eSports Ontology

Ana Álvarez Suárez & Gonzalo Mier Muñoz

16 de enero de 2017

1. Introducción

Actualmente, los deportes electrónicos o eSports son uno de los negocios en auge de internet. Los benificiarios no son sólo los creadores de los videojuegos asociados a los mismos, que en su mayoría se distribuyen gratuitamente, sino los jugadores que ganan millones en torneos, los patrocinadores que permiten profesionalizar esta actividad a cambio de publicidad personalizada o incluso las casas de apuestas, que creando una similitud con los deportes tradicionales, permiten apostar a favor o en contra de que ocurra determinado evento.

Debido al fenómeno, han surgido gran cantidad de sitios web que sumistran información relativa a los eSports, datos y estadísiticas, algunas para el espectador y otras para desarrolladores, que las utilizarán, por ejemplo, en webs de apuestas. Sin embargo, al ser un concepto moderno, no existe una manera estándar de proporcionar esa información y cada página utiliza nomenclatura, conceptos, estadísticas distintos, creando confusión en los uusuarios y en los desarrolladores. Con esta idea ha surgido eSport Ontology, para así de poder proporcionar a las casas de apuestas, apostadores o aficionados, una forma cómoda de acceder a la información de los últimos eventos para poder conocer a la perfección el estado de estos mismos, y en el caso de querer, apostar.

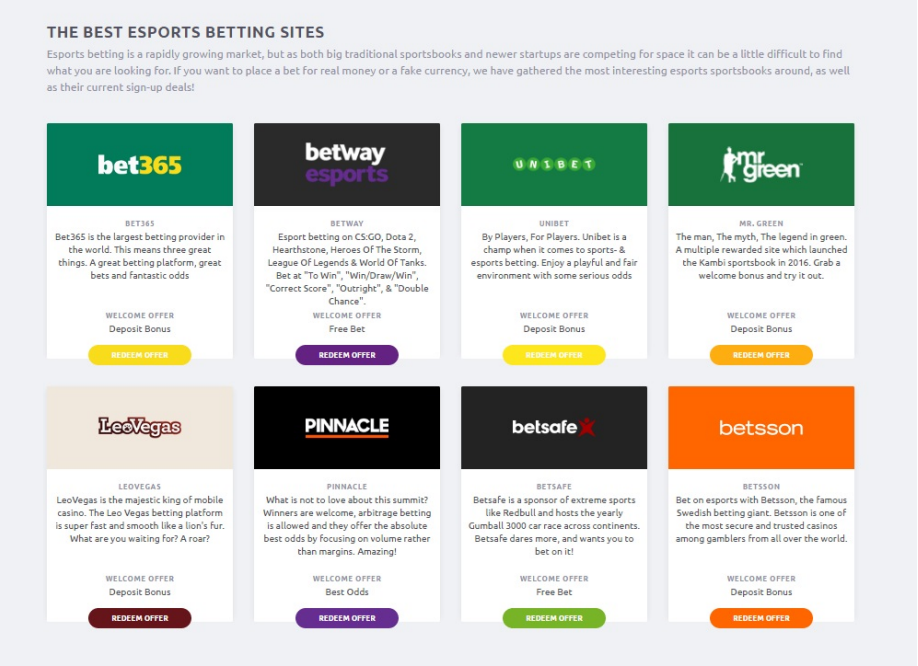

Figura 1: Algunas webs de apuestas

Algunos ejemplos de casas de apuestas online que podrían usar esta ontología serían: "bet365", "betway esports", "unibet",...; y ejemplos de webs de información de eSports: "gosugamer" o "esportsmatrix".

Esta ontología se ha planteado como una **ontología semi-pesada**, ya que no se necesitarán realizar inferencias complejas sobre los datos, pero se quiere que si algún dato es erróneo, sea capaz de descubrirlo. Por todo esto, se hará una ontología con los axiomas necesarios como para realizar inferencias simples, debido a que inicialmente no necesario razonar con los datos, aunque siempre cabe la posibilidad de extenderla.

