Thesis Defense Presentation

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

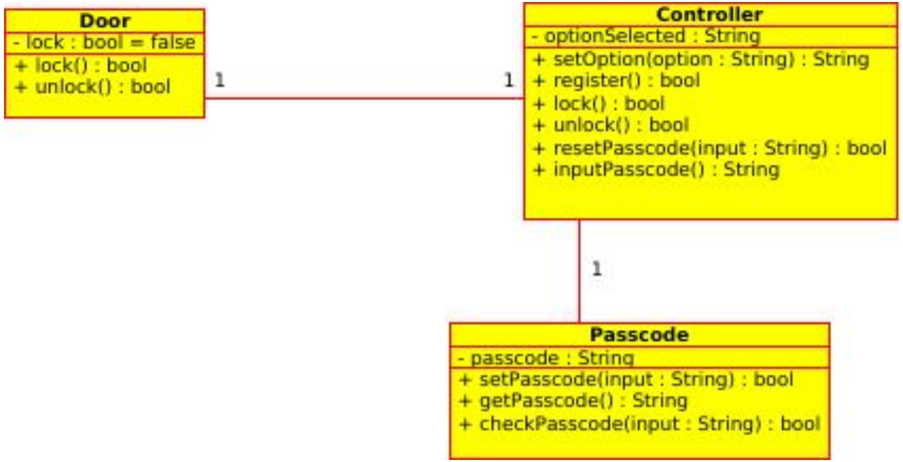
Software Evaluation: An Example

Automated Door Locking System:

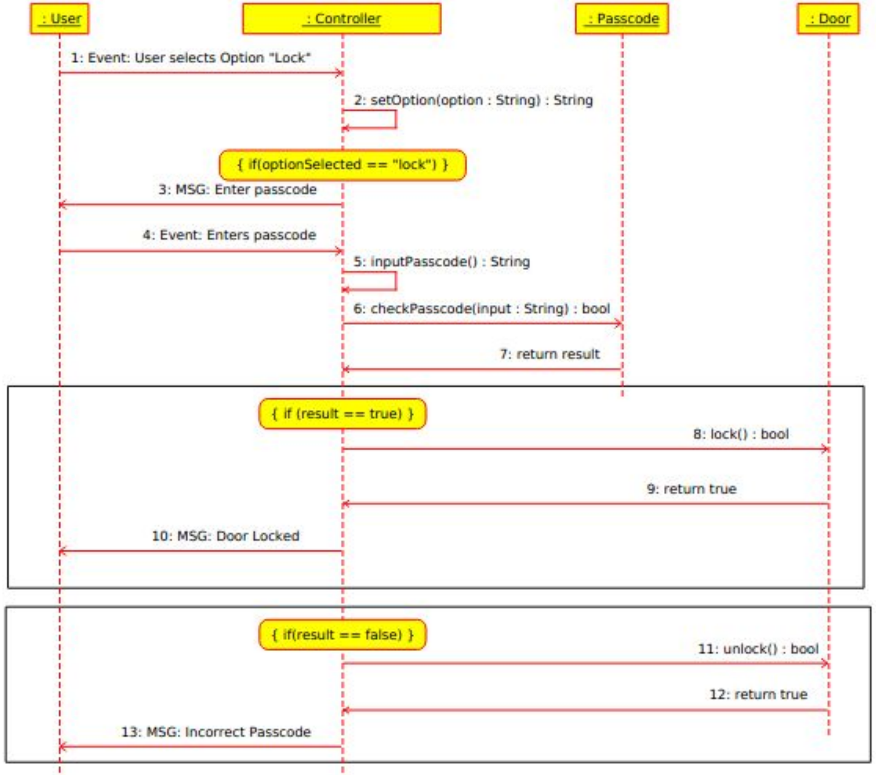
Requirements:

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.

Requirements Modelled using Unified Modelling Language Diagrams


Door
- lock : bool = false
+ lock() : bool
+ unlock() : bool
1
1
Controller
- optionSelected : String
+ setOption(option : String) : String
+ register() : bool
+ lock() : bool
+ unlock() : bool
+ resetPasscode(input : String) : bool
+ inputPasscode() : String
1
Passcode
- passcode : String
+ setPasscode(input : String) : bool
+ getPasscode() : String
+ checkPasscode(input : String) : bool

Class diagram

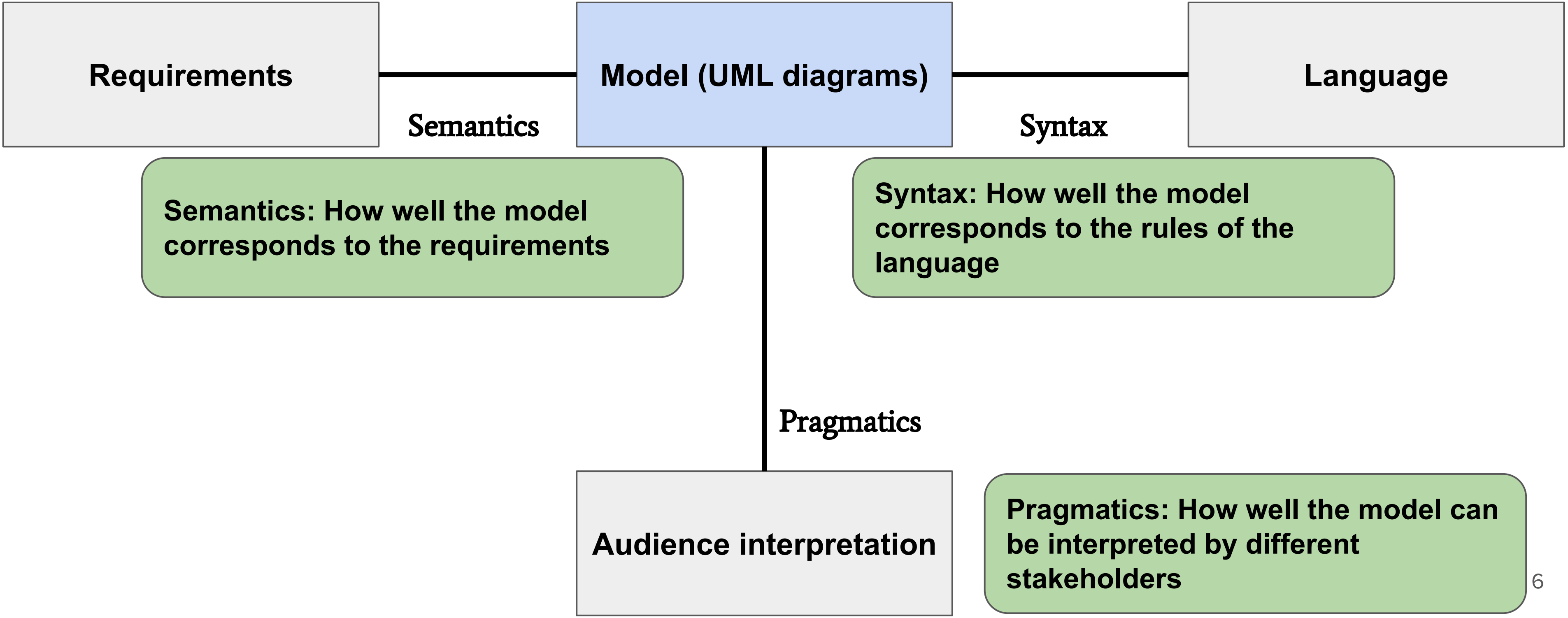

: User
: Controller
: Passcode
: Door
1: Event: User selects Option "Lock"
2: setOption(option : String) : String
{ if(optionSelected == "lock") }
3: MSG: Enter passcode
4: Event: Enters passcode
5: inputPasscode() : String
6: checkPasscode(input : String) : bool
7: return result
{ if (result == true) }
8: lock() : bool
9: return true
10: MSG: Door Locked
{ if(result == false) }
11: unlock() : bool
12: return true
13: MSG: Incorrect Passcode

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.

Sequence diagram for the lock use case

Perspectives on Evaluating a Given Design

(Lindland et al., 1994)



Requirements

Semantics

Model (UML diagrams)

Syntax

Language

Semantics: How well the model corresponds to the requirements

Syntax: How well the model corresponds to the rules of the language

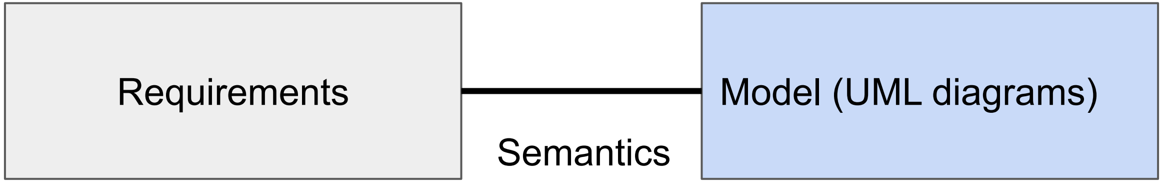
Pragmatics

Audience interpretation

Pragmatics: How well the model can be interpreted by different stakeholders

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



Requirements

Semantics

Model (UML diagrams)

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"

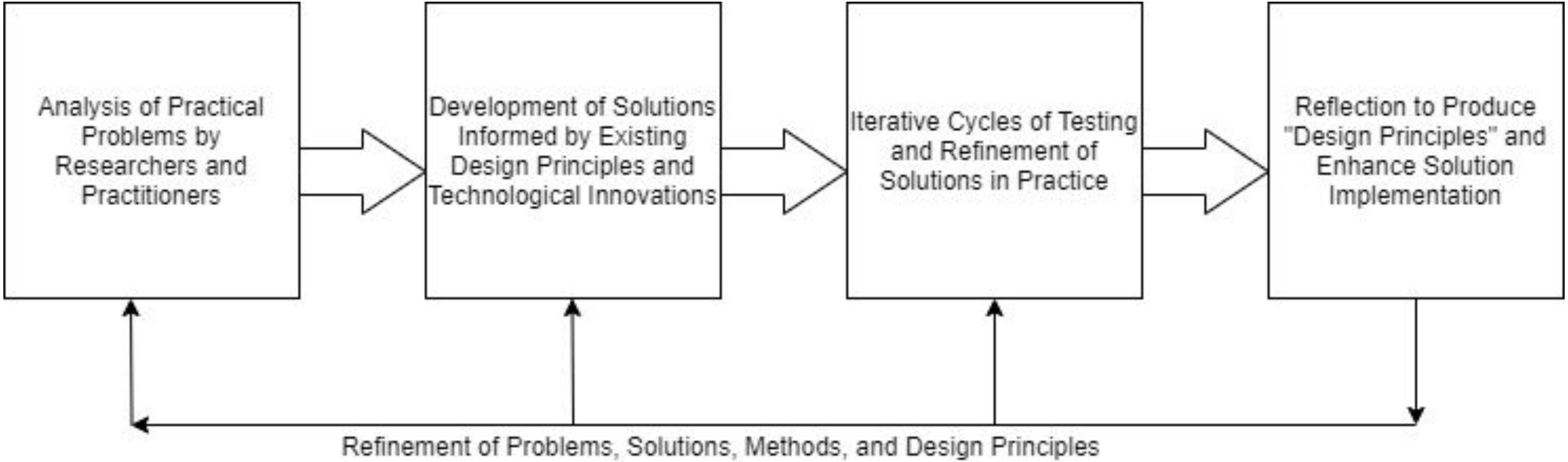
Key Questions Answered in this Thesis

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



Analysis of Practical Problems by Researchers and Practitioners

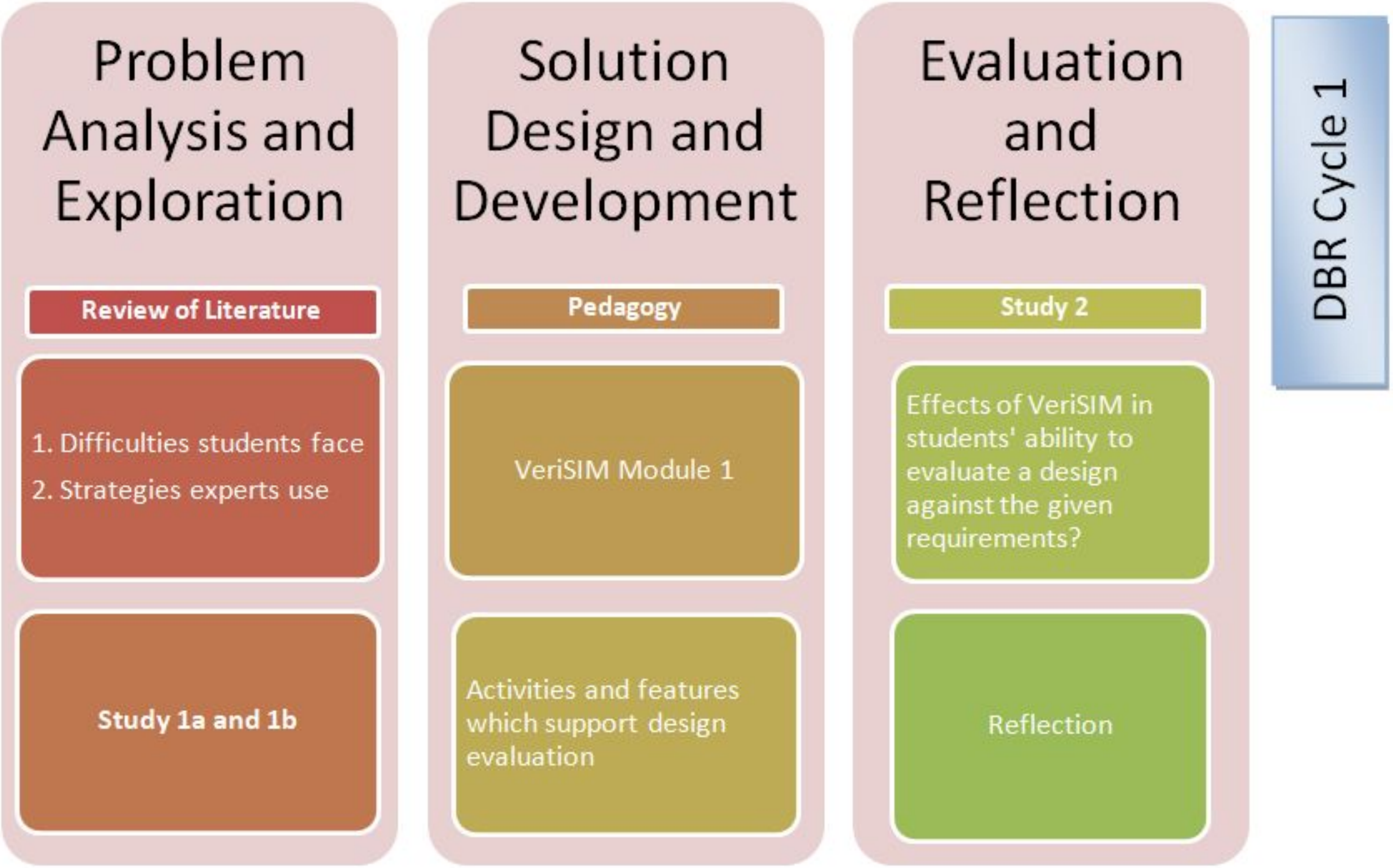
Development of Solutions Informed by Existing Design Principles and Technological Innovations

Iterative Cycles of Testing and Refinement of Solutions in Practice

Reflection to Produce "Design Principles" and Enhance Solution Implementation

Refinement of Problems, Solutions, Methods, and Design Principles

Adapted from (Plomp, 2013)



Problem Analysis and Exploration

Review of Literature

1. Difficulties students face
2. Strategies experts use

Study 1a and 1b

Solution Design and Development

Pedagogy

VeriSIM Module 1

Activities and features which support design evaluation

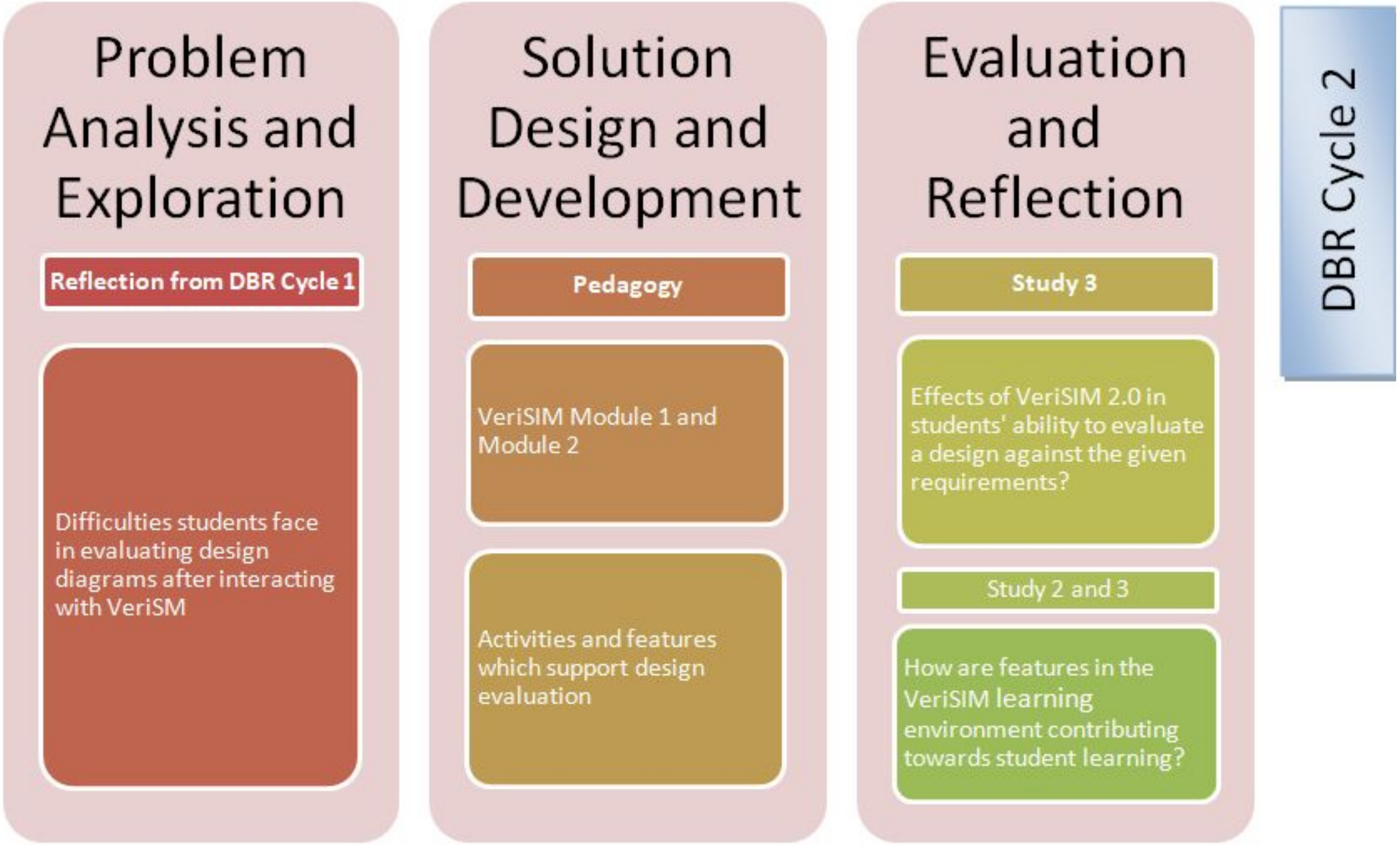
Evaluation and Reflection

Study 2

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

Reflection

DBR Cycle 1



Problem Analysis and Exploration

Reflection from DBR Cycle 1

Difficulties students face in evaluating design diagrams after interacting with VeriSM

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?

