Title	インタレストベースの交渉を使用した自動チャットボット（NDLtutor）を通じたオープン学習者モデルにおける対話の役割強化
Author(s)	SULEMAN, RAJA MUHAMMAD
Citation	
Issue Date	2016-09
Type	Thesis or Dissertation
Text version	ETD
URL	http://hdl.handle.net/10119/13816
Rights	
Description	Supervisor:池田　満，知識科学研究科，博士

Doctoral Dissertation

Enhancing the Role of Dialogue in Open Learner Models through an Automated Chatbot (NDLtutor) Using Interest-Based Negotiations

SULEMAN RAJA MUHAMMAD

Supervisor: Professor Mitsuru Ikeda

School of Knowledge Science
Japan Advanced Institute of Science and Technology

September, 2016

ABSTRACT

Negotiation mechanism using conversational agents (chatbots) has been used in Open Learner Models (OLM) to enhance learner model accuracy and provide opportunities for learner reflection. Using chatbots that allow for natural language discussions has shown positive learning gains in students. Traditional OLMs assume a learner to be able to manage their own learning and already in an appropriate affective/behavioral state that is conducive for learning. This thesis proposes a new perspective of learning that advances the state of the art in fully-negotiated OLMs by exploiting learner's affective & behavioral states to generate engaging natural language dialogues that train them to enhance their metacognitive skills. In order to achieve this, we have developed the NDLtutor that provides a natural language interface to learners. Our system generates context-aware dialogues automatically to enhance learner participation and reflection. This thesis provides details on the design and implementation of the NDLtutor and discusses two evaluation studies. The 1st evaluation study focuses on the dialogue management capabilities of our system and demonstrates that our dialog system works satisfactorily to realize meaningful and natural interactions for negotiation. The 2nd evaluation study investigates the effects of our system on the self-assessment and self-reflection of the learners. The results of the evaluations show that the NDLtutor is able to produce significant improvements in the self-assessment accuracy of the learners and also provides adequate support for prompting self-reflection in learners.

Keywords: Intelligent Tutoring System, Open Learner Model, Affect & Behavior modelling, Metacognition, Interest-Based Negotiation

ACKNOWLEDGEMENTS

First of all I would like to thank ALLAH the Most Gracious, Most Merciful for the countless blessings bestowed upon me, without which I would have never been able to be where I am today.

I would then like to show my utmost gratitude to Prof. Riichiro Mizoguchi for his never ending support, guidance, patience and kindness. I will always be indebted for his undoubting support and continuous encouragement throughout my Doctoral studies.

Special thanks to Prof. Mitsuru Ikeda for all his generosity and support. Thank you for providing valuable feedback on the research and for the constructive arguments during our discussions.

I would also like to thank Prof. Tsukasa Hirashima, Prof. Ho Tu Bao, Associate Prof. Hyunh Van-Nam, Associate Prof. Takaya Yuizono and Associate Prof. Hideaki Kanai for their comments and valuable feedback which helped me to improve the quality of this thesis.

Thank you to all the members of the Ikeda laboratory for their kindness. I would like to specially thank Ikue Osawa and Chen Wei for all their help which made life easier in JAIST. I would also like to thank Fernando and Tamires for their love and support.

Last but not the least; I would like to thank my beloved family; my parents, my brothers and sister and my nephews and nieces. Thank you for all the support, prayers and love.

TABLE OF CONTENTS

1. INTRODUCTION......**1**

1.1 Motivation......2

1.2 Objective......4

1.3 Approach......5

1.4 Organization of the Thesis......5

2. RELATED WORK......**7**

2.1 Learner Models......9

2.2 Classes of Open Learner Models......10

2.3 Interest-Based Negotiation......14

2.4 Affect & Behavioral Modeling......14

2.5 Dialogue-Based Tutoring Systems......15

3. ANALYSIS & DESIGN......**18**

3.1 Scope Definition......18

3.2 Negotiation-Driven Learning......18

3.3 Problem Analysis......19

3.3.1 Generating Dialogues for NDL......19

3.3.2 Facilitating Metacognitive Skills......20

3.3.3 Identifying Learner's States......22

3.3.4 System Architecture......24

3.4 Requirements Analysis......27

3.5 Wizard-of-Oz Experiment......29

3.5.1 Results......33

3.5.2 Classifying Student's Affective and Behavioral States 34

3.5.3 Inputs Related to Affective States 38

3.5.4 Inputs Related to Behavioral States 39

3.5.5 Revisiting Questions set for the Experiment 41

4. IMPLEMENTATION - NDLtutor 45

4.1 Overview..46

4.2 Database Structure 48

4.2.1 Domain Structure 48

4.2.2 User Utterance Library 50

4.2.3 System Utterance Library 52

4.2.4 Rules Library 52

4.2.4.1 Feedback Rules 52

4.2.4.2 Dialogue Move Rules 53

4.3 Natural Language Understanding 56

4.3.1 Natural Language Understanding in NDLtutor 58

4.4 Functional Modules 63

4.4.1 NLPE Class...63

4.4.2 Dialogue_Manager Class..64

4.5 Interface and basic functionality of NDLtutor...65

4.6 Phases of NDL 66

4.6.1 Initialization Phase..67

4.6.2 Domain Discussion Phase...67

4.6.3 Reflection Phase 68

4.7 Example Interaction....69

Initialization Phase....70

Domain Discussion Phase....72

Reflection Phase....76

5. EVALUATION 1 - DIALOGUE MANAGEMENT CAPABILITIES 79

5.1 Participants....79

5.2 Method....79

5.3 Learner Interactions....80

5.4 Results and Discussion....85

6. EVALUATION 2 – PEDAGOGICAL IMPLICATIONS....90

6.1 Participants....90

6.2 Method....90

6.3 Results....92

7. CONCLUSIONS & FUTURE WORK....105

7.1 Summary....105

7.2 Contributions....107

7.2.1 Contribution to Knowledge Science....108

7.3 Limitations & Future Work....110

References....114

Appendix A: List of Publications....123

Appendix B: NDLtutor Dialogue Logs (Evaluation 1)....125

Appendix C: NDLtutor Dialogue Logs (Evaluation 2)....136

LIST OF FIGURES

Figure 1: Research themes in the NDL Paradigm 7

Figure 2: Sample NDL dialogue (Reflection Phase) 21

Figure 3: NDL System Architecture 25

Figure 4: Response Library 32

Figure 5: Occurrences for each Affective State 38

Figure 6: Occurrences for each Behavioral State 40

Figure 7: NDLtutor Workflow……………………………………………………..47

Figure 8: NDLtutor Steps for Classifying Learner Input 61

Figure 9: NDLtutor Interface 66

Figure 10: Knowledgeable Learner Interaction with NDLtutor 81

Figure 11: Less knowledgeable learner Interaction with NDLtutor 82

Figure 12: NDLtutor dialogue excerpt showing system's adaptation to the learner's response patterns 84

Figure 13: NDLtutor dialogue excerpt showing system's confirmation of the student's confidence in his response 89

Figure 14: Self-assessment inaccuracy Before & After Negotiation with NDLtutor 94

Figure 15: Number of Topics with discrepancy Before & After Negotiation with NDLtutor 95

Figure 16: Self-Reflection scores (in ascending order) of individual students across all sessions 100

LIST OF TABLES

Table 1: List of selected Affective & Behavioral States of learner in NDL.................... 23

Table 2: Sample Rule for wizard to select system response...32

Table 3: Examples of Affective states corresponding to user inputs............................... 39

Table 4: Examples of Behavioral states corresponding to user inputs 40

Table 5: User Utterance Categories with sample sentences ……………………………50

Table 6: Post-Experiment Survey results.. 86

1. INTRODUCTION

The paradigm of Open Learner Models (OLM) was introduced in Intelligent Tutoring Systems in order to involve the learner in the overall learning experience (Bull, S. & Pain, H., 1995; Dimitrova, V., 2003). OLMs generate the Learner Model (LM) of a learner by diagnosing their knowledge during their interactions with the system (VanLehn, 1988). This is achieved by evaluating the learner's answers to a series of questions on a particular topic or domain. Previous LMs were encapsulated from the learners and were only accessible to the system. OLMs externalize the contents of the LM to promote independent learning. This is done in order to provide transparency and increase learner's trust in the system (Bull, S. & Judy K., 2010). Negotiated OLMs achieve this by maintaining separate belief bases for both the learner and the system. The term *belief* here is defined as the "confidence in one's abilities or knowledge". The learner is allowed to inspect (view) and edit their own belief base however they can only inspect (view) the belief base of the system. Negotiation mechanisms are used to resolve any conflict (difference) that might occur between the learner's belief base and that of the system. The result of this negotiation is used to update the LM accordingly.

Different approaches to negotiation have been deployed by previous fully negotiated OLMs which include menu-based interfaces (Bull, S. & Pain, H., 1995) and conceptual graphs (Dimitrova, V., 2003). Conversational agents or chatbots were introduced to allow for more flexible and naturalistic negotiations (Kerly, A. & Bull, S. 2006). The natural language interface provided by a chatbot (Kerly, A., Ellis, R., Bull, S. 2008) improves the quality of dialogues by easing the communication between the learner and the system. The use of a chatbot yielded positive learning gains and was successful in increasing self-assessment accuracy (Kerly, A., Bull, S., 2008). Through a successful trial with different age groups the research was able to identify the novelty and effectiveness of using a chatbot to discuss the LM content with the learners in the context of OLMs.

Research has shown that expert human-tutors are successful as they try to engage students according to their affective and behavioral states, which provides a sense of empathy and encourages learner involvement (Lepper, Mark R., et al., 1993). We believe current OLM implementation can be largely enhanced by explicit use of the information regarding such states of a learner to control the flow of the dialogue.

Improving the metacognitive abilities of the learner has always been a key role of OLMs (Bull, S., & Kay, J., 2013) and these systems have shown to be successful in promoting self-reflection. However, there is no explicit mechanism in current OLMs to scaffold the metacognitive processes. Self-reflection is implied implicitly, i.e. how the learner is reflecting or evaluating themselves is left on the part of the learner. The system does not explicitly involve the learner into a discussion that can motivate them to practice these skills more actively.

1.1 Motivation

Allowing the learner to edit their belief base results in scenarios where the learner's belief about their own knowledge is different from that of the system. Such events trigger an interaction where the system tries to negotiate the changes made by the learner in their belief base in an effort to remove the difference of beliefs between the learner’s belief base and the system’s belief base. The aim of this negotiation has been mainly used to increase the accuracy of the system's LM and enhance the role of the learner in the construction and maintenance of their LMs, which increases learner reflection (Bull, S. & Pain, H., 1995; Kerly, A., Ellis, R., Bull, S., 2008; Dimitrova, V., 2003).

A major research issue for Intelligent Tutoring Systems has always been related to maximizing learner engagement and control over their learning processes. During the interaction with an ITS, the learner might feel that he has improved his knowledge about a certain topic. At this point the learner should be allowed to inform the system about the changes in his knowledge level. This ability of the learner to inform the system about the change in his knowledge level inspires us to envision a new learning paradigm in which the system suspends the normal course of tutoring and engages in a dialogue with the learner about their belief of the knowledge of a specific topic. Contrary to the common practice in previous fully-negotiated OLMs which confined the scope of this negotiation by only allowing the learner to prove his claim by giving another MCQ test, we find it more effective if the system can engage the learner in a dialogue about a specific domain concept in a natural language environment. Therefore, instead of merely testing the learner's knowledge, the system should be able to help the learner construct their knowledge and fill any knowledge gaps or remove any misconceptions that might arise during this discussion. Moreover, we believe introduction of an explicit reflection phase dialogue at the end of each discussion would provide learners with a unique opportunity for dialogue-driven learning which is different from normal learning based mainly on problem-solving in conventional ITSs. Once this dialogue session has been completed, the system can resume the normal course of tutoring.

In the context of current OLM implementations negotiation is mainly used as a tool to improve the accuracy of the Learner Model while promoting self-reflection is only implied implicitly. The motivation of this research is thus to make use of the negotiation between the learner and the system as a unique opportunity for not only *learning* but also for *explicitly promoting metacognitive skills* in them.

1.2 Objective

A conflict may occur because the learner may be confused about their knowledge, or simply have a misconception which leads them to change their LMs. The system challenges the change made by the learner and requires them to justify himself. This creates an interesting prospect to involve the learner into a discussion about their belief and what led them to believe so. Humans become stronger advocates of their beliefs once they are challenged, and are intrinsically motivated to defend their beliefs (D. Gal, D.D. Rucker, 2010). This provides an excellent opportunity to involve an intrinsically motivated learner in a deep learning dialogue which not only discusses the domain knowledge but also encourages them to assess the discussion to promote self-reflection. By capturing this opportunity and making use of the context, we believe we can come up with a new learning paradigm. Based on this, the objectives of this research include:

1. Proposition of Negotiation-Driven Learning (NDL) which exploits the above mentioned opportunity.
2. Proposition of a conversational agent (chatbot) named NDLtutor that uses *Interest-Based Negotiation* in OLMs to engage the learners in a natural language dialogue targeted towards deeper learning.
3. Investigation of the impact and effects of NDLtutor on dialogue management for promoting metacognitive abilities in learners.

1.3 Approach

Learning is maximized by proactive participation of learners; we believe that such a context is ideal to engage a learner in a dialogue that explicitly targets the metacognitive skills of the learner and provides them the scaffolding to utilize and enhance these skills. Research on the effects of using learner's affective and behavioral states to shape negotiations has shown a positive impact on the overall learning gains (Du Boulay, Bennedict, et al., 2010; Fredrickson, BL. 1998). This has not been previously studied in the context of OLMs. In NDL we aim to exploit the utility created by the occurrence of a conflict by engaging a learner in a natural language dialogue according to their affective and behavioral states and promote metacognitive skills in them through reflective dialogues and self-assessments. This thesis aims at exploring the effects of negotiation in the context of fully-negotiated OLMs and its impact on the learning gains of the students, the use of behavioral modeling to understand the motivational state of the learner, and the potential of a deploying a conversational agent (chatbot) that uses Interest-Based Negotiation (IBN) to allow for a more open dialogue between the learner and the system.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows; Section 2 provides the background of our research in the context of the related work and literature review. Section 3 defines the scope of our research and introduces the paradigm of Negotiation-Driven Learning. Here we provide the outline of the system architecture and the details of the design of dialogues in NDL. We then describe the Wizard-of-Oz experiment which is used for selecting learner's affective & behavioral states for our system as well as generating system libraries for rules of dialog management and NLP matters. This is followed by the discussion on the objectives achieved during the WoZ experiment. The Section 4 introduces our implementation of the NDLtutor which is our realization of the NDL paradigm. Here we provide the description of the domain structure and the functional modules of the NDLtutor. Then we give a description of the different phases of NDL and then provide an example dialogue to illustrate how our envisioned system interacts with the learner. Section 5 discusses the first evaluation study which evaluates the dialogue management capabilities and validates the affective & behavioral states that were selected for our system. In Section 6 we provide an in-depth discussion on the second evaluation study that explores the effects of our system on the self-reflection and self-assessment skills of the learners. In Section 7 we conclude the thesis by providing a summary and highlighting the contributions made by our research to the artificial intelligence in education society and knowledge science and then briefly discuss the limitations together with future work.

2. RELATED WORK

Fig. 1 shows the research themes that motivate and influenced the research on the Negotiation-Driven Learning paradigm. This section provides an overview of these research areas and how they contribute to the development of NDL. Open Learner Models emphasize the active involvement of the learner in the process of improving the accuracy of the Learner Model. OLMs utilize different strategies of negotiation in order to allow the learner to discuss their LM with the system. These strategies are mainly differentiated on the amount of control the learner and the system have over the course of the dialogue. Fully Negotiated OLMs allow a larger degree of control to the learner as compared to other negotiation strategies by deploying interaction symmetry that provides the same dialogue moves to both the system and the learner (Bull, S. & Vatrapu, R. 2012). Allowing the learner to change their belief base gives them a sense of control over the process while having the ability to defend their beliefs against the system and ask for justification from the system inculcates a sense of trust in them.

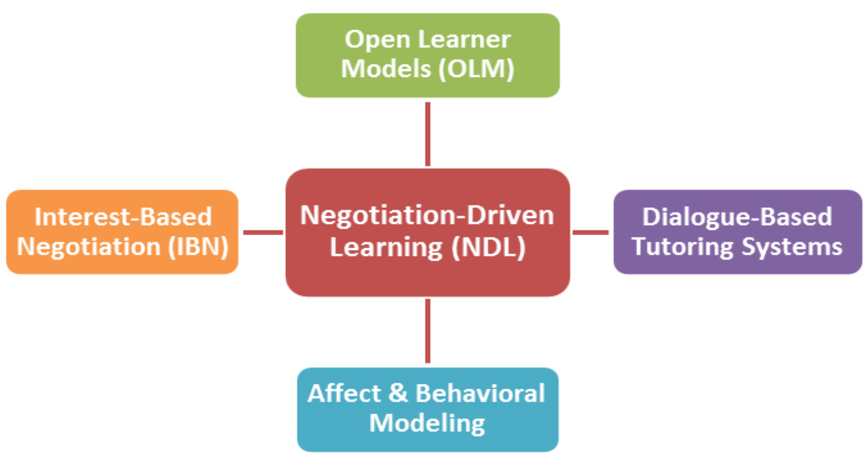


Fig.1. Research themes in the NDL Paradigm

Allowing the learner to interact with the system about his LM, opens up new doorways of interaction possibilities and diagnosis. The interactive nature of the dialogues provides an opportunity to promote reflective thinking in the learners. A very important aspect of OLMs has always been the active promotion of metacognitive skills in the learner (Bull, S., & Kay, J., 2013). It has been documented that students who have better metacognitive skills perform much better than students who have weak metacognitive skills (Swanson, H. L., 1990; Schraw, G. & Dennison, R.S., 1994). Since metacognitive skills do not have any observable manifestation, such skills are hard to acquire and gauge. However continuous stimuli can lead the learner into learning to use such skills more actively so that these skills are automatically used by the learner while they are learning. Improvement in the metacognitive skills of the learners has mostly been implied implicitly. Externalization has been considered as one of the major sources of self-reflection in learners. When they are able to view their LM and reflect upon their knowledge level. However, this self-reflection remains implicit and OLMs do not provide a clear platform to the learner to keep a track of their metacognitive abilities.

ITS systems are modelled to replicate expert or semi-expert tutors, since expert tutors have shown to have the maximum learning gain in learners. An important aspect of the expert tutors teaching tactics is the ability to react to the student's affective and motivational states (M. T. H. Chi et al, 2001). A learner's affective and behavioral states play a vital role in the outcome of their interaction with the system as a confident, interested, and motivated learner would interact very differently from a learner who is not confident, uninterested or demotivated. For an automated system to be able to replicate an expert tutor’s empathy, it needs to be able to classify the learner’s interactions as a possible outcome of a mental state. Intensive research on this effect would contribute to advancement of the field of OLMs.

2.1 Learner Models

Intelligent Tutoring Systems use Learner Models to provide adaptive and personalized content to the learners. The system diagnoses the learner’s knowledge during its interactions with the learner and uses this information to infer the corresponding learner model (VanLehn 1998). The learner model represents the current state of the learner’s knowledge. The basic tasks for any learner model include (Wenger 1987):

1. Storing information about the learner’s knowledge and expertise about a particular domain. This information allows the system to compare the learner’s knowledge with that of an expert module to generate evaluations and highlight area of weakness.
2. Representation of the learner’s knowledge level that allows for an insight into incorrect knowledge and misconceptions held by the learner.
3. Accounting for data by analyzing the information available to the system to generate the diagnosis for the learner. Such diagnostic process can vary depending upon the kind and amount of information available to the system.

Traditionally the learner model was encapsulated from the learner and only visible and available to the system for adaptive tutoring. It has been argued and that involving the learner in the process of constructing and maintaining their learner model not only promotes learner engagement but also has positive effects on their metacognitive skills (Bull & Pain, 1995; Kerly, A., Ellis, R., Bull, S., 2008; Dimitrova, V., 2003).

2.2 Classes of Open Learner Models

We can identify different classes of OLMs according to the level of control they provide to the learner over the LM. The learner's level of control can be defined as the learner's capability to change the contents of the LM. According to this specification, OLMs can be classified as:

1. *Inspectable*: An inspectable OLM can be considered as a read-only or view only OLM. The LM is completely controlled by the system and is only available to the learner for viewing. The learner has no right to change the contents of the LM directly. The learner can answer questions related to the domain in order to have their model updated. The externalization of the LM has shown to increase learner involvement and promote self-reflection and planning skills (Bull, S. & Judy K., 2010). All OLM implementations are considered inspectable since they allow the learner to view their learner models in one form or another.
2. *Co-operative*: These models allow the learner and system to jointly construct the learner model. The system asks the learners to provide complementary information required for the modeling process (Beck, Stern, & Woolf, 1997).
3. *Challenge*: These OLMs allow the learner to challenge the model generated by the system. EI-OSM (Zapata, R. et al., 2007) is one such system based on Toulmin's model of argumentation (Toulmin 1958). EI-OSM uses claims, data, warrants, backing and rebuttal to allow learner to add new arguments with supporting evidence. A teacher has the authority to determine which evidence has

the highest strength and the evidence supported by the teacher is considered stronger than the unapproved evidence provided by the learner.

Another OLM that allows the learner to challenge the system is xOLM (Van, L. et al., 2007). The learner is allowed to view the model and select the topic for discussion. The system provides is justification for the topic and the learner are provided with three options; 1) agree 2) disagree and 3) move on, to continue the interaction. If the learner agrees with the system, the system's beliefs are reinforced. In case of a disagreement the learner has to provide further information which is used to diagnose the model. Move on allows the learners to end the discussion with the system.

4. *Add-Evidence*: These OLMs allow the learner to provide additional evidence to be considered in the modeling process. ELM-ART (Weber & Brusilovsky, 2001) is an OLM that allows the learners to inspect and edit the contents of their learner model. ELM-ART is implemented as an adaptive interactive textbook where the learner informs the system about their knowledge by providing evidence to support their claim. Evidence can be in the form of answering questions, taking tests or performing tasks.

   Another OLM that allows the learner to provide evidence is TAGUS (Paiva, Self & Hartley, 1994). The learner can inform (*tell)* the system about the new evidence which is then analyzed by the system to take appropriate action.

5. *Editable*: Learners have full responsibility and control in editable OLMs. They are allowed to edit their learner model when they deem necessary without the intervention of the system. The system may offer some information regarding its

belief base which can be neglected or overridden by the learner. The changes made by the learner are directly reflected in the system's belief base which alters their learner model. Some examples of OLMs in this class are; C-POLMILE (Bull, S. & McEvoy, 2003), SASY (Czarkowski, Kay, & Potts, 2005) and Flexi-OLM (Mabbott & Bull, S., 2006).

6. *Persuaded*: Persuaded OLMs also allow the learner to change their learner models but they are required to demonstrate their competency before the system can agree with the changes they made. The system uses questioning techniques to analyze the learner's knowledge level and validate their claim. If the learner is not able to justify the change they made, their changes are rejected by the system and the learner model remains unchanged. Flexi-OLM (Mabbott & Bull, S., 2006) is an OLM that falls in this category.

7. *Negotiated*: Negotiated OLMs allow for a more collaborative approach towards constructing and maintaining the OLM. Negotiated OLMs use a separate set of beliefs for the learner and the system. The negotiation process is used to resolve the conflicts (discrepancies) between these sets of beliefs. There is an interaction symmetry which provides both the learner and the system with equal rights of interaction. The basic negotiation protocol allows for; ask for justification, provide justification, challenge justification, reject justification, provide proposal, accept proposal or reject proposal.

Mr. Collins (Bull, S., Brna, P., Pain, H., 1995) is the first fully negotiated LM which focuses on the discussion of the LM between the learner and the system. Mr. Collins uses a menu-based discussion which allows learners to challenge and respond to the system.

While Mr. Collins has been shown to promote learner reflection, the negotiation method used can be considered as restrictive. STyLE-OLM (Dimitrova, V., 2003) is another fully-negotiated system that allows learners to discuss their LM with the system. STyLE-OLM is proposed based on the idea that interaction is a stimulus for reflection. The dialog is constructed as a conceptual graph that allows the learner to see the explicit connections between the different arguments. However, some learners might find using the graphical interface difficult or distracting.

CALMsystem (Kerly, A., Ellis, R., Bull, S., 2008) addresses the problem of using menu selections and conceptual graphs for young learners. In order to provide an easier way to communicate with the system, CALMsystem proposes the use of natural language dialogue. CALMsystem follows the negotiation options provided by Mr. Collins and uses a chat-bot to provide a natural language dialogue. CALMsystem utilized the Lingubot™ (Creative Virtual, 2007) technology to build the chatbot. Domain-independent utterances do not affect the course of the dialogue which can be restrictive in a natural language dialogue system. CALMsystem laid the foundations of using natural language conversational agents in the context of OLMs. The background and guidelines provided by CALMsystem formed the basis of this research.

2.3 Interest-Based Negotiation

Negotiation is a vital form of human interaction which ranges from basic information exchange to more complex cooperation or coordination activities. As computer systems evolve to become autonomous agents, it was inevitable for such systems to be able to conduct a negotiation of their own in an automated way. Automated negotiation has found much interest and success in the field of e-commerce where automated autonomous agents negotiate over resources (tangible assets).

Interest-Based Negotiation (IBN) (Fisher, R., Ury, W., 1983) has gained attention from the research community since it provides a good alternative to Position-Based negotiation where all agents are considered adversaries. It is also known as win-win negotiation as all parties try to create a mutual gain. IBN allows the parties to reveal their underlying interest by specifying new information during the course of the dialogue. This information can be used to decide an alternate strategy in real-time which makes IBN more responsive. Since learning is a process of exchanging ideas and understanding problems, IBN seems much more suited for educational systems, as proposed in (Miao, Yuan, 2008). There are no current implementations of OLMs that have tried to utilize IBN as the main negotiation approach.

2.4 Affect & Behavioral Modeling

Research has shown that expert human tutors have a higher impact on learning than novice tutors and ITSs (Lehman, Blair, et al., 2008). This is not only due to the pedagogical strategies employed by such expert tutors but is also deeply rooted in the emotional (affective) and motivational (behavioral) strategies such tutors employ to engage the learners in leaning (Du Boulay, Bennedict, et al., 2010). Affect and behavior are closely entwined in a bi-directional relationship. Moreover a learner may not only experience a positive affective or behavioral state, but also a negative state. Such a negative state might even be necessary for a learner to be engaged in the process of learning. Understanding the state the learner is in can allow a system to be more empathetic towards them which leads to higher levels of engagement. It has been argued that while an exact estimation of a specific state might not be possible or even required, an approximation of these states can be as helpful in continuing the learning process. The terminology of “caring systems" encompasses such systems which are meta-affectively and meta-cognitively aware. Our research aspires to inherit such attributes to provide adequate support the learners to promote their cognitive and meta-cognitive skills.

