# Predicting Student Behavior Models in Ill-structured Problem-Solving Environment

Deepti Reddy Patil [1]

[1] *Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai, India*

**Abstract**

Ill-structured problem-solving requires novices to be scaffolded to ap-ply various cognitive and metacognitive skills. The cognitive skills are under-standing the problem, formulating it into subproblems, generating alternative de-signs, and selecting the optimal solution. Metacognitive skills are the ability to monitor, evaluate and improve upon their performance. We have developed an online teaching-learning environment named Fathom to teach ill-structured problem-solving skills in the context of solving software design problems in the data structures course. A total of 100 undergraduate CS students were trained using Fathom and results showed significant learning gains from pre-test to post-test. Posttest scores were not enough to analyze how learners at various levels (low, medium, and high) interact with the learning environment. This paper discusses our approach to building student models of the low, medium, and high performers using hidden Markov model methodology based on the log data generated in Fathom. These models will be used to predict their performance on new data which will help to intervene during their interactions with the learning environment.

**Keywords**

Ill-structured problem, Technology-enhanced learning environment, Student model, Hidden-Markov model, Software-design

## 1. Introduction

Ill-structured problems are complex because they have vaguely defined or unclear goals and unstated constraints; they possess multiple solutions and involve multiple criteria for evaluating solutions [14]. Software design is a complex and ill-structured activity in which a software designer must deal with issues such as understanding the unknown problem domain, eliciting requirements from multiple stakeholders' viewpoints, identifying alternative solutions, and making decisions based on selection criteria [1].

Novices find design daunting and face some difficulties like – the inability to structure a problem, fixation while creating a solution, and evaluation of the solution. The teaching-learning efforts in software engineering and software design need also to be directed toward students being able to perform ill-structured tasks such as structuring open problems, creating integrated solutions, and evaluating them. Research [1,16,17] shows that experts are able to deal with these issues by implicitly applying cognitive skills such as drawing diagrams to simulate scenarios that aid in eliciting require and constraints which may not be directly stated initially. However, novice tends to jump to a single solution without understanding the problem which affects the quality of the software. Hence, in addition to content knowledge, students need to be explicitly trained to effectively use these practices while solving software design problems.

We have designed and developed a technology-enhanced learning environment named Fathom, for the teaching-learning of ill-structured problem-solving skills in the context of solving software design problems in the data structures course. In Fathom, the learners are scaffolded towards applying the skills through structured guidance in the context of solving a software design problem. The targeted software

Workshop on Artificial Intelligence in Support of Guided Experiential Learning, Held in conjunction with the International Conference on Artificial Intelligence in Education (AIED) 07 July 2023, Tokyo, Japan
EMAIL: deepti.reddy@nmims.edu (A. 1)

design skills were: the ability to visualize the problem space before formulating sub-problems and the ability to generate alternative design options before selecting one solution based on evaluation criteria. The learning activities are designed with both cognitive and meta-cognitive scaffolds to aid learners in not just solving the problem but to monitor and improve upon their skills. The cognitive scaffolds include: prompts, hints, case-study, study material, drawing tools to aid visualization, workspace to record learners' responses, and metacognitive scaffolds include: system-evaluated feedback. The learners' actions during interaction with the Fathom were logged in the form of the triplet: <learner_id, timestamp, clicked_button >.

We conducted research studies with undergraduate engineering students (N=100) to evaluate the effectiveness of Fathom in learning these skills. The methodology used is a pretest-intervention-posttest research design. The results showed significant gains from the pretest to posttest in quality of problem formulation, and solution quality. However, the scores were not helping in providing insights into the interaction behavior of the high, medium, and low-performing students. To investigate the relationship be-tween learning performance and the use of strategies by low, medium, and high per-formers, it became important to examine how these activities came together as larger behavior patterns and strategies. Research shows that the hidden Markov models (HMMs) are the most appropriate, as they allow to identify some of the students' general behavior patterns from sequences of their interactions with the system [2]. The aim of this paper is to discuss the process of building a student model for high, medium, and low-performing groups of learners using HMM, analyze their interaction behavior, and use these models to predict the performance of the learner.

