An Intelligent Multi-Agent System Approach to Automating Safety Features for On-Line Real Time Communications

Agent Mediated Information Exchange

Katrinna MacFarlane

A thesis submitted to the University of Huddersfield in partial fulfilment of the requirements for the Degree of Doctor of Philosophy

The University of Huddersfield

September 2016

For Dan and Raiven, everything I do is for you, and for the rest of my family, whose faith in me is a never-ending source of inspiration.

Acknowledgements

I need to thank my children, Dan and Raiven, who have always been a source of inspiration, driving me to do the best that I can and, by their own determination to overcome many obstacles in the pursuit of their goals, have set me, as well as those around them, an excellent example of how to conquer adversity. I must thank my mother for encouraging to believe that I could do anything I set my mind to.

I would like to thank Dr Violeta Holmes for her support, encouragement, patience and guidance throughout the lifespan of this work. We never fail to come up with numerous solutions to every issue, her enthusiasm to make progress is infectious and has ensured that, even in times of hardship, despondency, and frustration, the work has carried on.

I also have to thank the rest of my family, friends and colleagues, who have had to put up with me through the highs and lows of developing the prototypes and writing the thesis. Their support and encouragement have been invaluable.

Part of this work has used the Protégé resource, which is supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

The work presented in this thesis contains elements of all works, both published and pending publication, as described in the Publications section of this thesis on page 6, as well as reports presented for progression through the stages of PhD study at the University of Huddersfield.

Copyright Statement

Abstract

Child safety online is a growing problem, governmental attempts to highlight and combat this issue have not been as successful as it was hoped, and still there are highly publicised cases of children, young people and vulnerable adults coming to harm as a result of unsafe online practices. This thesis presents the research, design and development of a prototype system called SafeChat, which will provide a safer environment for children interacting in online environments.

In order to combat such a complex problem, it is necessary to integrate various artificial intelligent technologies and autonomous systems. The SafeChat prototype system discussed within this research has been implemented in Java Agent Development Environment (JADE) and utilises Protégé Ontology development, reasoning and natural language processing techniques. To evaluate our system performance, comprehensive testing to measure its effectiveness in detecting potential risk to the user (e.g. child) is in constant development. Initial results of system testing are encouraging and demonstrate its effectiveness in identifying different levels of threat during online conversation.

The potential impact of this work is immense, when used as a plug-in to popular communications software, such as Facebook Messenger, Skype and WhatsApp. SafeChat provides a safer environment for children to communicate, identifying potential and actual threats, whilst maintaining the privacy of their discourse. The SafeChat system could be easily adapted to provide autonomous solutions in other areas of online threat, such as cyberbullying and radicalisation.

Publications

2016 Multi-Agent System for Safeguarding Children Online - SAI Intelligent Systems Conference, September 20-22, 2016 | London, UK, Publication pending

2009 MacFarlane, Katrinna and Holmes, Violeta (2009) Agent-Mediated Information Exchange: Child Safety Online. In: 2009 International Conference on Management and Service Science. IEEE, pp. 1-5

2008 Agent Mediated Information Exchange - In IMSI Conference Pisa, Italy (2008)

Posters

2010 Agent Mediated Information Exchange: Holmes, Violeta and MacFarlane, Katrinna (2009) Agent Mediated Information Exchange. In: University of Huddersfield Research Festival, 23rd March - 2nd April 2009, University of Huddersfield.

2009 Agent Mediated Information Exchange (Poster) - British Computer Society, Hopper Colloquium 2009.

Table of Contents

38,073 words.

Table of Figures

List of Tables

Chapter 1: Introduction

Global governmental efforts to address the issue of child safety in an online setting continue (OECD, 2012: Online). The Internet Taskforce on Child Protection was established in the United Kingdom in March 2001. The task force went on to release a comprehensive set of guidelines for safe practice on the internet aimed at parents and children in 2010. Whilst this was well publicised at the time, it failed to address incidents of children compromising their safety.

Recently, highly publicised cases of young people going missing as a result of meeting strangers contacted online (BBC, 2014: Online; Martin, 2010: Online; Hunt, 2014: Online), have emerged which suggests that the problem is still not being resolved with any great success. The whole issue is compounded by the fact that social networking mediums such as Facebook and sites like Ask.fm are becoming more popular than ever. Misuse of some of these sites, by anonymous users is leading to documented cases of young people going missing and in cases where cyber bullying is highlighted, committing suicide (NoBullying.com, 2015: Online).

Another contributing factor is the proliferation of ubiquitous computing. The omnipresent connected society is reaching children much earlier in their development, where even the most informed parents are allowing children as young as five access to mobile devices (Press Association, The Guardian. 2013: Online). Consequently, children have access to social media and instant messaging applications available on those platforms, exposing them to risk.

Recently the United Nations have published an updated version of the Convention on the Rights of the Child (UNICEF, 2015: Online). In article 17, it stipulates that children have a right to access information from mass media. In order to meet these requirements children must have access to the internet.

According to the Office for National Statistics findings outlined in the Internet Access – Households and Individuals 2015 report released in August (2015: Online), that the internet in the United Kingdom was used by 78% (39.3 million) of adults on a daily basis. Of these 96% of users, aged between 16 and 24 accessed the internet from a mobile or work device. As figure 1 indicates, both internet usage and computer usage is on the rise across all age groups.

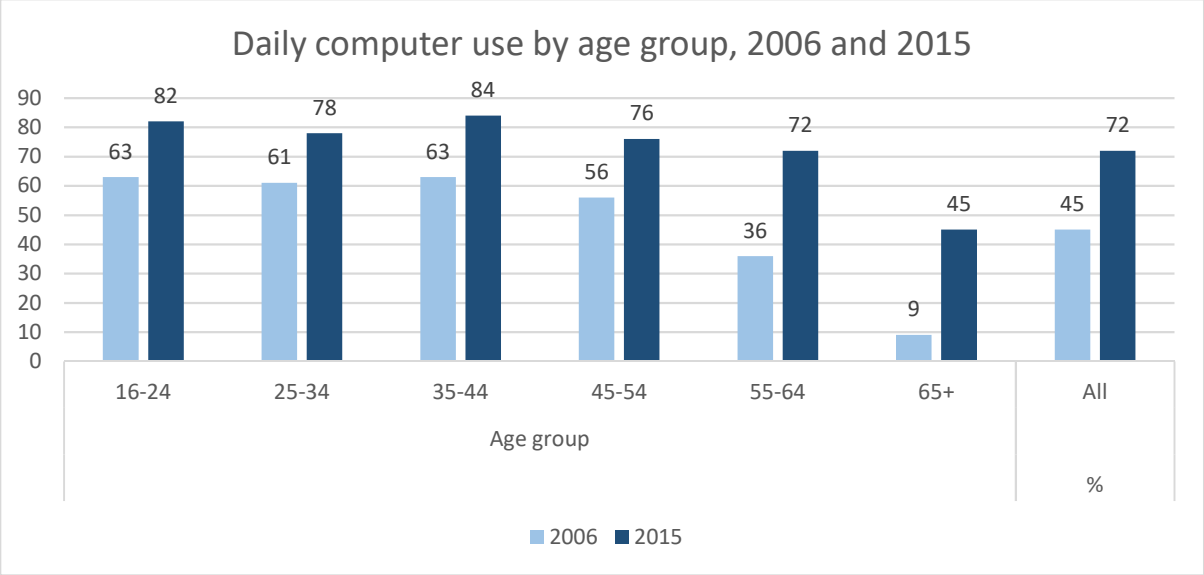


Figure 1: Daily computer use by age group in the United Kingdom

The report goes on to state that the number of households in the United Kingdom with internet access has risen to 86% (22.5 million) in 2015 from 57% in 2006.

In order to ascertain a level of adoption of government guidance for families with young children and adherence to the articles set out in the United Nations Convention, we have conducted a number of surveys over the last 7 years. At the outset of the project, research was carried out amongst 437 schoolchildren, 59% of those who took part regularly chatted to people over the internet. 24% of those who did chat online admitted to giving out an element of their own personal information, these included home telephone number, mobile telephone number and home address with 37 of the children saying that they had arranged to meet someone they had met online (MacFarlane and Holmes, 2009).

From December 2015 to March 2016 a focus group of 29 parents were asked to complete an online survey into online access, supervision, application and privacy for their children. A detailed breakdown of all the questions and their answers can be found in the appendix (item B). Whilst most parents stated that they did worry about their child's safety in an online setting, as seen in figure 2, they went on to confirm that they would let their children access applications and the internet unsupervised once they reached a certain age.

Figure 2: Breakdown of results of parent questionnaire – worry about online safety

When asked to specify a safe age for unsupervised access to online environments the parents gave a range of responses from the options available, over 50% of the parents consulted thought that only children over the age of 12 should be unsupervised on the internet, see figure 3.

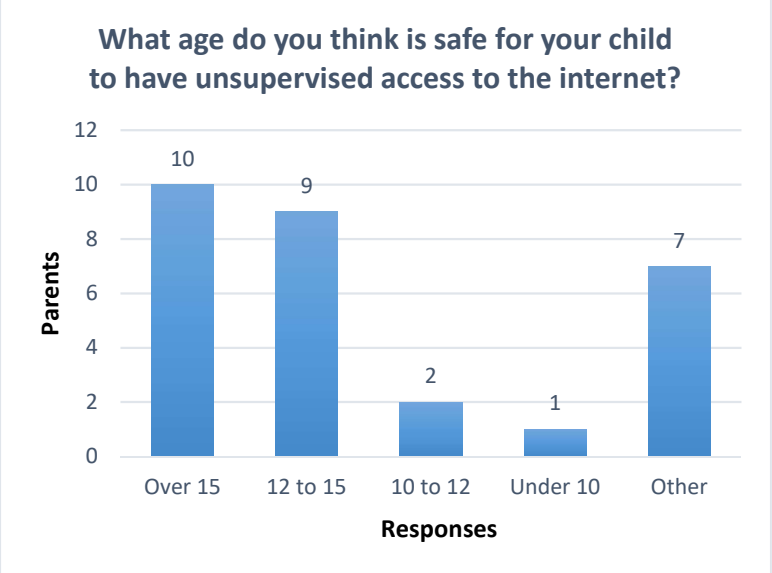


Figure 3: Breakdown of results of parent questionnaire – unsupervised access

Out of the 29 parents polled, twenty confirmed that they had allowed their child to have a mobile phone. Figure 4 shows a breakdown of the applications parents would be happy for their children to use. It will be demonstrated later in the thesis (Table 1, page 20) that most of these applications allow free unsolicited access to all registered users, and some of them have billions of subscribers. This could suggest that the peer pressure that children experience which encourages them to have the latest trend in gadgets and software may affect their parents. This is validated by Deborah Bothun in Consumer Intelligence Series (Bothun, 2014) where she articulates that parents "recognise the role of peer pressure and status in their kids' purchase requests".

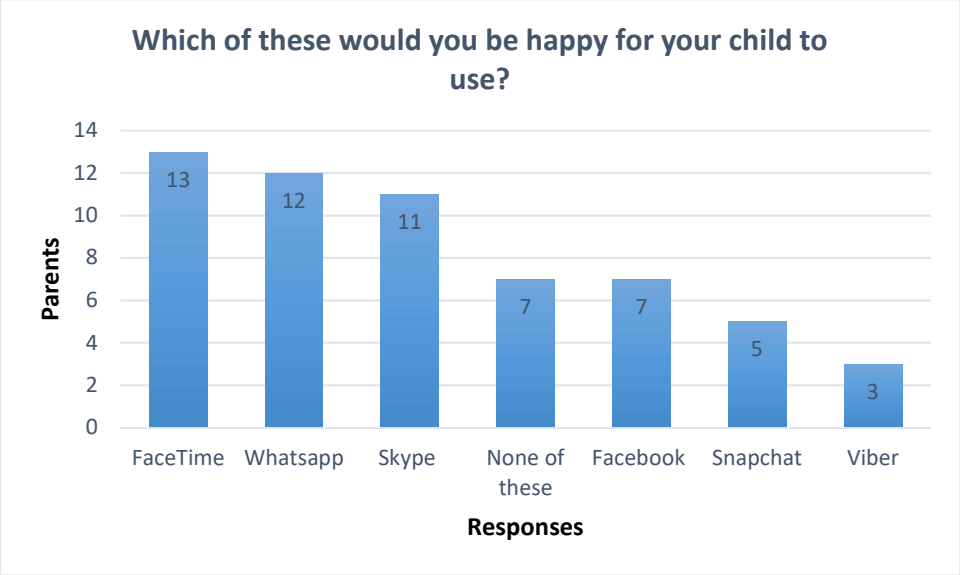


Figure 4: Breakdown of results of parent questionnaire – applications

Given that most parents, 23 of the 29 who took part in the survey, said that they were worried about the safety of their children whilst they were online, it is perhaps a little strange to see that some parents would allow children unsupervised access to the internet at 10-12 years old (2), and in one case younger than 10.

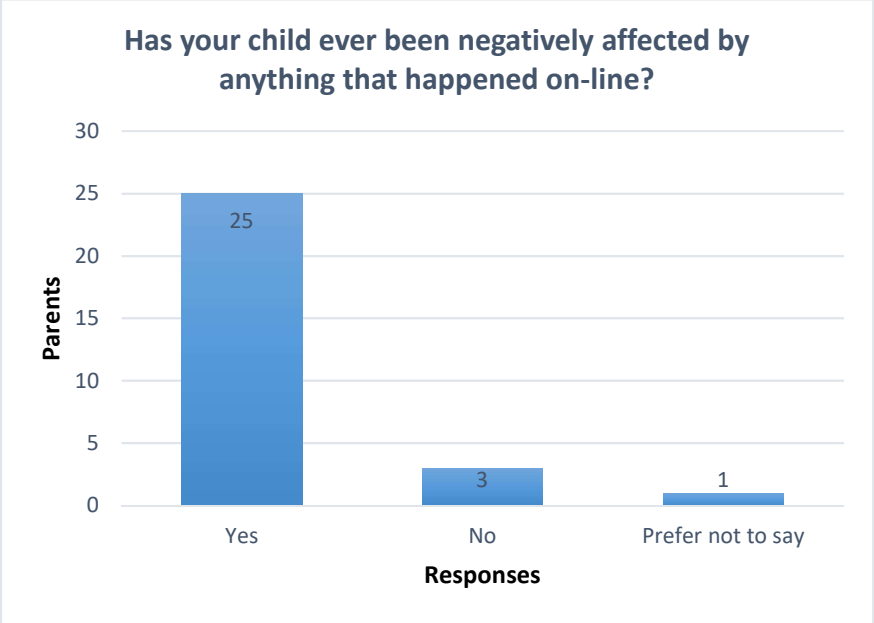


Figure 5: Breakdown of results of parent questionnaire – negative experience

Perhaps the most worrying statistic to come out of the short questionnaire was that 25 of the 29 parents polled, said that their child had been negatively affected whilst online, as seen in figure 5.

1.1 Problem Identification

The British Broadcasting Company (BBC) reported on 12th February 2016 an article on the case against a prominent premiership footballer who, it is alleged, has “groomed” a 15-year-old female fan via Facebook and text messaging. It is alleged that he talked her into meeting him for a signed shirt. The article goes on to say;

> *After the meeting they exchanged further messages, discussing whether Mr Johnson deserved a thank you kiss for the shirt-signing "or more"*
>
> (BBC [c], 2016: Online)

It is also alleged that the third time they met there was sexual contact. The defendant pleaded guilty to one count of sexual activity with a child and one charge of grooming and was subsequently sacked by his football club as a consequence. He is also unlikely to be chosen to add to his tally of twelve games played for the England national football team.

On the 10th July 2015, the Daily Mail reported that a seventeen-year-old girl had been found safe and well after disappearing to meet a man she had met on Facebook.

In another harrowing report, 25th November 2014, the Independent online newspaper reported that a 19-year-old computer engineer admitted to the murder of a 14-year-old boy that he had met playing online video games (Withnall, 2013: Online).

These three cases highlight some of the ways the dangers and risks associated with online grooming and unsafe practice can affect a child. Table 1 shows a range of the most popular communication applications across multiple platforms available today. It goes on to give the number of subscribers currently associated with these applications.

These applications currently offer no protection from the dangers associated with grooming to those who may find themselves most at risk of unsafe practices. The solution presented in this thesis could be adapted to provide that layer of protection that works, once activated, from within the application. This could be set up like a profanity filter, which is used in some online games, including World of Warcraft, to ensure that younger players are not able to see bad language/cursing in their chat windows.

Example Instant Communication Applications			
Application	**Available Platforms**	**Application Information**	**Subscriber Information**
Facebook	Mobile, Web, Desktop, Console	Harvard college students launched this application in 2004. Users create a profile and then add "friends", which could be people they know, people their friends know or people who request to be added to their friends list. Once someone has identified as a 'friend' they are able to exchange messages, pictures and links. Friends also have access, based on permission levels, to content and information on the user's profile page or 'wall'. Many other communication services, websites and application now encourage you to use you Facebook login details to access their services, this then allows those applications and services access to the user's profile information.	**1.18 billion** active users, Facebook (2015: Online)
Skype	Mobile, Web, Desktop, Console	Skype was originally released in 2003, with the first public beta being made available to user in August of that year, Emirates News (2015: Online). This application allows users to add contacts, receive contact requests and communicate via VOIP (Voice Over Internet Protocol), video and instant messaging. Skype to Skype calls are free, but if you wish to use the application to make telephone calls you need to pay using the debit based user account system. In 2010, it was sold to the Microsoft Corporation and was then integrated into their products (mobile and computer operating systems and devices). Microsoft then phased out MSN Messenger, which had offered some of the same services, and replaced it with Skype. Now Skype automates the integration of Microsoft, Facebook and Skype accounts for its users.	**300 million** active users as of November 2015, Statista (2015: Online)
Snapchat	Mobile	Snapchat was released in September 2011, it allows users to take a picture or a video and share it with their contacts for a limited time (1-10 seconds). Users can embed a text-based message with their picture or video, or manipulate them with filters and effects. Once content is received it can only live on the Snapchat dedicated server for the allotted time, but users can screenshot the message on	**200 million** active users as of November 2015, Statista (2015: Online)

		their mobile device to keep. In May 2014, the application added instant messaging and video chat functionality as stated in The Atlantic (2015: Online).	
WhatsApp	Mobile, Web	Originally released in January 2010, this application provides cross platform instant messaging functionality for smartphone users. WhatsApp allows users to send messages to individuals and groups, share media content and share their location, WhatsApp (2015: Online). It saves the user the cost of sending an SMS message. In February 2014, Facebook acquired WhatsApp for approximately 16 billion US dollars. Voice calling functionality has been integrated into the application since this acquisition.	**900 million** active users as of November 2015, Statista (2015: Online)
Viber	Mobile, Desktop	This application is very similar to Skype in its functionality, although it was initially intended for use on mobile devices, Viber have recently released a desktop version of the application. Viber allows users to send messages, voice and video calls to other registered users free of charge, it allows for group chats and public chats. Public chats allow users to sign up to a group conversation with celebrities from the work of fashion, music, brands and sport (Viber, Online). Viber also makes use of 'stickers' which are an enhancement of 'emoticons'. Stickers are small images depicting a thought, emotion or phrase that can be exchanged in place of a text based message.	**249 million** active users as of November 2015, Statista (2015: Online)
World of Warcraft	Desktop, Mobile	World of Warcraft is one of the most successful video game of its genre; massive multiplayer online role-playing game (MMORPG). At the height of its popularity in 2010 it boasted 12 million active subscribers, Statista (2015: Online) and has held several world records based on subscriber and revenue statistics. Once users purchase the game and set up an account they are able to communicate through direct, area, group, guild and raid chat channels. In 2011 Blizzard released a mobile application so that players could communicate with their friends when not logged into the game, us.battlle.net (2015, Online).	**5.5 million** active subscribers, (Statista, 2015: Online)
Minecraft	Desktop, Mobile,	Minecraft is a sandbox survival game, which saw its first release in May 2009. The game has gone on to achieve	**100 million** registered

	Console, Web	record levels of success and popularity amongst gamers of all ages, but particularly with younger players, 20.5% of players are under fifteen years old, Minecraft-Seeds (2015: Online). Players communicate in this game when they select the multiplayer option, where they can log onto available player hosted servers or choose to create their own server. Once logged into a server, players can communicate in an area chat channel and send messages directly to another player.	users (Makuch, E. 2014: Online)
SMS (Short Message Service), also known as text messaging	GSM, Web	This technology was conceptualized in the early 1980's, but was not offered to British consumers until 1993 by BT Cellnet (now known as O2). This technology is not reliant on internet connections and is one of the easiest and most effective ways to communicate, Nexmo (2015: Online). It boasts a read rate of 90% read rate of messages in minutes, which has led to SMS being the medium of choice for marketing and alert services. Up to 80% of all mobile telephone, users regularly use SMS messaging. In 2005 the average texts sent per user each month was 62, by 2014 that figure has increased to an average of 491 per user per month according to the Statistic Brain Research Institute (2015: Online). In June 2014, the total number of text messages sent was 561,000,000,000.	In 2014 the total number of global cellular phone subscriptions was **6,931,000,000** (Seth Institute, 2015: Online)

Table 1: Example Instant Communication Applications

There are however, many tools available to parents to help them tailor their child's access to internet sites and content. Popular web browsers, such as Microsoft Internet Explorer and Mozilla Firefox have easy to use built in functionality to help parents make choices. There are also applications that run inside the browser to provide an added layer of filter and protection.

Existing Applications for Online Child Protection			
Application	**Safety Features/Restrictions**	**Real-Time Protection**	**Automated**
PlayMessenger	This application offers all of the same functionality as standard communication applications; text, voice, video, group and person to person chat. Only users who have verified their adult status, by providing proof of all of the following; diver license number, social security number or	No	No

	credit card information can set up an account and subsequently set up their children's account. Parents can then set levels of trust, approve friends, block users, add language filters for all of their children. Parents have access to everything the child does using this application. (Social, 2015: Online)		
Net Nanny	Provides a suite of parental controls to allow parents to govern internet access for their children. Features include; internet filter, pornography blocker, time management system, profanity filter, social media monitoring, alerts and reporting, remote administration and user profile management. Parents have access to everything the child does using this application. (Net Nanny, 2015, Online).	No	No
K9 Web Protection	This small application runs inside the web browser and filters out unwanted content when activated. This application does not prevent any communication from websites unless the site is blocked. K9 does not affect any applications running outside of the web browser. (K9 Web Protection, 2015, Online).	No	No
Spyrix Free Keylogger	A key-logging application records all key strokes made. This tool will allow parents to monitor websites and messages sent via applications, but will not see the content of received messages. A parent can only go through this information after it is sent. Parents have access to everything the child does using this application. (Spyrix Software, 2015, Online).	No	No
RooKids	This application enables messaging over SSL (Secure Sockets Layer). It offers parental controls, a contacts list, time management, 24-hour support and a mechanism to report abuse of the system. Camera access in disabled in this application, and while parents have control of the contact list, the child's actual conversations remain private. (Rookids, 2015: Online).	No	No

Table 2: Existing applications aimed at protecting children

Other applications that are recommended to parents include tools to monitor a child's browsing habits, so that a parent can capture browsing history in a third party application (e.g. K9 web Protection). There are also tools to prevent children from accessing applications or allowing them to

install applications onto the device they are using. Table 2 examines some of the most popular software available to parents.

All of these different tools have the same fundamental drawback, they do not allow for the monitoring of interactions between users on the fly, meaning that there is no opportunity to act before the child places himself or herself in a position of risk. Key logging software might alert a parent to bad practices, but the parent would only be made aware of this after the fact. These applications also compromise the child's privacy, each one of these applications would give the parent the ability to access their child's private conversations.

Existing software tools could indeed help to find someone who is missing or provide evidence in a worst-case scenario. However, in order to provide a more robust and effective solution, we intend to utilise artificial intelligence techniques and integrate various technologies, Intelligent Agents, Ontology, Natural Language Processing and Web technologies, in the design and development of The SafeChat System.

1.2 Aims and Objectives

Based on initial investigations it was concluded that there is a need to check that safety guidelines are followed when children chat on-line. Early iterations of development sought to create an application that autonomously ensured that users followed safety measures and that would go on to prevent meeting arrangements.

Further research, into a range of issues, such as internet usage, application development and the sharp rise in the number of available applications for online communication led development in a different direction. Young people want to use popular applications, and popularity changes, as seen with the way Facebook became more popular than Twitter and Snapchat usage is on the rise. WhatsApp and Viber also attract a large number of users. Minecraft has become an immensely popular application and has communication embedded within the game, as seen in table 1.

The aim is now to develop a platform independent system, which is able to integrate with any online communication scenario, which would contain the built in functionality to block the transmission of personal data to other users autonomously and any data that involves the arrangements of meetings between users. The system could then report any attempts to transfer safety-compromising data to the user's nominated parent or guardian, as well as warn the user of the dangers involved.

It is important that any solution also provides transparency for the user; care must be taken to ensure that the users experience does not alter noticeably with the introduction of the software. If the computing overheads of the system are too great, users may look to bypass any safety measures.

An ideal solution will protect the privacy of the child. It is important that children are free to express themselves without fear of repercussions, and application that compromised this may also provoke users to try to bypass the system.

This leads to the following hypothesis:

> *"The development of an intelligent multi-agent monitoring system and application specific ontology will provide a safer environment for children and vulnerable adults when communicating online."*

In order to test this hypothesis, a range of technologies have been investigated. One part of the overarching SafeChat system will be a multi-agent system, which will need to demonstrate effective information blocking and threat level maintenance. An ontology will be produced to demonstrate provision of intelligence for use in decision making within the SafeChat system. The ontology should be imported into an environment where its rules, relationships and functionality can be interrogated using Simple Protocol and Resource Description Framework (RDF) Query Language (SPARQL). Finally, a simple chat server and client system will be developed to demonstrate that the multi agent system can be embedded and triggered for use when needed.

A range of test scenarios will be needed to ensure that the various parts of the system act as intended when faced with different input. Test conversations will be considered to help streamline the ontology and provide clarity on expected results.

1.3 Methodology and Implementation

Guba and Lincoln (2005) recommend that researchers select research methods that are appropriate to meet the aims and objectives of the work being carried out. A single method can be used if the research question is specific and highly focused (Holmbeck et al, 2002), but other avenues of research and the identification of varied approaches can be lost if a mixed method approach is not adopted. Mixed methodology approaches however, are not without limitation; they can be more time intensive and do rely on the researcher's skills when capturing, collating and analysing data (Belk, 2007). The nature of this work lends itself to a mixed method approach, because whilst numerical data can give a value to some of the issues raised, it cannot articulate the effect on the subject. In light of this, quantitive and qualitative data has been sought using primary and secondary sources.

In order to identify literature that is recent and relevant, specific search terms were employed for internet and journal searches. Where appropriate, the literature collated in this work has been published with the last ten years, however where applicable to the technological background, and where advances have been developed, it has been necessary at times to exceed this time frame.

Interviews were sought with representatives from government departments dealing with Online Safety, the Child Exploitation and Online Protection Centre (CEOP), and from a range of application providers (Facebook, Microsoft and WhatsApp) to try to gain insight into the scope of their response to the issues raised in this thesis, but no positive responses were received within the timeline of this research.

Observational research can show how people behave in certain settings, but for this research it was felt that the fact that subjects know that they are being observed would cause them to behave differently, and thus skew any data gathered.

The SafeChat system has been developed using a range of software and techniques. The Java runtime environment has been used, as it provides platform independence as well as compatibility with the JADE agent development framework.

The multi agent system part of the presented solution uses the Java Agent Development framework (JADE) developed by Telecom Italia to offer a simplified environment for the implementation of agent and multi-agent systems. The SafeChat multi-agent system has been programmed using the NetBeans integrated development environment (IDE). The simple chat system has also been developed using NetBeans.

To facilitate testing, a messaging application was developed in Java, which utilised a user registration system that used a MySQL database housed on an apache server, accessed, and administered via PhPMyAdmin all running locally using XXAMP.

The meeting ontology has been developed using Protégé, a free, open source ontology editor, which was created at Stanford University and can be used as a framework for building intelligent systems.

A Fuseki Server running in Apache Jena has been implemented in order to execute SPARQL queries on the meeting ontology to test its functionality and refine the dataset, defined relationships and attributes.

1.4 Research Contribution

In this research the main contribution presented through this thesis is:

Development of a novel system to monitor adherence to guidelines set out by the government to protect personal information and prevent children and vulnerable adults from making meeting arrangements autonomously. This novel system will be designed to provide or consider the following requirements:

- The system will be required to adapt to any other application (e.g. Facebook, Skype, etc.) or platform. This will ensure that this solution is available across the range of popular applications and connectivity devices available now and in the future.
- As much as possible, the system should be transparent to the user, system overheads need to be managed so that the user experience is not compromised.
- This system will protect the privacy of the child and monitor the threat level of discourse autonomously, and will not share the details of that discourse with anyone.
- The system will be designed in such a way that it can be easily adapted to respond to other threats, such as cyberbullying or radicalisation.

This research has been reviewed and published in the following papers, conference presentations and poster presentations:

- MacFarlane, Katrinna and Holmes, Violeta (2016) Multi-Agent System for Safeguarding Children Online - SAI Intelligent Systems Conference, September 20-22, 2016, London, UK, Publication pending
- MacFarlane, Katrinna and Holmes, Violeta (2009) Agent-Mediated Information Exchange: Child Safety Online. In: 2009 International Conference on Management and Service Science. IEEE, pp. 1-5
- MacFarlane, Katrinna and Holmes, Violeta (2008) Agent Mediated Information Exchange -In IMSI Conference Pisa, Italy
- MacFarlane, Katrinna (2009) Agent Mediated Information Exchange: (Poster) In: University of Huddersfield Research Festival, 23rd March - 2nd April 2009, University of Huddersfield.
- MacFarlane, Katrinna (2009) Agent Mediated Information Exchange (Poster) British Computer Society, Hopper Colloquium.

1.5 Organisation of Thesis

This thesis in divided into six chapters. Each chapter starts with a brief introduction, which articulates how the work presented in that section fits with the overarching structure of the research, and ends with a summary of the ideas, concepts and research presented.

Chapter 1: presents the fundamental problem addressed by this research. It contains an introduction and a discussion on the issues presented. Specific cases, existing solutions and the scope of the problem will be examined. This section will begin to look at the technologies, which will be employed in the development of the proposed solution as well as enter a brief discussion on how they will be used.

Chapter 2: presents the background research and a review of current literature. It articulates an analysis of previous studies and uses of technologies and techniques as they apply to similar problems. This chapter focuses on five areas of review; factors contributing to the motivation of study, artificial intelligence, agent technology, ontology development and natural language processing.

Chapter 3: This chapter discusses the methodology employed to create the component parts required to provide an overarching SafeChat system solution. It goes on to look at the development of the software parts of the system, such as the multi-agent system and the simple instant messaging system. This chapter will also discuss the methodology applied to developing the SafeChat Meeting Ontology.

Chapter 4: covers the implementation of the SafeChat system components. It articulates the implementation of the multi-agent system, the messaging application and the workings of the meeting ontology. This chapter also examines the implementation and refinement of the agent decision-making process.

Chapter 5: discusses the results presented by the component parts of the SafeChat system and is made up of three parts, the first presents the results of testing the multi-agent system, the second part examines the results of the chat application, and the third discusses the results of querying the ontology.

Chapter 6: is the final part of the thesis. It presents the conclusions as they apply to the research and development undertaken in the completion of the thesis, and associated work. This chapter will also discuss directions for future research or development of the work presented.

Appendices are at the end of the thesis. Appendix item A is a sample of ontology exported files. Appendix item B is a detailed breakdown of the responses to the survey discussed in chapter 1. Item C presents a selection of project plans associated with this research, and finally appendix item D presents a testing plan and results associated with the threat level agent component of the SafeChat multi-agent system.

Chapter 2: Literature Review

2.1 Introduction

This chapter is a critical and analytical review of relevant literature, which has been identified using specific search terms across a range of platforms. Although the search was global and included international works, only materials presented in English language were accepted. The literature will provide a background to the research process carried out across five main areas to identify the theoretical and factual background knowledge essential to the success of the SafeChat project. These areas are; motivation of study, artificial intelligence, agent technologies, ontology development and natural language processing.

A review of current literature across these specific areas will provide insight into what, if anything, has been done to address the issue already. It will also provide knowledge about what should be avoided in providing the proposed solution.

2.2 Motivation of Study

This project exists to provide children, young adults and vulnerable adults protection from placing themselves in danger and from those who would deliberately set out to do them harm. In order to fully explore the scope of this problem, there is a need to discuss aspects such as the levels and types of recorded crime, ways in which this population group uses the Internet, identify factors that contribute to users placing themselves at risk, how various stakeholder bodies are attempting to address safety issues and why an autonomous solution is needed to protect the privacy of the user.

Stakeholder Response

At a recent event in the United Arab Emirates, Baroness Shields, the Minister for Internet Safety and Security said;

> *"We are living in a remarkable time in which the intersection of technology and humanity is enriching people's lives and changing the ways we interact with each other. Technology offers endless benefits and possibilities to our children that we, as parents, never experienced. But as a result, childhood is being transformed beyond recognition.*
>
> *Technology empowers the curious, the creative, and the compassionate but equally, it empowers the criminal, the corrupt, and the coercive."*
>
> *(Shields, B.J., 2015: Online)*

The British Government formed the Child Exploitation and Online Protection Centre (CEOP) in April 2006 to respond to the growing issue of national and international threats to children from online

activities including, but not limited to; production and distribution of child abuse material, grooming and trafficking of children. CEOP has since been absorbed into the National Crime Agency and is now known as the Child Exploitation and Online Protection Command.

Writing in the CEOP Annual Review 2012- 2013, Rt. Hon. Theresa May, then Home Secretary stated that the centre works closely with a range of partners to protect children, including Microsoft, Visa and charities, such as the National Society for the Prevention of Cruelty to Children (NSPCC).

Recently the children's charity UNICEF published the updated articles contained in the United Nations convention on the Rights of the Child (2015: Online), which stipulates over forty different facets of protection for governments to work with adults to ensure the safety of the child. This project could contribute to some of those articles in providing a measure of safety for children; these are set out below in table 3.

United Nations Convention on the Rights of the Child		
Article	**Article Aim**	**Research Contribution**
11: Kidnapping and trafficking	This article states that governments must do everything possible to prevent children being removed from their country and ensure that children are allowed to return to their place of origin.	The solution presented seeks to prevent children from making meeting arrangements with strangers in an online environment. This will negate the opportunity for predators to target children for kidnap or trafficking.
13: Freedom of expression	Children should be free to express their thoughts and feelings, and to seek information as long as it is lawful.	This research seeks to offer a fully automated solution to preventing children from placing themselves at risk, which is intended to negate the need for a human to read a child's private conversations.
15: Freedom of association	Every child has the right to meet with other children and join with groups or organisations as long as this does not prevent others pursuing their rights.	This software will allow children to communicate freely and without fear of monitoring with other children and their friends. The only thing a child will be unable to do is communicate personal details or arrange meetings.
16: Right to privacy	This article says that every child has the right to a private personal, family and home life.	Once integrated, this software provides a deliberately autonomous solution in order to protect the privacy of the child.

17: Access to information from mass media	Although this article recognises a child's right to access reliable information from mass media, it goes on to stipulate that it should be in a form the child understands and places responsibility on the government to protect children from materials that could cause harm.	If parents or guardians feel that a child is safer online, they will be more open to allowing children the freedom to explore mass media for general information and education purposes.
19: Protection from all forms of violence	This article entrusts the government with the responsibility of protecting children from all forms of violence, bad treatment, abuse or neglect from their parents or guardians.	The solution presented will help protect children in an online environment autonomously, preventing children from being at risk of danger whilst also protecting their privacy.
34: Sexual exploitation	In this article the government is charged with ensuring that children are not exposed to sexual abuse or exploitation	The solution presented seeks to prevent children from making meeting arrangements with strangers in an online environment. This will negate the opportunity for predators to target children for sexual or other exploitation.
35: Abduction	This article places responsibility on the government to protect children from being abducted or sold.	The solution presented seeks to prevent children from making meeting arrangements with strangers in an online environment. This will negate the opportunity for predators to target children for abduction.

Table 3: Research Contribution to articles listed in the United Nations Convention on the Rights of the Child

Types of Recorded Crime

Online grooming refers to the use of digital technology to prey on minors in order to facilitate online or offline sexual contact (NSPCC, 2016: Online). The Cambridge Online Dictionary defines online grooming as *"the criminal activity of becoming friends with a child, especially over the internet, in order to try to persuade the child to have a sexual relationship"* (2015: Online). Offline grooming situations can happen in places like parks, parties, schools and shopping centres. Online grooming can take place in situations including, online games, social networking sites, chat rooms and communication applications. All of these scenarios offer opportunities for groomers to befriend children.

Sexual Grooming Offences – England and Wales 2004-2013

Year	2004/5	2005/6	2006/7	2007/8	2008/9	2009/10	2010/11	2011/12	2012/13
Recorded Offences	186	237	322	274	313	393	309	371	373

Table 4: Home Office crime statistics on sexual grooming

Table 4 shows the total number of grooming offences recorded over a nine-year period. In the 2011/12 period, 371 offences were recorded, but according to the Ministry of Justice, only 70 offenders were sentenced for meeting a child following sexual grooming in 2012, as seen in table 5.

Prosecutions and Convictions for Sexual Grooming Offences					
Offence	**Outcome**	**2009**	**2010**	**2011**	**2012**
Meeting a **girl** under the age of 16 following sexual grooming (Offender over 18)	Charged	33	47	49	59
	Convicted	40	60	51	63
	Sentenced	**41**	**62**	**48**	**62**
Meeting a **boy** under the age of 16 following sexual grooming (Offender over 18)	Charged	6	8	11	6
	Convicted	9	9	6	8
	Sentenced	**9**	**9**	**6**	**8**

Table 5: Ministry of Justice statistics on convictions of sexual grooming

Children on the Internet

The internet, along with mobile technology, has changed the way we all communicate. Young people can now connect and communicate directly from their computer or mobile device. Many studies have examined how children interact with strangers online and some of those findings are set out below in table 6.

Young people corresponding and meeting with strangers online and offline					
Source	**Reference /Time Period**	**Sample Size**	**Age Group**	**% of total respondents who talked to a stranger online**	**% of total respondents who had made offline contact**
UK Children Go Online (Livingstone and Bobber, 2005)	'Ever' done in lifetime. Conducted January – March 2004	1,257 (UK children who used internet at least once a week) Random location sampling	9-19 years	30% (377 individuals)	8% (101 individuals)

EU Kids Online, European Sample (Livingstone *et al.* 2011)	'Ever' done in lifetime. Conducted May – June 2010	25,142 internet-using children across 25 EU countries. Random stratified sample	9-16 years	30% (7,543 individuals)	9% (2,263 individuals)
EU Kids Online, UK Sample (Livingstone *et al.* 2010)	'Ever' done in lifetime. Conducted May – June 2010	1,032 internet using UK children. Random stratified sample	9-16 years	29% (299 individuals)	4% (41 individuals)
Harnessing Technology: The learner and their context (Enyon, 2009)	'Ever' done in lifetime. Conducted December 2008 – February 2009	941 internet using UK children. Random sample.	8, 12, 14, 17 - 19 years	27% (254 individuals)	7.6% (71 individuals)
Bridging the digital divide (Bryce, 2008)	'Ever' done in lifetime.	650 children in North West of England. Non-random sample	8-18 years	62% (403 individuals)	24% (156 individuals)
Agent-Mediated Information Exchange (MacFarlane and Holmes, 2009)	'Ever' done in lifetime. January – March 2007	437 children in Lancashire, UK. Non-random sample	11- 13 years	24% (105 individuals)	8.5% (37 individuals)

Table 6: Research into young people's online interactions with strangers

In a report for the European Commission on Grooming, Webster *et al* (2012) identify three categories of grooming victims;

- **Vulnerable individuals** – these young people seek affection or 'love' online and are responsive to approaches that are articulated in this way. These individuals tend to suffer from low self-esteem and are particularly vulnerable given that they are unlikely to discuss their relationships with anyone. It is difficult to act on this type of risk because often the victim feels that they have a genuine relationship with the groomer.
- **Risk takers** – these young people seek adventure and are over confident about their ability to handle risk. They often engage in multiple risk activities, deliberately seeking out older unknown people online and exposing themselves to risk of blackmail by exchanging or looking at pornographic images. Whittle *et al* (2013) discuss the fact

that development in adolescents is often typified by impulsive and risk taking behaviour.

- **Resilient individuals** – these young people are able to identify risk and avoid it. They will fend off unwanted contact from strangers and are likely to disclose any issues.

Child Exploitation and Online Protection Centre (2011: Online) have identified the following factors, which can make children more vulnerable to online abuse, when combined with frequent internet access:

- Personal issues; low self-esteem, confusion about their sexuality and loneliness
- Social isolation; perhaps through problems/dissatisfaction at school with limited support from their peer group or family
- Lack of parental monitoring or involvement in online activities; coupled with factors such family problems

Demographic information around victim characterisation show some clear patterns. In all aspects of research into this issue, there are many more reported instances of crime against girls than there are against boys. This may be true, but boys are less likely to report offences, so getting true statistics form a male victim perspective continues to be problematic. Whittle *et al* (2013) state that *'it is likely that the sexual abuse of boys online is grossly underreported, primarily due to negative stigma discouraging boys from reporting'.*

Teenagers are likely to be more vulnerable to online threats than very young children for a number of reasons, these include, but are not limited to (McGuire and Dowling, 2013);

- Teenagers are more likely to have access to smartphone, computers and other devices for their own use
- Older children are less likely to be supervised online
- They are more likely to be exposed to a greater range of online communication applications across multiple platforms
- Young people aged 12 and over are a likely to be curious about relationships and their own developing sexuality

Privacy and Transparency

Kemp and Moore (2006) state that *'Privacy is a difficult notion to define. Part of the problem is that privacy has been used to denote a wide number of interests including, personal information…'*

There is compelling evidence (Kemp and Moore, 2007; Mathiesen, 2013; Shmueli and Blecher-Prigat, 2011) that young people need privacy in order to develop and mature so even if parents were able to monitor every interaction this would also be damaging for the child. This then dictates a need for any protective measures implemented to preserve the autonomy and privacy of the individual wherever possible.

As shown previously most applications developed to ensure safety for children in online environments involve the need for parents to monitor closely online habits and interactions. Not only is this an invasion for the child, but time intensive for the parent, who would more than likely be uncomfortable intruding on their child's autonomy.

It is therefore safe to presume that children who value their privacy would avoid using software that provided functionality to monitor them. In a study of this issue Mathiesen (2013), 63% of young people between the ages of 12 and 19 reported that they had actively taken steps to protect the privacy of their online interactions from their parents. Mathiesen goes on to say that '*the argument here is not that children have an **absolute** right to privacy in their informal exchanges, but that failing to respect their online privacy is a serious matter and should not be done lightly, or as a general policy*'.

Young people will prefer to use the same software as their peers and this will include the more popular applications (Bothun, 2014). This leads to the conclusion that any successful solution needs to be developed in such a way that it can be applied to a range of applications, and that it should be transparent to the user, meaning it should be invisible and not interfere with the application's functionality.

2.3 Artificial Intelligence

Artificially intelligent machines can be seen as the Holy Grail for computer scientists and related researchers. In his book, On Intelligence (2004), Jeff Hawkins says;

> *"For half a century we've been bringing the full force of our species' considerable cleverness to trying to program intelligence into computers. In the process we've come up with word processors, databases, video games, the Internet, mobile phones and convincing computer animated dinosaurs. But intelligent machines still aren't anywhere in the picture."*

Definition of artificial intelligence

The Oxford English dictionary defines artificial intelligence as *"The theory and development of computer systems able to perform tasks normally requiring human intelligence, such as visual perception, speech recognition, decision-making, and translation between languages."* (2015: Online)

In their book Artificial Intelligence: A Modern Approach (2014), Russell and Norvig state that there are four categories of artificial intelligence and attribute each category with two definitions, as seen in table 7.

Definitions of Artificial Intelligence	
Thinking Humanly *"The exiting new effort to make computers think…**machines with minds**, in the full and literal sense."* (Hagueland, 1985) "[The automation of] *activities that we associate with human thinking, actions such as decision-making, problem solving, learning…"* (Bellman, 1978)	**Thinking Rationally** *"The study of mental faculties through the use of computational models"* (Charniak and McDermott, 1985) *"The study of computations that make it possible to perceive, reason and act."* (Winston, 1992)
Acting Humanly *"The art of creating machines that perform functions that require intelligence when performed by people."* (Kurzweil, 1990) *"The study of how to make computer do things at which, at the moment, people are better."* (Rich and Knight, 1991)	**Acting Rationally** *"Computational Intelligence is the study of the design of intelligent agents."* (Poole *et al.* 1985) *"AI ….is concerned with intelligent behaviour in artifacts"* (Nilsson, 1998)

Table 7: Russell and Norvig's definitions of Artificial Intelligence (2014, p.2)

The actual term 'Artificial Intelligence' is attributed to John McCarthy, who defined it in 1956 as *"the science and engineering of making intelligent machines".*

The foundations of artificial intelligence

Artificial intelligence draws on several other significant fields of study, all of which contribute concepts, perspectives and techniques, which can be considered when applying intelligence to machines or software systems.

- **Philosophy** –this discipline has long considered questions related to intelligence and how the mind works, which has in turn, has led to attempts to replicate this process in machines. Noted leaders in this field have contributed to the development of artificial intelligence in many ways. Aristotle (384-322 B.C.) created a set of laws, which he proposed, directed the rational part of the mind. His informal system of syllogisms (which means a kind of logical argument that applies deductive reasoning to help arrive at a conclusion based on at least two prepositions which are asserted or assumed to be true) for logical reasoning, which would theoretically, allow for mechanically generated conclusions to be drawn. Leonardo da Vinci (1452-1519) proposed a design for a mechanical calculator, which when constructed has been found to be functional.
- **Psychology** – It is perhaps, cognitive psychology that contributes most to the field of artificial intelligence. In this discipline, the brain is viewed as a processing device for information. Working with Frederick Bartlett at the Applied Psychology Unit in Cambridge, Kenneth Craik first specified the key properties needed by a knowledge based agent in 1943. Developments of his work led Donald Broadbent to produce *Perception and Communication* in 1958, which modelled psychological occurrences as information processing. This linked to ongoing work in the field of cognitive science that was emerging around the same time. Contributions from Noam Chomsky (1956) (*Three Models of Language*), Newell and Simon (1956) (*The Logic Theory Machine*) and George Miller (1956) (*The Magic Number Seven*) presented three papers which theorised that computer models could be used to replicate the psychology of logical thinking, memory and language.
- **Mathematics** – There are three areas, which contribute most to AI, these are logic, probability and computation.
    - In terms of logic, the work of George Boole (1815-164) that started the thinking around propositional or Boolean logic. This was then extended to include objects and relations by Gottlob Frege (1848-1925) and this created first order logic.

