Deliverable number:	D3.3
Deliverable title:	Question generation and answering

Delivery date:	2010/07/31
Submission date:	2010/08/31
Leading beneficiary:	University of Amsterdam (UVA)
Status:	Version 05 (final)
Dissemination level:	PU (public)
Authors:	Floris Linnebank, Jochem Liem, Bert Bredeweg

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

Abstract

This deliverable presents the question generation module in the DynaLearn software. Its role is to provide question and answer sets in several contexts ranging from a multiple choice quiz to questions, answers, and explanations in response to queries from the learner. It has the following main features: A mechanism for scope setting, constraints analysis and the application of selection heuristics. A question and answer construction mechanism based on model simulation input. A multiple choice distractor generation mechanism. A flexible formal output OWL format. A mechanism for interpreting questions and answers in response to learner queries and recombining and analysing output to produce elaborate explanations. The final main feature is a mechanism to construct a teachable agent challenge matching questions from one model with answers from another model.

Internal review

- Michael Wißner & René Bühling (UAU), Multimedia Concepts and Applications, University of Augsburg
- David Mioduser (TAU), Dept. of Education in Math, Science & Technology, Tel-Aviv University

Acknowledgements

The authors would like to thank Michael Wissner (UAU), René Buehling (UAU), Wouter Beek (UVA) and David Mioduser (TAU) for their contributions to the design, implementation and description of the work presented here.

Document History

Version	Modification(s)	Date	Author(s)
01	Outline + first text	2010-07-14	F. Linnebank
02	First version after comments	2010-07-21	F. Linnebank, B. Bredeweg
03	Question Formalisation section	2010-07-22	J. Liem
04	First draft for internal review	2010-07-22	F. Linnebank, B. Bredeweg
05	Final version after review	2010-08-20	F. Linnebank, B. Bredeweg

Contents

1. Introduction

An important module within the DynaLearn interactive learning environment is the question generation. This module is called upon from different contexts and has to work with largely varying inputs, particularly in terms of models and learner specific needs. It also has to impose sensible scoping on the generation process in order to deliver relevant questions in the context of the many questions that in principle can be generated.

Question generation is a subclass of Natural Language Generation (NLG; www.questiongeneration.org; Wang, Hao & Liu 2008). It is often done using templates that are applied to (prestructured) natural language corpora about the subject (Aldabe, Lopez de Lacalle, Maritxalar, Martinez & Uria, 2006; Wang et al., 2008). In some cases structured subject material, such as a computer program, is used to generate questions (Myller, 2007). A similar approach based on structured subject material was taken in previous work on question generation for Qualitative Reasoning (Goddijn, 2002; Goddijn, Bouwer, & Bredeweg, 2003). This work produced the QUAGS (QUestions about Garp Simulations) question generator which was used as a starting point for the current research.

The aim of Task 3.3 is to construct a question generator component in the DynaLearn system that provides input for several functions in several use cases. Firstly in the quiz use case (D2.1: Bredeweg et al., 2009) the quizmaster virtual character is supplied with questions and multiple-choice answers regarding the currently loaded model and its simulation. In this use case the quizmaster asks questions to the learner, whereby the latter learns about the model. The series of questions is adapted to the learner's performance. The system selects follow up questions based on previously given answers by the learner. A user model is maintained for this purpose (D5.2: Wißner et al., 2010) and therefore the question generator must be able to generate questions given a certain scope as input. Secondly in the teachable agent use case (D2.1: Bredeweg et al., 2009) the learner can construct questions and pose these to the teachable agent virtual character. Here, the question generator is responsible for providing answers and explanations in response to these questions. Also the teachable agent can be given a challenge quiz in which case the question generator is responsible for providing both the questions and answers.

The communication between components in the system is done via socket connections, using the OWL language for actual knowledge content and plain XML for simple machine state communications (D3.2: Liem et al., 2010).

The next section will introduce the question generator component of the DynaLearn system more in detail. Section 3 describes the formal structure of the output. Sections 4 and 5 present the role and implementation of the question generator in two use-cases, the multiple choice quiz and the teachable agent, respectively.

2. The Question Generator Component

The question generator component in the DynaLearn system provides a flexible information source providing pure question and answer sets, as well as analysis and summarizations of simulation behaviours. The question generator is a significantly extended and improved version of the QUAGS question generator (Goddijn et al., 2003) and also incorporates an additional reasoning layer to aggregate multiple questions/answer combinations into broader explanations. It is constructed using Prolog (www.swi-prolog.org) and takes as main input the simulation output of the qualitative reasoning engine in the DynaLearn system.

Question generation for DynaLearn should be fully domain independent such that the techniques used do not need adaptation for models of different domains. This is achieved by taking advantage of the structure provided in the model and simulation by the generic modelling and simulation ingredients. These structures are of course instantiated with domain specific concepts and therefore the questions are in fact focused on the domain itself. Simulation behaviour in single states or along simulation paths can be questioned thereby targeting a full understanding of the working of a model. Even in small models many questions can be generated (Goddijn et al., 2003), and restricting the amount of questions and selecting questions relevant to the learning objectives is therefore an important task.

Figure 2.1 illustrates the context in which the generator operates in the DynaLearn architecture. The Generate Questions inference in this figure is an extended and improved version of the QUAGS question generator and it is further enveloped with inferences to drive it and use its input for several types of output.

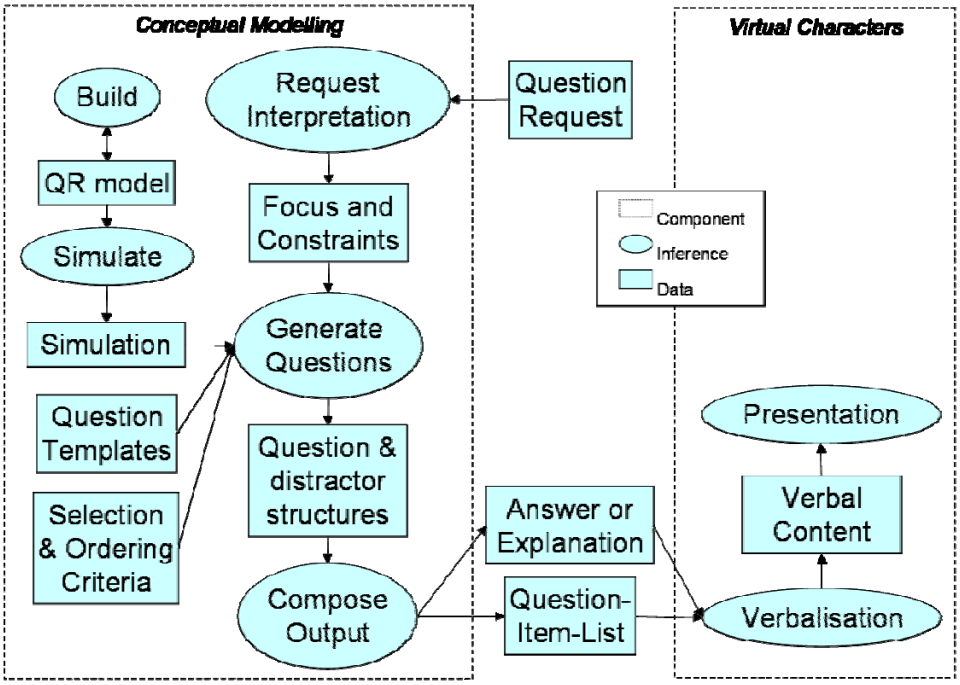

Figure 2.1: Context of the DynaLearn Question Generator component. The Question generator is invoked by a question request which is interpreted and which provides a focus for the main generator inference. Question structures are interpreted according to the request type to compose output that can be verbalised and presented.

In the next subsections the question types present in the question generator are discussed as well as a more detailed architectural description of the question generator, thereafter the scope setting is presented which provides a first set of constraints for limiting the output. Lastly heuristics used to further constrain output and provide a relevant set of questions are explained.

2.1. Question Generator Architecture

The question generator architecture is outlined in figure 2.2. Main input for the generator component is the qualitative simulation of the loaded model. Focus and constraints are input that guide the design and selection of questions such that a reasonable set of questions is produced instead of the very large set of questions that can be theoretically produced for any nontrivial model.

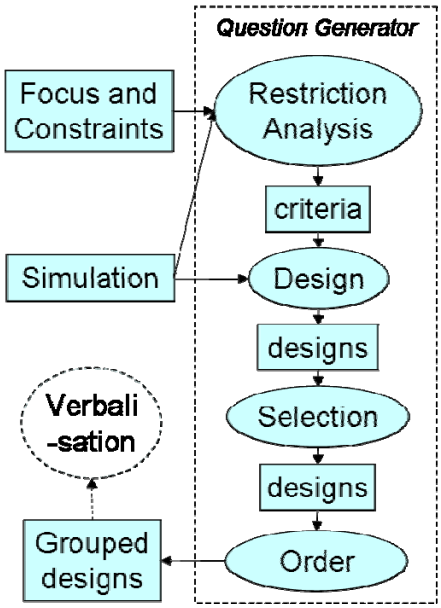

Figure 2.2: Question Generator Architecture: Main input for the question generator is the qualitative simulation and constraints. Output is a set of grouped question designs that can be verbalised into natural language.

The generation of questions is done in four steps. First the given restrictions are analysed with respect to the simulation and completed with built in heuristics. Then the resulting criteria lead to the generation of a set of question designs based on the simulation input and a set of question templates. Each of these question types has some associated constraints that determine when the question may be valuable. The amount of designs is generally still large at this point and therefore thirdly a selection inference determines the best set of questions given the full set of successful designs. Fourthly this final set of questions is put in a logical order with groups of questions for every state in the simulation. Verbalisation into natural language is done outside of the question generator component and this process as well as the format output by the question generator is discussed in section 3. The next subsection discusses the question types present in the question generator.

2.2. Question Types

The question generator can generate the 23 different question types listed in table 2.1[1]. Example quantity names are used from a model of a container with in and outflow. A dependency view of this model is given in figure 2.1. Note that the actual state of the model in this figure does not necessarily correspond to all questions. Although a virtually unlimited amount of question types is possible, the question types are designed to focus on explaining model behaviour. This is done by focussing on generic behavioural aspects of the simulation. Since simulation behaviour revolves around changing quantities, the question types target quantities and the causal model: the drivers and propagations of quantity changes.