2. Requerimiento de especificación de la Ontología

e-Sports Ontology Requirements Specification Document
1. Purpose
El propósito de nuestra Ontología de los e-Sport es obtener un modelo consensuado del dominio de los e-Sport o deportes electrónicos profesionales, el cual pueda ser utilizado de forma genérica por las diferentes casas de apuestas.
2. Scope
Dentro del dominio de los deportes solo se refiere a los deportes electrónicos y dentro de estos, nos centraremos en los "tower defence", con el nivel de granularidad que precise una casa de apuestas, aunque pueda ser extendido para otro uso.
3. Implementation Language
OWL (con sintaxis rdfs/xml)
4. Intended End-Users
Usuario 1: gerente de apuestas, que ofrecerá distintas apuestas en distintos deportes y se beneficiará de los resultados. Usuario 2: usuario que va a apostar, aficionado de e-Sports, que realice una apuesta sobre uno de los e-sport ofrecidos, o una simple consulta sobre datos de e-Sports.
5. Intended Uses
Uso 1: Describir e-Sport en función de sus ligas, eventos, hitos... Uso 2: Publicar una posible apuesta para un e-Sport en una web. Uso 3: Usuario apostador puede buscar entre las distintas apuestas y realizar alguna existente.
6. Ontology Requirements
a. Non-Functional Requirements
NFR 1. La ontología debe ser creada en inglés. NFR 2. La ontología debe ser reutilizable, y utilizar estándares del deporte y las apuestas.
b. Functional Requirements: Groups of Competency Questions

CQG1. Equipo (15 CQ)	CQG2. Jugador (16 CQ)
CQ1. ¿Qué equipos juegan en un partido? Cloud9 y Vici Gaming; Team Liquid y Origen	CQ16. ¿Qué jugadores juegan en un equipo? Balls, Meteos, Hai, Sneaky, LemonNation
CQ2. ¿Quién es el entrenador de un equipo? Pro0lly; YellowStar; Fochid	CQ17. ¿Dónde nació un jugador? Norteamérica, China, Korea
CQ3. ¿Qué patrocinadores tiene un equipo? Razer; Vodafone; G2A	CQ18. ¿Cuál es el nombre real de un jugador? An Le
CQ4. ¿Quién ganó un partido dado? Vici Gaming; Cloud9; Origen	CQ19. ¿En qué posición juega un jugador? Jungla; Calle superior, Calle inferior
CQ5. ¿Quién ganó un evento? Vici Gaming; Cloud9; Origen	CQ20. ¿Quién ha sido el MVP de un partido? Vasili; Meteos; Balls
CQ6. ¿Qué equipo mató al primer dragón / barón / roshan? Vici Gaming; Cloud9; Origen	CQ21. ¿Quién ha hecho la primera sangre en un partido? Vasili; Meteos; Balls
CQ7. ¿Qué equipo destruyó la primera torre? Vici Gaming; Cloud9; Origen	CQ22. ¿En qué equipos ha jugado el jugador? Cloud9, Vici Gaming y Team Liquid
CQ8. ¿De qué región es el equipo ganador de un evento? China; Europa; Norteamérica	CQ23. ¿Cuánto oro tiene cada jugador en un instante de un partido? 12546; 23545; 35842
CQ9. ¿Cuál es el primer equipo en anotar N muertes en un partido? Vici Gaming; Cloud9; Origen	CQ24. ¿Qué KDA tiene el jugador en un partido dado? 3/20/2; 10/2/5; 3/4/5
CQ10. ¿Qué equipos hay en un evento dado? Origen, Giants, Splyce, Fnatic...	CQ25. En un partido determinado, ¿con qué campeón juega un jugador? Amumu; Sion; Syndra; Shen
CQ11. ¿Cuál es el mejor equipo de una región en un juego? SKT Gaming en Korea	CQ26. ¿Qué jugador ha matado más veces a otro en esta partida? xPeke a Pepinero
CQ12. ¿Cómo de bueno es un equipo de una liga determinada? Posición 2 en la liga, porcentaje de victorias 85 %,	CQ27. ¿Qué campeón es el más elegido por determinado jugador? Zed; Amumu; sion; Syndra
CQ13. ¿Cuáles son las estadísticas del equipo en esa liga? oro total por partida 300k, número medio de barones derrotados 1.05, número medio de torres derribadas 6.25.	CQ28 ¿Con qué campeón se han ganado más partidas? Zed; Amumu; sion; Syndra
CQ14. ¿Cuáles son los siguientes partidos que va a jugar un equipo dado? Origen vs Splyce, LCS Ronda 5º., Origen vs Cloud 9, AllStarts 2016.	CQ29. ¿Ha jugado un jugador dado en un partido dado en su posición habitual? Si; No
CQ15. ¿Qué equipo tiene más probabilidad de ganar un cierto evento? SKT Gaming , se encuentra activo, ratio de victorias 1º, ranking 1º.	CQ30. ¿Cómo de bueno es un jugador dado? 3º en ranking de LOL. Porcentaje de victorias 67 %.
	CQ31. ¿Cuál es su K/D/A y estadísticas? KDA ratio 27, KDA medio 6.5/5/3, oro medio 31K.
	CQ32. ¿Cuál es el historial de partidas recientes de un jugador dado? Origen vs Splyce, LCS Ronda 4º, Origen vs Giants, LCS Ronda 4º.

