

RESEARCH

Free and Open Access

Fathom: a technology-enhanced learning environment for teaching-learning of expand-reduce skills in software design

Deepti Reddy Patil ^{1*}, Sridhar Iyer ², Sasikumar M. ³

*Correspondence:
deepti.reddy@nmims.edu
Department of Computer
Engineering,
Mukesh Patel School of
Technology Management &
Engineering,
SVKM's Narsee Monjee Institute
of Management Studies (NMIMS)
Deemed-to-University,
Mumbai-400056, India
Full list of author information is
available at the end of the article

Abstract

Software design problems are often characterized as ill-structured because the requirements are not clearly defined. These problems offer various solution paths and the criteria used to select the solution may not be known initially. The expertise and experience of designers play a crucial role in determining the quality of software design. Less experienced designers often tend to prematurely narrow down their options to a single solution without fully exploring the problem-solution space. This tendency has a negative impact on the overall quality of software design. Studies have shown that systematically expanding the problem space before reducing to problem formulation and exploring the solution space before reducing it to a single solution (expand-reduce skills) improves the quality of the design. We have designed and developed a technology-enhanced learning environment (TELE) named Fathom for scaffolding expand-reduce (ER) skills in software design problems in a data structures course. In this paper, we present three cycles of design, development, and evaluation of Fathom based on the design-based research (DBR) approach. In the first DBR cycle, we identified and evaluated Fathom's pedagogical features in learning ER skills. In the second cycle of DBR, the aim was to improve the design of Fathom for the learning and transfer of ER skills. Fathom was revised in the third cycle of the DBR to scaffold metacognitive skills. The main contribution of this research is pedagogical design for facilitating the learning of expand-reduce skills in solving software design problems.

Keywords: Software design problem, Ill-structured problem, Expand-reduce skills, Design-based research (DBR), Learning environment, Cognitive and metacognitive scaffolding

Introduction

Software design problems are sometimes ill-structured and complex, especially when the problem space and solution space are not well defined. The characteristics of an ill-



© The Author(s). 2024 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

structured problems are as follows: requirements may be broadly defined (Pressman, 2010); the designer may not be familiar with the area in which the software is to be created (Adelson & Soloway, 1985; Tang et al., 2008); and the designer has to turn partial needs into specifications (Guindon, 1990; Pressman, 2010). In the solution space, there may be various solution paths and alternative design possibilities, and the criteria for choosing the best solution may not be explicitly articulated (Jonassen et al., 2006). For example, in the software design problem “Design library management system,” the problem definition is broad and needs to be decomposed into subproblems. The solution space may consist of identifying appropriate data structures to store library books and using appropriate algorithms to perform various operations. Based on the needs and limitations applicable to the problem area, designers must choose acceptable data structures and algorithms (Guindon, 1990; Pressman, 2010; Tang et al., 2010).

The literature on solving ill-structured problems indicates that having the ability to comprehend and visualize the problem space is crucial for transforming an ill-defined problem into a well-defined problem. This approach is defined as expansionist thinking, which consists of comprehending the system by identifying the components and interrelationships between them, followed by the reductionist approach, which consists of decomposing the problem into solvable subproblems (Ackoff, 1979; Volkema, 1983). Similarly, in the solution space, alternative solutions may be developed using strategies such as brainstorming, mind mapping, attribute listing, and analogous thinking (Liu & Schonwetter, 2004). After generating solutions, one may reduce to a single solution using various evaluation strategies such as pros and cons analysis and decision matrix (Pugh, 1991). In this paper, we refer to the ability to explore the problem-solution space and eventually reduce towards the formulation of subproblems and solution design as expand-reduce (ER) skills.

Research has shown that a designer’s expertise and experience have a significant impact on the quality of software design (Adelson & Soloway, 1985; Tang et al., 2008). Expert designers are adept at visualizing the problem solution space, applying heuristic techniques to search the solution space, and quickly selecting an appropriate option. However, novices struggle to solve design challenges because they lack experience in solving such problems. They tend to reduce early in the solution design without visualizing the problem-solution space. This affects design quality because of reasons such as defining the problem too narrowly, failure to decompose problems into subproblems, and fixation on a specific solution without openly examining other options (Ellspermann et al., 2007; Zannier et al., 2007).

According to research, designers should use expand-reduce (ER) skills to increase design quality, especially when the designer is inexperienced or the problem area is unclear (Adelson & Soloway, 1985; Tang et al., 2008, 2010; Zannier et al., 2007). We propose a

technology-enhanced learning (TEL) environment named Fathom, designed and developed for undergraduate computer engineering students, with a focus on the learning of expand-reduce skills in the context of solving software design problems. The instructional design of Fathom is grounded in the framework suggested for scaffolding ill-structured problem-solving processes (Bannert & Mengelkamp, 2013; Jonassen, 2011; Xun & Land, 2004).

This paper presents the pedagogical design of Fathom and its implementation through three design-based research (DBR) cycles (Amiel & Reeves, 2008; Reeves, 2006). In the first DBR cycle, we investigated the challenges encountered by novices when tackling software design issues. This inquiry involved an exploratory study conducted with 40 second-year undergraduate engineering students. In this study, the students were given a software design problem to be solved using a worksheet. Student responses were evaluated and scored using a rubric designed to assess ER skills. The results showed that students scored low in ER skills. Based on the findings of Study 1 and insights gathered from the literature survey, we designed and developed Fathom-Ver1 (version 1). We tested the effectiveness of Fathom-Ver1 by conducting exploratory study 2 with the aim to investigate the effectiveness of Fathom-Ver1 in performing activities related to ER skills. A group of 49 second-year undergraduate engineering students took part in this research, engaging in problem-solving within Fathom-Ver1. Subsequently, they participated in a student perception survey and focus group interview. The analysis of our results showed that the average scores of the students were better in Study 2 than in Study 1; however, the survey and interview data analysis revealed that students had difficulty understanding the activities designed in Fathom. In the second DBR cycle, the design elements of Fathom-Ver1 are upgraded by adding worked examples, feedback, and drawing tools to the learning environment. The second DBR cycle focused on the learning and transfer of ER skills. Study 3 tested the effectiveness of Fathom-Ver2 in learning ER skills. The study was conducted with 52 second-year undergraduate computer engineering students. The research method was a one-group pretest-posttest study to measure learning gains before and after the intervention. The results showed that there was a significant improvement in scores for ER skills from pretest to posttest; however, log-data analysis showed that only 20% of students revised their responses after self-evaluation, which implies that learners did not exhibit metacognitive behaviors. The third cycle of DBR focused on strengthening metacognitive scaffolds to aid learners in evaluating their ER skills. Study 4 was conducted to evaluate Fathom-Ver3. The study involved 50 second-year undergraduate computer engineering students. The research method used was a one-group pretest-intervention-posttest. The results showed significant improvement in the quality of software design from pretest to posttest, which implies that Fathom-Ver3 was effective in learning ER skills in solving software design problems. Log-data analysis showed that the students evaluated

their responses and were taking appropriate corrective measures to improve their ER skills during the intervention, thus exhibiting metacognitive behaviors.

In the section “Theoretical background: problem-solving literature,” we discuss related work in software design problem-solving, ill-structured problem-solving, and the formulation of expand-reduce skills. In the sections “Design-based research” to “DBR cycle 3,” DBR cycles 1, 2, and 3 are discussed in detail, followed by a discussion, limitations, and conclusion.

Theoretical background: problem-solving literature

In this section, the importance of design problem-solving in engineering education is established, followed by a literature review on software design problem-solving skills. Next, the characterization of expand-reduce skills is discussed, followed by issues and challenges in existing studies and the research goal.

Design problem-solving in engineering education

One of the outcomes of engineering education is the ability to identify, formulate, and solve engineering problems and design a product under a given set of constraints, which directly or indirectly solves real-life problems and improves quality of life (ABET, 2013). Design problems are ill-structured, and the skills needed to solve design problems need to be taught explicitly to students during engineering education. However, engineering education focuses more on teaching content and solving well-structured problems (Cooperrider, 2008; Dym et al., 2005; Jonnasen, 2006). Computer engineering courses for undergraduates often involve design problem-solving skills as they require students to apply theoretical knowledge to real-world challenges. Computer engineering subjects in which design problem-solving skills are particularly crucial are database design (Mitrovic & Suraweera, 2016), UML design (Moritz & Blank, 2008), software design (Mitrovic & Weerasinghe, 2009; Nyhoff, 2005), and computer networks (Lian, 2012), etc. Our work focuses on software design problem-solving in data structures courses, as choosing appropriate data structures and algorithms is an important skill in software design, and the designer has to make design decisions based on the criteria relevant to the given problem (Tang et al., 2010).

Expand-reduce skills in software design

Based on the problem-solving literature, various cognitive and metacognitive processes and tools that expert designers implicitly use to expand-reduce during problem analysis and solution design phases are discussed in detail.

Problem analysis

The problem-solving literature argues that the quality of problem formulation depends on the ability to see the entire problem space and then decompose the problem into subcategories (Ackoff, 1979; Dennis et al., 1999). This process is categorized as the expansionist and reductionist (Volkema 1983) approach, in which the problem space is expanded before reducing (decomposing) the problem into subproblems. The expansionist approach involves understanding the system by identifying the parts and the interrelationships between parts (Ackoff, 1979). Studies have shown that various visualization techniques, such as drawing a cognitive map representing various actors/concepts as nodes and links as the interaction between them help in understanding the problem as a whole and improves the quantity and quality of the generated problem statements (Eden & Ackermann, 2001; Norese, 1995).

Many studies have been conducted with experts in software design problem-solving to study the cognitive and metacognitive processes involved (Adelson & Soloway, 1985; Guindon, 1990; Tang et al., 2008, 2010). In problem analysis, the designer must transform incomplete and ambiguous specifications into high-level system design. A high-level design describes the main software functions and subfunctions. Before the formal specifications are documented, expert designers analyze the problem by creating and simulating mental models at various levels of abstraction. Eventually, the subproblems are decomposed from these mental models. Designers create a mental model of the system using external representations, such as drawing a state transition diagram or listing entities with their respective properties and actions. The drawing chosen is based on the prior experience of the designer; for example, if the designer has prior experience in designing control systems, then he may prefer a state transition diagram and a software designer may prefer entity-action representation or data flow diagrams. If the problem to be solved is: “Design a library management system for a college” then different representations that can be used by the designers are a state transition diagram (Figure 1) or listing entities and corresponding actions (Figure 2).

