Chen, W. et al. (Eds.) (2016). Proceedings of the 24th International Conference on Computers in Education. India: Asia-Pacific Society for Computers in Education

Designing PHyTeR: a system to teach troubleshooting skill

Kavya ALSE[a*], Sridhar IYER[a] & Sasikumar M[b]
[a]*Interdisciplinary program in Educational Technology, IIT-Bombay, India*
[b]*CDAC Bombay, Mumbai, India*
*kavyaalse@iitb.ac.in

Abstract: Troubleshooting skill is required by undergraduates in Computer Science training to be IT professionals. Current curriculum in Computer Science doesn't give sufficient emphasis on teaching troubleshooting to learners. We are following educational design research (EDR) to engineer a solution to teach troubleshooting skill. In this paper we present an overview of EDR applied to this problem and the design of a system called 'PHyTeR'. PHyTeR is intended to teach troubleshooting to CS undergraduates by choosing scenarios from the domain of Computer Networks. A plan of evaluation to evaluate the design of the system is also discussed.

Keywords: Troubleshooting Skill, Teaching thinking skill, Design of learning environments, Computer Networks, Educational Design Research

1. Introduction

Computer Science undergraduates interact with various types of systems – it might range from running a program on an IDE to building a drone or building/interacting with multiple versions of libraries, APIs. Troubleshooting is an important component in building and working with all these systems. "Troubleshooting is a process which ranges from the identification of problem symptoms to determining and implementing the action required to fix that problem" (Schaafstal, Schraagen, & Berlo, 2000). According to Jonassen (Jonassen, 2010), troubleshooting is an ill-structured problem which needs troubleshooters to have an understanding of the system they are troubleshooting and keep track of troubleshooting process. Also it involves higher level cognitive activities like analyzing the behavior of the system, generating plausible explanations for errors seen, switching between different levels of details of the system etc. Troubleshooting like many other thinking skills is pan-domain since it is applicable in the domains of electrical, chemical, mechanical & computer science engineering.

Traditionally, tools like gdb, wireshark, circuit analyzers are used by educators to teach troubleshooting. However, the problems chosen to teach troubleshooting are usually simple compared to the open ended problems faced by professionals. Our solution approach is that by allowing students to practice troubleshooting skill with fairly complex problems would prepare them better for their profession. The tools will be of more use if the students understand the process of troubleshooting. The goal of this paper is to describe the solution approach that we are following to teach troubleshooting skill to Computer Science undergraduate students. Then the design of a system which we call 'PHyTeR' is described in detail followed by an evaluation plan for the solution.

2. Related Work

2.1 Troubleshooting skill

Troubleshooting is a moderately ill-structured problem because the troubleshooter has to determine what information is needed, require deep level understanding of the system. The competencies of troubleshooting skill has been defined by researchers from different domains (Johnson, 1987; Ross & Orr, 2009; Schaafstal et al., 2000). Some consider it as iterative testing of hypotheses and others argue that troubleshooting includes all processes between representing problem space and verification of the solution. We are considering the following competencies of troubleshooting which we have synthesized from literature: i) Problem Space Representation, ii) Hypothesis Generation, iii) Hypothesis Prioritization, and iv) Design & Run Tests.

2.2 Approaches to teach skills like troubleshooting

There have been worksheet based approaches to teach troubleshooting (Schaafstal, Schraagen, & Berlo, 2000). They give a process overview of troubleshooting but fail to give contextual, domain dependent & independent in-time scaffolds. One such approach was to train students to become better troubleshooters by giving them structured practice on authentic problems (Ross & Orr, 2009). There have also been many technology enhanced learning environments to teach complex skills like design, scientific inquiry and modeling especially for school children (Basu, Dickes, Kinnebrew, Sengupta, & Biswas, 2013; Sun & Looi, 2012; White et al., 2002) but for troubleshooting skill. These aim to teach reasoning & skills to students in different contexts. On the other hand there have been number tools which help college students and professionals to troubleshoot systems viz wireshark, debuggers built in IDEs etc. They give opportunity to learners to interact with real systems/simulations but don't help learners in the 'process' of troubleshooting. It has been argued that troubleshooting needs to be taught separately because teaching learners to build and design systems is not sufficient for them to troubleshoot systems (Jonassen, 2010).