<table>
<tr><th>CQG3. Partida (7 CQ)</th><th>CQG4. Juego y eventos (7 CQ)</th></tr>
<tr><td>CQ33. ¿Cuánto ha durado una partida dada? 30 min; 43 min 16 seg; 1 hora 2 min 60 seg
CQ34. ¿A qué evento pertenece un partido? All-Starts 2016; Campeonato Mundial 2016
CQ35. ¿De qué ronda es parte un partido? 1º; 3º; 4º
CQ36. ¿Cuántas torres han sido destruidas en un partido? 5; 8; 10
CQ37. ¿Qué mapa están jugando, dado un partido? La Grieta del Invocador
CQ38. ¿Cuántas partidas tiene un partido dado? 3; 5
CQ39. ¿Qué hitos hay en un partido de un videojuego dado? (LOL) Dragón, barón, primera sangre, primera torre, primer inhibidor, final.</td><td>CQ40. ¿Cuántas rondas hay en un evento? 5; 7; 8;
CQ41. ¿Cuándo acaba el evento? 1/10/15; 10 enero de 2016; 4/5/2014
CQ42. ¿Qué tipo de evento es? Liga regional; Campeonato Mundial, All-Star
CQ43. ¿Quién gana la ronda? Cloud9; Vici Gaming; Origen
CQ44. ¿Qué eventos hay en un videojuego dado? Ligas regionales, Campeonatos Mundiales, All-Star
CQ45. ¿Qué eventos se están jugando actualmente? Campeonato de España; All-Star 2016
CQ46. ¿Qué videojuego tiene más partidos tal día? CS:GO, 10 partidos</td></tr>
<tr><th colspan="2">7. Pre-Glossary of Terms</th></tr>
<tr><th colspan="2">a. Terms from Competency Questions + Frequency</th></tr>
<tr><td colspan="2">Partido 20
Equipo 16
Jugador 12
Evento 9
Ronda 3
Campeón 3
Videojuego 3
Estadísticas 2</td></tr>
<tr><th colspan="2">b. Terms from Answers + Frequency</th></tr>
<tr><td colspan="2">Cloud9/Origen 10
Campeonato 4
Amumu/Sión 3
victorias 3
Balls/Meteos/Hai 3
torre 2
China/Europa 2
Porcentaje 2</td></tr>
<tr><th colspan="2">c. Objects</th></tr>
<tr><td colspan="2">LOL, Amumu, Cloud9, Balls, Meteos, Origen, Korea, Maokai, jungla, etc...</td></tr>
</table>

Cuadro 1: Especificación de requerimientos de la Ontología

3. Planificación

Para la planificación se ha usado un modelo iterativo incremental, donde cada una de las iteraciones es en cascada.

Sólo llegamos a realizar dos iteraciones, pero por supuesto de ser un proyecto real se habría seguido en una mejora continua. Cada iteración esta comprendida por una fase de especificaciones, reuso, una de diseño, una de implementación, y, para terminar, se llevan a cabo las fases de evaluación y documentación del proyecto en cascada.

En la segunda iteración tanto la parte de especificaciones como la planificación estaban bastante claras y la necesidad de modificarlas fue mínima. En la segunda iteración el principal cambio fue que el sistema fue diseñado pensando en el reuso de una ontología que resultó problemática, por lo que se decidió rehacer esta iteración completa sin contar con esta ontología. Por último, tanto

la evaluación como la documentación han sido fases únicas, por lo que han sido colocadas en cascada.

Las fases de reuso y reingeniería de recursos ontológicos y no ontológicos han sido realizadas siempre en paralelo debido al alto paralelismo de estas tareas, ya que a la hora de buscar información, se buscaba de las dos clases indistintamente.