DBR Cycle 2

Scope of the Thesis

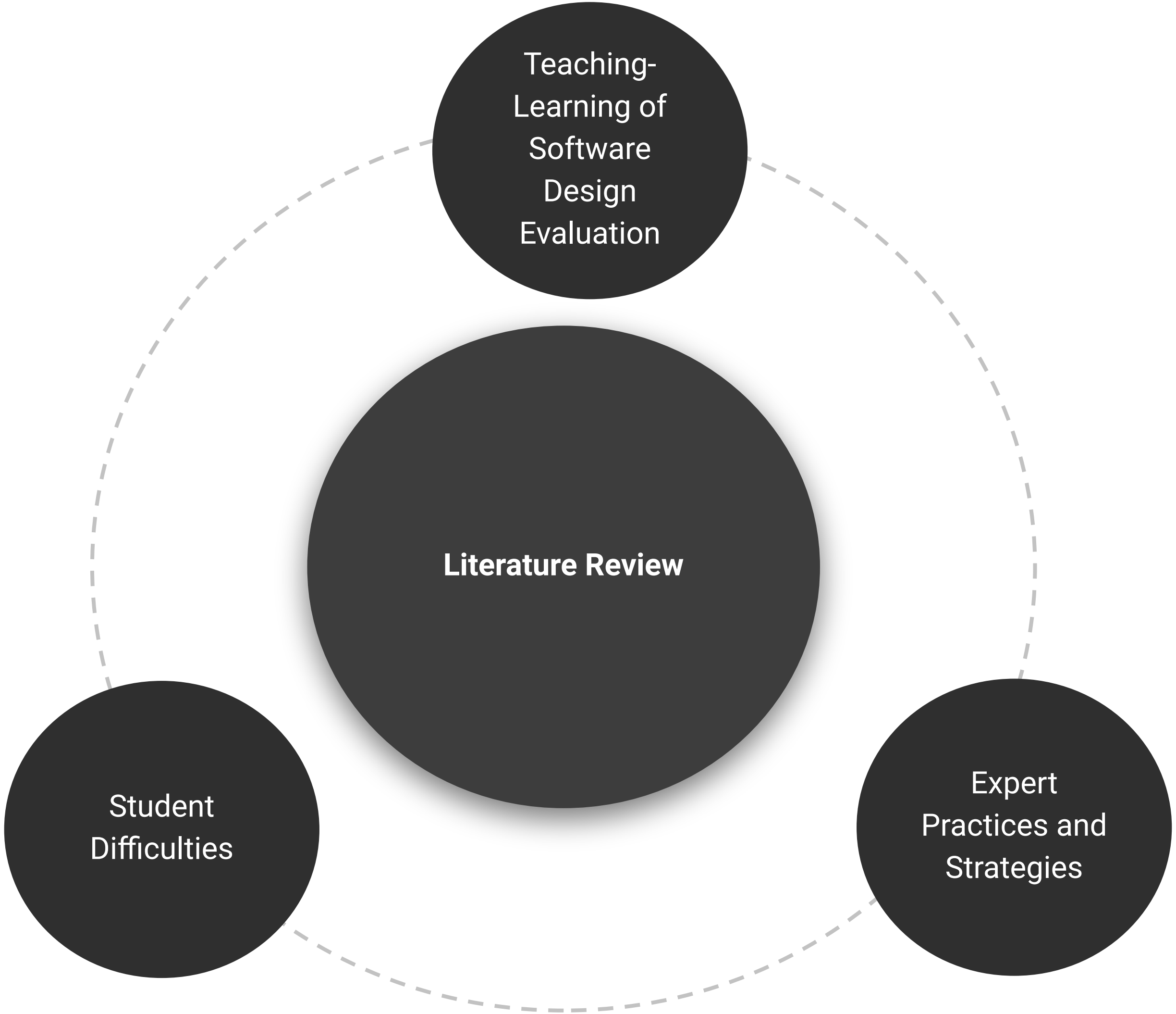
Objective	**Context**	**Target population**	**Intervention**
Develop design evaluation skills in students	Students provided with requirements and design diagrams (class and sequence diagrams)	Computer science undergraduates with basic understanding of class and sequence diagrams	VeriSIM: 1. Module 1 - 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

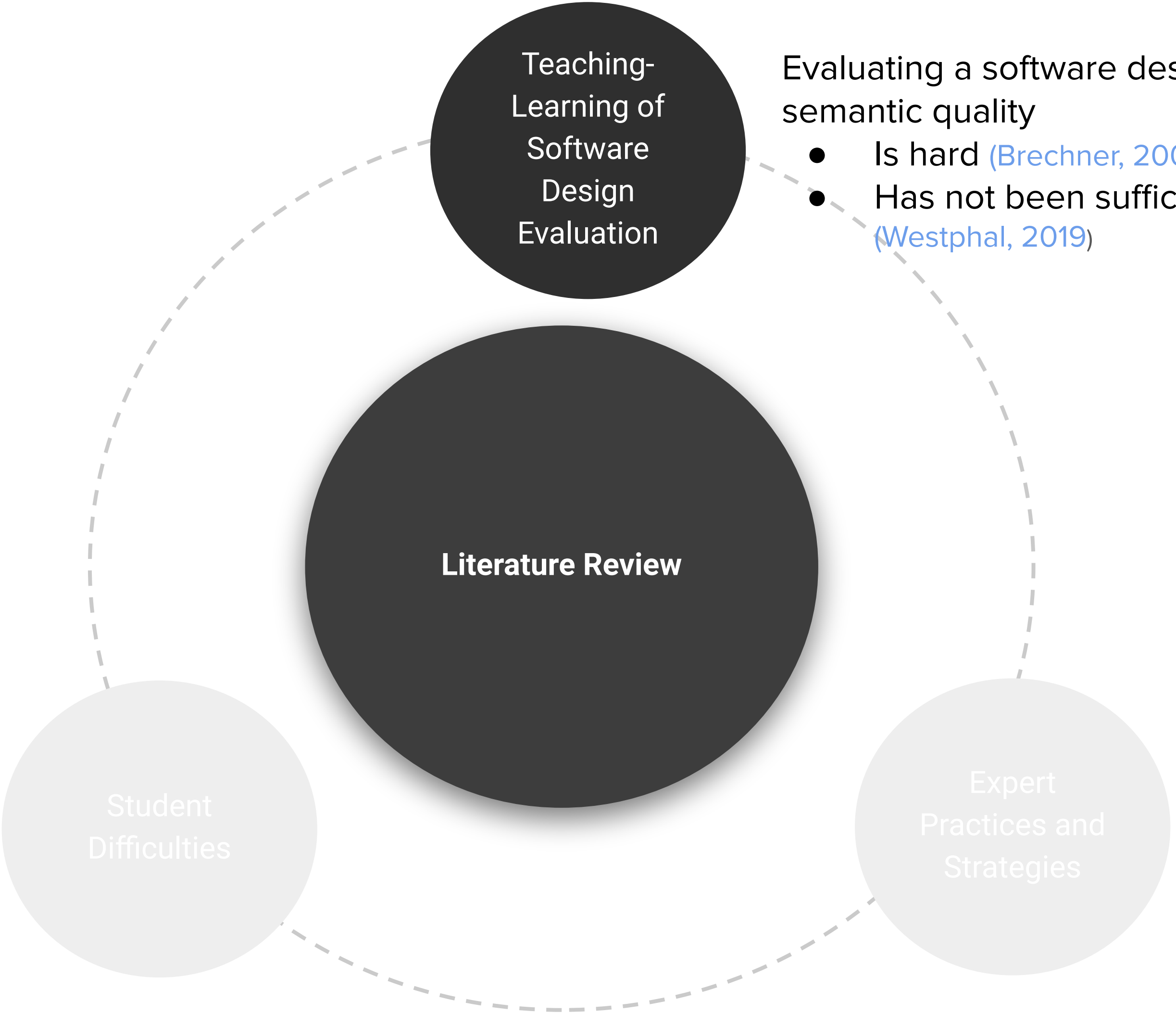


Literature Review

Teaching-Learning of Software Design Evaluation

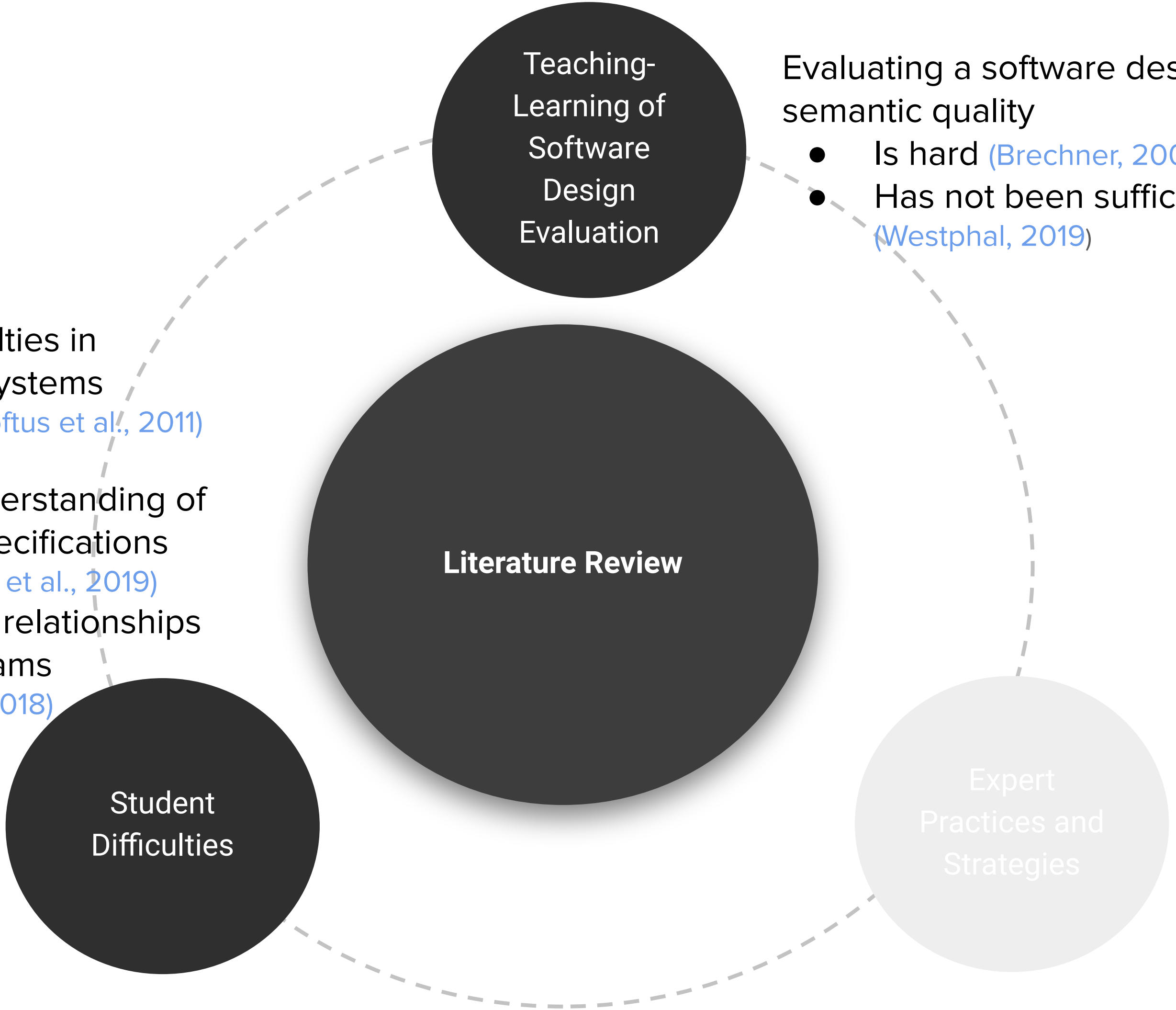
Student Difficulties

Expert Practices and Strategies


Teaching-Learning of Software Design Evaluation
Literature Review
Student Difficulties
Expert Practices and Strategies

Evaluating a software design for its semantic quality

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



Teaching-Learning of Software Design Evaluation

Literature Review

Student Difficulties

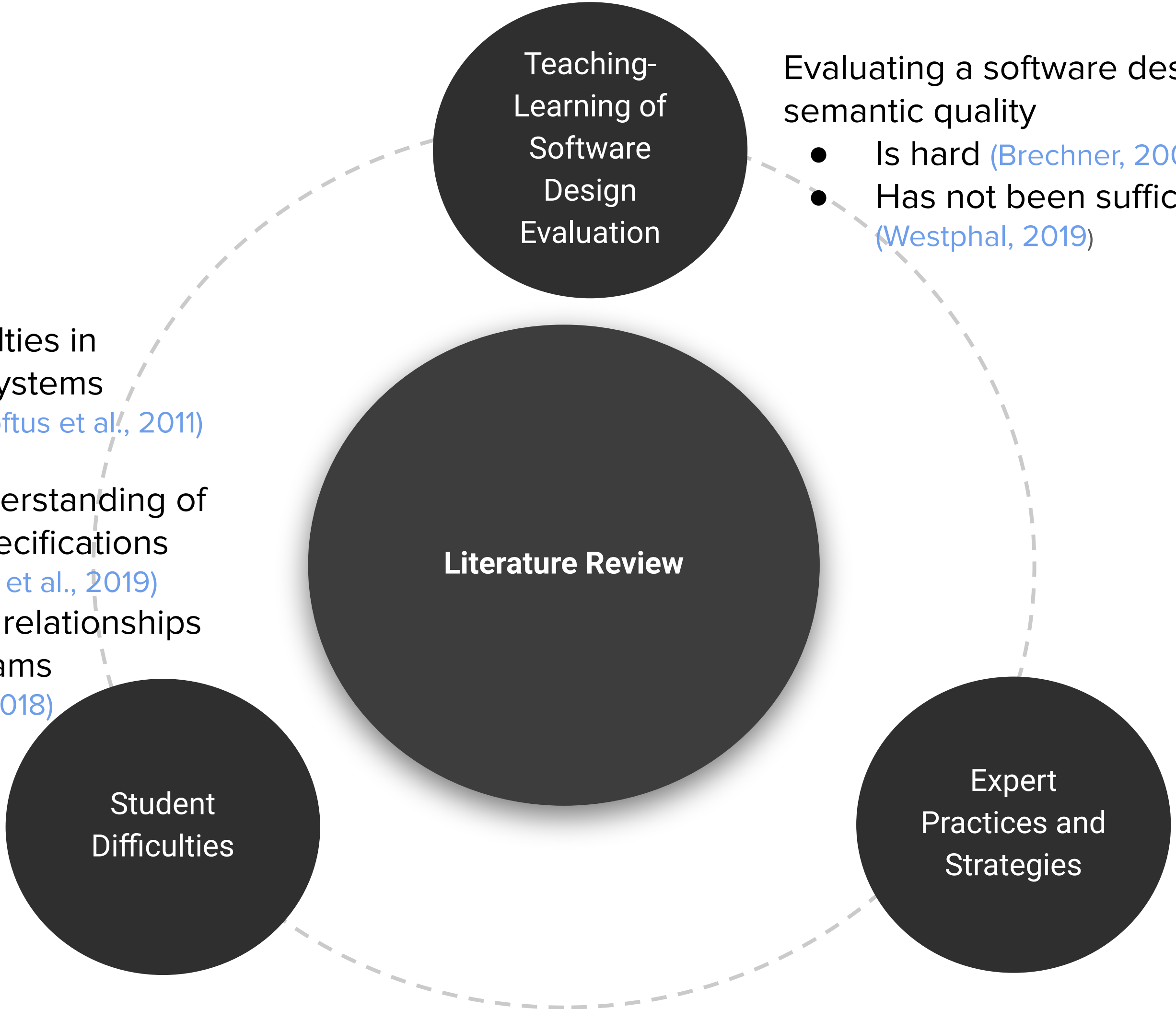
Expert Practices and Strategies

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)



Literature Review

Teaching-Learning of Software Design Evaluation

Evaluating a software design for its semantic quality

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

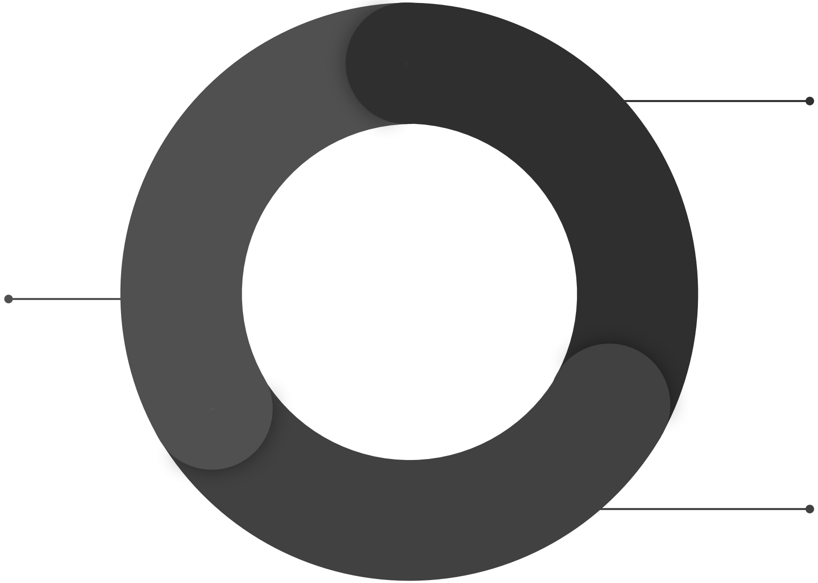
Student Difficulties

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

Expert Practices and Strategies

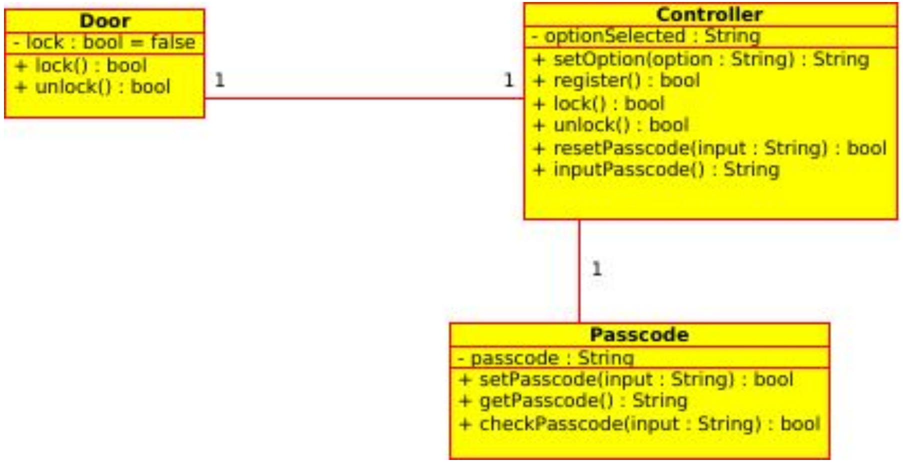


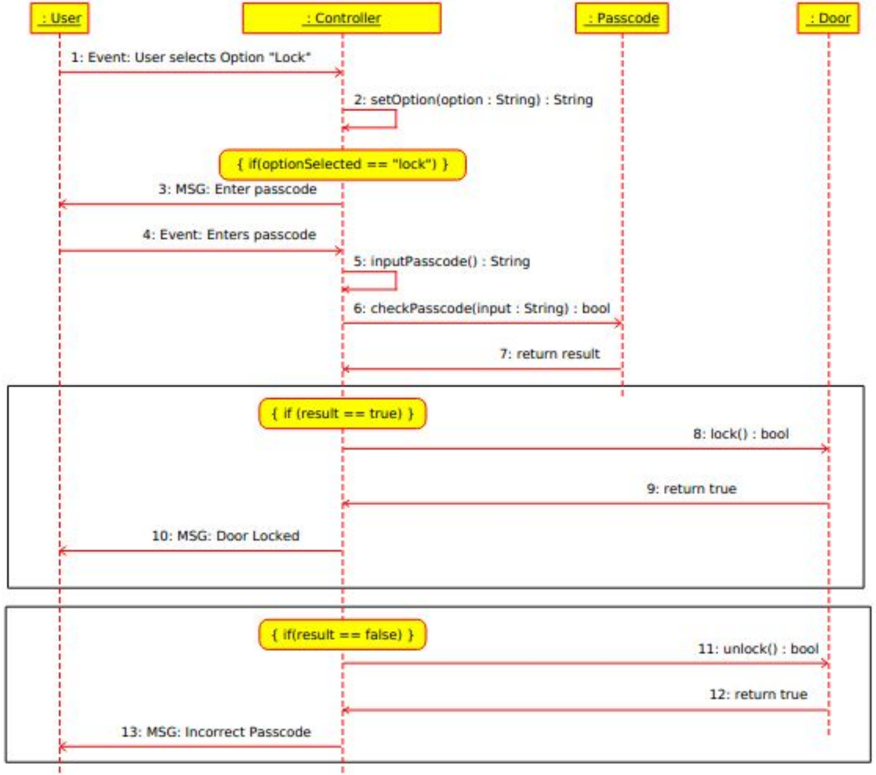
Experts create rich and detailed mental models of the design and requirements
(Adelson and Soloway, 1986;
Schumacher and Czerwinski, 1992)

Perform mental simulations on these models
(Gentner D, 1983)

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

What does the mental model of the software design look like?