Table 2.1: Question types

Nr.	Keyword	Example Question
1	GiveValue	*What is the value of Amount?*
2	ExplainEffectOfChange	*Amount changed from state S to this state; namely its derivative rose from min to zero, what is the effect of this change on Height?*
3	PredictValue	*What will be the value of Pressure in the next state?*
4	DescribeBehaviour	*What is going to happen with Amount during simulation?*
5	ExplainCause	*Why does Amount decrease?*
6	EffectiveYesNo	*Is the influence from Flow (Tap) on Amount effective?*
7	ExplainSubmission	*Why does Amount decrease, although Flow (Tap) is positive and Flow (Tap) has a positive influence on Amount?*
8	ExplainCorrespondence	*Why does Height have value min?*
9	ExplainCalculation	*Why does 'Quantity' have value min?*
10	EnumerateInfluences	*Which quantities have a direct influence on Amount?*
11	ExplainCausalChain	*How does Flow (Tap) influence Pressure?*
12	InterpretInequality	*Which quantity is greater, Flow (Tap) or Flow (Hole)?*
13	InterpretCausalChain	*Does Amount influence Flow (Hole)?*
14	SummarizeAggregateCausalChain	*Describe the influence of Flow (Hole) on Pressure.*
15	DescribeAlternativePath	*Amount changed from the state S to this state: its value decreased from plus to zero, what else could have happened?*
16	InterpretLoop	*We saw that 'QuantityA' influences 'QuantityB' and vice versa. Does 'QuantityA' influence itself via 'QuantityB'?*
17	EnumerateQuantitySpace	*Which values can Pressure adopt?*
18	PredictPossibleValues	*What can possibly be the value of Pressure in the next state?*
19	PredictWhichQuantityChanges	*Which quantity will be changed in the next state: Flow (Hole) or*

[1] In this table the 'Quantity (Entity)' notation is only used in ambiguous phrases, by default just the quantity name is given for brevity. Also questions not applicable to the example model use generic *'QuantityA' example names.*

		Amount?
20	DescribeCausal	*Describe the influence of Height on Pressure.*
21	DescribeCorrespondence	*What kind of connection is there between the values of Amount and Height?*
22	ExplainDerivativeEquality	*Why does 'Quantity' decrease?*
23	DirectInfluenceYesNo	*Does Amount influence Pressure directly?*

The constraints mentioned in the previous section associated with each question type are given in appendix A. Example answers for each question type are given in section 4.2.

2.3. Scope Setting & Selection Heuristics

One of the most important problems with automatic question generation is the large amount of possible questions. A moderately complex model may already produce roughly 72000 possible questions (Goddijn et al., 2003). The reason for this large number is the detailed character of the question types: they are about small aspects of the simulation, which occur many times. To control the amount of questions, we adopt two approaches. Firstly a certain part of the simulation can be selected as subject for the questions as well as a subset of the question types. This can be done by placing five *restriction methods* on the generation process. Secondly heuristics are used to construct and select the most interesting questions. The question generator is organised such that it automatically starts to apply heuristics when insufficient restrictions are given and the amount of questions becomes too large.

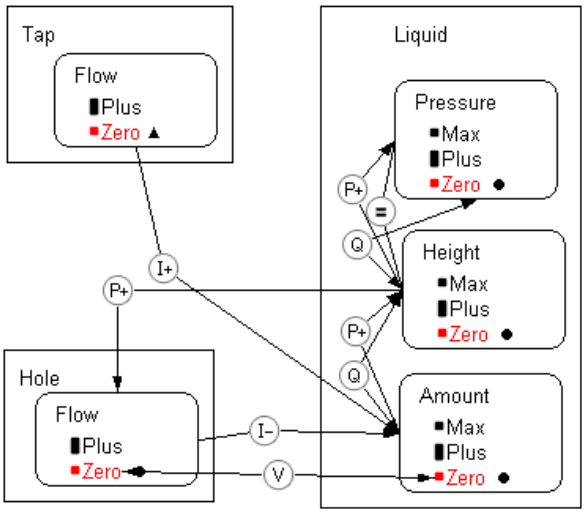

Figure 2.1: Dependency view of the contained liquid with in and outflow model. The tap has a flow that has a positive influence on the amount. Amount determines height and pressure. A hole in the container has a flow that negatively influences the amount in the container.

2.3.1. Restriction method 1: Question type criteria

If all question type criteria are undefined, the question generator takes only the question types that have as *behaviour* criterion[2] the value *real*. This yields the most general questions types.

2.3.2. Restriction method 2: Selection of states

The entered states define, together with the *perspective* criterion, the states for which questions will be generated. If the perspective criterion is *simulation run-through* then the following applies. The questions of this perspective criterion are about changes over states that are directly connected. Therefore states should succeed each other. If the entered states do not form a connected graph, intermediate states are added to make the graph connected. If no states are entered at all, the shortest path(s) between start and end state(s) are chosen.

If the perspective criterion is *causal model* then no graph is needed because the information for these questions can be found in a single state. Therefore only the entered states are used. Just in case no states were entered, the shortest path(s) between start and end state(s) are chosen.

If the perspective criterion is not defined then both causal model questions and simulation run-through questions must be possible and therefore a graph is constructed as with the simulation run-through perspective. If no states are entered, the question generator uses the start and end states and the shortest path(s) that connect(s) these states. When searching for a path the shortest one is preferred because of two reasons. First, the shortest path contains all changes needed to transform the start state into the end state. Changes that are not on this shortest path do not contribute to the final situation and can therefore be considered as 'superfluous' detail. Secondly, a short path minimises the amount of repeated information and as a consequence the amount of questions.

2.3.3. Restriction method 3: *System scope*

The system scope defines the part of the system for which questions may be generated. By defining a scope, separate entities can be investigated, but also specific processes. Processes tend to deal with a few quantities from several entities. Therefore, the scope may be defined in terms of entities or quantities. This method can be used to focus the question generation process for example in the quiz use case (see section 4). If no scope is entered, the question generator considers the whole system.

2.3.4. Restriction method 4: *Subject quantities*

This is an important restriction method. First, because selecting a small set of subject quantities excludes a large number of questions. Second, because all questions are about quantities, as a

[2] See appendix A.

consequence, the selection of the subject quantities determines for a large part which questions will be posed.

Subject quantities may also be defined as input to the question generator. All questions must then be about at least one of these quantities. If no quantities are defined, the question generator uses two heuristics for determining the subject quantities. The most significant heuristic is that quantities that change most during the selected part of the simulation (the selected states) are considered important. In addition to this heuristic, the question generator considers source or exogenous quantities as important, because they are the causes of the described behaviour in the simulation. An exogenous quantity is not influenced by the system itself and drives some behaviour in the system.

2.3.5. Restriction method 5: Forbidden subject quantities

If some quantities fall under the system scope but may not be explicitly named in the questions or answers, they can be indicated as forbidden. If no forbidden subject quantities are entered, the question generator leaves this restriction empty.

2.3.6. Selection Heuristics

In the selection phase, the full set of questions generated so far is considered: questions will only be dropped when some other question is better suited, because the goal is to make a diverse set with just a few questions of each type. One general mechanism is that questions about quantities with a high grade of change (investigated at restriction method 4) are considered more important than questions about quantities with a lower grade. Another mechanism used is that some questions have more value if they are combined with other questions, possibly from other types. For example, the question about a quantity is relevant when there is another question that asks why the quantity has this value, or what is going to happen with this value during the simulation. Besides deciding which questions should be selected, the question generator must also determine when the questions should be posed, because most questions are available in several states. Some questions trigger information that can be valuable in later stages; therefore these questions are posed in the first state possible.

3. Question Representation and Verbalisation

The Teachable Agent and Quiz use-cases (Bredeweg et al., 2009) make use of the questions generated by the Conceptual Modelling (CM) component. In both use-cases, the Virtual Characters (VCs) communicate the question (and in the Quiz mode they also communicate the answers) via a text balloon and spoken language. In the current implementation, the questions and answers are converted into natural language in the CM component.

In the envisioned final use-cases the VC component should dynamically generate the verbalisations of the questions and answers (based on previous interactions with the learner). Making the phrasing dynamic should allow the verbalisation to be part of an ongoing dialogue with the learner. For example, details that a learner already knows should not be repeated. Dynamically generating phrasings should also make it possible to research the effect of different verbalisations in an educational dialogue. For example, does friendlier or stricter wording have an effect on learning outcome? Furthermore, we aim to make the architecture general enough to make it possible to generate verbalisations in multiple languages.

To achieve these goals, the questions generated by the CM component should not be communicated to the VC component in natural language. Such a representation is difficult to convert into different sentences with different phrasings or other languages. Instead, the questions generated by the CM should get a formal representation that can be converted into different sentences. The obvious choice of a knowledge representation language is the Web Ontology Language (OWL), since the important representations within DynaLearn are also communicated in this format (D3.2; Liem et al., 2010). This representation should allow the flexibility to generate questions using different phrasings and in different languages. In the near future, the narrative planner in the VC component will use these representations as part of the dialogue generation (Interaction Manager; Wissner et al., 2010).

3.1. Representation of models and simulations

The OWL formalisation of the questions makes use of the many other representations used in DynaLearn. An overview of these representations is shown in Figure 3.1. The figure is a strongly simplified version of the representation meant to explain the conceptual structure of the formalisation. Each distinct representation can be considered a conceptual layer that makes new ontological commitments. Each conceptual layer defines a new vocabulary based on the terminology in the previous conceptual layer. The model fragment and super state layers are based on the model fragment and simulation shown in Figure 3.2 and 3.3. The definitions in OWL, QR vocabulary and model ingredient definitions layers are not shown in this document, but can be found in (D3.2; Liem et al., 2010).

