Studies into interactive didactic approaches for learning software design using UML

Stikkolorum, D.R.

Chapter 10

An Online Educational Agent: Automated Feedback for Designing UML Class Diagrams

Description of contributions of the authors
The research in this chapter was done in collaboration with Helen Anckar. At the time of writing Helen had conducted her bachelor project at the joint department of Computer Science of the University of Gothenburg and Chalmers University of Technology in Sweden. Her topic was to design and implement a proof of concept of an educational agent that gives students feedback when they are creating a class diagram.

The contributions of Dave Stikkolorum were: introducing the topic, taking part in the discussions about how to apply the concept of pedagogical agents in the field of software design education, supervising the research process, supporting the software design and programming process and writing this chapter.

This chapter describes our exploration of the automatic feedback to students during their process of designing a class diagram. To this end, we extended our online educational design tool WebUML with a feedback on demand feature. This feature

This chapter elaborates and extends on the bachelor thesis of Helen Anckar: Helen Anckar. Providing automated feedback on software design for novice designers. BSc thesis, Goteborg University, 2015

was implemented by means of a pedagogical agent. The feedback agent was evaluated by five students and resulted in future direction for research. We conclude the agent is still immature and further research is needed to be conducted in order to make such a technique useful for educators.

10.1 Introduction

As mentioned in the previous chapters of this dissertation, several studies point out that software design and its modelling activities are challenging topics for novices [80][69][113][141]. Given the fact that software design is a practical skill, we believe educational programs should offer students a 'learning by doing' approach (as discussed in Chapter 3) and facilitate the students with much exercise time to overcome their difficulties. At the same time, this requirement creates a challenge in organising and staffing a course:

- The need for much practical hours demand a large group of lecturers or teaching assistants (TAs) that have to be involved.
- It is a challenge to support the students at the right time. A lecturer or TA can only support a couple of students at one point in time. If the other students are stuck, they have to wait for help.
- Students could benefit from being guided during their design process. This enables the possibility to take smaller steps, instead of only having a evaluation at the completion of their task.

The above mentioned requirements are not easy to fulfil in the typical classroom. Both in classic lectures and online courses, high numbers of students challenge the process of evaluation and grading. In classic lectures, accompanied with practical workshops, 200 students or more are not an exception. Online, this number can even be higher. To overcome scalability problems, an automated approach of more tailored feedback is needed. Moreover, this enables students to practice at their own pace and in their own time.

The research in this study focuses on feedback. We investigate how the integration of a pedagogical agent in our online WebUML tool can support overcoming the aforementioned challenges. We aim to design an agent that is capable of guiding students towards their solution by giving feedback during the performance of their task.

There are several categories of criteria that lecturers use when giving feedback on the design of class diagrams (also available as checklist in Appendix E):

- Syntax – does the diagram follow the formal notation rules of the modelling language? At the time of writing UML 2.4 for example.
- Layout – Is the diagram readable? Does it not mix styles? For example: is an inheritance relationship drawn from top to bottom across the whole diagram?
- Consistency – is the diagram consistent with the other diagrams in the model? For example: does a class name correspond with a lifeline in a sequence diagram?
- Semantics – Does the diagram represent what it should? For example: a '1 .. *' multiplicity is applied. Which means '1 to many'. Was this meant?
- Requirements satisfaction – does the diagram support the (functional) requirements? For example: Does the class diagram have a set of classes that cover the responsibilities of the system that is designed?
- Design principles – are common design principles applied? For example: did the student take coupling and cohesion into account?
- Design decisions – were the right design decisions made and applied? For example: a pattern is applied to support a quality requirement.

Of these criteria, Syntax and Layout and - to some degree - Consistency and Design Principles can be addressed using well-defined formal- rules. The difficulty with evaluating students' designs lies mainly in evaluating the criteria: 'Semantics', 'Making design decisions' and 'Requirements satisfaction'.

Design principles can be considered in isolation and then evaluated as a rule. For example metrics for coupling and cohesion can be computed [25, 46] and compared against some threshold. However, a difficulty that arises is the balancing between the right amount of coupling versus the right amount of cohesion. Making such trade-off's are not easy to capture as a rule. Indeed, such decisions depend on knowledge in addition to the (technical) solution context, such as domain knowledge.