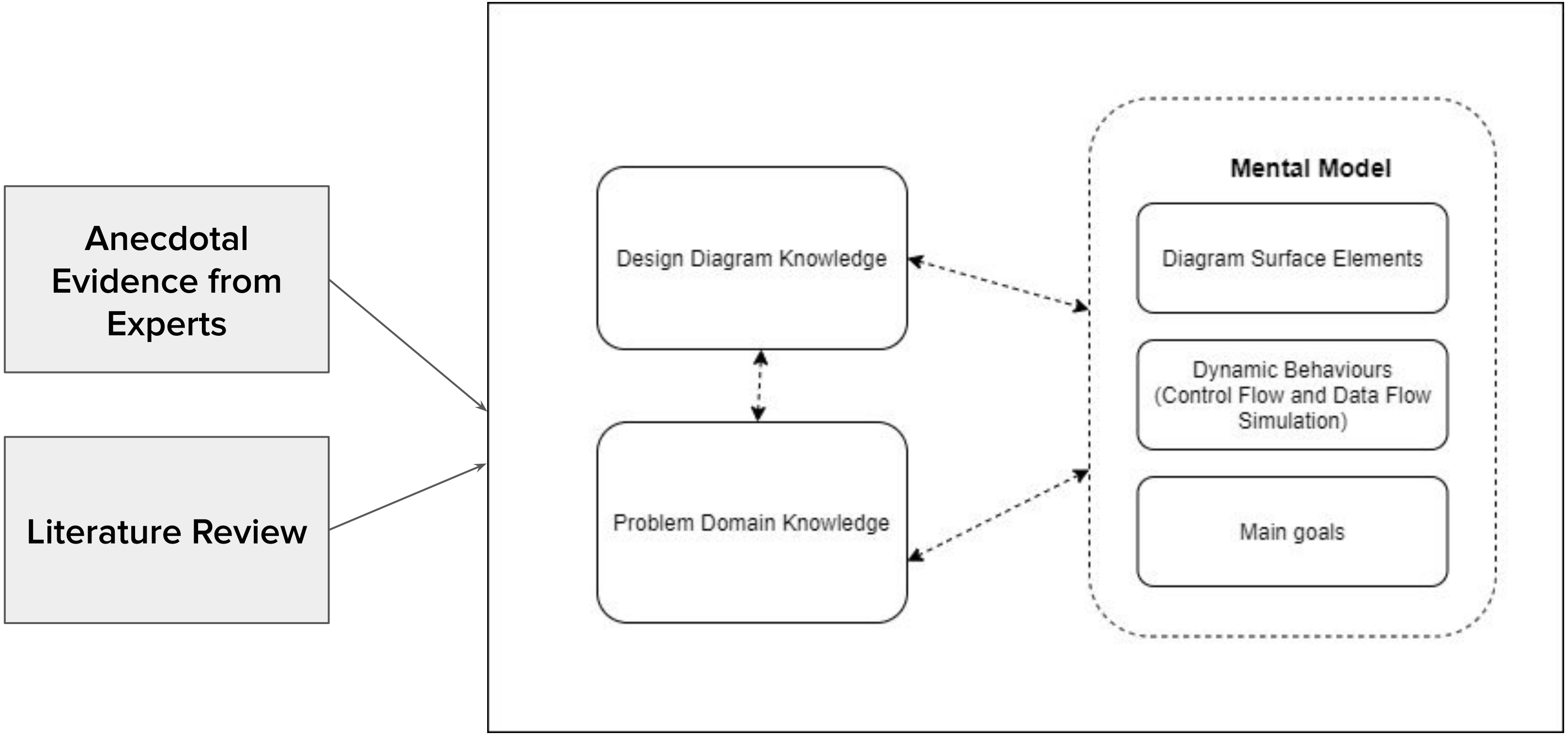

Door
- lock : bool = false
+ lock() : bool
+ unlock() : bool
1
1
Controller
- optionSelected : String
+ setOption(option : String) : String
+ register() : bool
+ lock() : bool
+ unlock() : bool
+ resetPasscode(input : String) : bool
+ inputPasscode() : String
1
Passcode
- passcode : String
+ setPasscode(input : String) : bool
+ getPasscode() : String
+ checkPasscode(input : String) : bool


: User
: Controller
: Passcode
: Door
1: Event: User selects Option "Lock"
2: setOption(option : String) : String
{ if(optionSelected == "lock") }
3: MSG: Enter passcode
4: Event: Enters passcode
5: inputPasscode() : String
6: checkPasscode(input : String) : bool
7: return result
{ if (result == true) }
8: lock() : bool
9: return true
10: MSG: Door Locked
{ if(result == false) }
11: unlock() : bool
12: return true
13: MSG: Incorrect Passcode

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

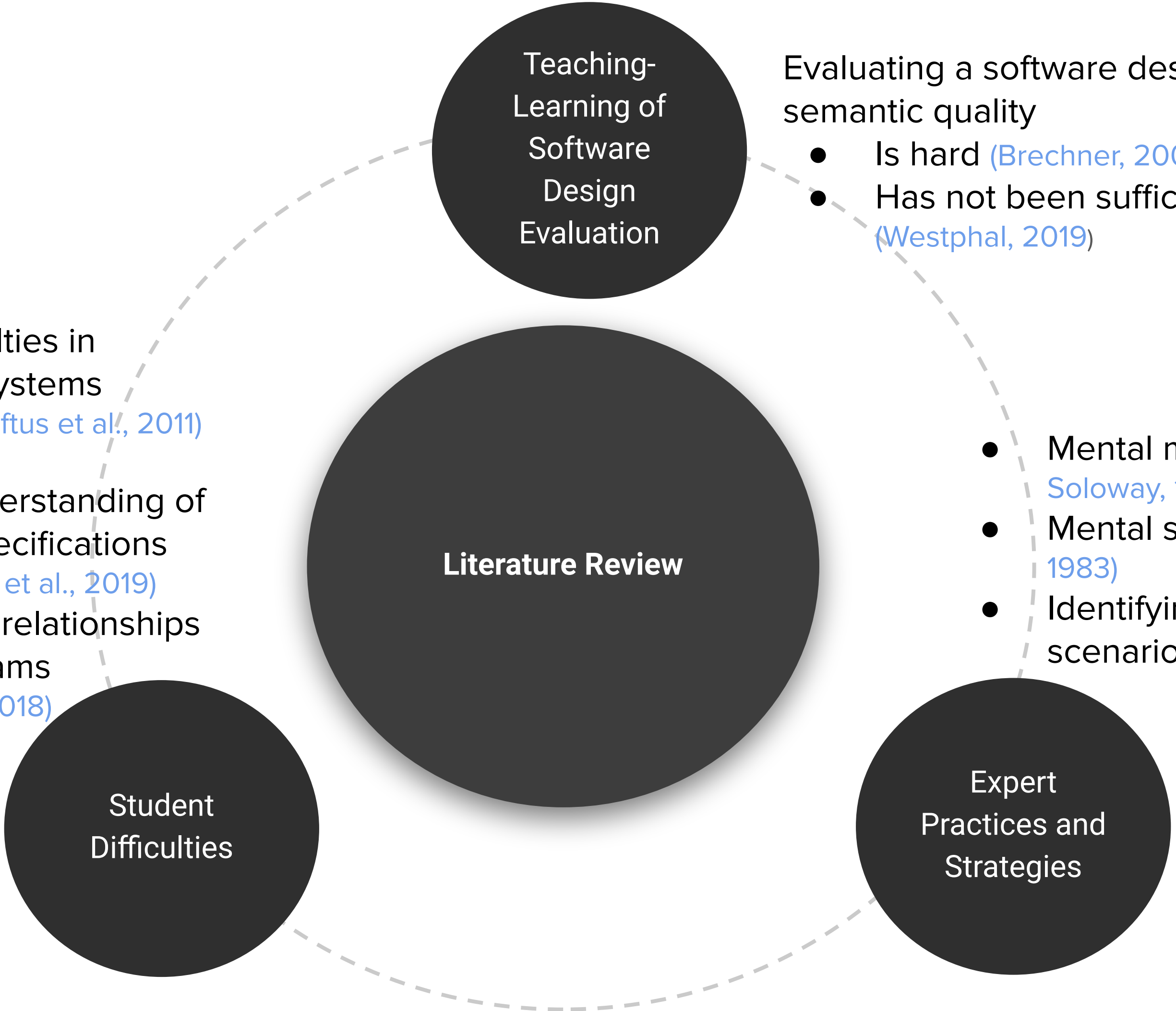
(Soloway and Ehrlich, 1984; Pennington, 1987; Von Mayrhauser and Vans, 1996)

Proposed Mental Model Elements for Design Diagrams



Anecdotal Evidence from Experts

Literature Review



Literature Review

Teaching-Learning of Software Design Evaluation

Evaluating a software design for its semantic quality

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

Student Difficulties

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

- 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

	Data Source	Data Analysis
RQ 1.1: How do students evaluate a software design against the given requirements?	Student response sheets	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

Add new functionality

Focus on **new elements** absent in the design

Novice Study - Study 1b

Qualitative Study - 6 computer engineering and information technology engineering students

	Data Source	Data Analysis
RQ 1.2: What defects are students able to identify in the design evaluation task?	Student responses on the task sheet	Student responses on the task sheet
RQ 1.3: What **reading strategies** do students use?	Video of students performing the task and screen capture	Thematic analysis of video data
RQ 1.4: What are the **elements in their mental model?**	Audio transcripts of the post-task interview	Thematic analysis of transcripts

More details

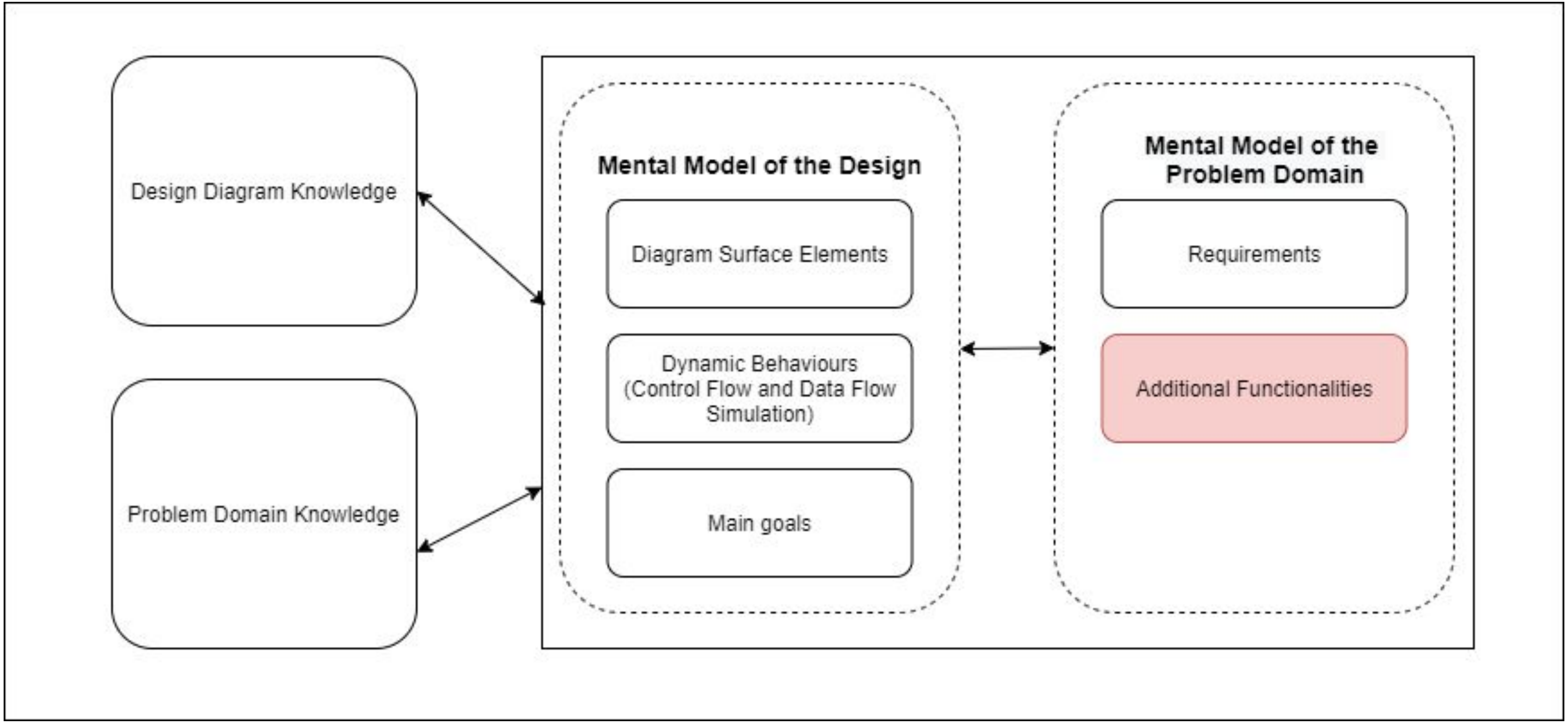
Study 1b - Findings

- 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



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

Requirements

Additional Functionalities

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

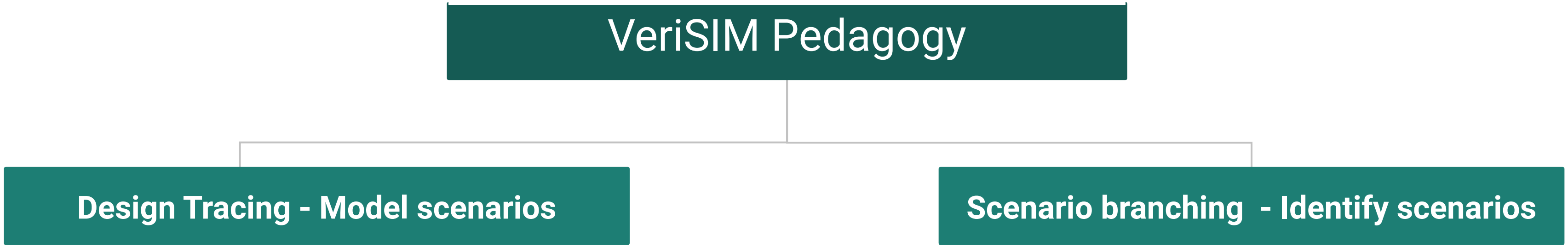
    Novice studies - Study 1a and 1b

    RQ 1: How do students evaluate a design against the given requirements?