- This led to the development of algorithms for computing results based on first order logic. Alan Turing (1912-1954) attempted to characterise what mathematical functions were computable in order to clarify which could be solved using a computer.
- Probability provides an invaluable mathematical contribution to the study of artificial intelligence. Probability was first discussed in terms of gambling as a way to predict payoffs and has quickly become invaluable in dealing with uncertain or incomplete circumstance. Thomas Bayes (1702-1761) proposed a rule to update probability when presented with new evidence, Bayes rule contributes to modern AI solutions to uncertain reasoning environments.

- **Economics** – In 1776, a Scottish philosopher, Adam Smith became the first person to treat economics as a science with the publication of *An Enquiry onto the Nature and Causes of the Wealth of Nations*. Economics is in essence, the study of how decisions are made that lead to preferable outcomes. In mathematics, the treatment of preferred outcomes was developed by Leon Walras (1834-1910), improved by Frank Ramsey in 1931 and further enhanced by Jon von Neumann and Oscar Morgenstern in *The Theory of Games and Economic Behaviour* (1944). Preferred outcome is now more often referred to as utility. When combined, probability theory and utility theory give us decision theory. Decision theory provides a structure by which decisions can be applied to uncertain environments.
- **Computer Engineering** – In order for artificial intelligence to become a reality, it needs a platform to work on. The platform of choice is the computer. The first working computer was the one built in 1940 by Alan Turing and his team to decipher German wartime transmissions. In 1943, the team built the Colossus, which was the first multi-purpose computer. There have since been massive advances in the technology, speed and storage in computer systems. Future increases to performance rely on parallelism rather than faster clock speeds, which is why there are more processing cores in modern machines.
- **Linguistics** – In terms of artificial intelligence, the study of linguistics relates to computational linguistics or natural language processing and is considerably more complex than early studies suggested (Russell and Norvig 2014). For a computer to understand language it has to have some knowledge of the subject matter and context, rather than just the syntax.

Current developments

IBM developed the Deep Blue supercomputer, which was the first computer to beat a grand master chess champion in 1996, when it won a game against then world champion Gary Kasparov, before being beaten in the match 4 to 2. In light of this, Deep Blue was upgraded, and went on to win a match against Kasparov in May 1997, before IBM decided to retire the Deep Blue project.

Since then IBM have gone on to develop Watson, the supercomputer that attempts to answer questions posed using natural language. In order to achieve this, the computer needs to draw on a number of areas of artificial intelligence and computer science. These include; natural language processing, information retrieval, distributed computing, parallel performance techniques and machine learning (Wagle, 2013).

According to IBM, Watson can run at up to 80 petaflops. It does this by using 2,880 power 7 processors running at 3.55Ghz., it has access to 16 terabytes of random access memory (RAM) and a further 4 terabytes of clustered on disk space. All of these resources are distributed across 90 IBM Power 750 servers, which are connected by a 10 GB Ethernet network (IBM, 2016: Online).

Utilising this processing and storage power enabled Watson to win Jeopardy, the US game show, where the statements are posed in the form of answers and contestants have to respond in the form of the correct question. The contestant who comes up with the correct question first wins the round.

IBM state (2016: Online) that IBM Watson learns a new subject by having all related materials loaded into its memory, things like word documents, PDF files and web pages. Then question and answer pairing are created to help train IBM Watson on the new subject, and then any new information on that subject is automatically added to IBM Watsons 'memory' as it is published. To answer a question, IBM Watson searches all of the related documentation to find all possible answers, it then applies a scoring algorithm to rate the quality of the presented evidence and then ranks all of the possible answers based on its score.

Although IBM Watson did very well in some subject areas on Jeopardy, the computer did not have it all its own way because its human adversaries were able to steal in with some correct answers (Endgadget, 2011: Online). It was clear when watching IBM Watson operate, that if the computer had access to the information, it would be able to select the answer quickly and more often than not, do so faster than its human counterpart. However, in areas where the computer did not have the information, like any human in this position, it was not able to answer.

This is due, in part, to the decision to load all stored data into RAM memory, in order to achieve a speedy answer and provide a fairer platform for its human adversaries, "*Watson is not connected to*

the Internet or any outside source of information" (IBM, 2016: Online). If Watson had to scan disk drives full of information, it would take longer to come up with all possible answers and therefore response times would be affected.

This problem could be addressed by allowing IBM Watson to have access to the internet. The increase in time to ascertain the most likely correct answer, due to the overheads and wait times involved in searching the vast amounts of data contained across the internet would influence Watson's performance. The quest for intelligent and improved response, measured in time and quality of result, make smarter internet search engines are another significant development area in artificial intelligence (Technologies 2016: Online).

In 2001, Tim Berners-Lee, the man largely known for helping with the creation of what we now call the World Wide Web, published an article in the May edition of the Scientific American journal. The article, co-authored by James Hendler and Ora Lassila, discussed the evolution of the web. They theorised that much of the data stored on the internet would benefit from added structure, which would better define the meaning of data so that a machine could more easily interpret data through a series of defined relationships and descriptors that would add context to data, helping the machine to respond to human requests. The article says, "*For the semantic web to function, computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning.*" (Berners-Lee, *et al.* 2001).

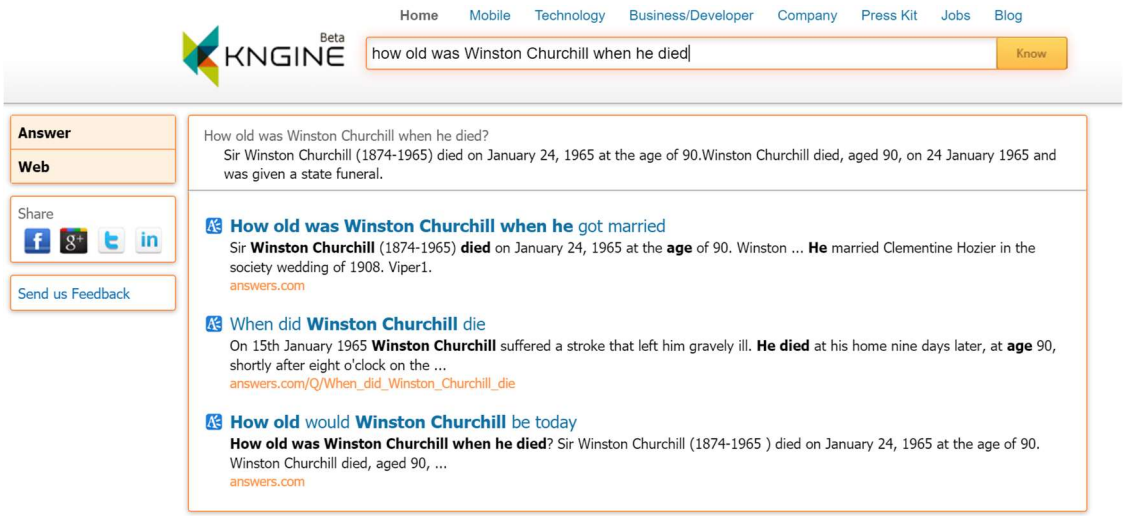


Figure 6: Semantic web search engine - Kngine

Since the release of the article, although nowhere near as quickly as Berners-Lee would have envisioned, the semantic web has been developing. The paper discussed how web agents and ontologies could work together using eXtesible Mark-up Language (XML) and Resource Description Framework (RDF) to add description data and define relationships in data that could be understood

by machines. These technologies have been used to create semantic search engines like Kngine, which will respond to a query posed in natural language and aspires to be a "*Web 3.0 Search Engine designed to provide customized meaningful search results*" (Lur, 2015: Online). Advances to popular search engines like Google also include semantic technologies, if we compare the same search in the two search engines we get similar results, as seen in figure 7, Google Search returns much more information, than Kngine (figure 6). This difference should be expected, Google is a global brand with almost unlimited resources, and Google have secured several patents related to semantic searching. Google patents include; assigning terms of interest to an entity, interactive query completion templates, knowledge graph based search system and identification of search units from within a search query, to name just a few (Bhattacharya, 2015: Online).

As the work into developing a computers understanding of natural language and the meaning and structure of stored data has progressed, the way humans interact with technology has evolved. Where once it was thought to be the norm to type away at a keyboard when using a computer, now users interact through the mediums of video and voice. Microsoft has the following aim "*Cortana aims to serve the mobile world and smart devices that will have no keyboards, mice, or even screens*" (Gallagher, 2015), for its embedded personal assistant software.

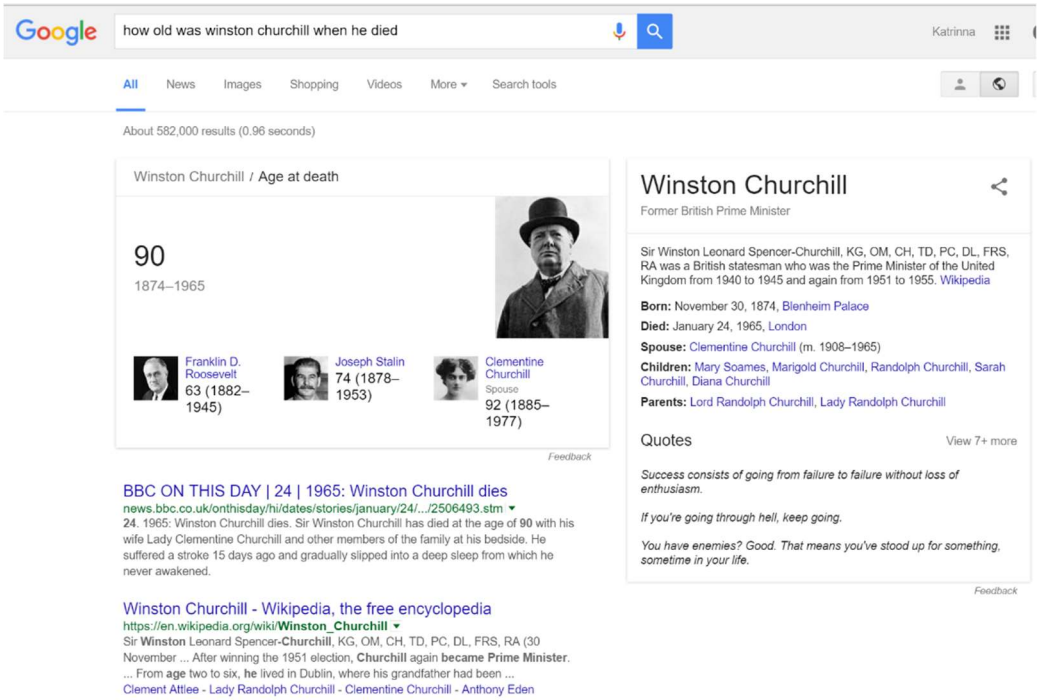


Figure 7: Google search results

Advancements in the development of touch screens mean that we are now able to manipulate software with ease. Human interactions are more immersive and varied than they once were, and intelligent systems, if they are to be successful, will have to consider these changes. This is articulated

well by James and Sebe (2007) when they said, "*The wide range of computing devices available, with differing computational power and input/output capabilities, means that the future of computing is likely to include novel ways of interaction*".

Speech or voice recognition is a development area within the computational linguistics section of the artificial intelligence field of study that does consider this interaction. According to Dictionary.com (Dictionary [a], 2016: Online), speech recognition is defined as;

> *"The computerized analysis of spoken words in order to identify the speaker, as in security systems, or to respond to voiced commands: the analysis is performed by finding patterns in the spectrum of the incoming sound and comparing them with stored patterns of elements of sound, as phones, or of complete words"*

Uses of this technology have traditionally been for security applications or for applications like Dragon Naturally Speaking, which lets users dictate speech into the computer to save time having to re-type. Other common uses included built in accessibility options in operating systems such as Microsoft Windows, who provide speech recognition as part of a suite of options and tools to *"make it easier to see, hear, and use your computer including ways to personalize your PC*" (Microsoft, 2016: Online). More recently, this technology has been embedded into applications, mobile devices and operating systems in the form of a virtual assistant.

These virtual assistants are designed to provide a single multi-faceted interface to a host of applications and services, using both text and more commonly, audio input. There are several available across a range of platforms, but perhaps the most notable being Apple's Siri and Google's Google Now, which have been developed to work on mobile device platforms. Another example is Microsoft's Cortana, which works on mobile platforms, and is embedded into the latest iteration of Microsoft's operating system, Windows 10 (Warren, 2015: Online).

These applications attempt to receive sound in the form of natural speech, translate this into instructions that the machine or applications understand, and then process the request into what it thinks is the desired outcome. This could involve looking up a phone number, searching for a web page or finding the nearest supermarket. They are most effective when they are linked to complex natural language processing (NLP) systems and the semantic web services used by more sophisticated search engines. Automated personal assistants are attempting to compute what is said by the user, what that actually means, and therefore what the user wants to happen consequently (Woodford, 2015: Online). These applications are an example tools that utilise several different artificial intelligence

Ethical Considerations

Recently, there have been a flurry of news articles (Aguirre et al., 2015: Online; Zolfagharifard, 2016: Online), online discussions and debate around the dangers of artificial intelligence and intelligent machines. Some the world's leading thinkers, such as Stephen Hawking, Elon Musk, Mark Zuckerberg and Stuart Russell, have raised concerns over the direction of some intelligent systems, whilst indicating support for the growing insistence that these technologies are used to improve society and for the betterment of mankind, Future of Life Institute (2015: Online).

Many people accept that there are many ways intelligent systems and machines can help society, but there are applications that give cause for concern. Many worry that robots will take jobs away from people, and that this will increase wealth divide. Stephen Hawking indicated that this concern has merit in an online question and answer session when he responded;

> *"If machines produce everything we need, the outcome will depend on how things are distributed. Everyone can enjoy a life of luxurious leisure if the machine-produced wealth is shared, or most people can end up miserably poor if the machine-owners successfully lobby against wealth redistribution. So far, the trend seems to be toward the second option, with technology driving ever-increasing inequality."*
>
> Stephen Hawking (2015: Online)

Martine Rothblatt (Terms, 2016: Online), biotech and satellite entrepreneur, predicted that stored personal data could be used to create intelligent digital clones, which could be used, or programmed to act autonomously. Christof Koch, chief scientific officer of the Allen Institute for Brain Science, warned that although he believed intelligent software could never be conscious, it could still harm us if not designed correctly (Senior, 2014: Online).

There have been numerous protests, some making headline news, around vigorous opposition to automated weapon systems. In a report (Watch, 2016: Online) delivered to the United Nations, the organisation Human Rights Watch articulated the dangers surrounding the use of such systems and offered guidance to limit their impact. The recommendations were to;

- *Prohibit the development, production, and use of fully autonomous weapons through an international legally binding instrument.*
- *Adopt national laws and policies that prohibit the development, production, and use of fully autonomous weapons.*

Human Rights Watch (2015: Online)

The clear consensus (Kablan, 2016: Online; Gray, 2016: Online) is that the focus should be on intelligent systems and machines that will help humankind and better society. This is evidenced by the fact that the open letter, "Research Priorities for Robust and Beneficial Artificial Intelligence" (Future of Life Institute, 2016: Online) invited readers to sign up to an agreement to the priorities set out in the letter. The agreement has had 8,600 signatures so far, many of those from leading stakeholders in the field of artificial intelligence research and the industries that apply intelligent systems. As with all areas of research, ethical considerations should be carefully considered into all of the possible uses of any outputs or system developments.

2.4 Software Agent Technology

In the preface to his book, An Introduction to Multi-Agent Systems (2009), Michael Wooldridge says that;

> *"Multiagent systems are systems composed of multiple interacting computing elements, known as agents. Agents are computer systems with two important capabilities. First, they are at least to some extent capable of autonomous action – of deciding for themselves… Second they are capable of interacting with other agents – not simply by exchanging data, but by engaging in analogues of the kind of social activity that we all engage in every day of our lives…"*

When initial research dictated that decision-making and autonomy were necessary requirements of this system, the solution gravitated toward a software agent or multi-agent based development. Software agents lend themselves well to complex problems where decision-making needs to be performed on an ad hoc basis without the direct involvement of the user or system (Russell and Norvig, 2014; Wooldridge, 2009; Macfarlane and Holmes, 2009). This ability to function autonomously is precisely what is needed in the SafeChat system.

Software Agent Definition

Google defines an agent as:

> *"a person who acts on behalf of another, in particular:*
> ***synonyms:*** *representative, negotiator, business manager, emissary, envoy, factor, go-between, proxy, surrogate, trustee, liaison, broker, delegate, spokesperson, spokesman, spokeswoman, frontman, mouthpiece;"*
>
> (Google [c], 2016: Online)

Google is clearly talking about a person in this instance, but most of the definition would still apply to a software agent. After all, the software agent is specifically designed to act on behalf of the system or the user, it represents them, and if designed properly, will act on their behalf to accomplish something.

Russell and Norvig classify a software agent as "*anything that can be viewed as perceiving its **environment** through **sensors** and acting upon that environment through **actuators***" (Russell and Norvig, 2014 p. 35). Whilst Michael Wooldridge says, *"An agent is a computer system that is **situated** in some **environment**, and that is capable of **autonomous action** in this environment in order to meet its delegated **objectives**"* (Wooldridge, 2009, p. 21).

Both of these definitions about a software agent are correct, but Wooldridge defines an agent more thoroughly for the purposes of the work presented in this thesis.

Software Agent Characteristics and Types

In 1995, the Knowledge Engineering Review published an article by Michael Wooldridge and Nicholas Jennings titled "Intelligent agents: theory and practice". In this article, they discuss notions of agency and apply characteristics in terms of weak and strong agents that are used to help distinguish agent types today, as seen in table 8.

Software Agent Characteristics		
Weak notions of agency	Strong notions of agency	Other notions of agency
Autonomy: agents operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state; **Pro-activeness:** agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviour by taking the initiative. **Reactivity:** agents perceive their environment and respond to it in timely fashion to changes that occur in it.	In addition to having the properties identified as weak notions of agency, a strong agent is either conceptualised or implemented using concepts that are applied to human beings. For example, it is quite common in artificial intelligence to characterise an agent using *mentalistic* notions, such as **knowledge**, **belief**, **intention**, and **obligation**.	**Mobility:** the ability of an agent to move around a network **Veracity:** agent will not knowingly communicate false information **Benevolence:** agents do not have conflicting goals and always try to do what is asked of it. **Rationality:** an agent will act in order to achieve its goals and will not act in such a way as to prevent its goals being achieved"

Social Ability: agents interact with other agents (and possibly humans) via some kind of agent-communication language.		

Table 8: Wooldridge and Jennings's notions of agency (Wooldridge and Jennings, 1995)

Agents will be made up of some or all of the characteristics listed, and depending on their function, can be categorised into types (Mahmoud, Unknown: Online), as seen in table 9.

Agent Types	
Type	**Description**
Rational Agent	• This agent will do the right thing – conceptually, e.g. if its job is to fill in a database it will be successful if every entry in a table is filled out correctly.
Reactive Agent	• This is also known as a reflex agent and will normally use a production system to determine what action to carry out based on current inputs • **Example:** spam mail filter • Does not perform well when the environment changes. • Does not deal well with unexpected events • Is of the form if *event* then *action*

Goal-based Agent	• These agents are more complex than reactive agents are.
	• They use search and/or planning techniques
	• No regard to technique or efficiency
	• **Example uses**:
	  ◦ Find pages on the Internet for A.I.
	  ◦ Design agent to load Web pages
	  ◦ Identify the goal
	  ◦ Report results

Utility-Based Agent	• These agents attempt to achieve some specified goal, usually using search or planning methods. • An agent, for example, might have the goal of finding interesting web pages. • The agent would have various actions it could perform such as fetching web pages and examining them • This agent will attempt to maximize efficiency
Interface Agent	• An interface agent acts as a personal assistant • **Example**: a tool used to help a user learn to use a new software package

	• Interface agents observe a user's behavior and make recommendations accordingly
Mobile Agent	• Mobile agents can move from one location to another • This can mean physical locations (for robots) or network locations • A computer virus is a kind of mobile agent. Viruses are usually autonomous but not intelligent • Mobile agents are efficient, but can pose a severe security risk • Mobile agents can be combined to produce a distributed computing architecture
Information Agent	• Also known as internet agents or bots • Information agents gather information from the Internet (or other source of data) • Can be static or mobile • Can be taught by example: "find me more information like this" • Information agents need to be sophisticated to deal with the "dirty" nature of much of the data on the Internet
Learning Agent	• Agents learn using mechanisms such as neural networks and genetic algorithms • Learning enables an agent to solve problems it has not previously faced, and to learn from past experience • Multi-agent learning can produce much more impressive results • Such learning can be centralized or decentralized – agents learn individually or contribute to the learning of the whole group
Robotic Agent	• Unlike software agents, robotic agents exist in the real world • Robots operate in a stochastic, inaccessible environment, and must also be able to deal with large numbers of other agents (such as humans) and other complicating factors • It is important for robotic agents to deal with change and uncertainty well

Table 9: Types of software agents (Coppin, 2016: Online)

Agent Task Environments

Depending on the system and its function, the environment an agent or agent system will have to work in can be categorised into general types (Wooldridge, 2009). Russell and Norvig (2014) articulate the following properties for the classification of agent environments;

Accessible versus inaccessible – In the accessible environment, an agent can monitor every aspect of the environment and will always have access to all of the information it needs to be able to assess the state of the environment. The inaccessible environment is more like the real world, where it is not possible to have all of this information at hand.

Deterministic versus non-deterministic – In a deterministic environment there can be a certainty on the effect of a selected action, it will achieve a set outcome. In a non-deterministic environment, the effect of an action is unknown.

Static versus dynamic – In a static environment it can be assumed that it will only change as a result of any impact of actions taken by the agent. Only if the agent's actions influence the environment will it change. In a dynamic environment, there are other factors, which could affect change, therefore the environment is more fluid and the agent cannot be certain that the environment will stay the same regardless of any actions it takes. The real world is a dynamic environment. If the environment itself is static and will not change over time, but the agent's parameters or performance is likely to change, this is classified as a **semi dynamic** environment

Discrete versus continuous – A discrete environment is one where there are a finite number of actions and rules available to the agent. In contrast, a continuous environment will have no such limitations.

Episodic versus sequential – In the episodic environment the agent works in an atomic fashion, in each round of execution the agent will receive a percept and then perform a given action. It is important to note that in each round of actions there is no reliance on actions taken in previous rounds. In a sequential environment, the current action could have an impact on all future decisions.

Single agent versus multi-agent – Although the difference between these two environments may seem obvious, it is worth noting that even a single agent may have to operate alongside other entities that it will have to treat as though they are other agents. This means that a single agent can be working in a multi-agent environment.

Known versus unknown – This distinction refers to the agents (or program designer's) knowledge or perception of the environment. In a known environment, the outcomes for actions taken are understood. Conversely, in an unknown environment the outcomes of actions taken will be uncertain.

Table 10 gives some sample systems or tasks and looks at the environments they would inhabit. From a programming and development point of view, the most complex environment to work in would be an inaccessible, non-deterministic, dynamic, continuous, unknown and multi-agent environment.

Environment Examples						
	Accessible	**Deterministic**	**Static**	**Discrete**	**Episodic**	**Agent(s)**
Crossword	Fully	Deterministic	Static	Discrete	Sequential	Single
Timed chess game	Fully	Deterministic	Static	Discrete	Sequential	Multi
Poker	Partially	Non-deterministic	Static	Discrete	Sequential	Multi
Taxi driving	Partially	Non-deterministic	Dynamic	Continuous	Sequential	Multi

Medical diagnosis	Partially	Non-deterministic	Dynamic	Continuous	Sequential	Single
Image analysis	Fully	Deterministic	Semi	Continuous	Episodic	Single
Part picking robot	Partially	Non-deterministic	Dynamic	Continuous	Episodic	Single
Interactive tutor	Partially	Non-deterministic	Dynamic	Discrete	Sequential	Multi

Table 10: Examples of task environments

Agent Architectures

Agent systems can be developed using different kinds of architectures, these architectures will provide the foundational mechanisms needed to support agents to behave appropriately in changing environments (Bellifemine *et al.* 2007).

Cognitive agent architecture will have some artificial intelligence and decision theory mechanisms to allow agents to behave in what is hoped to be a rational manner. Agents in this type of architecture with have goals that they need to achieve. The agents could have several states to choose from and each state will have a different level of utility, a rational agent will choose the state that yields the highest utility.

First Order Predicate Logic (FOPL) architecture uses a symbolic representation of knowledge, and uses inference in the logic deduction and theorem to help decide on a course of action (Russell and Norvig, 2014). The advantages of using this architecture are that it is simple and elegant in creation and has specifications that can be executed. It does however have some drawbacks, including a difficulty in representing changes over time, and because the decision making process works on deduction, it can take longer than necessary in some cases to reach the goal.

Beliefs Desires and Intentions (BDI) architecture is based on the theory of rational human actions and is arguable the most popular architecture in use (Rao and Georgeff, 1995). Beliefs are what the agent knows about the world or environment it lives in, desires are the goals that the agent wants to achieve, and intentions are desires that the agent is committed to achieve. BDI performs means end analysis and it weighs the competing alternatives in order to achieve the highest utility. Intentions play a critical role in BDI as they drive the means end analysis, they are persistent and influence future practical reasoning.

In a **reactive agent** architecture, an agent has Task Accomplishing Behaviours (TAB), a competence module represents each of these TAB's. Competence modules operate in parallel and are often responsible for a clearly defined simplified task. Reactive architecture has a hierarchical approach and lower level modules have a higher priority and can block higher-level module operations. Lower level

modules tend to be responsible for basic tasks, while higher-level modules are responsible for behaviour that is more complex and incorporate a subset of the tasks of the lower level modules.

Layered Agent architectures combine reactive and proactive behaviours (Bellifemine *et al.* 2007, page 5), they can be layered horizontally and have horizontal I/O flow, or vertically layered with vertical I/O flow. This architecture has a control subsystem, which contains a set of control rules.

Agent Communication

An agent's communication language is what provides it with the capability to exchange knowledge and information with other agents. A communication language can handle rules, actions and propositions and any semantics attached to them. This language describes a desired state in a declarative language instead of a procedure or method. This enables agents to have conversations rather than just exchange messages.

The type of messaging agents use in communication languages is known as 'Speech Acts', which are inspired by speech act theory. In this theory (Witek, 2015), the language used tries to be as close to the natural language used by people to achieve goals and desires. We can identify five kinds of speech acts; Representatives or Assertives, Directives, Comissives, Expressives and Declaritives. These communication languages need to be able to translate syntactically between languages, preserve semantic content between applications and be able to communicate complexity.

```
(KQML-performative
                    : sender      <word>
                    : receiver    <word>
                    : language    <word>
                    : ontology    <word>
                    : content     <expression>
                        …)
```

Figure 8: KQML performative message structure

Knowledge Query and Manipulation Language (KQML) is a protocol used in the exchange of information and knowledge among agents and applications (Finin *et al.* 1994). The structure of a KQML message can be seen in figure 8. A description of the parameters of a KQML message can be seen in table 11.

Parameter	Meaning
: sender	Sender of the message
: receiver	Intended recipient of the message
: language	Language in which the message is expressed

: ontology	Vocabulary of the "word" in the message
: content	Content of the message

Table 11: KQML parameters

Agents who communicate using KQML have a client server relationship and can communicate in either a synchronous or an asynchronous manner (Finin *et al*. 1994). For a synchronous communication, a sending agent waits from a reply, whilst in an asynchronous communication, the sending agent continues with its reasoning or actions, which would then be interrupted when replies arrive at a later point in time.

Agent Communication Language (ACL) was developed by the Foundation for Intelligent Physical Agents (FIPA), as seen in figure 9. It is quite similar to KQML but ACL has only two basic performatives **inform** and **request**; all other performatives are macro definitions defined in terms of these. The *inform* and *request* performatives, can be defined in two parts, *precondition* and *rational effect*. ACL does not provide facilitator services.

```
(inform
    :sender     agent1
    :receiver   agent2
    :content    (price good200 150)
    :language   sl
    :ontology   hpl-auction
)
```

Figure 9: FIPA agent communication language message structure

> *"FIPA is an IEEE Computer Society standards organisation that promotes agent-based technology and the interoperability of its standards with other technologies."*

(Dale, 2013: Online)

Agent Negotiation

Negotiation between the agents in multi-agent systems may involve exchange of information, the relaxation of original goals and mutual concession. This negotiation needs to be governed by a mechanism or a protocol in order to define the rules of engagement and to set the rules by which agents can come to agreements. These protocols, in order to achieve maximum utility, should have preferred outcomes, as seen in table 12.

Negotiation Protocols	
Preferred Outcome	**Benefit**

Guaranteed success	By agreeing to this action, the agent will definitely achieve its goals
Individual rationality	This is the most sensible course of action for the agent
Stability	This course of action will not compromise the success of the system or goals
Distribution	This course of action will delegate the task, or part of the task to ensure achievement of goals
Simplicity	This is the easiest/shortest/cheapest/most efficient way to achieve the goal
Maximum social welfare	This will achieve the goal with the best possible outcome for all parties

Table 12: Negotiation protocols and their benefits

Agent Decision Making

Agent and multi-agent systems provide autonomy by making decisions on behalf of the user (Wooldridge, 2009; Das, 2008). These decisions can be dependent on many variables and often involve reacting to uncertain events. Das articulates that, *"The concept of **epistemic states** is often used to represent an actual or possible cognitive state that drives the human-like behaviour of an agent."* Das goes on to propose three common models of epistemic states an agent can inhabit (Das, 2008). These are:

1. A **propositional model**, where the epistemic state is represented by propositions that are accepted by the agent. Propositions, in this case, are expressed in an object language using sentences.
2. A **probabilistic model**, where the epistemic state is defined by a measure of probability, which is derived from a collection of variable values. In this case, the measure of probability provides the agent with a degree of belief about the variable values contained in the epistemic state.
3. A **possible world model**, where the epistemic state is represented by possible worlds that includes the world that the agent is in, as well as worlds which are compatible with the agents pre-defined beliefs and knowledge. In this epistemic state, the worlds are made up of propositions that the agent considers to be true to its epistemological worldview.

Regardless of the agent's epistemic state, the agent needs to have access to some kind of knowledge domain in order to inform any decision making. An agent may not always be able to fully automate decision making, this is dependent on the type of system and knowledge domain it has access to. For

example, a medical diagnosis system, may present a range of outcomes which all match patient symptoms, from which a human doctor would then make the final decision.

Utility drives decision making in agent systems, utility in this case is the measurement of payoff or success secured by the decision. Wooldridge (2009, p225), compares utility to money, meaning that the benefit for the agent making decision can be compared to the amount of money which will be earned. He goes on to clarify that it is not that simple in reality. Often the agent is dealing with events or situations which are varied and where the outcome is measured against priorities assigned to the agent, so success is measured against how much the agent or agent system benefits from the final decision.

For example, we are seeing the introduction of driverless vehicles, which are driven by automated systems. These vehicles can plan a route based on what is known about the traffic at a given point in time, but if something happens on route, that plan would have to change in order for the desired outcome to be achieved. In fact, certain events would make the outcome unachievable (for example, a bridge collapsing or an accident blocking a key road in the selected route).

Regardless of the agent's epistemic state, decision making often has a degree of uncertainty attached. In terms of diagnosis, for medicine, vehicle faults, network errors etc., any decision making will involve a degree of uncertainty. When we try to apply propositional logic rules to diagnosis we see how the logic breaks down (Russell and Norvig, 2014). Consider the following rule;

Car won't start => flat battery

The problem with this simple rule is that it is not guaranteed to be true, there are several reasons why the car may not be starting; no fuel, faulty starter motor etc. Consider the refined rule;

Flat battery => car won't start

This rule is true, if a car has a flat battery, it cannot start. The problem with this rule is that changing the battery is not guaranteed to make the car start, there could be other problems with the car that would prevent it from starting. At best, the agent can only have a partial belief in these rules, therefore a mechanism for dealing for partial belief is needed. Russell and Norvig (2014) assert that the best tool for this problem is **probability theory**. A logical agent treats a rule as being true or false, whereas a probabilistic agent applies a numerical value of belief to each rule, 0 meaning the rule is absolutely false, or 1 meaning the rule is absolutely true.

When applied to the general rule, probability theory gives a summary of the uncertainty that emerges from a given rule. There is no certainty that replacing the flat battery of the car will make it start. But

using statistical analysis and other experiential information, the conclusion could be drawn that in 80% (or 0.8 in probability terms) of cases where a car with a flat battery had a new battery fitted the car then started. This gives the agent more information, and this along with other information (for example, the car's fuel status) can be used to evaluate and inform a decision in an uncertain environment. If the car battery is flat and the car has no fuel the probability that replacing the battery will enable the car to start may be reduced to 0% (0 in probability terms), but we can say that the probability of the car needing a new battery before it can start is 100% (1 in probability terms). These statements are not contradictory as each articulates a belief which is applied to a different knowledge state.

If we apply this to a more complex problem, like planning to get to an appointment on time using our automated vehicle, the vehicle could come up with one plan that would give us a 97% chance of getting to our appointment on time, but this plan may not be the rational choice, other plans may give higher probabilities of success, but may involve setting off a lot earlier and having to wait at the appointment area for a long time. In order to be able to differentiate between these plans the agent needs to have knowledge of **preferences** between the different possible **outcomes** of a plan. These outcomes must be completely specified in order to allow the agent to select the best decision. For our example this will involve, arriving on time and not having to wait at the appointment. **Utility theory** can be used to represent and reason with preferences, to enable agents to select a plan which yields the highest utility based on user preferences as they apply to a given scenario.

Probabilities, linked with preferences allocated based on utility give the foundation for decision making. As articulated by Russell and Norvig (2014, p491);

Decision theory => probability theory + utility theory

This means that a rational agent will decide on an action (or plan of actions) that yields the greatest possibility of success based on articulated user preferences.

Agent Frameworks

Agent frameworks or toolkits help the developer create robust agents, complete with the tools, attributes, features and rules needed to meet the challenges of the complex problem that agent faces (Serenko and Detlor, 2002). Whilst there is no universal definition for agent frameworks, they have been described as "An integrated tool suite for constructing intelligent software agents" (AgentBuilder, 2000), and alternatively "A software framework to make easy the development of agent applications…for interoperable multi-agent systems" (Bellifemine et al., 2000).

There are different categories of frameworks available, these are:

- Mobile Agent Toolkits
- Multi-agent Toolkits
- General Purpose Toolkits
- Internet Agent Toolkits

The SafeChat system is a multi-agent system, therefore when researching frameworks, multi- agent frameworks became the research focus. These are the most complex frameworks because, in most cases, an agent cannot solve a complex problem alone and needs co-operation with other agents to exchange data and information with or delegate tasks to. Table 13 details some common agent development frameworks.

Java Development Framework (**JADE**) – This framework was developed by researchers at the University of Parma. It provides a set of tools to support development and de-bugging of multi-agent systems. The JADE environment supports the use of ontology and is fully FIPA compliant. It is maintained and has a suite of add-ons, which help with cross platform and ontology integration support.

Java Agent Template Lite (**JATLite**) – This framework was developed by researchers at Stanford University and is used to build agents that are able to communicate effectively over the Internet through the Agent Message Route Facilitator, via a registration process.

ZEUS - The ZEUS toolkit was developed by British Telecom and consists of three main components: an agent component library, an agent building tool, and a suite of utility agents comprising name server, facilitator, and visualisation agent.

MadKit – MadKit was developed at the University of Montpellier. It is a multi-agent toolkit, which builds upon the AGR (Agent/Group/Role) organisational model. It allows high heterogeneity in agent architectures and communication languages, and various customisations.

Name	Open Source	Maintained	FIPA Compliant
MadKit	Yes	Yes	No
Zeus	Yes	No	No
JADE	**Yes**	**Yes**	**Yes**
JATLite	Yes	No	No

Table 13: Comparison of agent frameworks

Example Agent System Developments

When considering weather, a problem is complex enough to warrant a multi-agent system approach much has to be considered. Ayllet *et al.,* (1998) state that *"a multi-agent approach would be sensible for problems that are inherently (physically or geographically) distributed where independent processes can be clearly distinguished".* Wooldridge points towards increasing trends towards intelligence and delegation asserting the demand for more complexity in software solutions (Wooldridge, 2009). Most experts agree that agent systems are best suited to problems where there is a need to address complexity, planning, autonomy and decision-making.

Examining existing multi-agent systems gives an insight into the breadth of application areas and the levels of complexity and challenge being dealt with by system developers. It also, if the developers are kind, gives information on the pitfalls and shortcuts encountered throughout the development process.

Genghis was developed to act as a carpooling system to help alleviate the problems of congestion on the roads and in turn reduce harmful emissions to help the environment (Kothari, 2004).

In this system, a *UserAgent* represents a human user and interacts with the other agents in the system. ProxyAgents handle HTTP requests and act as middleware between the JADE containers and the web portal application. A *JourneyRoundupAgent* flags journeys that are underway or that cannot be joined. A *JourneyNotifyAgent* monitors needed and active journeys and reports any available matches to the *UserAgent*.

The idea is that the user logs on via a web portal and inputs a desired journey and time, the system then scans journeys that are active at that time and where possible reports matches back to the user. In the case of more than one match, results are ranked in accordance with a pre-set ranking criterion.

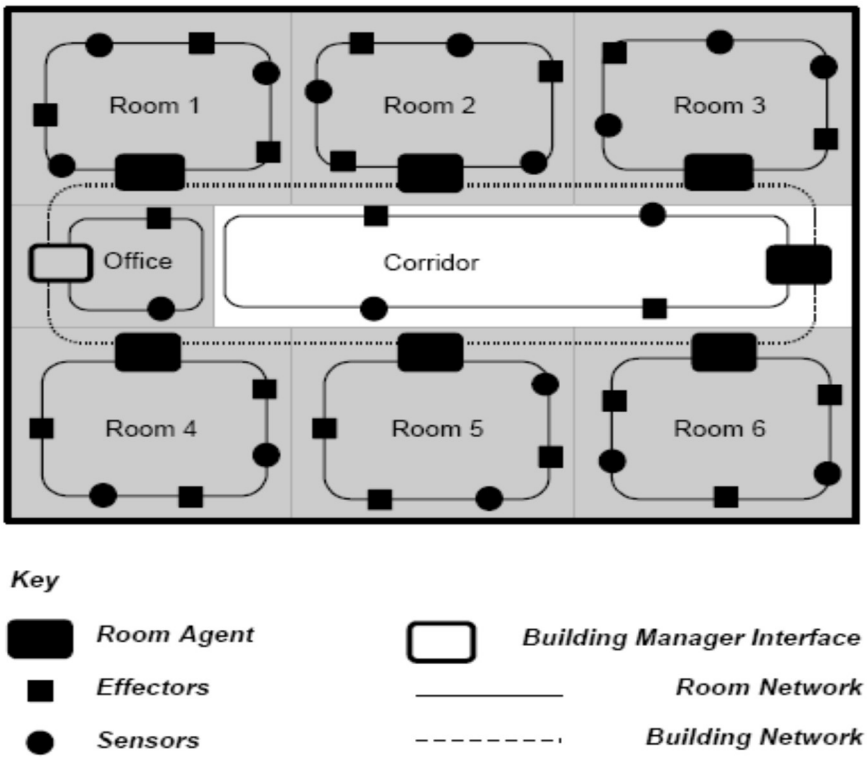


Figure 10: The Essex IB Model system diagram

The **Essex IB Model,** seen in figure 10**,** is a multi-agent intelligent building application (Callaghan *et al.* 2000). In this application software agents control the heating, lighting and security of a building autonomously, and learn from user input. Each room or area, such as a corridor, is assigned an agent. It is logical to assume that control and learning behaviour are different depending on the room we are in, for example, we could want our recreation room to be comfortably warm, while we might want our kitchen and bedrooms to be cooler.

In essence, the system is a collection of parallel, distributed agents, all monitoring their own area and responding to whatever is happening in that area. This is more effective because the agents learn to respond to the needs of the users of that room, rather than treating the whole building the same. This system would be ideal for something like a nursing home, where occupants have their own living areas. Agent communication is in effect; in cases of emergencies, an alarm can be spread quickly to each different area of the building.

Calvin is a multi-agent personal information retrieval system (Bauer T., Leak D.B., 2002). Each user is assigned a personal profile, which is developed by analysis agents. These analysis agents then generate descriptions of the current requirements of that user. Retrieval agents use these descriptions to query standard search engines. Results are then reported to a user interface agent to be presented back to the user.

Users interact with Calvin via a web interface; this interface also acts as an agent, by recording any web pages the user chooses to access and passing this information to the user analysis agents. Calvin has two retrieval agents, GoogleBot and AltaBot. These agents interact with Google and Alta-Vista and refine searches by presenting queries with the appropriate syntax and keywords.

2.5 Ontology Development

Ontology based solutions are being employed with great effect in the Semantic Web, where intelligent agents are being used to filter existing web pages to return only information that is relevant, or at least more relevant, to user queries (MacFarlane and Holmes, 2009). Ontology are being used in a wide variety of artificial intelligence research projects, according to (Davies *et al.* 2002 p.4) the reason they are so attractive in the field of artificial intelligence is because they are said to promise *"… A shared common understanding of a domain that can be communicated between people and application systems".*

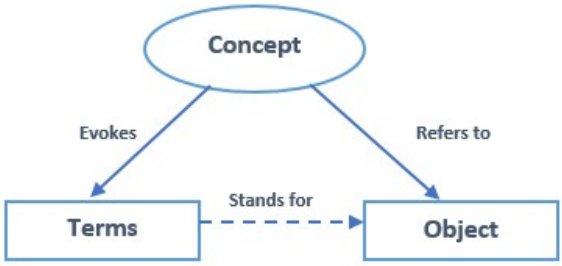


Figure 11: The meaning triangle in terms of ontology

In humans, the theory of communication is expressed generally, in communication context using the triangle of meaning (Ogden and Richards, 1927). They theorised that the triangle represented the three relationships between concepts (thoughts), terms (words) and objects (things). Figure 11 shows these relationships, expresses an indirect link between terms, and objects (Sawsaa, 2013). If a human is presented with a term he or she has no knowledge of, they will not understand the term and therefore cannot link to the object.

For example, if you ask a human to describe a C250, most will not know what is being asked of them, but if you ask them to describe a Mercedes, they will likely describe a car. The *term* Mercedes stands for the *object* car in this example. A computer is always in the position that it does not know the meaning of the term or its relationship to the object. Ontology development seeks to rectify this and provide the knowledge needed to appreciate the concept, terms and objects associated with a body of knowledge.

Definition of ontology

Google uses its improved semantic search techniques to define an ontology as "*a formal naming and definition of the types, properties, and interrelationships of the entities that really or fundamentally exist for a particular domain of discourse. It is thus a practical application of philosophical ontology, with a taxonomy*." when applied to computer science (Google, 2016: Online).

(Calero *et al.* 2010) conclude on page 10 that, in terms of a software engineer, an ontology *"is a formal, explicit specification of a shared conceptualisation. Conceptualisation refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon".*

According to Tom Gruber, an artificial intelligence specialist at Stanford University, an ontology is *"the specification of conceptualisations, used to help programs and humans share knowledge."* (Larose and Cruse, 2005: Online)

Overview of Ontologies

Ontology has its roots in early philosophy, where it refers to the study of existence, or being, and the organisation of reality (Jakus *et al.* 2013, page 29). In terms of computer science, it is used in the artificial intelligence discipline to represent the real world in a way that can be understood by a computer.

Ontologies are a mechanism to allow the organisation of related concepts, an example of this is illustrated in figure 12. By grouping concepts in this way and then applying a property or definition to the relationship between concepts, a clearer understanding of the overall entity is achieved.

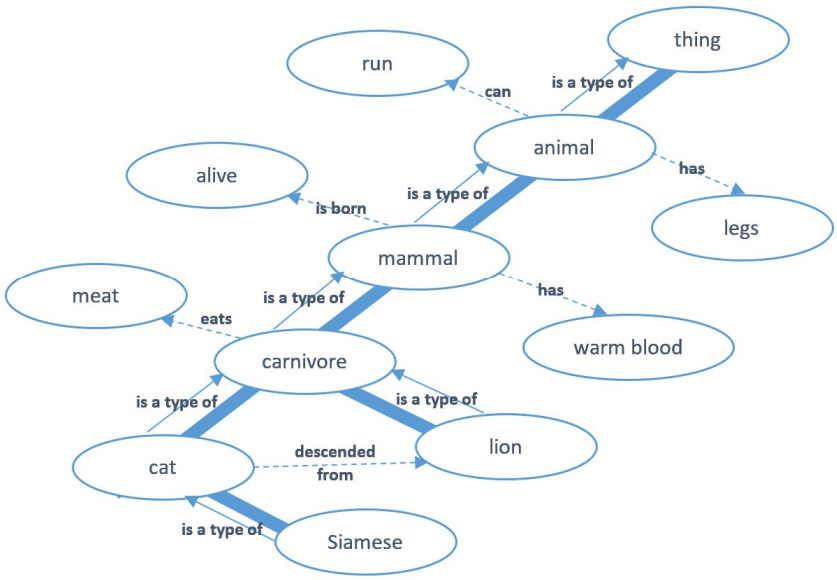


Figure 12: An example of a taxonomically structured ontology

The value of ontological development lies in the fact that they are often created to cover a whole domain of knowledge. If an ontology is modelled carefully, it can be re-used (Studer *et al*. 1998; Jakus *et al*. 2013; Corcho *et al*. 2010). That is not to say that ontologies cannot be developed to fit a specific task, but that where possible, these resources should be open and shared. The primary aspects of ontologies are (Studer *et al*. 1998; Jakus *et al*. 2013);

- Ontologies are *conceptualisations*. They are abstract models, which contain the relevant concepts needed to describe the real word, or chosen part thereof. This means that ontologies can be said to provide a surrogate reality in terms that can be processed by a machine.
- Ontologies should be *explicit*, meaning that the ideas and relationships and other components depicted in a given ontology are explicitly defined.
- Ontologies need be *formal*, in order to be used correctly by computer systems the use of natural language is not appropriate, due to the ambiguous and inconsistent way that natural language is expressed.
- Ontologies should be *shared*, especially when they contain accepted proven knowledge for a given domain of thought. If ontologies are to be shared, some consideration must be given to enabling revision of the knowledge contained in the ontology, in light of any newly accepted evidence or discovery.

Corcho *et al*. (2010) and Gruber (1993) suggest the following design principles for ontology engineering, as an objective set of guidelines which should be employed in the creation and evaluation of ontology development;

Clarity – the ontology should clearly articulate the intended meaning of defined terms. Definitions should be objectives and can be stated on formal axioms. Where possible they should be complete, but where it is necessary, partial definitions are acceptable. Natural language should be used to document definitions.

Employ **minimal encoding bias** – this means that the ontology conceptualisation should be specified at the knowledge level and not be dependent on a specific or personalised symbol level encoding. Adhering to this guideline helps with the sharing and interoperability of a given ontology.

Extendibility – this means that the ontology should able to define new properties or terms for special use based on the existing vocabulary used in the ontology. This should allow for additions without the need to revise existing definitions.

Coherence – this means an ontology should be easily understood, and should only allow inferences that are consistent with its definitions. If, under interrogation, a concept can be developed from the ontology that contradicts its stated definitions, the ontology is said to be incoherent.

Encourage **minimal ontological commitment** – Given that ontological commitment relies on the consistent use of vocabulary, it may be prudent to minimise ontological commitment by specifying the basic theory. The ontology should then only define terms that are essential to the communication of knowledge, which is consistent with that theory.