For constructing a diagram's layout certain styles can be followed. A modelling language's syntax is defined by a formal grammar. Modern modelling tools, such as Visual Paradigm [1] or Papyrus [2], are already equipped with systems for checking the syntax of diagrams and the consistency of these diagrams within a model. A large overview of consistency rules for UML is reported by Torre [150]. For layout, most modelling tools also provide functionality to guide the construction of the layout. Also [16] demonstrated good results in automatically evaluating layout of class diagrams using a machine-learning approach.

[1] https://www.visual-paradigm.com
[2] https://www.eclipse.org/papyrus

As mentioned above, the key challenge lies in evaluating the semantics, design decisions and requirements satisfaction. In this chapter we explore the possibilities for addressing these challenges.

As an extension of our WebUML editor, we developed a module that can be seen as a pedagogical agent that is capable of providing feedback to students by evaluating their class diagram. The aim of the agent is to provide students with feedback while they are working on a software design task in order to guide the student to the correct solution. The feedback agent evaluates the correctness of a student's design in order to give guidance for improvement. In our approach, the student solution is compared to a model solution and the feedback the student receives focuses on semantics, requirements satisfaction and the design decisions.

In this chapter we explore the possibilities and limitations of a feedback agent that guides students towards the solution of their software design task. The primary focus is on the role of the agent as a *guide*:

> the agent supports the student with hints about the quality of his design and suggests steps in the right direction. The agent aims to let the student discover the direction until a final situation is achieved.

In our approach the agent needs to *evaluate* some aspects of the design before giving feedback for guidance. However, the agent does not grade the student's task:

> as evaluator, the agent evaluates the quality of the student's software design. Evaluation takes place at the end of the student's task.

By researching this topic we aim to: i) improve online learning approaches, and ii) decrease the workload of lecturers.

The remainder of this chapter is organised as follows: Section 10.2 discusses related work. Section 10.3 discusses the research method of the study. Section 10.4 presents the results. We discuss the findings in Section 10.5 by answering the research questions. We close this chapter by concluding and proposing future work in Section 10.6

10.2 Related Work

This section describes existing research that is related to the evaluation of students' modelling products. We mainly focus on the following categories:

- tooling – the use of modelling tools in an educational context for supporting the

development of the students modelling skills

- help seeking – the way in which students seek help, at what typical moments, and what kind of topics they need help for
- providing feedback – providing the right type of feedback (task or self related) and the right amount (prevent cognitive overload) at the right time
- pedagogical agents – a digital, automated solution that recognises students' struggles, provides feedback and guides them towards solutions.

10.2.1 Interactive UML Learning Environments

Baghaei et al. present COLLECT UML, a tool for individual and collaborative learning of UML software design [10]. Their web based tool is capable of giving ondemand feedback by comparing submitted individual solutions with an ideal solution provided by the lecturer. The tool also supports a group phase in which the group solution exposes the differences between the individual solutions in order to enable group discussions for improvement of the group model. Their constrained-based tutor shows supporting messages on one side of the screen in order to guide them towards a correct solution. The feedback message are of different nature: Simple Feedback (the solution is right/wrong), Error flags (indicators for missing elements, such as classes, attributes/operations or types), and Hints (guiding from violated rules). The more help students need, the more detailed the feedback is (Hints).

Chen et al. designed CoLeMo, an online learning environment for collaborative UML modelling of geographically distributed students [24]. The PC tool uses pedagogical agents (see 10.2.4) for providing domain related advice and collaboration related advice during the activity of the design task students perform. By monitoring students in a chat area, responses to questions (agreement buttons) and activities in a shared workspace CoLeMo provides task related feedback and feedback on collaboration.

The modelling task related feedback is limited to the UML domain level, such as explaining a violation of a UML diagram construction rule. For example wrong use of the abstract class concept. The feedback on the collaboration is based on a weighted activity log. For example: additions of classes are assigned with a higher weight than renaming. Based on the logs the agent provides feedback about participation and warns when elements that were created by others are about to be deleted.

