scene file (i.e. dialog script) is selected from the scene library. Placeholders in this scene are filled with the appropriate data from the question, and the question and its answers can be presented by the character. Once the learner has selected one of the answers by clicking on it, the answer is evaluated and based on whether it was correct, the character displays an appropriate reaction, again by selecting and filling in a scene file.

Update user model based on learner’s answer Based on the concept that the asked question was about, evidence is entered into its corresponding question node in the user model. If the learner’s answer was correct, the node receives a “Known” evidence and an “Unknown” evidence otherwise. Then the user model is updated so the new information can be taken into account in the next interaction round.

Check for quiz ending condition As described above, if the “model” node in the user model has a high enough “Known” value, the quiz ends.
4. Critic

4.1. Introduction and Purpose

The Critic support type is based on the ontology based feedback generated by the ST module, as described in D4.2 “Ontology based feedback on model quality” [7]. Opposed to other support types which use a character to present content generated by a certain component within in the CM (such as the Basic Help or the Diagnosis), the Critic comments on and thus enhances the already existing presentation of the generated feedback.

To be more specific, the generated feedback is shown in a specific window within the software and can be inspected by the learner. But in addition, a learner can also call upon the Critic to get additional details on the information presented in this window, similar to the way in which human teachers might react when they see a feedback window on a learner’s screen.

There are five different kinds of comments that the character will use during an interaction: Category summarization, Score comment, Item description, Action suggestion and Category explanation. Specific examples of the different categories are given in the next section, the following description is more general. The first two kinds of comments are only used once in the whole interaction, more specifically right in the beginning, when the window appears for the learner to see. “Category summarization” does exactly that, i.e. it gives an overview about the number of feedback items in each of the different categories (Missing term, Extra term etc.). “Score comment” is a remark about the matching percentage between user model and reference model.

The other three types only come into account when the learner selects a feedback item from the list. The character first uses an “Item description” which contains more information and is less formal than the one already present in the window. From here, learners can ask how to proceed with this item (leading to an “Action suggestion”), ask for more and general information (“Category explanation”) or indicate their understanding of what needs to be done for this item. Through use of the dialog history we make sure that none of the comments is given more than once.

4.2. Interaction Flow and Examples

Let us suppose the learner has requested feedback on a certain model and is presented with the feedback window shown in figure 16.
Now the Critic appears and makes his two introductory comments (one after the other), “Category summarization” and “Score comment”, as shown in figure 17.

The mapping from the numerical matching rate to the character’s verbal rating can be seen in table 1.

<table>
<thead>
<tr>
<th>Matching rate</th>
<th>Verbal rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>0% - 39%</td>
<td>That really needs to improve.</td>
</tr>
<tr>
<td>40% - 59%</td>
<td>That could be better.</td>
</tr>
<tr>
<td>60% - 79%</td>
<td>That’s good!</td>
</tr>
<tr>
<td>80% - 99%</td>
<td>That’s very good!</td>
</tr>
<tr>
<td>100%</td>
<td>That’s perfect!</td>
</tr>
</tbody>
</table>

Next, the learner selects one of the items in the list, e.g. “Pipe”. The character will then give the appropriate description, along with the options to ask for the other comments as well. Figure 18 shows an example of how this interaction could look like.
Finally, to give a better overview of the different categories and the comments available for them, table 2 shows the different feedback categories with examples of the “Item description” and “Action Suggestion” comment types.

