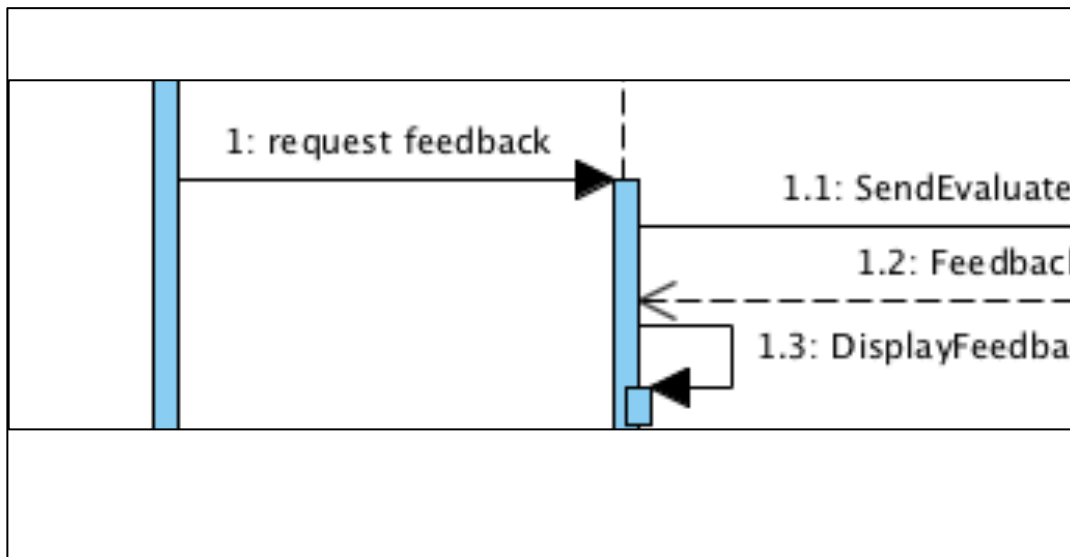




UNIVERSITY OF GOTHENBURG



## Providing Automated Feedback on Software Design for Novice Designers

*Bachelor of Science Thesis in the Programme Software Engineering and Management*

HELEN ANCKAR

University of Gothenburg  
Chalmers University of Technology  
Department of Computer Science and Engineering  
Göteborg, Sweden, June 2015

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet.

The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

### **Providing Automated Feedback on Software Design for Novice Designers**

HELEN ANCKAR, June 2015,

© HELEN ANCKAR, June 2015.

Examiner: JAN-PHILIPP STEGHÖFER

University of Gothenburg  
Chalmers University of Technology  
Department of Computer Science and Engineering  
SE-412 96 Göteborg  
Sweden  
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering  
Göteborg, Sweden June 2015

# Providing Automated Feedback on Software Design for Novice Designers

Helen Anckar

Department of Computer Science and Software Engineering  
University of Gothenburg  
Gothenburg, Sweden  
gusanche@student.gu.se

**Abstract.** *In this paper we describe the design and implementation of a pedagogical feedback agent, designed to provide on-demand feedback on software design. The implemented feedback agent was integrated with an online UML editor, and novice designers can design UML class diagrams within this environment and request formative feedback from the feedback agent. Tool evaluations were conducted with the purpose of improving the system and as way to observe how users respond to the provided feedback. Results from this research are encouraging and most of the evaluation participants found the automated feedback on design to be helpful.*

**Keywords:** Computer supported collaborative learning, Interactive Learning Environment, Pedagogical Agent, Formative Feedback, Cognitive Load.

## 1 Introduction

The Unified Modeling Language (UML) has evolved into a standard object-oriented modeling notation for designing quality conceptual models of software systems [1]. As a result of today's fast evolving technological advancements, and the standardisation and exuberant growth of UML, the variety in UML modeling tools has increased over the last decade. While these tools are equipped with a wealth of features to enable the design and development of basic to complex software systems, constructing complex UML diagrams such as class diagrams, use case diagrams and system sequence diagrams

can be quite a challenging and complex process, especially for novice software system designers [2]. These UML modeling tools normally employ functionalities that are too extensive and too complex for novice designers to use, which may consequently lead to cognitive overload in students with low prior knowledge in UML design [12, 21]. Some examples of the most common UML modeling tools currently available include Umodel<sup>1</sup> by Altova, IBM Rational Rhapsody by IBM, Papyrus<sup>2</sup>, Enterprise Architect<sup>3</sup> by Sparx Systems, Visual Paradigm<sup>4</sup>, and IntelliUML Teresa<sup>5</sup> by Beto Software.

Since UML has grown to become the de facto standard language for designing software artefacts produced by many established software development companies, software engineering and programming students at higher education institutions are expected to learn UML design well enough to effectively design higher quality software in industry. In an effort to help students learn UML design more effectively, researchers have developed a variety of Computer Supported Collaborative Learning systems for UML, for example [2, 5]. These CSCL systems employ pedagogical agents that help novice designers learn UML modeling skills by providing advice and domain-level feedback on design through hypertext, glossaries, and on collaboration [4, 12]. CSCL systems for UML are designed to support collaborative learning and are not developed with the aim of providing especially designed feedback on UML design to single-user designers. To the best of our knowledge, no feedback agent (FA) for UML has been designed to provide especially designed formative feedback to novices in online Interactive Learning Environment (ILE).

This paper presents a web based UML feedback agent that provides formative feedback on software design to novice designers based on the results from a comparative validation of the students solution against an ideal solution provided by an educator. An Interactive Learning Environment [29], is created by integrating the feedback

---

<sup>1</sup> ([http://www.altova.com/products\\_umodel.html](http://www.altova.com/products_umodel.html))

<sup>2</sup> (<https://www.eclipse.org/papyrus/>)

<sup>3</sup> (<http://www.sparxsystems.com/products/ea/>)

<sup>4</sup> (<http://www.visual-paradigm.com>)

<sup>5</sup> (<http://www.betosoftware.com/teresa/>)

agent with an already existing online UML editor called WebUML [14] that was designed and created by professor Dave Stikkolorum.

Our main objective with this research design study was to find out how an online UML modelling technology can support giving automated effective feedback to novice designers and how this feedback mechanism can best be designed and developed. More specifically:

**RQ.1** What properties should a feedback agent for an online Interactive Learning Environment for UML novice designers have?

**RQ.2** How does such an agent effectively provide effective feedback?

## 2 Related Work

This section presents areas of research concerning i) computer supported collaborative learning systems for UML that are similar to our area of research; ii) help-seeking behaviour in ILE's; and iii) pedagogical agents. We also explore the concept of feedback, and the different levels of feedback as elaborated in the feedback model by Hattie and Timperley [13].

