Fostering Software Design Evaluation Skills in Students using a Technology-enhanced Learning Environment

Prajish Prasad
154380001
under the guidance of Prof Sridhar Iyer
1st July 2021
Motivation

● NIST study -
  2002 - Software bugs cause the US economy - $59.5 billion (Newman, 2002)
● 2016 - $1.1 trillion (Cohane, 2017)
● 1/3rd of costs - earlier identification of software defects
● NASA study - Cost to fix bugs escalates exponentially as the project progresses (Haskins et al., 2004)
Importance of rigorous and effective software evaluation in earlier phases of the development cycle
Software Evaluation: An Example

Automated Door Locking System:

Requirements:

1. If the passcode hasn't been set yet, the user can register and enter a required passcode.
2. When the user chooses the lock option, and enters the correct passcode, the door should lock. If the passcode is incorrect, the door remains unlocked.
3. When the user chooses the unlock option, and enters the correct passcode, the door should unlock. If the passcode is incorrect, the door remains locked.
Requirement: When the user chooses the **lock option**, and enters the correct passcode, the door should lock. If the passcode is incorrect, the door remains unlocked.
Perspectives on Evaluating a Given Design

(Lindland et al., 1994)

- **Requirements**
- **Model (UML diagrams)**
  - **Semantics**: How well the model corresponds to the requirements
  - **Syntax**: How well the model corresponds to the rules of the language
  - **Pragmatics**: How well the model can be interpreted by different stakeholders
- **Language**
  - **Audience interpretation**
Teaching-Learning of Software Design Evaluation

- Software engineering courses - focus on syntax, but not much on semantics (Westphal 2019)
- Evaluating for semantic quality is difficult

Evaluating software designs for semantic quality:

Given a set of goals/requirements and a software system design (UML diagrams) does the design fully satisfy all these goals/requirements?
Broad Research Objective:

“Design and develop a technology-enhanced learning environment (TELE) which enables students to effectively evaluate a software design against the given requirements”
Key Questions Answered in this Thesis

1. **Existing gap** in teaching-learning of software design evaluation

2. Student **difficulties**

3. **Pedagogical strategies** for effective software design evaluation
Overarching Research Methodology: Design Based Research

Adapted from (Plomp, 2013)
Problem Analysis and Exploration

Reflection from DBR Cycle 1

Difficulties students face in evaluating design diagrams after interacting with VeriSIM

Solution Design and Development

Pedagogy

VeriSIM Module 1 and Module 2

Activities and features which support design evaluation

Evaluation and Reflection

Study 3

Effects of VeriSIM 2.0 in students' ability to evaluate a design against the given requirements?

Study 2 and 3

How are features in the VeriSIM learning environment contributing towards student learning?
Scope of the Thesis

Objective: Develop design evaluation skills in students

Context: Students provided with requirements and design diagrams (class and sequence diagrams)

Target population: Computer science undergraduates with basic understanding of class and sequence diagrams

Intervention: VeriSIM:
1. Module 1 - Self-paced TELE
2. Module 2 - Worksheet activity facilitated by instructor
Key Questions Answered in this Thesis

1. Existing gap in teaching-learning of software design evaluation

2. Student difficulties

3. Pedagogical strategies for effective software design evaluation
Teaching-Learning of Software Design Evaluation

Student Difficulties

Expert Practices and Strategies

Literature Review
Evaluating a software design for its semantic quality

- Is hard (Brechner, 2003)
- Has not been sufficiently explored (Westphal, 2019)
Evaluating a software design for its semantic quality
- Is hard (Brechner, 2003)
- Has not been sufficiently explored (Westphal, 2019)

Students have difficulties in designing software systems (Eckerdal et al., 2006; Loftus et al., 2011)
- Insufficient understanding of domain and specifications (Sien, 2011; Chren et al., 2019)
- Understanding relationships between diagrams (Burgueño et al., 2018)
Evaluating a software design for its semantic quality
- Is hard (Brechner, 2003)
- Has not been sufficiently explored (Westphal, 2019)