<table>
<thead>
<tr>
<th>Feedback Category</th>
<th>Item description</th>
<th>Action Suggestion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing Term</td>
<td>Apparently this term is part of the reference model but you don’t have it.</td>
<td>Why don’t you add it to your model?</td>
</tr>
<tr>
<td>Extra Term</td>
<td>This means that the term only exists in your model.</td>
<td>I suggest you simply remove the term from your model.</td>
</tr>
<tr>
<td>Missing Dependency</td>
<td>Ok, so this dependency is missing in your model.</td>
<td>How about adding the dependency to your model?</td>
</tr>
<tr>
<td>Extra Dependency</td>
<td>It seems that this dependency does not exist in the reference model.</td>
<td>Removing the dependency in question is probably the best idea.</td>
</tr>
<tr>
<td>Different Dependency</td>
<td>Ah, so the dependency connects the correct quantities, but its type is wrong.</td>
<td>Simply change the type so it’s the correct one.</td>
</tr>
<tr>
<td>Inconsistency</td>
<td>See, there’s an issue with your entity hierarchy.</td>
<td>Change your hierarchy so it matches the suggestions.</td>
</tr>
<tr>
<td>Terminology</td>
<td>Ok, this is something about the way you named your model ingredients.</td>
<td>Rename the term so it’s consistent with the reference model.</td>
</tr>
</tbody>
</table>

4.3. Knowledge Extraction and Representation

First, the list of feedback items is generated by the ST module and sent to the VC module as an OWL structure, the so-called “metamodel”. More details on the metamodel can be found in D4.2. From it the following data is extracted so it can be later used in the character’s dialog: The matching percentage, the number of feedback items in each category and the unique URI, category and description of each feedback item.
Then the character’s introduction can be constructed by filling in the placeholders in a set of scene files with the concrete values for the matching percentage and the items per category. When the learner then selects an item from the list, this item’s URI is sent to the VC module, where its category and description can be retrieved. Based on the category, a set of scene files are retrieved from the library. This set contains the character’s dialog for the other three comment types. With these, the rest of the character’s dialog can be constructed so it can react to the learner’s inputs accordingly. The description of a feedback item can be used to further customize the character’s dialog, i.e. instead of “Apparently, this term is missing in your model”, the character could also say “Apparently the term pipe is missing in your model”. Figure 19 shows an overview of the scene library’s structure for the Critic.

**Comment Types**

<table>
<thead>
<tr>
<th>Category summarization</th>
<th>Score comment</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scene</strong></td>
<td><strong>Score</strong></td>
</tr>
<tr>
<td><strong>Item description</strong></td>
<td><strong>Action suggestion</strong></td>
</tr>
<tr>
<td><strong>Category explanation</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 19: Scene library for the Critic**

The exact number of scenes for each comment type is different from category to category. If there is more than one scene for a comment, one scene will be selected randomly by the dialog manager as described in D5.2.
5. Diagnosis

5.1. Introduction and Purpose

The Diagnosis support type builds on the diagnosis component within the CM module, which is further described in D3.4 “Diagnosis and assessment” [8]. To offer a short recapitulation, this component helps learners to detect flaws in their model or to correct their expectations. Given the simulation of a model, the learner indicates his expectations through a window in the software. Following this, a certain number of so-called “probe questions” are generated by the system in order to pinpoint the mismatch between the real simulation and the learner’s expectations. Finally, the component compiles a list of model ingredients which are “responsible” for the mismatch, the so-called culprits.

In the scope of task 5.4, this component was enhanced with a virtual character (by default, this is the mechanic) to make the interaction described above more compelling and immersive for a human learner. This happens at two points during the interaction (confer next section): The asking of probe questions (and collecting the learner’s answers) as well as the presentation of the culprits.

While the first task can be handled through multiple-choice questions similar to the Quiz, we decided to make things more interesting when presenting the culprits to the learner: Based on the tutoring and communication strategies we identified in D5.2, the character does not outright tell the culprits to the learner but instead tries to lead the learner to this information. More specifically, the character makes use of Hints and Highlights to point the learner to the model ingredients in question. However, should the character run out of these before the learner indicates understanding, the character will, as a last resort, provide the correct answer directly. In this way, we ensure the learner can proceed with the further diagnosis and assessment of his model even if he cannot cope with a particular single issue.

5.2. Interaction Flow and Examples

Suppose that the learner opened the Tree & Shade model (see figure 20) and, after running a simulation on it, decided to start the Diagnosis.