Tourtoglou et al. [151] designed AUTO-COLLEAGUE, a collaborative UML environment specifically designed for the formation of groups. They use models of prior performance of the users for creating an advice for the formation of groups. With the

help of defined stereotypes for the level of expertise, performance types and personality the tool generates the advice. The UML knowledge is observed by a human trainer or recorded by a wizard drive test module. It seems that the tool does not provide feedback on the UML designs.

The above mentioned works show that with interactive learning environments additional possibilities for providing feedback to students are created. At the moment of conducting our research we noticed that there was more work that relates to the feedback on the design process than actual software design and software design decisions. This motivated to conduct more research on feedback on actual software design and, in the future, include design principles in the feedback mechanism.

10.2.2 Help Seeking in Interactive Learning Environments

Aleven et al. propose to use Nelson-Le Gall's help seeking model [96] for children's problem solving tasks in an interactive learning environment [4]. The help seeking model consists of 5 stages:

1. Become aware of need for help.
2. Decide to seek help.
3. Identify potential helper(s).
4. Use strategies to elicit help.
5. Evaluate help-seeking episode.

Although this model originally was designed for classroom contexts it should also be applicable for interactive learning environments [4]. The difference is the 'helper' role. In a classroom context this role is fulfilled by the lecturer, in an interactive learning environment, next to asking help from the lecturer, this could be a computer program or glossary.

Aleven et al. conducted a study [3] in which 15-year old students used a glossary and an intelligent on-demand hint system for help seeking. They concluded that the students rather use the hints than the glossary. For the hints, they tend to search for the more specific hints (close to the solution) than the high level (guiding) hints. Learning from the right answer, just as learning from examples, requires that the student has an understanding of why the answer is right [7]. Also, the students seem to wait long before using the help facilities. Even when making a high number of errors per step.

In our research, we want to take the stages of Nelson-Le Gall into account when designing the pedagogical agent. We want to design a help system that is guiding

and will not use a glossary kind of approach. Eventually we want to be able to recognise common students' design mistakes in order to teach them by reflecting on these mistakes. Our future modelling tools should be equipped with smart feedback systems that guide beyond reflection on syntax errors. Rather they reflect on students' decision behaviour and trigger and improve self-evaluation.

10.2.3 Providing Feedback

Hattie et al. propose a model for applying the right type of feedback that has the greatest impact in a certain situation [49]. It is based on three questions: *Where am I going? How am I going? and Where to next?* The levels of feedback that they identify are:

- feedback about the task – performance on a task,
- feedback about the processing of the task – understanding how to do the task,
- feedback about self-regulation – self monitoring and regulation, and
- feedback about the self level (personal level) – evaluation and affect on personal level unrelated to the task.

Interesting to mention is that, in the analysis of the studies they report, the largest effect of the feedback was via feedback on the task. This effect was larger than that of praise, reward and punishment. For applying feedback Hattie et al. emphasise that timing should be considered. Also, they explain there is a choice between negative and positive feedback. Which can both have a positive influence on learning. For example, the choice depends on the level of feedback and the attitude a student has to the task (does the student *want* to do a (learning) task or does he/she *have* to do it?). Moreover, in classroom situations a safe environment should be created to make the learning caused by the feedback effective, e.g. students should feel safe to make errors.

Kalyuga explains that the cognitive architecture of our brain should be taken into account when designing interactive learning environments [63]. Our cognitive system can be easily overloaded when offering too much information to process. Kalyuga warns that poorly designed feedback can lead to split-attention. The students then are driven away from the original task. Providing feedback or guiding on demand is one of the examples that could help to reduce the amount of unnecessary cognitive load. Another suggestion is to provide feedback sequenced with an increase of provided details (amount of information) in the feedback message.

10.2.4 Pedagogical Agents

In general a pedagogical agent is an autonomous piece of software that supports learners by interacting with them. Pedagogical agents consist of a software agent (typically using some sort of AI) and a virtual character. The characters impersonate a person that has a specific role to the students, e.g. a tutor, teacher or learning companion. The character can be static (e.g. a picture) or dynamic in the form of an animation to benefit the interaction between the agent and the student and the motivation and engagement of the student.