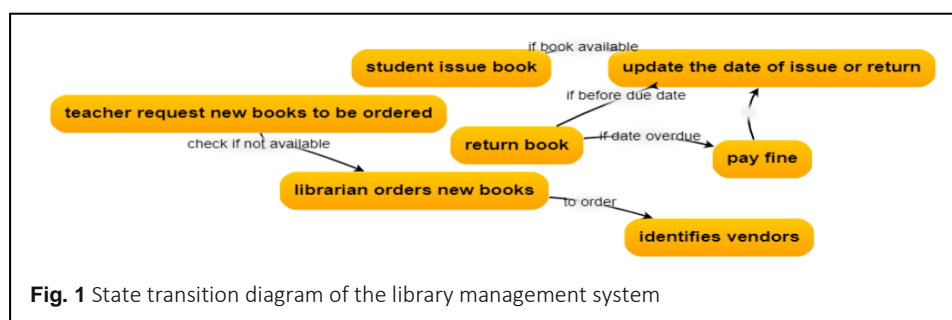


Fig. 1 State transition diagram of the library management system

Entity	Action
Librarian	Issue books, return books, maintain cards, order books, calculate fine
Teacher/student	Issue/return books, re-issue book, request new books
Book	Update date of issue or return

Fig. 2 Entity-action list of the library management system

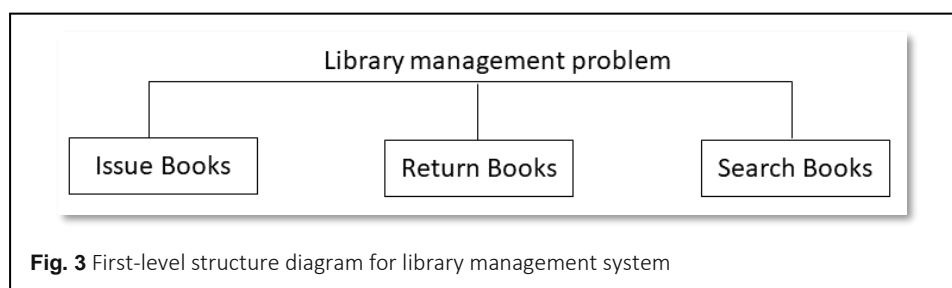
These diagrams act as mechanisms for problem understanding and problem decomposition into solvable subproblems. For example, the above diagram will trigger the inference of new functions, such as `issue_book (book-id)`, `return_book (book_id)`, and `calculate_fine (book-id, student-id)`. The drawing of the library problem in terms of the various entities/actors involved and the interactions between them helps in simulating various scenarios. This leads to the creation of mental models, which in turn trigger the decomposition of problems into solvable subproblems.

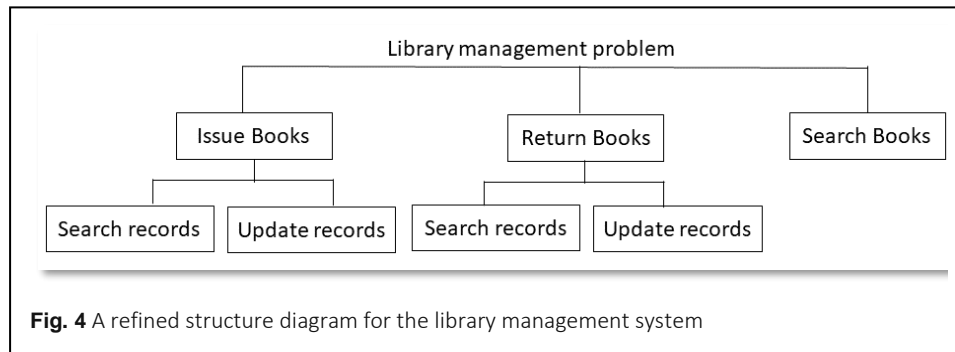
Solution design

Software design problems (Nyhoff, 2005) are often complex and cannot be visualized or anticipated at the outset of all details of a complete solution to the entire problem. One approach to tackling such complex problems is to decompose the original problem into simpler subproblems, each of which can be considered individually. Some subproblems may have to be further partitioned into simpler subproblems that can be solved directly.

For example, in the library management problem, the original problem is partitioned into simpler subproblems: issue, return, and search books. The mind map can be used to visualize the solution space at various levels of abstraction (Koustousov & Kudryavtsev, 2017). The first level of abstraction is illustrated in Figure 3.

Typically, one or more first-level subproblems may still be complex, and must be divided into smaller subproblems. For example, for issuing books, one must search for the book in the set of records and update the issue date in the corresponding record, as shown in Figure 4. This process may continue for several more levels of refinement until each





subproblem is sufficiently simple and straightforward to implement. This solution may consist of designing storage structures for data and identifying algorithms to process data. For example, for the search problem, book records may be stored in ascending order on one of the key fields (accession number) in an array, and the algorithm might be a binary search algorithm. However, the decision of which data structure and algorithm to use must be made based on the problem requirements and constraints to be satisfied.

Experienced designers are good at identifying the selection criteria; they may not perform an exhaustive search in the solution space and quickly reduce to a few alternatives based on the selection criteria, quickly perform a trade-off analysis of the available options, and make design decisions (Guindon, 1990). However, inexperienced designers benefit more by explicitly generating all possible alternatives, explicitly stating the selection criteria, evaluating alternative options against the selection criteria, and making a design decision by explicitly justifying the selection. This process helps the designer backtrack alternative solutions if the current solution is later found to be unviable (Tang et al., 2008, 2010).

In software design problem-solving, designers use various forms of external representation to expand the solution space for possible alternative design options. To generate a variety of design possibilities, sketches play an important role in supporting mental simulation, reviewing progress, and considering alternatives. The external representations suggested are mind maps, concept maps, tables, lists, etc., for designing and evaluating solutions (Guindon, 1990; Mangano et al., 2014). These representations were used to identify missing information and to ensure the completeness of the solution.

Convergent thinking techniques are used to reduce by evaluating alternative ideas using various criteria and selecting the appropriate idea. For example, making a decision on buying equipment for a company involves evaluating all alternatives based on various selection criteria, such as cost and usability. Some of the tools widely used for evaluating alternatives are the decision matrix (Pugh, 1981, 1991); and Analytic Hierarchy Process (AHP) (Saaty, 2008). The decision matrix (Pugh, 1981, 1991) is used to make design decisions by evaluating alternative designs based on multiple criteria and deciding which one best meets all criteria. Decisions based on multiple criteria are complex, resulting in

	Design1	Design2	Design3	Design4
Criteria1	1	3	4	3
Criteria2	2	5	4	2
Criteria3	4	3	1	3
Criteria4	3	2	1	4
Total score	10	13	10	12

Fig. 5 Example of decision matrix

inconsistent and irrational decisions. The Pugh Matrix provides a simple approach to comparing alternative designs against each criterion. The alternative designs and criteria are listed in the table shown in Figure 5, which demonstrates the use of a decision matrix to evaluate and select alternatives. Four design alternatives were evaluated against the four criteria. The designs are rated on a scale, of example, 1-5 against each criterion, and the total rating for each design is calculated. The design with the highest rating was selected for the study. The alternative is to weigh the criteria based on their importance over other criteria and multiply the rating with the weight of each criterion. The advantage of Pugh's decision matrix is that it is easy to implement. However, this method fails if the selection of criteria is incomplete or incorrect because the effectiveness of Pugh's matrix is related to the quality of the selection criteria. If the selection criteria are incorrect or incomplete, the selection decision can go wrong.

Analytic Hierarchy Process (AHP) is another popular technique proposed by Saaty (2008), which is used to choose among multiple criteria and choices; for example, selecting a car by evaluating alternative choices (Baleno, Honda city, Ford aspire) against multiple criteria (cost, style, mileage, reliability). The basic principle of AHP is to perform a pairwise comparison of criteria and perform a matrix calculation (eigenvector) to determine the relative importance of one criterion over another. The main advantage of AHP is its ability to rank choices in order of their effectiveness in meeting conflicting objectives. The drawback is the ambiguity in the interpretation of Saaty's rating scale and the complexity of mathematical calculations.

A summary of the various cognitive processes and tools used to expand-reduce the problem-solution space is shown in Figure 6.

In this paper, we characterize expand-reduce skills from expansionist-reductionist thinking (Ackoff, 1991; Ellspermann et al., 2007; Volkema, 1983), divergent-convergent thinking literature (Basadur et al., 2000; Howard et al., 2008; Liu & Schonwetter, 2004; Pugh, 1991; Saaty, 2008) and software design problem-solving literature (Guindon, 1990; Mangano et al., 2014; Tang et al., 2008, 2010). Expand-reduce (ER) skills are defined as the ability to expand-reduce the problem-solution space by understanding the problem from multiple perspectives before formulating subproblems, and generating solutions before selecting a single solution, as shown in Figure 7.

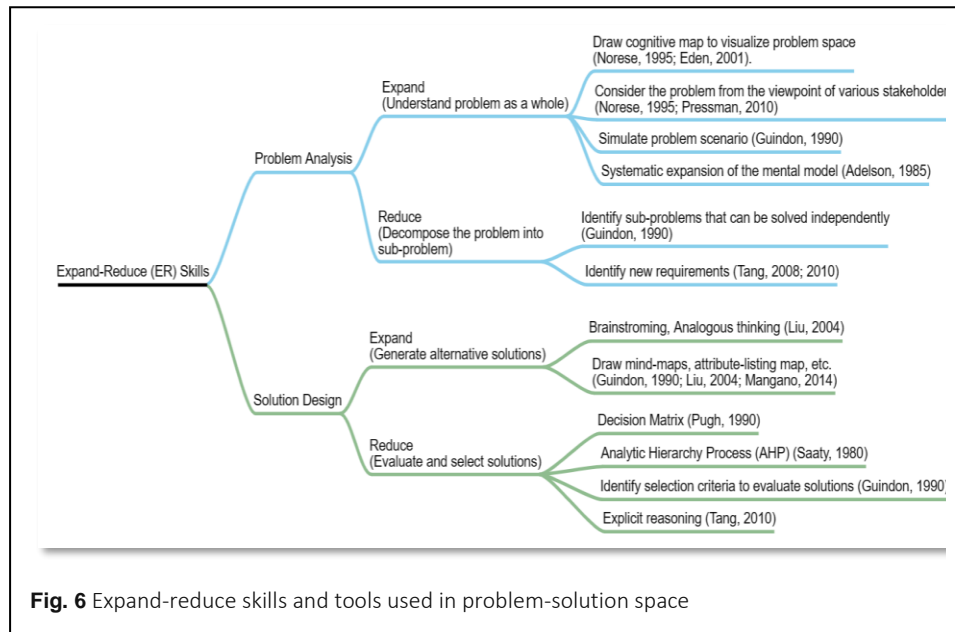


Fig. 6 Expand-reduce skills and tools used in problem-solution space

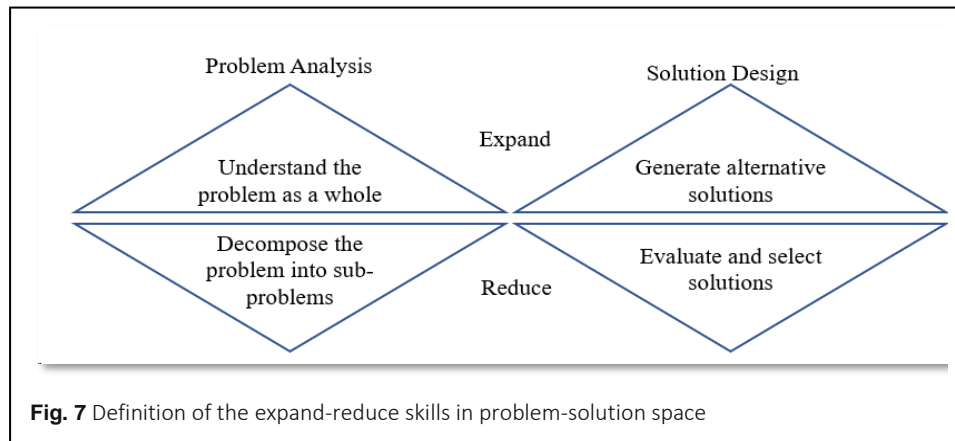


Fig. 7 Definition of the expand-reduce skills in problem-solution space

Teaching-learning techniques for problem-solving

The characteristics of software design problems are similar to those of ill-defined problems in which the start and end goals are unclear, and there is no algorithm defined to transform the start goal into the end goal. In addition, the start state is underspecified and the goal state is defined in terms of highly abstract features (Goel & Pirolli, 1992; Mitrovic & Weerasinghe, 2009). A literature survey was conducted to identify teaching-learning techniques suggested for ill-structured problem-solving. To improve students' performance in solving ill-structured problems, it is necessary to provide externalized support or scaffolding to facilitate cognitive and metacognitive processes (Bannert & Mengelkamp, 2013; Ge, 2013; Jonassen, 2011; Tang et al., 2018; Xun & Land, 2004). Various scaffolding techniques have been discussed in this section.