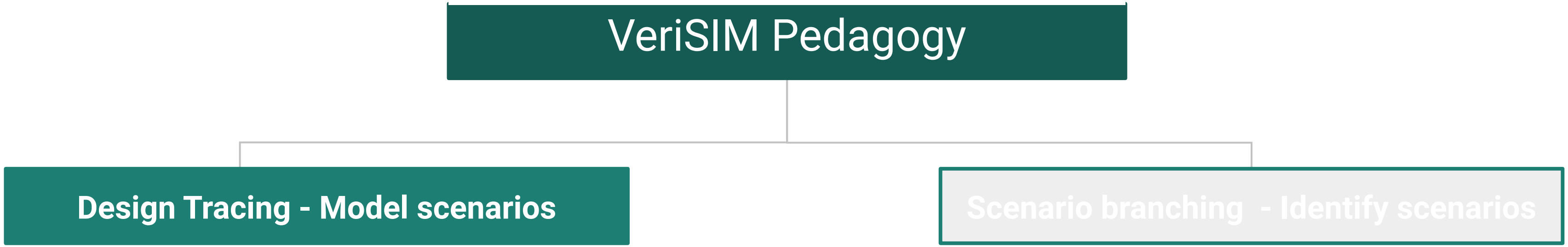
3. **Pedagogical strategies for effective software design evaluation**
    a. VeriSIM pedagogy
    b. Effectiveness Studies - Study 2 and 3
    c. How pedagogical features of VeriSIM are contributing towards learning

VeriSIM Pedagogy

Verifying Designs by **Sim**ulating Scenarios


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

VeriSIM Pedagogy



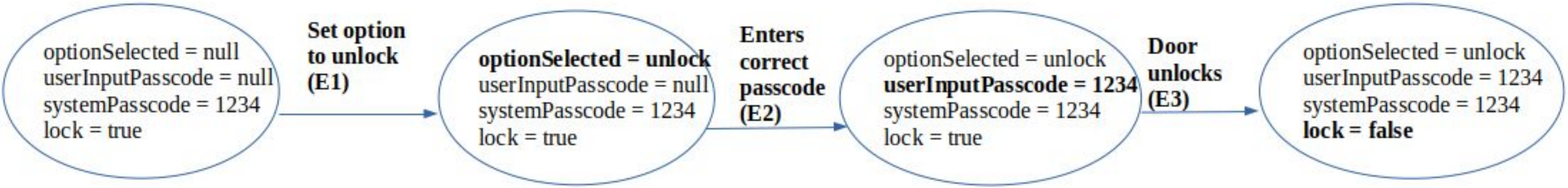
VeriSIM Pedagogy

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”



optionSelected = null
userInputPasscode = null
systemPasscode = 1234
lock = true

Set option to unlock (E1)

optionSelected = unlock
userInputPasscode = null
systemPasscode = 1234
lock = true

Enters correct passcode (E2)

optionSelected = unlock
userInputPasscode = 1234
systemPasscode = 1234
lock = true

Door unlocks (E3)

optionSelected = unlock
userInputPasscode = 1234
systemPasscode = 1234
lock = false

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:

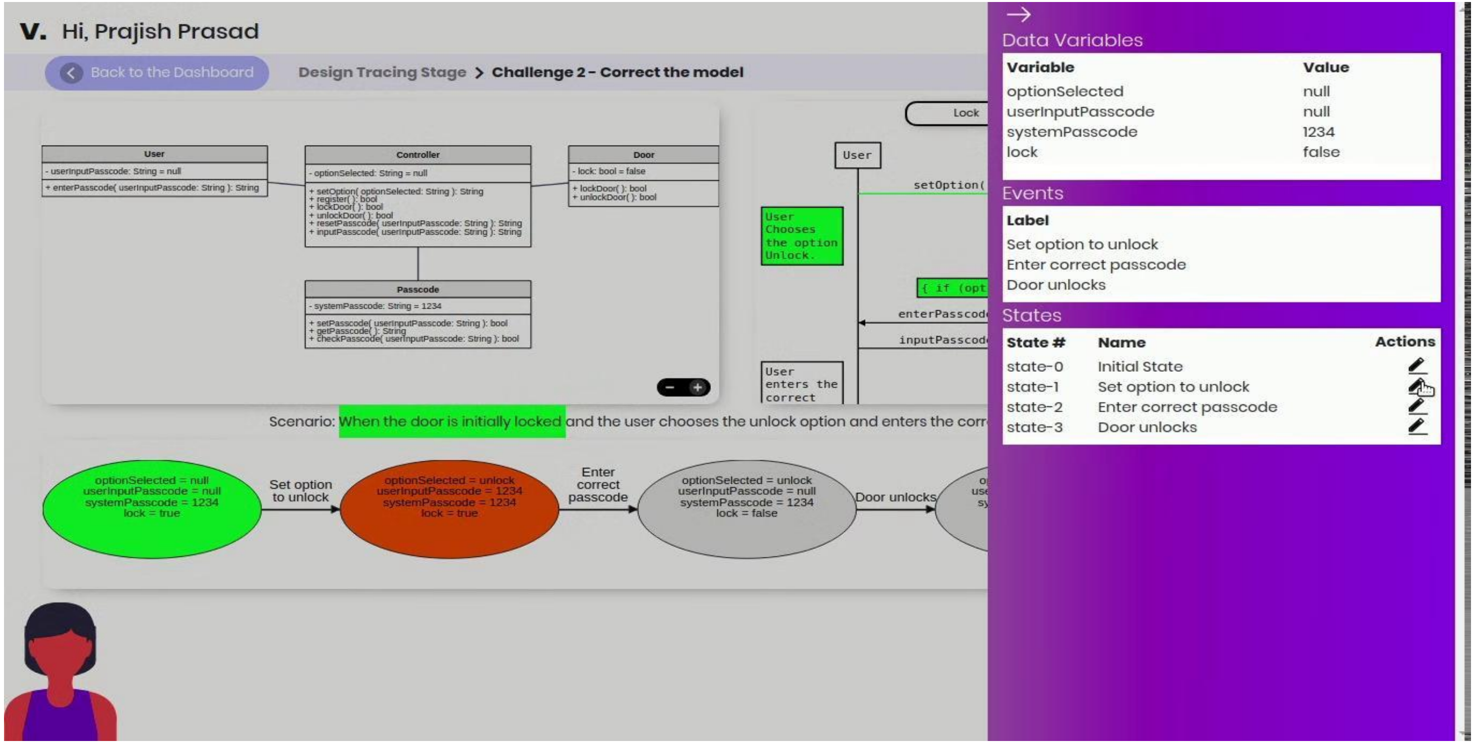
1. Explore the model
2. Correct the model
3. Complete the model
4. Construct the model

More details about VeriSIM here

Design Tracing Stage

Explore and Correct the Model

Model scenarios in the design using the design tracing pedagogy.



Hi, Prajish Prasad

Design Tracing Stage > **Challenge 2 - Correct the model**

Scenario: When the door is initially locked and the user chooses the unlock option and enters the corr

Data Variables

Variable	Value
optionSelected	null
userInputPasscode	null
systemPasscode	1234
lock	false

Events

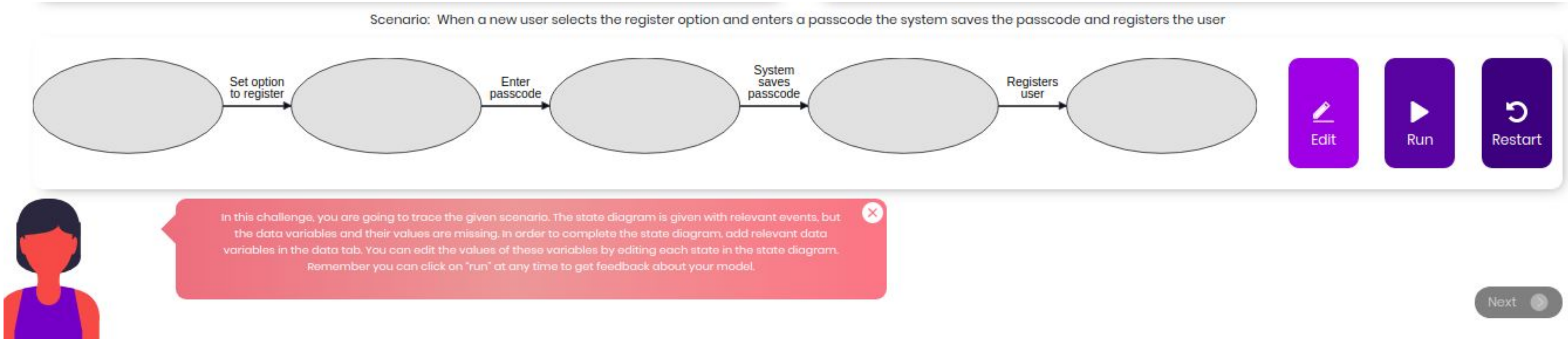
Label
Set option to unlock
Enter correct passcode
Door unlocks

States

State #	Name	Actions
state-0	Initial State	✎
state-1	Set option to unlock	✎
state-2	Enter correct passcode	✎
state-3	Door unlocks	✎

Watch the demo here

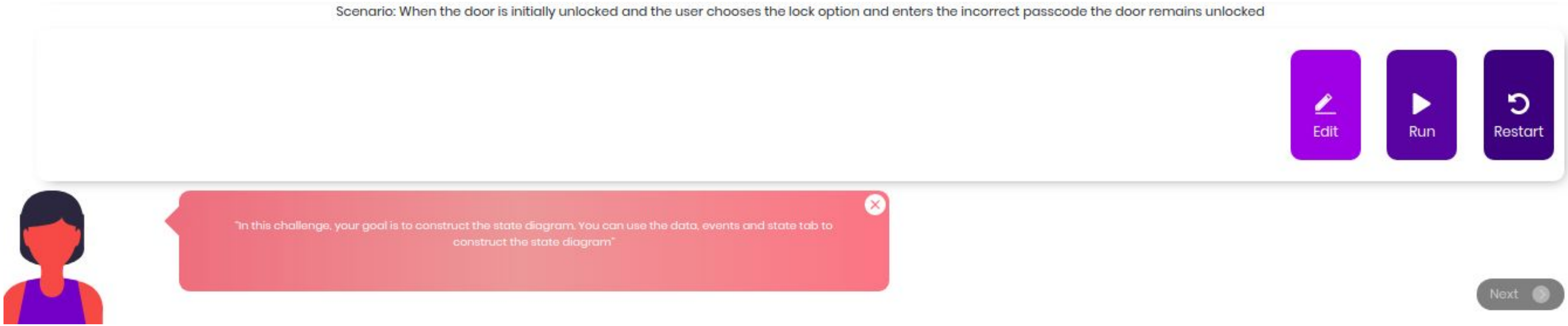
Design Tracing Stage- Challenge 3 - **Complete the Model**



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 data tab. You can edit 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.

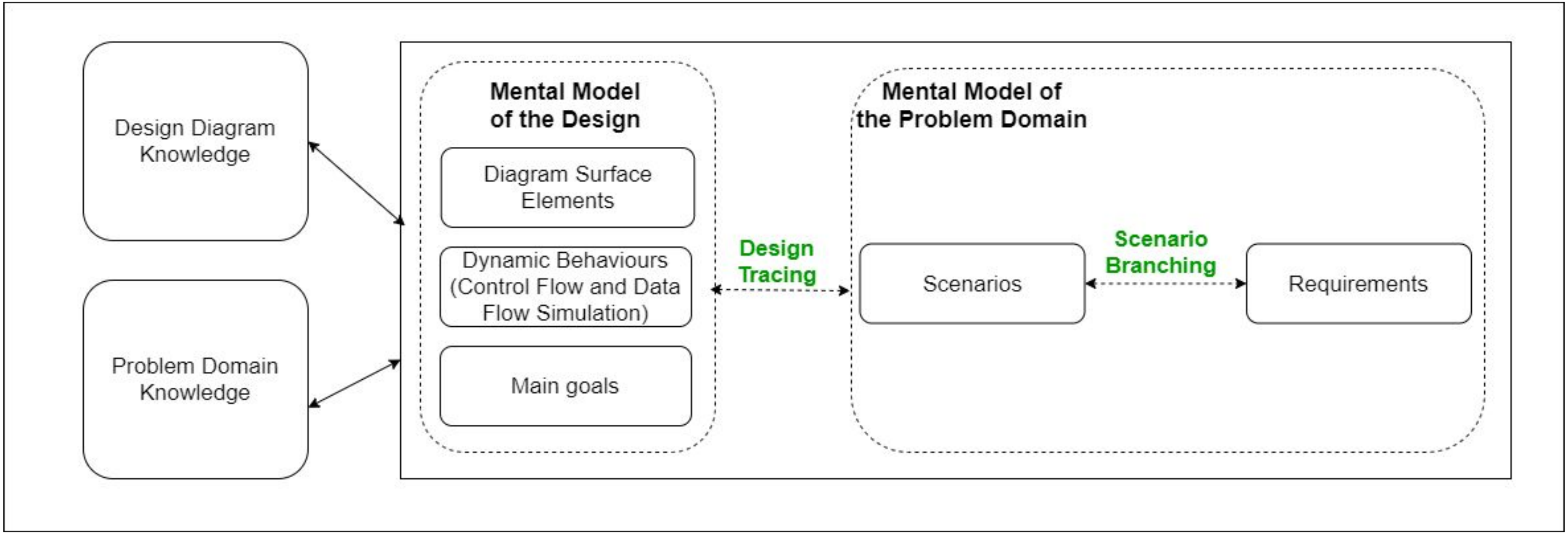
Design Tracing Stage- Challenge 4 - **Construct the Model**



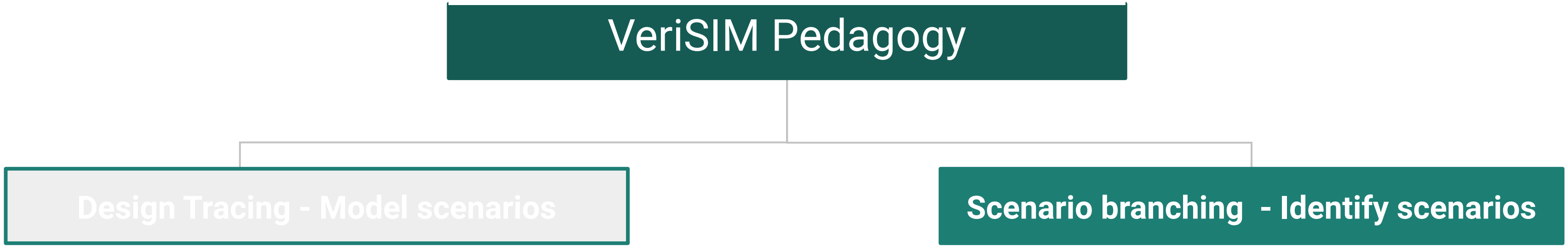
Scenario: When the door is initially unlocked and the user chooses the lock option and enters the incorrect passcode the door remains unlocked

"In this challenge, your goal is to construct the state diagram. You can use the data, events and state tab to construct the state diagram"

Connecting the Pedagogy to Mental Model Elements



VeriSIM Pedagogy: Scenario Branching Strategy



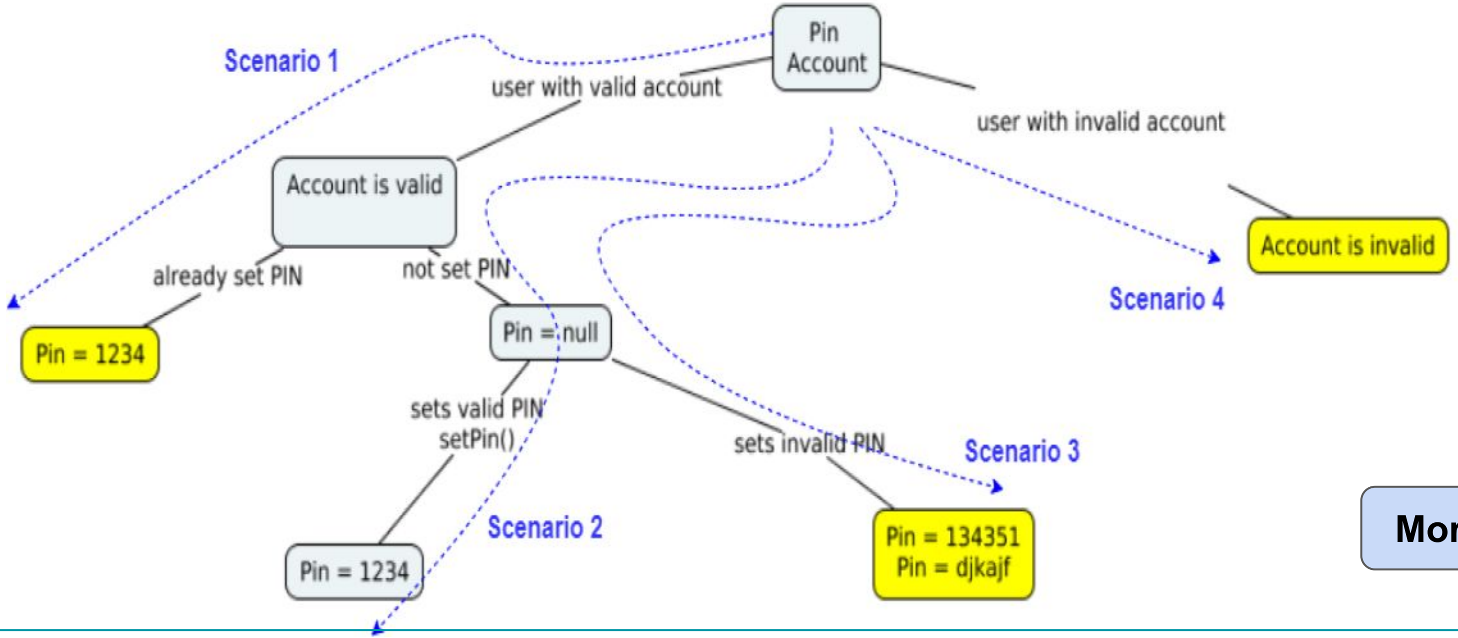
VeriSIM Pedagogy

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

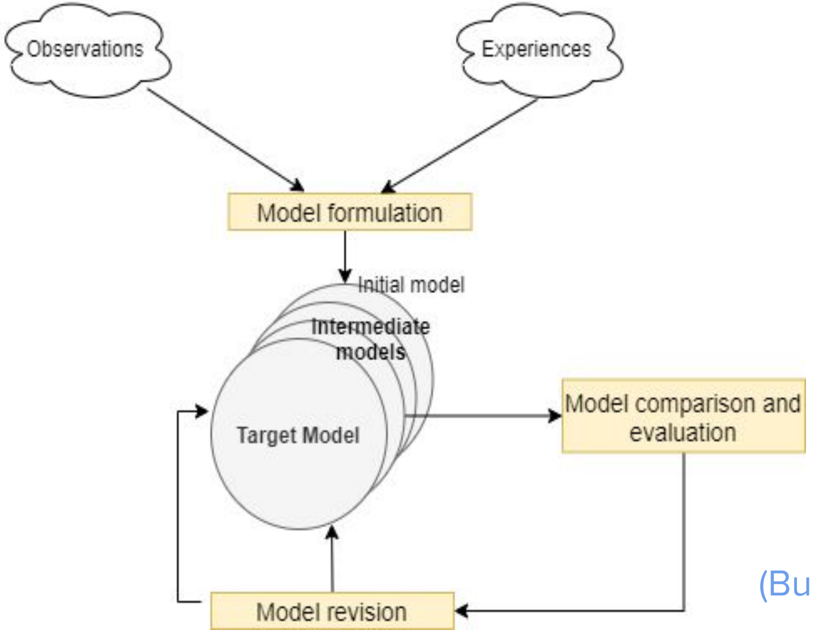


More details

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

Modelling Practices



Observations

Experiences

Model formulation

Initial model

Intermediate models

Target Model

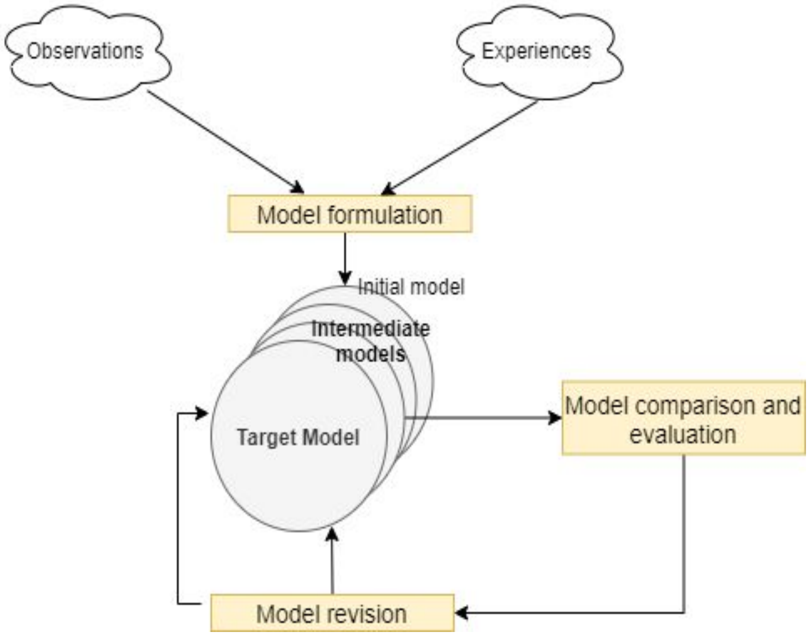
Model comparison and evaluation

Model revision

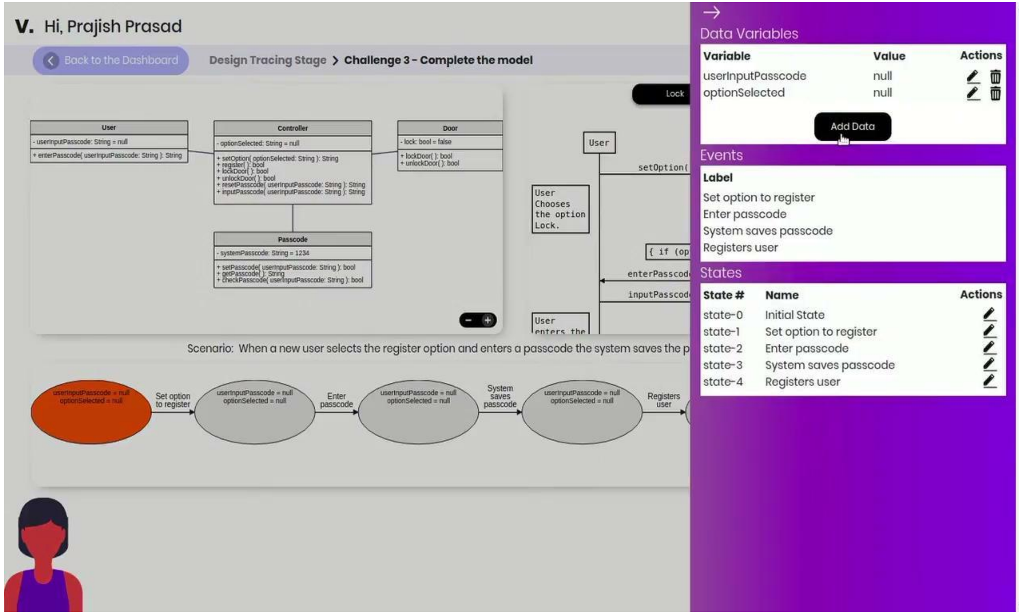
(Buckley et al., 2010)

