ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΝΝΟΙΟΛΟΓΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΟΝΤΟΤΗΤΩΝ ΣΕ ΔΙΑΔΙΚΤΥΑΚΕΣ ΠΛΑΤΦΟΡΜΕΣ ΚΟΙΝΩΝΙΚΗΣ ΔΙΚΤΥΩΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Γ. Σιδέρης

ΕΠΙΒΛΕΠΩΝ: Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Αθήνα Ιούνιος 2011

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ
ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΝΝΟΙΟΛΟΓΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΟΝΤΟΤΗΤΩΝ ΣΕ ΔΙΑΔΙΚΤΥΑΚΕΣ ΠΛΑΤΦΟΡΜΕΣ ΚΟΙΝΩΝΙΚΗΣ ΔΙΚΤΥΩΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Νικόλαος Γ. Σιδέρης

ΕΠΙΒΛΕΠΩΝ: Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 10η Ιουνίου 2011.

Αθήνα, Ιούνιος 2011

Θ.Βαρβαρίγου	Ε.Πρωτονοτάριος	Β.Λούμος
…..	….	….

....

Νικόλαος Γ. Σιδέρης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

ABSTRACT

Social Network Sites are the revolution in our days and beside the usage they found in our private life they started to have an important role to a new web market and web applications where people can fulfill their needs and develop some useful applications. Starting with a general review over social network sites their architectures and their use we move on the Semantic web and ontologies.

Ontologies are the tool for the Semantic Web to let machines understand and process data coming from the World Wide Web. There are many categories and different kind of ontologies and it is worth to review them. The concept of our thesis is to develop an ontology for the platform called SocIoS based on the media items from some important social network sites. SocIoS is a project started in NTUA in cooperation with other European technological institutes and universities. It will pave the way for building qualitative, functional and usable business applications exploiting the User Created Content and the Social Graph of users in Social Networks. By providing tools for cross-platform application development and deployment; support for SLAs and QoS; tools for UCC and social graph management; and most importantly, a usable framework to build services in and through Social Networks, SocIoS will provide incentives for the development of business applications. To meet our needs we started by reviewing the APIs of the SN sites and then we move on to the development of our ontology.

The ontology language we select to develop our ontology is OWL-DL and we use the protégé tool to design the ontology and a description logic reasoner (Pellet) to check the consistency of the ontology and automatically compute the class hierarchy.

Lastly apart from the ontology, I also created a web application that returns the demanded results from the API of DailyMotion based on the search keyword.

KEYWORDS: Ontology, Social network Site, Reasoner, API, Protégé, semantic Web, SocIoS

Special Thanks to

Finishing my diploma thesis I would like to thank some people who support me during my work and generally during the seven years that I had in the Technical university of Athens

First of all a great thanks to my supervisor Mr. Kostantinos Tserpes who dedicated a lot of his time to help me complete my thesis. My professors also who help me all these years with my courses and especially Mrs. Dora Varvarigou.

In conclusion I would like to thank my family for the support that I had during these tough months, providing me with the courage and the will to complete my task and also my friend Akis who help me when I encountered some problems with the web application.

Table of Contents

1. Introduction

The purpose of my thesis is to take a deeper view on the Social Networking (SN) sites and the ontologies which are the main and principal tool of the Semantic Web 3.0. We aim to develop an ontology for the media items of some of the most famous SN sites to be able to run application simultaneously at the sites and reclaim data from these sites for potentional new markets of internet services. To be able to use the data found in a site ontology is needed to make some basic rules and patterns for the machine process and understanding of these data.

The main requirements from doing this came out of a European funded project, named SocIoS (www.sociosproject.eu) in which among others, we attempted to model the media items "living" in SN sites. SocIoS, is a project led by NTUA with the collaboration of other European organizations and universities, that tries to benefit from the SN sites by leveraging a service marketplace using data from them.

The objective of this document is to explain how this modeling took place using an ontology-based data structure. At the same time, we built a mechanism that employs this ontology by applying it in Dailymotion, a media-based social network. The remaining of the document is dealing with introducing the reader to basic concepts of SN sites and programming using their APIs, with explaining the ontology that we analyzed and with the abovementioned implementation.

In detail: at the first four chapters we analyze the Social Network, in general terms, and the Social Network Sites. We talk about their history, the details about their architecture, their purpose and the most famous of them. Moreover we categorized them according to their purpose and the targeted area that they involved in.

Then in the chapter 5 we specify what is ontology, giving a definition and how it is used in the Semantic Web and how useful it is. Then we specify the ontologies languages that are used today and some of the tools that can design them. We categorize ontologies languages in two basic categories and give more attention in OWL and RDF which are the most famous ontology languages for the development of ontologies for the Web.

In the next chapter, we start the creation of the ontology. Firstly we studied the API guide of each SN site to understand how the applications of each site run and then we started to write down the fields for each site that describe the media items. Our ontology is connected with the media items of the SocIoS platform and it tries to define and make a common vocabulary for a media item of SocIoS according to the media items and their fields-values of the SN sites. That’s why it was necessary to have a better understanding for the APIs of the sites. We also made a web application which returns information about a video of DailyMotion according to a given keyword. The ontology language that we selected to develop our ontology is OWL-DL and we also used the Protégé Tool to design the ontology and after using the Pellet reasoner for the classification we have a completed ontology for the domain we talk about without worries about the consistency and of the ontology and the logical coherence.

2. Social Networks

2.1 Definition for a Social Network

Social network is a social structure made up of individuals (or organizations) called "nodes", which are tied (connected) by one or more specific types of interdependency, such as friendship, kinship, common interest, financial exchange, dislike, sexual relationships, or relationships of beliefs, knowledge or prestige.

Social network analysis views social relationships in terms of network theory consisting of *nodes* and *ties* (also called *edges*, *links*, or *connections*). Nodes are the individual actors within the networks, and ties are the relationships between the actors. The resulting graph-based structures are often very complex. There can be many kinds of ties between the nodes. Research in a number of academic fields has shown that social networks operate on many levels, from families up to the level of nations, and play a critical role in determining the way problems are solved, organizations are run, and the degree to which individuals succeed in achieving their goals.

In its simplest form, a social network is a map of specified ties, such as friendship, between the nodes being studied. The nodes to which an individual is thus connected are the social contacts of that individual. The network can also be used to measure social capital – the value that an individual gets from the social network. These concepts are often displayed in a social network diagram, where nodes are the points and ties are the lines.

Here is a photo example of a social network and the relationships involved in it:

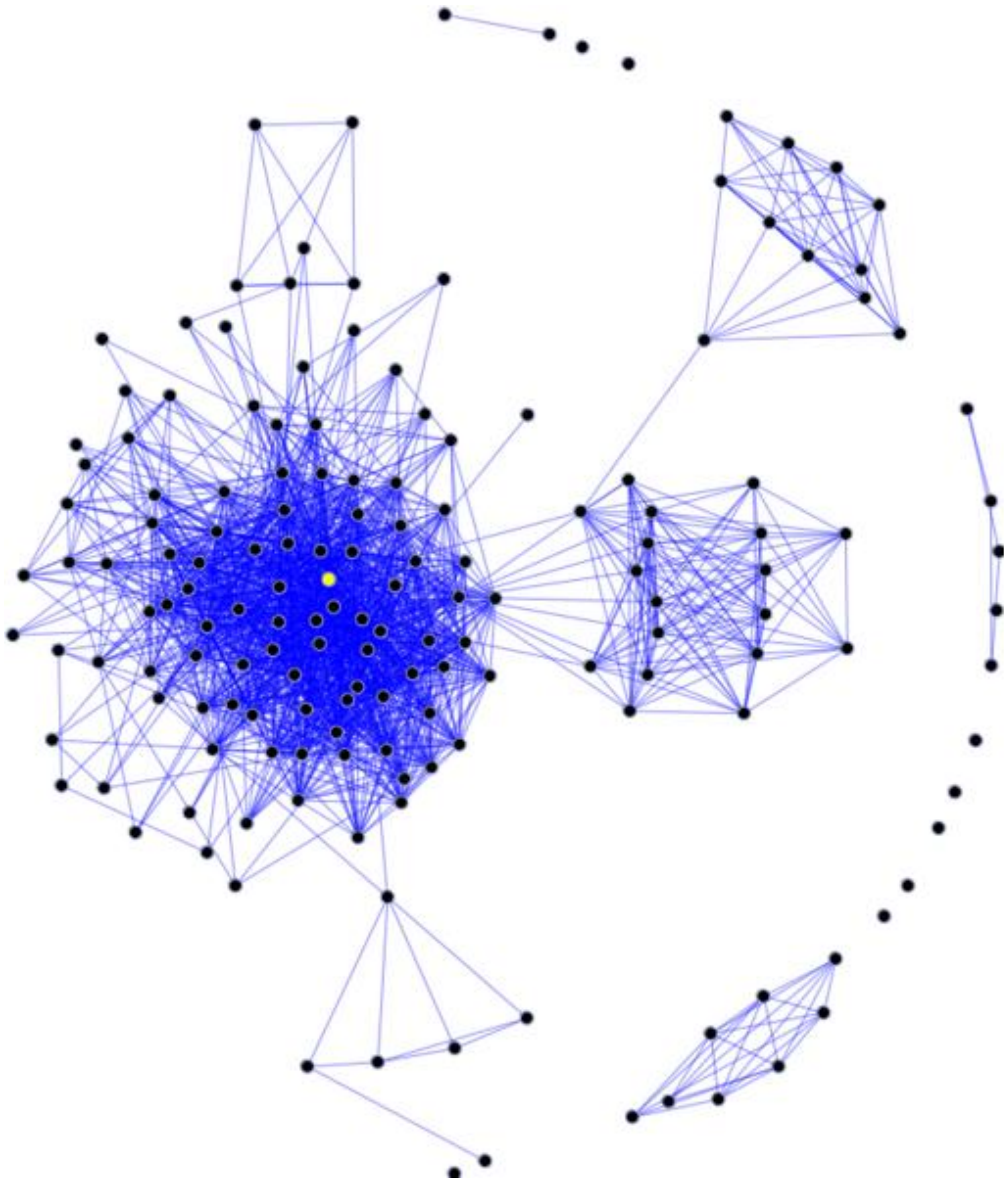


Figure 1: a social network site with it relationships

2.2 Social network analysis

Social network analysis (related to *network theory*) has emerged as a key technique in modern sociology. It has also gained a significant following in anthropology, biology, communication studies, economics, geography, information science, organizational studies, social psychology, and sociolinguistics, and has become a popular topic of speculation and study.

People have used the idea of "social network" loosely for over a century to connote complex sets of relationships between members of social systems at all scales, from interpersonal to international. In 1954, J. A. Barnes started using the term systematically to denote patterns of ties, encompassing concepts traditionally used by the public and those used by social scientists: bounded groups (e.g., tribes, families) and social categories (e.g., gender, ethnicity). Scholars such as S.D. Berkowitz, Stephen Borgatti, Ronald Burt, Kathleen Carley, Martin Everett, Katherine Faust, Linton Freeman, Mark Granovetter, David Knoke, David Krackhardt, Peter Marsden, Nicholas Mullins, Anatol Rapoport, Stanley Wasserman, Barry Wellman, Douglas R. White, and Harrison White expanded the use of systematic social network analysis.

Social network analysis has now moved from being a suggestive metaphor to an analytic approach to a paradigm, with its own theoretical statements, methods, social network analysis software, and researchers. Analysts reason from whole to part; from structure to relation to individual; from behavior to attitude. They typically either study *whole networks* (also known as *complete networks*), all of the ties containing specified relations in a defined population, or *personal networks* (also known as *egocentric networks*), the ties that specified people have, such as their "personal communities". The distinction between whole/complete networks and personal/egocentric networks has depended largely on how analysts were able to gather data. That is, for groups such as companies, schools, or membership societies, the analyst was expected to have complete information about who was in the network, all participants being both potential egos and alters. Personal/egocentric studies were typically conducted when identities of egos were known, but not their alters. These studies rely on the egos to provide information about the identities of alters and there is no expectation that the various egos or sets of alters will be tied to each other. A snowball network refers to the idea that the alters identified in an egocentric survey then become egos themselves and are able in turn to nominate additional alters. While there are severe logistic limits to conducting snowball network studies, a method for examining hybrid networks has recently been developed in which egos in complete networks can nominate alters otherwise not listed who are then available for all subsequent egos to see. The hybrid network may be valuable for examining whole/complete networks that are expected to include

important players beyond those who are formally identified. For example, employees of a company often work with non-company consultants who may be part of a network that cannot fully be defined prior to data collection.
Several analytic tendencies distinguish social network analysis:

- There is no assumption that groups are the building blocks of society: the approach is open to studying less-bounded social systems, from nonlocal communities to links among websites.
- Rather than treating individuals (persons, organizations, states) as discrete units of analysis, it focuses on how the structure of ties affects individuals and their relationships.
- In contrast to analyses that assume that socialization into norms determines behavior, network analysis looks to see the extent to which the structure and composition of ties affect norms.

The shape of a social network helps determine a network's usefulness to its individuals. Smaller, tighter networks can be less useful to their members than networks with lots of loose connections (weak ties) to individuals outside the main network. More open networks, with many weak ties and social connections, are more likely to introduce new ideas and opportunities to their members than closed networks with many redundant ties. In other words, a group of friends who only do things with each other already share the same knowledge and opportunities. A group of individuals with connections to other social worlds is likely to have access to a wider range of information. It is better for individual success to have connections to a variety of networks rather than many connections within a single network. Similarly, individuals can exercise influence or act as brokers within their social networks by bridging two networks that are not directly linked (called filling structural holes).

The power of social network analysis stems from its difference from traditional social scientific studies, which assume that it is the attributes of individual actors—whether they are friendly or unfriendly, smart or dumb, etc.—that matter. Social network analysis produces an alternate view, where the attributes of individuals are less important than their relationships and ties with other actors within the network. This approach has turned out to be useful for explaining many real-world phenomena, but leaves less room for individual agency, the ability for individuals to influence their success, because so much of it rests within the structure of their network.

Social networks have also been used to examine how organizations interact with each other, characterizing the many informal connections that link executives together, as well as associations and connections between individual employees at different organizations. For example, power within organizations often comes more from the degree to which an individual within a network is at the center of many relationships than actual job title. Social networks also play a key role in hiring, in business success, and in job performance. Networks provide ways for companies to gather information, deter competition, and collude in setting prices or policies.

2.3 References:

[1] Linton Freeman, *The Development of Social Network Analysis.* Vancouver: Empirical Press, 2006.
[2] Wellman, Barry and S.D. Berkowitz, eds., 1988. *Social Structures: A Network Approach*. Cambridge: Cambridge University Press.
[3] Hansen, William B. and Reese, Eric L. 2009. Network Genie User Manual. Greensboro, NC: Tanglewood Research.
[4] Freeman, Linton. 2006. *The Development of Social Network Analysis.* Vancouver: Empirical Pres, 2006; Wellman, Barry and S.D. Berkowitz, eds., 1988. *Social Structures: A Network Approach.* Cambridge: Cambridge University Press.
[5] Scott, John. 1991. *Social Network Analysis.* London: Sage.
[6] Wasserman, Stanley, and Faust, Katherine. 1994. *Social Network Analysis: Methods and Applications.* Cambridge: Cambridge University Press.

3. Social Network Sites

3.1 Abstract

Lastly social network pass from our lives to the Web live and particularly in many sites called as Social Networks (also called social networking sites). Social network sites (SNSs) are increasingly attracting the attention of academic and industry researchers intrigued by their affordances and reach. Here we describe features of SNSs and propose a comprehensive definition. We then present one perspective on the history of such sites, discussing key changes and developments. After briefly summarizing existing scholarship concerning SNSs, we conclude with some words about privacy rules.

Since their introduction, social network sites (SNSs) such as MySpace, Facebook, Cyworld, and Bebo have attracted millions of users, many of whom have integrated these sites into their daily practices. As of this writing, there are hundreds of SNSs, with various technological affordances, supporting a wide range of interests and practices. While their key technological features are fairly consistent, the cultures that emerge around SNSs are varied. Most sites support the maintenance of pre-existing social networks, but others help strangers connect based on shared interests, political views, or activities. Some sites cater to diverse audiences, while others attract people based on common language or shared racial, sexual, religious, or nationality-based identities. Sites also vary in the extent to which they incorporate new information and communication tools, such as mobile connectivity, blogging, and photo/video-sharing.

Scholars from disparate fields have examined SNSs in order to understand the practices, implications, culture, and meaning of the sites, as well as users' engagement with them. We begin by defining what constitutes a social network site and then present one perspective on the historical development of SNSs, drawing from personal interviews and public accounts of sites and their changes over time. Following this, we review recent scholarship on SNSs and attempt to contextualize and highlight key works.

3.2 A Definition for Social Network Sites

We define social network sites as web-based services that allow individuals to construct a public or semi-public profile within a bounded system, articulate a list of other users with whom they share a connection, and view and traverse their list of connections and those made by others within the system. The nature and nomenclature of these connections may vary from site to site.

While we use the term "social network site" to describe this phenomenon, the term "social networking sites" also appears in public discourse, and the two terms are often used interchangeably. We chose not to employ the term "networking" for two reasons: emphasis and scope. "Networking" emphasizes relationship initiation, often between strangers. While networking is possible on these sites, it is not the primary practice on many of them, nor is it what differentiates them from other forms of computer-mediated communication (CMC).

What makes social network sites unique is not that they allow individuals to meet strangers, but rather that they enable users to articulate and make visible their social networks. This can result in connections between individuals that would not otherwise be made, but that is often not the goal, and these meetings are frequently between "latent ties" who share some offline connection. On many of the large SNSs, participants are not necessarily "networking" or looking to meet new people; instead, they are primarily communicating with people who are already a part of their extended social network. To emphasize this articulated social network as a critical organizing feature of these sites, we label them "social network sites."

While SNSs have implemented a wide variety of technical features, their backbone consists of visible profiles that display an articulated list of Friends who are also users of the system. Profiles are unique pages where one can "type oneself into being". After joining an SNS, an individual is asked to fill out forms containing a series of questions. The profile is generated using the answers to these questions, which typically include descriptors such as age, location, interests, and an "about me" section. Most sites also encourage users to upload a profile photo. Some sites allow users to enhance their profiles by adding multimedia content or modifying their profile's look and feel. Others, such as Facebook, allow users to add modules ("Applications") that enhance their profile.

The visibility of a profile varies by site and according to user discretion. By default, profiles on Friendster and Tribe.net are crawled by search engines, making them visible to anyone, regardless of whether or not the viewer has an account. Alternatively, LinkedIn controls what a viewer may see based on whether she or he has a paid account. Sites like MySpace allow users to choose whether they want their profile to be public or "Friends only." Facebook takes a different approach—by default, users who are part of the same "network" can view each other's profiles, unless a profile owner has decided to deny permission to those in their network. Structural variations around visibility and access are one of the primary ways that SNSs differentiate themselves from each other.

After joining a social network site, users are prompted to identify others in the system with whom they have a relationship. The label for these relationships differs depending on the site—popular terms include "Friends," "Contacts," and "Fans." Most SNSs require bi-directional confirmation for Friendship, but some do not. These one-directional ties are sometimes labeled as "Fans" or "Followers," but many sites call these Friends as well. The term "Friends" can be misleading, because the connection does not necessarily mean friendship in the everyday vernacular sense, and the reasons people connect are varied.

The public display of connections is a crucial component of SNSs. The Friends list contains links to each Friend's profile, enabling viewers to traverse the network graph by clicking through the Friends lists. On most sites, the list of Friends is visible to anyone who is permitted to view the profile, although there are exceptions. For instance, some MySpace users have hacked their profiles to hide the Friends display, and LinkedIn allows users to opt out of displaying their network.

Most SNSs also provide a mechanism for users to leave messages on their Friends' profiles. This feature typically involves leaving "comments," although sites employ various labels for this feature. In addition, SNSs often have a private messaging feature similar to webmail. While both private messages and comments are popular on most of the major SNSs, they are not universally available.

Not all social network sites began as such. QQ started as a Chinese instant messaging service, LunarStorm as a community site, Cyworld as a Korean discussion forum tool, and Skyrock (formerly Skyblog) was a French blogging service before adding SNS features. Classmates.com, a directory of school affiliates launched in 1995, began supporting articulated lists of Friends after SNSs became popular. AsianAvenue, MiGente, and BlackPlanet were early popular ethnic community sites with limited Friends functionality before re-launching in 2005-2006 with SNS features and structure.

Beyond profiles, Friends, comments, and private messaging, SNSs vary greatly in their features and user base. Some have photo-sharing or video-sharing capabilities; others have built-in blogging and instant messaging technology. There are mobile-specific SNSs (e.g., Dodgeball), but some web-based SNSs also support limited mobile interactions (e.g., Facebook, MySpace, and Cyworld). Many SNSs target people from specific geographical regions or linguistic groups, although this does not always determine the site's constituency. Orkut, for example, was launched in the United States with an English-only interface, but Portuguese-speaking Brazilians quickly became the dominant user group (Kopytoff, 2004). Some sites are designed with specific ethnic, religious, sexual orientation, political, or other identity-driven categories in mind. There are even SNSs for dogs (Dogster) and cats (Catster), although their owners must manage their profiles.

While SNSs are often designed to be widely accessible, many attract homogeneous populations initially, so it is not uncommon to find groups using sites to segregate themselves by nationality, age, educational level, or other factors that typically segment society, even if that was not the intention of the designers.

3.3 The History of Social Network Sites

3.3.1 The Early Years

According to the definition above, the first recognizable social network site launched in 1997. SixDegrees.com allowed users to create profiles, list their Friends and, beginning in 1998, surf the Friends lists. Each of these features existed in some form before SixDegrees, of course. Profiles existed on most major dating sites and many community sites. AIM and ICQ buddy lists supported lists of Friends, although those Friends were not visible to others. Classmates.com allowed people to affiliate with their high school or college and surf the network for others who were also affiliated, but users could not create profiles or list Friends until years later. SixDegrees was the first to combine these features.

SixDegrees promoted itself as a tool to help people connect with and send messages to others. While SixDegrees attracted millions of users, it failed to become a sustainable business and, in 2000, the service closed. Looking back, its founder believes that SixDegrees was simply ahead of its time. While people were already flocking to the Internet, most did not have extended networks of friends who were online. Early adopters complained that there was little to do after accepting Friend requests, and most users were not interested in meeting strangers.

From 1997 to 2001, a number of community tools began supporting various combinations of profiles and publicly articulated Friends. AsianAvenue, BlackPlanet, and MiGente allowed users to create personal, professional, and dating profiles—users could identify Friends on their personal profiles without seeking approval for those connections. Likewise, shortly after its launch in 1999, LiveJournal listed one-directional connections on user pages. LiveJournal's creator suspects that he fashioned these Friends after instant messaging buddy lists —on LiveJournal, people mark others as Friends to follow their journals and manage privacy settings. The Korean virtual worlds site Cyworld was started in 1999 and added SNS features in 2001, independent of these other sites.

The next wave of SNSs began when Ryze.com was launched in 2001 to help people leverage their business networks. Ryze's founder reports that he first introduced the site to his friends—primarily members of the San Francisco business and technology community, including the entrepreneurs and investors behind many future SNSs. In particular, the people behind Ryze, Tribe.net, LinkedIn, and Friendster were tightly entwined personally and professionally. They believed that they could support each other without competing. In the end, Ryze never acquired mass popularity, Tribe.net grew to attract a passionate niche user base, LinkedIn became a powerful business service, and Friendster became the most significant, if only as "one of the biggest disappointments in Internet history".

In the next figure it is shown the way that SNSs had starting from the first site until the latest one.

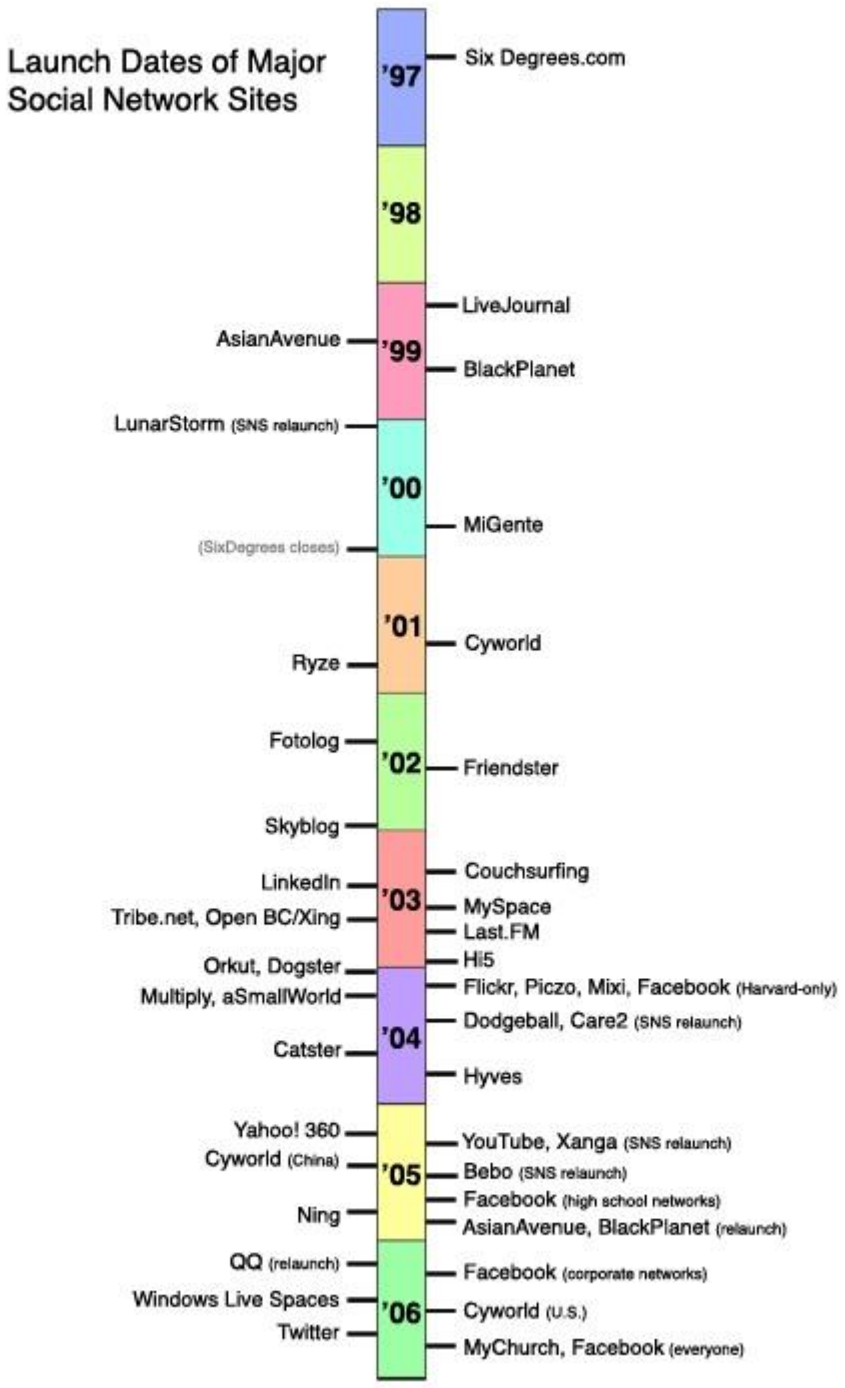


Figure 2. Timeline of the launch dates of many major SNSs and dates when community sites re-launched with SNS features

3.3.2. The Rise (and Fall) of Friendster

Friendster launched in 2002 as a social complement to Ryze. It was designed to compete with Match.com, a profitable online dating site. While most dating sites focused on introducing people to strangers with similar interests, Friendster was designed to help friends-of-friends meet, based on the assumption that friends-of-friends would make better romantic partners than would. Friendster gained traction among three groups of early adopters who shaped the site—bloggers, attendees of the Burning Man arts festival, and gay men and grew to 300,000 users through word of mouth before traditional press coverage began in May 2003

As Friendster's popularity surged, the site encountered technical and social difficulties. Friendster's servers and databases were ill-equipped to handle its rapid growth, and the site faltered regularly, frustrating users who replaced email with Friendster. Because organic growth had been critical to creating a coherent community, the onslaught of new users who learned about the site from media coverage upset the cultural balance. Furthermore, exponential growth meant a collapse in social contexts: Users had to face their bosses and former classmates alongside their close friends. To complicate matters, Friendster began restricting the activities of its most passionate users.

The initial design of Friendster restricted users from viewing profiles of people who were more than four degrees away (friends-of-friends-of-friends-of-friends). In order to view additional profiles, users began adding acquaintances and interesting-looking strangers to expand their reach. Some began Friendster launched in 2002 as a social complement to Ryze. It was designed to compete with Match.com, a profitable online dating site. While most dating sites focused on introducing people to strangers with similar interests, Friendster was designed to help friends-of-friends meet, based on the assumption that friends-of-friends would make better romantic partners than would. Friendster gained traction among three groups of early adopters who shaped the site—bloggers, attendees of the Burning Man arts festival, and gay men and grew to 300,000 users through word of mouth before traditional press coverage began in May 2003 massively collecting Friends, an activity that was implicitly encouraged through a "most popular" feature. The ultimate collectors were fake profiles representing iconic fictional characters: celebrities, concepts, and other such entities. These "Fakesters" outraged the company, who banished fake profiles and eliminated the "most popular" feature. While few people actually created Fakesters, many more enjoyed surfing Fakesters for entertainment or using functional Fakesters (e.g., "Brown University") to find people they knew.

The active deletion of Fakesters (and genuine users who chose non-realistic photos) signaled to some that the company did not share users' interests. Many early adopters left because of the combination of technical difficulties, social collisions, and a rupture of trust between users and the site. However, at the same time that it was fading in the U.S., its popularity skyrocketed in the Philippines, Singapore, Malaysia, and Indonesia.

3.3.3 SNSs Hit the Mainstream

From 2003 onward, many new SNSs were launched, prompting social software analyst Clay Shirky (2003) to coin the term YASNS: "Yet Another Social Networking Service." Most took the form of profile-centric sites, trying to replicate the early success of Friendster or target specific demographics. While socially-organized SNSs solicit broad audiences, professional sites such as LinkedIn, Visible Path, and Xing (formerly openBC) focus on business people. "Passion-centric" SNSs like Dogster help strangers connect based on shared interests. Care2 helps activists meet, Couchsurfing connects travelers to people with couches, and MyChurch joins Christian churches and their members. Furthermore, as the social media and user-generated content phenomena grew, websites focused on media sharing began implementing SNS features and becoming SNSs themselves. Examples include Flickr (photo sharing), Last.FM (music listening habits), and YouTube (video sharing).

With the plethora of venture-backed startups launching in Silicon Valley, few people paid attention to SNSs that gained popularity elsewhere, even those built by major corporations. For example, Google's Orkut failed to build a sustainable U.S. user base, but a "Brazilian invasion" made Orkut the national SNS of Brazil. Microsoft's Windows Live Spaces (a.k.a. MSN Spaces) also launched to lukewarm U.S. reception but became extremely popular elsewhere.

Few analysts or journalists noticed when MySpace launched in Santa Monica, California, hundreds of miles from Silicon Valley. MySpace was begun in 2003 to compete with sites like Friendster, Xanga, and AsianAvenue, according to co-founder Tom Anderson the founders wanted to attract estranged Friendster users. After rumors emerged that Friendster would adopt a fee-based system, users posted Friendster messages encouraging people to join alternate SNSs, including Tribe.net and MySpace. Because of this, MySpace was able to grow rapidly by capitalizing on Friendster's alienation of its early adopters. One particularly notable group that encouraged others to switch were indie-rock bands who were expelled from Friendster for failing to comply with profile regulations.