The structure of the paper is as follows: in section 2, the learning environment Fathom and the study conducted are discussed, followed by related work in learner modeling is discussed in section 3, and in section 4 the methodology used to build HMM model is discussed in detail with its analysis and conclusion.

## 2. Fathom Description

A Technology Enhanced Learning Environment named Fathom for teaching-learning of ill-structured problem solving [8], was designed and developed based on the principles of effective cognitive and metacognitive support for novice learners [3,4,5,6,7]. The cognitive and metacognitive support provided in Fathom is discussed in detail in this section.

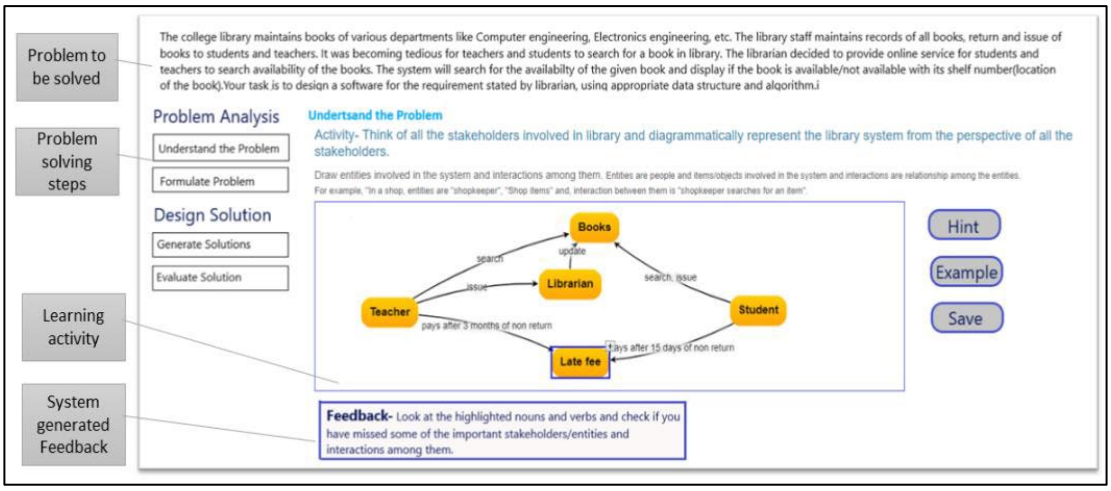
## 2.1. Cognitive and metacognitive scaffolds in Fathom

**Cognitive support** in Fathom was provided in the form of structured guidance to-ward solving the software design problem which is ill-structured in nature. The learning activities were designed for each step of problem-solving to direct learners' thinking toward applying the targeted skills as shown in Fig 1. The learning activities designed in Fathom are:

1. Understand the problem- In this activity learners were prompted to explore the prob-lem by identifying entities and users in the system. The learners were prompted to draw the model of the system to aid in visualizing the system from the perspective of various entities and the user's point of view.
2. Formulate problem- In this activity, the learners were prompted to write the sub-goals: the data to be stored and operations to be performed by the software to achieve the stated goal.
3. Generate solutions- This activity is designed to expand solution space by generating alternative solutions. The learners were prompted to draw cognitive maps to list alternative design options for each sub-problem.
4. Evaluate solutions- This activity prompted the evaluation of alternative solutions based on the identified selection criteria and constraints. This activity is designed to select the appropriate solution using a decision matrix. The decision matrix was used to allow learners to evaluate alternative solutions against the constraints and rank the solutions. Finally, justify their selected solution.

Each learning activity was scaffolded with question prompts, solved examples, demo videos, learning resources, and cognitive tools (drawing cognitive maps, pro-cons table, decision matrix, etc.).

**Metacognitive scaffolds** in Fathom were provided to allow learners to engage in assessing and improving on the skills performed. Post-activity, the learning environment was designed to assess the performance of the learner and provide corrective or positive feedback. The aim of the feedback was to ensure that the learners reflect on their skills, identify gaps and improve on their skills.



The college library maintains books of various departments like Computer engineering, Electronics engineering, etc. The library staff maintains records of all books, return and issue of books to students and teachers. It was becoming tedious for teachers and students to search for a book in library. The librarian decided to provide online service for students and teachers to search availability of the books. The system will search for the availabilty of the given book and display if the book is available/not available with its shelf number(location of the book).Your task is to design a software for the requirement stated by librarian, using appropriate data structure and algorithm.i

Activity- Think of all the stakeholders involved in library and diagrammatically represent the library system from the perspective of all the stakeholders.