Question prompts

The framework proposed by Xun and Land (2004) suggests the use of question prompts for scaffolding ill-structured problem-solving. The question prompts enable to direct students' attention to important aspects of problem-solving at both the cognitive and metacognitive levels. Question prompts have been empirically proven to be effective in guiding learners step-by-step throughout the entire process of a specific problem-solving task (Ge et al., 2005; Xun & Land, 2004). Question prompts can guide students' attention to specific aspects of their learning process, thereby helping them to monitor and evaluate problem-solving processes. Question prompts include procedural, elaboration, and reflection prompts, each of which serves different cognitive and metacognitive purposes. Procedural prompts help learner complete specific tasks, such as writing or problem-solving, and have been used successfully to help students learn cognitive strategies in specific content areas. Some examples of procedural prompts include the Identify entities involved in the problem. *Identify alternative solutions*. Elaboration prompts are designed to prompt learners to articulate their thoughts and elicit explanations. Some examples are as follows: *Why is it important? How does this affect the selection?* Reflection prompts encourage reflection on a metalevel that students do not generally consider. A few examples of reflection prompts can be used to help students justify the viability of the proposed solution against alternatives: *What are the pros and cons of this solution? Are there alternative solutions?*

Structured guidance

Learning problem-solving skills should be supported by providing a learning environment with authentic and complex problems. This allows learners to learn skills by engaging in activities in that field (Jonassen, 2011). The learning environment should provide structured guidance in the form of learning activities scaffolded with prompts and extra support that drives their thinking towards the systematic application of abilities (Bannert & Mengelkamp, 2013; Ge, 2013). The main components of the problem-based learning environment are worked examples, analogy, case studies, question prompts, etc. (Ge et al., 2005; Jonassen, 2011; Xun & Land, 2004).

Worked examples

Worked examples are effective in problem-solving for novices as they reduce the extraneous load and can concentrate on building cognitive schemas, which are used to solve similar problems in the future (Jonassen, 2011; McLaren & Isotani, 2011). Worked examples should break down complex solutions into smaller meaningful solution elements, and present multiple examples in multiple modalities for each type of problem. Providing analogous problems for students to compare with the problem to solve allows them to gain

more robust conceptual knowledge about problems (Xun & Land, 2004). In tutored problem-solving environments, worked and erroneous examples have proven to be effective learning activities. Studies have shown that alternating work examples and problem-solving have positive results in learning domain knowledge (Chen et al., 2019).

Cognitive tools

The research Wang et al. (2013) illustrated the possibilities of using visualization-based cognitive tools in learning environments to scaffold the entire problem-solving process. Tools such as concept maps, mind maps, and argument maps make thinking and learning in a problem context visible throughout the learning process. The framework by Kostousov and Kudryavtsev (2017) suggests using mind maps or concept maps in the problem analysis phase to expand the problem space. These tools will help in visualizing the main parts of a problem situation and actions in the problem-solving process as well as their relationships. In the design solution phase, a goal tree or decision tree can be used to expand the solution space because it can have multiple target states that are associated with different goals, and we do not know the best one. Argument mapping can help in choosing the best solution that has been identified by visualizing its benefits and limitations.

Issues and challenges of the existing studies

Prior work (Razavian et al., 2016; Tang et al., 2018) proposed a general model for the reflective design reasoning process, consisting of identifying the context and requirements, formulating and structuring design problems, creating solution options, and making design decisions. These techniques are referred to as design-reasoning techniques. In one study (Razavian et al., 2016), a list of reflective questions was used to improve design reasoning. The study involved groups of students in which the test groups were asked reflective questions by the lecturer, whereas for the control groups, the lecturer was only passively present to answer technical questions that the students might have had. It was concluded that active, externally induced reflection improved the quality of design reasoning. However, some participants found the questions difficult to use as there were many of them.

Another study Tang et al. (2018) proposed a simpler representation, the reminder card system, to test its reflective power on designers and its impact on design discourse. This study was conducted using a combination of students and professionals. The average reasoning techniques used by the teams in the test group were significantly higher than those used by the control group. This indicates that during design discourse, student teams in the test group are more cognizant of design reasoning. However, no significant differences were observed in the design outcomes in terms of the number of formulated problem statements, design options, and design decisions. Some participants mentioned that it would have been useful to guide how to use the cards during the design session;

perhaps, a structured approach or a method to prompt reasoning and reflection and to guide design discussions may be useful.

The studies do not directly examine whether the amount of explicit reasoning performed by designers leads to better results, that is, a better overall design.

Research goal

Our research goal was to design and develop a technology-enhanced learning environment for scaffolding design problem-solving in the context of software design problems in data structures courses for undergraduate engineering students. We propose the use of a design-based research methodology to design and develop a technology-enhanced learning environment for design problem-solving and evaluate its effectiveness in learning the skills and design outcome in terms of the quality of the overall design.

Design-based research

The design-based research (DBR) framework assists researchers in analyzing real-world problems, incorporating design principles with technical breakthroughs into solution development, testing, refining learning environments, and defining new design principles (Amiel & Reeves, 2008; Reeves, 2006). The DBR cycle comprises four phases: Problem Analysis, Solution Development, Evaluation, and Reflection. We conducted three DBR cycles, as discussed in detail in this section.

DBR cycle 1

The first DBR cycle aimed to examine a novice's ability to solve software design problems and identify teaching-learning techniques for ER skills. The work performed in each phase of the DBR is discussed in detail in this section.

Problem analysis

Study 1: We conducted a preliminary exploratory research study (Study 1) (Reddy et al., 2016) to assess the level of a novice in solving software design problems and to explore students' difficulty in applying ER skills. Exploratory research has been conducted to gain deeper insights into the problem and refine the problem definition (Shields & Rangarajan, 2013).

The research question (RQ) formulated for study 1 is:

RQ1. "What are the difficulties faced by the students while solving a software design problem with respect to ER skills?"

Based on a problem-solving literature survey, we designed a worksheet to solve design problems in the data structures course. In the worksheet, question prompts were given to enable the learner to systematically expand and reduce the problem-solution space. In the

problem analysis phase, prompts were designed to allow the learner to expand the problem space by listing all entities and actors involved. Later, the problem is decomposed into subproblems by identifying the data to be stored and the operations to be performed to solve the problem. Similarly, in the solution design phase, prompts were designed to generate multiple solutions by identifying alternative data structures and algorithms and selecting one based on the requirements of the problem.

Participants and research method: Forty second-year undergraduate computer engineering students participated in this study. This was a single-group study (Figure 8).

During the study, all students were given a software design problem to solve using a worksheet, as shown in Figure 9. The students individually solved the worksheet problem for almost two hours.

Data collection and analysis: The worksheet responses were evaluated and scored using the rubric designed to assess ER skills as shown in Table 1.

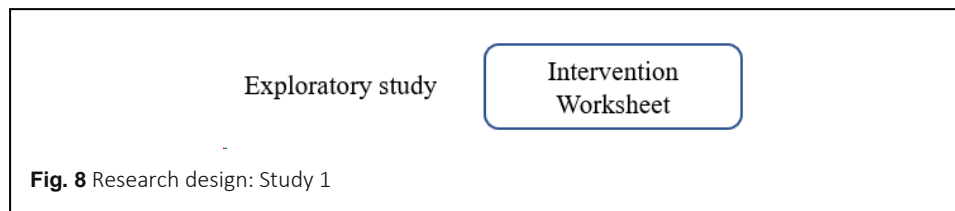


Fig. 8 Research design: Study 1

Problem: The local automobile retail shop sells parts for different car models. The shop owner wants to create an inventory control program that tracks the quantity of all the parts and creates a report of the parts that need to be ordered so there is minimal risk of items getting out of stock. Come up with multiple possible solutions by using appropriate data structures and operations for solving the above problem. Justify which solution is most efficient for the above-stated problem.

Phase 1. Understand and analyze the problem:	
1.1	List all the entities and actors:
1.2	List the data and operations performed from the perspective of each entity listed:
1.3	What is the requirement for the above problem?
1.4	Identify the data (listed in step 2), needed to solve the above requirement:
1.5	Identify the operations out of the list in step 3, needed to solve the problem:
Phase 2. Problem-solving:	
2.1	List the desirable Data Structures that can be used to solve the above problem:
2.2	For each Data Structure, give alternative solutions to solve requirements given in step 4 using data and operations identified in step 5 and 6 respectively:
2.3	Identify the efficient solution based on constraints and requirements in the problem. Justify?

Fig. 9 Worksheet activity of solving the software design problem

Table 1 Rubric to evaluate ER skills

Sub-skills/Score	3	2	1
Problem Analysis			
Understand the problem	All entities and interactions were identified correctly.	Missed a few entities and interactions.	Only a few entities and interactions identified.
Formulate the problem - sub-goal	Formulated the complete specification in terms of the data and operations to be performed.	Missed a few specifications in terms of the data and operations to be performed.	The specifications were incomplete.
Design Solution			
Generate solutions	All possible alternative solutions were generated.	Few possible alternative solutions are missed.	Generated only one solution.
Select and evaluate a solution	The solution is complete and correct. In the solution, the data structure is valid, and the operations are mapped to the correct algorithms. The selected solution is justified with clarity on how data is stored and how the algorithms are used in terms of constraints for the given problem.	The solution is partially correct OR the selected solution is justified with less clarity.	The solution is incomplete OR incorrect OR not explained clearly OR vaguely explained.

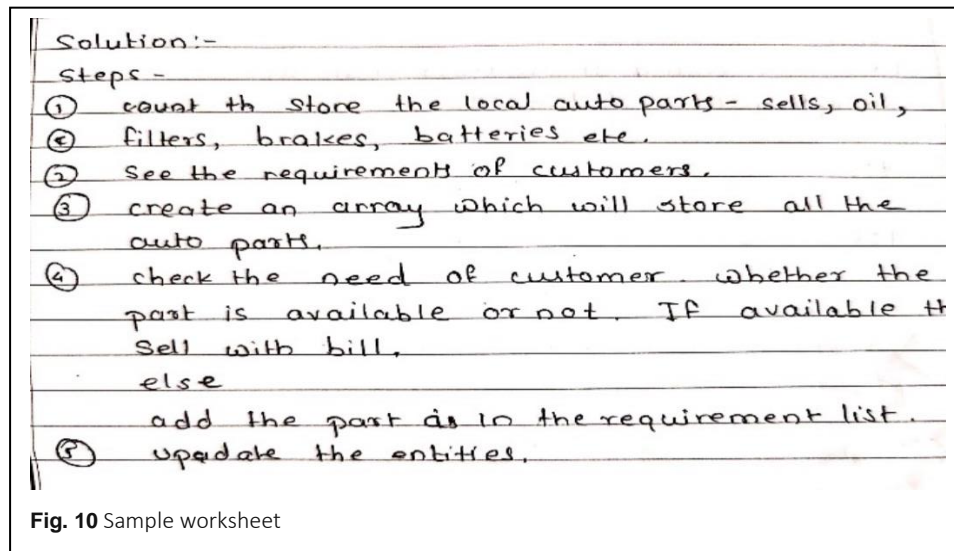
An expert educator with over ten years of teaching experience validated the rubric. The artifacts created during the intervention were independently examined by both instructors. Cohen's kappa (Vieira et al., 2010) was used to determine whether there was any inter-rater reliability between the two raters' scores. Agreement between the two raters was good ($\kappa = .634, p < .046$).