While MySpace was not launched with bands in mind, they were welcomed. Indie-rock bands from the Los Angeles region began creating profiles, and local promoters used MySpace to advertise VIP passes for popular clubs. Intrigued, MySpace contacted local musicians to see how they could support them. Bands were not the sole source of MySpace growth, but the symbiotic relationship between bands and fans helped MySpace expand beyond former Friendster users. The bands-and-fans dynamic was mutually beneficial: Bands wanted to be able to contact fans, while fans desired attention from their favorite bands and used Friend connections to signal identity and affiliation.

Furthermore, MySpace differentiated itself by regularly adding features based on user demand and by allowing users to personalize their pages. This "feature" emerged because MySpace did not restrict users from adding HTML into the forms that framed their profiles; a copy/paste code culture emerged on the web to support users in generating unique MySpace backgrounds and layouts.

Teenagers began joining MySpace en masse in 2004. Unlike older users, most teens were never on Friendster—some joined because they wanted to connect with their favorite bands; others were introduced to the site through older family members. As teens began signing up, they encouraged their friends to join. Rather than rejecting underage users, MySpace changed its user policy to allow minors. As the site grew, three distinct populations began to form: musicians/artists, teenagers, and the post-college urban social crowd. By and large, the latter two groups did not interact with one another except through bands. Because of the lack of mainstream press coverage during 2004, few others noticed the site's growing popularity.

Then, in July 2005, News Corporation purchased MySpace for $580 million (BBC, 2005), attracting massive media attention. Afterwards, safety issues plagued

MySpace. The site was implicated in a series of sexual interactions between adults and minors, prompting legal action. A moral panic concerning sexual predators quickly spread although research suggests that the concerns were exaggerated.

3.3.4 A Global Phenomenon

While MySpace attracted the majority of media attention in the U.S. and abroad, SNSs were proliferating and growing in popularity worldwide. Friendster gained traction in the Pacific Islands, Orkut became the premier SNS in Brazil before growing rapidly in India, Mixi attained widespread adoption in Japan, LunarStorm took off in Sweden, Dutch users embraced Hyves, Grono captured Poland, Hi5 was adopted in smaller countries in Latin America, South America, and Europe, and Bebo became very popular in the United Kingdom, New Zealand, and Australia. Additionally, previously popular communication and community services began implementing SNS features. The Chinese QQ instant messaging service instantly became the largest SNS worldwide when it added profiles and made friends visible while the forum tool Cyworld cornered the Korean market by introducing homepages and buddies.

Blogging services with complete SNS features also became popular. In the U.S., blogging tools with SNS features, such as Xanga, LiveJournal, and Vox, attracted broad audiences. Skyrock reigns in France, and Windows Live Spaces dominates numerous markets worldwide, including in Mexico, Italy, and Spain. Although SNSs like QQ, Orkut, and Live Spaces are just as large as, if not larger than, MySpace, they receive little coverage in U.S. and English-speaking media, making it difficult to track their trajectories.

3.3.5 Expanding Niche Communities

Alongside these open services, other SNSs launched to support niche demographics before expanding to a broader audience. Unlike previous SNSs, Facebook was designed to support distinct college networks only. Facebook began in early 2004 as a Harvard-only SNS. To join, a user had to have a harvard.edu email address. As Facebook began supporting other schools, those users were also required to have university email addresses associated with those institutions, a requirement that kept the site relatively closed and contributed to users' perceptions of the site as an intimate, private community.

Beginning in September 2005, Facebook expanded to include high school students, professionals inside corporate networks, and, eventually, everyone. The change to open signup did not mean that new users could easily access users in closed networks—gaining access to corporate networks still required the appropriate .com address, while gaining access to high school networks required administrator approval. Unlike other SNSs, Facebook users are unable to make their full profiles public to all users. Another feature that differentiates Facebook is the ability for outside developers to build "Applications" which allow users to personalize their profiles and perform other tasks, such as compare movie preferences and chart travel histories.

While most SNSs focus on growing broadly and exponentially, others explicitly seek narrower audiences. Some, like aSmallWorld and BeautifulPeople, intentionally restrict access to appear selective and elite. Others—activity-centered sites like Couchsurfing, identity-driven sites like BlackPlanet, and affiliation-focused sites like MyChurch—are limited by their target demographic and thus tend to be smaller. Finally, anyone who wishes to create a niche social network site can do so on Ning, a platform and hosting service that encourages users to create their own SNSs.

Currently, there are no reliable data regarding how many people use SNSs, although marketing research indicates that SNSs are growing in popularity worldwide. This growth has prompted many corporations to invest time and money in creating, purchasing, promoting, and advertising SNSs. At the same time, other companies are blocking their employees from accessing the sites. The rise of SNSs indicates a shift in the organization of online communities. While websites dedicated to communities of interest still exist and prosper, SNSs are primarily organized around people, not interests. Early public online communities such as Usenet and public discussion forums were structured by topics or according to topical hierarchies, but social network sites are structured as personal (or "egocentric") networks, with the individual at the center of their own community. This more accurately mirrors unmediated social structures, where "the world is composed of networks, not groups». The introduction of SNS features has introduced a new organizational framework for online communities, and with it, a vibrant new research context.

3.3.6 Impression Management and Friendship Performance

Like other online contexts in which individuals are consciously able to construct an online representation of self—such as online dating profiles and MUDS—SNSs constitute an important research context for scholars investigating processes of impression management, self-presentation, and friendship performance. In one of the earliest academic articles on SNSs, Boyd (2004) examined Friendster as a locus of publicly articulated social networks that allowed users to negotiate presentations of self and connect with others. Donath and Boyd (2004) extended this to suggest that "public displays of connection" serve as important identity signals that help people navigate the networked social world, in that an extended network may serve to validate identity information presented in profiles.

While most sites encourage users to construct accurate representations of them, participants do this to varying degrees. Marwick (2005) found that users on three different SNSs had complex strategies for negotiating the rigidity of a prescribed "authentic" profile, while Boyd (in press-b) examined the phenomenon of "Fakesters" and argued that profiles could never be "real." The extent to which portraits are authentic or playful varies across sites; both social and technological forces shape user practices. Skog (2005) found that the status feature on LunarStorm strongly influenced how people behaved and what they choose to reveal—profiles there indicate one's status as measured by activity (e.g., sending messages) and indicators of authenticity (e.g., using a "real" photo instead of a drawing).

Another aspect of self-presentation is the articulation of friendship links, which serve as identity markers for the profile owner. Impression management is one of the reasons given by Friendster users for choosing particular friends recognizing this, Zinman and Donath (2007) noted that MySpace spammers leverage people's willingness to connect to interesting people to find targets for their spam.

In their examination of LiveJournal "friendship," Fono and Raynes-Goldie (2006) described users' understandings regarding public displays of connections and how the Friending function can operate as a catalyst for social drama. In listing user motivations for Friending, Boyd (2006a) points out that "Friends" on SNSs are not the same as "friends" in the everyday sense; instead, Friends provide context by offering users an imagined audience to guide behavioral norms. Other work in this area has examined the use of Friendster Testimonials as self-presentational devices and the extent to which the attractiveness of one's Friends (as indicated by Facebook's "Wall" feature) impacts impression formation.

3.4 Networks and Network Structure

Social network sites also provide rich sources of naturalistic behavioral data. Profile and linkage data from SNSs can be gathered either through the use of automated collection techniques or through datasets provided directly from the company, enabling network analysis researchers to explore large-scale patterns of friending, usage, and other visible indicators (Hogan, in press), and continuing an analysis trend that started with examinations of blogs and other websites. For instance, Golder, Wilkinson, and Huberman (2007) examined an anonymous dataset consisting of 362 million messages exchanged by over four million Facebook users for insight into friending and messaging activities. Lampe, Ellison, and Steinfield (2007) explored the relationship between profile elements and number of Facebook friends, finding that profile fields that reduce transaction costs and are harder to falsify are most likely to be associated with larger number of friendship links. These kinds of data also lend themselves well to analysis through network visualization.

SNS researchers have also studied the network structure of Friendship. Analyzing the roles people played in the growth of Flickr and other social networks, Kumar, Novak, and Tomkins (2006) argued that there are passive members, inviters, and linkers "who fully participate in the social evolution of the network" . Scholarship concerning LiveJournal's network has included a Friendship classification scheme, an analysis of the role of language in the topology of Friendship, research into the importance of geography in friending, and studies on what motivates people to join particular communities. Based on Orkut data, Spertus, Sahami, and Büyükkökten (2005) identified a topology of users through their membership in certain communities; they suggest that sites can use this to recommend additional communities of interest to users. Finally, Liu, Maes, and Davenport (2006) argued that Friend connections are not the only network structure worth investigating. They examined the ways in which the performance of tastes (favorite music, books, film, etc.) constitutes an alternate network structure, which they call a "taste fabric."

3.5 Bridging Online and Offline Social Networks

Although exceptions exist, the available research suggests that most SNSs primarily support pre-existing social relations. Ellison, Steinfield, and Lampe (2007) suggest that Facebook is used to maintain existing offline relationships or solidify offline connections, as opposed to meeting new people. These relationships may be weak ties, but typically there is some common offline element among individuals who friend one another, such as a shared class at school. This is one of the chief dimensions that differentiate SNSs from earlier forms of public CMC such as newsgroups. Research in this vein has investigated how online interactions interface with offline ones. For instance, Lampe, Ellison, and Steinfield (2006) found that Facebook users engage in "searching" for people with whom they have an offline connection more than they "browse" for complete strangers to meet. Likewise, Pew research found that 91% of U.S. teens who use SNSs do so to connect with friends.

Given that SNSs enable individuals to connect with one another, it is not surprising that they have become deeply embedded in user's lives. In Korea, Cyworld has become an integral part of everyday life—Choi (2006) found that 85% of that study's respondents "listed the maintenance and reinforcement of pre-existing social networks as their main motive for “Cyworld use”. Likewise, Boyd (2008) argues that MySpace and Facebook enable U.S. youth to socialize with their friends even when they are unable to gather in unmediated situations; she argues that SNSs are "networked publics" that support sociability, just as unmediated public spaces do.

3.6 Privacy

Popular press coverage of SNSs has emphasized potential privacy concerns, primarily concerning the safety of younger users. Researchers have investigated the potential threats to privacy associated with SNSs. In one of the first academic studies of privacy and SNSs, Gross and Acquisti (2005) analyzed 4,000 Carnegie Mellon University Facebook profiles and outlined the potential threats to privacy contained in the personal information included on the site by students, such as the potential ability to reconstruct users' social security numbers using information often found in profiles, such as hometown and date of birth.

Acquisti and Gross (2006) argue that there is often a disconnect between students' desire to protect privacy and their behaviors, a theme that is also explored in Stutzman's (2006) survey of Facebook users and Barnes's (2006) description of the "privacy paradox" that occurs when teens are not aware of the public nature of the Internet. In analyzing trust on social network sites, Dwyer, Hiltz, and Passerini (2007) argued that trust and usage goals may affect what people are willing to share—Facebook users expressed greater trust in Facebook than MySpace users did in MySpace and thus were more willing to share information on the site.

In another study examining security issues and SNSs, Jagatic, Johnson, Jakobsson, and Menczer (2007) used freely accessible profile data from SNSs to craft a "phishing" scheme that appeared to originate from a friend on the network; their targets were much more likely to give away information to this "friend" than to a perceived stranger. Survey data offer a more optimistic perspective on the issue, suggesting that teens are aware of potential privacy threats online and that many are proactive about taking steps to minimize certain potential risks. Pew found that 55% of online teens have profiles, 66% of whom report that their profile is not visible to all Internet users. Of the teens with completely open profiles, 46% reported including at least some false information.

Privacy is also implicated in users' ability to control impressions and manage social contexts. Boyd (in press-a) asserted that Facebook's introduction of the "News Feed" feature disrupted students' sense of control, even though data exposed through the feed were previously accessible. Preibusch, Hoser, Gürses, and Berendt (2007) argued that the privacy options offered by SNSs do not provide users with the flexibility they need to handle conflicts with Friends who have different conceptions of privacy; they suggest a framework for privacy in SNSs that they believe would help resolve these conflicts.

SNSs are also challenging legal conceptions of privacy. Hodge (2006) argued that the fourth amendment to the U.S. Constitution and legal decisions concerning privacy are not equipped to address social network sites. For example, do police officers have the right to access content posted to Facebook without a warrant? The legality of this hinges on users' expectation of privacy and whether or not Facebook profiles are considered public or private.

3.7 References:

[7] Acquisti, A., & Gross, R. (2006). Imagined communities: Awareness, information sharing, and privacy on the Facebook. In P. Golle & G. Danezis (Eds.), Proceedings of 6th Workshop on Privacy Enhancing Technologies (pp. 36-58). Cambridge, UK: Robinson College.

[8] Adamic, L. A., Büyükkökten, O., & Adar, E. (2003). A social network caught in the Web. First Monday, 8 (6). Retrieved July 30, 2007 from http://www.firstmonday.org/issues/issue8_6/adamic/index.html

[9] Backstrom, L., Huttenlocher, D., Kleinberg, J., & Lan, X. (2006). Group formation in large social networks: Membership, growth, and evolution. Proceedings of 12th International Conference on Knowledge Discovery in Data Mining (pp. 44-54). New York: ACM Press.

[10] Bahney, A. (2006, March 9). Don't talk to invisible strangers. New York Times. Retrieved July 21, 2007 from http://www.nytimes.com/2006/03/09/fashion/thursdaystyles/09parents.html

[11] Barnes, S. (2006). A privacy paradox: Social networking in the United States. First Monday, 11 (9). Retrieved September 8, 2007 from http://www.firstmonday.org/issues/issue11_9/barnes/index.html

[12] BBC. (2005, July 19). News Corp in $580m Internet buy. Retrieved July 21, 2007 from http://news.bbc.co.uk/2/hi/business/4695495.stm

[13] Benzie, R. (2007, May 3). Facebook banned for Ontario staffers. The Star. Retrieved July 21, 2007 from http://www.thestar.com/News/article/210014

[14] Boyd, d. (2004). Friendster and publicly articulated social networks. Proceedings of ACM Conference on Human Factors in Computing Systems (pp. 1279-1282). New York: ACM Press.

[15] Boyd, d. (2006a). Friends, Friendsters, and MySpace Top 8: Writing community into being on social network sites. First Monday, 11 (12). Retrieved July 21, 2007 from http://www.firstmonday.org/issues/issue11_12/boyd/

[16] Boyd, d. (2006b, March 21). Friendster lost steam. Is MySpace just a fad? Apophenia Blog. Retrieved July 21, 2007 from http://www.danah.org/papers/FriendsterMySpaceEssay.html

[17] Boyd, d. (in press-a). Facebook's privacy trainwreck: Exposure, invasion, and social convergence. Convergence, 14 (1).

[18] Boyd, d. (in press-b). None of this is real. In J. Karaganis (Ed.), Structures of Participation. New York: Social Science Research Council.

[19] Boyd, d. (2008). Why youth (heart) social network sites: The role of networked publics in teenage social life. In D. Buckingham (Ed.), Youth, Identity, and Digital Media (pp. 119-142). Cambridge, MA: MIT Press.
[20] Boyd, d., & Heer, J. (2006). Profiles as conversation: Networked identity performance on Friendster. Proceedings of Thirty-Ninth Hawai'i International Conference on System Sciences. Los Alamitos, CA: IEEE Press.

4 Classification of Social Network Sites

4.1 Abstract

Social media as we mentioned above has become a popular way to share information and content with others. Social Networking Sites are the sites that host social media. For those new to social media, it is good to know what types of social network sites there are and what each does. Observing the list below, we can notice that some of the social networks may be more than one type of the categories.

4.2 Forums

Probably one of the first sites to allow for social interaction, forums have been around for a while. Forums are typically comprised of people with a similar interest. Users have conversations around a particular given topic and build up relationships with each other. They provide a great deal of information about a topic and are a great way to share your knowledge. Crunchyroll, Fotki and many other forums created at sites connecting with many different types like sports, education where people can sign change ideas propose solutions and discuss their topics. For example our university has its own forum where students can find help and notes about their lessons and exams. You can easily become a member, usually with a free sign up form, and then u can discuss about different subjects that you or another person can post. To ease your search in topics and find faster and easier what you might want, each forum may be categorized by the title of the topic been discussed. You can also send private messages to the other members of forum for a private conversation or you can report a post or a topic if you think it is not appropriate for you or someone else.

4.3 Blogs

People like to journal, and with the internet they wanted to share their lives more openly. Originally called web-logs, these personal journals have advanced. Some personal journals still exist, while other blogs discuss a particular niche or interest. Some of them are Blogster , Itsmy, LiveJournal , My Opera, Open Diary and Windows Live Spaces. These are some of the most famous blogs SN that may discuss about everything the “journal” wants. It also very common and recent to see in some sport, economic or politician sites blogs created by persons who want to share their ideas and exchange comments. About Blogster now, Blogster is a blogging community that features specific-interest blogs. Blogster maintains an online community of users who publish content, images, video and more. Blogster members can network and collaborate by creating a blog, building a personalized profile, creating friend lists, commenting on articles and interacting in an online community. Blogster is positioned on simplicity, and easy-to-use settings options. Unique characteristic on Blogster features a combination of blogging and social networking. Users create both their profiles and their blogs. They can also add other users with the same interests as friends, chat with other users, join groups, upload photos on a gallery, and even incorporate their blog RSS feed, Twitter and Flickr accounts.

Over time, Blogster has added several new features to its website like the Wall, a space on every user's profile page that allows members to post messages for the user to see, and Lifestream, which displays an aggregated chronological view of the users' activities within the site.

Blogster Groups was introduced on March 30, 2009, which aims to allow members to create professional or social groups. Members can network with same interests, share information, photos, and connections. Meanwhile, Blogster Photo System was introduced on June 26, 2009. Privacy settings can be set for individual photos, limiting the groups of members that can see a photo. For example, the privacy of a photo can be set so that only the user's friends can see the photo, while the privacy of another photo can be set so that all Blogster users can see it. Another feature of the Photos applications is the ability to embed the photo to a blog post and tag the photo. On October 29, 2009, Blogster groups were able to customize their blog with their own photos, design, and colors. Blogster launched its own Chat function on October 12, 2009, allowing users to chat their friends about different topics. On November 10, 2009 Blogster launched a new footer to allow members to chat one to one easily.

On October 30, 2009, Blogster introduced a new mail system. The inbox allows members to exchange and archive messages. Members may block contacts with whom they do not want to interact. Lastly members may also send virtual gifts to their friends.

Figure 3: a screenshot of blogster homepage

4.4 Micro-Blogging

Similar to blogs, this is a micro journal of what is happening right now. Microblog differs from a traditional blog in that its content is typically smaller in both actual and aggregate file size. Microblogs "allow users to exchange small elements of content such as short sentences, individual images, or video links. These sites can share what is going on in an individual life or can be information the individual wants to share. Major news events are now breaking online via microblogs. The best known microblog is Twitter. Some other famous micro-blogging sites are: Tumblr , Plurk , Presently . Focusing on Twitter we can say that Twitter.com is an online social network used by millions of people around the world to stay connected to their friends, family members and coworkers through their computers and mobile phones. Twitter is a website, owned and operated by Twitter Inc., which offers a social networking and microblogging service, enabling its users to send and read messages called tweets. The interface allows users to post short messages (up to 140 characters) that can be read by any other Twitter user. Users declare the people they are interested in following, in which case they get notified when that person has posted a new message. A user who is being followed by another user does not necessarily have to reciprocate by following them back, which makes the links of the Twitter social network directed. We call active users, the users that post at least twice. We also define the active time of an active user by the time that has elapsed between his first and last post. Twitter users are able to post direct and indirect updates. Direct posts are used when a user aims her update to a specific person, whereas indirect updates are used when the update is meant for anyone that cares to read it. Even though direct updates are used to communicate directly with a specific person, they are public and anyone can see them. Often times two or more users will have conversations by posting updates directed to each other.

4.5 PhotoSharing

Social Networking Sites are known for sharing of information, in this case site share photos. Users upload their pictures to Photo Sharing Sites. Rather than having to send individual pictures to family, you send a single link. You can tag your photos with keywords related to the image and allow people to comment. Some popular photosharing sites are Flickr, Picasa, Fotki, and My Opera . Let’s see some more details about Flickr.

Flickr offers two types of accounts: Free and Pro. Free account users are allowed to upload 300 MB of images a month and 2 videos. Also, if a free user has more than 200 photos on the site, they will only be able to see the most recent 200 in their photostream. The other photos that were uploaded are still stored on the site and links to these images in blog posts remain active. Free users can also contribute to a maximum of 10 photo pools. If a free account is inactive for 90 consecutive days, Flickr reserves the right to delete it. For a free account, no one (including the account owner) can access the original file. If the account is upgraded to a pro account, then the original files are available for download.

Pro accounts allow users to upload an unlimited number of images and videos every month and receive unlimited bandwidth and storage. Photos may be placed in up to 60 group pools, and Pro account users receive ad-free browsing and have access to account statistics. As soon as a Pro account expires, it reverts back to the restrictions of a free account, including Flickr reserving the right to delete an account that is "inactive for 90 consecutive days". Flickr asks photo submitters to organize images using tags (a form of metadata), which enable searchers to find images related to particular topics, such as place names or subject matter.

Flickr was also an early website to implement tag clouds, which provide access to images tagged with the most popular keywords. Flickr also enables users to organize their photos into "sets", or groups of photos that fall under the same heading. Sets can be displayed as a slideshow and shared by embedding them in websites. However, sets are more flexible than the traditional folder-based method of organizing files, as one photo can belong to one set, many sets, or none at all. Flickr's "sets", then, represent a form of categorical metadata rather than a physical hierarchy. Geotagging can be applied to photos in set. Any sets with geotagging can be related to a map using imapflicker. The resulting map can be embedded in a website. Sets may be grouped into "collections", and collections further grouped into higher-order collections. We have to define here the meaning of geotagging. Geotagging is the process of adding geographical identification metadata to various media such as photographs, video, websites, SMS messages, or RSS feeds and is a form of geospatial metadata. These data usually consist of latitude and longitude coordinates, though they can also include altitude, bearing, distance, accuracy data, and place names. It is commonly used for photographs, giving geotagged photographs.

Finally, Flickr offers a fairly comprehensive web-service API that enables programmers to create applications that can perform almost any function a user on the Flickr site can do.

4.6 Video Sharing

YouTube is the video sharing site that almost everyone has seen. Videos are shared online; keywords are added so people can search for those terms or for the video title. People can comment if the video owner allows. Like other forms of social media it allows for a more personal look of the actual user. Apart from YouTube some other video sharing sites are Gather.com, OneWorldTv, DailyMotion, and Google Video. Let’s give some more details about YouTube. In this Social Network site a few things that you can do without producing video content of your own:

- Commenting (on others videos, or channels)
- Sharing
- Creating playlists of your favorite YouTube videos
- Rate videos (1-5 stars)
- Favorite Videos (another playlist)

For those who are familiar with the various sharing options found on many blogs and social networking sites, you'll be glad to know that YouTube has this function as well. YouTube users can share a video using one of the popular social networking sites (including Twitter, MySpace, Facebook, Bebo, hi5). It also can be sent to one of your YouTube friends, or emailed.

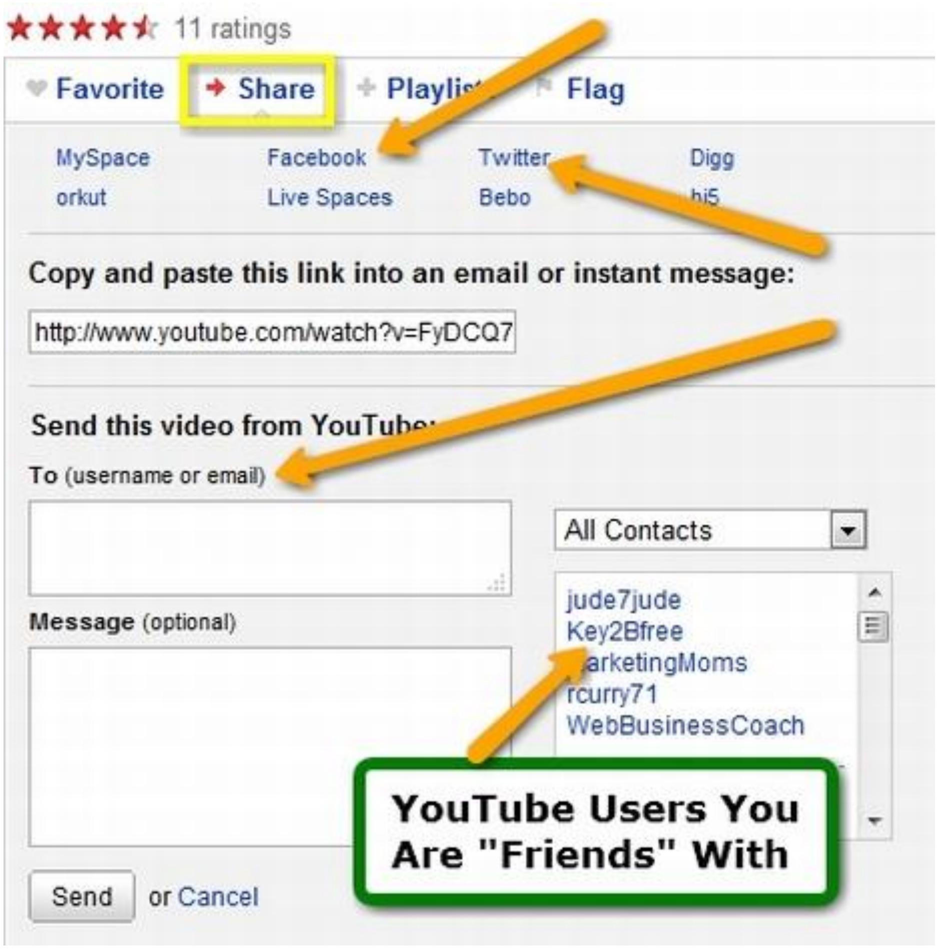
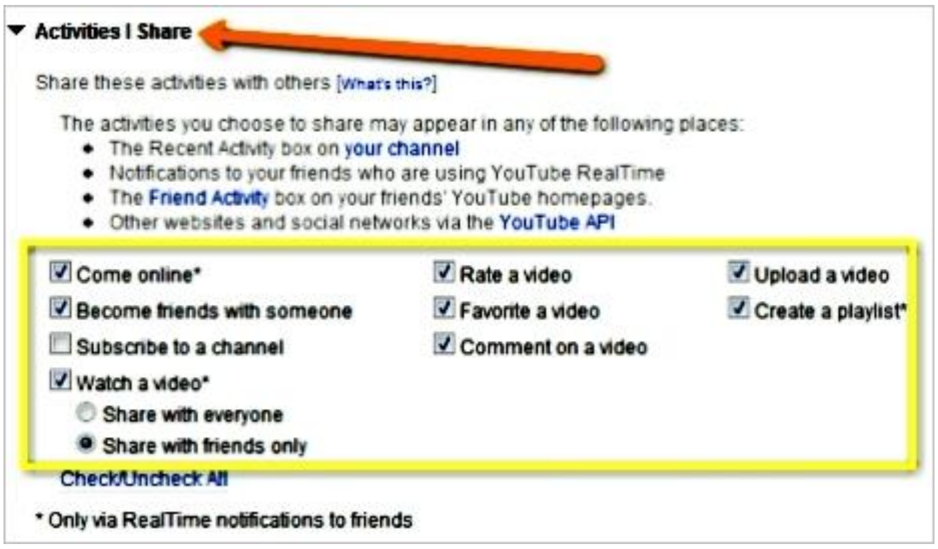


Figure 4: the possibilities of Youtube

YouTube also allows you to embed video content for a single video, a playlist, or a channel. While not comprehensive as Facebook or other feed activity updates, with the release of realtime updates, YouTube seems to be trending in that direction. The fact that this feature has been introduced along with other recent enhancements, there may be other activities will be streamed out as well. Here is a screenshot of the activities you can share:



Activities I Share

Share these activities with others [What's this?]

The activities you choose to share may appear in any of the following places:

- The Recent Activity box on your channel
- Notifications to your friends who are using YouTube RealTime
- The Friend Activity box on your friends' YouTube homepages.
- Other websites and social networks via the YouTube API

✓ Come online*	✓ Rate a video	✓ Upload a video
✓ Become friends with someone	✓ Favorite a video	✓ Create a playlist*
☐ Subscribe to a channel	✓ Comment on a video	
✓ Watch a video*		

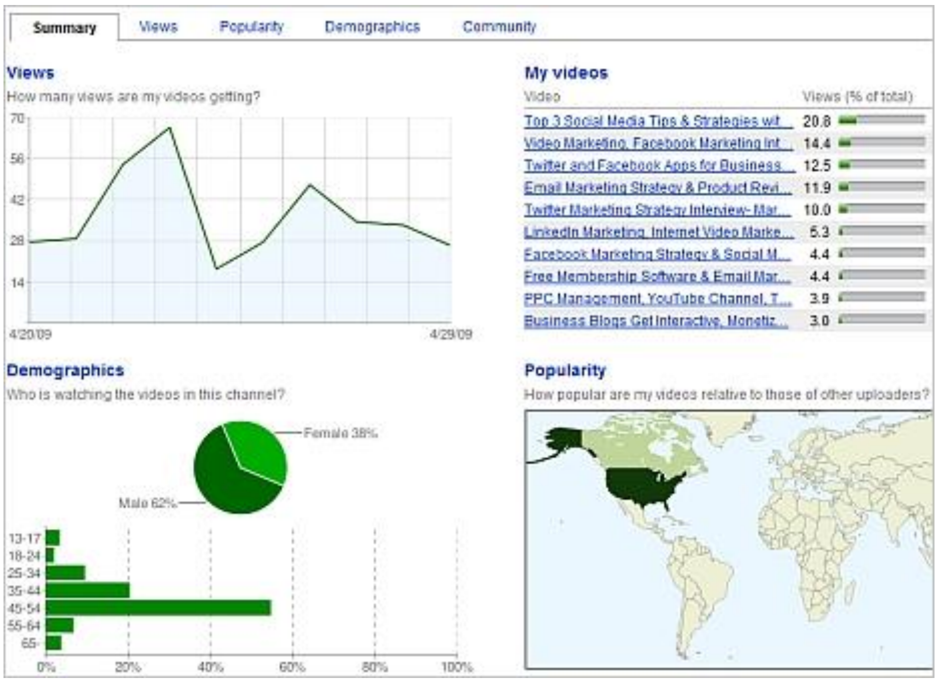
○ Share with everyone
● Share with friends only

Check/Uncheck All

* Only via RealTime notifications to friends

Figure 5: some of the activities you can share

One of the challenges with social media sites is the difficulty in measuring the effectiveness of your social activity. Understanding how your market responds to your content can help small business video publishers provide relevant content to grow their business and increase sales. Besides getting feedback from video ratings and user comments, there's Insight Statistics and Data.



My videos

Video	Views (% of total)
Top 3 Social Media Tips & Strategies wit...	20.8
Video Marketing, Facebook Marketing Int...	14.4
Twitter and Facebook Apps for Business...	12.5
Email Marketing Strategy & Product Revi...	11.9
Twitter Marketing Strategy Interview- Mar...	10.0
LinkedIn Marketing, Internet Video Marke...	5.3
Facebook Marketing Strategy & Social M...	4.4
Free Membership Software & Email Mar...	4.4
PPC Management, YouTube Channel, T...	3.9
Business Blogs Get Interactive, Monetiz...	3.0

Figure 6: a screenshot example measuring some of these statistics

- Insight: YouTube's reporting function helps you understand views, viewer demographics, popularity, and community.
- Community: The community tab of Insight reports on how other YouTube users are interacting with your video contents in the form of rating, comments, and favorite. Used properly, this information can drive future content you publish, making it more relevant and targeted.
- Hot Spots: The Hot Spots feature is available on a per video basis, and helps you understand the attention your video has at any point, compared to videos of similar length. Learn where you are losing interest, and make appropriate adjustments in future videos.

