University of Huddersfield

MEng Group Project

Cryptic Crossword Solver

Authors:
Mohammad Rahman
Leanne Butcher
Stuart Leader
Luke Hackett

Supervisor:
Dr. Gary Allen

Examiner:
Dr. Sotirios Batsakis

Friday, 9^{th} May 2014

Abstract

Cryptic crosswords are a unique style of crosswords in which the answer to each given clue is a word puzzle. An answer can only be obtained if the cryptic clue is read in the correct way. Often when the clue is surface read, the clue makes no sense at all. The challenge is to find a way in which the reading of the clue leads to a solution.

Cryptic crosswords are a popular type of puzzle found in many parts of the world. Most of the national UK newspapers will print cryptic crosswords of varying difficulty on a daily basis. Many users can often become frustrated when a clue appears to be unsolvable.

Clues that appear to be unsolvable are often all part of the fun of cryptic crosswords. However each clue can take some time to solve due to the fact that there are many forms of wordplay.

Contents

List of Figures

List of Tables

Chapter 1

Problem Analysis

An initial problem analysis has been conducted to ensure that the overall project remains focused upon the original problem.

This chapter will discuss key topics and will recommend that these topics are researched further to help support the project. The problem analysis will also define the problem in more detail, to help with understanding the project and its purpose.

1.1 Problem

A typical crossword involves a grid with black squares, not to be filled in by the solver, and white squares used by the solver to input their answers. The input comes from the solver working out the answer to given clues. These clues have an orientation, a length and a number associated with its related position within the grid. The fundamental difference between a typical crossword and a cryptic crossword are the clues themselves.

Cryptic crosswords are a popular type of puzzles found in many parts of the world. Most common-wealth national newspapers will print cryptic crosswords of varying difficulty on a daily basis.

Cryptic crosswords are a unique style of crosswords, in which the answer to each given clue is a word puzzle. An answer can only be obtained if the cryptic clue is read in the correct way. Often when the clue is surface read, the clue makes no sense at all. The challenge is to find a way in which the reading of the clue leads to a solution. To aid with solving cryptic crosswords, the clues are written to be within specific categories, such as reversals and anagrams, which have individual characteristics.

Many users can often become frustrated when a clue appears to be unsolvable. It is the vast range of possible clues that often makes solving not only challenging but interesting as well.

Fundamentally, the overall aim of this project is to develop a piece of software that is able to solve any given type of cryptic crossword clue.

1.2 Product

Within this group project, three components will be delivered. The first deliverable is the final, working piece of software. Whilst the second and third deliverables are written reports. The second deliverable is a group written report comprising of the all research and implementation details of the software product. The final deliverable will be each member's individual analysis and evolution of the project as a whole.

Based upon the given background and problem information it could be possible to develop a product that is able to solve the given problem.

The final product would be a piece of software that is able to understand a given clue and try to deduce what the answer to the clue is. This would require the software to have some form of natural language processing component as well as one or more cryptic crossword algorithms. Once a clue has been correctly "guessed" it can simply be returned to the user. It is the "guessing" of the answer that this project will primarily focus upon.

In order to gain maximum user coverage, the software must have an easy to use interface. The main reason for this is that the computer literacy of the intended users is not known - although basic computer literacy is assumed.

1.3 Client & Stakeholders

Dr Hugh Osborne, a lecturer from the University of Huddersfield will be the client for the group project. Dr Hugh Osborne has a keen interest in cryptic crosswords and the problem in the area which the group intends to help to eliminate. The role of the client for the group project will be to input ideas and potential requirements which Dr Hugh Osborne, as an experienced solver of cryptic crosswords, would consider to be necessary. As a client for the project, Dr Hugh Osborne will also be present for academic demonstrations.

Dr Gary Allen, Sotirios Batsakis and Colin Venters, all lecturers at the University of Huddersfield, will act as stakeholders for the group project. Dr Gary Allen will be the most involved external individual as the project supervisor. The role of project supervisor requires frequent meetings with the team to monitor the development of the project, provide guidance as well as opinion on certain aspects of the life cycle.

Sotirios Batsakis and Colin Venters, will have a less active role within the project during the project life cycle than the role of the client or the project supervisor. These stakeholders will play active roles at particular milestones of the group project such as providing guidance

for the proposal of the project and at the project demonstration approximately half way through the life cycle.

1.4 Users

Kathryn Friedlander and Philip Fine (Friedlander and Fine, 2009) carried out an investigation into whether the amount of cryptic crosswords completed by a solver determined how successful they were at solving them. To complete this study they gathered data from 241 people and have deduced the following facts about the user base (Friedlander and Fine, 2009):

- "209 M, 32 F"
- "mean age=53 years, range=23 – 83"
- "mean time spent=8 hours per week, range=1-30"

To support decisions made within the project life cycle an additional quantitative research method has been utilized to gain a larger understanding of the types of users the deliverable will attract. The research method used by the team is in the form of a survey.

The survey results gathered were seen as additional justifications for the purpose of the project. Moreover, data collected from the survey was expected to indicate the locations in which users complete cryptic crosswords to understand the potential need for the deliverable to be of a transportable nature.

The following questions were asked:

1. Do you play Cryptic Crosswords?
2. How often do you play?
3. How do you complete cryptic crosswords?
4. Do you often finish them?
5. If no to the previous question, what reason don't you finish them?
6. What is your age group?
7. When do you play Cryptic Crosswords?
8. What gender are you?
9. What is the Highest qualification you have?
10. What platform is your mobile phone on?

The survey was conducted between 18th November and 4th January. It was distributed across the University of Huddersfield portal message board, Facebook and Twitter.

1.4.1 Results

The survey that was outlined within the previous section managed to provide some interesting trends. Within this subsection a number of the trends will be highlighted, and discussed. It is hoped that the trends will be able to help guide the research, design and development processes throughout the project.

One of the key questions that was asked within the survey was "how do you complete cryptic crosswords?", which the question referred to the format in which users generally tended to play or favour.

Figure 1.1 illustrates the responses from the question. The over all statement is that 56% of those surveyed favoured a paper based format which perhaps is a little surprising in today's 'digital age'.

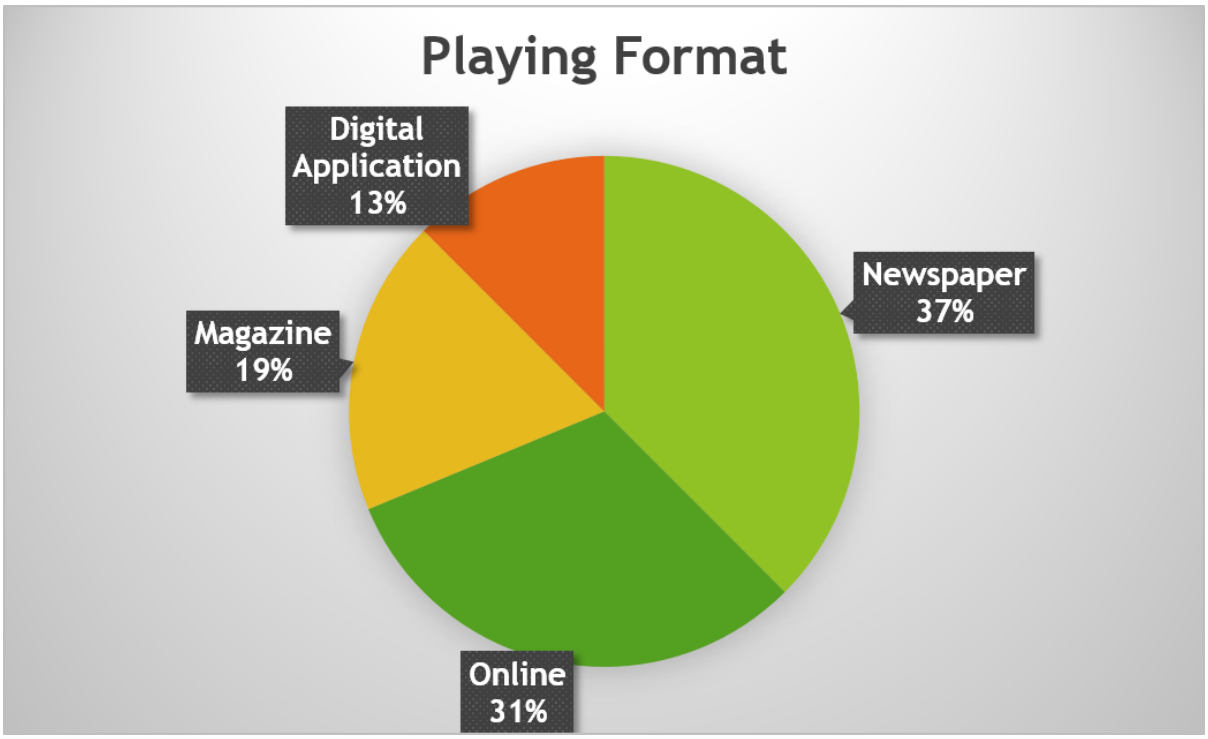

Figure 1.1: Pie chart illustrating how users complete cryptic crosswords

Following on from this it was also surprising that very few individuals use a mobile-based application to complete cryptic crosswords — especially as on-line crosswords were the second

most used format. It is this area that is intended to be a large basis for this project, and hence this will need to be investigated thoroughly.

The survey also tried to deduce the reasons behind why cryptic crossword users are perhaps sometimes unable to solve a clue. The understanding of the responses to this question is a critical part of the project, as that it is intended that the final product should be able to solve a given clue, and thus help a solver.

Figure 1.2 illustrates the responses from all users — both advanced and basic users. The chart shows that the most common reason as to why users are unable to complete crosswords is that they don't have the time to complete the crossword — something that the project may not be able to directly solve.

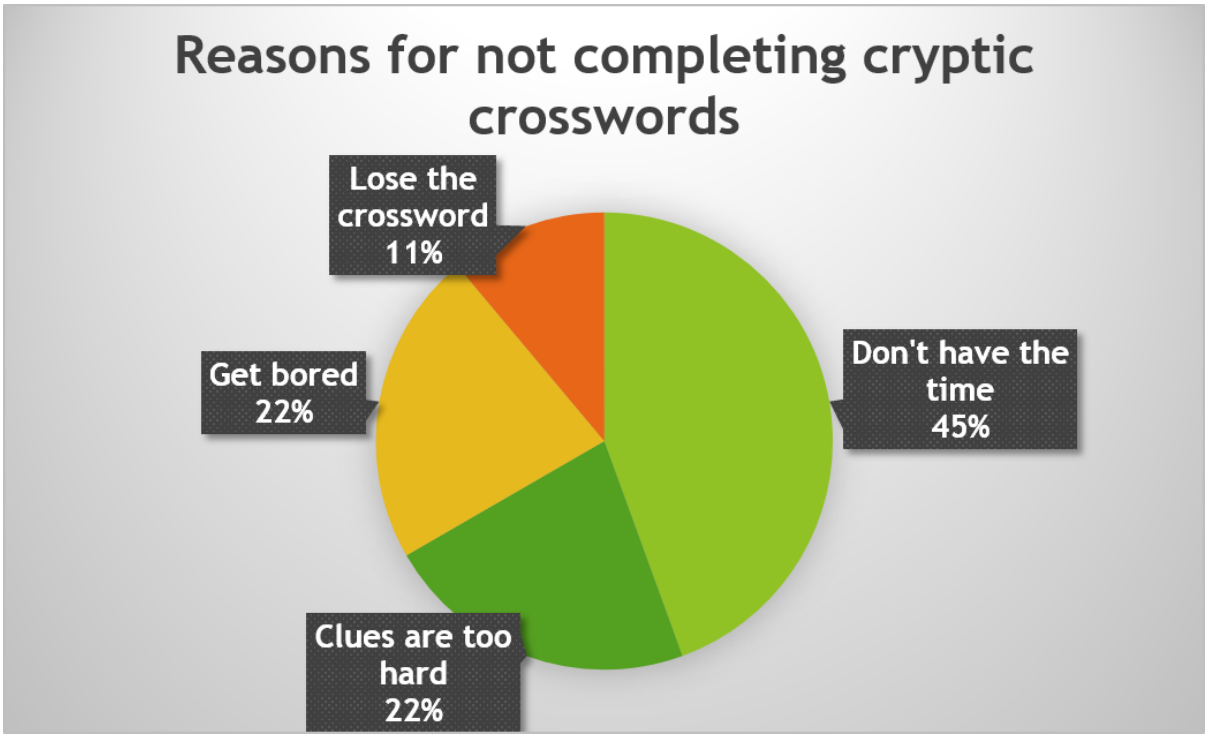

Figure 1.2: Pie chart illustrating why all users are unable to complete cryptic crosswords

However the next top two responses — "clues are too hard" and "get bored" — could be attempted to be solved within this project. These two responses could be linked together, for example do users get bored because the clues are too hard? Although the survey did not highlight this, it did illustrate a potential trend in the data that can be invested further.

Figure 1.2 illustrated responses from all users, if "advanced users" are removed then a more clear trend may emerge. For the purposes of this test an "advanced user" can be defined as a user who regularly tries to solve a cryptic crossword.

Figure 1.3 illustrates responses only from "basic users". Although the overall trend is the same, it does highlight a larger gap between the two other responses — "clues are too hard" and "get bored".

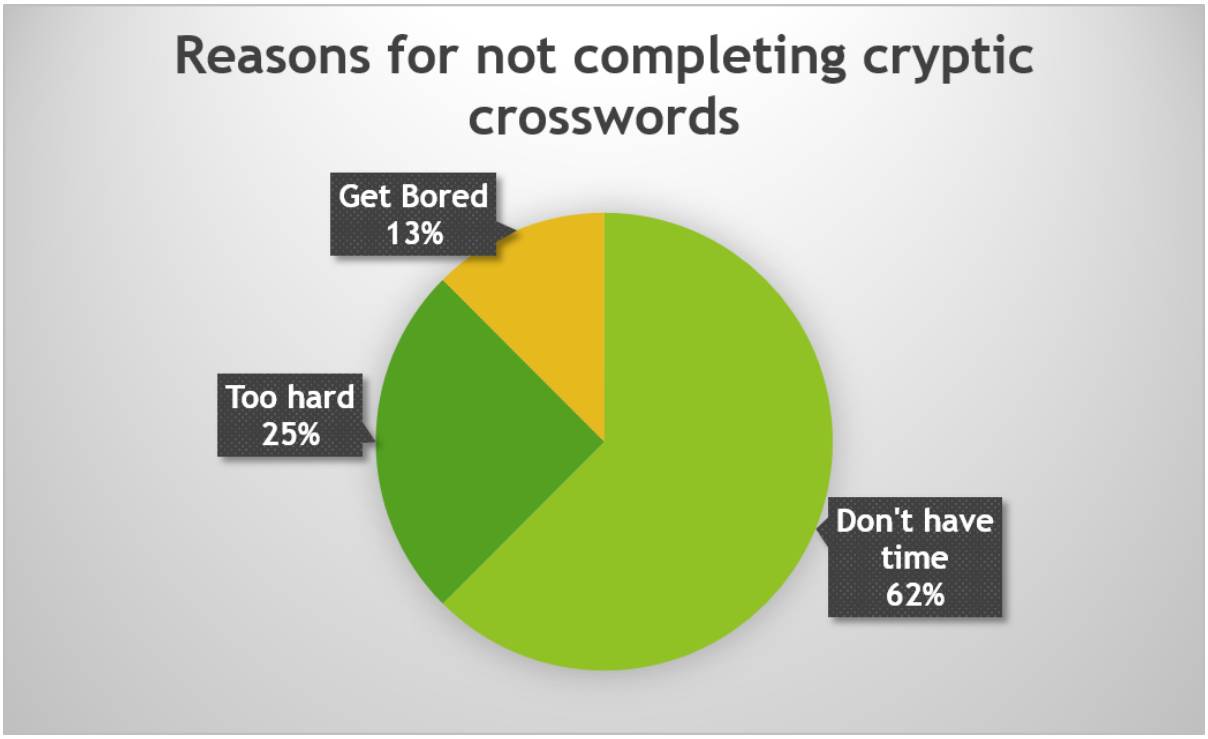

Figure 1.3: Pie chart illustrating why basic users are unable to complete cryptic crosswords

This identifies another possible trend in that users who can't solve clues may wish to 'learn' how to solve that clue. This factor will need to be taken in to consideration through the remainder of the project.

Additionally the survey tried to deduce when cryptic crosswords are completed. Figure 1.4 illustrates the responses. Generally speaking it could be said that cryptic crosswords are completed to "pass time". The top two answers indicated that users completed crosswords when "bored" or "travelling".

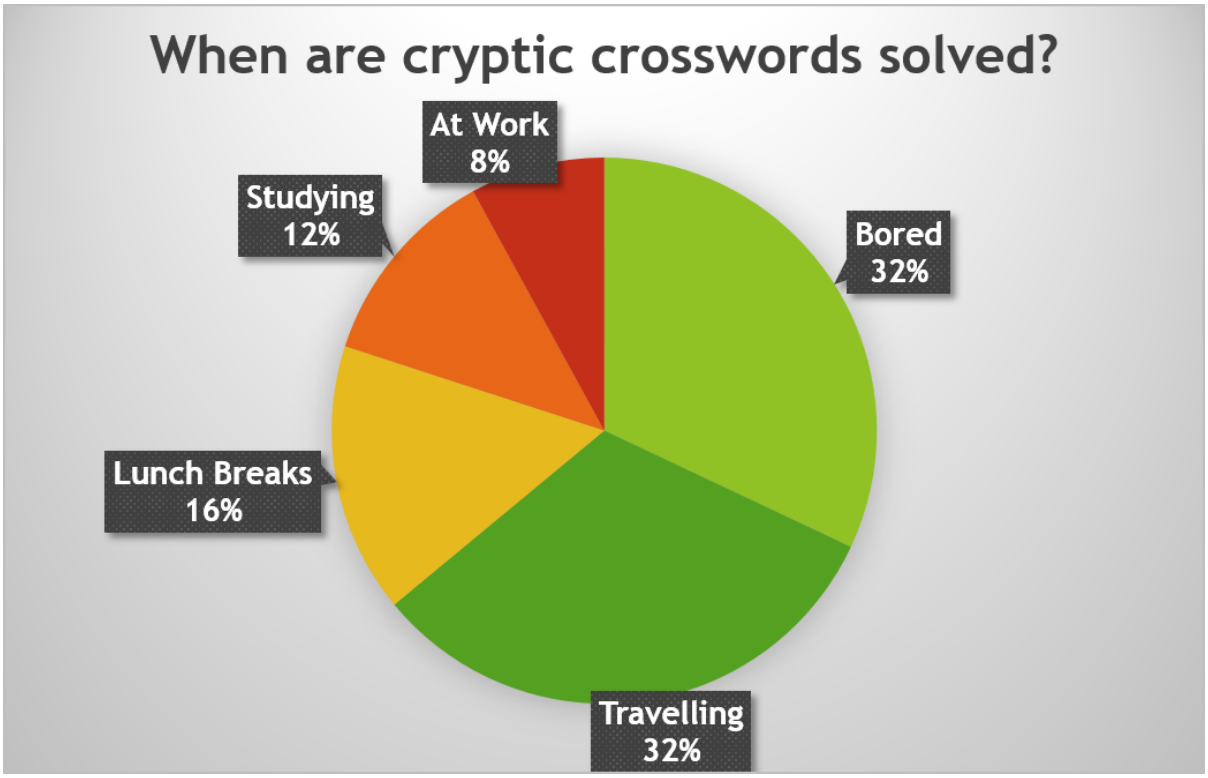

Figure 1.4: Pie chart illustrating when cryptic crosswords are solved

The responses to the question were quite important, as it highlights that the responses are people who could be categorised as people who are "on the go". This means that they are not fixed in one location, such as travelling from home to work back home again. Furthermore it highlights that there may be a potential gap in the market for a mobile based application to aid in the solving of cryptic crosswords.