Results of study 1: Worksheets were evaluated using a rubric (Table 1). The average scores and corresponding standard deviation (SD) for each sub-skill are listed in Table 2.

The results show that the worksheet helped solve the problem systematically; however, students scored low in providing proper explanations and justification of the selected solution. To answer our research question, the difficulties faced by students in solving problems with respect to ER skills are discussed. As shown in the sample solved worksheet in Figure 10, the problem was solved superficially, and the students failed to visualize the problem space and decompose the problems into solvable subproblems. In the solution

Table 2 Scores of the worksheet

	Phase 1. Understanding the problem		Phase 2. Problem-solving	
	Expand	Reduce	Expand	Reduce
	Understand system (Max score=3)	Formulate the problem (Max score=3)	Generate solutions (Max score=3)	Select solution (Max score=3)
Average	1.5	1.55	1.71	0.47
SD	0.64	0.86	0.89	0.60



space, the solution was incomplete, and alternative design options were not considered or evaluated. This shows that students struggled to expand-reduce in problem-solving areas, which affected the quality of solution design and reasoning. It was evident that students needed to be scaffolded to apply ER skills, as they lacked the cognitive skills required to solve ill-structured software design problems.

The next step was to design a learning environment for teaching and learning ER skills. Willis (2009) stated that instructional designers must make appropriate decisions regarding guiding theories of learning (e.g., behaviorism, cognitive science, constructivism), general teaching and learning strategies (e.g., direct instruction, student-centered instructions), and pedagogies (e.g., anchored instruction, tutorial, problem-based learning). Based on recommendations in the literature (Bannert & Mengelkamp, 2013; Ge, 2013) for successful instructional methodologies for ill-structured problem-solving skills, the solution was designed as discussed in the next section.

Solution design- Fathom-Ver1

Fathom-Ver1's goal is to create a learning environment with pedagogical features that will scaffold novice learners to complete ER tasks. The design principle of Fathom-Ver1 is to provide structured guidance by designing learning activities to direct learners' thinking towards systematically applying ER skills.

The design features of Fathom-Ver1 are:

- i. Technology-enhanced learning environment to scaffold design problem-solving in data structures course.
- ii. Structured step-by-step guidance using learning activities scaffolded with question prompts, examples, and cognitive tools.


In Fathom-Ver1, the design problem posed was - “Design software for a bank to allow customers to check their account balance.” The learners were guided to solve the problem by following the following learning activities:

Understand the Problem: In this activity, learners were prompted to explore the problem by identifying entities and users in the system. The learners were prompted to draw the model of the system to show the components, their properties, and the interconnection between the components on the paper, as shown in Figure 11. The prompts were supported by examples and hints to enhance the understanding of the activity to be performed.


Formulate Problem: In this activity, the learners were prompted to write the sub-goals in terms of the operations to be performed by the software to achieve the stated goal, as shown in Figure 12. Hint buttons were provided to help learners with explanations and examples.

Learning Activity- Understand the problem
Draw the model of the system to show the components, it's properties and the interconnection between components.

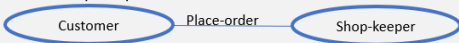
Components- Parts of whole system, e.g. actors, objects, as shown below.



Properties- attributes associated with each component, e.g., customer attributes are name, age, phone no., etc.



Interconnections- relationship between components, e.g. place_order is a relationship between customer and shop-keeper



Next

Hint on Components

Hint on Properties

Hint on Interconnection

Fig. 11 Fathom-Ver1- *Understand the Problem* activity

Learning Activity- Formulate Problem
Activity 1- Specify the goal to be achieved in the above stated problem

Goal

To provide online service to bank customers to check account balance

Activity 2- Refer to the model drawn in the previous step and formulate the problem by specifying how its components, properties and interconnections will be represented to achieve the stated goal.

The customer details and the account details will be stored in a data structure and the functions update_account and check_balance will be implemented.

Save

Hint on Activity 1

Goals are the requirements or functions to be achieved in the problem, e.g. design a software system to ..

Hint on Activity 2

Domain Knowledge

Fig. 12 Fathom-Ver1- *Formulate Problem* activity

Generate Solutions: This activity was designed to expand the solution space by generating alternative solutions. The learners were prompted to draw a mind map to list alternative design options for each subproblem, as shown in Figure 13.

Evaluate Solutions: This activity prompted them to evaluate alternative solutions based on the identified selection criteria. Learners were prompted to select an appropriate solution using a decision matrix. A decision matrix was used to allow learners to evaluate alternative solutions against the criteria and rank the solutions, as shown in Figure 14. Finally, the selected solution is justified.

Learning Activity- Formulate Problem

Activity 1- Draw mind map to list alternative design options for each sub-problem.
For example, for each data item, list all possible data structures and for each operation, list all possible algorithms, as shown in figure below-

Activity 2- List all possible alternative solutions

Solution 1

Solution 2

Solution 3

Hint on Activity 1

Mind map is a visualization tool that helps to explore all possible alternative design options.

Hint on Activity 2

Domain Knowledge

Fig. 13 Fathom-Ver1- *Generate Solutions* activity

Learning Activity- Evaluate

Activity 1- Identify the criteria to evaluate the alternative solutions

Criteria-

Activity 2- Evaluate solutions and rank using the decision matrix below-

Solutions	Execution time	complexity	Rank
Customer details stored in array and search using linear search	$O(n)$	Low	2
Customer details stored in linked list and search using linear search	$O(n)$	High	3
Customer details stored in array and search using binary search	$O(\log n)$	Med	1

Activity 3- Justify why the solution ranked 1 is optimal

Hint on Activity 1

Hint on Activity 2

Hint on Activity 3

Domain Knowledge

Fig. 14 Fathom-Ver1- *Evaluate Solutions* using decision matrix activity

Overall, in Fathom-Ver1, each activity is equipped with question prompts, hints, examples, and cognitive tools (drawing mind-maps, decision matrices, etc.) to systematically guide learners toward applying ER skills.

Evaluation: Study 2

We conducted Research Study 2 to evaluate the effectiveness of Fathom-Ver1. The research question (RQ) formulated for Study 2 is as follows:

RQ2. How effective is Fathom-Ver1 in applying ER skills while solving design problems in a data structures course?

Participants and research method: Study 2 (Reddy et al., 2017) was conducted to answer RQ2. The exploratory study (Shields & Rangarajan, 2013) was used for the following purposes:

- i. To investigate the effectiveness of prompts with additional scaffolds: explanation, example, and hint in performing the activity.
- ii. To understand how students are using the ER cognitive tools while performing the activity.
- iii. To assess the performance level of students’ ER skills with Fathom-Ver1.

Forty-nine undergraduate second-year computer engineering students participated in this study. Students interacted with Fathom-Ver1 for almost 2 hours individually, followed by a student perception survey and student interviews. This was a single-group exploratory study, as shown in Figure 15.

Data analysis and results: The data collected and analyzed are: student responses in Fathom, student perception survey, and interview data.

i. Scores of student’s responses: The students’ responses in Fathom-Ver1 were evaluated using the same rubric given in Table 1, and the average scores and standard deviation (SD) of ER skills are shown in Table 3.

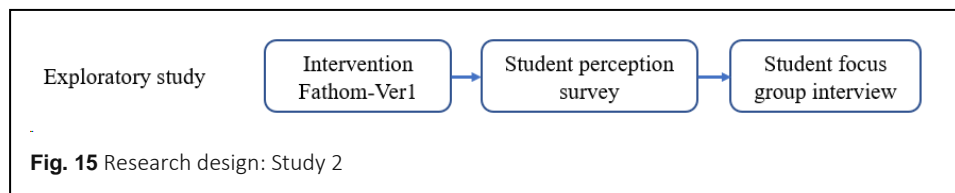


Table 3 ER scores of learners using Fathom-Ver1

	Phase 1. Understanding the problem		Phase 2. Design the solution	
	Expand	Reduce	Expand	Reduce
	Understand system (Max score=3)	Formulate the problem (Max score=3)	Generate solutions (Max score=3)	Select solution (Max score=3)
Average	2.10	1.52	1.9	1.50
SD	0.50	1.01	0.55	0.85

ii. *Student perception survey*: A student perception survey was conducted to reflect on how well the activities in the system helped in applying ER skills. The survey used a five-point Likert scale: 5(strongly agree), 4(agree), 3(neutral), 2(disagree), and 1(strongly disagree).

Q1. Identifying entities and interactions helped me to understand the working of the existing system from the perspectives of multiple stakeholders.

Q2. Understanding the whole system helped me to formulate the problem into subproblems.

Q3. The drawing of the mind map helped in generating multiple solutions.

Q4. The activity of identifying evaluation criteria parameters helped in analyzing solutions.

Q5. The decision matrix helped in evaluating and justifying the selected solution.

Two open-ended questions were asked to express their likes and dislikes regarding the system. The students' perception rating means and standard deviations (SD) are given in Table 4.

iii. *Focus group Interview*: The focus group interviews of four students were transcribed and evaluated to determine the students' perceptions of the activity's usefulness in learning ER skills and the problems they encountered while performing the activities.

During the interview, students perceived that step-by-step guidance, prompts, and examples helped to perform the activities. Some of the students' quotes on how the activities helped are given below:

1. *The overall idea of step-by-step guidance was good.*
2. *The activities helped us to figure out how many solutions are there, and see what are the advantages and disadvantages that helped us to deduct which is the best solution for the problem.*
3. *Hints and examples helped us a lot when we were not able to understand that the problem.*

Improvements are needed in the introductory and first phase—drawing the model, identifying and formulating sub-goals, as students found it difficult to understand the activity. Some of the students' quotes suggesting improvement are as follows:

1. *At the start, an introduction should be given to get clear idea of what to do, the content was not clear for a new user to understand. Prompts were not enough.*
2. *The activity in the design phase was nicely given and we moved faster, while struggling with first phase as we did not know what exactly is needed.*

Table 4 Student perception survey rating

Question no.	Q1	Q2	Q3	Q4	Q5
Mean	4.11	4.06	4.09	4.15	4.09
SD	0.52	0.6	0.65	0.59	0.69

Reflection from DBR cycle 1

RQ2 was answered based on the ER skill scores of the responses of the problem solved in Fathom-ver1, student perception survey, and interview data analysis. The analysis of our results showed that the average scores of the students were better in Study 2 than in Study 1, which shows that the systematic guidance in Fathom-Ver1 in the form of prompts, hints, examples, and cognitive tools helped in applying ER skills compared to the worksheet. The mean scores of the student perception survey showed that most of the students agreed that Fathom's learning activities and tools helped achieve the desired learning outcomes. The students' responses to the open-ended questions showed that the activities helped in developing analysis and designing skills, building multiple solutions, and providing knowledge on how to solve problems. However, some students found the entire process to be time-consuming. In the interview, students said that they had difficulty understanding the activities to be performed, especially in the learning activities in the first phase—problem analysis.