### 2.1 Tooling

Chen and colleagues designed and developed CoLeMo [2], a distributed collaborative UML modeling environment designed to help remotely situated students learn UML software design in groups [2]. The CoLeMo system is designed to provide advice pertaining to UML designs rules, as well as advice regarding group collaboration. CoLeMo is also designed to monitor designer activities (e.g which students are active during a modeling session), and to regulate and coordinate user participation.

Nilufar Baghaei et al. [23], present COLLECT UML, another collaborative learning environment that uses a meta constraint-based model to teach object oriented analysis and design. Like CoLeMo, COLLECT UML also provides designers with domain level feedback and as well as feedback on collaboration work. COLLECT UML was first designed to support single users before it was turned into a collaboration system. This system also provides a chat window

were collaborating students can communicate with each other while working on a common task.

Tourtoglou and Virvou [5] present AUTO-COLLEAGUE, a collaborative learning environment for UML designed to help novice designers (trainees) learn UML skills through collaboration with their peers and trainers (administer, supervisor or teachers). The tool is developed around a user-modelling component where students characteristics such as performance and personality are evaluated and accounted for as a basis for providing feedback on effective collaboration.

## 2.2 Help-Seeking in ILE's

The concept of help-seeking in any social context is one that a lot of people know too well, however the process behind this concept is rather complex. According to the help-seeking model presented by Nelson-Le Gall [24], help-seeking is a five step process that involves the following stages :

*i) Became aware of the need for help:* The student must know when to seek help; understand that he or she is struggling and will need help in order to progress with the task at hand.

*ii) Decide to seek help:* The student must look at available information and decide whether to elicit help.

*iii) Identify potential helpers:* In this step, the student must identify who to elicit help from. This could be from peers, teachers or by asking a question online.

*iv) Use strategies to elicit help:* The question must be formulated in an appropriate way to match the subject matter [25].

*v) Evaluate help-seeking episode:* In the final step, the learner must evaluate the help they received and determine whether it was useful or not. If the help was not useful, the student must seek further help and this may involve the identification of a new helper.

Although this model was built around the study of help-seeking in a social context such as classrooms, it can also be applied to help-seeking behaviour presented in Interactive Learning Environments

(ILE) as the different stages that students take during a help-seeking episode are very similar in both environments [12]. The only difference is that in an ILE, a helper (e.g help feature) would have already been identified. However, one can also argue that even in an ILE, a student would still have the choice to seek help from other sources e.g an external web page that contains detailed domain-related information.

Help in ILE systems can be designed to be accessible to students either on demand (i.e at a student's request) or whenever the system deems it necessary to provide help. Researchers argue that students who receive system initiated help are primarily less likely to receive the appropriate type of help they need as the system's help functions would not be able know a student's current cognitive state or pick up on any of the students verbal cues [28, 26]. Furthermore, students are less likely to use system initiated help in knowledge construction as the help message may have been presented at a time when they are least receptive to new information [27].

### 2.3 Providing Feedback

Hattie and Timperley conceptualise feedback as information that one receives from agents such as teachers, parents, friends and other sources regarding performance and or understanding [13]. They explain that for feedback to be effective, it must provide answers to three critical feedback questions: "Where am I going?", "How am I going?", and "Where to next?". They also present a feedback model that consists of four different levels:

- i) *The task (FT)*: This level involves feedback regarding a performed task, and whether the task is correct or incorrect.
- ii) *The Processing (FP)*: Focuses more on helping students learn the necessary processes required for completing a task.
- iii) *The Regulatory (FR)*: Here, feedback is directed at self-regulation such as instilling confidence in a student to encourage further engagement on a task.
- iv) *The Self (FS)*: This is the least effective feedback level of the four and involves giving feedback to students in the form of praise

through phrases such as "great job!" or "Well done". It is the least effective because feedback at this level alone rarely contains information about the task itself.

The most effective feedback involves a combination of the task performance level, the processing level and the regulatory level [13]. Students move through these levels in a sequential manner. This combination makes it possible for a student to know where he or she is, how he or she is doing and what he or she should focus on next in order to make progress with a task. This also consequently reduces frustration and trial-and-error behaviour in students.

## 2.4 Pedagogical Agents

Pedagogical agents are software agents such as autonomous or interface agents that support learning and provide assistance to students in for instance Interactive Learning Environments [15] or CSCLs such as CoLeMo and AUTO-COLLEAGUE [2, 5]. In short, "pedagogical agents are software agents with a pedagogical agenda" [16]. Pedagogical agents have been employed in a variety of intelligent tutoring systems (ITS) and are presented either as text-based agents (displaying only text in the user interface) or animated agents for example [18, 19]. Some agents are specifically designed for educational purposes and play specific roles where they exhibit expert knowledge in specific domains such as, facilitating collaboration in ITSs for example [2, 5] or providing assistance in ILEs [15].

While the above mentioned systems are all designed to help novice designers learn UML design skills, these systems may present some challenges for students who find it hard to seek help in social environments. Firstly, these systems are designed to employ collaboration between peers, which in itself is not a bad thing. However, this may defeat the very purpose of simplifying the learning experience for UML novices by employing way too many functionalities. Using a chat in a UML learning environment may present similar challenges to the ones that students in social settings such as classrooms are faced with [12]. Students working in collaborative learning environments may become reluctant to seek help as they do not want to be perceived as incompetent or weak by their peers or instructors [12].



The CSCL systems mentioned above are very similar in nature. They provide domain-level knowledge feedback and feedback on collaboration. However they do present one particular limitation: The domain-level feedback that they provide is limited to the feedback at task level (FT). According to Hattie and Timperley, too much feedback based solely on this feedback level can be detrimental to a student's performance [13]. We propose using UML constraints and combining three different levels of feedback (i.e the FT level, FP level and FR level) as the most effective way of modeling and presenting automated feedback.

