Semantic Systems - Ontology modeling of Computer Games

Teodor Chakarov - 12141198

Description

The market for video games is getting larger and larger. New game studio-companies are getting in. Some of them can generate big revenue from the very begining while others struggle to sell their first game for long period of time. Some of the companies who are doing well from the very beginning but start loosing the interest from their fan-base. On the other hand since there are a lot of games being developed, customers get overwhelmed with choices. They don't know which game is good before they try it. Maybe they also don't know about the options being out there.

Thats why our Ontology model aims to help both customers and developers. For example, the **customers** will:

- See all the games out there.
- Filter the games based on their prefference (genre, gameplay duration, price, game studio or etc.)
- Have the possibility to see the ratings given by other customers or professional game critics.
- Will have the possibility to see what type of payment is needed (monthly or just once) as well as what type of microtransactions are there (payments within the game).

Respectively, the **developers** will be able to:

- To evaluate how well the other games are doing by their rating. Also to see what type of characteristics they offer.
- What genre is the most risky based on the number of dowlnoads by each genre.
- They can also get feedback form professional critics.

The basic model will consist of the following class structure: 1) **Game** - will be the main class which will hold the information for each game. The class properties are *names, number of downloads, price, duration, the prizes it won and other* 2) **Requirements** - will store the information for a sertain game. There should be 2 for each game (minimal requirements and recommended). 3) **Payments** types for each game. They respectivley contain 2 subclasses (Microtransaction and License Type) 4) **Creators** - are the people responsible fo the game. Contianing the subclasses for game-studios and distributors 5) **Ratings** will store the raitings which are comming from the 2 subclasses which are customer and professional ratings.

In order to develop well structured and organized application we need to formu-

late some questions which will help us during the developement of the Ontology model. Based on them we can have a "path" to follow for creating the instances and the classes.

Question helping the development of the applicaiton

1. ***Which games are popular?** - as customers and developers will use this application we need to give the opportunity for comparing them.
2. **Is there a relationship between how many times one game was being sold and awards received?** - We can track if one game recieves award based on the success or critics choice
3. **What kind of games customers like to play?**
4. **Are the most popular games being developed in the last 5 year?**
5. **Which platform is preferred the most for each game genre?**
6. **Do the big companies sell more games in comparison to smaller onces?**

Steps of creating the apps

1. The first step is to create the classes for the ontology model. Based on the keywords, questions and ideas we need to construct the classes and sub-classes structure.
2. Then we created the data properties. They are showing the types of the attributes from the classes. For example the name of the game is string, the date of developement is date-time, number of downloads is integer and etc. They also show the cardinality of the data.
3. The object property are linking the individuals (instances of the classes) with eachother. They can be one or bi-directional, can have characterisitcs like Inverse, Transitive, Symmetric, Reflexive and more.

OWL Features

Disjointness: Can be found for each of the subclasses in System Requirements. One instance cannot be from multiple classes. Also the Ratings subclasses. **Constraint**: We have constraint on Payment - only 1 license is allowed per instance of Game. But the Microtransactions are not limited. Also the Game can have exactly two requirements (recommended and minimal). **Symmetric**: The object property "hasDistributor" and "isDistributor" are symmetric.