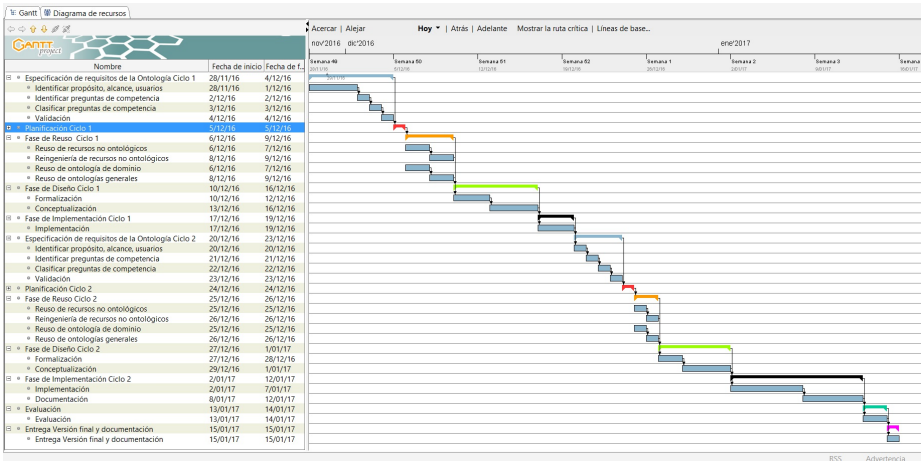

Figura 2: Planificación temporal de la realización del proyecto

4. Re-uso y re-ingeniería de recursos ontológicos y no ontológicos

4.1. Recursos no ontológicos

El modelo que se ha querido seguir de diseño ha sido inicialmente ABox, y posteriormente se ha querido poblar con otros datos webs que proporcionan esta información online.

Se ha elegido el modelo ABox dado que existían fuentes de datos muy completas que se podían reutilizar cómodamente para nuestro objetivo. Aquellas clases necesarias propias de los eSport pero que no pertenecían a ninguna ontología ya creada se han hecho usando el modelo TBox.

Para obtener los datos de los partidos, eventos y, en general, la mayoría de las instancias de la ontología se quiere usar la API de Abios. Esta página proporciona una API especializada para apostar en distintos eSports y en distintas casas de apuestas, por lo que se usará la base de datos de Abios como instancias, y se adaptará el esquema usado por la API para crear la eSport Ontology.

Sin embargo, esta adaptación no es directa, es compleja y necesita el uso de muchos otros recursos ontológicos, por lo que se usarán las webs lolesport y Gamepedia para añadir la información necesaria a la ontología.

4.2. Recursos ontológicos

Los recursos ontológicos que se pretendieron usar, tras buscar en diferentes fuentes de ontologías, en la primera iteración fueron: "Videogame Ontology" (VGO), la cual proporciona muchos de los conceptos relacionados con videojuegos, como jugador o personajes controlables o no; BBC Sport Ontology, la cual es una ampliación de la ontología de deportes, capaz de representar los complejos juegos olímpicos, pero como tenía los conceptos de descripción de eventos deportivos, era útil para la ontología; OWL Time, para manejar los tiempos en que suceden los eventos; FOAF, para la definición de los conceptos relativos a personas (jugadores, equipos..).

5. Diseño

Usando estas ontologías, nuestra ontología fue diseñada para estar dividida en las siguientes partes pequeñas que conforman la ontología de los eSports. Estas serían: Team Ontology, que usa foaf:agent para definir que es un jugador; Event Ontology, que usa OWL Time para calcular los tiempos; Videogame Ontology, que es una versión podada primero, y extendida (en amplitud) posteriormente de VGO; y Sport Ontology, que es una versión podada de la BBC Sport.

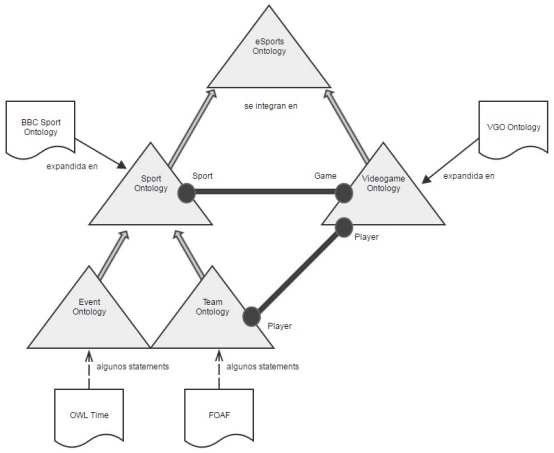

Figura 3: Esquema de diseño del ciclo 1

Desafortunadamente, BBC Sport fue descartada para nuestra ontología ya que había muchos problemas y errores a la hora de importarla, lo cual fue asumible, ya que de todos modos resultaba demasiado amplia, y tampoco tenía referencias anteriores así que no podemos considerarla un estándar necesario. Debido a esto, se decidió modificar la ontología de eSport para que cada una de las partes estuviesen mejor definidas y se relacionasen mejor entre ellas.

Videogame Ontology y Team Ontology se mantienen como en el anterior esquema. Event Ontology ahora usa OWL Event, para definir que es un evento y un subevento, y este a su vez de OWL Time. Por último, Sport Ontology usa dbo/schema vocab para algunas definiciones también. Con este formato, la ontología reusada mayormente ha sido VGO Ontology, siendo todo el resto de usos de otras ontologías un apoyo para definir las propias clases de eSport Ontology.