Gulz and Haake argue that the visual and aesthetic aspects of the animated agents should be taken seriously as they have an important impact on the user *Look is too important to be left or just happen* [42]. In another study they provide an overview of the different roles an agent can have [43]. The distinguish between:

- More authoritative roles – tutor, coach, instructor etc.
- Less authoritative roles – competing co-learner, collaborating learning companion, etc.

In their studies Gulz and Haake emphasise that *pedagogical agents are not a digital monsters threatening to replace teachers – they are only a tool with the potential to help teachers in daily classroom activities and hopefully help them to personalise the learning during the everyday activities*.

Mørch et al. see an important role for pedagogical agents as conceptual awareness mechanisms in their Computer-Supported Collaborative Learning environments. [91] The propose a design space that supports the adoption of pedagogical agents in interactive learning environments. The mention four dimensions that support the design of a pedagogical agent:

- presentation – how an agent should present itself to a user
- intervention – when the agent should present itself to a user
- task – the student's task (work / assignment)
- pedagogy – the kind of information that is presented.

In a meta-analysis of 43 studies Schroeder et al. [120] showed that pedagogical agents have a small, yet positive significant effect on learning. The effect was higher for K12 students than post-secondary students. Dincer et al. [34] conducted a study on

multiple-pedagogical agents: the use of multiple types of agents. They report that pedagogical agent use has a very important effect, especially on academic success. Based on their findings they suggest the use of pedagogical agents in their computer assisted software.

The effect of the pedagogical agent on cognitive load is still being discussed. Schroeder did not find a significant difference between groups (with or without the use of an agent) related to their training efficiency or instructional efficiency in his experiment [120]. Müller-Wuttke et al. see the pedagogical agent as an instrument to reduce cognitive load [93]

In our design we combine the findings of the works mentioned above. We aim at an agent that:

- acts as a guiding tutor,
- evaluates the solution of the student based on a lecturer's solution,
- gives feedback on the different task related and self regulation levels, and
- takes a student's cognitive load into account.

10.3 Research Approach

In this section we explain our research approach. We explain our choice for design research. We explain how we iterated over different (design) research steps and how we analysed the data.

10.3.1 Research Method

For developing the feedback agent we applied a design research approach [54]. Because of the involvement of designing, implementing and evaluating a software artefact as part of the study, this approach was found most useful. Figure 10.1 shows the overall process of the study. The different phases are described in the following subsections. The design evolved by iterating over the requirements, design, implementation and evaluation phases. The improvements are discussed in Subsection 10.4.1