Draw entities involved in the system and interactions among them. Entities are people and items/objects involved in the system and interactions are relationship among the entities.
For example, "In a shop, entities are "shopkeeper", "Shop items" and, interaction between them is "shopkeeper searches for an item".

**Feedback-** Look at the highlighted nouns and verbs and check if you have missed some of the important stakeholders/entities and interactions among them.

**Figure 1.** Overall Design Features of Fathom

## 2.2. Study and Results

The field study was done with 100 undergraduate computer engineering students to evaluate the effectiveness of Fathom in learning ill-structured problem-solving skills. The research methodology used was a pretest-intervention-postest design.

The procedure of the study is as follows:

### 2.2.1. Pre-test

First, all the participants were given a pre-test in which a shop-inventory problem was solved using a worksheet- "Design software system for a supermarket to keep track of items whose quantity is below the threshold at the end of the day."

The participants took 30 minutes to solve the problem, and after completing the work-sheets were collected from them.

### 2.2.2. Intervention

Immediately after the pre-test, the students interacted with the Fathom in which they solved a software design problem for library management. The participants interacted with Fathom for almost 2 hours.

### 2.2.3. Post-test

After the intervention, during the post-test, the participants were given a worksheet to solve a new problem- *"Design software for a bank to allow customers to do online banking (check balance, withdraw money, and check balance)*". The participants were told to apply the software design skills learned during the intervention. The participants took almost 30 minutes to solve the post-test problem.

The student responses generated during the pre-test, intervention, and post-test were evaluated to assess the quality of the problem formulation and solution design. The scores showed a significant gain from the pre-test to post-test in the quality of problem formulation (p=0.02, effect size= 0.75) and solution quality (p=0.00, effect size= 1.15). This shows that the pedagogical features of Fathom were effective in learning problem-solving skills.

However, the scores were not of much help in providing insights into how students interacted with different features in the environment and their effect on learning problem-solving skills. This paper focuses on log data analysis to model learners' behavior and analyze how high, medium, and low scorers interact with the learning environment and prediction of the performance based on their interaction behaviors.

## 3. Related Work in Learner Modeling

A Hidden Markov Model (HMM) is a finite state machine that has some fixed number of states. It provides a probabilistic framework for modeling a time series of multivariate observations. For a given observation sequence O, i.e., O1 , O2 , O3 ,………OT, the hidden Markov model (λ) is characterized using three parameters: λ= {A, B, n), where A is {aij} transition matrix, where aij represents the transition probability from state i to state j, B = {bj(Ot)} observation emission matrix, where bj(Ot) represents the probability of observing Ot at state j, π = {πi} the prior probability, where πi represent the probability of being in state i at the beginning of the experiment, i.e., at time t = 1

HMM is used as a classifier or predictor in various applications like for speech signal recognition [12, 15], DNA sequence analysis [9], handwritten characters recognition [12], natural language domains, etc. Another area of application is in educational data mining to build student behavior models in various interactive learning environments and to predict student learning behaviors. For instance, hidden Markov models (HMM) were used to model school students' behavior based on the trace data generated from Betty's brain system which used the pedagogy of learning by teaching [2]. In a later study, Jeong et al. (2010) applied the same HMM approach to study the learning behavior of adult professionals in an asynchronous online learning environment. In particular, their exploratory study was aimed at identifying the main phases of the student's learning process in the examined course and investigating the differences between high and low-performing students in terms of their transitions through the identified phases of the course.