DBR cycle 2

From the findings and reflections from DBR cycle 1, we identified problems in the design of the first version of Fathom, a potential reason for the problem, and redesign steps (Table 5).

In DBR cycle 2, the pedagogical features of the activities were improved to engage learners at both cognitive and metacognitive levels. Prior studies have shown that metacognitive scaffolds, including prompts, guiding questions, feedback, and self-reflection, significantly enhance students' problem-solving performance and awareness and regulation of their cognitive processes (An & Cao, 2014; Bannert & Mengelkamp, 2013; Ge, 2013; Geiwitz, 1994).

Research shows that the following components are necessary to engage learners in thinking at a metacognitive level in a learning environment:

- i. Feedback is a metacognitive activity that helps learners improve their learning (Narciss, 2013). Feedback is important for improving learning in various instructional contexts,

Table 5 Problems seen in DBR cycle 1, a potential reason for the problem, and redesign step

Sr. No.	Problems seen in DBR cycle 1	The potential reason for the problem	Redesign
1.	Difficulty in understanding the activity to be performed.	Lack of help and feedback from the learning environment.	The prompts have to be enhanced with solved examples and feedback generation based on student responses.
2.	Difficulty in drawing the diagrams to expand problem-solution space.	The drawing tools were not part of the learning environment.	The drawing tools will be integrated into the learning environment.

including online learning environments. Feedback on learners' responses informs the learner about their actual state of learning to regulate the further process of learning in the direction of learning standards (Narciss, 2013). In tutoring, formative feedback helps learners become aware of any gaps that exist between their desired and current state of knowledge or competencies. Studies have shown that feedback from online tutors enables students to acquire problem-solving skills in domains such as geometry, algebra, and computer programming languages (Cassel & Victor, 2015).

ii. In-action reflection is a metacognitive activity in which a learner reviews her cognitive process or experience. This review process allows learners to identify deficiencies in their performance and the specific skills they need to improve on (Rivera-Gutierrez et al., 2016). Learners can review their performance through a self-assessment on a performance scale. The process of self-assessment is not an easy task, because learners tend to overestimate or underestimate their performance. The self-assessment process can be improved by providing students with scaffolds to assess the scale accurately. In one such study, the authors (Rivera-Gutierrez et al., 2016) proposed the use of in-action reflections to develop interpersonal skills, such as empathy towards patients and medical practitioners. The students interacted with virtual agents that played the role of a patient. A study conducted with third-year dental students showed that the students were relatively accurate in their self-assessment of how empathetic they were to the virtual patient and were significantly more empathetic to the patient after their in-action self-assessment.

Solution Design- Fathom-Ver2

The design principles of Fathom-Ver 2 are the same as Fathom-Ver1, with one new principle added: to enable learners to apply metacognitive skills during problem-solving.

The design features of Fathom-Ver2 are:

- i. To integrate drawing tools into the system.
- ii. To provide worked examples for all steps of problem-solving.
- iii. Scaffold learners to reflect on their problem-solving skills using self-evaluation activities.

The learning activities were enhanced by doing the following revisions in Fathom-Ver2:

1. The learners are scaffolded to apply ER skills in the process of solving a library management problem in a college. The problem posed is- "Design a software system for a college library that will allow students and teachers to check the availability of the book in the library."
2. Drawing tools are integrated into *Understand the Problem* and *Generate Solutions* activities. In the *Understand the Problem* activity, the learners are prompted to draw a diagram to represent all entities and interactions to aid in visualizing the problem as a whole,

as shown in Figure 16. In the *Generate Solutions* activity, the learners are prompted to visually represent solution space using mind-map, as shown in Figure 17.

3. Worked example illustrating the problem-solving process: In every activity, help is provided, which learners may use to obtain a detailed explanation of the activity illustrated with an example. For example, on the clicking view demo in Figure 16, the video is played to illustrate the process of drawing a diagram for the shop-inventory problem, as shown in Figure 18.

Problem posed
The college library maintains books on various subjects taught in various departments like arts, science, and commerce. The library staff is involved in issuing books to students and teachers. The students need to return the book in 15 days while teachers can return it in three months. Librarian decides to provide online services to students and teachers to search for availability of the books in library. Your task is to design software to solve the librarian using appropriate data structures and algorithms.

Problem solving phases
 Problem Analysis
 Understand Problem
 Formulate Problem
 Design Solution
 Generate Solutions
 Evaluate and Select

Workspace for the learner to draw as prompted

Diagrammatically represent the library system by drawing entities involved in the system and interactions among them. Entities are people and objects involved in the system and interactions are relationship among the entities. For example, in a shopkeeper problem, customer and item are the entities and interaction is buys (customer-> buys -> items)

Self-evaluation- Rate your response on the scale shown below-
 ● **High-** Identified all possible entities/actions and interactions among them.
 ● **Medium-** Missed to identify few entities/actions and interactions among them.
 ● **Low-** Identified very few (one or two) entities/actions and interactions among them.

Fig. 16 Fathom-Ver2- *Understand the Problem* activity

Problem posed
The college library maintains books on various subjects taught in various departments like arts, science, and commerce. The library staff is involved in issuing books to students and teachers. The students need to return the book in 15 days while teachers can return it in three months. Librarian decides to provide online services to students and teachers to search for availability of the books in the library. Your task is to design software to solve the librarian using appropriate data structures and algorithms.

Problem solving phases
 Problem Analysis
 Understand Problem
 Formulate Problem
 Design Solution
 Generate Solutions
 Evaluate and Select

Workspace for the learner to draw mindmap

The partial mind map for your software design problem is shown below. Complete the map by identifying various data structures for each data item and various algorithms (linear search, binary search, bubble sort, insert, etc) for each operation.

Self-evaluation- Rate your response on the scale shown below-
 ● **High-** Identified all possible data structures and algorithms for each data item and operation respectively.
 ● **Medium-** Missed to identify few data structures and algorithms .
 ● **Low-** Identified very few (one or two) data structures and algorithms .

Fig. 17 Fathom-Ver2- *Generate Solutions* activity

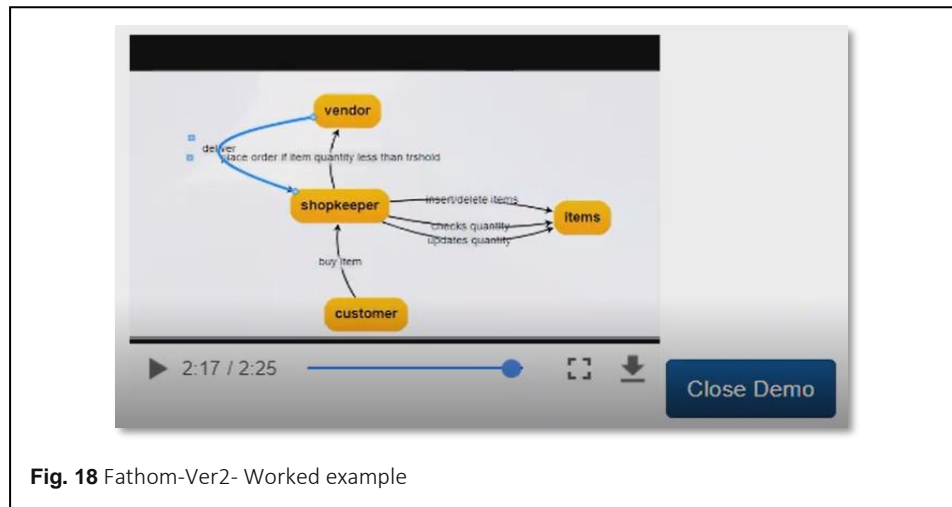


Fig. 18 Fathom-Ver2- Worked example

4. Learner's self-evaluation: After each activity, learners are directed to self-evaluate and reflect on their responses. Immediately after the learner saves their response, the system prompts them to self-evaluate their responses based on the rubric, as shown in Figure 16. Subsequently, feedback is generated, which states the corrective measures to be implemented to improve the response.

Evaluation: Study 3

We conducted Study 3 (Reddy et al., 2018) to evaluate the design features of Fathom-Ver2. The research question (RQ) formulated for Study 3 is:

RQ3. "How effective is Fathom-Ver2 in learning ER skills at both cognitive and metacognitive levels?"

Participants and research method: Study 3 was conducted with fifty-two second-year undergraduate computer engineering students. The research design used was a one-group pretest-posttest study (Cohen et al., 2002). This study aimed to investigate the effectiveness of Fathom-Ver2 in learning ER skills by comparing performance before and after the intervention. The research design is illustrated in Figure 19.

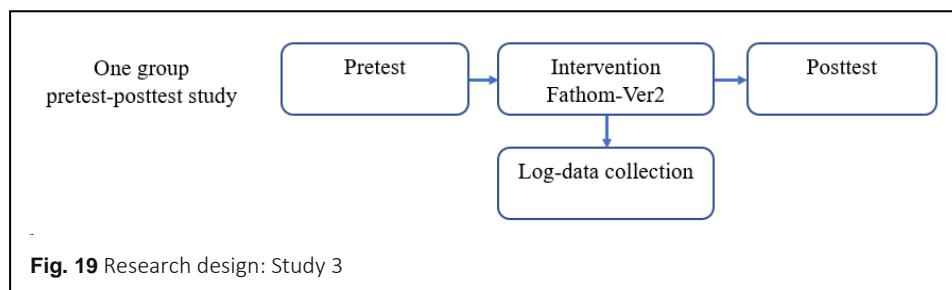


Fig. 19 Research design: Study 3

In the pretest, the students were given a worksheet to solve a shop inventory problem: “Design a software system for the supermarket to display items below threshold,” and the students were given one hour to solve the pretest problem individually by following the steps given below:

1. Write the broad goal to be achieved.
2. Write the sub-goals to be achieved (insert/delete/search).
3. Design a solution using appropriate data structure and algorithms.
4. Justify why the selected data structure is appropriate for the given problem.

Immediately after the pretest, the learners interacted with Fathom-Ver2 for 2 hours. The log data were collected from Fathom, in which the following data were recorded for each button click: <learner_id, timestamp _button, student_response>. After the training, the students were told to solve a new problem in Fathom-Ver2, in which the scaffolds were withdrawn.

Data collected and analyzed: To answer RQ3, we analyzed and compared the pretest-activity-posttest scores to measure the learning of ER skills. The log data of Fathom were analyzed to measure the effectiveness of the self-evaluation activity.

i. *Pretest-activity-posttest scores:* The student artifacts generated during the pretest, posttest, and intervention were evaluated using a rubric (Table 1). Of the fifty-two students, only forty-seven completed the training with Fathom and only seventeen completed the posttest. The mean scores with standard deviation (SD) of the pretest, activity, posttest, and comparison between the pretest-activity and pretest-posttest using the t-test (Derrick et al., 2017) are shown in Tables 6 and 7.