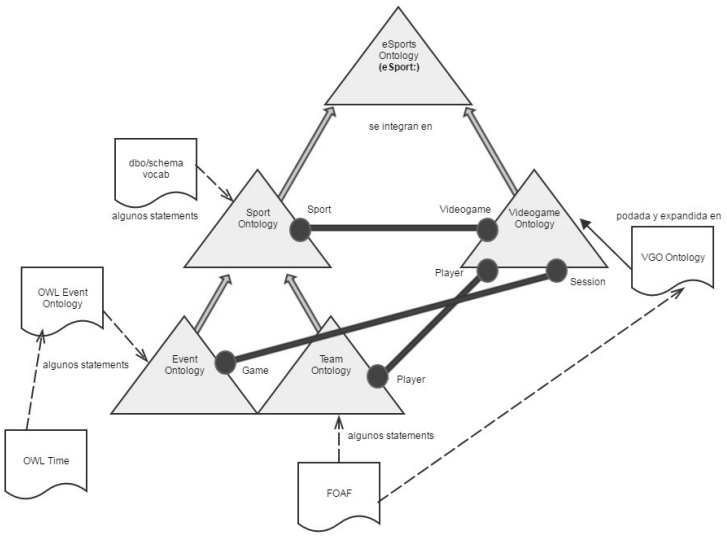

Figura 4: Esquema de diseño del ciclo 2

En esta sección se aprovechará para explicar las decisiones de diseño tomadas:

- Tenemos dos tipos de relaciones parte de, ambas hijas de "dcterms:partOf", una general transitiva que se utiliza para eventos y personas, y es por tanto general y con restricciones añadidas para evitar mala transitividad, y una intransitiva para temas de mapas, que no tienen pertenencia directa.
- "eSportClub" y "Team" no son lo mismo, por lo tanto se crean dos clases distintas. "Team" es un grupo de jugadores que juegan una partida en el mismo bando. Sin embargo, un "eSportClub" puede estar compuesto

por más de un equipo, que a su vez pueden estar dedicados a más de un deporte, por lo que "eSportClub" tampoco es dependiente del "eSport" que se este manejando.

- Una partida definida por "eSport:Game" se decide que sea equivalente a una "vgo:Session", ya que así podremos utilizar los conceptos del videojuego del eSport correspondiente expuestos en VGO, siriviendo así de nexo entre esas partes de la ontología.

- "PlayerRole" y "ChampionRole" no son lo mismo (de hecho son disjuntos). Esto se debe a que "PlayerRole" hace referencia a la calle del mapa donde juega un jugador, mientras que "ChampionRole" es la manera en la que juega, teniendo en cuenta las técnicas usadas y las características del campeón. Estos conceptos son habitualmente confundidos en las páginas de información. Decir además que los roles han sido *implementados como instancias*, siendo ése un patrón ABox, de gran expresividad.

- Para implementar la idea *Un jugador X juega un personaje Y en una sesión Z*, disponíamos de "vgo:involvesCharacter" y "vgo:involvesPlayer", propiedades de "vgo:Session", que no crean una correspondencia clara cuando hay varios jugadores con varios personajes en una partida. Como hemos visto, al ser una relación n-aria, lo que hay que hacer es crear una relación o clase intermedia llamada "CharacterPlayerRelation" para marcar una relación 3-aria inequívoca. Lo mismo ocurre con la relación *Un jugador X obtiene unas estadísitcas Y en una sesión/partida Z* y la relación intermedia GameStats, aunque de una manera menos directa.

- Debido a lo peculiar de esta ontología, se ha usado disjoints para todas las clases que se encontrasen en el mismo nivel, ya que en ningún caso podrían llegar a haber instancias coincidentes. Las únicas excepciones realizadas han sido "lastRound"/"lastMatch" y "Game"/"Match", debido a que en determinadas condiciones podrían llegar a coincidir.

- Se ha decidido no añadir "OWL Time" a la parte de VGO dado que no se pretende hacer razonamiento temporal con los eventos instantáneos, pero si en el futuro fuese necesario porque cambiasen las especificaciones, habría que añadirlo. En la parte de eSportsEvents sí que se utiliza, como definía la ontología OWL Event, pero no se han especificado propiedades de inferencia (before/after...), aunque serían de interés para versiones posteriores. Al igual pasa con "DataCube" para las estadísticas, el cual se añadiría como continuación de este trabajo.

- En vez de usar "vgo:Username", se podía haber usado "foaf:OnlineAccount", pero esta en fase de testing. Personalmente, creemos que VGO debería actualizar la terminología en cuanto este término este estable.

- Para definir cómo funcionan los tipos de eventos temporalmente, se han usado los patrones de secuencias explicados en la asignatura, que utili-

zan una propiedad "hasNext" y una clase puntero "lastSomething". El siguiente esquema muestra la estructura general de los torneos.