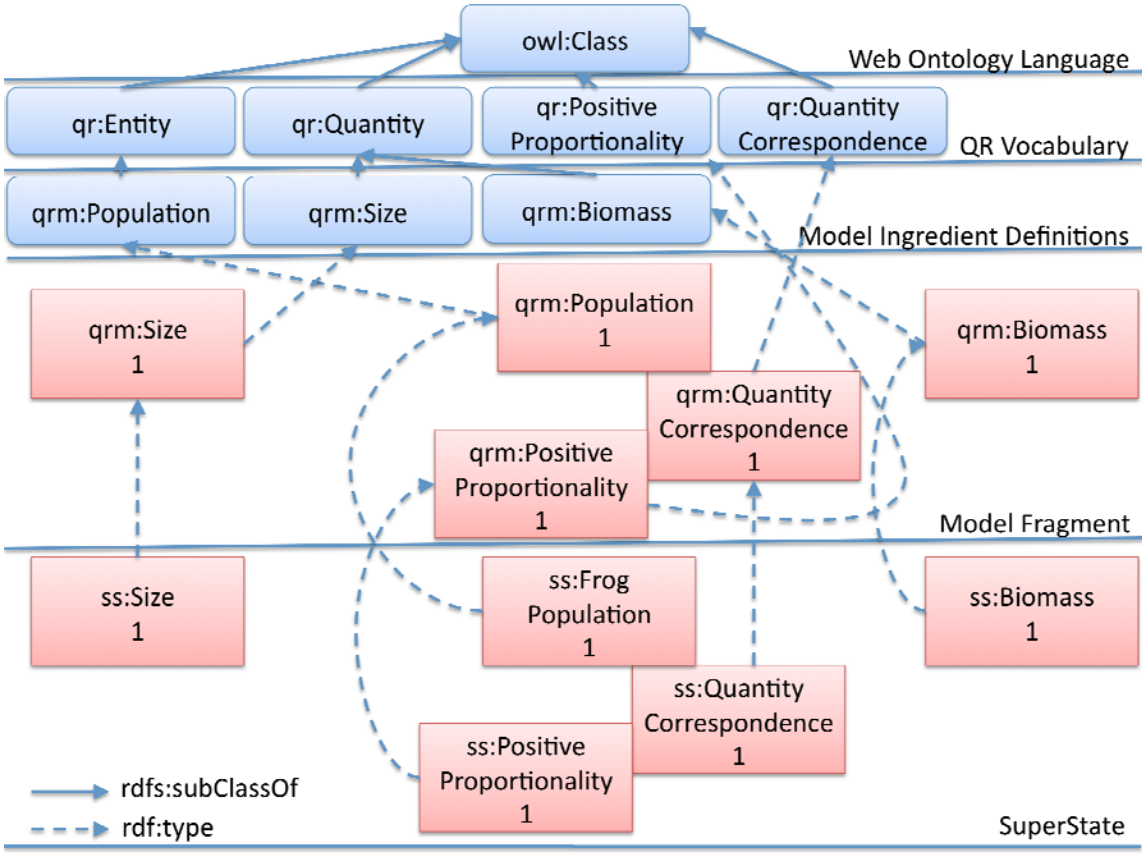

Figure 3.1: Different conceptual layers in the formalisation of the QR vocabulary, models, and simulations

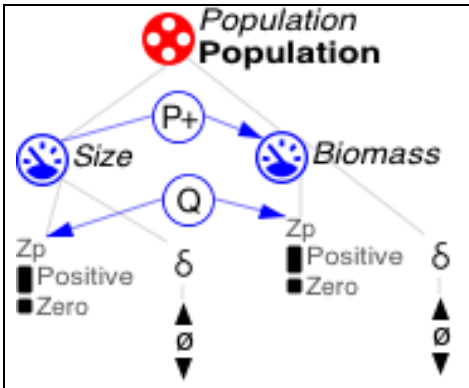

Figure 3.2: The Population model fragment

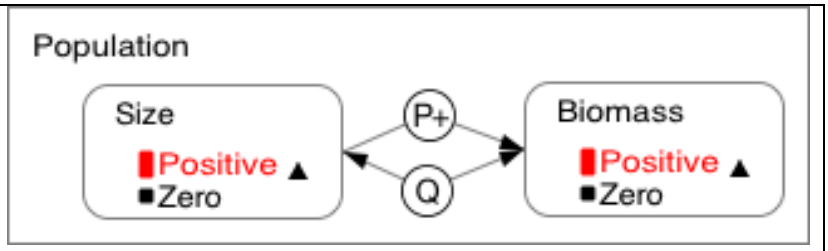

Figure 3.3: Simulation in which the Population model fragment fires

Each concept in a conceptual layer is defined by a URI, which is defined by a *namespace* and a *local name*. For example, the concept *Entity* is defined by the URI http://www.science.uva.nl/~jliem/ontologies/QRvocabulary.owl#Entity, which can be abbreviated by using the namespace 'qr' to refer to the URI of the ontology, and the local name 'Entity' to refer to the concept within the ontology (so qr:Entity can be used to refer to the full URI of the concept).

Previous research has identified different types of ontologies based on the type of ontological commitments that are made (van Heijst et al., 1995). We use the same vocabulary to talk about our conceptual layers. The OWL language[3] itself functions as our *representation ontology*. Its key representations are classes, instances and properties. Anything represented using OWL has one of these types. Using the OWL language, we created a *generic ontology* that formalises all the model ingredients that can be used in a DynaLearn QR model. For example, it provides definitions of the concepts entity, quantity, model fragment, etc. We call this representation the QR vocabulary[4]. The QR vocabulary only refers to concepts in the OWL language. For visualisation purposes only a reference to owl:Class is made in Figure 3.1, however most concepts in OWL are used in the QR vocabulary. The QR vocabulary is available on the web, and is also distributed within DynaLearn (to allow working offline).

Based on the QR vocabulary, representations of models can be generated (and such functionality is implemented in DynaLearn). Such model ontologies use the definitions in the QR vocabulary to represent the contents of the model. Since the resulting models are domain-specific, they can be considered *domain ontologies*.

Conceptually, the model ontologies can be separated into two parts. Firstly, the model ingredient definitions define the model ingredients that can be used in expressions, model fragments and scenarios (such as the entity Population and the quantities Size and Biomass in Figure 3.1). Secondly, the expressions, model fragments and scenarios describe situations using instances from the model ingredient definitions and instances of concepts in the QR vocabulary (such as causal relationships). The model fragment representation shown in Figure 3.1 is based on the model fragment shown in Figure 3.2. For simplicity, the representation of the quantity spaces is not shown.

Based on a scenario (which describes the initial situation of a specific system) and a library of model fragments a simulation can be generated by DynaLearn. Such a simulation is visualised as a state graph in which each state represents a qualitatively unique state of behaviour. For purposes of efficient representation, the concept of a super state has been developed. This super state represents the union of all the concepts in each state in the state graph (except the particular values it has in each state). In addition to this representation the state graph is also stored, but without its contents. This is done to prevent redundancy in the representation and to prevent the representation to become too large (in terms of memory and communication load within the DynaLearn software).

The super state visualised in Figure 3.1 represents the model fragments from which the model ingredients originate (based on the simulation shown in Figure 3.3, but without the values being shown). This allows a learner model (which is updated after a question is answered) to update both the knowledge about the simulation and the knowledge about the current simulation. The simulation representation is separate from the model representation (it is not stored when a model is exported to OWL), as simulations can be trivially recreated using the software (and simulation representations tend to inflate the size of the model files).

In Figure 3.1, the type relationships from the super state ingredients to the model ingredient definitions and QR vocabulary are not shown to keep the figure readable. These type relationships are equivalent to the type relationships originating from the model fragment. Furthermore, the relationships between the model ingredients in model fragments and simulations are not shown in Figure 3.1 (again to keep the figure readable). These relationships are visualised in Figure 3.4.

[3] http://www.w3.org/2002/07/owl#

[4] http://www.science.uva.nl/~jliem/ontologies/QRvocabulary.owl#

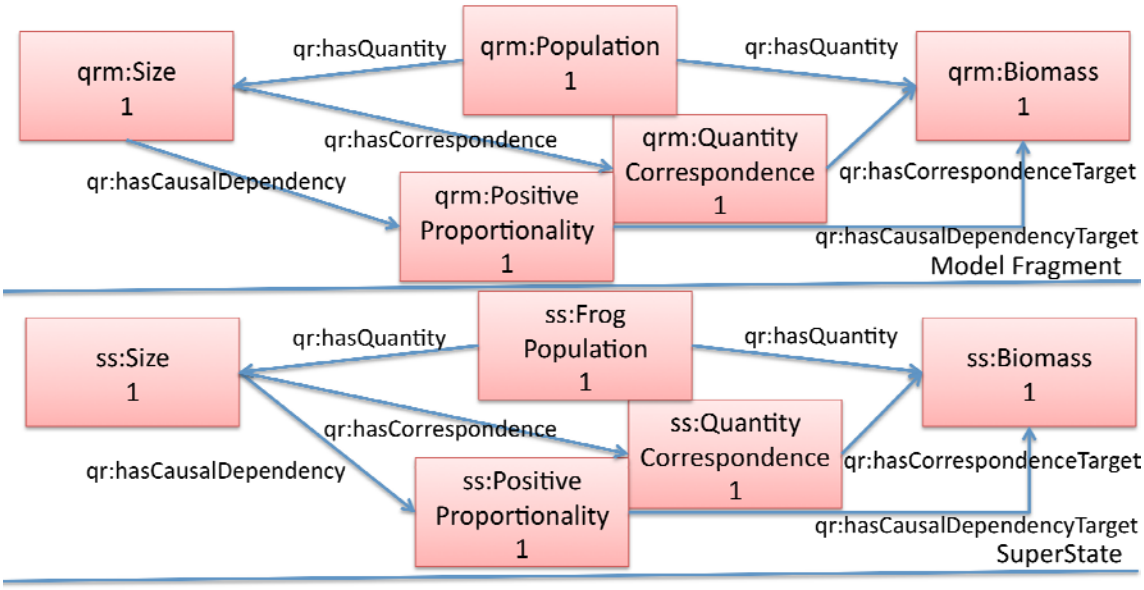

Figure 3.4: The relationships in the model fragment and super state layers

3.2. Representation of questions and question requests

To start the Quiz use-case, the CM component sends a request to the VC component. After this request, the VC component starts the quiz master character, and all further interaction is performed via this character. Firstly, the VC component sends a request to the CM component for the model representation and the super state representation of the current simulation. Based on the super state and the model, a Bayesian network is created that represents the knowledge of the learner. The VC component generates question requests that are sent to the CM component based on the current knowledge of the learner (as reflected by the Bayesian network). The question request is parsed by the CM component, and based on the current simulation a set of questions (and answers) are generated. These questions and answers are then sent to the VC component and communicated to the learner. Answering the question updates the Bayesian network and results in new question requests being sent to the CM component.