Theoretical Basis: Model-based Learning

Modelling Practices



Design Tracing - Model scenarios


V.
Back to the Dashboard
Lock
User
setOption(
User Chooses the option Lock.
{ if (op
enterPasscod
inputPasscod
User enters the
userInputPasscode = null optionSelected = null
Set option to register
userInputPasscode = null optionSelected = null
Enter passcode
userInputPasscode = null optionSelected = null
System saves passcode
userInputPasscode = null optionSelected = null
Registers user
Add Data

Hi, Prajish Prasad

Design Tracing Stage > Challenge 3 - Complete the model

```
User
- userInputPasscode: String = null
+ enterPasscode( userInputPasscode: String ): String

Controller
- optionSelected: String = null
+ setOption( optionSelected: String ): String
+ register( ): bool
+ lockDoor( ): bool
+ unlockDoor( ): bool
+ resetPasscode( userInputPasscode: String ): String
+ inputPasscode( userInputPasscode: String ): String

Door
- lock: bool = false
+ lockDoor( ): bool
+ unlockDoor( ): bool

Passcode
- systemPasscode: String = 1234
+ setPasscode( userInputPasscode: String ): bool
+ getPasscode( ): String
+ checkPasscode( userInputPasscode: String ): bool
```

Scenario: When a new user selects the register option and enters a passcode the system saves the p

Data Variables

Variable	Value	Actions
userInputPasscode	null	
optionSelected	null	

Events

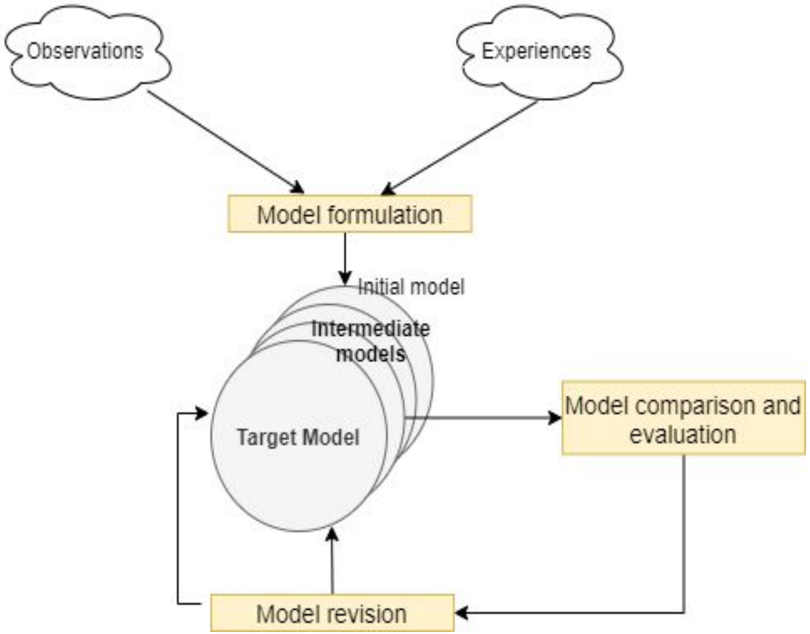
Label
Set option to register
Enter passcode
System saves passcode
Registers user

States

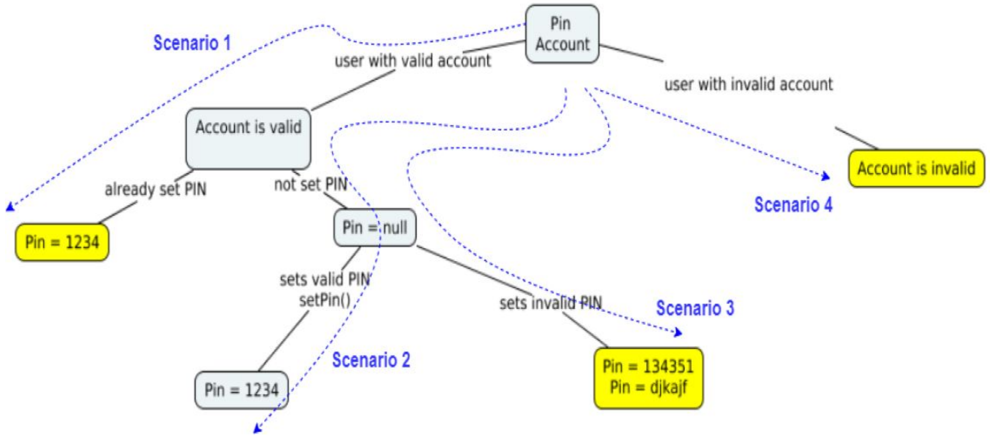
State #	Name	Actions
state-0	Initial State	
state-1	Set option to register	
state-2	Enter passcode	
state-3	System saves passcode	
state-4	Registers user	

Theoretical Basis: Model-based Learning

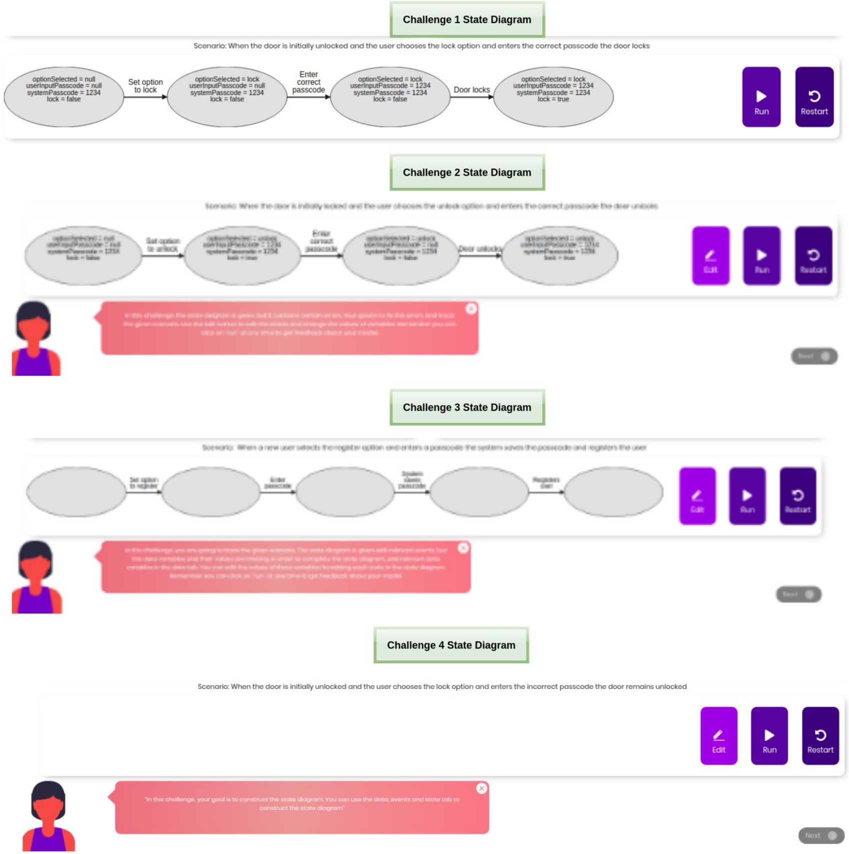
Modelling Practices



Scenario branching - Identify scenarios



Pedagogical Features: Model Progression



Challenge 1 State Diagram

Scenario: When the door is initially unlocked and the user chooses the lock option and enters the correct passcode the door locks

Challenge 2 State Diagram

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

In this challenge, the state diagram is given, but it contains certain errors. Your goal is to fix the errors and trace the given scenario. Use the edit button to edit the states and change the values of variables. Remember you can click on "run" at any time to get feedback about your model.

Challenge 3 State Diagram

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 data tab. You can edit the values of those variables by editing each state in the state diagram. Remember you can click on "run" at any time to get feedback about your model.

Challenge 4 State Diagram

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

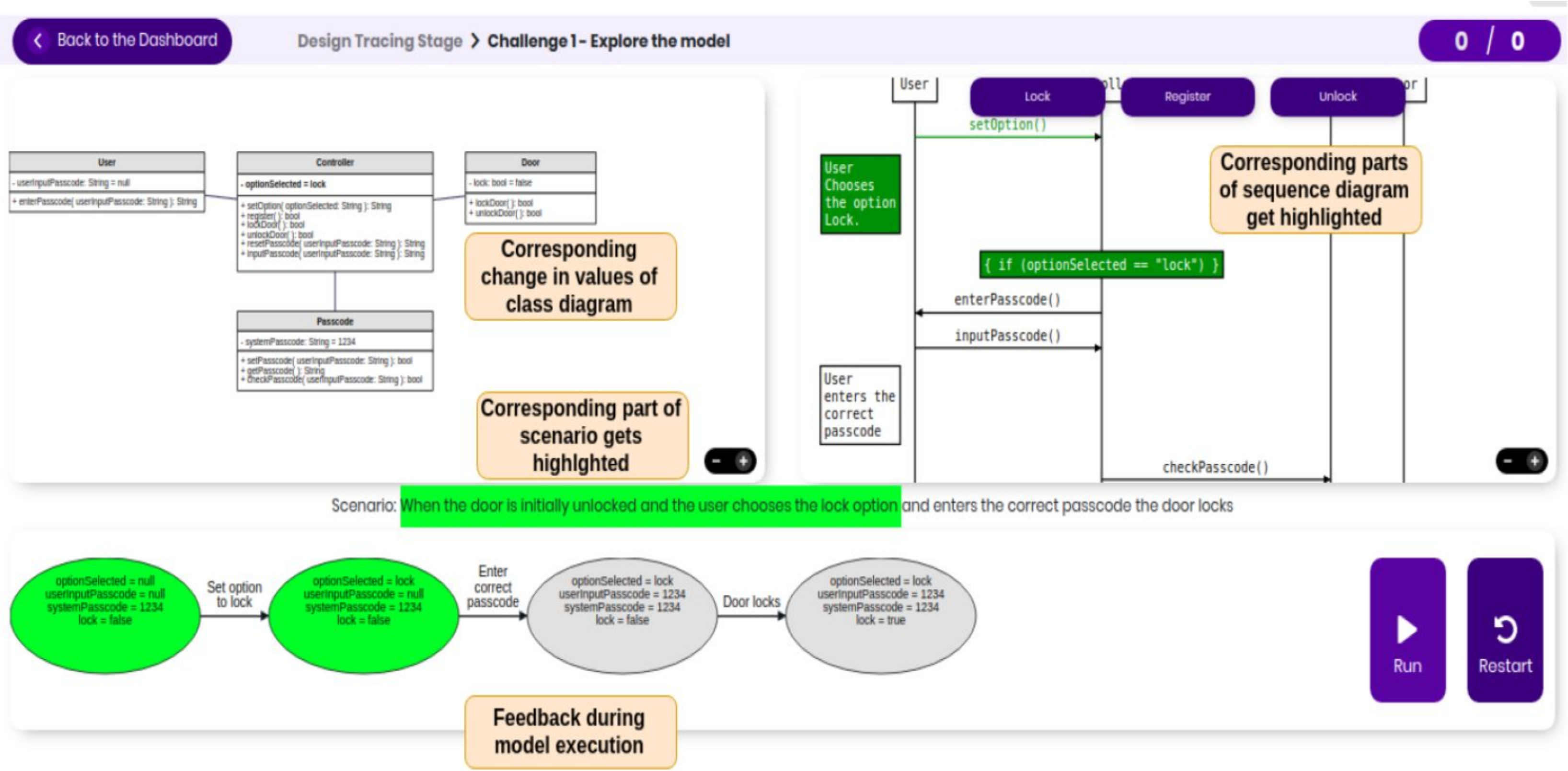
"In this challenge, your goal is to construct the state diagram. You can use the data, events and state tab to construct the state diagram"

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



Scenario: When the door is initially unlocked and the user chooses the lock option and enters the correct passcode the door locks

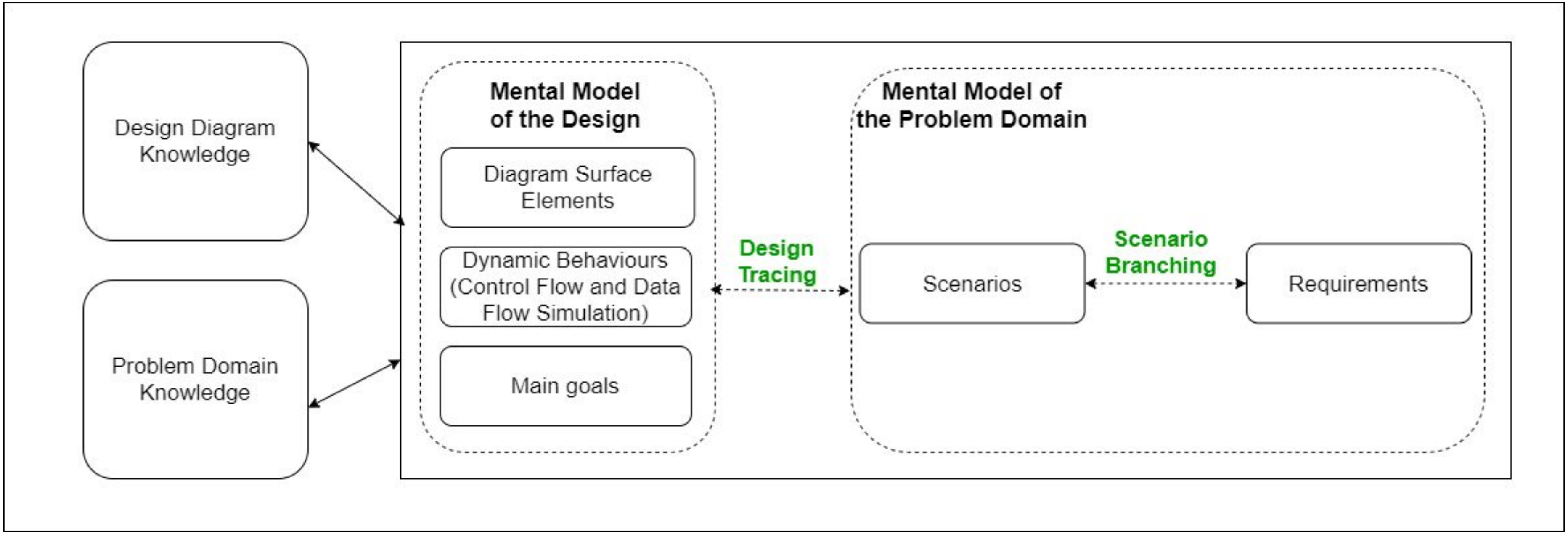
Corresponding change in values of class diagram

Corresponding parts of sequence diagram get highlighted

Corresponding part of scenario gets highlighted

Feedback during model execution

Connecting the Pedagogy to Mental Model Elements



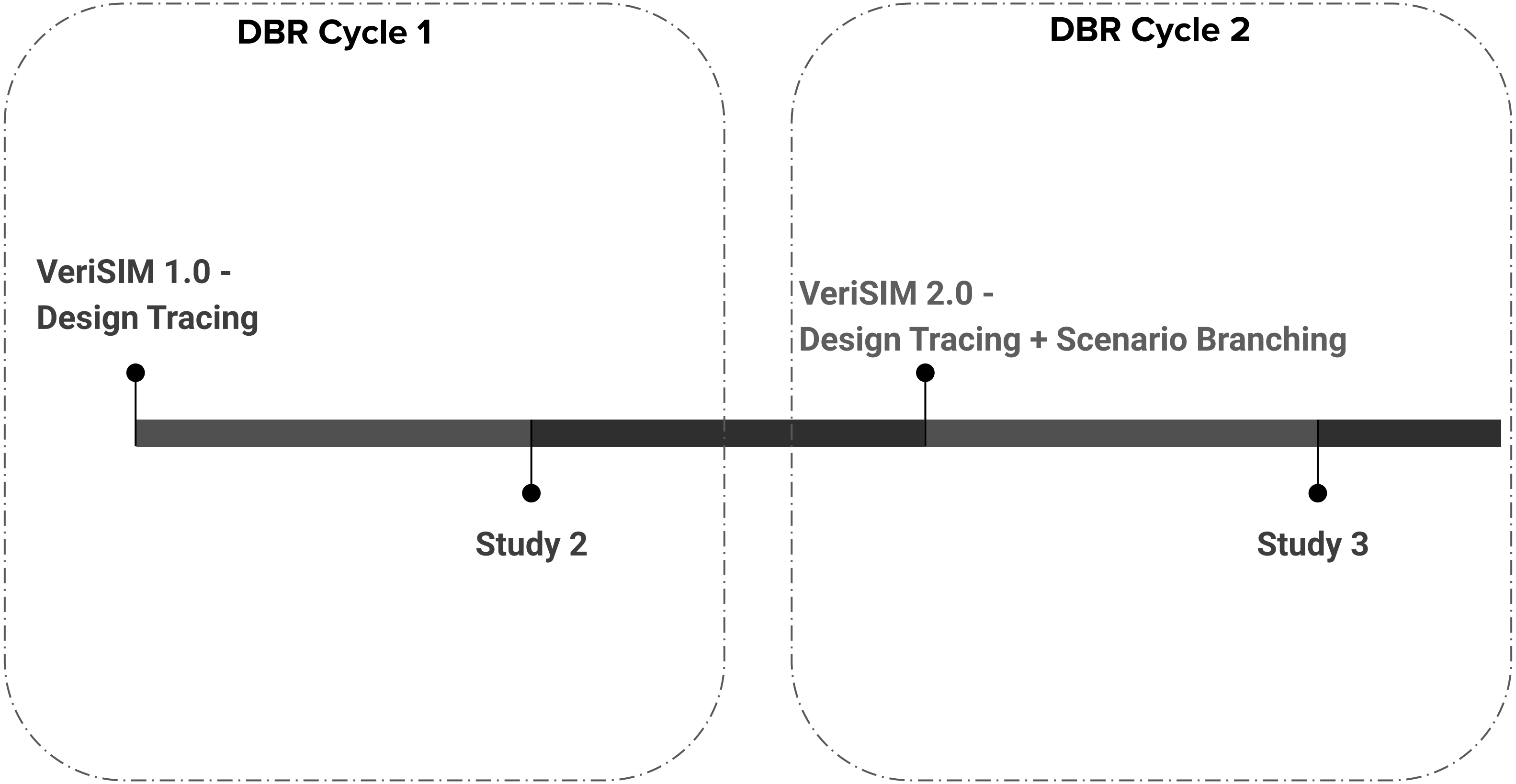
Key Questions Answered in this Thesis

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

DBR Cycle 2

VeriSIM 1.0 -
Design Tracing

VeriSIM 2.0 -
Design Tracing + Scenario Branching

Study 2

Study 3

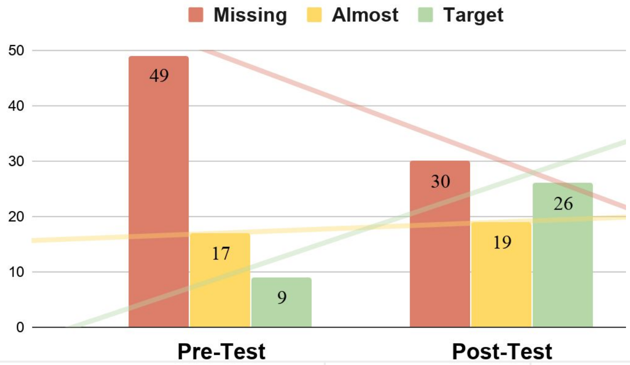
Study 2

	Data Source	Data Analysis
RQ 2.1 Does VeriSIM improve learners ability to **model a given scenario?**	Question in pre-test and post-test: **Explain the changes in the system on execution of this scenario**	Differences in pre-test and post-test question based on rubric
RQ 2.2 Does VeriSIM improve learners ability to **uncover defects?**	Question in pre-test and post-test: **Uncover defects in design diagrams**	Content analysis of “uncover defects” question in the pre-test and post-test