2.5 Metacognition

Metacognition is commonly defined as “thinking about one’s own thinking” or “what we know about what we know” (Puntambekar and Du Boulay, 1999). It involves understanding what one already knows, comprehending the task of learning and what skills would be required to solve it, the ability to monitor one’s actions, planning, debugging and evaluation (Schraw, G. & Dennison, R.S., 1994; Taylor, 1999). Metacognition definitions generally include 2 components (Schraw, G. & Dennison, R.S., 1994; Taylor, 1999; Flavell, 1987):

1. Knowledge about Cognition: Includes declarative knowledge, procedural knowledge and conditional knowledge to aid the reflective aspect of metacognition.

2. Regulation of Cognition: Includes planning, information management, comprehension monitoring, debugging strategies and evaluation processes which support in the control aspect of learning.

2.6 Dialogue-Based Tutoring Systems

ITS systems have come a long way from having simple human-computer interfaces to adopting conversational interfaces. Apart from the conventional text display and graphics such systems employ an automated conversational agent that is able to speak to the student using synthesized speech accompanied by facial expressions and gestures. This makes the learner’s experience more interactive and has also been shown to increase engagement.

Dialogue-based tutoring systems have deployed different forms of strategies to maximize learning. Knowledge construction dialogues (KCD) were used to encourage students to infer or construct the target knowledge in the ATLAS system (Freedman, R. 1999). KCDs connect principles and relate them to common sense knowledge to help students to discuss their knowledge. ATLAS was originally developed for CIRCSIM tutor and also provides a natural language interface to the learners. Immediate feedback strategy was employed in ANDES (Gertner and VanLehn, 2000; VanLehn, 1996) to help college and high-school physics students to do their homework problems. ANDES highlighted the use of real-time hints and feedback to help student solve given tasks.

One of the most successful systems in this category has been AutoTutor (Graesser et al., 1999, Person et al., 2001). It is an ITS that provides a natural language dialogue to interact with the learner. AutoTutor provides the learner with an interactive agent that speaks out the question in addition to displaying the text on the screen. AutoTutor engages the learner in a deep reasoning dialogue which requires the learner to provide comprehensive explanations. Autotutor's strength lies in its ability to handle learner responses during the course of the dialogue. Autotutor uses advanced statistical NLP techniques such as Latest Semantic Analysis (Graesser, A. C., et al, 2000) to analyze learner response and classify responses into corresponding speech acts. The role of AutoTutor is to act as a teacher and teach/construct knowledge.

3. ANALYSIS & DESIGN

3.1 Scope Definition

This thesis introduces a new learning paradigm of Negotiation-Driven Learning which allows a learner to interact with an ITS in a natural language interface. The research finds its motivation in different research fields i.e. Natural Language Processing, Affect & Behavioral Modeling, Dialogue-based Tutoring, and endeavors to combine best practices that have only been used separately in existing OLMs to develop an independent OLM that is capable of engaging learners in dialogues that promote metacognitive skills in them.

3.2 Negotiation-Driven Learning

This research proposes a new learning paradigm of Negotiation-Driven Learning which aims at *enhancing* the role of negotiations in OLMs to facilitate constructive learning. When a learner is involved in a learning exercise, they are not only learning something new, but they are also implicitly involved in learning how to learn. More often than not they are more inclined towards executing well-practiced strategies rather than monitoring themselves. NDL aims at encouraging learners to use these metacognitive skills more actively and effectively.

NDL acts as a component of the ITS which is triggered when a conflict between the beliefs of the system and the learner occur. During its interaction with the learner the system tries to understand why the learner holds a certain belief (cause of the conflict) and tries to help them understand why it might not be true. The system uses the information about the learner's affective and behavioral states to control the flow of the dialogue to ensure maximum engagement. An NDL dialogue session is concluded when the learner is able to defend their claim, or shows an understanding of their incorrect belief by accepting the system's justification/proposal. The system's LM is updated with the outcome of the dialogue and the ITS resumes the normal course of tutoring.

3.3 Problem Analysis

3.3.1 Generating Dialogues for NDL

NDL allows learners to interact with the system in a natural language interface. In order to accomplish this, the system follows the negotiation protocol proposed in (Miao, Yuan, 2008) to allow the learner to provide justification of their change. This protocol is consistent with other protocols that have been defined and used in previous versions of OLMs (Bull, S. & Pain, H. 1995, Dimitrova, V., 2003, Van Labeke, N., Brna, P. & Morales, R., 2007). The system asks the learner to justify the changes they make to their belief base. If the justification provided by the learner contains an incorrect idea, the system rejects it. If the justification provided by the learner contains an “assertion”, the system can ask for more information to accept it or provide a proposal to the learner to continue the dialogue further. The system initiates a reasoning process which is used to understand the motivation behind the change made by the learner. The system and the learner have equal rights to ask for further information; accept or reject a justification provided by the other party; therefore the system needs to be capable of deploying an alternative strategy in case a learner rejects its proposal/justification.

3.3.2 Facilitating Metacognitive Skills

Facilitating metacognitive skills has been the core of recent research on ITSs and OLMs (Bull, S., & Kay, J., 2007, Mitrovic, A. & Martin, B., 2002, Mitrovic, A. & Martin, B., 2007). Learners who are good at using their metacognitive skills perform better than those who are unable to use such skills actively (Garner, R. & Alexander, P.A., 1989, Schraw, G. & Dennison, R.S., 1994). NDL emphasizes the importance of actively using and enhancing these skills during an interaction between the learner and the system. Fig. 2 shows the dialogue session after a few dialogue moves encompassing domain-specific reasoning. Once the learner is able to answer the domain specific questions to an acceptable standard, the system requires them to summarize their answers and reflect upon their discussion with the system. This is done to reinforce their understanding and encourage self-assessment.

```
s133022-019_16_11_2015.txt - Notepad
File Edit Format View Help
NDLtutor:      Great!
               Let's reflect upon our discussion now!
NDLtutor:      On a scale of 0  to 10 (0:minimum, 10:maximum) how would you rate your first answer?

Student:       7

NDLtutor:      Ok. But your initial answer did not have a high score
NDLtutor:      How is your last summarized answer different from your first answer?

Student:       my first answer wasnt complete. my last answer looks good

NDLtutor:      Ok. Your initial answer was incomplete and lacked most of the concepts. Your final answer scored well as it
               is complete and covers all the aspects.
NDLtutor:      What were the concepts that you missed?

Student:       LIFO?

NDLtutor:      You're initial answer was missing the following concepts: LIFO, PUSH(), POP()
NDLtutor:      Did you encounter any misconceptions?

Student:       no i didnt

NDLtutor:      Is there anything still unclear?

Student:       no

NDLtutor:      Great!
NDLtutor:      What aspects of this topic do you think you need to concentrate more on?

Student:       i need to concentrate on remembering the concepts

NDLtutor:      Do you think you will be able to discuss this topic with a peer/firend now?

Student:       yes i think so

NDLtutor:      Are you sure? You don't seem confident in your answer.

Student:       yes i can
```

Fig. 2. Sample NDL dialogue (Reflection Phase)

The dialogue session in Fig. 2, highlights a major feature of NDL that distinguishes our approach from the current implementations of OLMs. The system engages in a domain discussion if the learner is unable to justify the change they made in their belief base. The domain discussion phase is used to analyze how much the student knows about a specific topic. If a learner is more knowledgeable or has improved/increased their knowledge they are able to answer the question within the first attempt. This provides the system with the information about their knowledge level in the topic. For less knowledgeable students who are not able to answer the question according to the defined standard (criteria), the system engages in a series of funneling questions in order to understand their level of understanding/knowledge of the topic. For such students, at the end of the domain dialogue session, the system explicitly encourages them for self-assessment by asking them to reflect upon the past interaction and evaluate how the discussion helped them formulate their final answers.

3.3.3 Identifying Learner's States

All ITSs aim to engage learners to maximize learning; however a learner's engagement highly depends upon the affective and behavioral state they are in (Lehman, Blair, et al., 2008). If a learner is in some sub-optimal state, the system needs to diagnose such states in order to help a learner move into an optimal state that is more conducive to learning. When a learner is in an optimal state of learning, they are more focused and learn better. Hence the system needs to ensure that such a state is maintained. There is an abundance of literature on modeling affect and motivation with varied views (Afzal, S. and Robinson, P., 2011, Burleson, W. & Picard, R. 2007, Conati, C. & Maclaren, H. 2009, Sidney D'mello and Art Graesser. 2013, Woolf, B. et al., 2010). However it is agreed that an exact estimation of such states is not required in practice as the main focus of an ITS is to improve the cognitive state of a learner, and the knowledge about these states support the system in its reasoning process (Du Boulay, Bennedict, et al., 2010) .

The process of learning requires the learner to be interested, motivated and confident to engage in a productive discussion with the system. Table 1 shows a list of Affective & Behavioral states that were selected to be used in NDL to model the affective/behavioral state of the learner. These states have been selected from previous research on the subject (Lehman, Blair, et al., 2008; Du Boulay, Bennedict, et al., 2010), and they provide a good approximation of the learner's mental state. How these states were shortlisted will be discussed in the experiment section of the paper. The precision of modeling these states is not of principal importance, but an approximation of these states can allow the system to engage the learner more actively.

Table 1

List of selected Affective & Behavioral States of learner in NDL

Affective States	
CONFUSED	Poor comprehension of material, attempts to resolve erroneous belief
FRUSTRATED	Difficulty with the material and an inability to fully grasp the material
ENGAGED	Emotional involvement or commitment
Behavioral States	
CONFIDENT	The feeling or belief in one's abilities or qualities
INTERESTED	Wanting to know or learn more about something
MOTIVATED	Having a motive or incentive to perform an action

Affective states are related to emotions or feelings and therefore are more prominent during the domain-independent discussions where learner responses are generally influenced by how they are feeling. On the other hand behavioral states are related to the interaction of the learner and hence domain-dependent discussions are mostly influenced by the behavioral states of the learner. Metacognitive states of a learner are more difficult to gauge as they are implicit in nature and are used subconsciously. However, understanding the *context* of a dialogue can help in estimating the approximate metacognitive state of the learner. Further discussion about these states will be continued in the Wizard of Oz experiment section.

3.3.4 System Architecture

We propose the use of Interest-Based Negotiations (IBN) (Fisher, R., Ury, W., 1983) in NDL. IBN aims at exploring underlying interests of the parties rather than their negotiating positions and considers negotiating parties as allies working together for mutual gain, which is the essence of the negotiation process.

Since negotiation is a process of understanding, we make use of IBN to generate the dialogues in NDL. To realize the envisioned interactions in our system we extend the computational model proposed in (Xuehong Tao, et al., 2006) on the automation of IBN. Fig. 3 shows the architecture of our system which consists of the following functional components:

State Reasoner: handles all the state-related tasks. It generates the State Model (SM) for the learner by translating learner inputs to the corresponding affective and behavioral states. The State Updater (SU) updates all these state in real-time with each transaction. It also stores previously held states of the learner to understand learner progression.

Dialogue Manager: consists of the Rules Checker (RC) which is an inference engine and uses the information from the SM in conjunction with the LM in order to select the next system move with the maximum utility according to the current context. The Context Analyzer (CA) submodule uses the information from the SR and the NLPE in order to articulate the current context. It also consists of the Discourse Manager (DiM) that controls the flow of the overall dialogue.

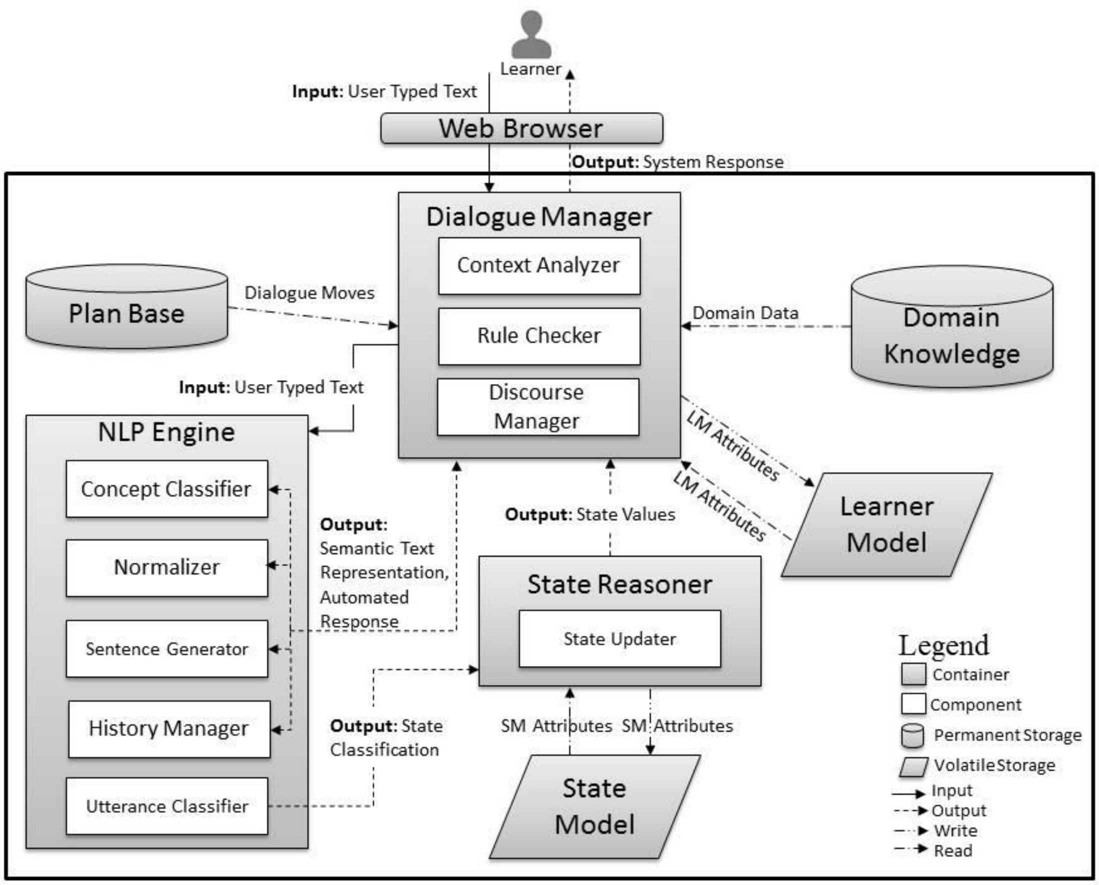


Fig. 3. NDL System Architecture

NLP Engine: this is the core module for providing a Natural Language interface to the learner. NDL does not require a complete NLP understanding as we are interested in the concept-level cognition of the learner's input. To accomplish this, the NLPE consists of submodules which include:

Concept Classifier: uses a Normalized Distance Compression algorithm to return a list of concept identifiers that most closely match the learner input.

Normalizer: manages stemming and spell checking for the learner input.

Sentence Generator: uses the concepts identified along with the current context to generate a list of possible utterances of the system. These possibilities are matched with the response library and the best matching phrase is selected to generate sentences automatically.

History Manager: stores information about the concepts used by the system and the concepts expressed by the learner. This information is passed to the RE, which uses it to classify the current context.

Utterance Classifier: uses a Cosine Similarity Index algorithm to return a list of state classifiers that are identified from the learner input.

Plan Base: holds the different negotiation moves available to the system according to a specific context. The information regarding the consequences of using a move in a specific context and state are used to update a move's adequacy to that context in the PB.

3.4 Requirements Analysis

Realizing interactions envisaged in NDL, it requires that the system not only understands the learner's characteristics but is able to comprehend their answers to provide a proper response. In NDL we wanted to introduce a more flexible, open, and natural method of interaction between the learner and the system. The use of chatbots has been documented to ease the negotiation process and improve engagement levels (Kerly, A. & Bull, S. 2006, Kerly, A., Ellis, R., Bull, S 2008). In light of these previous studies on the use of chatbots in OLMs we put forward the following questions for ourselves:

Q1. Can a conversational agent provide a more natural and flexible negotiation interface to the learner than a menu-based system?

Q2. What kind of dialogue moves would be required to facilitate such a negotiation?

Q3. What will be the challenges of implementing such a chatbot?

Q4. Which affectvie & behavioral states of a learner we need to pay attention to for realizing usable IBN-based dialogues?

To find the answers to these questions, we conducted a Wizard-of-Oz (WoZ) experiment. Natural language dialogue is complex in nature and the interaction patterns differ from learner to learner. Such inconsistencies were to be faced in negotiating the LM with learners; therefore, we required empirical data in order to support our system design. The WoZ approach has been shown to be valuable for collecting data in scenarios which require complex interactions between the users and the systems (Dahlbäck, N., Jönsson, A., and Ahrenberg, L., 1993). Our experiment design was based on the structure and guidelines on conducting a WoZ experiment provided by the previous study on CALMsystem (Kerly, A., Ellis, R., Bull, S. 2008). Building upon the findings of the previous studies our experiment design included a self-annotation mechanism for students to annotate their input according to the option they think best describes their current affective and behavioral state. We also used the findings of the previous study to generate a list of possible outcomes/markers that could be related compared afterwards. Since in the WoZ experiments, users are under the impression that they are interacting with a system, many application-specific characteristics of a textual dialogue can be elicited.

For this experiment we created an independent OLM. The domain of "Data Structures" was used for this experiment. The system gave a multiple-choice questions test to capture their understanding. These test scores were used to analyze the performance of each student and the results were used to generate the learner model. At the end of the test, the learner was allowed to update their belief base about their knowledge in the corresponding topic. This allowed for the wizard to initiate a dialogue in the case of a conflict occurring between the system's set of beliefs and the learner's set of belief. Ensuring a mixed-initiative dialogue system, the participants were also allowed to initiate a dialogue with the system by themselves at any time. During their interaction with the system, the participants typed their inputs and were required to annotate each input according to a drop-down list of states provided to them (self-annotation). They had the liberty to select multiple states which they thought best represented their mental state or they could provide a new/different state not available in the list.

3.5 Wizard-of-Oz Experiment

Wizard-of-Oz experiments have been shown to be very useful in eliciting application-specific characteristics of dialogues in complex interaction scenarios (Dahlbäck, N., Jönsson, A., and Ahrenberg, L., 1993). In a WoZ experiment, the participants are under the impression that they are interacting with a live system however the role of the system is "played" by a human which is commonly refered to as the "wizard". In our experiment, the wizard played the role of a "chatbot" which allowed the participants to discuss their learner model through a natural language interface. The encapsulation of the human experimenter from the participants ensures that the participants interact with the system as they would do in a natural setting. Another benefit of using the WoZ approach is that the interaction data can be collected without implementing a complete system.

The study was conducted with the students of Bahria University, Islamabad, Pakistan. A total of 45 students from semester of the Software Engineering course participated in the experiment. All participants had completed the compulsory courses of computer programming (C++, OOP, and Data Structures) as a course requirement. One of the present authors acted as a secondary experimenter while the experiment was conducted and supervised by the local instructor (Senior Lecturer in the SE department). The author was available via an online connection throughout the duration of the experiment. The participants were given an introduction to ITSs and OLMs by the secondary experimenter through a Skype video conferencing session. The session included an introduction to the aims and objectives of ITSs and their real-life applications. The participants were also introduced to the different categories of OLMs and were shown the interfaces and interaction possibilities provided by some OLMs, specifically Mr. COLLINS (Bull, S. & Pain, H., 1995), STyLE-OLM (Dimitrova, V., 2003) and CALMsystem (Kerly, A., Ellis, R., Bull, S., 2008). Ann initial survey was conducted to understand their expectations from such a system.

The participants were provided with a web interface to interact with the system. All interactions between the system and the participants were logged and the interaction transcripts were stored for future analysis. Once the participants had completed their sessions with the system, another survey was conducted to get their feedback about the system and the interaction possibilities it provided.

The participants were randomly divided into 3 groups; 1 uncontrolled group and 2 controlled groups. This was done in order to ensure that the system responses generated during each phase would be valid enough for a diverse group of learners. The experiment was conducted in 3 phases where in the 1st phase with the uncontrolled group, there was no negotiation protocol set for the wizard. The wizard conducted open-ended dialogues with the participants without following any set of rules. The dialogue scenarios captured in these interactions were translated into IF/THEN clauses in order to generate the initial 'rules library'. The interaction logs were also used to generate a corpus for system responses that constituted the first version of the response library. Fig. 4 shows a screenshot of the response library available to the wizard.

In order to generate the response library, the protocol discussed previously was used to classify system utterances. The strategies used are:

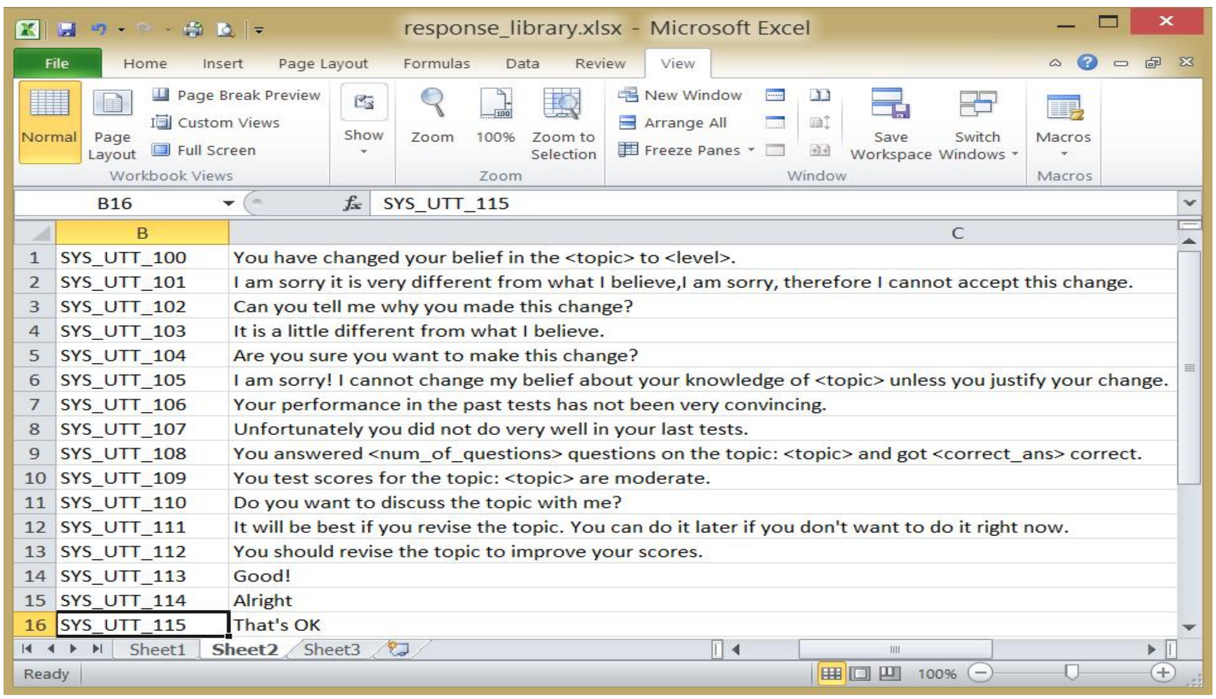
1. ASK for JUSTIFICATION: ask to justify a response/claim.

2. GIVE JUSTIFICATION: provide justification for the last utterance/action.
3. ACCEPT JUSTIFICATION: accept the claim if it is justified.
4. REJECT JUSTIFICATION: reject a claim if it is not justified.
5. GIVE PROPOSAL: propose an alternative solution
6. ACCEPT PROPOSAL: accept a proposal.
7. REJECT PROPOSAL: reject a proposal.
8. PROVIDE FEEDBACK: provide feedback corresponding to the last action.

Both the rules library and the response library were saved in MS Excel file for quick access to appropriate response to the learner. Each system response was given unique identifier SYS_UTT_ #, where '#' was a unique numerical value. This allowed the wizard to only select and copy/paste the corresponding system utterance in the next phase.

The 2nd phase with the controlled group 1 was conducted under controlled conditions where the wizard used the rules and response libraries generated from the analysis of the interactions in the previous phase under the protocol guidelines to respond to student inputs. During this phase there were certain scenarios which did not occur in the previous phase and hence had no corresponding rules in the rules library to select an appropriate response from the response library. In such situations, the wizard had the liberty to improvise the response and such a situation was highlighted for future analysis of the dialogues.

The 3rd phase was conducted with the controlled group 2. The interaction logs of the first two phases were used to update the rules and response libraries. The analysis of the first two phases allowed for the improvement of the rules and response libraries for the wizard by including missing rules and responses for new dialogue scenarios. The 3rd phase of the experiment was almost completely automated with 85% of the wizard responses being generated by using the rules library. The results from this phase were again used to update the libraries to accommodate missing rules or responses. Table 2 shows an example of a rule used by the wizard in order to select a corresponding system response.



	B	C
1	SYS_UTT_100	You have changed your belief in the <topic> to <level>.
2	SYS_UTT_101	I am sorry it is very different from what I believe,I am sorry, therefore I cannot accept this change.
3	SYS_UTT_102	Can you tell me why you made this change?
4	SYS_UTT_103	It is a little different from what I believe.
5	SYS_UTT_104	Are you sure you want to make this change?
6	SYS_UTT_105	I am sorry! I cannot change my belief about your knowledge of <topic> unless you justify your change.
7	SYS_UTT_106	Your performance in the past tests has not been very convincing.
8	SYS_UTT_107	Unfortunately you did not do very well in your last tests.
9	SYS_UTT_108	You answered <num_of_questions> questions on the topic: <topic> and got <correct_ans> correct.
10	SYS_UTT_109	You test scores for the topic: <topic> are moderate.
11	SYS_UTT_110	Do you want to discuss the topic with me?
12	SYS_UTT_111	It will be best if you revise the topic. You can do it later if you don't want to do it right now.
13	SYS_UTT_112	You should revise the topic to improve your scores.
14	SYS_UTT_113	Good!
15	SYS_UTT_114	Alright
16	SYS_UTT_115	That's OK

Fig. 4. Response Library

The students were divided randomly in 3 equal groups for the 3 phases of the experiment. This meant that for each phase, we had 15 students interacting with the wizard. Each group had a single interaction session with the wizard. The interactions in the 1st phase were the longest as there was no set negotiation protocol, so the students and the wizard indulged in a very open discussion. The interaction times of the 2nd phase were considerably shorter as a negotiation protocol was introduced and the discussion was more directed. The average interaction time in this phase was 20 minutes. The 3rd phase saw the shortest interactions as it used the formalized rules and response library. Average interaction time for the 3rd phase was 16 minutes. All of the interactions were concluded successfully with the student either accepting the wizard's proposal or retaining their initial stance about their knowledge level.