With this in mind, the final question tried to deduce which mobile platforms users owned and used. Figure 1.5 illustrates the responses, and unsurprisingly the top two platform choices was Apple's iOS and Google's Android operating systems.

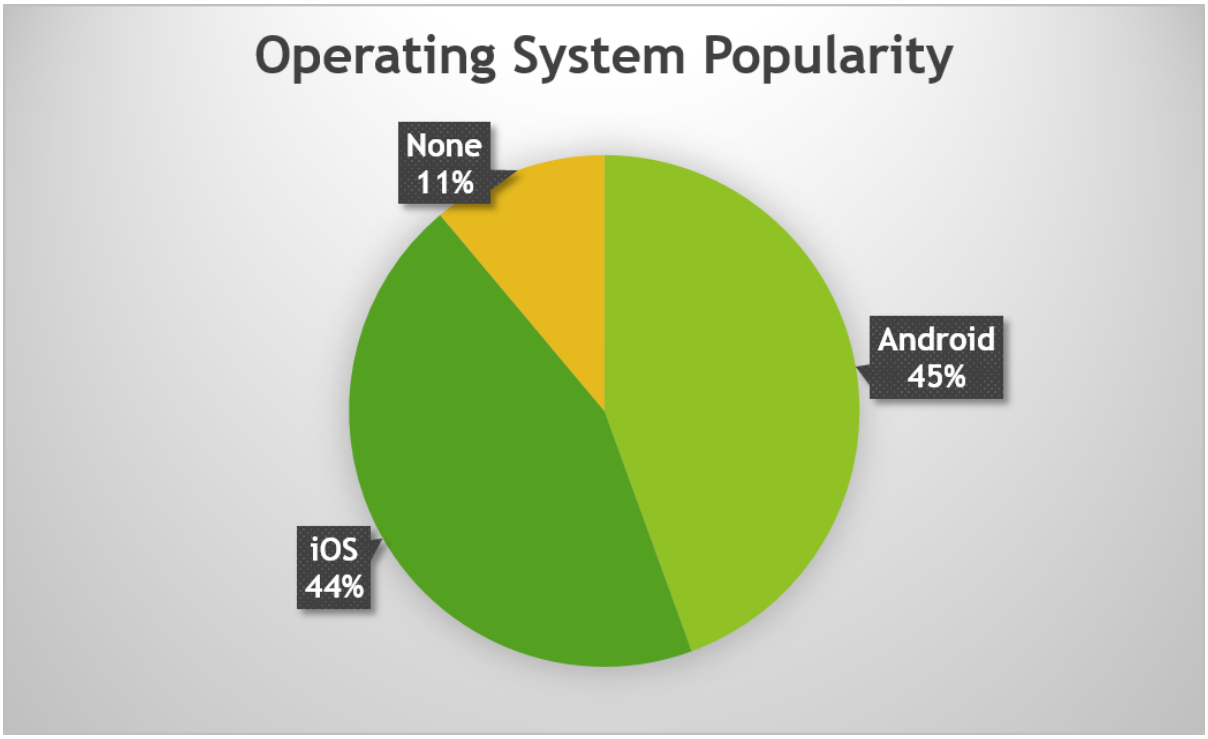

Figure 1.5: Pie chart illustrating the ownership of mobile devices by operating system

The chart indicates that if a mobile-based application was feasible then the application should run on at least iOS or Android.

The responses and results presented from the survey will be used to direct the 'formal' academic research that will form part of the written deliverable of the project. The specific research areas will be covered in the following section.

1.5 Research Areas

Before undertaking the project an initial review was conducted. The review's objective was to determine the feasibility of the project as a whole. The review also covered whether or not the project has been completed before.

From the outlined background and problem information it is clear that cryptic crosswords are a popular form of entertainment. It is also clear that some clues are particularly difficult to solve, and users may often ask other people for help in solving a given clue.

1.5.1 Cryptic Crosswords

A review of the national UK newspapers was conducted to determine whether or not there is a pattern in cryptic crosswords. Of all the newspaper's websites that were reviewed (The Guardian, The Times, The Independent and The Mirror) it was clear that all cryptic crosswords are of the same style.

Each clue is categorised as being either 'across' or 'down' with its corresponding grid number, as well as containing the number of letters the answer should be. An example is show below:

12. The seamstress's sensation? (4, 3, 7) =>PINS AND NEEDLES

The Guardian's website utilises web standard technologies such as HTML and CSS, and also provides an option to solve a clue. The Mirror's website follows a similar approach to the Guardian's website; however solutions can only be obtained by dialing a premium telephone number.

The Times and the Independent both utilise a different approach and that is to serve a Java applet. Both Java applets allow the user to solve a clue should they get stuck. The Times provides puzzles as part of their paid subscription service.

All of the above newspapers publish cryptic crosswords upon a daily basis, with the solutions to the crosswords appearing in the next day's newspaper.

Following from the crossword review, a second review into cryptic crossword solvers was undertaken. The objective of this review was to determine whether or not computerised cryptic crossword solvers exist. The three cryptic crossword solvers that were identified were One Across, Crossword Tools and Cryptic Solver.

Each of the solvers manages to solve some clues with the same answers, with other clues providing a range of possible answers.

Crossword Tools (Crossword Tools, 2013) is a paid subscription based service, which allows users to enter a clue and a pattern. A pattern can contain part of the answer or the number

of letters the answer has. If multiple answers are available, they are displayed. An example is shown below:

Kind of dog (10) =>the answer is 10 letters long.

Kind of dog (?????????r) =>the answer is 10 letters long, final letter is 'r'.

Cryptic Solver (Cryptic Solver, 2013) is a free service that offers the same functionality as Crossword Tools. Although Cryptic Solver does provide the correct answer, it does not necessarily provide the correct answer at the top of the list.

Finally One Across (One Across, 2013) provides all the same functionality as the previous two solvers, along with a score. The score is linked to the number of people who have used the given answer (effectively it's a ratings system). One Across uniquely highlights how it has managed to deduce the answer, showing the break downs of each sentence. As with Cryptic Solver, One Across is a free service that doesn't require a subscription.

1.5.2 Natural Language Processing

In order to correctly solve a clue, some form of natural language processing may be required. It is the natural language processing that could try to deduce the meaning of a clue. It is the meaning that can then be aligned with possible answers.

An example of natural language processing can be found within the One Across application. Given a clue (and a pattern) it will try to provide an accurate solution:

Spin broken shingle (7) =>ENGLISH

In order for the answer to be obtained, One Across will follow a natural language processing path and will provide it's trace path. The trace path shows how the clue has been broken down to get to the answer. The trace path for the above clue can be found below:

'spin' is the definition.

'broken' means to anagram 'shingle' to get ENGLISH.

ENGLISH matches 'spin' with confidence score 100%.

1.5.3 Application Platform

The existing products that have been discussed within this problem analysis have all been accessible via a browser. Although this is an acceptable platform, there could be a better platform that allows uses to utilise the technology easier.

As previously mentioned, most crosswords are designed for users who have a few minutes to spare on the move. With the recent trends in owning a smartphone or tablet, there may be a gap in the market for a high quality mobile cryptic crossword solver.

An in-depth review will need to be conducted in order to deduce the viability of this proposal.

Chapter 2

Research

In the previous chapter a detailed approach to what the project will cover was discussed. In order to correctly implement a working piece of a software a number of areas will need to be researched thoroughly.

Based upon the problem analysis, the the main topics that have been identified are:

- Cryptic Crosswords
- Natural Language Processing
- Application Platform
- Web Services

2.1 Crosswords

Arthur Wynne produced the first crossword puzzle which was printed on December 21st 1913. A crossword is a puzzle which involves the solver resolving the answer to a clue and placing it in the correct space within the grid.

A grid is made up of black and white squares, the black squares are blanks and the white squares are where the solver must place the answers. A crossword grid comes with a set of clues. The clues are usually arranged based upon their positing within the grid. Clues that that appear downwards in the grid are kept separate to the clues that appear across the grid.

The white squares which are used for the first letter of an answer to a clue usually have a number in the top left hand corner to indicate the clue which links to this area of the grid.

There are different types of crosswords such as quick, cryptic and double-clue. A quick crossword has clues which simply define the answer. A cryptic crossword is more complex as it has word play as well as simple definitions and many different types of clues. A double-clue crossword combines the two and allows for a simpler option for the solver when the cryptic clues become too difficult.

2.1.1 Cryptic Crosswords

Most cryptic clues consist of two different parts, the word play and the definition itself. The definition is like a clue found within a quick crossword and the word play is an indication to the answer.

Clue types such as double definition and purely cryptic break the usual format of cryptic clues. Double definitions miss out the word play whereas the purely cryptic clues miss out the usual simpler definition and become a fully cryptic definition. Other types of clue add to the usual format when smaller clues are embedded within the larger clue to assist with the word play.

Punctuation within clues should always be disregarded unless it is a question mark or an exclamation mark. Punctuation such as commas and hyphens are used to distract the solver from the answer usually by attempting to dictate how the clue is read.

A question mark tells the solver that the clue requires creative thinking to work out the answer which could be witty in nature. An exclamation mark can mean that the word play and the definition may intersect which is otherwise known as a clue of the type "& lit". Articles within clues can also be very important and should not be disregarded when reading the clue.

Cryptic clues which have a particular tense will always be for a clue with the same tense. Similarly a plural clue determines that the answer will also be plural.

2.1.2 Crossword Clue Types

An important skill needed to solve a cryptic crossword is to be able to spot the type of clue given. Below is a list of the most common types of cryptic clue and expected rules the clue should follow so they are identifiable.

Purely Cryptic

Although clues within a cryptic crossword usually include both a definition and word play, this type of clue is an exception because the whole clue is a definition written in an unusual way.

Word play within other clue types assist the user in being able to determine their answer is correct as well as solve them, because there is no word play more than one answer could be found which are incorrect. Possible indicators that denote a purely cryptic clue are:

- Question mark
- Exclamation mark

Example: *Frames for summer's activities? (5)* - (Upadhyay, 2008c)
Answer: ABACI

- "Summer" as in a person who does mathematical sums
- An abacus is a frame which holds moving beads
- As the clue is plural so must the answer be, hence abaci

Hidden

The answer for a hidden clue is concealed within the clue itself and can be spread over more than one word as well as possibly being hidden in reverse. The clue will have a definition, an indicator that the answer is hidden within the clue and a word or set of words which have the answer in them. Possible indicators that denote a hidden clue are:

- Word/s e.g. contains, in, within, held by, from
- Large words with a hidden word indicator before it may have the answer inside them
- A clue which seems inelegantly written or a clue which contains proper nouns

Example 1: *Metal concealed by environmentalist (4)* - (Upadhyay, 2008e)
Answer 1: IRON

- "Concealed by" is a phrase indicator for hidden clues
- "Environmentalist" is a large word with an indicator in front
- "Metal" is the definition so the answer is a type of metal which can be found within the word "environmentalist", hence iron

Example 2: *Mountain range in central Taiwan (5)* - (Upadhyay, 2008e)
Answer 2: ALTAI

- "in" is a word indicator for hidden clues
- "central Taiwan" contains a proper noun which indicates the answer is hidden here
- "Mountain range" is the definition
- Without knowledge of mountain ranges the answer could be narrowed down down to the following words (assuming "central" would not be within the clue without a purpose):
  - TRALT
  - RALTA
  - ALTAI
  - LTAIW
- If some of the crossword is completed within the area this clue is placed, the correct answer could be found through trial and error.

Charades

A charade clue forms its answer with the use of smaller answers to smaller clues within the main clue. Abbreviations and first/last letters of words are common within charade clues to make up the complete answer.

Two or three parts are usually within the charade clue to solve the correct answer, they may not be in the right order however, and word indicators will be used to warn the solver.

Other types of clues can also be used within charade clues for the different parts such as reversals and homophones, if this is the case there will be indicators for the specific type. Possible indicators that denote a charade clue are:

- Words e.g. with, follows, behind, after to indicate joining of answers to parts of the clue

Example: *Prior belted one that is ultimately right (7)* - (Upadhyay, 2008b)
Answer: EARLIER

- "Prior" is the definition
- "belted one" gives earl
- "that is" gives the abbreviation for i.e. or ie
- "right" gives the abbreviation for r
- All the segments put together give the word "earlier" which can also mean "prior"

Anagrams

Anagram clue types have a definition, a word or phrase to indicate the clue is of this type and an element called "fodder".

An anagram is a word whose letters can be rearranged to form another word; within an anagram clue the letters to rearrange are known as "fodder" and are placed next to the indicator. Possible indicators that denote an anagram clue are:

- Words which could mean change or shifting
- A clue which seems inelegantly written or a clue which contains proper nouns

Example: *Toy breeds trained to find out a place for pearls (6,3)* - (Upadhyay, 2008f)
Answer: OYSTER BED

- "trained" indicates the clue is an anagram
- "Toy breeds" is an abnormal phrase and is the "fodder" of the clue
- to find out a place for "pearls" is left to become the definition
- "Toy breeds" is then moved around to give oyster bed

Homophones

A homophone is a word which sounds like another word but has a separate meaning. This type of clue has a definition, a word or phrase which means the same as the homophone to find and an indicator. Possible indicators that denote a homophone clue are:

- Words which indicate hearing or sound e.g. said, heard
- Normally the indicator for a homophone is next to the word or phrase which is to be used to find a homophone

Example: *Refer to a location, reportedly (4)* - (Upadhyay, 2008j)
Answer: CITE

- "reportedly" is the homophone indicator
- "a location" is the phrase which needs to be used to find a homophone
- "Refer to" is the definition
- A location can be otherwise known as a "site", hence cite

Acrostics

An acrostic clue commonly involves picking the first letter from a group of words and putting them together to form the answer. It is possible that the clue will require the last or middle letters from words to solve them or that the letters should be put together in reverse order.

This clue type has a definition and an indicator as well as "fodder" which in this case means the group of words the necessary letters will come from. Possible indicators that denote an acrostic cryptic clue are:

- Words which could mean start or beginning
- If the clue is unusually long and so is the number indicator to determine how long the answer should be

Example: *Some URLs recommended for beginners to explore online (4)* - (Upadhyay, 2008a)
Answer: SURF

- "beginners" is the indicator
- As the clue states the answer should be of length four it is assumed "to explore online" is the definition as there are only three possible letters for an acrostic in the phrase
- "Some URLs recommended for" is the "fodder" for the clue. The indicator implies that the first letters should be taken from the first letters of the words within the "fodder", hence surf

Palindromes

A palindrome is a word which reads and looks the same when it is reversed. This type of clue has an indicator and a definition. Possible indicators that denote a palindrome clue are:

- Phrases which may mean either way or going around in circles

Example: *Unacceptable, going up or down (3,2)* - (Connor, 2012a)
Answer: NOT ON

- "going up or down" is an indicator as it could mean "in either direction" like the format of a palindrome
- "Unacceptable" is then left as the definition which gives the answer, not on

Reversals

A reversal requires the solver to reverse a number of letters to give a new word. The clue consists of a definition, an indicator that the clue is a reversal clue and some "fodder" which is a phrase or word which could contain the letters to be reversed or a smaller clue which leads to the letters which need to be reversed. Possible indicators that denote a reversal clue are:

- Words/phrases used for directional purposes e.g. left, up
- The word indicators may also be relative to the direction the clue should be placed within the crossword (down or across)

Example: *Stop the flow in crazy get-up (3)* - (Upadhyay, 2008h)
Answer: DAM

- "Stop the flow" is the definition
- "get-up" is the indicator that the clue is a reversal
- Another word for "crazy" is mad which reversed gives the answer dam

"& lit"

"& lit" clues, which means "and literally so", is a type of clue where the definition and the word play are the same and are not split out into separate phrases or words as with other clues. The definition is the whole clue and the word play can be one or more of any of the normal clue types such as anagrams and charades. Possible indicators that denote a "& lit" clue are:

- Exclamation mark

Example: *Cop in male form (9)* - (Upadhyay, 2008g)
Answer: POLICEMAN

- The whole clue is the definition
- "form" is an indicator for an anagram clue
- "Cop in male" can be rearranged to policeman which is also the answer to a "Cop in male form"

Double Definition

A double definition clue has no word play and is purely a clue with two (or possibly more) definitions which lead to the same answer. Possible indicators that denote a double definition clue are:

- Possibly shorter than most clues (2 or 3 words)
- Although it is advisable to ignore punctuation when solving cryptic clues, a double definition may have a piece of punctuation separating the definitions.

Example: *Robust author (5)* - (Upadhyay, 2008d)
Answer: HARDY

- Both words are separate definitions which could both mean hardy

Containers

A container clue includes three definitions and an indicator. One of the definitions is for the final solution whereas the other definitions describe two separate answers where one is contained within the other.