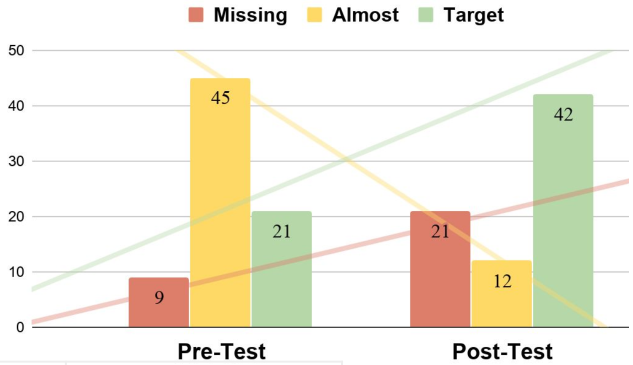
More details

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

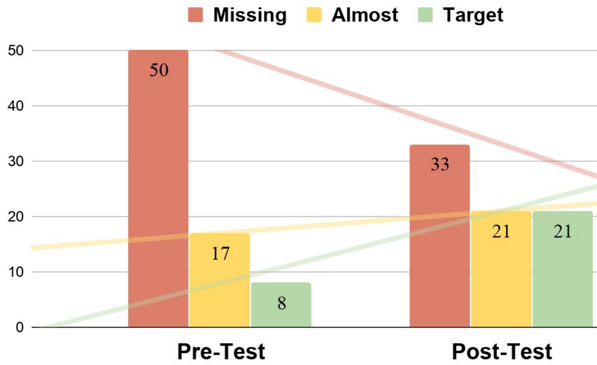
Data criteria



Event criteria



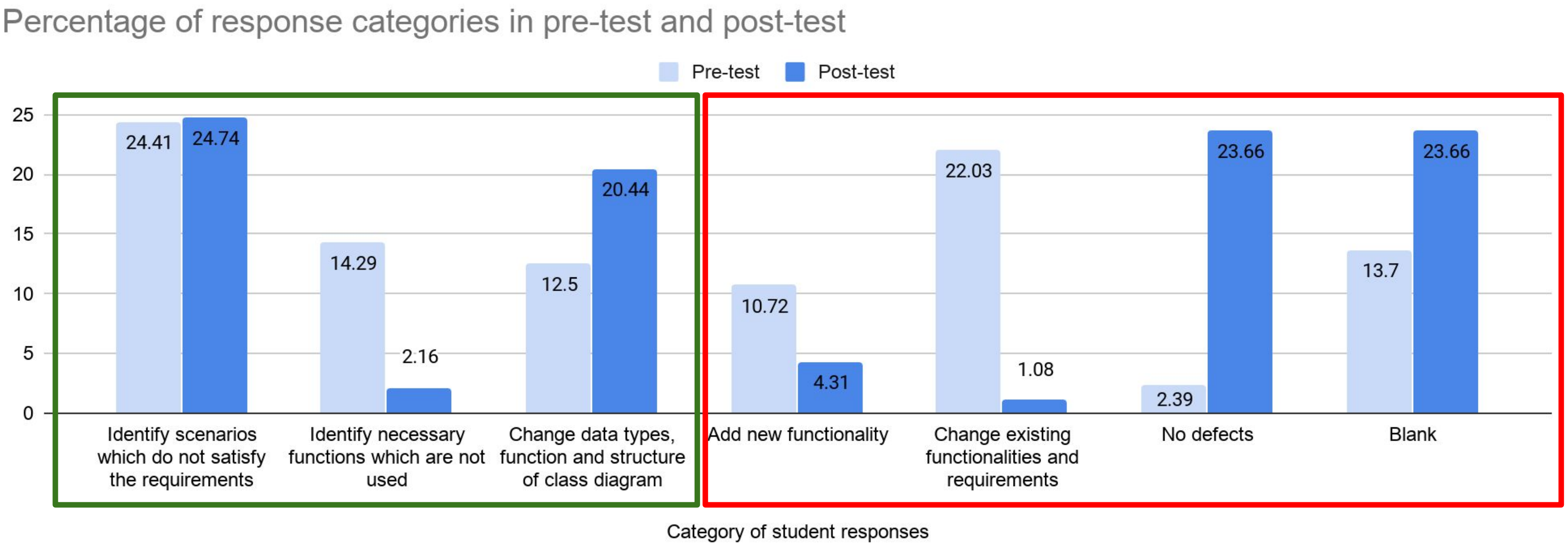
State change criteria



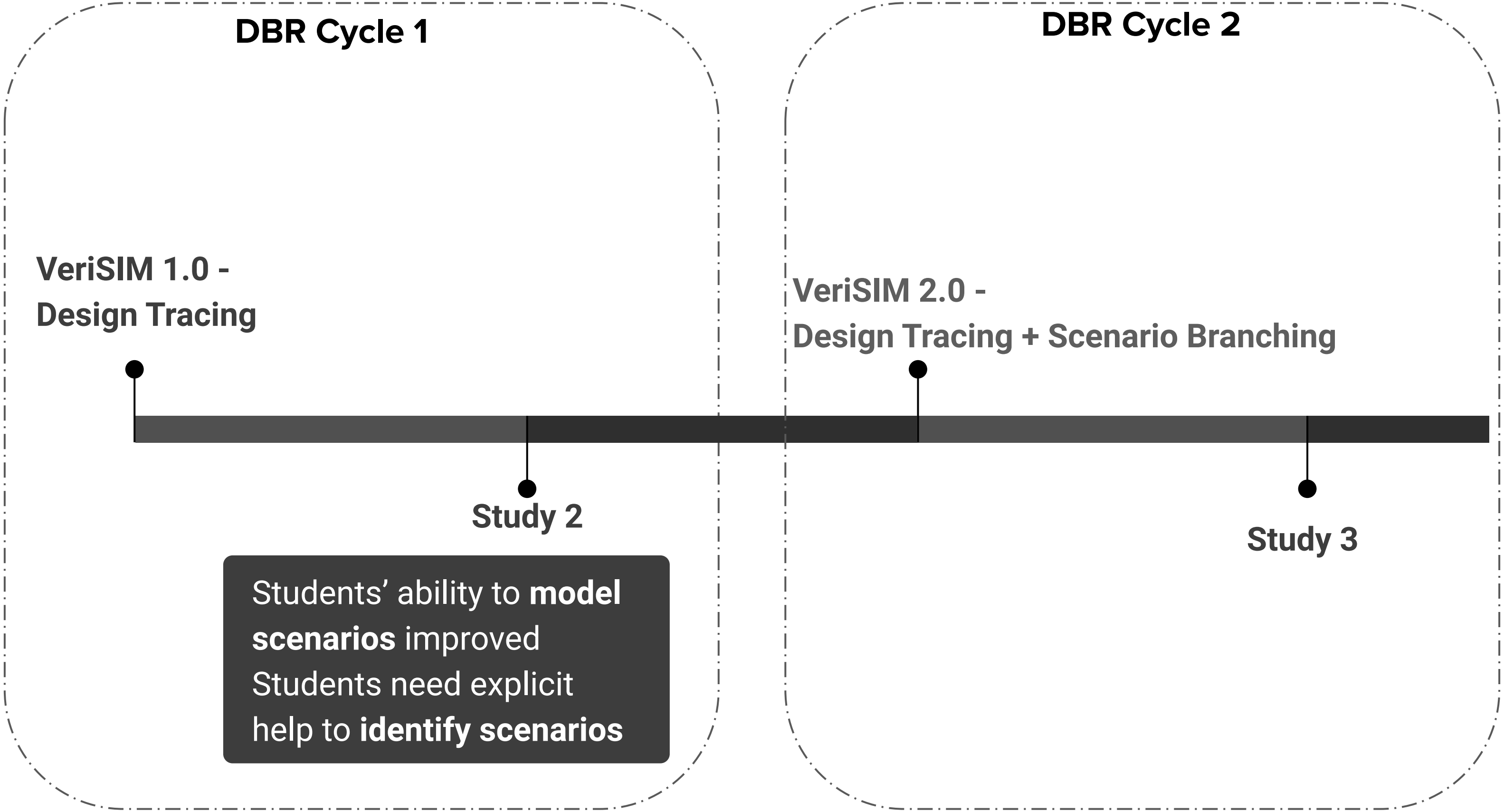
	Pre-test Mean (SD)	Post-test Mean (SD)	Paired t-test (p value)
Identifying relevant data variables	0.47(0.70)	0.95(0.87)	0.00
Identifying relevant events	1.16(0.62)	1.28(0.88)	0.17
Simulating state change	0.44(0.68)	0.84(0.84)	0.00
Total	2.07(1.70)	3.07(2.09)	0.00

Statistically significant **improvement in students' ability to model scenarios**

Study 2: Results - RQ 2.2: Ability to uncover defects



No difference in ability to **identify scenarios not satisfying the requirement**



DBR Cycle 1

DBR Cycle 2

VeriSIM 1.0 -
Design Tracing

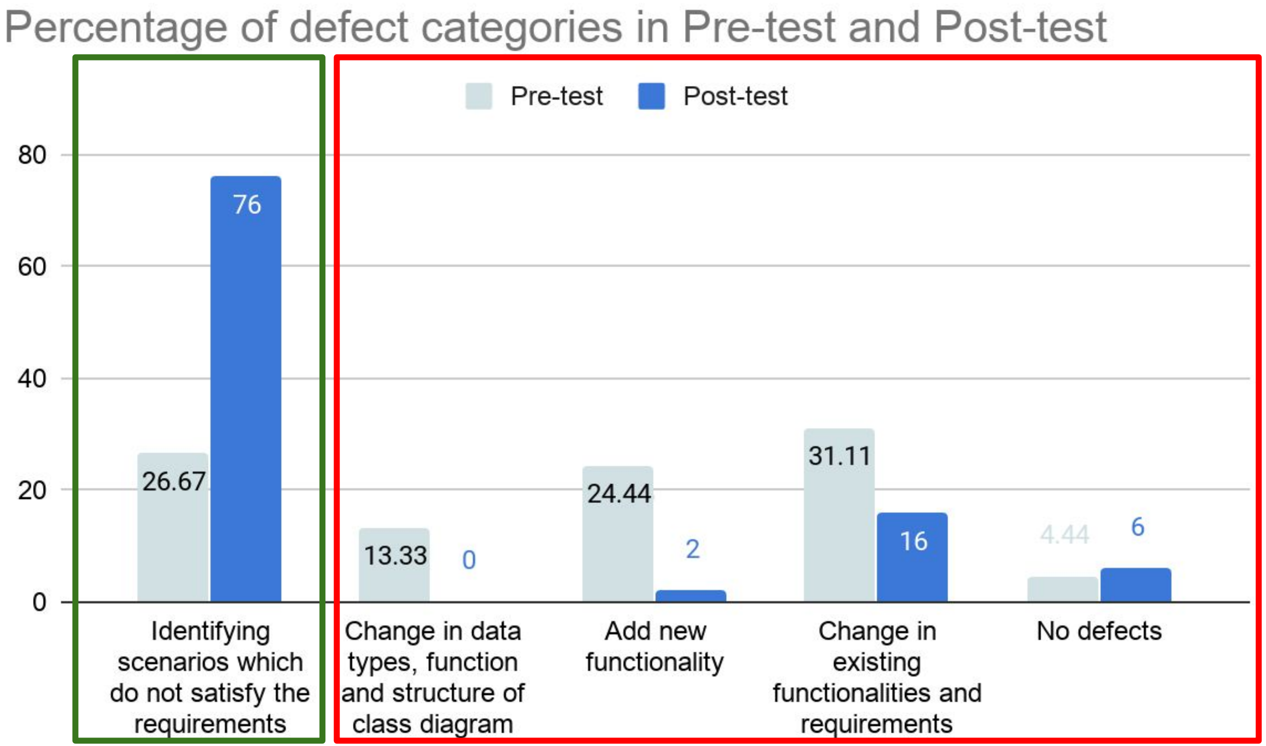
VeriSIM 2.0 -
Design Tracing + Scenario Branching

Study 2

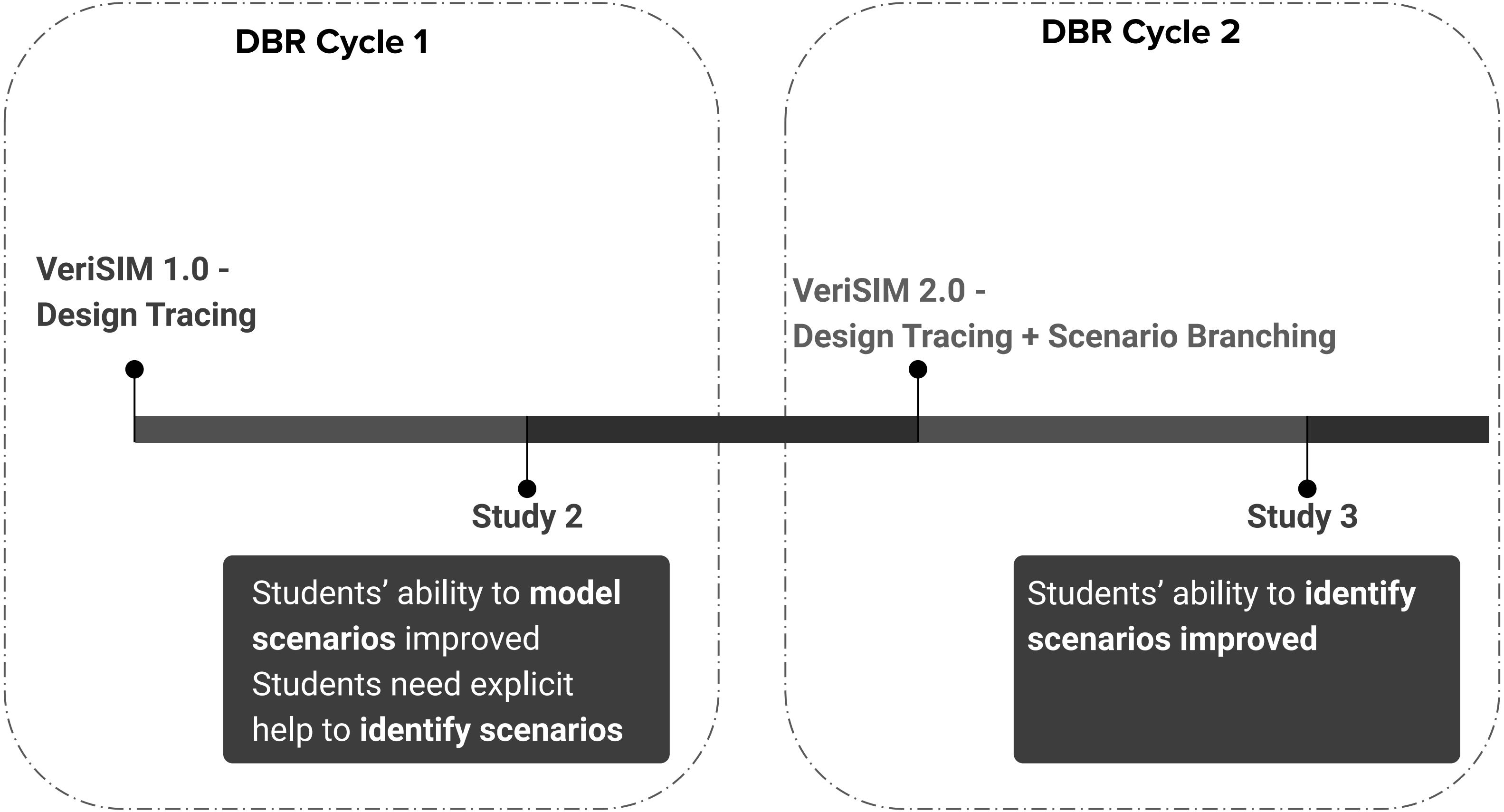
Study 3

Students' ability to **model scenarios** improved
Students need explicit help to **identify scenarios**

Study 3: Results - RQ 3.2: Identify defects



More details



DBR Cycle 1

DBR Cycle 2

VeriSIM 1.0 -
Design Tracing

VeriSIM 2.0 -
Design Tracing + Scenario Branching

Study 2

Students' ability to **model scenarios** improved
Students need explicit help to **identify scenarios**

Study 3

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

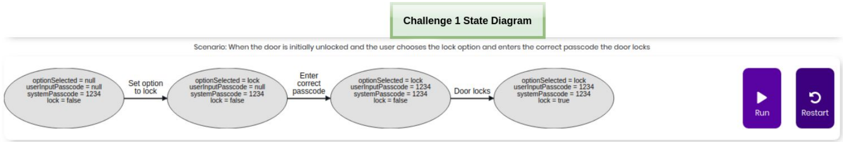
3. **Pedagogical strategies for effective software design evaluation**
    a. VeriSIM pedagogy
    b. Effectiveness Studies - Study 2 and 3
    RQ 2 and RQ 3: What are effects of the VeriSIM pedagogy in students' ability to evaluate a design against the given requirements?
    c. Pedagogical features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

RQ 4: How are features in VeriSIM contributing towards student learning?