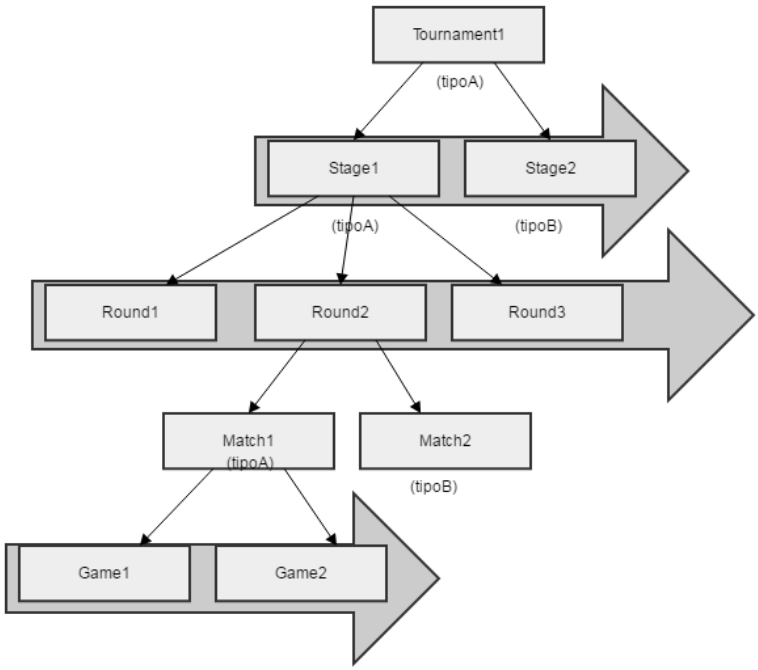

Figura 5: Esquema de funcionamiento de un evento temporalmente

- Se añadieron también algunas restricciones de cardinalidad como que "Game" siempre tiene exactamente una "vgo:session" y dos equipos "Team" asociados.

6. Implementación

Para la implementación del diseño se ha utilizado el lenguaje owl facilitado por la herramienta *Protégé* de la Universidad de Standford, versión 5. Tanto la documentación html obtenida directamente de la ontología como los ficheros owl de la misma se encuentran adjuntos a esta memoria.

7. Evaluación

Como primera parte de la Evaluación, se utilizaron **Razonadores** ya implementados en la herramienta Protégé. Ni Hermi ni Fact producen ningún tipo de inconsistencia ni inferencias equivocadas, aunque inicialmente hubo varias instancias que daban problemas por no estar bien definidas de acuerdo con las restricciones explicitadas, se resolvió con facilidad.

Además, en una segunda parte, se utilizó la herramienta *OOPs!*. En una primera pasada, las fuentes de error obtenidas fueron las de la imagen.

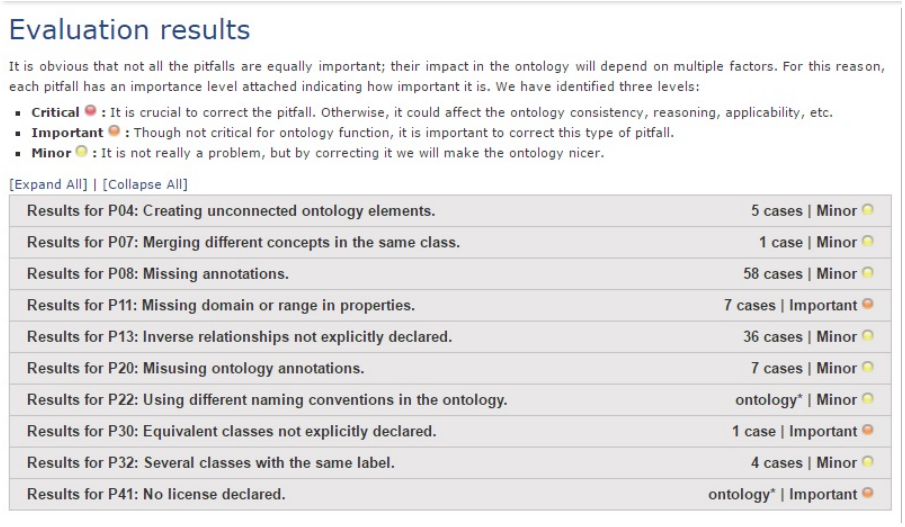

Figura 6: Errores de OOPS!