2.3 Expert systems to troubleshoot

Another thread of literature related to troubleshooting is where expert systems were developed to do better troubleshooting or 'better' debuggers were designed. Some examples of designing better debuggers were providing visualizations of programs being executed, providing stack traces etc. is intended to reduce the cognitive load on troubleshooters (Hejmady, 2011). Studies on design of expert systems have emphasized the need for students representing the problem space and having a functional understanding of the components of a system (Kleer, Williams, & De Kleer, 1989).

3. Research Methodology – Educational Design Research

Educational Design Research (McKenney & Reeves, 2014) is a way of engineering practical solutions to problems in educational domain. EDR can be used in designing policies, educational products, processes or programs (McKenney & Reeves, 2014). Along with developing solution, an important aspect of EDR is to develop theory/understanding about the development of solution or how the solution works in a context. EDR is an iterative process of cycles consisting of problem analysis, design & development of solution and evaluation of solution. The following diagram shows the EDR process employed in designing the TEL environment 'PHyTeR'. Based on the available literature on troubleshooting skill we have designed task structure of the TEL environment. Literature on expert and novice studies informed the affordances and scaffolds. Expert and novice studies in the domain of Computer Networks are planned to obtain detailed inputs from the domain which would further inform design decisions.

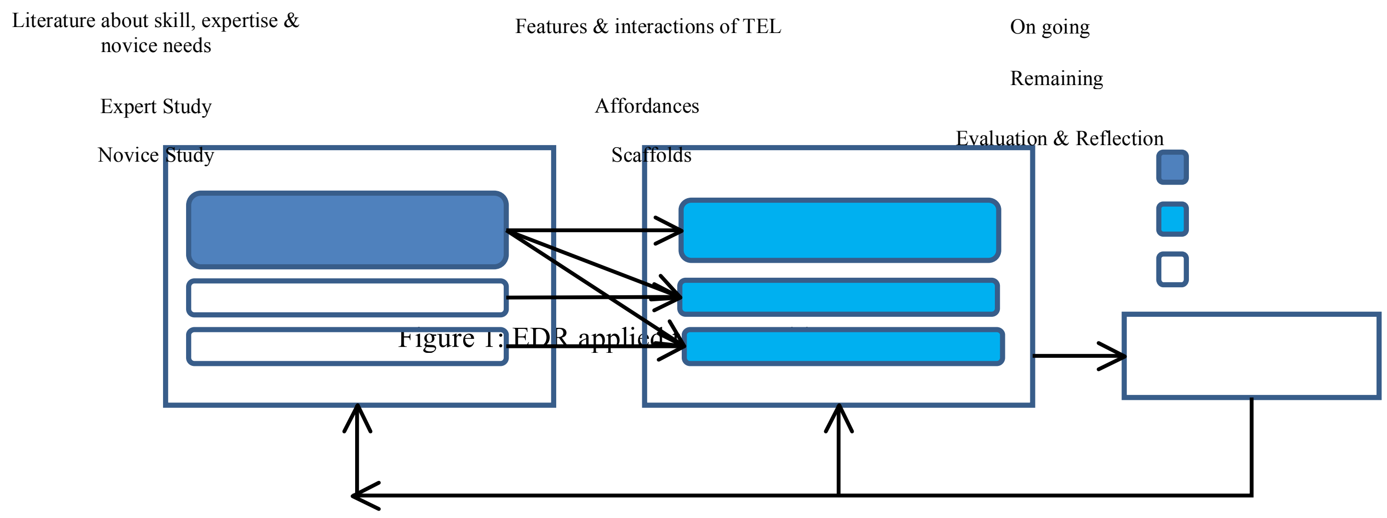


Figure 1: EDR applied

4. Theoretical Basis for solution

Following are the theoretical backings for our conjectures about learning. Modeling is aimed at developing mental models about troubleshooting. External representations and scaffolds help in reducing the complexity of the task.

4.1 Model based learning

Models are concrete representations of abstract objects/systems in the real world. Usually models are built by reducing the complexities of real world to focus on few aspects of interest (Seel & Blumschein, 2008). Modeling helps as a concrete external representation while interacting with it and also helps in 'meaning-making' in the process of building it. An internal 'mental model' is created when people interact with the above modeling activities and 'meaning' appears when these mental models become coherent, rich with experiences, ideas, thoughts (Seel & Blumschein, 2008). Modeling is also the activity suggested for decomposing complex processes to simpler elements and mechanics (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013).