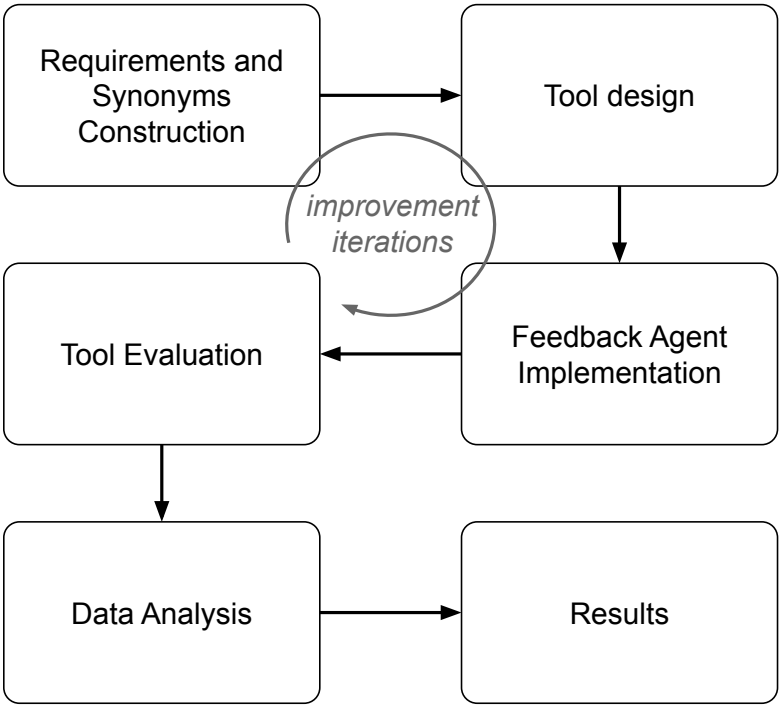

Figure 10.1: *The overall research approach for the feedback agent study*

10.3.2 Requirements Collection

First we collected the requirements for the construction of the feedback agent. To this end, we analysed the experiences of students designing UML diagrams from the study that was discussed in Chapter 6 [137]. In this study we asked students to record the questions that arose when facing a difficulty while performing a class design task in order to uncover common difficulties. The students created a class design for a game (see Appendix C for the game description). We used the game designs for further analysis of the research that is described in this chapter. The aim of the analysis was to investigate what particular software design mistakes students make in an assignment. These mistakes point to the topics that students could use feedback on during their tasks. The following requirements that the feedback agent should fulfil were collected based on the analysis of the designs of Chapter 6:

Naming and Principles

- Offer domain-level feedback on UML naming conventions.

Satisfying requirements

- Inform designers of missing interface classes.
- Provide advice on missing classes.
- Provide advice on missing attributes.
- Provide advice on missing operations.

Paradigm and generic design concepts

- Provide advice regarding inheritance.
- Feedback on general design principles.

10.3.3 Design of the Feedback Agent

The feedback agent was build as an extension for WebUML, our self made online UML editor (discussed in Chapter 4). For extending and future possibilities, such as connections with other sub modules or the integration in other systems, the agent is designed as a separate module. Figure 10.2 shows a high level overview of WebUML, including the feedback agent.

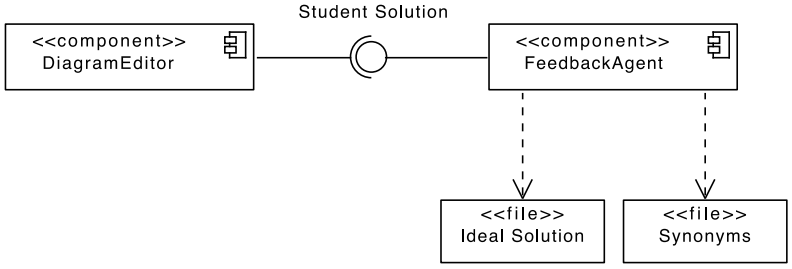

Figure 10.2: *The high level overview of WebUML including the feedback agent*

We choose to check a student's diagram against an ideal solution. Although we will continue exploring other approaches such as using machine learning, we choose to focus on a proven (less flexible) approach that delivered us a prototype for further investigation.

The feedback of the agent is shown to the user in a text balloon of an avatar. Using such a simulated human-like interface is common when using pedagogical agents.

In addition to the feedback agent a progress bar is shown at the bottom of the WebUML editor. The progress bar supports students' need for confirmation that was also found in Chapters 3, 7 and 8.

The general work flow for receiving feedback with the feedback agent is as follows:

1. The user presses a button for reviewing his/her design.
2. The feedback agent receives the data of the model that is kept by the main program.
3. The feedback agent evaluates the data and sends the evaluation back to the main program in the form of a feedback message. In addition, a progress value for the progress bar is also sent.
4. The main program sends the feedback message to the text balloon of the avatar and updates the progress bar.
5. The user reads the feedback message and possibly improves the design.

Figure 10.3 shows a screenshot of WebUML with the integrated feedback agent enabled.

The feedback is triggered 'on-demand'. This is done because of Nelson-Le Gall's second stage: decide to seek help. We wanted to examine the decision of asking for help rather than the reaction on a offer for help. In addition, letting the user decide is less intrusive than forcing help during modelling. When a student is in the need for help he/she has to press the 'help-button'. After pressing for help. the feedback agent is activated. The agent compares the student's solution with the 'ideal' solution that lectures prepare and store in a database file. This solution file also contains a set of synonyms. The synonyms make the comparison more flexible. It supports a reasonable flexible interpretation of the assignment text and textual representation of the student solution.

