Learning Activity

Technical Debt

Learning Goal

Technical debt is a concept in programming that reflects the extra development work that arises when code that is easy to implement in the short run is used (to allow for quicker application release) instead of applying the best overall solution. Therefore, it results in "interest", that is, the extra work needed to refactor (restructure) the code in the future.
The goal of this learning activity is to allow students to understand this concept and to make them able to manage efficiently the "technical debt" of a software development project.

Learning Objectives and Outcome

After playing this scenario, learners will be able to:

- Understand the concept of "technical debt"
- Apply the best procedures and techniques to balance "technical debt" and software releases

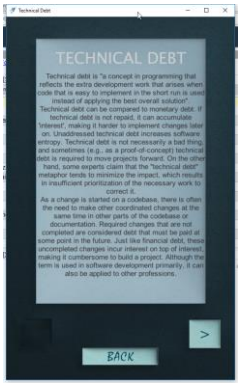
How to Use LEAP Game

In this game, the player takes the role of a software development team manager working on a project that will last for 10 sprints. His/her job is to create the highest amount of new software value by the end of the project. To that purpose the player has to balance the software development process that generates new value (NV) but increases technical debt (TD) and the actual reduction of TD through several investment measures available.

During each turn, the software development team has a finite capacity to create new software value and deal with technical deb. At the beginning of the game, the player has a certain points available to create new value (NV), and a certain number of points for technical debt (TD).

Each turn (representing a sprint), the player rolls a dice for creating new value and takes the total of all points rolled. Then he/she rolls the technical debt dice and totals that number. The net new value (NNV) created in each turn is the NV total minus the TD total.

To lower the burden of technical debt during the game, the player has four different measures: reduced complexity, continuous integration, increased test coverage and code review. The player can invest in one at a time and each investment in a TD-reducing measure reduces the NV dice for a few turns. At that point, the player gets the bonus capability for dealing with technical debt, and can invest in another TD-reducing measure. In each sprint, the player can also opt to make no investment at all.

How to play	
Initial screen allows to start a new game or to see the configuration options.	
The options menu allows to configure the language of the game, to see some instructions of the game and to play a game tutorial. The first time you play the game you should check the instructions and the tutorial. Show the students how to change the language of the game.	 

Playing the tutorial is important because it allows to see what is the effect of applying an investment. **Reduced complexity** Effect: remove 7 points on average from TD and add them in NV. Cost: 7 points on average from NV to TF for 3 sprints. **Continuous integration** Effect: remove 3.5 points on average from TD and add them in NV. Cost: 10.5 points on average from NV to TD for 2 sprints. **Increased test coverage** Effect: remove 3 points from TD each sprint. Cost: 3.5 points on average from NV to TD for 3 sprints. **Code review** Effect: Lowers the TD by a random amount. Cost: 3.5 points on average from NV to TD for 2 sprints. **No investment** Effect: none Cost: none	
Start the game and explain the learners the five options, just to recap the concepts learned in the tutorial. After that, let the players choose the options and follow the 10-sprint cycle.	 

In the end, check the results of all the students. Have the students with the higher score explain their method to the other students.	

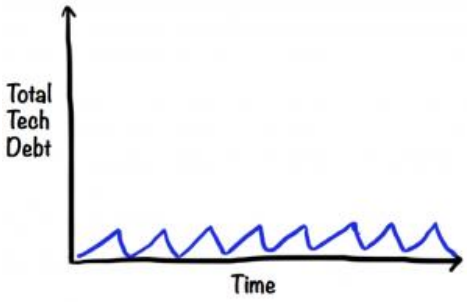
Class Collaboration

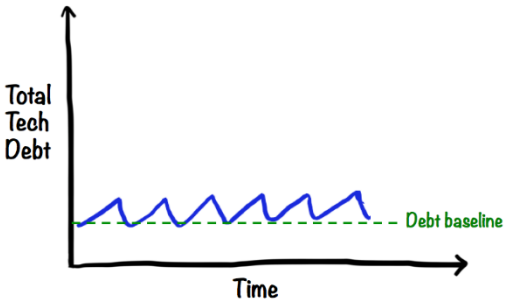
Questions to stir discussion in the classroom:

- Does it make any sense not to develop the best solution immediately?
- Does the need for refactoring imply that the existing code is bad and inefficient?
- Does the concept of “Technical debt” make sense in other software development technologies like in the waterfall models?
- What would be the ideal curve for technical debt?

Assessment

Have the students play the game and assess the technical debt curve they achieved, comparing with the “ideal” (left image) or “real ideal” (right image) curves proposed by Henrik Kniberg. Make students explain the relation of the curve they achieved with these two.





Auxiliary materials

- Techopedia, Technical debt, Available at: https://www.techopedia.com/definition/27913/technical-debt

- Wikipedia, Technical debt, Available at: https://en.wikipedia.org/wiki/Technical_debt
- Ward Cunningham, Debt Metaphor, Available at: https://www.youtube.com/watch?v=pqeJFYwnkjE
- Steve McConnell, 10x Software Development Best Practices: Technical Debt, Available at: http://www.construx.com/10x_Software_Development/Technical_Debt/
- Henrik Kniberg, Good and Bad Technical Debt, Available at: http://blog.crisp.se/2013/10/11/henrikkniberg/good-and-bad-technical-debt