4.2 External Representations

Troubleshooting requires learners to have a picture of system being troubleshot & its components, keep track of the process of troubleshooting and interpret the cause and effect relationship within the system. Kirsh (Kirsh, 1995) argues how such complex activity can be not just supported but also enhanced and mediated by using external representations. External visual representation have shown to reduce the difficulty of process of problem –solving. Experts have also been shown to use visual representations frequently during problem-solving to support reasoning (Moreno, Ozogul, & Reisslein, 2011).

4.3 Scaffolds

Quintana (Quintana et al., 2004) argue that even though introducing software tools to learners might make the task more difficult initially, there is credit in using software tools to scaffold learners in structuring an ill-structured problem, giving guidance and allowing them plan and monitor their performance. Narciss (Narciss, 2013) provide a framework for providing such scaffolds using interactive contextual feedback at different levels: cognitive, metacognitive and motivational. We are using some of these in the design of PHyTeR.

5. Design of 'PHyTeR'

Our design is based on a framework called 'TELoTS framework' (Murthy, Iyer, & Mavinkurve). This framework gives actions & guidelines on developing a smart learning environment specifically for thinking skills like troubleshooting. Design starts with defining the competencies of a skill, coming up with learning objectives and assessments to evaluate the competencies. Then the framework suggests analyzing expert actions and integrating the requirements with design principles to come-up with the design of the system.

Based on this, we have designed a system called 'PHyTeR' to teach troubleshooting in the domain of Computer Networks for Computer Science undergraduate students. 'PHyTeR' stands for P – Problem Space Representation, Hy – Hypothesis generation & prioritization, Te – Testing which are the competencies and R – Reflection. Next sub-sections describe the design in detail.

5.1 Key Features of 'PHyTeR'

PHyTeR is intended to

a. Support students to switch between big picture and small picture
b. Provide in-time scaffolding & reflection questions to guide the process
c. Provide modeling and interaction with simulations in the same interface

5.2 Design of learning activities

The system provides learners with troubleshooting scenarios that are ordered from simple to complex. There is a task corresponding to each competency and a reflection task at the end. At the beginning of the task, the students will be explained how that task is a step towards troubleshooting. Reflection questions will be asked when the students take decisions like selecting a hypothesis or complete a task. The system features were derived based on learning outcomes for the competency, assessment rubrics for the competency and inputs from expert novice literature. The following table shows how the learning design elements for the task of problem space representation were derived.

Table 1: Components of design decisions

Competency	Learning Outcome	Input from expert study	Input from novice studies	Learning design principles	Features in PHyTeR
Problem Space Representation	Students will be able to analyze the system in structure and function terms	Experts describe the system in functional terms	Novices have inadequate system understanding	Make students describe structure, & function of the system	Question prompts to identify relevant components, link between them and their functions

5.2.1 Problem Space Representation Activity

Experts are said to have a rich representation of the problem that they are trying to solve (Jonassen, 2010). With respect to troubleshooting, this representation consists of various devices present in the network, the links between them (structure). It also includes a representation of the function of each device and how they combine to produce the function of the whole network.

PHyTeR helps students in building a representation of the problem by using 'annotated topology builder'. The topology creator has a device bar consisting different types of network devices like terminals, links, access points etc. Students will have to drag and drop the devices on the topology space to create the annotated topology. They will have to annotate the devices with protocols and configurations that have to be set on the device for it to work correctly. Learners might not have all the required domain knowledge. For this purpose, there is a domain book which has computer networks related information (text or animations or video).

The topology that the learners create is like a reference for them to perform next activities. Let us consider an example scenario which consists of many devices and one of the device (terminal T2) is not able to connect to another device (Terminal T1). An example of the annotated topology is shown in figure 2.

5.2.2 Hypothesis Generation Activity

Once students have an understanding of the problem space, they need to come up with multiple plausible causes for the error. Learners are asked to generate hypotheses and attach these hypotheses to the device & component to which it is related. There is a general window where hypotheses not

R1 SW1 SW2 R3 T2

Function: To forward packets
Protocol: IPv4, IPv6, ARP
Configuration: DNS, Subnet mask, Firewall

related to any specific device can be noted down. Examples for the hypotheses: Internet is down, firewall is blocking the port etc.