We investigated how to equip the feedback agent with the appropriate feedback messages. As feedback is a powerful tool, it can also have negative effect when not properly applied [49]. The feedback agent was designed to give feedback in sequential stages, not providing too much detail at once, in order to prevent extraneous load.

As mentioned in Section 10.2 Hattie et al. explain four levels of feedback [49]: Task Level, Process Level, Self-regulation Level and Self Level.

The feedback agent implements three types of feedback and for these we focus on the levels that are the closest to the actual design task, rather than on the general self-level. Figure 10.4 shows two moments of the avatar's feedback during a student's assignment process. Table 10.1 shows examples of feedback messages that are currently implemented in the feedback agent.

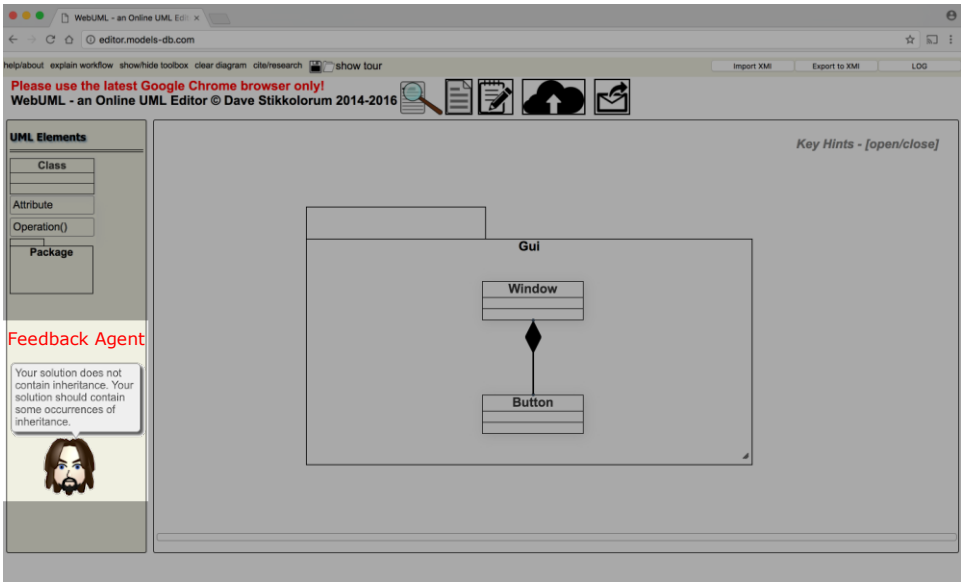

Figure 10.3: *WebUML including the feedback (highlighted) agent*

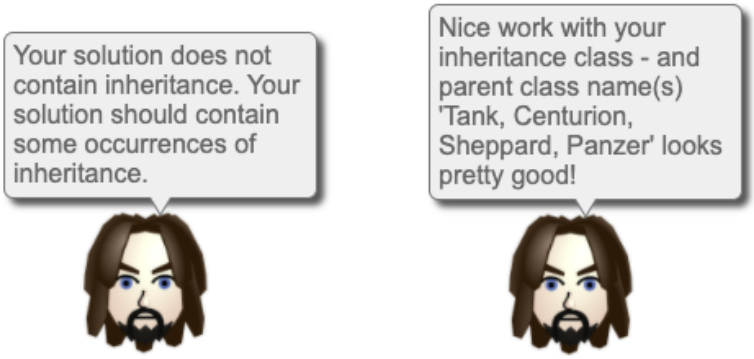

Figure 10.4: *left: feedback on missing elements, right: feedback on good progress*

Table 10.1: *Examples of feedback messages used by the feedback agent*

Feedback Message	Feedback Level
Your classes Tank, Centurion look good but you are still missing some important classes.	Task
Read the task carefully and see if you can find the remaining classes by identifying nouns, preferably complete singular nouns.	Processing
Class name(s) Tank looks good.	Regulatory

10.3.4 Implementation and evaluation

In this section we describe the evaluation steps we performed. We evaluated the tool by asking students to perform a class diagram task. We collected observations on their behaviour while designing and performed post-task interviews (Table 10.2). Based on the observations and interviews, we adjusted and extended the feedback agent. In total 1 first year and 4 third years students participated in the evaluation of our agent. The students were studying at a bachelors IT program. They had a basic understanding of UML and software design.