Table 6 Pretest and activity scores

N=47	Phase 1. Understand the Problem		Phase 2. Design the solution	
	Expand	Reduce	Expand	Reduce
	Understand system (Max score=3)	Formulate the problem (Max score=3)	Generate solutions (Max score=3)	Select solution (Max score=3)
Pretest Average (SD)	0.00	1.38 (0.60)	1.00 (0.85)	0.93 (0.70)
Activity Average (SD)	2.48 (0.47)	2.09 (0.52)	1.83 (0.51)	1.92 (0.62)
P value (T-test)	0.00	0.00	0.00	0.00

Table 7 Pretest and posttest scores

N=17	Phase 1. Understand the Problem		Phase 2. Design the solution	
	Expand	Reduce	Expand	Reduce
	Understand system (Max score=3)	Formulate the problem (Max score=3)	Generate solutions (Max score=3)	Select solution (Max score=3)
Pretest Average (SD)	0.00	1.5 (0.68)	1.2 (0.98)	0.7 (0.58)
Posttest Average (SD)	2.41 (0.62)	2.4 (0.65)	2.11 (0.34)	2.3 (0.71)
P value (T-test)	0.00	0.00	0.00	0.00

Student-ID	Timestamp	Clicked_button	Student_responses
114A1001	October 14, 2017, 12:03 pm	formulatebutton	
114A1001	October 14, 2017, 12:03 pm	SaveButton_goal	to maintain records of the books and check for availability of books
114A1001	October 14, 2017, 12:03 pm	SaveButton_subgoal	check for books
114A1001	October 14, 2017, 12:04 pm	Generate solution	
114A1001	October 14, 2017, 12:14 pm	SaveButton_GenSol	Array+ linear search
114A1001	October 14, 2017, 12:14 pm	SaveButton_Self-eval	
114A1001	October 14, 2017, 12:16 pm	SaveButton_GenSol	Array+ linear search, Linked-list
114A1001	October 14, 2017, 12:18 pm	hint_generatesolution	
114A1001	October 14, 2017, 12:19 pm	hint_generatesolution	

Fig. 20 Raw log data collected in Fathom

ii. *Log data analysis*: The sample log data are presented in Figure 20.

To simplify the interpretation, we parsed the log data into a sequence of activities. For example, all the edits made in the *understanding_problem* activity, such as drawing the diagrams and saving in the first attempt as UP, edits in *formulation-problem* activity as FP, *generate_solution* activity as GS, and *evaluate_solution* activity as EV. The resources accessed (hints, notes, examples, etc.) were coded as RA. The subsequent save, in which the responses are modified after the self-evaluation activity, is coded as REDO. Examples of the resultant sequences of a student are as follows.

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

The percentage of students with REDO codes in the resultant sequences was calculated.

Results and reflection

RQ3 was answered using the ER scores of student responses in the pretest-activity-posttest and log data analysis. The scores showed significant improvement in the quality of problem formulation and solution from pretest to activity and from pretest to posttest, which shows that students were able to expand and reduce effectively while solving the problem in Fathom and transfer those skills to the posttest problem. The drawing tools integrated into the learning environment in the *Understand the Problem* activity provided the affordance to save, edit diagrams, and navigate back and forth during problem formulation. This activity helped create and simulate a mental model of the system as a whole. In the *Generate Solutions* activity, the integration of the drawing tool for drawing a mind map representing all the possible alternative design options for each subproblem helped visualize the solution space and further identify the selection criteria for evaluating solutions. However, the log data analysis revealed that only 20% of the students revised their responses after self-evaluation, indicating that students either overestimated or underestimated their responses and lacked the ability to self-evaluate.

DBR cycle 3

From the findings of DBR cycle 2, it was evident that the students were able to perform the activities, but were weak in the self-evaluation of ER skills. The aim of DBR cycle 3 was to redesign metacognitive activities in Fathom. The design features are the same as the previous version with one change: to scaffold learners to apply metacognitive skills using system-generated feedback and peer review instead of self-evaluation activity.

Solution design of Fathom-Ver3

The literature suggests that expert feedback, peer review, and active learning-based collaboration (Ge, 2013; Narciss, 2013), are effective in improving learners' metacognitive skills during problem-solving. Peer review is a metacognitive activity that enables learners to see alternative perspectives from peers' responses and helps them notice things that they might not have thought about. By compelling learners to examine their thinking after reviewing their peers' responses, learners engage in metacognitive activities and self-regulation during problem-solving (Ge, 2013).

In Fathom-Ver3, the self-evaluation activity is replaced by system-generated feedback and peer review. The new features added are as follows:

System-generated feedback: Automated feedback is generated by evaluating the gaps and providing an elaborate explanation of the gaps and an action plan to monitor and revise the skill. Positive or corrective feedback is generated at the end of each activity after the learner saves their response. Figure 21 shows the feedback generated at the end of the step-understand problem. The feedback was generated by the system by semantically comparing the learners' responses to the experts' solutions.

Peer review: This is a new feature that allows collaboration with peers during each activity, as shown in Figure 22. To encourage learners to actively evaluate their peers' responses,

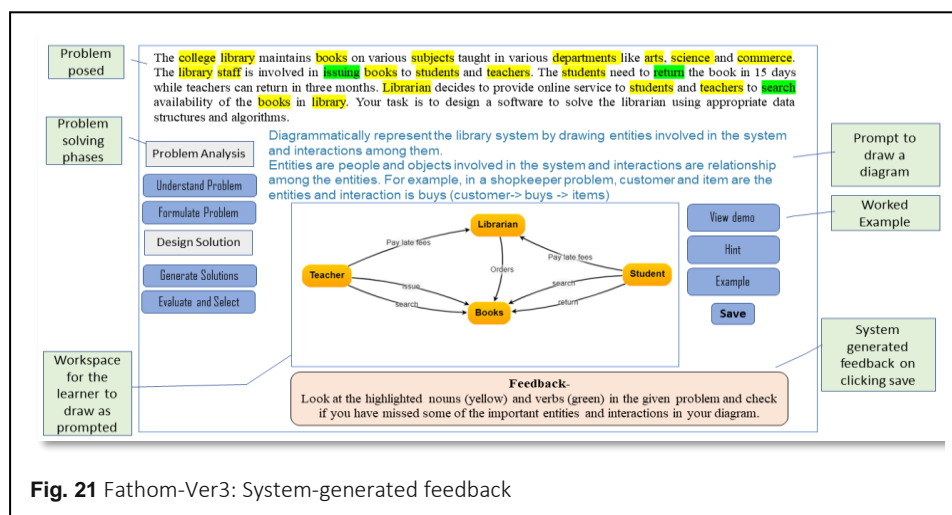


Fig. 21 Fathom-Ver3: System-generated feedback

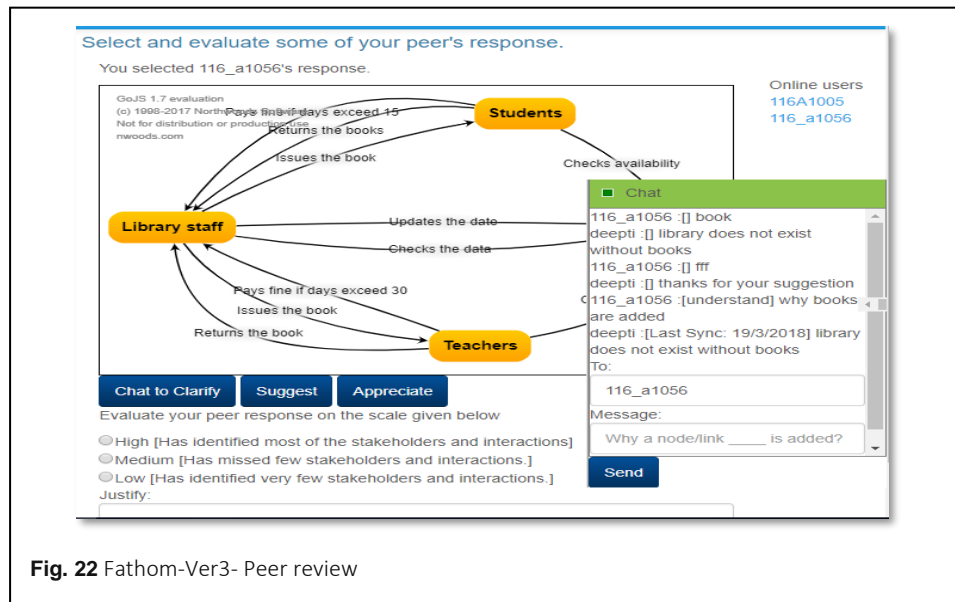


Fig. 22 Fathom-Ver3- Peer review

effective collaborative learning methods (Soller et al., 1999) were used to allow them to actively demand explanations and justifications from their peers. The aim is to allow learners to see how their peers have performed the activity, evaluate their responses against the given scale of high, medium, and low, and justify the evaluation. This helps the learner to see others' perspectives on doing, thinking, and evaluating the gap.

Evaluation: Study 4

This study aimed to investigate the effectiveness of cognitive and metacognitive scaffolds of Fathom-Ver3 in learning ER skills. The research question (RQ) formulated for Study 4 is:

RQ4. How effective is Fathom-Ver3 in learning ER skills at both the cognitive and metacognitive levels?

Participants and research method: The study involved fifty undergraduate second-year computer engineering students. The research design used was a one-group pretest-posttest study (Cohen et al., 2002) to investigate the effectiveness of Fathom-Ver3 in learning ER skills by comparing ER skills before and after the intervention. The research design of Study 4 is shown in Figure 23.

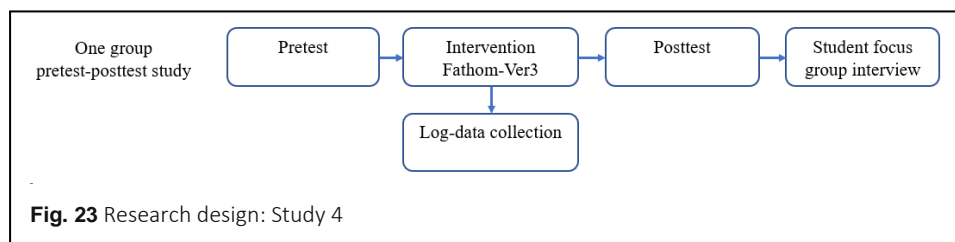


Fig. 23 Research design: Study 4

During the pretest, the participants were given a shop-inventory problem to be solved on a worksheet for 30 min, followed by an intervention in which students interacted with Fathom-Ver3 for two hours. After the intervention, the participants were given a bank problem to solve on a worksheet with no scaffolds.

Data analysis: The data collected were the responses generated during the pretest, intervention, and posttest; interview data of four students; and log data in the form of user clicks.

- i. Pretest and posttest scores: A rubric (Table 1) was used to evaluate the student artifacts created during the pretest and posttest.
- ii. Interview data: To identify meaningful units of analysis, content analysis of interview transcriptions was performed in terms of how the students perceived the activities to be effective in learning ER skills and the difference in problem-solving from pretest to posttest.
- iii. Log data analysis: As discussed in Study 3, in a similar way, log data analysis was performed with additional code added for peer review as PR. The percentage of students with REDO and PR codes in their resultant sequences was calculated.

Results: The data analyzed were pretest and posttest scores, interview data, and log data.