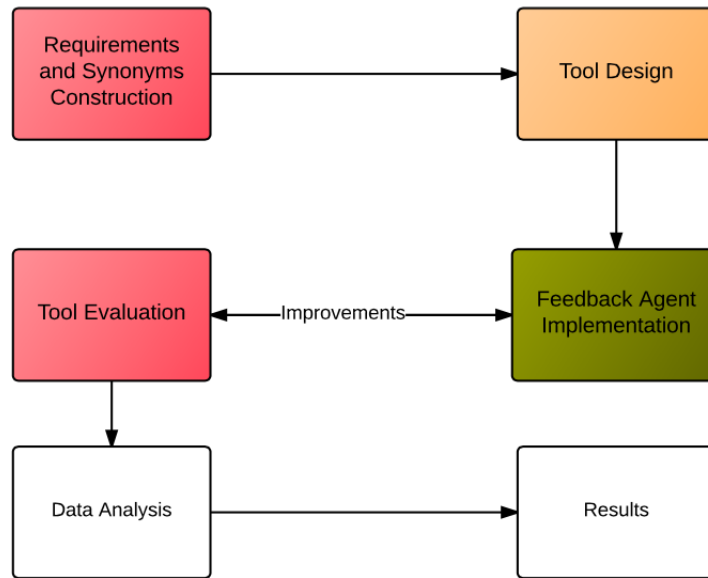
Using our knowledge in the different kinds of help seeking behaviour that students may present in ILE's, one of our goals was to design a system that could support students who would otherwise not seek help in for example classroom settings, to seek help from the system. In order to provide effective feedback to novice designers, we designed a system that provides the kind of feedback that aims to answer the three critical questions "where am I going?", "how am I going?" and "where to next?". We combine three different feedback levels, i.e the FT level, FP level and FR level to produce the most effective type of feedback as presented by [13].

Unlike the CSCL's mentioned above, our focus with this research was not only about providing feedback to students using UML-based constraints, rather we also focused on exploring help-seeking behaviour in ILE's, the different types of feedback, as well as ways in which to provide effective feedback to support students in attaining their goals in UML design.

### 3 Method

A design research approach was used for this study [8]. Research Design was the most suitable for this research because the study involved the implementation and evaluation of a designed software artefact to address a specific problem domain [8]. The study was conducted iteratively and incrementally, and included observational field studies (monitoring the use of the developed feedback agent) as the evaluation method of our choice.

Evaluations involving 5 participants were conducted and necessary improvements were made to the tool based on the feedback



**Fig. 1.** Overall Research Framework

that we received. A UML design task about a tank game was used as a modelling task by students who modelled all the student design solutions explored during this research and as a modelling task for tool evaluators during the evaluation processes. Fig. 1 presents the different phases of our research process.

### 3.1 Identifying Design Mistakes and Possible 'Synonyms'

A total of 59 UML design solutions of the tank game were manually investigated with the purpose of identifying common design errors made by students when designing class diagrams. The discovered mistakes represented areas where students needed learning support, and the following requirements were developed around them to support learning:

- Offer domain-level feedback on UML naming conventions and general design principles.
- Inform designers of missing interface classes.
- Provide advice on missing classes

- Provide advice on missing attributes
- Provide advice on missing operations
- Provide advice regarding inheritance

We also manually investigated these models to identify the most commonly used names by students for classes, associations, attributes and operations. The common names were then used to build a 'synonyms' database of possible names that students could use when implementing the task in the ILE. The possible names were used for the synonyms database as there is no single best solution for a problem, and there are often several possible class names, attributes or operation names for the same requirements. For example some class names such as PlayerController are not 'real' words and would therefore have no real synonyms. Therefore, creating possible 'synonyms' based on the most frequently used names for classes seemed the most plausible approach to take. The UML models were collected from three different sources as follows:

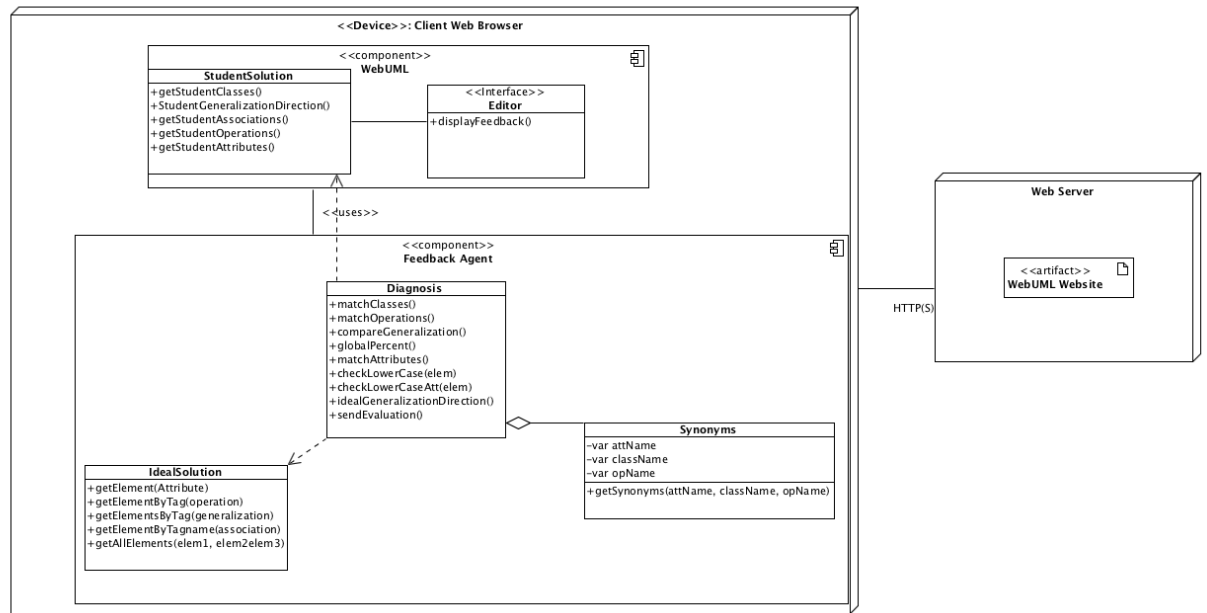
*Makerere university, Uganda.* 27 of the models we studied were designed by software engineering students at Makerere university in Kampala, Uganda. The data from Uganda was subset data collected for a separate research experiment. However, since the task in the experiment was the same as the task given to students for this research, and to increase the external validity of this research by increasing the number of data points used, we decided to use the models submitted by the students in Uganda as data for this research. The students modelled and submitted their task solutions on-online using WebUML.

*Leiden University, Netherlands.* A further 28 models were designed by software engineering students at the university of Leiden in the Netherlands. These models were retrieved from a free online UML repository (<http://models-db.com/>).

*Gothenburg University, Sweden.* And finally, 4 of the models that we studied were designed by software engineering students at the university of Gothenburg.

### 3.2 Design and Development

We developed a pedagogical agent designed to provide guidance and feedback to novice designers during design sessions. The feedback agent takes in as input, an ideal solution designed by an educator and uses this solution as part of the basis (the other being design principles) for providing feedback to novice designers. Feedback is triggered on-demand when a student presses the validation button.