They are similar to charade clues in the way that other types of clues can be used within container clues such as anagrams and charades themselves. Possible indicators that denote a container clue are:

- Word indicators which could be used to indicate the inner word (e.g. inside, held) or the outer word (e.g. outside, external)

Example: *Building for the workers in principle (8)* - (Upadhyay, 2009a)
Answer: TENEMENT

- "principle" is the definition for the outer word which gives tenet

- "in" is the indicator that the answer for the inner word will be placed within the outer word (tenet)
- "the workers" is the definition for the inner word which is men
- 'Building' is the overall definition which could mean tenement which is also given when men is put within tenet

Deletions

Deletion clues require the solver to retrieve the answer by looking at the definition and the word play and removing the correct letters from the correct word. The clue will not usually have the word which needs letters removing from it directly within the clue. Possible indicators that denote a deletion clue are:

- Words to indicate letters should be removed from a certain place within the word such as its first or last letter
- Words to indicate certain letters should be removed from the word found. These could be words that can be abbreviated .

Example: *Little shark edges away from diver's equipment (3)* - (Upadhyay, 2009b)
Answer: CUB

- "edges away from" indicates that the first and the last letter should be removed from the answer which comes from "divers equipment"
- "divers equipment" is otherwise known as scuba
- Removing the "s" and the "a" from scuba leaves the word cub which is also the answer given from the definition "Little shark"

Spoonerisms

A spoonerism clue has a definition and word play which is usually a phrase which describes another. This phrase, which is usually two words long, is taken and the first letter of each is swapped around to gain a new phrase which is in turn the answer to the clue. Possible indicators that denote a spoonerism clue are:

- The word "Spooner" or "Spooners" is placed within the clue

Example: *Spooner's cheerful enthusiast? He'll get you across (8)* - (Connor, 2012b)
Answer: FERRYMAN

- "Spooners" indicates that this clue is a spoonerism
- "Hell get you across" is used as the definition
- "cheerful enthusiast" could also be known as a merry fan
- Swapping the first letters from the words merry and fan give the answer ferryman

Pattern

This type of clue has a definition, an indicator that the clue is of the type pattern and a phrase or word which has the answer within them arranged as a pattern. These patterns can be odd or even letters joined together to make a word or possibly letters picked from regular intervals.

Pattern word play can also be joined together with other types of clues such as charades to form a more complex answer. Possible indicators that denote a pattern clue are:

- Words which could mean even, odd or routine

Example: *Beasts, free, ginned, we hear regular losses there! (8)* - (Upadhyay, 2009d)
Answer: REINDEER

- "regular" indicates it is a pattern clue as well as "losses" to indicate the dropping of letters.
- "Beasts" is the definition.
- "free, ginned, we hear" holds the answer reindeer by picking out the first "r" within free and each letter alternately from then on.

Substitutions

A substitution clue involves removing letters from a word and replacing it with another to retrieve the answer. There are two definitions within the clue, one definition to retrieve the word to substitute letters from and another to define the final answer.

The letter or letters to substitute are usually an abbreviation which can be found within the clue itself. Possible indicators that denote a substitution clue are:

- Words which mean substitution e.g. replace, switch, exchange

Example: *Unexciting story gets mark for length (4)* - (Upadhyay, 2008i)
Answer: TAME

- "mark for length" is an indicator for a substitution clue
- "mark" can be abbreviated to "m" and "length" can be abbreviated to "l", therefore replace "l" with "m"
- "story" is the definition for the word which needs the substitution and could be defined as a tale
- Replacing the "l" in tale with "m" gives the word tame which can also mean unexciting

Shifting

A shifting clue has an indicator, a definition of the final answer and another definition for the word which needs to be used to shift a letter to a different position within the word to find the final answer.

The shifting of a letter could be moving the first letter to the last position in the word or in a more complex clue letters could be shifted within the middle of the word. Possible indicators that denote a shifting clue are:

- Words e.g. shift, change, move
- Phrases e.g. head to foot

Example: *Character needs help, head to foot (4)* - (Upadhyay, 2009e)
Answer: BETA

- "head to foot" indicates moving a letter from the front of a word to the end
- "help" is the definition for the word which requires letter shifting and can also be defined as abet
- Moving the first letter of "abet" to the end gives the word beta which is a "Character"

Exchange

An exchange clue is similar to a shifting clue however instead of only one letter shifting positions within a word, two letters within a word exchange places to form a new word.

Typically the letters to exchange will be the first and last letters of a word or two letters next to each other, however it is possible more than one letter on each side will need to be swapped.

For example, the word "rage" can be split into two sections "ra" and "ge" which can then be exchanged to make the word "gear". Possible indicators that denote an exchange clue are:

- Words e.g. swap, exchange, change

Example: *Doomed king switching sides? True (4)* - (Upadhyay, 2009c)
Answer: REAL

- "switching sides" indicates that this clue is an exchange clue
- A "Doomed king" can also be known as a "Lear"
- "True" is the definition which can also be defined as "real" which can be gained by exchanging the first letter of "Lear" ("L") with the last letter ("r")

2.2 Natural Language Processing

An inherent disparity between the languages used and understood by human beings and those which are interpreted by machines introduces the requirement for a system which is able to provide a form of compatibility between the two. Natural language can be described as a system of communication that has not consciously been invented (Collins, 2004) and typical examples of this are the languages used by human beings to communicate with each other. English and Korean are two such examples of natural languages and these evolve over time as words are added and removed and the rules that define the language are refined or adapted, meaning there is no immutable grammar for the language.

Conversely, formal (or artificial) languages are those which have been fabricated for a specific purpose, and can be described using precise and unambiguous mathematical rules (Jiang et al., 2010). An example of such a mathematical rule is the BackusNaur Form (BNF) grammar. Programming languages such as Java or Python are examples of artificial languages, as these have been created with a specific purpose in mind. The vocabularies of these languages are well defined and free from ambiguities, making interpretation by the languages corresponding compiler a black or white task where the given language to compile is either valid or invalid.

Natural language processing (NLP) allows for the manipulation of natural languages by a computational device (Bird, 2009) and there are a number of identified steps of the NLP process which have been developed, from scanning and parsing, to meaning extraction and integration. In the context of solving cryptic crosswords, a given clue would need to be broken down into meaningful components which would allow, as an example, the word play and the definition components to be identified and passed forward to the appropriate algorithms to identify the correct answer.

```
Reversible fasteners pinning baker, European
(TOP (NP (NP (NNP Reversible) (NNS fasteners)) (VP (VBG pinning) (NP (NN baker,)
)) (. European)))
```

Figure 2.1: Creating a parse tree for a given sentence

(Halpern, 2013)

2.2.1 The Steps of Natural Language Processing

Scanning

A number of steps are typically involved in the processing of a natural language. Assuming the input format of the natural language is in a textual form as sentences (rather than as speech or handwriting); these chains of text may be broken down into their separate components - the words which make up the sentences. This process is known as scanning, and may itself comprise of several sub-steps. The first step could likely involve breaking down the given textual input into separate sentences (Apache, 2013).

```
C:\Users\u0955187\Downloads\apache-opennlp-1.5.3-bin\apache-opennlp-1.5.3\bin>op
ennlp SentenceDetector en-sent.bin < input.txt
Loading Sentence Detector model ... done (0.068s)
Reversible fasteners pinning baker, European
```

Figure 2.2: A single sentence is already in the desired state

Once the separation of any sentences present in the input has occurred, these can be further divided into the separate words to achieve an array-like data structure, such as [*Reversible*, *fasteners*, *pinning*, *baker*, *European*], a process which is referred to as tokenising. It is important to consider the inclusion and placement of any punctuation provided in the input text, as this may provide some indication as to the separation of the cryptic crossword components and increase the chances of successfully calculating the answer. In the example provided, European corresponds to the definition component of the clue, where Reversible fasteners pinning baker relates to the word play component. Both components have been clearly separated by a comma, though not all cryptic clues are presented in this fashion.

Parsing

Parsing is a process used to ensure that a given sentence conforms to a grammar and thus showing that it is syntactically correct (McCluskey, 1999), and this process can be demonstrated by creating a parse tree to represent the structure of a given sentence. It may become apparent when attempting to solve the clues of a cryptic crossword that certain elements of the provided clue sentences bear some relation to the correct answer. To give an example, it may prove more likely that the adjectives or verbs present in the input are synonyms of the correct answer, where there may be little probability that the answer stems from a synonym of any present determiners (the, this, a, some) or prepositions (on, beneath, over, under). Likewise, adjectives or adverbs that hint at a particular action such as *reversible* or *backwards* could reveal themselves as more likely to be indicators that describe in what way the clue should be manipulated.

Referencing

The ambiguous nature of cryptic clues may not allow for a straightforward parse of an already ambiguous (natural) language, possibly contributing to the complexity of the process of solving. It may be the case that multiple parses of the same clue have to be taken forward to be processed before the solution can be found. Once a parse tree has been selected for the given input, a number of steps can be performed, which attempt to reduce the level of ambiguity. Lee McCluskey outlines four methods which can be applied to achieve this:

- **Contextual Disambiguation:** Such as referring to segments of a previous piece of text. Fred Bloggs may have been declared in a previous sentence, and from that point referred to as he.
- **Physical Constraints:** This refers to the literal meaning of a sentence of phrase, and uses known facts to assist in the reduction of ambiguity. The phrase I answered the door in my pyjamas could be taken to mean that the subject was either wearing their pyjamas when they answered the door, or that the door was wearing the subjects pyjamas when they answered it. As it is unlikely that the door was physically wearing pyjamas, the parse suggesting this could be disregarded.
- **Default Roles in Known Verb Structures:** Some verbs may only be applied to a subset of subjects in order to make reasonable sense. In the example sentence fruit flies like a banana, one parse may take fruit as the noun, and flies as the verb. In this scenario, fruit is physically incapable of flying, and so an alternative parse which taken fruit flies as the noun-phrase is more likely to be the correct parse of the sentence.
- **General Defaults:** To say that one is going to visit an old friend may imply that the friend has been known for a long time, or that the friend is of old age. While the correct interpretation may be context specific, the likelihood is that one particular interpretation will hold precedence over another. In this case, the friend is likely to have been known for a long time.

(McCluskey, 1999)

A parsing algorithm is used to obtain a complete parse tree for a given sentence, providing insight into the complete structure of the sentences. It is possible that this may not be required for the task at hand; a full parse may provide more information than what is required to solve a cryptic clue. An alternative mechanism exists, chunking, which may also be referred to as *light parsing*. This technique divides a sentence into a series of non-overlapping chunks of text, which provide an overview of the input sentence (Litman, 2003). One of the major benefits of this technique is the removal of the need to resolve ambiguity, as chunks of the sentence are non-recursive, unlike parsing which produces components of a sentence which may be nested in each other.

```
Reversible_NNP fasteners_NNS pinning_VBG baker,_NN European._.
```

Figure 2.3: Assigning *Parts of Speech* tokens to an input sentence

Meaning Extraction

Meaning extraction refers to the process of taking the parse and storing it in a way which accurately represents the information it models (McCluskey, 1999). For example, the following cryptic crossword clues all map to the same answer, and whilst they are written differently, they each imply the same meaning of *DRILL*:

- *Doctor gives patients exercise*
- *Doctor needing a doctor for practice*
- *GP not fit for practice*

(Gordius, 2003)

Effective meaning extraction will ensure that each distinct clue may help to ensure that variations of the same will be mapped to a single, internal representation and this will allow for the answer to be retrieved from a database without having to recalculate what has already been processed once.

2.2.2 Natural Language Processing Libraries

There exists a range of NLP libraries that offer many natural language processing functions for a variety of programming languages. Development on a number of these libraries has become stale, and others are available only for use after purchasing a commercial licence. Below are a selection of available libraries which have recently been updated and are available under some form of free software licence.

Library	**Licence**	**Last Updated**	**Supported Languages**
OpenNLP	Apache 2.0	April 2013	Java
NLTK	Apache 2.0	November 2012	Python
Stanford CoreNLP	GNU General Public Licence	June 2013	Java

Table 2.1: A comparison of available NLP libraries

OpenNLP

OpenNLP provides the ability to carry out a range of natural language processing functions, including parsing, chunking and name finding, where the latter aims to recognise proper nouns in specified text such as person names. The features of the library utilise a maximum entropy model, and this allows the performance of the software to be enhanced as it uses training data to learn (Apache, 2013). This allows for each component of the toolkit to refine as the subset of training data increases over time, with the aim of increasing the accuracy of the corresponding NLP component. The library is available for use in a Java environment or can be accessed directly through a command line interface, but the general usage of the library requires providing an applicable model and language data as input, for which the model will be utilised for the language processing.

Models A number of models are provided for use with the NLP library. These include, but are not limited to, models to assist in the process of parsing, chunking and detecting sentences. A number of libraries also exist which allow for the identification of person names, company names, times, dates or location in input text. These existing models may be further trained in a bid to increase their effectiveness.

DocumentCategoriser Another prominent feature of the library is the ability to classify input text into a range of predetermined categories using the aforementioned maximum entropy model. Once a model has been created, which contains a series of example inputs along with their corresponding categories, further inputs to the system will be paired with a *best outcome* classification.

```
<terminated> NLP [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 1, 2013 3:18:08 PM)
Category for: Posy says no, according to Spooner is: SPOONERISM (0.18832631809541703)
Category for: Preparation for documents once Spooner's transporting luggage is: SPOONERISM (0.005477603261500726)
Category for: Preserve holy one appointed by Spooner is: SPOONERISM (0.003470437596580894)
Category for: Theme''s proper order reversed is: REVERSAL (0.016981012268904402)
Category for: This compiler going to the stake? Quite the reverse! is: REVERSAL (3.198100077871112E-4)
Category for: This vehicle reverses into its club is: REVERSAL (0.01006333130189762)
```

Figure 2.4: A demonstration of the OpenNLP DocumentCategorizer

NLTK

The Natural Language Toolkit is an alternative library which exists with an open-source licence variant. NLTK exists for use with the Python programming language and possesses a similar infrastructure to OpenNLP, where trained models are used to allow each component of the software package to function correctly (NLTK Project, 2012). The package is

supplemented by a comprehensive e-book, detailing and providing examples of the usage of each component available.

2.3 Mobile Platforms and the Market

The demand for applications to be portable on mobile phones and tablets has become a fashion as well as accessibility on the go. In the current market Cryptic crosswords are available in a variety of newspapers, magazines and online websites. These are everyday media in which a person can access their favourite Cryptic Crosswords and participate in. With the expansion of mobile phone and tablet applications research has shown that a unique Cryptic Crossword solver has not been incorporated in any of the mobile operating systems. There are applications, which can be downloaded, that are pre-solved crosswords but there is no real solver, which solves Cryptic Crosswords in real time. In fact research has shown that there is only a very small handful of Cryptic Crossword applications across all mobile operating systems.

2.3.1 Mobile Operating Systems

The Oxford English dictionary states an "Operating System" as 'the low-level software that supports a computers basic functions, such as scheduling tasks and controlling peripherals' (Dictionaries, 2011).

A mobile operating system has the same definition but supports the basic functions for handheld devices such as mobile phones and tablets. In the current market the term mobile operating system is associated with smartphones rather than mobile phones due to the powerful processors, which are embedded in the mobile phone devices.

The official first mobile operating system for smartphones was by IBM introduced in 2000 as the 'Simon' but it was Ericson who created the first all in one smartphone with the Symbian OS, which incorporated a keyboard hence, the term smartphone was introduced. This ran on various mobile phones by companies such as Ericcson, Samsung but predominantly on smartphones by Nokia. This created a path for a market, which was to be dominated by others in the coming future (Akman, 2013).

By 2007 from the back of a successful campaign selling music devices known as the iPod, Apple introduced the Apple IPhone which came with their fist mobile operating system the IOS which was a full operating system used from the Mac OS X 10 (Apple, 2007).

By July 2008 Apple released IOS 2.0 and with this came the app store. This was a revolution to the mobile market allowing a platform for third party developers to sell and market their own applications for the mobile operating system. On the 7th January 2013 Apple announced that they have had more than 40 Billion downloads of apps through their app store (Apple, 2013).

In September 2008 Google released their own version of a mobile operating system Android which had its similarities of marketing with its very own app store known as the android

market which since has been rebranded to the Google play store.

In April 2009 Blackberry also launched its own application store called the Blackberry World, which works with their mobile operating system the Blackberry 10 (Cha, 2009).

Finally windows released their version of an app store for distribution of applications on October 26th 2012 (Businessline, 2012).

Although there are now several platforms that run on a mobile device, Android and iOS combine for 91.1% of the Worldwide Smartphone OS Market (IDC, 2013).

This shows that although blackberry has been in the market since 2009 there isnt much of a rise to interests in their apps and while Windows is fairly new it has a lot of ground to cover to catch up with their main competitors.

For the purpose of this project it is pretty clear to what is demanded from consumers in the real world with Apple and Google being the two main competitors for mobile applications. In order to decide on what platform the project will be suitable for to design, maintain and deploy a good working product below is the following page contains a table which covers some of the reasons which could be possible to allow the team members to come to a decision to what pathway the project will be going in.

Platform	**Programming Language**	**Open Source**	**Open API/SDK**	**License**
Android	Java	Yes	Yes	Apache
IOS	Objective C	N/A	Yes	Proprietary
Blackberry	Java, C++	N/A	Yes	Proprietary
Windows	C, C++, C#	N/A	Yes	Microsoft

Table 2.2: A comparison of mobile platforms

Platform	**Latest Version**	**Debugging**	**Hardware / Software Requirements**	**Emulator**
Android	4.3 Jelly Bean	Yes	Windows / Mac	Yes
IOS	IOS 7	Yes	Intel Based Mac	Yes
Blackberry	Blackberry 10	Yes	Windows / Mac	Yes
Windows	Windows Phone 8	Yes	Windows	Yes

Table 2.3: A further comparison of mobile platforms

Platform	**Underlying OS**	**Development Environment**	**Submission To App Store**	**Development Cost**
Android	Linux	Eclipse	Unlimited	$25 One Off cost
IOS	Darwin	XCode	Unlimited	£60 Yearly / $99
Blackberry	Blackberry OS	Eclipse	100 Per Year	$100 One off cost
Windows	Windows	Visual Studio	Unlimited	$19 Yearly / Free for Dreamspark Students

Table 2.4: More comparisons of mobile platforms

Although these are the four main types of platforms available in the market for mobile development there is a growing amount of organisations, which are developing tools to allow developers to create cross platform application with ease. Two of these are Appcelerator and Adobe AIR.