No other social networking site provides this kind of data for free.

4.7 Professional

Networking has long been touted as an asset in professional circles. Networking allows the person to connect with people and by those connections meet other professionals that person knows. Professional social media allows for that type of opportunity online. Thus if you move away from your home town you can still remain connected to your ex-coworkers and others in your profession. LinkedIn is one of the top professional social networking sites.

The purpose of LinkedIN site is to allow registered users to maintain a list of contact details of people they know and trust in business. The people in the list are called Connections. Users can invite anyone (whether a site user or not) to become a connection. This list of connections can then be used in a number of ways:

- A contact network is built up consisting of their direct connections, the connections of each of their connections (termed *second-degree connections*) and also the connections of second-degree connections (termed *third-degree connections*). This can be used to gain an introduction to someone a person wishes to know through a mutual, trusted contact.

- It can then be used to find jobs, people and business opportunities recommended by someone in one's contact network.
- Employers can list jobs and search for potential candidates.
- Job seekers can review the profile of hiring managers and discover which of their existing contacts can introduce them.
- Users can post their own photos and view photos of others to aid in identification.
- Users can now follow different companies and can get notification about the new joining and offers available.
- Users can save (i.e. bookmark) jobs which they would like to apply for.

LinkedIn also allows users to research companies with which they may be interested in working. When typing the name of a given company in the search box, statistics about the company are provided. These may include the ratio of female to male employees, the percentage of the most common titles/positions held within the company, the location of the company's headquarters and offices, or a list of present and former employees.

The feature LinkedIn Answers, similar to Yahoo! Answers, allows users to ask questions for the community to answer. This feature is free and the main difference from the latter is that questions are potentially more business-oriented, and the identity of the people asking and answering questions is known.

The searchable LinkedIn Groups feature allows users to establish new business relationships by joining alumni, industry, or professional and other relevant groups. LinkedIn groups can be created in any subjects and by any member of LinkedIn. Some groups are specialized groups dealing with a narrow domain or industry whereas others are very broad and generic in nature.

4.8 Social

After you have worked with people a while, they may become your friends. Then you have friends from church, from your school, your job or other activities you may participate and friends you have met in a wide variety of ways. Social networking sites are about being social, so there are sites purely for allowing users to stay in touch with people whom they know. In these sites people can chat, post their ideas, their videos, and pictures or even play some games to have fun. While MySpace was the leader, the current leader is Facebook and it is rapidly growing around the world. Some other famous sites like these are Hi5, Bebo, Badoo, Haboo, Netlog and also Twitter.

About Facebook now, it is maybe the most famous Social Network site. In Facebook users can create profiles with photos, lists of personal interests, contact information, and other personal information. Users can communicate with friends and other users through private or public messages and a chat feature. They can also create and join interest groups and "like pages" (formerly called "fan pages", until April 19, 2010), some of which are maintained by organizations as a means of advertising.

To allay concerns about privacy, Facebook enables users to choose their own privacy settings and choose who can see specific parts of their profile. For example you can exclude particular friends from watching your wall or your photos. The website is free to users, and generates revenue from advertising, such as banner ads. Facebook requires a user’s name and profile picture (if applicable) to be accessible by everyone. Users can control who sees other information they have shared, as well as who can find them in searches, through their privacy settings. The media often compares Facebook to MySpace, but one significant difference between the two websites is the level of customization. Another difference is Facebook’s requirement that users utilize their true identity, a demand that MySpace does not make. MySpace allows users to decorate their profiles using HTML and Cascading Style Sheets (CSS), while Facebook only allows plain text. Facebook has a number of features with which users may interact. They include the Wall, a space on every user's profile page that allows friends to post messages for the user to see, post videos, images or even some test like applications that connect you with a friend. Pokes, which allows users to send a virtual "poke" to each other (a notification then tells a user that they have been poked). Photos, where users can upload albums and photos and Status, which allows users to inform their friends of their whereabouts and actions. Depending on privacy settings, anyone who can see a user's profile can also view that user's Wall. In July 2007, Facebook began allowing users to post attachments to the Wall, whereas the Wall was previously limited to textual content only.

Over time, Facebook added features to its website. On September 6, 2006, a News Feed was announced, which appears on every user's homepage and highlights information including profile changes, upcoming events, and birthdays of the user's friends. This enabled spammers and other users to manipulate these features by creating illegitimate events or posting fake birthdays to attract attention to their profile or cause. Initially, the News Feed caused dissatisfaction among Facebook users, some complained it was too cluttered and full of undesired information, while others were concerned it made it too easy for others to track individual activities (such as relationship status changes, events, and conversations with other users). As time pasts Facebook add more features and facilitate the communication and the "spy" games in our friends' life. For example with the free API given in internet it gives the possibility to everyone creates his own application and then tries it to his Facebook or even his mobile phone. Besides we have to notice that Facebook is compatible with the majority of mobile phones and a new way for young people to communicate instead of texting or calling their friends.

4.9 Bookmarking

The major purpose of these social networking sites is the sharing of information. If you read a web page that is interesting, informative, or inspirational you may choose to bookmark it. By doing this you are saying that you like it. Your friends will learn what you liked and can choose to read it. Pages that many people bookmark become popular and drive others to that page as well. Digg, Delicious, and StumpleUpon are very popular bookmarking websites. Let's summarize some details about Delicious. Delicious uses a non-hierarchical classification system in which users can tag each of their bookmarks with freely chosen index terms. A combined view of everyone's bookmarks with a given tag is available; for instance, the URL "http://www.delicious.com/tag/wiki" displays all of the most recent links tagged "wiki". Its collective nature makes it possible to view bookmarks added by other users.

Delicious has a "hotlist" on its home page and "popular" and "recent" pages, which help to make the website a conveyor of Internet memes and trends. Delicious is one of the most popular social bookmarking services. Many features have contributed to this, including the website's simple interface, human-readable URL scheme, a novel domain name, a simple REST-like API, and RSS feeds for web syndication.

Use of Delicious is free. The source code of the site is not available, but a user can download his or her own data through the site's API in an XML or JSON format, or export it to a standard Netscape bookmarks format.

All bookmarks posted to Delicious are publicly viewable by default, although users can mark specific bookmarks as private, and imported bookmarks are private by default. The public aspect is emphasized; the site is not focused on storing private ("not shared") bookmark collections.

4.10 Other formats

There are other types of social networking sites out there; some are very similar to the ones mentioned here. Many sites have a major form of socializing and then use the other types and other ways for users to interact. People love to socialize and interact, so this is a growing field with ever more sites.

4.11 References:

[21] LinkedIn Answers unlocks the world's best source of business knowledge: trusted professionals". Linkedin.com. 2007-01-16. http://www.linkedin.com/static?key=press_releases_011607. Retrieved 2009-12-07.
[22]"Help: Free Accounts, Upgrading and Gifts". Flickr. http://www.flickr.com/help/limits/. Retrieved 2010-03-16.
[23]"flickr.com - Traffic Details from Alexa". Alexa Internet, Inc. http://www.alexa.com/siteinfo/flickr.com. Retrieved 2010-07-30
[24]"FB.com acquired by Facebook". January 11, 2011. http://namemon.com/news/1-latest-news/115-fbcom-acquired-by-facebook.
[25] "Edit Your Profile". Facebook. Archived from the original on February 27, 2008. http://web.archive.org/web/20080227212605/http://www.facebook.com/sitetour/profile.php. Retrieved March 7, 2008.
[26] "Search Privacy". Facebook. http://www.facebook.com/privacy/?view=search. Retrieved
[27] Barton, Zoe (April 28, 2006). "Facebook goes corporate". ZDNet. Archived from the original on May 26, 2008. http://web.archive.org/web/20080526001748/http://news.zdnet.com/2100-9588_22-6066533.html. Retrieved March 9, 2008.

[28]"Choose Your Privacy Settings". Facebook. http://www.facebook.com/settings/?tab=privacy. Retrieved September 10, 2009.
[29] Stone, Brad (May 25, 2007). "Facebook Expands Into MySpace’s Territory". *The New York Times*. http://www.nytimes.com/2007/05/25/technology/25social.html. Retrieved March 8, 2008.
[30] Ciccone, David (May 7, 2009). "Facebook Connect fully integrated into Mobility Today". http://mobilitytoday.com/news/009500/facebook_connect_mt. Retrieved September 10, 2010.
[31] Sullivan, Mark (July 24, 2007). "Is Facebook the New MySpace?". *PC World*. http://www.pcworld.com/article/id,134635-c,categories/article.html. Retrieved April 30, 2008.
[32] Cite: http://ezinearticles.com/?Types-of-Social-Networking-Sites&id=4019476
[33] Social networks that matter: Twitter under the microscope
[34] Bernardo A. Huberman1, Daniel M. Romero1;2 and Fang Wu1
[35] Whyte, Murray (June 1, 2008). "Tweet, Tweet – There's Been an Earthquake". *Toronto Star*. http://www.thestar.com/News/Ideas/article/434826. Retrieved February 23, 2011.
[36] Blogster (November 11, 2009). "Blogster New Footer"

5. Ontologies

5.1 A definition for ontologies

We are on the brink of a new generation of World Wide Web (WWW) called the Semantic Web. Unlike the existing WWW, where data content is primarily intended for human consumption, the Semantic Web will provide data whose content is also machine process able. This will enable a wide range of intelligent services such as information brokers, search agents; information filters etc., a process that Berners-Lee describes as "Bringing the Web to its full potential".

The development of ontologies will be central to this effort. Ontologies are metadata schemas, providing a controlled vocabulary of terms, each with an explicitly defined and machine process able semantics. By defining shared and common domain theories, ontologies help both people and machines to communicate more effectively. They will therefore have a crucial role in enabling content-based access, interoperability and communication across the Web, providing it with a qualitatively new level of service: the Semantic Web.

In order for ontologies to fulfill their role in the semantic integration of the Web, there will need to be some standardization of Web ontology languages. An ontology is a “a specification of a conceptualization ”, whereby a conceptualization is a collection of objects, concepts and other entities that are presumed to exist in some domain and that are tied together with some relationships. A conceptualization is a simplified view of the world, a way of thinking about some domain. Ontologies belong to the knowledge representation approaches that have been discussed above and they aim to provide a shared understanding of a domain both for the computers and for the humans. Thereby, ontology describes a domain of interest in such a formal way that it can be processed by computers. The outcome is that the computer system knows about this domain. Ontology is a formal classification schema, which has a hierarchical order and which is related to some domain. An ontology comprises the logical component of a “Knowledge Base”. Typically, a knowledge base consists of ontology, some data and also an inference mechanism. Ontology, comprising the logical component of the knowledge base, defines rules that formally describe how the field of interest looks like. The data can be any data related to this field of interest that is extracted from various resources such as databases, document collections, the Web etc. The inference mechanism would deploy rules in form of axioms, restrictions, logical consequences and other various methods based on the formal definition in the ontology over the actual data to produce more information out of the existing one. For example the ontology component a family knowledge base may say that a mother is a person, who as at least 2 children. The actual data, extracted from some database, may contain the information that Queen Sylvia of Sweden has 2 children. On the basis of this actual data and the formal definition of the mother, the knowledge base would infer that Queen Sylvia of Sweden is a mother.

Ontology may take a variety of forms, but necessarily it will include a *vocabulary of terms,* and some *specification of their meaning*. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms.

As we said above, Gruber defines ontology as "the specification of conceptualizations, used to help programs and humans share knowledge". The *conceptualization* is the couching of knowledge about the world in terms of entities (things, the relationships they hold and the constraints between them). The *specification* is the representation of this conceptualization in a concrete form. One step in this specification is the encoding of the conceptualization in a knowledge representation language.

Guarino defines ontology as "an intentional semantic structure which encodes the implicit rules constraining the structure of a piece of reality." A *formal* ontology has some underlying logical structure which allows us to *reason* about the concepts in the ontology. The goal is to create an agreed-upon vocabulary and semantic structure for exchanging information about that domain.

5.2 The most famous ontology languages

Ontologies are formal theories about a specific domain; therefore they require a formal logical language to express them. Most languages for formalizing ontologies seem to have emerged based on two approaches. The first-order predicate logic (FOL) with the frame based languages and the description logic-based languages.

Languages based on the former approach are more generic and they applied the frame concept to the structuring of language properties. Frame languages are usually software languages. They are rather focused on the recognition and description of objects and classes, and relations and interactions are considered as "secondary». In general, "frame" in this context means "something that can be/(has to be) fulfilled", whereas XML-RDF based languages such as OWL (Web Ontology Language) are description languages without going so far as to take the leap to first-order logic, specific for the development of Web ontologies. Even though RDF (Resource Description Framework) is most commonly mentioned as a language, it is rather a data model that is independent of any domain or implementation. It has been developed to provide meaning to the Web documents by decorating the Web documents with metadata in order to achieve terminological consensus on the Web. XML fell short for this purpose as it can help only for the syntactic structuring of the documents. Thus, researchers set out for developing languages that support semantics and that built on XML to benefit from its advantages such as the syntactic structure. RDFS and OWL languages are outcomes of such an attempt. Both languages are based on RDF, which is a data model developed for describing Web resources with metadata. As we said before RDF is not a language but a data model that is independent of any domain or implementation. As a data model RDF is graph based and it consists of nodes and edges. Nodes correspond to objects or *resources* and the edges correspond to *properties*. The labels on the nodes and on the edges are Uniform Resource Identifiers (URIs). Resources are all things being described by RDF expressions.

A resource may be an HTML document, it can be a part of a Web page e.g. a specific HTML or XML element within the document source or it can be a collection of pages e.g. an entire Web site.

Properties are specific attributes that describe resources and they have a defined meaning. A property together with its value for a specific resource makes a *statement* about that resource. Statements consist of a specific resource together with a named property plus the value of that property for that resource. Thus, an RDF statement is a triple, whose parts are the *subject,* the *predicate,* and the *object*. The object of a statement that is the property value, can be another resource, it can be a literal for example a resource specified by a URI, it can be a simple string or some other primitive datatype defined by XML. Reification is possible in RDF, so statements can be made about statements. RDF itself does not define any primitives for creating ontologies, it provides basis for several other ontology definition languages such as RDFS.

RDF Schema or RDFS has been developed in order to define the vocabulary used in RDF data models by specifying which kinds of properties apply to which kinds of objects, what values the objects can take and what kinds of relations between those objects exist. RDF Schema extends RDF by providing vocabulary to build logical object-oriented schema, including a simple typing system, sub-classes, sup-properties, inheritance, etc. Therefore, RDFS is considered as a first move towards an ontology language for the Web. RDFS offers a fix set of modeling primitives such as *rdfs:Class*, *rdf:Property* or the *rdfs:subClassOf* relationship to define RDF vocabularies for some specific application. In RDFS it is possible to define classes of classes, classes of properties, classes of literals that are strings, integers, booleans and so forth and classes of statements. Using RDFS properties, which are *rdf:type*, *rdfs:subClassOf* and *rdfs:subPropertyOf*, it is possible to define instanceOf relationship between resources and classes, subsumption relationship between classes and subsumption relationship between properties, respectively. Using *rdfs:domain* and *rdfs:range* properties it is possible to restrict the resources that can be subjects or objects of the property. As we have mentioned, RDFS is regarded as only a first move towards an ontology language because it is considered to be not expressive enough to qualify as a full ontology language. There are a number of things that cannot be said in RDFS. For example, *disjoint*, *union*, *intersection* and *complement* classes cannot be defined, cardinality restrictions are not present and properties cannot be declared as *transitive*, *symmetric* or *inverse* of each other. Yet, researchers have determined that such features are essential for an ontology language if it is to provide efficient reasoning support. Therefore, they have set out for the development of a more expressive ontology language.

OWL has been developed with such a motivation. It is an outcome of the collaborative efforts of US American and European researchers, whose goal has been to develop an ontology language other than RDFS that can be commonly adopted and that will facilitate the semantic interoperability on the Web. The Web Ontology Working Group of World Wide Web Consortium (W3C) describes OWL as *"a language designed for use by applications that need to process the content of information instead of just presenting information to humans"*. OWL ontologies have three components. These are classes, individuals, also called instances, and properties. In other formalisms properties are sometimes called as roles, relations, or attributes. OWL classes are interpreted as sets that contain individuals. Classes can be organized into a superclass-subclass hierarchy. When a class is declared to be the subclass of another, then every instance of the first class will also be the instance of the second one. In OWL DL, the superclass-subclass relationships can be computed automatically by an automatic inference mechanism. Classes can be declared to be *union*, *intersection* and *complement* classes. They can also be *equivalent* to each other. Finally, there are *enumerative* classes in OWL, which are classes that are defined by precisely listing the individuals that are the members of the class. Exactly these individuals make up the class. For example, the class *Kansas City Jazz Musician* can be defined as being made up of exactly the members (the individuals) "Count Basie" and "Dizzy Gillespie".

OWL individuals are the objects of the domain that we are interested in. Referring to the example above "Count Basie" and "Dizzy Gillespie" are some of the individuals of our domain, say, and the domain of Jazz Musicians. Further individuals could be then "Billy Oliday", "Miles Davis", "Thelonious Monk", "Duke Ellington" and so forth. OWL p*roperties* are binary relations on individuals i.e. they link two individuals together. There are two types of properties in OWL. *Object Properties* relate objects to other objects like in "Chet Baker" *plays Instrument* "Trumpet". *Datatype Properties*, relate objects to datatype values. For example, "Chet Baker" *died at the Age of* "59". Like in RDFS, properties in OWL have also *domains* and *ranges*. Similar to the case with classes, OWL properties may have *subproperties*, so that it is possible to form hierarchies of properties. For example, the property *is Jazz*

Musician may have the more specific property *is West Coast Jazz Musician* as its subproperty.

Restrictions in OWL are the *quantifier* restrictions, the *has-value* restriction and the *cardinality* restrictions. The quantifier restrictions are declared using the two OWL constructs *owl:allValuesFrom* (semantically equivalent to the universal quantifier “∀”) and the *owl:someValuesFrom* (semantically equivalent to the universal quantifier “ ∃”). The has-value restriction is declared using the construct *owl:hasValue* (“ ∋”). The *owl:hasValue* is a restriction on the value that some property can take by exactly specifying what that value is. For example, *is the city of Olympic Games 2004 owl:hasValue* “Athens”. Using the cardinality restrictions on properties, we can describe the class of individuals that have at least “ < ”, at most “ > ” or exactly “ = ” a specified number of relationships with other individuals or datatype values. Properties in OWL can be declared to be *transitive*like in *is Older than* property, they can be *symmetric* like in *is Married To* property or they can be *functional*, which states that a property has at most one value such as the property *age*.

One benefit of writing ontologies using OWL is that they can be processed by an inference mechanism i.e. by a reasoner. Thus, it is possible for a reasoner to check for subsumption relations in OWL ontologies and to compute the inferred class hierarchy. A reasoner can also check for consistency of OWL ontologies and can determine whether or not it is possible for a class to have any instances.

5.3 Tools and programs to design an Ontology

5.3.1 Ontolingua

Ontolingua provides a distributed collaborative environment to browse, create, edit, modify, and use ontologies. The ontology server architecture provides access to a library of ontologies, translators to other languages, and an editor to create and browse ontologies. Remote editors can browse and edit ontologies.

Applications can access any of the ontologies in the ontology library using OKBC. The original Ontolingua language, as described by Gruber, was designed to support the design and specification of ontologies with a clear logical semantics and built on KIF. KIF was extended with additional syntax to capture intuitive bundling of axioms into definitional forms with ontological significance; and a Frame Ontology to define object-oriented and frame-language terms. The Ontolingua Server has extended the original language in two ways. First, it provides explicit support for building ontological modules that can be assembled, extended, and refined in a new ontology. Second, it makes an explicit separation between ontology's presentation and representation.

5.3.2 Protégé

Protégé, which has been developed by Stanford's Medical Informatics Section in USA, is one of the most commonly used editors. It is compatible with the latest standards in the field of ontological engineering. Protégé allows direct editing in OWL, has a well developed import and export mechanism for OWL and for other recent ontology languages. It is open source and can be freely obtained from the World Wide Web. It is available on different platforms like Windows, Mac OS, Solaris, Linux, Unix and its capabilities can be extended by downloading various plug-ins that are designed for the tool. Classes (or concepts) of the domain to be modeled are visualized in a taxonomic hierarchy in Protégé. It is possible to define the instances of the model, so that for each class associated instances can be created directly in the model. The instances automatically become related to their classes by *instanceOf* relationship. Slots in Protégé describe properties of classes and instances. Facets specify constraints on allowed slot values. Axioms and rules cannot be explicitly represented, extra plug-ins need to be downloaded. Protégé does not allow synchronous editing of an ontology by multi-users, yet it is possible to import and export ontologies in different formats such as text files, database tables and RDF files. Since OWL has become standard ontology language for the Web, Protégé supports the editing of OWL ontologies by an OWL plug-in. This can be separately downloaded and be integrated into the editor. Thus primitives of the OWL language become available for use in Protégé to produce OWL ontologies.

The reasoner RACER provides reasoning support for Protégé. This tool can be separately downloaded to on the local computer. When it is run, it checks for the consistency of the ontologies created by Protégé and infers the classification tree of the ontology based on the subclass-superclass relationships. Several mailing lists such as *protégé-users*, *protégé discussion*, *protégé-beta* exist that are really active and that are helpful for the developers.

5.4 Classification for ontology languages

We can categorize ontology languages in 2 categories:

Weak languages:

- From conceptual modeling: E-R diagrams, UML diagrams.
- The Resource Description Framework (RDF) and its extension RDFS with schema vocabulary.

More expressive, logic-based languages:

- Description logics and the standard OWL (Web Ontology Language)
- Datalog and rule-based languages
- Conceptual Graphs
- Predicate logic

In general there are two strategies for achieving a standard:

- Defining a "small" set of modeling primitives that are common across the community, and defining a proper semantics for them
- Defining a "large" set of modeling primitives that are present in some of the approaches in a community and glue them together.

There are a (large) number of existing standards and standardization efforts which are either directly or indirectly involved in the provision of ontology language standards.

Some of the languages and efforts are listed below along with a brief description of the language or effort. Note that this list contains a number of languages or language specifications that are not actual standards. For example, OIL and DAML+OIL are language specifications that have not been accepted as standards by a recognized body. However, such languages have been used in much recent ontology work, and have taken on the mantle of *de facto* standards and will heavily influence any further development or standardization. For this reason they are included here.

Fol languages and frame based languages:

CKML	CLIPS	Conceptual-graphs	CycL
GRAIL	KIF	UML	OML
OKBC	F-logic		

Table 1

Description languages specific for Web ontologies:

DAML+OIL	XML-schema	XOL	SHOE
ebXML	RDF and RDF(s)	RuleML	OIL
OCML	Thesauri	Topic Maps	OWL 2
DAML-ONT	FOAF		

Table 2

5.4.1 CKML & OML

The Ontology Markup Language [OML] is an ontology specification language based on Conceptual Graphs. It allows the representation of concepts organized in taxonomies, relations and axioms in first order logic. Conceptual Knowledge Markup Language [CKML] is an extension of OML.

5.4.2 CLIPS

CLIPS is a productive development and delivery expert system tool which provides a complete environment for the construction of rule and/or object based expert systems. CLIPS is used throughout the public and private community including: all NASA sites and branches of the military, numerous federal bureaus, government contractors, universities, and many companies. CLIPS provides a cohesive tool for handling a wide variety of knowledge with support for three different programming paradigms: rule-based, object-oriented and procedural. Rule-based programming allows knowledge to be represented as heuristics, or "rules of thumb," which specify a set of actions to be performed for a given situation. Object-oriented programming allows complex systems to be modeled as modular components (which can be easily reused to model other systems or to create new components). The procedural programming capabilities provided by CLIPS are similar to capabilities found in languages such as C, Pascal, Ada, and LISP.

5.4.3 Conceptual Graphs

Conceptual graphs (CGs) are a system of logic based on the existential graphs of Charles Sanders Peirce and the semantic networks of artificial intelligence. They express meaning in a form that is logically precise, humanly readable, and computationally tractable. With a direct mapping to language, conceptual graphs serve as an intermediate language for translating computer-oriented formalisms to and from natural languages. With their graphic representation, they serve as a readable, but formal design and specification language. CGs have been implemented in a variety of projects for information retrieval, database design, expert systems, and natural language processing. A proposal for a CG standard has been developed, based on work from the NCITS.T2 Committee on Information Interchange and Interpretation. In addition, a more up to date working draft of an ISO CG Standard has also been produced.

5.4.4 CycL

The CycL language has been developed within the Cyc project. Cyc is an artificial intelligence project that attempts to assemble a comprehensive ontology and knowledge base of everyday common sense knowledge, with the goal of enabling AI applications to perform human-like reasoning. CycL is a formal language whose syntax derives from first-order predicate calculus (the language of formal logic). In order to express real-world expertise and common sense knowledge, however, it goes far beyond first order logic. The vocabulary of CycL consists of terms: semantic constants, non-atomic terms (NATs), variables, numbers, strings, etc. Terms are combined into meaningful CycL expressions, ultimately forming meaningful closed CycL sentences (with no free variables.) A set of CycL sentences forms a knowledge base.

5.4.5 DAML+OIL

DAML+OIL builds on work from both the OIL and DAML-ONT initiatives. It provides modeling primitives commonly found in frame-based languages (such as an asserted subsumption hierarchy and the description or definition of classes through slot fillers) and has a clean and well defined semantics. The version of March 2001 extends DAML+OIL (December 2000) with values from XML Schema datatypes. DAML+OIL is effectively alternative presentation syntax for a Description Logic (SHIQ with the addition of concrete datatypes) with an underlying RDFS based delivery mechanism.

The presence of the well-defined semantics in terms of SHIQ allow the use of description logic reasoners such as FaCT or RACER, in particular to support the tasks of classification and inconsistency detection.

The development of DAML+OIL was the responsibility of the Joint US/EU ad hoc Agent Markup LanguageCommittee (DAML Joint Committee). Many members of that committee are now part of the WebOnt Committee, so it is likely that further development will take place within that forum.

5.4.6 DAML-ONT

The original language proposal from the DAML program, now super ceded by DAML+OIL. Again, as with OIL, DAML-ONT was a proposal for a language specification rather than a ratified standard.

5.4.7 ebXML

The mission of ebXML is to provide an open XML-based infrastructure enabling the global use of electronic business information in an interoperable, secure and consistent manner by all parties. This is to be achieved through a modular suite of specifications that enables enterprises of any size and in any geographical location to conduct business over the Internet. EbXML will provide a standard method to exchange business messages, conduct trading relationships, communicate data in common terms and define and register business processes.

EbXML is sponsored by OASIS and UN/CEFACT. OASIS, the Organization for the Advancement of Structured Information Standards, is a non-profit, international consortium that creates interoperable industry specifications based on public standards such as XML and SGML, as well as others that are related to structured information processing. UN/CEFACT is the United Nations body whose mandate covers worldwide policy and technical development in the area of trade facilitation and electronic business. Headquartered in Geneva, it has developed and promoted many tools for the facilitation of global business processes including

UN/EDIFACT, the international EDI standard. Its current work programme includes such topics as Simpl-edi and Object Oriented edi and it strongly supports the development and implementation of open interoperable, global standards and specifications for electronic business.

5.4.8 FIPA

The Foundation for Intelligent Physical Agents (FIPA) is a non-profit organisation aimed at producingstandards for the interoperation of heterogeneous software agents.

FIPA98 (Specification Part 12 Ontology Service) deals with technologies enabling agents to manage explicit, declaratively represented ontologies. It specifies an ontology service provided to a community of agents by a dedicated Ontology Agent. The FIPA 98 specification is now considered to be obsolete. A more recent Ontology Service specification has been produced. The model of agent communication in FIPA is based on the assumption that two agents, who wish to converse, share a common ontology for the domain of discourse. It ensures that the agents ascribe the same meaning to the symbols used in the message. For a given domain, designers may decide to use ontologies that are explicit, declaratively represented (and stored somewhere) or, alternatively, ontologies that are implicitly encoded with the actual software implementation of the agent themselves and thus are not formally published to an ontology service. The FIPA specification deals with technologies enabling agents to manage explicit, declaratively represented ontologies. An ontology service for a community of agents is specified for this purpose. It is required that the service be provided by a dedicated agent, called an Ontology Agent (OA), whose role in the community is to provide some or all of the following services:

• discovery of public ontologies in order to access them,

• maintain (for example, register with the DF, upload, download, and modify) a set of public ontologies.
• translate expressions between different ontologies and/or different content languages,
• respond to query for relationships between terms or between ontologies, and,
• facilitate the identification of a shared ontology for communication between two agents.

As with the earlier specification, this deals only with the communicative interface to such a service while internal implementation and capabilities are left to developers. The interface is specified at the agent communication level as opposed to a computational API. Therefore, the specification defines the interaction protocols, the communicative acts and, in general, the vocabulary that agents must adopt when using this service.

5.4.9 F-Logic

F-Logic is a deductive, object oriented database language which combines the declarative semantics and expressiveness of deductive database languages with the rich data modeling capabilities supported by the object oriented data model. The basic constructs of F-Logic are objects. Objects model real world entities and are internally represented by object identifiers which are independent of their properties. Following the object oriented paradigm, objects may be organized in classes. Furthermore, methods represent relationships between objects. F-Logic forms the core of systems such as Ontobroker.

5.4.10 GRAIL

The GRAIL formalism was a product of the European GALEN project, now the responsibility of the OpenGALEN organization. GRAIL is effectively a description logic with a primitive role hierarchy, transitive roles and restricted forms of concept inclusion axioms. The GALEN Terminology architecture also includes functionality to support lexical operations and multilingual support. The language has primarily been used to produce models of medical terminology (the GALEN Common Reference Model [CRM]).

A suite of modeling tools known as the OpenKnoME are available for GRAIL modeling.

5.4.11 KIF

Knowledge Interchange Format KIF provides a declarative language for describing knowledge and for the interchange of knowledge among disparate programs.

KIF has a declarative semantics (i.e. the meaning of expressions in the representation can be understood without appeal to an interpreter for manipulating those expressions). It is logically comprehensive i.e. it provides for the expression of arbitrary sentences in the first-order predicate calculus, and it provides for the representation of knowledge about knowledge. The following are some of its features:

• Declarative semantics. It is possible to understand the meaning of expressions in the language without appealing to an interpreter for manipulating the expressions.

• Logically comprehensive. It provides for the expression of arbitrary sentences in predicate calculus.

• Metaknowledge. This allows us to make all knowledge representation explicit and permit us to introduce new knowledge representation constructs without changing the language.

• Translatability. It enables practical means of translating declarative knowledge bases to and from typical knowledge representation languages.

• Readability. Although KIF is not intended as a language for interaction with humans, it is useful for describing representation language semantics and assisting humans with knowledge base translation problems.

KIF is not intended as a primary language for interaction with human users (though it can be used for this purpose). Different programs can interact with their users in whatever forms are most appropriate to their applications (for example frames, graphs, charts, tables, diagrams, natural language, and so forth). As a pure specification language, KIF does not include commands for knowledge base query or manipulation. OKBC is complementary to such language specifications.

There is a "draft proposed American National Standard" (dpANS) for KIF. The IEEE Standard Upper Ontology (SUO) Study Group Knowledge Interchange Format group [SUO Study Group] is also working towards a standardization of KIF in order to support the SUO project.