![Figure 20: The Tree & Shade model](image)

First, the “Express expectations” window appears (figure 21) and the learner indicates the expectation, that “growth” should have a value of “zero” in state 4.
Based on this, the diagnosis component determines the first probe questions which can then be asked by the character. Figure 22 shows an example of this: The question is asked as a multiple-choice question so learners can easily give their input.

Depending on the learner’s answer and the initial expectations, more than one of these probe questions might be asked in a similar manner. But eventually, the component will arrive at one or more culprits which can then be presented by the character. As mentioned above, the character will make a number of statements (hints and highlights) about the culprit, getting more and more specific until the learner either successfully identifies it or the character runs out of statements and identifies it for the learner.

To continue our example, let’s suppose that the culprit is the “P+” between “size” and “shade”. Figure 23 shows which statements the character might use to point the learner towards this.
After this, the learner can either start a new round of Diagnosis or, if all problems have been sufficiently addressed, continue building their model.

Figure 23 shows some concrete example for the hints and highlights, but here are some more general ideas that we used when creating them:

- “The problem is not with….”: By selecting one of the other ingredient categories (those the culprit does not belong to) to complete this statement, a number of hints can be easily generated. Also, those hints are very general and serve as a good starting point. Of course, the character should not use too many of those, as this could get boring easily.

- “You should take a closer look at your causal dependencies / value assignments / quantity spaces etc.”: Grouping together similar categories (e.g. influences and proportionality as causal dependencies) is a way of easily creating highlights. However, because all groups are mutually exclusive, only one of them can be used in any given explanation.

- “The ingredient you’re looking for is a proportionality / influence / correspondence etc.”: Depending on the exact ingredient category, these highlight are already very concrete, especially if there are only a few elements of the respective ingredient in the model.

5.3. Knowledge Extraction and Representation

Once learners have indicated their expectations and the first step of diagnosis is complete, probe questions need to be sent to the VC module. As with all communication between the CM and VC modules (see, for example, the representation of the Basic Help in D5.3 “Basic help and teachable agent” [9]), the representation of a probe questions is based on an XML structure. The data contained in this representation is extracted and filled into placeholders in appropriate scene files.

A similar structure is also used to communicate the list of culprits, from which both the exact description of a culprit as well as its ingredient category (such as Proportionality, Influence etc.) are extracted.

Based on the category, a number of scene files are selected from the scene library. For each category a certain number of these scenes exist, and they are sorted into “levels”. Each scene contains a single statement, like the ones used in the previous example. To begin, the character will select a scene from the lowest level and each time the learner indicates a need for more help, the level is increased. Once a level is
reached for which no scene file exists, the character will use the exact description of the culprit instead. Figure 24 shows an example overview of this process.

![Image: Scene library for culprit presentation](image)

The exact number of levels and the number of scenes per level depends on the category. If there is more than one scene for a level, one scene will be selected randomly by the dialog manager as described in D5.2.
6. Integrated Dialogs

The idea behind the concept of "Integrated Dialogs" is that the characters should not only appear by themselves, presenting "their" support type, but also interact with the other characters already on screen, showing they are aware of them and respond to what they are saying.

To include this idea in the DynaLearn software, we identified three situations that would benefit from integrated dialogs and that learners could find themselves in: Introducing the characters and their support types, getting to know a model and model building support. In the three following sections we will discuss each of these situations in more detail. Please note that these situations are not recognized automatically, but are rather handled the same way as the normal support types: They can be requested by the learner through the software’s interface.

6.1. Introducing the characters and support types

This mode is intended for learners who have only little experience with the characters and the support they offer. When this mode is started, one randomly selected character will appear, greet the learners, give a short introduction about the software and then presents them with a list of possible questions to ask, all pointing towards the different characters and the support types they offer. When a question is selected, the character whose area of expertise the questions falls in will take the turn and present himself and explain how he can support the learners. Then he will offer the remaining questions to the learner again. This goes on until learners indicate they have no more questions. Figure 25 shows an example of a short interaction in this mode.