Students have difficulties in designing software systems (Eckerdal et al., 2006; Loftus et al., 2011)

- Insufficient understanding of domain and specifications (Sien, 2011; Chren et al., 2019)
- Understanding relationships between diagrams (Burgueño et al., 2018)
Expert Practices and Strategies

Experts create rich and detailed mental models of the design and requirements (Adelson and Soloway, 1986; Schumacher and Czerwinski, 1992)

Reasoning Strategies - Generating scenarios, Tradeoff analysis (Tang et al., 2010)

Perform mental simulations on these models (Gentner D, 1983)
What does the mental model of the software design look like?
1. Knowledge
   a. Domain knowledge
   b. Design diagram knowledge
2. Diagram surface elements
3. Main goals
4. Control flow and data flow - dynamic behaviours in the design

(Soloway and Ehrlich, 1984; Pennington, 1987; Von Mayrhauser and Vans, 1996)
Proposed Mental Model Elements for Design Diagrams

Anecdotal Evidence from Experts

Literature Review

- Design Diagram Knowledge
- Problem Domain Knowledge

Mental Model
- Diagram Surface Elements
- Dynamic Behaviours (Control Flow and Data Flow Simulation)
- Main goals
Students have difficulties in designing software systems (Eckerdal et al., 2006; Loftus et al., 2011)

- Insufficient understanding of domain and specifications (Sien, 2011; Chren et al., 2019)
- Understanding relationships between diagrams (Burgueño et al., 2018)

Evaluating a software design for its semantic quality

- Is hard (Brechner, 2003)
- Has not been sufficiently explored (Westphal, 2019)

- Mental modeling (Adelson and Soloway, 1986)
- Mental simulation (Gentner D, 1983)
- Identifying and simulating scenarios (Tang et al., 2010)
Key Questions Answered in this Thesis

1. Existing gap in teaching-learning of software design evaluation
   a. Literature Review

2. Student difficulties
   a. Novice studies - Study 1a and 1b
      RQ 1: How do students evaluate a design against the given requirements?

3. Pedagogical strategies for effective software design evaluation
Novice Study - Study 1a

100 final year computer engineering and information technology engineering students

RQ 1.1: How do students evaluate a software design against the given requirements?

Data Source

Student response sheets

Data Analysis

Content analysis
Study 1a - Findings

- Identify scenarios which do not satisfy requirements
  - Focus on **dynamic behaviours and main goals** in the design
- Change data types, functions of class diagram
  - Focus on **diagram surface elements** elements in the design
- Change existing functionalities and requirements
  - Focus on **new elements** absent in the design
- Add new functionality
**Novice Study - Study 1b**

**Qualitative Study** - 6 computer engineering and information technology engineering students

<table>
<thead>
<tr>
<th>RQ 1.2: What defects are students able to identify in the design evaluation task?</th>
<th>Data Source</th>
<th>Data Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Student responses on the task sheet</td>
<td>Thematic analysis of transcripts</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ 1.3: What reading strategies do students use?</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Video of students performing the task and screen capture</td>
<td>Thematic analysis of video data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ 1.4: What are the elements in their mental model?</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio transcripts of the post-task interview</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Study 1b - Findings

- Able to do a **superficial search** on the design diagrams.

- Have difficulty in **identifying scenarios** where the design does not satisfy the requirement.

- Difficulty in **simulating the control flow and data flow** within design diagrams.
Novice studies - Connecting to the Mental Model Elements
Scaffolding students to **identify and model relevant scenarios** in the design can lead to effective software design evaluation.
Key Questions Answered in this Thesis

1. Existing gap in teaching-learning of software design evaluation
   a. Literature Review

2. Student difficulties
   Novice studies - Study 1a and 1b
   RQ 1: How do students evaluate a design against the given requirements?

