Table of Contents

[Introduction 2](#_Toc467539303)

[Aim 3](#_Toc467539304)

[Objective 3](#_Toc467539305)

[Scope 3](#_Toc467539306)

[Design 4](#_Toc467539307)

[Classes 6](#_Toc467539308)

[Object Property 11](#_Toc467539309)

[Implementation 15](#_Toc467539310)

[Evaluation and Use 16](#_Toc467539311)

[Critical Reflection 18](#_Toc467539312)

[Reference 20](#_Toc467539313)

[Appendix 21](#_Toc467539314)

# Introduction

Is the application area a suitable one for a semantic based application? Are the aims both useful and achievable?

Semantic application used to narrow down searches based on syntax and data.

This document presents the Video Game Ontology, an ontology created for describing knowledge of Video Games. The ontology focuses on modelling events happening inside video games as well as players and their playing behaviour.

To share common understanding of the structure of information among people or software agents

θ To enable reuse of domain knowledge

θ To make domain assumptions explicit

θ To separate domain knowledge from the operational knowledge

θ To analyse domain knowledge

What is the domain that the ontology will cover?

θ For what we are going to use the ontology?

For what types of questions the information in the ontology should provide answers? θ Who will use and maintain the ontology? Hospital Management System To save records for patients, staff, symptoms etc. What could be the suggested medicine for a patient with certain symptoms? Health applications, Hospital Management systems, Hospital IT staff.

It is the idea of having data on the Web defined and linked in a way that it can be used for

more effective discovery, automation,

integration and reuse across various

applications.

## Aim

Build a semantic application. The Video Game Ontology is aimed at modelling video game related information. The main goal is to capture knowledge about events that happen in video games and information about players.

## Objective

* Create an ontology using protégé by creating classes, object and data properties
* Add instances to the ontology
* Create SPARQL end point using Jena Fuseki

## Scope

This project is limited to creating an ontology using protégé. Building the application using Jena-fuseki as endpoint, Java application and PHP web application.

# Design

Use of RDF, RDF schema and Ontology design.

The Ontology-driven recipe querying application developed for this thesis is built up in three main parts; the OWL ontology, OWL server and the Web interface to interact with the system. The ontology constructed is the only information resource used for the application. All data such as the text representing the recipe, recipe category names etc. are stored in the ontology file constructed.

Camel case naming convention for classes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Classes** | **Sub-classes** | | |
| Engine |  |  |  |
| Genre |  |  |  |
| Release |  |  |  |
| Dimension |  |  |  |
| Player Mode |  |  |  |
| Video Game | Killzone |  |  |
| Mortal Kombat |  |  |
| Generation | First |  |  |
| Second |  |  |
| Third |  |  |
| Forth |  |  |
| Fifth |  |  |
| Sixth |  |  |
| Seventh |  |  |
| Eighth |  |  |
| Person | Director |  |  |
| Artist |  |  |
| Writer |  |  |
| Producer |  |  |
| Programmer |  |  |
| Designer |  |  |
| Composer |  |  |
| Platform | Web |  |  |
| Console | Home | Play Station |
| Xbox |
| Handheld | Game Boy |
| Game Boy Advanced |
| Hybrid |  |
| PC |  |  |
| Language | Programming Language |  |  |
| Script Language |  |  |
| Firm | Developer |  |  |
| Publisher |  |  |
| Distributor |  |  |

## Classes

Engine (Game Engine):

A game engine is a software framework designed for the creation and development of video games. Developers use them to create games for consoles, mobile devices and personal computers. The core functionality typically provided by a game engine includes a rendering engine (“renderer”) for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, scene graph, and may include video support for cinematics. The process of game development is often economized, in large part, by reusing/adapting the same game engine to create different games, or to make it easier to "port" games to multiple platforms.

Genre (Video game genre):

A video game genre is a classification assigned to a video game based on its gameplay interaction rather than visual or narrative differences.

Release:

Every video game has a release date in some cases these dates differ in terms of location and platform.

Dimension (Video game graphics):

A variety of computer graphic techniques have been used to display video game content throughout the history of video games. The predominance of individual techniques have evolved over time, primarily due to hardware advances and restrictions such as the processing power of central or graphics processing units.