For purposes of representing the questions, answers, and question requests, a question ontology[5] was developed. A question and question request import the question ontology, but also the super state and QR vocabulary. The question request is represented using concepts in the question ontology, but also refers to concepts in the super state and QR vocabulary. Each question request consists of a question request identifier (which identifies this request uniquely), and a reference to the super state that represents the simulation that the question should be about. An example question request is shown in Figure 3.5 (with the question namespace called 'dl', and in which the request and super state properties are not shown).

[5] http://staff.science.uva.nl/~jliem/ontologies/QuestionOntology.owl

Each question request instance can have properties that represent each of the selection criteria discussed in Section 2.3. For example, a question about a particular concept can be requested by creating a question request with the *hasConcept* property to one of the QR concepts in the QR vocabulary (e.g. qr:CausalDependency, qr:Magnitude or qr:Derivative). Questions about a particular entity in the simulation can be requested using the *hasSystemScope* property to one of the entities in the super state (e.g. qrm:Tree1). And finally, questions about particular quantities, but in which other quantities are not involved can be requested using the *withSubjectQuantity/withNonSubjectQuantity* properties referring to particular quantities in the super state. Similar constraining properties exist for answer method difficulty, questions about particular states, and particular types of behaviour (submissive, dominant). However, these types of constraints are not yet used in the current implementation (they are envisioned to be used once the dialogue functionality matures). A particular one of the 23 question types (Table 2.1, Section 2.2) can be selected by using the *withQuestionType* property to one of the question type instances in the question ontology.

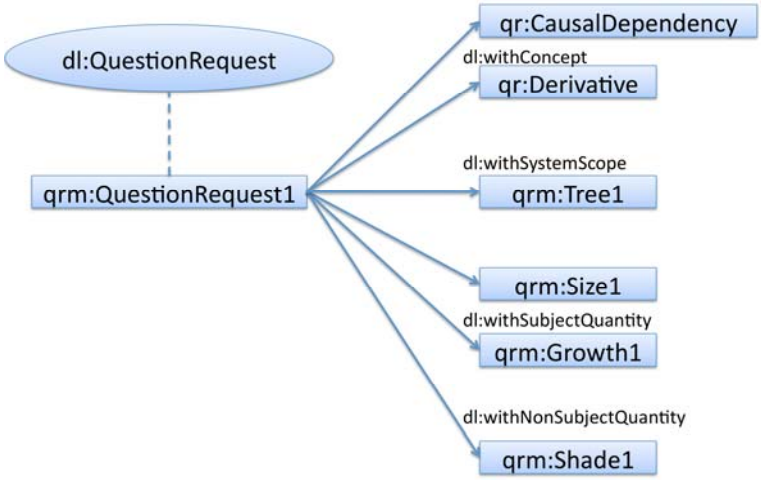

Figure 3.5: Formalisation of a question request

The structure of a question is shown in Figure 3.6. A question is identified by a unique question identifier, and refers to the question request it was based on. Furthermore, a reference to the super state that it is based on is included. Each question has precisely one question type (represented in the same way as for question requests). Furthermore, each question type is identified by a particular question type number and a question type keyword. Question types have been modelled as instances to allow potential other features of question types to be represented in the future, such as default templates for sentences generated for particular types of questions.

The main content of a question, the actual question itself, is divided into two parts: the context and the requested information. The context represents the framework of the question, i.e. the information that is needed to correctly answer the question. The requested information represents the knowledge that should be provided in the answer. Consider the following situation in state 1. There is a population with a natality and mortality rate. The natality rate positively influences the size of the population, while the mortality rate has a negative influence of the size. In this state the natality rate is greater than the mortality rate. Consider the following question: “In state 1, why does the size of the population increase, although the mortality has a negative influence on the size?”

The question context indicates that the question is about state 1. In the context, the population entity, the mortality and size quantities, and the negative influence are represented. The question type indicates that this is a ‘why’ question, and there is a special why relationship to the increasing derivative value of the size quantity. This representation of the context allows the context part of the

question to be converted into natural language (while choosing to exclude certain aspects that are already known by the learner).

The requested information part of the question also represents the correct answer. In this case it includes all three quantities and their influences and the inequality relationship between the natality and mortality rates. Although strictly speaking, this information suffices as an answer, the natality influence is also indicated as being dominant, and the mortality influences are indicated as being submissive. This allows the correct answer to be generated: “Natality has a positive influence on size, and mortality has a negative influence on size, and natality is greater than mortality, as a result, size increases.”

For incorrect answers similar structures are used, but they are based on either fictional descriptions of the state, or descriptions that are irrelevant to the question that has to be answered.

To summarize, the developed formalisation allows questions, answers, and question requests to be represented in OWL. The representation allows different verbalisations to be generated, as it provides only the essential information, without specifying an explicit order. Furthermore, since DynaLearn has multi-language support (and the descriptions in different languages are also in the OWL representations of the model and the simulation), questions and answers can potentially also be generated in different languages. These features of the representation fulfil the requirements set of the Quiz and Teachable Agent use-cases, while allowing possible future extensions to DynaLearn.

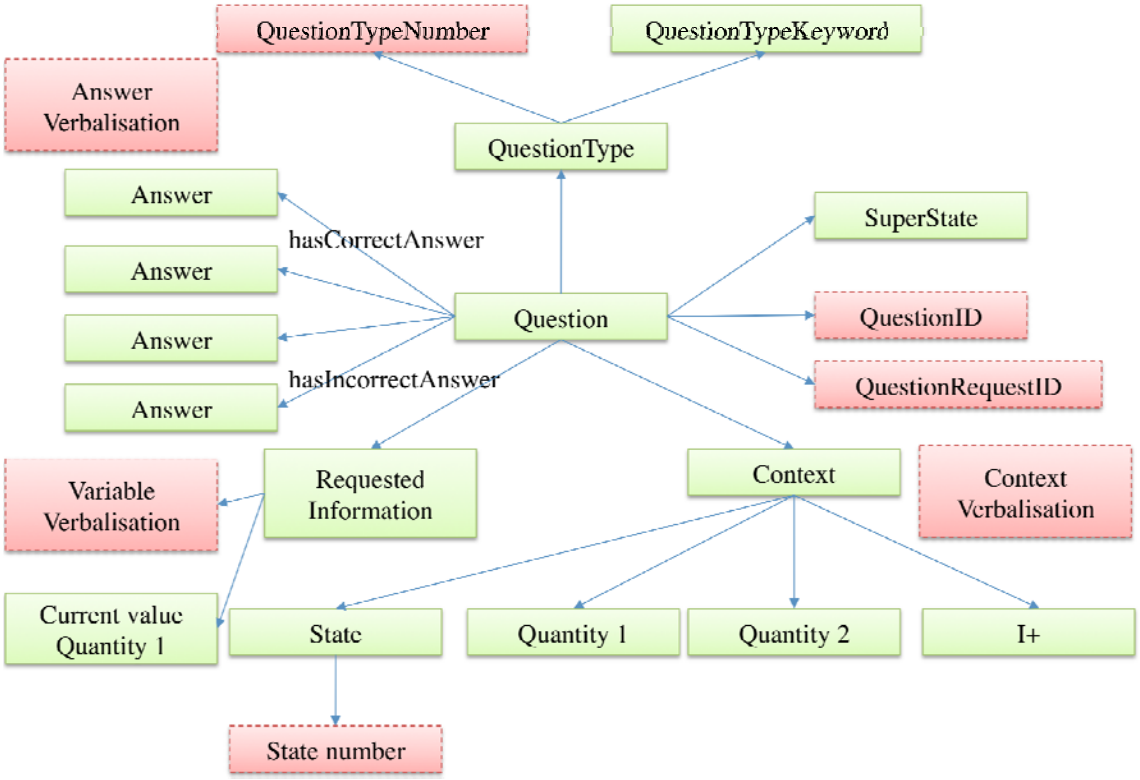

Figure 3.6: The structure of questions in OWL. The dashed boxes represent XML data types, while the lined boxes represent instances of OWL classes.

4. Use Case: Multiple-Choice Quiz

In the DynaLearn multiple-choice use case (D2.1: Bredeweg et al., 2009), the quizmaster character presents the learner with an automatically generated multiple choice based on the model simulation. The objective of this quiz can be to improve understanding of the workings of the model or to assess the level of understanding of the learner.

A Multiple Choice quiz consists of a series of items each containing a question (the stem), a correct answer (the key) and some incorrect answers (distractors). Automatic generation of distractor items is previously done using natural language processing techniques (Mitkov & Ha, 2003; Mitkov, Ha, Varga & Rello, 2009), and using lexically structured databases such as WordNet (Brown, Frischkoff & Eskenazi, 2005). The DynaLearn system is based however on the structural and semantic information contained in the model and its ingredients. This approach has the advantage of being able to use precise information relevant to conceptual modelling and it can only be used with a finite set of predefined question types.

The quality of multiple choice items is an important topic in education since poor quality items may give away the correct answer or frustrate the learner. Boland, Lester and Williams (2010) distinguish several important aspects. Firstly the item should test important material and be of an appropriate level of difficulty. Secondly, stems should be focused, contain most of the information, indicate a single correct answer and be positively phrased. Finally, distractors should be concise, independent, unambiguous and similar in content, length and grammar. These guidelines have been followed as much as possible during the implementation of the automatic distractor generation procedures for each question type.

Scope setting in the quiz use case is done by the Interaction Manager of the Virtual Character module (D5.2: Wißner et al., 2010). The user model in this component is determines the most poorly known model ingredients and these are 'selected' by setting subject quantities and concept constraints (see section 2.3 and appendix A). The next section will discuss distractor generation techniques for each question type.

4.1. Multiple Choice Distractor Generation

The generation of distractors for the multiple choice items is based on the structure and semantics provided by the correct answer and in the rest of the model and its ingredients. Each question type has a different procedure to generate distractors but some features of distractor generation are general. Firstly the amount of distractors is set to three by default, but this number can be changed in the configurations file. Also some question types generate larger or smaller numbers of distractors given the complexity of the model. For instance there are only two possible values for a quantity with a {*zero, plus*} quantity space and if this is the only quantity space in the model then more than two distractors (including unknown) cannot be generated for a questions concerning its value. In this case there are too little distractors the item will contain less than the set amount of options. Secondly since for some question types more distractors than needed are generated, a procedure is in place to randomly select distractors. Thirdly all choices (the key and the distractors) are always placed in

random order. These last two measures ensure that students will not be able to find patterns in distractor sets that allow them to choose the correct alternative without considering content.