Table 10.2: *Interview questions during research design feedback agent*

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

10.3.5 Data analysis and Results

We collected data as part of the design research process. This data was obtained by observing and interviewing the users who trialled the system. This qualitative data was collected during subsequent iterations of the design process and was transcribed and coded.

10.4 Results

In this section we present our findings on the design of the feedback agent.

10.4.1 Design Improvements per Iteration

We started with an initial design and iteratively improved the feedback agent as discussed in Section 10.3. By observing the students during their task, analysing their solutions and listening to their feedback we improved the following:

- Iteration 1 – In the initial design the feedback was mainly focused on task level related feedback. This type of feedback seemed to induce students to use a trial-and-error strategy. Some students solutions lacked important classes, thus we extended the feedback agent with feedback about missing elements, for example missing classes, attributes or operations.
- Iteration 2 – We added feedback about deleting or editing of elements. Students are informed that classes, attributes or operations should be omitted because they should not be present in the design. Elements can be irrelevant to the assignment case and therefore should be omitted. Another example could be that elements should not appear as class in the design, but as attribute. Students can receive feedback about renaming elements when for example a mandatory parent class has wrongly named child classes.
- Iteration 3 – The last addition for the feedback agent was the suggestion to look for hints in the assignment text.

10.4.2 Students' Interviews

The coding and labelling process of the answers to the students' interview questions led to 3 dominant themes: i) the pedagogical agent, ii) the usability of the tool and iii) the user interface.

- Pedagogical Agent – Two main matters were reflected in the interview answers on the theme of *Pedagogical Agent*:
    1. Agent's Guiding Role is Appreciated – All participants appreciated the confirmation of classes, attributes or operations that we constructed in the

right way. Also, they valued the possibility of the feedback agent to advise them about the next steps in a task.

"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."

2. Need for More Help – 3 participants explicitly mention the need for more hints during the task. They suggest to give more information about elements such as missing and irrelevant classes. It could be due to the increasing feedback approach (Section 10.3) that students have this need. Also, when their design skills are low they ask for more information:

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

- Tool Usability – The students find the tool easy to use. This was mostly confirmed by observation during the tasks, but was also explicitly mentioned by a student. Another student appreciated the ability to move back and forth between the assignment text and the editor. During the experimental design of the research in Chapter 6 we decided about having the possibility to integrate the reading of the assignment text in the WebUML environment in order to let the student stay focused on the task.

  "It's quite easy to use"
  "I like the ability to move back and forth from the assignment text to editing mode"

  2 students mentioned the lack of the possibility of changing the size of a class, for example, when having long attribute names. 2 students mentioned that they want to be able to select more than one class for moving around the whole diagram in stead of moving one class at a time.

- User Interface – 3 students did not notice the progress bar. It should be improved or reconsidered. It could be due to a bad choice of colour that students mistake the progress bar for a scroll bar or just don't observe it at all.

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

  1 student seems to appreciate the progress bar and seems to use it in the way we design it:

  "It's helpful you know, if I were to do this without the progress bar, I would still have something missing e.g. attributes, or methods."

- Observations – Although the students recognised the benefits of using a feedback agent, in some cases they ignored the suggestions of the feedback agent and left the errors unchanged.

  Interesting to observe is that the students that had poor UML skills were more willing to change their design according to the suggestions of the feedback agent

than the students that had good UML skills. The students with prior UML design experience maintained their design although the feedback agent did not identify it as a good solution. They were confident about their solution to be also good.

10.5 Discussion - Strengths, Possibilities, Limitations and Future Directions

In this section we discuss the study of the feedback agent by discussing strengths, possibilities, limitations and future directions.