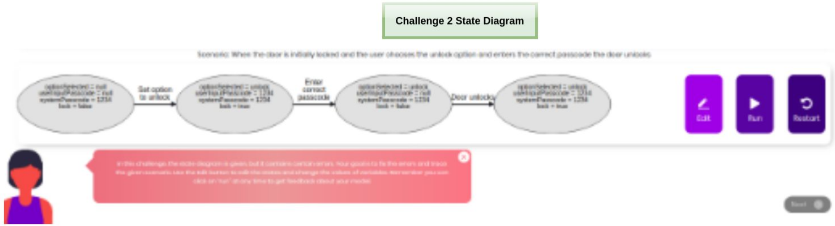
- Key Features in VeriSIM:
  - Model progression of Challenges
  - Model execution visualization (Run)
  - Scenario branching

- Data Sources -
  - Interaction Logs - 48 students who gave consent (Study 2 and 3)
  - Focus group interviews

Model Progression of Challenges

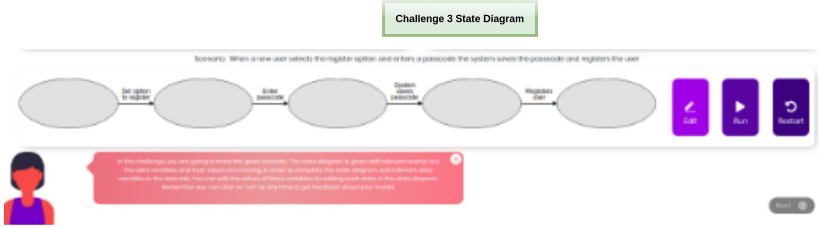


Scenario: When the door is initially unlocked and the user chooses the lock option and enters the correct passcode the door locks



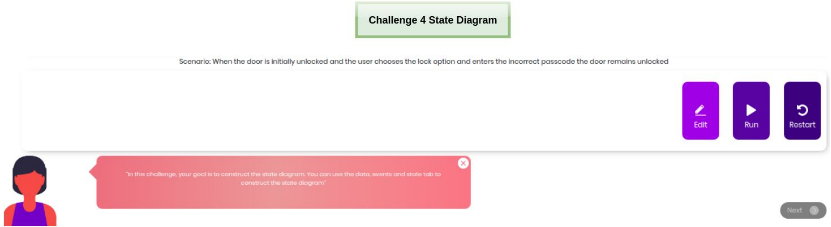
Scenario: When the door is initially locked and the user chooses the unlock option and enters the correct passcode the door unlocks

In this challenge the state diagram is given but it contains certain errors. Your goal is to fix the errors and trace the given scenario. Use the Edit button to edit the states and change the values of variables. Remember you can click on 'Run' at any time to get feedback about your 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 data tab. You can edit 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.

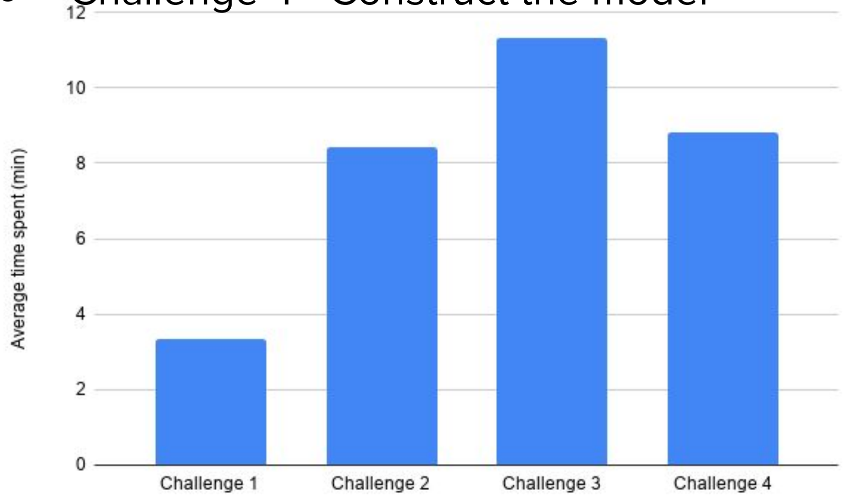


Scenario: When the door is initially unlocked and the user chooses the lock option and enters the incorrect passcode the door remains unlocked

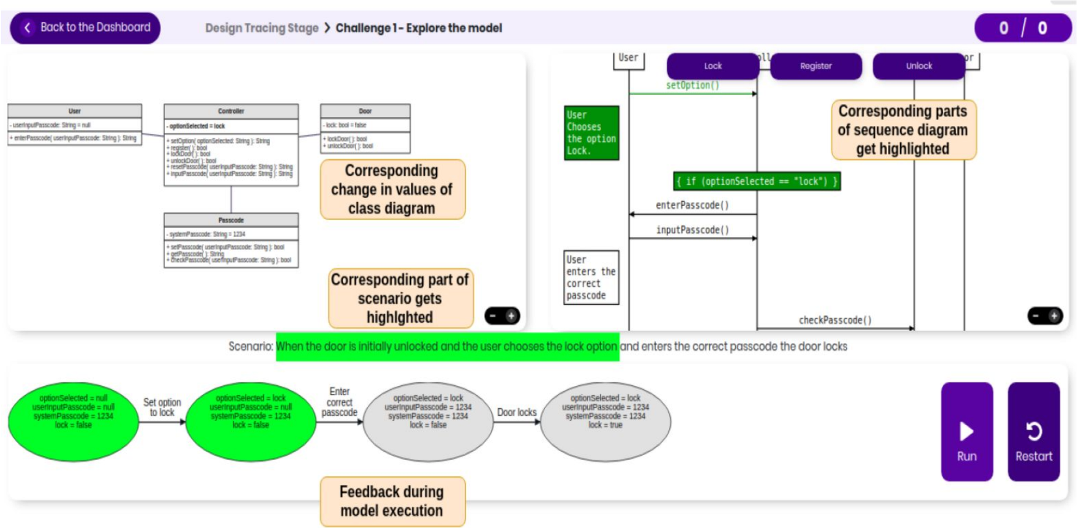
"In this challenge, your goal is to construct the state diagram. You can use the data, events and state tab to construct the state diagram"

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



Scenario: When the door is initially unlocked and the user chooses the lock option and enters the correct passcode the door locks

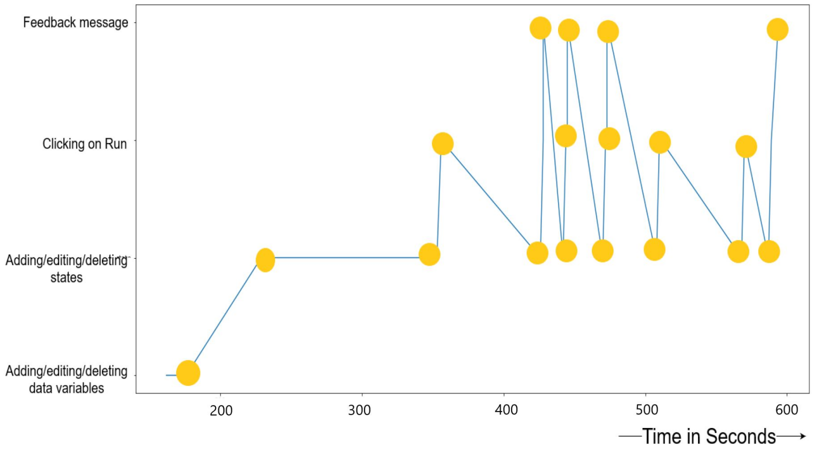
Corresponding change in values of class diagram

Corresponding parts of sequence diagram get highlighted

Corresponding part of scenario gets highlighted

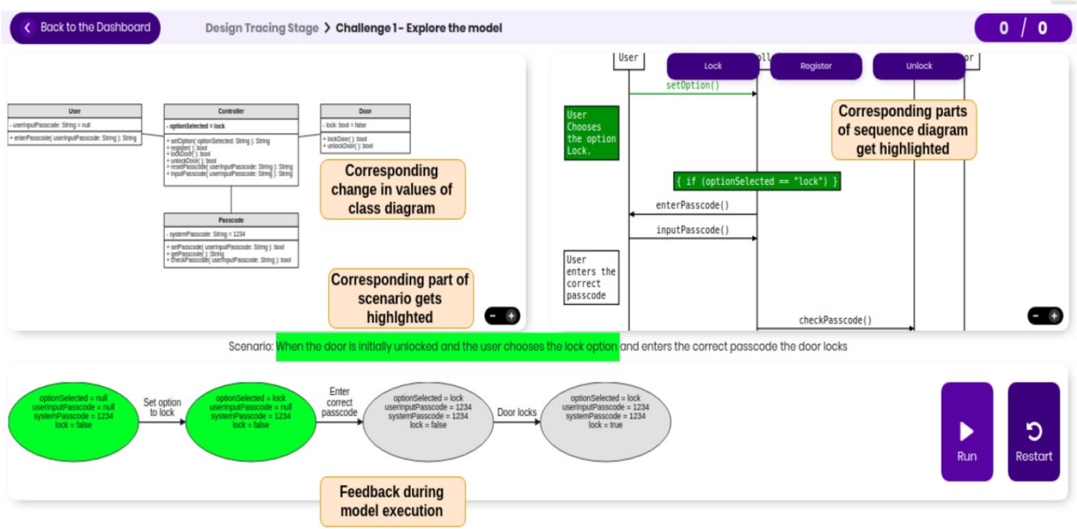
Feedback during model execution

From interaction logs:



Students use the model execution visualization feature while modelling scenarios

Model Execution Visualization



Scenario: When the door is initially unlocked and the user chooses the lock option and enters the correct passcode the door locks

Corresponding change in values of class diagram

Corresponding part of scenario gets highlighted

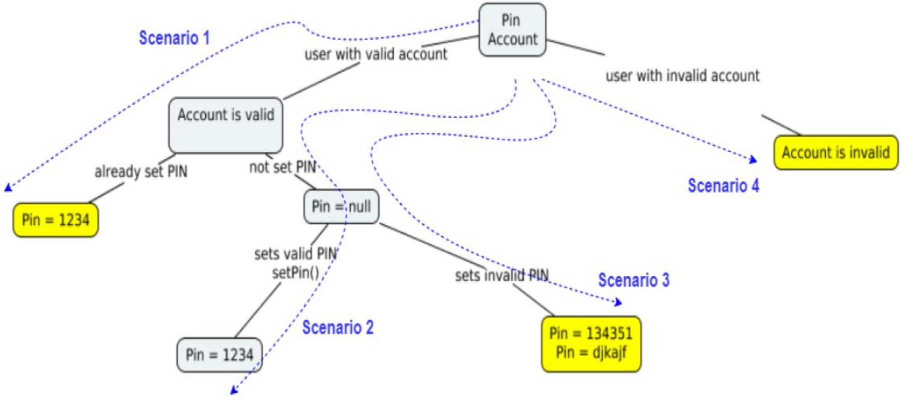
Corresponding parts of sequence diagram get highlighted

Feedback during model execution

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

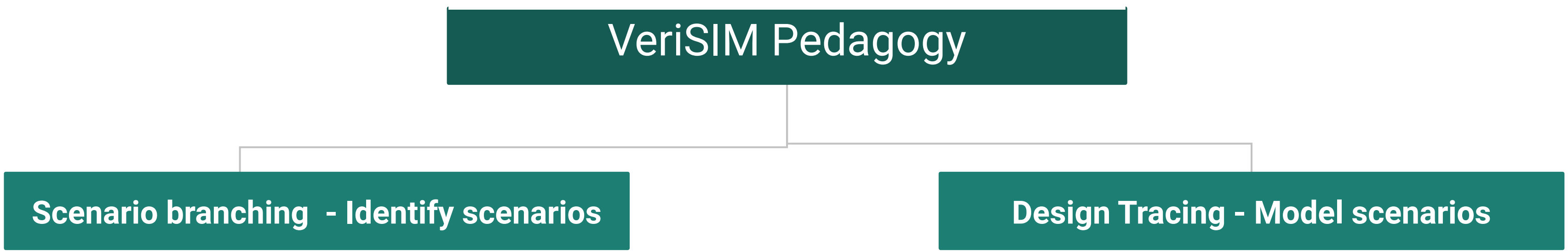
Students have difficulties in developing a rich and consistent mental model of the design

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
   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

1. Prasad, P., & Iyer, S. (2020, August). How do Graduating Students Evaluate Software Design Diagrams?. In Proceedings of the 2020 ACM Conference on International Computing Education Research (pp. 282-290).

2. Prasad, P., & Iyer, S. (2020, June). VeriSIM: a learning environment for comprehending class and sequence diagrams using design tracing. In Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering Education and Training (pp. 23-33).

Posters

1. Prasad, P., & Iyer, S. (2020, November). Inferring Students' Tracing Behaviors from Interaction Logs of a Learning Environment for Software Design Comprehension. In Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research (pp. 1-2).

2. Reddy, D., Alse, K., Lakshmi, T.G., Prasad, P., & Iyer, S. (2021, March). Learning Environments for Fostering Disciplinary Practices in CS Undergraduates. In SIGCSE 2021: To appear.

3. Prasad, P. (2018, August). Developing Students' Cognitive Processes Required for Software Design Verification. In Proceedings of the 2018 ACM Conference on International Computing Education Research (pp.284-285). ACM.

Bibliography - I

Adelson, B., Soloway, E., 1986. A model of software design. International Journal of Intelligent Systems 1 (3), 195–213.

Bolloju, N., Leung, F. S., 2006. Assisting novice analysts in developing quality conceptual models with uml. Communications of the ACM 49 (7), 108–112.

Brechner, E., 2003. Things they would not teach me of in college: what microsoft developers learn later. In: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications. ACM, pp. 134–136

Buckley, B. C., Gobert, J. D., Horwitz, P., O'Dwyer, L. M., 2010. Looking inside the black box: assessing model-based learning and inquiry in biologica™. International Journal of Learning Technology 5 (2), 166–190.

Burgueño, L., Vallecillo, A., Gogolla, M., 2018. Teaching uml and ocl models and their validation to software engineering students: an experience report. Computer Science Education 28 (1), 23–41.

Chren, S., Buhnova, B., Macak, M., Daubner, L., Rossi, B., 2019. Mistakes in uml diagrams: analysis of student projects in a software engineering course. In: Proceedings of the 41st International Conference on Software Engineering: Software Engineering Education and Training. IEEE Press, pp. 100–109.

Cohane, R., November 2017. Financial cost of software bugs. URL https://medium.com/@ryancohane/financial-cost-of-software-bugs-51b4d193f107

Eckerdal, A., McCartney, R., Moström, J. E., Ratcliffe, M., Zander, C., 2006. Can graduating students design software systems? ACM SIGCSE Bulletin 38 (1), 403–407.

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

Bibliography - II

Haskins, B., Jonette, S., Dick, B., Moroney, G., Lovell, R., Dabney, J., 2004. Error cost escalation through the project life cycle. In: Proceedings of the 14th Annual INCOSE International Symposium.

Kopainsky, B., Alessi, S. M., Pedercini, M., Davidsen, P. I., 2015. Effect of prior exploration as an instructional strategy for system dynamics. Simulation & Gaming 46 (3-4), 293–321.

Kopainsky, B., Alessi, S. M., Pedercini, M., Davidsen, P. I., 2015. Effect of prior exploration as an instructional strategy for system dynamics. Simulation & Gaming 46 (3-4), 293–321.

Li, P. L., 2016. What makes a great software engineer. Ph.D. thesis.

Lindland, O. I., Sindre, G., Solvberg, A., 1994. Understanding quality in conceptual modeling. IEEE software 11 (2), 42–49.

Loftus, C., Thomas, L., Zander, C., 2011. Can graduating students design: revisited. In: Proceedings of the 42nd ACM technical symposium on Computer science education. ACM, pp. 105–110.

Mulder, Y. G., Lazonder, A. W., de Jong, T., 2011. Comparing two types of model progression in an inquiry learning environment with modelling facilities. Learning and Instruction 21 (5), 614–624.

Mulder, Y. G., Bollen, L., de Jong, T., Lazonder, A. W., 2016. Scaffolding learning by modelling: The effects of partially worked-out models. Journal of research in science teaching 53 (3), 502–523.

Newman, M., 2002. Software errors cost us economy 59.5 billion annually. NIST Assesses Technical Needs of Industry to Improve Software-Testing.

Bibliography - III

Pennington, N., 1987. Comprehension strategies in programming. In: Empirical studies of programmers: second workshop. Ablex Publishing Corp., pp. 100–113.

Plomp, T., 2013. Educational design research: An introduction. Educational design research, 11–50.

Schumacher, R. M., Czerwinski, M. P., 1992. Mental models and the acquisition of expert knowledge. In: The psychology of expertise. Springer, pp. 61–79.

Sien, V. Y., 2011. An investigation of difficulties experienced by students developing unified modelling language (uml) class and sequence diagrams. Computer Science Education 21 (4), 317–342.

Soloway, E., Ehrlich, K., 1984. Empirical studies of programming knowledge. IEEE Transactions on software engineering (5), 595–609.

Tang, A., Aleti, A., Burge, J., van Vliet, H., 2010. What makes software design effective? Design Studies 31 (6), 614–640.

Von Mayrhauser, A., Vans, A. M., 1996. Identification of dynamic comprehension processes during large scale maintenance. IEEE Transactions on Software Engineering 22 (6), 424–437.

Westphal, B., 2019. Teaching software modelling in an undergraduate introduction to software engineering. In: 2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems Companion (MODELS-C). IEEE, pp. 690–699.

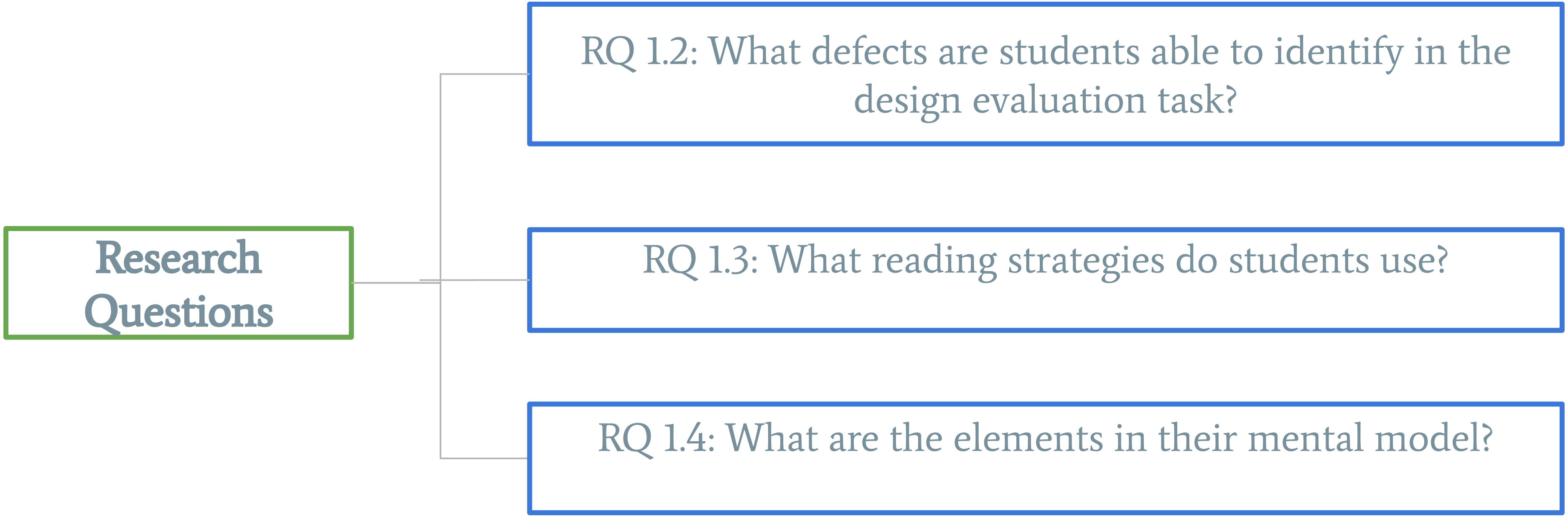
Wijnen, F. M., Mulder, Y. G., Alessi, S. M., Bollen, L., 2015. The potential of learning from erroneous models: comparing three types of model instruction. System dynamics review 31 (4), 250–270.