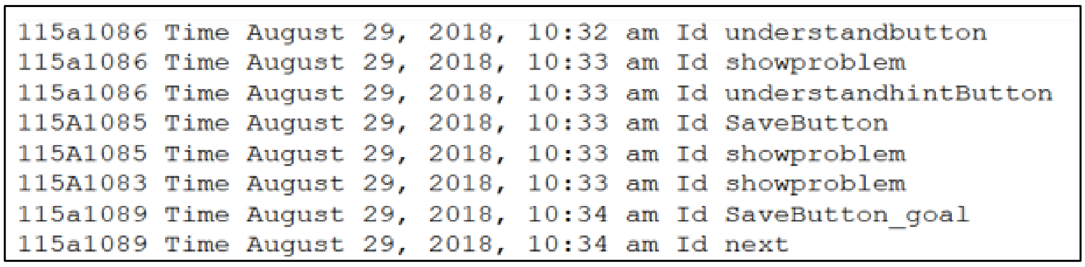
We propose to use HMM similar to the work proposed by Jeong (2008) to investigate how engineering students interact with learning environments designed for complex problem solving and analyze student behaviors to get insights on how the learning environment facilitates the learning of complex problem solving among high, medium and low performers.

## 4. Methodology for Obtaining Behavior Model

Our approach involves four steps that appear in most data mining applications [2]: (i) log data collection that records student interactions with the system; (ii) data cleaning by parsing the generated log files and splicing the information into desired activity sequence data that will form the input to the HMM generating algorithm; (iii) construct the HMM models; and (iv) interpret generated models as student learning behaviors and compare models across low, medium and high performers, (v) predict student behaviors and evaluate the model. We describe each of these steps in greater detail below.

### 4.1. Log data collection and processing

The log data was collected from Fathom in the form of triplet <learner_id, timestamp clicked_button>. The sample log data collected is as shown in Fig. 2.



```
115a1086 Time August 29, 2018, 10:32 am Id understandbutton
115a1086 Time August 29, 2018, 10:33 am Id showproblem
115a1086 Time August 29, 2018, 10:33 am Id understandhintButton
115A1085 Time August 29, 2018, 10:33 am Id SaveButton
115A1085 Time August 29, 2018, 10:33 am Id showproblem
115A1083 Time August 29, 2018, 10:33 am Id showproblem
115a1089 Time August 29, 2018, 10:34 am Id SaveButton_goal
115a1089 Time August 29, 2018, 10:34 am Id next
```

**Figure 2.** Raw log data collected in Fathom

The log data in its raw form is very difficult to comprehend and needs to be processed before we can perform any operations on it. The log files consist of all the activities carried out by the students in the form of button clicks, edits made in the drawing tools and the text fields, access to hints, and examples, etc.

The other dataset we worked on was the score sheet of post-test to identify low, medium, and high scorers. The students scoring low (score<2) in quality of problem formulation and solution were categorized as low scorers, students scoring medium (score=2) were categorized as medium scorers, and others were categorized as high scorers. Out of 100 students, 16 students did not complete the posttest, hence we considered only 84 students, out of which 20 students were categorized as low scorers, 36 as medium scorers, and 28 as high scorers. The log sequences were then assigned to each student and three separate input dataset was created as input for the hidden Markov model.

## 4.2. Parsing the Log Files

In this study, we derive learners' behavior patterns by analyzing the sequence of their interactions with the system. The system had four major steps: understand_problem, formulate_solution, generate_solutions and evaluate_solution. In each activity, learners' were prompted to provide responses and save them. During the activity, access to additional resources was provided in the system in the form of solved examples, notes, and hints. Post-activity, the system evaluated the responses and gave positive or corrective feedback to the learner. After, reading the feedback, learners were allowed to evaluate and modify their responses.

To simplify the interpretation task we mapped learners' actions in each activity into one aggregate activity. For example, all the edits made in the understand_problem activity, like drawing the diagrams and saving in the first attempt as UP, accessing re-sources as RA, and then redoing after saving the responses as REDO, etc. All student activities were expressed as the six activities summarized in Table 1.

**Table 1**
Student activities and related actions

| Activity | Student actions |
|---|---|
| UP | Saving the diagram drawn in the Understand_Problem activity in the first attempt. |
| FG | Saving the responses in the Formulate_Problem activity in the first attempt. |
| GS | Saving the solutions generated in the Generate_Solutions activity in the first at-tempt. |
| EV | Saving the evaluation of solutions against criteria and constraints using the decision matrix in the Evaluate_Problem activity in the first attempt. |
| RA | Accessing resources in the form of hints, examples, notes, etc |
| REDO | Modifying the responses after the first attempt |

For example, all occurrences where the sequence follows "solving a problem state and saving to check feedback and then redoing the same activity" are replaced by the REDO state (representative of redoing the task). For example, the sequence: "UP", "Save", "UP", "Save", is replaced by "UP", “REDO”. All occurrences where the sequence follows solving a problem state and then checking the resources for hints, notes, examples, etc., are replaced by the RA state. Examples of the resultant sequences of two students are shown in Fig. 3.