3. Pedagogical strategies for effective software design evaluation
   a. VeriSIM pedagogy
   b. Effectiveness Studies - Study 2 and 3
   c. How pedagogical features of VeriSIM are contributing towards learning
VeriSIM Pedagogy

Verifying Designs by Simulating Scenarios

- Design Tracing - Model scenarios
- Scenario branching - Identify scenarios
VeriSIM Pedagogy

- Design Tracing - Model scenarios
- Scenario branching - Identify scenarios
VeriSIM Pedagogy: Design Tracing Strategy

Scenario:
When the door is initially locked and the user selects the unlock option and enters the correct passcode, the door unlocks

Construct a state diagram which models the scenario
VeriSIM Learning Environment

- VeriSIM Learning platform
- Web-based learning environment - Developed using Vue.js, Node.js and MongoDB

Design Tracing Stage - 4 challenges:

1. Explore the model
2. Correct the model
3. Complete the model
4. Construct the model

More details about VeriSIM [here](#)
Model scenarios in the design using the design tracing pedagogy.

Watch the demo [here](#)
Design Tracing Stage- Challenge 3 - **Complete the Model**

Scenario: When a new user selects the register option and enters a passcode, the system saves the passcode and registers the user.

In this challenge, you are going to trace the given scenario. The state diagram is given with relevant events, but the data variables and their values are missing. In order to complete the state diagram, add relevant data variables in the state tab. You can see the values of these variables by editing each state in the state diagram. Remember, you can click on “run” at any time to get feedback about your model.

Design Tracing Stage- Challenge 4 - **Construct the Model**

Scenario: When the door is initially unlocked and the user chooses the lock option and enters an incorrect passcode, the door remains unlocked.

In this challenge, your goal is to construct the state diagram. You can use the state events and state tab to construct the state diagram.
Connecting the Pedagogy to Mental Model Elements

- Design Diagram Knowledge
- Problem Domain Knowledge

**Mental Model of the Design**
- Diagram Surface Elements
- Dynamic Behaviours (Control Flow and Data Flow Simulation)
- Main goals

**Mental Model of the Problem Domain**
- Scenarios
- Requirements

Design Tracing
Scenario Branching
VeriSIM Pedagogy: Scenario Branching Strategy

- Design Tracing - Model scenarios
- Scenario branching - Identify scenarios
VeriSIM Pedagogy: Scenario Branching Strategy

Identify scenarios for each requirement in the design using a concept map

Requirement: A user with a valid account can register his/her ATM and set a PIN if he/she has not set a PIN yet. The PIN should be of length 4 and should contain only numbers.
Theoretical Basis: Model-based Learning

(Buckley et al., 2010)
Theoretical Basis: Model-based Learning

Modeling Practices

Observations → Experiences → Model formulation → Initial model → Intermediate models → Target Model → Model comparison and evaluation → Model revision

Design Tracing - Model scenarios
Theoretical Basis: Model-based Learning

Modelling Practices

Scenario branching - Identify scenarios

Observations

Experiences

Model formulation

Initial model

Intermediate models

Target Model

Model comparison and evaluation

Model revision
Pedagogical Features: Model Progression

Progressively learn to construct the model (Mulder et al. 2011)

1. Prior exploration of model (Kopainsky et al., 2015)
2. Learning from erroneous models (Wijnen et al., 2015)
3. Learning from partial models (Mulder et al., 2016)

Challenge 1-3 help learners construct the model for a given scenario
Pedagogical Features: Visualize Model Execution
Connecting the Pedagogy to Mental Model Elements

Design Diagram Knowledge

Mental Model of the Design
- Diagram Surface Elements
- Dynamic Behaviours (Control Flow and Data Flow Simulation)
- Main goals

Mental Model of the Problem Domain
- Scenarios
- Requirements

Problem Domain Knowledge

Design Tracing

Scenario Branching
Key Questions Answered in this Thesis

1. Existing gap in teaching-learning of software design evaluation
   a. Literature Review

2. Student difficulties
   a. Novice studies - Study 1a and 1b
      RQ 1: How do students evaluate a design against the given requirements?