Table 2

Sample Rule for wizard to select system response.

IF	User has changed their belief in topic and the difference between their belief value and the system's belief value is greater than 2
THEN	Highlight User Change: {SYS_UTT_100} REJECT CHANGE: {SYS_UTT_101} ASK for JUSTIFICATION: {SYS_UTT_102}

3.5.1 Results

The interaction logs and the conversation transcripts form the WoZ experiment were transcribed and analyzed in order to understand the kind of dialogues the participants engaged in with the system. In the 45 conversations between the student's and the wizard there were a total of 195 negotiation fragments. The number of user initiated conversations was 80. The mean interaction time was 27.4 minutes. Off-topic discussions or small talk constituted 13.4% of all conversations. 45.6% of the conversations were related to domain-specific discussions while the remaining 41% conversations constituted the inputs used to approximate learner characteristics.

While off-topic conversation during a tutoring session may be seen as counter-productive to the construction of knowledge, it has been found to be an effective strategy to keep the learners engaged. Expert tutor utilize off-topic conversations in scenarios where the learner seems to be disengaged or frustrated. It is seen as useful strategy to build a sense of trust and empathy using a dialogue that does not require the learner to recall domain or task-oriented knowledge. Having the ability to engage at a certain level of small talk allows the system to provide responses to user inputs that are not related to the domain or the task at hand. This gives the system the ability to hold more naturally flowing dialogues with the learners.

3.5.2 Classifying Student's Affective and Behavioral States

Affect relates to the emotional reaction (feeling) one has towards an attitude object (learning task). For example, if a student is confused about a mathematical concept (attitude object), whenever they are exposed to a problem related to that concept, they feel confused. Behavior relates to how one behaves when exposed to an attitude object. Considering the previous example, if the student is confused about a concept, they are most likely to avoid it and be less interested in taking on the problem.

There are many unknown categories of learner's mental states and an in-depth evaluation of all these states was out of the scope of our study. For the initial classification of the participant's affective states we used Ekman's six "basic" emotions (Ekman, P., 1973) and a set of learning-focused affective states identified in (Graesser, A. C., et al., 2006; D'Mello, S. K., et al. 2007) for our study. The list of affective states that was used for this study include: *confusion, engagement, frustration, curiosity, eureka, surprise, anger, fear and sadness*. Similarly for the classification of behavioral states, we used only the "states" identified in (De Vicente, A., & Pain, H., 2002) based on the theories of motivation in education (Malone, T. W., and Lepper, Mark R., 1987; Keller, J. M., 1983). Choosing between different states is not a trivial task; therefore, we concentrated on the states that would have a deeper impact on the outcome of an interaction. We limited our study to the states that characterize a student's behavior while interacting with a human tutor which include: *confidence, interest, satisfaction, effort and motivation* along with their negative dimensions. The occurrence frequencies of the states were used as the measure of acceptance which narrowed the affective states list to; *confused*, *frustrated*, and *engaged*. Whereas the behavioral states selected were; *confident*, *interested* and *motivated*.

The interaction logs generated during the experiment consist of self-annotated typed input by the participants. There is no gold standard for understanding and detecting the mental state of a learner from an interaction log. To this effect we employ the Multiple-judge strategy (Graesser, A. C., et al., 2006) to manually annotate the interaction logs. The judges included the participants (self-annotations) and 2 expert judges (assistant professors) and 2 intermediate judges (lecturers). One of the expert judges was a professor of psychology while one of the intermediate judges was a lecturer in linguistics. This selection of judges provided us with a diverse pool of experience which was very helpful during the discussions over the annotated utterances. The judges were provided with the learner interactions along with the list of affective and behavioral states classified for this study. They were also given the liberty to add a new state if they deemed necessary in order to capture the approximation of the participant's mental state. We are aware of the subjective nature of this classification scheme which might not reflect the true mental state of a learner. However, we have previously emphasized that an approximation scheme can be considered sufficient to control the flow of the dialogues. An incorrect classification of a learner state does not drastically impede the dialogue course as the system uses the context and dialogue history to ensure an effective flow of the dialogue. We will discuss this topic in the evaluation section below.

The judges were provided specific guidelines for annotating the transcripts. They were required to highlight any markers in the student's input that might point towards a specific attribute of their mental state. For example using "*Ummm…*" in the beginning of an utterance was classified by tutors as a sign of "low confidence" or "guessing". A similar "vocal" sound is associated to a thinking person. However, it was noticed during the experiments that when the students were thinking, they did not type "*ummm*", but rather made the vocal sound. Another important aspect of annotation was the consideration of "context" while annotating the transcripts. Context plays an important role in helping to decipher the rationale behind a specific utterance and in most cases the thought process involved. For example, if a student is asked a question related to the domain and they answer;

"*I don't know…. But I think it is ……*"

This input from the student is treated as "confused" and their answer is "not confident" but he is considered to be "interested" as he is trying to answer the question.

Similarly, the very basic utterance “OK” can have multiple meanings which can be elicited if the context in which the utterance occurs is known. The strategy to highlight markers in text and convey a context was very helpful in fine-tuning the rules in the library.

The annotated transcripts from the judges were compared with each other to find the matching and conflicting annotations. The list of conflicting annotations was discussed with all the expert judges in order to reach a consensus regarding a specific learner utterance and its relation to a specific affective of behavioral state.

The self-annotated lists of the participants were then matched with the agreed upon judge’s annotated list in order to generate a list of student utterances classification according to the affective and behavioral states. A list of utterances with no matches, or mismatches was also generated during this process. These lists were deliberated upon by the judges in order to remove any discrepancy between the annotated values. As mentioned previously, the panel of judges included an expert tutor of psychology and a lecturer in linguistics. This diversity of experience helped the panel to annotate utterances mismatching annotations to generate a complete list.

An interesting observation during the analysis of the inputs was the positive and negative dimensions of the specified states and how they affected the course of the dialogue. It was observed that in case of affective states, a negative affective state required more system involvement than a positive affective state. For example, if a learner was confused (negative state), the system had a better opportunity to help him realize his confusion than when he was not confused (positive state), in which case the system intervention was minimum. Contrary to this, the dimensions of behavioral states played a much greater role in the interactions between the learner and the system. A confident learner reacted differently than a learner who was not confident. It was observed that both positive and negative dimensions of behavioral states impacted the system's interactions with the learner.

3.5.3 Inputs Related to Affective States

In Fig. 5 we can see the distribution of the occurrence of the affective states in the learner inputs. These occurrences were calculated by comparing the tutor's annotated data from the experiment with the self-annotated data of the learners during their discussion with the system. The findings were consistent with previous study (D'Mello, S. K., et al. 2007). The most often occurring affective states were selected and identified as *confusion*, *engagement* and *frustration*. The rest of the affective states were almost non-existent in both tutor and participant annotations. Table 3 shows a list of a few learner inputs and their corresponding affective state.

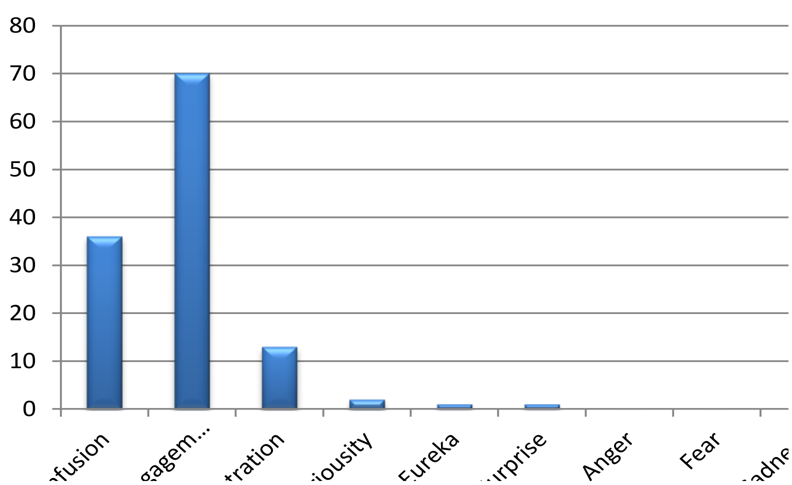


Fig. 5. Occurrences for each Affective State

Table 3

Examples of Affective states corresponding to user inputs

User Input	Affective State
I don’t understand	Confused
No! I still don’t understand	Confused, Frustrated
I don’t know	Confused, Frustrated
I don’t need your help!	Frustrated
What is this?	Confused
How?	Confused
I can’t do this	Frustrated
Wow I did it!	Engaged
Yes, I think I got it	Engaged
I know it	Engaged

3.5.4 Inputs Related to Behavioral States

In Fig. 6 we can see the distribution of the occurrence of the behavioral states in the learner inputs. The states with the highest occurrence frequencies i.e. *confident*, *interested* and *motivated* were selected for the classification of the learner utterances. Frequencies of the corresponding negative states were also added to the chart to show the number of occurrences along both positive & negative dimensions. Table 4 shows a list of learner inputs and corresponding behavioral states.

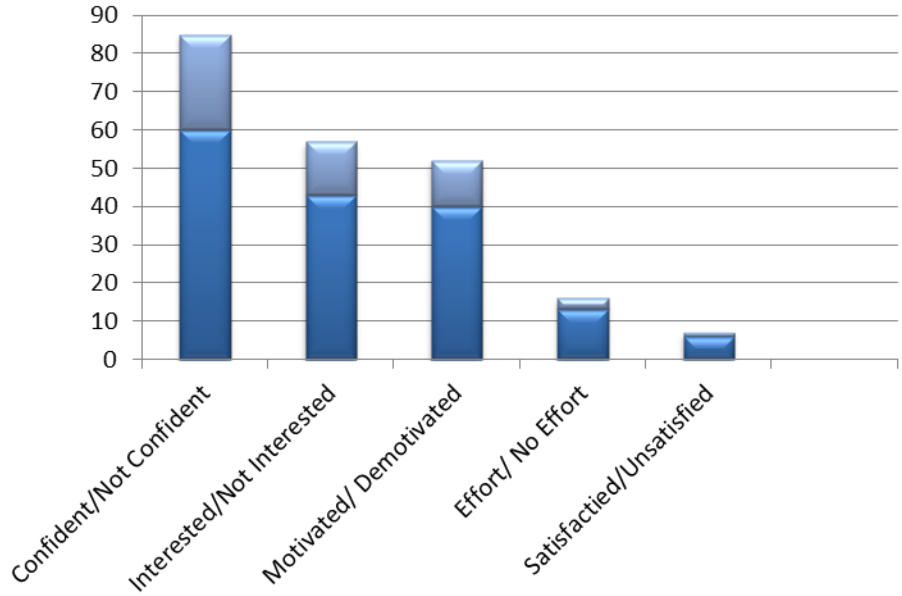


Fig. 6. Occurrences for each behavioral state

Table 4

Examples of Behavioral states corresponding to user inputs

User Input	**Behavioral State**
Yes I know	Confident
Ok, Yes, Yeah, Yeah sure, Sure (Context Dependent)	Confident, Motivated
I want to discuss this	Motivated, Interested
No (Context Dependent)	Uninterested
I’m not sure	Not confident
I don’t think so	Not confident
I don’t want to …	Uninterested
I can’t do this	Demotivated
I want to solve this	Motivated

3.5.5 Revisiting Questions set for the Experiment

At the end of the experiment, we analyzed the user interactions with the system, the observations made by the authors during the experiment and the discussions with the tutors' panel while annotating the learner utterances, in order to answer the questions we set for ourselves before the experiment. The first question we put forward was:

Q1. Can a conversational agent provide a more natural and flexible negotiation interface to the learner than a menu-based system?

The participants had never used an ITS before and therefore they did not have a hands-on experience of using a menu-based OLM. However as mentioned earlier, in the pre-experiment setup, the participants were given an introductory lecture on OLMs and the interaction possibilities provided by a few OLMs. They were shown interfaces and interaction fragments of previous system in order to familiarize them with the concept and applications of OLMs. In the post-experiment survey the participants noted that the natural language dialogue conducted by the wizard was a very natural and realistic form of interaction as it closely related to some form of chat messaging provided by many SNS and SMS application they use to in their daily lives. Majority of the participants were of the opinion that using a natural language negotiation approach would allow the student to interact with the system more openly. On the question of replacing the natural language interface with a menu-based interface, most of the participants answered in the negative as they thought it would make them feel controlled and confined. The authors are aware that the results from this question do carry a bias as the participants had no prior experience of a menu-based system. However, previous WoZ experiments have concluded that students did prefer a chatbot over a menu-based interaction system (Kerly, A., Hall, P., Bull, S., 2007). Now we consider the second question:

Q2. What kind of dialogue moves would be required to facilitate such a negotiation?

The answer to this question was investigated during the analysis of the results of the experiment. The negotiation protocol that was provided to the wizard proved to be sufficient in handling the course of negotiations from different participants. It was noted that apart from following the negotiation protocol, the system also needs to be able to handle a fair amount of off-topic discussions or small talk. This was in-line with the findings and guidelines provided by previous WoZ experiment to study the use of a conversational agent in an OLM (Kerly, A., Ellis, R., Bull, S. 2008). This became more evident in the interaction of less interested/motivated participants. However, it was also noted that almost all participants did engage in some form of small talk with the wizard during their interactions. Therefore, the discourse manager not only needs to follow a negotiation protocol, but also needs to be able to deal with small talk initiated by the learner, or in some cases initiated by the system itself in order to engage the learner and keep continuity in the discussion. Another finding that resonated with results of previous WoZ experiment is consideration that the system should be able to keep track and control the level of small talk during an interaction. This will be essential to ensure that the learner does not spend too much time off-topic.

Q3. What will be the challenges of implementing such a chatbot?

This question relates to the challenges we could foresee for our system after conducting and analyzing the experiment. Our findings were in line with previous work on the use of chatbots in OLMs (Kerly, A. & Bull, S. 2006, Kerly, A., Ahmad, N., Bull, S. 2008, Kerly, A., Hall, P., Bull, S. 2007). The more prominent challenge was the implementation of a natural language interface that will be able to handle vast array of user inputs. The research on Natural Language Processing (NLP) has been continuing for years and there is no single, best NLP approach that can be used to generate a 100% realistic dialogue environment. Keeping this limitation in mind, we needed to decide upon the tradeoff between the usability of an NLP technique and its complexity. Spending too much effort and time on implementing the NL interface would negatively affect the scope of the project. Hence it was decided to keep the complexity of the NLP to a minimum and with each development iteration try to improve upon it.

The second challenge that was acknowledged was the complex nature of learner states and identifying such states automatically. Since we will not be using any sensory information and only use the typed input to generate an approximation of the learner's state, this will make the task simpler but the accuracy of the resulting states will remain questionable. Further research will be required in this respect in order to maximize the usability of the state model at acceptable cost.

The third challenge identified by the analysis of the experiment was related to the user experience. Learners with different knowledge level use the system differently and their interaction patterns also vary significantly. A chatbot in a learning environment needs to be able to adapt to this change in character and keep the learner engaged and on topic. As identified earlier, small talk can act as a good strategy to bring back the learner who loses interest, but the system needs so ensure that the small talk should not hinder the learning process.

Lastly the experiment also gave insight to the problem of authoring a chatbot script from scratch. Most of the current chatbot implementations use specialized script formats that increase the learning curve and require some time to generate. This is normally due to the fact that domain-dependent and domain-independent dialogue fragments are merged into the same script. To minimize this complexity a scheme that separates the domain-dependent and domain-independent utterances and uses a mechanism to merge them at runtime would allow tutors to concentrate more on the domain-dependent section of the chatbot. This would result in faster development times and maintenance tasks would be more simplified.

Q4. Which affective and behavioral states of a learner we need to pay attention to for realizing usable IBN-based dialogues?

The Wizard of Oz experiment aided us to short-list the affective and behavioral states that were most prominent during the interactions between the participants and the wizard. As discussed in detail in the previous sections, the deliberation between the experts over the annotated logs allowed us to finalize a list of 3 affective and 3 behavioral states that will be used to control the flow of the dialogue in our system. The sufficiency of these affective & behavioral states was evaluated in the first evaluation study which will be discussed in Section 5.

4. IMPLEMENTATION - NDLtutor

The Wizard-of-Oz experiment discussed in section 3.5 allowed us to collect interaction data between the user and the system. The analysis of this data allowed us to understand the possible interaction patterns that need to be handled by our system i.e. the kind of typed inputs expected from the users and the appropriate responses of the system. The data collected during this experiment was used to generate the *user utterance library*; a collection of common user utterances classified according to their affective & behavioral state values and utterance category, *rules library*; a collection of IF/ELSE clauses that are used by the system to generate its next move and, *system utterance library*; a collection of system utterances that the chatbot uses to interact with the user according to the system move selected through the rules library.

To evaluate the feasibility of the architecture we defined for NDL, the validity of the affective/behavioral states and the effects on the self-reflection and self-assessment skills of the learners, we implemented a system called NDLtutor. It is a *rule-based system* that uses a *chatbot* to provide a natural language interface to the learner to discuss their LM with the system in the data structures domain. Since NDL can act as a component of an ITS, we implemented NDLtutor as an independent OLM. The NDLtutor diagnoses the knowledge gaps of the learner during discussions about the learner's beliefs and promotes metacognition by using reflective dialogue strategies.

Some of the design rationales of the NDLtutor were as follows:

1. Separation of domain-dependent and domain-independent content.
2. Minimize Natural Language Understanding complexity by using using sentence classification to understand learner input.

3. Generating system responses at run-time.
4. Employing data-centered approach using a Relational Database Management System (RDBMS) to store all content to ensure system portability and future extensibility.
5. Using Open Source technology to allow for the project to be distributed on a public forum to encourage collaborative development.

4.1 Overview

The workflow of the dialogue in NDLtutor is shown in Fig. 7. The system receives the learner's input through the interface. The system classifies the input by matching it with entries in the database. The classification process is discussed in detail in section 4.3. Once the input is successfully classified by the system, the system uses the information from the classified input to set/update the corresponding elements in the working memory and the State Model of the learner. The current elements of the working memory are *matched* against the rules in the rules library to identify the applicable rule. The chosen rule is then *applied* which results in the selection of the next action/response of the system and also adds/updates the elements of the working memory. The system uses the values of the affective and behavioral state attributes from the State Model to *adjust* the selected system response to make it more reactive. Finally, the termination condition is *checked*; if it is fulfilled, the session is concluded and depending upon the results of the session, the Learner Model is updated. This allows the NDLtutor to be more empathetic in its responses to the learner's input. In the NDL paradigm, we believe that learning is maximized by participation; such a dialogue design provides sound foundations for the system to actively engage learners and encourage participation.

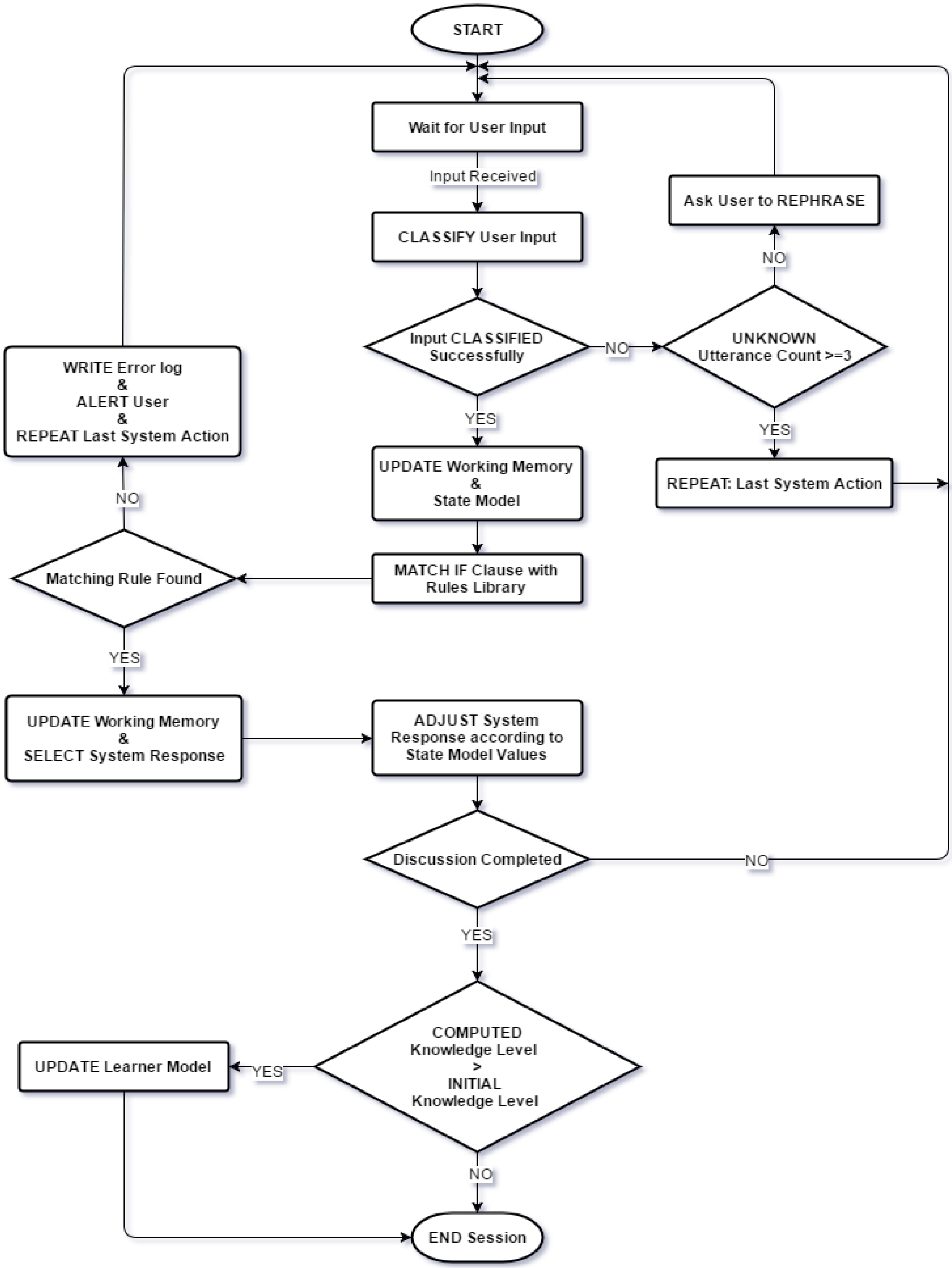


Fig. 7. NDLtutor Workflow

4.2 Database Structure

The fundamental component of the NDLtutor is the backend database of the system. The NDLtutor uses MySQL 5.5 community edition as the backend database. The database consists of the following main tables:

1. Domain_Topics
2. Topic_MCQs
3. Topic_Discussion_Questions
4. Topic_Discussion_Answers
5. Topic_Discussion_Question_Concept_List
6. Concept_Funnel_Questions
7. Concept_Funnel_Answers
8. Topic_ Discussion_Question_Misconcept_List
9. Misconcept_Funnel_Questions
10. Misconcept_Funnel_Answers
11. USR_Utterance_Library
12. SYS_Utterance_Library

4.2.1 Domain Structure

The domain content is stored in the Domain_Topics table. Each domain can be divided into a set of topics with each topic given a unique Topic_ID. The Domain_Topics table allows the author to input the content for each topic that can be displayed in the Content Area during the normal course of tutoring.

The Topic_MCQs table is used to store the MCQs related a specific topic which is identified by the unique Topic_ID. These MCQs are used to gauge the learner's level of understanding in that topic. Each MCQ in the table has a unique MCQ_ID which can be used by the system to generate a list of unanswered MCQs for future tests.

The Topic_Discussion_Questions table stores the general domain discussion question for each topic. This is the first question that the system asks a learner when they agree to discuss the topic with the system. The table stores the text of the question and the unique Question_ID used to access other tables with related data.

The Topic_Discussion_Answers table stores the answers to each topic question identified by the unique Question_ID. A single question can have multiple answers, therefore each answer is specified a specific category i.e. Expert, Intermediate, Novice, Incorrect. This answer category is used by the system during the scoring process.

The Topic_Discussion_Question_Concept_List table stores the concepts related to each topic. The Question_ID is used to identify the concepts related to a specific question. Moreover, the table allows the author to input multiple values for a single concept as a "Related_Concept". This allows the author to specify multiple synonymous terms for a single concept. If any of those terms is found in the learner's answer, the system considers the concept is covered. The Topic_Discussion_Question_Misconcept_List table has the same structure but is used to store the common misconceptions related to a question.

The Concept_Funnel_Questions and Concept_Funnel_Answers tables are used to store the questions and answers related to the concepts. These questions are asked by the system if the system identifies that the learner's answer is missing a certain concept.

4.2.2 User Utterance Library

The USR_Utterance_Library table holds the common user utterances during their interaction with the system. This data is used by the system to classify an input by the learner into a specific utterance class and act. Each user utterance has a value of the affective & behavioral state as identified by the analysis of the WoZ experiment. Each of these attributes can have a value of; -1 (negative connotation), 1 (positive connotation), NULL (unspecified for the utterance). These state values are used to update the State Model of the learner according to their inputs at run-time. Furthermore, each user utterance also has Usr_Utterance_Category attribute which is used to specify the speech act of the learner. This helps the system to narrow down the possible response range and improves processing times.