“115A1086”: ["UP", "FG", "UP", "FG", "UP", "FG", "FG", "GS", "REDO", "EV", "GS", "FG", "RA", "GS", "REDO", "EV", "REDO", "EV", "UP", "FG", "GS"],
"115A1090": ["UP", "FG", "GS", "EV", "RA", "UP", "RA", "FG", "REDO", "RA", "UP", "RA", "FG", "REDO", "RA", "GS", "RA", "REDO", ”EV”, "RA", "EV", "REDO", "RA", "GS", "REDO", "EV", "REDO", "RA", "GS", "RA", "GS", "REDO", "REDO", "FG", "EV", "GS", "REDO", "EV", "GS", "RADO", "EV", "GS", "REDO"]

**Figure 3.** Parsed data for two students

## 4.3. Training and testing the HMMs

The first step in interpreting this behavior data was to build hidden Markov models from the sequence of observable events. A hidden Markov model is characterized by three sets of parameters: initial probability vector π, state transition probability matrix, A, and output probability matrix, B [15]. In order to train the HMM, we divided the dataset into two sets, one training set and one test (recall) set in the ratio of 80:20.

The difficult part of the modeling process is to determine the optimal set of parameters and the size of the model (number of states) that maximizes the likelihood of the input sequences. Jeong (2010) compared two common iterative convergence optimization schemes, the Baum-Welch and the segmental K-Means algorithms to achieve the optimal model parameters, which include (π, A, B) and the number of states in the model. The results showed that the optimal number of states is six using both Baum-Welch and the segmental K-Means algorithms, which we have followed in our model-ing. We used the Viterbi algorithm for sequential decoding and calculating transition probabilities between states.

The parsed activity sequences of three groups-low, medium, and high performers were used to derive three sets of hidden Markov models as shown in Fig. 4, Fig 5, and Fig. 6 respectively.