**Fig. 2.** The Main Components

Once feedback is requested, the pedagogical feedback agent first compares class names, associations, inheritance association direction, attribute names and method names in the student's solution against the same parameters in the ideal solution for direct hits. If some parameters are still missing, the FA goes through a 'database' of allowed possible class element names. The possible names are not only limited to possible synonyms, rather we also created a list of possible names using the most common names in the class diagrams

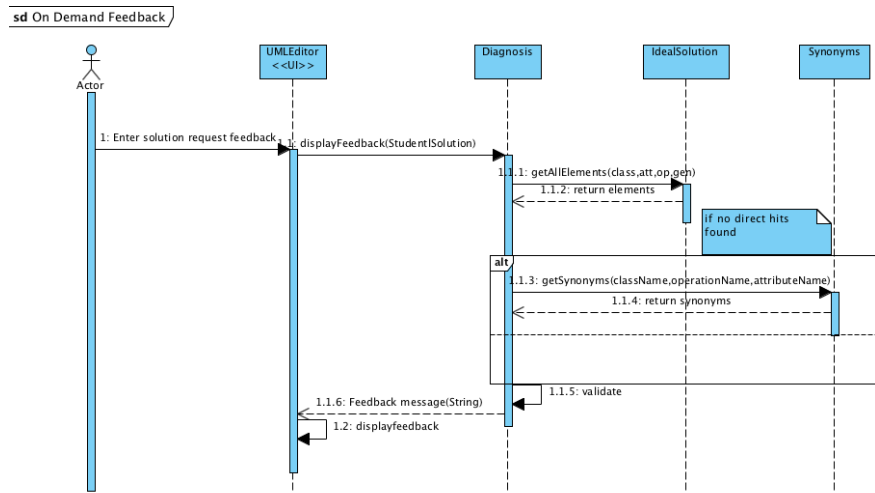


Fig. 3. System Sequence Diagram

that we studied in the elicited data from the Uganda experiment, UML repository and the Gothenburg university students. We created the homemade synonyms for cases where students use names that are not real words or cannot be synonyms of any existing words for example Iplayer, or PlayerController. An example of the possible classes extracted from the elicited data is shown in Fig.3. below:

```
var classes = TAFFY([
  {name: "Tanks"}, {name: "Tanker"}, {name: "ShermanTank"},
  {name: "Sherman_Tank"}, {name: "ShermanTank"}, {name: "Sherman "},
  {name: "CenturionTank"}, {name: "Centurion_Tank"}, {name: "Centurion_tank"},
  {name: "PanzerTank"}, {name: "Panzer_Tank"}, {name: "Silver_Bullets"}, {name: "IPlayer"},
  {name: "I_Player"}, {name: " Panzer"}, {name: "PlayerControl"}, {name: "PlayerController"},
  {name: "Worlds"}, {name: "Level"}, {name: "level"}, {name: "Levels"}, {name: "Levels"},
  {name: "Scores"}, {name: "Scoreboards"}, {name: "Metal_Bullets"}, {name: "MetalBullet"},
  {name: "drive"}, {name: "Gold_Bullet"}, {name: "Gold_Bullets"}, {name: "GoldBullets"},
  {name: "Metal_Bullets"}, {name: "PlayerInterface"}, {name: "ScoringBoard"}, {name: "ScoreBoard"}]);
query = classes().select("name");
```

Fig. 4. Possible class names

The FA also checks student solution’s class names, attribute names, and operations names for uppercase (first letter in a class name) or lowercase (first letter in attributes and operations names). If class names starting with lower case letter are found, appropriate feedback is provided (e.g ”class names must always start with a cap-

ital letter”). Operation names starting with an uppercase letter are also checked before necessary feedback is provided.

A method called `globalPercent` is also updated every time the feedback agent makes a correct 'diagnosis' and the result from this method is presented as a progress bar in the user interface to indicate progress towards the ideal solution.

## Feedback Messages

Feedback has the power to influence how people learn and achieve their goals [13]. However, providing feedback directed at the wrong level, in the wrong way or at the wrong time may result in undesired consequences [13]. Since our aim was to design feedback that was directed at the right level, presented well and at the right time, we thought it a necessary step to investigate how people process new information by trying to understand the concepts of cognitive load, and how best to present feedback to students that are learning new design skills.

Research on cognitive learning has demonstrated the importance of considering cognitive load when designing efficient Interactive Learning Environments [20, 21]. Kalyuga [21] explains that the human cognitive architecture has limitations and these limitations must be carefully considered since such environments are expected to support the construction of new knowledge in students as a consequence of deep cognitive processes. As humans, despite our long term memories (LTM) ability to store unlimited amounts of data for unlimited amounts of time, our cognitive systems employ a restrictive mechanism functionality that limits the range of immediate changes to our LTM [21]. Working memory (WM) is responsible for processing newly acquired knowledge and for integrating this knowledge with the knowledge that already exists in LTM. And unlike LTM, its capacity is very limited and is easily overloaded when exposed to too much new information [21]. Extraneous cognitive load can result from poorly designed instructional systems where too many new elements of information are introduced into working memory and/or are introduced too fast to be successfully incorporated into LTM structures. [21].

In order to minimise the risk of causing extraneous cognitive load and to support efficient learning in students, we designed the FA to dispense feedback messages in sequential stages [21]. Feedback about a particular task element implementation is only released once the student has a particular number of class diagram elements implemented in his or her solution. For example, a student solution containing more than 10% of correctly implemented class names and less than 20% of expected operations names, triggers a feedback response containing feedback information regarding class names only, and zero feedback on implemented operations.

**Table 1.** Feedback Design

<i>FeedbackMessages</i>	<i>FeedbackLevel</i>
Your classes Tank, Centurion look good but you are still missing some important classes.	FT Level
Read the task carefully and see if you can find the remaining classes by identifying nouns, preferably complete singular nouns.	FP Level
Class name(s) Tank look good	FR Level

Table 1 presents an example of the different levels of feedback that are presented to a novice designer regarding classes during a help-seeking episode in the ILE. We employ a combination of three different feedback levels (i.e the task performance level, the processing level and the regulatory level) to provide the most effective feedback to novices as explained by [13]. With these levels of feedback, novice designers can see where they are, how they are doing and how they can move forward to attain their goal of completing a task. Our aim with this feedback is to move the designer sequentially from task to processing, and then to regulation [13]. According to Hattie and Timperley, feedback that is presented in this manner has the most effect on how students learn [13].

Initially, the feedback agent only provided information that was directed at the task level (FT). However, during a couple of evaluation sessions we noticed that students who after some attempts to complete the task but got stuck, seemed to quickly reduce their cognitive effort, and turned to trial-and-error tactics to complete their design task. Fortunately, since our system development was done iteratively, we managed to optimise our feedback messages to increase effectiveness and reduce frustrations-levels in students. Improvements to the FA's feedback messages were conducted iteratively as follows:

**Phase 1:** The first stage of improvements were conducted to include more feedback about which important classes or class elements were missing from the students solution as the first feedback the feedback agent provided was mostly directed at the task level and seemed to induce trial-and-error strategies in the evaluators.