Table 5

User Utterance Categories with sample sentences

User Utterance Category	Example Matched input
know_more	“i know more” “i improved my knowledge” “i’m better prepared now”
know	“i know” “i understand” “i know this”
not_know	“i don’t know” “i don’t know more details” “that’s all i know” “i dunno”
approve	“yes” “yeah” “ok” “okay” “alright”
disapprove	“no” “nope” “I don’t want to” “no thanks” “nah” “i wont”
ask_help	“!hint” “!help” “ i need help” “what should i do” “can you help me”
assert	“i want to prove myself” “I want to change my level”
enquire	“am i right?” “is this right?” “was it right? ” “is this correct? “am i correct?
confirm	“really?” “is it so” “should i” “i’m sure”
negate	“my proficiency is not” “its not”
smalltalk_who_r_u	“who are you” “are you a bot” “are you a girl” “are you a guy”
smalltalk_laugh	“haha” “hehe” “rofl” “lmao” “:D” “:P”

smalltalk_tell_joke	“tell me a joke” “i wana hear a joke” “please tell me a joke”
smalltalk_bored	“im bored” “youre boring” “this is boring”
greeting	“hello” “hi” “hi there” “hey”

As mentioned previously, the data collected during the WoZ experiment was used to generate the user utterance library. The learner utterances were analyzed by a panel of experts to categorize them into categories that specified the intention of such utterances according to the speech act classification theories (Ballmer et al., 1981). Different speech act theories classify learner’s inputs into broad categories such as Austin’s Speech Act Thoery (Austin, 1962) defines locutionary acts, illocutionary acts and prelocutionary acts whereas Searle’s Speech Act (Searle, 1975) include assertives, directives, commisives, expressives and declaratives etc. Such broad categorization makes the task of finer-grained categorization very difficult. Our analysis of the data collected during the WoZ experiment hightlighted the different actions/intentions of the learners during their interaction with the system. The interaction logs were deliberated on by the panel of expert to define a set of distinct categories that could encompass all the interactions in terms of the *intention/action* that were witnessed during the experiment under the light of the speech act theories. This was done to provide a fine-grained categorization scheme for learner utterances. The idea was to make the process of understanding the learner *intentions/actions* from their utterance simpler by using a custom set of specific speech act categories that could help us realize the dialogues envisioned in the NDL paradigm. The 15 categories shown in Table 5 are used by the NDLtutor to understand the intention/action of the learner and help the NDLtutor to select its next action that is most suited to the learner’s current action.

4.2.3 System Utterance Library

The SYS_Utterance_Library table holds the system utterances that are used by the system to generate responses to the learner during an interaction. Each system utterance has a unique SYS_UTT_ID to keep track of the last utterance of the system as well as to identify unique utterances. Every system utterance also has a Class attribute. This Class attribute is stored by the system in the session during an interaction. This allows the system to keep track of the last moves of the system and generate a context for the next expected move.

4.2.4 Rules Library

The WoZ experiment allowed us to create a set of IF/THEN rules that are used to select the system's action i.e. next move or response. The rules library is stored as a collection of IF/THEN clause where the IF condition is composed of a number of parameters that must be matched for the condition to be true. Once a corresponding IF statement is found, the THEN part is used to select the system's action accordingly. The current rules library consists of 33 rules that are used to handle different scenarios based upon the current context of the dialogue. The rules are divided into 2 broad categories:

1. Feedback Rules
2. Dialogue Move Rules

4.2.4.1 Feedback Rules

Feedback rules are related to the feedback delivery mechanism of the NDLtutor. There are 3 types of feedbacks provided by NDLtutor:

— *Positive feedback:* “That’s great!” “Good Job” “Great work”

— *Negative feedback*: “No” “I’m afraid that’s not correct” “I’m sorry that’s wrong”

— *Neutral feedback:* “Ok” “Hmmm…”

Feedback rules are used to provide immediate feedback to the learner’s input. Feedback rules are mostly used in combination with the Dialogue Move rules to complement the continuity of dialogue. An example of a feedback rule is as follows:

IF CURR_SYS_MOVE = **DOM_QUESTION** AND USR_UTT = **ANSWER** AND CONCPT_COV = **HIGH** AND ANS_CAT != **MISCONCPT** AND NUM_OF_TRIES = **1** AND ANS_SCORE = **HIGH**

THEN POS_FDBK (Positive Feedback)

Or

IF CURR_SYS_MOVE = **DOM_QUESTION** AND USR_UTT = **ANSWER** AND ANS_CAT = **BAD_ANSWER**

THEN NEG_FDBK (Negative Feedback)

4.2.4.2 Dialogue Move Rules

Dialogue Move Rules (DMR) are used to select the next move or action of NDLtutor according to the current context. NDLtutor’s moves include:

— *Prompts:* “Restate/Explain”, “Review”, “Restate/Rephrase”

— *Splices:* “A stack is an Abstract Data Type (ADT) that stores elements in reverse order”

— *Hints:* “Stacks have 2 basic operations _____ and ____”

— *Smalltalk:* “Jokes” “self-introduction”

— *Questions:* "Topic Questions" "Concept Funneling Questions" "Misconception Funneling Questions" "Reflection Questions"

DMR ensure that the system generates an adequate response to the learner's input to conduct a meaningful dialogue. These rules are used in all the 3 phases of the dialogue i.e. initialization, domain discussion and reflection phase. They differ in the terms of the parameters that are used to construct the IF query clause. For example for the Initialization Phase, the *current system move* and the *user utterance* are used to decide the *action* and *next system move.* Below we see an example of such a DMR for the Initialization Phase:

IF CURR_SYS_MOVE = **PRFMNC_OVRVIEW_PROPOSAL** AND
USR_UTT = **APPROVE**

THEN SHOW_TOPIC_PRFMNC (Action)
AND
NXT_SYS_MOVE = **DOM_DISCUSSION_PROPOSAL**

Another example for the DMR with the parameters for the Domain Discussion Phase is as follows:

IF CURR_SYS_MOVE = **DOM_QUESTION** AND USR_UTT = **ANSWER**
AND CONCPT_COV = **LOW** AND ANS_CAT != **MISCONCPT** AND
NUM_OF_TRIES = **2** AND ANS_SCORE = **LOW**

THEN NEU_FDBK (Neutral Feedback)
AND

QUESTION_SPLICE

AND

CONCPT_FUNNEL = **START** (Start asking questions regarding missing concepts)

It is important to highlight here that in the current implementation of the NDLtutor, the system uses a predetermined sequence of questions to help the learner reflect upon the discussion and analyze their performance during the Reflection phase. These questions are sequenced such that they encourage the learner to *evaluate* their answer, *monitor* their performance and *plan* what they should concentrate on.

As mentioned previously we also introduced an informal formative assessment in the form of a Reflection Score in the reflection phase. The NDLtutor asks the learner 8 questions in total, out of which 5 questions currently are used to calculate a reflection score. An example of a Reflection Scoring Rule is as follows:

IF CURR_SYS_MOVE = **IDENTIFY_MSG_CONCPTS** AND USR_UTT = **ANSWER** AND CONCPT_COV = **LOW**

THEN NEG_FDBK (Negative Feedback)

AND

LIST_MSG_CONCPTS

AND

RF_SCORE = 0

The *current context* of the dialogue plays a vital role in selecting and triggering a rule from the Rules Library. Global parameters values are only updated when the global context changes while all local parameters are assigned values at runtime with respect to the learner's inputs by the context analyzer for each interaction. The final IF query is generated by combining all those parameteric values in the current context which are *not null*.

4.3 Natural Language Understanding

Understanding natural language dialogue is not a trivial task and therefore is considered as an AI-hard problem. Natural Language Understanding (NLU) incorporates a diverse set of functions including; relation extraction, paraphrase & natural language inference, semantic parsing, sentiment analysis, summarization, vector space models etc. All these tasks require complex in-depth analysis and statistics to get a basic understanding of text which can still not be considered as "full" understanding (Liddy et. al, 2003).

Since NDLtutor provides a natural language interface to the user, we needed to select a feasible approach to handle the natural language processing of the system. Syntactic and semantic analyses have traditionally been used to understand natural language text but such methods bring with them the complexity of generating grammatical/statistical models of the language. No single NLP approach is considered as an industry-standard for handling natural language input. As an alternate to complex statistical and grammatical Natural Language Understanding approaches, previous chatbot implementations such as ALICE (ALICE 2002, Wallace 2003), PARRY (Colby, K. 1973) and ELIZA (Weizenbaum, 1966) use a less complex pattern-matching approach on a predefined set of categorized input patterns with corresponding templates (Shawar, B, A., Eric, A., 2002). These chatbots use specialized markup files for storing their knowledge. For example ALICE uses the AIML (Artificial Intelligent Markup Language) to store its knowledge in the form of topics and categories. Each category consists of a pattern which represents the user's input and a template which holds the chatbot's response. When a user types in their input it is matched with the pattern in each category and if a match is found, then the corresponding template is used to generate the chatbot's response. ALICE allows searching for input patterns using regular expressions and wild cards.

Similar to the pattern-matching approach of ALICE, to minimize the complexity of complete NLU in the implementation of NDLtutor, we turned the NLU process into sentence classification using a predefined set of input categories with a *text-similarity* measure. As described earlier, we used the Wizard of Oz experiment to inform the design of the NDLtutor. The analysis of the WoZ experiment provided us in-depth insights about the possible interaction patterns and characteristics as well as plausible concrete utterances. Some observable characteristics of the user interactions were:

1. Limitied vocabulary encountered during the WoZ experiment which makes it predictable.
2. Frequent use of common phrases/sentences by the participants.
3. Limited use of complex, multi-phrase utterances.

It was evident from the data collected during the WoZ experiment that the scope of the input texts from the learners was limited by the boundaries of the task-specific dialogue. This means that the learner inputs were largely similar and followed an identical sequence of flow. Hence the task of understanding all kinds of inputs during an interaction between the learner and the system can be appropriately exhaustive in terms of resources; this complexity can be minimized by investigating the nature of the expected dialogues and identifying/storing common vocabulary, phrases and sentences for sentence matching.

Following this approach the interaction data collected in the WoZ experiment is used to build the chatbot knowledge-base (separated as domain-dependent & domain-independent utterances) which is stored in the database tables. We use a text similarity-based measure to match the learner's input text (typed sentences) with the utterance text (*usr_utterance_text*) in the *user utterance library* or the *domain answer tables* to generate an utterance classification. The resulting match allows us to allocate the learner's input a *category* which is used by the system to understand the *action/intent* of the learner.

Using a text similarity approach also helped us to avoid the overhead of having exact matching sentences in the database, i.e. the system is able to match "*I don't know"* with "*I dont knw"*, hence we don't have to worry about different forms of a word or small misspellings. To calculate the text similarity, we use the Normalized Compression Distance (NCD) approach, which is a *text similarity* metric based on the theoretical concept of *Kolmogorov Complexity* (Cilibrasi et. al, 2005). NCD is used to measure the similarity between two fragments of text. The similarity between two text fragments is defined as the degree of difficulty into transforming them into each other. NCD is considered as a basic-medium text similarity string and works well with text inputs.

4.3.1 Natural Language Understanding in NDLtutor

Before discussing the process of Natural Language Understanding of the NDLtutor, it must be noted that the system divides learner inputs into 2 broad categories:

1. **Domain-dependent utterances:** domain-dependent utterances are related to the domain and contain the domain content. Such utterances are generated by the learner to provide *answers* to the system's domain/task specific questions. Domain-depedent utterances are only *expected* during the Domain Discussion phase. Domain-dependent utterances are stored in the chatbot knowledgebase in the Topic_Discussion_Answers, Concept_Funnel_Answers, and Misconcept_Funnel_Answers (*domain answer*) tables. Such utterances are considered as *bag-of-words* which means that while matching the learner's answer to such utterances, the system can ignore the sentence structure and use the *keywords/concepts* to match and to evaluate a learner's answer.
2. **Domain-independent utterances:** domain-independent utterances consist of phrases that do not contain any domain information and are used to control the flow and continuity of the dialogue by the learner. Hence any utterance that is not *classified* as an *answer* is automatically considered as a domain-independent utterance. The domain-independent utterances are stored in the *user utterance library* of the chatbot knowledge-base. The surface-level structure of the domain-independent utterances is crucial for the classification process since the classification scheme uses a threshold value to find matching utterances in the knowledge-base. This means that the learner's input needs to be quite similar to the domain-independent utterance for the classification scheme to identify it properly.

The user utterance library consists of common user utterances that are classified according to affective and behavioral state values and a specific category. When a learner types in their input through the chatbot interface, the learner's input text is matched with the entries in the user utterance library using NCD. If a match is found, then the corresponding values of the affective & behavioral states and category is used by the NDLtutor to classify (understand) the learner's input text and respond accordingly. Learner inputs are categorized as *domain-dependent* utterances, and *domain-independent* utterances. As mentioned earlier, domain-dependent utterances include the learner's inputs used to *answer* the domain/topic discussion questions during the Domain Discussion phase. Therefore, the domain-dependent utterances are limited to the scope of the Domain Discussion phase. Domain-independent utterances occur in all the 3 phases of NDL and can be used by the learner to express how they feel, inform the NDLtutor about their belief or ask the NDLtutor for help etc.

The learner's input goes through a series of checks before it is classified as a specific utterance by the NDLtutor. These checks are influenced by the nature of the utterance i.e. (domain-dependent/domain-independent). Fig. 8 shows a conceptual flow of how the learner's input is processed by the NDLtutor.

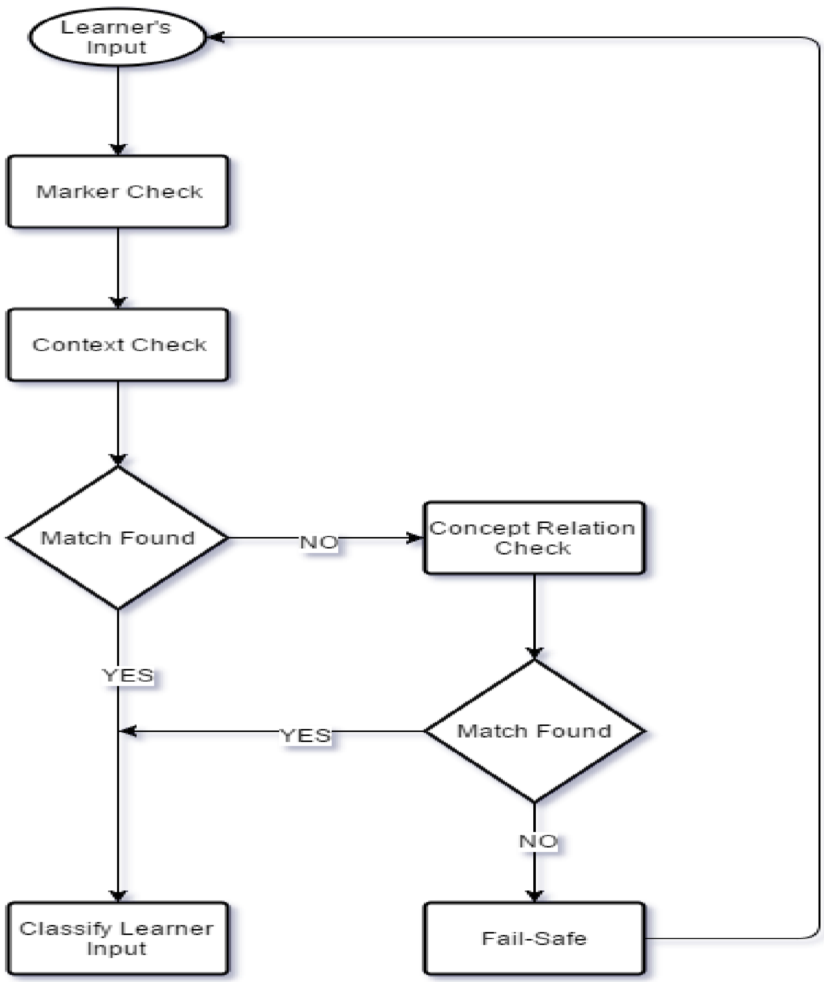


Fig. 8. NDLtutor Steps for Classifying Learner Input

Common Checks: The first round is the *Marker Checks*; when the learner types in input text, the system checks whether the input contains any *special character markers*. These markers include; exclamation marks (!), questions marks (?), periods (…). If any such marker is found, the flag is raised so that the system understands that the resulting classification needs to be updated according to the marker. For example; *No I don't want to!!!* is classified as *disapprove*, however the exclamation marks (!!!) show that the student is *frustrated* by the system repeating its proposal to discuss the topic. The second marker check is used to examine whether the learner's input *starts-with* a marker. For example; *I think, ummm. "I think"* is interpreted as a marker for showing that the learner is *not confident*. If a learner's input begins with this marker, it is automatically considered as a *not confident* utterance. The last marker check is used to inspect the learner's input for the *WH* markers. WH marker includes; *What, Why, Who, How* and indicates a *question/call for action* from the learner. For example; *what should I do?* Starts with the WH marker *what.* This narrows down the classification possibilities of this input to a learner question. The remaining text is then used to identify the type of question. In this example, the input is classified as *help_seeking.*

Domain-dependent utterance checks: After the *Marker Checks* are completed the system uses the *Context Check* to classify the input. In the *Context Check* phase the current context is used to match the input with the corresponding table in the database. This means that if the system asked the learner a *domain question*, then the *expected* user input according to the *current context* is a *domain answer* (domain-dependent utterance), therefore the system looks for a match of the learner's input in the Topic_Discussion_Answers, Concept_Funnel_Answers, and Misconcept_Funnel_Answers tables in the database accordingly. The system uses the sentence structures of the utterances to find a possible match. If a match is found, the learner's input is classified as a *domain answer*, however if no match is found in these tables the system classifies the input as *unknown.*

If a domain-dependent input is classified as *uknown* after the *Context Check,* then the *Concept Relation Check* is carried out by the system. The last phase tries to find a match using the sentence structure of the utterances. However the learner input's sentence structure might be very different from the domain-dependent utterance's sentence structure stored in the database. To overcome this, the system *tokenizes* the input and matches it with the concept list for the current domain question. If a concept is matched in the learner's input with the concept list, then the system classifies the input as a *domain answer,* as it contains a related concept, which means that the learner is trying to answer but his answer text is very different from the text in the answer tables.

Domain-indepdent utterance checks: Any input that is not classified as a domain answer is classified as a domain-independent utterance. Such inputs can occur in all the 3 phases of the NDL dialogue. All such inputs are matched with the *user_utterance_library* according to the surface-level structure of the utterances to classify the input into its corresponding category. If no match is found in the user_utterance_library the input is classified as *unknown.*

Fail-Safe: If all steps fail to classify the learner's input then the system deploys the *Fail-Safe* step which marks the *unknown* utterance as *unidentified* and asks the learner to *rephrase* their input.

4.4 Functional Modules

4.4.1 NLPE Class

It provides the Natural Language Processing functions for the NDLtutor. It defines the following member functions:

- *Tokenizer ($string)*: breaks the learner's input text into words (tokens).
- *Normalizer ($string)*: stems the input using the Porter stemmer algorithm.

- *Utterance_Classifier($string)*: uses Normalized Compression Distance (NCD) algorithm to match the learner's input with the User_Utterance_Library. It returns a set of learner states and a unique utterance classifier. We have tested the Utterance_Classifier() with different threshold settings and benchmarked 60% as the minimum score to qualify for a match.
- *Answer_Scorer($string)*: this returns the score of the learner's utterance when it is classified as an answer. The Answer_Scorer() uses the NCD to match the answer to the list of answers (expected/misconceptions/bad) and returns the score of the highest match. Again a threshold of 60% is set as the minimum score for a match.
- *Concept_Classifier(array)*: matches the tokens in the learner's input text with the list of concepts associated with the expected answer and return a list of missing concepts.
- *Sentence_Generator()*: Discourse Manager (DiM) calls this function to populate a template response from the System_Utterance_Library according to the selected system move.

4.4.2 Dialogue_Manager Class

It controls the dialogue capabilities of the system. It uses the following member functions to achieve this task:

- *Context_Analyzer()*: this function constructs and maintains the *session* variables in order to generate the *current context*. The current context is used for user utterance classification as well as selecting next system move tasks.

- *Rule_Checker()*: accesses the Rules_Library to match rule conditions according to current values of parameters such as user utterance type, number of tries, misconception identified, concept coverage etc.
- *Discourse_Manager()*: this function uses the system move selected by the Context_Analyzer and the Rule_Checker to generate the system response by calling the Sentence_Generator() functionality of the NLPE class.

4.5 Interface and basic functionality of NDLtutor

Fig. 9 shows the interface of the NDLtutor. The interface is divided into 3 columns. The left column contains the learner's own belief base and the LM generated by the system. The learner is allowed to change their belief base using a drop-down list. The middle column contains the section which provides the MCQ tests to assess the knowledge level of a learner in a topic. These results are used to generate the LM of the learner. The right column provides the learner with the chatbot interface. The chatbot provides the following modes of interaction:

1. *Conflict resolution*: This form of interaction is initiated by the system when the learner's change generates a conflict between the belief base of the system and the learner.
2. *Discussion*: The chatbot allows the learner to initiate a discussion about a topic by using the DISCUSS keyword. The result of this discussion is reflected in the system's LM.
3. *Help*: The learner can also ask for quick explanations using the HELP keyword. This functionality allows the learner to use the chatbot to quickly search for

terms/concepts they want to know more about. The system in help mode acts like a glossary and provides the basic definition for that term.

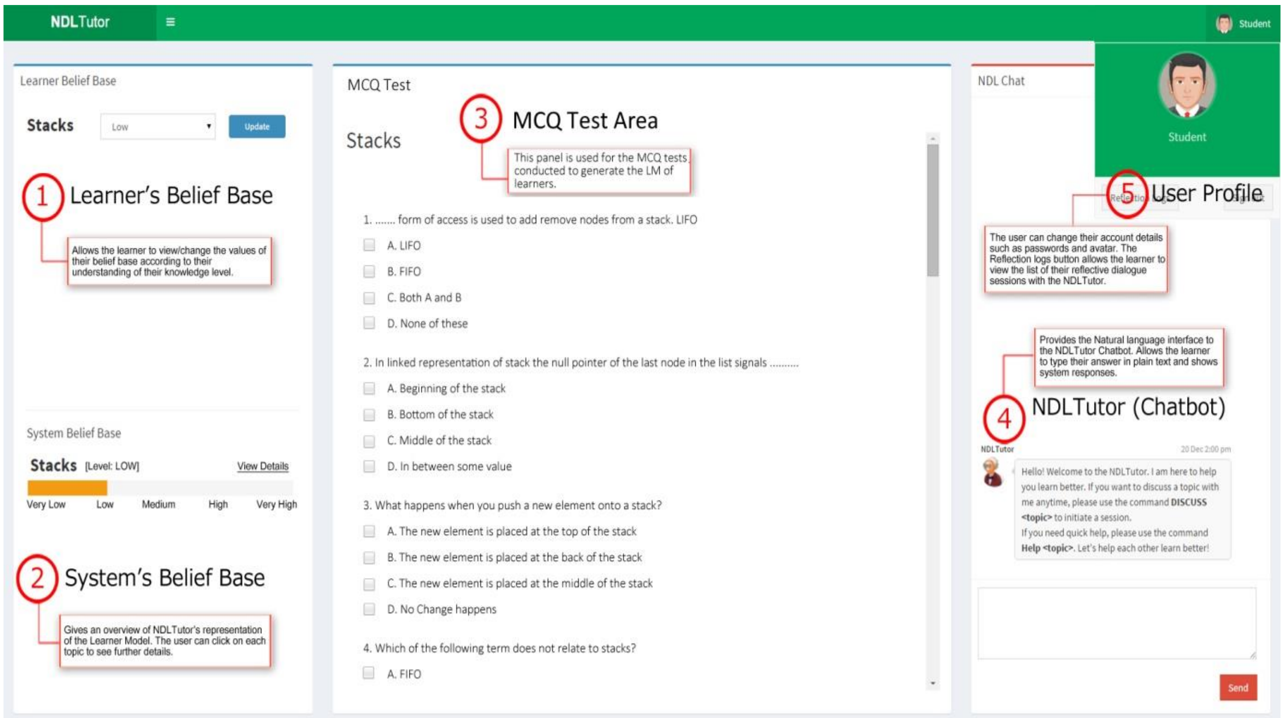


Fig. 9. NDLtutor Interface

4.6 Phases of NDL

NDL acts as a component of an ITS which is normally triggered by an event such as a conflict between the learner's belief in their knowledge and the system's belief about the same. If necessary, an NDL dialogue may also be initiated explicitly by the learner by using the <DISCUSS> command followed by the "topic" they wish to discuss with the system. Every NDL dialogue is comprised of 3 phases: Initialization, Domain Discussion, and Reflection. Dividing a dialogue into 3 phases makes it easier to handle the different inputs expected during each phase, hence the rules can be specifically written for a phase.

4.6.1 Initialization Phase

The first phase of an NDL dialogue is the initialization phase. This phase is used to initialize the values of the State Model for the current dialogue as well as set a foundation for the dialogue to follow. In the initialization phase, the system tries to understand what triggered the dialogue. If a conflict is the cause of a dialogue being triggered, then the system tries to identify the reason behind the learner's change.

During the initialization phase, the system asks the learner for a justification for their action. A weak justification is challenged by the system until the learner is able to justify himself. The values from the State Model are used to select the next move of the system during this phase. The initialization phase can lead to two possible outcomes for the learner; domain discussion or take a test to prove their claim. A learner who chooses to take a test is still encouraged to discuss the topic with the system before they take the test.

4.6.2 Domain Discussion Phase

Once a learner agrees to discuss the topic with the system, the domain discussion phase is initiated. This phase is directly related to the domain knowledge of the learner and discusses the selected topic with the learner using the natural language interface. The discussion starts with a focal question about the topic. The learner responses are classified according to the Utterance Classifiers (UT) and annotated with the State Model (SM) values. Once a response has been classified, the system uses this information to generate the system response accordingly. Each focal question in the domain discussion phase has a list of attributes related to it, which include:

- List of Correct Answers according to the degree of completeness i.e. (EXPERT, INTERMEDIATE, and NOVICE).
- List of concepts related to the topic that constitutes a good answer.
- List of common misconceptions related to the topic/concept.

When a learner's input is classified as an answer, it is firstly matched with the misconception list. If the learner's answer contains any misconception, the system initiates a remedial dialogue which focuses on the identified misconception and the funnels through the related concepts in order to identify the cause of the misconception. If the answer does not contain any misconception, then the system matches the content of the input with the list of correct answers to score the degree of completeness. Finally the learner's input is tokenized in order to match the concepts related to the questions. The scores of the learner's answer and concept coverage are then used to calculate a final score. This score is used by the RC in order to match the corresponding rule in the rule-base to generate the next system response/move. As mentioned before, the implementation details are not a part of this paper and will be discussed in a separate paper. The domain discussion phase is completed once the learner is able to provide an answer with an acceptable degree of completeness along with a medium to high concept coverage.

4.6.3 Reflection Phase

The last phase of the NDL dialogue is one of the core features that distinguish our system from the current OLMs. The reflection phase of NDL is initiated at the end of the domain discussion. This phase is utilized for the explicit reflection for the learner and does not discuss the domain rather the dialogue which just occurred between the learner and the system. During this phase the system engages the learner into a dialogue that encourages self-assessment in the learner. The system discusses the learner's final answer with respect to his initial answer and the discourse that led them to it. The learner is asked open-ended questions that make them compare their answers and assess how they were able to improve upon them. There is no correct or wrong answers in this phase, however the system does keep a track of the learner responses and uses the dialogue history and the learner's verbosity in order to advance the dialogue. This phase allows the learner to reflect upon their discussion with the system. They are encouraged to identify the causes of confusion and how they were able to clarify them. This phase offers the learner a series of questions explicitly targeted towards self-reflection and evaluation. Moreover the learner is also encouraged to evaluate their learning strategy and how they can improve it. The discussions from this phase are saved and are available to the learner as assessment logs they can access at any time.