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

Study 1b: Data Sources and Analysis

	Data Source	Data Analysis
RQ 1.2: What defects are students able to identify in the design evaluation task?	Student responses on the task sheet	Student responses on the task sheet
RQ 1.3: What reading strategies do students use?	Video of students performing the task and screen capture	Thematic analysis of video data
RQ 1.4: What are the elements in their mental model?	Audio of the post-task interview	Thematic analysis of audio data

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

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

Focussed on surface level parts of the diagrams

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

Back to main slides

Study 2 - Details

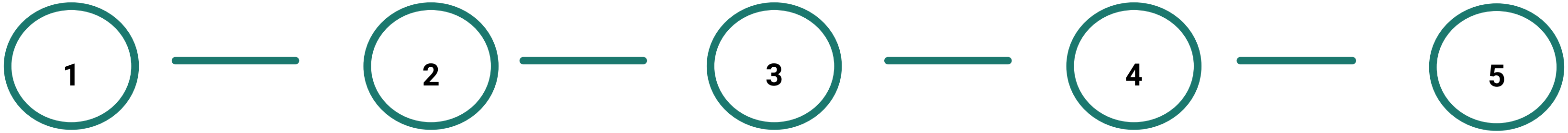
Study 2: Effect of VeriSIM on Students' Evaluation Skills:

	Data Source	Data Analysis
RQ 2.1 Does VeriSIM improve learners ability to **model a given scenario?**	Question in pre-test and post-test: **Explain the changes in the system on execution of this scenario**	Differences in pre-test and post-test question based on rubric
RQ 2.2 Does VeriSIM improve learners ability to **uncover defects?**	Question in pre-test and post-test: **Uncover defects in design diagrams**	Content analysis of "uncover defects" question in the pre-test and post-test

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



Pre-registration

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

Pre-test

Design of ATM system:

- Class diagram
- 3 sequence diagram

Questions:

- Execute the given scenario
- Identify defects based on the requirement

Interaction with VeriSIM

Post-test

Design of library system:

- Class diagram
- 3 sequence diagram

Questions:

- Execute the given scenario
- Identify defects based on the requirement

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?

Study 2: Data Source and Analysis

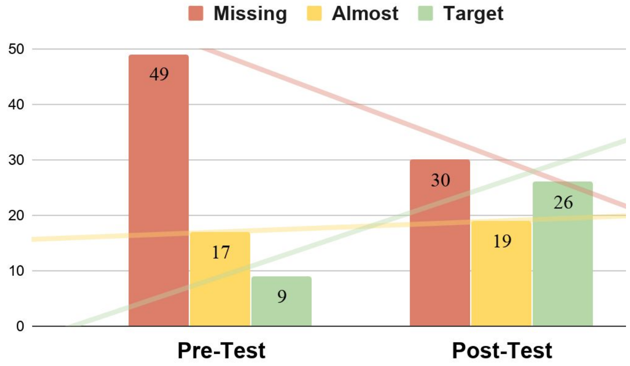
	Data Source	Data Analysis
RQ 2.1 Does VeriSIM improve learners ability to model a given scenario?	Question in pre-test and post-test: **Explain the changes in the system on execution of this scenario**	Differences in pre-test and post-test question based on rubric
RQ 2.2 Does VeriSIM improve learners ability to uncover defects?	Question in pre-test and post-test: **Uncover defects in design diagrams**	Content analysis of “uncover defects” question in the pre-test and post-test

Study 2: Results - RQ 2.1: Model a given scenario: Rubric

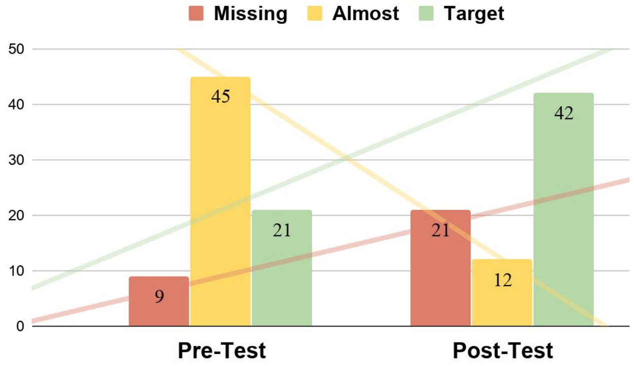
	Missing (0)	Almost (1)	Target (2)
Identifying relevant data variables	Missing all relevant data variables from the class diagram	Identifies some relevant variables Adds irrelevant data variables	Identifies all relevant data variables No irrelevant data variables added
Identifying relevant events	Missing all relevant events Separation of events is not seen	Identifies some relevant events Identifies some irrelevant events Separation of events is unclear	Identifies all relevant events No irrelevant events included Separation of events is clear
Simulating state change	No mention of state change of variables	State change of some variables are mentioned with variable-value pairs	State change of all variables are clearly mentioned with correct variable-value pairs

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

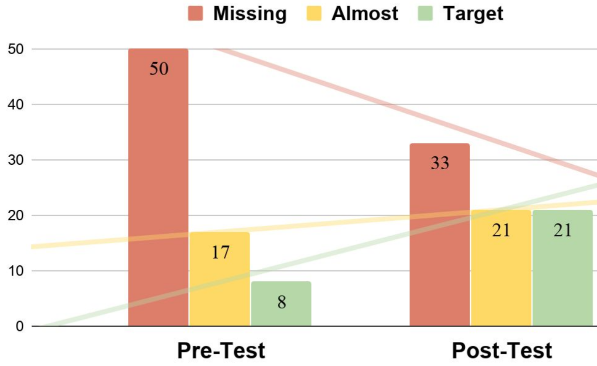
Data criteria



Event criteria



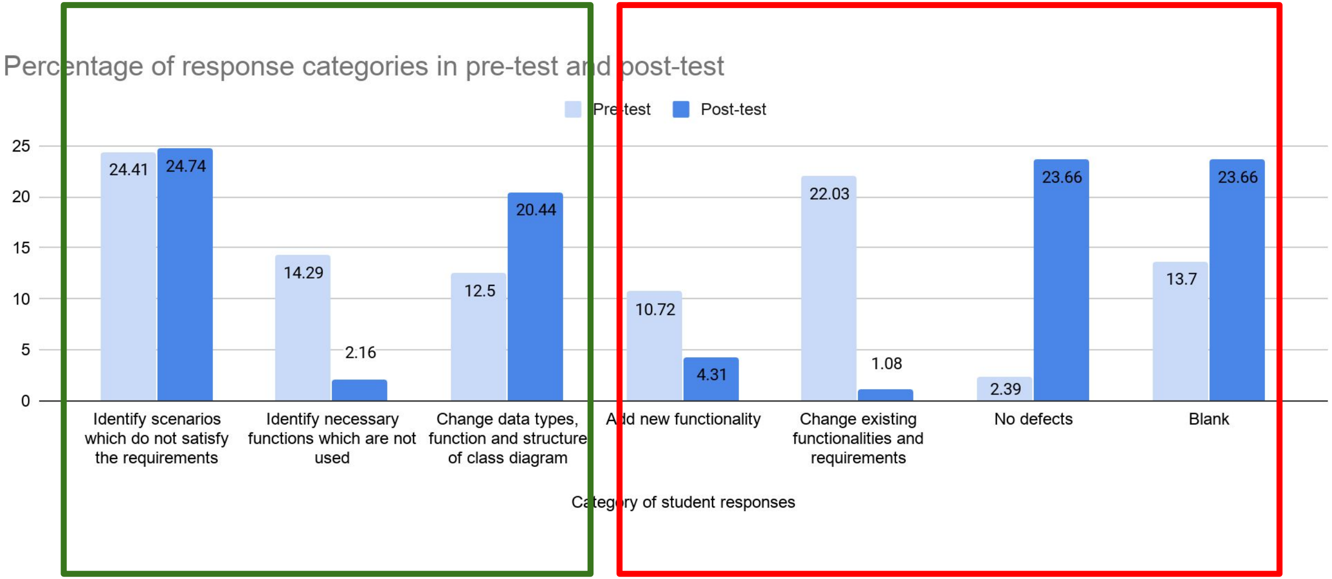
State change criteria



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

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**

Back to main slides

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

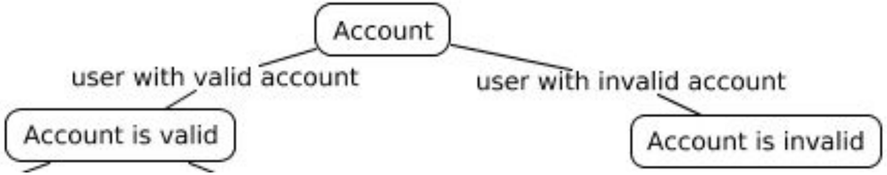
Scenario Branching Strategy

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.

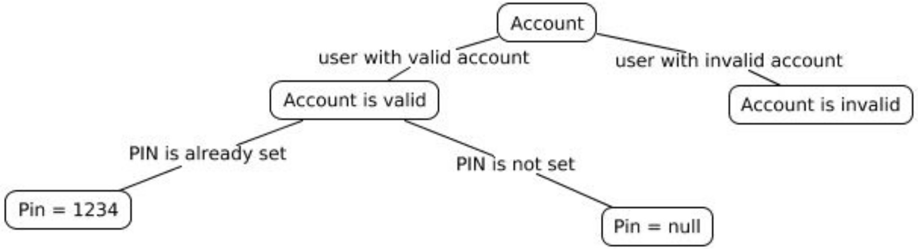
Steps:

- 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

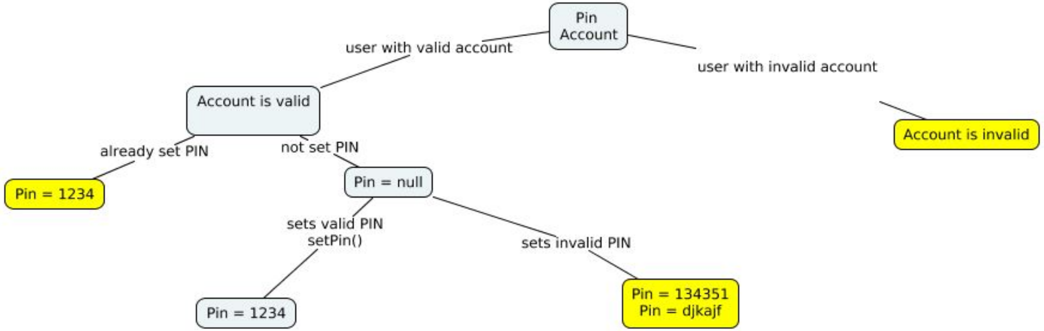


Scenario Branching Strategy

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**



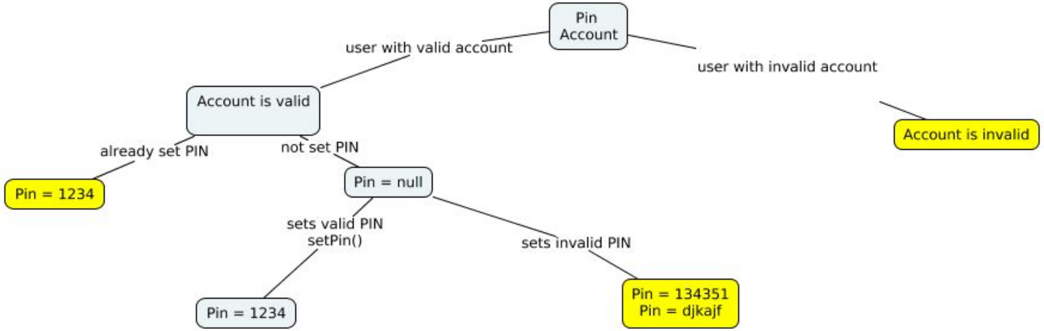
- Scenario 1: User with a valid account has already set a Pin
- Scenario 2: User with a valid account has not set a Pin and sets a valid Pin
- Scenario 3: User with a valid account has not set a Pin and sets an invalid Pin
- Scenario 4: User has an invalid account

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.

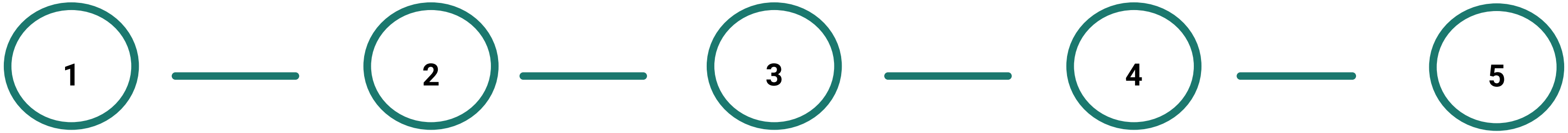
Back to main slides

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



1

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

2

VeriSIM - Module 1

Design Tracing Pedagogy

3

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?

4

VeriSIM - Module 2

Scenario branching pedagogy worksheet

5

Post-test and focus group interviews - 2

Design of a streaming website

- Class diagram
- 3 sequence diagram

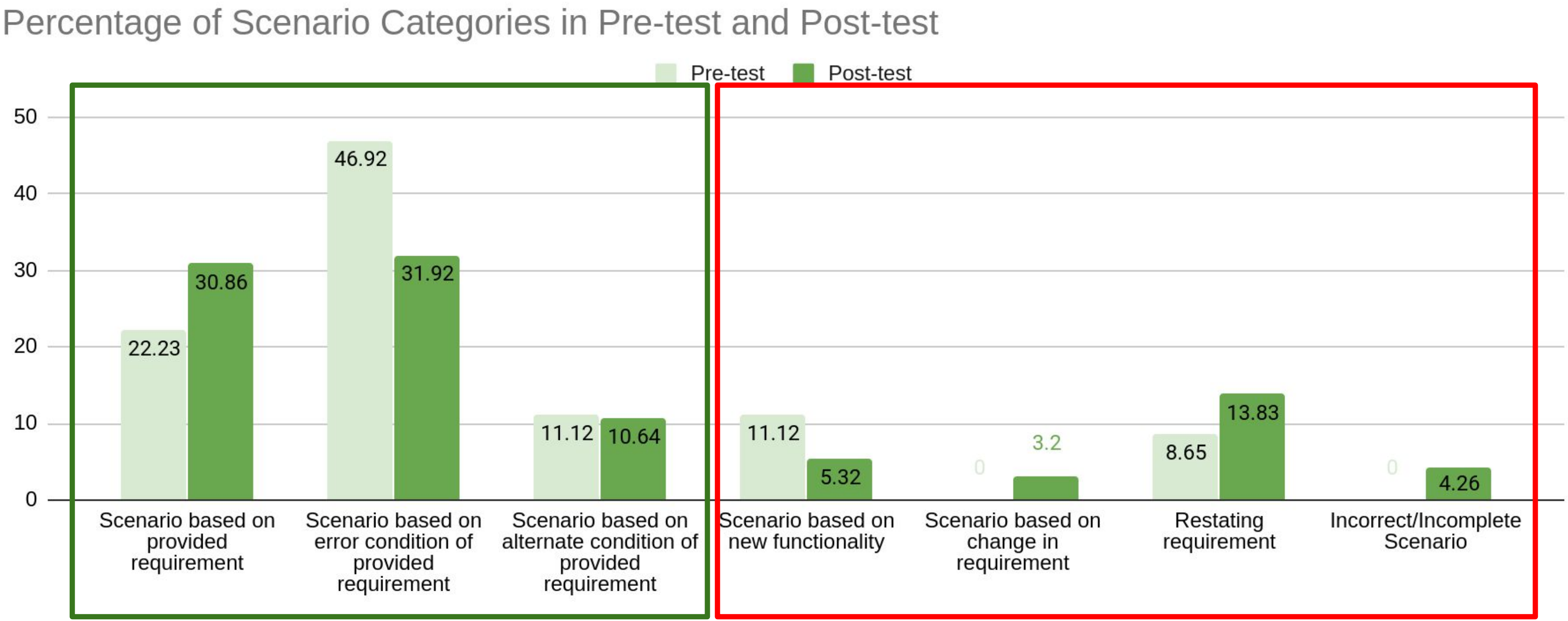
Questions:

- 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

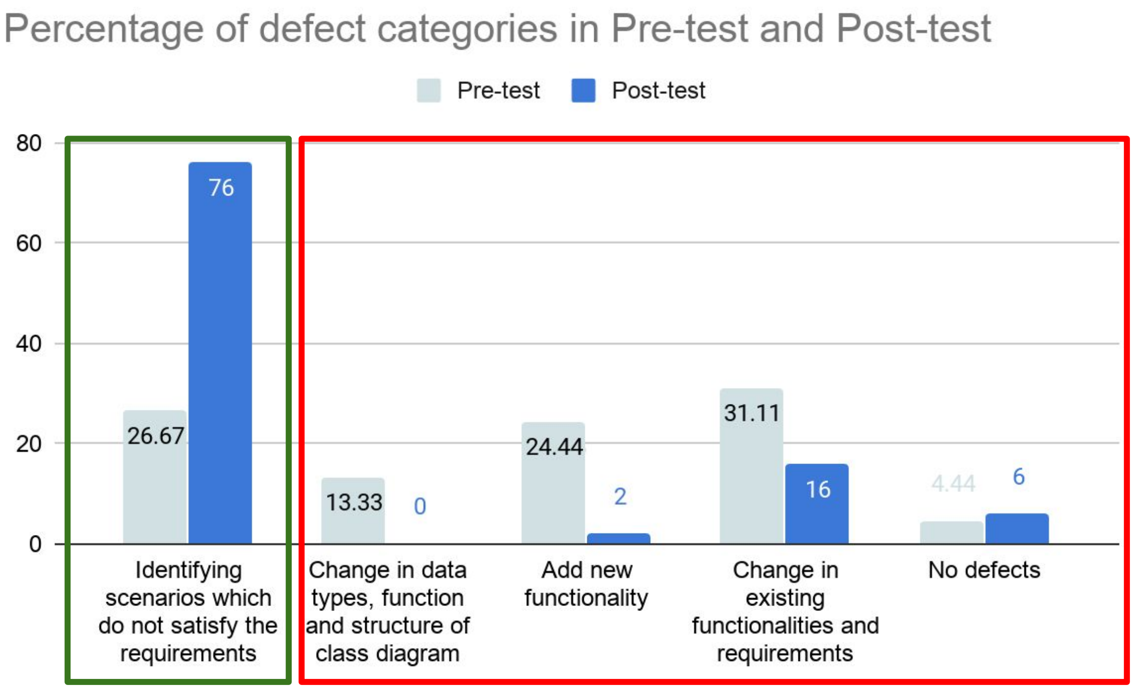
Study 3: Results - RQ 3.1: Identify Scenarios



Total number of responses in Pre-test: 81

Total number of responses in Post-test: 94

Study 3: Results - RQ 3.2: Identify Defects



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

Back to main slides