Platform	**Programming Language**	**Open Source**	**Open API/SDK**	**Underlying OS and License**	**Development Environment**
Appcelerator	JavaScript	Yes	Yes	Linux Apache 2.0	Eclipse Based IDE / Titanium Studio
Adobe AIR	ActionScript, HTML, CSS, JavaScript	No	Yes	Darwin Proprietary	Adobe AIR

Table 2.5: A comparison of cross platform development tools

2.3.2 Appcelerator

Appcelerator is a platform created by Appcelerator Inc to allow developers to create cross platform native applications for Android and IOS. They later introduced compatibility for Blackberry 10 and are in the process of developing for Windows Phone. The main development environment used to create applications is an Eclipse based IDE known as Titanium Studio. Developers can create great looking native apps using JavaScript. The use and license of Appcelerator falls under Apache 2.0.

2.3.3 Adobe AIR

Adobe Integrated Runtime (Adobe AIR) uses adobe tools such as flash to allow developers to create platform independent web apps. Unlike Appcelerator this means that applications created can only be web based and not native. For this reason a lot of developers avoid using Adobe AIR. It supports all the major vendors for mobile applications but apps created in Adobe AIR are a lot slower.

2.3.4 Native Apps

A native app is a platform independent application designed to work with a particular mobile OS. The app is installed on the device whether its a smartphone or a tablet.

Advantages

- Faster at accessing device features such as camera and accelerometer
- Easy to find in app stores such as Google Play and Apple app store
- Secure as they go through an approval phase from vendors

Disadvantages

- Expensive to develop
- Expensive to maintain
- Approval process can take from days to months
- Support of the app can be hard to maintain due to different people using different versions of operating system installed.

2.3.5 Web Apps

A web application is really a website designed to look and feel like an app. This is wrapped by the web browser, which means an Internet connection is required to use the app. Google Chrome and Safari are examples of Web apps.

Advantages

- Easy to maintain
- Can be compatible with any device
- Approval is not required
- Easy to maintain and update without affecting the user to update the software

Disadvantages

- Limited to what can be accessed on the devices
- Hard to provide support over various Web browsers
- Not easy to find and promote, users will have to browse websites
- Can be insecure due to the application being web based

There are various platforms available for mobile devices. The most popular platforms are the as previously mentioned. Although they are in contest the use and popularity of platforms vary from region to region. An article published by Sabri (2013) states that windows phone is the most popular platform in Latin America. Although the platform was produced later than the other platforms this has become popular due to the lower prices of devices. Devices such as the IPhone and the IPad are a lot dearer and can cost a user a couple of hundred pounds. In September Forbes reported that the most popular platform on mobile devices is Android.

Germany	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	78.7	78.7	0.0
BlackBerry	0.6	0.5	-0.1
iOS	11.1	9.5	-1.6
Windows	3.8	8.8	5.0
Other	5.7	2.6	-3.1

GB	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	62.7	56.3	-6.4
BlackBerry	10.1	3.7	-6.4
iOS	21.4	27.5	6.1
Windows	4.5	12.0	7.5
Other	1.3	0.5	-0.8

France	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	62.7	63.3	0.6
BlackBerry	8.2	4.2	-4.0
iOS	13.5	17.5	4.0
Windows	5.6	10.8	5.2
Other	10.2	4.1	-6.1

Italy	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	58.6	71.6	13.0
BlackBerry	3.9	2.1	-1.8
iOS	15.0	14.4	-0.6
Windows	10.3	9.5	-0.8
Other	12.2	2.4	-9.8

Spain	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	84.9	90.8	5.9
BlackBerry	5.6	0.7	-4.9
iOS	2.8	5.3	2.5
Windows	2.1	2.2	0.1
Other	4.7	1.0	-3.7

USA	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	60.7	55.1	-5.6
BlackBerry	2.1	1.8	-0.3
iOS	33.9	39.3	5.4
Windows	2.6	3.0	0.4
Other	0.8	0.8	0.0

China	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	63.2	72.4	9.2
BlackBerry	0.2	0.0	-0.2
iOS	23.5	20.8	-2.7
Windows	4.7	2.1	-2.6
Other	8.5	4.7	-3.8

Japan	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android		48.6	
BlackBerry		0.3	
iOS		47.4	
Windows		0.8	
Other		2.9	

Australia	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	65.9	62.1	-3.8
BlackBerry	1.6	0.5	-1.1
iOS	25.8	28.7	2.9
Windows	3.7	6.5	2.8
Other	3.1	2.2	-0.9

EU5	3 m/e Aug 2012	3 m/e Aug 2013	% pt. Change
Android	68.8	70.1	1.3
BlackBerry	5.8	2.4	-3.4
iOS	14.1	16.1	2.0
Windows	5.1	9.2	4.2
Other	6.1	2.1	-4.0

Figure 2.5: Mobile phone market share September 2013

(Jones, 2013)

2.3.6 Currently available Cryptic Crossword apps

After performing various research on the apple app store, the Google play store and blackberry world, it was discovered that there is not many Cryptic Crosswords available to download. There were two, which could be clearly defined as Cryptic Crossword applications, and these have been analysed in the following sections.

2.3.7 Puzzler Super Cryptic Crosswords

Platform: Apple iOS
Price: £3.99
Compatibility: iOS 4.3 or Later
Website: https://itunes.apple.com/gb/app/puzzler-super-cryptic- crosswords/id616060420?mt=8

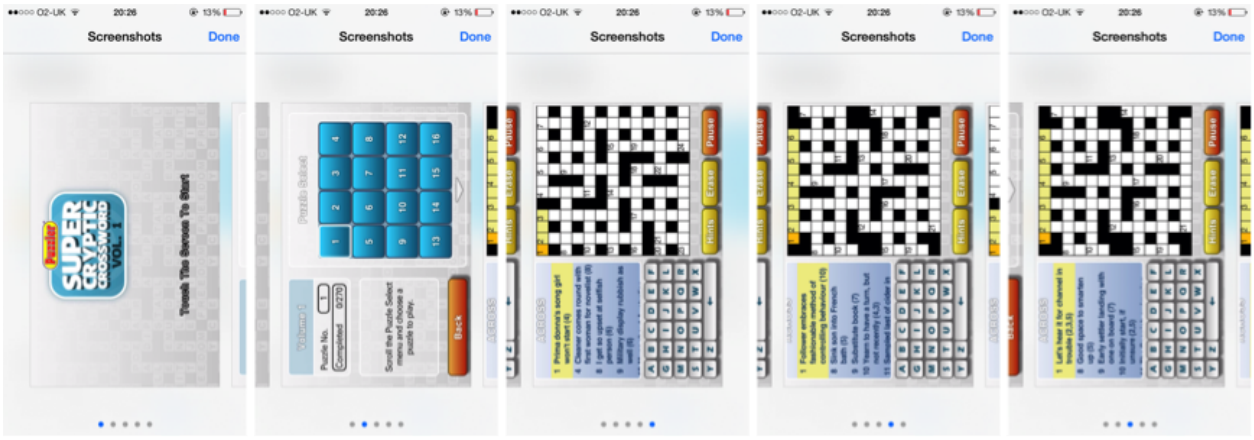

Figure 2.6: Screenshots of Puzzler Super Cryptic Crosswords on IPhone

This application contains 270 pre built crosswords. The interface is really easy to use and it clearly shows what has been completed and what needs to be completed. Whats nice about this application is that the crosswords are in various levels and as you complete one crossword you can move on to the next. What was noticed is that the application already stores the answers and also the application allows hints, which makes it a little easier to use. The other noticeable thing about this application is that it didnt fully use the native features of the mobile phone. Like the keyboard is a custom keyboard.

2.3.8 Cryptic Crossword

Platform: Apple iOS, Android
Price: Free with 2 puzzles on Apple devices- In app purchase avalable - £1.99 on Android Devices
Compatibility: iOS 6.0 or Later
Website: https://itunes.apple.com/gb/app/cryptic- crossword/id661608021?mt=8

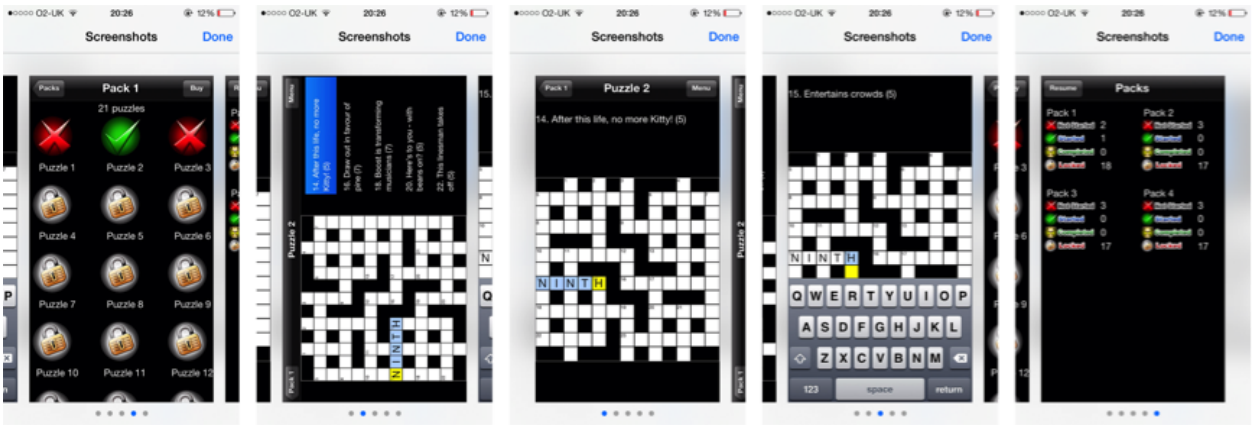

Figure 2.7: Screenshots of Cryptic Crossword on IPhone

The Cryptic Crosswords app comes with 4 different packages and a total of 81 puzzles. Each pack can be bought for £0.69 or all 4 for a discounted price of £1.99. Each package contains various cryptic crossword puzzles. After having a play around with this application at first point of contact it is noticed that look and feel of the application is not that great. The use of colours and the layout styles of the application can be better. The app consists of 81 puzzles, which can be played only after each crossword has been completed. Some of the features of the app are:

- Checking answears
- Revealing a letter, a word or the whole puzzle
- Clearing the puzzle
- Moving the character bar to next box
- Greying out completed clues

These are some of the features the application uses but there are plenty more.

2.4 Web Services

Over the past decade web services have exploded into the computing space. However the concepts that underlie web services are not new. Web services originally evolved from the Remote Procedure Call mechanism that was found in a software development framework used in the 1990s (Kalin, 2013).

During the late 1990s, XML-RPC was developed, which was a stripped down, light weight version of the Remote Procedure Call mechanism. The XML-RPC system only supported a small number of data types along with a number of simple commands. XML-RPC contained two key features, which are use of XML serialise/deserialise for data types and the reliance on HTTP for transport XML-RPC (Kalin, 2013).

XML-RPC is designed to be as lightweight as possible, and thus can be supported on a wide range of devices. XML-RPC was ultimately implemented fully and became known as SOAP. As well as SOAP, another implementation of XML-RPC occurred, which was entitled REST. Both of these technologies fall under the term 'web services'.

Since 2001 a vast range of companies have adopted the web services movement including (but not limited to) IBM, Oracle, Hewlett-Packard, Amazon, Google, Facebook and Twitter (Sullivan, 2001; Kalin, 2013).

Web services generally tend to reside upon public networks such as the Internet. However it is possible for a web service to run within a private network, such as a company's internal Intranet.

2.4.1 What are Web Services?

Although there are many companies adopting web services, the term web service has a diverse and loose definition (Kalin, 2013). During the initial explosion, many providers created heavily detailed plans upon the direction of their web service, but failed to exactly define what a web service is.

It was only until the explosion subsided that authors were able to define what a web service is (Kalin, 2013). Kalin (2013) Kalin highlighted three common characteristics between web service providers:

1. Can be thought of as a 'webified application'
2. Typically delivered over Hyper Text Transport Protocol (HTTP)
3. Typically has some form of distributed nature allowing for components to be deployed and executed across multiple devices.

For the purposes of this project a web service will be defined as:

> A service that contains one or more software components that are designed to allow machine-to-machine interaction over a network using standard protocols.

Web services follow the client-server model, which is the standard architecture for accessing a website. However unlike the traditional approaches to client/server models (such as a web server/web page setup), web services do not provide the end user with a Graphical User Interface (GUI).

The web service will provide the end user with machine readable data — i.e. the data must be put into a pre-defined GUI. This architectural design concept is not new and has been around for a number of years. Web services often can be thought of as imitating mainframes — i.e. a 'dumb terminal' sends a request to a service hosted upon a central computer system.

Web services can be broadly categorised into the distributed software systems category (Kalin, 2013). Broadly speaking a distributed software system is a system that is often split up into various components. Each component can run upon a separate physical machine, and is able to communicate with other parts of the system by passing 'messages' around. Although a web service does fit into that broad definition, there are several features that are unique to a web service.

Firstly web services heavily depend upon open, industry-standard, vendor-independent protocols such as HTTP, JSON and XML. By adding networking, data formatting and security features, web services can effectively lower start-up costs and promote interoperability between new and existing services (Kalin, 2013).

It is the interoperability that allows web services to promote language transparency. This means that web services and client programs do not need to be developed in the same

language. Many popular languages (e.g. C/C++, Java, and Python) provide inbuilt libraries or frameworks in support of web services (Kalin, 2013).

Finally web services are designed to be modular in design. This allows new services to be brought online in staggered stages, as well as allowing for laying of existing services. Again as previously mentioned each new service, can be written in the same language as the previous service, or use a completely new language (Kalin, 2013).

2.4.2 Web Service Categories

Web services can be divided into two distinct groups — SOAP based and REST-style (Kalin, 2013). Interestingly the distinction could be described as being little at most, but they are not necessarily directly compatible with each other.

SOAP

SOAP originally stood for Simple Object Access Protocol, but is often referred to as Service Oriented Architecture (SOA) Protocol (Kalin, 2013). At a glance the name change doesn't appear to be too trivial, but it is acutely an example of the technology becoming better defined (Kalin, 2013).

SOAP utilises concepts that can be seen throughout the industry, but none more so than the use of XML. One of the major advantages of XML, is that it is able to provide flexible, self-describing data structures that can easily be produced and read.

SOAP tries to imitate the postal system — i.e. allowing two machines to send and receive letters. In this analogy, the letter is the raw XML data, and the envelope is an additional data layer that wraps around the letter. The envelope adds additional information to the request, such as which operation is being requested, and may also include authentication and session information in envelope headers (Gershon, 2004).

In order to ensure one client or service can 'talk' to another service, SOAP responses must use a Web Services Definition Language (WSDL). The WSDL defines the inputs (e.g. parameters), the outputs, the operations, the protocols and the network addresses that are required and used by the service (Gershon, 2004).

The underlying implementation is loosely coupled with WSDL, which means the provider is able to change the implementation, without negatively impacting the end service users. It is the configurable services aspect that is the central concept behind all service oriented architectures (Gershon, 2004).

REST

REST stands for Representational State Transfer, and is a relatively new architecture for creating web services. Despite its relatively new architecture it is actively used by some of the larger vendors such as Google and Amazon (DOSPINESCU and PERCA, 2013).

REST relies upon the emerging architecture known as resource-oriented architecture. Essentially, these resources are a number software components that can be combined together to create reusable functionality.

As well as using a resource-oriented architecture, REST makes clever and effective use of open standard web technologies, such as the Hypertext Transfer Protocol (HTTP), the Uniform Resource Identifier (URL) and the Extensible Mark-up Language (XML) (DOSPINESCU and PERCA, 2013).

Although not all of the features have been implemented, (mostly because they are layout properties rather than data properties) the major concepts found in web technologies have been implemented and the most notable of these features are:

1. Data from the client is transmitted to the server via the URI
2. The server will perform the operation described by the HTTP method (such as GET, DELETE)
3. The URI for each resource will contain the server name and address

As previously mentioned, HTTP methods are widely used within REST. A HTTP method will describe the necessary action (Create, Read, Update and Delete — CRUD) that is required to be performed by the server (DOSPINESCU and PERCA, 2013).

The HTTP methods follow another standard in terms of the basic functions of a database management system. It must be said that REST and the HTTP protocol are mutually exclusive — REST doesn't require HTTP (DOSPINESCU and PERCA, 2013).

Table 2.6 describes common HTTP verbs and the associated CRUD operation.

HTTP verb	**CRUD operation**
POST	Create a new resource
GET	Read a resource
PUT	Update an existing resource
DELETE	Delete a given resource

Table 2.6: HTTP verbs mapped to the associated CRUD operation.

There are additional optional verbs, such as HEAD, TRACE, CONNECT, OPTIONS and INFO, but these may not be implemented by the server and/or service for security reasons. Every HTTP request will include a verb to indicate which CRUD operation should be performed upon the resource (DOSPINESCU and PERCA, 2013).

SOAP vs REST

Both REST and SOAP utilise standard protocols when communicating, and also originate from a similar specification. The real difference between the two technologies is that SOAP utilises it's own application protocol by extending current protocols.

This causes a number of issues, such as protocol standardisation. Although SOAP is based upon the HTTP protocol, each client will have to correctly understand the new extended protocol — via an additional layer of software or libraries. This adds weight to the overall technology.

SOAP describes functions, and the types of data, which requires large amount of documentation in order to use the service. As well as this there are several protocols and technologies that directly relate to it, such as Web Services Description Language, Web Servicing Addressing, XML Schema Definitions.

All binary data that is to be transmitted must be first encoded in a supported format (e.g. base64), which increases processing power at both the client and server ends. All requests are transmitted via XML, which is much slower to parse and interpret than other text-based human readable data, such as JavaScript Object Notation (JSON).

REST on the other hand is based upon uniform interfaces. This means the various clients will have a small understanding of the web service, but not necessarily how it operates or what it will return.