5.4.12 OCML

The Operational Conceptual Modeling Language (OCML) is a modeling language from the Open University's (OU) Knowledge Media Institute (KMi). It supports the construction of knowledge models by means of several types of constructs. It allows the specification and operationalization of functions, relations, classes, instances and rules. It also includes mechanisms for defining ontologies and problem solving methods, the main technologies developed in the knowledge modeling area. About a dozen projects in the (KMi) are currently using OCML to provide modeling support for applications in areas such as knowledge management, ontology development, e-commerce and knowledge based system development. OCML modeling is also supported by a large library of reusable models, providing a useful resource for the knowledge modeling community.

5.4.13 OIL

The Ontology Inference Layer (OIL) was a proposal for a web-based representation and inference layer for ontologies, which combined the widely used modeling primitives from frame-based languages with the formal semantics and reasoning services provided by description logics. It is compatible with RDF Schema, and includes a precise semantics (OIL Semantics) for describing term meanings (and thus also for describing implied information). OIL presents a layered approach to a standard ontology language. Each additional layer adds functionality and complexity to the previous layer. This is done such that agents (humans or machines) who can only process a lower layer can still partially understand ontologies that are expressed in any of the higher layers. Although not an official standard endorsed by a standards body, OIL has proved a major influence on the development of DAML+OIL, with many of the ideas developed in OIL being present in the DAML+OIL language.

5.4.14 OKBC

Open Knowledge Base Connectivity (OKBC) is not an ontology representation language but is an application programming interface for accessing knowledge bases stored in knowledge representation systems (KRSs).

OKBC is a successor of Generic Frame Protocol (GFP) which was primarily aimed at systems that can be viewed as frame representation systems and was jointly developed by Artificial Intelligence Center of (SRI) International and the Knowledge Systems Laboratory (KSL) of Stanford University.

OKBC provides a uniform model of KRSs based on a common conceptualization of classes, individuals, slots, facets, and inheritance. OKBC is defined in a programming language independent fashion, and has existing implementations in Common Lisp, Java, and C. The protocol transparently supports networked as well as direct access to KRSs and knowledge bases.

OKBC consists of a set of operations that provide a generic interface to underlying KRSs. This interface isolates an application from many of the idiosyncrasies of a specific KRS and enables the development of tools (e.g., graphical browsers, frame editors, analysis tools, and inference tools) that operate on many KRSs. It has been successfully used in several ongoing projects at SRI and Stanford University. The development of OKBC is being overseen by a working group, chaired by Richard Fikes.

5.4.15 RDF and RDF(S)

RDF (Resource Description Framework) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. XML is a universal metalanguage for defining markup. It provides a uniform framework, and a set of tools like parsers, for interchange of data and metadata between applications. However, XML does not provide any means of talking about the semantics (meaning) of data. For example, there is no intended meaning associated with the nesting of tags, it is up to each application to interpret the nesting. Let us illustrate this point using an example. Suppose we want to express the following fact: David Billington is a lecturer of Discrete Mathematics.

There are various ways of representing this sentence in XML. Three possibilities are:

```
<course name="Discrete Mathematics">
<lecturer>David Billington</lecturer>
</course>

<lecturer name="David Billington">
<teaches>Discrete Mathematics</teaches>
</lecturer>

<teachingOffering>
<lecturer>David Billington</lecturer>
<course>Discrete Mathematics</course>
</teachingOffering>
```

Note that the first two formalizations include essentially an opposite nesting although they represent the same information. So there is no standard way of assigning meaning to tag nesting. Although often called a “language” RDF (Resource Description Framework) is essentially a data model. Its basic building block is an object-attribute-value triple, called a statement. The preceding sentence about Billington is such a statement. Of course, an abstract data model needs a concrete syntax in order to be represented and transmitted, and RDF has been given syntax in XML. As a result, it inherits the benefits associated with XML. However, it is important to understand that other syntactic representations of RDF, not based on XML, are also possible. XML-based syntax is not a necessary component of the RDF model. RDF is domain-independent in that no assumptions about a particular domain of use are made. It is up to users to define their own terminology in a schema language called RDF Schema (RDFS). The name RDF Schema is now widely regarded as an unfortunate choice. It suggests that RDF Schema has a similar relation to RDF as XML Schema has to XML, but in fact this is not the case. XML Schema constrains the structure of XML documents, whereas RDF Schema defines the vocabulary used in RDF data models. In RDFS we can define the vocabulary, specify which properties apply to which kinds of objects and what values they can take, and describe the relationships between objects. For example, we can write: *Lecturer* is a subclass of *academic staff member*.

This sentence means that all lecturers are also academic staff members. It is important to understand that there is an intended meaning associated with "is a subclass of". It is not up to the application to interpret this term; its intended meaning must be respected by all RDF processing software. Through fixing the semantics of certain ingredients, RDF/RDFS enables us to model particular domains. We illustrate the importance of RDF Schema with an example. Consider the following XML elements:

```
<academicStaffMember>Grigoris
Antoniou</academicStaffMember>
<professor>Michael Maher</professor>
<course name="Discrete Mathematics">
     <isTaughtBy>David Billington</isTaughtBy>
</course>
```

Suppose we want to collect all academic staff members. A path expression in Xpath might be:

//academicStaffMember

The result is only Grigoris Antoniou. While correct from the XML viewpoint, this answer is semantically unsatisfactory. Human readers would have also included Michael Maher and David Billington in the answer because

• all professors are academic staff members (that is, professor is a subclass of academicStaffMember);
• courses are only taught by academic staff members.

This kind of information makes use of the semantic model of the particular domain and cannot be represented in XML or in RDF but is typical of knowledge written in RDF Schema. Thus RDFS makes semantic information machine accessible, in accordance with the Semantic Web vision.

RDF emphasizes facilities to enable automated processing of Web resources. The broad goal of RDF is to define a mechanism for describing resources that makes no assumptions about a particular application domain, nor defines (a priori) the semantics of any application domain. The definition of the mechanism should be domain neutral, yet the mechanism should be suitable for describing information about any domain. As a result of many communities coming together and agreeing on basic principles of metadata representation and transport, RDF has drawn influence from several different sources. The main influences have come from the Web standardization community itself in the form of HTML metadata and PICS, the library community, the structured document community in the form of SGML and more importantly XML, and also the knowledge representation (KR) community.

There are also other areas of technology that contributed to the RDF design; these include object-oriented programming and modeling languages, as well as databases. While RDF draws from the KR community, readers familiar with that field are cautioned that RDF does not specify a mechanism for reasoning.

RDF: Basic Ideas

The fundamental concepts of RDF are resources, properties, and statements. We can think of a resource as an object, a “thing” we want to talk about. Resources may be authors, books, publishers, places, people, hotels, rooms, search queries, and so on. Every resource has a URI, a Uniform Resource Identifier. A URI can be a URL (Uniform Resource Locator, or Web address) or some other kind of unique identifier; note that an identifier does not necessarily enable *access* to a resource. URI schemes have been defined not only for Web locations but also for such diverse objects as telephone numbers, ISBN numbers, and geographic locations. There has been a long discussion about the nature of URIs, even touching philosophical questions (for example, what is an appropriate unique identifier for a person?), but we will not go into detail here. In general, we assume that a URI is the identifier of a Web resource.

Properties

Properties are a special kind of resources; they describe relations between resources, for example “written by”, “age”, “title”, and so on. Properties in RDF are also identified by URIs (and in practice by URLs). This idea of using URIs to identify “things” and the relations between them is quite important. This choice gives us in one stroke a global, worldwide, unique naming scheme. The use of such a scheme greatly reduces the homonym problem that has plagued distributed data representation until now.

Statements

Statements assert the properties of resources. A statement is an object attribute-value triple, consisting of a resource, a property, and a value. Values can either be resources or literals. Literals are atomic values (strings), the structure of which we do not discuss further.

The foundation of RDF, in other words, is a model for representing named properties and property values. The RDF model draws on well-established principles from various data representation communities. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources and an RDF model can therefore resemble an entity-relationship diagram. (More precisely, RDF Schemas — which are themselves instances of RDF data models — are ER diagrams.) In object-oriented design terminology, resources correspond to objects and properties correspond to instance variables.

The RDF data model is a syntax-neutral way of representing RDF expressions. The RDF data model, as specified in RDFMS, defines a simple model for describing interrelationships among resources in terms of named properties and values. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources. As such, the RDF data model can therefore resemble an entity-relationship diagram. The RDF data model, however, provides no mechanisms for declaring these properties, nor does it provide any mechanisms for defining the relationships between these properties and other resources. That is the role of RDF Schema.

Resource description communities require the ability to say certain things about certain kinds of resources. For describing bibliographic resources, for example, descriptive attributes including "author", "title", and "subject" are common. For digital certification, attributes such as "checksum" and "authorization" are often required. The declaration of these properties (attributes) and their corresponding semantics are defined in the context of RDF as an RDF schema. A schema defines not only the properties of the resource (e.g., title, author, subject, size, color, etc.) but may also define the kinds of resources being described (books, Web pages, people, companies, etc.).

Here is an example of an RDF schema:

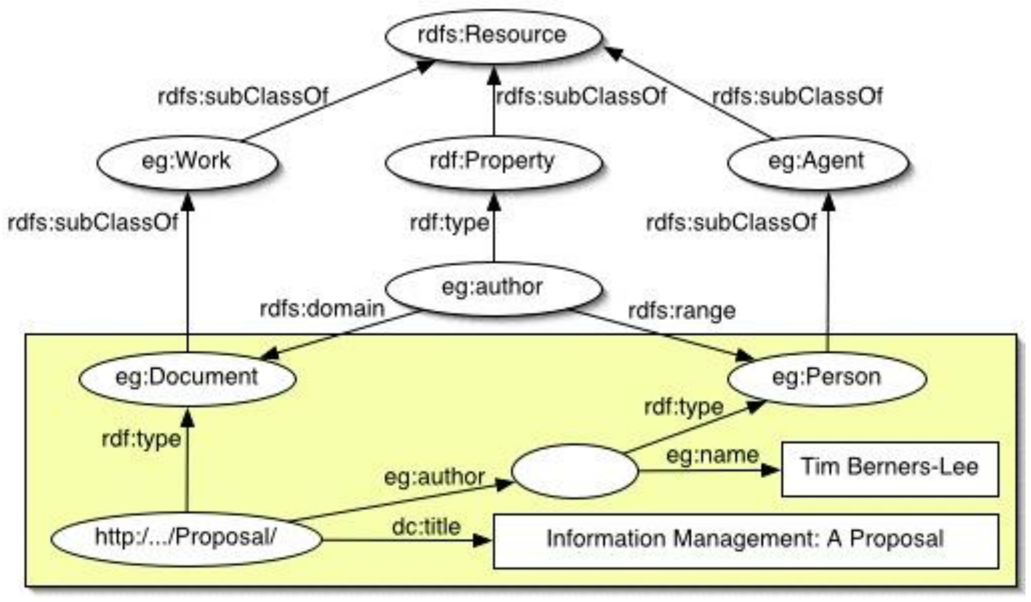


Figure 7: An RDF schema

The RDFS specification does not specify a vocabulary of descriptive elements such as "author". Instead, it specifies the mechanisms needed to define such elements, to define the classes of resources they may be used with, to restrict possible combinations of classes and relationships, and to detect violations of those restrictions. More succinctly, the RDF Schema mechanism provides a basic type system for use in RDF models. It defines resources and properties such as [rdfs:Class] and [rdfs:subClassOf] that are used in specifying application-specific schemas.

RDF(S) itself, however, lacks the necessary rich concept forming operators that one would expect in an ontology representation language, and is thus not seen as a candidate for such a language. However, richer languages can be built on RDF and RDF(S). Indeed the layered nature of this approach can provide a benefit. Applications that are aware or “understand” RDF(S) can still process, for example, DAML+OIL ontologies. Although the application may not be able to extract and use all the information within the ontology (such as being able to make inferences about complex definitions), basic information about class hierarchies may be available to such an application.

5.4.16 RuleML

Rules in (and for) the Web have become a mainstream topic since inference rules were marked up for Ecommerce and were identified as a Design Issue of the Semantic Web, and since transformation rules were put to practice for document generation from a central XML repository. Rules have also continued to play an important role in Intelligent Agents and AI shells for knowledge-based systems, which need a Web interchange format, too. The Rule Markup Initiative has taken initial steps towards defining a shared Rule Markup Language (RuleML), permitting both forward (bottom-up) and backward (top-down) rules in XML for deduction, rewriting, and further inferential-transformational tasks.

5.4.17 SHOE

The SHOE project, from The University of Maryland's Parallel Understanding Systems Group (PLUS), provided an extension to HTML which allowed the incorporation of machine-readable semantic knowledge in HTML or other World-Wide Web documents. In SHOE, ontologies were fairly simple structures, with an ISA hierarchy of classes/categories, plus a set of atomic relations between these categories, and a set of inferential rules in the form of simplified horn clauses. Categories inherit relations defined for parent categories. Most web pages with *SHOE* annotations tend to have tags that categorize concepts; therefore there is no need for complex inference rules to perform automatic classification. This approach extends HTML with a set of object-oriented tags to provide structure for knowledge acquisition. It associates meaning with content by committing web pages to existing ontologies. These ontologies permit the discovery of implicit knowledge through the use of taxonomies and inference rules, allowing information providers to encode only the necessary information into their web pages. An ontology tag delimits the machine-readable portion of the ontology. Some other tags complement the definition of ontologies.

5.4.18 Topic Maps

Topic Maps provide a formalization of the notion of a back-of-book index. There is an ISO standard [ISO/IEC 13250] for topic maps, and the claim is that they provide a mechanism for describing knowledge structures and associating them with resources. Topic Maps are based around three notions, that of Topic, Association and Occurrence.

A topic represents any "thing" whatsoever – a person, entity, concept – regardless of whether it exists or has any specific characteristics. The notion of topic types allows the specification of classes of topics – thus topics represent both classes and instances (in the traditional frame-based or description logic senses).

An occurrence of a topic is an information resource that is deemed to be relevant to the topic in some way. Topic associations allow the representation of relationships between topics. Topic maps are strongly connected to notions of indexing. As such, they may prove to be a useful mechanism when employed as indexing structures within the Semantic Web.

However, their lack of built in concept forming operators or constructs with detailed semantics suggests that they may not provide an appropriate formalism for the representation of rich ontologies. TopicMaps.Org is an independent consortium of parties interested in developing the applicability of Topic Maps to the Web. Their work includes the development of version 1.0 of an XML grammar for interchanging Web-based Topic Maps, called XML Topic Maps (XTM) Version 1.0, written by the Topicmaps.Org Authoring Group. All versions of the XTM Specification will be permanently licensed to the public.

5.4.19 UML

The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system. UML fuses the concepts of Booch, OMT and OOSE, resulting in a single, common, and widely usable modeling language for users of these and other methods.

UML focuses on a standard modeling language, not a standard process. Although the UML must be applied in the context of a process, experience has shown that different organizations and problem domains require different processes. (For example, the development process for shrink-wrapped software is an interesting one, but building shrink-wrapped software is vastly different from building hard-real-time avionics systems upon which lives depend.) Therefore, the efforts concentrated first on a common metamodel (which unifies semantics) and second on a common notation (which provides a human rendering of these semantics). The UML authors promote a development process that is use-case driven, architecture centric, and iterative and incremental. The (UML Specification) is provided by the Object Management Group (OMG), an open membership, not-for-profit consortium that produces and maintains computer industry specifications for interoperable enterprise applications. OMG's membership roster, about 800 strong, includes virtually every large company in the computer industry, and hundreds of smaller ones. There are a large number of tools supporting UML and the creation of UML models.

UML includes mechanisms such as Use Case, Sequence and Activity Diagrams that allow the description of behavior as well as structure. In this way, UML provides more than just an ontology representation language. UML class diagrams are more or less in correspondence with ER schemas, having classes, attributes and relations, with is-a links and cardinality constraint. Moreover, a full FOL additional constraint language (OCL) can be used. A drawback of UML, however, is its lack of agreed upon formal semantics. Languages like UML have a great advantage in that they provide graphical representations for ontologies, and have a wealth of tools to support the creation and manipulation of those models. The price to pay; however is that they are usually less expressive than languages such as (DAML+OIL).

5.4.20 XML schema

The Document Type Definition (DTD) of XML provides a mechanism for declaring constraints on the use of markup. Automated processing of XML documents, however, requires more rigorous and comprehensive facilities in this area. Requirements are for constraints on how the component parts of an application fit together, the document structure, attributes, data-typing, and so on. XML Schemas express shared vocabularies and allow machines to carry out rules made by people. They provide a means for defining the structure, content and semantics of XML documents. XML Schema also provides a facility for defining datatypes which has been used by other language specifications (e.g. DAML+OIL).

5.4.21 XOL

An early proposal for an XML based ontology exchange language, designed in response to a study of languages from the BioOntology Core Group was the language XOL. The semantics of XOL were based on OKBC-Lite (a simplified form of OKBC). XOL was an influence on early drafts of OIL, but is no longer being actively developed.

5.4.22 OWL 2

The OWL 2 Web Ontology Language, informally OWL 2, is an ontology language for the Semantic Web with formally defined meaning. OWL 2 ontologies provide classes, properties, individuals, and data values and are stored as Semantic Web documents. OWL 2 ontologies can be used along with information written in RDF, and OWL 2 ontologies themselves are primarily exchanged as RDF documents. Here is an overview of OWL 2 language, showing its main building blocks and how they relate to each other. The ellipse in the center represents the abstract notion of an ontology, which can be thought of either as an abstract structure or as an RDF graph. At the top are various concrete syntaxes that can be used to serialize and exchange ontologies. At the bottom are the two semantic specifications that define the meaning of OWL 2 ontologies.

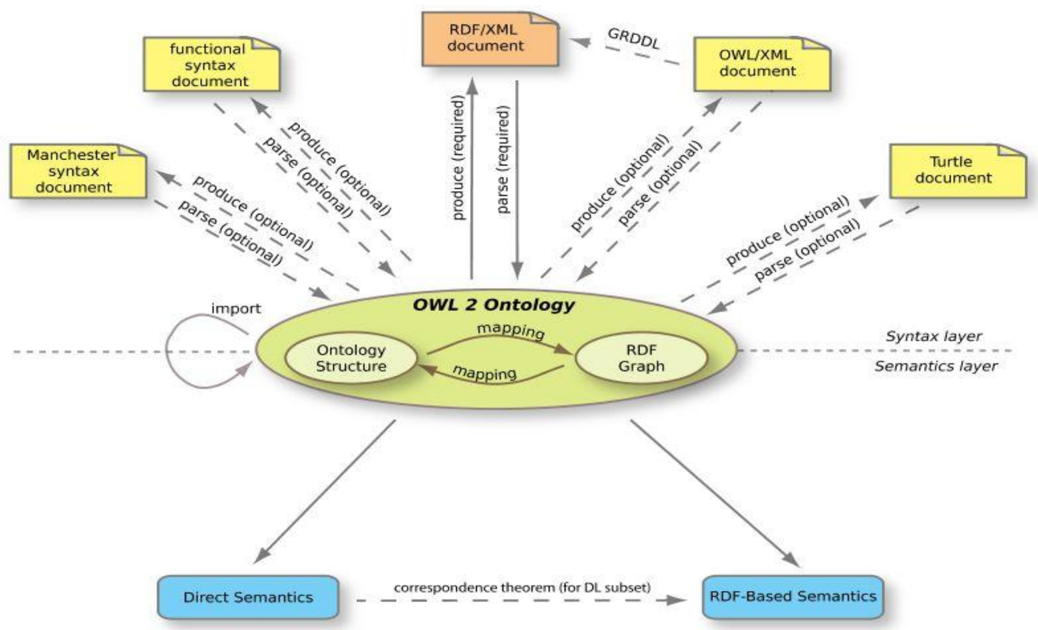


Figure 8: A preview of OWL

Three Sublanguages of OWL

The full set of requirements for an ontology language that seem unobtainable: efficient reasoning support and convenience of expression for a language as powerful as a combination of RDF Schema with a full logic.

Indeed, these requirements have prompted W3C'sWeb OntologyWorking Group to define OWL as three different sublanguages, each geared toward fulfilling different aspects of this full set of requirements.

OWL Full

The entire language is called OWL Full and uses all the OWL language primitives. It also allows the combination of these primitives in arbitrary ways with RDF and RDF Schema. This includes the possibility (also present in RDF) of changing the meaning of the predefined (RDF or OWL) primitives by applying the language primitives to each other. For example, in OWL Full, we could impose a cardinality constraint on the class of all classes, essentially limiting the number of classes that can be described in any ontology.

The advantage of OWL Full is that it is fully upward-compatible with RDF, both syntactically and semantically: any legal RDF document is also a legal OWL Full document, and any valid RDF/RDF Schema conclusion is also a valid OWL Full conclusion. The disadvantage of OWL Full is that the language has become as powerful as to be undecidable, dashing any hope of complete (or efficient) reasoning support.

OWL DL

In order to regain computational efficiency, OWL DL (short for Description Logic) is a sublanguage of OWL Full that restricts how the constructors from OWL and RDF may be used. Essentially application of OWL's constructors to each other is disallowed, thus ensuring that the language corresponds to well-studied description logic. The advantage of this is that it permits efficient reasoning support. The disadvantage is that we lose full compatibility with RDF. An RDF document will in general have to be extended in some ways and restricted in others before it is a legal OWL DL document. Every legal OWL DL document is a legal RDF document.

OWL Lite

An even further restriction limits OWL DL to a subset of the language constructors. For example, OWL Lite excludes enumerated classes, disjointness statements, and arbitrary cardinality. The advantage of this is a language that is both easier to grasp (for users) and easier to implement (for tool builders). The disadvantage is, of course, a restricted expressivity.

Ontology developers adopting OWL should consider which sublanguage best suits their needs. The choice between OWL Lite and OWL DL depends on the extent to which users require the more expressive constructs provided by OWL DL. The choice between OWL DL and OWL Full mainly depends on the extent to which users require the metamodeling facilities of RDF Schema (e.g., defining classes of classes, or attaching properties to classes). When using OWL Full as compared to OWL DL, reasoning support is less predictable because complete OWL Full implementations will be impossible. There are strict notions of upward compatibility between these three sublanguages:

- Every legal OWL Lite ontology is a legal OWL DL ontology.
- Every legal OWL DL ontology is a legal OWL Full ontology.
- Every valid OWL Lite conclusion is a valid OWL DL conclusion.
- Every valid OWL DL conclusion is a valid OWL Full conclusion.

OWL still uses RDF and RDF Schema to a large extent:

- All varieties of OWL use RDF for their syntax.
- Instances are declared as in RDF, using RDF descriptions and typing information.

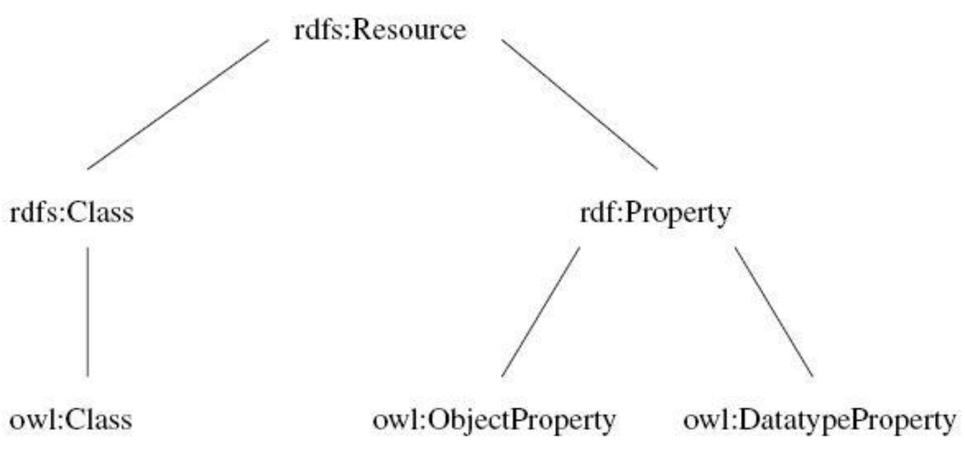


Figure 9: Subclass relationships between OWL and RDF/RDFS

•OWL constructors like owl:Class, and owl:DatatypeProperty, and owl:ObjectProperty are specializations of their RDF counterparts.

The figure above shows the subclass relationships between some modeling primitives of OWL and RDF/RDFS. One of the main motivations behind the layered architecture of the Semantic Web is a hope for downward compatibility with corresponding reuse of software across the various layers. However, the advantage of full downward compatibility for OWL (any OWL-aware processor will also provide correct interpretations of any RDF Schema document) is only achieved for OWL Full, at the cost of computational intractability

Syntaxes

In practice, a concrete syntax is needed in order to store OWL 2 ontologies and to exchange them among tools and applications. The primary exchange syntax for OWL 2 is RDF/XML (RDF Syntax); this is indeed the only syntax that must be supported by all OWL 2 tools.

While RDF/XML provides for interoperability among OWL 2 tools, other concrete syntaxes may also be used. These include alternative RDF serializations, such as Turtle, an XML serialization, and a more "readable" syntax, called the Manchester Syntax, that is used in several ontology editing tools. Finally, the functional-style syntax can also be used for serialization, although its main purpose is specifying the structure of the language.

Name of Syntax	Specification	Status	Purpose
RDF/XML	Mapping to RDF Graphs, RDF/XML	Mandatory	Interchange (can be written and read by all conformant OWL 2 software)
OWL/XML	XML Serialization	Optional	Easier to process using XML tools
Functional Syntax	Structural Specification	Optional	Easier to see the formal structure of ontologies
Manchester Syntax	Manchester Syntax	Optional	Easier to read/write DL Ontologies
Turtle	Mapping to RDF Graphs, Turtle	Optional, Not from OWL-WG	Easier to read/write RDF triples

Table 3

Profiles

OWL 2 Profiles are sub-languages (syntactic subsets) of OWL 2 that offer important advantages in particular application scenarios. Three different profiles are defined: OWL 2 EL, OWL 2 QL, and OWL 2 RL. Each profile is defined as a *syntactic restriction* of the OWL 2 Structural Specification, i.e., as a subset of the structural elements that can be used in a conforming ontology, and each is more restrictive than OWL DL. Each of the profiles trades off different aspects of OWL's expressive power in return for different computational and/or implementational benefits.

OWL 2 EL enables polynomial time algorithms for all the standard reasoning tasks; it is particularly suitable for applications where very large ontologies are needed, and where expressive power can be traded for performance guarantees. OWL 2 QL enables conjunctive queries to be answered in LogSpace (more precisely, AC^0) using standard relational database technology; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to access the data directly via relational queries (e.g., SQL). OWL 2 RL enables the implementation of polynomial time reasoning algorithms using rule-extended database technologies operating directly on RDF triples; it is particularly suitable for applications where relatively lightweight ontologies are used to organize large numbers of individuals and where it is useful or necessary to operate directly on data in the form of RDF triples.

Any OWL 2 EL, QL or RL ontology is, of course, also OWL 2 ontology and can be interpreted using either the Direct or RDF-Based Semantics. When using OWL 2 RL, a rule-based implementation can operate directly on RDF triples and so can be applied to an arbitrary RDF graph, i.e., to any OWL 2 ontology. In this case, reasoning will always be *sound* (that is, only correct answers to queries will be computed), but it may not be *complete* (that is, it is not guaranteed that all correct answers to queries will be computed). Theorem PR1 of the Profiles document states, however, that (in general) when the ontology is consistent with the structural definition of OWL 2 RL, a suitable rule-based implementation performing ground atomic queries will be both sound and complete

5.4.23 FOAF

FOAF (an acronym of Friend of a friend) is a machine-readable ontology describing persons, their activities and their relations to other people and objects. Anyone can use FOAF to describe him or herself. FOAF allows groups of people to describe social networks without the need for a centralized database. The most important component of a FOAF document is the FOAF vocabulary The FOAF vocabulary defines both classes (e.g. foaf:Agent, foaf:Person, and foaf:Document) and properties (e.g. foaf:name, foaf:knows, foaf:interests, and foaf:mbox) grounded in RDF semantics. In contrast to a fixed standard, the FOAF vocabulary is managed in an open source manner, i.e. it is not stable and is open for extension. Therefore, inconsistent FOAF vocabulary usage is expected across different FOAF documents.
FOAF is a descriptive vocabulary expressed using the Resource Description Framework (RDF) and the Web Ontology Language (OWL). Computers may use

these FOAF profiles to find, for example, all people living in Europe, or to list all people both you and a friend of yours know. This is accomplished by defining relationships between people. Each profile has a unique identifier (such as the person's e-mail addresses, a Jabber ID, or a URI of the homepage or weblog of the person), which is used when defining these relationships.

The FOAF project, which defines and extends the vocabulary of a FOAF profile, was started in 2000 by Libby Miller and Dan Brickley. It can be considered the first Social Semantic Web application, in that it combines RDF technology with 'Social Web' concerns.

