Gameplay

Scenario

You are an IT company planning to enter security software business. You are presented with a set of possible features and their dependencies, which could be implemented in your development process. Choosing to implement certain features over the other implies earning different profits and satisfying various groups of customers. As a manager, you are to decide whichever feature to prioritise.

Instructions

Players are presented with a Tree Diagram. The diagram is an example of Release Planning — where certain tasks depend on each other, and the decision is to be made about which tasks to prioritise. There are two crucial variables: Revenue and Reputation. Revenue is within the range of 0-200, and Reputation is within 0-10. Both are equally important, and the final scoring is reflecting the optimal balance of both. The final scoring is determined by the sum of both revenue and reputation, where reputation is multiplied by 20 to equate to the range of revenue (0-10 * 20 = 0-200, respectively), hence:

Total Score = Final Current Revenue + Final Current Reputation * 20

The team with the greatest score is the winner of the game.

Earning early revenue from the features means wider possibilities of implementing more features the next turn (which we call a *sprint*). This brings benefit of revenue over reputation. However, it is easier to gain reputation points as they cost less than those features that provide high revenue returns. Therefore, it is recommended to keep a balance of both, in relation of "1 Reputation = 20 Revenue".

The other two variable are: Cost and Effort. Cost is simply the amount of money that has to be paid for the feature from the budget. Therefore, players are limited by the budget they are provided with (cannot go negative) and the sprint period. One sprint can only last 30 days. The variable *effort*, in fact, refers to how long it takes to implement the feature. The sum of effort has to equate or be less than 30 (per sprint).

Each turn players are supposed to take notes of the costs and earnings on the Spread Sheet provided. There are two tables: accumulated and planned assets. The first table (accumulated assets) is showing the budget that is available to be spent (players are not allowed to exceed the budget limits), current revenue that will be earned after this sprint is complete, and current reputation that has been accumulated over the sprints. By default, budget is set to 800 to allow players for the optimal use of strategies and planning of resources — this should provide at least 3 features to be implemented for the first *sprint* and just enough revenue returns for future planning. Current revenue is set to -50 — this is the salary that has to be spent on labour each sprint; hence, players are not allowed to spend over 750 for the first round.

The second table (planned assets) is storing information of decisions that are being made during each sprint. Correspondingly, is includes Costs — which is the sum of all costs per feature to be paid this sprint; Extra Revenue — the planned revenue returns that will be added to the Current Revenue; Extra Reputation — gains in reputation scores, which will also be added to the Current Reputation. After the second table is filled, players will add the extra gains and subtract the costs from the budget. Additionally, they will add current revenue to the budget, and record all this transactions in the first table.

Each time the first table is filled with information, before making future planning and filling the second table, an event card is taken. The event card is meant to introduce randomness to the game. One event card affects all of the teams, therefore, they face similar problems or gain almost equal benefits.

The other type of uncertainty is introduced by rolling a die. Each team rolls a die just once per sprint, and the probability is allocated as follows:

Variable Reputation			Variable Revenue		
1 (low)	2 (medium)	3 (high)	4 (low)	5 (medium)	6 (high)
Medium Revenue			Medium Reputation		

From this table, if a player receives die value of 1 — this mean low reputation and medium revenue. Similarly, rolling a 6 will give high revenue and medium reputation. This way players are more likely to have medium values for either of two, with some occasional luck (or not). Most importantly, this represents individual uncertainty as each team is rolling their own die. Teams are rolling their dies after the decision is made about which set of features to implement for the current sprint.

There might be cases when players run out of money, but an event card is demanding an amount larger than the budget available. During such moments of the game, players are allowed to take a *loan*. This will supply 200 extra credits for their personal use. However, during the next rounds and before the end of the final sprint, the team will have to pay 250 credits back; otherwise, they lose and the final score is not counted. Therefore, some of the teams not being able to pay for unexpected events will be introduced to the concept of loaning and paying the interest rates.

All in all, players are following these **steps**:

1. Plan your implementations;
2. Roll a die and choose your option;
3. Record values in table 2, and then in table 1;
4. Take an event card; (possibly, take a loan)
5. Repeat.