3. Pedagogical strategies for effective software design evaluation
   a. VeriSIM pedagogy
   b. Effectiveness Studies - Study 2 and 3
      RQ 2 and RQ 3: What are effects of the VeriSIM pedagogy in students’ ability to evaluate a design against the given requirements?
Refinement of the Pedagogy

DBR Cycle 1
VeriSIM 1.0 - Design Tracing
Study 2

DBR Cycle 2
VeriSIM 2.0 - Design Tracing + Scenario Branching
Study 3
Study 2

RQ 2.1 Does VeriSIM improve learners ability to **model a given scenario**?

RQ 2.2 Does VeriSIM improve learners ability to **uncover defects**?

**Data Source**

- Question in pre-test and post-test: **Explain the changes in the system on execution of this scenario**
- Question in pre-test and post-test: **Uncover defects in design diagrams**

**Data Analysis**

- Differences in pre-test and post-test question based on rubric
- Content analysis of “uncover defects” question in the pre-test and post-test

More details
Study 2: Results - RQ 2.1: Ability to model scenarios

**Data criteria**

<table>
<thead>
<tr>
<th></th>
<th>Pre-test Mean (SD)</th>
<th>Post-test Mean (SD)</th>
<th>Paired t-test (p value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifying relevant data variables</td>
<td>0.47(0.70)</td>
<td>0.95(0.87)</td>
<td>0.00</td>
</tr>
<tr>
<td>Identifying relevant events</td>
<td>1.16(0.62)</td>
<td>1.28(0.88)</td>
<td>0.17</td>
</tr>
<tr>
<td>Simulating state change</td>
<td>0.44(0.68)</td>
<td>0.84(0.84)</td>
<td>0.00</td>
</tr>
<tr>
<td>Total</td>
<td>2.07(1.70)</td>
<td>3.07(2.09)</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Statistically significant improvement in students’ ability to model scenarios
No difference in ability to identify scenarios not satisfying the requirement
Students’ ability to model scenarios improved. Students need explicit help to identify scenarios.
Study 3: Results - RQ 3.2: Identify defects

Percentage of defect categories in Pre-test and Post-test

<table>
<thead>
<tr>
<th>Category</th>
<th>Pre-test</th>
<th>Post-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifying scenarios which do not satisfy the requirements</td>
<td>26.67%</td>
<td>76%</td>
</tr>
<tr>
<td>Change in data types, function and structure of class diagram</td>
<td>13.33%</td>
<td>0%</td>
</tr>
<tr>
<td>Add new functionality</td>
<td>24.44%</td>
<td>2%</td>
</tr>
<tr>
<td>Change in existing functionalities and requirements</td>
<td>31.11%</td>
<td>16%</td>
</tr>
<tr>
<td>No defects</td>
<td>4.44%</td>
<td>6%</td>
</tr>
</tbody>
</table>
Students' ability to model scenarios improved
Students need explicit help to identify scenarios

Study 2

DBR Cycle 1

VeriSIM 1.0 - Design Tracing

Study 3

DBR Cycle 2

VeriSIM 2.0 - Design Tracing + Scenario Branching

Students’ ability to identify scenarios improved
Key Questions Answered in this Thesis

1. Existing gap in teaching-learning of software design evaluation
   a. Literature Review

2. Student difficulties
   a. Novice studies - Study 1a and 1b

3. Pedagogical strategies for effective software design evaluation
   a. VeriSIM pedagogy
   b. Effectiveness Studies - Study 2 and 3
      RQ 2 and RQ 3: What are effects of the VeriSIM pedagogy in students’ ability to evaluate a design against the given requirements?
   c. Pedagogical features of VeriSIM
      RQ 4: How are features in VeriSIM contributing towards student learning?
RQ 4: How are features in VeriSIM contributing towards student learning?