**Phase 2:** The second phase of improvements were conducted to include feedback that advises novices to either rename or delete class diagram elements names that were considered wrong or irrelevant to the task by the feedback agent.

**Phase 3:** And finally, feedback specifying where and what students could look for in the text was included during the third phase of improvements implementation.

## User Interface

The user interface is very simple and minimalistic. An icon button with a link to a textual description of the task to be implemented and a feedback request button are provided at the top of the screen. Students can read the text describing the task and construct their design solutions in the workspace. They can switch back and forth from the text to editing mode whenever they want. Students decide when to submit their solutions and request feedback from the FA. Once feedback is requested, the agent displays the appropriate level of feedback in the left column of the screen. Novice designers can go



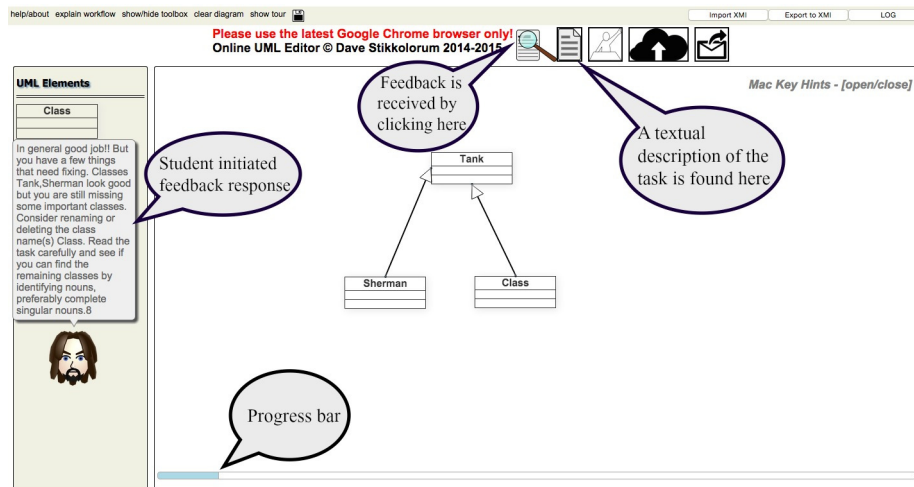


Fig. 5. User Interface

through this information, and continue with the task when they feel they have understood the feedback.

A feedback progress bar was also added to the bottom part of the user interface. It indicates overall progress and reflects data about matched classes, attributes, and operations. Progress bars or 'percent done indicators' are a graphical technique of informing users on how the task their currently undertaking (or that the system is undertaking) is proceeding [20]. Progress bars are used in a lot of systems to indicate for instance, the progress of a file transfer, or as a way to provide an estimation of the time remaining before a long task is completed. Research on progress bars indicates just how important it is to use them in user interfaces [20]. According to Myers [20], users would much rather use a system that provides a progress indicator as a feature in its user interface than one without.

The progress bar implemented in this system is by no means a grading system, it only reflects how the student is doing with regards to whether or not he or she has implemented the most important elements of the given task. It is a type of feedback that serves as progress reassurance and helps students feel better about the system [20]. At a quick glance, students can see just how far they are from

the ideal solution and decide whether to seek more help by reading the feedback displayed on the same window more closely.

### 3.3 Data Collection and Evaluation

The data were collected qualitatively through an observational design evaluation method with the purpose of uncovering new requirements, collecting feedback on how students feel about the tool's usability, the feedback they received from the feedback agent, and what they felt needed to be improved. Five first and third-year software engineering students enrolled at the university of Gothenburg volunteered to take part in the evaluations. The students learned UML modeling concepts during their first year at university and had a good idea of what UML is about. The students were given a UML design task to implement under observation and new improvements were iteratively made to the FA based on the feedback.

#### The Task

For the evaluation, students were given a task to model. The task involved modeling a tank game which required the realisation of requirements such as different types of tanks, world, levels, a player, different types of bullets and a way to manage the recording of scores.

#### Data Collection Methods

The data were collected iteratively using qualitative methods i.e interviews, and observations. Evaluation participants were asked to construct design solutions of the tank game and there was no set time frame for completing the task. Participants were free to stop whenever they felt they were done.

*Observations:* Observations were conducted in order to observe how the students used the online UML editor to design the task and how they respond to feedback.

*Interviews:* The participants were also interviewed using open ended questions and the interview sessions were recorded with their consent. The interview sessions were aimed at having the students

help us identify any problems with the tool, verify the results obtained from the observations and to evaluate the artefact [8]. The following questions were asked during the interview sessions:

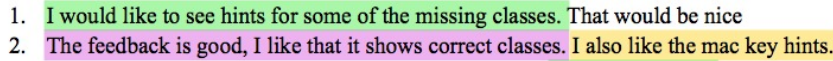
1. What kind of feedback do you think would be have been great to see now?
2. What do you like about the implemented features?
3. What do you like about the tool as a whole?
4. What do you think about the progress bar?
5. How do you think the tool could be improved?
6. Which part of the feedback helped you solve the problem?

### Data Analysis

The collected data were analysed thematically, as this allowed us to go deeper into the information that we got from the evaluations. Thematic analysis is a qualitative data analysis approach that can be used to analyse, identify, assign and describe themes within data in research [11]. All units of data from the interviews (transcripts) were given a particular code and examined in more detail. Themes were later developed around these codes. The data were analysed qualitatively [9, 10] in four steps as follows:

**Step 1:** *Reading Through Transcripts:* Transcripts were thoroughly read through several times. Notes about our first impression of the data were recorded.

**Step 2:** *Coding Relevant Data:* Relevant data such as sections, phrases or sentences that shared commonalities were labelled using the same colour. For instance, sentences that contained the words 'progress bar' were colour coded with the colour 'yellow' to represent features that the students found useful regarding the progress bar located at the bottom of the user interface, and 'underlined yellow' text was used to represent user interface improvements suggested by the students. Fig.5 below shows the different colour themes that were used to organise the data before major themes were assigned.

- 
1. I would like to see hints for some of the missing classes. That would be nice
  2. The feedback is good, I like that it shows correct classes. I also like the mac key hints.

**Fig. 6.** Colour coded data

**Step 3:** *Selecting the Most Important Codes:* The coded text was read through again and the codes that we thought were of most relevance or occurred more than once (e.g most common interview questions responses or recurring words, sentences or phrases) were selected for labelling.