Tras solucionar algunas, se consideró que las siguientes "pitfalls" (no críticas) no eran solucionables o no resultaba interesante solucionarlas:

- P04 y P07 fallan porque no podemos subir a OOPs el archivo de la ontología importada ya podado, así que cree que hay clases desconectadas o equívocas que nosotros ya hemos borrado.
- P08 y P13 fueron minimizadas, pero sigue quedando alguna, especialmente relaciones sin inversas que decidimos dejar así, por ejemplo, para dejar intacto un patrón, o porque no eran necesarias.
- P20, las anotaciones que fallan son de la importación y decidimos dejar las originales y P22, VGO, de nuevo, utiliza otras nomenclaturas.
- P30 dice que la clase "Champion" puede ser equivalente a la clase "Friendly", pero esto no es cierto, ya que hay más posibles tipos de "Friendly" además de ese.

- P32 Hemos comprobado que ninguna comparte el mismo label, el error probablemente proviene de que algunas sólo se diferencian entre sí por números y OOPs no lo distingue.

8. *Linked Data*

En esta sección, explicaremos cómo sería el proceso de implantación de datos enlazados en la ontología a través de un ejemplo.

En nuestra opinión, los datos que tenemos se dividen en dos tipos: constantes y dinámicos. Los constantes son la información propia del juego (tipos de mapas y personajes...), que cambiará sólo con los parches de los juegos, digamos casi anualmente, y que es necesaria para la coherencia de la ontología. Los dinámicos son todos aquellos relacionados con eventos, estadísticas y equipos, que cambiarán constantemente y deben tener un flujo continuo de actualización.

Para la obtención de los datos, utilizamos servicios web llamados APIs de datos. Como la API Abios que íbamos utilizar originalmente nos puso problemas para acceder a ella gratuitamente, utilizamos la API Pandascore, de peor calidad y más difícil de ajustar a la ontología, pero suficiente para este ejemplo.

Figura 7: Resultados

Accediendo a esa API, vemos por ejemplo en la primera URL el tipo de información que guarda del equivalente a la clase "Tournament". Tiene Data Properties que no hemos implementado como "season" y otros datos que necesitamos en nuestra ontología como la fecha de inicio aparecen nombrados pero vacíos. La última URL se corresponde a un único partido, que será lo que añadamos a nuestra ontología como ejemplo. En él, sí aparecen las cosas que necesitamos para describirlo: equipo y jugadores participantes ("roster" para ellos), fechas, resultados y multitud de estadísticas, algunas son "InstantaneousEvents" asociados a la partida en general y otras "PlayerGameStats", asociados a jugadores. Aparece una gran cantidad de ellos, aunque nuestras propiedades sólo abarcan

un par, puesto que la idea es que la próxima versión implementaría las estadíticas de una forma más completa.

Esta API está diseñada para ser manejada desde Ruby. Para poblar la ontología, simplemente necesitaríamos realizar un programa cuyo código hiciera las queries para cada clase, en un proceso de búsqueda en profundidad para poder crear también las relaciones o propiedades, y que lo pasara a formato rdf.

Hemos decidido añadir los datos de esa última query de un único "Game" a mano en Protégé, cuyos resultados se muestran en parte las imágenes, ya que sólo aparece un jugador por facilidad. Otras formas mejores incluirían generar rdf mediante la utilización de *rdb2rdf* con bases de datos relacionales (que no nos sería útil) o pasarlo a csv y luego usar LODRefine.