The next section will discuss the specific generation of distractors per question type.

4.1.1. Question Type: Give Value

The *GiveValue* question results in a question of the form: *What is the value of Amount?* A possible correct answer is: *The value of Amount is plus.*

Distractors are generated by substituting all other values of the quantity space for the correct value. A future improvement may also include the substitution of values from other quantity spaces to increase the range of the distractor generation function. This use of such 'foreign' values should be limited however because they may be too obviously incorrect. Because of the required plausibility of distractors it was decided not to use values outside the quantity space associated with the quantity in this question type.

4.1.2. Question Type: Explain Effect Of Change

The *ExplainEffectOfChange* question results in a question of the form: *Amount changed from state S to this state; namely its derivative rose from min to zero, what is the effect of this change on Height?* A possible correct answer is: *Height's derivative rose from min to zero.*

Distractors are generated in two ways. Firstly this is done by constructing fake derivative changes. An increasing derivative might be said to move up from *min* to *plus*, thus making an inconsistent jump. The same is possible for a decreasing derivative. Also a change in the opposite direction may be generated or a change with the correct direction but an incorrect interpretation: *increased* instead of *decreased* and vice versa. Secondly fake but plausible changes for the value are generated based on the quantity space of the quantity.

4.1.3. Question Type: Predict Value

The *PredictValue* question results in a question of the form: *What will be the value of Pressure in the next state?* A possible correct answer is: *Pressure's value will rise from plus to max from this state to state S.* Distractors are generated by constructing plausible changes for the value based on the quantity space of the quantity.

4.1.4. Question Type: Describe Behaviour

The *DescribeBehaviour* question results in a question of the form: *What is going to happen with Amount during simulation?* A possible correct answer is: *Amount will increase and reach maximum value and then stay steady ending in state S via states S1 and S2.* Distractors are generated by changing the order of changes in the correct behaviour paths and removing changes from behaviour paths.

4.1.5. Question Type: Explain Cause

The *ExplainCause* question results in a question of the form: *Why does Amount decrease?* A possible correct answer is: *Flow (Hole) is positive and Flow (Hole) has a negative influence on Amount.*

Distractors are generated in two ways. Firstly if other influences are also active but not effective they may be presented as an explanation although their effect is in fact submissive in the state because of it being smaller or because of ambiguity. Secondly the correct answer may be changed by substituting different causal relations for the correct relation and or substituting different values or derivatives for the source of change.

4.1.6. Question Type: Effective Yes No

The *EffectiveYesNo* question results in a question of the form: *Is the influence from Flow (Tap) on Amount effective?* A possible correct answer is: *Yes, because Flow (Tap) is positive, Amount increases.*

Distractors are generated in two ways. Firstly this is done by substituting fake statuses using the set of effective, submissive and balanced causal relation in the model. Secondly this is done by constructing fake explanatory information by substituting incorrect causal relations and or substituting different values or derivatives for the source of change.

4.1.7. Question Type: Explain Submission

The *ExplainSubmission* question results in a question of the form: *Why does Amount decrease, although Flow (Tap) is positive and Flow (Tap) has a positive influence on Amount?* A possible correct answer is: *Flow (Hole) is positive and Flow (Hole) has a negative influence on Amount and Flow (Hole) is greater than Flow (Tap).*

Distractors are generated in four ways. Firstly this is done by reversing the explanatory inequality. Secondly the explanatory value may be substituted with the quantity's derivative or vice versa. Thirdly the causal relation may be substituted with a different one. Finally distractors are generated by searching for a propagating relation after the causal relation of interest. This relation is then used to incorrectly explain the submissive pattern.

4.1.8. Question Type: Explain Correspondence

The *ExplainCorrespondence* question results in a question of the form: *Why does Height have value min?* A possible correct answer is: *Amount has value min and there is a quantity correspondence between Amount and Height.* Distractors are generated by substituting the explanatory quantity and value with another quantity and its value from the simulation.

4.1.9. Question Type: Explain Calculation

The *ExplainCalculation* question results in a question of the form[6]: *Why does Flow have value min?* A possible correct answer is: *Flow is calculated by Pressure(Left) minus Pressure(Right) and Pressure(Left) is smaller than Pressure(Right).*

Distractors are generated in two ways. Firstly the explanatory inequality may be reversed. Secondly instead of a calculation a fake correspondence or simple binary equality may be proposed to explain the target quantity's value.

4.1.10. Question Type: Enumerate Influences

The *EnumerateInfluences* question results in a question of the form: *Which quantities have a direct influence on Amount?* A possible correct answer is: *Flow(Tap) and Flow(Hole).*

Distractors are generated in two ways. Firstly the set of influences on a quantity may be changed by removing items rendering it incomplete. Secondly alternate influences may be collected and inserted in the set. This is done by searching for influences away from the target quantity to other quantities. A combination of both these approaches is also used.

4.1.11. Question Type: Explain Causal Chain

The *ExplainCausalChain* question results in a question of the form: *How does Flow (Tap) influence Pressure?* A possible correct answer is: *Flow(Tap) has a positive influence on Amount, which propagates its change to Height, which propagates its change to Pressure.*

Distractors are generated in three ways. Firstly the causal path can be reversed. Secondly influences can be changed into proportionalities and vice versa in the causal path. Thirdly positive causal relations can be changed into negative relations and vice versa. These three methods yield several incorrect explanations of the causal chain.

[6] Example question taken from a different model that includes calculation: Communicating Vessels. www.garp3.org.

4.1.12. Question Type: Interpret Inequality

The *InterpretInequality* question results in a question of the form: *Which quantity is greater, Flow (Tap) or Flow (Hole)?* A possible correct answer is: *Flow(Hole) is bigger.* Distractors are generated by proposing different inequalities from the set: bigger, smaller, equal, unknown.

4.1.13. Question Type: Interpret Causal Chain

The *InterpretCausalChain* question results in a question of the form: *Does Amount influence Flow (Hole)?* A possible correct answer is: *Yes.* Since this a pure yes/no question, the only possible distractor is the opposite answer.

4.1.14. Question Type: Summarize Aggregate Causal Chain

The *SummarizeAggregateCausalChain* question results in a question of the form: *Describe the influence of Flow (Hole) on Pressure.* A possible correct answer is: *Flow (Hole) has a negative influence on Pressure.* Distractors are generated by generating alternative aggregate relations with different signs, different causal relation types (influence / proportionality) or multiple paths and ambiguous aggregate results.

4.1.15. Question Type: Describe Alternative Path

The *DescribeAlternativePath* question results in a question of the form: *Amount changed from the state S1 to this state: its value decreased from plus to zero, what else could have happened?* A possible correct answer is: *Amount could have stayed plus going to state S2.*

Distractors are generated by modifying the set of changes in the correct answer with plausible other changes. For these generated incorrect changes the values of the quantity space of the quantity are used as well as the actual follow up states from the state of interest. Each element of the set of changes in the correct answer may or may not be changed thereby producing a complete set of alternative paths. To ensure plausibility of these paths these are checked for consistency, states may not occur twice in the same path.

4.1.16. Question Type: Interpret Loop

The *InterpretLoop* question results in a question of the form[7]: *We saw that 'QuantityA' influences 'QuantityB' and vice versa. Does 'QuantityA' influence itself via 'QuantityB'?* A possible correct answer

[7] This pattern is quite rare in example models and therefore a generic phrasing is used in the example question and answer.

is: *No, the end of the path from 'QuantityA'to 'QuantityB' can't be joined to the start of the path from 'QuantityB' to 'QuantityA'.* Since this is in fact a pure yes/no question, the only possible distractor is the opposite answer.

4.1.17. Question Type: Enumerate Quantity Space

The *EnumerateQuantitySpace* question results in a question of the form: *Which values can Pressure adopt?* A possible correct answer is: *point (zero), plus and point(max).*

This question type effectively asks for an enumeration of the values of a quantity space and distractors are generated by modifying this set of values. This is done in two ways. If there are no other quantity spaces present in the model the only possible mutations of the value set are deletions. If there are other quantity spaces in the model, then the values of these quantity spaces are used to mutate the correct value set. In this case no deletions are done, only substitutions. Furthermore all distractor sets have the same amount of their elements substituted, roughly 20%, thereby ensuring some homogeneity amongst the distractor sets.

4.1.18. Question Type: Predict Possible Values

The *PredictPossibleValues* question results in a question of the form: *What can possibly be the value of Pressure in the next state?* A possible correct answer is: *Pressure can decrease from plus to zero, going to state(s) S1, or stay plus going to state(s) S2.*

Distractors are generated for this question type by modifying the set of changes in the correct answer with plausible other changes. For these generated incorrect changes the values of the quantity space of the quantity are used as well as the actual follow up states from the state of interest. Each element of the set of changes in the correct answer may or may not be changed thereby producing a complete set of alternative paths. To ensure plausibility of these paths these are checked for consistency, states may not occur twice in the same path. Apart from having only one-step paths this procedure is similar to that of question type: *DescribeAlternativePath.*

4.1.19. Question Type: Predict Which Quantity Changes

The *PredictWhichQuantityChanges* question results in a question of the form: *Which quantity will be changed in the next state: Flow (Hole) or Amount?* A possible correct answer is: *Both, going to state(s) S1 and S2.*

Distractors are generated for this question type in two ways. Firstly the set of successor states in which the changes occur can be modified. This is done by deleting elements in it. Secondly the set of changing quantities may be modified: Instead of both quantities changing, only one may be said to change or even none. Similarly if the correct answer is one of the latter options any other option may be substituted.

4.1.20. Question Type: Describe Causal

The *DescribeCausal* question results in a question of the form: *Describe the influence of Height on Pressure.* A possible correct answer is: *Changes in Height are followed by changes in Pressure.*

Distractors are generated for this question type by substituting a different causal relation. This may involve changing positive causal relations into negative relations and vice versa and or changing influences into proportionalities or vice versa.

4.1.21. Question Type: Describe Correspondence

The *DescribeCorrespondence* question results in a question of the form: *What kind of connection is there between the values of Amount and Height?* A possible correct answer is: *All their values correspond, they will always have the same value.*

Distractors for this question type are generated in two ways. Firstly this is done by changing the correspondence type of the correct answer. There are four types of correspondences: undirected quantity correspondences, directed quantity correspondences, undirected value correspondences and directed value correspondences. These can all be used and in case of substituting a value correspondence plausible corresponding values are selected. Secondly another type of relation can be implied and this is done by either proposing that the quantities are in fact not related or by proposing that they are both influenced by the same third quantity.