Player Mode:

Video game:

A video game is an electronic game that involves human interaction with a user interface to generate visual feedback on a video device such as a TV screen or computer monitor.

Generation:

Video game consoles can be categorised according to the periods they were released.

Person:

This person class represents people. People involved in the development of the video game.

Director (video game director):

The game director is a creative lead over the whole project, and acts as the final arbiter of what is and is not permitted in the game.

Producer (Video game producer):

A video game producer is the person in charge of overseeing and funding development of a video game.

Artist (Video Game Artist):

A game artist is an artist who creates art for one or more types of games. Game artists are responsible for all of the aspects of game development that call for visual art.

Writer (Video game writer):

The video game writer is part of the design team, during pre-production, and creates the main plot of a video game but can also focus on the dialogue, the character creation and development.

Programmer (Video game programmer):

A game programmer is a software engineer, programmer, or computer scientist who primarily develops codebases for video games or related software, such as game development tools.

Designer (Video game designer):

Designers devise what a game consists of and how it plays. They plan and define all the elements of a game.

Composer:

Composer creates the soundtrack that accompanies video games.

Platform (Video game platform):

Platform is where any video game is executed. It may be the hardware or OS, even a web browser as long as the game is executed in it.

Web:

Web stands for browser based games. Games played over the internet using web browsers.

Console (Video Game Console):

A video game console is an electronic, digital or computer device that outputs a video signal or visual image to display a video game that one or more people can play.

Home (Home video game console):

A home video game console or simply home console is a video game device that is primarily used for home gamers.

Handheld (Handheld video game console):

A handheld game console is a small, portable self-contained video game console with a built-in screen, game controls, and speakers.

Hybrid (Hybrid video game console):

A hybrid video game console is a type of video game console that has its own designated display and controllers and can run independently, making it a hybrid of handheld and home video game consoles.

PlayStation:

PlayStation is a gaming brand created by Sony Interactive Entertainment. They have created several Video Game Console.

Xbox:

Xbox is a gaming brand created by Microsoft. They have created several Video Game Console.

PC:

PC stands for video games installed on personal computers this differs from operating system.

Language:

Language stands for languages designed to communicate instructions to a machine, particularly a computer.

Programming Language:

A programming language is a formal computer language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behaviour of a machine.

Script Language:

A script language is a programming language that supports scripts, programs written for a special run-time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator.

Firm:

Companies involve in development and distribution of video games.

Developer (Video Game Developer):

A video game developer is a company that specializes in video game development – the process and related disciplines of creating video games.

Publisher (Video Game Publisher):

A video game publisher is a company that publishes video games that they have either developed internally or have had developed by a video game developer.

Distributor (Video Game Distributor):

A video game publisher is a company that distributes the video game to shops.

## Object Property

|  |  |  |
| --- | --- | --- |
| **Object Property** | **Domain** | **Range** |
| HasArtist | Video game | Artist |
| HasComposer | Video game | Composer |
| HasDesigner | Video game | Designer |
| HasDeveloper | Video game | Developer |
| Engine |
| HasDimension | Engine | Dimension |
| HasDirector | Video game | Director |
| HasDistributor | Video game | Distributor |
| HasEngine | Video game | Engine |
| HasGenre | Video game | Genre |
| HasPlatform | Video game | Platform |
| HasPlayerMode | Video game | Player Mode |
| HasProducer | Video game | Producer |
| HasProgrammer | Video game | Programmer |
| HasProgrammingLanguage | Engine | Programming Language |
| HasPublisher | Video game | Publisher |
| HasReleaseDate | Video game | Release |
| Console |
| HasScriptLanguage | Engine | Script Language |
| HasWriter | Video game | Writer |

HasArtist:

HasArtist object property demonstrates the relationship between all instances of Video game which is the domain and range Artist.

HasComposer:

HasComposer object property demonstrates the relationship between all instances of Video game which is the domain and range Composer.

HasDesigner:

HasDesigner object property demonstrates the relationship between all instances of Video game which is the domain and range Designer.

HasDeveloper:

HasDeveloper object property demonstrates the relationship between all instances of Video game and Engine class as domain and range Developer.