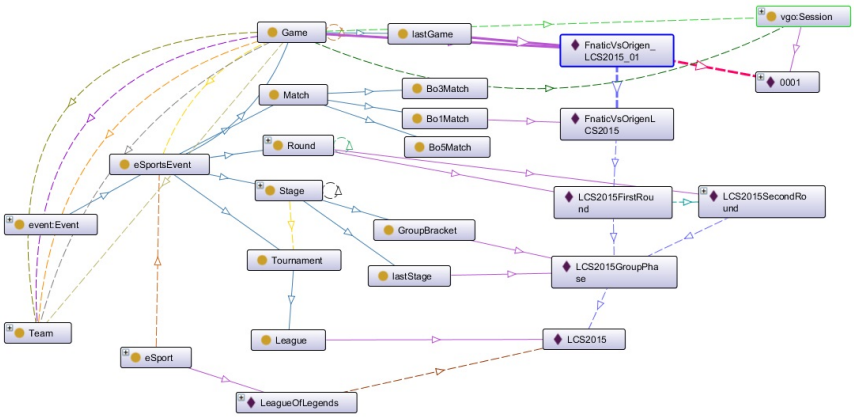

Figura 8: Implementación de eventos

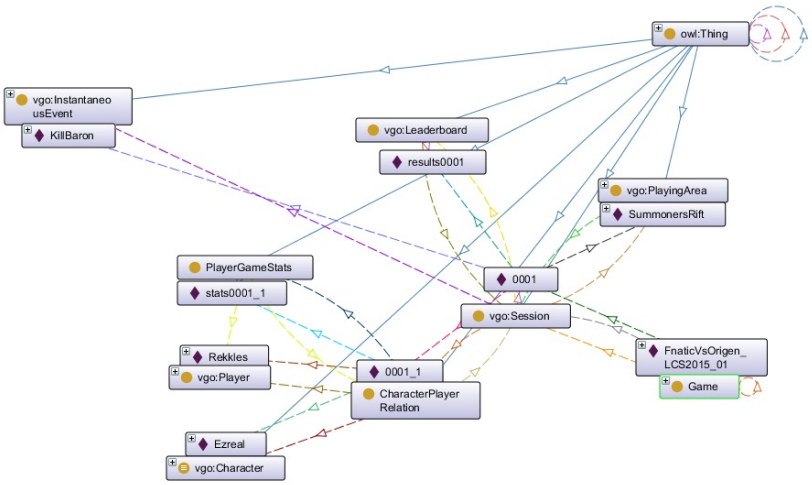

Figura 9: Implementación de una partida

Para poder realizar consultas sobre nuestra ontología con datos enlazados, necesitamos publicar estos. Se publican gracias al sistema de URIs que define el Linked Data.

En este caso, nuestro "namespace" tiene la forma:
https://www.theesportslov.org/eSports-ontology
Así que, por usar el mismo dominio, podemos publicar los datos en:
https://www.theesportslov.org/resources.
Así, una clase tendría la forma:
https://www.theesportslov.org/eSports-ontology# MiClase
Mientras que una instancia *https://www.theesportslov.org/resources/MiInstancia.*

Es interesante señalar que estamos utilizando URIs de tipo "slash", que se suelen utilizar para los datos ya que van directamente a la fila de interés, y no tipo "hash", ya que estas descargan la página entera. Aunque en nuestro ejemplo tendremos datos mínimos y por tanto daría igual, es mejor pensar de cara a la gran cantidad de datos que idealmente tendríamos.

Es decir, para representar una partida, tendríamos la clase de la ontología asociada:
https://www.theesportslov.org/eSports-ontology# Game

Mientras que para representar una instancia de partida:
https://www.theesportslov.org/resources/Games/eulcs15r2fnaticvsorigen1

En este caso la URIs no tienen porqué ser opacas, pero por ejemplo para jugadores sí, puesto que existe la posibilidad de que se repitan nombres.

Además de la publicación, para poder realizar "SPARQL queries" necesitaríamos utilizar algún servicio de "EndPoint" como "Virtuoso".

9. Conclusiones

Se ha logrado construir una ontología sobre deportes electrónicos o eSport usando Protégé, aplicando la metodología Neon. Para ello, se han reutilizado las ontologías: Event, TIME Owl, FOAF y VGO, en un intento por seguir la pauta de consenso propia de la ingeniería ontológica.

Además, se ha extraído información de bases de datos online, haciendo uso de APIs para poblar el esquema con instancias. Y por último, se han creado las URIs de una posible publicación de datos enlazados.

Como trabajo futuro para mejorar la ontología:

- Incorporar RDF Data Cube Vocabulary. Esta ontología proporciona facilidad para asociar a una instancia una estructura de datos del formato elegido por el diseñador. Vendría muy bien a la hora de guardar estadísticas de los jugadores y equipos en partidos y trabajar con ello. De la parte de estadísticas sólo hemos añadido una como ejemplo en lugar de la gran cantidad que irían, lo cual queda pendiente para una segunda versión, utilizando el vocabulario mencionado.
- Añadir más axiomas a las clases para que se pueda realizar inferencias más complejas. Aunque hemos añadido alguna restricción de prueba, este dominio tan artificial se presta a gran cantidad de las mismas para comprobación de coherencia. Por ejemplo, podríamos añadir cosas como que "Matches del mismo tipo de Stage son del mismo tipo" o que "Si el eSport es LOL, entonces tenemos exactamente 5 jugadores por equipo en un partido".
- Añadir como instancias todos los datos constantes existentes. Obtener y publicar los datos, los dinámicos obtenidos desde distintas APIs constantemente desde un servidor para tener información completa, actualizada y consensuada.

Referencias

Páginas web:

- http://www.neon-project.org/nw/NeOn_Book
- http://vocab.linkeddata.es/vgo/
- http://www.bbc.co.uk/ontologies/sport
- https://www.w3.org/TR/vocab-data-cube/
- https://www.w3.org/TR/2016/WD-owl-time-20160712/
- http://xmlns.com/foaf/spec/
- http://motools.sourceforge.net/event/event.html

Diapositivas:

- Asunción Gómez Pérez & Mari Carmen Suárez Figueroa. ONTOLOGY DESIGN PATTERNS. Ingeniería Ontológica, MUIA. 2016-2017