4.1.22. Question Type: Explain Derivative Equality

The *ExplainDerivativeEquality* question results in a question of the form[8]: *Why does 'QuantityA' decrease?* A possible correct answer is: *Because the derivative of 'QuantityA' is equal to the derivative of 'QuantityB'.*

Distractors for this question type are generated in two ways. Firstly this is done by changing the explanatory relation to a derivative correspondence or positive proportionality, which both are plausible candidates for producing this behaviour. Note that this question is only presented if no such relations are in fact present in the model. Secondly distractors are created by selecting a random third quantity and proposing the same three relations for this quantity: derivative equality, derivative correspondence and positive proportionality.

4.1.23. Question Type: Direct Influence Yes No

[8] This pattern is quite rare in example models and therefore a generic phrasing is used in the example question and answer.

The *DirectInfluenceYesNo* question results in a question of the form: *Does Amount influence Pressure directly?* A possible correct answer is: *Yes.* Since this a pure yes/no question, the only possible distractor is the opposite answer.

5. Use Case: Teachable Agent Ask, Explain & Challenge

In the DynaLearn teachable agent use case the learner is given an assignment to build a model on a specific subject, whereby the model constructed represents the knowledge of a teachable agent character (D2.1: Bredeweg et al., 2009). The learner is in fact teaching the virtual character and thereby this use case is an application of the 'learning by teaching' approach to education. This paradigm is based on the finding that teaching may produce strong learning effects (Bargh & Schul, 1980; Roscoe & Chi, 2007). Previous work on teachable agents has shown that the personification by the virtual character increases motivation and may provide a socially acceptable way of attributing suboptimal performance partly to the agent thereby protecting the self-image (Biswas et al., 2001; Chase, Chin, Oppezzo & Schwartz, 2009). The DynaLearn teachable agent use case aims to profit from these benefits by providing personification via the virtual character, scaffolding of the teaching-learning process and also a sense of competition by allowing students and their teachable agents to be virtually challenged resulting in a grade. For an added sense of competition this challenge could be taken as a public quiz in front of the whole class.

The teachable agent use case is implemented on DynaLearn learning space two, basic causal model (D2.1: Bredeweg et al., 2009) and has the structure outlined in figure 5.1. Firstly the course facilitator or teacher should supply the learner or student with an assignment containing subject material or directions on how to obtain this. Also an expert model must be supplied. This model is flagged as a teachable agent expert model and loading it from a file or from the model repository will cause DynaLearn to enter teachable agent mode. In this mode the expert model is not shown to the student but all names of quantities and entities are loaded in a blank student model. Supplying the learner with these terms allows the student model and expert model to be compared later on.

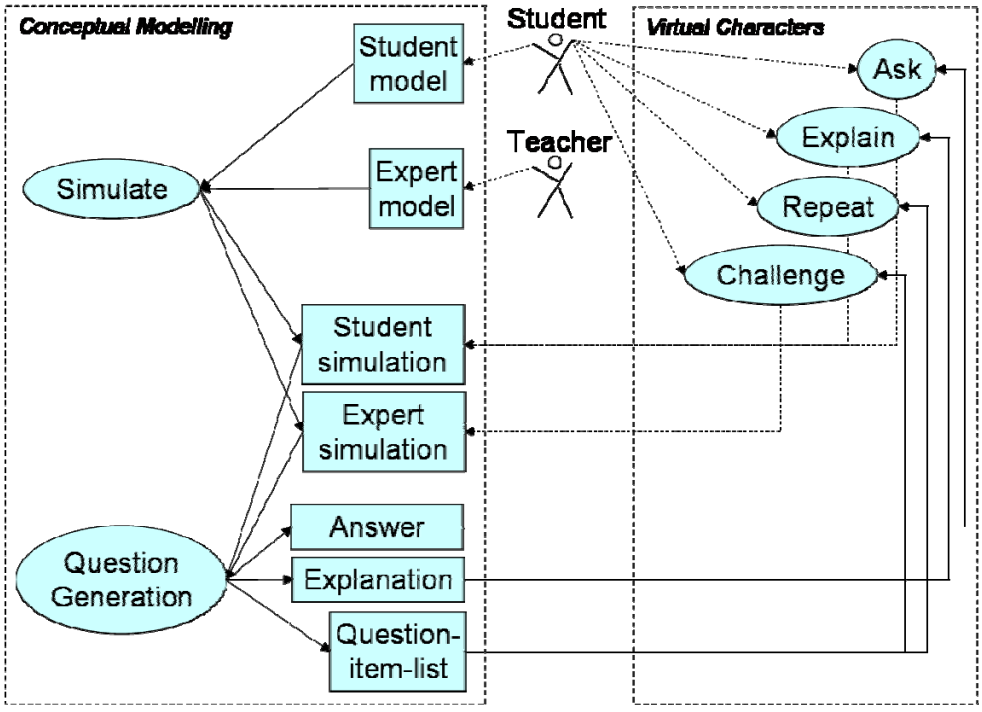

Figure 5.1: Teachable Agent Overview. The student builds a student model about which questions can be asked to the teachable agent. Explanations of answers can also be based on the student

simulation. The teachable agent's knowledge can be challenged by a question set generated from the expert simulation and answering based on the student simulation.

Secondly the student teaches his virtual character by building a model of the subject matter. The teachable agent can reason about this knowledge by simulating the model and the question generator is used to answer questions about the subject composed by the student. These questions asked 'test' if the teachable agent has understood the material and thereby if the model behaves as the student expects it to. More detailed explanations of the reasoning behind given answers can also be generated. Finally the student and his/her teachable agent can be tested by sending the teachable agent to a challenge. In this challenge the teachable agent is quizzed by the quizmaster using questions generated from the expert model and simulation. Answers are generated from the student model and simulation. The challenge can be taken in a group setting and the resulting grade is an indication of how well the student tutored his teachable agent. Furthermore incorrectly answered questions may cue the learner to possibilities for improvement. By all these means the teachable agent use case is expected to improve learner motivation and provide scaffolding in modeling, model understanding and advancement in the direction of an expert model.

The role of the question generator is central in ask, explain and challenge interactions. The next subsections will discuss the implementation of each of these cases respectively.

5.1. The 'Ask' Interaction

The Ask interaction lets the student ask a question to the teachable agent of the form: "If *Amount increases,* then what happens to *Pressure*?" Figure 5.2 provides an overview of the interaction which starts when the student clicks on the ask button floating above the teachable agent character. Source quantity (*Amount* in this example), direction of change (*increases)* and target quantity (*Pressure*) are then selected from drop-down lists in a question composition dialog.

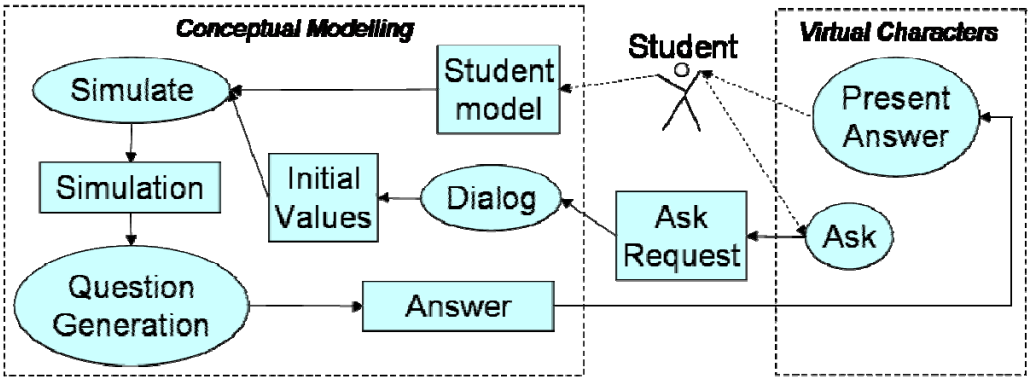

Figure 5.2: Ask Interaction Overview: The student starts the Ask interaction after which a question composition dialog is completed. This triggers a simulation and question generation inference that supplies the answer to the question which is communicated by the virtual character.

The information from this dialog is used to run a small simulation in the background. In this simulation the causal model is stripped from all irrelevant relations because these could provide uninitialized causal paths. Figure 5.3 provides two examples in a hypothetical causal graph of such irrelevant

relations. In this example source quantity A has a known behaviour selected by the student (increasing or decreasing). The effect on target quantity B is of importance but it can only be calculated if all causal influences on it are known. Therefore quantity Y should be initialized to zero. Quantity X however, cannot be initialized to zero because this would contradict the behaviour of A. The solution that avoids unnecessary computation of initial values is therefore to only include causal relations on paths from source to target quantity.

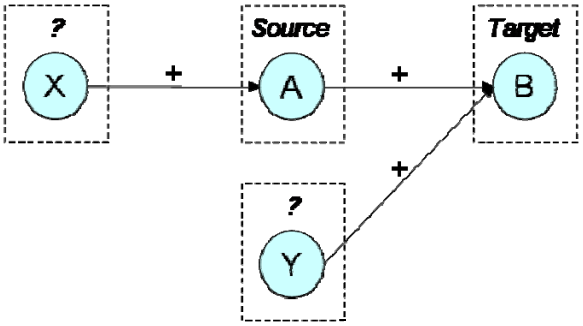

Figure 5.3: Hypothetical causal graph. Source quantity A has known behaviour, therefore quantity X should be left uninitialized not to cause contradictions. Quantity Y should however be initialized to zero or be removed to be able to calculate the effect of A on target quantity B.

The resulting simulation is used to generate a question of type 14 using different conditions however in terms of heuristics guiding its construction; also paths of one step are used. Furthermore a different verbalisation of the answer is used to match the learner question format. The answer of this question is communicated to the learner via the teachable agent. Figure 5.4 shows an example of an ask interaction. The learner can use the repeat function to hear the answer again or use the explain function described in the next section to hear about the reasoning that led to the answer.