**Step 4:** *Labelling the Themes:* Finally in step 4, all the collected themes were labelled and the relations between these themes were noted. Any data concerned with user interface improvements such as changing the colour of the progress bar were placed under the 'UI' label. Data that suggested that the students would have wanted more help than was provided by the feedback agent were organised under the 'Pedagogical Agent' label, and all sentences that seemed to suggest how the tool could be improved or how easy the tool features are to use were organised under the 'Usability' label. We ended up with three major categories i.e 'Usability', 'Pedagogical Agent' and 'User interface', each of which has two sub-categories namely 'useful features' (what the students found useful) and 'improvement suggestions' (improvement suggestions). Fig.6 illustrates two of the three major themes, and their subthemes.

### 3.4 Internal and External Threats to Validity

*External:* Our study focused on a small group of students. There could be an ocean of data out there that we could not reach due to our limited number of participants and resources. Because of this, although the feedback agent was implemented using requirements and feedback data provided by software engineering students at the university of Gothenburg, the views and expressions about how the tool works and whether or not it helps students learn UML modeling skills more efficiently and effectively because of the type of feedback provided cannot really be generalised. We tried to

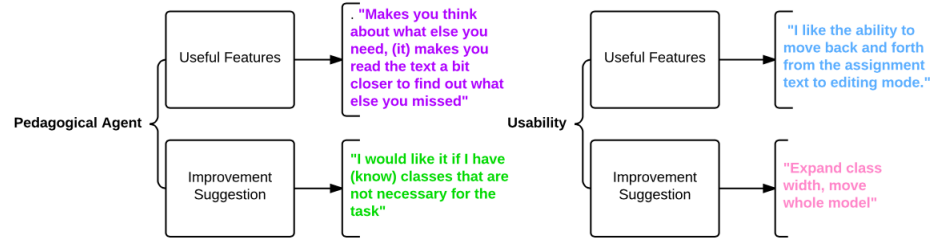


Fig. 7. Thematic Analysis

avoid this by staying as open- minded as possible and by accepting as many improvement suggestions as was possible to implement from the small number of participants that we had.

*Internal:* There is a possibility that only the features that we thought were worth implementing would be implemented, despite an indication of other more important features in the data collected from the students. We tried to overcome this by studying the improvement suggestions carefully and selecting the ones that occurred most frequently for improvement iterations and by discussing these with the students who had participated in the evaluations and with other stakeholders.

## 4 Results

Five software engineering students from the university of Gothenburg volunteered to participate in the evaluation of the tool. The participants were third year students who were familiar with UML design albeit with varying proficiency. The evaluations were aimed at investigating what the students felt about the system, how useful they found the feedback messages to be, what they thought about the user interface and the general usability and feel of the ILE. Themes obtained from the thematic analysis are key to determining what all the participants think about the feedback presented by the pedagogical agent as well what they think about performing a UML task in the online learning environment.

During data analysis, each labelled theme was assigned a total of 2 subthemes, 'useful features and 'improvement suggestions'. The subthemes 'useful features' describe the benefits experienced by the students through the use a particular feature of the ILE. The sub-theme 'improvement suggestions' was assigned to characteristics of data referring to improvements in the pedagogical agent, the usability of the system as well as suggested improvements in the user interface.

#### 4.1 Pedagogical Agent

The Pedagogical agent theme contained the largest number of codes. A total of eleven codes were identified under this theme. The theme presented here reflects two things:

i) By reading the feedback presented by the pedagogical agent, the participant's showed the ability to identify what they needed to do next in order to complete or make any progress with the task.

Five out five participants were pleased to receive affirmation about which classes, attributes or methods that they constructed were correct. They also expressed satisfaction on the pedagogical agents ability to advise them on what they needed to do next in order to move forward with the task. It is fair to say that all of the participants benefited from getting feedback that informed them of their current status, and as well as feedback that pointed them towards where to go next with the task.

*I: Which part of the feedback helped you solve the problem?*

*\*: "The feedback is good, I like that it shows correct classes. The part where it says, classes Tank, centurion and the others look good. I know that these classes are correct. Also, the part that says find classes by identifying nouns in the text."*

Some mixed messages where also noted. Despite acknowledging the benefits of knowing which elements of the class diagrams were correct, we observed that some of participants remained adamant

about deleting or renaming some of their 'incorrect' class diagram elements and neglected to follow some of the advice from the feedback agent. We also noticed a few participants switch to trial-error-tactics, but what we found particularly interesting was that the students who switched to trial-and-error tactics also admittedly had lower UML proficiency, and were also more willing to modify their class diagram elements completely in accordance to the feedback provided by the pedagogical agent. The participants that seemed to have more higher prior knowledge in design, maintained some parts of their solutions that they felt were correct despite them not being confirmed as good solutions by the feedback agent.

ii) The participant's need for more help in the form of hints from the feedback agent.

Almost all participants expressed the need for more help in the form of hints. Suggested improvements included adding a bit more information about the missing classes, irrelevant classes or class elements and hints related to them. The requests for hints or more help is likely to be because of the way feedback is designed to be presented (i.e a little feedback at a time and then increasing from there to minimise or even avoid extraneous cognitive load), however we cannot draw anything conclusive given the small sample size of the evaluation participants.

*\*: "Someone like me who's not good at UML, I need more hints".*

## 4.2 Tool Usability

We identified the second largest number of codes in the area of usability. A total of nine codes were identified in the area of usability. This theme reflects the student's experience with using the features provided in ILE and what they feel could be improved. There were aspects of the transcript that highlighted that the majority of the participants thought that the tool was easy to use. This was also confirmed during observation. Participants showed no indication of struggling with either switching between reading the task and returning to the editing pane, or dragging and dropping the different

class elements. They found the tool to be relatively easy to use.

*I: What do you like about the tool as a whole?*

*\*: "Its quite easy to use".*

From the observations and interviews, we noted how some of the participants seemed to want to expand the width of the class after constructing long class names or attribute names and declaring types. The majority of participants also expressed the wish to have the tool support the locking of classes so as to enable them to move the whole model around the modeling pane.

*\*: "There should be a way to move the model. Find a way to resize the class."*

*\*: "Make it possible to expand class width."*

### 4.3 User Interface

This theme contained the least amount of codes. These codes all referred to how difficult or easy it was for the participants to notice the progress bar and what the participants thought about having a progress bar in the user interface. Most of the participants did not notice the progress bar. This is evidence enough to suggest that the progress bar design needs to be improved. The participants seemed to either not see the progress bar or mistook it for a normal scroll bar. This could have been due its colour as one participant suggested or its position within the user interface.

*I: What do you think about the progress bar?*

*\*: "The progress bar is very vague, colour-wise. I thought it was a scroll bar. Maybe add percent or something?"*

The majority of the participants did not notice the progress bar, and only a few suggested improvement changes. The suggested changes included using a stronger, bolder colour and including some



text that indicated progress by either labelling the progress bar with a word or phrase or by including a percent-done numeric symbol.