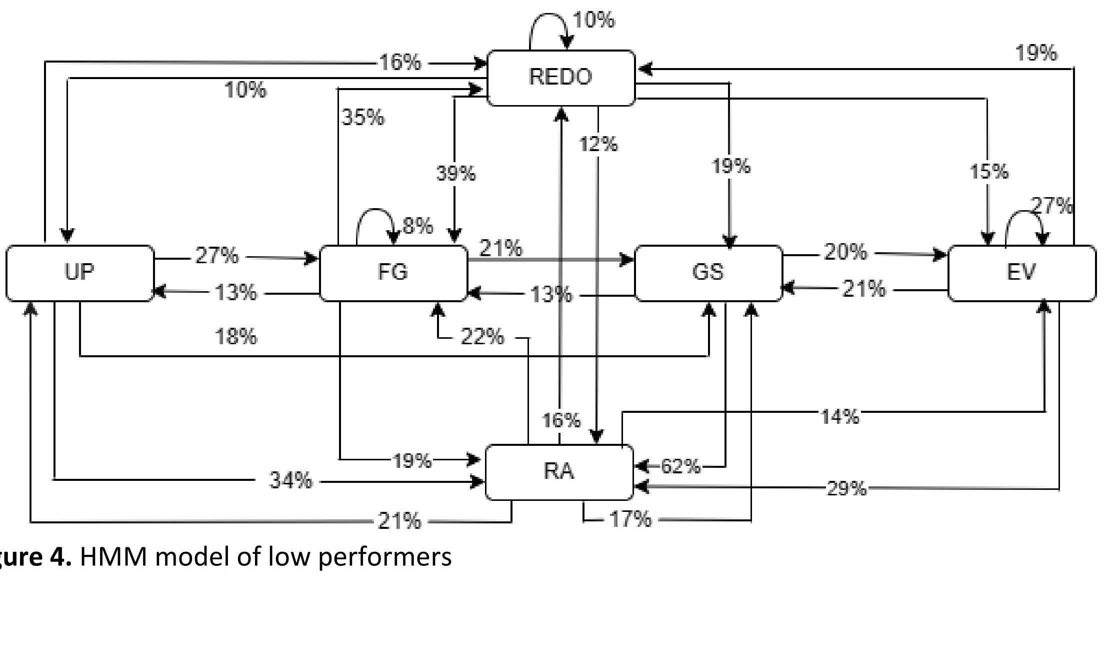


**Figure 4.** HMM model of low performers

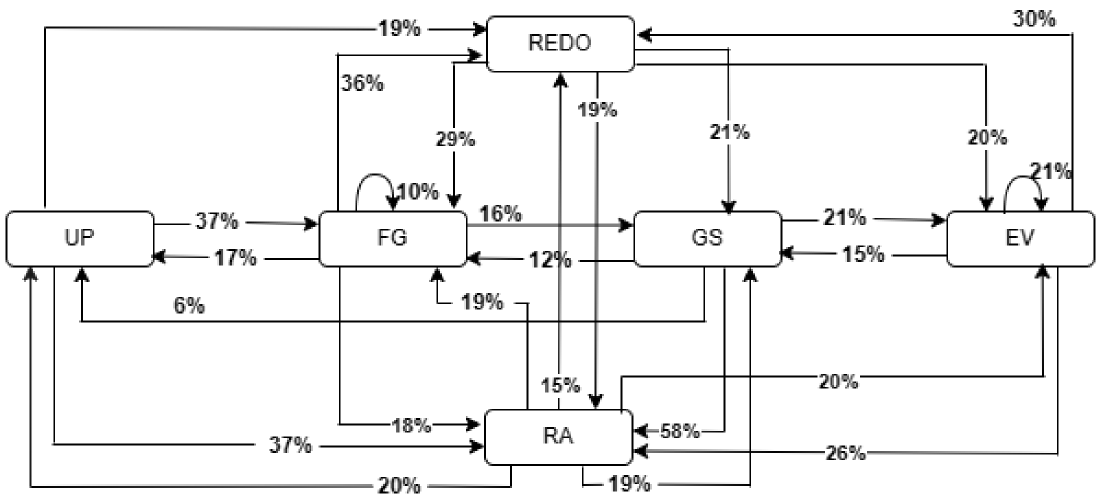


**Figure 5.** HMM model of medium performers

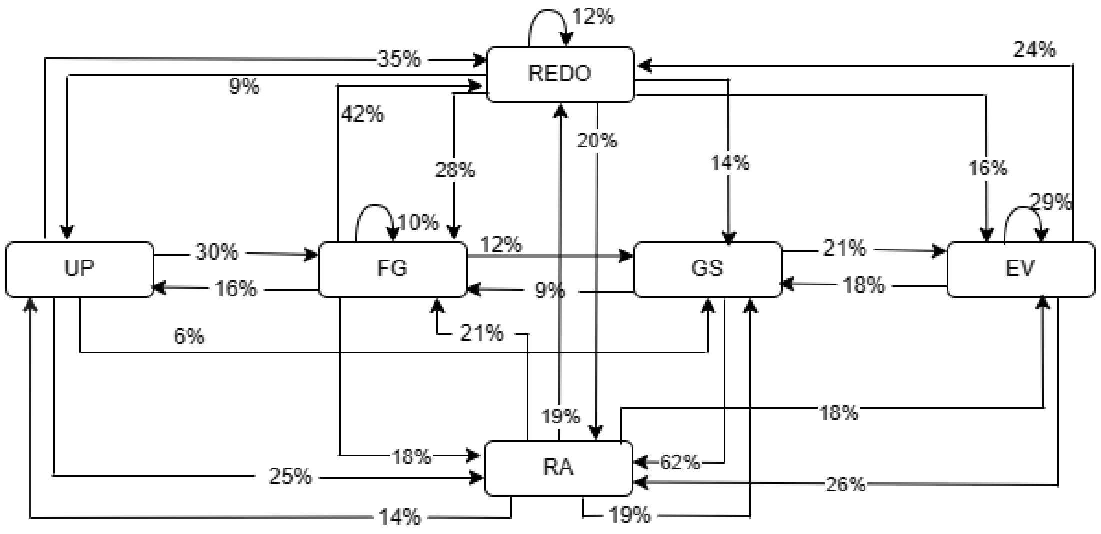


**Figure 6.** HMM model of high performers

Each model is made up of a set of states, the activity patterns (the output probability) associated with each state, and the transition probabilities between states. The transition probability associated with a link between two states indicates the likelihood of the student transitioning from the current state to the indicated state. For example, the HMM model states student in the high-performer group (Fig. 6) in state UP would demonstrate a 30% likelihood of transitioning to state FG, and 25% likelihood of transitioning to state RA. Likelihoods less than 5% were not represented in the figure, ex-plaining why these numbers do not sum to 100%. HMMs are so named because their states are hidden. That is, they are not directly observed in the input sequences, but provide an aggregated description of the student's interactions with the system. Sequences of states may be interpreted as the student's learning behavior patterns. We investigate further by testing these models for the prediction of low, medium, and high performers against the test data.

For each sequence O in test data, the probability is calculated using each model, for example, P(O|low), P(O|medium), and P(O|high). The sequence is classified into high, medium or low based on the one with the high probability. A multi-class confusion matrix was calculated to measure the performance of our prediction model as shown in the figure. Each entry in a confusion matrix denotes the number of predictions made by the model where it classified the classes correctly or incorrectly.

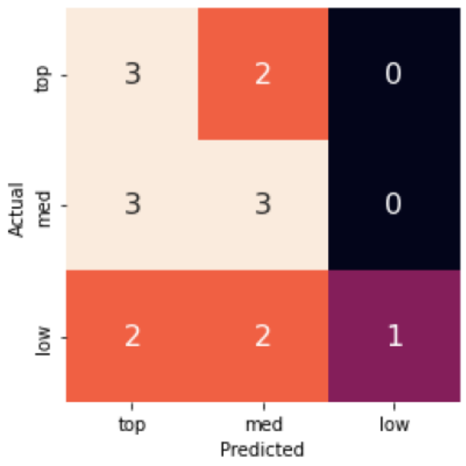


**Figure 7.** Confusion matrix for our model

The F1 score in our overall model is 0.44 and low, medium, and high scores are 0.33, 0.46, and 0.46 respectively.