- **Key Features in VeriSIM:**
  - Model progression of Challenges
  - Model execution visualization (Run)
  - Scenario branching

- **Data Sources -**
  - Interaction Logs - 48 students who gave consent (Study 2 and 3)
  - Focus group interviews
Model Progression of Challenges

Challenges in increasing order of difficulty.

- Challenge 1 - Explore the model
- Challenge 2 - Correct the model
- Challenge 3 - Complete the model
- Challenge 4 - Construct the model
Model Execution Visualization

Students use the model execution visualization feature while modelling scenarios.
Model Execution Visualization

From focus group interviews

- Helped map a particular state to the corresponding part of the scenario
- Understand the relationship between the scenario and different diagrams
- Visual feedback helped learners identify which parts had errors.
Scenario Branching Strategy

Focus group interview:

- Structuring and breaking down the design problem
- Macro-view of the design problem
- Identify scenarios missing in the design diagrams
Summary and Contributions
Review of Literature

Experts create a rich mental model of the design, use various reasoning techniques, and perform mental simulations.

Students have difficulties in developing a rich and consistent mental model of the design.

Novice studies - Study 1a and 1b

Have difficulty in identifying and simulating scenarios where the design does not satisfy the requirement.

Able to do a superficial search on the design diagrams.

Difficulty in simulating the control flow and data flow within design diagrams.

VeriSIM Pedagogy

Scenario branching - Identify scenarios

Design Tracing - Model scenarios

Evaluation studies - Study 2 and Study 3

- Students’ ability to model scenarios improved
- Students’ ability to identify defects improved
- Pedagogical features in VeriSIM contribute towards effective learning of design evaluation
Contributions

1. **Unpacking learner difficulties** while evaluating design diagrams
   Quantitative and qualitative investigations on how students evaluate design diagrams and difficulties which they face

2. **Pedagogies for evaluating design diagrams** -
   The design tracing and scenario branching can be used by instructors in software design courses

3. **VeriSIM learning environment** -
   a. Directly used by instructors as well as students to be trained in evaluating design diagrams against the requirements - [https://verisim.tech](https://verisim.tech)
   b. Design features of VeriSIM - used by learning environment designers in related contexts.
Implications

● **Teaching-learning of Software Design**
  ○ Equip students to identify specific scenarios and model them
  ○ Provide activities to help students progressively model scenarios in the design

● **Characterization of student mental models for design diagrams**

● **Model-based learning paradigm for computing disciplines**
Generalizability

- Extension to **other UML diagrams**
  - Underlying principle of identifying and modelling scenarios can be extended to other design diagrams

- Extension to teaching-learning of **software design creation**
  - While creating a design based on the given requirements, students can identify and model various scenarios in their own designs
Limitations

● Learner characteristics
  ○ Personal, social, emotional and cognitive characteristics
  ○ Prior experience working with software designs

● Scoping the construct and skills involved in ‘evaluation’
  ○ Other perspectives - Syntactic and pragmatic deficiencies
  ○ Inter-personal and collaboration skills (Li, 2016)
Future Work

● Developing an instructor interface for the VeriSIM learning environment

● Using eye-tracking for a deeper understanding of how students evaluate a design

● Investigating the effects of evaluation before creation of designs
Acknowledgements

- Friends and Family
- EdTech department Family
- Bhupender Singh - Design and Development of VeriSIM
- Kinnari Gatare - UI/UX Design of VeriSIM
- Herold, Lakshmi - Initial design, planning of activities in TELE
- Colleagues from Fr. CRCE and SIES
Thesis-related Publications

Conference Papers


Posters


Bibliography - I


Gentner, D., Gentner, D. R., 1983. Flowing waters or teeming crowds: Mental models of electricity. Mental models 99, 129.
Bibliography - II


Thank You
Extra Slides
Study 1b - Details
Study 1b: Characterizing Students’ Evaluation Process

Research Questions

RQ 1.2: What defects are students able to identify in the design evaluation task?