The following FOAF profile states that Jimmy Wales is the name of the person described here. His e-mail address, homepage and depiction are resources, which means that each of them can be described using RDF as well. He has media as an interest, and knows Angela Beesley (which is the name of a 'Person' resource):

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

<#JW>
    a foaf:Person ;
    foaf:name "Jimmy Wales" ;
    foaf:mbox <mailto:jwales@bomis.com> ;
    foaf:homepage <http://www.jimmywales.com/> ;
    foaf:nick "Jimbo" ;
    foaf:depiction
<http://www.jimmywales.com/aus_img_small.jpg> ;
    foaf:interest <http://www.media.org> ;
    foaf:knows [
        a foaf:Person ;
        foaf:name "Angela Beesley"
                       ] .
<http://www.media.org>
    rdfs:label "info" .
```

5.5 References

[37] T. R. Gruber. *A Translation Approach to Portable Ontologies.* Knowledge Acquisition, 5(2), 1993, pp. 199--220
[38] G. Antoniou, F. van Harmelen. *Semantic Web Primer*. In Cooperative Information Systems. MIT Press Cambridge, April 2004
[39] E. Miller, F. Manola. *RDF Primer*. tech. report, World Wide Web Consortium (W3C) Recommendation 10 February 2004, http://www.w3.org/TR/rdf-primer

[40] F. Yergeau, T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler *Extensible Markup Language (XML) 1.0* tech. report, World Wide Web Consortium (W3C) 4th February 2004, http://www.w3.org/TR/2004/REC-xml-20040204/
[41] G. Schreiber, M. Dean. *OWL Web Ontology Language Reference*. tech. report, World Wide Web Consortium (W3C) Recommendation 10 February 2004, http://www.w3.org/TR/owl-ref/
[42] O. Lassila T. Berners-Lee, J. Hendler. *The Semantic Web*. Scientific American, 184(5), 2001, pp. 34--43.
[43] Orgnet.com Social network analysis software and services for organizations and their consultants http://www.orgnet.com/
[44] http://www.ksl.stanford.edu/software/ontolingua/
[45] http://ontolingua.stanford.edu/doc/release/okbc/
[46] http://protege.stanford.edu
[47] http://www.w3.org/2002/07/owl
[48] http://www.w3.org/RDF/
[49] http://www.ontologos.org/OML/OML%20root.html
[50] http://www.ghg.net/clips/CLIPS.html
[51] http://www.cs.uah.edu/~delugach/CG/
[52] http://users.bestweb.net/~sowa/cg/cgstandw.htm
[53] http://users.bestweb.net/~sowa/cg/cgstand.htm
[54] http://www.ncits.org/tc_home/t2.htm
[55] http://www.cyc.com
[56] http://www.daml.org
[57] http://www.unece.org/cefact
[58] http://www.ebxml.org
[59] http://www.oasis-open.org/
[60] http://www.unece.org/cefact
[61] http://www.fipa.org/
[62] http://www.opengalen.org/open/CRM/
[63] http://logic.stanford.edu/kif/kif.html
[64] http://kmi.open.ac.uk/projects/ocml/
[65] http://www.ontoknowledge.org/oil
[66] http://www.ai.sri.com/~okbc/
[67] A Semantic Web Primer
[68] http://www.w3.org/RDF/
[69] http://www.dfki.uni-kl.de/ruleml
[70] http://www.cs.umd.edu/projects/plus/SHOE/
[71] http://www.topicmaps.org
[72] http://www.omg.org/technology/documents/formal/uml.htm
[73] http://www.w3.org/XML/
[74] http://www.ai.sri.com/pkarp/xol/
[74] http://www.w3.org/TR/owl2-overview/
[75]http://wiki.foaf-project.org/w/DataSources
[76] T.R. Gruber. Towards Principles for the Design of Ontologies used for Knowledge Sharing. In *Proc. of International Workshop on Formal Ontology*, Padova, Italy, 1993. Ed. N. Guarino. Available as Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University.
[77] M. Uschold, M. King, S. Moralee & Y. Zorgios. The Enterprise Ontology. *Knowledge Eng. Rev.*

6. The Development of our Ontology

6.1 Set of principles that are useful in the Development of Ontologies

- Clarity and objectivity: The ontology should provide a glossary of the vocabulary used in providing objective definitions and precise meaning in natural language form.
- Completeness: A definition expressed by a necessary and sufficient condition is preferred over a partial definition.
- Coherence: It should permit inferences that are consistent with the definitions.
- Maximal monotonic extendibility: New general or specialized terms should be included in the ontology in such a way that does not require the revision of the existing definitions.
- Minimal ontological commitment: It should make as few axioms as possible about the world being modeled.
- Ontological Distinction Principle: Classes carrying different identity criteria should be disjoint. This principle is discussed in more detail.

6.2 Ontology Development Process

The ontology development process refers to the tasks you carry out when building ontologies. Adapting the IEEE software development process to ontology development process, the tasks identified are classified into three categories as shown in the figure below.

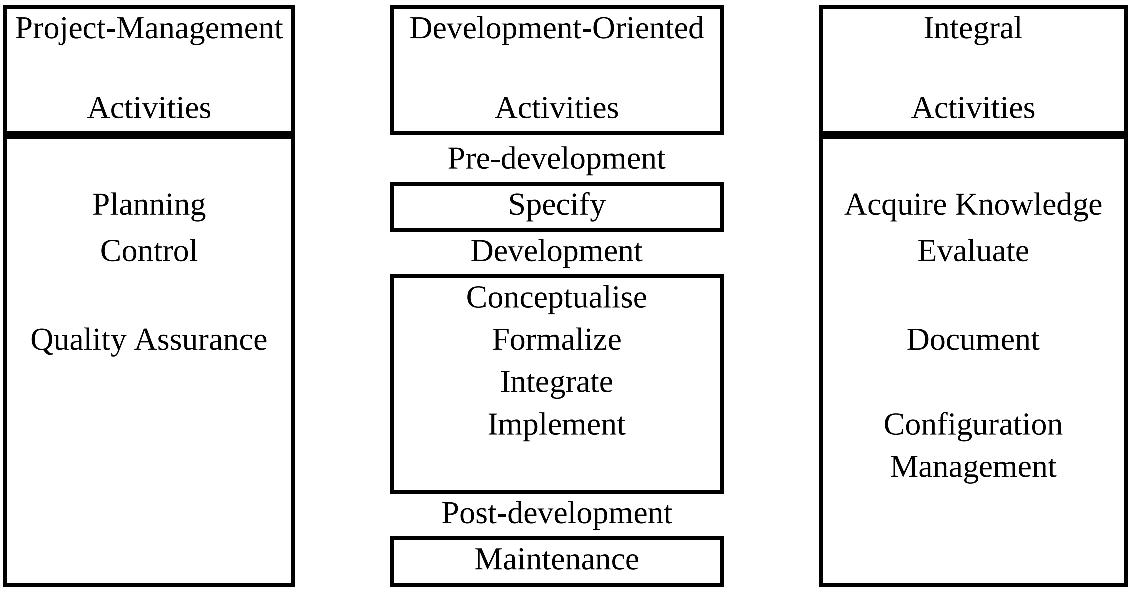


Figure 10: Ontology development process

6.2.1 Project Management Activities

Their main aim is to assure a well-running ontology. These tasks are usual in the classical software development process. They are simply briefly reminded:

- Planning: It is the ordered list of the tasks to be done, represented for example by Gantt diagrams. They also provide information on the resources allocated to the different tasks (i.e. human, budget, software tools, hardware platform).
- Control: Its goal is to guarantee that the planned tasks are done in the way they were intended to be performed. This should prevent typically from delays, errors and omission.
- Quality assurance: It assures that each delivery of tasks is compliant to a given quality standard.

6.2.2 Development Activities

The following tasks describe the practical skills, techniques and methods used to develop ontology:

- Specify: The scope of the ontology under consideration must be defined, its goal, its foreseen usage and end-users' needs. The degree of formality of the writing of this requirement specification may vary, from informal text to more structured framework (e.g. set of competence questions).
- Conceptualize: Its goal is to build a conceptual model that describes the problem and its solution.
- Formalize: This activity transforms the conceptual model into a formal model that is semi-computable. Conceptual graphs, frame-oriented or description logic representations could be used to formalize the ontology.
- Integrate: Ontologies are built to be reused. Accordingly, duplication of work in building ontologies has less sense than in the traditional object-oriented software development. So, reuse of existing ontologies is encouraged. Nevertheless, a general method to integrate ontologically heterogeneous taxonomic knowledge is not known. This specification allows the assertion of some relationships between ontologies.
- Implement: Codification of the ontology in a formal language. For a reference framework for selecting target languages see.
- Maintain: Additions and modifications of ontology should be possible.

6.2.3 Integral Activities

These activities are prominent tasks, since all the development-oriented tasks are fully dependent on the quality achieved during these tasks. The interaction between development-oriented and integral activities will be explicated in the life cycle of the ontology (below).

- Acquire knowledge: Elicitation of knowledge will be done via KBSs knowledge elicitation techniques. As a result, the list of the sources of knowledge and the rough description of the techniques used in the elicitation process will be available.
- Evaluate: Before publishing an ontology, make a technical judgment with respect to a framework of reference.

- Document: To allow reuse and sharing of ontologies, a well written documentation is absolutely needed.
- Configuration management: It is the task of keeping records of each release issued during the development of the ontology. This is a classical task in software development.

6.2.4 Specification

The goal of the specification is to produce either an informal, semi-formal or formal ontology specification document written in natural language. The following information should at least be included:

1. Purpose of the ontology: its intended uses (e.g., teaching, manufacturing, arts, etc.), end-users (e.g., actor and roles) and use case scenarios (e.g., teacher, unit production manager, researcher, etc.). That is the clearly defined domain of application.
2. Degree of formality used to codify the ontology. This ranges from informal natural language to a rigorous formal language.
3. Scope of the ontology: the detailed summary of its content.

The formality of the ontology specification document varies depending on whether a natural language, competency questions or a middle-out approach is used.

For example in a middle-out approach, you can use a glossary of terms to define an initial set of primitive concepts and using these concepts to define new ones. It is also advisable to group concepts in concepts classification trees. The use of these intermediate representations will allow not only the verification, at the earliest stage, of relevant terms missed and their inclusion in the specification document, but also the removal of terms that are synonyms and irrelevant in the ontology. The goal of these checks is to guarantee the conciseness and completeness of the ontology specification document. The middle-out approach, as opposed to the classical bottom-up or top-down approaches, allows identifying some primary concepts of the ontology, in a first stage. Then, it allows specializing or generalizing when needed. As a result, the terms in use are more stable, and so less re-work and overall effort are required.

As mentioned by some authors, and in fact already used in traditional software development at the analysis phase, the use of motivating scenarios (use cases), that present the problem as a story of problems or examples and a set of intuitive solutions, are very useful. Those scenarios could consist of a set of informal competency questions that are the questions that ontology must be able to answer in natural language. Then, the set of informal competency questions are translated into a formal set of competency questions using first-order logic (or higher). This formal set is also used to evaluate the extensions of the ontology.

As an ontology specification document cannot be tested for overall completeness, someone may find new relevant term to be included at anytime and anywhere. A good ontology specification document must have the following properties:

- Conciseness: each and every term is relevant, and there are no duplicated or irrelevant terms.
- Partial completeness: coverage of the terms.
- Realism: meanings of the terms and relationships making sense in the domain.

6.2.5 Knowledge Acquisition

Knowledge acquisition is an independent phase in the ontology development process. However, it is coincident with other phases. Most of the acquisition is done simultaneously with the requirements specifications phase, and decreases as the ontology development process moves forward.

Experts, books, handbooks, figures, tables and even other ontologies are sources of knowledge from which the knowledge can be elicited and acquired, used in conjunction with techniques such as: brainstorming, interviews, questionnaires, formal and informal texts analysis, knowledge acquisition tools, etc ... For example, if you have no clear idea of the purpose of your ontology, the brainstorming technique, informal interviews with experts, and examination of similar ontologies will allow you to elaborate a preliminary glossary with terms that are potentially relevant. To refine the list of terms and their meanings, formal and informal texts analysis techniques on books and handbooks combined with structures and non-structured interviews with experts might help you to build concepts classification trees and to compare them with figures given in books.

6.3 The first steps for the Development of an Ontology for the SN Sites included in SocIoS Platform

To be able to meet the necessary conditions mentioned above (Clarity and objectivity, Completeness, Coherence, Maximal monotonic extendibility, Minimal ontological commitment, Ontological Distinction Principle) we should fix some things that will help us to understand the ontology and the definition of the elements that it has.

Firstly we start with the definition of the SocIoS platform (homepage on http://www.sociosproject.eu/). Social Network (SN) environments are the ideal future service marketplaces. It is well known and documented that SN users are increasing at a tremendous pace. Web 2.0 and SN Sites (SNS) have attracted more than 125 million regular users within just 5 years of existence. The number of the potential customers is huge, coming from almost every societal class, cultural background, and age. The requirements are: a computer, a browser, network access, and the natural need for socializing. The latter results in a great number of SN service consumers who, through their interactions, create complex social graphs that are valuable to them.

Taking advantage of these social dynamics as well as the vast volumes of amateur content generated every second is a major step towards creating a potentially huge market of services. Application developers can be anyone from an individual home user who plays around with Facebook, to a company exploiting the SN spaces to deliver its services. Providing developers with cross-platform tools that enable them to manage the dynamically generated content and complex social interactions by allowing them to build, deploy and potentially sell services that combine data and functionality from two or more different SN services disregarding the underlying SN implementation, will create an agile and profitable market of services and will bring the Internet of Services concept a step closer to realization.

A study launched by the European Commission on user-created content (UCC) revealed the true potential and dynamics of the social graph and of content as well as indicated an opportunity for Europe. Exploiting this opportunity the SocIoS outcomes are expected to be consumed primarily by the SN users, where Europe holds a leading and active role, rather than by the SN sites which are dominated by the US stakeholders. The SocIoS framework will provide an alternative for service provision and consumption in the SN space adding a European perspective and alternative. The main artifact that SocIoS plans to present is a framework for allowing application developers of variable level of expertise to combine content and services from a wide range of SN sites into complex workflows. The approach for services development and deployment is “write once, run everywhere”. The resulting services can be deployed on the developer’s own server or inside the SocIoS platform. There they can be discovered and used by other users of the platform, allowing SocIoS to orthogonally address the debate regarding interoperability through a trusted third party (TTP) or an ad-hoc, distributed solution. The terms of service usage, including billing information, will be defined by Service Level Agreements. The SocIoS framework will make sure to develop applications that will adhere to all the related regulation, be it national, or enforced by the individual platforms. Overall SocIoS aims to support:

- the full exploitation of the main SN assets namely, UCC and social graphs
- the SN user -be it individual or organization- to easily develop, deploy, share and configure applications
- the integration of the developed applications to a service provisioning market
- the business applications
- the extendibility of the SocIoS platform
- the sustainability and interoperability of the SocIoS platform
- the viability of the SocIoS platform through the adherence to the respective regulations and privacy protection

The SocIoS framework consists of 5 main components, namely:

- A SOA infrastructure implementing a service discovery, service composition and workflow enactment mechanism that allows users to discover available services that match their criteria, combine them into workflows and compose new services that implement the resulting workflow. The resulting services can be hosted either on the platform or on the developer’s own web server. Services that can be used to compose workflows will include SNs, services built and deployed inside the platform by other users, services built in the platform but hosted on remote servers as well as third party services hosted on remote servers that are advertised in the platform. For services hosted on remote servers and thus prone to failure due to unavailability a special dynamic discovery mechanism will be developed that detects changes in location or status that may cause unavailability and takes proactive measures in order to keep the registry up to date. One issue here is that currently most SNS provide access to their data and functionalities through APIs and not as services. Special provision will be made to accommodate such SNS so that they are transparently incorporated in the service composition process. More specifically, wrapper services that expose all functionality of the SNs APIs as services will be created and hosted in the platform. Those wrapper services constitute the core SocIoS services and will be referred to as such for the remainder of the proposal.
- An API that will grant developers with single point access to the underlying SN APIs. Using the API, users will be able to programmatically create applications that draw data from or perform actions on a wide range of SNs without having to familiarize themselves with the different concepts, data structures and functions of each SN. Through the API, users will also be able to invoke third party services that are accessible through the SocIoS platform. By using a simple set of functions, the common user will be able to combine the functionality or data from the aforementioned services into a new application tailored to his/her needs. The SocIoS API functions and data structures (object model) will be designed in a way that they are comprehensible to the user by incorporating concepts that are familiar to the users from their involvement in the SNs. The SocIoS API will also be able to

support the adaptive incorporation of changes in the SN APIs or the (semi-)automatic inclusion of new ones.

- A usable user interface that provides access to the aforementioned discovery mechanism and API. Through the interface the users will be able to perform search actions based on specific keywords, select the services that they wish to include into their workflows, specify the environment(s) upon which they want to deploy it, the ways that they want to control it and set its access rights. The interface will give to its end-user the opportunity to find and recruit available SN services from various platforms and combine them in order to build the application as if they were mere functions or objects. This will be done in a transparent way to the user, with usability being the most significant requirement at this phase.
- A toolset for supporting the business extensions of the developed applications. This toolset will affect both the API and API implementation layers but most importantly it will live in the service infrastructure providing to SLA-aware features. The purpose of this toolset will be to allow the incorporation of quality notions in the SocIoS services while investigating all the implications that will occur from that concept. Such are: the inclusion of SLAs associated to the services, the management of the SLA lifecycle, the addition or integration with payment mechanisms, the incorporation of non-functional requirements in the development phases and the investigation of the viability of the various occurring business models.
- A toolset for third-party services support. An enhancement of the SOA infrastructure so as to incorporate in the service management process, services that have been developed outside the SocIoS framework. Examples of such services that would be of special interest for SocIoS are: a) services for the management of the UCC (information, media, etc) that thrives in the SN platforms, and b) services for the encapsulation of existing models/analyses of SNs that will take into account the differentiation that a SN environment must make in order to operate as a service market in contrast to the typical IT service provisioning environments. With this toolset, the final applications and services that will be built using SocIoS will constitute workflows that

combine SocIoS services and third-party web services. Core artifacts of this toolset will be a data and interface specification transformation mechanisms.

Exploiting these components and the underlying technologies SocIoS will develop two integrated_solutions for the news broadcasting and TV production vendors that desire to exploit SN content and dynamics to extend their market share by providing more qualitative and interactive services. The development of these two products will provide guidance to the technical work of the SocIoS project but also will evaluate the technologies.

:

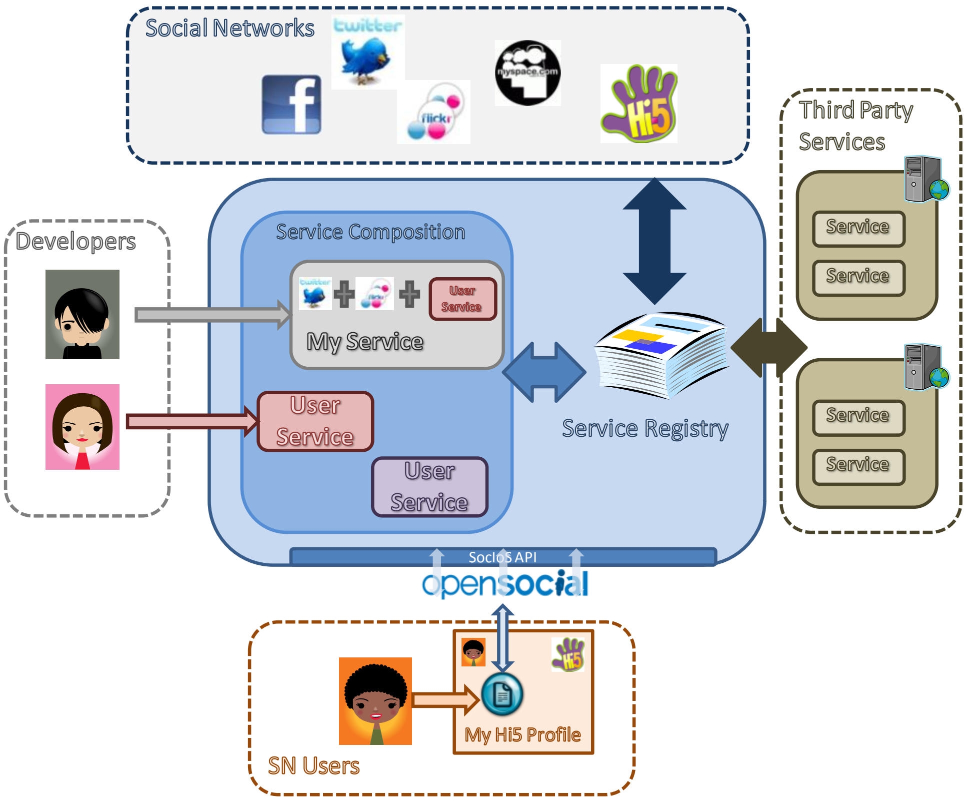


Figure 11: Overall project concept

There will be three types of services available through the platform and accessible through the SocIoS API:

- The SocIoS core services are wrapper services for the SNs with open APIs. The functionalities of the specific SNS (‘containers’ in the language of OpenSocial) will be exposed as services by integrating their APIs and mapping their entities to the SocIoS ontology
- The user created services are services created by independent developers and made available through the SocIoS platform. The developers will be able to develop and deploy services that utilize the content and social graphs living in the SNSs. In most cases, the developers will be self-motivated SN users with experience in programming who wish to make their applications available to a wider audience and potentially charge for their use.
- Third-party services that are hosted on an external host and can be discovered through the platform. The host will need to support the SocIoS SOA infrastructure in order to complete the control of the new workflow lifecycle and to offer SLA support and third-party services integration. A key role in this process is played by the supported Service Registry, operating as a record for SocIoS service (all services exposed in the SocIoS framework) endpoints.

The SocIoS API will enable common users to access all three types of the aforementioned services and – by using a few simple commands – combine them in order to create a new application with enhanced functionality tailored to their needs. For SN platforms compatible with SocIoS (BounceIt and platforms supported by OpenSocial), it will be possible for the user to include the application in their SN profile. Apart from the common users, independent developers are also potential end-users of the API as it will provide them an easy way of accessing the various services available in the platform.

As the picture depicts, different SocIoS artifacts are consumed by different end-user categories because SocIoS proposes the provision of a framework addressing the value chain from end-to-end.

With regard to our work we tried to develop and define the ontology for the objects of MEDIA SocIoS. For that reason had to learn and record fields that characterize the media items for each of these SN sites: Flickr, Twitpic, Facebook, Dailymotion, Youtube and OpenSocial. To do that we found in the pages of the sites at internet, their API documentation that contains the fields and the items which consist everything shown on the site. Each site has its own code (API),but simply finding the code for each one of them describing how the items related to Media (photos, videos) were able to compare and then define the corresponding fields for SocIoS and make the appropriate interface.

Our sources for each SN site were essentially the pages that give free access by anyone who wants to develop an own application in the site or want to learn some things on the operation of these SN sites. More specifically, let's start first with the definition of the API and then look at each individual SN.

An application programming interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. It serves as an interface between different software programs and facilitates their interaction, similar to the way the user interface facilitates interaction between humans and computers.

An API can be created for applications, libraries, operating systems, etc., as a way of defining their "vocabularies" and resources request conventions (e.g. function-calling conventions). It may include specifications for routines, data structures, object classes, and protocols used to communicate between the consumer program and the implementer program of the API.

Turning now to each API for the Social Network sites, the necessary information were found from:

- http://www.flickr.com/services/api for Flickr
- http://developers.facebook.com/ for Facebook
- http://www.dailymotion.com/doc/api/index.html for Dailymotion and to be more specified Rest Api reference and Rest API.

We can also note here the difference between the rest API and the API documentation that characterizes the majority of our sites. Representational State

Transfer (REST) is an architectural style, defined by Dr. Roy Fielding's PhD thesis, Architectural Styles and the Design of Network-based Software Architectures. It is the architectural style used by the World Wide Web. An API is an application programming interface. A REST(ful) API is an API that follows the REST architectural style. So REST is the model of the Web. For example, your browser makes a request to a URL and receives a response. That request can be a GET or POST (or a PUT, DELETE, or HEAD) and the response can be anything -- HTML, an image file, a PDF, XML. The point is that anything, not just a browser, can make that sort of request and process the response. And thus, you have REST.

- http://www.youtube.com/dev for Youtube
- http://twitpic.com/api.do for Twitpic
- http://code.google.com/intl/el-GR/apis/opensocial/docs/0.8/reference for OpenSocial

We must also mention here that in some cases the API documentation was only free to members of the site or u may need an API key which was given to someone who had subscripted.

After downloading and searching the API documentation of each SN site we write down all of the items that describe the media items of each site. So we make the appropriate tables that follow below:

Flickr	Facebook		
photo_id	id		
owner	from		
secret		name	
server		category	
title		id	
ispublic	name		
isfriend	picture		
isfamily	source		
description	height		
license	width		
date_upload	images		
date_taken		height	
owner_name		width	
icon_server		source	
original_format	link		
last_update	icon		
geo	created_time		
tags	position		
machine_tags	updated_time		
o_dims	comments		
views		id	
media		from	
path_alias			name
various_urls			id
url_t			
url_o			
latitude			
longitude			
o_height			
o_width			

Table 4: The features of a media item for Flickr and Facebook

Twitpic		Youtube	
id		Title	
short_id		Uploaded by	
user_id		Date Uploaded	
source		Video ID	
message		Description	
views		Copyright	
width		Aspect Ratio	
height		Categories	
size		Credits	
type		Ratings	
status_id		Web Player URL	
in_reply_to_status_id		Keywords	
in_reply_to_user_id		Average rating	
location		View count	
timestamp		Favorite count	
user	id	Thumbnails	
	twitter_id		Thumbnail URL
	username		Thumbnail Time Index
	name	Media	
	location		Media Location
	website		Media Type
	bio		Duration
	avatar_url		
	User Twitter profile appearance info		
	utc_offset		
	geo_enabled		
	public_updates		
	public_tagging		
	banned		
	timestamp		
	events		
	images		
comment			
	id		

	user_id		
	message		
	timestamp		
	user_info again		
event			
	id		
	user_id		
	name		
	description		
	timestamp		

Table 5: The features of a media item for Twitpic and YouTube

OpenSocial		**DailyMotion**	
album_id		url	
created		id	
description		status	
duration			waiting
file_size			processing
id			ready
language			published
last_updated			rejected
location			deleted
	latitude		encoding_error
	longitude	tags	
mime_type		title	
num_comments		channel	
num_views		description	
num_votes		date	
rating		language	
start_time		country	
tagged_people		city	
tags		published	
thumbnail_url		allow_comments	
title		private	
type		creative	
url		official	
		fields	
Album	id		
	location		
	mediaItemCount		
	mediaMimeType		
	mediaType		
	owernId		

	thumbnailUrl		
	title		

Table 6: The features of a media item for OpenSocial and Dailymotion

Note here that about Dailymotion the selected field may be one of the elements listed below:

- owner_url
- owner_avatar_small_url
- owner_avatar_medium_url
- owner_avatar_large_url
- thumbnail_url
- thumbnail_small_url
- thumbnail_medium_url
- thumbnail_large_url
- rating
- ratings_total
- views_total
- views_last_hour
- views_last_day
- views_last_week
- views_last_month
- comments_total
- bookmarks_total
- embed_html
- embed_url
- aspect_ratio

Then we try to do the matchup between the media items of each SN site and the media items that we have defined in SocIoS. If an item is not included in the SN API then we fill the table with the value “not supported”. The result of this try is shown in the table follows above:

SocIoS API	**Flickr**	**Facebook**	**Twitpic**	**Youtube**	**OpenSocial**	**Dailymotion**
item_id	photo_id	id	id	Video ID	id	id
thumbnail_url	url_t	picture	*not supported*	Thumbnail URL	thumbnail_url	thumbnail_url
url	url_o	source	*not supported*	Web Player URL	url	url
type	media	*not supported*	*not supported*	"video"	type	"video"
owner_id	owner	id	id	*not supported*	owernId	owner
license	license	*not supported*	*not supported*	Copyright	*not supported*	*not supported*
date_uploaded	date_upload	updated_time	timestamp	Date Uploaded	last_updated	created_time
date_created	date_taken	created_time	not supported	not supported	created	*not supported*
lat	latitude	not supported	location	not supported	latitude	*not supported*
lon	longitude	not supported	not supported	not supported	longitude	*not supported*
description	description	not supported	message	Description	description	description
title	title	name	*not supported*	Title	title	title
tags	tags	category	*not supported*	Keywords	tags	tags
views	views	*not supported*	views	View count	num_views	views_total
height	o_height	height	height	*not supported*	*not supported*	*not supported*
width	o_width	width	width	*not supported*	*not supported*	*not supported*
rating	*not supported*	*not supported*	*not supported*	Average rating	rating	rating
sns_id*	1	2	3	4	5	6
category	*not supported*	*not supported*	*not supported*	Categories	*not supported*	channel
owner_name	owner_name	name/from	name	Uploaded by	*not supported*	owner_screenname

Table 7: The matchup between a media item of SocIoS and a media item of a SN site

That was the basic and the fundamental step to create our ontology. Having this table we can proceed to the development of our ontology. The definitions and the relations needed to create out ontology are mentioned above and as we can see in some cases some fields are empty (not supported) at the SN. But this may not be a problem because when u want to describe and take a media item from the SocIoS network the user can give each field that exists according the desirable SN site (Facebook, Twitpic, Flickr, Youtube, Dailymotion, open social)

It is worth also to mention here that Open Social API is basically an initiation from Google that support the API of many different sites. The creation and the development of our ontology will be described in more details in next chapter. Before that we can take a look at a site-web application we have developed about Dailymotion API that responds to our demands about media items and information about them

6.4 A Web Application for Dailymotion API

As we mentioned above Dailymotion describes its rules, relations and items in API documentation which is free for everyone. After studying the API and writing down all the necessary fields for the description of a media item we made a Web application using Php programming language and Html.

The application's purpose is to open a page where a user can give a general keyword or the owner name of a video and returns a json file that gives all the details of the video. For example we may want to take a video that is related to the word "mountain". Filling the box that demands the keyword we can take the result in json form. Most of the sites can return the values for the fields that describe a video into a json form or xml form or both of them if supported. But here in Dailymotion only json form is supported.
Let's see now what is a json form and we can read it and understand it.

6.4.1 JSON files

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by: (colon) and the name/value pairs are separated by, (comma).

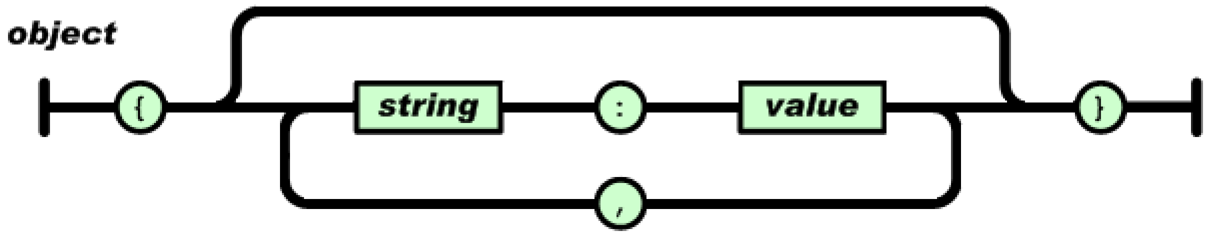


Figure 12: An object in a Json file

An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by , (comma).

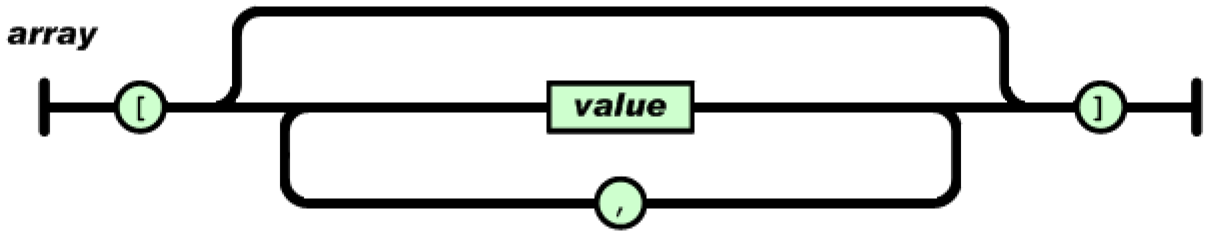


Figure 13: An array in a json file

A value can be a string in double quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.

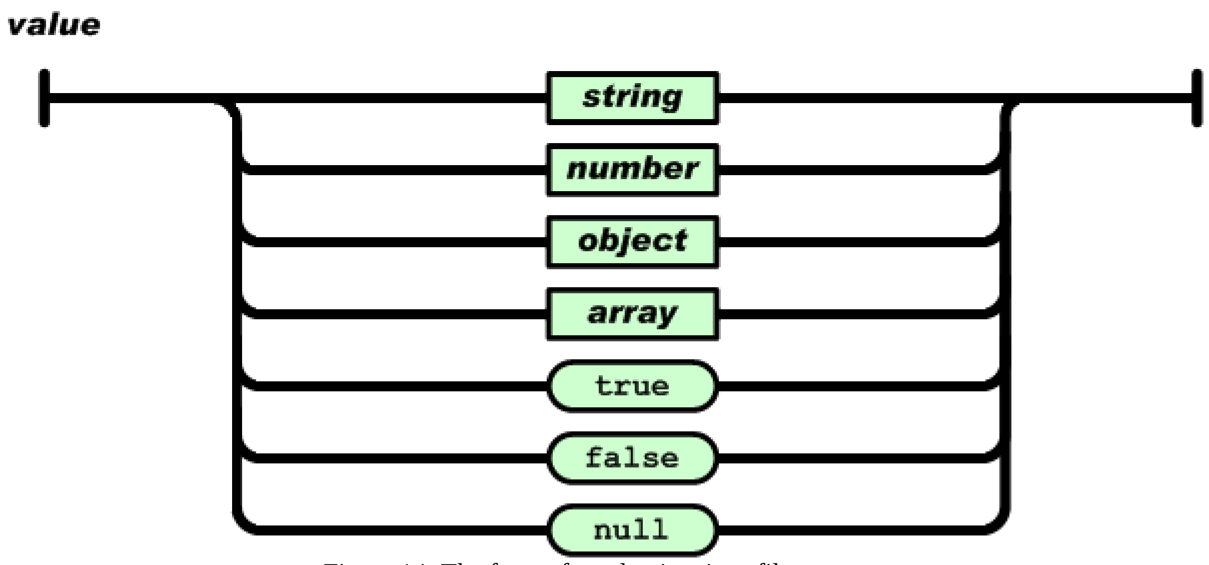


Figure 14: The form of a value in a json file

A string is a sequence of zero or more Unicode characters, wrapped in double quotes, using backslash escapes. A character is represented as a single character string. A string is very much like a C or Java string.

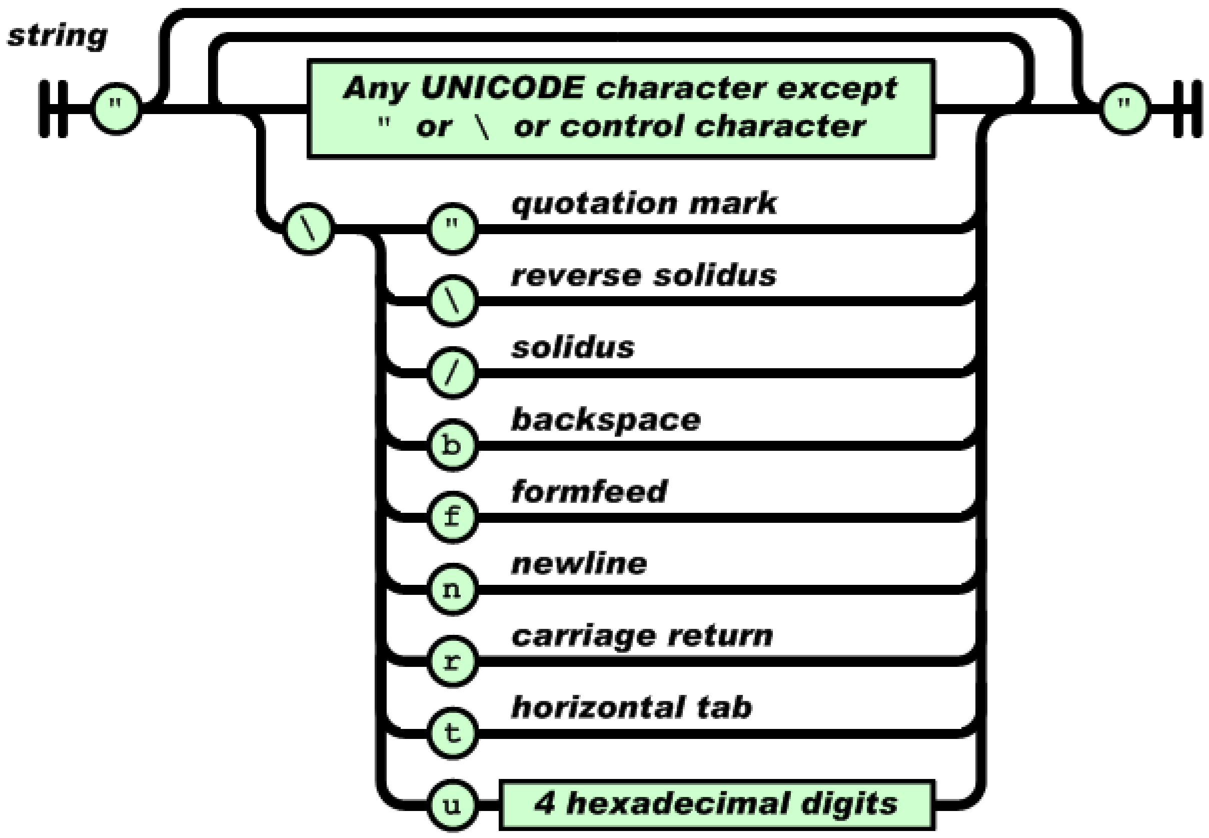


Figure 15: The form of a string n a json file

A number is very much like a C or Java number, except that the octal and hexadecimal formats are not used.

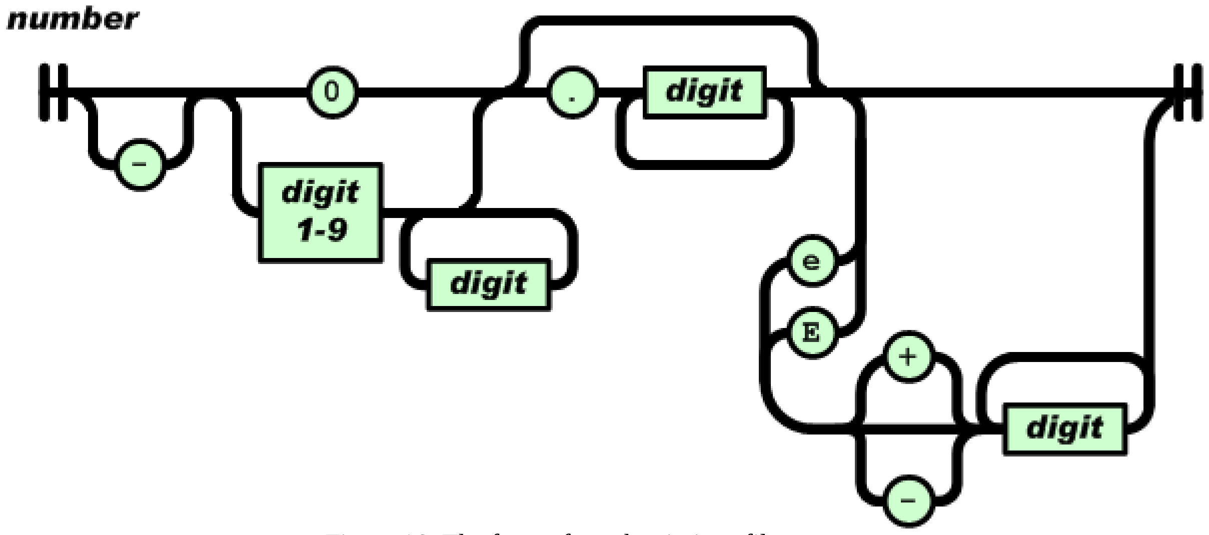


Figure 16: The form of number in json files

To be more clarifying let’s see an example of json file:

{"page":1,"limit":10,"has_more":true,"list":[{"id":"xi621j","thumbnail_url":
\"http:\/\/static2.dmcdn.net\/static\/video\/176\/515\/30515671:jpeg_preview
_large.jpg?20110413162617","url":"http:\/\/www.dailymotion.com\/video\/xi621j
_how-to-plan-your-vacation-to-st-ucia_lifestyle","owner":"shellyysen",
"created_time":1302704033,"description":"http:\/\/bookingsstlucia.com\/ -
Vacations are always special. You can plan a vacation every year so that you can
take break from your regular routine. It is very important to plan your vacation
properly. Without a proper planning you cannot enjoy your holiday. There are
various places in the world where you can plan your vacations. First of all you
need to decide about the kind of place you would like to visit. Some people are
fond of beaches while others want to be in between the mountains.","title":"How To
Plan Your Vacation To StLucia","tags":["luxury","vacations","island","best","
caribbean","lucia","hotels","saint"],"views_total":0, "rating":0, "channel":
"lifestyle","owner_screenname":"shellyysen"}

This is a json file result that we have writing down the general keyword mountain. That says that we are in page=1 the limit of our results is 10 and it has more results. The id of the video is “xi621j” and the thubanil_url is:

http://static2.dmcdn.net/static/video/176/515/30515671:jpeg_preview_large.jpg?20110413162617.

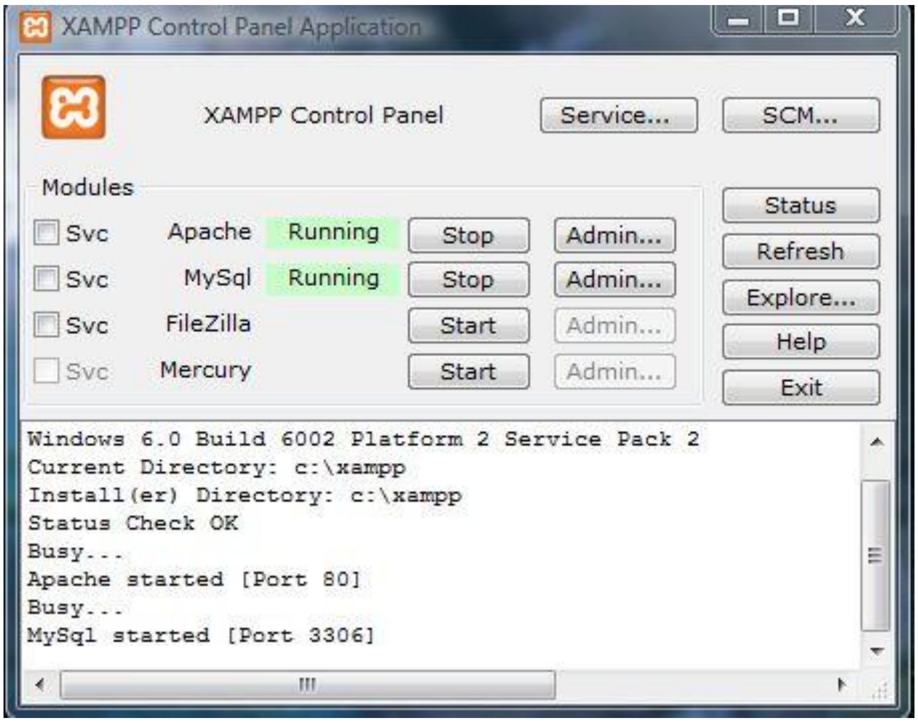
In addition it give us the url which is: http://www.dailymotion.com/video/xi621j_how-to-plan-your-vacation-to-st-lucia_lifestyle, the owner of the video: shellyysen the created_time:1302704033 and a description of the video as we can see it in Dailymotion page "http:\/\/bookingsstlucia.com\/ - Vacations are always special. You can plan a vacation every year so that you can take break from your regular routine. It is very important to plan your vacation properly. Without a proper planning you cannot enjoy your holiday. There are various places in the world where you can plan your vacations. First of all you need to decide about the kind of place you would like to visit. Some people are fond of beaches while others want to be in between the mountains". More over we can see the title, the tags, the total views, the rating, the channel and the owner screename.

As we can see the json file is written in an informal way that is very close to the natural language so it is easy to be read and understood from users and machines both.

6.4.2 The development of the Web Application

To make now the web application we follow some steps. Firstly we downloaded and used the XAMPP apache web server. XAMPP is an open, cross-platform web server, contains primarily the Apache HTTP Server, MySQL database, and interpreters for scripts written in the PHP and Perl programming languages. After the download we run the xamp as shown in the picture below:

We start the apache and the MySQL server and we were ready to write our program in php and then run the site in local host to see if it is ok.


XAMPP Control Panel Application
XAMPP Control Panel
Service...
SCM...
Modules
Svc Apache Running Stop Admin...
Svc MySql Running Stop Admin...
Svc FileZilla Start Admin...
Svc Mercury Start Admin...
Status
Refresh
Explore...
Help
Exit