Other general software development techniques should be applied to ontology development, for example clear naming conventions, to help those working with the ontology to understand its content. It is important to note that, although an ontology is produced as a piece on software, it is not a program and will not run as an application.

Ontology Modelling Approaches

There are three approaches to designing and developing software, all of which are well established in the field (Sawsaa, 2013). When deciding on the appropriate one for ontology development, it is important to weigh the advantages and disadvantages of each before selecting the one that is best suited to the task.

Top-down – this approach is also known as stepwise design. Essentially the system or problem is broken down in an effort to gain insight into the sub-systems that it contains. Using this approach, an overview of the system is created, specifying but not detailing the lower levels of the system. For example, in top down design, you would examine the concept first, and then identify entities within that concept, illustrated in figure 13. This approach is useful for most projects; it provides greater control of the design encourages object-oriented programming with encapsulation and has proven to be successful for functional programming. It is challenging to ensure that the initial design will deliver an adequate stem using this approach, testing has to wait until a large part of the system is complete and it is more likely that redundancies will appear in the system (Volin, 2010: Online).

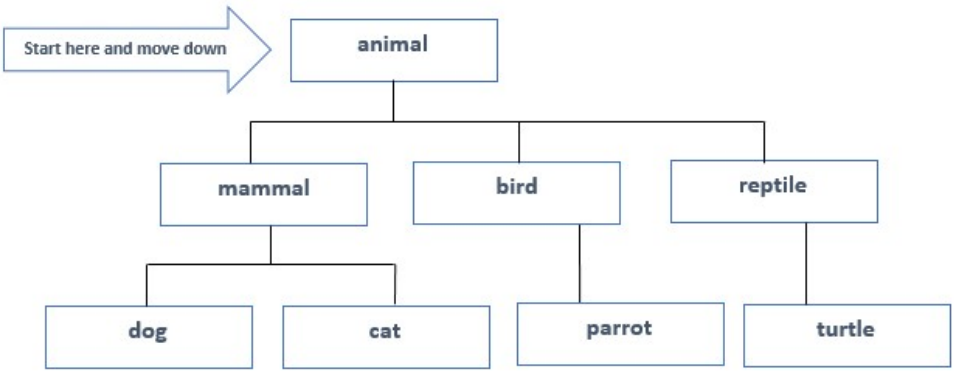


Figure 13: An example of the top down modelling approach

Bottom-up – this is the opposite of the top down approach, here the base parts of the system are defined and linked together to compose the total system, as seen in figure 14. This method has some advantages, testing can take place earlier in the design process and it leads to reusable code. It may prove more difficult to link things together in this approach and the resulting system could be unorganised and difficult to navigate. Finally, and most importantly in the case of ontology design, this approach may lead to ambiguities of the relationships between elements in the overarching design (Sawsaa, 2013; Volin, 2010: Online).

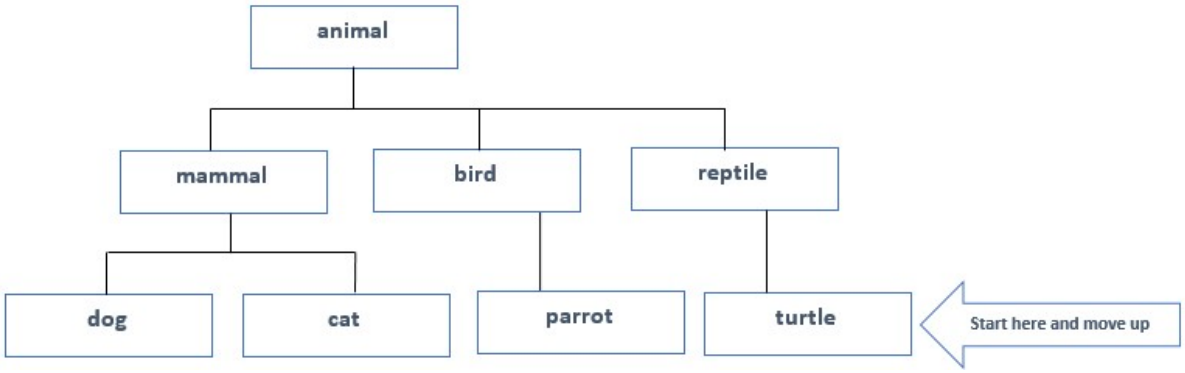


Figure 14: An example of the bottom up modelling approach

Middle-out – this approach starts by examining what is already known about the problem and then using that to define the upper levels of the concept, then lower levels are subsequently added. This can be seen in figure 15. In this example, the types of animal are the focus for the beginning of the development (Sawsaa 2013).

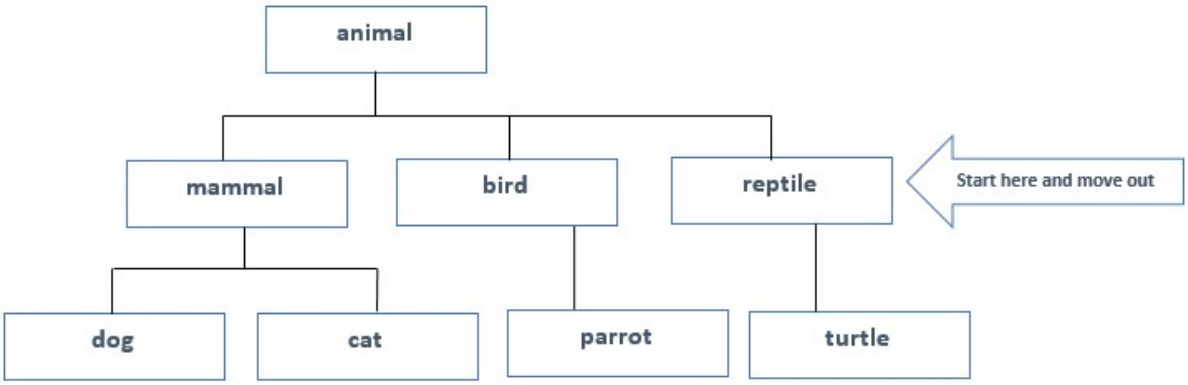


Figure 15: An example of the middle-out modelling approach

Ontology Structure

An ontology is made up of different component parts. The names of these components are different depending on the selection of ontology language employed. There are two types of ontology component, the first articulates the entities within the domain, whilst the second describe the ontology itself or enable its use (Lord, 2010: Online). Stephens (2001: Online), stated that an ontology was made up of four main components; concepts, relationships, instances and axioms (Corcho *et al*. 2010).

- **Concept** – from an ontology perspective, a concept represents a group of entities or things that exist within a domain. *Car* is a concept within the *vehicle* domain.
- **Relationships** – articulate the interactions between concepts or their properties
- **Instances** – are the elements represented by the concept, for example, a Mercedes in an instance of the concept *car*
- **Axioms** – are used to represent values for classes or instances

This is still the case in modern ontology development, but the components of an ontology can be broken down further and more components identified to better understand the way an ontology functions (Sawsaa, 2013);

- *Classes or entities*: clearly identified things that represent concepts
- *Instances or individuals*: present elements contained within the ontology
- *Property*: link relationships between instances, or from instances to other values, such as data and they can be functional, symmetric or transient. Semantic Links between entities can be;
    - *Hierarchical* - superclass and subclass
    - *Equivalent* – similar to a synonym of terms
- *Associative relation*: describes links between concepts

- *Restrictions*: provide information about entities or classes about how they can be used
- *Axioms*: represent information that is true, axioms can be used to infer new knowledge

Categories of Ontology

Many authors have developed a set of categories for ontologies, but they have done so from slightly different perspectives, this has resulted in diversity in classification of ontologies that have differing applications. Sawsaa (2013) and Calero *et al*. (2010) cite the work of Guarino (1998), who defined a classification for ontology based on their generality.

High-level ontology – these ontologies discuss general concepts, for example things like; material, objects, space and time. This type of ontology is free of specific domains or problems, and are used by a number of interested parties to combine principles.

Domain ontology –refers to an ontology, which considers concepts related to generic domains (for example, information systems, medicine or astrophysics). A domain ontology uses the concepts articulated in high-level ontologies.

Task ontology – describes the vocabulary related to a generic activity or task, such as sales or development. A task ontology uses the concepts articulated in high-level ontologies.

Application ontology – These ontologies are created to describe concepts as they relate to a specific application.

Sawsaa states, "… *ontologies are categorised from different approaches and have many classifications based on their structure*" (Sawsaa, 2013: p27). Ontologies are very different from each other; they perform different functions and contain varied structures, which make them distinctive. Table 14 presents alternative approaches to ontology categorisation.

Approach	Ontology Categorisations
Sowa (2000)	• Formal Ontology • Informal Ontology • Domain Ontology
Jurisica *et al*. (2004)	• Static ontologies • Dynamic ontologies • Intentional ontologies • Social ontologies
Fensel (2003)	• Generic or common sense ontologies

	• Representational ontologies • Domain ontologies • Method and task ontologies
Lassila and McGuinness (2001)	• Controlled vocabularies • Glossaries • Thesaurus • Informal is-a relationships • Formal instance • Frames, value restrictions • General logic constraints

Table 14: Examples of different ontology categorisations

Ontology Methodologies

When considering which development methodology to employ, careful consideration must be given to the following factors; the principles, methods, processes, practices and other activities used to design, build, test and deploy the ontology in question (Gasevic *et al*. 2006 p. 65). There are several methodologies available and they have been evaluated in the literature. Examining the work of (Fernandez Lopez, 1999; Corcho *et al*., 2002; Staab and Studer, 2009; Swasaa 2013) leads to the conclusion that;

- ***Most*** of the methodologies proposed for ontology development focus on *constructing* ontologies.
- There are methodologies for ontology development, which include consideration of *merging*, *maintaining*, *re-engineering*, *maintaining* and *evolving* ontologies.
- Some methodologies use *enhanced* software development practices and process to develop ontologies

(Noy and McGuinness, 2001) say, "*Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain"* in their proposal for a simple methodology. The 7 steps contained in their methodology are;

Step 1: Determine the domain and scope of the ontology - this should help develop a concise vision of the constraint of the ontology, what it will be used for, the knowledge and related queries encapsulated by the ontology and guidelines on the maintenance of the ontology.

Step 2: Consider re-use of existing ontologies – given that ontology development is demanding, it is a good idea to check to see if someone else has already done the work of developing an ontology that fits your domain of interest and made it publicly available, so that it can be tweaked or refined to suit a similar application or task. It is important to consider things like language support and other software environment interoperability issues when selecting this route.

Step 3: Articulate the important terms in the ontology – this is the first step to constructing the ontology, and here consideration is given to things like; what are the terms the ontology need to discuss? What properties do the terms have? What needs to be said about those terms?

Step 4: Define the classes and the class hierarchy of the ontology. This step works closely with Step 5 in this methodology and can be performed using one of the three ontology modelling approaches. Top down (identifying the upper most concepts and classes first), bottom up (identifying the most specific classes and concepts first), middle out (starting from some important middle layer classes and expanding in both directions), or by using a combination of these approaches.

Step 5: Define the properties of the classes in the ontology – this step describes the internal structure of concepts by applying properties to defined classes. Properties can be extrinsic (for example; *name*, *duration* and *use*) or intrinsic (for example; *weight*, *height* and *colour*). In this step, relations to other classes and individuals within those classes is also articulated.

Step 6: Define the facets of the properties applied to the classes in the ontology – properties can have different facets describing the type of value it represents. These are things like property value and type, the allowed values (range and domain), number of values (cardinality), and other attributes that can be applied to the property.

Step 7: Create instances – the final step in this ontology development methodologies involves creating instances of the classes defined within the hierarchy of the ontology. Creating and individual instance of a class is a three-step process. The first step is (1) choose a class, (2) create and individual instance of that class and (3) define the property values as they apply to the specific individual instance of the class.

An example of a more comprehensive methodology is the Methontology Framework (Lopez et al., 1999). Using this methodology, it is stipulated that the start of the design process requires the standardisation and characterisation of the complete ontology like cycle – from the specification of

requirements through to plans for upkeep – as well as approaches and practices which will steer the development of the ontology through its life cycle (Gasevic *et al*., 2006; Swasaa 2013). The Methontology framework includes the following elements;

- identification of the ontology development process;
- a life cycle model which is based on evolving prototypes;
- the Methontology methodology, which details steps for completing each activity, the practices involved, the outputs of each activity stage, and an evaluation technique for the developed ontology.

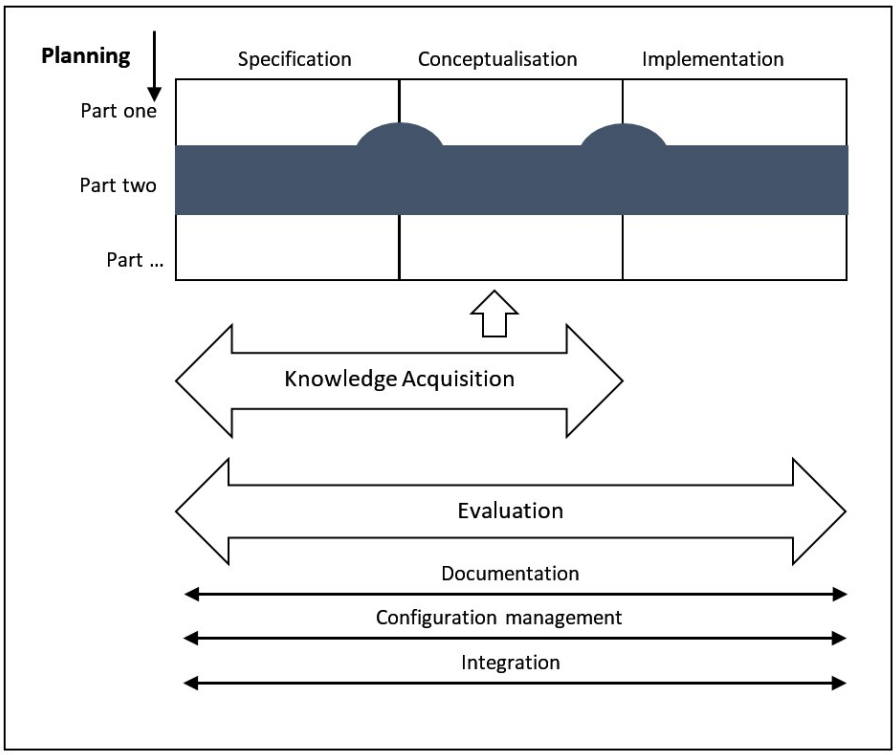


Figure 16: An example of ontology development life cycle (Lopez et al., 1999)

In Methontology, once development starts the activity is started there are three stages, the first of which is *specification* (Calero *et al*., 2010). This is where the goals, purpose, strategy, scope and terminology of the ontology is identified. This is where the reasons for the ontology are articulated, and designers may choose to formulate formal and informal questions to establish and solidify the constraint of the ontology (Fernandez-Lopez *et al*., 1997). This is an important part of ontology development, it helps to ensure that the knowledge contained within the ontology is current, useful and accurate (Sawsaa, 2013).

The second stage is *conceptualisation*. This where the knowledge gained through the specification stage is organised given structure. This is then modelled using semi-formal concept design techniques so the designer and knowledge expert can easily understand the knowledge represented by the ontology and agree on accuracy and relevance (Gasevic *et al.*, 2006).

The final stage is the *implementation* stage, this is where the products of the first two stages are implemented, usually in an ontology development environment (such as, Protégé), to produce the concepts, hierarchies and relationships that bring together the working ontology.

As seen in figure 16, other processes run alongside these three main stages of ontology development. These involve evaluation, documentation, configuration management and integration. This provides a level of quality assurance for the ontology development life cycle (Lopez et al., 1999; Gasevic et al., 2006). The Methontology methodology is particularly effective when building an ontology from scratch or re-engineering, or re-purposing an existing ontology (Fernandez-Lopez, 1999).

A four-stage ontology development methodology was proposed in 1995 by Uschold and King (Calero et al., 2010). The four stages are:

1. Identify the purpose of the ontology
2. Build the ontology
    i. Identify the concepts and relationships within the domain of interest
    ii. Produce clearly articulated definition for the agreed concepts and relationships
    iii. Create the vocabulary or terms to refer to the concepts and relationships
3. Evaluate the ontology
4. Document the ontology

As well as this, *"…it should include a set of techniques methods principles and guidelines for each stage as well as indicating what relationships exist between the stages"* (Uschold and King 1995).

(Gasevic *et al.*, 2006) and (Noy and McGuinness, 2001) state that no methodology can be identified as the best because it depends what the proposed ontology needs to achieve. This claim makes sense, because ontology development is by nature an iterative process, and depending on the results of testing, the selected methodology may need to be revised.

Ontology Tools

There are a range of tools available for modelling and building ontologies. Selecting an existing tool helps to ensure that the created ontology adheres to the design rules set out to make ontologies shareable, extendable and coherent (Sawsaa, 2013). Standard ontology development tools now

include representation languages, graphical development environments and ontology-learning tools (Gasevic *et al.*, 2006). Table 15 describes some example tools.

Tool	Information	Use
CmapTools	CmapTools is an application, which allows users to create graphical representations of knowledge models, which are known as concept maps, as seen in figure 18. This software is a *"result of research conducted at the Florida Institute for Human and Machine Cognition (IHMC). It empowers users to construct, navigate, share and criticize knowledge models…" (*IMHC, 2014: Online)	Ontology planning (graphical)
OntoStudio	This is a commercial modelling environment user to develop and maintain ontologies. *"OntoStudio is also able to import many structures, schemas and models. Some of OntoStudio's most important functions are the mapping tool, which can be used to match heterogeneous structures quickly and intuitively, the graphic rule editor which specialists can use to model complex correlations or the integrated test environment…"* (GmBH, 2012: Online). A trial version of this software is available, but it only lasts for a month.	Ontology development
Protégé	Protégé was developed by the Stanford Medical Informatics team at Stanford University. According to the website, it is an *"ontology editor and framework for building intelligent systems"* (Protégé, 2016: Online). The ontology editor is the heart of this application; it allows the creation of standalone and shared ontologies. There are a range of add-ons for the application (e.g. SnoRocket, one of range of different reasoners, which will check the logic of the created ontology and OWLViz a tool that provides a visual representation of the elements and relationships for a given ontology). This software is open source and therefore is free to use under licence.	Ontology development
WebODE	This is a collection of tools for ontology engineering, which is based on an application server. This suite of tools was initially developed in 1999, development continued until 2006 when support was withdrawn. "*WebODE was built as a scalable, extensible, integrated workbench that covers and gave support to most of the activities involved in the ontology development process (conceptualization, reasoning, exchange, etc.) and supplied a comprehensive set of*	Ontology development

	ontology related services that permit interoperation with other information systems." (WebODE, 2016: Online). It supports a host of languages, including; XML, RDF(S), OIL, DAML and OWL.	

Table 15: Examples of ontology tools

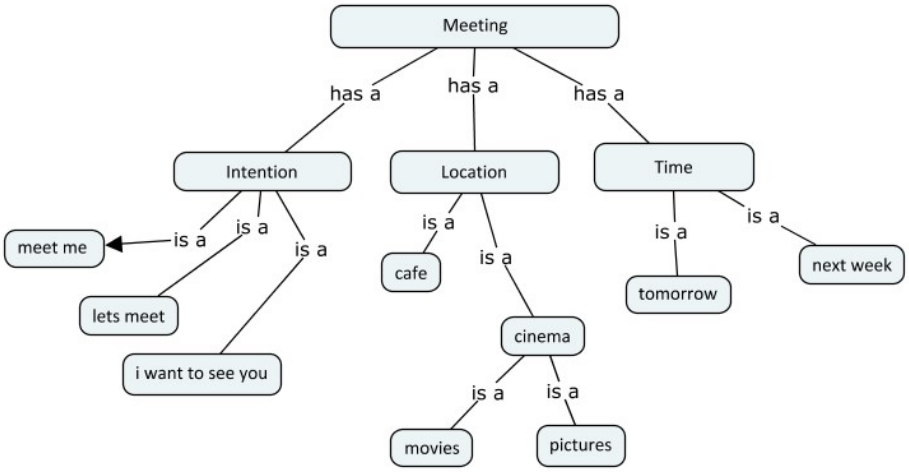


Figure 17: An example of an ontology concept map created with CmapTools

There are several ontology representation languages to choose from, table 16 presents an overview of some of the more prominent ones. Some have been developed purely by AI researchers (Gasevic *et al.*, 2006), whilst others, from the late 1990's have been developed in conjunction the World Wide Web Consortium (W3C) (Antoniou et al. 2012).

Language	Description
DAML+OIL	This is a semantic markup language, which builds on some of the earlier standards like RDF and RDF Schema to provide richer modelling primitives (Connolly *et al.,* 2001).
Loom	This is a knowledge representation language, which was developed by researchers at the University of Southern California in the Information Sciences Institute (Loom, 2006). This language is based on description

	logics, meaning that the declarative knowledge in Loom is made up of definitions, facts and default rules.
KIF	Knowledge Interchange Format (KIF) is a first order logic based language designed to enable computer systems to share information and re-use knowledge (Genesereth *et al.,* 1992)
OIL	Ontology Interchange Layer (OIL) is a semantic markup language and used description logics implemented using frame based representation primitives (Fensel *et al.,* 2001).
OWL	Web Ontology Language (OWL) is designed for use by applications to share knowledge by asserting context to declared entities. This language is the evolution of DAML+OIL and is currently the most popular representation language (OWL, 2004).
RDF	Resource Description Framework (RDF) is a model for enabling the exchange of data on the internet. It has features for data merging from differing data representations and supports the evolution of representations over time (RDF Working Group, 2014).
RDF Schema	RDF Schema is a data modelling vocabulary for data, which is represented using RDF (Brickley and Guha, 2014). The combination of RDF and RDF Schema is known as RDF(S).

Table 16: Ontology representation languages

Ontology Evaluation

Evaluating an ontology can prove to be problematic, there is no one size fits all solution and as is the case with all ontology development tools and techniques, different ontologies require their own evaluation methods (Sawsaa, 2013; Calero *et al.,* 2010). The most well-known and enduring ontologies (for example, SENSUS, Cyc Ontologies etc.) have no real documentation or publications that articulate how they were evaluated.

Gomez-Perez (2001), recognised that there is a lack of specific mechanisms for evaluating ontologies and knowledge sharing technologies in general and went on to propose a series of steps that would go some way to creating a framework for ontology evaluation, these are;

1. Evaluate the ontology from 2 perspectives, the developer and the user.
2. Create a set of terms for the ontology, ensuring that standard definitions exist for those terms

3. Define a criterion, by which the technical and user evaluation process can be measured
4. Develop tool or mechanisms which will allow some evaluation of an ontology during its creation
5. Create tools to evaluate existing technologies
6. Include evaluation methods in tools used to create ontologies

Later a different approach was offered, which involved evaluation the ontology from two perspectives; ontology validation and ontology verification (Gomez-Perez *et al.,* 2004). The validation process evaluates weather the ontological definitions match those of the real world (depending on the purpose of the ontology), whilst the verification process ensures that the ontology is built in the correct way to support the implementation of its definitions. The emphasis of evaluating ontologies in this way is to ensure that concepts are designed correctly. There are two areas of concern for this evaluation method;

Consistency – there are no contradictions in the classes or elements and the data presented will not give contradictory conclusions

Completeness – the ontology successfully represents the whole domain of knowledge represented by the ontology and does not have any completeness errors

The validation process can be performed automatically by a Description Logic Reasoner (DLR), and these can be found embedded within ontology development tools or added as a bolt on application. An example of this is FaCT++, which is a DLR that can be used within the Protégé ontology development application (FaCT++Reasoner, 2016: Online). This validation approach is important to assess the quality of the knowledge represented in the ontology, and as such has some criteria to assess the level of quality. The quality criteria are (Sawsaa, 2013);

1. **Consistency** – there should be no contradiction between the concepts presented in the ontology
2. **Completeness** – the ontology should cover the whole area of knowledge as it applies to the domain, all concepts should be fully defined and there should be no missing concepts
3. **Conciseness** – there should not be any needless or redundant information presented in the ontology
4. **Clarity** – the ontology declares concepts which effectively present the knowledge domain concerned

5. **Generality** – the ontology can be used for a more than one purpose in the same domain
6. **Robustness** –the ontology can support future changes to the domain
7. **Semantic data richness** – the ontology is rich and diverse in its conceptualisation
8. **Subject coverage** – the ontology covers the particular domain well

The has been the proposition of cohesion metrics to evaluate ontologies (Yao et all, 2005). They propose an approach which involves testing cohesion by measuring the number of root classes, the number of leaf classes and the average depth of inheritance between tree of leaf nodes.

Another approach to ontology evaluation involves the investigation of how well the ontology fits the domain of knowledge that it represents (Brewster *et al.,* 2004). This involves comparing the ontology concepts and relations against text documents which contain domain specific knowledge.

Ontology evaluation is required throughout the ontology life-cycle, particularly during the stages of pre-modelling, before release and after release (Bilgin *et al.,* 2015). Ontologies are seen as vital to the knowledge description domain and pivotal to the development of the semantic web (Hlomani and Stacey, 2014), therefore mechanisms to evaluate their effectiveness and improve their development are just as important.

Example Ontologies

LinKBase is a natural language based ontology for the medical profession and is defined as "A conceptual representation of medical information computers can understand and process" (Nuance, 2016: Online). The aim of LinKBase is to provide third party application with an ontology that supports Natural Language Processing(NLP) and Natural Language Understanding (NLU) for biomedical data applications (van Gurp *et al.,* 2006).

This ontology has been structured using Basic Formal Ontology (BFO), which provides an upper level ontology framework for the creation of scientific research focused ontologies (Stenzhorn, 2008: Online).

> *"Theories of endurants and perdurants, mereology, topology, universals and particulars, biological classes and instantiations, space and time and granular partitions are all included in the BFO theory."*
>
> (van Gurp *et al.,* 2008)

There are more than 570,000 types articulated in LinKBase, which represent entities in the real world. To allow for semantic reasoning they have been structured hierarchically using a real life

approach; **child** types represent a subclass of a **parent** type in every single instance. This means that relationships in this ontology are able to maintain consistency in their meanings, as seen in figure 18. For example, *rash* will never occur as a subclass of *allergic reaction* because a rash is not purely caused by an allergic reaction (van Gurp *et al.,* 2006). This ontology also boasts a lexicon of over 1.5 million terms, which are used to represent types in the real world. Terms can be translations, synonyms, singular or plural forms of the type they represent.

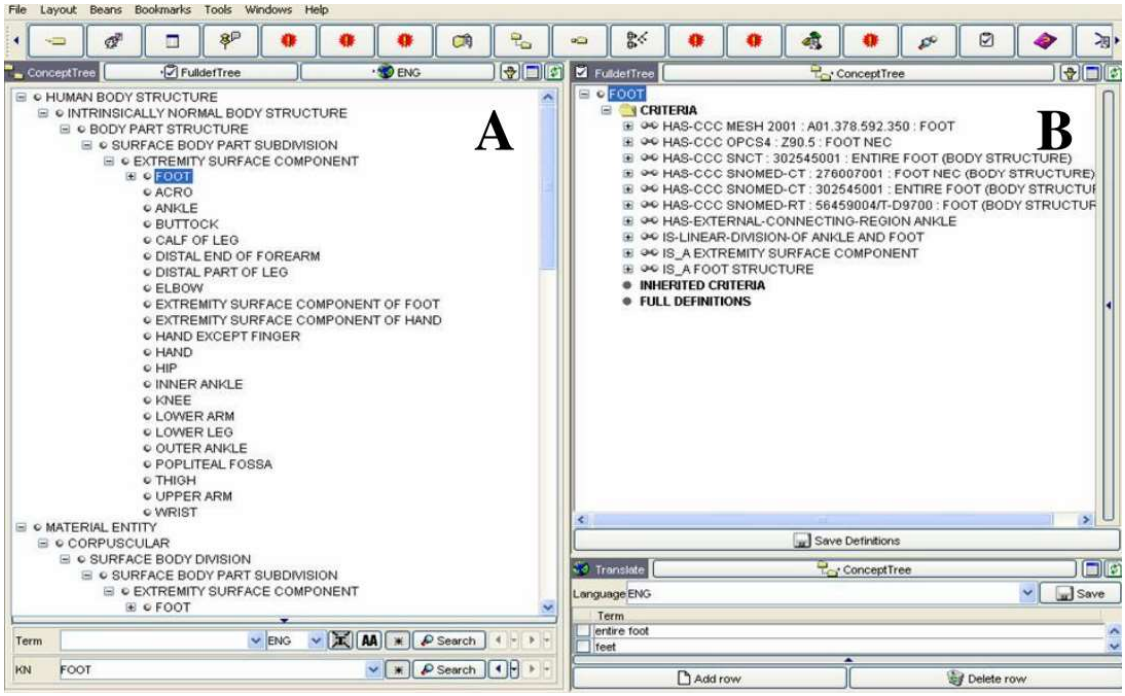


Figure 18: LinKBase structure (van Gurp et al., 2006)

The creators of LinKBase have been successful in building and maintaining a useful and relevant ontology that integrates with other software to provide knowledge for the medical domain, this is evident in the fact that it is used by Nuance (a company that provides healthcare solutions to healthcare professionals) to support medical services to individuals and businesses (Nuance, 2016: Online).

WordNet is a lexical database of the English language. It is made up of Synsets (cognitive groups of nouns, verbs, adverbs and adjectives) each describing a singular concept. These synsets are linked together by conceptual-semantic and lexical relationships (Princeton, 2016: Online). WordNet can be explored using a web browser, as seen in figure 19, and is freely available for public download.

WordNet has many applications, (Pal *et al.,* 2014) used it to research query expansion (QE), a widely used technique, which tries to increase the occurrence of relevant matches made between a query and the data being queried. They theorise that adding semantically related terms to a query, that the user will receive more relevant returns and experience improved query performance. They conclude that after experimentation, their proposed way of querying WordNet outperformed WordNet's own methods for QE.

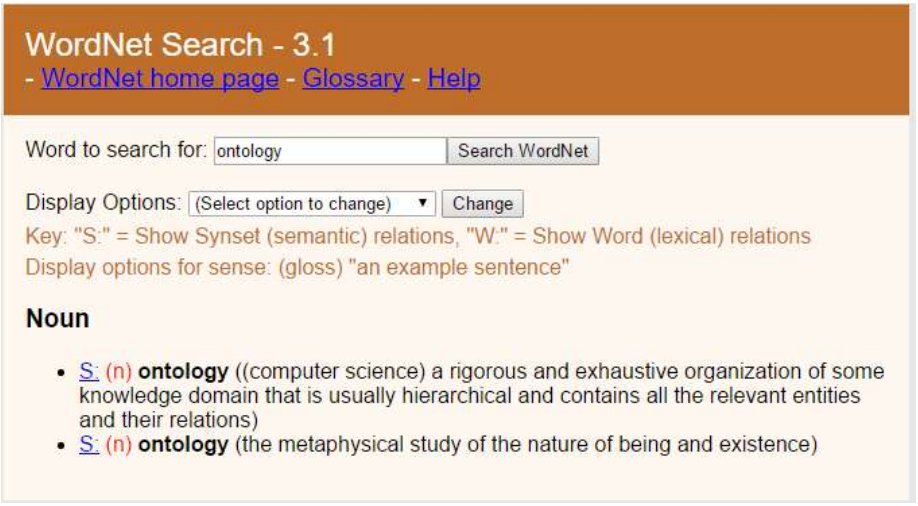


Figure 19: Result of the search for "ontology" using WordNet in a web browser

WordNet has been used as a semantic hub to help increase the integration of linked open data (LOD) (Ballatore *et al.,* 2014). The way synsets work make this ontology attractive to other developments as the links between words and word senses is encapsulated well in the construction of the synset and its applied relationships. Bellatore *et al.,* go on to document the development of the *Voc2WordNet* algorithm, used to generate syntactic mappings between vocabularies and the WordNet ontology. They conclude that WordNet provides a shared semantic grounding to enable interoperability for heterogeneous vocabularies.

WordNet does face challenges, it was developed in English, using all of the rules that apply to tense etc. that the English language follows, and this has proven to be problematic when translating it into different language sets. Overall it has been a successful in providing a solid foundation for other knowledge based semantic applications.

2.6 Natural Language Processing

An alternative solution to the meeting detection problem would be to employ conventional Natural Language Processing (NLP) techniques directly. As stated by Jackson and Moulinier (2002) the term NLP refers to the function of software or hardware used to analyse spoken or written language within a computer system (MacFarlane and Holmes, 2009). Dictionary.com defines natural language processing in the technology section as:

> *"artificial intelligence*
> *(NLP) Computer understanding, analysis, manipulation, and/or generation of natural language.*
> *This can refer to anything from fairly simple string-manipulation tasks like stemming, or building concordances of natural language texts, to higher-level AI -like tasks like processing user queries in natural language."*
>
> (Dictionary [b], 2016: Online)

Elizabeth Liddy in the paper, Natural Language Processing (2001), gives the following, more articulated definition:

> *"Natural Language Processing is a theoretically motivated range of computational techniques for analyzing (sic) and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications."*
>
> (Liddy, 2001: Online)

Overview

The human race displays a unique capacity for language, when compared to any other known species. Humans developed the ability to talk to one another around 100,000 years ago, and began recording experience and thought through written communication around 7,000 years ago. Although some animals display vocabularies made up of hundreds of signs, only humans have proven to be able to communicate in many different ways using an unquantifiable number of messages on any topic using discrete signs (Russell and Norvig, 2014).

A computer needs to understand language for two main reasons, one to communicate more effectively with humans, and the other is to provide the computer with the capacity to understand knowledge presented in the wealth of written information available today (DeAngelis and Solutions, 2016: Online).

The challenges facing any language detection solution, centre around the fact that the knowledge of language needed to understand complex language behaviour can be broken down into six categories, as set out by Jurafsky and Martin (2008)

- **Phonetics and Phonology** – phonetics relates to the production of speech sounds and how they are then perceived, whist phonology relates to the way speech sounds are interpreted
- **Morphology** – this is the study and description of the structure of a given languages' linguistic units (words and parts of words), for example, parts of speech, root words, affixes, intonations, or the context implied.
- **Syntax** – the rules used in the formation of grammatical sentences, specifically phrase structure and word order.
- **Semantics** – the meaning of language, split into two main areas of concern; logical semantics (sense, reference, presupposition and implication) and lexical semantics (word meaning and word relationships).
- **Pragmatics** – deals with the ways in which language is used to accomplish goals, including deixis (the function or use of deictic words, forms, or expressions (Google [d], 2016: Online), conversation semantics, and the organisation of text.
- **Discourse** – linguistic occurrences (written or spoken communication) that are bigger than a single utterance, for example, discussion and debate.

Manning and Schutze (2001, p. 18) further explain the problem when they assert that an effective NLP solution must be able to disambiguate decisions involving word sense, word category, syntactic structure and semantic scope.

Applications of Natural Language Processing

There are many applications for NLP, there are five main areas of application (Sandhu, 2016: Online), these are;

Machine Translation – a lot of the worlds stored information can be found online and needs to be more accessible. The challenge of making this wealth of information available to all, regardless of language or location has become too large for human translators. Machine translation solutions are being developed and offered, Google is one of the largest companies working in this field and uses a proprietary statistical engine to power its Google Translate service (Chitu, 2010: Online).

Information Extraction – NLP is being wildly implemented in applications which strive to make sense of written information for one reason or another. The computer may be looking for news pertinent to the stock market, scanning for mentions of a particular incident on social media or collating information for medical purposes. For example, the Protein Active Site Template Acquisition (PASTA) and the Enzyme and Metabolic Pathways (EMPathIE) projects (Humphreys *et al*., 1999).

Summarisation – In an age where the amount of digital text available is staggering and far exceeds our capacity to understand it all, we need a way to present information in a more condensed way in the first instance. This enables us to search for more appropriate information in a more expedient manner. Applications are available to provide summaries of documents for a variety of purposes (SMMRY, 2016: Online), and this branch of NLP will become increasingly useful as a marketing asset (S, 2016: Online).

Fighting Spam – Spam filtering is a very important and increasingly challenging field of NLP. The false-positives and false-negatives are at the crux of spam management, which relies on extracting and correctly identifying the meaning of strings of texts. Bayesian spam filtering is proving to be more effective in the battle against the plethora of unwanted emails being received (Graham, 2002: Online)

Query Resolution – Search engines provide a gateway to immense amounts of information and digital resources. NLP techniques are being used to help search engines recognise natural language questions (Al-Rfou et al., 2016: Online). This is one of the major challenges for NLP and is a large research focus area (Sandhu, 2016: Online).

2.7 Summary

For this project, it would be beyond the scope, both in terms of resources and time to create a single intelligent machine like IBM's Watson. As is the case with Cortana and Siri, there is no need for a dedicated super computer because the problem is much more focused and needs to run in a smaller, more measured environment. In order to be effective, the system will need access to semantic knowledge and natural language processing techniques, which eventually will have to develop into the realms of speech recognition, taking into consideration the changing interactions between humans and their computing devices.

Therefore, efforts will need to focus on developing a system that uses the same various component technologies as some of the current developments presented, so that the SafeChat system will be able to automate decisions based on what is known about the ongoing interaction between two users.

Any proposed system has taken into consideration ethical issues. System processes need to be designed to act autonomously to protect users in an online communication environment. Conversations should not be recorded, although anonymising discourse data may help in future development cycles if machine-learning techniques were to be employed to refine system performance. These recorded or stored conversations could easily be anonymised to protect the privacy of those involved.

Agent technologies are an integral part of the new semantic web features, and will be used to make searching for information much more effective. (Bellifemine *et al.* 2007 p.77) state that the JADE platform uses the FIPA compliant Agent Communication Language to communicate and syntax is known as content language. Research suggests, as seen in table 13, that the Java Agent Development Environment (JADE) environment is the best suited to the requirements of an agent system. This is because it is fully FIPA compliant, supports ontology development, is fully maintained and has useful add-ons to assist integration across multiple platforms (e.g. the Leap add-on allows agents to function on standard mobile phone devices) (JADE Team, 2016: Online).

Research into existing systems, starting on page 53, has been highly beneficial. Small but ultimately, crucial concepts lead to further developments. For example, the Genghis system, much like IBM's Watson, applies a ranking system to the possible outcomes presented. The SafeChat system will need to rank discourse according to current level of risk, any ultimately use this cumulatively to decide on the overall safety level of a conversation.

Initial research centred upon the development of a single messaging application with built in protection, but users would be free to join several virtual 'chat rooms'. The Essex- IB model provided an insight into multi-faceted, distributed agent systems, which will still need to be considered, as a user is likely to be communicating using multiple applications.

The agents any proposed solution will be required to communicate for the appropriate actions to be taken and to assure system efficiency, research into how agents communicate has proven to be vital in understanding how this should work, and has also influenced the choice of multi-agent framework.

Research into this negotiation amongst agents and agent system development informs essential techniques required to ascertain maximum utility from the agents in a system.

Utility theory could be used in conjunction with probability theory to refine agent decision making and would enable a multi-agent system to allocate a level of threat to a given conversation and decide which messages to allow and which need to be blocked to ensure that safety of the user is maintained.

With regards to any proposed solution, the environment in which it must operate (mobile, desktop, wireless, fast and slow connectivity and limited storage space), and the specific nature of the domain of knowledge required, the most logical conclusion would be to create a focused ontology, which is application specific. This would limit the computing overheads involved in checking message content and inter system communications.

It is proposed that an ontology, once completed will provide a comprehensive vocabulary for referring to the terms in a specific subject area (Gasevic et al. 2006 p.51), which in this case is the conceptualisation of a meeting. With this knowledge, a system would be able to make informed decisions on elevating the threat levels contained within a message or conversation, and then take the appropriate defined actions based on the current value of said threat level.

An ontology will need to adhere to accepted ontology design practice, it would need to be updated, as any language oriented ontology would. Language is an evolving thing, it changes and adapts constantly, an ontological solution should take this into consideration. The ontology will need to be clear, concise and fit for purpose, but also be lean in its design, so as to minimise computational overheads and preserve transparency.

Protégé is widely used to develop ontological solutions, this suite of tools provides a graphical user interface to develop ontologies, has built in description logic reasoners to validate ontologies, and offers functionality to export an ontology into a range of ontology languages. The ontology language support that Protégé provides is hugely beneficial for use in agent systems, as it will enable third party developers, who wish to embed the ontology into their messaging applications, flexibility to do so with minimal encoding bias.

Most uses of natural language processing and by extension, NLP based Ontologies, are working with existing information sets or scanning documents. Any proposed system will need to achieve this ‘on the fly’, therefore the ontology must act as the data set. NLP techniques are embedded in the ontological engineering. The definition of relationships between words can provide the knowledge and understanding of natural language so that a proposed system would be able to correctly identify meeting elements within communications.

Chapter 3: SafeChat System Design

3.1 Introduction

This chapter will discuss software development methodologies and identify the methods used to develop the SafeChat system. It will go on to detail the design and development of the various components of the system.

The research articulated in chapters 1 and 2 of this thesis indicates that there is a need to provide an autonomous solution to ensuring the safety of children communicating online. The SafeChat system has been developed to achieve this goal.

The system needs to provide an autonomous solution that protects the privacy of its users, to block the transfer of personal information, provide an indication of the threat or safety level of current discourse between two users, and act if the threat level threshold is exceeded. The system must also have as limited a computational footprint as possible to be transparent to the user.

3.2 Software Design Methodologies

The SafeChat System, as seen in figure 20, is made up of three component software systems, each diverse enough in their composition to require a slightly different methodology for their development. The multi-agent system and the simple chat system are more obviously aligned to more traditional software development life cycle paradigms. The SafeChat Meeting Ontology requires the application of an ontology development methodology.

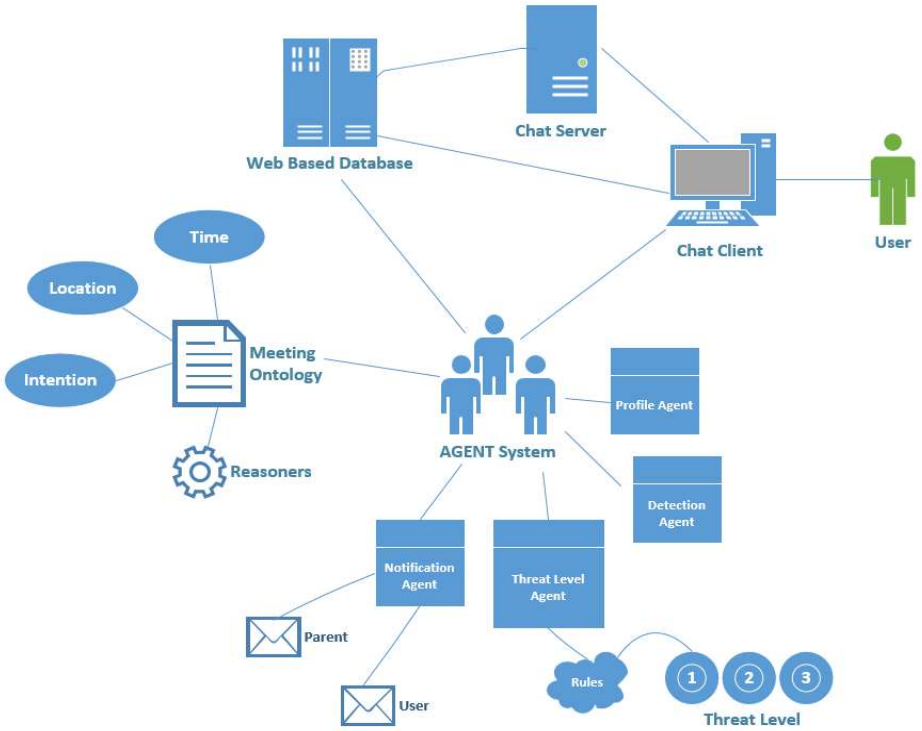


Figure 20: The SafeChat System

There are many Software Development Lifecycle (SDLC) methodologies available (Purcell, 2016), each with its advantages and disadvantages depending on the application area. SLDC has five main areas of consideration (Requirements Analysis, System Design, Implementation, System Testing, and Evolution). Table 17 describes the leading software development models (TechTalk, 2015: Online; Purcell, 2016: Online).

SDLC Model	Description
Waterfall Model	This is the oldest SDLC model. The Waterfall model follows sequential step by step process; each preceding step must be complete before moving to the next stage of development. The steps are; requirements analysis => software design => implementation => verification => maintenance. The problem with this method lies in the inability to review steps or introduce new knowledge at any stage, meaning that the finished product may be obsolete or incomplete by the time it is produced (Purcell, 2016: Online).
Spiral Model	The spiral SDLC model was developed in response to the weaknesses apparent in the waterfall model. This model includes risk analysis, prototype development, measurements, software design, next phase development and review steps. Information and feedback is gathered and acted upon as the cycle is repeated. This method helps to provide indication of risk, costs and functions, and users can be involved in the system early due to the prototyping. The weaknesses of this model are in time spent on continual assessment and risk evaluation. This is a complex model and the spiral may continue indefinitely if the process is not tightly managed (Purcell, 2016: Online).
Rapid Prototyping	This method puts less emphasis on planning and more onto development and is iterative in nature. It only has four steps; planning, design, system development and implementation. The strengths associated with this model are; time and money savings, focused code development. This method employs modelling concepts to identify and capture data and processes. Identified weaknesses of this method includes; risk of achievement, difficult to merch with legacy systems, works better for modular systems and developers must deliver to tight schedules. (TechTalk, 2015: Online)
Iterative Model	This model implements a small prototype containing some of the software requirements and iteratively adds and enhances the evolving version of the

	software until all of the requirement are met and working as intended (Spence, 2005: Online). The advantages of this method is that feedback is obtained between each phase of development from coding teams and end users, this can then fully inform and enhance the end product, as seen in figure 21.
Object-Oriented Model	In object oriented development there are five conceptual tools to help the programmer, these are; encapsulation, data protection, inheritance, interface and polymorphism. The idea behind this model is that classes, methods, relationships and properties of objects are developed from a real world view (McKay, 1988- 2016: Online).
Agent-Oriented Model	Early agent oriented software development models attempted to design multi agent systems using a societal view of the system, defining relationships between concepts in much the same way as they are defined in our own society (Wooldridge *et al.,* 2000). The Gaia model proposed by Wooldridge et al., defines concepts as either abstract or concrete. Abstract concepts include; roles, permissions, responsibilities, protocols. Concrete concepts include; agent types, services and acquaintances. Another agent oriented methodology developed to support analysis and design activity for agent systems is Tropos (Bresciani *et al.,* 2004). In this methodology there are five main areas of development; early requirements, late requirements, architectural design, detailed design and implementation.
Agile Software Model	Agile software development methods try to reduce the risks associated with software development, by developing software in short time periods, often referred to as iterations. These iterations can last from one to four weeks typically and one includes all the work involved in completing this iteration of the development, including; planning, requirements analysis, design, coding, testing, and documentation (Association of Modern Technologies Professionals, 2016: Online). This method promotes collaboration with the end users of systems and requirements change, or requirements creep, is a welcomed part of development (Greer *et al.,* 2011). Unremitting design improvement, system testing and continuous integration are all features of Agile software development models (Misra and Singh, 2016).