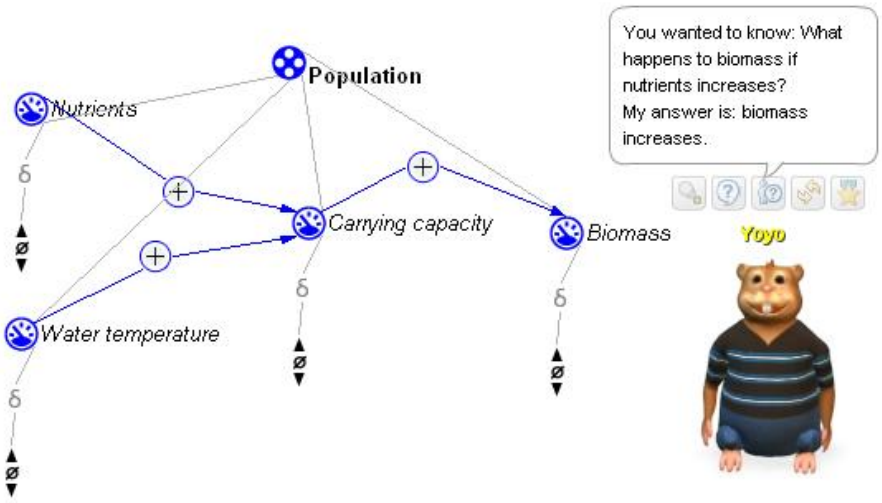

Figure 5.4: Teachable Agent Ask Interaction Answer. After the learner has composed the question the teachable agent replies.

5.2. The 'Explain' Interaction

The Explain interaction provides a step by step walkthrough of the reasoning done to calculate the answer to a question from the Ask interaction. This interaction can therefore only take place after an Ask interaction was successfully completed. Figure 5.5 provides an overview of the interaction which starts when the learner clicks on the explain button floating above the teachable agent character. The simulation is used to generate questions of type 11 and 20 for every causal path from source to target quantity. The answer(s) to these questions together with the (aggregate) answer from the ask interaction are used in the question explanation construction inference.

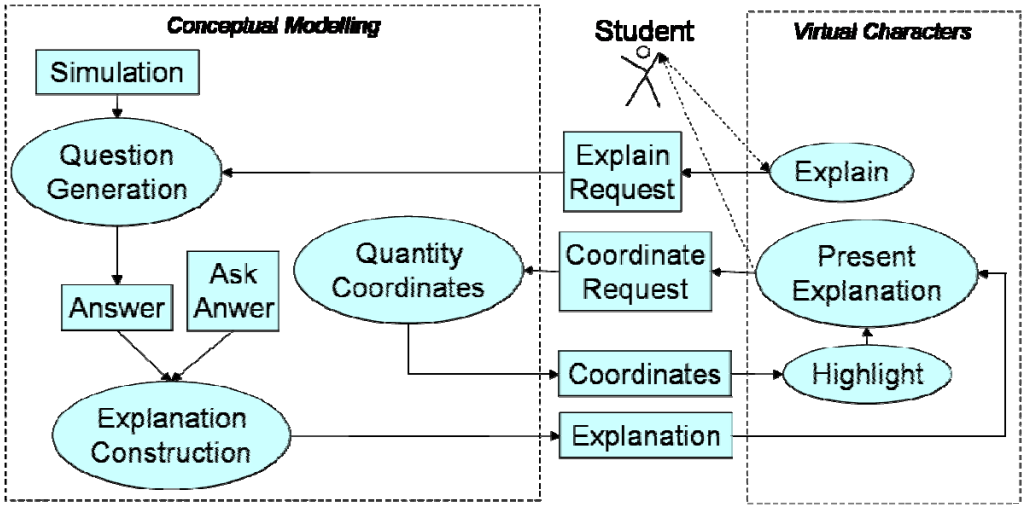

Figure 5.5: Explain Interaction Overview. The student initializes the interaction. The Answer and simulation from the Ask Interaction are used and from all information an explanation is constructed involving all causal paths. The explanation is presented by the teachable agent highlighting all quantities as they are mentioned.

The generated explanation enumerates all causal links and their effects before summarizing them. In case of multiple causal paths this inference groups all paths with positive outcomes and those with negative outcomes together and these groups are then contrasted to explain why an ambiguous outcome is observed. The explanation is presented by the teachable agent and during this explanation each mentioned quantity is highlighted by requesting its coordinates and then projecting a star or laser beam to effectively implement a 'pointing' behaviour from the virtual character. To achieve this, elements that need to be highlighted are marked in the explanation. Upon encountering these marks the teachable agent requests the element's coordinates needed.

The explanation can be requested again using the repeat function. The last interaction of this use case is the challenge which is discussed in the next section.

5.3. The 'Challenge' Interaction

In the Challenge interaction the knowledge given to the teachable agent by the student is tested in a quiz composed of questions based on an expert model. Figure 5.6 gives an overview of this interaction which starts when the student clicks on the challenge button above the teachable agent character. Both the expert model supplied by the teacher and the student model are simulated in the background. The expert model of course remains invisible to the student at all times. The causal models are extracted from these simulations and compared yielding an objective score of how well the student model matches the expert model. This objective score and information on the causal models is used to select questions generated from the expert simulation that will be answered correctly or incorrectly in a similar ratio as the objective score. Answers are generated from the student model. It is essential for the student model to use the same terms as the expert model to allow matching of expert model questions and student model answers. As mentioned before this is ensured by supplying the student with all terms in the expert model when the teachable agent use case is started.

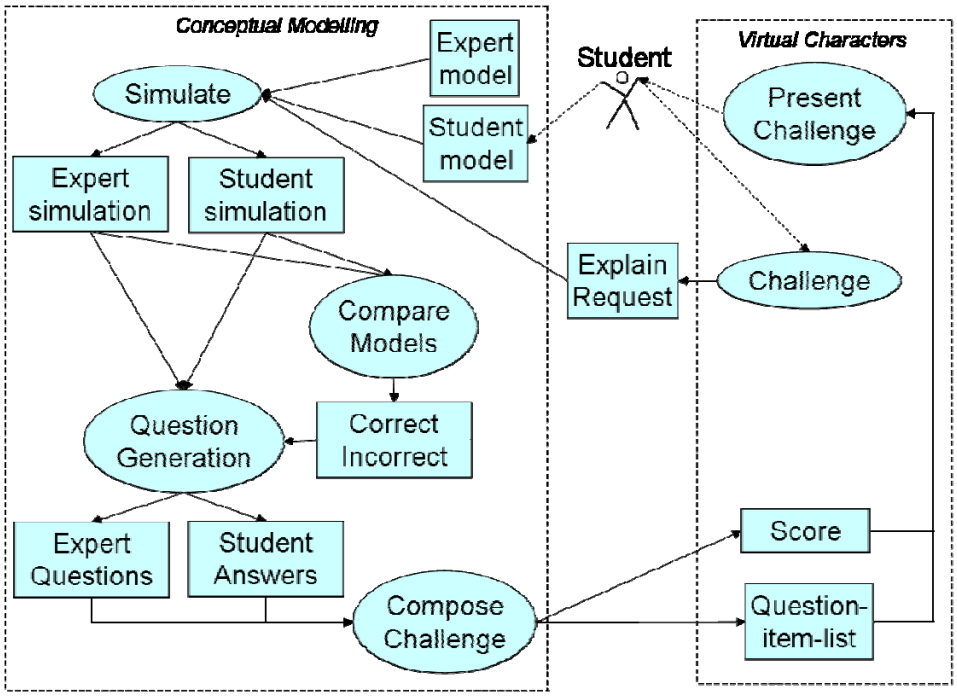

Figure 5.6: Challenge Interaction Overview. The challenge is initiated by the student. Expert and Student models and simulations are compared and questions are generated and selected according to the correct/incorrect ratio. The resulting challenge consists of expert model questions and student model answers and is presented by the quizmaster to the teachable agent.

The resulting challenge is presented to the student whereby the quizmaster poses the questions to the teachable agent who answers them. The challenge results in a score and the student may review all results by clicking on a button floating above the quizmaster character.

The amount of questions can be set by the teacher in the configuration file of the software. The next sections will discus the model comparison algorithm and question selection algorithm in more detail.

5.3.1. Causal Model Comparison And Challenge Question Selection

The challenge is composed of expert model questions and student model answers. Items are about relations between quantities and can be viewed as belonging to one of five categories, because for every relation there are five possibilities:

- The relation is in the expert model and in student model: Correct.
- The relation is in the expert model but not in student model: Missed.
- The relation is in the student model but not in expert model: Over-complete.
- The relation is in the student model but not in expert model, yet it is a correct shortcut of a longer path in the norm model: Aggregate-correct.
- The relation is not in the student model and not in expert model: Trick Question.

The model comparison algorithm first takes every causal[9] relation in the expert model and determines if it is present in the student model thus yielding the set of correct relations and the set of missed relations. Then the algorithm takes every causal relation in the student model and determines if it is present in the expert model. This yields the set of over-complete relations and aggregate-correct relations. The set of trick question relations is not determined because it was decided not to pose questions of this category. There are several reasons for this choice. Firstly the student does not gain any insight from observing his teachable agent correctly answering that some two random quantities are unrelated. Secondly the relation may in fact be relevant in the real world but not modelled in the expert model because of simplifying assumptions or a choice in modelling perspective. This last argument also holds for the over-complete relations category but in this case the relation is surely relevant since it was in fact modelled by the student.

Having these sets of relations in each category an objective score for the student model can be determined based on the amount of relations in each set. The measure used is the following:

Score = Correct + (Aggregate-correct / 2) / (Correct + Missed + Over-complete + Aggregate-correct)

Note that Aggregate-correct relations are counted as 50% correct and only a model with only correct relations will therefore achieve a 100% score. A model with all correct relations but also some erroneous extra relations will however not achieve a 100% score. Therefore the possible student strategy of putting any relation in the model that seems plausible will not result in a good score.

The sets of relations in each category are input for the challenge question selection algorithm. This algorithm first determines the amount of questions in each category given the total amount of items that the challenge needs to have. The distribution of questions over categories is done such that is matches the hit-miss-over-complete ratio of the model as well as possible. The second step is actually selecting questions and this is primarily done on a random basis from the pool of questions in each category. There are however four question types used in the challenge: single relation behavioural questions (type 20), aggregate relation behavioural questions (type 14), single relation yes/no questions (type 23) and aggregate relation yes/no questions (type 13). The ratio at which each of these types is used in each pool can be set in the configuration file. By default the ratio of behavioral questions is 75% and of the yes no questions, direct relation questions take up 66%. For Aggregate-correct relations the system will try to select both a correctly answered question (type 14) and an incorrectly answered question (type 23). Using this overall system of question distribution calculation it is assured that the challenge score corresponds to the student's objective score as closely as possible for challenges of any number of questions.[10]

[9] Note that this comparison algorithm can be extended to include any other modelling ingredient beyond causal relations.