```
Windows 6.0 Build 6002 Platform 2 Service Pack 2
Current Directory: c:\xampp
Install(er) Directory: c:\xampp
Status Check OK
Busy...
Apache started [Port 80]
Busy...
MySql started [Port 3306]
```

Figure 17: The start of Xampp

Then we open a notepad editor and we write the code followed. The code is written in php and html.

```
 <?php
 if(isset($_POST['formSubmit']) == "Submit")
 {
       //Error messages
 $errorMessage = "";
 if(empty($_POST['formKeyword']))
 {
 $errorMessage .= "<li class='error'>Παρακαλώ εισάγετε ένακeyword.</li>";
 }
 $varKeyword = $_POST['formKeyword'];
 if(empty($errorMessage))
 {
 $url='https://api.dailymotion.com/videos?search='.$varKeyword.'&fields=id,
 thumbnail_url,url,owner,created_time,description,title,tags,views_total,rating,
 channel,owner_screenname'; // this can be set based on whatever

 header('Location: '.$url);          //για να πας URL
 }
 }

 if(isset($_POST['formSubmit1']) == "Submit")
 {
       //Error messages
       $errorMessage = "";
 if(empty($_POST['formOwner']))
 {
 $errorMessage.= "<li class='error'>Παρακαλώ εισάγετε έναν videoowner.</li>";
 }
 $varOwner = $_POST['formOwner'];
 if(empty($errorMessage))
 {
 $url1='https://api.dailymotion.com/videos?user='.$varOwner.'&fields=id,
 thumbnail_url,url,owner,created_time,description,title,tags,views_total,
 rating,channel,owner_screenname'; // this can be set based on whatever

 header('Location: '.$url1);
 }
 }
 ?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <link rel="stylesheet" type="text/css" href="css/style.css" />
    <title>Φόρμα Αναζήτησης</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
     <table class="outter" width="100%" align="center">
         <tr>
             <td align="center"><img src="images/ntua.png" alt="" border="0"/>
             </td>
        </tr>
      </table>
      <table class="main" cellpadding="8" align="center">
      <tr>
      <td class="mainIn" >
      <?php
             if(!empty($errorMessage))
             {
              echo("<p>There    was    an    error    with    your    form:</p>\n"
                          echo("<ul>" . $errorMessage . "</ul>\n");
             }
       ?>
       <form action="form.php" method="post">
       <div class="blockHeader">
       <h3>
           Φόρμα Αναζήτησης για την DailyMotion
       </h3>
       </div>
       <table >
       <tr>Αναζήτηση με βάση:</tr>
          <tr>
           <td>Γενικό keyword:</td>
           <td><input type="text" name="formKeyword" value="" size="50" /></td>
           <td><input type="submit" name="formSubmit" value="Submit" /> </td>
          </tr>
           <tr>
            <td>Video owner:</td>
            <td><input type="text" name="formOwner" value="" size="50"/></td>
            <td><input type="submit" name="formSubmit1" value="Submit" /> </td>
          </tr>
        </table>
        </form>
        </td>
        </tr>
       </table>
</body>
```

Finishing with this file we save it as form.php and we saved it in htdocs at Xampp installation directory file. Then we made two more files at the same folder (C:\xampp\htdocs\site). Firstly a folder called images to save the background and the image we use in our web application and a folder called css which includes a file called style.css. This file helps us improve and making the background and the style-form our web application-page better. The code written in these pages follows below:

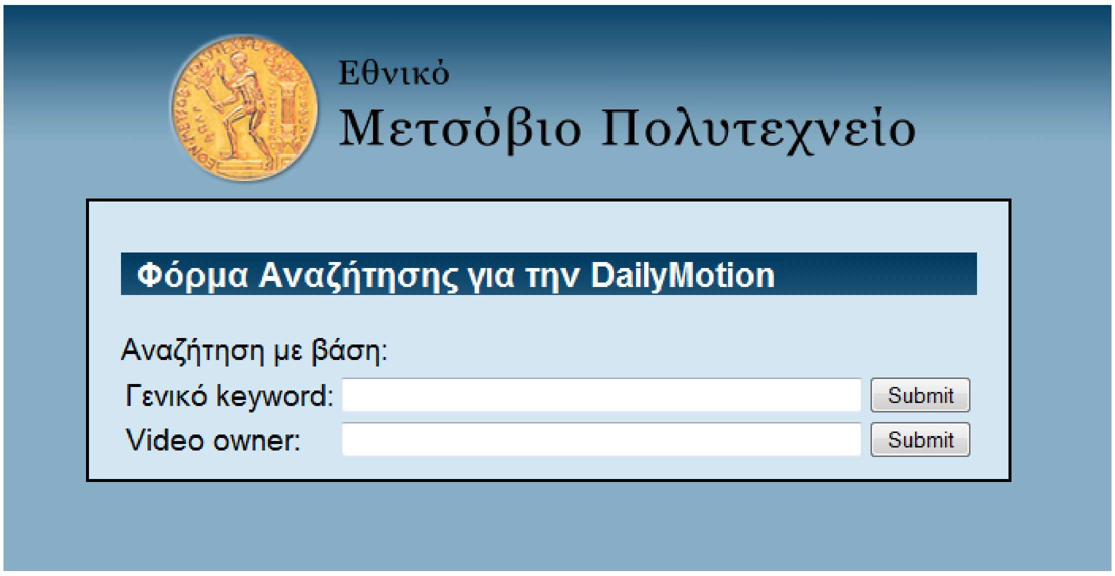
```
body{
      background-color: #87AEC5;

      font-family: arial;
      font-size: 120%;
      margin: 0px;
      padding: 20px 0 0 0px;
      background-image: url(../images/BannerTileBackground.gif);
      background-repeat: repeat-x;
}     // we put here the background image and we cross it over the
      X-direction to fulfill the page //and we also define the font
      size and type of the letters shown in the main body of the
      page
.blockHeader
{
      background-image: url(../images/BannerTileBackground.gif);
      background-repeat: repeat-x;
      color:white;
      padding-left: 10px;
}                          //the same for the header of the page
table.main
{
      background-color:#D2E7F4;
      font-size: 100%;
      color:black;
      border:2px solid black;
      padding: 0 10px 0 10px;
    //we can modify the font size and the style of the search table
}
p
{
  color: #C02020;     // the color for the first error message
}