Table 17: Software development life cycle models

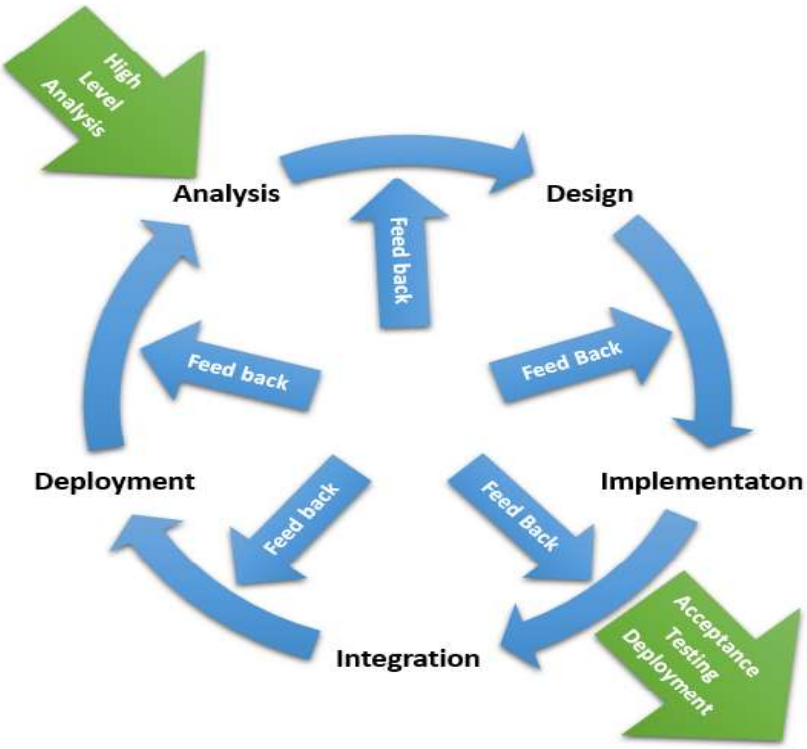


Figure 21: The iterative development method

The development method chosen for each component system of the SafeChat system is different, but they share the same principles, as seen in all the presented SDLC's. The SafeChat multi agent system has been designed using elements from agent oriented, iterative and agile software development methods. The simple messaging application is more traditional and therefore uses elements from iterative, object oriented and agile software development methods. The SafeChat Meeting Ontology has been developed using the Methontology ontology methodology discussed in Chapter 2, sub-section 2.5.

3.3 Agent System Design

The SafeChat agent system, as seen in figure 22, can be defined as a system of agents working together to autonomously ensure that safe practice guidelines are followed, which will provide safety for children communicating in an online environment. The system needs to block the transfer of personal information, which could be used to target the child, and it needs to block attempts at arranging meetings with children. The system needs to achieve this autonomously in order to protect the child's privacy, and to a certain extent, allow the child to act autonomously in ways which would not place them in danger.

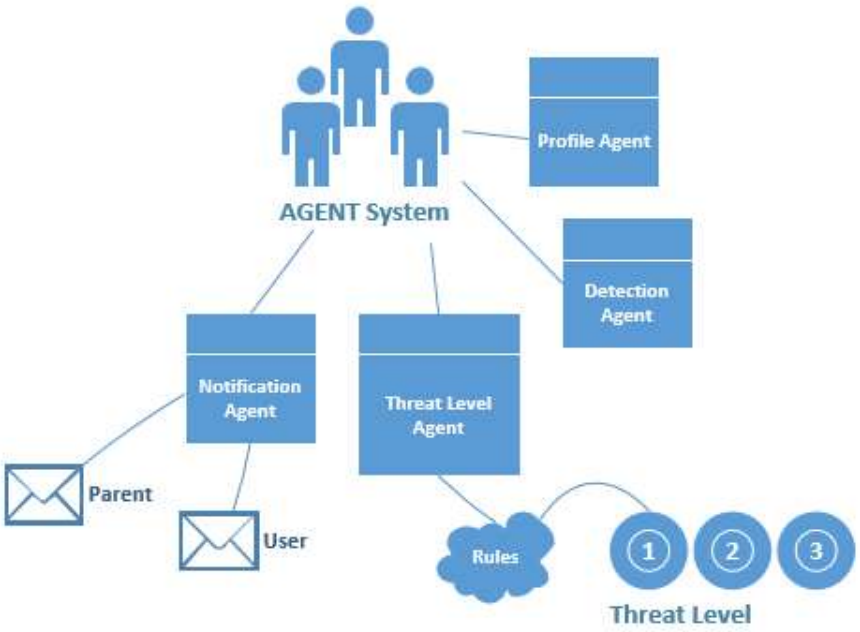


Figure 22: The SafeChat Multi-Agent System

During a recent development iteration, an agent was removed from the system. The WarningAgent was responsible for sending warnings to the user in cases where a message was found to be unsafe (MacFarlane and Holmes, 2009). There are two types of unsafe messages in the system, the first contains any elements of personal information. The other type of unsafe message contains elements that make up meeting arrangements. Another agent, the NotificationAgent is responsible for sending a warning message to the registered parent of the child in cases where messages were deemed to be unsafe. The responsibility for all warnings/notifications now resides with the NotificationAgent, thus rendering the WarningAgent superfluous to system requirements.

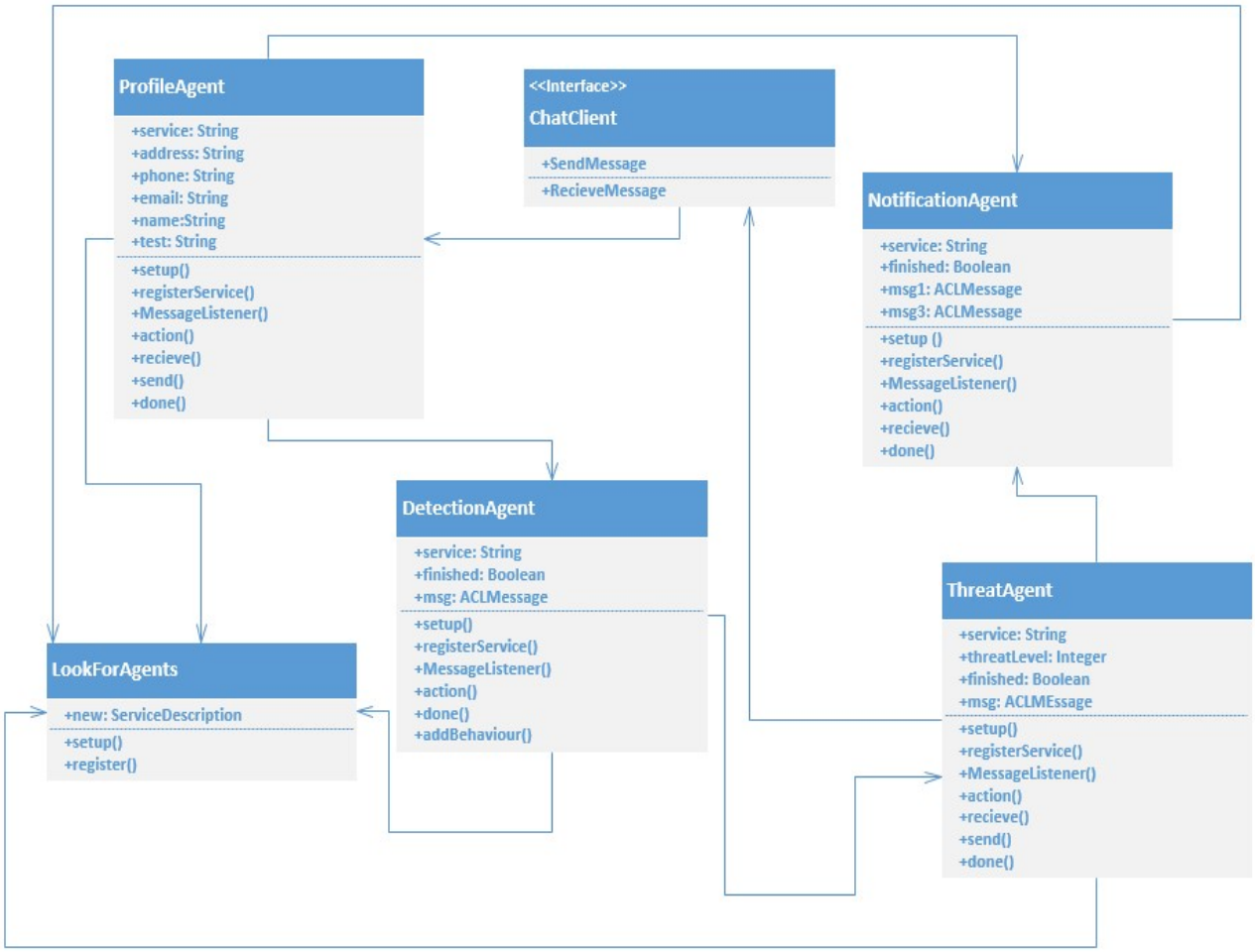


Figure 23: SafeChat multi-agent system class diagram

As seen in figure 23, the SafeChat multi agent system has five agents in its current iteration, here is a description of each agent and its intended functionality;

ProfileAgent - This agent performs a comparison between the message and the stored user information, if it finds a match the message is tagged as unsafe. The ProfileAgent informs the NotificationAgent what kind of information was being sent, so that the NotificationAgent can then take appropriate action, and the message is blocked. If the message is safe, the ProfileAgent sends the message through to the DetectionAgent.

DetectionAgent - This agent captures each safe message from the ProfileAgent; it passes it to the ontology query mechanism and waits for a numerical response, once received the numerical response and the original message is sent through to the ThreatAgent.

ThreatAgent - This agent will monitor and maintain a threat level for the current conversation. This agent is a pro-active agent and it will check all numerical data passed to it from the DetectionAgent passed through the system, in order to effectively maintain an accurate measure of the current threat level. While the threat level is deemed as safe, messages will be allowed through the system. If the threat level exceeds safety

parameters, the messages are blocked and the NotificationAgent will be informed that warnings need to be issued.

NotificationAgent - This is a reactive agent, it does nothing until it receives instructions from the ProfileAgent or NotificationAgent that there is a problem, when this happens it triggers and email to the parent of the child and transmits a warning message to the user with details of the transgression.

LookForAgents - This is used as a housekeeping agent to monitor running agents and ensure that the agents are successfully registering with the Directory Facilitator.

All of the agents in the system register their services with the Directory Facilitator in the Java Agent Development Framework (JADE). This will allow agents to find services offered by other agents in this co-operative multi-agent system. This functionality is not used in the current iteration of the system, but in future developments it may be required.

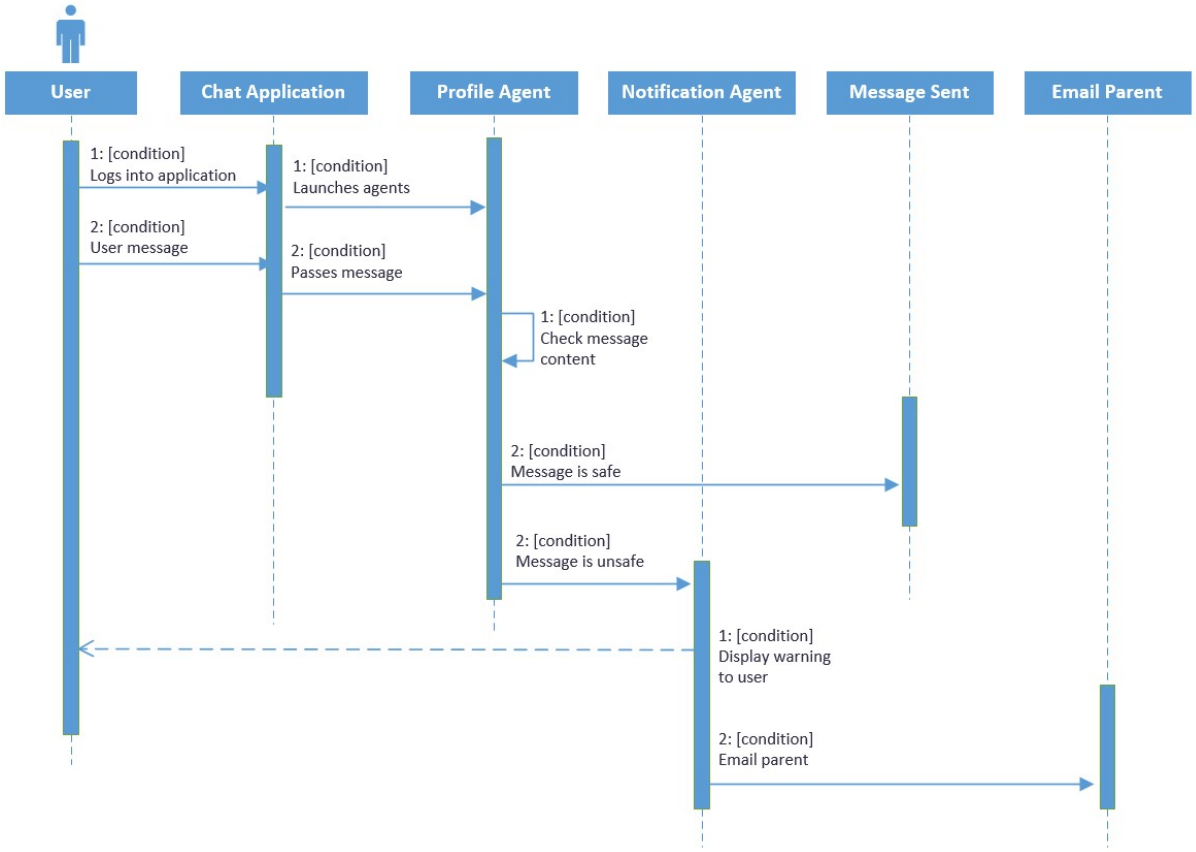


Figure 24: Personal information checking agent system sequence diagram

The steps involved in checking for personal information within a message are outlined in figure 24. As shown, the message content is checked against stored information and if a match is identified, the

Profile agent takes one course of action, if there is no match the message continues through the system.

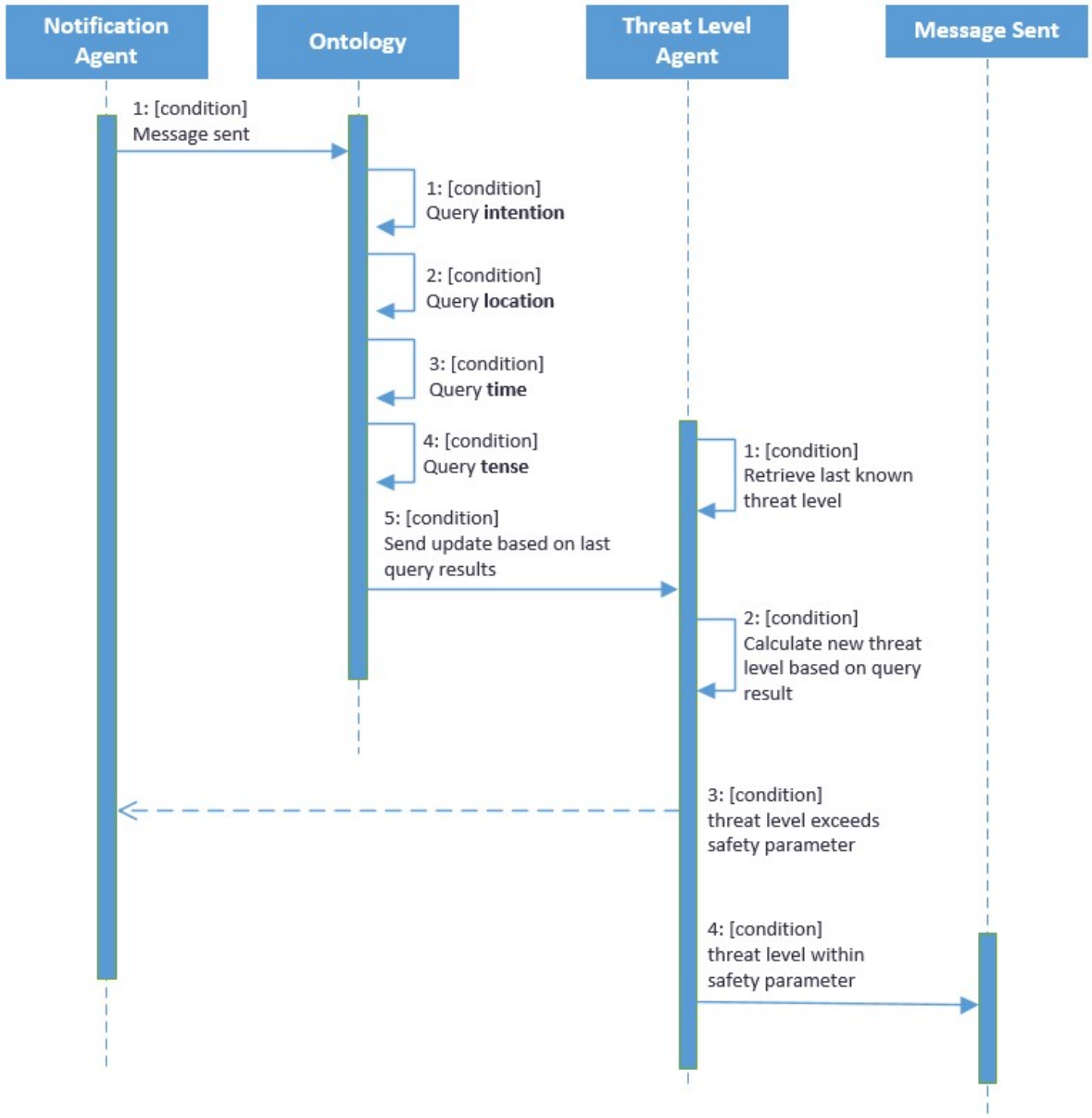


Figure 25: Meeting checking agent system sequence diagram

Detecting meeting elements within a message or conversation provides a greater challenge (MacFarlane and Holmes, 2016). The agent system needs access to knowledge about language, and this can be provided by the ontology. What the agent then does with this knowledge is outlined in figure 25. The agent will need to record the last known threat level of the last known conversation between the same users in order to maintain the true threat level of the interactions between particular users. The last known threat level could be stored in a secure database or encrypted file.

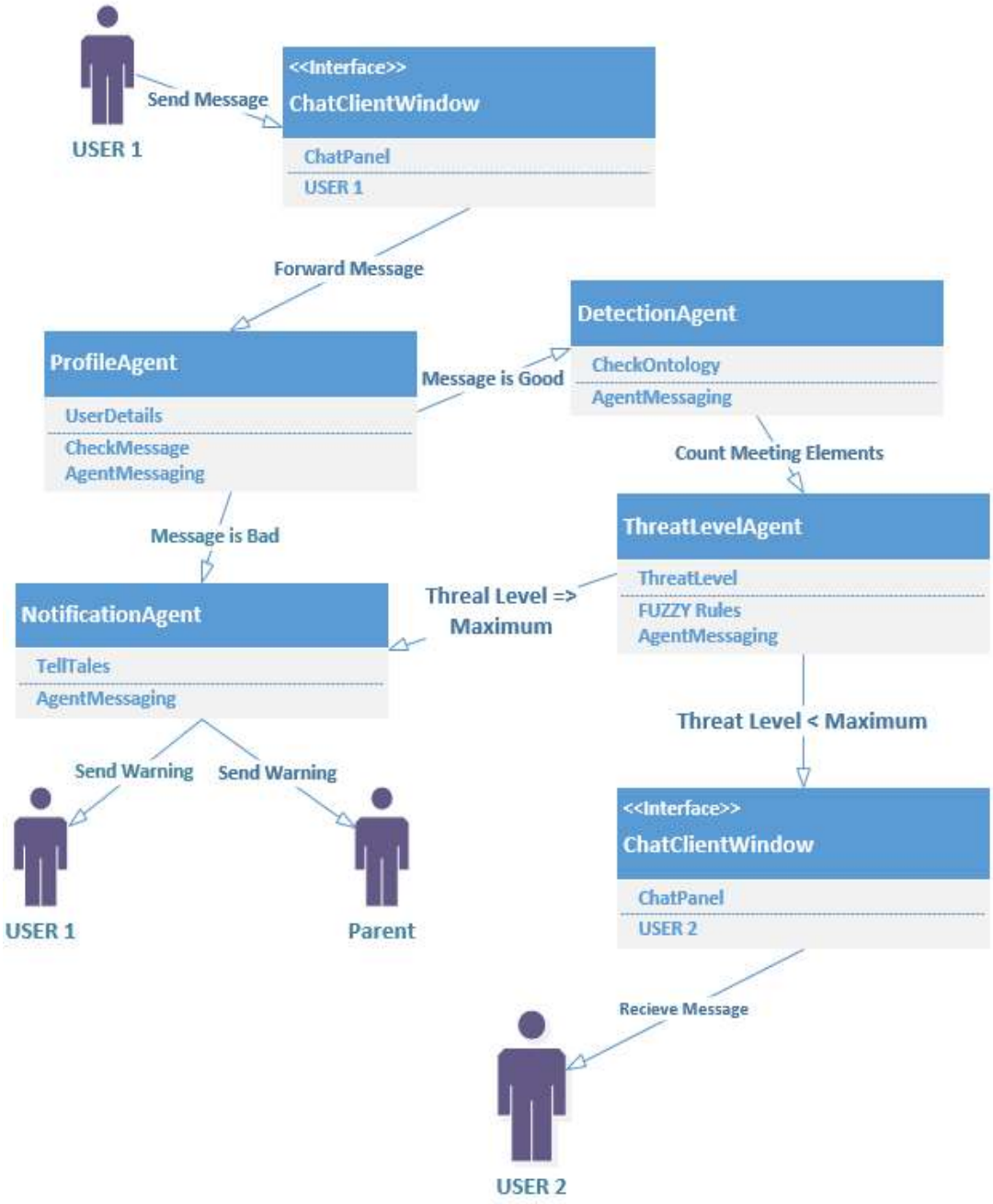


Figure 26: SafeChat message flow

Figure 26 illustrates the route of each message through the multi-agent system, and indicates that there are many steps a message must go through before it is released to its intended recipient. This will present a challenge for the system, as extensive delays will be noticed by the users, which will in turn compromise the transparency of the system.

3.4 Message Application Design

Initially the focus of the messaging application was to develop a fully working, platform independent instant messaging application, which would be used specifically for children. The prototype development included functionality for; chat server, chat client, database connectivity, message exchanges and friends (or contacts) lists.

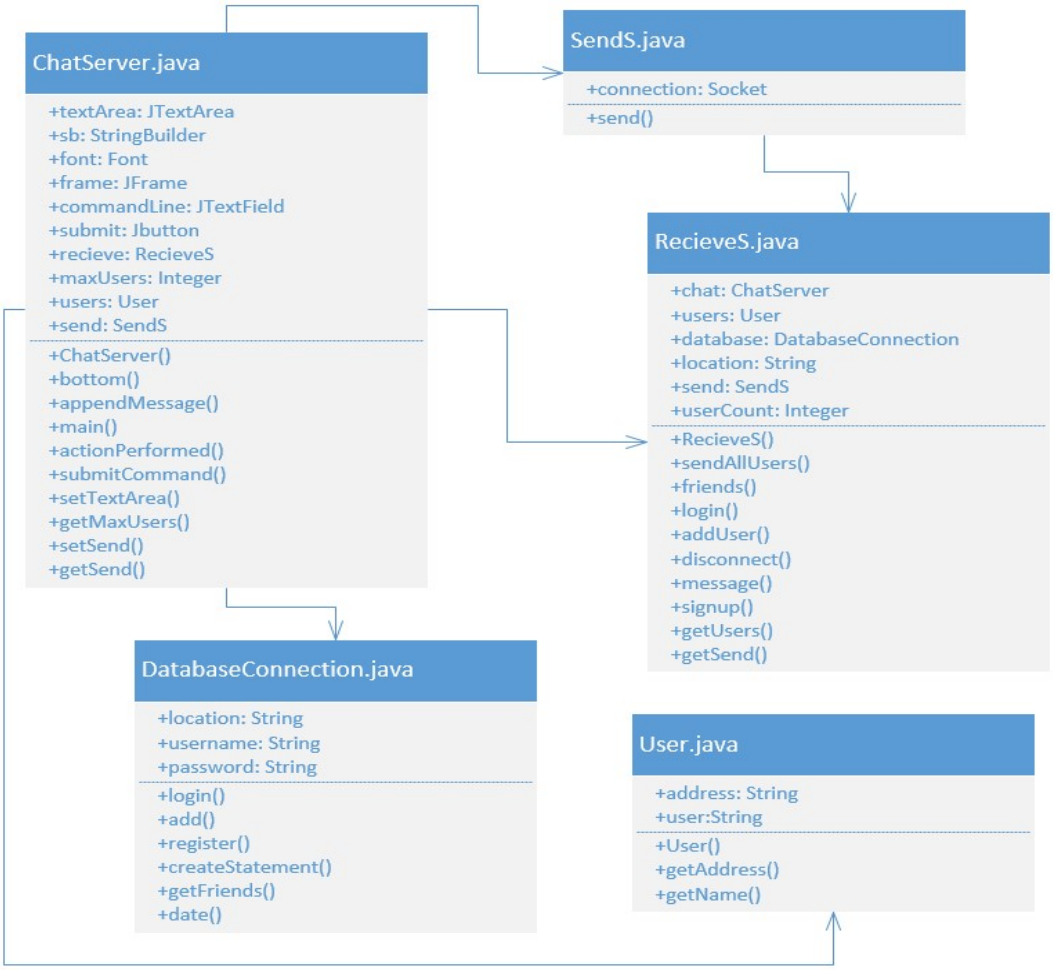


Figure 27: Chat Server class diagram

The messaging application needs to be in two parts, one part is the Chat Server, this part connects with the database and handles user login, sending messages and receiving messages (as seen in figure 28). The Chat Client part of the application allows the user to login and creates the chat interfaces needed for the exchange of messages. Other functionality includes; sending/receiving messages, maintaining friends lists and, most importantly, the SafeChat multi-agent system in launched from within the Chat Client application.

Figure 28: Chat Client classes

3.5 SafeChat Meeting Ontology Design

The first step in planning the ontology involved specifying the purpose, scope and goals of the ontology. Research suggested that the Methontology development method would be best suited to provide a robust, reusable ontology. The SafeChat system requires knowledge about the language of meetings. It more specifically, needs to be able to identify the components of a meeting, figure 29 identifies the elements that make up a meeting in conversation.

Next, the system needs to understand what these elements are and how they relate to each other. Using a top down approach, a taxonomy was developed for the meeting ontology and a snapshot of this can be seen in figure 30.

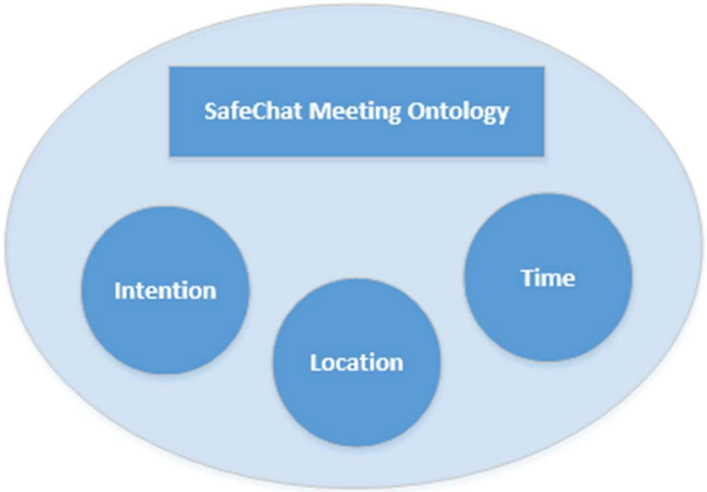


Figure 29: The components of a meeting

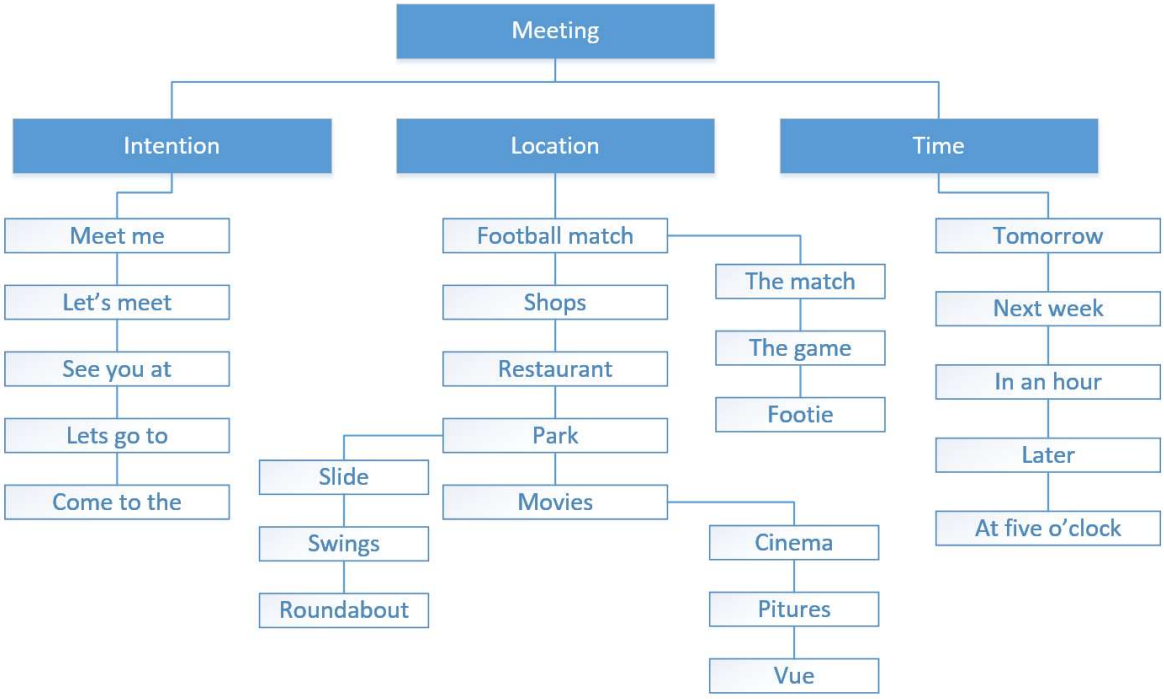


Figure 30: SafeChat ontology taxonomy

In developing the taxonomy, the relationships between classes begins to take shape and further planning can then take place to identify these relationships, as seen in figure 31. This is the last step before beginning production of the ontology. To test the robustness of the relationships, reasoners can be used to ensure that the relationships, classes and entities are defined appropriately, a small prototype ontology can be developed, so that time is not wasted.

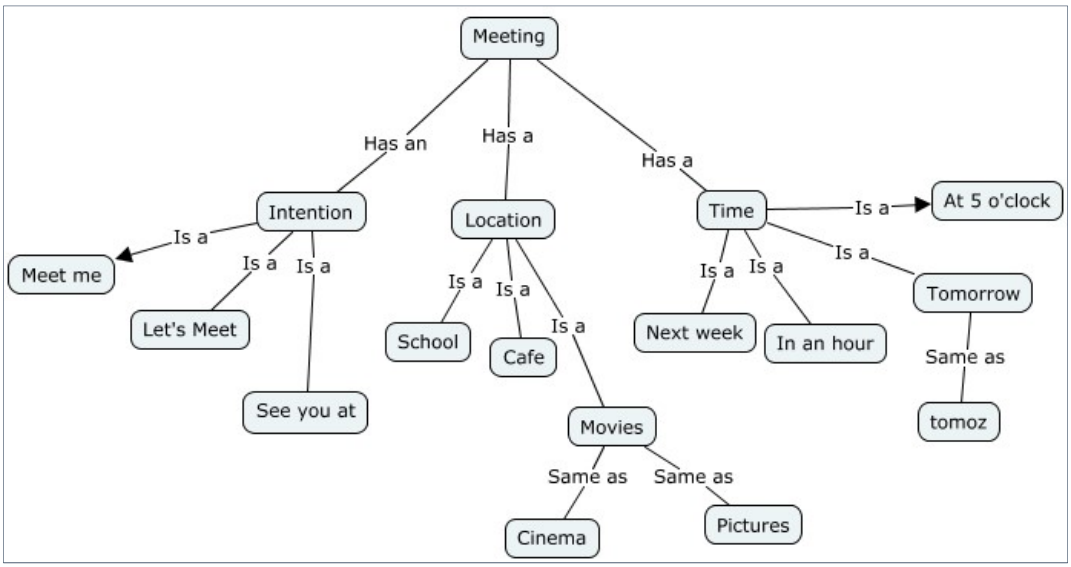


Figure 31: Relationships between entities and classes

3.6 Summary

This chapter discusses the design of the SafeChat system. It presents the different software design methodologies and articulates those chosen for each aspect of the system. The three main components of the overarching system require different approaches and these have been discussed as they apply to each part.

The multi-agent system design focuses on the role of each agent, their decision making and their utility. In cases where that utility has not been worth the cost of running an agent, as was the case with the WarningAgent, decisions can be taken at the design stage to address this. Figure 26 shows the route a message takes through the system, this gives the developer assurance that the system is taking the appropriate steps, but also highlights the possible overheads needed to ensure safety. By seeing this in a diagrammatical model, the risk to system transparency becomes clear, and steps can be taken at design stage to address this issue, where appropriate.

Initially, the messaging application was being developed as a stand-alone application, much like the PlayMessenger application discussed in Chapter 1 on page 19. Concurrent research and the surge in popularity of smart phones and social media applications meant that this initial idea had to be abandoned in favour of an approach that would provide SafeChat functionality to any messaging application. Efforts therefore focused on initialising the multi-agent system from the messaging application.

In designing the ontology, it was clear that selecting the classes and defining their relationships closely would be the key to producing a working, effective ontology. The use of diagrammatical tools in mapping these relationships proved to be crucial in defining these relationships correctly.

Chapter 4: Implementing the SafeChat System

4.1 Introduction

This chapter will describe the implementation of the SafeChat system components. It will discuss the development of the code and articulate decisions taken to ensure that system functionality is as intended. Problems encountered will be examined along with solutions developed to solve them. Images will be used extensively in this chapter to illustrate points of discussion.

4.2 The Agent System

The agent system was developed in the NetBeans IDE, using the Java programming language. In order for the agents to function properly the Java Agent Development (JADE) libraries must be imported into the project so that they can be called upon once the agents are instantiated from the agent class. As seen in figure 32, there are two libraries, the jade.jar and the commons-codec-1.3.jar.

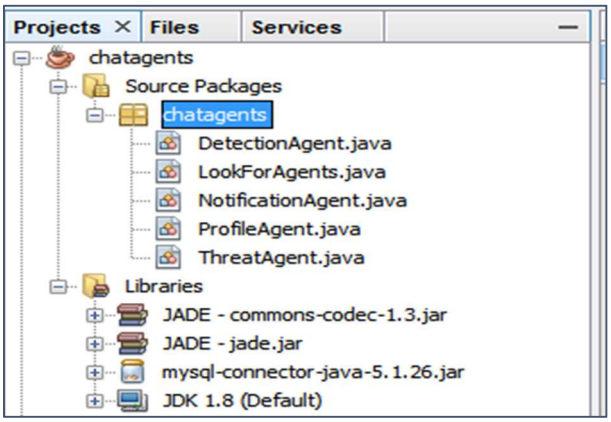


Figure 32: The agents created in the NetBeans IDE

Once the libraries are imported the agents can be developed and will call on them for a variety of functions. In figure 33, in code lines 8 through to 11 library components are being called to enable the java class to function as an agent in the JADE environment.

```
6   package chatagents;
7
8   import jade.core.*;
9   import jade.domain.DFService;
10  import jade.domain.FIPAAgentManagement.*;
11  import jade.domain.FIPAException;
12
13  public class LookForAgents extends Agent{
14
15          protected void setup(){
16      doWait(5000);//Allows previous agent time to load
17                  ServiceDescription sd=new ServiceDescription();
18                  sd.setType("utility");
19                  sd.setName(getLocalName());
20                  register(sd);
21                  try{
22                          DFAgentDescription dfd=new DFAgentDescription();
23                          DFAgentDescription[] result=DFService.search(this,dfd);
24
25                          System.out.println("Search returns: "+result.length+ " elements");
26                          if(result.length>0)
27                                  System.out.println(" "+result[0].getName());
```

Figure 33: LookForAgents code snippet

In NetBeans, agents can be launched by providing the appropriate runtime arguments in the project properties menu. The command *-gui* will ensure that the JADE GUI will boot so that agents can be tracked, checked, monitored or terminated. Each agent argument then needs a unique identifier or name and the argument for each agent is; *uniqueName:packageName.AgentName.* In figure 34 the SafeChat system agents are being instantiated all at once and the GUI is being loaded from the argument line.

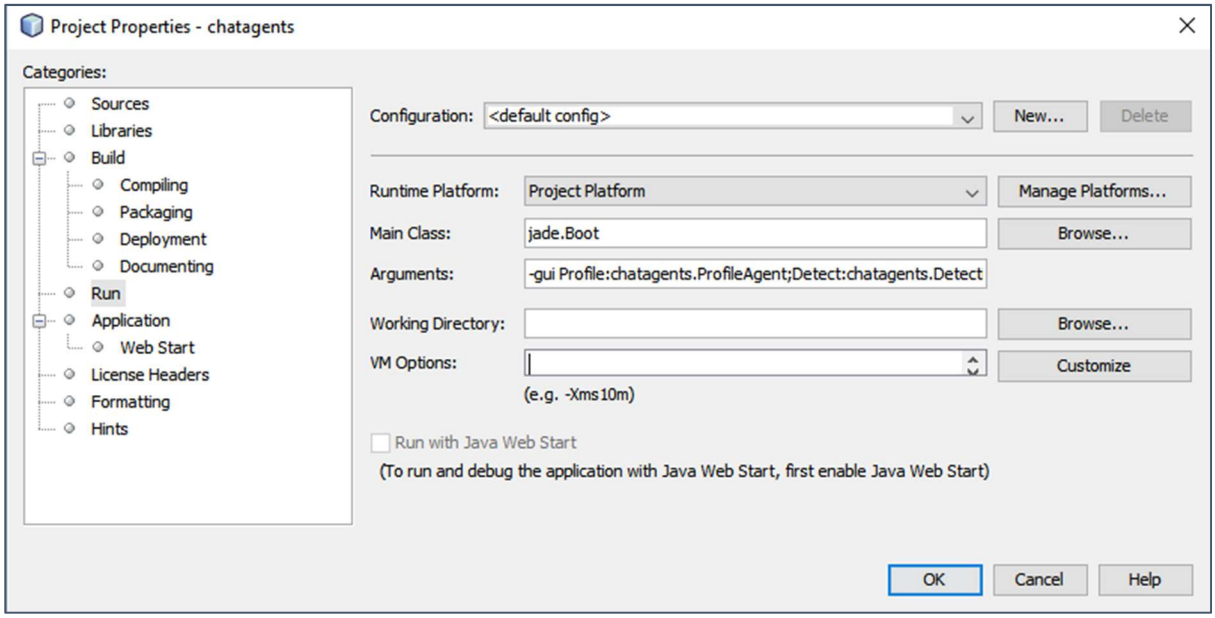


Figure 34: The runtime arguments to create an instance of each agent and launch the JADE GUI

Figure 35 shows the agents running in the JADE GUI with the unique names that were allocated in the runtime arguments.

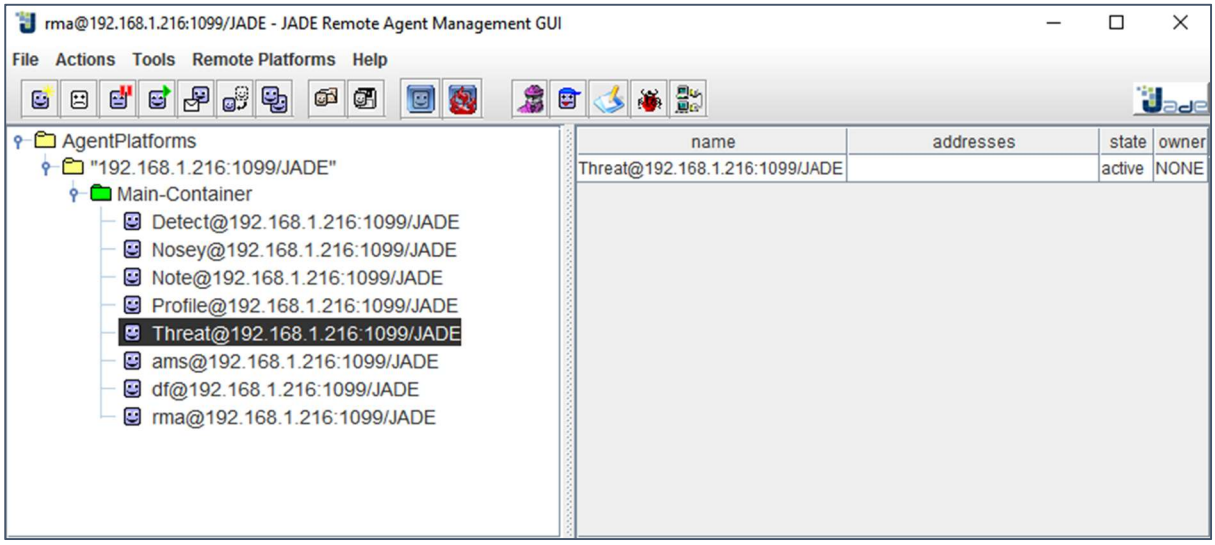


Figure 35: All agents running from the JADE GUI

All agents in the SafeChat system register their services with the Directory Facilitator, which acts as a registry for available services information for all agents so that other agents can find them and are aware of services offered. This also helps to check that the agents are behaving as expected.

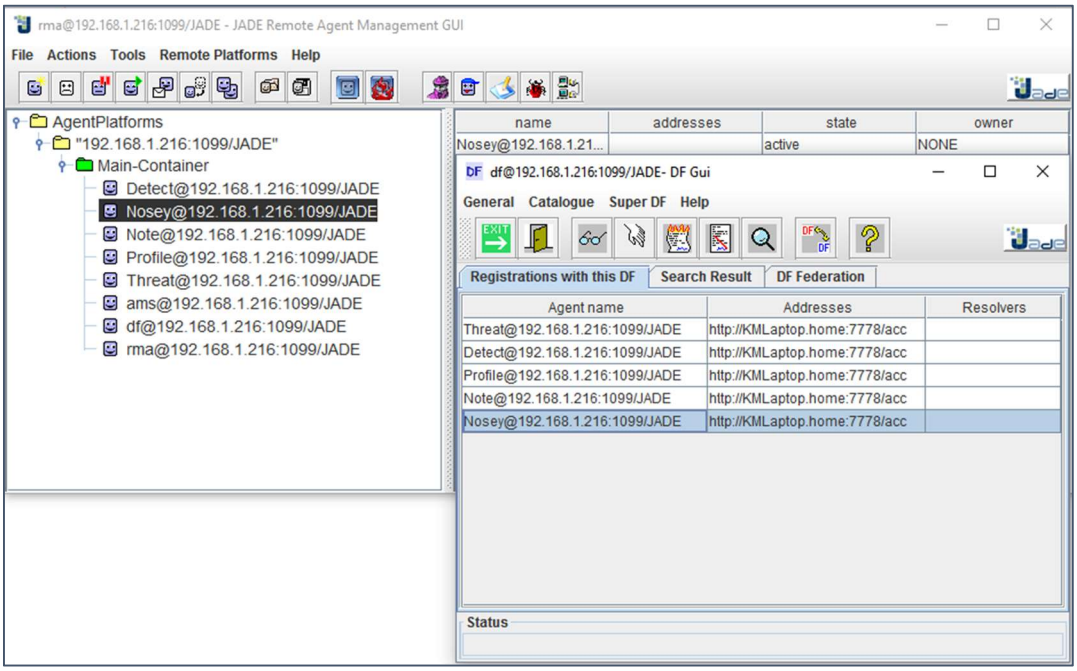


Figure 36: Agents successfully registered with the Directory Facilitator

Once the agents are setup and running, their individual functionality can be added. The LookForAgents agent is merely a housekeeping agent and therefore only the four active agents will be discussed in detail in this section.

The Notification Agent

This agent has absorbed the functionality of an earlier agent, the Warning agent. In the event of a user transmitting personal data, this agent displays a warning to the user and notifies the user's registered parent that a transgression has occurred.

To achieve this, the agent has a **MessageListener** class and the following methods;

- a **setup** method which defines the agents name, location and services
- a **registerService** method which registers the defined services with the directory facilitator
- an **action** method which receives messages and performs actions based on their content, seen in figure 37

```
63      @Override
64        public void action() {
65          ACLMessage msg1 = receive();
66          if (msg1!=null){
67                  String test=msg1.getContent();
68                  if (test.equals("bad")){
69                  doWait(2000);//we sleep here to give agents a chance to start.
70                  System.out.println("**********************************************************************");
71                  System.out.println("  "+getLocalName()+" sends:");
72              System.out.println("  Dear Parent,");
73              System.out.println("  Your child tried to send a message today that had some personal");
74              System.out.println("  information in it, might be a good time to have a chat about the");
75              System.out.println("  dangers of this, thanks, The Safe-Chat Team!");
76              System.out.println("**********************************************************************");
77              doWait(500);
78              System.out.println("**********************************************************************");
79              System.out.println("  "+getLocalName()+" sends:");
80              System.out.println("  Dear User,");
81              System.out.println("  Sorry your last message could not be sent it appeared");
82              System.out.println("  to contain unsafe content and that is not allowed");
83              System.out.println("**********************************************************************");
84                  ACLMessage msg3 = new ACLMessage(ACLMessage.INFORM);
85                  msg3.addReceiver(new AID("Warn", AID.ISLOCALNAME));
86                  msg3.setLanguage("English");
87                  msg3.setContent("warn");
88                  send(msg3);
```

*Figure 37: The notification agent **action** method*

The Detection Agent

This agent will receive messages from the instant messaging application and from the ontology and send the content to the profile agent and the threat level agent. For the purposes of testing, messages have been hard coded into this agent so that system functionality can be thoroughly tested.

