ANUKARNA: A Software Engineering Simulation Game for Teaching Practical Decision Making in Peer Code Review

Ritika Atal * and Ashish Sureka+
*Indraprastha Institute of Information Technology, Delhi (IIITD), India
Email: ritika13103@iiitd.ac.in
+ABB (India)
Email: ashish.sureka@in.abb.com

CMCE 2015
1st December 2015
What is Code Review?

- The practice of examining the source code in order to find mistakes and improve its overall quality.
- One of the standard practices of software engineering.
- Often known as *peer review*.

Current education models are content driven and instructor centric.

Limited opportunities for students to experience the presented concepts practically.

Students remain unaware of dynamics and functioning involved in development of real software projects at an organization level.

This is where *Simulation* - the process of recreating reality in virtual form comes to our rescue!!!

Through simulation, valuable hands-on experience is accumulated without incurring the high cost of performing the actual exercise.
Learning Models Used

1) DISCOVERY LEARNING:
• Individual learns a piece of knowledge most effectively if they discover it on their own.

2) LEARNING THROUGH FAILURE:
• Most memorable lessons are those that are learned as a result of failure.
Research Aim

1) To develop a web-based interactive educational SE simulation game or environment for teaching benefits and best-practices of peer-code review process.

2) To investigate a learning framework and model based on discovery learning, learning from failure, evidence and reasoning for teaching concepts on the practice of peer code review.

3) To evaluate the proposed learning framework and tool by conducting experiments and collecting feedback from users.
In this game we define 12 learning objectives covering multiple aspects of peer code review.

These learning objectives are captured in our pre and post game questions.

Each question is designed keeping in mind the learning objectives of the game.

Based on the learning objectives we come up with a situation or a decision which best questions that objective.

An evaluation question is then formed around this decision which challenges the player's knowledge to its best.
Unexpected Events

- We introduce unexpected events and unforeseen circumstances in the game to make it more realistic.
  - Internal conflicts between the team members
  - Attrition and change in deadline
  - Demand from the customer

- Unexpected events are unobservable and our goal is to examine the response and the decision making ability of the player to unexpected circumstances.
• We calibrate our scoring system such that incorrect decisions (quick and dirty solutions) lead to accumulation of technical debt.

• We take the maximum range value i.e. 5 as standard TD (Technical Debt) point against which all calculations are made.

deviation Value = (standardTD – decision Weight)

\[
\text{sumTD} = \text{sumTD} + \text{deviation Value}
\]

\[
\text{TDavg} = \frac{\text{sumTD}}{\text{decisionCounter}}
\]

\[
\text{TD} = \left(\frac{\text{TDavg}}{\text{standardTD}}\right) * 100
\]
Time

• It is required by the player to properly plan the project time and associated decisions to meet project deadline.

• Every decision that a player takes in game has a positive or negative impact. The magnitude of impact varies depending on the choice or decision made by the player.

• Series of decisions taken by a player either prevent them from meeting the deadline or they are able to complete the project successfully

\[
\text{remainingTime} = \text{remainingTime} - \text{pathTime} \\
\text{timeScore (days)} = \text{remainingTime}
\]
At the beginning of game, player is allotted a budget of Rs 2 million to complete the project.

The scoring system tracks player’s utilization of this budget.

If at any point of time player has consumed the entire budget value, they can go no further in game.

Budget remaining at the end of game is the combined effect of the cost consumed to perform review, developer’s recruitment, buying tools, team incentives etc.

\[
\text{remainingCost} = \text{remainingCost} - \text{pathCost}
\]
\[
\text{costScore (Rs)} = \text{remainingCost}
\]
Quality

- The scoring system captures the quality standards from the beginning of the project in terms of defect density (defects/kLOC).
- Code bases with a defect density of 1.0 (or 1 defect for every 1000 lines of code) are considered good quality software.
- Each path that player takes has a defect percentage (defect%) associated with it. Depending on player’s decision, this defect% either increases or decreases the software quality standards.

\[
\text{decisionQlty} = \text{defect\%} \times \text{projectQlty} \\
\text{projectQlty} = \text{projectQlty} + - \text{decisionQlty} \\
\text{qualityScore (defects/kLOC)} = \text{projectQlty}
\]
Each game consists of a problem space, initial state and single (or a set of) goal states.

A problem space is a mathematical abstraction in form of a tree where

- Root represents starting game state
- Nodes represent states of the game (decisions in our case)
- Edges represent moves and leaves represent final states

Demonstrates the different paths a player can follow and the weights associated with each path based on it's correctness.

Game tree has a branching factor of 4 with a solution depth of 9. Time complexity for its traversal is $O(4^9)$. 

Game Tree
Final Scoring

**STEP 1**

- As the player proceeds in game, the weight associated with each decision keeps on accumulating and is stored in $P_{sum}$.

$$P_{sum} = \Sigma W_d \text{(out of 50)}$$

**STEP 2**

- Project attributes sum ($P_{attr\_sum}$) holds the average of equivalent values of these three project attributes

$$P_{attr\_sum} = (C_f + T_f + Q_f)/3 \text{ (out of 50)}$$
Final Scoring

STEP 3

• Score sum ($S'_f$) is obtained by adding the values calculated in step 1 and 2.

$$S'_f (\text{out of 100}) = P_{\text{sum}} + P_{\text{attr_sum}}$$

STEP 4

• We then make use of $TD_{\text{avg}}$ to calculate the final score ($S_f$).

$$S_f = S'_f \times (1 - TD_{\text{avg}} / \text{standardTD})$$
An average mean score of 4.44 is obtained in pre game survey which increase to 7.75 in post game survey.

Improvement in absolute term ranges from 1.5 (Student ID 2) to 5.67 (Student ID 10)
Average improvement of 74.54% i.e. 3.31 in score, post game play.
Post Game Survey Score Values of Questions

A mean pre game score of 6.86 is observed for the survey questions, which then raises to 11.98 in post game survey.

Improvement in absolute term ranges from 2 (Q9 & Q11) to 12 (Q7).

Average improvement of 74.63% i.e. 5.12 in score, post game play.

Explores the improvement trend that exists for each and every question, before and after the game play.
Post Game Ratings given by Players in Section II

Learning Done

Scoring & Analysis

Game evaluation questions

Retry Motivation

New Concepts

Rating by players

Outliers

(3, 3, 3, 3)

Median

q3 (4)

q1 (3.75)

q3 (5)

q1 (4)

Median

q3 (5)

q1 (3.75)

(5, 5, 5)

(2)

Section II - Requires the player to rate the game on different aspects
Conclusion

- Tool successfully exposes students to different aspects of peer code review along with the complexities involved in project management.
- Proposed learning framework and model based on discovery learning and learning from failure, has relevant impact on student's knowledge in short time.
- Simulation games can be used as an alternative to test concepts taught in classroom lectures.
- Scoring the decisions, providing feedback, presenting unexpected scenarios to students engage them and encourage them to perform better.
Thank You