4.7 Example Interaction

Example: A <u>confident</u> student who is <u>confused</u> about his knowledge level but is <u>interested</u> to discuss his beliefs with the learner

The learner in this example completed the topic of Stack and gave a test where he could not perform very well. The system updated his LM to reflect a "Medium" understanding of the topic. The learner then changed his belief to "Very High", which was very different from that of the system. Therefore the wizard interrupted the proceeding by initiating the following dialogue:

Initialization Phase

[I.1]	Tutor	You have changed your belief in the topic of Stacks to Very High. I am sorry but this is very different from what I believe, therefore I cannot accept this change. Can you tell me why you made this change?
[I.2]	Student	I know more now
[I.3]	Tutor	That's great! However your performance in the topic of Stack has not been very convincing. Do you want me to show you an overview of your performance in the topic of Stack?
[I.4]	Student	Ok
[I.5]	Tutor	You answered 10 questions on the topic of Stacks and got 03 answers correct.
[I.6]	Student	but I'm better prepared now
[I.7]	Tutor	That's great! In order to justify your change, you will have to take a Test to improve your score. Before you take the test would you like to discuss the topic of Stack with me?

[I.1]When the system identified a conflict between the belief values of the learner and the system, the chatbot interrupted the proceedings by initializing the dialogue. The first utterances of the system are a direct cause of the user action i.e. change made to their belief base. The system highlights the value set by the learner which caused the conflict. The system then asks the learner to justify their claim.

[I.2] The learner response is classified by the system as a “weak_confident_justification”. The learner claims that they know more which is classified as a “confident act” but the use of “I think” in the start of their utterance is classified as a “weak act”.

[I.3] since the learner showed some confidence, the wizard starts with a positive feedback to reinforce their confidence. The learner’s justification has been marked as weak and their claim cannot be justified the system rejects their justification. As the current behavioral state of the learner has a positive value, the system gives the proposal to view their performance.

[I.6] The learner’s utterance is classified as confident and interested as he continues to show interest in defending his claim.

[I.7] as the system already provided the learner an overview of their performance and the learner did not revert back his change, the system gives him the proposal to discuss the topic in order to gauge his knowledge level. Discussing the topic instead of directly giving the test allows the system to highlight the weaknesses, confusions and misconceptions the learner might have about the topic during the discussion.

Domain Discussion Phase

[DD.1]	Tutor	What do you know about a Stack?
[DD.2]	Student	It is a data structure that we use to hold elements. We can take elements in and out of it
[DD.3]	Tutor	Can you elaborate on that?
[DD.4]	Student	It is a data structure that holds elements. We use operations to take elements in and out of it.
[DD.5]	Tutor	Ok. A stack is an abstract data type that stores elements in reverse order. What can you tell me about the principle behind storing elements in a stack?
[DD.6]	Student	The first element we put in the stack is the first element that we take out.
[DD.7]	Tutor	So you think the first element we put in the stack is the first element that we take out. I'm afraid that's not correct. Can you explain the order in which elements are stored in a stack?
[DD.8]	Student	On top of each other
[DD.9]	Tutor	Good! Which of the following is an example of such an ordering

scheme:

A) A pile of books
B) A line at a bank counter

[DD.10] Student A

Misconception funneling continues…

[DD.17] Tutor That's right!

So let's revisit the question:

Question What can you tell me about the principle behind storing elements in a stack?

Missing concepts funneling continues…

[DD.25] Tutor Great! Can you summarize your answer now?

[DD.26] Student A stack is an abstract data type that stores elements in reverse order. The principle of storing elements is called LIFO. It states that the LAST element INSERTED in the stack is the FIRST element that is taken OUT of it. It has 2 major operations Push and Pop. Push operation is used to insert an element on the stack and Pop operation is used to remove an element from the stack.

DD.3] This system utterance is a result of the following rule being triggered:

IF user utterance is an answer and the answer does not contain any misconception and the answer score is low and the concept coverage is low and it is the first try by the learner

THEN prompt the learner to elaborate on their answer.

The above rule is intended to allow the learner to rethink about their answer and add more details to it. This encourages the learner to think about the knowledge they have and decide what new information they can add to their previous answer.

[DD.5] This system utterance is a result of the following rule:

IF user utterance is an answer and the answer does not contain any misconception and the answer score is low and the concept coverage is low and number of tries by the learner is greater than 1

THEN give the learner a neutral feedback and provide a partial statement about the answer start funneling the missing concepts

As described previously, each topic has a set of concepts related to it. When the learner's provides an answer, their answer is tokenized and matched with the list of topics which generates a set of missing concepts. The system then starts to funnel through these concepts one by one in order to help the learner build a concrete answer.

[DD.7] The rule behind this system utterance is as follows:

IF user utterance is an answer and the answer contains a misconception
THEN repeat the learner's answer and provide negative feedback and funnel misconception

Just as every topic has a list of concepts associated with it; it also has a list of common misconceptions. When a learner's answer is matched with a misconception, the system highlights the misconception by repeating the learner's answer. The system then start an inner loop of funneling through the concepts related to the misconception.

[DD.17] once all of the concepts related to the misconception have been discussed/covered, the system then asks the learner the same question which triggered the misconception. This is done in order to reinforce that the learner is able to answer the same question without the misconception.

[DD.25] As with the case of a misconception, once the system and the learner have discussed all the missing concepts in the learner's initial answer, the system asks the learner to summarize their answer. This gives the opportunity to the learner to collect all the information into one concise answer. The learner's answer is again scored and checked for any misconceptions or missing concepts. If any such irregularities are found, these are highlighted by the system and a corresponding feedback and proposal are generated depending upon the previous dialogue moves.

Reflection Phase

[R.1]	Tutor	Let's reflect upon the discussion we just had in order to understand what we gained from it. How does your initial answer compare with your final answer?
[R.2]	Student	My first answer was not complete. I missed out important details. The last answer looks much more complete and I think it covers the important points about a stack. I think it is a proper definition of a stack.
[R.3]	Tutor	What was most confusing for you?
[R.4]	Student	I didn't find anything confusing
[R.5]	Tutor	You did hold a misconception regarding the following question: What can you tell me about the principle behind storing elements in a stack? Were you able to clarify your confusion?
[R.6]	Student	Yes
[R.7]	Tutor	What confusions do you still have that you need to clarify
[R.8]	Student	I think I don't have any more confusions about stack
[R.9]	Tutor	What worked well for you?
[R.10]	Student	When you broke down the problem to highlight my misconception. Thinking about what I know about stacks and

imagining it in my mind helped me highlight my misconception and clear out my confusion.

[R.11]	Tutor	What did not work so well for you that you should change?
[R.12]	Student	I need to have a more logical flow in my answers. I should think about what I know and learn to relate the knowledge I have with the question at hand.
[R.13]	Tutor	Good job! I have updated my belief base about your knowledge in the topic Stack to HIGH.

[R.1] This system utterance initiates the reflection phase. The system asks the learner to compare their initial answer to their final answer. There is no correct or wrong answer for this phase. However the system does react to the input given by the user in this phase.

[R.5] this utterances shows how the system reacts to the learner's answers in this phase. As the misconception flag had been raised during the domain discussion phase, the system expected the learner to talk about their confusion of "storing principle in stacks". However when the learner claims to have had no confusions, the system highlights the question which generated the misconception so that learner can evaluate their input.

[R.13] after the dialogue is completed, the system updates the LM accordingly. If the learner is able to provide a suitable answer to the initial question by the end of discussion, the system updates its belief base according to the learner's change.

5. EVALUATION 1 - DIALOGUE MANAGEMENT CAPABILITIES

The first evaluation study of the NDLtutor was conducted to assess the dialogue management capabilities of the system, use of affective and behavioral states to control dialogue flow and using a natural language interface as the communication medium. This evaluation focused on:

1. Quality of dialogues produced by the system.
2. Completion of meaningful dialogues.
3. Use of affective and behavioral states to control the flow of dialogues
4. Use of reflection dialogues as a means to promote metacognition and self-assessment.

5.1 Participants

The participants for this evaluation were 20 students from the undergraduate Software Engineering program at Bahria University Islamabad, Pakistan. These students were at the time enrolled in the data structures course and had just recently been introduced to the topic of stacks. The students had no previous experience of using an ITS system.

5.2 Method

Before the start of the session the students were given an overview of the system and the functionality available to them by the first author through a video conference session on Skype. They were introduced to the interface and the possible modes of interaction they could use. They were encouraged to inspect/change their belief base whenever they felt necessary. An initial LM was generated using the test scores of the students in their class exam and the lecturer's personal feedback about each student. The LMs were intentionally altered to show the student's knowledge level to be less than their original knowledge level. This was done to motivate the students to challenge the system's representation. The experiment was conducted in the computer lab of the Software Engineering department and a local instructor (Senior Lecturer) was present at the time of the experiment. The first author was also virtually present via Skype to answer any question.

Each student logged into the system for an individual session which typically lasted between 15 to 20 minutes. Individual logins were provided so that logs of individual interactions could be recorded in the database. The students could view the system's representation of their LM which was a simple skill-meter. To reinforce interaction symmetry the students were allowed to invoke the chatbot directly by using the DISCUSS <topic> command. Hence a negotiation session could be initiated by the system when a student made a conflicting change to their belief base or it could be initiated by the student by using the DISCUSS command. Out of 20 negotiation sessions recording in the experiment, 18 (90%) sessions were initiated by the system whereas 2 (10%) sessions were student-initiated. A post-experiment survey was conducted to get the student's feedback about the system. Self-reflection dialogue logs were also used to analyze the learner's interest and reaction to the dialogue itself.

5.3 Learner Interactions

During the course of the evaluation, different interactions were seen depending upon different characteristics of the students. The major characteristics that influenced a session include:

1. *Knowledge level* – the difference between the knowledge levels of the students had a major impact on the interaction. The interaction time of the more knowledgeable students was considerably shorter than that of the less knowledgeable students. This was an obvious observation since the more knowledgeable students were able to justify the change they had made by discussing the topic with the system. Their answers were more concrete and well-formed which left little room for the NDLtutor to continue the dialogue. Fig. 10 shows such an interaction log. The student's answer score is high and the concept coverage is high as well, therefore the system does not need to deploy any funneling strategy. Another important observation from this interaction is that the reflection phase is also influenced by the depth of the discussion. Since the depth of dialogue is so shallow that the system cannot engage the learner in a reflective dialogue regarding their discussion.

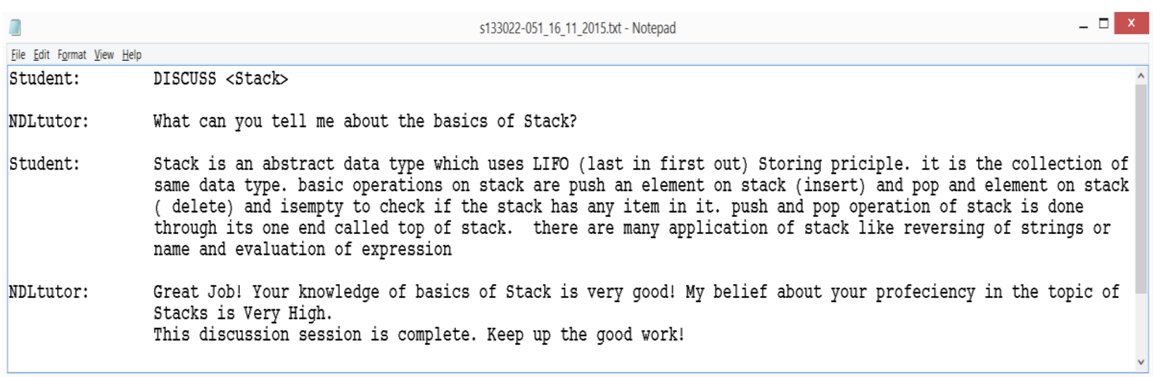
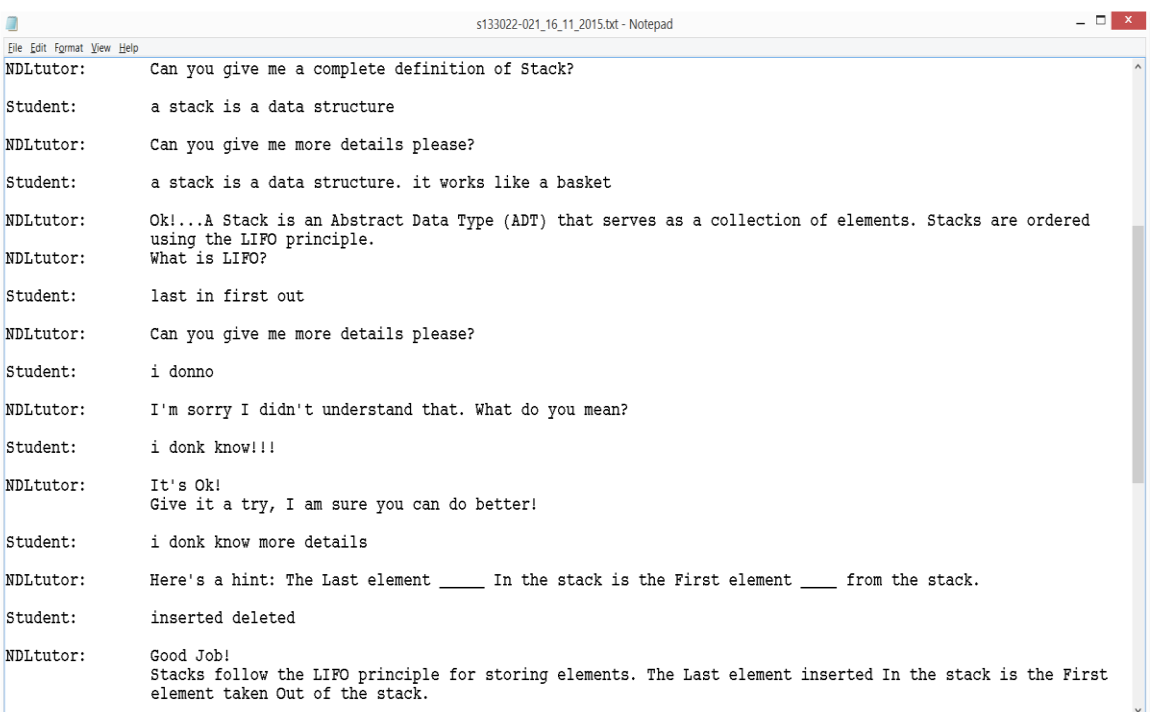


Fig. 10 Knowledgeable Learner Interaction with NDLtutor

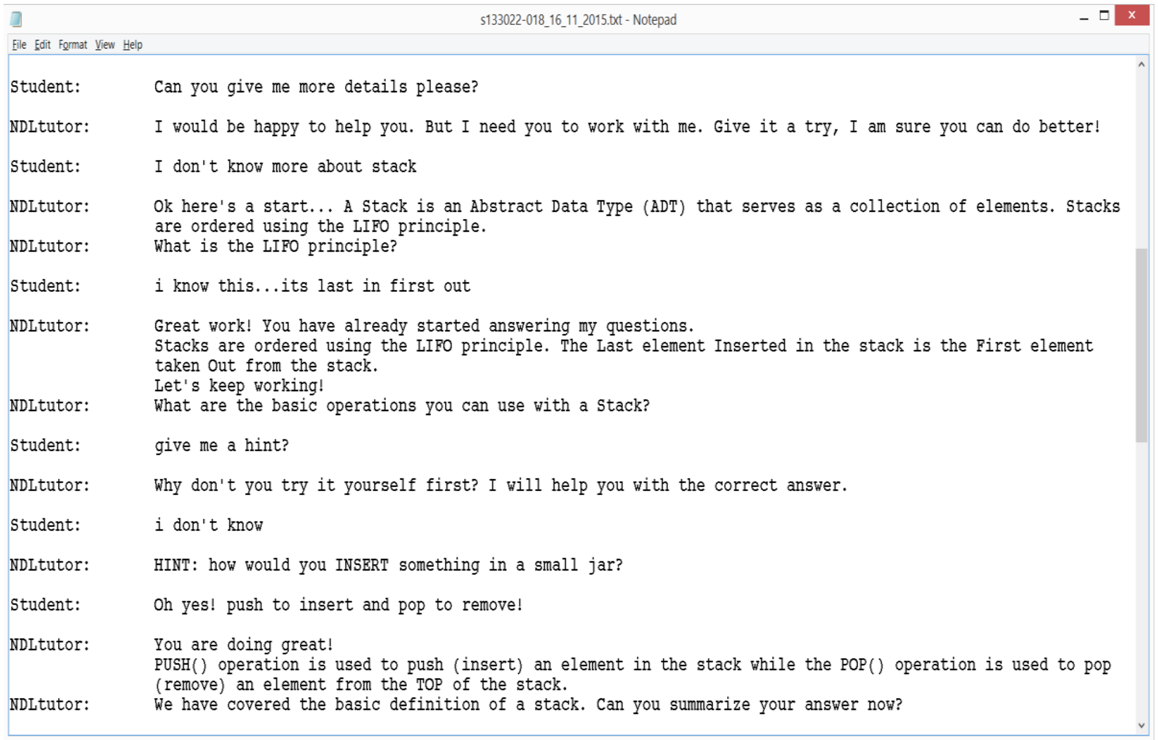
Contrary to this, the interaction sessions of the less knowledgeable students were longer and provided more insight to the evaluation. These interactions followed different paths depending upon the student's reply and therefore the NDLtutor needed to make more strategic dialogue decisions. The basic markers for such students were the low answer score and low concept coverage. This provided more room for discussion as the system could ask a series of funneling questions in order to cover the topic. This category of students was the main focus of our study as they allowed us to test our system's dialogue management capabilities. The dialogue fragment in Fig. 11 shows an interaction log of a student with low knowledge of stacks. As seen the student's answer is not complete and allows the system to engage in a funneling discussion about the topic. Such an interaction also provided a gradual transition into the reflection phase.




```
NDLtutor:       Can you give me a complete definition of Stack?

Student:        a stack is a data structure

NDLtutor:       Can you give me more details please?

Student:        a stack is a data structure. it works like a basket

NDLtutor:       Ok!...A Stack is an Abstract Data Type (ADT) that serves as a collection of elements. Stacks are ordered
                using the LIFO principle.
NDLtutor:       What is LIFO?

Student:        last in first out

NDLtutor:       Can you give me more details please?

Student:        i donno

NDLtutor:       I'm sorry I didn't understand that. What do you mean?

Student:        i donk know!!!

NDLtutor:       It's Ok!
                Give it a try, I am sure you can do better!

Student:        i donk know more details

NDLtutor:       Here's a hint: The Last element _____ In the stack is the First element ____ from the stack.

Student:        inserted deleted

NDLtutor:       Good Job!
                Stacks follow the LIFO principle for storing elements. The Last element inserted In the stack is the First
                element taken Out of the stack.
```

Fig. 11. Less knowledgeable learner Interaction with NDLtutor

2. *Affective and Behavioral states* – one of the main research issues we are investigating in this study is the impact of using affective and behavioral state of a learner to make dialogue control decisions. The influence of such states was clearly observed during the review of the interaction logs. Fig. 12 shows an excerpt of such a dialogue where the NDLtutor is able to identify a specific state and use this information to control the dialogue flow. The student is not confident about his knowledge of stacks but shows interest to interact with the NDLtutor and remains on topic. The NDLtutor provides maximum scaffolding to the student as they appear to try harder with every question answered. The student tries to ask for help repeatedly and this is caught by the NDLtutor. To cope with this, the system encourages the student to try to answer by himself before he could receive help/hint. In the future development iteration, the use of hints/help feature would be further formalized so that the NDLtutor can ensure maximum input from the student before providing assistance on domain knowledge.



s133022-018_16_11_2015.txt - Notepad

File Edit Format View Help

Student: Can you give me more details please?

NDLtutor: I would be happy to help you. But I need you to work with me. Give it a try, I am sure you can do better!

Student: I don't know more about stack

NDLtutor: Ok here's a start... A Stack is an Abstract Data Type (ADT) that serves as a collection of elements. Stacks are ordered using the LIFO principle.

NDLtutor: What is the LIFO principle?

Student: i know this...its last in first out

NDLtutor: Great work! You have already started answering my questions.
Stacks are ordered using the LIFO principle. The Last element Inserted in the stack is the First element taken Out from the stack.
Let's keep working!

NDLtutor: What are the basic operations you can use with a Stack?

Student: give me a hint?

NDLtutor: Why don't you try it yourself first? I will help you with the correct answer.

Student: i don't know

NDLtutor: HINT: how would you INSERT something in a small jar?

Student: Oh yes! push to insert and pop to remove!

NDLtutor: You are doing great!
PUSH() operation is used to push (insert) an element in the stack while the POP() operation is used to pop (remove) an element from the TOP of the stack.

NDLtutor: We have covered the basic definition of a stack. Can you summarize your answer now?

Fig. 12. NDLtutor dialogue excerpt showing system's adaptation to the learner's response patterns

The student's confidence was found to be more of a personality trait and not directly associated with their knowledge level as we observed less knowledgeable students to show confidence in their interactions as well. However interest and engagement levels were found to be more influenced by the student's knowledge level. Students with very low knowledge of the topic inclined to show less interest in the discussion and repeatedly asked the system for help. This highlighted an important caveat that was not fully taken into consideration during the initial analysis of the system. The act of *gaming* the system was seen in some interactions where the less knowledgeable students were uninterested in the domain discussion and repeatedly asked the system to provide them with help.

Similarly the students used the system's answers during the domain discussion phase and copy-pasted them as their final answer to receive a high score. These new insights were recorded for the next development iteration of the system.

5.4 Results and Discussion

As stated earlier the main focus of the evaluation was the *dialogue management capabilities* of the NDLtutor. The results collected from the experiment consisted of two parts; the interaction logs and the post-experiment survey. Table 6 shows the results of the survey conducted at the conclusion of the experiment phase. The findings were in line with previous researches on tutorial dialogue and learning effectiveness (M. G. Core, et al., 2003, C. Rose et al., 2003, S. Katz, D. Allbritton and J. Connelly, 2003).

While analyzing the results of the survey, the most prominent discovery was the high rate of acceptance from the students. In our understanding, a major factor leading to this outcome was the "Asian culture" influence. We had actually discovered this in one of our earliest survey's for another study.

Table 6

Post-Experiment Survey results

	<strongly agree……..strongly disagree>					Mean
	(5)	(4)	(3)	(2)	(1)	
Do you think discussing a topic with the chatbot was a good way of justifying your proficiency in that topic?	16	3	1	0	0	4.75
Do you think discussing a topic with the chatbot helped you improve your understanding?	13	4	2	1	0	4.45
Was the chatbot able to correctly understand what you wanted to say?	16	2	0	1	1	4.55
Were the system's reactions to your inputs valid?	14	4	1	0	1	4.5
Did the chatbot make the negotiation process easy?	14	2	3	1	0	4.45
Did the use of off-topic discussion/small talk make dialogue feel realistic/natural?	4	7	6	2	1	3.61
Did you find the reflection dialogue beneficial?	16	1	3	0	0	4.65
Would you be interested to use a similar system in the future as a study resource?	18	1	1	0	0	4.85

Asian students tend to be very respectful and polite in their interactions with their tutors. This is a major factor that influences their reactions and it was again prominent in the results of this survey. Having highlighted this, we do recognize that the students were actually very interested and impressed by their interactions with the NDLtutor. They were intrigued by the idea of discussing a topic with a computer tutor in a natural language setting. The authors received multiple emails and Facebook comments from students showing interest in the NDLtutor and volunteering for future experiments. The interaction logs were analyzed in the light of four major criteria set for the experiment.

1. *Quality of dialogues produced by the system* – the first criterion was related to the quality of the dialogues generated automatically by the system. It is imperative that the system is able to generate dialogue that engage and motivate students. The analysis of the interaction logs revealed that the system was indeed able to initiate and conduct fruitful dialogues with the student. The user utterance classification scheme that was defined in the earlier section was validated by reviewing the interaction logs and further supported by the survey results where 90% of the students agreed that the system was indeed able to understand their inputs. In the case of a mismatch the system asked the students to rephrase what they had said which proved to be a good strategy to improve the system's understanding of the inputs. All the sessions were completed successfully which showed the robustness of the system's dialogue management capabilities.

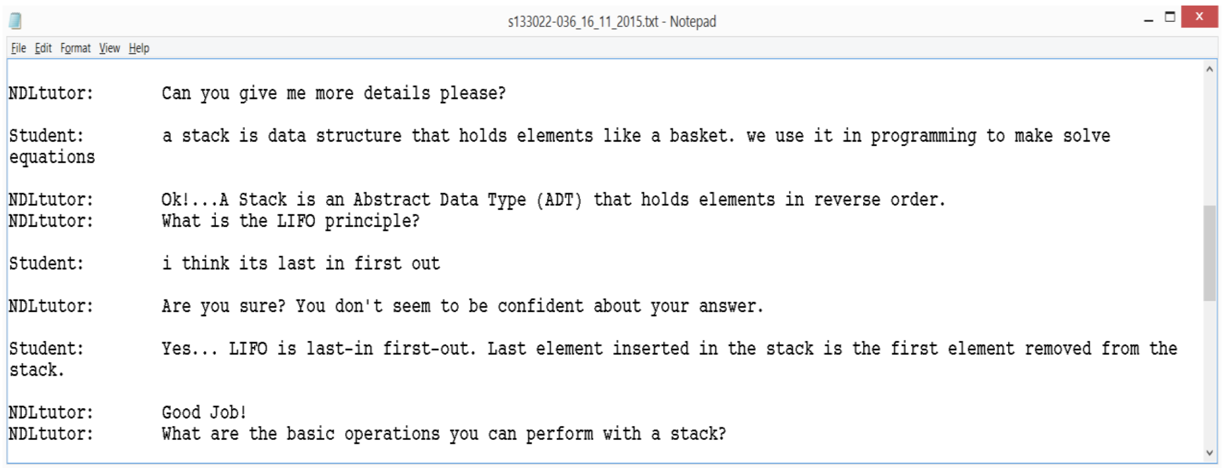
2. *Completion of meaningful dialogues* – as discussed above, all dialogue sessions terminated successfully with the mutual agreement between the student and the NDLtutor. The inclusion of small talk in the system corpus proved to be a valuable decision during the system design phase. The post-experiment survey showed that students thought that a minimum amount of small talk made the system feel more realistic and natural. The students also appreciated the

misconception funneling functionality of the system and found it to be really helpful in correcting their erroneous beliefs. Moreover it provided them with a chance to discuss the topic in more detail which promoted deeper learning.