This agent is made up of the following components;

- a **setup** method which defines the agents name, location and services
- a **SimpleBehaviour** which allows messages to be sent to the other agents in the system
- two **action** methods**,** one to send a message to the profile agent, whilst the other sends a message to the threat agent, seen in figure 38
- a **registerService** method which registers the defined services with the directory facilitator

```
63      @Override
64        public void action() {
65          ACLMessage msg1 = receive();
66          if (msg1!=null){
67                  String test=msg1.getContent();
68                  if (test.equals("bad")){
69                  doWait(2000);//we sleep here to give agents a chance to start.
70                  System.out.println("**********************************************************************");
71                  System.out.println("  "+getLocalName()+" sends:");
72              System.out.println("  Dear Parent,");
73              System.out.println("  Your child tried to send a message today that had some personal");
74              System.out.println("  information in it, might be a good time to have a chat about the");
75              System.out.println("  dangers of this, thanks, The Safe-Chat Team!");
76              System.out.println("**********************************************************************");
77              doWait(500);
78              System.out.println("**********************************************************************");
79              System.out.println("  "+getLocalName()+" sends:");
80              System.out.println("  Dear User,");
81              System.out.println("  Sorry your last message could not be sent it appeared");
82              System.out.println("  to contain unsafe content and that is not allowed");
83              System.out.println("**********************************************************************");
84                  ACLMessage msg3 = new ACLMessage(ACLMessage.INFORM);
85                  msg3.addReceiver(new AID("Warn", AID.ISLOCALNAME));
86                  msg3.setLanguage("English");
87                  msg3.setContent("warn");
88                  send(msg3);
```

*Figure 38: The detection agent **action** method sending a message to the threat agent*

The Profile Agent

The profile agent maintains a record the user's personal details, again in the prototype system this has been hardcoded for ease of testing, but in a live application setting, this information would be drawn from the details stored in the messaging application, or set up by the parent. This agent checks the content of received messages against the stored details and then sends a message to the notification agent based on the outcome of the test.

```
69      @Override
        public void action() {
71        address = "47 green street";//these values would be retrieved from chat application when live
72        phone = "01254292929";
73        email = "kate@kate.com";
74        name = "katrinna macfarlane";
75
76         doWait(2500); //we sleep here
77         ACLMessage msg = receive();
78         //System.out.println(msg);
79           if (msg!=null){
80                   String test=msg.getContent();
81                   if (test.equals(address)){
82                   ACLMessage msg1 = new ACLMessage(ACLMessage.INFORM);
83                   msg1.addReceiver(new AID("Note", AID.ISLOCALNAME));
84                   msg1.setLanguage("English");
85                   msg1.setContent("bad");
86                   send(msg1);
87                   finished = true;
88           }
```

*Figure 39: The profile agent **action** method which compares the message content with user details*

This agent contains the following code components;

- a **setup** method which defines the agents name, location and services
- a **registerService** method which registers the defined services with the directory facilitator
- a **MessageListener** class which allows messages to be received by the agent
- an **action** method which has been designed to cycle through user details to look for any matches and take actions based on the result of the test. If the message does not match any stored details it is allowed to progress and no action is taken. If a match is found, the message is blocked, seen in figure 39.

The Threat Agent

This agent provides two main services, it calculates and maintains the current threat level for a given conversation and based on that, provides feedback to the user on the safety level of a given conversation. It receives messages from the detection agent.

This agent contains the following code components;

- a **setup** method which defines the agents name, location and services

- a **registerService** method which registers the defined services with the directory facilitator
- a **MessageListener** class which allows messages to be received by the agent
- an **action** method which "listens" or waits for a message to be received, once a message is received the agent converts the string into a numerical value and then breaks the numerical value down to ascertain the presence and type of meeting elements recorded in the code. Once this has been done the threat level is calculated and the safety level is generated based on this calculation. This information is then passed to a file so that the current threat level can be recorded, and the safety level can be displayed to the user.

Figure 40 shows a part of the code responsible for receiving a message in string format. The *integer. parseInt* instruction (on code line 71) converts the content of the string into an integer value. This integer value is then broken down into separate numerical values, with each number signifying the presence, or otherwise of a meeting element. The message was originally made up of three characters in a binary format. For example, 010 or 111, each character represents a meeting element with 1 indicating the presence of an element and 0 meaning this particular element is not present.

```
59      public class MessageListener extends SimpleBehaviour {
60
61          private boolean finished = false;
62              public MessageListener(Agent agent){
63              super(agent);
64          }
65          @Override
              public void action() {
67              ACLMessage msg = receive();
68                //System.out.println(msg);
69                if (msg!=null){
70                          String msg1=msg.getContent();
71                          int msgCon1 = Integer.parseInt(msg1);
72                              System.out.println("The Message recieved was " +msgCon1);
73                      // The meeting element variables are declared below:
                          int testT = 0;
                          int testLoc = 0;
                          int testTim = 0;
                          int testInten = 0;
78      // Here we take the integer value and separate it into its component values
79      // This is why we have to add the CheckBit value - the number cannot start with a 0
80       //so a CheckBit with a value of 3 has been inserted
81                  testInten = (msgCon1 % 10);
82                  testTim = (msgCon1 % 100)/ 10;
83                  testLoc = (msgCon1 % 1000)/ 100;
84                   testT = (msgCon1 % 10000)/ 1000;
85       //Here we display the values to screen for testing purposes
86      System.out.println("********************************************************************** ");
87       System.out.println("Checkbit:" + testT + " Location:" + testLoc + " Time:" + testTim + " Intention:" + testInten);
88        System.out.println("********************************************************************** ");
```

Figure 40: Conversion of string to integer and allocation of meeting element values

Initial calculations using the three-digit message, or code produced some success, but if the leading element was not present and the message started with a zero, the calculation did not work. This is because a mathematical calculation is being used on a numerical value to break it down into separate values. This means that the zero will be discounted by the programme as though it does not exist and an incorrect result is returned. To overcome this a simple check bit has been included at the start of the code with a value of 3. This ensures that the number never starts with a zero and the correct values are allocated to each element.

```
122     if (threatLevel <= 1) {
123           safeLevel = "Green";
124         }
125         else if (threatLevel <= 2){
126         safeLevel = "Amber";
127         }
128         else{
129         safeLevel = "Red!";
130         }
131       System.out.println("************************************************************************** ");
132       System.out.println("The current Safety Level for this conversation is: " + safeLevel);
133       System.out.println("************************************************************************** ");
```

Figure 41: Creation and calculation of safety level for user

Figure 41 shows the code snippet that allocates a safety level for the current conversation. This could be communicated to the user to help them monitor their own safety.

Refining the Decision-Making Process

In the SafeChat system, the threat level agent needs to monitor messages and conversation and calculate a level of threat based on the presence of elements that make up a meeting arrangement. Currently it does this by assigning a numerical value of each meeting element. This is because in conventional conversation, we could see locations and times being discussed frequently, as people talk about their day they would note the timing of events as naturally as they would the events themselves. The sample conversation in figure 42 is an example of this:

Dan: You will never guess what happened today
Sara: Oh?
Dan: Well **this morning** Jack came into school and told everyone he got picked for the football team
Sara: Wow that's great news
Dan: Yes, his dad took him to **Nando's** because that's where they went on his birthday and they asked if he **wants to go to** Disneyland on holiday

Figure 42: Sample Conversation

A human could quite easily see that this conversation did not contain a meeting arrangement and identify as safe to let this conversation continue. As long as the intention part of an ontology was carefully coded the computer would detect a location and a time in this conversation. If the system elevates the threat for every instance of a location and time element, the system would report false positives, figure 43 shows a sample conversation that has two locations and an intention.

Joe: Hi there, how are you today?

Betty: I'm ok - bored though

Joe: really - what do you feel like doing

Betty: I **want to go to McDonalds** but no one will take me

Joe: that's too bad - hope you feel better soon

Betty: what are you doing today?

Joe: I'm going to the **movies** with my dad

Betty: aww I **want to go** too

Joe: Got to run - see you later

Figure 43: Meeting elements in the sample conversation

If we counted every element when used the threat level would continue to grow. If this was the case, the system would take action against the conversation in figure 41, but this would be a false positive.

```
99    if ((testLoc + lastLoc)<= 0){
100         location=0;
101         }
102         else{
103         location=1;
104         }
105         if ((testTim+ lastTime )<= 0){
106         time=0;
107         }
108         else{
109          time=1;
110         }
111         if ((testInten+lastInten )<= 0){
112         intention=0;
113         }
114         else{
115         intention=2;
116         }
117                 // This part of the code sets the current threat level of the conversation
118                 int threatLevel;
119
120                 threatLevel = (location+time+intention);
```

Figure 44: Threat level calculation

Without some way of further refining the decision-making process the system would likely encounter a high number of false positives and be rendered all but useless. The whole premise is for the system

to be protective without being restrictive, as identified previously, if the system proved to be restrictive or problematic for users, they would stop using it or look for more open, unsafe alternatives.

The system is set therefore, to record the last known threat level at an element level, so that it knows if a particular element has been present in the conversation previously. And the system is coded not to increase the threat level when another element of the same kind is recorded.

Figure 44 is the code which controls the threat level calculation, the time and location elements are set to a value of 1 if they are present, even if they were present previously. **testLoc** is the current location value and **lastLoc** is the previous location value in code line 99, regardless of both values, the result will be a value of 1 if a location has been discussed. Finally, the intention element carries a higher value that the other meeting elements. This is because an intention is a definite statement about a meeting and cannot relate to erroneous discourse, so when an intention is present the threat will always be greater. By allocating the value of 2 to the intention element we ensure that the safety level of the conversation is set to the value red whenever it is accompanied by another element and amber when it is the lone element present.

4.3 The Messaging System

Initial attempts at creating a messaging application failed, because they were being created in the same environment as the chat agents. A messaging application has been developed in its own environment to enable testing of the system. The application is in two parts; the chat server and the chat client, as seen in figure 45.

The chat application provides a range of functionality; it incorporates a database to register and authenticate users. For this prototype system, a local database has been configured to enable testing and ease of validation using XXAMP. The system also provides functionality for a contacts or friends list and sending/receiving messages. Development of the chat system ceased when it became clear that focus needed to shift from a single child safe application, to a system that could be adapted or plugged in to any online communication medium.

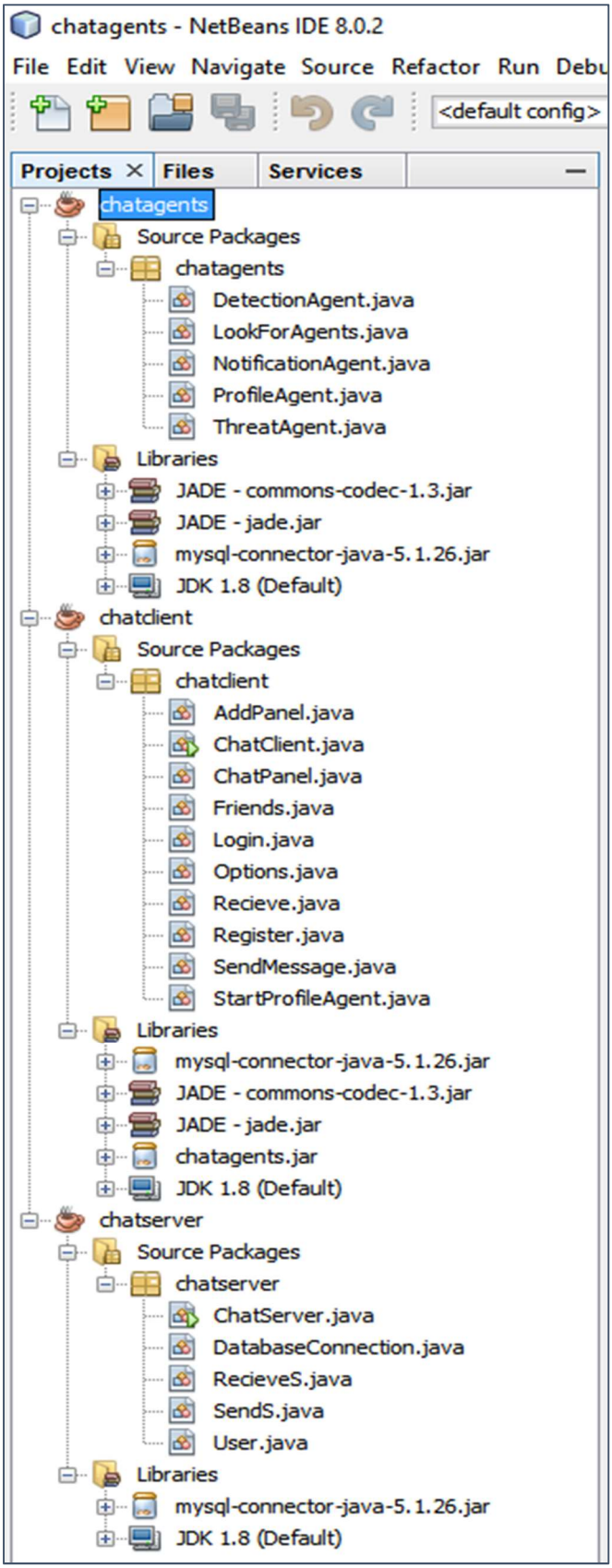


Figure 45: All SafeChat java applications running independently

The Chat Server

The chat server application needs to be running before the chat client can run. The client logs on to the server and the server application opens the connection to the user database to authenticate users. It is made up of five java classes, these are;

- the **ChatServer** class is the main class and instantiates the application, defines the server interface (seen in figure 46) and calls upon functionality from other classes, such as **SendS.**
- the **DatabaseConnection** class opens the connection to the database, populates the database with new user details and updates and retrieves the friend or contacts information from the database.
- the **ReceiveS** class enables the chat server to receive messages
- the **SendS** class handles outgoing messages
- the **User** class manages information about currently logged in users

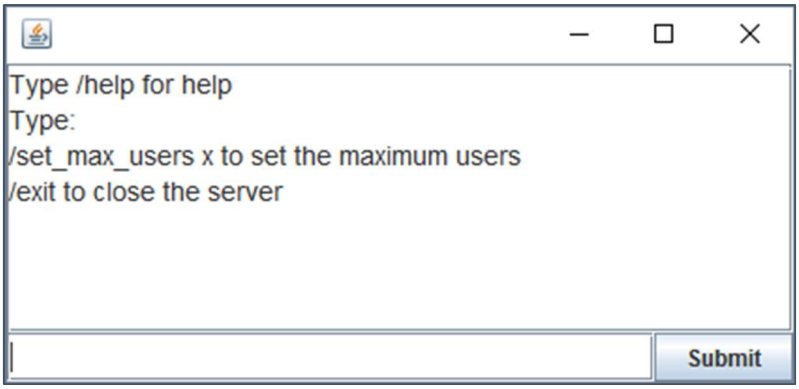


Figure 46: The chat server interface

The Chat Client

The chat client allows users to log on, register and communicate with their friends. Initially this was to be developed further and be the only application that would offer the SafeChat multi-agent system protection, but this application still provides a valuable resource for testing and developing the system as a whole. It contains the following java classes;

- the **AddPanel** class controls elements of the client user interface
- the **ChatClient** class is the main class and controls other elements of the user interface and calls upon methods from the other classes in the chat client application
- the **ChatPanel** class controls some user interface elements, updates and refreshes the contents of the chat window and friend's list
- the **Friends** class manages messages between friends

- the **Login** class connects to the database to authenticate users and in the event of a successful login, instantiates the SafeChat multi-agent system
- the **Options** class provides an options interface
- the **Receive** class handles incoming messages
- the **Register** class handles new user registration, it populates the database with user details and provides a layer of data validation
- the **SendMessage** java class handles all outgoing messages
- the **StartProfileAgent** class creates the start-up environment for the SafeChat agents and calls the agent from the chatagents.jar, seen in figure 47

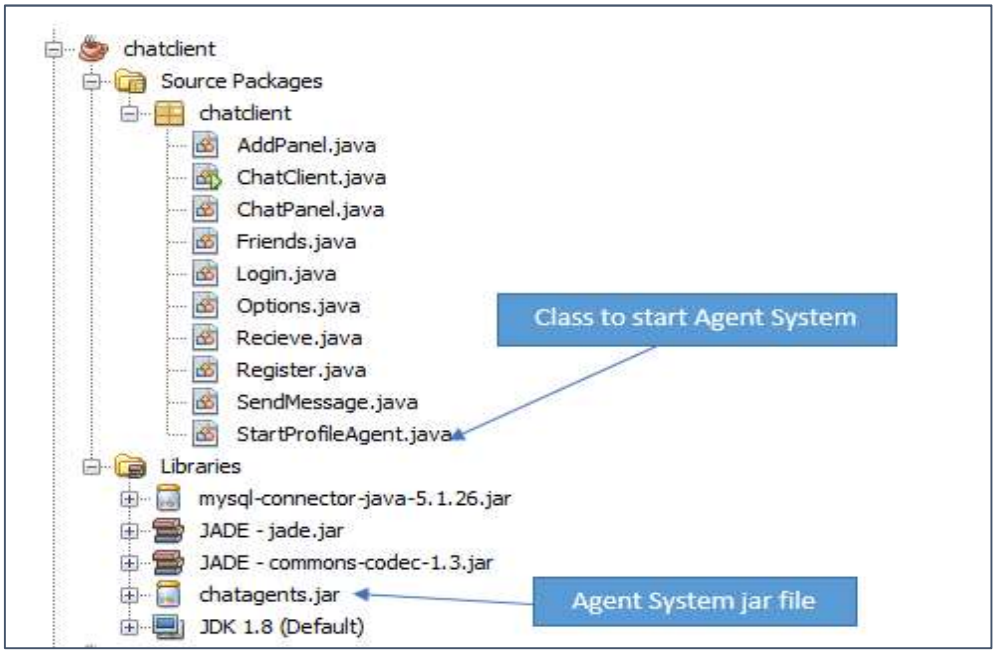


Figure 47: Chat client setup to instantiate the SafeChat agents

4.4 Implementing the Ontology

Protégé has been selected as the best tool to design and implement an ontology solution for the Safe-Chat system. It offers a package of tools and add-ons to construct ontology solutions using the Web Ontology Language (OWL) W3 (2010: Online) developed as a vocabulary extension to the Resource Development Framework (RDF) (MacFarlane and Holmes, 2009). OWL is a semantic mark-up language developed by the WC3 to allow the sharing and publication of ontology solutions on the Internet (Horridge *et al.* 2004).

Creating ontology in Protégé involves creating its classes and subclasses and then defining the relationships between those classes.

A meeting arrangement is made up of three elements; a **time**, a **location** and an **intention**, these elements would then make up the main classes in the ontology, seen in figure 48.

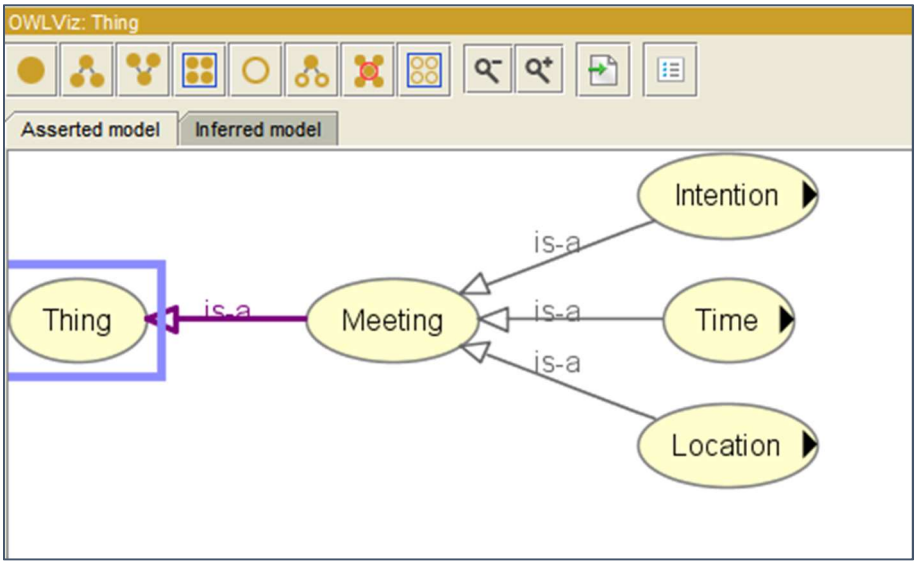


Figure 48: The three element classes with their relationships

Setting up the location class is more challenging, but it is possible to add fine grain definition by adding more subclasses. For example, it can be specified that a location can have the subclass ***Cinema***, and then go on to add that Cinema can also have the subclasses of all other terms that describe a cinema including brand names and slang terms like ***movies***, ***pictures*** and ***Apollo***. Similarly, the location ***restaurant*** can have the subclasses ***cafe, diner*** and ***McDonalds***. Each subclass can be further broken down into further subclasses (MacFarlane and Holmes, 2009). MacDonald's is an example of this in figure 49.

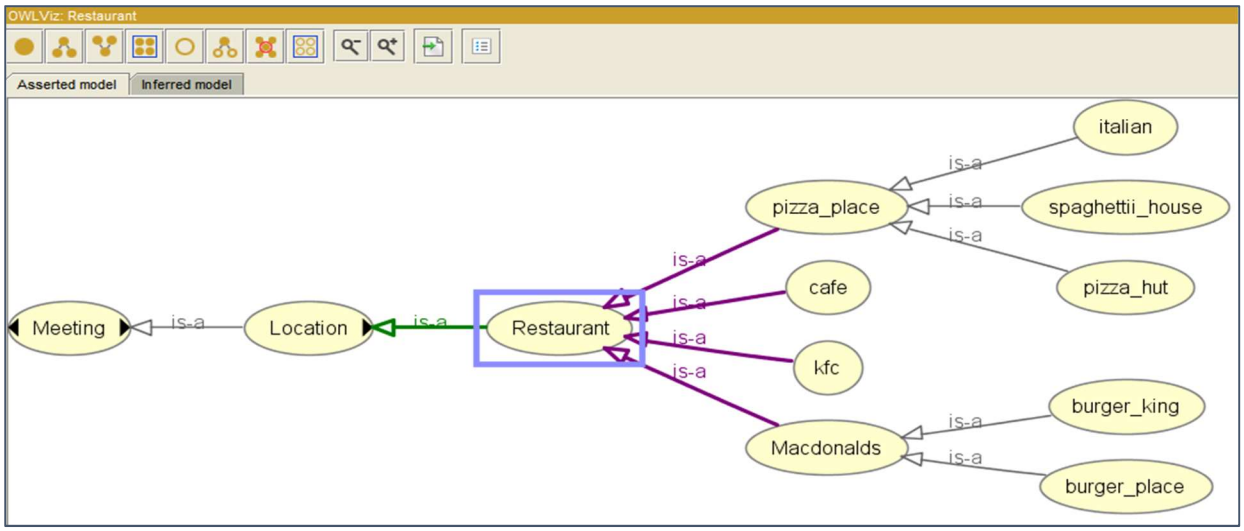


Figure 49: Here subclasses have been added to the location class

An intention can be made up of many phrases, for example "let's meet", "meet me" or "can I see you?" This class and the time class are trickier to construct as words are used in phrases, rather than one word identifiers, as seen in figure 50.

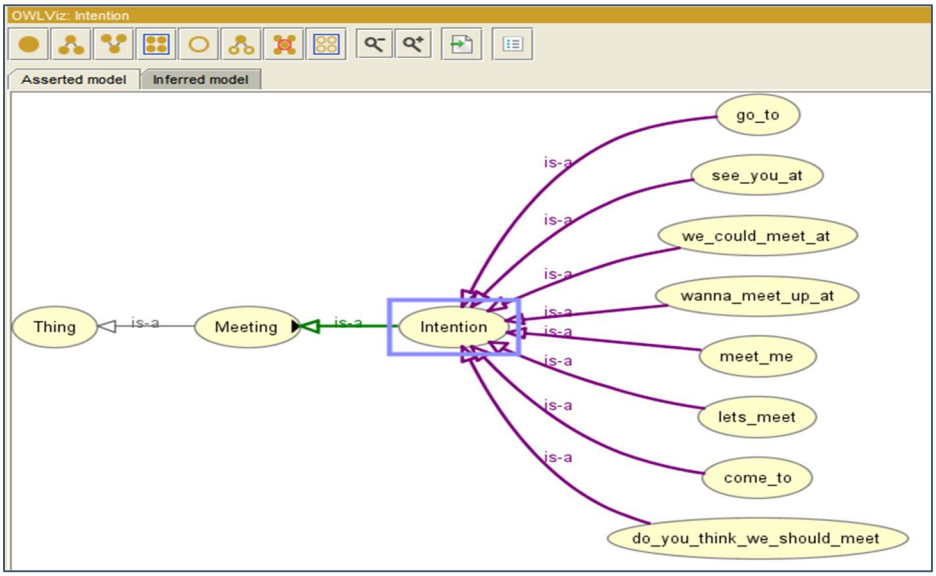


Figure 50: The Intention subclass

Another complication when it comes to interacting with natural language is that we often have several ways of saying the same thing. This is further complicated when the use of colloquial slang is introduced into the vocabulary. We can overcome this by applying relationships accurately (as can be seen in the time class in figure 51) and including known slang and abbreviations in the ontology.

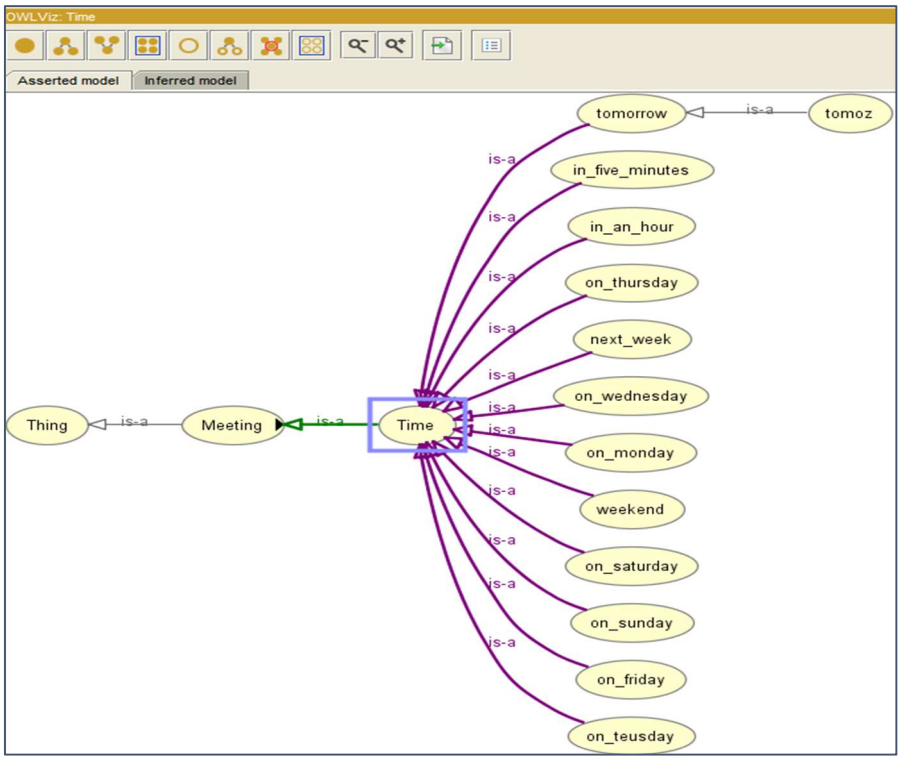


Figure 51: The time class

Given the range of relationships, and the complicated structure (seen in figure 52), it is important to maintain a logical, informed decision making process for the agents using the ontology. To check the

logic applied to the ontology, a reasoner can perform a logic test on the ontology, Protégé comes with several built-in reasoners and they have been developed for different kinds of ontologies.

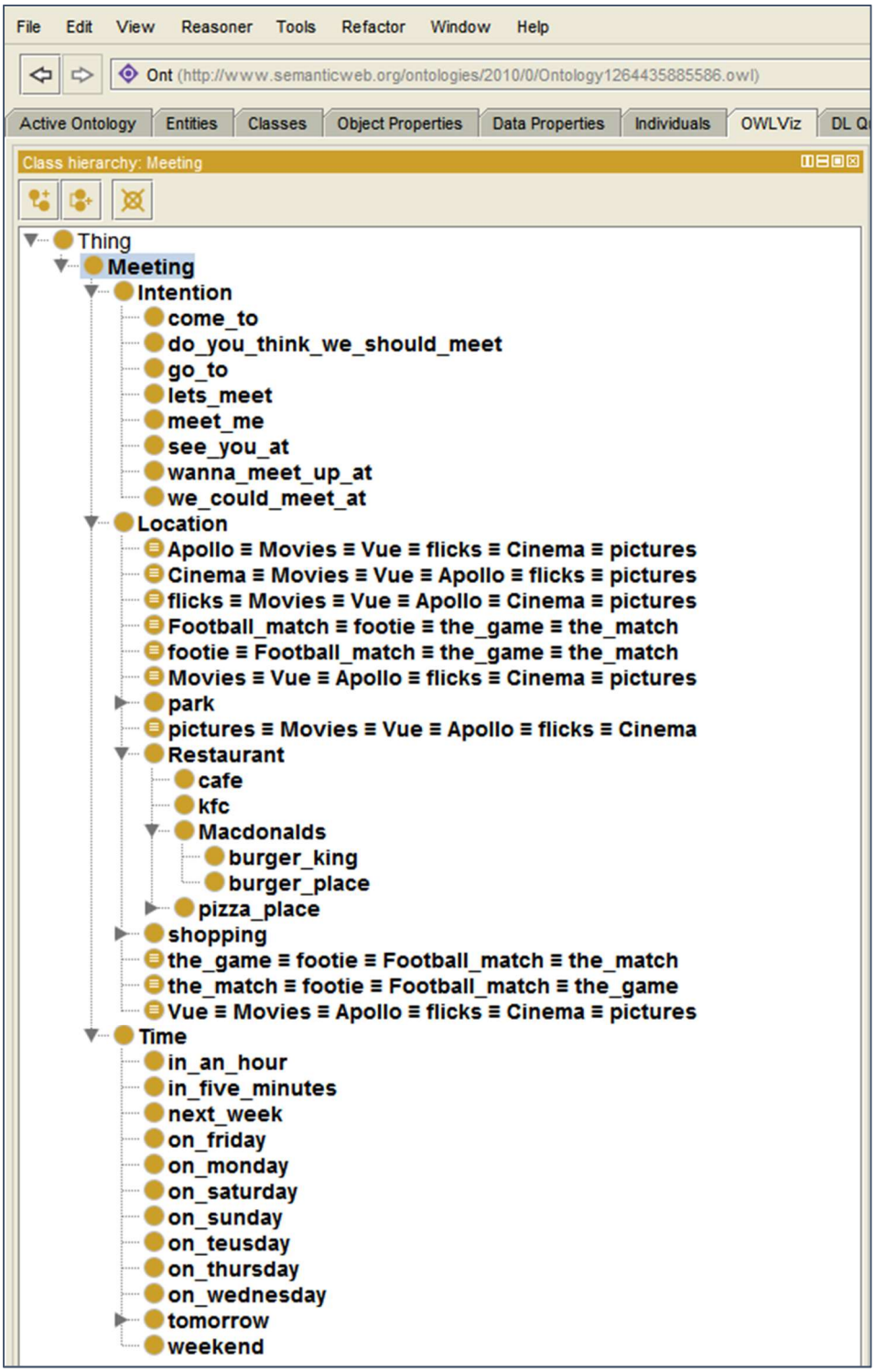


Figure 52: The SafeChat Meeting Ontology classes

4.5 Summary

This chapter examined the implementation of the three sub-systems that make up the SafeChat system. The agents within the multi-agent system have been presented and their respective methods were discussed and their usage demonstrated. The agent framework is presented and this illustrates the environment that the agents inhabit once instantiated. Work on refining the agent decision making process is also discussed in this chapter. Initially the agents had a three-digit code to work with, but this was problematic and the solution identified to overcome this problem has been presented. The weightings for each of the elements within the meeting ontology could not be equal, because the way language is used must be considered.

The messaging system is presented, and although it is now not necessary for the SafeChat system, working on it and developing a working system helped inform the multi-agent system development, and provided insight into how these systems operate.

The last sub-section of this chapter presents the implementation of the ontology design. Here, Protégé has been used to articulate a meeting, using classes, sub-classes and relationship definition.

Chapter 5: Testing and Results

5.1 Introduction

This chapter will focus on the results of testing each component part of the SafeChat System. In cases where initial results did not match the desired or expected outcome in the first instance, this will be reported alongside steps taken to resolve the issues.

5.2 SafeChat Multiagent System Results

This part of the thesis will focus on presenting the results of testing each aspect of the multi-agent system, from launching the agents to individual agent performance.

Launching the Agents

All agents launch successfully when the system is started (figure 53), they all specify their name and location information as expected. The only issues encountered with this part of system performance were encountered early on in the development stage, when the java integrated development environment (IDE) was changed from JCreator to NetBeans. This was a deliberate move to a more industry standard IDE, JCreator did not offer as much integration support as NetBeans and agents had to be started from command line driven batch files. The move to NetBeans meant that agent start-up arguments could be developed and defined from within the IDE.

The java files compile successfully and when the project runs the JADE user interface is triggered as expected and can be used to verify that the agents have registered with the directory facilitator.

Agent Functionality

A message is sent from the Detection agent to test the functionality of the Profile agent; this message can be changed to test weather personal details are successfully blocked or are allowed through. Initial problems were encountered when all agents ran at the same time, some outputs became garbled as all agents worked at the same time. This was resolved by using the doWait (); java command and carefully cascading the running order of the agents.

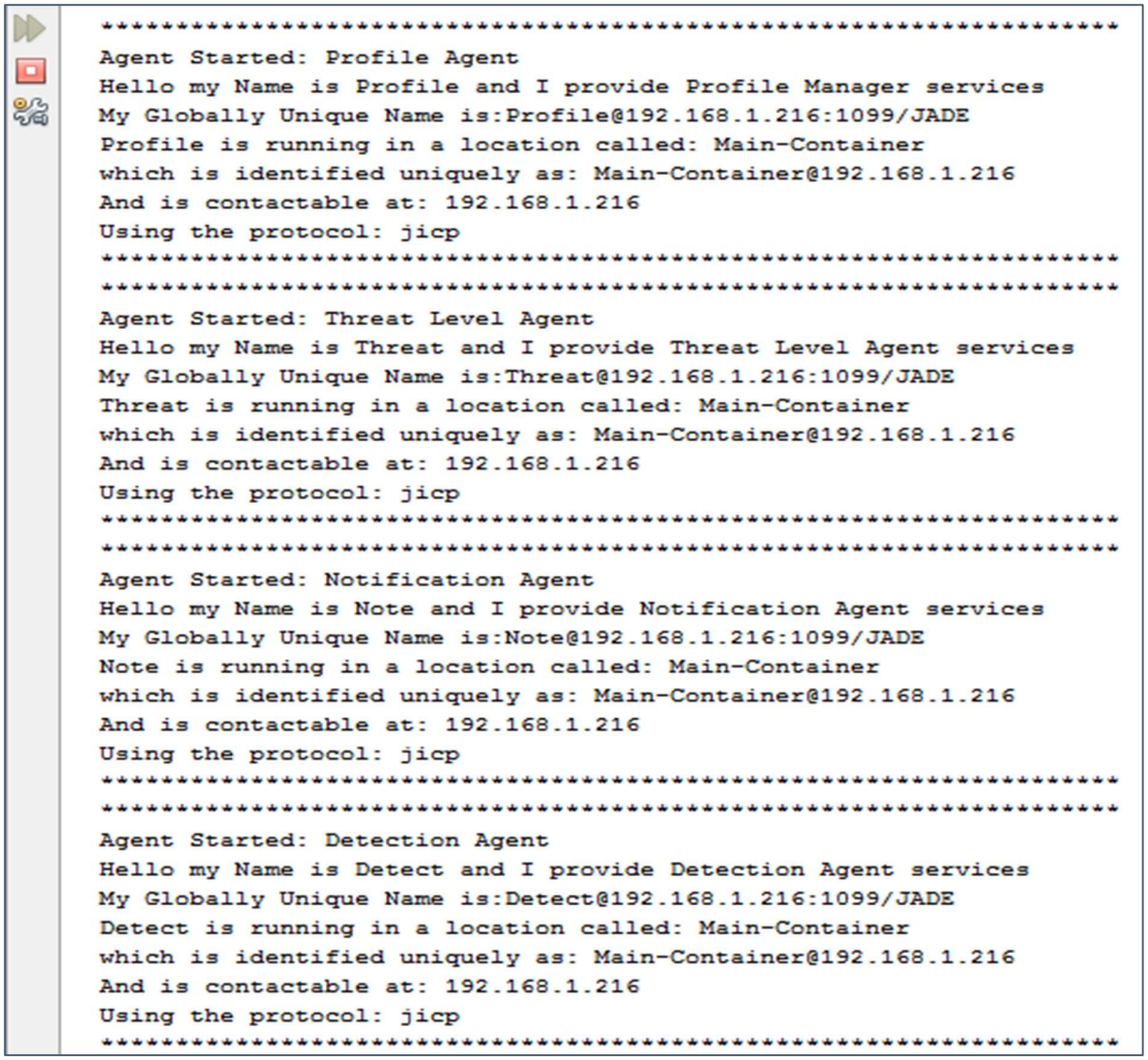


Figure 53: All agents running successfully

If the message contains personal data, it is successfully blocked and the Notification agent successfully receives the appropriate instructions to warn the user and nominated parent, as seen in figure 54. If the message does not contain personal details then the message is allowed thorough and the Notification agent does not take action, this can be seen in figure 55.

```
**********************************************************************
Detect: Message Sent(INFORM
 :sender  ( agent-identifier :name Detect@192.168.1.216:1099/JADE  :addresses (sequence http://KMLaptop.home:7778/acc ))
 :receiver  (set ( agent-identifier :name Profile@192.168.1.216:1099/JADE ) )
 :content  "47 green street"
)
**********************************************************************
  Profile sends:
  The message was bad:BlockMessage
  The message was (INFORM
 :sender  ( agent-identifier :name Detect@192.168.1.216:1099/JADE  :addresses (sequence http://KMLaptop.home:7778/acc ))
 :receiver  (set ( agent-identifier :name Profile@192.168.1.216:1099/JADE ) )
 :content  "47 green street"
)
**********************************************************************
**************************************************
  Note sends:
  Dear Parent,
  Your child tried to send a message today that had some personal
  information in it, might be a good time to have a chat about the
  dangers of this, thanks, The Safe-Chat Team!
**************************************************
****************************************************************************************************
  Note sends:
  Dear User,
  Sorry your last message could not be sent it appeared
  to contain unsafe content and that is not allowed, if you need more information please visit
  https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/177099/DFE-00004-2012.pdf
****************************************************************************************************
```

Figure 54: Output if the message contains personal details

Test results for the threat level agent are encouraging. The message is successfully received from the Detection agent and takes the format of a four-digit string (e.g. 3010). The Threat agent then converts this to a numerical value and then performs a mathematical calculation to break 3010 down into single digit values for the check bit, location, time and intention variables. Figure 56 shows this is being done accurately by the prototype system.

```
**********************************************************************
Detect: Message Sent(INFORM
 :sender  ( agent-identifier :name Detect@192.168.1.216:1099/JADE  :addresses (sequence http://KMLaptop.home:7778/acc ))
 :receiver  (set ( agent-identifier :name Profile@192.168.1.216:1099/JADE ) )
 :content  "My dog is called spot"
)
  Profile sends:
  The message was good:No action taken!
  The message was My dog is called spot
**********************************************************************
```

Figure 55: Output if the message does not contain personal details

```
**********************************************************************
Detect: Message Sent(INFORM
 :sender  ( agent-identifier :name Detect@192.168.1.216:1099/JADE  :addresses (sequence http://KMLaptop.home:7778/acc ))
 :receiver  (set ( agent-identifier :name Threat@192.168.1.216:1099/JADE ) )
 :content  "3010"
)
The Message recieved was 3010
**********************************************************************
Checkbit:3 Location:0 Time:1 Intention:0
**********************************************************************
```

Figure 56: Output of the threat agent message calculation

Once the threat agent has broken the message down correctly it then allocates a value to the current time, location and intention variables. This is based on not only the current value of each meeting element, but also on the last known element in the conversation. This is important because the meeting arrangements may take place over several conversation sessions, and would certainly involve more than one message. The threat agent is also responsible for setting a safety level for the current conversation, figure 57 shows that outputs are changing appropriately and that the threat and safety levels are working as intended.

```
The Message recieved was 3000
**********************************************************************
Checkbit:3 Location:0 Time:0 Intention:0
**********************************************************************
**********************************************************
The current Threat Level for this conversation is: 0
**********************************************************
**********************************************************************
The current Safety Level for this conversation is: Green
**********************************************************************
```

```
The Message recieved was 3011
**********************************************************************
Checkbit:3 Location:0 Time:1 Intention:1
**********************************************************************
**********************************************************
The current Threat Level for this conversation is: 3
**********************************************************
**********************************************************************
The current Safety Level for this conversation is: Red!
**********************************************************************
```

```
The Message recieved was 3010
**********************************************************************
Checkbit:3 Location:0 Time:1 Intention:0
**********************************************************************
**********************************************************
The current Threat Level for this conversation is: 2
**********************************************************
**********************************************************************
The current Safety Level for this conversation is: Amber
**********************************************************************
```

```
The Message recieved was 3111
**********************************************************************
Checkbit:3 Location:1 Time:1 Intention:1
**********************************************************************
**********************************************************
The current Threat Level for this conversation is: 4
**********************************************************
**********************************************************************
The current Safety Level for this conversation is: Conversation Terminated
**********************************************************************
```

Figure 57: Output of the threat agent threat and safety levels

The threat level agent successfully creates a meet.txt file to record the last known threat levels for each element. This can be used to inform the agents decision making process.

5.3 SafeChat Chat Application Results

Once the XXAMP application launches the apache server and the MySQL server, the chat server application can be started, as seen in figure 58.

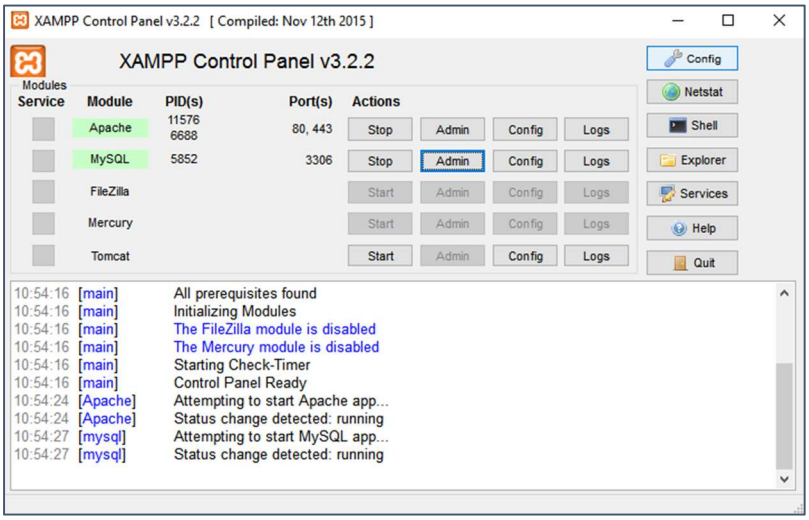


Figure 58: XXAMP control panel

This needs to be running because the chat server application needs to connect to the MySQL database to authenticate users. Once this is running, the chat client application can be started. Figure 59 shows the chat interface that appears when a validated user logs into the application. It also shows the chat server monitoring messages being sent through the chat interface. This is where message content would be captured by the SafeChat System as it records the message sender, receiver and content.

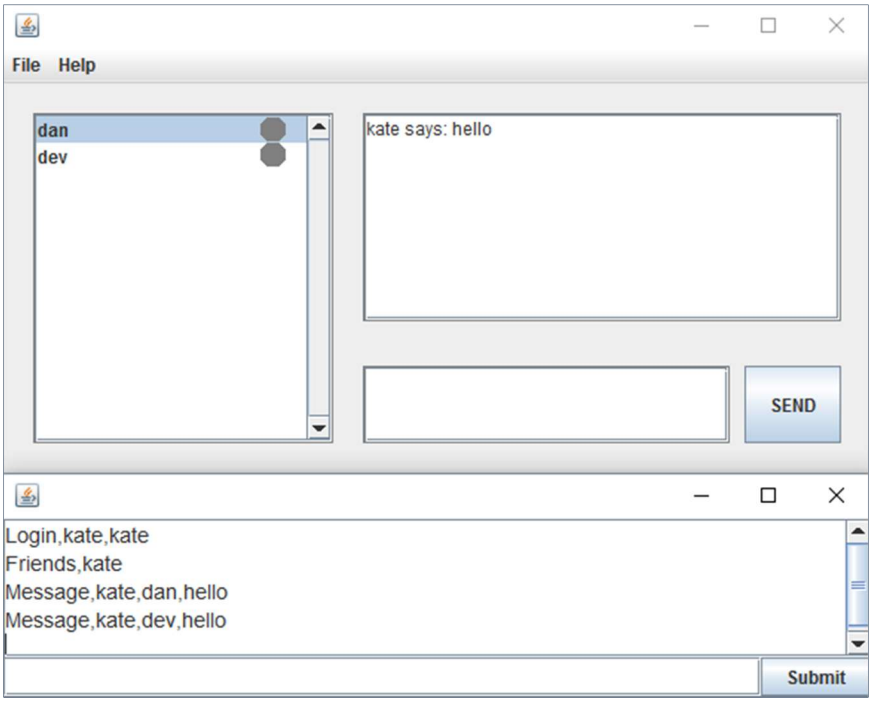


Figure 59: Chat client running

The next test for the chat application is that of initialising the autonomous agents. For testing purposes the Profile agent is being triggered, and will only trigger once an authenticated registered user successfully logs into the application. Figure 60 shows that this functionality is working as intended.

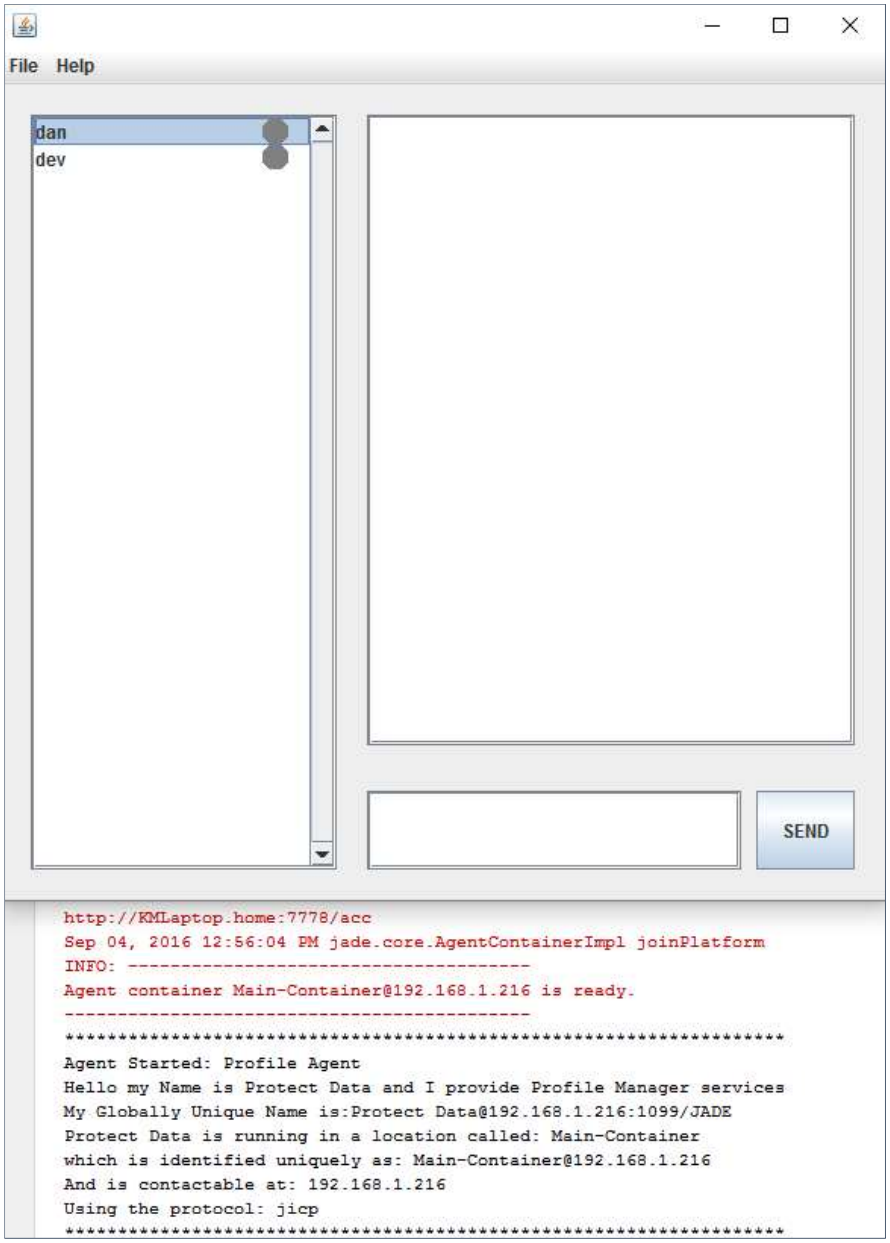


Figure 60: Profile agent triggered successfully

5.4 SafeChat Meeting Ontology Results

There are several ways to examine the ontology prototype available from within the Protégé application. Elements can be reviewed by using the search function, this will return information based on usage and occurrence and can be seen in figure 61.