Figure 2: An interface to interact with the simulator

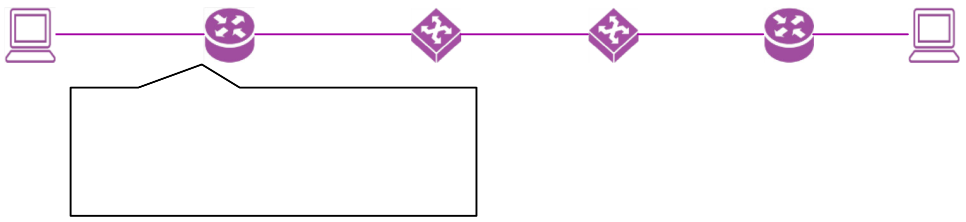


Figure 2: Annotated Topology Map

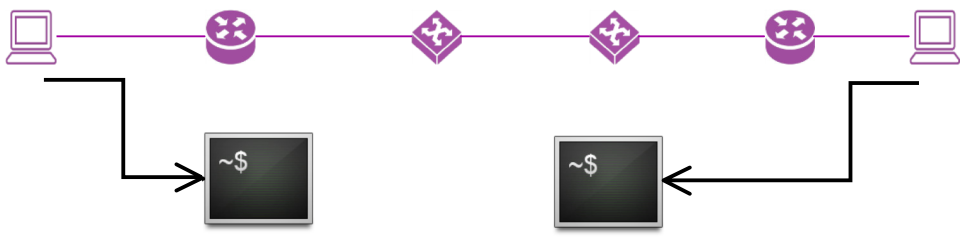
5.2.3 *Hypotheses Prioritization Activity*

With a number of hypotheses at hand, the progress of troubleshooting depends on the order of selection of hypothesis to test. This is done by prioritizing the hypotheses using different strategies. One example of a strategy is described in a study which suggested that experts suggests hypothesis at higher level (system/sub-system level) and then generate hypotheses at specific level (device/component level) to investigate further (Johnson, 1987). This sort of strategy information is given as hints to learners when they click on the hint button. A list of all the hypotheses generated by learners is displayed along with all details related to hypotheses (the device/component to which it is related etc.). They have to drag and order the hypotheses according to the prioritization strategy.

When the learners select one hypothesis, they will be asked a justification question – to justify their prioritization of the hypotheses. This is intended as a metacognitive reflective prompt and has been claimed to enhance linking between the error and plausible explanations.

5.2.4 Design & Run Test Activity

After a hypothesis is selected for testing, the learners have to design a test. For this they have choose the testing methods (commands to be executed, logs to be checked, configurations to be checked etc.). Then they have to predict the result that would be obtained by performing the test. Then an interface to the real erroneous topology (with a simulator in background) is displayed as shown below:



This topology has simulated network devices and connections between them. The learners can click on any of these devices to open a console to interact with the corresponding device. When learners have completed the test and obtained the result, they have to compare it with the predicted result. If the predicted and observed results match, it means they have found a causal relationship. They will have to record this in their causal map book. If the predicted and observed results don't match, then it means either the hypothesis was incorrect (then they have to test a new hypothesis) or the test that they performed is incorrect (then they have to redesign the test). Based on the interpretation of the result students might want to generate new hypotheses or discard previously generated hypotheses. At this stage students can use a 'Route Map' which shows them a summary of all the previous tests performed and results obtained. This is intended to help them in switching between 'small picture' (design & perform a test) to 'big picture' of the overall troubleshooting process.

5.2.5 *Reflection Activity*

Apart from the justification reflection activities spread in between the activities, learners will do a 'Reflection by summarizing' activity. Here the students will have to give a summary of the troubleshooting process that they did and then compare it with an expert solution.