## 5. Analysis of HMM Patterns

The analysis of transition shows certain patterns in both low, medium, and high per-formers. The likelihood percentage of high scorers transitioning to REDO state in each activity is more compared to low and medium scorers. For instance, the high scorers' transitions from UP to REDO state with 35% likelihood compared to 19 % and 16% in medium and low scorers respectively. This shows that high scorers were more responsive to the feedback given by the system and went back to the same activity to improve their responses.

The likelihood percentage of low and medium scorers transitioning to RA (resource access) state is higher than high scorers. For example, the likelihood percentage of the low and medium scorers transitioning from UP to RA state is 34% and 37% respectively compared to 25% in high scorers. The low and medium scorers were accessing re-sources like hints, videos, and learning material more often which indicates that they had difficulty comprehending the activity. The difference between low and medium scorers is that low scorers' transition probability reduced towards the last phase, while medium scorers showed consistency in doing tasks till the end.

Our next level of analysis consisted of examining the interactions among the meta-cognitive states and their transitions in our models. These interactions inform us about students' typical learning behavior patterns among high, medium, and low performers. We find that the students in the low and medium-scorer groups tend to stay mainly in the cognitive task of doing activities in UP, FG, GS, CR, and EV states, while the high and medium-scorer students tend to transition to the higher-level states such as REDO state. High scorers tend to transition between doing and redoing the task, and occasion-ally referring to the help provided by the system, exhibiting metacognitive behavior. While low and medium scorers tend to be in a cognitive state of doing the activity and are less likely to monitor and reflect on their skills. The resource usage rate is high and REDO is low in low scorers which indicates that low scorers have difficulty in doing or comprehending the activities compared to high scorers. The model evaluated against test data had an accuracy and F1 score of 0.44. The prediction may be improved by providing more observations in training the model.