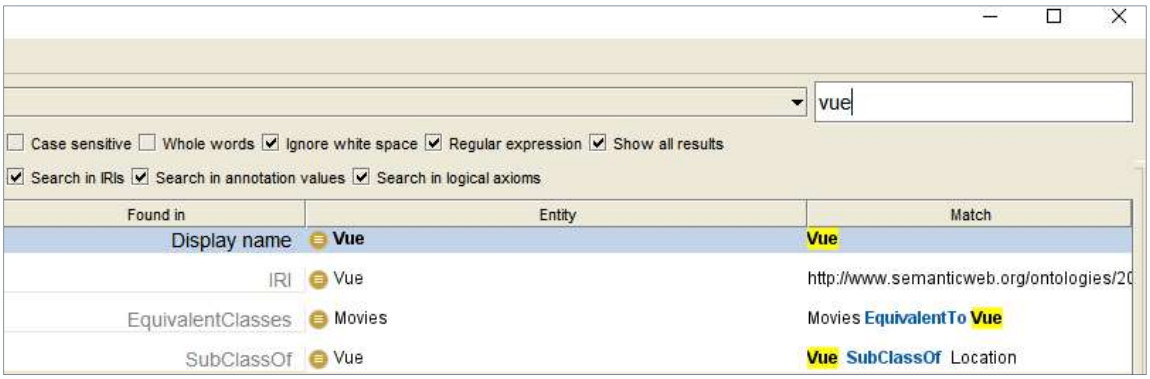


Figure 61: Result of Protégé search for the "vue" element

Another way to check that the relationships between classes and elements are set robustly, is to use the object property view. This shows component ranges and equivalences and can be seen in figure 62.

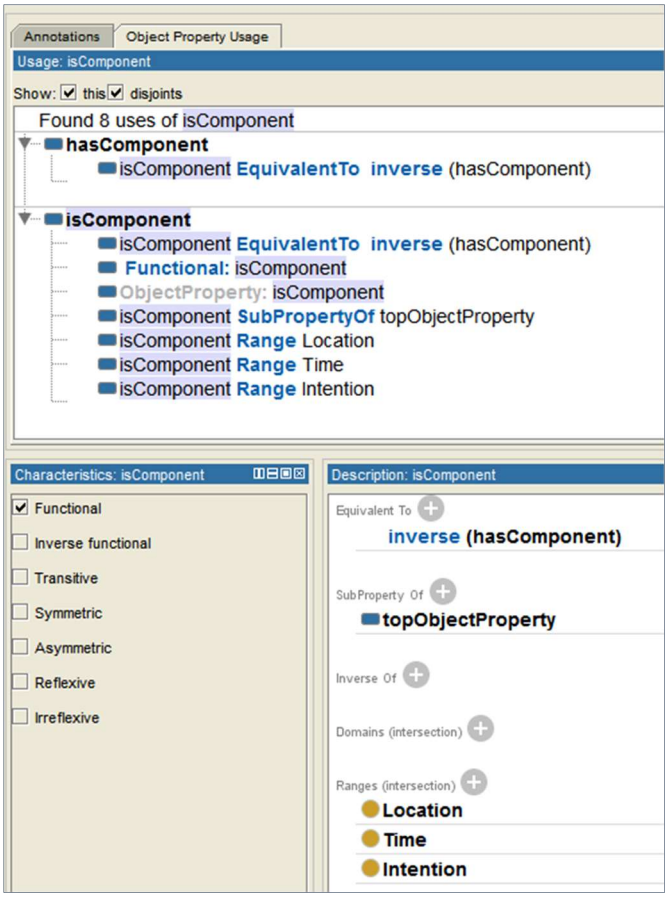


Figure 62: Relationships and ranges in the ontology

Protégé also provides a class usage view to help ensure that the ontology relationships and uses are accurate and informed. This can be evidenced in figure 63.

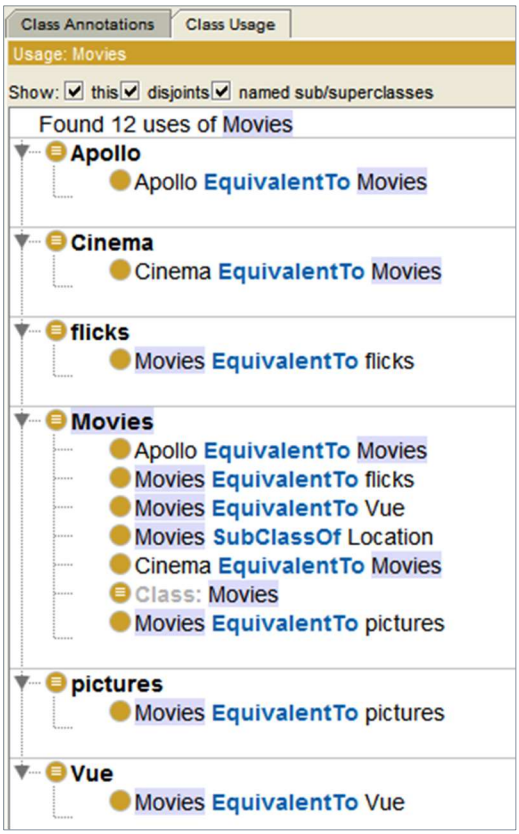


Figure 63: Class usage for the "Movies" element of the ontology

Another way to test functionality and run bespoke queries on the ontology is to export it into an Apache Jena Fuseki local server, as seen in figure 64. Once this has been completed, it is possible to load the ontology into memory and execute SPARQL queries on it.

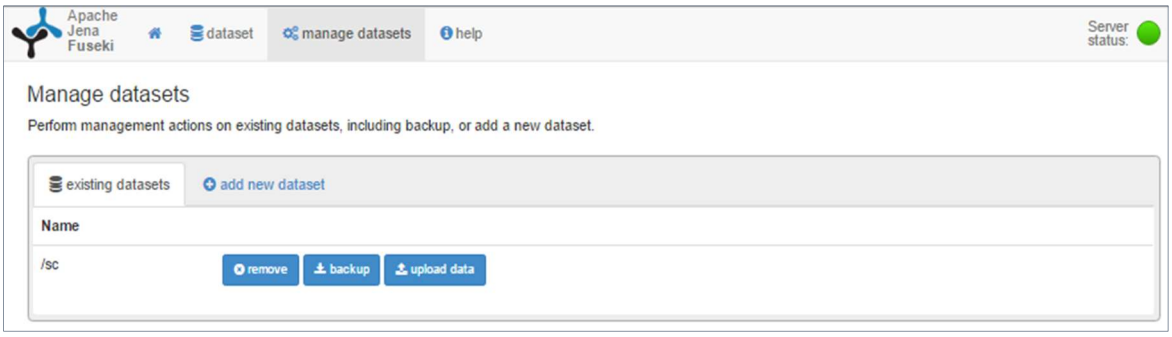


Figure 64: Apache Jena Fuseki server

Once a query has been executed on the ontology the results can be exported into several file formats for closer inspection. Figure 65 shows a SPARQL query to select triple information from the ontology.

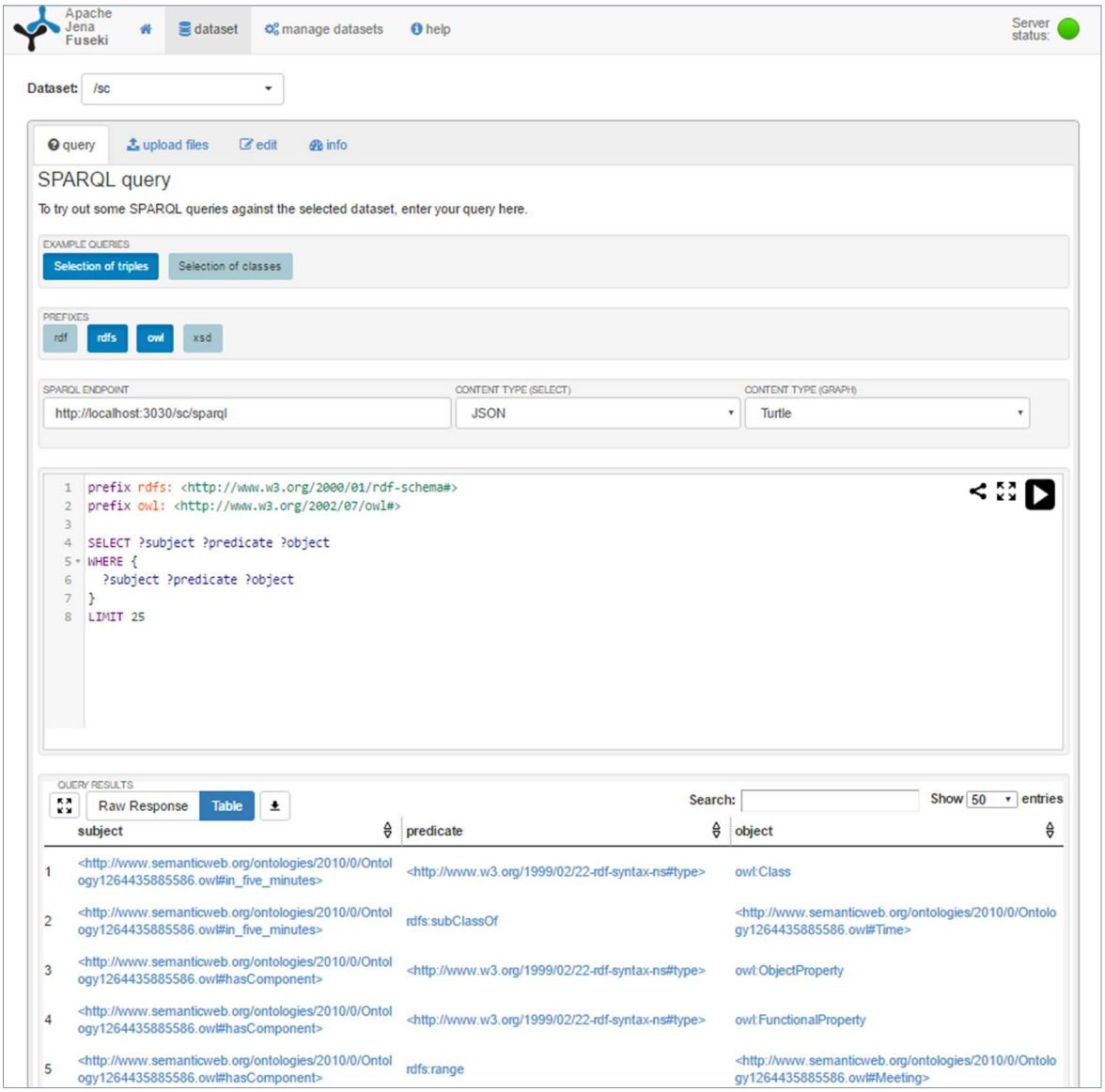


Figure 65: SPARQL triple query

5.5 Summary

The testing strategy for the SafeChat system relied on iterative development techniques. Functionality was developed for the agents in the multi-agent system, and then the agents were systematically executed to ensure behaviour was as expected. This saved time where the same functionality was needed in more than one agent, for example, when a message was successfully sent from one agent and received by its intended recipient, that code could be re-used in other agents. Each function was tested and the results recorded. Figure 66 shows the testing plan for the maintenance of the threat and safety levels performed by the threat agent.

Threat Agent Testing Plan														
lastLoc	lastTime	lastInten	testLoc	testTim	testLoc	Expexted location	Expected time	Expected intention	Actual location	Actual time	Actual intention	Expected Safety	Actual Safety
0	1	0	1	0	1	1	1	1	1	1	1	Red	Red
0	0	0	0	1	1	0	1	1	0	1	1	Red	Red
1	0	1	0	0	0	1	0	1	1	0	1	Red	Red
0	1	1	1	1	1	1	1	1	1	1	1	Red	Red
0	0	1	0	1	0	0	1	1	0	1	1	Amber	Amber
1	1	0	0	1	0	1	1	0	1	1	0	Amber	Amber
1	0	0	1	0	0	1	0	0	1	0	0	Green	Green

Figure 66: Threat agent testing plan

Results were 100% accurate once the check bit had been added to the integer, and around 33% accurate before this correction was made.

The messaging application underwent the same strategy, with each function being tested and corrected before moving onto the next function or class. Some problems were encountered triggering the agents from the messaging application. This was due to the JADE environment not shutting down fully between tests. Once this was discovered and dealt with, the agents triggered successfully all the time.

In testing the ontology, SPARQL was used to ensure that the declared classes and their relationships appeared as intended. The Protégé development suite has built in tools to check relationships within the ontology are declared accurately. The ontology will need to be integrated with a live system and be extended before it can be fully tested.

Further evidence of testing can be found in Appendix Item D.

Chapter 6: Conclusion

6.1 Introduction

This thesis has been broken down into six chapters, chapter 1 presented the problem using case studies, it reviewed and critically evaluated existing solutions and attempted to define the scope of the problem. Chapter 2 contains an examination of the literature and presented research into available technologies which may provide a solution to the problem. Chapter 3 presented the methodologies used to design the SafeChat system. Chapter 4 discussed the implementation of the SafeChat system and each component system was examined and debated. Chapter 5 examined the results of testing the SafeChat system. Finally, this chapter presents the conclusion of research that began some time ago by the results of a survey inspired by the authors' concerns about child safety online. It culminates in the design, development and implementation of the SafeChat System.

The research problem and the proposed solution are evaluated against the aims of the research. The achievements of the research are discussed as well as the limitations presented by the solution. The research began by identifying the problem and the proposed solution eventually became the development of a prototype system, comprising a multi-agent system working alongside an application specific ontology and an instant messaging application. Iterative development has insured that all three component applications have benefited from a degree of testing at various stages to ensure effective operation. The current iterations of the multi-agent system, meeting ontology and messaging application are presented in this thesis. At the end of this chapter future developments and possible future research areas are discussed.

6.2 Research Findings

As this work progressed the hypothesis identified was;

> *"The development of an intelligent multi-agent monitoring system and application specific ontology will provide a safer environment for children and vulnerable adults when communicating online."*

This led to the creation of the development of the SafeChat system, which went through an iterative process to provide automated, adaptable and transparent protection in an online setting.

The main achievement of this study is the creation of working prototypes that will serve as a first step towards creating safer environments for children, young people and vulnerable adults communicating online. The SafeChat multi-agent system performs as expected and correctly manages and maintains a threat and safety level of interactions between users.

The Chat application was superfluous to the system in many ways once the decision to create an application independent add-on system was made. It did however, still prove to be useful in proving that the agents could be triggered from an external application and in working out when this would be appropriate in the conversation life cycle.

The SafeChat meeting Ontology has moved through three of the four stages of its creation processes: specification, conceptualisation and formalisation, and will be fully evaluated once the opportunity to work with a live application is realised.

6.3 Research Contribution

In this research, the main contributions presented in the thesis are:

The development of a novel system to monitor adherence to guidelines set out by the government to protect personal information and prevent children and vulnerable adults from making meeting arrangements autonomously. This novel system was designed to provide or consider the following requirements:

- The SafeChat system will monitor the threat level of discourse autonomously and take action should pre-determined safety levels be compromised.
    - The system clearly prevents the transfer of personal data, and as seen in figure 54, the system will block such a message and inform the user and designated parent that a transgression has occurred
    - The system maintains a threat level and a safety level for discourse by tracking the number and type of meeting elements within a conversation and calculating a threat level based on these values, as seen in figure 56.
    - Based on the current threat level of the conversation the Threat Agent also calculates a safety level for the user and displays this for then, this is evidenced in figure 57 and is acting as expected when the values are adjusted using the testing plan shown in figure 66.
- This system will protect the privacy of the child and will not share the details of that discourse with anyone.
    - The system will only take action if conversation contains personal information or a number of meeting elements which exceeds safety parameters and only records the last known threat level, this ensures that all other conversation is allowed freely through the system.
- The system has been developed to be able to adapt to any other application (e.g. Facebook, Skype, etc.) or platform. This will ensure that this solution is available

across the range of popular applications and connectivity devices available now and in the future.

    - A simple chat client has been developed and figure 60 shows that the agent system can be successfully triggered from a separate environment.

- The system has been designed to use an application specific ontology, so that other ontologies can be developed to respond to other threats, such as cyberbullying or radicalisation.
    - Although the meeting ontology has not been tested in a live scenario, ontology relationships have been interrogated to ensure that they are fit for purpose and the Protégé development tools show that these relationships are articulated properly, as seen in figures 61, 62 and 63. This is further demonstrated in figure 65, which shows the initial results of a simple SPARQL triple query.

The potential impact of this work is immense, when implemented as a plug-in to widely used chat software, for example, Facebook chat, Snap Chat, WhatsApp etc. It will provide a safe environment for children to communicate, identifying potential and actual threats, whilst maintaining the privacy of the discourse. This potential was demonstrated as part of the testing and evaluation of the system. This solution could only be realised by integrating a selection of relevant components, informed by the results of current research into artificial intelligence, ontology and agent technologies.

6.4 Research Limitations

The inability to secure live testing of the ontology is one limitation of this work, and efforts to secure opportunities to do so will continue.

It has proved to be problematic that no other similar works exist for this problem, there is nothing available to evaluate against, and as a consequence, no good practice that can be drawn upon.

Attempts were made throughout the time of study to work with industry developers in order to integrate the system or parts of it, into live communications software. This was never realised, but would have provided a wealth of information and direction for future studies.

The scope of the work attempted proved to be a lot larger than anticipated. The change in focus from a single application to a system that could work in any application, whilst absolutely the right decision, made development times longer and meant that a large quantity of development time working on the chat application, was, on reflection, not fully utilised.

6.5 Future Work

There are many directions this work could take. Whilst the prototypes presented are functional they would benefit from further refinement and added sophistication. This would be better if it was done on a live system, so there is a need to resource partners willing to invest some time and manpower into the realisation of a complete live, working system.

The SafeChat meeting ontology requires further consideration in order to be evaluated thoroughly, it needs to be expanded and then applied in order for this to be completed. As with all language based problems, a solution to the nature of language evolution needs to be sourced and applied to the ontology. One idea for this is an artificial neural network, which could be developed to detect new slang words, and by analysing the context in which the words are used, apply an informed meaning to them. The ontology will also need to be expanded in order to effectively recognise abbreviations and text speak. Consideration must also be given to the size of the ontology, as it must fully cover the domain of knowledge, but this comes at a cost, computationally speaking as the larger the ontology becomes, the longer it will take to process queries. Poor spelling could also render the ontology ineffective, a solution to this could be to apply spell checking mechanisms to messages, although this could further compromise system transparency.

Further work on analysing online discourse on a larger scale could theoretically inform the threat agents decision making process. This could be done by anonymising existing data and using Hadoop to look for patterns of meeting arrangements, which in turn could inform a more defined set of rules for threat management, possibly including the introduction of probability measures.

Once a working system has been completed, the system could easily be adapted to manage other threats, for example, cyberbullying, identity theft etc.

As we develop more sophisticated and higher speed networks and communication applications, voice and video based systems will be employed more commonly. Applications like Apple's FaceTime and Microsoft's Skype already employ these features and Sony have launched their PlayStation Network, which utilises voice over IP communications. This means that the SafeChat solution would become obsolete if these technologies became more commonplace. Future iterations must begin to look at ways to provide the same level of safety across these mediums, which, because of their higher bandwidth usage will make transparency of use, a more vital consideration.

It is the authors' intention to present this work to UNICEF in order to contribute to the implementation of the articles set out in their charter on the rights of the child.

References

Aguirre, A., Brynjolfsson, E., Calo, R., Dietterich, T., George, D., Hibbard, B., Hassabis, D., Horvitz, E., Kaelbling, L.P., Manyika, J., Muehlhauser, L., Osborne, M., Parkes, D., Roff, H., Rossi, F., Selman, B. and Shanahan, M. (2015) Research priorities for robust and beneficial artificial intelligence. Available at: http://futureoflife.org/data/documents/research_priorities.pdf (Accessed: 1 May 2016).

AI open letter - FLI - future of life institute (no date) Available at: http://futureoflife.org/ai-open-letter/ (Accessed: 1 May 2016)

Al-Rfou, R., Pickett, M., Snaider, J., Sung, Y.-H., Strope, B. and Kurzweil, R. (2016) Conversational Contextual cues: The case of Personalization and history for response ranking. Available at: https://arxiv.org/pdf/1606.00372v1.pdf (Accessed: 26 July 2016).

Antoniou, G., Groth, P., van Harmelen, F., Hoekstra, R. (2012) A Semantic Web Primer 3rd Ed. MIT Press. Cambridge, Massachusetts, London, England.

Association of Modern Technologies Professionals. (2016) Software development methodologies. Available at: http://www.itinfo.am/eng/software-development-methodologies/ (Accessed: 2 August 2016)

Aylett, R., Brazier, F., Jennings, N., Luck, M., Nwana, H. and Preist, C. (1998) Agent Systems and Applications. The Knowledge Engineering Review, vol. 13, no. 3, pp. 303-308.

Ballatore, A., Bertolotto, M. and Wilson, D.C. (2014) Linking geographic vocabularies through WordNet. Annals of GIS, 20(2), pp. 73–84.

Bauer T. and Leak D.B. (2002) Handling Complex Information Environments: A Multi-Agent Framework. Proceedings of the Fourth International Bi-Conference Workshop on Agent-Oriented Information Systems

BBC [c] (2016) Former Sunderland footballer Adam Johnson 'abused position in society'. Available at: http://www.bbc.co.uk/news/uk-england-35559931 (Accessed: 1 May 2016).

Bellifemine, F., Caire, G., Greenwood, D. (2007) Developing Multi-Agent Systems with JADE. John Wiley and Sons Ltd. Chichester, England.

Belk, R.W. (ed.) (2007) Handbook of qualitative research methods in marketing. Cheltenham, UK: Edward Elgar Publishing.

Bellifemine, F., Caire, G., Poggi, A., Rimassa, G. (2008) JADE: A software framework for developing multi-agent applications. Lessons learned. Information and Software Technology vol. 50, pp. 10-21

Bellifemine, F., Caire, G., Trucco, T., Rimissa, G., Mungenast, R. (2007) JADE Administrators Guide, Online

Bellifemine, F., Poggi, A., Rimassa, G. (2001) Developing Multi-Agent Systems with a FIPA-Compliant Agent Framework. Software- Practice and Experience 31, 103-128

Berners-Lee, T. Hendler, J. Lassila, O. (2001) The Semantic Web. Scientific American, May 2001.

Bhattacharya, J. (2014) How Google processes queries in a semantic web environment. Available at: https://ahrefs.com/blog/google-processes-queries-semantic-web-environment/ (Accessed: 1 May 2016).

Bilgin, G., Dikmen, I. and Birgonul, M.T. (2014) Ontology evaluation: An example of delay analysis. Procedia Engineering, vol. 85, pp. 61-68.

Bothun, D. (2014) Consumer Intelligence Series: Technology, media and content usage among kids and teens. PCW LLP.

Bresciani, P., Perini, A., Giorgini, P., Giunchiglia, F. and Mylopoulos, J. (2004) Tropos: An Agent-Oriented Software Development Methodology. Autonomous Agents and Multi-Agent Systems, vol. 8, no. 3, pp. 203-236.

Brewster, C., Alani, H., Dasmahapatra, S. and Wilks, Y. (2004) Data driven Ontology evaluation. Available at: http://www.cbrewster.com/papers/BrewsterLREC.pdf (Accessed: 19 July 2016).

Brickley, D. Guha, R. V. (eds) (2014) RDF schema 1.1. Available at: https://www.w3.org/TR/rdf-schema/ (Accessed: 31 May 2016)

Calero, C., Ruiz, F., Piattinni, M. (eds) (2010) Ontologies for Software Engineering and Software Technology. Springer-Verlag. Berlin, Germany.

Callaghan, V. Clarke, G. Pounds-Cornish, A. Sharples, S. (2000) Buildings as Intelligent Autonomous Systems: A Model for Integrating Personal and Building Agents. Conference proceedings: 6th Intelligent Autonomous Systems.

Cambridge free English dictionary and thesaurus (2016) Available at: http://dictionary.cambridge.org/ (Accessed: 1 May 2016).

Child exploitation and online protection centre - internet safety (2013) Available at: https://www.ceop.police.uk/Media-Centre/Press-releases/2013/ALARMING-NEW-TREND-IN-ONLINE-SEXUAL-ABUSE/ (Accessed: 1 May 2016).

Child exploitation and online protection centre - internet safety (no date) Available at: https://www.ceop.police.uk/safety-centre/ (Accessed: 1 May 2016).

Chitu, A. (2010) How Google translate works. Available at: http://googlesystem.blogspot.co.uk/2010/08/how-google-translate-works.html (Accessed: 26 July 2016).

Connolly, D., van Harmelen, F., Horrocls, I., McGuinness, D., Patel-Schneider, P., Stein, L. (2001) DAML+OIL (march 2001) reference description. Available at: https://www.w3.org/TR/daml+oil-reference (Accessed: 31 May 2016)

Coppin chapter 19.ppt (2016) Available at: http://studylib.net/doc/15495156/coppin-chapter-19.ppt (Accessed: 6 September 2016).

Corcho, O., Fernandez-Lopez, M., Gomez, Perez, A., Lopez-Cima, A. (2004) Building Legal Ontologies with Methontology and WebODE. Law and the Semantic Web.

Dale, J. (2016) Welcome to the foundation for intelligent physical agents. Available at: http://www.fipa.org/ (Accessed: 1 May 2016).

Das, S.K. 2008, Foundations of decision-making agents: logic, probability and modality, World Scientific, London;Singapore;.

Davies, J., Fensel, D., Van Harmelen, F. 2007, Towards the Semantic Web (Ontology Driven Knowledge Management). John Wiley and Sons Ltd. Chichester, England.

DeAngelis, S.F. and Solutions, E. (2016) The growing importance of natural language processing. Available at: http://www.wired.com/insights/2014/02/growing-importance-natural-language-processing/ (Accessed: 21 July 2016).

Dictionary [a] (2016) The definition of voice recognition. Available at: http://www.dictionary.com/browse/voice-recognition (Accessed: 1 May 2016).

Dictionary [b] (2016) 'The definition of natural language processing', in Available at: http://www.dictionary.com/browse/natural-language-processing (Accessed: 21 July 2016).

Engadget (2011) IBM's Watson supercomputer destroys humans in jeopardy | Engadget. Available at: https://www.youtube.com/watch?v=WFR3lOm_xhE (Accessed: 1 May 2016).

FaCT++ Reasoner (2016) Available at: http://owl.cs.manchester.ac.uk/tools/fact/ (Accessed: 19 July 2016).

Features | Roo kids (2016) Available at: http://www.rookidsapp.com/features/ (Accessed: 1 May 2016).

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D.L. and Patel-Schneider, P.F. (2001) OIL: an ontology infrastructure for the Semantic Web, IEEE Intelligent Systems, vol. 16, no. 2, pp. 38-45.

Fernandez Lopez, M. (1999) Overview of Methodologies For Building Ontologies, In Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods (KRR5) Stockholm, Sweden.

Finin, T., Fritzson, R., McKay, D. and McEntire, R. (1994) KQML as an agent communication language. ACM, pp. 456.

Gallagher, S. (2015) Cortana for all: Microsoft's plan to put voice recognition behind anything. New York: Condé Nast Publications, Inc.

Gasevic, D., Djuric, D., Devedjic, V. (2006) Model Driven Engineering and Ontology Development. Springer-Verlag. Berlin, Germany.

Genesereth, M.R., Fikes, R.E., Bobrow, D., Brachman, R., Gruber, T., H, P., Letsinger, R., Lifschitz, V., Macgregor, R., Mccarthy, J., Norvig, P., Patil, R. and Schubert, L. (1992) Knowledge interchange format version 3.0 reference manual. Available at: https://www.cs.auckland.ac.nz/courses/compsci367s2c/resources/kif.pdf (Accessed: 31 May 2016)

Gómez-Pérez, A. (2001) Evaluation of Ontologies. International Journal of Intelligent Systems, vol. 16, no. 3, pp. 391-409.

Google [a] (2016) Available at: https://www.google.co.uk/webhp?sourceid=chrome-instantandion=1andespv=2andie=UTF-8#q=how+old+was+winston+churchill+when+he+died (Accessed: 1 May 2016).

Google [b] (2016) Available at: https://www.google.co.uk/webhp?sourceid=chrome-instantandion=1andespv=2andie=UTF-8#q=internet%20usage%20statistics%20uk (Accessed: 1 May 2016).

Google [c] (2016) Available at: https://www.google.co.uk/webhp?sourceid=chrome-instantandion=1andespv=2andie=UTF-8#q=agent+definition (Accessed: 12 May 2016).

Google [d] (2016) Available at: https://www.google.co.uk/webhp?sourceid=chrome-instantandion=1andespv=2andie=UTF-8#q=deixis (Accessed: 21 July 2016).

Graham, P. (2002) A plan for Spam. Available at: http://www.paulgraham.com/spam.html (Accessed: 26 July 2016)

Greer, D. and Hamon, Y. (2011) Agile Software Development. Software: Practice and Experience, vol. 41, no. 9, pp. 943-944.

Gruber, T. (1993) Towards principles for the design of ontologies used for knowledge sharing. In Guarino, N., Poli, R. (Ed's) International Workshop on Formal Ontology in Conceptual Analysis and Knowledge Representation. Padove, Italy

Guarino, N. (1998) Formal Ontology in information systems: Proceedings of the 1st international conference June 6-8, 1998, Trento, Italy, Vol. 1. United States: IOS Press, US.

Guba, E. G., and Lincoln, Y. S. (2005). Paradigmatic controversies, contradictions, and emerging confluences. In Denzin N. K., and Lincoln, Y. S. (Ed's), The sage handbook of qualitative research (3rd Ed.). Sage pp191-215

Hawkins, J., Blakeslee, S. (2005) On Intelligence. Owl Books. New York, USA.

Hlomani, H. and Stacey, D. (2014) An extension to the data-driven ontology evaluation. Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration (IEEE IRI 2014)

Holmbeck, G. N., Li, S. T., Schurman, J. V., Friedman, D., and Coakley, R. M. (2002). Collecting and managing multisource and multimethod data in studies of paediatric populations. Journal of Paediatric Psychology, 27, pp 5–18

Home Office and Ministry of Justice (2013) An overview of sexual offending in England and wales. Available at: https://www.gov.uk/government/statistics/an-overview-of-sexual-offending-in-england-and-wales (Accessed: 1 May 2016).

Horridge, M., Knublauch, H., Rector, A, Steverns, R., Wroe, C. (2004) A Practical Guide to Building Ontologies Using the Protégé- OWL Plugin and CO-ODE Tools. The University of Manchester, United Kingdom.

Hsu, F. (1999) IBM's Deep Blue Chess grandmaster chips. IEEE Micro, vol. 19, no. 2, pp. 70-81

Humphreys, K., Demetriou, G. and Gaizauskas, R. (1999) 'Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures', Biocomputing 2000, . doi: 10.1142/9789814447331_0048.

Hunt, K. (2014) Ex-parish council chairman Christopher Worrall jailed after booking Tunbridge wells hotel to abuse teenage boy groomed on Facebook. Available at: http://www.kentonline.co.uk/tunbridge-wells/news/parish-council-chairman-jailed-21139/ (Accessed: 1 May 2016).

IBM systems magazine - it's technical, dear Watson (2016) Available at: http://www.ibmsystemsmag.com/ibmi/trends/whatsnew/It%E2%80%99s-Technical,-Dear-Watson/ (Accessed: 1 May 2016).

IBM. (2016) Text Analytics, Watson, and healthcare. Available at: https://vimeo.com/25516026 (Accessed: 1 May 2016).

Innovation, I. (2015) Stephen hawking: Should we fear artificial intelligence? Available at: http://www.internationalinnovation.com/stephen-hawking-should-we-fear-artificial-intelligence/ (Accessed: 1 May 2016).

Investors (2015) Facebook reports First quarter 2015 results. Available at: http://investor.fb.com/releasedetail.cfm?ReleaseID=908022 (Accessed: 1 May 2016).

Irowe, S. (2012) *Pros and cons of research techniques*. Available at: https://staceylrowe.wordpress.com/2012/01/16/pros-and-cons-of-research-techniques/ (Accessed: 11 December 2016).

JADE Ontologies (no date) Available at: http://www.iro.umontreal.ca/~vaucher/Agents/Jade/Ontologies.htm (Accessed: 1 May 2016).

jade-develop] getting protégé OWL individuals exported into JADE agent knowledgebase (no date) Available at: http://jade.tilab.com/pipermail/jade-develop/2010q2/015306.html (Accessed: 1 May 2016).

Jaimes, A., Sebe, N. (2007) Multimodal human–computer interaction: A survey. Computer Vision and Image Understanding, vol. 108, no. 1, pp. 116-134.

Jakus, G., Milutinovic, V., Omerovic, S., Tomazic, S. (2013) Concepts, Ontologies, and Knowledge Representation. Springer, London, United Kingdom.

Jurafsky, D., Martin, J. (2008) Speech and Language Processing (An Introduction to Natural Language Processing, computational Linguistics and Speech Recognition). Prentice-Hall, Inc. New Jersey, USA.

K9 web protection (2010) Available at: http://www1.k9webprotection.com/ (Accessed: 1 May 2016).

Kablan, A. (2016) AI is not a threat to humanity, but an Internet of 'smart' things may be! Available at: http://techcrunch.com/2016/04/20/artificial-intelligence-is-not-a-threat-to-humanity-but-an-internet-of-smart-things-may-be/ (Accessed: 1 May 2016).

Kemp, R., Moore, A. (2007) Privacy. Library Hi Tech vol. 25, pp. 58-57

Kngine (2012) Ask me anything. Available at: http://kngine.com/#Know!q=how%20old%20was%20Winston%20Churchill%20when%20he%20died (Accessed: 1 May 2016).

Kothari, A. (2004) Ghengis - A Multi-Agent Carpooling System. PDF, Online.

Larose, G. and Kruse, P. (2005) What is ontology? - definition from WhatIs.com. Available at: http://whatis.techtarget.com/definition/ontology (Accessed: 1 May 2016).

Liddy, E.D. (2001) Natural language processing recommended citation. Available at: http://surface.syr.edu/cgi/viewcontent.cgi?article=1043andcontext=istpub (Accessed: 21 July 2016).

Loom (2006) Available at: http://www.isi.edu/isd/LOOM/ (Accessed: 31 May 2016).

Lopez, M.F., Gomez-Perez, A., Sierra, J.P. and Sierra, A.P. 1999, "Building a chemical ontology using Methontology and the Ontology Design Environment" IEEE Intelligent Systems and their Applications, vol. 14, no. 1, pp. 37-46

Lord, P. (2010) Components of an Ontology. Available at: http://ontogenesis.knowledgeblog.org/514 (Accessed: 1 May 2016).

Lur, X. (2012) Kngine: The smartest search engine ever? Available at: http://techxav.com/kngine-the-smartest-search-engine-ever/ (Accessed: 1 May 2016).

M. Uschold, and M. King. (1995) Proc. of IJCAI95's WS on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada,

MacFarlane, K., Holmes, V. (2009) Agent-Mediated Information Exchange: Child Safety Online, in Management and Service Science. MASS 2009.

Makuch, E. (2014) Minecraft passes 100 million registered users, 14.3 million sales on PC. Available at: http://www.gamespot.com/articles/minecraft-passes-100-million-registered-users-14-3-million-sales-on-pc/1100-6417972/ (Accessed: 1 May 2016).

Manning, C., Schutze, H. (2001) Foundations of Statistical Natural Language Processing. MIT Press. Cambridge, Massachusetts, London, England.

Martin, A. (2014) 'Canterbury cannibal' found GUILTY of sexually grooming a child. Available at: http://www.dailymail.co.uk/news/article-2699897/Canterbury-cannibal-told-14-year-old-girl-wanted-kill-eat-GUILTY-sexually-grooming-child.html (Accessed: 1 May 2016).

Mathiesen, K. (2013) The Internet, Children and Privacy: The case against parental monitoring. Springer Science and Business, Dordrecht: online.

McGuire, M., Dowling, S. (2013) Cyber-crime: A review of the evidence, Home Office Research Report 75, Chapter 3: Cyber-enabled crimes-sexual offending against children

McKay, V. (1988- 2016) What is a software development process? Available at: http://www.selectbs.com/analysis-and-design/what-is-a-software-development-process (Accessed: 1 August 2016).

Microsoft (2015) Accessibility in windows 10. Available at: https://www.microsoft.com/enable/products/windows10/ (Accessed: 1 May 2016).

Misra, S.C. and Singh, V. (2015) Conceptualizing open agile software development life cycle (OASDLC) model. International Journal of Quality and Reliability Management, vol. 32, no. 3, pp. 214-235.

Noy, N.F. and McGuinness, D.L. (2001) Ontology development 101: A guide to creating your First Ontology. Available at: http://protege.stanford.edu/publications/ontology_development/ontology101.pdf (Accessed: 2 May 2016).

NSPCC (2016) Grooming. Available at: https://www.nspcc.org.uk/preventing-abuse/child-abuse-and-neglect/grooming/ (Accessed: 1 May 2016).

Nuance. (2016) Nuance healthcare. Available at: http://www.nuance.com/for-healthcare/resources/clinical-language-understanding/ontology/index.htm (Accessed: 19 July 2016).

OECD (2012) The protection of children online. Available at: https://www.oecd.org/sti/ieconomy/childrenonline_with_cover.pdf (Accessed: 1 May 2016).

Ogden, C.K., and Richards, I.A. (1927). Meaning of meaning. Harcourt, Brace and Company, New York

OWL - semantic web standards (2004) Available at: https://www.w3.org/2001/sw/wiki/OWL (Accessed: 1 May 2016).

Pal, D., Mitra, M. and Datta, K. (2014) Improving query expansion using WordNet. Journal of the Association for Information Science and Technology, vol. 65, no. 12, pp. 2469-2478.

Parents' guide to the Internet (2001) Available at: http://www2.ed.gov/pubs/parents/internet/index.html (Accessed: 1 May 2016).

Press Association. (2013) Nearly one in 10 children gets first mobile phone by age five, says study. Available at: http://www.theguardian.com/money/2013/aug/23/children-first-mobile-age-five (Accessed: 1 May 2016)

Princeton (2016) About WordNet - WordNet - about WordNet. Available at: https://wordnet.princeton.edu/ (Accessed: 19 July 2016).

Purcell, J.E. (2016) Comparison of software development Lifecycle methodologies. Available at: http://software-security.sans.org/resources/paper/cissp/comparison-software-development-lifecycle-methodologies (Accessed: 1 August 2016).

Rao A.S., Georgeff, M. (1995) BDI Agents: From Theory to Practice. In Proceedings of the 1st International Conference on Multi-Agent Systems pp .312-319 San Francisco, CA.

RDF Working Group (2014) RDF - semantic web standards; Available at: https://www.w3.org/RDF/ (Accessed: 31 May 2016).

Russell, S., Norvig, P. (2014) Artificial Intelligence: A Modern Approach 3rd Ed. Pearson Education Ltd. Harlow, Essex, England.

Sandhu, R. (2016) 5 applications of natural language processing technology how will NLP shape the future of the tech world? Available at: http://newtech.about.com/od/semanticweb/tp/5-Applications-Of-Natural-Language-Processing-Technology.htm (Accessed: 26 July 2016).

Sawsaa, A. (2013) A Generic Model of Ontology to Visualize Information Science Domain(OIS). Doctoral thesis, University of Huddersfield

Senior (2014) What it will take for computers to be conscious. Available at: https://www.technologyreview.com/s/531146/what-it-will-take-for-computers-to-be-conscious/ (Accessed: 1 May 2016).

Serenko, A. and Detlor, B. (2002) AGENT TOOLKITS: A GENERAL OVERVIEW OF THE MARKET AND AN ASSESSMENT OF INSTRUCTOR SATISFACTION WITH UTILIZING TOOLKITS IN THE CLASSROOM. Available at: http://www2.econ.iastate.edu/tesfatsi/Agent_Toolkits_Working_Paper.SerenkoDetlor2002.pdf (Accessed: 1 May 2016).

Seth and Institute, S.B.R. (2015) Text message statistics. Available at: http://www.statisticbrain.com/text-message-statistics/ (Accessed: 1 May 2016).

Shields, B.J., Home Office and Department for Culture, Media and Sport (2015) UK Internet safety and security minister's speech to the WeProtect summit. Available at: https://www.gov.uk/government/speeches/uk-internet-safety-ministers-speech-to-the-weprotect-summit (Accessed: 5 May 2016).

Shmueli, B., Blecher-Prigat, A. (2011) Privacy for Children. Columbia Human Rights Law Review.

Smmry (2016) About. Available at: http://smmry.com/about (Accessed: 26 July 2016).

Social, P.D. (2016) PLAYMessenger - kids safe chat – Android Apps on Google play. Available at: https://play.google.com/store/apps/details?id=com.pgdigital.playmessengerandhl=en_GB (Accessed: 1 May 2016).

Staab, S. and Studer, R. (eds.) (2009a) Handbook on ontologies. 2nd edn. Berlin: Springer-Verlag Berlin and Heidelberg GmbH and Co. K.

Statista. (2016) Leading social networks worldwide as of April 2016, ranked by number of active users (in millions). Available at: http://www.statista.com/statistics/272014/global-social-networks-ranked-by-number-of-users/ (Accessed: 1 May 2016).

Statistics (2009) Minecraft. Available at: https://minecraft.net/stats (Accessed: 1 May 2016).

Stenzhorn, H. Basic formal Ontology (BFO) (2008) Available at: http://ontology.buffalo.edu/bfo/ (Accessed: 19 July 2016).

Stories of 7 teen suicides because of ask.FM bullying (2013) Available at: http://nobullying.com/stories-of-7-teen-suicides-because-of-ask-fm-bullying/ (Accessed: 1 May 2016).

Studer, R., Benjamins, V.R. and Fensel, D. (1998) Knowledge Engineering: Principles and methods. Data and Knowledge Engineering, vol. 25, no. 1-2, pp. 161-197.

Technologies, S. (2016) How are we building smarter search engines in the big data age?. Available at: http://www.searchtechnologies.com/smart-search-engines-with-big-data (Accessed: 1 May 2016).

TechTalk. (2015) System development life cycle (SDLC) approaches. Available at: https://tech-talk.org/2015/01/21/system-development-life-cycle-sdlc-approaches/ (Accessed: 1 August 2016).

Terms (2016) Martine Rothblatt on AI, mind clones and the future of the self. Available at: http://www.campaignlive.com/article/martine-rothblatt-ai-mind-clones-future-self/1338436 (Accessed: 1 May 2016).

UNICEF (2015) A Summary of the UN Convention on the Rights of the Child, United Kingdom, Online.

Van Gurp, M., Decoene, M., Holvoet, M., Casella, M. and Santos (2006) LinKBase®, a philosophically-inspired Ontology for NLP/NLU applications. Available at: http://ceur-ws.org/Vol-222/krmed2006-p08.pdf (Accessed: 19 July 2016)

Volin, V. (2010) SuperProfundo. Available at: http://superprofundo.com/2010/12/13/top-down-and-bottom-up-pros-and-cons/ (Accessed: 1 May 2016).

Wagle, K. 2013, IBM Watson: Revolutionizing healthcare? Young Scientists Journal, vol. 6, no. 13, pp. 17.

Warren, T. (2014) The story of Cortana, Microsoft's Siri killer. Available at: http://www.theverge.com/2014/4/2/5570866/cortana-windows-phone-8-1-digital-assistant (Accessed: 1 May 2016).

Watch, H.R. (2016) Mind the gap. Available at: https://www.hrw.org/report/2015/04/09/mind-gap/lack-accountability-killer-robots (Accessed: 1 May 2016).

Webster, S., Davidson, J., Bifulco, A., Gittschalk, P., Carreti, V., Pham, T. (2012) European Online Grooming Report. European Comission, Safer Internet Plus Programme.

Whittle, H., Hamilton-Giachritsis, C., Beech, A. and Collings, G. (2013) "A review of young people's vulnerabilities to online grooming", Aggression and Violent Behaviour vol. 18, no. 1, pp. 135-146

Witek, M. (2015) 'Linguistic under determinacy: A view from speech act theory', Journal of Pragmatics, 76, pp. 15–29. doi: 10.1016/j.pragma.2014.11.003.

Withnall, A. (2014) Breck Bednar murder: Teenage video gamer admits killing British schoolboy after meeting arranged online. Available at: http://www.independent.co.uk/news/uk/crime/breck-bednar-murder-teenage-video-gamer-admits-killing-british-schoolboy-after-meeting-arranged-9882590.html (Accessed: 1 May 2016).

Woodford, C. (2015) Voice recognition software. Available at: http://www.explainthatstuff.com/voicerecognition.html (Accessed: 1 May 2016).

Wooldridge, M. and Jennings, N.R. (1995) Intelligent agents: theory and practice. The Knowledge Engineering Review,vol. 10, no. 2, pp. 115-152.

Wooldridge, M. and Jennings, N.R. (1999) SOFTWARE ENGINEERING WITH AGENTS: Pitfalls and Pratfalls. IEEE INTERNET COMPUTING, June pp. 20-27.

Wooldridge, M., Jennings, N.R. and Kinny, D. (2000) The Gaia Methodology for Agent-Oriented Analysis and Design. Autonomous Agents and Multi-Agent Systems, vol. 3, no. 3, pp. 285-312.

Wooldridge, M.J. 2009, An Introduction to Multi Agent Systems, 2nd Ed, Wiley, Chichester.

Yao, H., Orme, A.M. and Etzkorn, L. (2005) Cohesion metrics for Ontology design and application. Journal of Computer Science, 1(1), pp. 107–113.

Zolfagharifard, E. (2016) Elon musk opens AI GYM to train machines on Atari games. Available at: http://www.dailymail.co.uk/sciencetech/article-3564752/Elon-Musk-opens-AI-GYM-train-machines-Atari-games-Billionaire-says-initiative-help-computers-think-like-humans.html (Accessed: 1 May 2016).