3. *Use of affective and behavioral states to control the flow of dialogues* – as seen in Fig. 9 above, the use of affective and behavioral states to control the flow of dialogues allowed the system to be more flexible and naturalistic in its responses to the students than the negotiation mechanisms of other existing OLMs. One interesting observation from the interaction logs was that in the case of the system identifying the student's mental state erroneously, the impact on the dialogue was not drastic. This was due to the fact that the system used the information about the student's states in conjunction with the current context of the dialogue. An example of such an occurrence is seen in Fig. 13. The system identifies the student's behavior as "not confident" and raises this point to confirm its classification. The student reacts by reaffirming their belief in what they had said. Since their answer was correct, the system accepts their justification and proceeds to the next dialogue move.

4. *Use of reflection dialogues as a means to promote metacognition and self-assessment* – the survey results in Table 6 confirm that the students found the reflection phase to be very helpful in promoting self-reflection. Irrespective of the fact that the more knowledgeable students did not have the reflection phase in their interaction, all the participants unanimously agreed to the usefulness of

having a reflective dialogue at the end of the domain discussion. The option of viewing the reflection logs was also welcomed by all the participants. The students accepted that a reflection log would allow them to reflect upon their learning periodically.


s133022-036_16_11_2015.txt - Notepad

File Edit Format View Help

```
NDLtutor:       Can you give me more details please?

Student:        a stack is data structure that holds elements like a basket. we use it in programming to make solve
equations

NDLtutor:       Ok!...A Stack is an Abstract Data Type (ADT) that holds elements in reverse order.
NDLtutor:       What is the LIFO principle?

Student:        i think its last in first out

NDLtutor:       Are you sure? You don't seem to be confident about your answer.

Student:        Yes... LIFO is last-in first-out. Last element inserted in the stack is the first element removed from the
stack.

NDLtutor:       Good Job!
NDLtutor:       What are the basic operations you can perform with a stack?
```

Fig. 13. NDLtutor dialogue excerpt showing system's confirmation of the student's confidence in his response

Due to the limited empirical data, the question about how the students may use such reflection logs was out of the scope of this evaluation. This is an interesting prospect to investigate and therefore will be a part of the future evaluations of the system. A few dialogue logs presenting the diverse interactions that took place during this evaluation are added in Appendix B.

6. EVALUATION 2 – PEDAGOGICAL IMPLICATIONS

The second evaluation study of the NDLtutor was conducted to assess the pedagogical implications of the NDLtutor. This evaluation focused on:

1. Improvement in Self-assessment accuracy.
2. Effects (if any) of the reflection phase on the Self-reflection skills of learners.

6.1 Participants

The participants for this evaluation were 20 students from the undergraduate Software Engineering program at Bahria University Islamabad, Pakistan. 15 students had participated in the 1st evaluation while the remaining 5 had no previous experience of using an ITS system.

6.2 Method

As with the first evaluation, before the start of the session the students were again given an overview of the system and the functionality available to them by the first author via Skype video conferencing session. This was done to accommodate the 5 new students who had volunteered for the study. They were introduced to the interface and the possible modes of interaction they could use. The domain was extended to include the topics of Queues and Linked Lists in addition to the topic of Stacks. The students were asked to concentrate on one topic per session. A single topic was selected per session to ensure maximum concentration and engagement of the students

For this evaluation the system implementation was updated so that the students had to make an initial self-assessment for each topic after logging into the system. The self-assessment scores were divided into a 5 confidence bands namely; Very Low, Low, Moderate, High, Very High. Each of these bands had a corresponding numerical value assigned to it as follows; Very Low = 0, Low = 1, Moderate = 2, High = 3 and Very High = 4. Once the students completed the self-assessment they were provided with the option of taking the MCQ test. The MCQs for the topic of Stack were updated from the previous version of the system in order to generate fresh results. The system's learner model for a topic was updated once the student completed the MCQ test for that specific topic. Once the student completed the MCQ test for a selected topic, the system's learner model was updated for that topic. The system then asked the student to confirm their initial self-assessment or update it if they deemed necessary. As the students confirmed/updated their belief base, conflicts occurred between the belief base of the learner and that of the system and at this point the system initiated a dialogue session for the corresponding topic. At the end of the dialogue session the system either accepted the student's change (system's belief base changed) or rejected it (system's belief base remained unchanged). When the student logged off from the system, they were alerted about any discrepancies between the belief bases as a last resort to encourage them to review their belief base in contrast to that of the system.

It is worth mentioning here that in the 1st evaluation study, we intentionally manipulated (reduced) the system's belief score about the learner's knowledge level to motivate the learners to challenge the system in order to evaluate the dialog management capacity of NDLtutor for maximum dialogue interactions. However in the 2nd evaluation study, no such manipulations were made to ensure a natural dialog activity of learners in the normal context. As a consequence not all of the students engaged in a dialogue with the system for every topic.

To analyze the effects on the self-reflection of the students, the reflection phase was updated to include a scoring mechanism. The reflection phase consisted of 5 questions allotted one point, hence 5 points per reflection session for each topic. These questions were structured specifically so that the answers could be quantified in terms of a numerical value. For example, the learner was asked to score their initial answer on a scale of 0 to 10 (0: minimum, 10: maximum). The student's answer was then compared with the system's score of their initial answer. This was done to test whether after completing the domain discussion phase, the student would be able to evaluate their initial answer better. If the student's scoring of their initial answer matched with that of the system, they were awarded a single point. Details about these measures are described together with the results presented below. A few excerpts of the dialogue logs from this evaluation are added in Appendix C.

6.3 Results

This evaluation focused on the effects of the NDLtutor on the self-assessment and self-reflection skills of the students. To gauge the effects on the self-assessment of the students, we used two discrepancy measures introduced in a previous study on the evaluation of CALMsystem (Kerly, A., Ellis, R., Bull, S., 2008). The selected measures are:

1. *Self-assessment accuracy*

The self-assessment scores for a student were calculated as the numerical sum of the student's belief across all 3 topics. Hence the highest possible self-assessment score for a student could have a value of 12. The self-assessment error was calculated for 2 cases; a) Before Negotiation and b) After Negotiation. Fig. 14 shows the results of the self-assessment evaluation. The mean self-assessment error before negotiation for all the 20 students was 1.6 with a standard deviation of 0.860. The mean self-assessment error for all the 20 students was reduced to 0.65 after negotiation with a standard deviation of 0.653. Hence significant improvements ($t= 3.83$, $p<0.0005$) in self-assessment were made by the students after negotiating with the NDLtutor. Fig.14 shows that the students did change their self-assessments after negotiating with the NDLtutor and their final self-assessments at the end of the evaluation study, more closely matched with the system's assessment about their knowledge. Out of the 19 students that engaged in a dialogue with the system, 17 (89.4%) students made changes to their belief base that resulted in the reduction of the self-assessment error whereas 2 (10.5%) students did not make any changes to their belief base after negotiation. The belief bases of 8 (42%) students matched completely with that of the system at the end of the experiment.

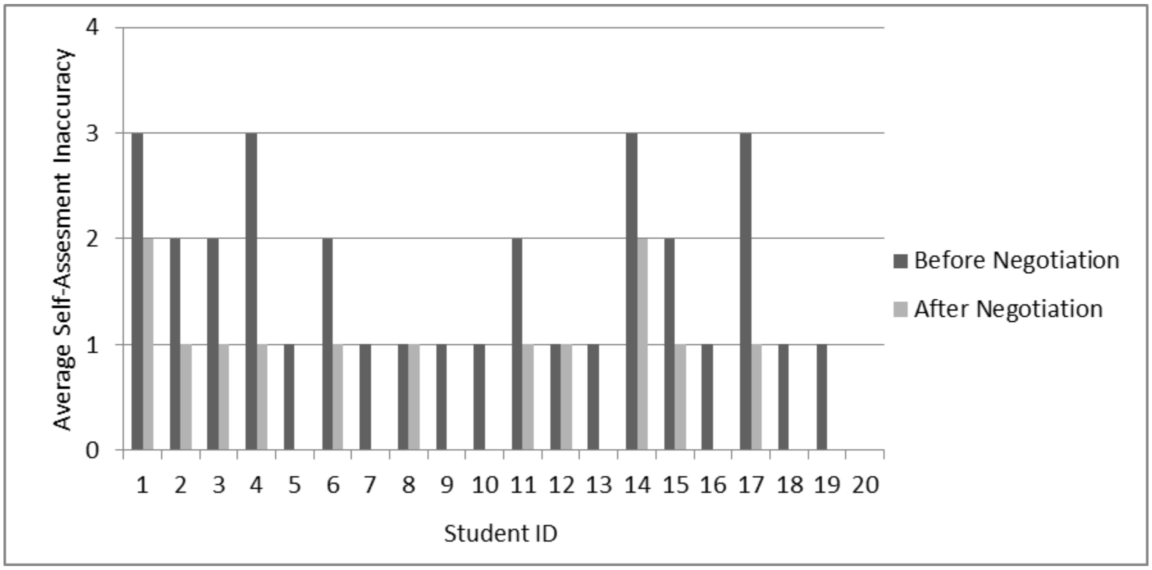


Fig. 14. Self-assessment inaccuracy Before & After Negotiation with NDLtutor

2. *No. of Topics with discrepancy*

The second discrepancy measure adopted from the previous study on CALMsystem was the reduction in the No. of Topics with discrepancy. This measure was calculated as the difference of the number of topics where the student's belief base value was different from that of the system before negotiation, to number of topics where the student's belief base value was different from that of the system after negotiation. The mean number of topics with discrepancy before negotiation was 1.45 across the 3 topics for all the students. The mean number of topics with discrepancy after negotiation reduced to 0.65 indicating that there was significant reduction ($t=3.72$, $p<0.0006$) in the number of discrepancies after negotiating the topics with the NDLtutor. Fig. 15 shows the number of topics with discrepancy for each individual student. Out of the 19 students that engaged in negotiation with the system, the number of topics with discrepancy reduced for 15 (78.9%) students whereas the number of topics with discrepancy did not change for 4 (21%) students at the end of the experiment. The high percentage of students with reduction in the number of topics with discrepancy indicates that the students did in fact reassess (review) their belief bases after negotiating with the NDLtutor.

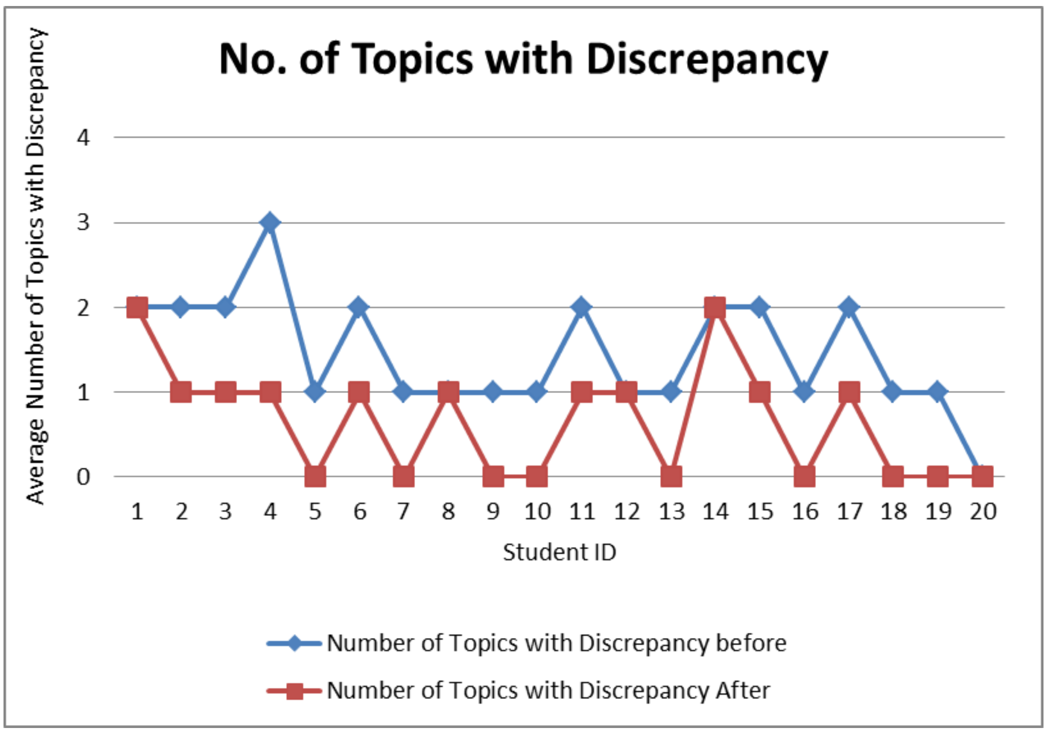


Fig. 15. Number of Topics with discrepancy Before & After Negotiation with NDLtutor

3. Effects in Self-Reflection

Promoting metacognitive skills of the learners has always been one of the major objectives of OLMs. Opening up the learner model to the learner was intended to maximize learner participation as well as promote self-reflection in learners (Bull, S. & Vatrapu, R., 2012). Learner's self-assessment of their belief in their knowledge level is considered as a reflective activity. An improvement in self-assessment has been used as an indicator for promotion of self-reflection (Kerly, A., Ellis, R., Bull, S., 2008, Dimitrova, V., 2003). How the learner is reflecting is mostly implicit as self-reflection is domain/task-independent. This implicitness of self-reflection skills of a learner and their ability to use such skills makes formally analyzing and assessing such skills a difficult task. It has been argued in the research on assessing and explicitly promoting self-reflection in learners that the system should focus on providing the learners with the tools to engage in some form of reflective activity such as; self-assessment of their belief-base, skill diaries (Long, Y., & Aleven, V. 2013) and self-explanations (S. Gross, S., et. al, 2015).

Based on the concept of skill dairies we introduced a reflection phase at the end of each dialogue session in the NDLtutor. The idea is to encourage the learner to reflect upon their discussion with the system. Our aim is to provide the learner with support in a domain-independent form of interaction that helps them in analyzing/realizing how they answered the system's questions during the domain discussion, what were the problems they encountered, what concepts they missed or what misconceptions were highlighted during discussion. To enable the system to analyze the learner's input during this phase, we introduced an informal formative assessment that uses 5 questions which can be quantified by the system to generate a reflection score at runtime. The reflective score for each student is calculated for each individual session by the system. The 5 questions carrying 1 point each are as follows:

Q1. *On a scale of 0 to 10 (0: Minimum, 10: Maximum) how would you rate your first answer?* – This question is used to analyze the learner's ability to evaluate their initial answer. The value of scale provided by the learner is converted into a percentage value and compared with the system's evaluation of the learner's initial answer i.e. (answer score + concept coverage). If the learner's evaluation score matches the evaluation score of the system (permitted variance: ±15%), the learner is awarded 1 point, otherwise zero point.

Q2. *How is your last summarized answer different from your first answer?* – The answer to this question is tested for learner's verbosity and their ability to identify the incompleteness of their initial answer. The learner's answer is analyzed for statements relating to incompleteness of their initial answer as well as missing details. If the learner's answer includes these markers, the system awards 1 point.

Q3. *What were the concepts that you missed?* – This question is used to check whether the learner is able to recall the concepts they missed in their initial answer. If the learner had missed some concepts during the domain discussion, then the system asks them to list these concepts. The learner is awarded 1 point if he is able to list all the concepts he missed during the domain discussion. If the learner did not miss any concepts, the system accepts "No" as an answer and awards 1 point.

Q4. *Did you encounter any misconceptions?* – Similar to Q3, the learner is asked to state the misconceptions (if any) that were encountered during the domain discussion. If no misconceptions were encountered the system accepts “No” as an answer and awards 1 point.

Q5. *Did you improve your understanding/knowledge on the topic?* – Whether the learner’s belief about their understanding of the topic changed after their interaction with the system. The learner’s answer is analyzed with respect to the score of their final summarized answer in the domain discussion phase. If the learner’s final answer score is higher than their initial answer score then the expected answer to this question is “Yes”, which earns the learner 1 point. On the other hand, if the learner provides “No” as an answer to this question and their final answer score is low, the system allocates 1 point and asks them to elaborate on the reasons that might have hindered their learning.

It is necessary to state that this reflective score is not intended to be used as a formal assessment measure; instead we argue that such a score can be used to study the different correlations between the learner’s self-assessment beliefs, their performance in the domain dialogue, and their responses in the reflection phase over a period of time. Fig. 16 shows the reflection scores of all the 20 students that participated in the evaluation. The scores are shown for each reflection session a student engaged in. Here is it important to point out that the reflection phase was only initiated for a student who was unable to provide a high scoring answer to the initial Domain Discussion Question in the domain discussion phase. This means that not all of the students engaged in a reflection session across all the 3 topics. This explains the empty columns of the students for some sessions in Fig. 16. That is to say, an empty column does not show a 0 reflection score, it only indicates that the student did not engage in a reflection phase for the specific session. The mean reflection score across all the 3 sessions was 3.82 which show that most students were able to get high scores during the reflection phase. 5 (25%) students engaged in the reflective dialogue of Session-I (Stacks). This number increased in Session-II (Queues) to 12 (60%). A similar number of students 12 (60%) engaged in the reflective dialogue for Session-III (Linked Lists). The mean reflection score of Session-I was 2.8 whereas the mean reflection scores for Session-II and Session-III were 3.83 and 4.25, respectively. The stats reveal that more students engaged in the reflective dialogue than those who engaged in the initial session, and as the complexity of the domain topic increased. Some further observations are as follows:

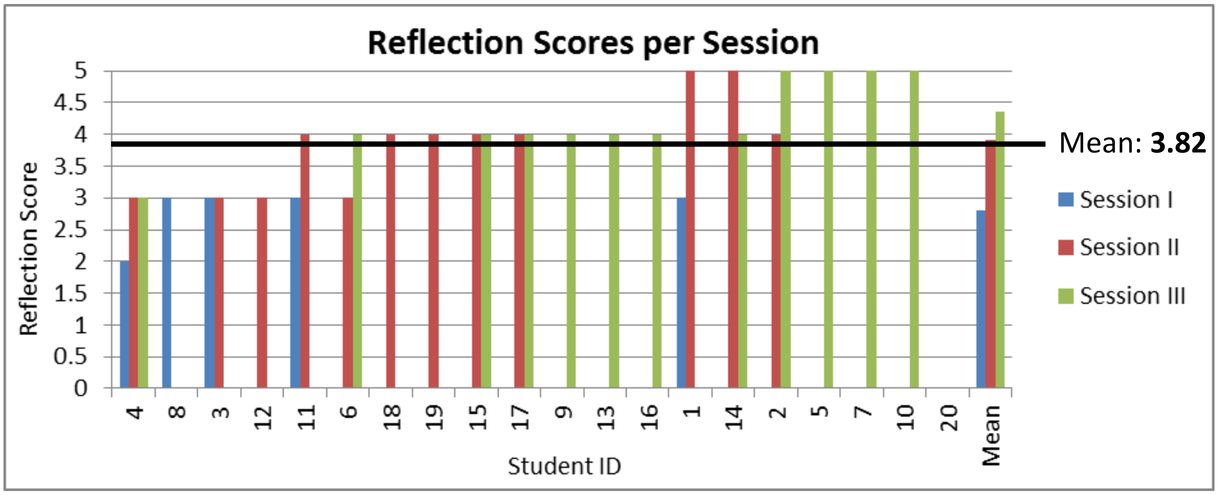


Fig. 16. Self-Reflection scores (in ascending order) of individual students across all sessions

1. Not all students engaged in reflective dialogues for all the 3 sessions. As defined earlier, a reflective dialogue is only conducted after the student has completed the domain discussion dialogue with the system. If there is no discrepancy between the learner's belief and that of the system, or if the learner accepts the system's belief value and updates their own belief base to match the system's belief without challenging the system, in such a scenario no dialogue session is conducted and hence the student does not engage in a reflection dialogue. For instance, in Fig. 16, Student#1, has no reflection score for session-III as this student did not challenge the system and accepted the system's inference about his knowledge level on the topic of Linked Lists. Only Student#4 engaged in all the 3 reflection sessions.

2. When there is a discrepancy between the system's belief and that of the student, then the system initiates a dialogue, if the student is able to answer the domain question in their first attempt to an acceptable standard, then the system does not have any room for a reflective discussion. Hence for students with high level of knowledge, the possibility of engaging in a reflection session is minimal. This can be seen in Fig. 16 as student # 20 has no reflection scores. This student was able to prove their knowledge during the MCQ tests for all topics, so that the system had no rationale to challenge his beliefs.

3. We found two encouraging suggestions. The first one is that the reflection score for each student remained neutral or positive and did not decline over multiple sessions except Student #14. The second is that the average of the reflection scores for each session increases as shown in the "Mean" column of Fig. 16. Although we cannot claim it with statistical significance in this experiment, whether interacting with the system multiple times had an effect on the learner's answers or did it play any role in training the learners to answer better is an interesting topic and it would be worth exploring in future studies.

4. Another interesting observation is that the reflection scores of the learners suggest a direct correlation to their confidence in their knowledge level of the topic. To find the correlation between the learner's confidence in their

knowledge and their reflection score, we calculated the Spearman's rank correlation coefficient for *tied* data. The average correlation coefficient between the learner's confidence in their knowledge and their reflection score across all 3 topics was found to be 0.674, which shows a positive correlation that is statistically significant at the 0.01 level (for n=20) between the learner's confidence in a topic and the reflection scores. In fact, students who chose low confidence values i.e. "Very Low" and "Low" in their belief bases tended to accept the system's inference without challenging the system. These students were also observed to generally have a below average reflection score in the reflection phase (Student#1, Student#3, Student#4, Student#8 and Student#11 in Fig.16). However, the students who were more confident about their knowledge level and chose "High" or "Very High" values in their belief bases challenged the system more and also scored higher in the reflection phase. For instance, all the students in Session-III had chosen "High" or "Very High" as their belief base value for the topic of Linked List. The only exception was Student#4 who had chosen "Moderate" and this student scored below average in the reflection dialogue. Whether or not there is a direct and strong correlation between the student's confidence in their knowledge level and their reflection scores is an interesting observation. However, this requires a greater number of interaction sessions to be further investigated.

5. While the above observation presents the correlation between learner's confidence of his/her knowledge level and the reflection score, here we

discuss about the correlation that was observed between the learner's *actual* knowledge level which is in the system's belief base and his/her reflection score. The correlation between the learner's knowledge as assessed by the system and their reflection scores was also calculated using the Spearman’s rank correlation coefficient for *tied* data. The average correlation coefficient between the learner's knowledge as assessed by the system and their reflection scores across all 3 topics was found to be 0.68, which shows a positive correlation that is statistically significant at the 0.01 level (for n=20) between the learner’s knowledge level in a topic and their reflection scores. From further analysis of the results we were able to define 3 broad categories of students; 1) Below Average, 2) Average and 3) Above Average according to their knowledge level as assessed by the system during the MCQ tests. Students in each category shared similar characteristics. The Below Average students showed the tendency to only challenge the system’s beliefs on the basic/easier topics i.e. Stacks or in some cases Queues. These students were mostly unable to defend their claim during the domain discussion phase and also scored below average in the reflection phase (Blue bars in Fig. 16). The Average category of the students can be considered as the ideal candidates for the system as these students demonstrated an average level of knowledge and high confidence in their assessments. These students challenged the system’s beliefs across all the topics and were mostly successful in defending their beliefs. Their reflection scores (Red bars in Fig. 16) were also closer to the overall mean score. The Above Average students were the ones who had a high or very high level of knowledge and were also confident about their beliefs. Such students mostly engaged in the dialogue with the system for the advanced topic i.e. Linked List. The Green bars in Fig. 16 show that almost all the students who engaged in the dialogue related to the advanced topic had an above average score. This observation is in line with previous research that students who have better metacognitive skills perform much better than students who have weak metacognitive skills (Swanson, H. L., 1990, Schraw, G. & Dennison, R.S., 1994).

7. CONCLUSIONS & FUTURE WORK

7.1 Summary

Open Learner Models maximize learner involvement by engaging them in a process of collaboratively constructing and maintaining their learner model (Bull, S., Brna, P., Pain, H., 1995, Dimitrova, V., 2003, Kerly, A., Ellis, R., Bull, S., 2008). The research on OLMs has shown to produce significant learning gains. Negotiated OLMs utilize different interaction strategies to enhance self-assessment and promote self-reflection in learners. Conversational agents have been used in this regard to provide naturalistic mode of interaction between the learner and the system. This not only eased the communication process but also improved the self-assessment accuracy of the learners. Following the success of using chatbots in OLMs, this study investigated the possibilities of enhancing the capabilities of such chatbots and their implications on the learner's learning.

This research introduced the paradigm of Negotiation-Driven Learning (NDL) which uses a chatbot employing Interest-Based Negotiation strategy to discuss the learner model with the students. We discussed the use of approximations of a learner's affective & behavioral states in order to control the flow of dialogues. Such a scheme enables a more reactive and responsive dialogue between the learner and the system and yields significant self-assessment improvement in learners. We also highlighted the explicit reflection phase of NDL for the promotion of metacognitive skills using a reflective dialogue at the end of every session which can also be used as a self-reflection log by the learners.

This thesis provides the details of the architecture of our system, the design and implementation, and presents the discussion on the results of two evaluation studies. Our system consists of 5 main components that interact with each other to provide the learners with an open-ended, natural language dialogue interface. We have discussed in details the Wizard-of-Oz experiment that was conducted to collect the data to support our system design. The data that we collected during the experiments was analyzed to select 3 affective and 3 behavioral states used to control the dialogue in NDL. The results from the last phase of the WoZ experiment showed that the data we collected and the resulting rules and response libraries allowed the wizard to conduct negotiations with the learners in the domain of Data Structures almost automatically. We then discussed the implementation details of our system the NDLtutor and presented two evaluation studies of our system which has been developed using the architecture obtained by the WoZ experiment. We evaluated the interactions between the students and the NDLtutor to highlight the potential benefits of using our approach. We have argued that our approach provides new insights into combining best practices that have only been used separately in existing OLMs to develop an intelligent tutoring system that is capable of engaging learners in dialogues that promote metacognitive skills in them.

NDL follows the notion that learning is maximized by participation in the learning process and negotiation provides an excellent opportunity to challenge the learners which promotes metacognitive skills by motivating them to think more objectively about their learning. NDL finds its roots in the theory of repetition in learning. We believe that continuously engaging learners in dialogue that encourage them to utilize their metacognitive abilities allows them to use such abilities more efficiently over time and our proposed approach has the potential to achieve these desired results.