HasDimension:

HasDimension object property demonstrates the relationship between all instances of Engine which is the domain and range Dimension.

HasDirector:

HasDirector object property demonstrates the relationship between all instances of Video game which is the domain and range Director.

HasDistributor:

HasDistributor object property demonstrates the relationship between all instances of Video game which is the domain and range Distributor.

HasEngine:

HasEngine object property demonstrates the relationship between all instances of Video game which is the domain and range Engine.

HasGenre:

HasGenre object property demonstrates the relationship between all instances of Video game which is the domain and range Genre.

HasPlatform:

HasPlatform object property demonstrates the relationship between all instances of Video game which is the domain and range Platform.

HasPlayerMode:

HasPlayerMode object property demonstrates the relationship between all instances of Video game which is the domain and range Player Mode.

HasProducer:

HasProducer object property demonstrates the relationship between all instances of Video game which is the domain and range Producer.

HasProgrammer:

HasProgrammer object property demonstrates the relationship between all instances of Video game which is the domain and range Programmer.

HasProgrammingLanguage:

HasProgrammingLanguage object property demonstrates the relationship between all instances of Engine which is the domain and range Programming Language.

HasPublisher:

HasPublisher object property demonstrates the relationship between all instances of Video game which is the domain and range Publisher.

HasReleaseDate:

HasReleaseDate object property demonstrates the relationship between all instances of Video game which is the domain and range Release Date.

HasScriptLanguage:

HasScriptLanguage object property demonstrates the relationship between all instances of Engine which is the domain and range Script Language.

HasWriter:

HasWriter object property demonstrates the relationship between all instances of Video game which is the domain and range Writer.

**Application Specification**

Protégé ontology editor

Protégé is developed by Stanford Medical Informatics. It has been selected as the ontology editor for the thesis because it is suitable to the project in many ways. It is a widely used free ontology editor especially for constructing and maintaining OWL ontologies. OWL is different than other ontology languages in that it supports a richer set of operators such as AND, OR, and Negation. Protégé supports all the advanced properties of OWL language and provides all the functionality to maintain OWL ontologies. Because of its wide popularity, it was easy to obtain support for it. 53 Plenty of tutorials, documentation and support forums are available for the editor. The functionality of Protégé can be extended by various plugins available online on the home page for Protégé. In addition, different wizards to ease the work are provided.

Jena Fuseki

a SPARQL end-point accessible over HTTP.It provides REST-style interaction with RDF data. It is a SPARQL server that provides REST-style SPARQL HTTP Update, SPARQL Query, and SPARQL Update using the SPARQL protocol over HTTP.

Jena

Jena is a Java framework for writing Semantic Web applications.

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. Jena is open source and grown out of work with the HP Labs Semantic Web Progrsamme. The Jena Framework includes:

• A RDF API

• Reading and writing RDF in RDF/XML, N3 and N-Triples

• An OWL API

• In-memory and persistent storage

• RDQL- a query language for RDF

• SPARQL query engine

Jena is a Java API which can be used to create and manipulate RDF graphs. Jena has object classes to represent graphs, resources, properties and literals. The interfaces representing resources, properties and literals are called Resource, Property and Literal respectively. A graph is called a model and is represented by the Model Interface [4]. To build applications exploiting the ontology, we need an API allowing us to access and manipulate directly an ontology written in OWL since it is reliable, mature and offers a good compatibility with most of the other RDFS/OWL API.

Sparqllib

Using this library, queries in the SPARQL language are transferred to Apache Jena Fuseki Server and results are gained to store other needed data, MySQL database was preferred because of good compatibility with PHP.

# Implementation

Use of systems or your own software to implement your design.

This Application will be implemented using an endpoint, a java application and a Web application.

Jena Fuski

Setting Up Jena Fuseki

1. Download latest version of jena fuseki
2. Open command prompt
3. Open jena fuski directory
4. Type fuseki-server
5. Open web browser
6. Type localhost:3030
7. Create new dataset
8. Upload ontology RDF triples

Type in sparql query to query dataset

# Evaluation and Use

SPARQL queries to prove your system works.

# Critical Reflection

Compare your application with alternative approaches.

# Reference

# Appendix