Overall analysis shows that high and medium scorers show more metacognitive behaviors, while low and medium scorers exhibited more help-seeking behaviors. How-ever, medium scorers showed consistency in interactions till the end, and low scorers' interactions were reduced towards the last step. This analysis is useful to enhance the learning experience in the learning environment Fathom, and predict student behaviors based on their interaction patterns in the learning environment, and provide timely help to low performers.

## 6. Conclusion

In this paper, we discussed the process of creating student models representing learning patterns of high, medium, and low performers in learning ill-structured problem-solving skills in the technology-enhanced learning environment, named Fathom. The model was built using the hidden Markov model (HMM) using the log data generated in Fathom. The analysis shows that high scorers exhibit metacognitive behaviors in terms of the ability to do the activity, monitor, and reflect on their skills. While, low and medium scorers tend to rely more on the resources given in the system and exhibit more help-seeking behavior, which implies that they have difficulty comprehending and doing the activity. While low scorers are not retaining the doing of tasks till the end while medium scorers were consistent in completing the tasks till the end. The model accuracy may be improved in the future by training the model with more observations.

## 7. References

[1] Tang, A., Aleti, A., Burge, J., & van Vliet, H. (2010). What makes software design effec-tive?. Design Studies, 31(6), 614-640.

[2] Jeong, H., & Biswas, G. (2008, June). Mining student behavior models in learning-by-teaching environments. In Educational data mining 2008.

[3] Geiwitz, J. (1994). Training Metacognitive Skills for Problem Solving. ADVANCED SCIENTIFIC CONCEPTS INC PITTSBURGH PA.

[4] Ge, X. (2013). Designing learning technologies to support self-regulation during ill-structured problem-solving processes. In International handbook of metacognition and learning technologies (pp. 213-228). Springer, New York, NY.

[5] Bannert, M., & Mengelkamp, C. (2013). Scaffolding hypermedia learning through metacognitive prompts. In International handbook of metacognition and learning technologies (pp. 171-186). Springer, New York, NY.

[6] Jonassen, D. (2011). Supporting problem solving in PBL. Interdisciplinary Journal of Problem-Based Learning, 5(2), 8.

[7] Narciss, S. (2013). Designing and evaluating tutoring feedback strategies for digital learning. Digital Education Review, (23), 7-26.

[8] Reddy, D., Iyer, S., & Sasikumar, M. (2018, December). Technology Enhanced Learning (TEL) Environment to Develop Expansionist-Reductionist (ER) Thinking Skills through Software Design Problem Solving. In 2018 IEEE Tenth International Conference on Tech-nology for Education (T4E) (pp. 166-173). IEEE.

[9] Liebert M A (2004), Use of runs statistics for pattern recognition in genomic DNA se-quences. Journal of Computational Biology, Vol. 11, pp. 107-124.

[10] Kovanović, V., Gašević, D., Joksimović, S., Hatala, M., & Adesope, O. (2015). Analytics of communities of inquiry: Effects of learning technology use on cognitive presence in asynchronous online discussions. The Internet and Higher Education, 27, 74-89.

[11] Jeong, H., Biswas, G., Johnson, J., & Howard, L. (2010, June). Analysis of productive learn-ing behaviors in a structured inquiry cycle using hidden Markov models. In Educational Data Mining 2010.

[12] Vinciarelli A and Luettin J (2000), Off-line cursive script recognition based on continuous density HMM, Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition, Amsterdam, pp. 493-498.

[13] Xie H, Anreae P, Zhang M, Warren P (2004), Learning Models for English Speech Recognition, Proceedings of the 27th Conference on Australasian Computer Science, pp. 323-329.

[14] Jonassen, D., Strobel, J., & Lee, C. B. (2006). Everyday problem solving in engineering: Lessons for engineering educators. Journal of engineering education, 95(2), 139-151.

[15] Rabiner, L., & Juang, B. (1986). An introduction to hidden Markov models. ieee assp mag-azine, 3(1), 4-16.

[16] Adelson, B., & Soloway, E. (1985). The role of domain experience in software design. IEEE Transactions on software engineering, (11), 1351-1360.

[17] Guindon, Raymonde. "Knowledge exploited by experts during software system design." International Journal of Man-Machine Studies 33.3 (1990): 279-304.