7.2 Contributions

We highlight the contributions of this research in the light of the research objectives we mentioned at the beginning of this thesis:

(1) *Proposition of a new learning paradigm of Negotiation-Driven Learning (NDL)* – The research successfully proposed and elaborated the feasibility and applicability of a new learning paradigm of NDL for enhancing the role of negotiation in the context of OLMs by using approximations of the learner's affective and behavioral states to control the flow of dialogue in a natural language setting.

(2) *Proposition of a conversational agent named NDLtutor that uses Interest-Based Negotiation in OLMs to engage the learners in a natural language dialogue targeted towards deeper learning* – The research resulted in the design and implementation of the NDLtutor which is the concretization of the NDL paradigm. The NDLtutor is an independent OLM, that can work as a standalone system or can be integrated with an existing ITS to provide a natural language interface to the learners for interacting with the system.

(3) *Demonstration of the impact and effects of NDLtutor's dialogue management for promoting metacognitive abilities in learners* – To investigate and demonstrate the impact and effects of the NDLtutor, we carried out 2 evaluations. The first

evaluation focused on the dialogue management capabilities of the NDLtutor. The results from this evaluation were very promising as the system was able to conduct successful dialogue sessions with the learners in a completely autonomous manner. This evaluation highlighted the NDLtutor's ability to handle various types of learner inputs and engage learners in constructive dialogues.

The second evaluation focused on the improvement of self-assessment accuracy of the learners and the promotion of self-reflection. The results from this evaluation showed that the NDLtutor was indeed able to improve the self-assessment accuracy of the learners and the number of topics where the learner's and the system's belief had a conflict did reduce. We were also able to identify the different statistically significant relationships between the learner's confidence in their knowledge, their actual knowledge and their self-reflection skills.

7.2.1 Contribution to Knowledge Science

Knowledge co-creation is the process where knowledge is created by the collaboration of individual knowledge, organizational knowledge and machine knowledge. Knowledge co-creation is comprised of *sharing*, utilizing and synthesizing knowledge from different source to create collaborative knowledge. A major research issue in the knowledge co-creation process is the conflicting knowledge held by entities involved in the interaction. This conflict of knowledge might occur due to the incorrect knowledge held by an entity or due to missing information regarding the change in the knowledge space of an entity. In the scope of Open Learner Models, such a situation can occur where a learner's beliefs about their knowledge change and are different from that of the system. This new information can cause a conflict between the system's belief about the learner's knowledge and the learner's belief base. This research contributes to the field of knowledge science by addressing the *knowledge sharing* problem between a learner and the system by providing a negotiation mechanism which allows them to share knowledge amongst each other to resolve conflicts.

The negotiation process requires all the concerned parties to be motivated to engage in an interaction for mutual gain. Open Learner Models focus on maximizing the motivation of the learners to increase their engagement in the learning process by involving them in the knowledge sharing process. Following these guidelines, this thesis proposes a new learning paradigm of Negotiation-Driven Learning which uses an artificially intelligent conversational agent to enhance the role of dialogues in the context of OMLs through context-aware reasoning using Interest-Based Negotiations. The goal is to provide a platform to the learner which maximizes their motivation to collaborate with the system for knowledge sharing and helps them regulate their learning. To achieve these goals, our system tries to classify and understand learner's typed input by using Natural Language Techniques and provides real-time responses that are adapted to each individual input. The system learns about the learner's affective and behavioral characteristics by generating a State Model for each individual learner. This allows the system to be more reactive in its responses as well as control the flow of dialogue by ensuring maximum learner participation. The data and results produced by the evaluation of our system will be used to generate future models of the learner's interactions and can provide insights into the different relationships between the learner's mental states and their performance.

7.3 Limitations & Future Work

This thesis has several limitations that can provide directions for future work.

Providing a Natural Language interface to learners can ease the communication process but adds to the overall complexity. NLP is a research field in its own right and a complete understanding of learner inputs is out of the scope of this study. The current system matches the learner's inputs with the predefined answers in the database. If an answer is matched the system uses this information to respond to the learner. However the system does not know what exactly was lacking in the learner's answer. A better understanding of this would allow the system to provide more intelligent follow-up moves.

For the current implementation of the NDLtutor, the system only covers the basic definitions of domain topics. This makes the expected answers to the system questions quite short and relatively easy to score. However, as the complexity of the expected answers increase, the current NLU techniques of the system might not provide a suitable solution and more power NLU techniques would need to be considered.

In order to minimize such complexity we use the Normalized Compression Distance (NCD) and Cosine Similarity Index algorithms to find the matching utterances in the classification process. The main reason for using less complex NLP techniques that the NDLtutor does not adopt the role of a teacher, but only reinforces and discusses what the learner knows to improve their understanding. However, more advanced NLP techniques will definitely allow to system to provide better scaffolding to the learner and understand their responses better.

Identifying learner's mental states is not a trivial task and it cannot be claimed that the NDLtutor is able to identify all such mental states. This study uses 6 states to control the dialogue flow, whereas there might be more states in certain situations that affect the outcome of an interaction. Our system tries to identify such states by matching a learner's input with the predefined utterance library where each utterance has related affective and behavioral attributes. This makes our task very simple; however it does not garauntee a perfect match to the learner's actual mental state. A simplifying assumption for this study was to use an approximation of the learner's states to keep the results as realistic as possible. Future work needs to investigate this further and a more refined list of states can be generated for controlling dialogue flow along with a mechanism to handle situations where the system identifies the states incorrectly.

The issue of engaging students with different knowledge levels is also one of the limitations that need to be considered in a real setting. Even though the NDLtutor tries to engage more knowledgeable students by asking them bonus questions, such students do not enter into a reflective dialogue with the system, therefore they might not have a reflection log for any of the sessions. How can the system engage such student is an interesting future direction?

The NDLtutor employs a very basic scoring mechanism to generate a reflection score for a student. During the reflective dialogue, the student's answers might vary according to their understanding of the domain as well as their own learning. The system needs to be capable of handling a varied form of inputs from the students. Again, this is an NLP issue that can elevate the power of the system to provide adequate feedback and responses.

We presented an evaluation framework based on the previous work on using chatbots in OLMs (Kerly, A., Ellis, R., Bull, S., 2008) and evaluated our system accordingly. Our findings were consistent with the previous research and showed a significant improvement in learner's self-assessment abilities after negotiating with the NDLtutor. We also evaluated the explicit reflection phase we introduced in our system in order to support and promote self-reflection in learners. The informal formative assessment of the reflection phase provided evidence that a reflection dialogue can engage learners to analyze and assess their understanding of their knowledge. The results of the evaluation also highlighted characteristics in the interaction patterns that were common in different categories of learners. Our findings demonstrate that the NDLtutor does provide adequate support to the learners to promote reflective thinking. Further in-depth analysis into the forms of reflection supported by our system would be a very interesting future direction.

One of the basic design considerations for our system was the separation of the domain-dependent and domain-independent content of the system. Using a backend database to store the system utterances in a different table from the domain content and using a sentence generator to generate system responses at run-time, allows for adding new domain knowledge without major changes to the system. Similarly, adding a new domain to the system should also benefit from this design decision. A full-fledged evaluation of domain portability is needed along with possibility of providing an *Authoring Tool* for authoring new domains constitutes the future work.

Another interesting future direction would be to investigate the existence of domain transferability of the NDLtutor. Whether the student's interaction patterns would change with a new domain, or would they achieve higher reflection scores after continuously engaging in the reflection phase dialogues with the system requires further investigations.

References

Afzal, S. and Robinson, P. 2011. Natural affect data: Collection and annotation. In New Perspectives on Affect and Learning Technologies, R. Calvo and S. D'Mello, Eds. Springer, 44–70.

ALICE. 2002. A.L.I.C.E AI Foundation, http://www.alicebot.org/

Austin, J. L. 1962. How to do things with words. London: Oxford University Press, p. 1.

Ballmer, Thomas T., and Waltraud Brennenstuhl. "Author's Motivation for a Speech Act Classification." *Speech Act Classification*. Springer Berlin Heidelberg, 1981. 13-14.

Beck, J., Stern, M., & Woolf, B. P. (1997). Cooperative Student Models. In B. Du Boulay & R. Mizoguchi (Eds.), *Artificial Intelligence in Education* (pp.127–134). Amsterdam: IOS Press

Bull, S. & Pain, H. (1995): "Did I Say What I Think I Said, And Do You Agree With Me?": Inspecting and Questioning the Student Model, in J. Greer (ed.), AIED95, AACE, Charlottesville VA, 501-508.

Bull, S. & Vatrapu, R. (2012): "Negotiated Learner Models for Today". ICCE.

Bull, S. & Judy K. (2010): "Open learner models". Advances in intelligent tutoring systems. Springer Berlin Heidelberg, 2010. 301-322

Bull, S., & Kay, J. (2013): "Open learner models as drivers for metacognitive processes". In International Handbook of Metacognition and Learning Technologies (pp. 349-365), Springer New York.

Bull, S. & Kay, J. (2007): Student Models that Invite the Learner In: The SMILI☺ Open Learner Modelling Framework, International Journal of Artificial Intelligence in Education

17(2) (2007), 89-120.

Bull, S., & McEvoy, A. T. (2003). An Intelligent Learning Environment with an Open Learner Model for the Desktop PC and Pocket PC. In U. Hoppe, F. Verdejo, & J. Kay (Eds.), *Artificial Intelligence in Education* (pp. 389–391). Amsterdam: IOS Press.

Bull, S., Brna, P., Pain, H. (1995): Mr. Collins: a collaboratively constructed, inspectable student model for intelligent computer assisted language learning. Instructional Science, 23 (1995), pp. 65–87

Burleson, W. & Picard, R. 2007. Evidence for gender specific approaches to the development of emotionally intelligent learning companions. IEEE Intell. Syst. 22, 4, 62–69.

C. Rose, D. Bhembe, S. Siler, R. Srivastava and K. VanLehn, "The Role of Why Questions in Effective Human Tutoring," Proceedings of Artificial Intelligence in Education, 2003.

Cilibrasi, R. and Vitanyi, P. (2005). Clustering by compression. IEEE Transactions on Information Theory, 51:1523–1545

Creative Virtual. (2007). "Creative Virtual UK web site." Retrieved 1 April 2007, from www.creativevirtual.com

Chetty, J. & van der Westhuizen, D. (2014). Implementing Metacognition Skills for Learners Studying Computer Programming. In J. Viteli & M. Leikomaa (Eds.), Proceedings of EdMedia: World Conference on Educational Media and Technology 2014 (pp. 726-731). Association for the Advancement of Computing in Education (AACE).

Colby, Kenneth. 1973. Simulation of belief systems. In R. Schank and K. Colby (eds.). Computer models of thought and language, 251-286. San Francisco: Freeman.

Conati, C. & Maclaren, H. 2009. Empirically building and evaluating a probabilistic model of user affect. User Model. User-Adapt. Interact. 19, 3, 267–303.

Czarkowski, M., Kay, J., & Potts, S. (2005). Web Framework for Scrutable Adaptation. In *Workshop on Learner Modelling for Reflection, 12th. International Conference on Artificial Intelligence in Education* (pp. 11–18).

D. Gal, D.D. Rucker (2010): “When in doubt, shout! Paradoxical influences of Doubt on Proselytizing”. In: Psychological Science, 21 (11), pp. 17011707.

D’Mello, S.K., Picard, R.W., Graesser, A.C. (2007): “Towards an affect sensitive Autotutor.” Special Issue on Intelligent Education Systems – IEEE Intelligent Systems 22(4), 53–61.

Dahlbäck, N., Jönsson, A. and Ahrenberg, L. (1993): “Wizard of Oz Studies – Why and How”. International Workshop on Intelligent User Interfaces '93. ACM: 193-200.

De Vicente, A., & Pain, H. (2002): “Informing the detection of the students’ motivational state: an empirical study.” In Intelligent tutoring systems (pp. 933-943). Springer Berlin Heidelberg. (2002)

Dimitrova, V.: “STyLE-OLM (2003): Interactive open learner modelling”. In: International Journal of Artificial Intelligence in Education, 13, 35-78.

Du Boulay, Benedict, et al. (2010): “Towards systems that care: a conceptual framework based on motivation, metacognition and affect.” International Journal of Artificial Intelligence in Education 20.3: 197-229.

Ekman, P. (1973): “Universal facial expressions in emotion.” Studia Psychologica 15(2), 140–147.

Fisher, R., Ury, W. (1983): “Getting to Yes: Negotiating Agreement without giving in”. Penguin books, New York.

Flavell, J. H. (1987). Speculations about the Nature and Development of Metacognition. In *Metacognition, Motivation, and Understanding*. F. Weinert and R. Kluwe, Eds. Hillsdale, NJ., Lawrence Erlbaum Associates Inc.**:** 21-29

Fredrickson, BL. What good are positive emotions? Review of General Psychology. 1998; 2:300–319.

Freedman, R. ATLAS (1999): A Plan Manager for Mixed-Initiative, Multimodal Dialogue. Paper presented at the 1999 AAAI Workshop on Mixed-Initiative Intelligence, 19 July, Orlando, Florida.

Garner, R. & Alexander, P.A. (1989). Metacognition: Answered and unanswered questions, Educational Psychologist, 24, 143-158

Gertner, A. S., and VanLehn, K. (2000). ANDES: A Coached Problem-Solving Environment for Physics. In Intelligent Tutoring Systems: Fifth International Conference, ITS 2000, eds. 133-142, New York, Springer.

Graesser, A. C.; Wiemer-Hastings, K.; Wiemer-Hastings, P.; Kreuz, R.; and the Tutoring Research Group. AUTOTUTOR (1999): A Simulation of a Human Tutor. Journal of Cognitive Systems Research 1(1): 35–51.

Graesser, A.C., McDaniel, B., Chipman, P., Witherspoon, A., D'Mello, S., Gholson, B. (2006): "Detection Of Emotions During Learning With Autotutor." In: Proceedings of the 28th Annual Conference of the Cognitive Science Society, pp. 285–290. Erlbaum, Mahwah.

Graesser, A. C., et al. "Using latent semantic analysis to evaluate the contributions of students in AutoTutor." *Interactive Learning Environments*8.2 (2000): 129-147.

Gross, S., B. Mokbel, B. Hammer, N. Pinkwart (2015). Learning Feedback in Intelligent Tutoring Systems. *KI - Künstliche Intelligenz*, 1-6.

Keller, John M. (1983): "Motivational design of instruction." Instructional design theories and models: An overview of their current status 1 (1983): 383-434. Associates.

Kerly, A., Ellis, R., Bull, S. (2008). “CALMsystem: A Conversational Agent for Learner Modelling”. In: Knowledge-Based Systems 21(3), pp. 238-246.

Kerly, A., Ahmad, N., Bull, S. (2008). “Investigating Learner Trust in Open Learner Models Using a ’Wizard of Oz’ Approach”. Intelligent Tutoring Systems. Springer

Kerly, A. & Bull, S. (2006). The Potential for Chatbots in Negotiated Learner Modelling, in M. Ikeda, K. Ashley & T-W. Chan (eds), Intelligent Tutoring Systems: 8th International Conference, Springer-Verlag, Berlin Heidelberg, 443-452

Kerly, A., Hall, P., Bull, S. (2007). Bringing chatbots into education: Towards natural language negotiation of open learner models. Knowledge Based Systems. 20, 2 (March 2007), 177-185.

Kerly, A., Bull, S. (2008): Children’s Interactions with Inspectable and Negotiated Learner Models. In: Woolf, B.P., Aïmeur, E., Nkambou, R., Lajoie, S. (eds.) ITS 2008. LNCS, vol. 5091, pp. 132–141. Springer, Heidelberg

Lehman, Blair, et al. (2008): “What are you feeling? Investigating student affective states during expert human tutoring sessions.” Intelligent Tutoring Systems. Springer.

Lepper, Mark R., et al. (1993): “Motivational techniques of expert human tutors: Lessons for the design of computer-based tutors.” Computers as cognitive tools. 75-105.

Liddy, Elizabeth D., et al. "Natural language processing." *Encyclopedia of library and information science* 2 (2003).

Long, Y., & Aleven, V. (2013). Skill diaries: Improve student learning in an intelligent tutoring system with periodic self-assessment. In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Proceedings of Artificial Intelligence in Education (AIED) 2013* (pp. 219–228). Berlin: Springer.

M. G. Core, J. D. Moore and C. Zinn, "The role of initiative in tutorial dialogue," Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics, pp. 67-74, 2003

M. T. H. Chi, S. A. Siler, H. Jeong, T. Yamauchi and R. G. Hausmann, "Learning from human tutoring," *Cognitive Science*, vol. 25, pp. 471-533, 2001

Mabbott, A., & Bull, S. (2006). Student Preferences for Editing, Persuading, and Negotiating the Open Learner Model. In M. Ikeda, K. Ashley, & T. W. Chan (Eds.). *Intelligent Tutoring Systems: 8th International Conference* (pp. 481–490). Berlin Heidelberg: Springer-Verlag.

Malone, T. W., and Lepper, Mark R. (1987): “Making learning fun: A taxonomy of intrinsic motivations for learning.” In R. E. Snow and M. J. Farr, editors, Conative and Affective Process Analyses, volume 3 of Aptitude, Learning, and Instruction, chapter 10, pages 223–253. Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey.

Miao, Yuan. (2008): “An intelligent tutoring system using interest based negotiation.” Control, Automation, Robotics and Vision, ICARCV.

Mitrovic, A. & Martin, B., (2002): Evaluating the Effects of Open Student Models on Learning, in P. de Bra, P. Brusilovsky & R. Conejo (eds), Proceedings of 2nd International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Springer-Verlag, Berlin Heidelberg, (2002), 296-305.

Mitrovic, A. & Martin, B. Evaluating the Effect of Open Student Models on Self-Assessment. International Journal of Artificial Intelligence in Education 17(2) (2007), 121-144

Paiva, A., Self, J., & Hartley, R. (1994). Externalizing Learner Models. In *Proceeding of World Conference on Artificial Intelligence in Education* (pp.509–516). Washington DC.

Person, N. K.; Graesser, A. C.; Kreuz, R. J.; Pomeroy, V.; and the Tutoring Research Group (2001). Simulating Human Tutor Dialogue Moves in AUTOTUTOR. International Journal of Artificial Intelligence in Education.

S. Katz, D. Allbritton and J. Connelly, "Going Beyond the Problem Given: How Human Tutors Use Post-Solution Discussions to Support Transfer," International Journal of Artificial Intelligence in Education, vol. 13, pp. 79-116, 2003.

Searle, John R. (1975), "A Taxonomy of Illocutionary Acts", in: Günderson, K. (ed.), *Language, Mind, and Knowledge*, (*Minneapolis Studies in the Philosophy of Science*, vol. 7), University of Minneapolis Press, p. 344-69.

Self, J. A. (1999a). The defining characteristics of intelligent tutoring systems research: ITSs care, precisely. *International Journal of Artificial Intelligence in Education*, 10, 350-364.

Schraw, G. & Dennison, R.S. (1994). Assessing Metacognitive Awareness, Contemporary Educational Psychology 19, 460-475

Shawar, B, A., Eric, A. "A comparison between ALICE and Elizabeth chatbot systems." (2002)

Sidney D'mello and Art Graesser. 2013. AutoTutor and affective AutoTutor: Learning by talking with cognitively and emotionally intelligent computers that talk back. *ACM Trans. Interact. Intell. Syst.* 2, 4, Article 23 (January 2013)

Suleman, Raja M., Mizoguchi, Riichiro, Ikeda, Mitsuru (2015): "Negotiation-Driven Learning". Proceedings of 17th International Conference on Artificial Intelligence in Education (AIED).

Swanson, H. L. (1990) Influence of metacognitive knowledge and aptitude on problem solving, *Journal of Educational Psychology,* 82(2), 306-314.

Toulmin, S. E. (1958). *The Uses of Argument*. Cambridge, UK: Cambridge University Press.

Van Labeke, N., Brna, P., & Morales, R. (2007). Opening up the Interpretation Process in an Open Learner Model. *International Journal of Artificial Intelligence in Education*, *17*(3), 305–338.

VanLehn, K. (1996): Conceptual and Meta-learning during Coached Problem Solving. In Proceedings of the Third Intelligent Tutoring Systems Conference, eds. C. Frasson, G. Gauthier, and A. Lesgold, 29–47. Berlin: Springer-Verlag.

VanLehn, K. (1988). Student Modelling. In M. Poison & J. Richardson (eds) Hilisdale, ’NJ: Erlbaum. 55-77. 55. In M. Poison & J. Richardson (Eds.), Foundations of Intelligent Tutoring Systems (pp. 55–77). Hilisdale, NJ: Erlbaum

Wallace, Richard. 2003. The elements of AIML style. ALICE AI Foundation.

Weber, G., & Brusilovsky, P. (2001). ELM-ART: An Adaptive Versatile System for Web-based Instruction. *International Journal of Artificial Intelligence in Education*, *12*, 351–384.

Weizenbaum, Joseph. 1966. ELIZA-A computer program for the study of natural language communication between man and machine. Communications of the ACM 10.8: 36-45.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*. California: Morgan Kaufmann Publisher.

Woolf, B., Arroyo, I., Muldner, K., Burleson, W., Cooper, D., Dolan, R., & Christopherson, R. 2010. The effect of motivational learning companions on low achieving students and students with disabilities In Proceedings of the 10th International Conference on Intelligent

Tutoring Systems, J. Kay and V. Aleven, Eds. Springer, 327–337.

Xuehong Tao, et al. (2006): “Interest Based Negotiation Automation.” Proceedings of International Conference on Intelligent Computing, Kunming, China, August, 2006.Volume 4115.

Zapata-Rivera, D., Hansen, E., Shute, V. J., Underwood, J. S., & Bauer, M. (2007). Evidence-based approach to interacting with open student models. *International Journal of Artificial Intelligence in Education, 17*(3), 273–303.

Appendix A: List of Publications

Journal Paper

1. Raja M. Suleman, Riichiro Mizoguchi, Mitsuru Ikeda. "A New Perspective of Negotiation-Based Dialog to Enhance Metacognitive Skills in the Context of Open Learner Models". *International Journal of Artificial Intelligence in Education (IJAIED),* Date of Submission: 11th September, 2015, Status: "Accepted", Date of Acceptance: 1st July, 2016, No. of pages: 50.

Conference Papers

1. Raja M. Suleman, Riichiro Mizoguchi, Mitsuru Ikeda. "NDLtutor: An Automated Conversational Agent to Facilitate Metacognitive Skills in Fully-Negotiated OLMs (Short Paper)". *13th Internationa1 Conference on Intelligent Tutoring Systems*, Vol. 9684, Springer 2016, pp 354-360, 7th-10th June 2016, Zagreb, Croatia.
2. Raja M. Suleman, Riichiro Mizoguchi, Mitsuru Ikeda. "Negotiation-Driven Learning". *Artificial Intelligence in Education: 17th International Conference, AIED 2015, Madrid, Spain, June 22-26, 2015. Proceedings*. Vol. 9112. Springer, 2015, pp 470-479, 22nd – 26th June, 2015, Madrid, Spain.

Conference Poster Papers

1. Raja M. Suleman, Riichiro Mizoguchi, Mitsuru Ikeda. "Negotiation-Driven Learning (Poster Paper)". 9th European Conference on Technology Enhanced Learning. In: Open Learning and Teaching in Educational Communities Lecture Notes in Computer Science Volume 8719, 2014, pp 590-591, 16th-19th September, 2014, Graz, Austria.
2. Raja M. Suleman, Riichiro Mizoguchi, Mitsuru Ikeda. "Negotiation-Driven Learning (Poster Paper)". 12th Internationa1 Conference on Intelligent Tutoring Systems. In: Intelligent Tutoring Systems Lecture Notes in Computer Science Volume 8474, pp 652-653, 5th-9th June 2014, Honolulu, Hawaii, USA.

Others (Doctoral Consortium Papers)

1. Raja M. Suleman, Riichiro Mizoguchi, Mitsuru Ikeda. "Negotiation-Driven Learning: A New Perspective of Learning Using Negotiation (Doctoral Consortium Paper)". *Artificial Intelligence in Education: 17th International*

Conference, AIED 2015, Madrid, Spain, June 22-26, 2015. Proceedings. Vol. 9112. Springer, 2015, pp 868-872, 22nd – 26th June, 2015, Madrid, Spain.

2. Raja M. Suleman, Riichiro Mizoguchi, Mitsuru Ikeda. "Negotiation-Driven Learning: A New Perspective of Learning Using Negotiation (Young Researcher Track Paper)". 12th Internationa1 Conference on Intelligent Tutoring Systems. In: Intelligent Tutoring Systems Lecture Notes in Computer Science Volume 8474, pp 689-691, 5th-9th June 2014, Honolulu, Hawaii, USA.

Appendix B: NDLtutor Dialogue Logs (Evaluation 1)

<u>Dialogue Log#1</u>

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is Medium. This is different from my belief about your proficiency in the topic of Stacks. What makes you think you have Medium level of proficiency?

Student: i think i know more than that

NDLtutor: Are you sure? You don't seem to be confident about your answer.

Student: yes i'm sure!

NDLtutor: That's great!
But your performance in the topic of Stacks was not very convincing.

Student: i know

NDLtutor: Do you want me to give you an overview of your performance?

Student: no i already know it

NDLtutor: To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: ok

NDLtutor: Great!
Can you give me a complete definition of Stack?

Student: A stack is a data structure. it holds elements like a basket. we can put in elements and take out elements

NDLtutor: Can you give me more details please?

Student: it is a data structure that holds elements. We can put elements in it and take them out. E.g. basket of clothes

NDLtutor: Ok!
A Stack is an Abstract Data Type (ADT) that holds elements in reverse order.
What is LIFO?

Student: last in first out

NDLtutor: Can you give me more details please?

Student: LIFO is last in first out

NDLtutor: Ok!
Stacks follow the LIFO principle for storing elements. The Last element inserted In the stack is the First element taken Out of the stack.
What are the basic operations you can perform on a stack?

Student: push n pop

NDLtutor: Can you explain more?

Student: push is when we put an element on the stack. Pop is when we take element out of the stack

NDLtutor: Good job!
Push() operation is used to insert an element in the stack. Pop() operation is used to take an element out of the stack.
We have covered the basic definition of a stack. Can you summarize your answer now?

Student: it is a data structure that holds elements. It follows the LIFO principle for storing elements. The Last element inserted In the stack is the First element taken Out of the stack. It has 2 basic operations push and pop. Push operation is used to insert an element in the stack. Pop operation is used to take an element out of the stack.

NDLtutor: Great!
Let's reflect upon our discussion now!
How is your final answer different from your initial answer?