- i. *Pretest and posttest scores:* The pretest and posttest scores of the fifty students are shown in Table 8. A t-test (Derrick et al., 2017) was used to compare the average pretest-intervention-posttest scores.
- ii. *Interview data:* The purpose of the interview data analysis was to determine how students perceived that the Fathom-Ver3 features helped them acquire ER skills. The following are students' perceptions of their overall learning of ER skills during Fathom-Ver3 training:
 1. In the *Understand the Problem* task, drawing the diagram helps to envision all of the entities present and how they are linked to one another, as well as establish essential requirements that the software must meet. The following is a quote from one of the students: "*it helped to visualize the problem, and it helped me to see that all the other entities are linked so that it simplified the problem.*"

Table 8 Pretest and posttest scores

N=50	Phase 1. Understand the Problem		Phase 2. Design the solution	
	Expand Understand system (Max score=3)	Reduce Formulate the problem (Max score=3)	Expand Generate solutions (Max score=3)	Reduce Select solution (Max score=3)
Pretest Average (SD)	0.00	1.34 (0.59)	1 (0.19)	1.35 (0.58)
Posttest Average (SD)	1.63 (0.79)	1.69 (0.68)	2.09 (1.22)	1.99 (0.44)
P value (T-test)	0.00	0.01	0.00	0.00

2. The *Understand the Problem* activity assisted in dividing the problem into subproblems. The corresponding quote: *“In the pretest, I could not think of alternative solutions because did not know how to break the problem and think of each subproblem as a separate unit which has different solution options.”*
 3. Examples were useful in determining which way to think.
 4. The feedback helped me understand the gap and was simple to comprehend.
- iii. *Log data analysis:* Log data analysis showed that 80% of the students evaluated their responses with the help of feedback and taking control action plans to improve. However, very few students engaged in higher-level activities, such as peer evaluation and chat facilities, to seek clarification or justification from peers.

Reflection

RQ4 is answered using ER scores of student responses in the pretest-posttest, interview, and log data analysis. The pretest-posttest scores showed significant improvement in the problem formulation and solution from pretest to posttest, which shows that students learned ER skills using Fathom-Ver3. Thus, the pedagogical features of Fathom-Ver3 were effective in teaching and learning ER skills in software design problem-solving. The improvement was seen at both cognitive and metacognitive levels, as students were able to reflect on their responses after reading the system-generated feedback, and learners were improving their skills.

Discussion

We iterated through three DBR cycles to design and develop an intervention, Fathom, with the learning objective of teaching-learning ER skills in the context of solving software design problems. The aim of the first DBR cycle was to conduct exploratory studies to understand the level of scaffolding needed for novices to learn ER skills. Studies 1 and 2 showed that only question prompts with an explanation of new terms and hints were not effective in triggering learners to apply the necessary cognitive skills to expand-reduce the problem-solution space. The reason for this might be that the students did not have prior experience in solving complex problems. The aim of DBR cycle 2 was to redesign Fathom with improved scaffolding to direct learners' thinking towards applying ER skills and reflecting on them. Fathom-Ver2 was equipped with prompts, solved examples, hints, ER cognitive tools (drawing tools, decision matrix, etc.), and self-reflection activities. Study 3 evaluated the effectiveness of Fathom-Ver2 and showed that scaffolding was effective in applying ER skills; however, self-reflection was ineffective. The students were either overestimating or underestimating their skills; thus, very few students were involved in

improving their responses. The third DBR cycle aimed to revise metacognitive activities for the effective monitoring and control of ER skills. In Fathom-Ver3, the self-reflection activity was replaced with system-generated feedback and peer review. Study 4 showed that students not only applied ER skills effectively but also actively improved their responses after reading the feedback.

We discuss the following local learning theories based on our research studies assessing the learning of ER skills:

Various cognitive tools (entity-interaction, mind-maps, decision matrix) aid novices in applying ER skills

The drawing of the entity-interaction diagram provided the affordance to systematically expand and visualize the entire problem/solution space. This helped to structure the problem into subproblems and consider the evaluation criteria to select the optimal solution. Our findings support previous studies that have also shown that various visualization techniques, such as drawing various entities and interactions among them, help in understanding the problem as a whole and improving the quantity and quality of the problem statements generated (Eden & Ackermann, 2001; Norese, 1995). This helps novices create and simulate mental models at various levels of abstraction. Eventually, from these mental models, the subproblems are decomposed (Adelson & Soloway, 1985; Guindon, 1990; Tang et al., 2008, 2010). The drawing of the problem and solution space helps simplify the understanding of the complex problem and identify the requirements to be addressed in the solution.

Drawing a mind map is effective in expanding the solution space. It enables the learner to represent subproblems (data items and operations) of the solution design and branch out each subproblem into various alternative design options (data structures and algorithms). Next, multiple solutions were generated by selecting valid combinations of design options for each subproblem.

The process of explicitly identifying selection criteria and using a decision matrix to evaluate alternative solutions against the selection criteria, ranking them, and selecting a solution helped to justify how the selected alternative is better than other alternatives. This allows the learner to reflect on and assess the effectiveness of the solution in achieving the stated requirements. Our findings support the claims made in previous research that stated that cognitive tools such as entity-interaction diagrams, mind maps, and decision tables helped to expand and reduce the problem and solution space (Ackoff, 1979; Adelson & Soloway, 1985; Guindon, 1990; Mangano et al., 2014; Tang et al., 2008, 2010).

Feedback is necessary for novices to regulate the learning of ER skills

Feedback assists students in identifying performance gaps, which is essential for monitoring and improving their performance. Feedback is the most effective when delivered immediately after the exercise, addressing both positive and negative elements

to motivate students by praising their efforts and directing their focus to areas where they can improve. Similar to prior studies (Bannert & Mengelkamp, 2013; Ge, 2013; Narciss, 2013), our findings also support the idea that feedback should be expanded with information to guide learners to see clues, hints, examples, peers' responses, and domain-related comments to improve.

Peer evaluation, self-evaluation, and active collaboration are advanced features to enhance the acquisition of ER skills

This feature was created to help learners apply metacognitive thinking skills by comparing their performance to that of their peers, evaluating the gap, using the chat feature to seek or provide an explanation, challenging peers' responses, or appreciating the work of others. However, during the training period, the majority of the students did not actively use this capability, implying that during training, learners focused their efforts on comprehension and learning abilities at a cognitive level, and external input was necessary to scaffold their metacognitive skills. Novices struggle to self-regulate metacognitive activities by utilizing self-evaluation or peer-evaluation features. Self-evaluation activity is not effective in regulating metacognitive behaviors, as students tend to either overestimate or underestimate their performance and are unable to take appropriate action plans to improve their skills. This does not support the findings of previous studies (Ge, 2013; Rivera-Gutierrez et al., 2016), which showed that self-evaluation activities and peer review helped students improve their problem-solving skills in the learning environment.

Fathom was effective in the teaching-learning of ER skills

Studies have shown that after training novices in Fathom, they were able to learn ER skills. The quality of the problem formulation and solution design improved from pretest to posttest, which implies that the scaffolding provided in Fathom was effective in the teaching-learning of ER skills. The pretest responses show that novices tend to converge early on a single solution without spending much time on problem exploration and alternative solution generation. This leads to incomplete problem formulation, which affects the design quality. The scaffolding provided in Fathom directed the learners to think explicitly about exploring the problem space and solution space before reducing the problem formulation and solution design. The cognitive tools provided in the system aided in effectively visualizing the problem and solution space. Metacognitive support allowed learners to reflect on their skills and improve them accordingly. Thus, we support previous findings (Bannert & Mengelkamp, 2013; Jonassen, 2011; Xun & Land, 2004) which state that for novices, explicit training is needed on using various cognitive and metacognitive tools using various scaffolding mechanisms such as prompts, drawing tools, solved examples, hints, and expert feedback.

Limitations

The limitations of this study are discussed in this section from the learners, instructors, and domain and research perspectives.

1. **Learner characteristics:** Our research findings are confined to computer engineering students who have completed a data structures and algorithm course, and are fluent in English and computer use.
2. **Topic and domain:** This research is carried out as part of a data structures course for computer engineering students that focuses on software design problem-solving. This has not been tested for design challenges in other related courses or in engineering fields.
3. **Near vs. far transfer:** Experiments were conducted to determine whether ER abilities could be transferred over the same course. Because the trials were not longitudinal, we did not test for far transfers.

Conclusion

In this paper, we discussed three DBR cycles following the design and development of the intervention Fathom used for the teaching-learning of ER skills in the context of solving software design problems. The overall contributions are the characterization and importance of ER skills in solving ill-structured software design problems, identification of ER cognitive tools to expand-reduce problem-solution space, and cognitive and metacognitive scaffolds. In future work, we aim to test and validate another type of design problem and conduct longitudinal studies over one year on the same set of students to test the transfer of ER skills.

Abbreviations

TELE: Technology-enhanced learning environment; ER skills: Expand-reduce skills; DBR: Design-based research; AHP: Analytic Hierarchy Process; RQ: Research question; SD: Standard deviation.

Acknowledgements

The authors would like to acknowledge Prajish Prasad and Kavya Also for providing suggestions in the design and development of the system, and Shitanshu and Anurag for helping in the design of the research studies.

Authors' contributions

DRP has designed and developed the learning environment, conducted research studies, collected and analyzed data, and writing the manuscript. SI and SM supervised the whole process of design and development of learning environments, literature survey, design of the research studies, and majorly in the review of the manuscript. All the authors read and approved the final manuscript.

Authors' information

Deepti Reddy Patil, Ph.D is an Associate Professor in the Department of Computer Engineering, Mukesh Patel School of Technology Management & Engineering, NMIMS University, Mumbai, 400056. Sridhar Iyer, Ph.D is a Professor in Inter-disciplinary Program in Educational Technology, Indian Institute of Technology Bombay, Powai, Mumbai 400076. Sasikumar M., Ph.D, is an Executive Director, at CDAC Mumbai, India.

Funding

Not applicable.

Availability of data and materials

Not applicable.

Declarations**Competing interests**

The author declares that he has no competing interests.