ul                   // the color for the second error message
{
  color: #C02020;
}
```

Finishing with the creation and saving of these files we could run our web application. We open a web browser (internet explorer, Mozilla Firefox, Google chrome etc) and we type http://localhost/site/form.php

The result is shown below:


Εθνικό
Μετσόβιο Πολυτεχνείο

Φόρμα Αναζήτησης για την DailyMotion

Αναζήτηση με βάση:

Γενικό keyword: Submit

Video owner: Submit

Figure 18: the search box of the web application

Filling now the boxes we can take the result we wish. This can be either a json file with the information of a video been searched by a general Keyword either a json file with the information of a video been searched by the video owner.

Let’s see now how this result comes up. In the php code we write this line:

$url='https://api.dailymotion.com/videos?search='.$varKeyword.'&fields=id,thumbnail_url,url,owner,created_time,description,title,tags,views_total,rating,channel,owner_screenname';

Which says that the $varKeyword which is the variable with the keyword we inserted is filling the URL we want to direct and an http GET request will be performed. The URL: 'https://api.dailymotion.com/videos is used in the Api of Dailymotion to get a list of videos according to a set of parameters. These parameters can be one of the listed below:

• Fields (array) : id, title, tags, channel, description, url, tiny_url,created_time, modified_time, type, private, explicit, published, duration,owner, owner_screenname, owner_url,owner_avatar_small_url,owner_avatar_medium_url,owner_avatar_large_url,thum bnail_url, thumbnail_small_url,thumbnail_medium_url, thumbnail_large_url, rating,ratings_total,views_total,views_last_hour,views_last_day,views_last_week,views_last _month,comments_total,bookmarks_total,embed_html,embed_url,aspect_ratio.

• page (number) - The page number to show. (Min 1, Max 100, Default 1)

• limit (number) - The number of items to show per page. (Min 1, Max 100,
Default 10)

• sort (string) - How the results should be sorted or ordered. (Default recent)
Allowed Values: recent, visited, visited-hour, visited-today, visited-week, visitedmonth, commented, commented-hour, commented-today, commented-week, -month, rated, rated-hour, rated-today, rated-week, rated-month, discussed, discussed-hour, discussed-today, discussed-week, discussed-month,
relevance, random and

• filters (array) - A list of filters available to reduce the result set. Allowed Values:featured, hd, official, creative, creative-official, buzz, buzz-premium,

• localization (string) - The localization of the content. Use detect to use the best localization for the API client localization or a country code returned by thelocale.list method.

• search (string) - A search query.

• user (string) - Limits the results to videos uploaded by this user.

• group (string) - Limits the results to videos of this group.

• playlist (string) - Limits the results to videos from this playlist.

• tag (string) - Limits the results to videos with this tag.

• channel (string) - The channel, i.e. main category, of the video.

So by typing:

https://api.dailymotion.com/videos?search=$varKeyword&fields=id,thumbnail_url,url,owner,created_time,description,title,tags,views_total,rating,channel,owner_screenname'

We take the result from the search we did according the $varKeyword which is the general keyword we give in the box. It is correct to assume that in the fields listed above we can add more of the permitted listed fields we have in the top. For example we may want to take the aspect_ratio but not the title of the video so we change the http Get response and we write:

https://api.dailymotion.com/videos?search=$varKeyword&fields=id,thumbnail_url,url,owner,created_time,description,<u>aspect_ratio</u>,tags,views_total,rating,channel,owner_screenname'

(the difference is underlined as we can see)

The same situation is occurring when we search according the owner name. The only difference we have is that instead of $varKeyword we use another variable $varOwner which takes the value of the owner name we entered in the box of the search page.

It is helpful to mention here the meaning of each value that are permitted in the fields list (array)

• id (string) – The id of the video.
• title (string) – The title of the video.
• tags (array) – The list of tags for the video.
• channel (string) – The short channel name of the video.
• description (string) – The description of the video.
• url (string) – The URL of the video.
• tiny_url (string) – The tiny URL of the video.
• created_time (date) – The date the video was uploaded to the site.
• modified_time (date) – The date the video was last modified.
• type (string) – The content type of the video (can be official, creative or null).
Allowed Values: ugc, creative and official
• private (boolean) – True if the video is private.
• explicit (boolean) – True if the video is explicit.
• published (boolean) – True if the video is published.
• duration (number) – The duration of the video in seconds.
• owner (string) – The id or username of the owner of the video (prefer owner_screenname to show the user name).
• owner_screenname (string) – The username or fullname of the owner of the video, depending on user preference.
• owner_url (string) – The URL of the owner of the video.
• owner_avatar_small_url (string) – The URL of the avatar of the owner (40px by 40px).
• owner_avatar_medium_url (string) – The URL of the avatar of the owner (80px by 80px).
• owner_avatar_large_url (string) – The URL of the avatar of the owner (160px by 160px).
• thumbnail_url (string) – The URL of the video thumbnail (full size respecting ratio).
• thumbnail_small_url (string) – The URL of the video thumbnail (80px by 60px).
• thumbnail_medium_url (string) – The URL of the video thumbnail (160px by 120px).

• thumbnail_large_url (string) – The URL of the video thumbnail (320px by 240px).
• rating (number) – The average number of stars the video has (float between 0 and 5).
• ratings_total (number) – The number of users who voted for the video.

• views_total (number) – The number of views on the video since its publication.

• views_last_hour (number) – The number of views in the last sliding hour.

• views_last_day (number) – The number of views in the last 24 sliding hours.
• views_last_week (number) – The number of views in the last 7 sliding days.
• views_last_month (number) – The number of views in the last 30 sliding days.
• comments_total (number) – The total number of comments on the video.
• bookmarks_total (number) – The total number of times a video has been added to users' favorites.
• embed_html (string) – The HTML embed code.
• embed_url (string) – The URL to embed the video.
• aspect_ratio (number) – The aspect ratio of the video frame (i.e.: 1.33333 for 4/3, 1.77777 for 16/9...).

6.5 The Development of our Ontology

As we mentioned before the scope of our thesis was to develop an ontology for the media items of the SocIoS platform. To achieve this we used the Protégé tool 4.1 version and for the reasoning of our ontology we plug in and we use the Pellet 2.2.

The required fields of the media item we wanted to describe were found in the table 7 in the first column. These are listed below with some comments to make them clear:

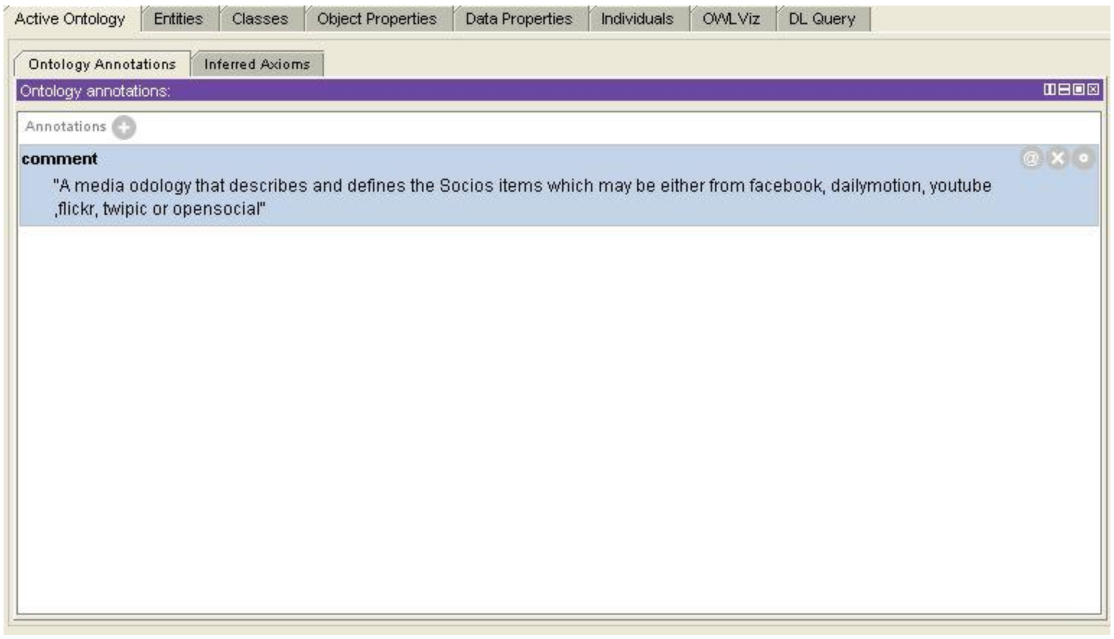
- item_id : the id of the media item
- thumbnail_url: The thubnail_url of the media item
- url: the url of the media item
- type: the type of the media item (video, photo)
- owner_id: the id of the owner who uploaded the specific media item
- license: the license that the media may have
- date_uploaded: the date that the media was uploaded in the site

- date_created: the date that the media has been created, for example when a photo has been taken
- lat: the latitude of the media item we describe
- lon: the longitude of the media item we describe
- description: the description of the media item as we can see it in the site
- title: the title that the media item has, for example the title of a video
- tags: the tags that the media item has, these may be some keywords that are used for the search of a video or a photo
- views: the total views that a media item has in the site
- height: the height of the media item in the page of the web browser
- width: the width of the media item in the page of the web browser
- rating: the rating that a media item may have in the site
- sns_id: using this id we can differentiate the media items from the different SN sites. So if the sns_id=1 the media item is coming from Flickr , if it is 2 it's coming from Facebook, 3 is for Twitpic, 4 is for Youtube, 5 is for OpenSocial and 6 for Dailymotion. This id is very useful to understand which field is required each time. As we see in the table each field of the SocIoS media item may have different defined name in each SN site or it may not be supported. So we use this id to understand which field we must use according to which SN site media item we have. For example if the sns_id is 2 and we have to fulfill the tags field the appropriate name of the SN site field that we expect is category because as we see in the table 2 is for Facebook site and the matchup between tags field of SocIoS and Facebook field is category.
- category: the category of the media item, for example a media item may refer to a category for fun sport or music.
- owner_name: the name of the uploader of the media item.

The basic try we had to accomplish in our effort of the ontology development was to do the matchup between the fields of a media item of Social Network site (SN site) and the fields of the SocIoS. To do this the table 7 was very helpful and necessary. This happen because according to this table we saw each field name in the SN site and if it supported. Some sites don't support some fields in their API so in the definition of our ontology we have to exclude in some fields if they are not be supported. For example if we want to describe the category field for the Facebook we see that this feature is not supported in the Facebook so we must not include it in our definition of a Facebook media item. Another subject we have to take into our consideration was the type of each field. According again the same table we can observe that some fields are of type integer or text or datetime or double. So it was necessary to our definition to mention the appropriate the value type that a field may have.

All these matters we described, we managed to accomplish them using the protégé tool. Let’s see now the steps we take to manage them and to design our ontology.

Firstly we run the protégé and in the first tab calling *Active Ontology* we add in the *annotations tab* a comment about what our ontology describe.



comment
"A media odology that describes and defines the Socios items which may be either from facebook, dailymotion, youtube ,flickr, twipic or opensocial"

Figure 19: Ontology annotations

Then selecting the tab classes we started creating our classes. *Class* is the general term to describe everything we want to define in our ontology starting from a field for example or the whole media item of a SN site or the SocIoS platform.

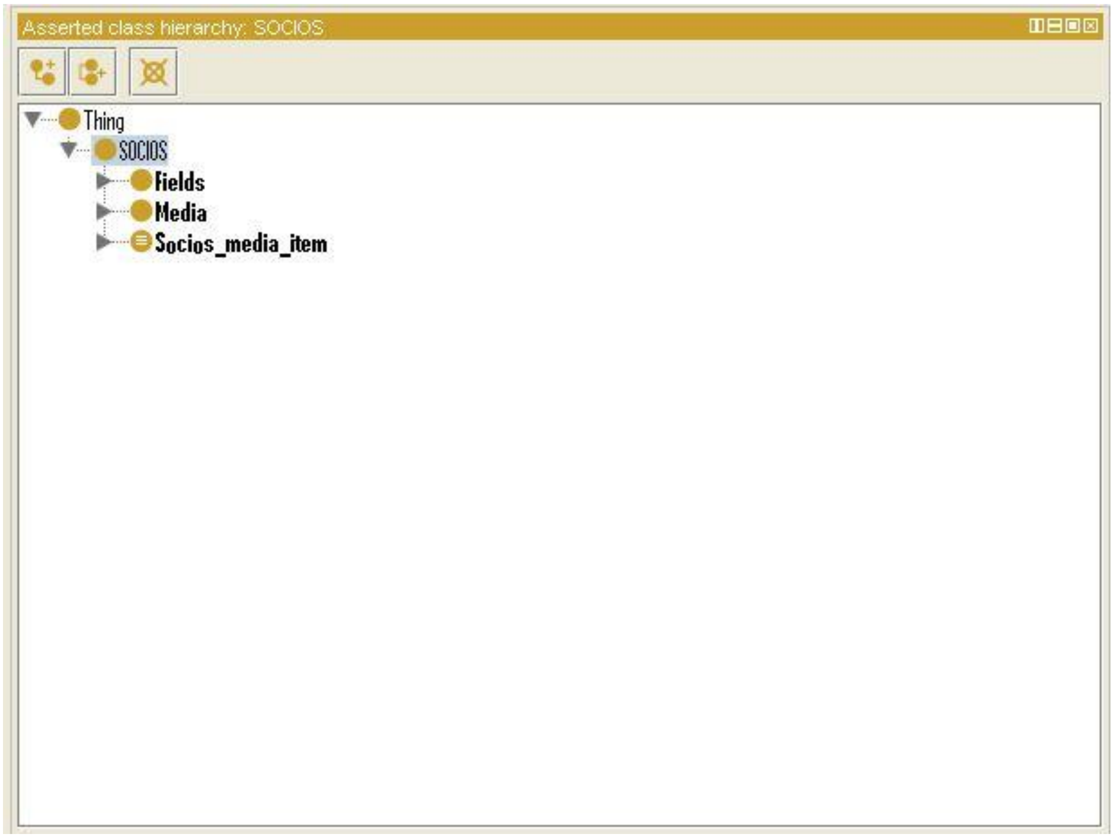


Figure 20: Create a class

First of all we create a *subclass* of the *Thing* (which is the general superclass of Protégé) called SOCIOS. To do this we select the *add subclass* box. Then we make 3 subclasses of the SOCIOS class called Fields, Media and SocIoS_media_item.

Then in each of these 3 classes we made other subclasses like before. To define what we want. In the “Fields” class, we create the classes listed below with the appropriate comments to understand what they describe. The “Fields” class includes all the features that a media item has as we find them in each SN site.

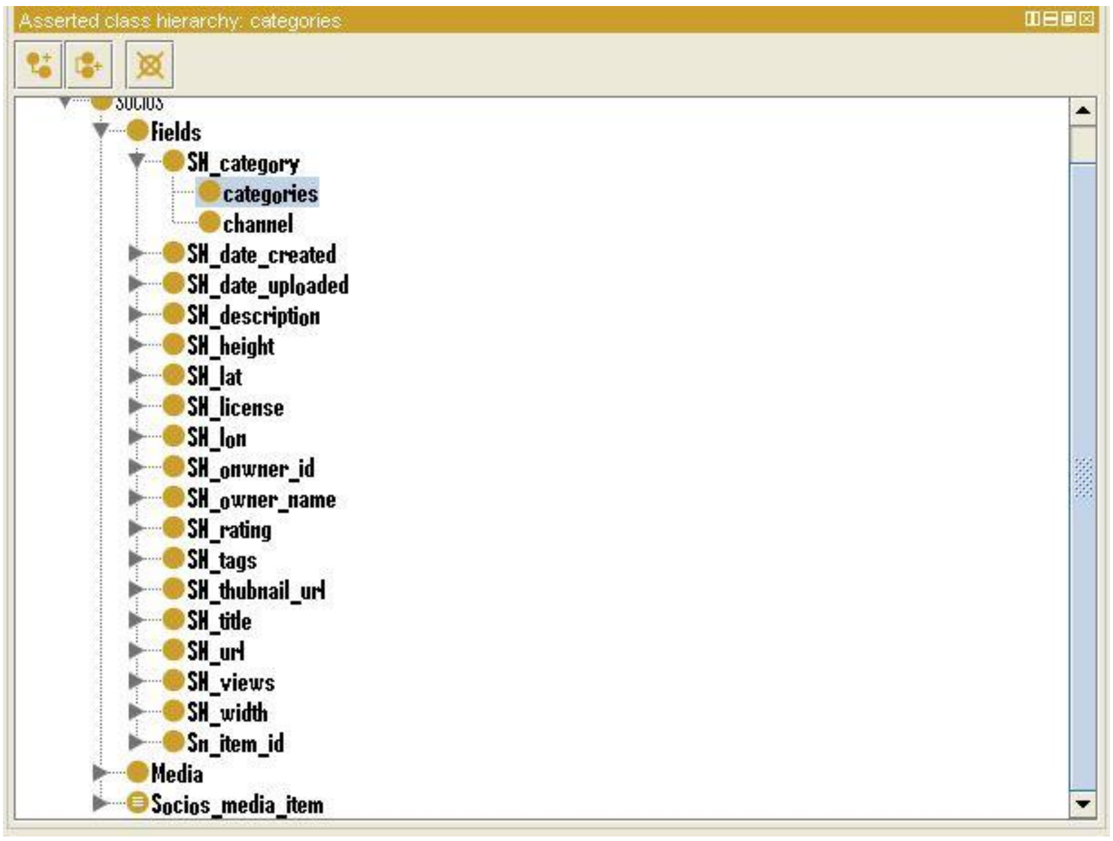
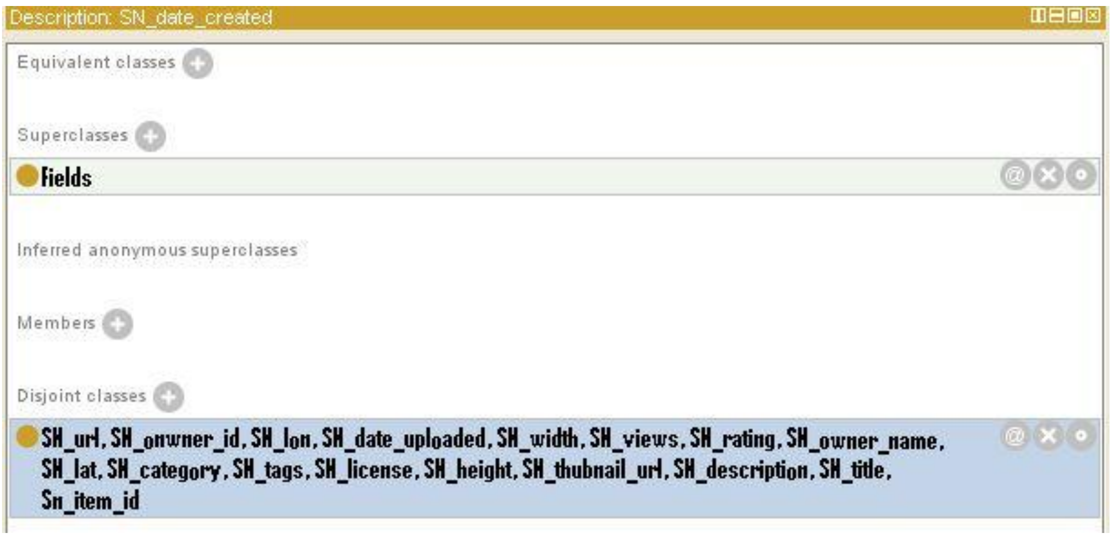


Figure 21: create a subclass

So each of these classes describe the fields that the SN sites have. For example the SN_category has 2 subclasses: categories and channel. This happen because according to the table 7 only youtube and dailymotion support this field and the equivalent fields are categories and channel. Similarly Sn_item_id has the subclasses photo_id, video_id and id according to the table 7. After finishing with the other subclasses of the "Fields" class we had the fundamental classes to rule each media item of a SN site. Then we had to disjoint these subclasses because an individual (object) can't be an instance of more than one of these subclasses of the "Fields". For example an individual can't be a SN_category and a SN_lat at the same time.

To do that in the description tab of a subclass we add the *disjoint classes* options the other subclasses.


Description: SN_date_created

Equivalent classes

Superclasses

Fields

Inferred anonymous superclasses

Members

Disjoint classes

SH_url, SH_onwner_id, SH_lon, SH_date_uploaded, SH_width, SH_views, SH_rating, SH_owner_name, SH_lat, SH_category, SH_tags, SH_license, SH_height, SH_thubnail_url, SH_description, SH_title, Sn_item_id

Figure 22: Disjoint classes

To finish with the class of "Fields" we had to add the appropriate value type of each of the subclasses. To do that we select *Annotations tab* for each of the subclass and we add at the *comment* in the *constant tab* one of the *type* listed. For example for the SN_license the vale type is string so we select it in the type.

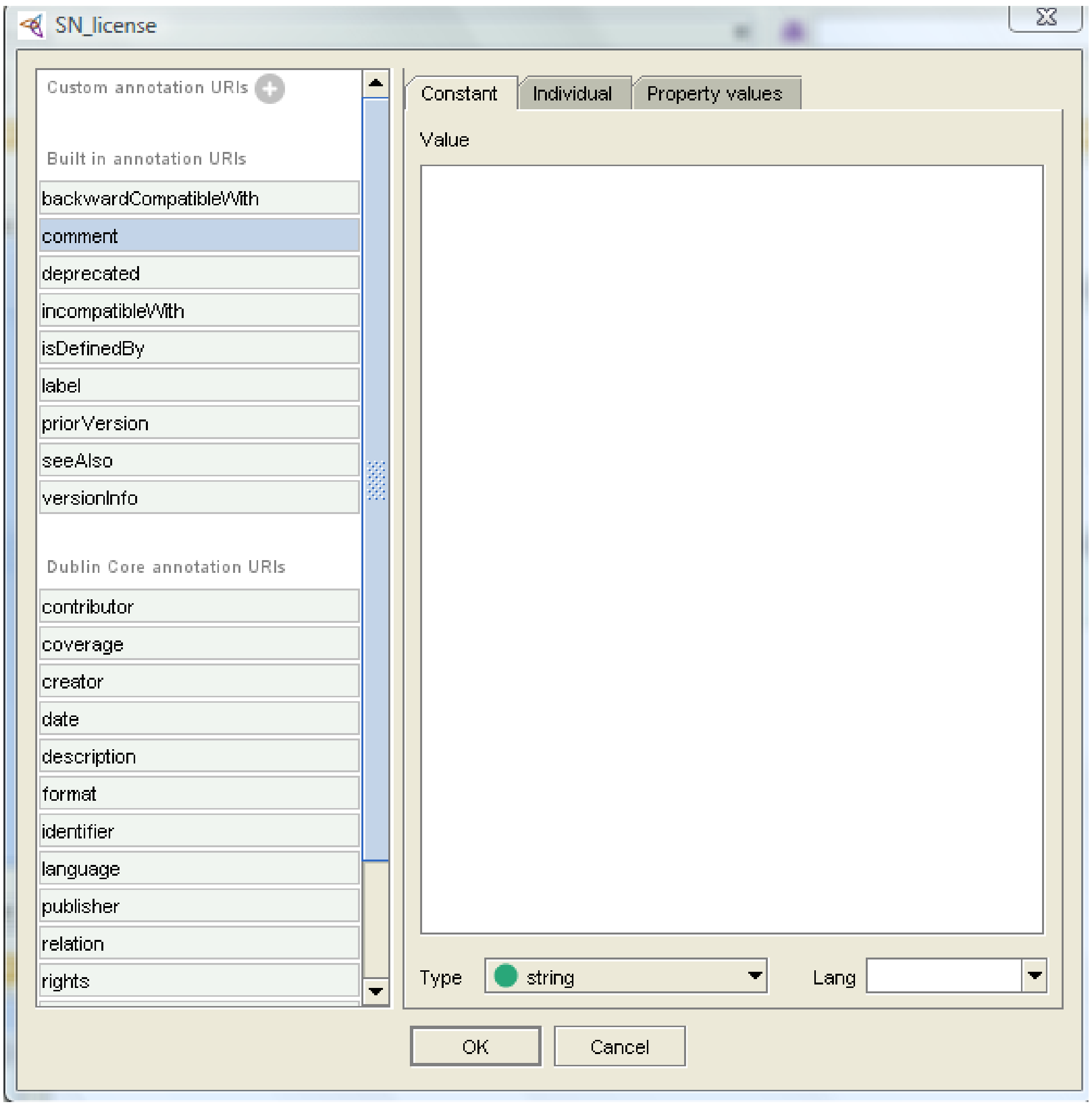


Figure 23: Select a value type

After finishing with the Fields class we move on to the next subclass of the "SOCIOS" class, the Media class. This class has been created in the same way like we described before and it is describing the fields that the SocIoS platform has according to the table 7. The result is shown in the next picture.

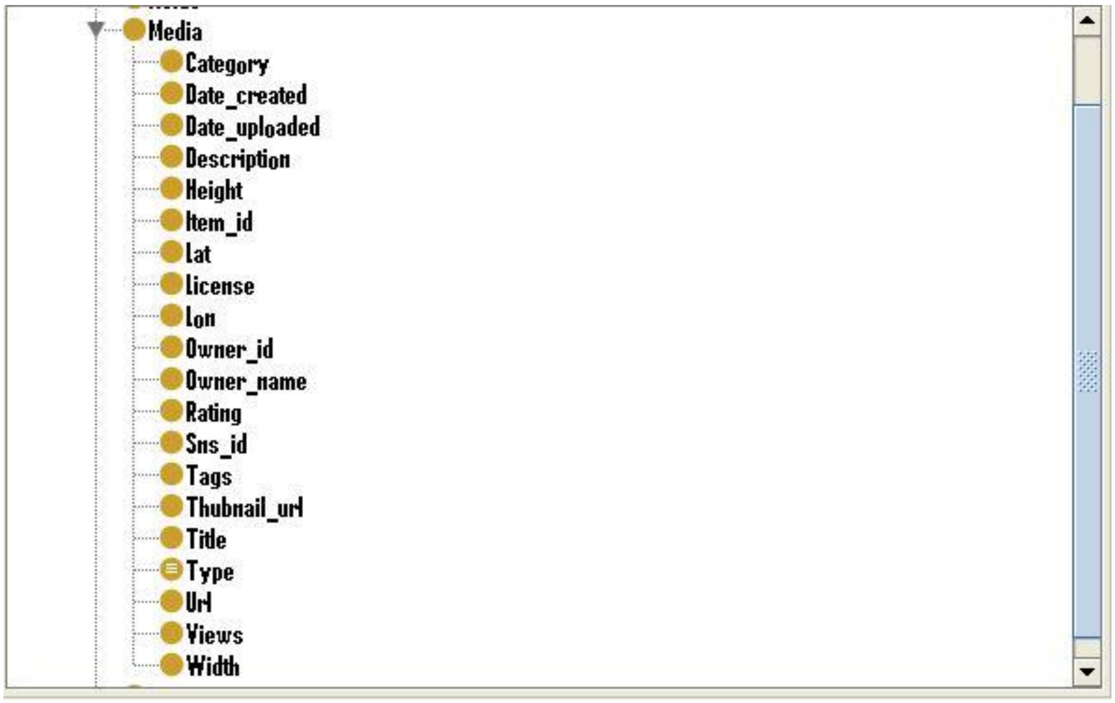
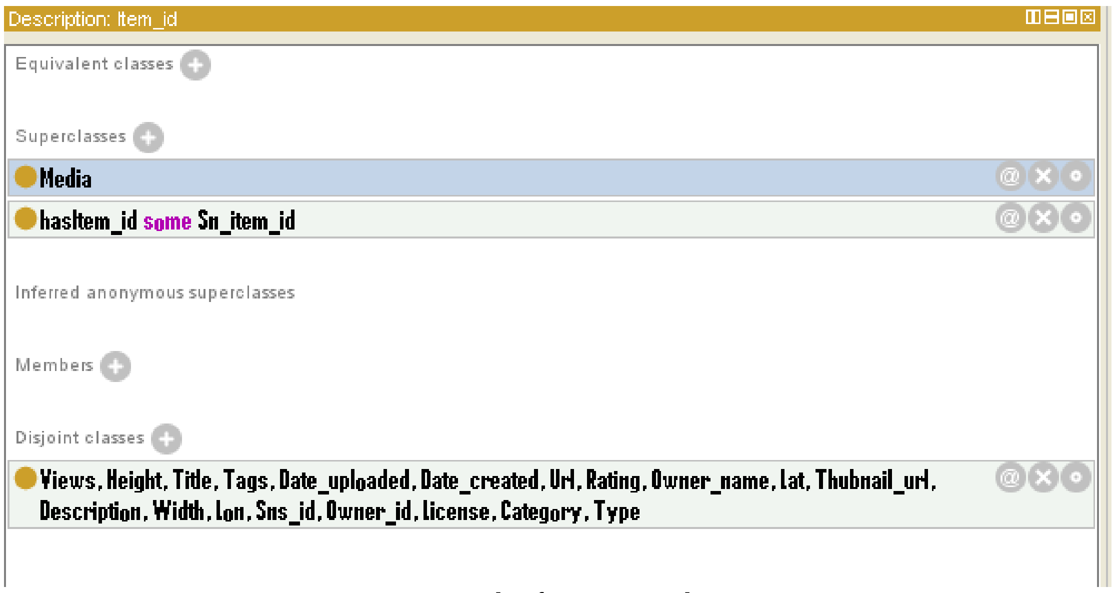


Figure 24: Subclasses of Media

Each media item of the SocIoS platform has these classes but each of these classes has a value from the SN site for example a photo in Facebook has a name and a category, so the title and the tags in the SocIoS have to be completed with theses values. The same happens with the other subclasses of the “Media” class. To do that we had to select each of the subclass of the “Media” class and at the *description tab* add at the *superclass tab* the appropriate existential restriction as it is called using the keyword *some* and the appropriate object property. Existential restrictions, also known as ‘someValuesFrom’ restrictions, or ‘some’ restrictions describe the set of individuals that have at least one specific kind of relationship to individuals that are members of a specific class. For example a member of “category” class must have at least one member of class “SN_category”. This happen because as we said before the “Media” class describes the fields that the media item has in SocIoS network but these fields must have at least one value from the fields of one of the SN site that we described before in the “fields” class. Let’s see an example below:


Description: Item_id

Equivalent classes

Superclasses

Media

hasItem_id some Sn_item_id

Inferred anonymous superclasses

Members

Disjoint classes

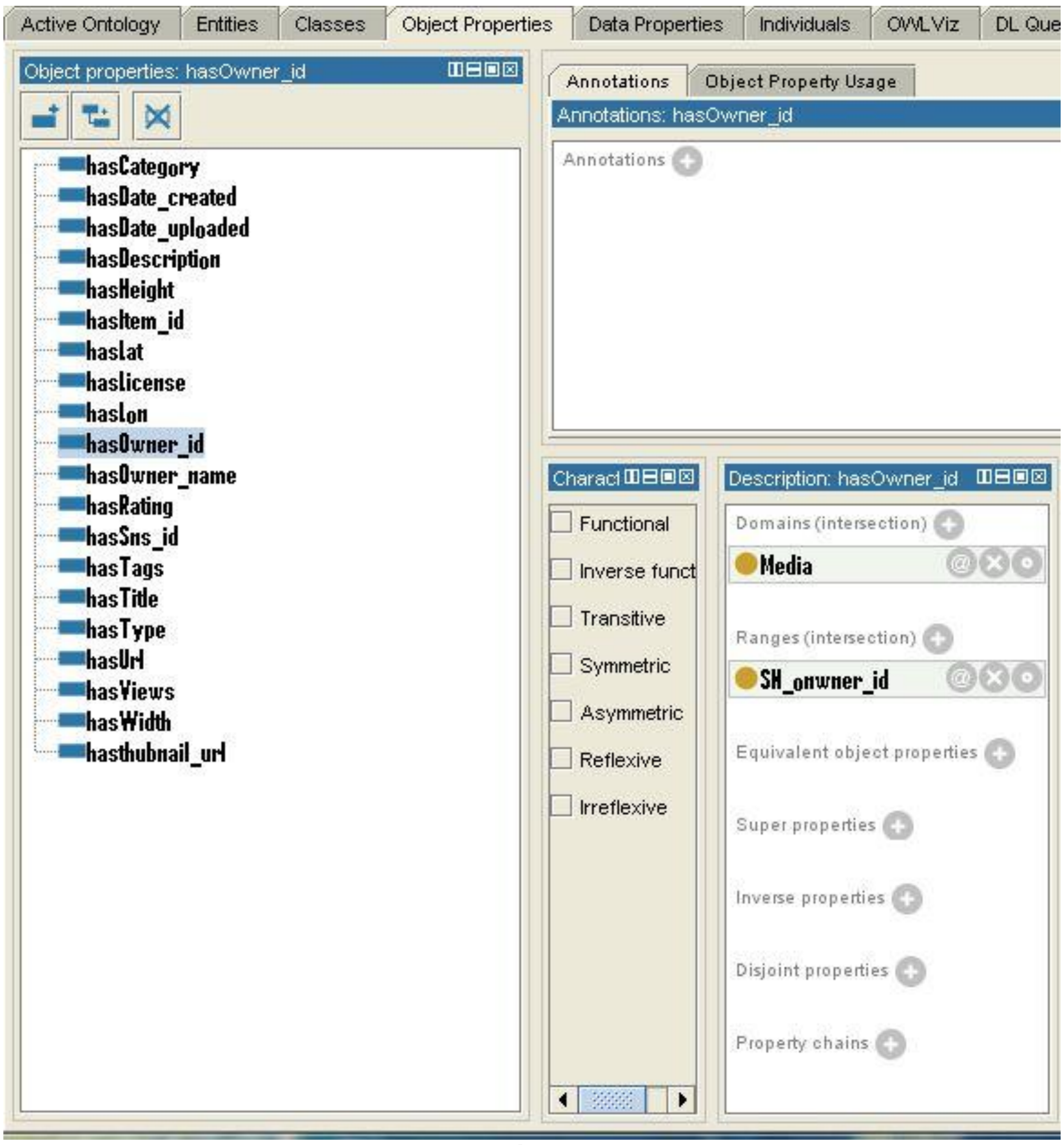
Views, Height, Title, Tags, Date_uploaded, Date_created, Url, Rating, Owner_name, Lat, Thubnail_url, Description, Width, Lon, Sns_id, Owner_id, License, Category, Type

Figure 25: Example of an existential restriction

At the superclasses tab the line "hasItem_id some SN_item_id" represents the existential restriction we mentioned before telling us that an instance of the "hasItem_id" must have at least one value from the "Sn_item_is". The "hasItem_id" is an object property that we create in such a way we will show in the next paragraph. This is necessary in the definition of our ontology because if a media item of SocIoS take its values in the fields from the inserted SN site media item. Moreover we have to mention also that we also disjoint these classes in the same way we did before for the same reason.

6.5.1 Object Properties

OWL Properties represent relationships. There are two main types of properties, Object properties and Datatype properties. Object properties are relationships between two individuals. Object properties link an individual to an individual. Properties may be created using the *Object Properties* tab shown below:



Object properties: hasOwner_id

hasCategory
hasDate_created
hasDate_uploaded
hasDescription
hasHeight
hasItem_id
hasLat
hasLicense
hasLon
hasOwner_id
hasOwner_name
hasRating
hasSns_id
hasTags
hasTitle
hasType
hasUrl
hasViews
hasWidth
hasthubnail_url

Annotations: hasOwner_id

Charact

- [ ] Functional
- [ ] Inverse funct
- [ ] Transitive
- [ ] Symmetric
- [ ] Asymmetric
- [ ] Reflexive
- [ ] Irreflexive

Description: hasOwner_id

Domains (intersection)
● Media

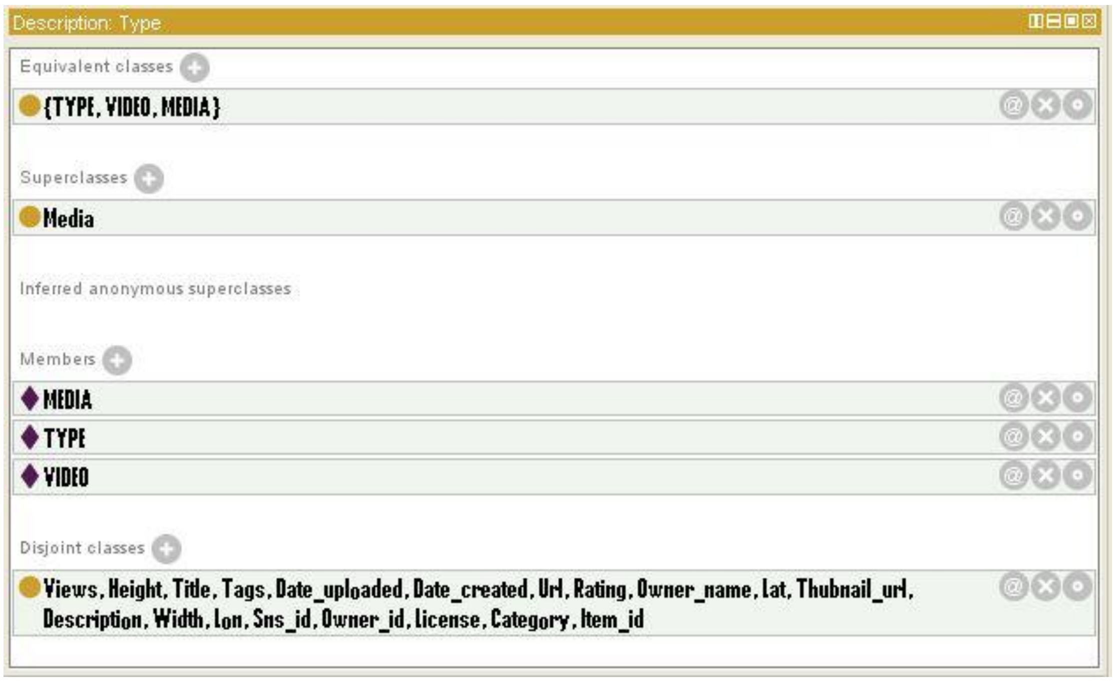
Ranges (intersection)
● SN_onwner_id

Figure 26: Object properties

We can see here that we create an object property for each subclass of “Media” class to describe the fact that a class of “Media” must have at least one subclass of the class “Fields”. We also add for each property the *Ranges* (intersection) tab to define that this property describes individuals from a certain class. For example the property hasTags ranges at the SN_tags class because when we define the subclass of “Media” Tags in the *superclass tab* we added the restriction “hasTags some SN_tags”.

6.5.2 Individuals

We must also add here that in the "Media" class the Type subclass is a lit bit different in the *description view*.


Description: Type

Equivalent classes

● {TYPE, VIDEO, MEDIA}

Superclasses

● Media

Inferred anonymous superclasses

Members

◆ MEDIA

◆ TYPE

◆ VIDEO

Disjoint classes

● Views, Height, Title, Tags, Date_uploaded, Date_created, Url, Rating, Owner_name, Lat, Thubnail_url, Description, Width, Lon, Sns_id, Owner_id, License, Category, Item_id

Figure 27: An example of individuals

The difference between this class and the others class is that this class is a *defined class*. And it can take only 3 values MEDIA, TYPE or VIDEO. According to table 7 the type field is a characteristic for only Flickr, Youtube, OpenSocial and Dailymotion and the permitted values are these 3. So we define the class "Type" in such a way that it is necessary to give each time one of these 3 values. To do that we open the *individuals* tab and we add these 3 values as individuals.

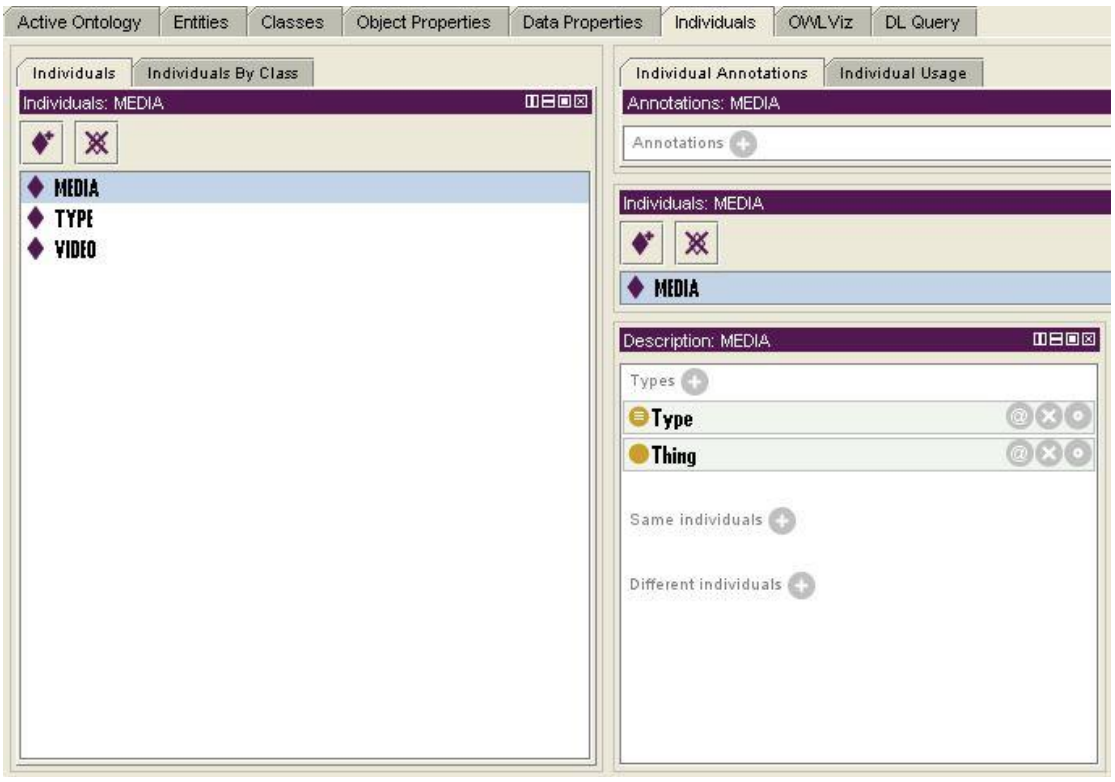
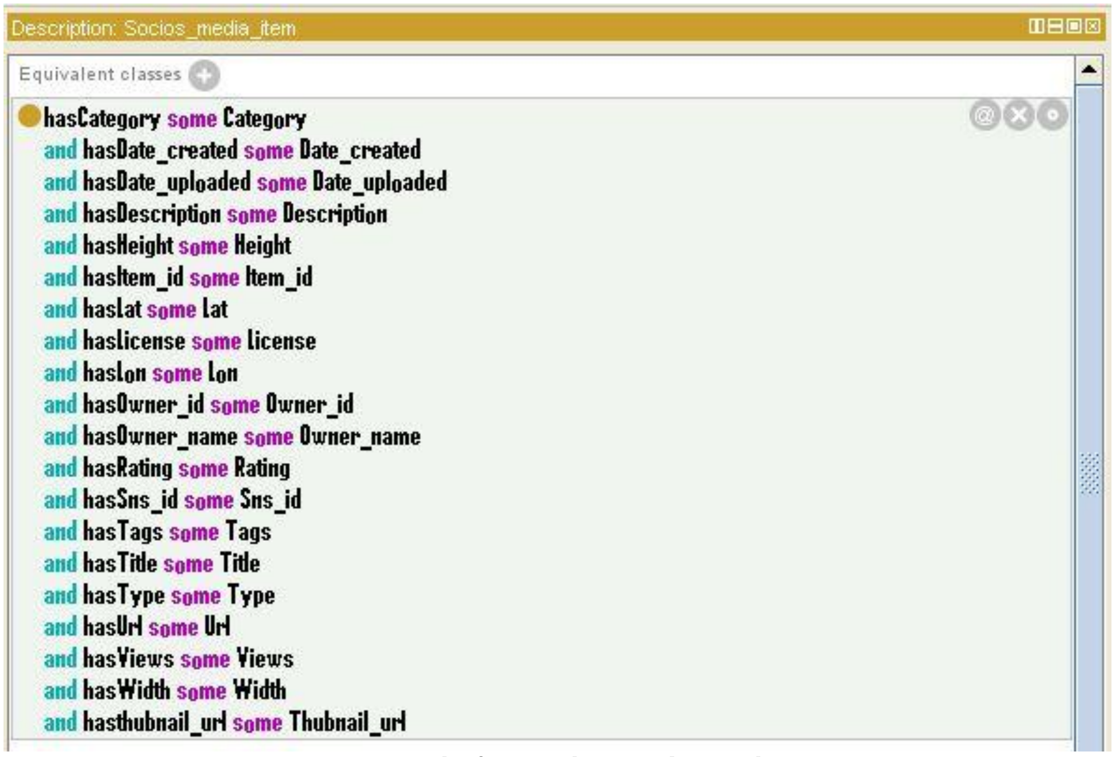


Figure 28: Create an individual

For each of individual we also add at the *Description tab* at the *Types* section the target for these individuals which are the “Type” class. So now at the *Classes* tab we select the class “Type” and at the *equivalent classes* section of *description view* we type {TYPE, MEDIA, VIDEO} and then make the class defines by selecting it and selecting the *convert to defined class* at the *edit* tab. All of the classes that we have created so far have only used necessary conditions to describe them and they were called primitive classes. Necessary conditions can be read as, “If something is a member of this class then it is necessary to fulfill these conditions”. With necessary conditions alone, we cannot say that, “If something fulfils these conditions then it must be a member of this class”. A class that only has necessary conditions is known as a Primitive Class. The difference with the defined classes is that a class that has at least one set of necessary and sufficient conditions is known as a Defined Class. So the “Type” class has now the view that we shown before and the *members* tab has been auto completed because of the individuals that we have created and the target we put.

6.5.3 The definition of the Media item

To complete now our ontology we have to distinguish the media items from each SN site. In fact we have find a way to understand if a media item is coming from Facebook or Flickr or one of the other SN sites. To achieve this we used the Sns_id class which has a unique value for each SN site. So we define each media item for each SN site. For example a media item for Facebook must have a Sns_id= 2 and it must have at least one Category, one Tags, one Title and all the other subclass of "Media" class. In the same way we defined all the media items for the other SN sites. We start be creating another subclass of the "Thing" class called "SocIoS_media_item" then we convert this class as a defined class in the same way we described before and at the *equivalent classes* section we typed the lines we show below to define the necessary and sufficient conditions to have a media item of SocIoS in general. The difference between the different SN sites will be clarified in the next step.