RQ 1.3: What reading strategies do students use?

RQ 1.4: What are the elements in their mental model?
Study 1b: Study Procedure

- 6 computer engineering and information technology engineering students (3 in third year, 3 in final year)
- **Familiar with class and sequence diagrams** - were introduced to UML diagrams in the previous semester.

- Students - provided with requirements and design diagrams - 1 Class diagram, 3 sequence diagrams for a door locking system
- **Task** - *For each requirement, your task is to provide a logical explanation for how the design satisfies/does not satisfy the requirement. You are free to use any notation/diagrams to support your explanation*
Study 1b: Study Procedure

1. Participant provided with task sheet and design diagrams
   Task sheet contains requirements
   Design diagrams provided in the Umbrello interface

2. Researcher explains task to participant
   Participant has to check whether the requirements are being satisfied by the design

3. Participant performs the task
   Participant is free to work silently or think aloud. Researcher takes observation notes and is available for answering any queries

4. Post-task Interview
   Participants elaborate and discuss how they went about solving the task
<table>
<thead>
<tr>
<th>RQ 1.2: What defects are students able to identify in the design evaluation task?</th>
<th>Data Source</th>
<th>Data Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student responses on the task sheet</td>
<td>Thematic analysis of video data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ 1.3: What reading strategies do students use?</th>
<th>Data Source</th>
<th>Data Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video of students performing the task and screen capture</td>
<td>Thematic analysis of audio data</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ 1.4: What are the elements in their mental model?</th>
<th>Data Source</th>
<th>Data Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audio of the post-task interview</td>
<td>Thematic analysis of video data</td>
<td></td>
</tr>
</tbody>
</table>
Study 1b: Results

RQ 1.2: What defects are students able to identify in the design evaluation task?

Able to identify defects which involve a superficial search on the design diagrams

RQ 1.3: What reading strategies do students use?

Single and multiple switches between design diagrams and requirements

Focussed on surface level parts of the diagrams

RQ 1.4: What are the elements in their mental model?

Lacked deep exploration of the design - flow of messages and how values change of variables

Back to main slides
Study 2 - Details
### Study 2: Effect of VeriSIM on Students’ Evaluation Skills:

<table>
<thead>
<tr>
<th>RQ 2.1</th>
<th>Question in pre-test and post-test: <strong>Explain the changes in the system on execution of this scenario</strong></th>
<th>Differences in pre-test and post-test question based on rubric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does VeriSIM improve learners ability to <strong>model a given scenario?</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>RQ 2.2</th>
<th>Question in pre-test and post-test: <strong>Uncover defects in design diagrams</strong></th>
<th>Content analysis of “uncover defects” question in the pre-test and post-test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Does VeriSIM improve learners ability to <strong>uncover defects?</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Study 2: Study Procedure

- 86 final year computer engineering and information technology engineering students (48 male and 38 female)
- Familiar with class and sequence diagrams - had a software engineering course in the previous semester
Study 2: Study Procedure

1. Pre-registration
   - Basic information - overall percentage in last semester, rate their confidence in understanding of object-oriented design, class and sequence diagrams

2. Pre-test
   - Design of ATM system:
     - Class diagram
     - 3 sequence diagram
   - Questions:
     - Execute the given scenario
     - Identify defects based on the requirement

3. Interaction with VeriSIM
   - Design of library system:
     - Class diagram
     - 3 sequence diagram
   - Questions:
     - Execute the given scenario
     - Identify defects based on the requirement