Author details

¹ NMIMS University, India

² Indian Institute of Technology Bombay, India

³ CDAC Mumbai, India

Received: 17 June 2023 Accepted: 2 April 2024

Published online: 1 January 2025 (Online First: 7 May 2024)

References

- ABET Engineering Accreditation Commission. (2013). *Criteria for accrediting engineering programs: Effective for reviews during the 2014-2015 accreditation cycle*. ABET.
- Ackoff, R. L. (1979). The future of operational research is past. *Journal of the Operational Research Society*, 30(2), 93–104. <https://doi.org/10.1057/jors.1979.22>
- Adelson, B., & Soloway, E. (1985). The role of domain experience in software design. *IEEE Transactions on Software Engineering*, 11, 1351–1360. <https://doi.org/10.1109/TSE.1985.231883>
- Amiel, T., & Reeves, T. C. (2008). Design-based research and educational technology: Rethinking technology and the research agenda. *Journal of Educational Technology & Society*, 11(4), 29–40.
- An, Y. J., & Cao, L. (2014). Examining the effects of metacognitive scaffolding on students' design problem solving and metacognitive skills in an online environment. *Journal of Online Learning and Teaching*, 10(4), 552–568.
- Bannert, M., & Mengelkamp, C. (2013). Scaffolding hypermedia learning through metacognitive prompts. In R. Azevedo & V. Aleven (Eds.), *International handbook of metacognition and learning technologies* (pp. 171–186). Springer, New York, NY. https://doi.org/10.1007/978-1-4419-5546-3_12
- Basadur, M., Pringle, P., Speranzini, G., & Bacot, M. (2000). Collaborative problem solving through creativity in problem definition: Expanding the pie. *Creativity and Innovation Management*, 9(1), 54–76. <https://doi.org/10.1111/1467-8691.00157>
- Cassel, S., & Victor, B. (2015). A structured approach to training open-ended problem-solving. In *Proceedings of 2015 IEEE Frontiers in Education Conference* (pp. 1–4). IEEE. <https://doi.org/10.1109/FIE.2015.7344088>
- Chen, X., Mitrovic, A., & Mathews, M. (2019). Learning from worked examples, erroneous examples, and problem solving: Toward adaptive selection of learning activities. *IEEE Transactions on Learning Technologies*, 13(1), 135–149. <https://doi.org/10.1109/TLT.2019.2896080>
- Cohen, L., Manion, L., & Morrison, K. (2002). *Research methods in education*. Routledge.
- Cooperrider, B. (2008). The importance of divergent thinking in engineering design. In *Proceedings of the 2008 American Society for Engineering Education Pacific Southwest Annual Conference* (pp. 27–28). American Society for Engineering Education.
- Dennis, A. R., Aronson, J. E., Heninger, W. G., & Walker, E. D. (1999). Structuring time and task in electronic brainstorming. *MIS Quarterly*, 95–108.
- Derrick, B., Broad, A., Toher, D., & White, P. (2017). The impact of an extreme observation in a paired samples design. *Advances in Methodology and Statistics*, 14(2), 1–17.
- Dym, C. L., Agogino, A. M., Eris, O., Frey, D. D., & Leifer, L. J. (2005). Engineering design thinking, teaching, and learning. *Journal of Engineering Education*, 94(1), 103–120. <https://doi.org/10.1002/j.2168-9830.2005.tb00832.x>
- Eden, C., & Ackermann, F. (2001). SODA—the principles. In J. Rosenhead & J. Mingers (Eds.), *Rational analysis for a problematic world revisited* (pp. 21–41). John Wiley & Sons Inc.
- Ellspermann, S. J., Evans, G. W., & Basadur, M. (2007). The impact of training on the formulation of ill-structured problems. *Omega*, 35(2), 221–236. <https://doi.org/10.1016/j.omega.2005.05.005>
- Ge, X. (2013). Designing learning technologies to support self-regulation during ill-structured problem-solving processes. In R. Azevedo & V. Aleven (Eds.), *International handbook of metacognition and learning technologies* (pp. 213–228). Springer, New York, NY. https://doi.org/10.1007/978-1-4419-5546-3_15
- Ge, X., Chen, C. H., & Davis, K. A. (2005). Scaffolding novice instructional designers' problem-solving processes using question prompts in a web-based learning environment. *Journal of Educational Computing Research*, 33(2), 219–248. <https://doi.org/10.2190/5F6J-HHVF-2U2B-8T3G>
- Geiwitz, J. (1994). *Training metacognitive skills for problem solving*. Advanced Scientific Concepts.
- Goel, V., & Pirolli, P. (1992). The structure of design problem spaces. *Cognitive Science*, 16(3), 395–429. [https://doi.org/10.1016/0364-0213\(92\)90038-V](https://doi.org/10.1016/0364-0213(92)90038-V)

- Guindon, R. (1990). Knowledge exploited by experts during software system design. *International Journal of Man-Machine Studies*, 33(3), 279–304. [https://doi.org/10.1016/S0020-7373\(05\)80120-8](https://doi.org/10.1016/S0020-7373(05)80120-8)
- Howard, T. J., Culley, S. J., & Dekoninck, E. (2008). Describing the creative design process by the integration of engineering design and cognitive psychology literature. *Design Studies*, 29(2), 160–180. <https://doi.org/10.1016/j.destud.2008.01.001>
- Jonassen, D. (2011). Supporting problem-solving in PBL. *Interdisciplinary Journal of Problem-Based Learning*, 5(2), 8.
- 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. <https://doi.org/10.1002/i.2168-9830.2006.tb00885.x>
- Kostousov, S., & Kudryavtsev, D. (2017). *Towards a framework of using knowledge tools for teaching by solving problems in technology-enhanced learning environment*. International Association for Development of the Information Society.
- Lian, H. (2012). *Network design problems, formulations and solutions*. University of Texas at Dallas.
- Liu, Z., & Schonwetter, D. J. (2004). Teaching creativity in engineering. *International Journal of Engineering Education*, 20(5), 801–808.
- Mangano, N., LaToza, T. D., Petre, M., & van der Hoek, A. (2014). How software designers interact with sketches at the whiteboard. *IEEE Transactions on Software Engineering*, 41(2), 135–156. <https://doi.org/10.1109/TSE.2014.2362924>
- McLaren, B. M., & Isotani, S. (2011). When is it best to learn with all worked examples?. In G. Biswas, S. Bull, J. Kay & A. Mitrovic (Eds.), *Artificial Intelligence in Education. AIED 2011. Lecture Notes in Computer Science, vol 6738* (pp. 222–229). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-21869-9_30
- Mitrovic, A., & Suraweera, P. (2016). Teaching database design with constraint-based tutors. *International Journal of Artificial Intelligence in Education*, 26, 448–456. <https://doi.org/10.1007/s40593-015-0084-6>
- Mitrovic, A., & Weerasinghe, A. (2009). Revisiting ill-definedness and the consequences for ITSs. In V. Dimitrova, R. Mizoguchi, B. du Boulay & A. Graesser (Eds.), *Proceedings of the 2009 conference on Artificial Intelligence in Education: Building Learning Systems that Care: From Knowledge Representation to Affective Modelling* (pp. 375–382). IOS Press.
- Moritz, S., & Blank, G. (2008). Generating and evaluating object-oriented designs for instructors and novice students. In V. Aleven, K. Ashley, C. Lynch & N. Pinkwart (Eds.), *Intelligent Tutoring Systems for Ill-Defined Domains: Assessment and Feedback in Ill-Defined Domains* (pp. 35–45). IEEE.
- Narciss, S. (2013). Designing and evaluating tutoring feedback strategies for digital learning. *Digital Education Review*, 23, 7–26.
- Norese, M. F. (1995). MACRAME: A problem formulation and model structuring assistant in multiactorial contexts. *European Journal of Operational Research*, 84(1), 25–34. [https://doi.org/10.1016/0377-2217\(94\)00315-4](https://doi.org/10.1016/0377-2217(94)00315-4)
- Nyhoff, L. R. (2005). *ADTs, data structures, and problem solving with C++*. Pearson/Prentice Hall.
- Pressman, R. S. (2010). A practitioner's approach. *Software Engineering*, 2, 41–42.
- Pugh, S. (1981). Concept selection: A method that works. In *Proceedings of the International Conference on Engineering Design* (pp. 497–506). The Design Society.
- Pugh, S. (1991). *Total design: Integrated methods for successful product engineering*. Addison-Wesley.
- Razavian, M., Tang, A., Capilla, R., & Lago, P. (2016). In two minds: How reflections influence software design thinking. *Journal of Software: Evolution and Process*, 28(6), 394–426. <https://doi.org/10.1002/smr.1776>
- Reddy, P. D., Iyer, S., & Sasikumar, M. (2016). Teaching and learning of divergent and convergent thinking through open-problem solving in a data structures course. In *Proceedings of 2016 International Conference on Learning and Teaching in Computing and Engineering* (pp. 178–185). IEEE.
- Reddy, P. D., Iyer, S., & Sasikumar, M. (2017). FATHOM: TEL environment to develop divergent and convergent thinking skills in software design. In *Proceedings of 2017 IEEE 17th International Conference on Advanced Learning Technologies* (pp. 414–418). IEEE.
- Reddy, D., Iyer, S., & Sasikumar, M. (2018). Technology enhanced learning (TEL) environment to develop expansionist-reductionist (ER) thinking skills through software design problem solving. In *Proceedings of 2018 IEEE Tenth International Conference on Technology for Education* (pp. 166–173). IEEE.
- Reeves, T. (2006). Design research from a technology perspective. In *Educational design research* (pp. 64–78). Routledge.
- Rivera-Gutierrez, D., Kleinsmith, A., Childs, G., Pileggi, R., & Lok, B. (2016). Self-assessment through interactive in-action reflections to improve interpersonal skills training. In *Proceedings of 2016 IEEE 16th International Conference on Advanced Learning Technologies* (pp. 143–147). IEEE.
- Saaty, T. L. (2008). Decision-making with the Analytic Hierarchy Process. *International Journal of Services Sciences*, 1(1), 83–98. <https://doi.org/10.1504/IJSSCI.2008.017590>
- Shields, P. M., & Rangarajan, N. (2013). *A playbook for research methods: Integrating conceptual frameworks and project management*. New Forums Press.
- Soller, A., Lesgold, A., Linton, F., & Goodman, B. (1999). What makes peer interaction effective? Modeling effective communication in an intelligent CSCL. In *Proceedings of the 1999 AAAI Fall Symposium: Psychological Models Of Communication In Collaborative Systems* (pp. 116–123). Cape Cod.

- Tang, A., Aleti, A., Burge, J., & van Vliet, H. (2010). What makes software design effective?. *Design Studies*, 31(6), 614–640. <https://doi.org/10.1016/j.destud.2010.09.004>
- Tang, A., Bex, F., Schriek, C., & van der Werf, J. M. E. (2018). Improving software design reasoning—A reminder card approach. *Journal of Systems and Software*, 144, 22–40. <https://doi.org/10.1016/j.jss.2018.05.019>
- Tang, A., Tran, M. H., Han, J., & van Vliet, H. (2008). Design reasoning improves software design quality. In S. Becker, F. Plasil & R. Reussner (Eds.), *Quality of Software Architectures. Models and Architectures. QoSA 2008. Lecture Notes in Computer Science*, vol 5281 (pp. 28–42). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-87879-7_2
- Vieira, S. M., Kaymak, U., & Sousa, J. M. (2010). Cohen’s kappa coefficient as a performance measure for feature selection. In *Proceedings of International Conference on Fuzzy Systems* (pp. 1–8). IEEE.
- Volkema, R. J. (1983). Problem formulation in planning and design. *Management Science*, 29(6), 639–652.
- Wang, M., Wu, B., Chen, N. S., & Spector, J. M. (2013). Connecting problem-solving and knowledge-construction processes in a visualization-based learning environment. *Computers & Education*, 68, 293–306. <https://doi.org/10.1016/j.compedu.2013.05.004>
- Willis, J. W. (Ed.). (2009). *Constructivist instructional design (C-ID): Foundations, models, and examples*. IAP.
- Xun, G. E., & Land, S. M. (2004). A conceptual framework for scaffolding III-structured problem-solving processes using question prompts and peer interactions. *Educational Technology Research and Development*, 52(2), 5–22.
- Zannier, C., Chiasson, M., & Maurer, F. (2007). A model of design decision-making based on empirical results of interviews with software designers. *Information and Software Technology*, 49(6), 637–653. <https://doi.org/10.1016/j.infsof.2007.02.010>

Publisher’s Note

The Asia-Pacific Society for Computers in Education (APSCE) remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Research and Practice in Technology Enhanced Learning (RPTEL)
is an open-access journal and free of publication fee.