## 5 Discussion

The objective of this study was to explore how a feedback agent for an online ILE for UML for novice designers could be developed, as well as how the feedback agent would best be designed to effectively provide feedback to novice designers. To meet this objective, we first had to investigate and understand what feedback means, what influence feedback has on students and which different types of feedback exist. We also had to understand help-seeking behaviour in ILE's, cognitive load in students and how feedback would be presented as this would benefit the design of the agent. For instance, we designed the agent to present feedback messages in a sequential manner to minimise extraneous load in students [21]. In order to provide effective feedback and to enhance learning, we designed and implemented a pedagogical agent that combines three different types of feedback levels (i.e FT, FP and FR). When combined, these levels of feedback provide the most effective feedback aimed at answering three critical questions as elaborated by Hattie and Timperley [13]. The pedagogical feedback agent was developed incrementally and iteratively and was integrated with an online UML editor to create a learning environment where novice designers could solve tasks and receive on-demand formative feedback. Qualitative evaluations were then carried out and the data were analysed using thematic analysis.

### The Feedback Agent

The results detailed in section 4 highlight important findings as to how students respond to receiving automated feedback. We received mostly positive feedback about the feedback provided by the pedagogical agent. What seems particularly important in terms of which parts of the feedback helped the participants attain their task goals was their ability to identify what they needed to do next in order to move forward in their task. However, because the initial feedback design contained no information about which class diagram elements were irrelevant to the task, some participants seemed to switch to

trial-and-error tactics to 'find' the classes or class elements whose names did not appear in the feedback message. The trial-and-error strategies displayed by some participants during the evaluation highlighted the need to fine tune the feedback design, and structure. The feedback that these students received from the pedagogical agent was mostly at task level (FT). Hattie and Timperley [13], stress that too much feedback at this level may remove a student's focus on the strategies necessary to attaining their goal, to focusing on their immediate goal. Improvements in the way feedback is presented were made to provide more formative feedback in order to reduce frustration, trial-and-error tactics and to encourage more cognitive effort from the students [13].

### **Why did some students maintain their 'incorrect' answers despite feedback to suggest otherwise?**

Aleven et al. [12], explain how academic goal orientation plays an import role in the willingness of students to seek help. Students who possess a high level of intrinsic orientation show an interest in the subject matter and aim to understand and learn. Having an interest in a subject encourages students to ask questions when they feel the need for help. On the other hand, students with extrinsic orientation also want to learn, but are usually mostly interested in achieving high scores and making a good impression on their teachers or peers [12]. Although one cannot draw generalised conclusions because of the small number of participants, in our study intrinsic and extrinsic behaviours were typified by some of students willingness to seek more help from the agent and by the reluctance to change their "correct" models in the other students. It should also be noted, however, that the participants modelled their tasks under observation. This may have created a social climate and may have affected the way that the participants responded to feedback and their behaviour towards seeking help. The response regarding the tool's usability as a whole was mostly positive. The participants found the tools features to be relatively easy to use.

### **The Progress Bar**

Feedback regarding the progress bar indicated that most of the students did not notice it in the user interface and as such merits further consideration for improvements. Based on [20], we believe that implementing a progress bar correctly would be beneficial. Progress bars “help novices feel better about the system by showing that a command has been accepted and the task is progressing successfully” [20]. Furthermore, the design of the progress bar, its colour, size and its location within the user interface may be important factors to consider.

### **Locking of Classes into One Moveable Model**

With the feedback regarding the tool as a whole, the students suggested the locking of classes into one moveable model. Although the locking of classes into one model may seem a separate issue from the feedback agent, the evaluators responses indicated otherwise. The response signified the importance of having a seamless interaction of all features involved in the learning environment to minimise frustration in students and to make students feel that they are indeed working in this one environment. When students interact with the modelling tool (i.e WebUML), they interact with one tool and not with the feedback agent alone as separate tool; they implement their design solutions by directly interacting with the modelling tool before requesting feedback from an agent that is embedded within the modelling tool. If tasks take long to implement because students have to move things around the workspace one at a time to fit in more items (perhaps as advised by the feedback agent), frustration issues may come into play. Locking the classes together could help serve as an indirect beneficial support for learning by allowing students to focus more on the task at hand and not on issues they feel are getting in the way of implementing their task.

## **6 Conclusion**

This paper describes the implementation of a feedback agent designed to provide automated feedback on software design to novice

designers. We designed a pedagogical agent and integrated it with an already existing online UML modeling tool called WebUML [14]. With the goal of providing effective feedback to support learning, we designed a system that could provide on-demand formative feedback that gradually increases in detail to support learning and reduce any chances of extraneous cognitive load [21]. We conducted evaluations with the purpose of discovering how students respond to the automated feedback presented by the pedagogical agent, how they felt about the learning environment as a whole, and to elicit improvement suggestions for the tool. The results from this research are encouraging and most of the evaluation participants thought it would be useful to use such a tool when solving UML assignments.

## 7 Future Work and Limitations

Although we received mostly positive feedback about the pedagogical agent from the evaluations, there are still some issues that merit further careful thought.

- How to have the model provide hints, when to provide these hints and what type of hints to provide? Many of the evaluation participants seemed to suggest the implementation of 'hints' to aid in their strategies for constructing their solutions. Hints could either be system initiated after a set number requests for feedback by the student, or user initiated.
- A way to measure the effectiveness of the feedback provided? Although the feedback about the pedagogical agent was mostly positive, statistically testing its effectiveness by conducting an experiment would have been desirable. However, due to the time limitations we had, we thought it wise to leave this type of evaluation for future research. An experiment with a reasonable enough sample size would give us more insight as to how effective providing formative feedback is. We would then base our final findings not only on qualitative results but on quantitative results of a large sample size too. This could also affect the generalizability of our findings.

- Lock classes for easier manageability of the model within the editing pane. We strongly feel that locking the classes together into one movable diagram would improve user experience and minimise frustration in students.
- How to adapt the feedback agent to evaluate other tasks besides the tank game? The feedback agent discussed in this paper was built around a specific UML modelling task (i.e the tank game). Adapting it to evaluate a variety of different UML design tasks and solutions is definitely worth careful thought.