6. Proposed evaluation of 'PHyTeR'

The first part of evaluation is to validate the competencies, troubleshooting scenarios and system design by experts in Computer Networks. This is required because the competencies and current design are based on literature from other domains like chemical plants and mechanical systems. The second part of the study is to evaluate that PHyTeR actually helps in learning troubleshooting. For this, as a first step a study is intended to improve the user interface of the system and ensure that learners doesn't have any difficulty in using the system and do the required tasks. This would complete the first cycle of EDR. The research questions that we are considering for initial evaluation of the system are: i) Does the features or scaffolds in the system help learners to complete a troubleshooting task? ii) What is the perception of students with respect to learning and usability? A single group pre-post study with 30 students will be conducted to evaluate their troubleshooting abilities before & after using the system.

7. Conclusion

The paper described the design of a system called 'PHyTeR' to teach troubleshooting by considering problems from the domain of computer networks. An evaluation plan based on design is proposed which we intend to use in the evaluation of the solution.

Acknowledgements

We would like to thank all the people at IDP-ET, IIT Bombay who helped in defining and designing the system.

References

Basu, S., Dickes, A., Kinnebrew, J. S., Sengupta, P., & Biswas, G. (2013). CTSiM: A Computational Thinking Environment for Learning Science through Simulation and Modeling. *The 5th International Conference on Computer Supported Education*.

Hejmady, P. (2011). *A Cognitive Model and Gaze-Based Evaluation of Multiple Representation use during Program Comprehension and Debugging*.

Johnson, S. D. (1987). Knowledge and skill differences between expert and novice service technicians on technical troubleshooting tasks. Retrieved from http://eric.ed.gov/?id=ED290043

Jonassen, D. H. (2010). *Learning to solve problems: A handbook for designing problem-solving learning environments*.

Kirsh, D. (1995). The intelligent use of space. *Artificial Intelligence*, *73*(1-2), 31–68.

Kleer, J. De, Williams, B. C., & De Kleer, J. (1989). Diagnosis with behavioral modes. *International Joint Conference On Artificial Intelligence*, 1324–1330. Retrieved from http://portal.acm.org/citation.cfm?id=1623967

McKenney, S., & Reeves, T. . (2014). Educational Design Research. In *Handbook of Research on Educational Communications and Technology* (Vol. Springer N, pp. 131–140).

Moreno, R., Ozogul, G., & Reisslein, M. (2011). Teaching with concrete and abstract visual representations: Effects on students' problem solving, problem representations, and learning perceptions. *Journal of Educational Psychology*, *103*(1), 32–47.

Murthy, S., Iyer, S., & Mavinkurve, M. (n.d.). Pedagogical Framework for Developing Thinking Skills using Smart Learning Environments. *(Under Review), IDP-ET, IIT-B.*

Narciss, S. (2013). Designing and evaluating tutoring feedback strategies for digital learning environments on the basis of the interactive tutoring feedback model. *Digital Education Review*, *23*(1), 7–26.

Quintana, C., Reiser, B. J., Davis, E. A., Krajcik, J., Fretz, E., Duncan, R. G., … Soloway, E. (2004). A Scaffolding Design framework for Software to Support Science Inquiry. *Journal of Learning Sciences*, *13*(3), 337–386.

Ross, C., & Orr, R. R. (2009). Teaching structured troubleshooting: integrating a standard methodology into an information technology program. *Educational Technology Research and Development*, *57*(2), 251–265.

Schaafstal, A., Schraagen, J. M., & Berlo, M. Van. (2000). Cognitive Task Analysis and Innovation of Training : The Case of Structured Troubleshooting. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, *42*(1), 75–86.

Seel, N. M., & Blumschein, P. (2008). Modeling and Simulation in Learning and Instruction: A Theoretical Perspective. In P. Blumschein, D. Jonassen, & W. Hung (Eds.), *In P. Blumschein, W. Hung, D. Jonassen & J. Strobel (Hsg.)Model-based approaches to learning: Using systems models and simulations to improve understanding and problem solving in complex domains* (pp. 17–40).

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351–380.

Sun, D., & Looi, C.-K. (2012). Designing a Web-Based Science Learning Environment for Model-Based Collaborative Inquiry. *Journal of Science Education and Technology*, *22*(1), 73–89.

White, B. Y., Frederiksen, J., Frederiksen, T., Eslinger, E., Loper, S., & Collins, A. (2002). Inquiry Island : Affordances of a Multi-Agent Environment for Scientific Inquiry and Reflective Learning. *Proceedings of the Fifth International Conference of the Learning Sciences (ICLS).*, 1–12.