REST doesn't need to operate over HTTP, and doesn't contain the complexity that SOAP provides. Rather than utilising XML, REST uses the standard HTTP methods to describe what a service should do. For example obtaining a resource would use GET, and for creating a resource POST would be used.

Clients do not require additional REST supporting libraries. As long as the language supports HTTP, the client will be able to consume a REST HTTP service easily.

Unlike SOAP, REST can deliver binary data without having to encode, and responses can be formatted to either XML or the more popular JSON (due to speed increases).

2.4.3 Clients

As previously mentioned, the broader web server architecture follows the client-server application model. When designing a client-server application, a decision has to be made as to which operations (or parts of operations) should be performed upon the client and the server.

This decision is vitally important as it can affect the speed to which a system can be brought to market. It might also affect any additional extensions or updates that the system might receive in the future, as well as affecting the design flexibility.

In order to simplify the design the client will need to fall into one of the two categories — 'thin' client or a 'thick' client.

Thin

A thin client is a computer system that depends largely upon a main server, or a number of servers in order to complete any computation tasks. The client has no knowledge of how to process data, it simply knows how to pass data to another entity, and receive data from another entity.

A recent example of a thin client is Google's Chromebook. Unlike typical computers where by the applications are installed locally upon the computer, the Chromebook allows for applications to be installed within the cloud — upon an external server.

The thin client design presents a number of advantages and disadvantages. Firstly an application that is hosted upon a central server can be easily updated — as there is only one code base. Once the application has been updated, this will be pushed immediately to all thin clients.

This obviously provides an advantage in some use cases such as trying to sell goods over the Internet. For example if a product's price changes, the update will only need to be applied once to the central server, rather than having to update all clients wishing to purchase the product.

Thin clients will utilise powerful servers to do the majority of the processing. This allows for the thin clients to be less powerful, and hence the overall costing to reduce.

However this will mean that thin clients will have poorer response times. The main reason for this is the fact that the majority of the operations are being complete upon another machine (potentially many miles away). Simple operations such as populating a menu, might require a request to the main server, thus increasing the overall time to achieve something.

Resources within a thin client network will need to be managed more effectively. Thin clients will use more bandwidth upon the network, and will make more connections to the server. This would require the server to be able to handle lots of potentially fast and slow connections, with each connection using a wide range of internal server resources (CPU, Memory etc).

Thick

A thick client is a computer system that has little dependency upon a main server, or a number of servers in order to complete computational tasks. The client will still require a limited connection to a server, but will not use the connection as often in comparison to a

thin client. A thick click will often be able to perform many operations without a connection to a network.

An example of a thick client would be a standard desktop installation. The desktop installation might provide various pieces of software that are installed locally upon the computer. For example the computer would be able to produce various documents regardless of the state of the network connection.

The thick client design presents a number of advantages and disadvantages. Firstly due to the fact that clients are able to do more of the computational work, server specifications do not need to be as high. This allows for cheaper servers to be purchased, and few overheads in terms of running and maintenance costs.

This will also lead to an increase in server capacity, again due to the fact that the client is carrying out more work. This ultimately means that the server is required to do less work, and can hence support a larger number of users.

Thick clients have an increased advantage over thin clients in terms of network connectivity. Thick clients do not require a constant connection to a server. This in turn frees up bandwidth that is being used upon the network, as well as reducing server loads.

Finally the end user is able to store files and applications locally upon the machine. This in turn allows for a faster application start up time, and a reduced file access time. Hence increasing the speed of operations, as well as reducing bandwidth upon the network.

However thick clients are more expensive to purchase, deploy and maintain. The reason being is that there will be more computers with higher specifications. This can lead to more expensive repair bills, should systems fail.

Fixing and troubleshooting become more difficult, simply because there are more machines to troubleshoot and fix should problems occur. This is obviously not a problem if there was a central server, such as found within the thin client model.

Chapter 3

Development Methodologies

In order to ensure all objectives and goals that have been set within this project are completed to the highest quality and upon time, a software development methodology will need to be chosen. Dividing a larger project into a set number of defined processes may seem like additional unnecessary work, but the advantages of this process far outweigh the disadvantages (Knott and Dawson, 1999).

The defined processes combine together to form part of a process model. The process model will allow for the following achievements (Knott and Dawson, 1999):

- Adding an element of control and planning
- Allowing for progress to be mapped visually
- Providing a structured approach to development
- Allowing for a higher quality of code and documentation to be produced

The Systems Development Life Cycle (SDLC) was one of the first formalised methodologies for building software. The SDLC utilises a methodical and structured approach to analysing, designing, building and testing software, to which many methodologies follow this rigid structure (Elliott, 2004).

3.1 Waterfall

One of the main aspects to the waterfall model is the fact that the project is expected to progress down the primary path (Cadle et al., 2010).

The waterfall model takes the major components of any project (requirements, design, implementation, testing and maintenance) and assigns each component a stage of its own. Each component is delivered as the flow down the primary path is completed (Cadle et al., 2010).

The waterfall model also supports backtracking (i.e. reverting back to a previous deliverable). This allows for project managers to check that the project has not expanded its defined scope, and to also ensure that each deliverable flows into the next correctly. It also allows for slight modifications to be made, however making many large changes might affect the project in the long run (Cadle et al., 2010).

3.1.1 Advantages

Cadle et al. (2010) states that the waterfall model houses a number of advantages, including:

- Provides a rigid project structure, that is easy to follow and review
- Deliverables are delivered in project order, one at a time
- Can work well for smaller projects, or for projects where by the requirements will not change.

3.1.2 Disadvantages

However Cadle et al. (2010) also goes on to state that the waterfall model houses a number of potential problems, including:

- Changes are difficult to implement the further a project is down its primary path
- Large projects may not benefit from the rigid structure
- A working piece of software is not delivered until late into the project

3.2 Spiral

A common feature found in the waterfall model is that all requirements are stated at the start of the project. It is these requirements that will form the basis of all work, along with any project planning (Cadle et al., 2010).

The spiral model forms its basis around iteration and prototyping to try to explore the requirements and develop the solution. During each turn around the spiral, a set of requirements are analysed and developed using prototyping (Cadle et al., 2010).

3.2.1 Advantages

Cadle et al. (2010) states that the spiral model houses a number of advantages, including:

- A high amount of risk analysis is conducted, and thus risk is more likely to be avoided
- The model allows for approval from clients, and large amounts of documentation to be produced
- Software can start to be produced earlier, in comparison to the waterfall methodology
- Additional functionality can be added on at any time during or after the project

3.2.2 Disadvantages

The spiral model allows for a high level of control, without too much restriction. However Cadle et al. (2010) states that this can cause difficulties such as:

- A thorough investigation into all of the requirements cannot be achieved early, therefore some requirements (and their priorities) may get completely missed
- The spiral model is based upon the clients knowing exactly what they want, which is unlikely
- A risk analysis must be conducted, and requires highly specific expertise to complete. IF a risk analysis is not completed, then the project may completely fail

3.3 Agile

The agile software development methodology is designed to "reduce risk by delivering software systems in short bursts or releases" (Dawson, 2009).

Each release (sometimes referred to as iterations) will involve minimal planning and will cover all the major SDLC components: analysis, design, implementation and testing. The agile development model also heavily promotes collaboration/development between team members (Dawson, 2009).

3.3.1 Advantages

One of the main advantages of using the agile development model is that software is developed in rapid cycles, which ultimately results in smaller constant incremental releases of software. As well as this major advantage, Dawson (2009) states the following advantages of using the agile development model:

- The methodology surrounds the concept of regular face-to-face meetings as opposed to in-depth documentation
- Utilises a close working relationship between the client and the developers, thus providing continuous delivery of useful software
- Uses shorter, iterative time scales (usually weeks rather than months or years), which results in working software being delivered frequently
- Easily able to change the requirements at any stage (however late the changes are)

3.3.2 Disadvantages

However many of the disadvantages of agile development model are surrounded by the lack of a rigid documentation, as Dawson (2009) also suggests the following disadvantages:

- There is often a lack of emphasis on necessary documentation (user documentation, design documentation etc.), which is normally skipped to save time
- The uncertainty of a specification may lead to poor code and/or structure
- The project can become confused if the original specification is not clear from the start
- Some software deliverables can be difficult to allocate the correct amount of resources (time, effort etc.) at the start of the project

3.4 Rapid Application Development

The Rapid Application Development (RAD) model is an extension to the incremental development methodology. The RAD model states that all requirements should be treated as mini projects, and that they should be completed in parallel. Each of the mini projects are ran like a normal project, and hence time scales need to be adhered to (ISTQB Exam Certification, 2013).

Upon completion of the mini project, the customer is able to review the output, and provide value feedback regarding to the delivery and the requirements. RAD will follow a somewhat simpler primary path, allowing for business modelling, data modelling, process modelling, application generation, testing and turnover (ISTQB Exam Certification, 2013).

3.4.1 Advantages

ISTQB Exam Certification (2013) states that there are many advantages of adopting the RAD model within a team:

- A reduced development time, due to the fact that the business modelling and data modelling processes should cover all aspects
- The combination of Data modelling and Process modelling should allow for the increased ability to reuse components
- Reviews of delivered outputs are constantly reviewed by the customer, allows for early feedback to be gained
- Parts of the system are integrated at an earlier stage, which allows for fewer integration issues towards the end of the project

3.4.2 Disadvantages

However, ISTQB Exam Certification (2013) also states that there can be disadvantages of adopting the RAD model within a team:

- There is a high dependency upon an overall strong team and strong individual performances for identifying business requirements
- The model will only work for systems that can be modularised
- The model assumes that the team members are highly skills designers and developers, with an even higher dependency upon modelling skills

3.5 Summary

In order to achieve the best possible product, it is clearly evident that the project should be developed utilising a feature driven approach. This will allow for any revisions, modifications, and changes to be considered and implemented with as little delay as possible, as well as little impact upon the rest of the project.

The projects requirements are not set directly by an external client, and hence it is possible for the requirements to be changed. It is because of these uncertainties that an agile development methodology would be best adopted by this project.

This methodology will not only allow for the requirements to change, but can allow for substantial research to be able to take place upon new topic areas if needed. Agile development methodologies allow for multiple releases of software, which fundamentally means that the team is able to use prototyping techniques to find the best outcome to a given problem.

Chapter 4

Software Specification

Robertson and Robertson (2013) state that 'a requirement is something the product must do to support its owner's business, or a quality it must have to make it acceptable and attractive to the owner.' The aim of gathering the requirements is ensure that all ambiguities are removed before a product is developed.

Within this chapter the project's aims and objectives will be outlined, as well as project constraints and assumptions.Software requirements will be categorised as functional and non-functional requirements, to which a full MoSCoW analysis will be conduced.

Finally a risk assessment into the various project related risks will be conducted to help identify any potential risks. The risk assessment will also identify how the risks can be reduced.

4.1 Aims & Objectives

4.1.1 Aims

- Produce a software application that accepts the input of cryptic crossword clues from a user
- Produce a software application that will present a set of results to a user for the cryptic crossword clue which has been input
- Effectively adhere to a project plan as a team
- Effectively abide by the guidelines of a chosen software methodology
- Construct systematic and comprehensive documentation for the academic report

4.1.2 Objectives

- To research cryptic crosswords themselves and the various types of cryptic crossword clue
- To study relevant software architectures associated with the group project
- To investigate the field of artificial intelligence, in particular natural language processing and justify the need for it within the project to solve cryptic crossword clues
- To research the various software methodologies which exist and select the most appropriate for the project
- To understand and effectively follow a collection of requirements
- Evaluate the progress of the project at numerous instances throughout the life cycle of the project

4.2 Constraints

The following section describes the constraints that effect the design of the product. The product that is to be developed cannot be successful unless these constraints have been accomplished.

- The project will have four developers operational throughout the life cycle
- The user interface to be developed for the software application must be developed with considerations regarding its simplicity for the usage on mobile phones
- Resources to test the software application on mobile phones will cover the following platforms:
  - Android
  - BlackBerry
  - iOS
- The product shall require an Internet connection
- The product shall make use of the Apache OpenNLP Library
- Possible external resources to be used for the project, such as a dictionary and a thesaurus, may have limits on the usage
- The product shall be completed on or before the 9th May 2014
- The product shall be developed using all freely available tools and the only constraint of budget shall be the time the project resources put in to the product

4.3 Facts and Assumptions

- Answers to Cryptic Crosswords are usually published the following day
- Two clues that are identical, may not lead to the same solution
- Equipment needed to test the application on different platforms will be available from the University
- A server to host the web service will be available from the University
- An external resource (The Guardian) has given permission to use their cryptic crossword data to use for our test data
- Four team members working on the project throughout the year at approximately 25% of their academic study time
- Project scope will remain the same throughout the project

4.4 Scope

4.4.1 Organisation Structure

The project team is based largely upon democratic discussions and decisions, however to ensure that team deadlocks do not occur a project leader has been chosen. Figure 4.1 reflects the hierarchy of the team.

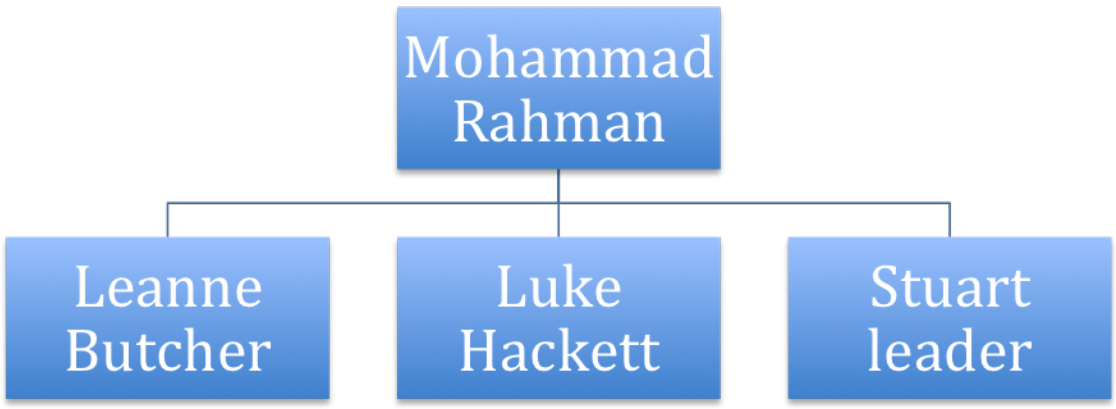

Figure 4.1: Hierarchical Structure of the team

4.4.2 Methodology

It has been decided that the team will follow an Agile software development model. This will allow the team to split the larger task down into smaller, more manageable 'chunks', that allows for a good quality analysis, evaluation, development and planning (on to the next 'chunk'). An iterative approach is best suited for this project due to the nature of changes and updates the product will require in future builds.

This method of development also allows for a more feature-driven approach to the project. Ultimately this allows for more important features and aspects of the project to be completed first.

4.4.3 Meetings

A weekly meeting will take place between all project members to discuss all aspects of the project. This includes (but is not limited to) project issues, software development issues, research findings, possible improvements and code reviews.

4.4.4 Product

Within the cryptic crossword area, there is a need for a cryptic crossword clue solver in the form of a software application. This has been justified previously within the document through research and investigating existing applications within the field. The main purposes of the software deliverable are to allow the input of a cryptic crossword clue by a user and output a potential results or number of results.

Once the project has been completed by the team, the following deliverables will have been accomplished:

- A software application which accepts input from the user and outputs appropriate solutions.
- Two written reports that:
  - document the entire software development process from a group perspective
  - analyse and evaluate the project as a whole

Furthermore the subsequent internal components will be implemented to aid with the goals of the project:

- A storage area to collect and store the cryptic crossword clues and their details used for test data as well as, clues successfully solved by the application which a user has input.
- A service which will allows connections from web browsers and potentially mobile phones to ensure the running of the software application.

There are specific criteria which have been identified as important features which must be completed to deem the project as adequately finished:

- A document outlining the full software development life cycle of the project
- A software application which can be accessed via either:
  - A web browser **and/or**
  - A portable device (e.g. smartphone or tablet)
- A service which can be connected to by either a mobile phone or a web browser which will sufficiently solve a cryptic crossword clue and output a result or number of results

- A storage area accessible by the service which stores the cryptic crossword clues and their associated details

Furthermore, there are specific criteria which have been deemed as out of the scope of the project:

- The user will not be able to access the storage area to browse through data
- The software application will not be implemented to have the ability to generate cryptic crosswords from the data stored

The restrictions listed below are the justifications for the project scope:

- The time scale of the project
- External priorities from the academic year such as other module assignments and examinations
- Limited resources for testing the software application on restrict the platforms which the software application will be fully compatible with

4.5 Functional Requirements

4.5.1 MoSCoW Analysis

Functionality	**Must**	**Should**	**Could**	**Won't**
Retrieve and process cryptic clues and output possible solutions	Yes	-	-	-
Obtain characteristics of the clue's solution (e.g. number of words, word lengths and known letters) and match these against possible solutions	Yes	-	-	-
Provide a web interface for users to interact with the system	Yes	-	-	-
Provide a mobile-friendly interface for users to interact with the system	-	Yes	-	-
Store clues and their corresponding solutions for future retrieval, including the clue type and solution length	-	Yes	-	-
Take feedback from a user on a solution's accuracy and use this to rank solutions for a given clue	-	Yes	-	-
Relay the process (solution trace) followed to arrive at a proposed solution to the user	-	Yes	-	-
Holistically determine a confidence score for each proposed solution, and relay this to the user	-	Yes	-	-
Document and store the process (solution trace) followed to solve a given clue	-	-	Yes	-
Solve clues which take cues from the clue's orientation or clue number in the crossword	-	-	Yes	-
Provide real-time feedback of the processes being followed by the application in an attempt to calculate the correct solution	-	-	Yes	-
Take annotations provided by the user of the known aspects of a clue, such as the definition word(s), fodder or indicator word(s)	-	-	Yes	-
Store a clue's orientation and clue number in the database	-	-	Yes	-
Use stored data to generate complete cryptic crosswords	-	-	-	Yes
Provide a personalised service, including a history of a user's interactions with the system	-	-	-	Yes

Table 4.1: MoSCoW analysis of the project's functional requirements

4.6 Non-Functional Requirements

4.6.1 Performance

The performance of the system should be good, however it must be noted that the system is intended to run as a web service upon a server. The quality of the connection between the end user and the web service will be out of the scope of this project.