Bibliography

24, E. and Administrator, S. (2016) Happy birthday Skype: Even monkeys use it now. Available at: http://www.emirates247.com/news/happy-birthday-skype-even-monkeys-use-it-now-2013-08-28-1.519060 (Accessed: 1 May 2016).

Abretti, M.-L. (2013) More than a million British kids get first mobile phone by the time they're five. Available at: http://www.uswitch.com/media-centre/2013/08/more-than-a-million-british-kids-get-first-mobile-phone-by-the-time-theyre-five/ (Accessed: 1 May 2016).

AI open letter - FLI - future of life institute (no date) Available at: http://futureoflife.org/ai-open-letter/ (Accessed: 1 May 2016)

Aoqui, L.G.F. (2014) Build a real-time chat app with Node-RED in 5 minutes. Available at: http://www.ibm.com/developerworks/cloud/library/cl-rtchat-app/ (Accessed: 1 May 2016).

Ashrafian, H. (2015) AIonAI: a humanitarian law of artificial intelligence and robotics", Science and engineering ethics. vol. 21, no. 1, pp. 29-12.

Association, P. (2016) Nearly one in 10 children gets first mobile phone by age five, says study. Available at: http://www.theguardian.com/money/2013/aug/23/children-first-mobile-age-five (Accessed: 1 May 2016).

Bagić Babac, M. and Jevtić, D. (2014) AgentTest: A specification language for agent-based system testing. Neurocomputing, vol. 146, pp. 230-248.

BBC [b] (2014) Boy held over internet grooming of girl in Manchester. Available at: http://www.bbc.co.uk/news/uk-england-manchester-28265452 (Accessed: 1 May 2016).

Blizzard. (2011) Remote guild chat now on Android, iPhone, and iPod touch [UPDATED 4/26]. Available at: http://us.battle.net/wow/en/blog/2653949/remote-guild-chat-now-on-android-iphone-and-ipod-touch-%5Bupdated-426%5D-4-26-2011 (Accessed: 1 May 2016).

Bogdan, S., Rigo, I., Miklic, D. (2008) Fuzzy Logic Navigation in Multi Agent Systems. IEEE International Symposium on Intelligent Control

Branisso, L., Kato, E., Pedrino, E. (2013) A Multi-Agent System Using Fuzzy Logic to Increase AGV Fleet Performance in Warehouses, in III Brazilian Symposium on Computer Systems Engineering

Chalabi, M. (2016) 36m Brits use the internet every day - but what are they all doing? Available at: http://www.theguardian.com/news/datablog/2013/aug/08/36-million-brits-internet-every-day-habits-use (Accessed: 1 May 2016).

Chandrasekar, R. (2014) "Elementary? Question answering, IBM's Watson, and the Jeopardy! challenge", Resonance, vol. 19, no. 3, pp. 222-241.

Chaware, S. Rao, S. (2010) Integrated Approach to Ontology Development Methodology with Case Study. International Journal of Database Management Systems (IJDMS), vol. 2, no. 3, pp. 13-19

Child-monitoring App pulled amid security fears (2015) Available at: http://news.sky.com/story/1580261/child-monitoring-app-pulled-amid-security-fears (Accessed: 1 May 2016).

Cia, Y., Yeung, C., Leung, H. (2012) Fuzzy Computational Ontologies in Contexts: Formal Models of Knowledge Representation with Membership Degree and Typicality of Objects, and Their Applications. Springer. London, United Kingdom.

Coplien, J.O., Bjŗnvig, G. and Books 24x7, I. (2011). Lean architecture for Agile software development, 1. Aufl.;1; edn, Wiley, Chichester.

Core JavaTM: Volume I—Fundamentals, ninth edition (2012) Available at: http://my.safaribooksonline.com/9780137082346 (Accessed: 1 May 2016).

Cosslett, R.L. (2014) The ignore no more app is wrong because parenting should be about trust. Available at: http://www.theguardian.com/commentisfree/2014/aug/19/ignore-no-more-app-parenting (Accessed: 1 May 2016).

Cresswell, S., McCluskey, T., West, M. (2013) Acquiring Domain Models Using LOCM. Knowledge Engineering Review.

de Blanc, P. (2011) Ontological Crises in Artificial Agents' Value Systems. Machine Intelligence Learning Institute.

Dentler, K., Cornet, R., ten Teije, A., de Keizer, N. (2011) Comparison of Reasoners for Large Ontologies in the OWL 2 EL Profile. Semantic Web vol. 1, pp. 1-5

Eckel, B. (2006) Thinking in Java 4th Ed. Pearson Education, New Jersey, USA

Elkan, C. and Greiner, R. (1993) 'Building large knowledge-based systems: Representation and inference in the cyc project', Artificial Intelligence, 61(1), pp. 41–52.

Enough is enough: Protecting our children online (no date) Available at: http://www.enough.org/inside.php?tag=statistics (Accessed: 1 May 2016).

EnoughSM, E.I. (2009) InternetSafety101.org: Statistics. Available at: http://www.internetsafety101.org/Predatorstatistics.htm (Accessed: 1 May 2016).

Enyon, R. (2009) Harnessing Technology: The learner and their context. Mapping young people's uses of technology in their own contexts. Oxford, BECTA

Ermolayev, V., Davidovsky, M. (2012) Agent-Based Ontology Allignment: Basics, Applications, Theoretical Foundations and Demonstration, in ACM 2012

Estival, D., Nowak, C., Zschorn, A. (2004) Towards Ontology Based Natural Language Processing. Human Systems Integrations Group, Australia.

Fallenstein, B., Soares, N. (2015) Vingean Reflection: Reliable Reasoning for Self-Improving Agents. Technical report 2015–2. MIRI

Fensel, D. (2003) Ontologies: A silver bullet for knowledge management and electronic commerce. 2nd edn. Germany: Springer-Verlag Berlin and Heidelberg GmbH and Co. K.

FERNÁNDEZ-LÓPEZ, M., GÓMEZ-PÉREZ, A. (2002) Overview and analysis of methodologies for building ontologies. The Knowledge Engineering Review, vol. 17, no. 2, pp. 129-156.

Ferndndez, M., Gomez-Perez, A. and Juristo, N. (1997) METHONTOLOGY: From ontological Art Towards ontological engineering. Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering.

Finin, T.; Fritzson, R.; McKay, D.; McEntire, R. (1994) KQML as an agent communication language. Proceedings of the third international conference on Information and knowledge management - CIKM '94. p. 456.

Ford, P. (2015) Our fear of artificial intelligence. Available at: https://www.technologyreview.com/s/534871/our-fear-of-artificial-intelligence/ (Accessed: 1 May 2016).

Friedrich, A. (2015) Fight the cold cyberwar with analytics. Available at: http://www.ibmbigdatahub.com/blog/fight-cold-cyberwar-analytics (Accessed: 1 May 2016).

Fuentes-Fernandez, R., Hassan, S., Pavon, J., Galan, J., Lopez-Parades, A. (2012) Metamodels for Role-Driven Agent-Based Modeling. Computation and Mathematical Organisation Theory vol. 18, pp .91-112

Gani, A. (2015) Kate Winslet says children being harmed by social media. Available at: http://www.theguardian.com/film/2015/nov/01/kate-winslet-says-children-being-harmed-by-social-media (Accessed: 1 May 2016).

Gate (2016) Available at: https://gate.ac.uk/ (Accessed: 12 May 2016)

Gawich, M., Badr, A., Ismael, H., Hegazy, A. (2012) Alternative Approaches for Ontology Matching, in International Journal of Computer Applications vol. 49, no. 18.

GmbH, semafora systems – (2012) Semafora systems: Home. Available at: http://www.semafora-systems.com/en/ (Accessed: 12 May 2016).

GO Consortium. (2012) Gene Ontology Annotations and resources. Nucleic Acids Research, 41(D1), pp. D530–D535.

Gómez-Pérez, A., Fernández-López, M. and Corcho, O. (2004) Ontological engineering: with examples from the areas of knowledge management, e-commerce and the Semantic Web. Springer, London.

Gomez-Sans, J., Fernandez, R., Arroyo, J. (2010) Model Driven Development and Simulations with the INGENIAS Agent Framework. Simulation Modelling Practice and Theory vol. 18, pp. 1468-1482

Gray, R. (2016) Pepper the 'emotional' humanoid becomes first robot to attend SCHOOL. Available at: http://www.dailymail.co.uk/sciencetech/article-3540307/Pepper-grows-Emotional-humanoid-robot-enroll-SCHOOL-Japan.html (Accessed: 1 May 2016).

Grier, D.A. (2008) Edward Elgar's Facebook Page. Computer vol. 41, no. 11, pp. 6-8.

Griss, M., (2001) Software Agents as Next Generation Software Components, in Component-Based Software Engineering: Putting the Pieces Together.

Gunasekera, K., Zaslavsky, A., Loke, S., Krishnaswamy, S. (2008) Context Driven Adaptation of Mobile Agents in Mobile Data Management Workshops, MDMW 2008.

Hanewald, R. (2013) From Cyber Bullying to Cyber Safety: Issues and Approaches in Educational Contexts. Nova Science Publishers, New York, USA.

Hinks, J. (2015) Best free parental control software: 5 programs to keep your kids safe in 2015. Available at: http://www.techradar.com/news/software/applications/best-free-parental-control-software-9-programs-to-keep-your-kids-safe-1140315 (Accessed: 1 May 2016).

Holmes, Violeta and MacFarlane, Katrinna (2009) Agent Mediated Information Exchange. In: University of Huddersfield Research Festival, 23rd March - 2nd April 2009, University of Huddersfield.

Home (2016) 'World of Warcraft', 18 April. Available at: http://eu.battle.net/wow/en/?- (Accessed: 1 May 2016).

Home. (2015) PLAYMessenger, your child's first messaging app. Available at: http://www.playmessenger.com/home/ (Accessed: 1 May 2016).

Huhns, M., Singh, M. (1997) Ontologies for Agents, IEEE Internet Computing, 81-83

IHMC (2014) CmapTools. Available at: http://cmap.ihmc.us/ (Accessed: 12 May 2016).

Inc, W. (2016) WhatsApp: Home. Available at: https://www.whatsapp.com/ (Accessed: 1 May 2016).

Internet Society. (2015) Global Internet report 2015. Available at: http://www.internetsociety.org/globalinternetreport/?gclid=CIDl3baz78gCFdVAGwodP9INwA (Accessed: 1 May 2016).

Iqbal, R., Azrifah, M., Murad, A., Mustapha, A. and Sharef, N.M. (2013) 'An analysis of Ontology engineering methodologies: A literature review', Research Journal of Applied Sciences, Engineering and Technology, vol. 6 no. 16, pp. 2993–3000.

Jackson, P., Moulinier, I. (2002) Natural Language Processing for Online Applications (Text Retrieval, Extraction and Categorization). John Benjamins Publishing Co. Amsterdam, Netherlands.

Jaishankar, K. (2011) Cyber Criminology: Exploring Internet Crimes and Criminal Behaviour. CRC Press, Boca Raton, USA.

Janowicz, K., Cuenca-Grau, B. and Hoekstra, R. (2010) 'The knowledge Reengineering bottleneck', 1(1).

Java and MySQL - create a login authentication (2016) Available at: http://stackoverflow.com/questions/15165024/java-mysql-create-a-login-authentication (Accessed: 1 May 2016).

Jurisica, I., Mylopoulos, J. and Yu, E. (2004) Ontologies for Knowledge Management: An Information Systems Perspective. Knowledge and Information Systems, vol. 6, no. 4, pp. 380-401.

Jurisica, I., Mylopoulos, J., Yu, E. (2004) Using Ontologies for Knowledge Management: An Information Systems Perspective, in KAISj04

K.L. (1998) How to pass values between classes? (beginning java forum at Coderanch). Available at: http://www.coderanch.com/t/531304/java/java/pass-values-classes (Accessed: 1 May 2016).

Kent and Courier, S. (2014) Gay Scout leader travelled 230 miles to have sex with a teenage boy in Tunbridge wells he groomed on internet. Available at: http://www.courier.co.uk/Gay-Scout-leader-traveled-230-miles-sex-teenage/story-22018626-detail/story.html (Accessed: 1 May 2016).

Khoo, C.S.G. and Na, J. (2006) Semantic relations in information science. Annual Review of Information Science and Technology, vol. 40, no. 1, pp. 157-228.

Klein, E., Potter, S. (2004) An Ontology for NLP Services, School of Informatics. University of Edinburgh, Edinburgh, Scotland

Kless, D., Milton, S., Kazmierczak, E. and Lindenthal, J. (2015) Thesaurus and ontology structure: Formal and pragmatic differences and similarities. Journal of the Association for Information Science and Technology, vol. 66, no. 7, pp. 1348-1366.

Kourtesis, D., Paraskakis, I., Simons, A. (2012) Policy Driven Governance in Cloud Application Platforms: An ontology-based approach, in FOIS 2012.

Ladd, S. (1997) Java Algorithms. McGraw Hill, Berkshire, England

Lassila, O. and Mcguinness, D. (2001) The role of frame-based representation on the semantic web. Available at: http://www.ida.liu.se/ext/epa/ej/etai/2001/018/01018-etaibody.pdf (Accessed: 26 April 2016).

Lehman, J. (2016) Teens and privacy: Should I spy on my child? Plus: The 4 tactics kids use when they get caught. Available at: https://www.empoweringparents.com/article/teens-and-privacy-should-i-spy-on-my-child-plus-the-4-tactics-kids-use-when-they-get-caught/ (Accessed: 1 May 2016).

Liben, L., Muller, U. (2015) Handbook of Child Psychology and Developmental Science, 7th Ed. John Wiley and Sons, New Jersey, USA.

Liefooghe, A. (2012) Deal with Cyber Bullying. Strategic HR Review vol. 11, no. 4

Lincoln, S., Robards, B. (2014) 10 Years of Facebook. New Media and Society vol. 16, no. 7, pp. 1047-1050.

Livingstone, S., Haddon, L., Gorzig, A. Olafsson, K. (2010) Risks and Safety for Children on the Internet: the UK Report. London: LSE, EU Kids Online.

Livingstone, S., Haddon, L., Gorzig, A. Olafsson, K. (2011) Risks and Safety for Children on the Internet: the perspective of European Children. Full Findings. London: LSE, EU Kids Online.

LLC, W.A. (2015) Alpha: Computational knowledge engine. Available at: http://www.wolframalpha.com/ (Accessed: 1 May 2016).

Lord, P., Stevens, R.D., Goble, C.A. and Horrocks, I. (2005) 'Description logics: OWL and DAML+OIL', in Encyclopedia of Genetics, Genomics, Proteomics and Bioinformatics. Wiley-Blackwell.

Luck,M., Ashri, R., D'Inverno, M. (2004) Agent Based Software Development. Artech House Inc. Norwood, MA, USA.

M. Uschold, and M. King. (1995) Proc. of IJCAI95's WS on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada,

Madhu, G. Govardhan, A. Rajinikanth, T. (2011) Intelligent Semantic Web Search Engines: A Brief Survey. International journal of Web and Semantic Technology vol. 2, no. 1.

Mahmoud, Q. Software Agents: Characteristics and Classification, http://www.cis.uoguelph.ca/~qmahmoud/post/agentsoft.pdf.

Maniraj, V., Sivakumar, R. (2010) Ontology Languages – A Review, International Journal of Computer Theory and Engineering vol. 2, pp. 1793-820

Marr, B. (2015) Big Data: Using Smart Big Data, Analytics and Metrics to Make Better Decisions and Improve Performance. John Wiley and Sons Ltd. Chichester, England.

Maynard, D., Li, Y., Peters, w. (2008) NLP Tecniques for Term Extraction and Ontology Population, in Ontology Lerning and Population: Bridging the Gap Between Text and Knowledge, pp. 10-27

Messieh, N. (2010) Top 7 semantic search engines as an alternative to Google. Available at: http://www.makeuseof.com/tag/top-7-semantic-search-engines-alternative-google-search/ (Accessed: 1 May 2016).

Meyer, R. (2014) The new terminology of Snapchat. Available at: http://www.theatlantic.com/technology/archive/2014/05/the-new-terminology-of-snapchat/361651/ (Accessed: 1 May 2016).

Miller, G. (1995) WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

Miller, P. (2011) Theories of Developmental Psychology 5th Ed. Worth Publishers, New York, USA.

Milojicic, D. (2000) Agent systems and applications. IEEE Concurrency, vol. 8, no. 2, pp. 22-23

Minecraft. (2013) Our Blog - Minecraft seeds. Available at: http://minecraft-seeds.net/blog/minecraft-player-demographics/ (Accessed: 1 May 2016).

Morelli, A.O. and Dombeck, M. (no date) Allowing for children's privacy - self-esteem. Available at: http://www.communitycounselingservices.org/poc/view_doc.php?type=docandid=37625andcn=96 (Accessed: 1 May 2016).

Muehlhauser, L., Salamon, A. (2013) Intelligence Explosion: Evidence and Import. In Singularity Hypotheses. Springer.

Murray, R. (2011) How to Write a Thesis 3rd Ed. McGraw Hill, Berkshire, England

ndex.html (no date) Available at: https://gate.ac.uk/ (Accessed: 1 May 2016).

News, S. (2014) Google reports Gmail user's 'child abuse images'. Available at: http://news.sky.com/story/1312967/google-reports-gmail-users-child-abuse-images (Accessed: 1 May 2016)

Noy, N.F. and McGuinness, D.L. (2001) Ontology development 101: A guide to creating your First Ontology. Available at: http://protege.stanford.edu/publications/ontology_development/ontology101.pdf (Accessed: 2 May 2016).

Nwana, H.S. (1996) Software agents: an overview. The Knowledge Engineering Review, vol. 11, no. 3, pp. 205-244.

O'Leary, M. 2009, Wolfram Alpha: Not Quite the Alpha Dog, Information Today, Inc, Medford.

Ontologies for agents - introduction to ontologies and semantic web - tutorial (no date) Available at: http://www.obitko.com/tutorials/ontologies-semantic-web/ontologies-for-agents.html (Accessed: 1 May 2016).

Ontology (information science) (2016) in Wikipedia. Available at: https://en.wikipedia.org/wiki/Ontology_(information_science)#Editor (Accessed: 1 May 2016).

Ontology Support in JADE (no date) Available at: https://web.fe.up.pt/~eol/SOCRATES/Palzer/ontologysupportJADE.htm (Accessed: 1 May 2016).

Ontoprise: Video of the latest features of OntoStudio, a professional modelling environment (no date) Available at: http://www.semafora-systems.com/fileadmin/user_upload/product-videos/OntoStudio/OntoStudio3-Video.html (Accessed: 1 May 2016).

Ontosaurus: Loom web Browser (1998) Available at: http://www.isi.edu/isd/ontosaurus.html (Accessed: 12 May 2016).

OWL API by owlcs (no date) Available at: http://owlcs.github.io/owlapi/ (Accessed: 1 May 2016).

OWL API by owlcs (no date) Available at: http://owlcs.github.io/owlapi/ (Accessed: 2 May 2016)

OWL web Ontology language overview (2004) Available at: https://www.w3.org/TR/owl-features/ (Accessed: 31 May 2016).

P999: What teenage messages really mean (2015) Available at: http://news.sky.com/story/1547640/p999-what-teenage-messages-really-mean (Accessed: 1 May 2016).

Phillips, E., Pugh, D. (2015) How to Get a PhD 6th Ed. McGraw Hill, Berkshire, England

PhPMyAdmin (no date) Available at: http://sost.blackburn.ac.uk/phpmyadmin/index.php?db=kmandtoken=2214d5f4e913ed7d42704fad500c50d4 (Accessed: 1 May 2016).

Picton, P. (2000) Neural Networks 2nd Ed. Palgrave, Hampshire, England

Poggi, A., Rimassa, G., Tomaiuolo, M. (2001) Multi-User and Security Support for Multi-Agent Systems, Online

Pranoto, H., Gunawan, F.E. and Soewito, B. (2015) "Logistic Models for Classifying Online Grooming Conversation",Procedia Computer Science vol. 59, 357-365

Princeton (2016) Citing WordNet - WordNet - citing WordNet. Available at: https://wordnet.princeton.edu/wordnet/citing-wordnet/ (Accessed: 1 May 2016).

Promoting Internet Safety through Public Awareness Campaigns Guidance for Using Real Life Examples Involving Children or Young People Issued by the Home Office Taskforce for Child Protection on the Internet November 2005

Protégé (2016) Available at: http://protege.stanford.edu/ (Accessed: 1 May 2016)

Protégé (2016) Available at: http://protege.stanford.edu/ (Accessed: 1 May 2016).

Rabhi, F., Lapalme, G. (1999) Algorithms: A Functional Programming Approach. Pearson Education, Essex, England.

Rector, A., Drummond, N., Horridge, M., Rogers, J., Knublach, H., Stevens, R., Wang, H., Wroe, c. (2004) OWL Pizzas: Practical Experience of Teachin OWL-DL: Common Errors and Common Patterns

Renkl, M. (2016) A child's need for privacy. Available at: http://www.parenting.com/article/a-childs-need-for-privacy (Accessed: 1 May 2016).

Reporter, D.M. (2010) Teenage girl missing after going to meet a man she had fallen for on Facebook. Available at: http://www.dailymail.co.uk/news/article-1260830/Teenage-Facebook-user-missing-going-meet-man-met-online.html (Accessed: 1 May 2016).

Ridley, D. (2012) The Literature Review: A Step-by-Step Guide for Students 2nd ED. Sage Publications LTD. London, United Kingdom

Rouse, M. (2006) What is semantic web? - definition from WhatIs.com. Available at: http://searchsoa.techtarget.com/definition/Semantic-Web (Accessed: 1 May 2016).

Rydel, T.J., Ravichandran, K.G., Tulinsky, A., Bode, W. and Huber, R. 1990, "The Structure of a Complex of Recombinant Hirudin and Human (alpha)-Thrombin",Science, vol. 249, no. 4966, pp. 277.

S, V.M. (2016) Free calls and messages. Available at: http://www.viber.com/en/ (Accessed: 1 May 2016).

Sales, T., Barcewlos, P., Guizzardi, G. (2012) Identification of Semantic Anti-Patterns in Ontology-Driven Conceptual Modelling via Visual Simulation, in FOIS 2012.

Sample, I. (2016) Neuroscientists create ‘atlas’ showing how words are organised in the brain. Available at: https://www.theguardian.com/science/2016/apr/27/brain-atlas-showing-how-words-are-organised-neuroscience?CMP=fb_gu (Accessed: 21 July 2016).

Schindler, W. 2004, Silanterminierte Polyether mit Alpha-effekt. Adhaesion Kleben und Dichten, vol. 48, no. 12, pp. 28.

Shaffer, D., Kipp, K. (2014) Developmental Psychology: Childhood and Adolesence 9th Ed. Wadsworth, California, USA.

Sheldon, F., Elmore, M., Potol, T. (2003) An Ontology-Based Agent System Case Study. Applied Software Engineering Research Group, Oak Ridge National Lab.

Sicilia, M., García-Barriocanal, E., Sánchez-Alonso, S. and Rodríguez-García, D. (2009) Ontologies of engineering knowledge: general structure and the case of Software Engineering. The Knowledge Engineering Review, vol. 24, no. 3, pp. 309-326

Simon, J., Dos Santos, M., Fielding, J. and Smith, B. (2006) Formal ontology for natural language processing and the integration of biomedical databases. International Journal of Medical Informatics, vol. 75, no. 3, pp. 224-231

Site, J. (2015) JAVA agent dEvelopment framework. Available at: http://jade.tilab.com/ (Accessed: 1 May 2016).

Site, J. (2016) Add-Ons. Available at: http://jade.tilab.com/download/add-ons/ (Accessed: 1 May 2016).

Soares, N. (2015) Formalizing Two Problems of Realistic World-Models. Technical report 2015–3. MIRI

Soares, N., Fallenstein, B. (2015) Toward Idealised Decision Theory. Machine Intelligence Learning Institute.

Social media domestic abusers face jail (2015) Available at: http://news.sky.com/story/1613587/social-media-domestic-abusers-face-jail (Accessed: 1 May 2016).

Solidtuber (2013) Java GUI with MySQL database for login authentication error. Available at: http://www.dreamincode.net/forums/topic/330773-java-gui-with-mysql-database-for-login-authentication-error/ (Accessed: 1 May 2016).

Sowa, J.F. (1999) Knowledge representation: logical, philosophical, and computational foundations. PWS, Boston, Mass; London

Spence, I. (2005) What is iterative development? Available at: http://www.ibm.com/developerworks/rational/library/may05/bittner-spence/ (Accessed: 1 August 2016).

Staff writer. (2014) Lake county man arrested for sexual contact with minor he met online. Available at: http://legacy.wkyc.com/story/news/local/lake-county/2014/07/23/lake-county-man-arrested-for-sexual-contact-with-minor-he-met-online/13047595/ (Accessed: 1 May 2016).

Strickland, J. (2006) How Google works. Available at: http://computer.howstuffworks.com/internet/basics/google1.htm (Accessed: 1 May 2016).

Stula, M., Stipanicev, D., Maras, J. (2013) Distributed Computation Multi-Agent System. New Generation Computing vol. 31, pp. 187-209

Swartout B, Patil R, Knight K, Russ T. Towards distributed use of large-scale ontologies. In: Spring Symposium Series on Ontological Engineering, Stanford University, CA, 1997, p 138-148.

Systems, W.M. (2006) User authentication using mySQL and JDBC. Available at: https://www.wowza.com/forums/showthread.php?1365-User-Authentication-Using-mySQL-and-JDBC (Accessed: 1 May 2016).

Tagg, C. (2012) The Discourse of Text Messaging: Analysis of SMS Communication. Continuum International Pub. Group, London.

Tanha (2008) Software development discussion community. Available at: https://www.daniweb.com/programming/software-development/threads/119702/login-form-verification-mysql-java (Accessed: 1 May 2016).

TED (2015) Monica Lewinsky: The price of shame. Available at: https://www.youtube.com/watch?v=H_8y0WLm78U (Accessed: 1 May 2016).

Tolentino, J. (2015) Why are people still using SMS in 2015? Available at: http://thenextweb.com/future-of-communications/2015/02/16/people-still-using-sms-2015/ (Accessed: 1 May 2016).

Torrance, S. (2013) Artificial agents and the expanding ethical circle. AI and SOCIETY. vol. 28, no. 4, pp. 399-414

Turing, A.M. (1950) Computing Machinery and Intelligence. Mind, LIX (236), pp. 433–460

UKCCIS (2012) Online Offending Behaviour: Findings from the European Online Grooming Project, Online

UKCCIS (2012) Young People's Online Behaviour: Findings from the European Online Grooming Project, Online

Unicef (no date) What is the UNCRC? | children's rights | Unicef UK. Available at: http://www.unicef.org.uk/UNICEFs-Work/UN-Convention/ (Accessed: 1 May 2016).

Uschold, M. Gruniger, M. (1996) Ontologies Principles Methods and Applications. Knowledge Engineering Review, Vol. 11 No. 2

van Aart, C., Pels, R. (2002) Creating and Using Ontologies in Agent Communication

van der Hof, S., van den Berg, B., Schermer, B. (2014) Minding Minors Wandering the Web: Regulating Online Child Safety, T.M.C. Asser Press, Dordrecht.

van der Vet, P. Mars, N. (1998) Bottom-Up Construction of Ontologies. IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 10, NO. 4, PP. 513-526

Van Gurp, M., Holvoet, M., Casella, M. and Santos (2008) LinKBase® and SNOMED: Some distinct features and impact on NLP. Available at: http://ceur-ws.org/Vol-410/Paper05.pdf (Accessed: 19 July 2016)

Vila, X., Schuuster, A., Riera, A. (2007) Security for a Multi-Agent System Based on JADE. Computers and Security vol. 26, pp. 391-400

Wallace, M., Wray, A. (2011) Critical Reading and Writing for Postgraduates 2nd Ed. Sage Publications LTD. London, United Kingdom.

Ward, M. (2013) Why Minecraft is more than just another video game. Available at: http://www.bbc.co.uk/news/magazine-23572742 (Accessed: 1 May 2016).

WebODE (2016) Available at: http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/old-technologies/60-webode/ (Accessed: 12 May 2016).

Weinberger, M. (2015) The 11 top-grossing video games of all time. Available at: http://www.techinsider.io/the-11-top-grossing-video-games-of-all-time-2015-8 (Accessed: 1 May 2016).

Weiss, G. (1999) Multi-Agent Systems (A Modern Approach to Distributed Artificial Intelligence). The Massachusetts Institute of Technology, USA.

What is an Ontology? (no date) Available at: http://www.cs.man.ac.uk/~stevensr/onto/node3.html (Accessed: 1 May 2016).

What is IBM Watson? (no date) Available at: http://www.ibm.com/smarterplanet/us/en/ibmwatson/what-is-watson.html (Accessed: 1 May 2016).

Winson, A. (2000) Wiley higher education supplementary Website. Available at: http://www.wiley.com/legacy/wileychi/bellifemine_jade/ (Accessed: 1 May 2016).

WordNet search - 3.1 (2016) Available at: http://wordnetweb.princeton.edu/perl/webwn?s=ontologyandsub=Search+WordNetando2=ando0=1ando8=1ando1=1ando7=ando5=ando9=ando6=ando3=ando4=andh= (Accessed: 19 July 2016).

Yang, S., Lee, D. and Chen, K. (2011) A new ubiquitous information agent system for cloud computing - Example on GPS and Bluetooth techniques in Google Android platform. pp. 1929

Zhao, C., Wysocki, B.T., Liu, Y., Thiem, C.D., McDonald, N. and Yi, Y. (2015) Spike-Time-Dependent Encoding for Neuromorphic Processors. ACM Journal on Emerging Technologies in Computing Systems (JETC), vol. 12, no. 3, pp. 1-21.

Zitouni, I. (2014) Natural Language Processing of Semitic Languages. Springer, London.

Appendices

A. Exported Ontology Files
B. Parent Survey Results
C. Project Plans
D. Testing Resources

Appendix A – Exported Ontology Files

```
package Meeting;

import java.util.Collection;

import org.protege.owl.codegeneration.WrappedIndividual;

import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLOntology;

/**
 *
 * <p>
 * Generated by Protege (http://protege.stanford.edu). <br>
 * Source Class: Intention <br>
 * @version generated on Mon Aug 03 09:17:45 BST 2015 by
Katrinna
 */

public interface Intention extends Meeting {

    /* ***************************************************
     * Common interfaces
     */

    OWLNamedIndividual getOwlIndividual();

    OWLOntology getOwlOntology();

    void delete();

}
```

```
package Meeting;

import java.util.Collection;

import org.protege.owl.codegeneration.WrappedIndividual;

import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLOntology;

/**
 *
 * <p>
 * Generated by Protege (http://protege.stanford.edu). <br>
 * Source Class: Location <br>
 * @version generated on Mon Aug 03 09:17:45 BST 2015 by
Katrinna
 */

public interface Location extends Meeting {

    /* ***************************************************
     * Common interfaces
     */

    OWLNamedIndividual getOwlIndividual();

    OWLOntology getOwlOntology();

    void delete();

}
```

```
package Meeting;

import java.util.Collection;

import org.protege.owl.codegeneration.WrappedIndividual;

import org.semanticweb.owlapi.model.OWLNamedIndividual;
import org.semanticweb.owlapi.model.OWLOntology;

/**
 *
 * <p>
 * Generated by Protege (http://protege.stanford.edu). <br>
 * Source Class: Time <br>
 * @version generated on Mon Aug 03 09:17:45 BST 2015 by
Katrinna
 */

public interface Time extends Meeting {

    /* ***************************************************
     * Common interfaces
     */

    OWLNamedIndividual getOwlIndividual();

    OWLOntology getOwlOntology();

    void delete();

}
```

```
package Meeting;

import org.semanticweb.owlapi.apibinding.OWLManager;
import org.semanticweb.owlapi.model.IRI;
import org.semanticweb.owlapi.model.OWLClass;
import org.semanticweb.owlapi.model.OWLDataFactory;
import org.semanticweb.owlapi.model.OWLDataProperty;
import org.semanticweb.owlapi.model.OWLObjectProperty;

/**
 * Vocabulary class to provide access to the Manchester OWL
API representatives for
 * various entities in the ontology used to generate this
code.<p>
 *
 * Generated by Protege (http://protege.stanford.edu).<br>
 * Source Class: ${javaClass}
 *
 * @version generated on Mon Aug 03 09:17:45 BST 2015 by
Katrinna
 */

public class Vocabulary {

     private static final OWLDataFactory factory =
OWLManager.createOWLOntologyManager().getOWLDataFactory();

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Apollo
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class APOLLO.<p>
     *
     */
    public static final OWLClass CLASS_APOLLO =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Apollo"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Cinema
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class CINEMA.<p>
     *
     */
    public static final OWLClass CLASS_CINEMA =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Cinema"));
```

```
    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Football_match
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class FOOTBALL_MATCH.<p>
     *
     */
    public static final OWLClass CLASS_FOOTBALL_MATCH =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Football_match"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Intention
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class INTENTION.<p>
     *
     */
    public static final OWLClass CLASS_INTENTION =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Intention"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Location
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class LOCATION.<p>
     *
     */
    public static final OWLClass CLASS_LOCATION =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Location"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Macdonalds
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class MACDONALDS.<p>
     *
     */
    public static final OWLClass CLASS_MACDONALDS =
```

```
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Macdonalds"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Meeting
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class MEETING.<p>
     *
     */
    public static final OWLClass CLASS_MEETING =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Meeting"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Movies
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class MOVIES.<p>
     *
     */
    public static final OWLClass CLASS_MOVIES =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Movies"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Restaurant
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class RESTAURANT.<p>
     *
     */
    public static final OWLClass CLASS_RESTAURANT =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Restaurant"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Time
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class TIME.<p>
```

```
     *
     */
    public static final OWLClass CLASS_TIME =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Time"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#Vue
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class VUE.<p>
     *
     */
    public static final OWLClass CLASS_VUE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#Vue"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#asda
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ASDA.<p>
     *
     */
    public static final OWLClass CLASS_ASDA =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#asda"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#book_store
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class BOOK_STORE.<p>
     *
     */
    public static final OWLClass CLASS_BOOK_STORE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#book_store"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#borders
     */
```

```
    /**
     * A constant to give access to the Manchester OWL api
representation of the class BORDERS.<p>
     *
     */
    public static final OWLClass CLASS_BORDERS =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#borders"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#burger_king
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class BURGER_KING.<p>
     *
     */
    public static final OWLClass CLASS_BURGER_KING =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#burger_king"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#burger_place
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class BURGER_PLACE.<p>
     *
     */
    public static final OWLClass CLASS_BURGER_PLACE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#burger_place"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#cafe
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class CAFE.<p>
     *
     */
    public static final OWLClass CLASS_CAFE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#cafe"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
```

```
5586.owl#do_you_think_we_should_meet
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class DO_YOU_THINK_WE_SHOULD_MEET.<p>
     *
     */
    public static final OWLClass
CLASS_DO_YOU_THINK_WE_SHOULD_MEET =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#do_you_think_we_shoul
d_meet"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#flicks
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class FLICKS.<p>
     *
     */
    public static final OWLClass CLASS_FLICKS =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#flicks"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#footie
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class FOOTIE.<p>
     *
     */
    public static final OWLClass CLASS_FOOTIE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#footie"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#game
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class GAME.<p>
     *
     */
    public static final OWLClass CLASS_GAME =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
```

```
ologies/2010/0/Ontology1264435885586.owl#game"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#game_exchange
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class GAME_EXCHANGE.<p>
     *
     */
    public static final OWLClass CLASS_GAME_EXCHANGE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#game_exchange"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#game_store
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class GAME_STORE.<p>
     *
     */
    public static final OWLClass CLASS_GAME_STORE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#game_store"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#hmv
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class HMV.<p>
     *
     */
    public static final OWLClass CLASS_HMV =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#hmv"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#in_an_hour
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class IN_AN_HOUR.<p>
     *
```

```
     */
    public static final OWLClass CLASS_IN_AN_HOUR =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#in_an_hour"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#in_five_minutes
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class IN_FIVE_MINUTES.<p>
     *
     */
    public static final OWLClass CLASS_IN_FIVE_MINUTES =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#in_five_minutes"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#italian
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ITALIAN.<p>
     *
     */
    public static final OWLClass CLASS_ITALIAN =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#italian"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#kfc
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class KFC.<p>
     *
     */
    public static final OWLClass CLASS_KFC =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#kfc"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#lets_meet
     */

    /**
```

```
     * A constant to give access to the Manchester OWL api
representation of the class LETS_MEET.<p>
     *
     */
    public static final OWLClass CLASS_LETS_MEET =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#lets_meet"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#meet_me
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class MEET_ME.<p>
     *
     */
    public static final OWLClass CLASS_MEET_ME =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#meet_me"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#morrisons
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class MORRISONS.<p>
     *
     */
    public static final OWLClass CLASS_MORRISONS =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#morrisons"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#music_store
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class MUSIC_STORE.<p>
     *
     */
    public static final OWLClass CLASS_MUSIC_STORE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#music_store"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#next_week
```

```
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class NEXT_WEEK.<p>
     *
     */
    public static final OWLClass CLASS_NEXT_WEEK =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#next_week"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#on_friday
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ON_FRIDAY.<p>
     *
     */
    public static final OWLClass CLASS_ON_FRIDAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#on_friday"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#on_monday
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ON_MONDAY.<p>
     *
     */
    public static final OWLClass CLASS_ON_MONDAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#on_monday"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#on_saturday
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ON_SATURDAY.<p>
     *
     */
    public static final OWLClass CLASS_ON_SATURDAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#on_saturday"));

    /* ************************************************
```

```
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#on_sunday
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ON_SUNDAY.<p>
     *
     */
    public static final OWLClass CLASS_ON_SUNDAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#on_sunday"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#on_teusday
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ON_TEUSDAY.<p>
     *
     */
    public static final OWLClass CLASS_ON_TEUSDAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#on_teusday"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#on_thursday
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ON_THURSDAY.<p>
     *
     */
    public static final OWLClass CLASS_ON_THURSDAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#on_thursday"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#on_wednesday
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class ON_WEDNESDAY.<p>
     *
     */
    public static final OWLClass CLASS_ON_WEDNESDAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
```

```
ologies/2010/0/Ontology1264435885586.owl#on_wednesday"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#park
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class PARK.<p>
     *
     */
    public static final OWLClass CLASS_PARK =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#park"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#pictures
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class PICTURES.<p>
     *
     */
    public static final OWLClass CLASS_PICTURES =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#pictures"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#pizza_hut
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class PIZZA_HUT.<p>
     *
     */
    public static final OWLClass CLASS_PIZZA_HUT =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#pizza_hut"));

    /* ***************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#pizza_place
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class PIZZA_PLACE.<p>
     *
```

```
     */
    public static final OWLClass CLASS_PIZZA_PLACE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#pizza_place"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#safeway
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class SAFEWAY.<p>
     *
     */
    public static final OWLClass CLASS_SAFEWAY =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#safeway"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#see_you_at
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class SEE_YOU_AT.<p>
     *
     */
    public static final OWLClass CLASS_SEE_YOU_AT =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#see_you_at"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#shopping
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class SHOPPING.<p>
     *
     */
    public static final OWLClass CLASS_SHOPPING =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#shopping"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#spaghettii_house
     */

    /**
```

```
     * A constant to give access to the Manchester OWL api
representation of the class SPAGHETTII_HOUSE.<p>
     *
     */
    public static final OWLClass CLASS_SPAGHETTII_HOUSE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#spaghettii_house"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#supermarket
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class SUPERMARKET.<p>
     *
     */
    public static final OWLClass CLASS_SUPERMARKET =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#supermarket"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#tesco
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class TESCO.<p>
     *
     */
    public static final OWLClass CLASS_TESCO =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#tesco"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#the_game
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class THE_GAME.<p>
     *
     */
    public static final OWLClass CLASS_THE_GAME =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#the_game"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#the_match
```

```
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class THE_MATCH.<p>
     *
     */
    public static final OWLClass CLASS_THE_MATCH =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#the_match"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#the_park
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class THE_PARK.<p>
     *
     */
    public static final OWLClass CLASS_THE_PARK =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#the_park"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#the_slide
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class THE_SLIDE.<p>
     *
     */
    public static final OWLClass CLASS_THE_SLIDE =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#the_slide"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#the_swings
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class THE_SWINGS.<p>
     *
     */
    public static final OWLClass CLASS_THE_SWINGS =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#the_swings"));

    /* ************************************************
```

```
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#the_wreck
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class THE_WRECK.<p>
     *
     */
    public static final OWLClass CLASS_THE_WRECK =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#the_wreck"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#tomorrow
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class TOMORROW.<p>
     *
     */
    public static final OWLClass CLASS_TOMORROW =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#tomorrow"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#tomoz
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class TOMOZ.<p>
     *
     */
    public static final OWLClass CLASS_TOMOZ =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#tomoz"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#wanna_meet_up_at
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class WANNA_MEET_UP_AT.<p>
     *
     */
    public static final OWLClass CLASS_WANNA_MEET_UP_AT =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
```

```
ologies/2010/0/Ontology1264435885586.owl#wanna_meet_up_at"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#we_could_meet_at
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class WE_COULD_MEET_AT.<p>
     *
     */
    public static final OWLClass CLASS_WE_COULD_MEET_AT =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#we_could_meet_at"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#weekend
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class WEEKEND.<p>
     *
     */
    public static final OWLClass CLASS_WEEKEND =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#weekend"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#wh_smiths
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class WH_SMITHS.<p>
     *
     */
    public static final OWLClass CLASS_WH_SMITHS =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#wh_smiths"));

    /* ************************************************
     * Class
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#whsmiths
     */

    /**
     * A constant to give access to the Manchester OWL api
representation of the class WHSMITHS.<p>
     *
```

```
     */
    public static final OWLClass CLASS_WHSMITHS =
factory.getOWLClass(IRI.create("http://www.semanticweb.org/ont
ologies/2010/0/Ontology1264435885586.owl#whsmiths"));

    /* ***************************************************
     * Object Property
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#hasComponent
     */

    /**
     * A constant to give access to the Manchester OWL API
representation of the object property HASCOMPONENT.<p>
     *
     */
    public static final OWLObjectProperty
OBJECT_PROPERTY_HASCOMPONENT =
factory.getOWLObjectProperty(IRI.create("http://www.semanticwe
b.org/ontologies/2010/0/Ontology1264435885586.owl#hasComponent
"));

    /* ***************************************************
     * Object Property
http://www.semanticweb.org/ontologies/2010/0/Ontology126443588
5586.owl#isComponent
     */

    /**
     * A constant to give access to the Manchester OWL API
representation of the object property ISCOMPONENT.<p>
     *
     */
    public static final OWLObjectProperty
OBJECT_PROPERTY_ISCOMPONENT =
factory.getOWLObjectProperty(IRI.create("http://www.semanticwe
b.org/ontologies/2010/0/Ontology1264435885586.owl#isComponent"
));

    /* ***************************************************
     * Object Property
http://www.w3.org/2002/07/owl#topObjectProperty
     */

    /**
     * A constant to give access to the Manchester OWL API
representation of the object property TOPOBJECTPROPERTY.<p>
     *
     */
    public static final OWLObjectProperty
OBJECT_PROPERTY_TOPOBJECTPROPERTY =
factory.getOWLObjectProperty(IRI.create("http://www.w3.org/200
2/07/owl#topObjectProperty"));

    /* ***************************************************
     * Data Property
http://www.w3.org/2002/07/owl#topDataProperty
```

```
     */

    /**
     * A constant to give access to the Manchester OWL API
representation of the data property TOPDATAPROPERTY.<p>
     *
     */
    public static final OWLDataProperty
DATA_PROPERTY_TOPDATAPROPERTY =
factory.getOWLDataProperty(IRI.create("http://www.w3.org/2002/
07/owl#topDataProperty"));


}
```

Appendix B – Survey Responses

Surveys for Pages

Results

You can view 11 more answers with the free plan. Upgrade to a Premium plan to remove this limitation.

How many children do you have? 29 answers (0 locked) View as pie chart	1-3	18 votes	62.1%
	1	6 votes	20.7%
	4 or more	5 votes	17.2%
Do you have children under 18? 29 answers (0 locked) View as pie chart	Yes	23 votes	79.3%
	No	6 votes	20.7%
Do you worry about the safety of your child(ren) when they are on-line? 29 answers (0 locked) View as pie chart	Yes	23 votes	79.3%
	No	6 votes	20.7%

Please help us understand why you selected the answer above?

28 answers (0 locked)

I do limit and monitor currenlty what the children have access to as although we have discussed internet safety and this is also discussed at school I feel they are still too young to fully unsderstand the dangers that are out there online

now old enough to make own choices

I realise that the internet can be abused by those wishing to do harm

More access to Internet rather than just PCs as it was years ago

There are a lot of horror stories in the media of online predators and how they use social media to groom children.

Too easy for an adult to groom children, pretending to be child themselves

Immaturity of children can expose them to a lot of wrong stuff out there! !!!

I worry about strangers on Facebook, adding my daughter.

It is not just about the anonymity of other users online i.e. in chat rooms but the free availability of inappropriate content and the increasing rise in social media and apps such as snapchat, Instagram, Pinterest where they can post things that could potentially affect their lives off line through bullying and then later on employment prospects because they are too naive to understand exactly what it is they are putting out there and for how long.

You don't know who's on the other side of any Internet interactions

Displaying the last 10 answers. **View more**

What age do you think is safe for your child to have free access to the internet?

29 answers (0 locked)

View as pie chart

Over 15	10 votes	34.5%
12-15	9 votes	31.0%
Other	7 votes	24.1%
10-12	2 votes	6.9%
Under 10	1 vote	3.4%

Does your child have a Facebook account?

29 answers (0 locked)

View as pie chart

No	18 votes	62.1%
Yes	11 votes	37.9%

What age do you think is safe for your child to have a Facebook account?

29 answers (0 locked)

View as pie chart

Over 15	14 votes	48.3%
12-15	11 votes	37.9%
Other	2 votes	6.9%
10-12	2 votes	6.9%
Under 10	0 votes	0%

Have you discussed internet safety with your child?

29 answers (0 locked)

View as pie chart

Yes	22 votes	75.9%
I didn't have to - my kids know more about this than I do!	3 votes	10.3%
No	3 votes	10.3%
I wouldn't know where to start.	1 vote	3.4%
I didn't have to - the school did this for me.	0 votes	0%

Does your child have a mobile phone?

29 answers (0 locked)

View as pie chart

Yes	20 votes	69.0%
No	9 votes	31.0%

What age do you think is safe for your child to have a mobile phone?

29 answers (0 locked)

View as pie chart

12-15	12 votes	41.4%
10-12	12 votes	41.4%
Over 15	3 votes	10.3%
Other	1 vote	3.4%
Under 10	1 vote	3.4%

Have you used or heard of the following applications? Please select all that apply.