Student: my final answer is complete definition

NDLtutor: Your initial answer was incomplete and lacked most of the concepts. Your final answer scored well as it is complete and covers all the aspects.
Did you find anything confusing?

Student: I was a little confused when you keep asking me to explain more. You did not tell me what you want to know

NDLtutor: Ok!
Do you have any confusion now?

Student: No

NDLtutor: Great!
What did you find helpful in our discussion?

Student: the way you asked questions

NDLtutor: After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Low.
Please review the topic to improve your level of proficiency in the topic of Stacks.
This session has concluded.

Dialogue Log#2

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is Medium.
This is different from my belief about your proficiency in the topic of Stacks.
What makes you think you have Medium level of proficiency?

Student: i know more

NDLtutor: That's great!
But your performance in the topic of Stacks was not very convincing.

Student: yeah i wasn't prepared

NDLtutor: Do you want me to give you an overview of your performance?

Student: i know i did bad, but i know more then that

NDLtutor: To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: ok

NDLtutor: Great!
Can you give me a complete definition of Stack?

Student: a stack is a data structure that holds elements on top of each other like a basket

NDLtutor: Can you give me more details please?

Student: a stack is a data structure. it holds elements like a basket

NDLtutor: Ok!
A Stack is an Abstract Data Type (ADT) that holds elements in reverse order.
What is LIFO?

Student: last in first out

NDLtutor: Can you give me more details please?

Student: Stacks use LIFO to hold elements. LIFO is last in first out

NDLtutor: Ok!
Stacks follow the LIFO principle for storing elements. The Last element inserted In the stack is the First element taken Out of the stack.
What are the basic operations you can perform on a stack?

Student: push and pop

NDLtutor: Can you explain more?

Student: push is used to insert an element on the stack and pop is used to take an element out of the stack

NDLtutor: Good job!
Push() operation is used to insert an element in the stack. Pop() operation is used to take an element out of the stack.
We have covered the basic definition of a stack. Can you summarize your answer now?

Student: a stack is an abstract data type that follows LIFO principle to hold elements. the last element inserted in the stack is the first element taken out of stack. it uses push function to insert elements in the stack and pop function to remove elements from the stack.

NDLtutor: Great!
Let's reflect upon our discussion now!
How is your final answer different from your initial answer?

Student: i missed lifo and operations in my initial answer

NDLtutor: Your initial answer was incomplete and lacked most of the concepts. Your final answer scored well as it is complete and covers all the aspects.
Did you find anything confusing?

Student: no

NDLtutor: Ok!
Do you have any confusion now?

Student: no

NDLtutor: Great!
What did you find helpful in our discussion?

Student: you explained the answer

NDLtutor: After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Low.
Please review the topic to improve your level of proficiency in the topic of Stacks.
This session has concluded.

<u>Dialogue Log#3</u>

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is High.
This is different from my belief about your proficiency in the topic of Stacks.
What makes you think you have High level of proficiency?

Student: i think its high

NDLtutor: Are you sure? You don't seem to be confident about your answer.

Student: yes

NDLtutor: That's great!
But your performance in the topic of Stacks was not very convincing.

Student: i know

NDLtutor: Do you want me to give you an overview of your performance?

Student: no

NDLtutor: To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: yes

NDLtutor: Great!
Can you give me a complete definition of Stack?

Student: a stack is a data structure that holds elements on top of each other like a basket

NDLtutor: Can you give me more details please?

Student: a stack is a LIFO data structure like a basket

NDLtutor: Ok!
A Stack is an Abstract Data Type (ADT) that holds elements in reverse order.
What is LIFO?

Student: last in first out

NDLtutor: Can you give me more details please?

Student: LIFO is how stack hold elements

NDLtutor: Ok!
Stacks follow the LIFO principle for storing elements. The Last element inserted In the stack is the First element taken Out of the stack.
What are the basic operations you can perform on a stack?

Student: push and pop

NDLtutor: Can you explain more?

Student: push inserts an element on the stack and pop removes an element out of the stack

NDLtutor: Good job!
Push() operation is used to insert an element in the stack. Pop() operation is used to take an element out of the stack.
We have covered the basic definition of a stack. Can you summarize your answer now?

Student: a stack is an abstract data type that follows LIFO principle to hold elements. the last element inserted in the stack is the first element taken out of stack.it uses push and pop function. push inserts an element on the stack and pop removes an element out of the stack

NDLtutor: Great!
Let's reflect upon our discussion now!
How is your final answer different from your initial answer?

Student: more details

NDLtutor: Your initial answer was incomplete and lacked most of the concepts. Your final answer scored well as it is complete and covers all the aspects.
Did you find anything confusing?

Student: no

NDLtutor: Ok!
Do you have any confusion now?

Student: no

NDLtutor: Great!
What did you find helpful in our discussion?

Student: how you break down the definition in small questions

NDLtutor: After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Medium.
Please review the topic to improve your level of proficiency in the topic of Stacks.
This session has concluded.

Dialogue Log#4

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is Very High.
This is different from my belief about your proficiency in the topic of Stacks.
What makes you think you have Very High level of proficiency?

Student: because i know stacks

NDLtutor: That's great!
But your performance in the topic of Stacks was Moderate.
Do you want me to give you an overview of your performance?

Student: was it?

NDLtutor: I'm sorry I didn't understand that. What do you mean?

Student: was my performance moderate?

NDLtutor: You attempted 10 questions and got 06 correct answers.
To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: ok

NDLtutor: Great!
Can you give me a complete definition of Stack?

Student: a stack is a data structure that uses LIFO principle to store elements. LIFO means that the last element inserted in the stack is the first element removed from the stack. there are 2 basic functions of stacks. we use push function to insert element in and pop function to remove element from it.

NDLtutor: Great work!
Your knowledge of basics of Stack is very good! After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Very High
Keep up the good work!
This discussion session is complete.

Dialogue Log#5

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is Medium. This is different from my belief about your proficiency in the topic of Stacks. What makes you think you have Medium level of proficiency?

Student: because i am better than that

NDLtutor: That's great!
But your performance in the topic of Stacks was not very convincing.

Student: i didnt prepare for the test

NDLtutor: Do you want me to give you an overview of your performance?

Student: no

NDLtutor: To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: hmmm... ok

NDLtutor: Great!
Can you give me a complete definition of Stack?

Student: An abstract data type that contains elements from top to bottom

NDLtutor: Can you give me more details please?

Student: Can you give me more details please?

NDLtutor: I would be happy to help you. But I need you to work with me.
Give it a try, I am sure you can do better!

Student: I don't know more about stack

NDLtutor: Ok here's a start... A Stack is an Abstract Data Type (ADT) that serves as a collection of elements. Stacks are ordered using the LIFO principle.
What is LIFO?

Student: i know this...its last in first out

NDLtutor: Great work! You have already started answering my questions.
Stacks follow the LIFO principle for storing elements. The Last element inserted In the stack is the First element taken Out of the stack.
Let's keep working!
What are the basic operations you can perform on a stack?

Student: give me a hint

NDLtutor: Why don't you try it yourself first? I will help you with the correct answer.

Student: i don't know

NDLtutor: Stacks have 2 basic operations _____ and ____

Student: Oh yes! push and pop

NDLtutor: Good job!
Push() operation is used to insert an element in the stack. Pop() operation is used to take an element out of the stack.
We have covered the basic definition of a stack. Can you summarize your answer now?

Student: Yes... A Stack is an Abstract Data Type (ADT) that serves as a collection of elements. Stacks are ordered using the LIFO principle. The Last element Inserted in the stack is the First element taken Out from the stack. PUSH() operation is used to push an element in the stack. POP() operation is used to take an element out of the stack.

NDLtutor: Great!
Let's reflect upon our discussion now!
How is your final answer different from your initial answer?

Student: i just copied your answers to make my final answer.

NDLtutor: Your initial answer was incomplete and lacked most of the concepts. Your final answer scored well as it is complete and covers all the aspects.
Did you find anything confusing?

Student: no

NDLtutor: Ok!
Do you have any confusion now?

Student: no

NDLtutor: Great!
What did you find helpful in our discussion?

Student: you gave all the answers

NDLtutor: After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Low.
Please review the topic to improve your level of proficiency in the topic of Stacks.
This session has concluded.

Dialogue Log#6

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is Very High.
This is different from my belief about your proficiency in the topic of Stacks.
What makes you think you have Very High level of proficiency?

Student: because i know stacks well

NDLtutor: That's great!
But your performance in the topic of Stacks was Moderate.
Do you want me to give you an overview of your performance?

Student: is this ai?

NDLtutor: I'm sorry I didn't understand that. What do you mean?

Student: are you a bot?

NDLtutor: I am the NDLtutor!
I'm here to help you learn better :)
Let's keep working!
Do you want me to give you an overview of your performance?

Student: ahan

NDLtutor: You attempted 10 questions and got 06 correct answers.
To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: i didnt ask for the overview you just gave me. i just said ahan

NDLtutor: I'm sorry I didn't understand that. What do you mean?

Student: lol

NDLtutor: I'm finding it hard to understand. Can you rephrase that please?

Student: laughing out loud :P

NDLtutor: Let's get back to learning :)
Can you give me a complete definition of Stack?

Student: why dont you give me a complete definitionof Stack?

NDLtutor: Give it a try, I am sure you can do better!

Student: i know what stacks are... do you?

NDLtutor: I'm sorry I didn't understand that. What do you mean?

Student: haha

NDLtutor: Let's get back to learning :)
Can you give me a complete definition of Stack?

Student: this is fun... anyways a stack is an abstract data type which uses LIFO storing priciple. .basic operations on stack are push an element which inserts it on stack and pop an element which removes it from the top of the stack and isempty to check if the stack has any item in it.

NDLtutor: Great work!
Your knowledge of basics of Stack is very good! After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Very High
Keep up the good work!
This discussion session is complete.

Dialogue Log#7

Student: DISCUSS <Stack>

NDLtutor: Can you give me a complete definition of Stack?

Student: Stack is an abstract data type which uses LIFO (last in first out) Storing priciple. it is the collection of same data type. basic operations on stack are push an element on stack (insert) and pop and element on stack (delete) and isempty to check if the stack has any item in it. push and pop operation of stack is done through its one end called top of stack. there are many application of stack like reversing of strings or name and evaluation of expression

NDLtutor: Great work!
Your knowledge of basics of Stack is very good! My belief about your profeciency in the topic of Stacks is Very High.
My belief about your proficiency in the topic of Stacks is Very High.
Keep up the good work!
This discussion session is complete.

Appendix C: NDLtutor Dialogue Logs (Evaluation 2)

Dialogue Log#1

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is Very High. This is different from my belief about your proficiency in the topic of Stacks. What makes you think you have Very High level of proficiency?

Student: I know stacks

NDLtutor: That's great!
But your performance in the topic of Stacks was High.
Do you want me to give you an overview of your performance?

Student: no just ask me the question

NDLtutor: To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: yes

NDLtutor: Great!
Can you give me a complete definition of Stack?

Student: a stack is an abstract data type (ADT) which uses LIFO storing principle to store elements. it has 2 basic operations push() and pop(). push inserts an element on stack and pop removes it from the top of the stack

NDLtutor: Great work!
Your knowledge of basics of Stack is Very High!
Here’s a bonus question for you. Let’s see if you can get this one:
Stacks are called LIFO structures, while Queues are called ______ structures?

Student: FIFO

NDLtutor: Great!
Your knowledge of basics of Stacks is very good! After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Very High
Keep up the good work!
This session has concluded.

Dialogue Log#2

NDLtutor: You have indicated that your proficiency level in the topic of Stacks is Very High.
This is different from my belief about your proficiency in the topic of Stacks.
What makes you think you have Very High level of proficiency?

Student: I know more then that

NDLtutor: That's great!
But your performance in the topic of Stacks was High.
Do you want me to give you an overview of your performance?

Student: no

NDLtutor: I'm sorry!
To update my belief I need to be convinced about your knowledge level in the topic of Stacks.
Do you want to discuss the topic of Stacks with me?

Student: Ok

NDLtutor: Great!
Can you give me a complete definition of Stack?

Student: it is a data structure. it uses LIFO to store elements which means last-in first-out. There are 2 main operations push and pop. push inserts an element on stack and pop removes it from the top of the stack.

NDLtutor: Great work!
Your knowledge of basics of Stack is Very High!
Here's a bonus question for you. Let's see if you can get this one:
Which of the following is an example of a Stack?

A) A pile of books
B) A line at a fee counter

Student: A

NDLtutor: Good job!
Your knowledge of basics of Stacks is very good! After our discussion I have changed my belief in your proficiency level in the topic of Stacks to Very High
Keep up the good work!
This session has concluded.

Dialogue Log#3

NDLtutor: You have indicated that your proficiency level in the topic of Queues is High.
This is different from my belief about your proficiency in the topic of Queues.
What makes you think you have Very High level of proficiency?

Student: i want to improve my proficiency level

NDLtutor: That's great!
But your performance in the topic of Queues was Low.
Do you want me to give you an overview of your performance?

Student: no… just ask me questions

NDLtutor: I'm sorry!
To update my belief I need to be convinced about your knowledge level in the topic of Queues.
Do you want to discuss the topic of Queues with me?

Student: yes

NDLtutor: Great!
Can you give me a complete definition of Queue?

Student: queue is a data structure which uses FILO. it has 2 main operations enquee and dequeue.

NDLtutor: So you think Queue uses FILO storing principle? I'm afraid that's not correct.
In a Queue does INSERTION & DELETION take place at the same end?

Student: yes

NDLtutor: I'm afraid that's not correct.
In a Queue INSERTION takes place at the REAR while DELETION takes place at the FRONT.
What principle do cars at a drive-through follow?

Student: ummm… i dont understand

NDLtutor: I'm sorry there is no help/hint for that. Please try to answer it yourself

Student: i don't knw

NDLtutor: First car to come In the Queue is the First car to get Out of the Queue.
So now can you tell me what storing principle do Queues use?

Student: first in first out

NDLtutor: That's good! Can you say more?

Student: the first element in the queue is the first element out of queue

NDLtutor: Good work!
Queues use the FIFO principle to store elements. The First element inserted IN the Queue is the First element removed OUT of a Queue.
What are the basic operations you can perform on a Queue?

Student enqueue and dequeue

NDLtutor: Can you explain more please?

Student: queue has 2 operations enqueue and dequeue

NDLtutor: Enqueue is used to ________ elements in a Queue

Student: Insert

NDLtutor: Good job!
Dequeue is used to ________ elements from a Queue

Student: remove

NDLtutor: Great!
Enqueue() operation is used to insert an element in the Queue. Dequeue() operation is used to take an element out of the Queue.
We have covered the basic definition of a Queue. Can you summarize your answer now?

Student: queue is a data structure that uses FIFO which is first element inserted in the queue is the first element taken out of a queue. It has 2 operations enqueue which inserts elements and dequeue which removes elements from queue.

NDLtutor: Good job!
Let's reflect on our discussion now!
On a scale of 0 to 10 (0: minimum, 10: maximum) how would you rate your first answer?

Student: 1

NDLtutor: That's right. Unfortunately your initial answer didn't get a good score.
How is your last summarized answer different from your initial answer?

Student: my first answer was wrong. the last answer is correct

NDLtutor: Your initial answer was incomplete. It was missing concepts and it contained a misconception

What were the concepts that you missed?

Student: FIFO and the functions

NDLtutor: You missed: FIFO, Enqueue, Dequeue
Did you encounter any misconceptions?

Student: yes… I thought queues use FILO but the right answer was FIFO

NDLtutor: That's right. You thought Queue uses FILO storing principle.
Queues use the FIFO principle to store elements. The First element inserted IN the Queue is the First element removed OUT of a Queue.
Is there anything still unclear?

Student: no

NDLtutor: Great!
What aspects of this topic do you think you need to concentrate more on?

Student: the proper definition and the storing principle

NDLtutor: Do you think you will be able to discuss this topic with a peer/friend now?

Student: yes

NDLtutor: Great!
Ok the last question…. Did this discussion improve your understanding/knowledge?

Student: yes it did. it was really helpful

NDLtutor: That's great!
I'm afraid you have not been able to prove your level of proficiency in the topic of Queues to be High; therefore I cannot update my belief base at this time.
Please review the topic to improve your level of proficiency in the topic of Queues.
Keep working hard!
This session has concluded.

Dialogue Log#4

NDLtutor: You have indicated that your proficiency level in the topic of Queues is Very High.
This is different from my belief about your proficiency in the topic of Queues.
What makes you think you have Very High level of proficiency?

Student: i think its very high

NDLtutor: Are you sure? You don't seem to be confident about your answer.

Student: yes

NDLtutor: That's great!
But your performance in the topic of Queues was High.
Do you want me to give you an overview of your performance?

Student: Ok

NDLtutor: You attempted 10 questions and got 08 correct answers.
To update my belief I need to be convinced about your knowledge level in the topic of Queues.
Do you want to discuss the topic of Queues with me?

Student: yes

NDLtutor: Great!
Can you give me a complete definition of Queue?

Student: a queue is a data structure which uses first-in first-out principle. queue has 2 main operations enqueue and dequeue.

NDLtutor: Can you explain more please?

Student: queue is a data structure which uses FIFO first-in first-out principle. queue has 2 main operations enqueue and dequeue. enqueue is used to add elements in the queue while dequeue is used to delete elements from the queue.

NDLtutor: Good Job!
Your knowledge of basics of Queues is very good! After our discussion I have changed my belief in your proficiency level in the topic of Queues to Very High
Keep up the good work!
This session has concluded.

Dialogue Log#5

NDLtutor: You have indicated that your proficiency level in the topic of Queues is Very High.
This is different from my belief about your proficiency in the topic of Queues.
What makes you think you have Very High level of proficiency?

Student: i know queues

NDLtutor: That's great!
But your performance in the topic of Queues was High.
Do you want me to give you an overview of your performance?

Student: no

NDLtutor: I'm sorry!
To update my belief I need to be convinced about your knowledge level in the topic of Queues.
Do you want to discuss the topic of Queues with me?

Student: ok

NDLtutor: Great!
Can you give me a complete definition of Queue?

Student: a queue is a data structure in which the first element inserted is the first element taken out. it has 2 operations enqueue and dequeue.

NDLtutor: Can you explain more please?

Student: isnt that the answer?

NDLtutor: I'm sorry I didn't understand that. What do you mean?

Student: this is the correct answer… a queue is a data structure in which the first element inserted is the first element taken out. it has 2 operations enqueue and dequeue.

NDLtutor: Ok!
What storing principle do Queues use?

Student: FIFO

NDLtutor: Can you explain more please?

Student: queues use FIFO which is first in first out!

NDLtutor: Good job!
Queues use the FIFO principle to store elements. The First element inserted IN

the Queue is the First element removed OUT of a Queue.
We have covered the basic definition of a Queue. Can you summarize your answer now?

Student: a queue is a data structure which uses FIFO principle which is first in first out in which the first element inserted is the first element taken out. it has 2 operations enqueue and dequeue.

NDLtutor: Good job!
Let's reflect on our discussion now!
On a scale of 0 to 10 (0: minimum, 10: maximum) how would you rate your first answer?

Student: 9

NDLtutor: Ok! Your initial answer was not a complete definition.
How is your last summarized answer different from your initial answer?

Student: its not different

NDLtutor: Your initial answer was incomplete.
What were the concepts that you missed?

Student: fifo

NDLtutor: That's right! Good job!
Did you encounter any misconceptions?

Student: No

NDLtutor: That's right. You have no misconceptions about the basics of Queues!
Is there anything still unclear?

Student: no

NDLtutor: Great!
What aspects of this topic do you think you need to concentrate more on?

Student: i know the topic

NDLtutor: Do you think you will be able to discuss this topic with a peer/friend now?

Student: ofcourse

NDLtutor: Great!
Ok the last question…. Did this discussion improve your understanding/knowledge?

Student: i already knew it

NDLtutor: Ok!
Your knowledge of basics of Queues is very good! After our discussion I have changed my belief in your proficiency level in the topic of Queues to Very High
Keep up the good work!
This session has concluded.

Dialogue Log#6

NDLtutor: You have indicated that your proficiency level in the topic of Linked Lists is Very High.
This is different from my belief about your proficiency in the topic of Linked Lists.
What makes you think you have Very High level of proficiency?

Student: i know. im just trying to improve my scores

NDLtutor: Good!
But your performance in the topic of Linked Lists was Low.
Do you want me to give you an overview of your performance?

Student: no

NDLtutor: I'm sorry!
To update my belief I need to be convinced about your knowledge level in the topic of Linked Lists.
Do you want to discuss the topic of Linked Lists with me?

Student: yes

NDLtutor: Great!
Can you give me a complete definition of Linked List?

Student: it is a data structure like a collection of nodes. each node is connected to each other. simple linked list called singly linked list.

NDLtutor: Can you explain more please?

Student: no

NDLtutor: Give it a try!

Student: i don't know more

NDLtutor: Ok!
A Linked List is a collection of nodes linked together by references from a node to

another node.
Each Node in the Linked List has a reference to what?

Student: Other Node

NDLtutor: Ok!
Each node contains one or more data fields AND a reference to the next Node.
The last node contains a _____ reference to indicate the end of the list?

Student: empty

NDLtutor: I'm afraid that's not correct. Try again!

Student: empty or null

NDLtutor: Good job!
The last node contains a NULL reference to indicate the end of the list.
What are operations you can perform on a Linked List?

Student: append, prepend, insert, delete, update

NDLtutor: Can you explain more?

Student: Append and prepend adds new node to linked list. delete removes a node from list and update is used to update a node in the list

NDLtutor: Hmmm…. Can you improve that?

Student: append and prepend operations are used to add new nodes to linked list. the delete function is used to delete a node and the update function is used to update the node.

NDLtutor: What is the operation used to go through the Linked List from beginning to end?

Student: i dont know

NDLtutor: Give it a try!

Student: i dont remember

NDLtutor: Ok
The basic operations that can be performed on a linked list are:
1. Traversing a linked list – going through the linked list
2. Append - adding a new node (to the end) of the list
3. Prepend – adding a new node (to the beginning) of the list
4. Inserting – adding a new node to a specific position on the list
5. Deleting – deleting a node from the list
6. Updating – update a node in the list

NDLtutor: We have covered the basic definition of a Linked List. Can you summarize your answer now?

Student: a linked list is a collection of nodes that is linked together by references between each node. the last node of the linked list has a reference to null. The main operations on a linked list are traversing (going though), append – adding a new now to the end of the list, prepend – adding a new node to the beginning of the list, inserting a new node at a specific position, deleting a node from the list and updating a node in the list.

NDLtutor: Good job!
Let's reflect on our discussion now!
On a scale of 0 to 10 (0: minimum, 10: maximum) how would you rate your first answer?

Student: 4

NDLtutor: Hmmm! Your initial answer was not a complete definition and it was missing many basic concepts.
How is your last summarized answer different from your initial answer?

Student: Its much better

NDLtutor: Your initial answer was incomplete. Your final answer contains much more information.
What were the concepts that you missed?

Student: traverse

NDLtutor: That's right!
You missed: Each Node has a reference to Next Node, The last Node has a reference to NULL, Traversing a linked list
Did you encounter any misconceptions?

Student: No

NDLtutor: That's right. You have no misconceptions about the basics of Linked List!
Is there anything still unclear?

Student: no

NDLtutor: Great!
What aspects of this topic do you think you need to concentrate more on?

Student: I need to remember what the operations do

NDLtutor: Do you think you will be able to discuss this topic with a peer/friend now?

Student: yes

NDLtutor: Great!
Ok the last question…. Did this discussion improve your understanding/knowledge?

Student: yes it did

NDLtutor: That's great!
I'm afraid you have not been able to prove your level of proficiency in the topic of Linked List to be Very High; therefore I cannot update my belief base at this time.
Please review the topic to improve your level of proficiency in the topic of Linked List.
Keep working hard!
This session has concluded.

Dialogue Log#7

NDLtutor: You have indicated that your proficiency level in the topic of Linked Lists is Very High.
This is different from my belief about your proficiency in the topic of Linked Lists.
What makes you think you have Very High level of proficiency?

Student: i think my level is Very High

NDLtutor: Are you sure? You don't seem to be confident about your answer.

Student: yes

NDLtutor: Great!
But your performance in the topic of Linked Lists was High.
Do you want me to give you an overview of your performance?

Student: no

NDLtutor: I'm sorry!
To update my belief I need to be convinced about your knowledge level in the topic of Linked Lists.
Do you want to discuss the topic of Linked Lists with me?

Student: ok

NDLtutor: Great!
Can you give me a complete definition of Linked List?

Student: a linked list consists of nodes where each node contains one or more data fields and a reference to the next node. the entry point is called the head. the basic operations of the linked list are traverse, append, prepend, insert and delete.

NDLtutor: Good!
The last node contains a _____ reference to indicate the end of the list?

Student: null

NDLtutor: Great!
What does the Update operation do to a Linked List?

Student: it is used to update the node in the list. it can be used to set the data of a node.

NDLtutor: Great!
The basic operations that can be performed on a linked list are:
1. Traversing a linked list – going through the linked list
2. Append - adding a new node (to the end) of the list
3. Prepend – adding a new node (to the beginning) of the list
4. Inserting – adding a new node to a specific position on the list
5. Deleting – deleting a node from the list
6. Updating – update a node in the list

NDLtutor: We have covered the basic definition of a Linked List. Can you summarize your answer now?

Student: a linked list consists of nodes where each node contains one or more data fields and a reference to the next node. the entry point is called the head. the last node contains a null reference to indicate the end of the list. the basic operations of the linked list are traverse, append, prepend, insert, delete and update.

NDLtutor: Good job!
Let's reflect on our discussion now!
On a scale of 0 to 10 (0: minimum, 10: maximum) how would you rate your first answer?

Student: 9

NDLtutor: Ok!
How is your last summarized answer different from your initial answer?

Student: not very different. just missed a few things in the initial answer

NDLtutor: Your initial answer was incomplete. Your final answer contains complete information.
What were the concepts that you missed?

Student: Null reference and update operation

NDLtutor: That's right!

You missed: The last Node has a reference to NULL, Updating a Node in a linked list.
Did you encounter any misconceptions?

Student: No

NDLtutor: That's right. You have no misconceptions about the basics of Linked List!
Is there anything still unclear?

Student: no

NDLtutor: Great!
What aspects of this topic do you think you need to concentrate more on?

Student: i don't know… im good at this topic

NDLtutor: Do you think you will be able to discuss this topic with a peer/friend now?

Student: yes i can

NDLtutor: Great!
Ok the last question…. Did this discussion improve your understanding/knowledge?

Student: well it helped me revise a couple of things

NDLtutor: That's great!
After our discussion I still think your proficiency level in the topic of Linked Lists is High.
Please keep working hard!
This session has concluded.