4.6.2 Legal and Access

Within the project a number of data sources will be used. Firstly an English dictionary will need to be accessed in order to allow the system to deduce if a given arrangement of characters formulates an English word.

In order to achieve this, an offline dictionary and an online dictionary will need to be used. The offline dictionary is a short dictionary and is provided as part of the GNU/Linux operating system under the GNU Licence Agreement.

A number of cryptic crossword clues and solutions have been obtained from the Guardian's website. The data will be used as a training set of data, when testing the final software. The Guardian has given us permission to use their data as long as the project remains within an academic environment, and that no profit is made from the project.

4.6.3 Backup and Recovery

The project will make use of the git revision control system over a secure shell connection. The service provider is github, who offer a distributed setup to ensure that data is always available. Using a secure revision control system will ensure that:

1. A copy of the project is stored upon a remote, secure server;
2. Changes are able to be tracked;
3. Issues and comments are able to be raised in a secure environment.

The above measures will in effect be able to reduce against the loss, damage or theft of all project information.

4.6.4 Archiving and Retention

The University will retain a copy of the project (software product and the final report) for assessment and evaluation purposes.

4.6.5 Maintainability

The project – including the software product – will be maintained up until the project hand in deadline: 9th May 2014.

4.6.6 Availability

The software will not contain any request or time limitations for users, and thus should be available at all times. However it must be stated that the server availability that will be hosting the web service is out of the scope of the project.

4.6.7 Capacity

The end product should be able to handle a number of requests from different users at the same time. As the project is only intended as an academic project, large scale user support will be out of the scope of this project.

4.6.8 Usability Requirements

Functionality	**Must**	**Should**	**Could**	**Won't**
Provide a text field for the user to input the cryptic clue to be solved	Yes	-	-	-
Provide a drop-down box allowing the user to input the number of words in the solution, with an initial upper-limit of 10 words	Yes	-	-	-
Dynamically provide individual text boxes for each word of the solution, which allow the length of the words to be specified. Also provide check-boxes between these text-boxes to define whether they are separated by a space or a hyphen	Yes	-	-	-
Dynamically provide text boxes to represent each character of each word of the solution, allowing the user to input any known characters	Yes	-	-	-
Provide a table of results which allow the user the to view the possible solutions	Yes	-	-	-
Alert the user to required fields with red asterisks	Yes	-	-	-
Have a consistent layout avoid unnecessary scrolling	-	Yes	-	-
Alert the user to invalid input through validation checks with error messages	-	Yes	-	-
Provide a confidence rating in the table of possible solutions	-	Yes	-	-
Allow the user to select a proposed solution in the corresponding table and mark this as correct	-	Yes	-	-
Display help buttons to indicate to the user the purpose and use of each control	-	-	Yes	-
Provide a button to submit the user input to the application for processing	-	-	Yes	-
Provide a group of radio buttons for the user to select the clue's orientation in its containing crossword	-	-	Yes	-
Provide a text box to input the clue's number within its containing crossword	-	-	Yes	-
Provide mechanisms to accommodate for users with difficulties, such as colour-blindness or poor eyesight	-	-	Yes	-

Table 4.2: MoSCoW analysis of the project's usability requirements

4.7 Risk Assessment

An investigation into the potential risks that may present themselves during the course of this project will allow the group to effectively minimise their impact should they become a reality.

Many facets of risk management contribute to the potential for a project that will be completed on time and with little interruption from unexpected occurrences. According to McManus (2003), not only will risk management improve the flow through the project due to a reduced overall project risk, but fewer surprises will allow for a more accurate schedule, and hence a greater likelihood of finishing on time.

4.7.1 Risk Identification

Risk (Threat)	**Description**	**Prob (%)**	**Impact (/10)**	**P-I Value**
Workload constraints	Fluctuations in the workload from university modules mean that certain times of the year are typically busier than others, as time and effort have to be delegated to the tasks at hand. This is especially applicable when assignment deadlines are imminent.	80	7	5.6
Underestimated system complexity	Without a thorough investigation into the project's requirements and comprehensive designs, it may become evident that the team has not accurately ascertained the complexity of the task at hand.	60	6	3.6
Project scope incorrect	A misunderstanding of the scale of the project may present significant problems and have adverse effects on the team's ability to produce a quality product on time.	40	6	2.4
Lack of understanding	The team may simply not understand a given task and be able to continue with the project in the expected fashion.	40	4	1.6
Inadequate facilities/resources	Potential resources that may be required include web servers, database management systems, phone app development kits and dictionaries / thesauruses. Seamless access to each and every resource may prove difficult.	70	2	1.4
Ineffective team structure	Explained elsewhere in this document, the team have adopted a largely democratic model. As pressures and workloads increase, conflicts may arise and the team structure may hamper the group's ability to resolve any issues effectively.	40	3	1.2
Skills mismatch	The team may find that the skills required to complete the project are beyond those which are currently held. This has been identified as a risk as the group's collective experience is relatively low.	15	4	0.6
Workforce reduction	The most likely cause for this risk to become a reality is if a team member leaves the university course. As there are only 4 members working on the project, this would equate to a 25% reduction in the workforce.	5	8	0.4

Table 4.3: Probability-impact table for the project's risks

4.7.2 Risk Classification

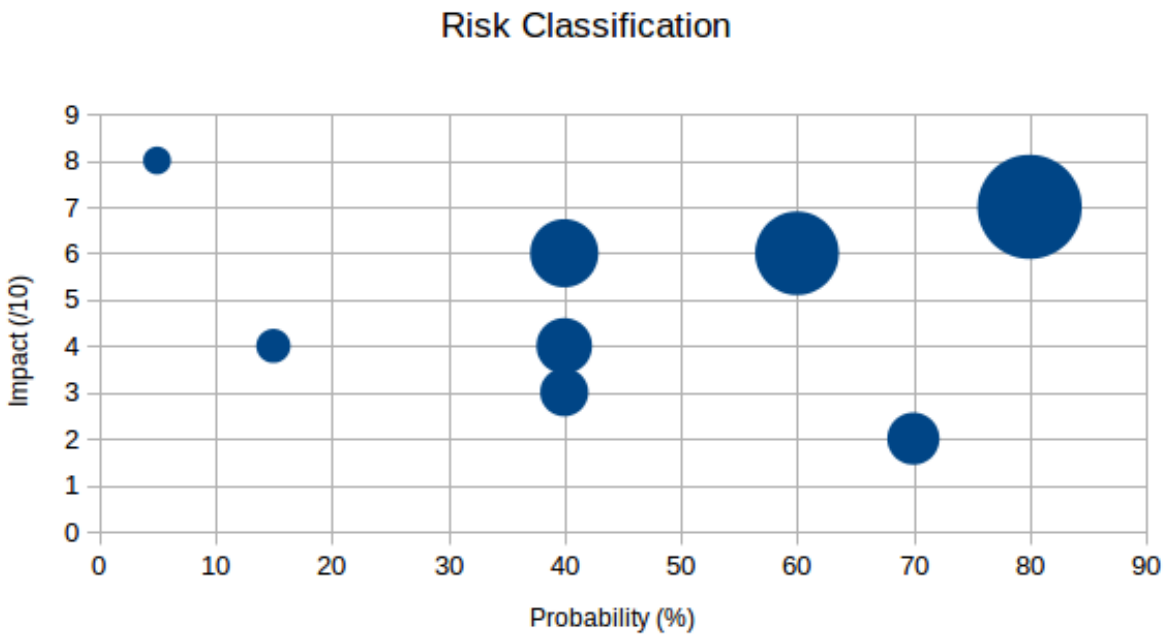

Figure 4.2: A visualisation of the identified risks of the project

Figure 4.2 demonstrates the impact and likelihood of the identified risks occurring. The risk with the greatest PI (*Probability / Impact*) score is the risk of constraints from other university commitments interfering with the expected progression of the project. This is further compounded by the fact that each member of the team is enrolled on the same course of study, and so the impact from such commitments will be magnified across each and every member.

In contrast, the risk with the lowest identified PI score is that of a group member leaving the course, and hence the project. While this has the largest impact score identified, the risk of such an occurrence is minimal. This is categorised by Stoeller (2003) as an *alligator*. This is defined as a risk with a high impact, but a low probability of becoming a reality.

4.7.3 Risk Prevention

Workload constraints The appearance of this risk balances on each team member's ability to effective manage their time. If a consistent effort is maintained throughout the course

of the academic year, there will likely be a greater chance of this risk being prevented. Otherwise an sudden increase in workload will detract from the team's collective ability to work on the project. Project planning should include contingencies for unexpected occurrences, especially around the time of pressures or deadlines from other course modules.

Underestimated system complexity A thorough set of design documents based on the gathered requirements will minimise the potential for this risk to show itself. A range of diagrams from use-case diagrams to sequence diagrams will allow for a greater understanding of the identified tasks, thus reducing the risk of encountering unexpected complexity.

Project scope incorrect & Lack of understanding A recurring theme of this project is that a lack of planning may turn into magnified issues further down the line. The team members should be able to gather a reasonable understanding to the project's scope during the earlier phases. Conclusions may be drawn from the research and requirements analysis phases which hold some indication to the potential size of the project.

Inadequate facilities/resources The team will need to draw on a host of resources in order to efficiently work on the project. Repository and versioning systems will be used to maintain documentation and a code base. Development environments and development kits will be used to aid the software development, and servers and frameworks may be required to host any applicable software. To prevent the risk of issues arising from any of these software packages, it may be best to forecast which software will be required along with an estimation of when this is likely to be. Consequently provisions can be made in advance to ensure the software is set up.

Ineffective team structure Despite a unanimous vote for a democratic structure within the group, a team leader has been selected (*Mohammad*) who will hold the capacity to make an overriding decision on the project if the team cannot come to an agreement unaided. It is also the responsibility of Mohammad to delegate work where he feels it necessary. Such actions may collectively contribute to minimising the impact of this risk, but they may not allow for it to be entirely prevented.

Skills mismatch To ensure the team possess all the skills required to create the proposed solution, the gathered requirements, and in particular the technical requirements will outline the technologies earmarked to be used. As this phase is positioned relatively early on in the project, there should be relatively little chance of realising we require skills we do not possess further on in the project.

Workforce reduction A decision to leave the course before completion will likely be one that will not be taken lightly be the respective student. Consequently this is a threat for which there is little that could possibly be done to prevent it, other than allowing for an amiable working environment amongst each team member.

4.7.4 Risk Mitigation

Workload constraints Effective project planning should prevent this risk from manifesting itself in the project, but should it occur, it may be possible to balance the project workload more effectively amongst the team members. It may be possible that the anticipated workload of each team member can be mapped, and planned work on the project can be temporarily postponed until these constraints no longer present an issue.

Underestimated system complexity An assessment of the implications on the project should be conducted, and the effects investigated. If it would not be possible to incorporate the additional complexity using the time and resources available, the project scope may be reduced to allow for a smaller, but fully functional product. Research into development methodologies suggest that an agile methodology will be adopted. This will allow for a fast reaction to any change seen in the scope or structure of the project.

Project scope incorrect The group should draw on experiences from encountering this risk, and reflect on why it became a reality. A revised project scope should then be created, using these deductions to ensure the risk doesn't manifest itself twice.

Lack of understanding & Skills mismatch The obvious solution to mitigate this risk is to acquire the understanding necessary to carry forward with the work. Contingencies in the project plan may allow for such tasks to be carried out with little impact on the overall schedule. A variety of resources should be utilised to ensure the most effective acquisition of the knowledge necessary to proceed.

Inadequate facilities/resources In the case that a particular resource is unavailable, the team should conduct an assessment of why this is the case and what can be done to rectify the issue. If the issue is a delay in the time it takes to acquire a particular resource, the most viable option could be to search for alternatives or rearrange the schedule to complete other tasks which are non-dependent on this particular task.

Ineffective team structure If the chosen team structure is proving ineffective, it may be necessary to adopt a different set-up for the benefit of the group. With the team members

working with one another over a span of months, each members' working style in a group environment should become apparent, and this may help to shape any new structure.

Workforce reduction The time frame for the project's completion is fixed, but a reduction in workforce may necessitate a respective reduction in project scope or complexity. For the remaining three team members to attempt to complete the work set for four would possibly result in a product which is either unfinished or below standard. While it may prove frustrating to reduce the scope, this should allow for a subset of the complete solution, but which is just as refined.

4.8 Feasibility Study

A feasibility study will be carried out to identify the difficulty of the implementation needed for each type of cryptic crossword clue with the aid of the research carried out previously. The study will focus on the resources needed for each implementation which will contribute to the level of difficulty for each clue type. Furthermore, using the previous research resources within the cryptic crossword field, a measurement of how common each clue type is within a cryptic crossword will be presented to assist in the understanding of the necessity of the clue type within the project.

Aspects such as how common clue types are featured within research resources and how regular indicators are found in the project test data will contribute to the judgement made on how regular a type of clue is. The justification of the use of specific resources for a clue type will come from research resources and inspecting project test data.

Clue Type	**Possible Resources**	**Difficulty**	**Regularity**
Hidden	Dictionary	Low	Common
Anagrams	Dictionary	Low	Common
Acrostics	Dictionary	Low	Intermediate
Pattern	Dictionary	Low	Intermediate
Homophones	Homonym Dictionary	Medium	Common
Charades	Abbreviations, Dictionary, Thesaurus	Medium	Common
Deletions	Abbreviations, Dictionary, Thesaurus	Medium	Common
Reversals	Abbreviations, Thesaurus	Medium	Common
Palindromes	Dictionary, Thesaurus	Medium	Intermediate
Double Definition	Dictionary, Thesaurus	Medium	Intermediate
Substitutions	Abbreviations, Dictionary	Medium	Rare
Shifting	Dictionary, Thesaurus	Medium	Rare
Exchange	Dictionary, Thesaurus	Medium	Rare
Spoonerisms	Dictionary, Thesaurus	Medium	Rare
Containers	Abbreviations, Dictionary, Thesaurus	High	Common
Purely Cryptic	-	High	Common
& lit	Abbreviations, Dictionary, Thesaurus	High	Intermediate

Table 4.4: Feasibility Study for Clue Types

Chapter 5

Design

This chapter will describe the various design decisions that were taken in order to allow for the system to be developed. The chapter will present a number of artifacts that are expressed using the the Unified Modeling Language (UML).

Additional non-UNL diagrams will also be presented to describe the database and user requirement design decisions.

5.1 Primary Path

Before being able to correctly design a large, complex system a good understanding of the main scenario through through the system will be required. The main scenario is "a sequence of event or actions" (Lunn, 2003) in which many smaller scenarios may occur.

The main scenario of the proposed cryptic crossword solver system is outlined below:

1. Input the cryptic clue to be solved
2. Input the pattern of the solution
3. Process the data based upon a number of algorithms
4. Output the most confident results

Based upon the previously given main scenario, the primary path of the entire system can be proposed. The primary path is defined as the most commonly used 'route' though a system with as few variances as possible (Lunn, 2003).

The proposed primary path for the cryptic crossword solver system is outlined below:

1. Input the cryptic clue to be solved
2. Input the solution length
3. Input the pattern of the solution
4. Determine the type of clue that has been given (e.g. anagram)
5. Run the given clue type algorithm
6. Rank each of the results based upon a pre-defined criteria
7. Output the top ranking results

The primary path outlined above takes a relatively generic approach with regards to how a clue should be solved. Ideally the clue would be categorised into a certain type of clue, thus requiring that particular solver to be run.

However the primary path does not illustrate the steps that are required to be taken if the clue can not be categorised — does the system simply stop or would it try an alternative approach?

These additional steps may often be 'optional', and hence are described as alternative paths. The alternative path will contain small and subtle changes to the primary path (Lunn, 2003).

If the system is unable to determine the type of clue that has been given then an alternative approach is to be used. In this instance, the primary path would house the following additions and changes:

3 Determine the type of clue that has been given (e.g. anagram)

    3.1 Start a new instance of each solver

4 Run the each of the solvers separately

The above alternative path shows that if the clue can not be categorised, then the system will attempt to solve the clue utilising a brute force approach. The above alternative path, could also be regarded as a 'worst case scenario'.

The outlined primary and alternative paths are a simple way of illustrating the the general ideas behind the system. These paths will be actively referred to throughout the design process.

5.2 Activity Diagrams

Within the Primary Path section a primary path analysis was conducted, ensuring that common paths throughout the proposed system are identified. Within this section a number of activity diagrams will be presented that will break the primary path into a number of simple processes.

An activity is a task in a process that allows for small meaningful work to happen. A process may have a number of activities linked together to form a work flow (Lunn, 2003).

Figure 5.1 on page 80, illustrates the main activity diagram for the complete system. The activity diagram follows what was discussed within the Primary Path section, and expands upon some of the generalised points. Over the following subsections key parts of the main activity diagram will be discussed in more detail.