The participating students react very positive to the feedback agent. The most important strength of the tool is that the students indicate that they are helped by the agent to move forward in their task by receiving the help hints. In this sense the agent fulfils the role of a guide. Students mention that they appreciate the possibility to get confirmation about their progress from the feedback agent. This makes them more confident about the direction that the are heading. This is in line with our findings in Chapters 3, 7 and 8 in which students mention this and teaching assistants observe the same about confirmation of the progress of the task.

We observed that students that have poor UML knowledge and skills are more willing to use the feedback agent than more experienced students. In general it is logical that students don't need the agent when they are confident about their skills. To make the agent attractive for these students we could expand the feedback with messages that supports more in depth information that challenges students to achieve higher design quality. However, the challenge to trigger them asking for help in the first place will remain.

The progress bar was perceived as a helpful addition but its design needs to be modified. At the moment it can be mistaken with for example a scroll bar.

A limitation is the fact that some students ignore the feedback of the agent. In our case, the students that neglected the hints of the feedback agent were more experienced than the ones that seemed to follow the suggestion more closely. Also letting students do the task under observation could have influenced their behaviour. Maybe they did not want to look 'dumb' in front of the observer by following the suggestions. Also, because the detection of alternative solutions is a challenge, we have to accept the ignoring of the hints by the students.

In addition, in the future the introduction of disconfirmation hints could be explored. Hatie et al. [49] point out there is evidence that this can be more potent on the self level.

At the same time they show that only the self level is not effective. A combination of the self level and the task level is preferable. At the task level disconfirmation should be accompanied by corrective feedback to be powerful. Combined this could lead to feedback messages such as "you are not on the right track, you are missing important classes. Did you find all class candidates in the assignment text or did you maybe select them to be an attribute?".

We did not take into account irrelevant elements, such as an irrelevant class that is present in the student's class design. The missing of hints about irrelevant classes could have encouraged the trial-and-error behaviour of the participants. Hinting about irrelevant elements could have motivated them to analyse the text again or try to look at the problem from another angle instead of trying to 'find' the classes by asking the feedback agent for confirmation. The introduction of irrelevancy could stay a limitation in general.

It is known that design problems can have multiple solutions and thus could have classes that are being recognised as irrelevant by a feedback agent that makes this distinction. Having a large pool of synonyms related to the words in the assignment could be a good start. Maybe natural language processing should be involved. Further investigation is needed for this direction.

The feedback agent compares a student solution with an ideal solution, therefore it cannot handle multiple solutions in a broad sense. Although the introduction of synonyms helps to allow flexibility in the naming, this still leaves challenges when variations of design structures are made by students. Future research could focus on the introduction of design principles in the agents' diagnosis as measure for the quality of the design. The judgement of the design would then be quality driven in stead of a 'perfect' fit driven approach.

10.6 Conclusion

In this chapter we described our approach in designing a pedagogical agent for giving feedback to students while learning to make UML class designs. We explored the possibilities for addressing the challenges of guiding students in using the right semantics, satisfying the requirements of assignments and help them with design decisions.

We presented our feedback agent, a software module that extends WebUML. We used a design research approach to iteratively develop a pedagogical agent that uses 3 task related feedback levels. We aimed to overcome the above mentioned challenge by using the method of letting the feedback agent compare the student's work with an

ideal solution that is prepared by a lecturer.

Early findings from observing and interviewing a small group of students that tested the prototype of the feedback agent gives us preliminary insights into the benefits and limitations of this approach. The students appreciate the guiding role of the feedback agent: the way it confirms steps that were done in the right way and suggestions for next steps. Students mention they would like to see more help during their task. We observed that some students ignore the hints and leave errors unchanged.

Based on the comments of the students, it seems the agent can be well used in the role of a guide that supports students towards the fulfilment of their task. More experienced students sometimes did not follow the advice offered by the agent even though this could improve their diagram. Future research should investigate how this type of students can be motivated to use the feedback agent.

Although still immature, we see a future for the use of the feedback agent in our future improvements of WebUML and/or other interactive learning environments.