```
hasCategory some Category
 and hasDate_created some Date_created
 and hasDate_uploaded some Date_uploaded
 and hasDescription some Description
 and hasHeight some Height
 and hasItem_id some Item_id
 and hasLat some Lat
 and hasLicense some License
 and hasLon some Lon
 and hasOwner_id some Owner_id
 and hasOwner_name some Owner_name
 and hasRating some Rating
 and hasSns_id some Sns_id
 and hasTags some Tags
 and hasTitle some Title
 and hasType some Type
 and hasUrl some Url
 and hasViews some Views
 and hasWidth some Width
 and hasthubnail_url some Thubnail_url
```

Figure 29: the features that a media item has

Then we made six subclasses of "SocIoS_media_item" one for each SN site to define the media item for each site. These subclasses are Dailymotion_querry, Facebook_querry, Flickr_ querry, Twitpic_ querry, Opensocial_ querry and Youtube_ querry.

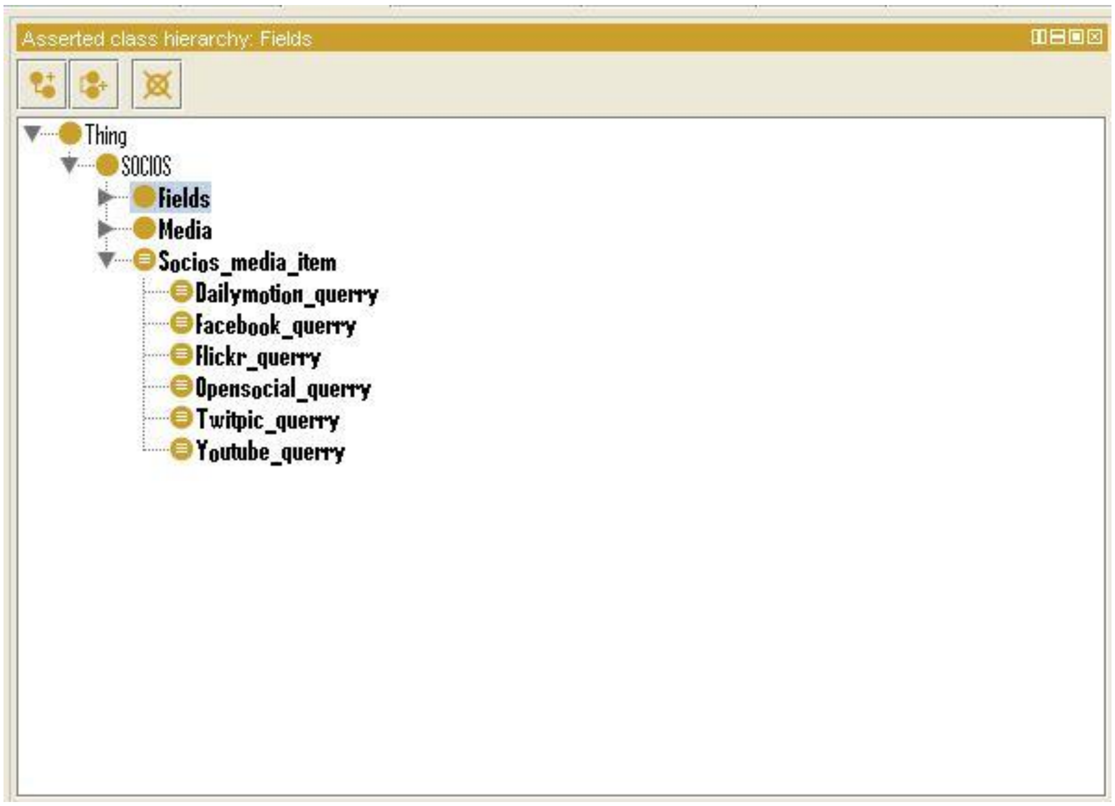
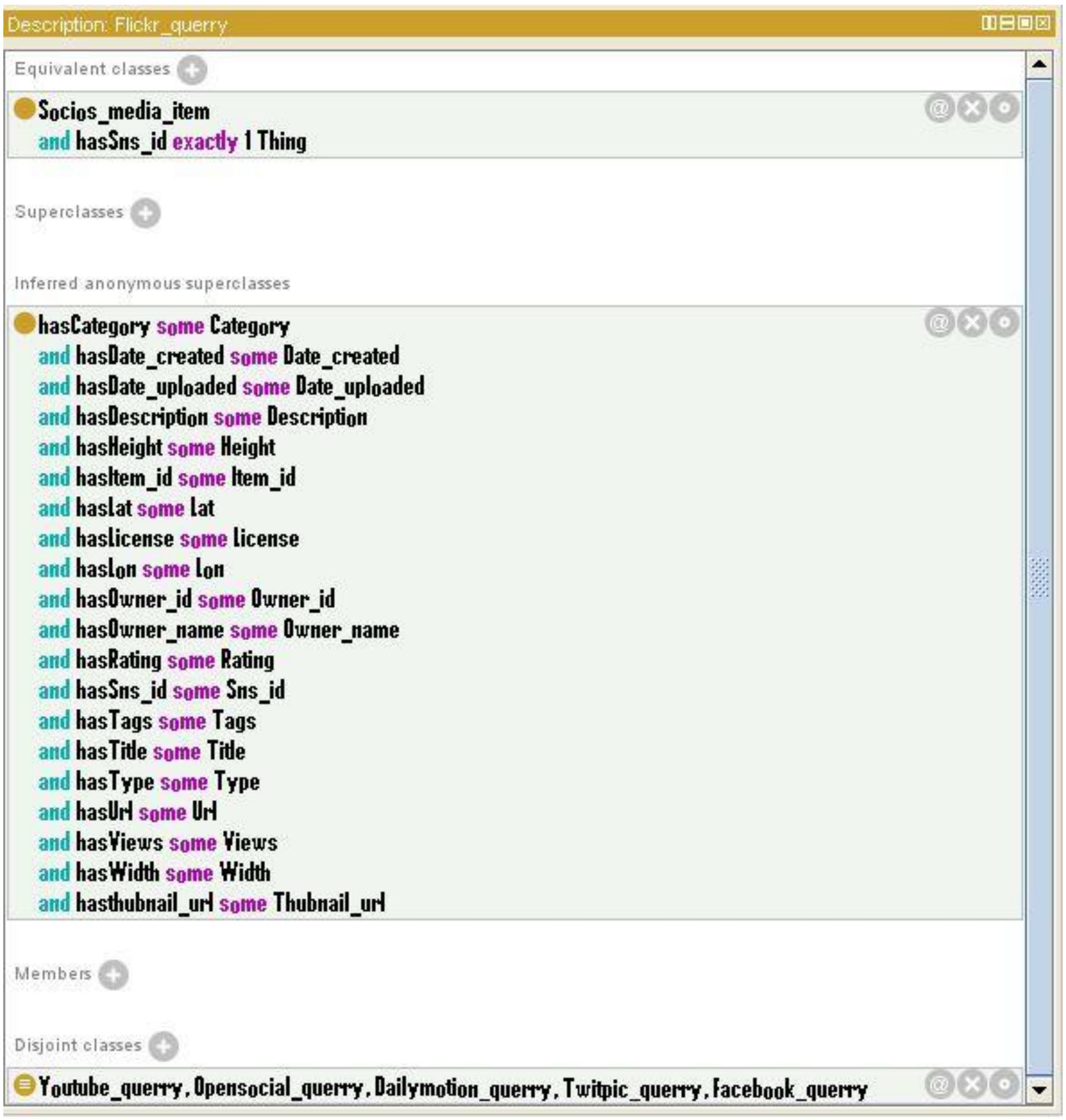


Figure 30: Media items for each SN site

Then we converted also these classes to defined classes and at the *equivalent classes* section we define for each SN site the exact value for the Sns_id and that it must be a “SocIoS_media_item”. So here we used another type of restrictions the Qualified Cardinality Restrictions and the advantage of inheritance that ontologies in OWL have. In OWL we can describe the class of individuals that have at least, at most or exactly a specified number of relationships with other individuals or datatype values. The restrictions that describe these classes are known as Cardinality Restrictions. For a given property P, a Minimum Cardinality Restriction specifies the minimum number of P relationships that an individual must participate in. A Maximum Cardinality Restriction specifies the maximum number of P relationships that an individual can participate in. A Qualified Cardinality Restriction specifies the exact number of P relationships that an individual must participate in. So here for each SN site we specified the exact value (1,2,3,4,5 or 6) for the Sns_id and by defining that each class is “SocIoS_media_item” we add all the features of the

"SocIoS_media_item" to the queries we created. The next figure gives us an example for the flickr_querry.


Description: Flickr_querry

Equivalent classes

```
Socios_media_item
  and hasSns_id exactly 1 Thing
```

Superclasses

Inferred anonymous superclasses

```
hasCategory some Category
  and hasDate_created some Date_created
  and hasDate_uploaded some Date_uploaded
  and hasDescription some Description
  and hasHeight some Height
  and hasItem_id some Item_id
  and hasLat some Lat
  and hasLicense some License
  and hasLon some Lon
  and hasOwner_id some Owner_id
  and hasOwner_name some Owner_name
  and hasRating some Rating
  and hasSns_id some Sns_id
  and hasTags some Tags
  and hasTitle some Title
  and hasType some Type
  and hasUrl some Url
  and hasViews some Views
  and hasWidth some Width
  and hasthubnail_url some Thubnail_url
```

Members

Disjoint classes

```
Youtube_querry, Opensocial_querry, Dailymotion_querry, Twitpic_querry, Facebook_querry
```

Figure 31: Features of a Flickr media item

As we notice in the picture we also disjoint the classes we created because a media item from a SN site can't be a media item from another SN site. In the *equivalent classes* section we added in the *class expression editor:* SocIoS_media_item *and* hasSns_id *exactly* *. Where * is the appropriate number for each SN site and hasSns_id is an object property we create to give for each media item a Sns_id.

6.5.4 Reasoners in ontologies

Now the next step to complete our task was to use the reasoner to be sure about the correct hierarchy and consistency of our ontology. Let’s see some details about reasoners and how useful are in the development of ontologies. A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. The notion of a semantic reasoner generalizes that of an inference engine, by providing a richer set of mechanisms to work with. The inference rules are commonly specified by means of an ontology language, and often a description language. Inference commonly proceeds by forward chaining and backward chaining. In fact, virtually all querying of OWL ontology (and its imports closure) should be done using a reasoner. This is because knowledge in ontology might not be explicit and a reasoner is required to deduce implicit knowledge so that the correct query results are obtained.

Forward chaining starts with the available data and uses inference rules to extract more data until a goal is reached. An inference engine using forward chaining searches the inference rules until it finds one where the antecedent (If clause) is known to be true. When found it can conclude, or infer, the consequent (Then clause), resulting in the addition of new information to its data. Inference engines will iterate through this process until a goal is reached.

For example, suppose that the goal is to conclude the color of a ball named THE BALL depending on the sport either soccer or basketball, given it is round and it weights 8 libres and that the rule base contains the following four rules:

1. If X is round and it weights 8 libres - Then X is a soccer ball
2. If X is round and it weights 15 libres - Then X is a basket ball
3. If X is a soccer ball - Then X is white and black
4. If X is a basket ball - Then X is orange

This rule base would be searched and the first rule would be selected, because its antecedent (If THE BALL is round and it weights 8 libres) matches our data. Now the consequent (Then X is a soccer ball) is added to the data. The rule base is again searched and this time the third rule is selected, because its antecedent (If THE BALL is a soccer ball) matches our data that was just confirmed. Now the new consequent

(Then THE BALL is white and black) is added to our data. Nothing more can be inferred from this information, but we have now accomplished our goal of determining the color of THE BALL.

Backward chaining starts with a list of goals (or a hypothesis) and works backwards from the consequent to the antecedent to see if there is data available that will support any of these consequents. An inference engine using backward chaining would search the inference rules until it finds one which has a consequent (Then clause) that matches a desired goal. If the antecedent (If clause) of that rule is not known to be true, then it is added to the list of goals (in order for one's goal to be confirmed one must also provide data that confirms this new rule).

For example, suppose that the goal is to conclude the color of a ball named THE BALL depending on the sport either soccer or basketball, given it is round and it weights 8 libres and that the rule base contains the following the same rules like before. This rule base would be searched and the third and fourth rules would be selected, because their consequents (Then THE BALL is white and black, Then THE BALL is orange) match the goal (to determine THE BALL's color). It is not yet known that THE BALL is a soccer ball, so both the antecedents (If THE BALL is a soccer ball, if THE BALL is a basket ball) are added to the goal list. The rule base is again searched and this time the first two rules are selected, because their consequents (Then X is a soccer ball, Then X is a basket ball) match the new goals that were just added to the list. The antecedent (If THE BALL is round and it weights 8 libres) is known to be true and therefore it can be concluded that THE BALL is a soccer ball, and not a basket ball. The goal of determining THE BALL 's color is now achieved (THE BALL is white and black if it is a soccer ball, and orange if it is a basket ball, but it is a soccer ball since it is round and it weights 8 libres. Therefore, THE BALL is white and black).

6.5.5 The Reasoner Pellet

As we said before in our ontology we used the reasoner Pellet. Pellet is the first sound and complete OWL-DL reasoner with extensive support for reasoning with individuals (including nominal support and conjunctive query), user-defined datatypes, and debugging support for ontologies. It implements several extensions to OWL-DL including combination formalism for OWL-DL ontologies, a non-monotonic operator, and preliminary support for OWL/Rule hybrid reasoning. Pellet is written in Java and is open source.

Pellet started as a proof of concept system to help meet the W3C's implementation experience requirements for the Web Ontology Language (OWL). It has since become a practical and popular tool for working with OWL. Pellet has been the first reasoner to support all of OWL-DL, i.e. the Description Logic (DL) *SHOIN*(D), and recently has been extended to support the new features proposed in the so-called OWL 1.1 effort,1 i.e. the DL *SROIQ*(D). OWL 1.1 extends OWL-DL with qualified cardinality restrictions, complex subproperty axioms (between a property and a property chain), local reflexivity restrictions, reflexive, irreflexive, symmetric and anti-symmetric properties, disjoint properties. Pellet is implemented in Java and is open sourced under a liberal license. It offers panoply of features including conjunctive query answering, rule support, *E*-Connection reasoning, and axiom pinpointing, among others. To make its reasoning capabilities easily accessible to users, Pellet provides various interfaces including a command-line interface, an interactive Web form for zero-install use, DIG server implementation, and API bindings for RDF/OWL toolkits Jena and Manchester OWL-API.

Pellet, in its core, is a Description Logic reasoner based on tableaux algorithms. The tableaux reasoner checks the consistency of a knowledge base and all the other reasoning services are reduced to consistency checking. The reasoner is designed so that different tableaux algorithms can be plugged in. The default algorithm handles *SROIQ*(D) but there are several other tableaux algorithms implemented, e.g. for non-monotonic extensions and for integration with rules.

To use the Pellet to the protégé and to see the results such as the inferred classes or any inconsistent class we select at the *Reasoner* tab the Pellet.

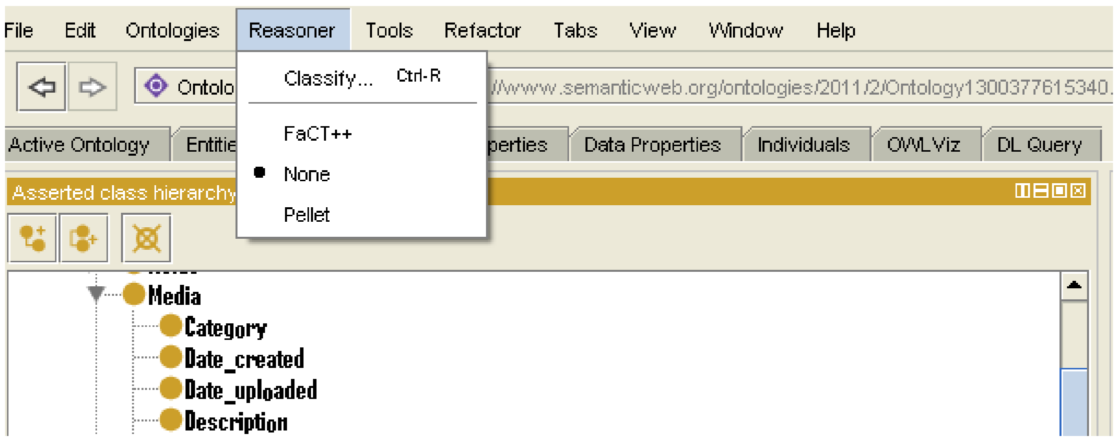
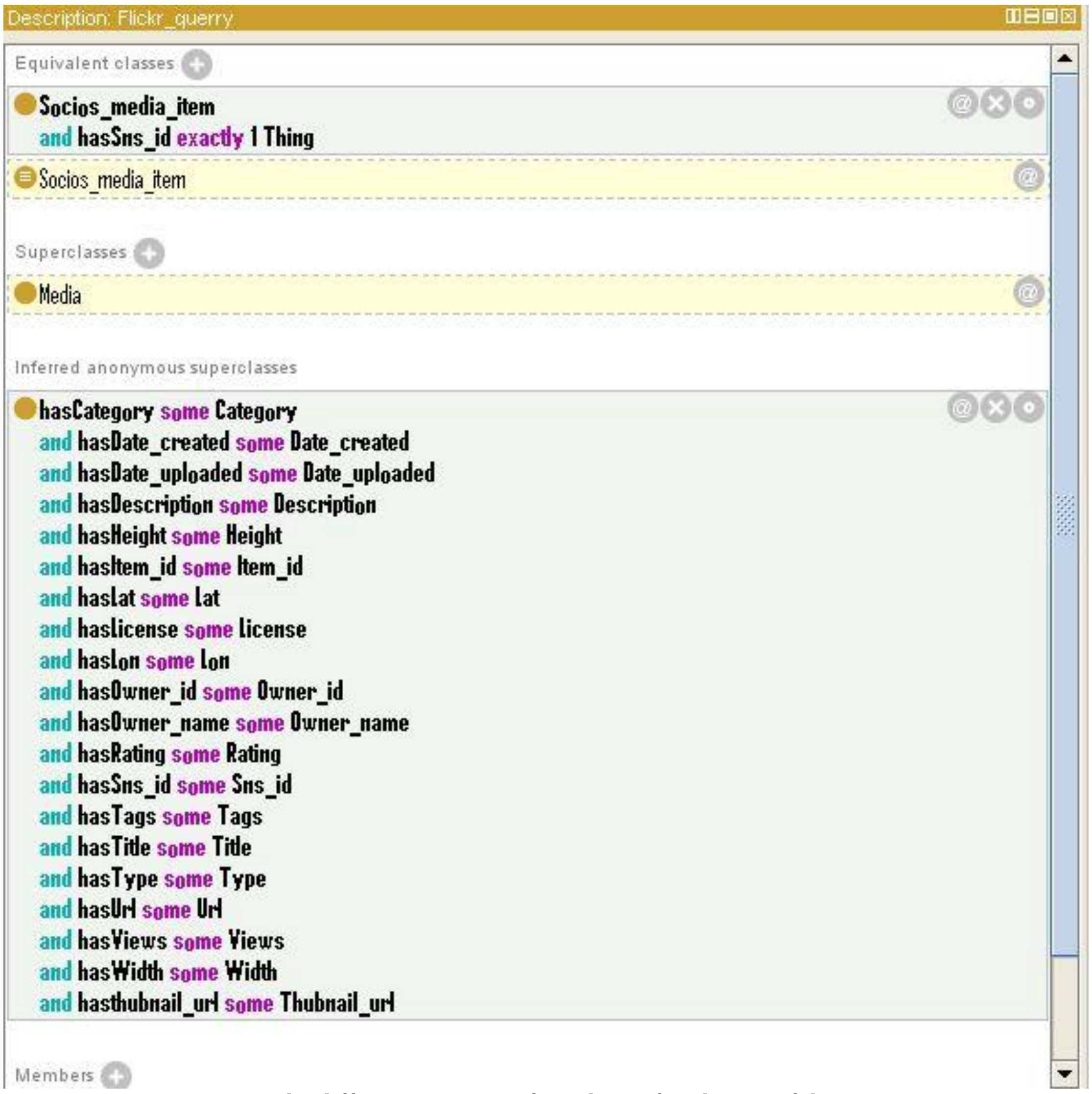


Figure 32: Classify our ontology using a reasoner

Then we see the differences in the description view of some classes in a shading yellow box like. For example for the class “SocIoS_media_item” the subclass Flickr_querry is a little bit different now and we can compare it with the picture 10 that we showed before. The result is below:



```
Socios_media_item
  and hasSns_id exactly 1 Thing
Socios_media_item
```

```
Media
```

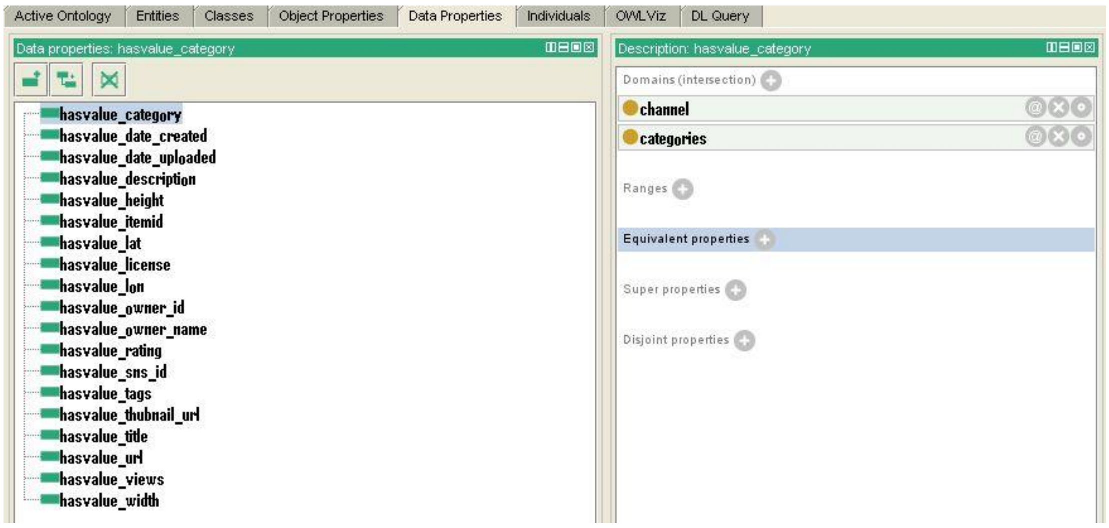
```
hasCategory some Category
  and hasDate_created some Date_created
  and hasDate_uploaded some Date_uploaded
  and hasDescription some Description
  and hasHeight some Height
  and hasItem_id some Item_id
  and hasLat some Lat
  and hasLicense some License
  and hasLon some Lon
  and hasOwner_id some Owner_id
  and hasOwner_name some Owner_name
  and hasRating some Rating
  and hasSns_id some Sns_id
  and hasTags some Tags
  and hasTitle some Title
  and hasType some Type
  and hasUrl some Url
  and hasViews some Views
  and hasWidth some Width
  and hasthubnail_url some Thubnail_url
```

Figure 33: The difference in view of ontology after the use of the reasoner

The inferred class hierarchy tell us that flickr_querry is also a SocIoS_media_item something that is true because from the definition of this class we define it as a SocIoS_media_item. So after the reasoning the result – inferred class is shown. We must also note here that if the reasoner finds any mistakes about the class hierarchy or consistency the result is shown in red letters and it declares where the problem is.

6.6 Applied Example

The main purpose of the ontology that we have created was to make a common vocabulary for the terms and fields of the social network sites, as we meet them in their APIs, that are been used in the SocIoS platform. So using the data properties that we created we can pass some values to the classes that we want and make some instances for the fields of a media item.



Data properties: hasvalue_category

- hasvalue_category
- hasvalue_date_created
- hasvalue_date_uploaded
- hasvalue_description
- hasvalue_height
- hasvalue_itemid
- hasvalue_lat
- hasvalue_license
- hasvalue_lon
- hasvalue_owner_id
- hasvalue_owner_name
- hasvalue_rating
- hasvalue_sns_id
- hasvalue_tags
- hasvalue_thubnail_url
- hasvalue_title
- hasvalue_url
- hasvalue_views
- hasvalue_width

Description: hasvalue_category

Domains (intersection)
- channel
- categories

Ranges

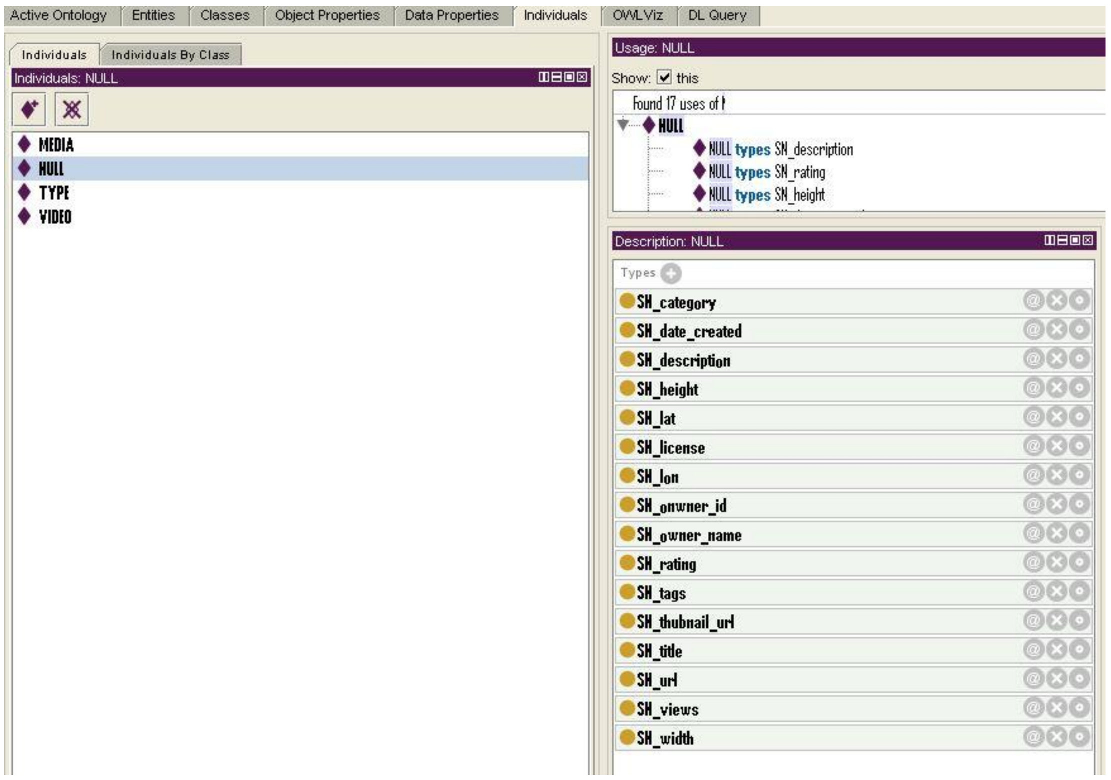
Equivalent properties

Super properties

Disjoint properties

Figure 34: create a data property

For example we may want to create a media item for Facebook. Using the *data properties* that are shown above we make an instance with values for all of the fields of the ‘Media’ class. For the classes that are not supported in Facebook as it happens the class”SN_lat’ we select the *individual* “Null’ that we created for that reason. We made an *individual* called ‘Null’ and we targeted almost all of the SN_classes because some fields are not supported in some Social network Sites (according to the table 7). So we can have the choice to select the ‘Null’ value, that is a member of these classes, to represent that for a particular site a value for a field must be empty-null.



Individuals: NULL

- MEDIA
- NULL
- TYPE
- VIDEO

Usage: NULL

Show: ☑ this

Found 17 uses of N

- NULL
  - NULL types SN_description
  - NULL types SN_rating
  - NULL types SN_height

Description: NULL

Types

- SN_category
- SN_date_created
- SN_description
- SN_height
- SN_lat
- SN_license
- SN_lon
- SN_onwner_id
- SN_owner_name
- SN_rating
- SN_tags
- SN_thubnail_url
- SN_title
- SN_url
- SN_views
- SN_width

Figure 35: create the null individual

After creating some instances for media items and uploading the ontology in the Web (semantic web) we can search for some of these instances according to some passed values. A suggested web application to show us a usage of our ontology is to return the values of the fields as they are in the API of each Social network Site. For example we may search according to the fields-values of the SocIoS platform and then return the fields-values of the social network site that we select. This is easy due to the common vocabulary that we created. So we may search for an 'owner_id'=nikos and for 'tags'=sport as it is shown in the SocIoS and by selecting the Sns_id that we want, each SN site has a unique Sns_id, a json or xml form may return with the values of this media item as it is represented in the API of the Social network Site.

6.7 Results and discussion

Knowledge management becomes more and more important as the volume of information transferred within a company, organization, the internet and generally in all features of modern society is increasing. The more effective and faster data processing results in the increase of corporations' profitability, reduction of various research programs' cost and in the provision of more qualitative consumer services.

The most effective knowledge management can be achieved by developing an ontology, which models the specific domain, seeking relationships between data and rules that govern these relationships. There are two levels at which ontologies play a key role in sharing and management of knowledge. First, ontologies enable humans to seek and distribute knowledge; secondly, ontologies facilitate the machines to communicate with each other, thus having greater potential for information processing.

In conclusion we can say that the ontology of the media items of the SocIoS platform may be very useful for many applications running in the Social network Sites and for the other usages of the SocIoS platform. Taking a look to the thesis we can do some steps to understand the structure of the social network sites and the ontologies which are very important nowadays that Semantic web is the spotlight. Ontologies started to have place to many different sections of our lives and they describe many things that we use. More specifically, in the field of social networks, the development of ontologies can help to organize the immense amount of information that is currently in Internet and to facilitate the search in this.

6.8 References

[78] http://www.json.org/
[79] Dailymotion Apis documentation
[80] http://owlapi.sourceforge.net/reasoners.html
[81] www.sciencedirect.com - Pellet: A practical OWL-DL reasoned
[82] Developers.facebook.com
[83] code.google.com/Apis/OpenSocial
[84] dev.twitpic.com/docs/
[85] www.flickr.com/services/api
[86] code.google.com/Apis/youtube/developers_guide_protocol.htm
[87] FP7-ICT-2009.1.2_DoW_SocIoS_v1.18
[88] Exploiting Social Networks for Building the Future Internet of Services- SocIoS (Project number: 257774. Call identifier FP7-ICT-2009.1.2; Internet of Services, Software and Virtualisation

7. Conclusion

7.1 Future applications

The ontology that we created can be the base for future web applications running in the Social networking Sites. As we already mentioned the Semantic web has many possibilities for the data mining, extracting data from a web page and using that data to run applications in many different sections.

In recent years, Social networking Sites have become an integral part of everyday life, especially for young people, and have many features that can be used either to develop new markets or to tap into data that are in them. For that reasons an ontology to describe the fields that describe media items or other features of these sites is necessary. The improvement and expansion of such an ontology could contribute to creating a "global Social Network Site" which will address to all users to search for any personal data or activity recorded in personal pages of the SN sites.

For the section that we discuss it would be very useful to have the common vocabulary for the Social network sites to develop applications that may run in each Social network. That happens because the application that we meet in the sites are based on the API of each Social network Site and each API is based on the fields that we defined in our ontology. So now we can translate the fields of one social network site to another and try to run an application to more than one site. For example we may want to search for the uploaded videos that a user has at his account on Facebook and on Dailymotion. Using the ontology for the media items, we can accomplish shortly this effort because the values of the requested fields that we may want to search can be easily searched and found to another Social network Site.

In addition to reducing the time and cost of search, the application ontology will could contribute in the search of personal information or by organizing information which provide the ontologies, the use of statistics is made easier and pooling the findings more effectively, consistently, to have a better view of the data stored in the SN sites.