29 answers (0 locked) (156 votes)

View as pie chart

WhatsApp	28 votes	96.6%
Facebook messenger	28 votes	96.6%
Skype	28 votes	96.6%
Snapchat	27 votes	93.1%
Face Time / iMessage	26 votes	89.7%
Viber	19 votes	65.5%

Which of these would you be happy for your kids to use? Please select all that apply

29 answers (0 locked) (58 votes)

View as pie chart

Face Time / iMessage	13 votes	44.8%
WhatsApp	12 votes	41.4%
Skype	11 votes	37.9%
None of the above	7 votes	24.1%
Facebook messenger	7 votes	24.1%
Snapchat	5 votes	17.2%
Viber	3 votes	10.3%

Has your child ever been negatively affected by anything that happened on-line?

29 answers (0 locked)

View as pie chart

No	25 votes	86.2%
Yes	3 votes	10.3%
I prefer not to answer	1 vote	3.4%

How important is your child's privacy to you

29 answers (0 locked) with a median of 4.8

View as pie chart

1	0 votes	0%
2	0 votes	0%
3	3 votes	10.3%
4	1 vote	3.4%
5	25 votes	86.2%

Please help us by explaining your answer

29 answers (0 locked)

I cannot tick the above question but the childrens privacy is extremely important to me

internet use should be controlled or supervised until children know how to protect themselves

All children need privacy, research suggest that if a child does not have privacy, even from their parents, it can be damaging to their development

If you wouldn't do it in real life don't do it in social medua

Ultimately a lack of respect for a child's privacy can erode the bond between parent and child, with many children who are spied upon by their parents often losing trust in those parents. It's a recognised issue for many families, and there are several groups that are actually in part dedicated towards addressing this.

Find a balance between protecting them yet also allowing an element of privacy

My child is only 3 still but I get worried at times about the random videos that appear in the suggested section of you tube!!! As well as impact on health due to using Internet.

I feel all children should be able to feel safe on line

As stated previously it is important that children do not naively put things out there that may impact on their future lives negatively.

I want my children to be able to use the Internet, which makes their privacy a primary concern and necessity

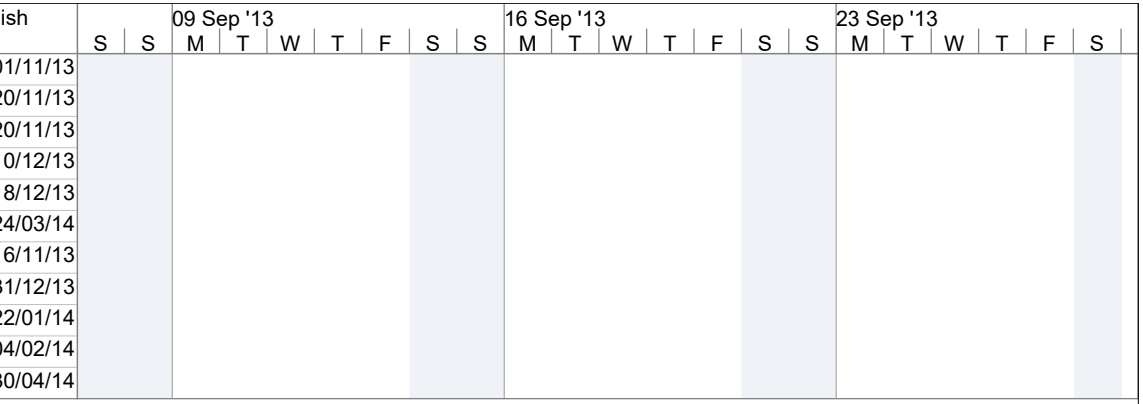
Displaying the last 10 answers. **View more**

Statistics **Participants** **Share results** **All surveys**

Account Contact us FAQ Examples

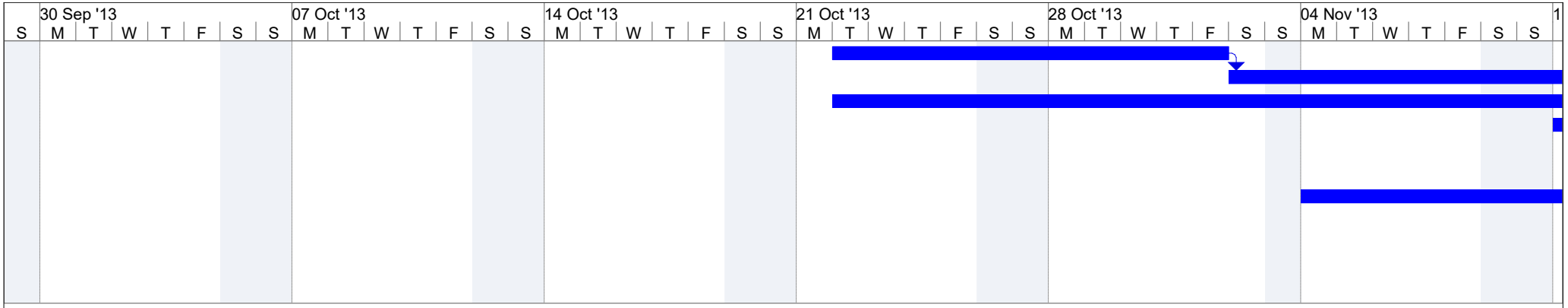
Appendix C – Project Plans

ID	Task Name	Duration	Start	Finish
1	Database Connection	9 days	Tue 22/10/13	Fri 01/11/13
2	Agent Integration with Data/Chat	15 days?	Sat 02/11/13	Wed 20/11/13
3	Develop Threat Level Agent	24 days?	Tue 22/10/13	Wed 20/11/13
4	Modify crisp rules -> fuzzy rules	24 days?	Mon 11/11/13	Tue 10/12/13
5	Finalise class relationships in the ontology	24 days?	Mon 18/11/13	Wed 18/12/13
6	System Testing Starts	69 days?	Thu 19/12/13	Mon 24/03/14
7	Different Ontology Reasoners	11 days?	Mon 04/11/13	Sat 16/11/13
8	Crisp vs Fuzzy rules in ThreatAgent	11 days?	Wed 18/12/13	Tue 31/12/13
9	Test Conversations	16 days?	Wed 01/01/14	Wed 22/01/14
10	Real Conversations	9 days?	Thu 23/01/14	Tue 04/02/14
11	Finalise Writeup	28 days	Mon 24/03/14	Wed 30/04/14



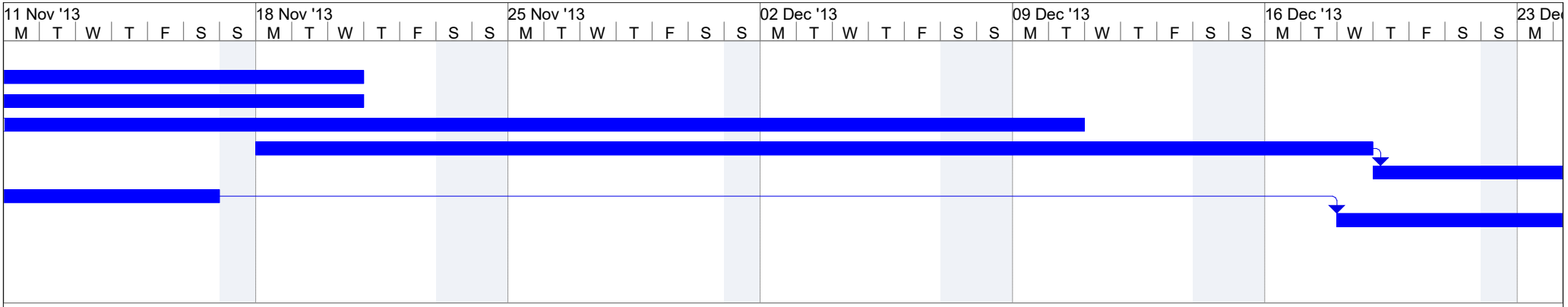
Project: Latest Project Plan.mpp
Date: Sat 03/09/16

Task	Inactive Milestone	Finish-only
Split	Inactive Summary	External Tasks
Milestone	Manual Task	External Milestone
Summary	Duration-only	Progress
Project Summary	Manual Summary Rollup	Deadline
External Tasks	Manual Summary	
External Milestone	Start-only	



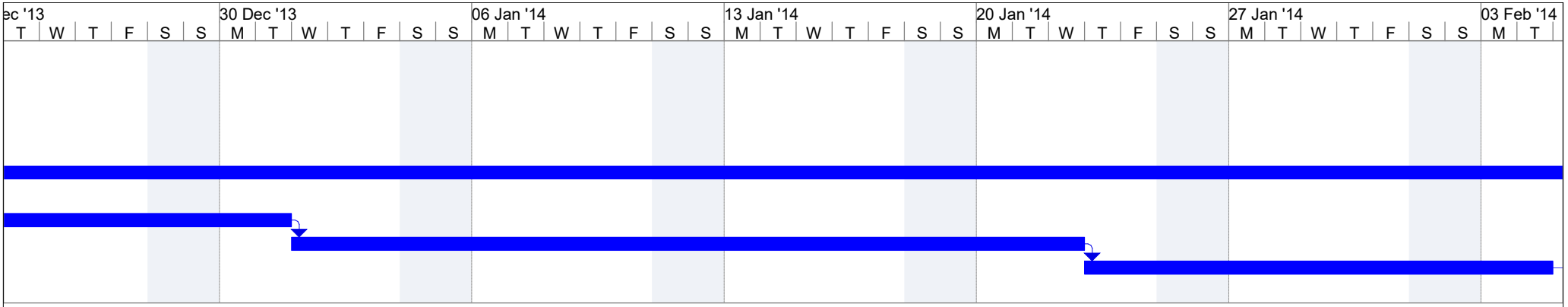
Project: Latest Project Plan.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



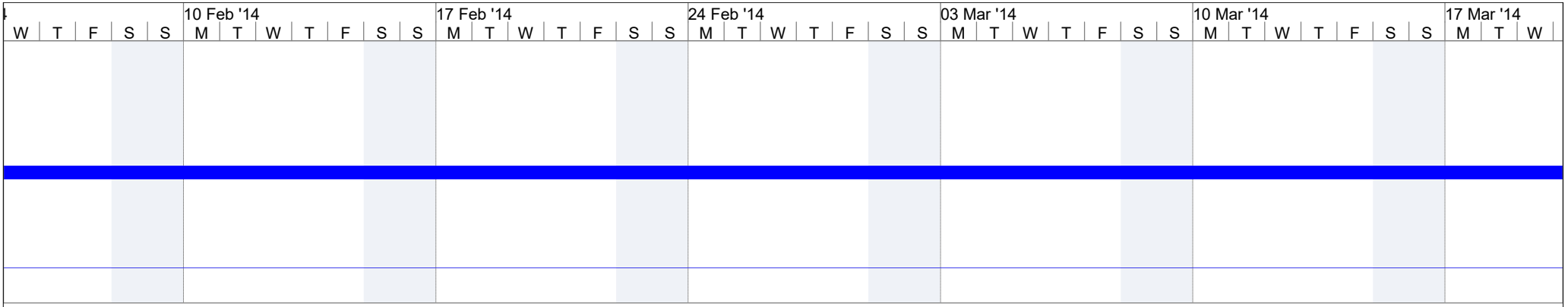
Project: Latest Project Plan.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



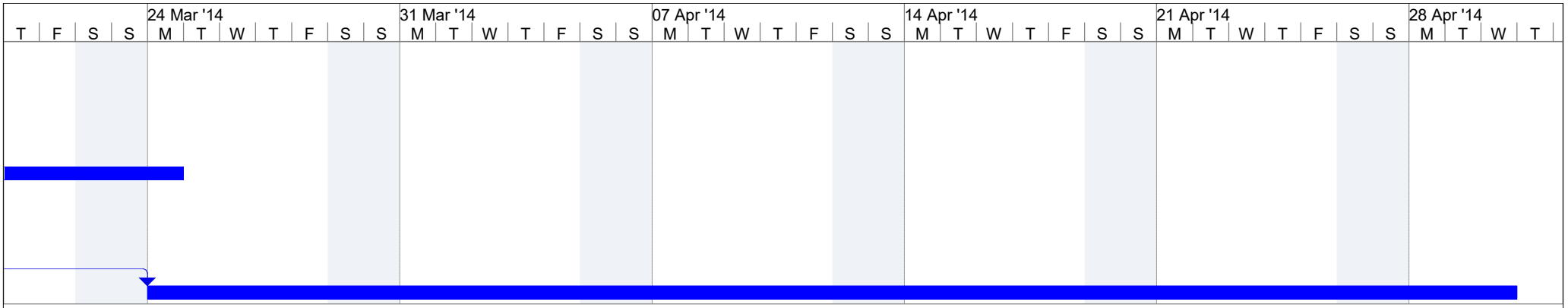
Project: Latest Project Plan.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



Project: Latest Project Plan.mpp
Date: Sat 03/09/16

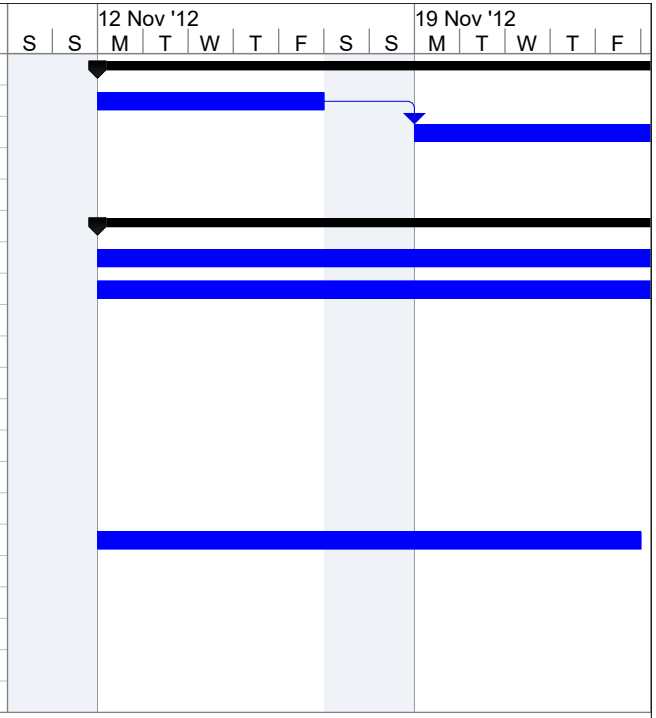
Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



Project: Latest Project Plan.mpp
Date: Sat 03/09/16

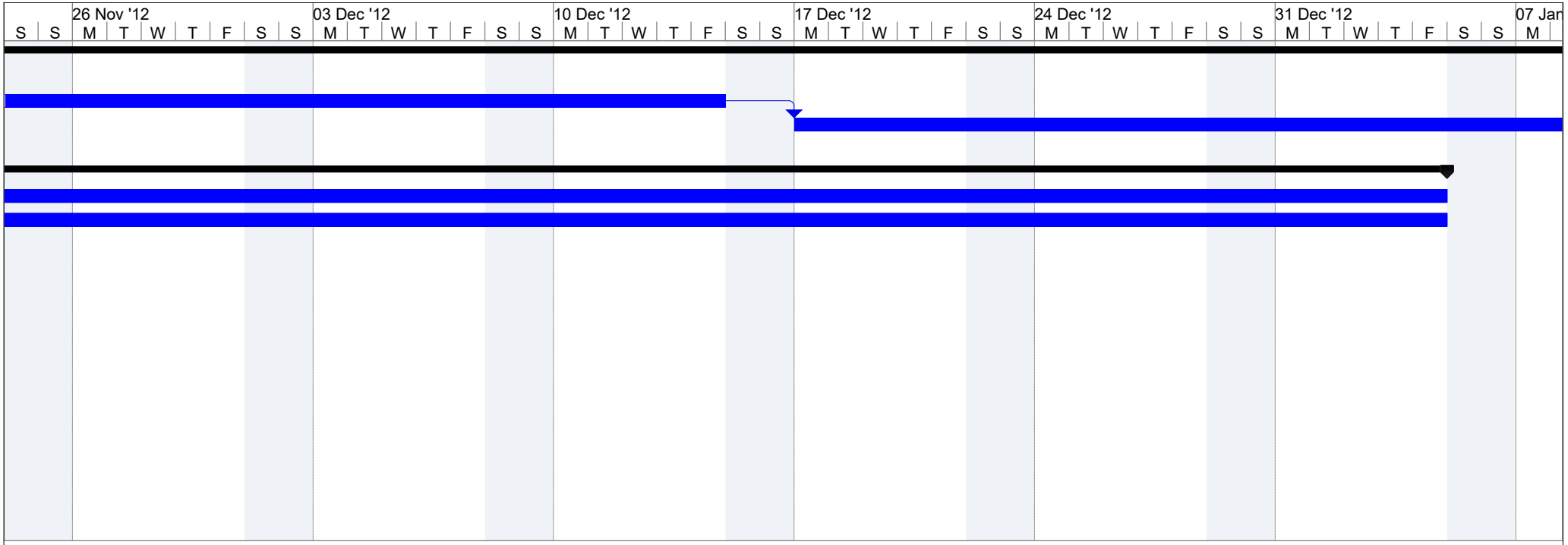
Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			

ID	Task Name	Duration	Start	Finish	Predecessors
1	**Generate working chat system**	**45 days**	**Mon 12/11/12**	**Fri 11/01/13**	
2	Create mobile develoment suite (NetBeans, JADE, Protégé)	1 wk	Mon 12/11/12	Fri 16/11/12	
3	Develop GUI's and integrate existing agents	4 wks	Mon 19/11/12	Fri 14/12/12	2
4	Connect to database to enable registration	4 wks	Mon 17/12/12	Fri 11/01/13	3
5	Write up progress to thesis - First Stage	1 wk	Mon 14/01/13	Fri 18/01/13	4
6	**Produce a working Ontology**	**40 days**	**Mon 12/11/12**	**Fri 04/01/13**	
7	Kate - Design a test set large enough to produce valid test results	8 wks	Mon 12/11/12	Fri 04/01/13	
8	Violeta - Enlist help to auto populate	8 wks	Mon 12/11/12	Fri 04/01/13	
9	Write up progress to thesis - Second Stage	1 wk	Mon 21/01/13	Fri 25/01/13	5
10	*Produce Ontology Paper*	2 wks	Mon 28/01/13	Fri 08/02/13	9
11	**Re-engineer MAS**	**40 days**	**Mon 28/01/13**	**Fri 22/03/13**	**5**
12	Meeting detection agent	2 wks	Mon 28/01/13	Fri 08/02/13	9
13	Threat level icrementation and notification	3 wks	Mon 11/02/13	Fri 01/03/13	12
14	Ontology integration	3 wks	Mon 04/03/13	Fri 22/03/13	13
15	Write up progress to thesis - Third Stage	1 wk	Mon 25/03/13	Fri 29/03/13	14
16	*Produce Paper on System Enhancements*	2 wks	Mon 12/11/12	Fri 23/11/12	
17	**Mobile system adaptation**	**50 days**	**Mon 01/04/13**	**Fri 07/06/13**	
18	Research JADE Leap	4 wks	Mon 01/04/13	Fri 26/04/13	15
19	Export system onto mobile platforms	4 wks	Mon 29/04/13	Fri 24/05/13	18
20	*Produce Mobile Platform Paper*	2 wks	Mon 27/05/13	Fri 07/06/13	19
21	**Complete and submit Thesis**	10 wks	Mon 10/06/13	Fri 16/08/13	20



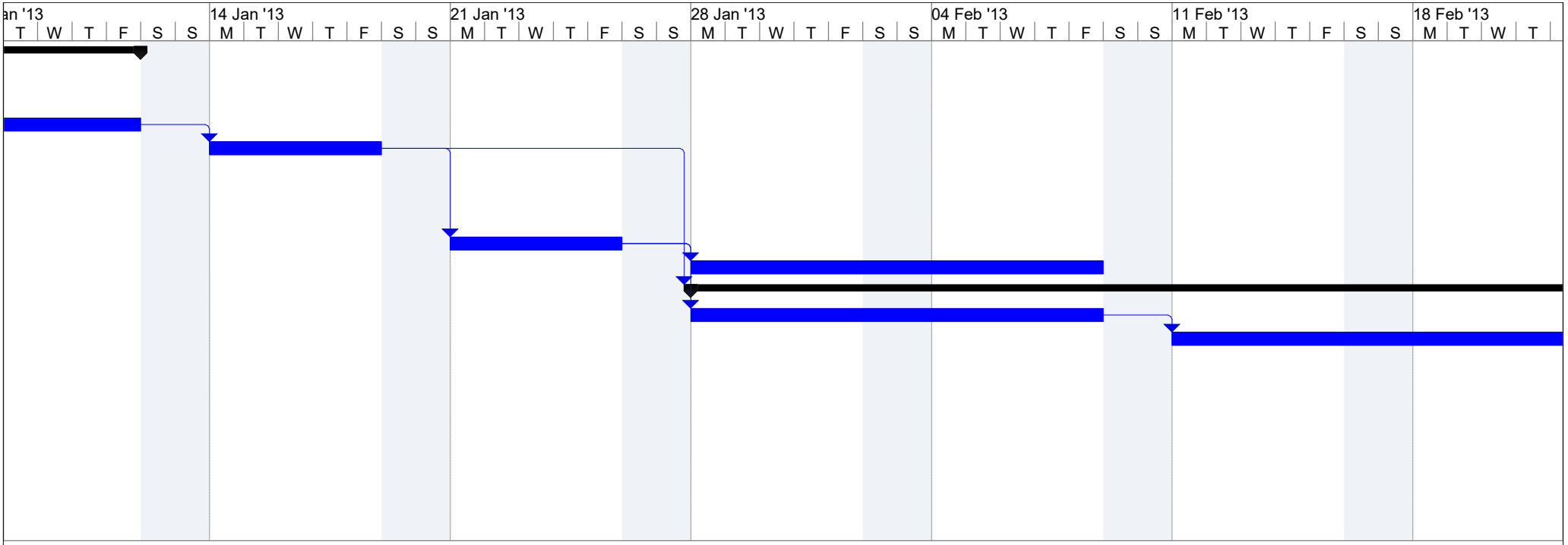
Project: SafeChat.mpp
Date: Sat 03/09/16

Task	Inactive Milestone	Finish-only	
Split	Inactive Summary	External Tasks	
Milestone	Manual Task	External Milestone	
Summary	Duration-only	Progress	
Project Summary	Manual Summary Rollup	Deadline	
External Tasks	Manual Summary		
External Milestone	Start-only		



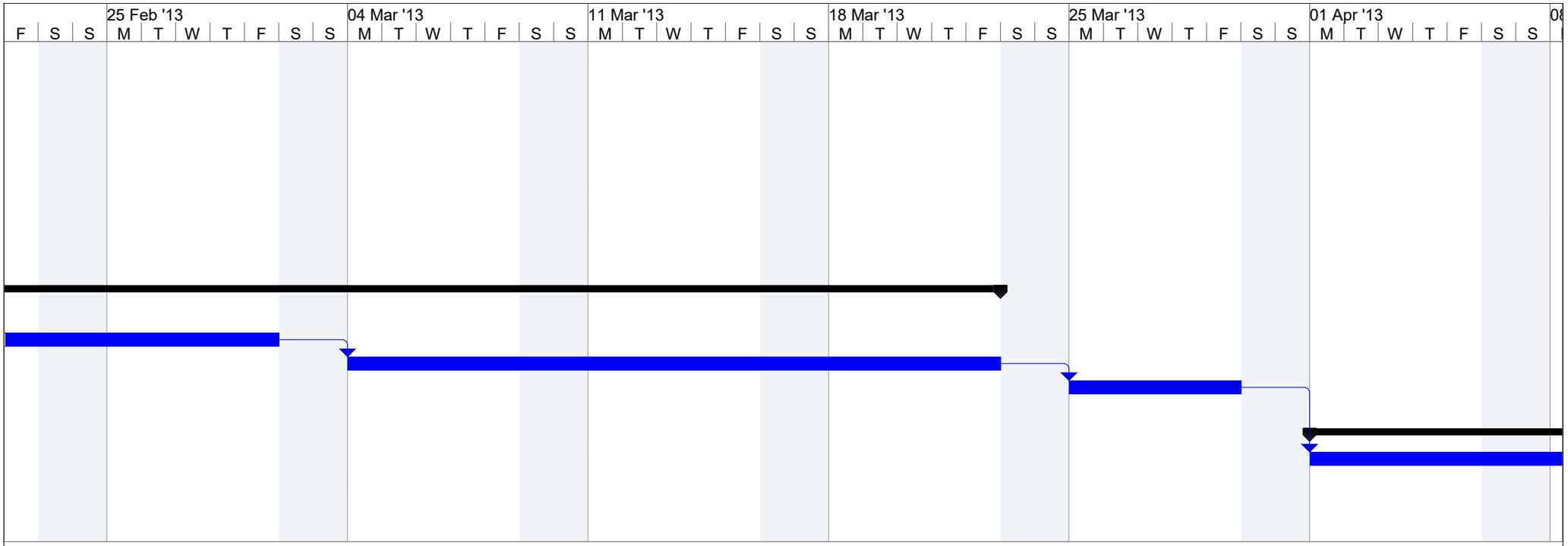
Project: SafeChat.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



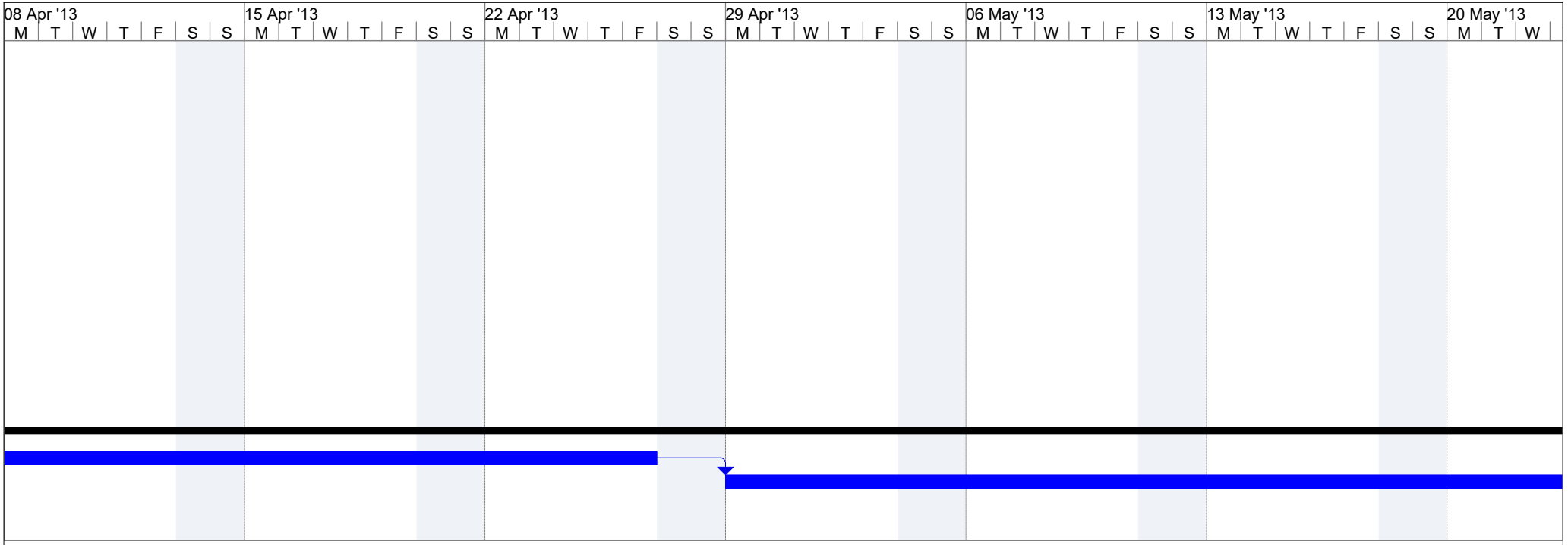
Project: SafeChat.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



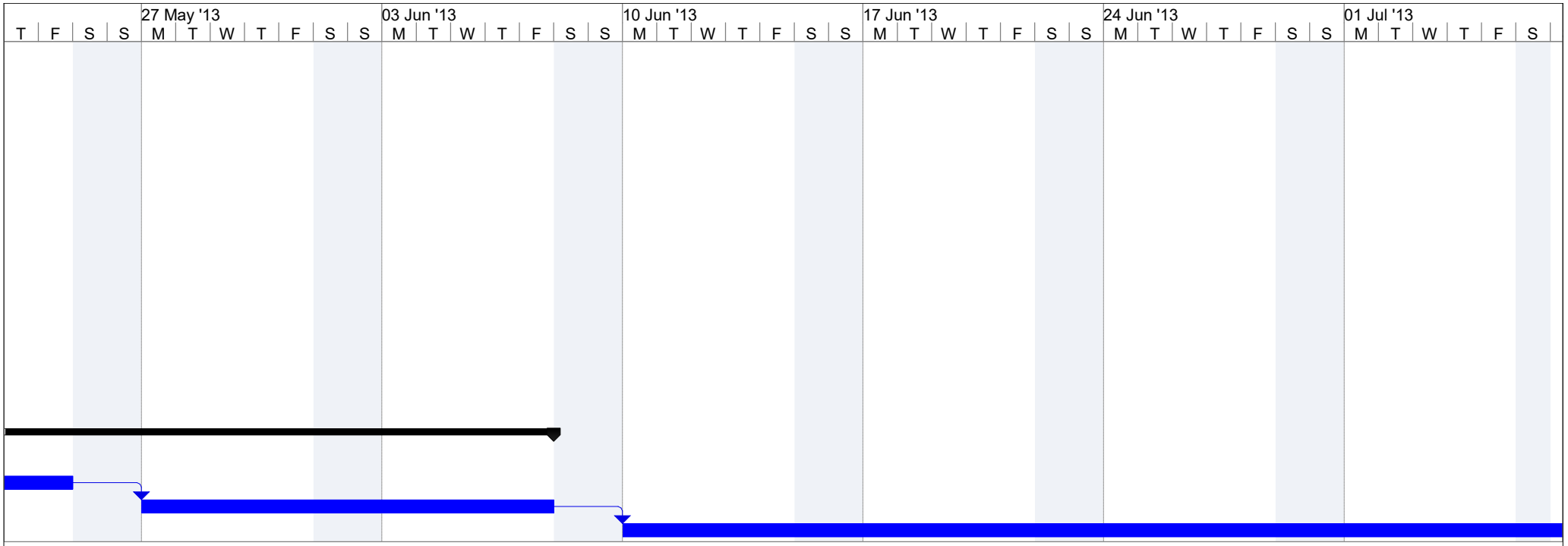
Project: SafeChat.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



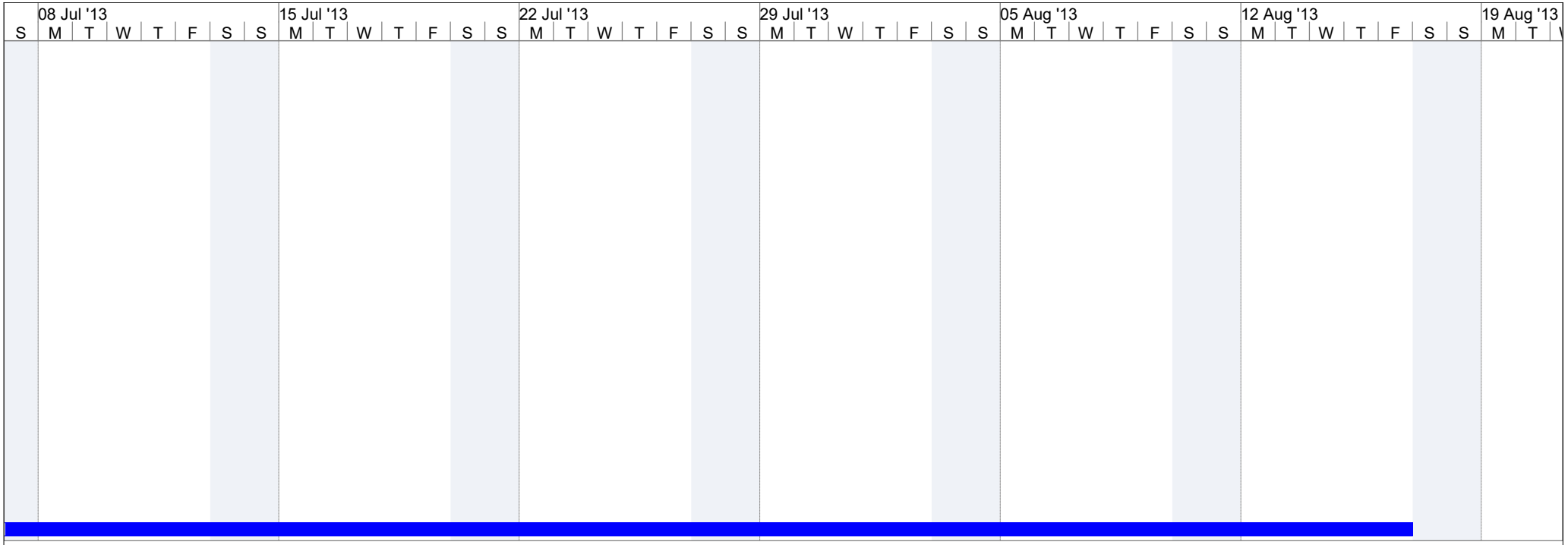
Project: SafeChat.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



Project: SafeChat.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			



Project: SafeChat.mpp
Date: Sat 03/09/16

Task		Inactive Milestone		Finish-only	
Split		Inactive Summary		External Tasks	
Milestone		Manual Task		External Milestone	
Summary		Duration-only		Progress	
Project Summary		Manual Summary Rollup		Deadline	
External Tasks		Manual Summary			
External Milestone		Start-only			

Safe Chat Development Plan

1. Produce a working ontology
    - Investigate the possibility of finding a faster way to populate it – or decide on a cut off point for a realistic working test set.
2. Generate GUI's
    - Chat windows
    - Registration/ log in
3. Threat level Agent
    - We need this to increment (into the database or text file) every time there is a threat generated by the user, then once the number gets to a specified level the system can take action.
4. Export options for ontology
    - Need to research different options in an effort to find the most efficient. Might even have to split the ontology?
5. Modify existing agents
    - Detection agent – needs to search through ontology efficiently (look at various search algorithms).
    - Register/ Log in needs to be resolved somehow.
    - Profile agent needs to read database or file.
    - Notification agent needs to read from database or file.
6. Generate new documentation for system
    - Flow diagrams.
    - Class diagrams.
    - Project plans- Gantt charts

Appendix D – Testing Resources

```
/*
 * Agent Mediated Information Exchange: Child Safety Online
 * SafeChat Development Files
 * Katrinna MacFarlane 2016
 */
package database;
import java.sql.*;
import java.sql.DriverManager;

public class DataBase {

    public static void main(String[] args) {

        String url  = "jdbc:mysql://localhost:3306/km";
        String user = "root";
       String password  = "";

          try{
              Connection myCon =
DriverManager.getConnection(url, user, password);

              Statement myStmt = myCon.createStatement();

              String sql = "insert into tbl_Users "
                                  + " (Last_Name, First_Name,
Email)"
                                  + " values ('Violeta',
'Holmes', 'v.holmes@hudd.com') ";

              myStmt.executeUpdate(sql);
              System.out.println("Insert Complete");
          }
          catch (Exception exc){
              exc.printStackTrace();
          }
    }

}
```

```
/*
 * Agent Mediated Information Exchange (2015)
 * Katrinna MacFarlane
 * PhD Working Examples
 */
package test;

import java.util.Scanner;

public class getConversation {
    public void getMessage(){
       String conversation;
       Scanner sc = new Scanner(System.in);

       Test sendConvo =new Test();

       System.out.println("Enter your text");
       conversation = sc.nextLine();

       sendConvo.printConvo(conversation);
       sc.close();
 }
}
```

```
/*
 * Agent Mediated Information Exchange (2016)
 * Katrinna MacFarlane
 * PhD Working Examples
 */
package testpackage;

public class TestPackage {

    public static void main(String[] args) {

        String msg1 = "2010";
        int msgCon1 = Integer.parseInt(msg1);

        System.out.println(msgCon1);

        int testT = 0;
        int testLoc = 0;
        int testTim = 0;
        int testInten = 0;

        testInten = (msgCon1 % 10);
        testTim = (msgCon1 % 100)/ 10;
        testLoc = (msgCon1 % 1000)/ 100;
        testT = (msgCon1 % 10000)/ 1000;

        System.out.println("Location= " +testLoc + " Time = "
+ testTim + " Intention = " + testInten);

        int intention;
        int time;
        int location;

        if (testLoc <= 0){
                    location=0;
             }
                else{
                    location=1;
                    }
        if (testTim <= 0){
                    time=0;
             }
                else{
                    time=1;
                    }
        if (testInten <= 0){
                    intention=0;
             }
                else{
                    intention=2;
                    }

       int threatLevel = 0;

        threatLevel = (location+time+intention);

        System.out.println("The current threat level for this
```

```
conversation is: " + threatLevel);
    }

}
```

```
/*
 * Agent Mediated Information Exchange (2016)
 * Katrinna MacFarlane
 * PhD Working Examples
 */
package test;

import java.util.Scanner;

public class Test {
    public static void main(String args[]){
        getConversation callClass = new getConversation();
        callClass.getMessage();
    }

    public void printConvo(String recievedConvo){
        System.out.println("You entered: "+ recievedConvo);
    }
}

     /* //now we need to parse the conversation and look for
elements
                    // then do if location == true - send
message
                    // then do if time == true - send message
                    // then do if intention == true - send
message

      String search = (new String (conversation));
      String searchString "cinema";
           if searchString = (true){
           searchString = location;
                   }
           else {location = "";
       }

 doWait(2500); //we sleep here
                        ACLMessage msg = receive();
                        //System.out.println(msg);
                            if (msg!=null){
                           String test=msg.getContent();
                           if (test.equals(location)){
                           ACLMessage msg1 = new
ACLMessage(ACLMessage.INFORM);
                          msg1.addReceiver(new AID("Threat",
AID.ISLOCALNAME));
                           msg1.setLanguage("English");
                           msg1.setContent("elevate");
                           send(msg1);
                           finished = true;
                        }
                        else{
                            if (test.equals(time)){
                            ACLMessage msg1 = new
ACLMessage(ACLMessage.INFORM);
                             msg1.addReceiver(new AID("Threat",
AID.ISLOCALNAME));
```

```
                        msg1.setLanguage("English");
                        msg1.setContent("elevate");
                        send(msg1);
                        finished =true;
*/
```

```
/*
 * Agent Mediated Information Exchange (2016)
 * Katrinna MacFarlane
 * PhD Working Examples
 */
package testone;

public class TestOne {

    public static void main(String[] args) {

         String msg1 = "3101";
        int msgCon1 = Integer.parseInt(msg1);

        System.out.println(msgCon1);

        int testT = 0;
        int testLoc = 0;
        int testTim = 0;
        int testInten = 0;



        testInten = (msgCon1 % 10);
        testTim = (msgCon1 % 100)/ 10;
        testLoc = (msgCon1 % 1000)/ 100;
        testT = (msgCon1 % 10000)/ 1000;

        System.out.println("Location= " +testLoc + " Time = "
+ testTim + " Intention = " + testInten);
    }

}
```

```
/*
 * Agent Mediated Information Exchange (2016)
 * Katrinna MacFarlane
 * PhD Working Examples
 */
package testtwo;

import java.io.*;

public class TestTwo {

    public static void main(String[] args) {

    File f = new File("meet.txt"); // Here we open a file to
record the last
                              //known threat levels for each of
the meeting elements

    String msg1 = "3110"; // For testing purposes we are using
a hard coded message
                                                  //this will be
sent from the DetectionAgent in a live system

// Here the message is converted from a string into integer
values
    int msgCon1 = Integer.parseInt(msg1);
// Here we print out the message to ensure accuracy
        System.out.println("The result of the meeting element
query is: " + msgCon1);

// The meeting element variables are declared below:
        int testT = 0;
        int testLoc = 0;
        int testTim = 0;
        int testInten = 0;

// Here we take the integer value and separate it into its
component values
// This is why we have to add the CheckBit value
//the number cannot start with a 0 so a CheckBit with  value
of 3 is being used
        testInten = (msgCon1 % 10);
        testTim = (msgCon1 % 100)/ 10;
        testLoc = (msgCon1 % 1000)/ 100;
        testT = (msgCon1 % 10000)/ 1000;

System.out.println("Checkbit:" + testT + " Location:" +
testLoc + " Time:" + testTim + " Intention:" + testInten);

// This part of the code sets the current value of the meeting
elements
        int intention;
        int time;
        int location;

        if (testLoc <= 0){
                    location=0;
```

```
            }
               else{
                   location=1;
                   }
        if (testTim <= 0){
                   time=0;
            }
               else{
                   time=1;
                   }
        if (testInten <= 0){
                   intention=0;
            }
               else{
                   intention=2; //Intention caries more
weight than the other elements
                   }
// This part of the code sets the current threat level of the
conversation
       int threatLevel;

        threatLevel = (location+time+intention);
        System.out.println("The current Threat Level for this
conversation is: " + threatLevel);

// This part of the code sets the current Safety Level of the
conversation
// This could be displayed to the user for information
        String safeLevel = "";

        if (threatLevel <= 1) {
            safeLevel = "Green";
        }
        else if (threatLevel <= 2){
               safeLevel = "Amber";
        }
        else{
            safeLevel = "Red!";
        }
       System.out.println("The current Safety Level for this
conversation is: " + safeLevel);

// This part of the code writes the last known threat levels
for each element to the meet.txt file
        int chkBit;
        int lastLoc;
        int lastTim;
        int lastInten;

        chkBit=3;
        lastLoc=location;
        lastTim=time;
        lastInten=intention;

        try{
           FileWriter fw =new FileWriter(f);
           fw.write( chkBit+ " " +lastLoc + " " + lastTim + "
```

```
" + lastInten);
            fw.close();

        }catch (Exception ex){

        }
    }
}
```

```
/*
 * Agent Mediated Information Exchange (2016)
 * Katrinna MacFarlane
 * PhD Working Examples
 */
package filewriter;

import java.io.*;

public class TestThree {

    public static void main(String[] args) {
       File f = new File("test.txt");
        try{
           FileWriter fw =new FileWriter(f);
           fw.write("Hello");
           fw.close();

        }catch (Exception ex){

        }
    }

}
```

Conversation 1

Joe: Hi there, how are you today?

Betty: I'm ok - bored though

Joe: really - what do you feel like doing

Betty: I **want to go** to **McDonalds** but no one will take me

Joe: that's too bad - hope you feel better soon

Conversation 2

Joe: Hi there, how are you today?

Betty: I'm ok - bored though

Joe: really - what do you feel like doing

Betty: I **want to go to McDonalds** but no one will take me

Joe: that's too bad - hope you feel better soon

Betty: what are you doing today?

Joe: I'm going to the **movies** with my dad

Betty: aww I **want to go** too

Joe: Got to run - see you later

Conversation 3

Joe: Hi there, how are you today?

Betty: I'm ok - bored though

Joe: really - what do you feel like doing

Betty: I want to go to **McDonalds** but no one will take me

Joe: that's too bad - hope you feel better soon

Betty: what are you doing today?

Joe: I'm going to the **movies** with my dad

Betty: aww I want to go too

Joe: I will ask my dad if you can come

Betty: no maybe we can go **next week**

Joe: cool - **lets meet** on **Saturday**

Conversation 4

Joe: I would really like you to **meet me** at the **movies** on **Saturday**, we can see the new adventure film **Tomorrow** Never Dies

Betty: ok

Conversation 5

Joe: Hi there, how are you today?

Betty: I'm ok - bored though

Joe: really - what do you feel like doing

Betty: I want to go play on my iPad-but its broke

Joe: that's too bad - hope you feel better soon

Conversation 6

Betty: hello my name is Betty

Joe: hey Betty - I'm joe

Betty: how are you Joe?

Joe: fine thank you, how are you?

Betty: good thanks, where do you live?

Joe: I live in **Manchester**

Betty: oh that's nice - what is your last name joe?

Joe: my last name is Bloggs

Conversation 7

Betty: You will never guess what happened today

Joe: oh?

Betty: **this morning** Jack came into class and told everyone he got picked for the team

Joe: wow- that's good news

Betty: Yes, his dad took him to **McDonalds** because that's his favourite and he asked if he **wants to go to** Disneyland on holiday

Conversation 8

Joe: Hi there, how are you today?

Betty: I'm ok - bored though

Joe: really - what do you feel like doing

Betty: I want to go to **McDonalds** but no one will take me, or to the **match** – or even the **movies**

Joe: that's too bad - hope you feel better soon, I went to the **movies** last week

Betty: what did you see?

Joe: I saw **Tomorrow** Never Dies, it's my favourite

Betty: aww I want to see that too

Joe: I will ask my dad if you can come **next week**

Betty: ok that would be great

"subject"	"predicate"	"object"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#in_five_minutes"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#in_five_minutes"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#hasComponent"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#ObjectProperty"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#hasComponent"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#FunctionalProperty"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#hasComponent"	"http://www.w3.org/2000/01/rdf-schema#range"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Meeting"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_match"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_match"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Macdonalds"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Macdonalds"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Restaurant"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Apollo"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Apollo"	"http://www.w3.org/2002/07/owl#equivalentClass"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Movies"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Apollo"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_thursday"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_thursday"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#come_to"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#come_to"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Intention"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Football_match"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Football_match"	"http://www.w3.org/2002/07/owl#equivalentClass"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#footie"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Football_match"	"http://www.w3.org/2002/07/owl#equivalentClass"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_game"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Football_match"	"http://www.w3.org/2002/07/owl#equivalentClass"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_match"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Football_match"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#tomorrow"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#tomorrow"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Meeting"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#burger_king"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#burger_king"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Macdonalds"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Intention"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Intention"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Meeting"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pictures"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pictures"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#kfc"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#kfc"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Restaurant"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#cafe"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#cafe"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Restaurant"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_wreck"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_wreck"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#park"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#spaghettii_house"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#spaghettii_house"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pizza_place"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#we_could_meet_at"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#we_could_meet_at"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Intention"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_friday"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_friday"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_slide"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#the_slide"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#park"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#do_you_think_we_should_m	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#do_you_think_we_should_m	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Intention"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#shopping"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#shopping"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_saturday"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_saturday"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#wh_smiths"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"

"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#wh_smiths"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#whsmiths"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#italian"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#italian"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pizza_place"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#meet_me"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#meet_me"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Intention"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#asda"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#asda"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#supermarket"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_monday"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_monday"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Meeting"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#lets_meet"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#lets_meet"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Intention"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#burger_place"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#burger_place"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Macdonalds"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#whsmiths"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#whsmiths"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#book_store"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_teusday"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#on_teusday"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Time"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#music_store"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#music_store"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#shopping"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#park"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#park"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Cinema"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Cinema"	"http://www.w3.org/2002/07/owl#equivalentClass"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Movies"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Cinema"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#flicks"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#flicks"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Location"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pizza_place"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pizza_place"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#Restaurant"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#game_store"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#game_store"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#shopping"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pizza_hut"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pizza_hut"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#pizza_place"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#tomoz"	"http://www.w3.org/1999/02/22-rdf-syntax-ns#type"	"http://www.w3.org/2002/07/owl#Class"
"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#tomoz"	"http://www.w3.org/2000/01/rdf-schema#subClassOf"	"http://www.semanticweb.org/ontologies/2010/0/Ontology1264435885586.owl#tomorrow"