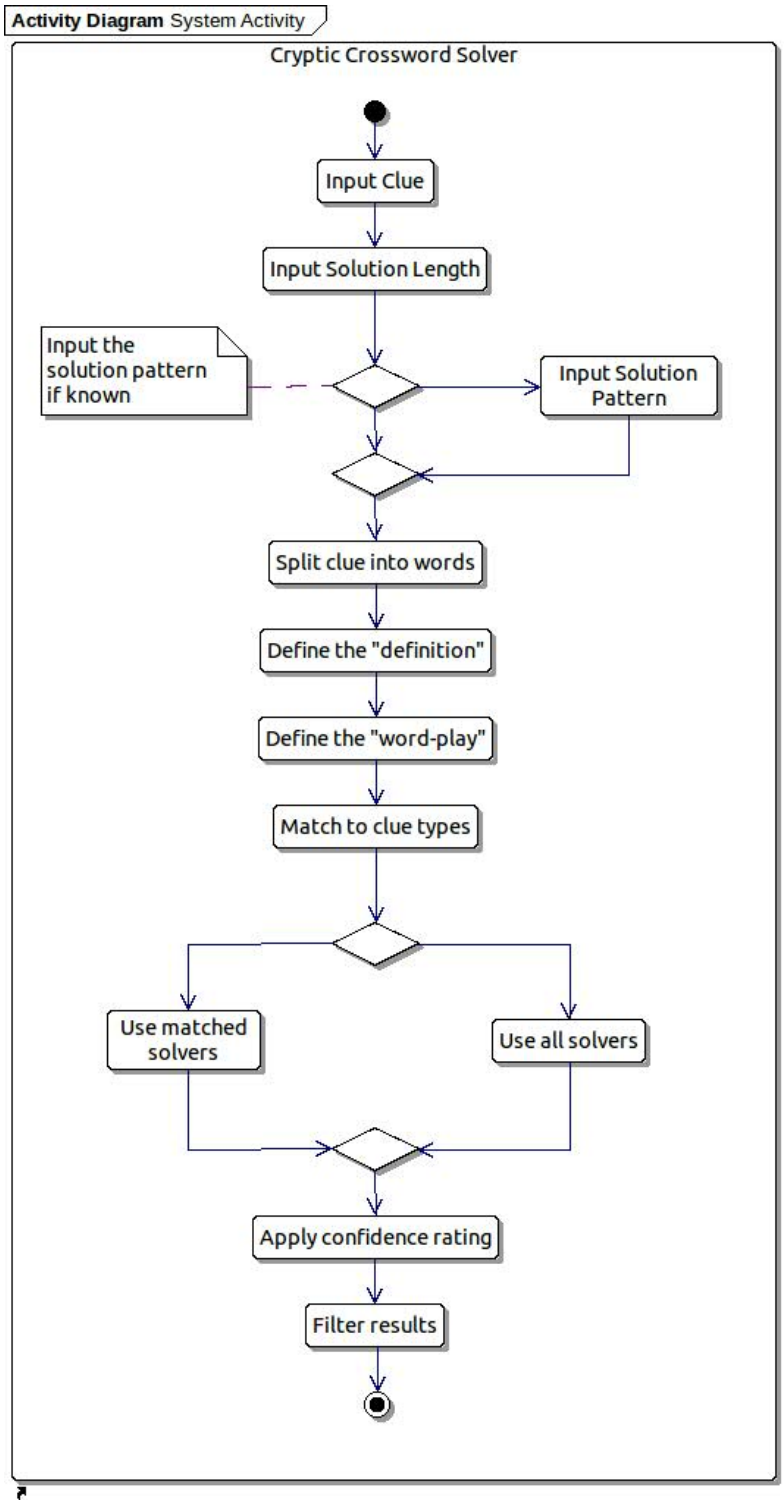

Figure 5.1: The complete Activity diagram

5.2.1 Clue Input

Figure 5.2 illustrates how the data should be input into the system. In order for the system to solve a given clue it requires the following three inputs:

- the clue
- the solution length
- the solution pattern

At first it may seem that the solution length is unnecessary, as it could be computed from the solution pattern. The reason for including the solution length is that it provides an additional validation element — much like having to type in a password twice when creating a new online account.

The solution pattern allows a user to predefine the required answer, based upon a number of characters. The solution pattern will accept all letters as well as three predefined 'special characters', as outlined below:

? ⇒ wild card character

- ⇒ hyphenated word

, ⇒ word separator

For example, the pattern "'????e?-in-???" could match to "mother-in-law" or "father-in-law", just as "??????? d????" could match to "sitting ducks". As previously mentioned, a user may simply present the correct number of wild card characters to denote an unknown pattern.

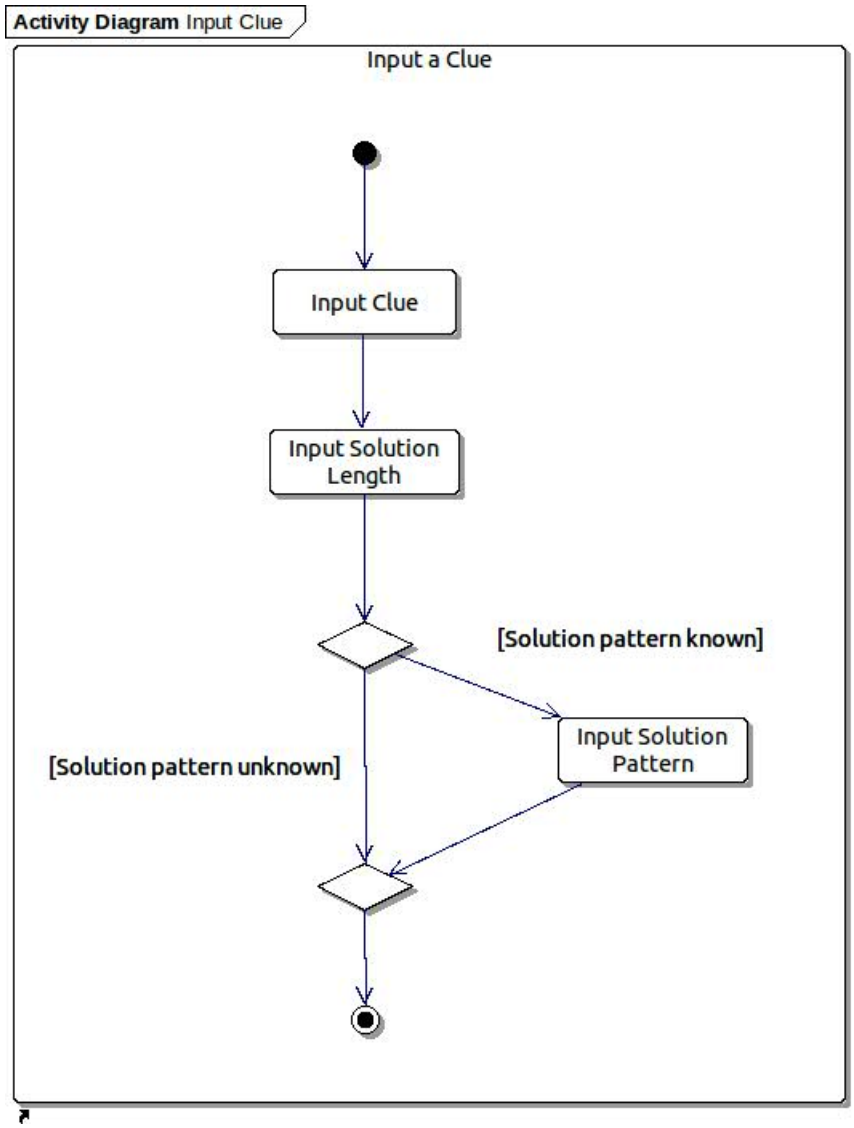

Figure 5.2: Activity diagram showing how a user would input a clue

5.2.2 Choosing a Solver

Once the user has successfully input the three variables, the system will then try to determine the type of clue. There are a number of actions that must be performed in order to deduce the type of clue, as shown in figure 5.3.

The system will try to match the given clue to a list of properties that each clue has. For example spoonerism clue types, will have the word "spooner" within the clue.

If the match is successful then the system will bump the ranking of that particular clue type. Once the system has finished applying the categorisation 'tests' to the clues, it will select

the top solver to try to solve the clue.

If the chosen solver is unable to solve the clue, then the next ranking solver will try to solve the clue. This process will continue until a set of results have been returned, or if all solvers have been used.

If the categorisation fails for some reason, then the system will try to brute force it's way to answer, by using all solvers.

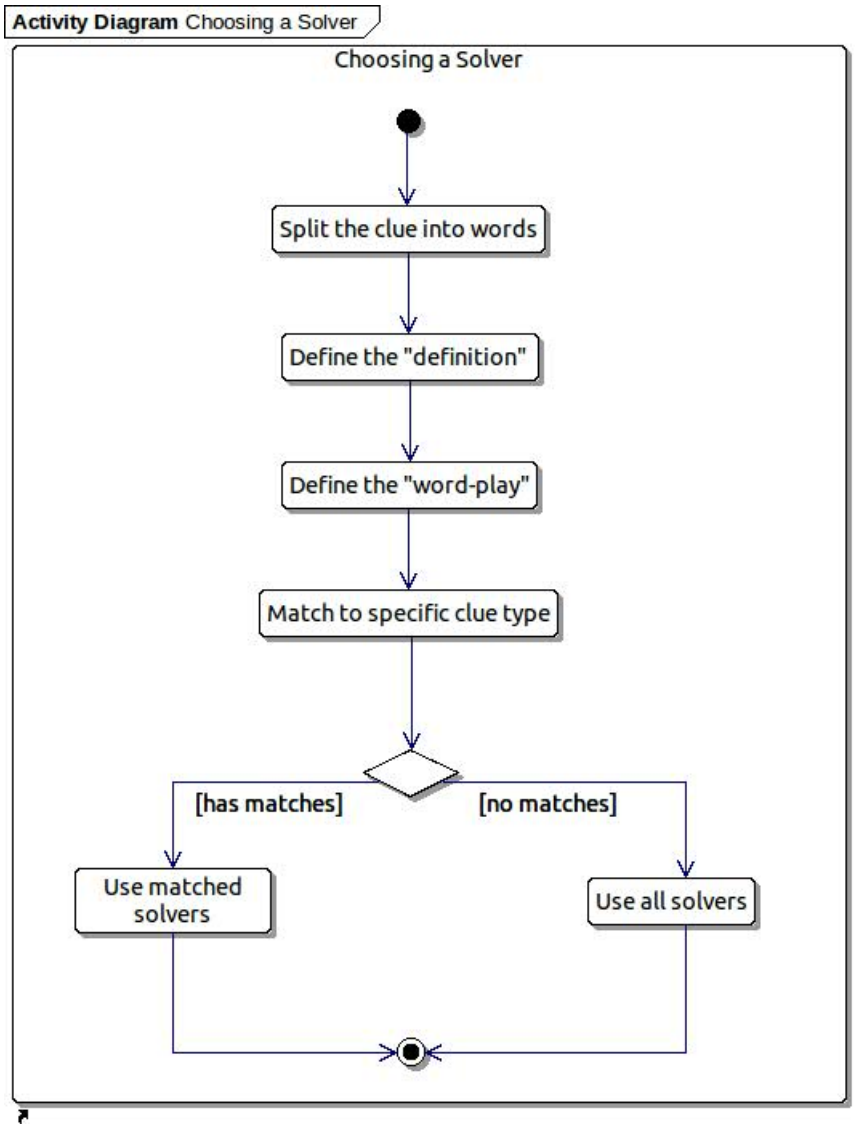

Figure 5.3: Activity diagram showing how a solver is to be chosen

5.2.3 Handling Results

Once the solving process has finished, the system will then attempt to apply a 'confidence rating'. The confidence rating of a solution is how close the system 'thinks' the solution is to the correct answer.

For example, if a solution had a confidence rating of 100% it would deemed as the only answer, whilst 0% would indicate there is no chance of the solution being correct.

It is intended that if there are a large number of results, then the system may choose to filter the results down. For example if thousands of results are generated, then the system may choose to only return results that have a confidence rating higher than 70%.

Figure 5.4 shows the handling of results as an activity diagram.

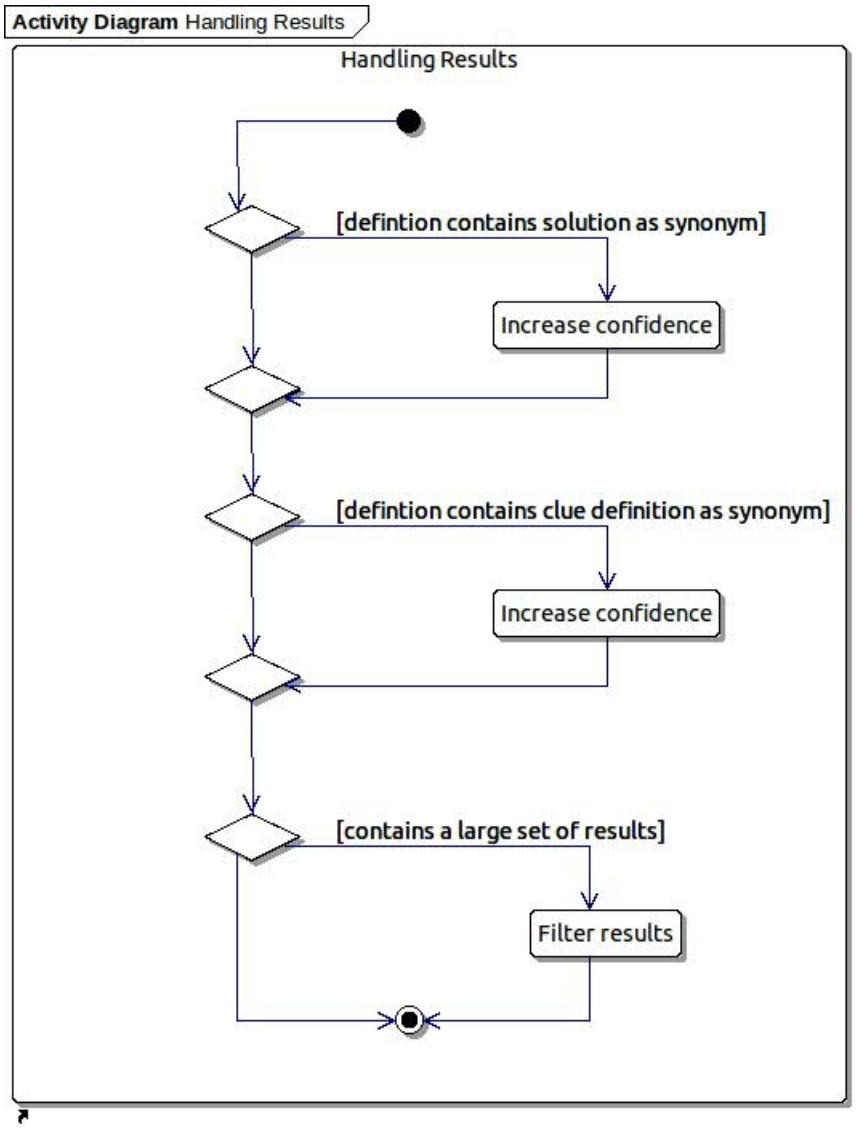

Figure 5.4: Activity diagram showing how results should be handled

5.3 Use Case

A Use Case diagram illustrates "a meaningful interaction with a computer system", and provides various approaches to analysing system design requirements (Lunn, 2003). Within this section the use case of the entire proposed system will be presented and discussed. The use case has been split into two diagrams to increase the readability of the diagram.

The primary intended use of the system is to be able to submit a clue to be solved, which requires the clue, solution length and solution pattern. Figure 5.5 illustrates the three parameter requirements in order to submit a clue to be solved.

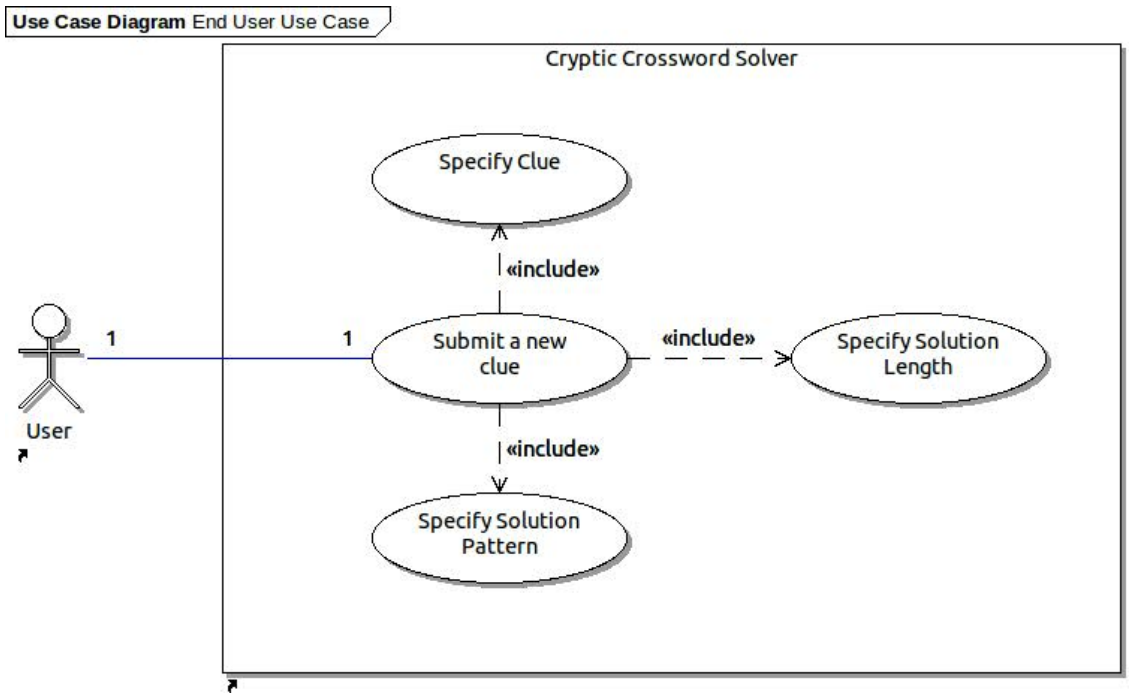

Figure 5.5: Use case illustrating the actor's range of possibilities

It must be stated that the solution pattern could be unknown. In this instance a set wild card characters must be given to the system. This was previously shown in figure 5.2 on page 82.

The remainder of the Use case is shown in figure 5.6. Once the user has input the various data elements the system will then attempt to solve the clue.

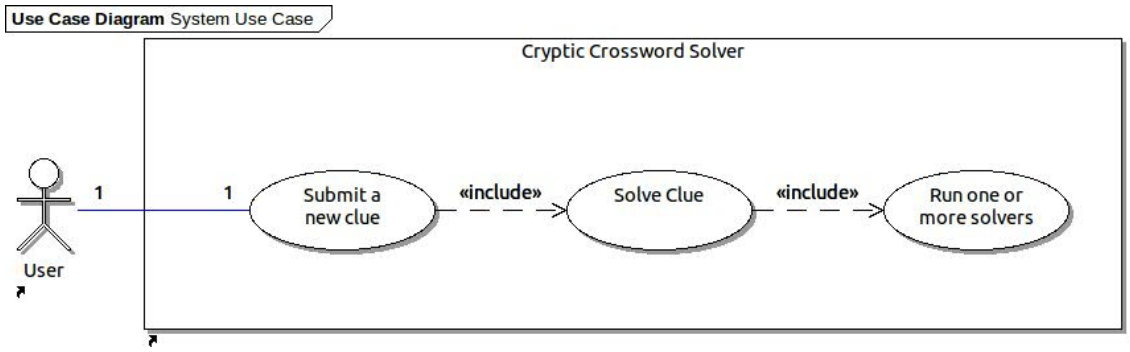

Figure 5.6: System Use Case

The use case demonstrates that there is very little user input required, and once the data parameters have been entered there will be no additional user interaction required.