4. Post-test

5. Focus group interviews
   - Questions
     - What are the main things you learnt from the workshop?
     - What according to you is design tracing?
     - What is the usefulness of constructing the state diagram?
<table>
<thead>
<tr>
<th>Study 2: Data Source and Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RQ 2.1 Does VeriSIM improve learners ability to model a given scenario?</strong></td>
</tr>
<tr>
<td><strong>RQ 2.2 Does VeriSIM improve learners ability to uncover defects?</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data Source</th>
<th>Data Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Question in pre-test and post-test: <em>Explain the changes in the system on execution of this scenario</em></td>
<td>Differences in pre-test and post-test question based on rubric</td>
</tr>
<tr>
<td>Question in pre-test and post-test: <em>Uncover defects in design diagrams</em></td>
<td>Content analysis of “uncover defects” question in the pre-test and post-test</td>
</tr>
</tbody>
</table>
### Study 2: Results - RQ 2.1: Model a given scenario: Rubric

<table>
<thead>
<tr>
<th></th>
<th>Missing (0)</th>
<th>Almost (1)</th>
<th>Target (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Identifying relevant data variables</strong></td>
<td>Missing all relevant data variables from the class diagram</td>
<td>Identifies some relevant variables</td>
<td>Identifies all relevant data variables</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Adds irrelevant data variables</td>
<td>No irrelevant data variables added</td>
</tr>
<tr>
<td><strong>Identifying relevant events</strong></td>
<td>Missing all relevant events Separation of events is not seen</td>
<td>Identifies some relevant events</td>
<td>Identifies all relevant events</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Identifies some irrelevant events Separation of events is unclear</td>
<td>No irrelevant events included Separation of events is clear</td>
</tr>
<tr>
<td><strong>Simulating state change</strong></td>
<td>No mention of state change of variables</td>
<td>State change of some variables are mentioned with variable-value pairs</td>
<td>State change of all variables are clearly mentioned with correct variable-value pairs</td>
</tr>
</tbody>
</table>
Study 2: Results - RQ 2.1: Ability to model scenarios

<table>
<thead>
<tr>
<th>Data criteria</th>
<th>Pre-test Mean (SD)</th>
<th>Post-test Mean (SD)</th>
<th>Paired t-test (p value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifying relevant data variables</td>
<td>0.47(0.70)</td>
<td>0.95(0.87)</td>
<td><strong>0.00</strong></td>
</tr>
<tr>
<td>Identifying relevant events</td>
<td>1.16(0.62)</td>
<td>1.28(0.88)</td>
<td>0.17</td>
</tr>
<tr>
<td>Simulating state change</td>
<td>0.44(0.68)</td>
<td>0.84(0.84)</td>
<td><strong>0.00</strong></td>
</tr>
<tr>
<td>Total</td>
<td>2.07(1.70)</td>
<td>3.07(2.09)</td>
<td><strong>0.00</strong></td>
</tr>
</tbody>
</table>

Statistically significant improvement in students’ ability to trace scenarios
Study 2: Results - RQ 2.2: Ability to uncover defects

Total number of responses in Pre-test: 145
Total number of responses in Post-test: 71
Summary: Study 2: Reflection - Cycle 1

- There is a statistically significant *improvement in students’ ability to model scenarios*
- Students perceive that design tracing is helping them
  - Develop an *integrated understanding of design diagrams*
  - Evaluate design diagrams better

- Spread VeriSIM over multiple days to avoid fatigue
- Design tracing <-> Evaluating design diagrams
  Students need explicit help to *generate and identify scenarios which do not satisfy the requirements*
Scenario Branching Strategy
Scenario Branching Strategy

Requirement: A user with a valid account can register his/her ATM and set a PIN if he/she has not set a PIN yet. The PIN should be of length 4 and should contain only numbers.

Steps:

- Identify subgoals in the requirement

Subgoals:

- User with valid account
- Sets a PIN if a PIN hasn’t been set yet
- PIN should be of length 4 and should contain only numbers
Scenario Branching Strategy

Requirement: A user with a valid account can register his/her ATM and set a PIN if he/she has not set a PIN yet. The PIN should be of length 4 and should contain only numbers.