**Automated Insertion of 'Synonyms'.** One possible way of achieving this would be to have an educator enter data about a UML design task (i.e possible class diagram element names) into a Google Spreadsheet. Since the feedback agent is designed to perform a comparative validation of the students solution against an ideal solution by checking specified UML class diagram parameters (checks any UML class diagram in xml format), the only data that an educator would have to provide in the spreadsheet would be data regarding allowable alternative names that students can use for a given task. The alternative names would then be converted into a Taffy-DB compatible data file that the feedback agent currently makes use of. This conversion process could be automated using a tool called Google Apps Script. With Google App Scripts one can manipulate the provided Google spreadsheet data using JavaScript. A url to the spreadsheet could be added to the querying source code of the feedback agent for querying of future insertions and automated conversions. The small snippet of code in Fig.7 below demonstrates how a Google spreadsheet is read before an automated conversion of data can be performed. A user page where educators can upload ideal solutions in xml format to a database could also be implemented.

**Acknowledgments.** I would especially like to thank Dr. Michel Chaudron, professor Dave Stikkolorum and Ho Quang Truong for their invaluable support and guidance during this research. I would also like to thank the volunteers who participated in the evaluations for their time and constructive feedback.

```

function readRows() {
var spreadsheet = SpreadsheetApp.openById('SPREADSHEET_ID')
.getActiveSheet();
var readRange = spreadsheet.getRange(5, 1, 500, 20);
var values = readRange.getValues();
//some more code here
}

```

**Fig. 8.** Spread Sheet querying

## References

- [1] Budgen, D., Burn, A.J., Brereton, O.P., Kitchenham, B.A., Pretorius, A.: Empirical evidence about the UML: A systematic literature review. *Software - Practice and Experience* (2010)
- [2] Chen, W., Pedersen, R.H., Perttersen, ...: CoLeMo: A collaborative learning environment for UML modeling. *Interactive Learning Environments*. Vol. 14. No. 3, 233–249 (2006)
- [3] Auer, M., Tschurtschenthaler, T., Biffl, S.: A Flyweight UML Modeling Tool for Software Development in Heterogeneous Environments. In: *29th IEEE EUROMICRO Conference New Waves in System Architecture* (2003)
- [4] Baghaei, N., Antonija, M., Irwin, W.: Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams *LNAI 4511*, 147, 217–2007 (2007)
- [5] Tourtoglou, K., Virvou M.: User Modeling in a Collaborative Learning Environment for UML *IEEE Computer Society* (2008)
- [6] Eshuis, R., Wieringa, R.: Tool Support for Verifying UML Activity Diagrams. In: *IEEE Transactions on Software Engineering* Vol.30, No.7, (2004).
- [7] Tessmer, M.: Planning and conducting formative evaluations. *IEEE Kogan Page Limited* (1993).
- [8] Hevner, A.R., March, S.T., Park, J., Ram, S.: Design Science in Information Systems Research. *IEEE MIS Quarterly*, Vol.28, No.1, 75–105. *IEEE Press*, New York (2001)
- [9] Bryman, A.: *Social Research Methods*. Oxford University Press, (2012)
- [10] Kvale, S., Brinkmann, S.: *Learning the Craft of Qualitative Research Interviewing*. SAGE (2015)
- [11] Tessmer, M.: Recommended Steps for Thematic Synthesis in Software Engineering In: *International Symposium on Empirical Software Engineering and Measurement* (2011)

- [12] Aleven, V., Stahl, E., Schworm, S., Fischer, F., Wallace, R.: Help Seeking and Help Design in Interactive Learning Environments. In: Review of Educational Research. Vol. 73, No. 3277–320. IEEE Press, New York (2003)
- [13] Hattie, J., Timperley, H.: The power of Feedback. In: Review of Educational Research, Vol. 77, No. 1, 81–112. (2007)
- [14] Dave, R., Stikkolorum, Truong, H., Chaudron, M.R.V.: Revealing Students UML Class Diagram Modelling Strategies with WebUML and LogViz. Accepted for presentation at the Euromicro SEAA conference, August 26-28th and publication in the conference proceedings (2015)
- [15] Morch, A., ondahl, S., Dolonen, J.A.: Supporting Conceptual Awareness with Pedagogical Agents. In: Information Systems Frontiers, Springer Verlag (Germany), Vol 7. No. 1, 39–53 (2005)
- [16] Haake, M.: Embodied Pedagogical Agents: From Visual Impact to Pedagogical Implications. ISRN LUTMDN/TMAT -1032-SE EAT (2009)
- [17] Ayala, G., Yano, Y.: GRACILE: A framework for collaborative intelligent learning environments. In: Journal of the Japanese Society of Artificial Intelligence, Vol.10, No.6, 157–170(1995)
- [18] Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhja lms-son, H., et al.: Embodiment in conversational interfaces. In: Proceedings of CHI 99, Pittsburgh, PA: ACM Press (1999)
- [19] Mayer, R.E., Moreno, R.: Nine Ways to Reduce Cognitive Load in Multimedia Learning. In: EDUCATIONAL PSYCHOLOGIST, Vol. 38, No.1, 43-52 (2003)
- [20] Myers, B.A., The Importance of Percent-Done Indicators for Computer-Human Interfaces. In: Proceedings of CHI 85, April (1985)
- [21] Kalyuga, S.: Enhancing Instructional Efficiency of Interactive E-learning Environments: A Cognitive Load Perspective. Educ Psychol Rev, 19:387-399 DOI 10.1007/s10648-007-9051-6 (2007)
- [22] Martens, R.L., Valcke, M.M.A, PORTIER, S.J.: Interactive Learning Environments to Support Independent Learning: The Impact of Discernibility of Embedded Support Devices. Science Educ. Vol.28, No.3, 185 – 197
- [23] Baghaei, N., Mitrovic, A., Warwick, I.: Supporting collaborative learning and problem-solving in a constraint-based CSCL environment for UML class diagrams. Computer-Supported Collaborative Learning 2:159 –190,DOI 10.1007/s11412-007-9018-0 (1997)
- [24] S, Nelson-Le Gall,: Help-seeking: An understudied problem-solving skill in children. In: Developmental Review, Vol.1, 224-246 (1981)
- [25] Stahl, E., Bromme, R.: Not everybody needs help to seek help: Surprising effects of metacognitive instructions to foster help-seeking in an online-learning environment. In: Computers and Education, Vol.53 1020–1028 (2009)

- [26] Wood, H., Wood, D.: Help-seeking, learning and contingent tutoring. *Computers and Education*, Vol.33 153–169 (1999)
- [27] Schworm, S., Renkl, A.: Learning by solved example problems: Instructional explanations reduce self-explanation activity. In:W. D. Gray & C. D. Schunn (Eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society*, 816–821(2002)
- [28] Anderson, J.R.: *Rules of the mind*. Hillsdale, NJ: Erlbaum.
- [29] Masthoff, J., VanHoe R.: *Appeal: A Multi-Agent Approach to Interactive Learning Environments*. European Workshop on Modeling Autonomous Agents, MAAMAW. Berlin, Springer-Verlag (1996)