Through the use of the dialog history, characters "know" who acted before them (and what was said) and can make references to it.
6.2. Getting to know a model

In this mode, we combine the Basic Help and the Quiz in order to make the exploration of a new model more interesting and engaging to a learner. The idea is that both characters take turns in explaining a model ingredient (Basic Help) and then asking a question about it (Quiz).

After the user has selected a What is? or Why? question and listened to the answer, a quiz question will be generated for one of the ingredients related to that answer. For a What is? question, this can either be the enquired ingredient itself or one that is somehow related to it. For a Why? question, this can be any ingredient that is part of the causal chain behind the answer.

The Basic Help is used in its normal way, except that no follow up questions can be asked since this would break the flow of this interaction mode. For the Quiz, the only change is the selection of the ingredient to ask about, this is done by the method described above and not through the user model. However, the user model is updated with the learner's answers so if a “real” Quiz is run later, this information can be taken into account. Figure 26 shows an example of this mode.

![Figure 26: Combining Basic Help and Quiz](image)

The two characters will also acknowledge each other’s presence and make appropriate comments.

An easy to realize variant of this mode would be that the learner can ask the Basic Help character a couple of successive questions first, followed by some quiz questions about all ingredients covered before. This way the back and forth between both involved characters would be reduced.

6.3. Model building support

With this mode, be combine the support types of both the Diagnosis and the Critic to assist learners when they run into trouble while building their models, or more specifically, when the model does not work in the way they intended. When a learner first activates this mode, both characters will appear and remind the learner of the specific support type each of them can offer and in which situation this support would be appropriate and helpful. Figure 27 shows an example of what the characters might say in this situation.
After this and based on the characters’ suggestions, the learner has to decide which support type to actually use. If necessary, the characters could give a short reminder of how to do that. Then the support type is presented by the corresponding character in the same way as described in the respective sections above, with one difference: When appropriate, the other character will join the dialog and make remarks and comments. Figure 28 shows two examples of how this would look like.

As we can see, this works in both directions: On the one hand, the Mechanic can remind the learner if model ingredients from the feedback played a central role in the diagnosis before (i.e. they were culprits or the learner had expectations about them). On the other hand, when the Mechanic is giving hints about a culprit during the Diagnosis, the Critic can give additional hints if the ingredient in question was part of a previous feedback.

A possible addition to this mode could be that, if certain “patterns” are present the characters make additional comments on those. For example, for the first situation shown in figure 28, the fact that a term should be removed that was the culprit of an earlier diagnosis could be seen as the reason that the diagnosis was necessary in the first place and the characters could point this out (“See? That’s why you had different expectations earlier!”).
7. Conclusion & Outlook

In this document we introduced three new virtual character based support types that are now available in the DynaLearn software.

For two of the new support types we created new virtual character models, following the guidelines to role and personality design that we elaborated in Deliverable 5.1 "Models and basic animations for characters".

We showed how each of new support types takes knowledge generated in either the ST (Critic) or CM (Quiz and Diagnosis) module and transforms it into a presentation by the virtual characters. This transformation makes use of the dialog system with all its subcomponents that we established and described in the scope of Deliverable 5.2 "Basic tutorial tactics for virtual agents": Dialog management, verbalization engine, user model and tutoring strategies.

We also demonstrated how the different characters and support types can be integrated with each other with only minor changes, again using the different parts of our dialog system, and also how these integrated dialogs could further enhance a learner's experience with the software.

To conclude this final deliverable within work package 5, we can say that all tasks for the VC module are completed and can be included in Milestone 3 – Full system and be tested in the upcoming evaluations of work package 7.

These evaluations are also tie-ins for future work as their insights and results will show opportunities to make improvements to specific details of the characters’ dialogs and behavior.
References


[8] Beek, W., and Bredeweg, B. (2011). Diagnosis and assessment. DynaLearn, EC FP7 STREP project 231526, Deliverable D3.4