Steps:

- Identify subgoals in the requirement
- Identify relevant variables and different possibilities of these variables

User with valid account

Sets a PIN if a PIN hasn’t been set yet
Scenario Branching Strategy

Requirement: A user with a valid account can register his/her ATM and set a PIN if he/she has not set a PIN yet. The PIN should be of length 4 and should contain only numbers.

Steps:

- Identify subgoals in the requirement
- Identify relevant variables and different possibilities of these variables
- Identify relevant scenarios based on the requirement

- Scenario 1: User with a valid account has already set a Pin
- Scenario 2: User with a valid account has not set a Pin and sets a valid Pin
- Scenario 3: User with a valid account has not set a Pin and sets an invalid Pin
- Scenario 4: User has an invalid account
Scenario Branching Strategy

Requirement: A user with a valid account can register his/her ATM and set a PIN if he/she has not set a PIN yet. The PIN should be of length 4 and should contain only numbers.

Steps:

● Identify subgoals in the requirement
● Identify relevant variables and different possibilities of these variables
● Identify relevant scenarios based on the requirement
● **Identify scenarios which are not satisfying the requirement**

● Scenario 1: User with a valid account has already set a Pin
● Scenario 2: User with a valid account has not set a Pin and sets a valid Pin
● Scenario 3: User with a valid account has not set a Pin and sets an invalid Pin
● Scenario 4: User has an invalid account
Implementation of Scenario Branching Strategy to VeriSIM

**Worksheet**
- Learners provided with requirements and design diagrams
- Worksheet outlines how to construct the scenario tree for a requirement
- Students are required to **construct the scenario tree** for the remaining requirements.

**CMAP Tool**
- Nodes - contain values of the identified data variables
- Links - denote different possible scenarios for the subgoals.
- Mentally trace each path and identify all possible scenarios.
Study 3 - Details
Study 3: Effects of VeriSIM 2.0 in Students’ Evaluation Skills

- 18 second year computer engineering and information technology engineering students
- Part of a Software design workshop
- Familiar with class and sequence diagrams - were introduced to UML diagrams a few days prior.
Study 3: Study Procedure

### Registration and Pre-test
Design of ATM system:
- Class diagram
- 3 sequence diagram

Questions:
- Identify scenarios for each requirement
- Identify defects based on the requirement

### VeriSIM - Module 1
Design Tracing Pedagogy

### Focus group interviews - 1
Questions:
- What are the main things you learnt from the workshop?
- What according to you is design tracing?
- What is the usefulness of constructing the state diagram?

### VeriSIM - Module 2
Scenario branching pedagogy worksheet

### Post-test and focus group interviews - 2
Design of a streaming website
- Class diagram
- 3 sequence diagram

Questions:
- Identify scenarios for each requirement
- Identify defects based on the requirement
Study 3: Data Sources and Data Analysis

RQ 3.1 Does VeriSIM improve learners ability to identify scenarios in a given design?

Content analysis of “identify scenarios” question in the pre-test and post-test

RQ 3.2 Does VeriSIM improve learners ability to uncover defects?

Content analysis of “uncover defects” question in the pre-test and post-test
Study 3: Results - RQ 3.1: Identify Scenarios

Percentage of Scenario Categories in Pre-test and Post-test

Total number of responses in Pre-test: 81
Total number of responses in Post-test: 94
Study 3: Results - RQ 3.2: Identify Defects

Percentage of defect categories in Pre-test and Post-test

- Identifying scenarios which do not satisfy the requirements: Pre-test 26.67%, Post-test 76%
- Change in data types, function and structure of class diagram: Pre-test 13.33%, Post-test 0%
- Add new functionality: Pre-test 24.44%, Post-test 2%
- Change in existing functionalities and requirements: Pre-test 31.11%, Post-test 16%
- No defects: Pre-test 4.44%, Post-test 6%

Total number of responses in Pre-test: 45
Total number of responses in Post-test: 50