[10] Note that the objective score remains hidden.

The challenge score and the objective score may diverge when the challenge has very few questions. In this case the scale of the student score is effectively very coarse and may not match the objective score very well. A last point of consideration is that when the challenge is very long compared to the size of the model, the amount of questions generated may not be sufficient to fill the challenge and in this case the challenge is automatically shortened.

6. Conclusion

The question generator component in the DynaLearn system provides input for several functions in the use cases ‘quiz’ and ‘teachable agent’. It has a set of 23 question types covering a wide range of qualitative reasoning phenomena. These question types are used flexibly; both producing straight forward question answer sets, as well as producing elaborate explanations of behaviour by combination and analysis of output. A formal OWL output format is used that allows flexible verbalisation into a natural language of choice.

The question generator can be driven towards certain questions or it can apply heuristics to produce a small diverse set of questions that are highly relevant to the simulation at hand. Multiple choice quizzing is supported with the automatic generation of high quality distractors for each question type enabling learning by questioning or assessment test approaches. Modelling can be supported by generating answers to user composed questions as well as accompanying explanations. Question generation can also be applied to model assessment with generation of a quiz effectively comparing a reference model and student model and providing a fair score according to student model quality.

7. Discussion

This document presented the question generator component of the DynaLearn system. The multiple choice quiz is expected to require the learner to thoroughly inspect the model thereby improving understanding of the subject matter. Also it is expected that the quiz may prove valuable in assessing learner understanding of a model. These expectations may be confirmed by evaluations in the context of WP7. The same holds for the content provided to the teachable agent.

Another question concerns the approach of question generation taken with respect to the model and simulation. Currently the question generator mostly uses the simulation as its knowledge base and this imposes a concrete perspective on the modelled knowledge. General behavioural patterns are specifically instantiated in the simulation, for example in case of ambiguous influences multiple states are generated in which the ambiguity behind the behaviour is implicit. The question generator currently deals with such patterns spanning the simulation by running specifically trimmed simulations and combining and analysing the output of several question types. Future work may research the possibility of generation of questions about more of these high level concepts in model behaviour based on analysis of the model fragment library. A difficulty in this alternative approach may be that the ultimate role that model fragments play is only known when they are applied in the simulation.

Future work may also involve the extension of the teachable agent use case to learning spaces 3 and 4 thereby including more modelling ingredients into the workspace that need to be addressed by the teachable agent. The inclusion of more modelling ingredients in this use case is expected to be relatively easy with respect to the question generator component's role of supplying content information since there are already question types present concerning these modelling ingredients.

Another option for future research is introducing more interactivity with the learner in the teachable agent challenge. This may be done when the teachable agent provides an incorrect answer in response to a question of the quizmaster. In this situation the learner might be given a multiple choice response option where by the key is of course based on the underlying expert model and the distractors based on the expert model as well as the student model. This way the learner's true understanding of the subject matter may be assessed. This information might be stored in the user model providing input for future diagnosis of modelling efforts.

References

Aldabe, I., Lopez de Lacalle, M., Maritxalar, M., Martinez, E., & Uria, L. (2006). ArikIturri: An Automatic Question Generator Based on Corpora and NLP Techniques*, Intelligent Tutoring Systems, 4053,* p.584-594.

Bargh, J. A., & Schul, Y. (1980). On the Cognitive Benefits of Teaching. *Journal of Educational Psychology, 72(6),* p.583-604.

Biswas, G., Schwartz, D., Bransford, J., & the Teachable Agents Group at Vanderbilt (TAG-V). (2001). Technology support for complex problem solving: From SAD environments to AI. *Smart Machines in Education,* p.71-98.

Bredeweg, B. (ed.), André, E., Bee, N., Bühling, R., Gómez‐Pérez, J.M., Häring, M., Liem, J., Linnebank, F., Thanh Tu Nguyen, B., Trna, M., & Wißner, M. (2009). Technical design and architecture, DynaLearn, EC FP7, STREP Project no. 231526, Deliverable D2.1.

Boland, R. J., Lester, N. A., & Williams, E. (2010) Writing Multiple-Choice Questions, *Academic Psychiatry, 43,* p.310-316.

Brown, J. C., Frischkoff, G. A., & Eskenazi, M. (2005). Automatic Question Generation for Vocabulary Assessment. *Proceedings of Human Language Technology conference and Conference on Empirical Methods in Natural Language Processing,* p.819-826.

Chase, C. C., Chin, D. B., Oppezzo, M. A., Schwartz, D. L. (2009). Teachable agents and the protégé effect: Increasing the effort towards learning. *Journal of Science Education and Technology, 18(4)*, p.334-352.

Goddijn, F. (2002). Quags - Automatische Vraaggeneratie bij Kwalitatieve Simulaties, master thesis, University of Amsterdam.

Goddijn, F., Bouwer A., & Bredeweg, B. (2003). Automatically Generating Tutoring Questions for Qualitative Simulations. *Proceedings of the 17th International workshop on Qualitative Reasoning, QR'03*, p.87-94.

van Heijst, G., Falasconi, S., Abu-Hanna, A., Schreiber, G. and Stefanelli, M. A case study in ontology library contruction. Artificial Intelligence in Medicine, 7(3):227–255, June 1995.

Liem, J., Beek, W., Linnebank, F., & Bredeweg, B. (2010) API for data and knowledge exchange, EC FP7, STREP Project no. 231526, Deliverable D3.2.

Mitkov, R., & Ha, L. A. (2003). Computer-Aided Generation of Multiple-Choice Tests. *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing,* p.17-22.

Mitkov, R., Ha, L. A., Varga, A., & Rello, L. (2009). Semantic similarity of distractors in multiple-choice tests: extrinsic evaluation. *Proceedings of the EACL 2009 Workshop on GEMS: GEomerical Models of Natural Language Semantics,* p.49-56.

Myller, N. (2007). Automatic Generation of Prediction Questions during Program Visualization, *Electronic Notes in Theoretical Computer Science, 178,* p.43 - 49.

Roscoe, R.D., and M.T.H. Chi, (2007). Understanding Tutor Learning: Knowledge-Building and Knowledge-Telling in Peer Tutors' Explanations and Questions, *Review of Educational Research, 77(4)*, p.534-574.

Wang, W., Hao, T., & Liu, W., (2008) *Automatic Question Generation for Learning Evaluation in Medicine*, Advances in Web Based Learning – ICWL 2007, vol 4823/2008, p.242-251.

Wißner, M., Häring, M., Mehlmann, G., Bühling, R., Milosevic, U., Bredeweg, B., & André, E. (2010) Basic Tutorial Tactics for Virtual Agents, EC FP7, STREP Project no. 231526, Deliverable D5.2.

Appendix A: Question Type Constraints

Table A.1: Question Type Constraints

Question-Type	P	C	B	S	A	Covers	Keyword
1	cm, sr	vl	re	pr	rp	Values	giveValue
2	sr	cr	Re	ot	es		explainEffectOfChange
3	sr	vl	Re	ot	rp		predictValue
4	sr	x	re	ot	ep		describeBehaviour
5	cm, sr	cr / dr	re	pr	es	▲ ▼ I+ I- P+ P-	explainCause
6	cm	cr	al	pr	es	I+ I- P+ P-	effectiveYesNo
7	cm	cr / ie	su	pr	Ep	I+ I- P+ P- < >	explainSubmission
8	cm	cp	re	pr	es	values	explainCorrespondence
9	cm	ca	re	pr	es	values	explainCalculation
10	cm	cr	al	pr	ss	I+ I- P+ P-	enumerateInfluences
11	cm	cr / ca	re	pr	sp	chains: I+ I- P+ P-	explainCausalChain
12	cm	ie	re	pr	rp	< > = ≤ ≥	interpretInequality
13	cm	cr	al	pr	sp	chains: I+ I- P+ P-	interpretCausalChain
14	cm	cr	al	pr	Ep	I+ I- P+ P-	summarizeAggregateCausalChain
15	sr	x	Re	ot	rp		describeAlternativePath
16	cm	cr	al	pr	sp	chains: I+ I- P+ P-	interpretLoop
17	cm, sr	vl	al	pr	rp	quantity spaces	enumerateQuantitySpace
18	sr	vl	Re	ot	rp		predictPossibleValues
19	sr	x	Re	ot	rp		predictWhichQuantityChanges
20	cm	cr	al	pr	rp	I+ I- P+ P-	describeCausal
21	cm	cp	al	pr	rp	Q Q^ V V^	describeCorrespondenceConnection
22	cm, sr	dr	re	pr	es	d=	explainDerivativeEquality
23	cm	cr	al	pr	rp	I+ I- P+ P-	directInfluenceYesNo

Table A.2: Constraint Types

P		Perspective (P): What kind of information will be used
	sr	Simulation run-through (sr) - *changing values*
	cm	Causal model (cm) – *reasons of change*
C		Concept (C): Domain-independent subject
	cr	Causal relation (cr)
	ie	Inequality (ie)
	cp	Correspondence (cp)
	vl	Value (vl)
	dr	Derivative (dr)
	ca	Calculation (ca)
B		Behaviour (B): Amount of contribution to the behaviour of the system
	re	Real (re) – *representing the behaviour*
	al	All (al) – *makes no distinction on this subject*
	su	Submissive (su) – *dominated by other effects, therefore not contributing*
S		InfoState (S): Whether the needed information is available in the present state or not
	pr	Present (pr)
	ot	Other (ot)
A		Answer method (A): Learner's strategy to find the answer, from easy to hard:
	rp	Report (rp) – *question indicates place of answer*
	ss	Search singular (ss) – *answer is to be found near the references in the question*
	sp	Search plural (sp) – *answer must be found somewhere*
	es	Explain singular (es) – *learner must reason with uniform information to answer*
	ep	Explain plural (ep) – *learner must reason with pluriform information*

e-mail: Info@DynaLearn.eu
website: www.DynaLearn.eu