A design decision has been taken so that there is minimal user interaction with the service. This will be particularly useful for those who are using a mobile device, as inputting data can be laborious in comparison to a desktop machine.

5.4 Sequence Diagrams

Within the previous Use Case section a number of key, generic functions were identified. In order for a sequence diagram to be developed, the generic functions will be converted into a set of objects and interactions. Each of the objects and interactions will eventually form the classes that will be used to build the system. Therefore a sequence diagram can be described as a "way of describing a journey through a system" (Lunn, 2003).

The two sequence diagrams that will be described within this section focus upon submitting a clue and solving a clue.

5.4.1 Submitting a Clue

Figure 5.7 illustrates a user sending data to the web service (servlet). Within this diagram a number of key elements such as how a clue will be solved have been removed. This is to ensure that the diagram is as simple as possible, as this diagram will only be focusing upon how a user sends and receives data.

The Solving a Clue subsection on page 89 describes in detail how a solver will solve the clue.

To simplify the diagram the user is submitting a 'block' of data, which in fact would be the three main parameters — the clue, the solution length and the solution pattern. This has been shown upon the diagram as a 'block' of data, because all three parameters will be sent to the system at the same time.

Once the data has arrived at the servlet, the data will be validated. If the data is deemed to be invalid, the servlet will send an error response back, finishing the current instance.

If the data is deemed to be valid, then the system will try to categorise the clue, and solve the clue using the required solver(s). This is will be discussed in more detail in the Solving a Clue subsection on page 89.

Once a set of solutions have been generated, the the in-memory set will be converted to either JSON or XML depending upon the client's request. Once converted the data is returned as part of the response, satisfying the client's request.

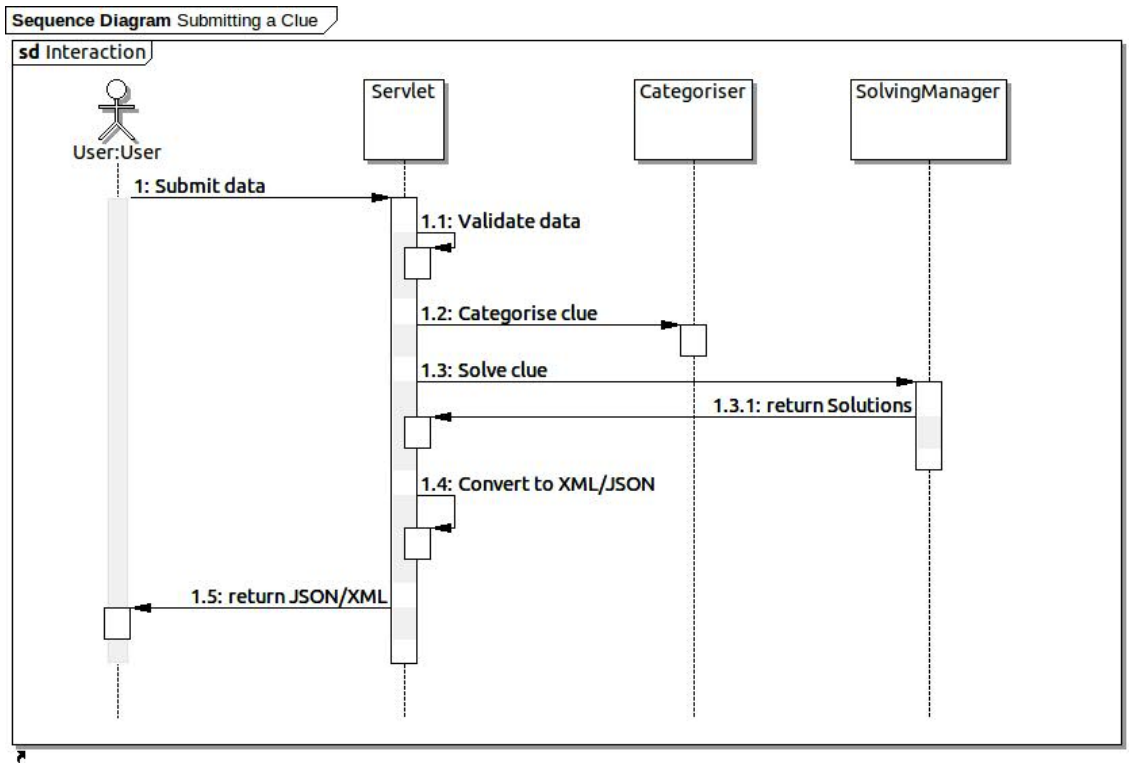

Figure 5.7: Sequence digram illustrating a user submitting a clue

5.4.2 Solving a Clue

Figure 5.8 illustrates how a given clue will be solved. This sequence diagram assumes that valid data has been passed to the 'SolvingManager'. This process was described in the previous subsection entitled 'Submitting a Clue' on page 88.

The 'SolvingManager' object will manage the process of solving a given clue. In order to do this it will have to house several processes, including distributing the clue out to one or more solvers, and merging all results.

Figure 5.8 illustrates the distributing and solving of the clue as a synchronous process, however this was used for illustrative purposes only. The system will in fact distribute and solve the clue upon an asynchronous basis. This will mean that each solver will be running at the same time, and thus reducing the total amount of time needed to solve the clue.

The system will wait however, for all solvers to finish, ensuring that all solutions can be returned to the user. The merging of the results will simply remove any duplicate solutions, upon a first come basis. So if a solution is already in the list, it will not be re-added because a different solver has managed to find the same solution.

Finally the confidence ratings will be adjusted. The adjustments are made upon a number

of factors, and were described in figure 5.4 on page 85.

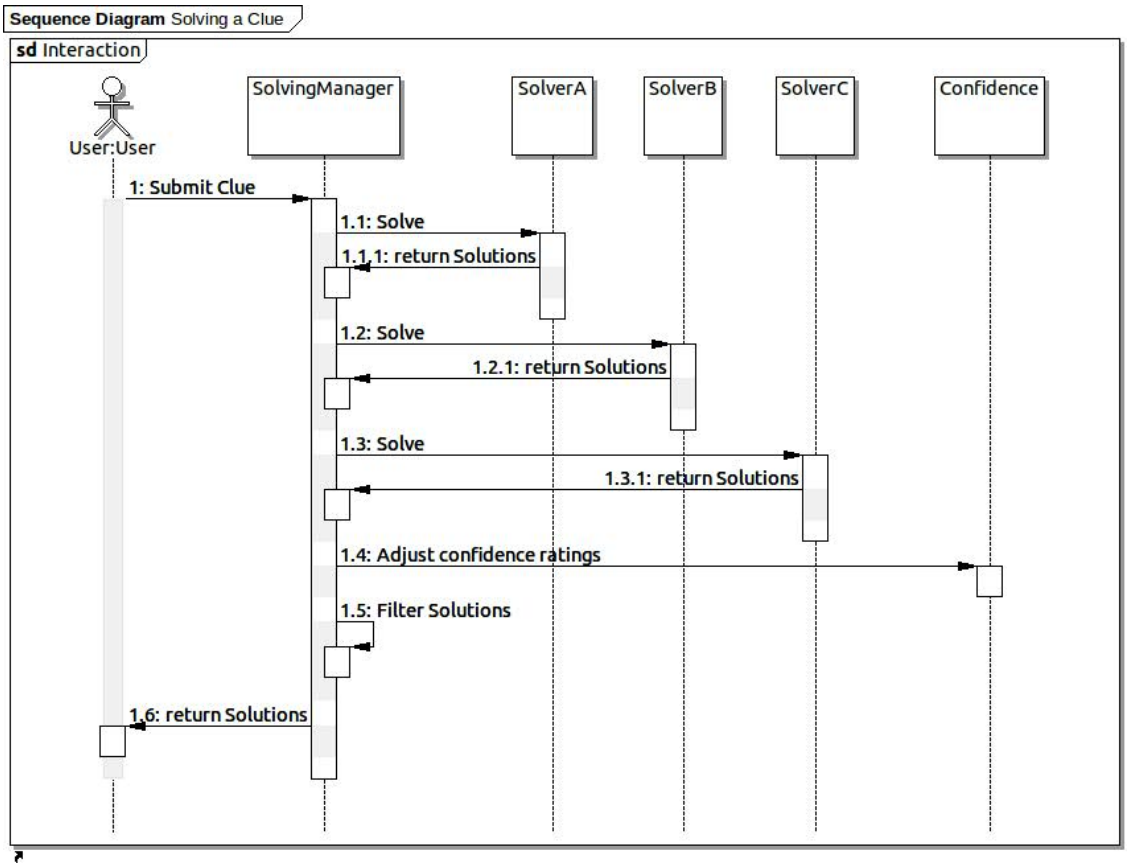

Figure 5.8: Sequence digram illustrating the system solving a clue

5.5 Class Diagrams

Within this section a range of class diagrams will be presented an discussed. “A class diagram describes the structure of a system by highlighting the system’s classes, attributes, methods, and relationships with other classes” (Lunn, 2003).

Figure 5.9 illustrates the seven packages that make up the cryptic crossword solver. Over the following subsections, each of the packages will be described in more detail.

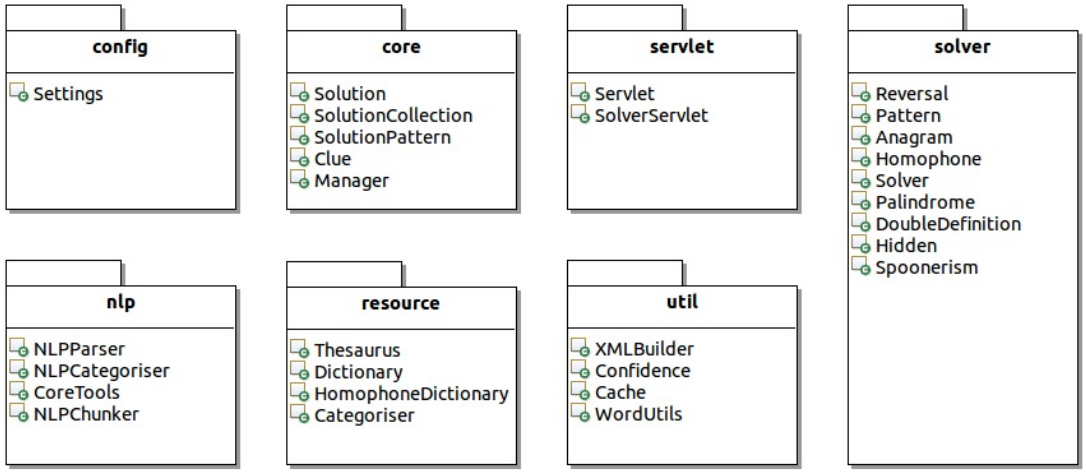

Figure 5.9: Package overview of the Cryptic Crossword Solver system

The architecture of the system will utilise a ‘plug and play’ approach. This approach, will allow for various numbers of solvers to be added at run time, rather than a compile time.

From a development point this will allow for the underlying system to be developed and tested utilising a limited number of solvers. Once the underlying system has been completed, solvers can then be added at any time, without having to alter any previously written code.

5.5.1 Config

The configuration class only contains one class, Settings. The Settings class is designed to hold all application specific settings, and each setting can be altered based upon the mode it is being run under — for example development, testing or production.

The Settings class follows the Singleton design pattern which ensures that the class will have only one instance (Gamma, 1995). This prevents various numbers of the same Settings objects being used in memory, and thus wasting valuable space.

Figure 5.10 illustrates the config package, containing the Settings class.

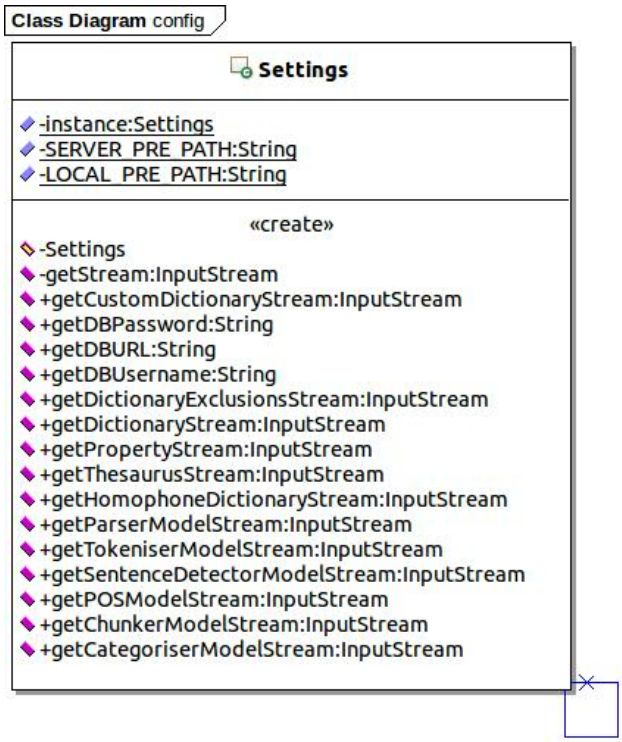

Figure 5.10: Config package class diagram

5.5.2 Core

The core package contains the various classes that are deemed to be at the heart of the system. A class has been devoted to the clue, a solution and a solution pattern providing various specific functionalities.

For example the solution pattern provides methods that allow for words to be computed, without having to work out the length of each word every time. The clue class is able to deduce the best solutions, based upon a collection of solutions.

The SolutionsCollection provides a simple interface to handle any number of solutions. The SolutionsCollection uses the HashSet as it's base class, which prevents duplicate solutions to be stored within the same set. It also provides a faster lookup, in comparison to List based structures.

Finally the manager class is the heart of the system and was originally illustrated in the subsection Solving a Clue on page 89. It will be able distribute a clue and it's solution pattern to the various solvers asynchronously. It also handles the merging of results within the `distributeAndSolve` method.

Figure 5.11 illustrates the core package, containing the various core classes.

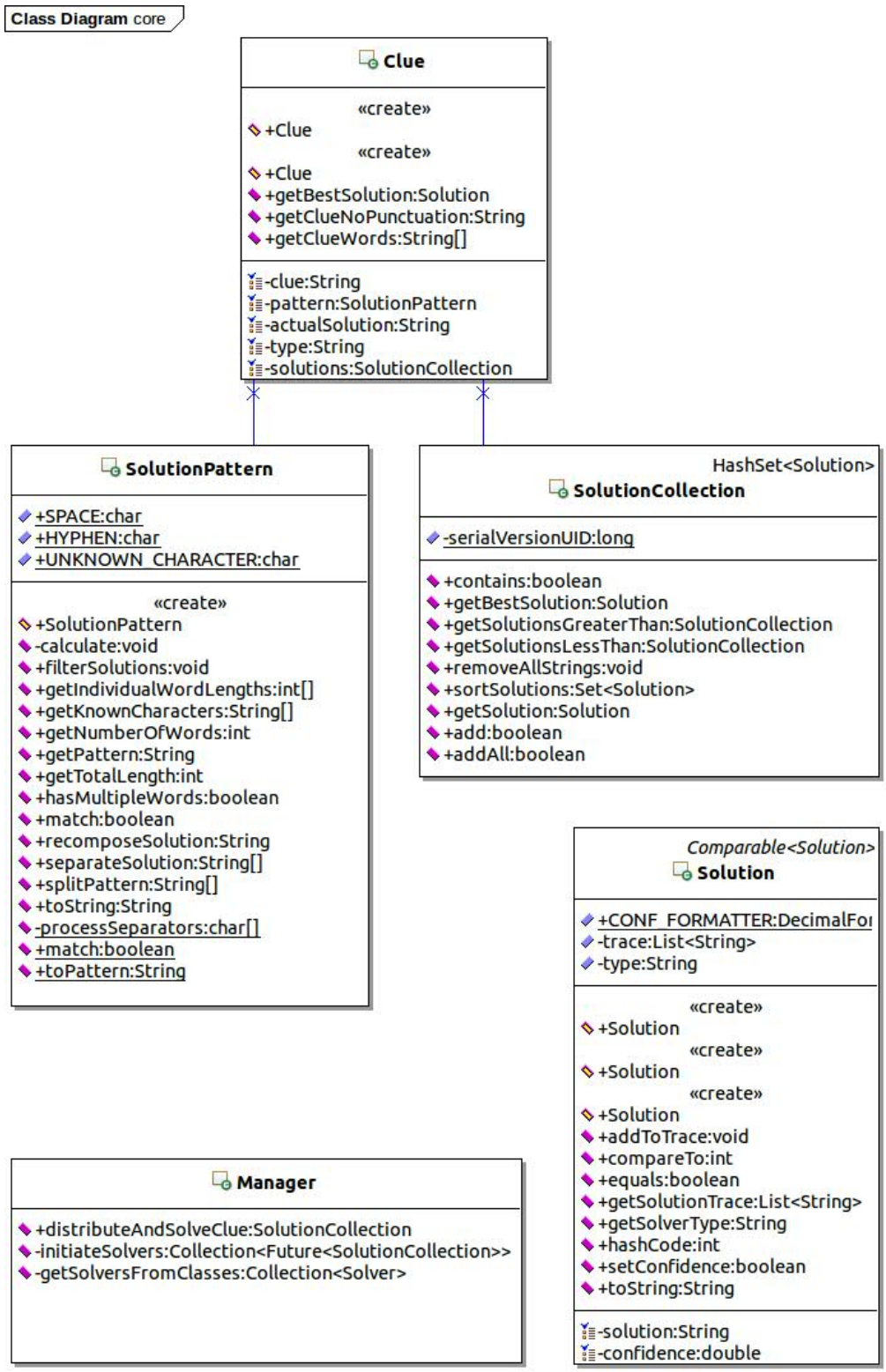

Figure 5.11: Core package class diagram

5.5.3 NLP

The NLP package contains the various classes that focus on providing the system with an interface to the Apache OpenNLP library. This will allow the system to be able to use natural language processing techniques from within the application.

Figure 5.12 illustrates the NLP package, containing the various NLP related classes.

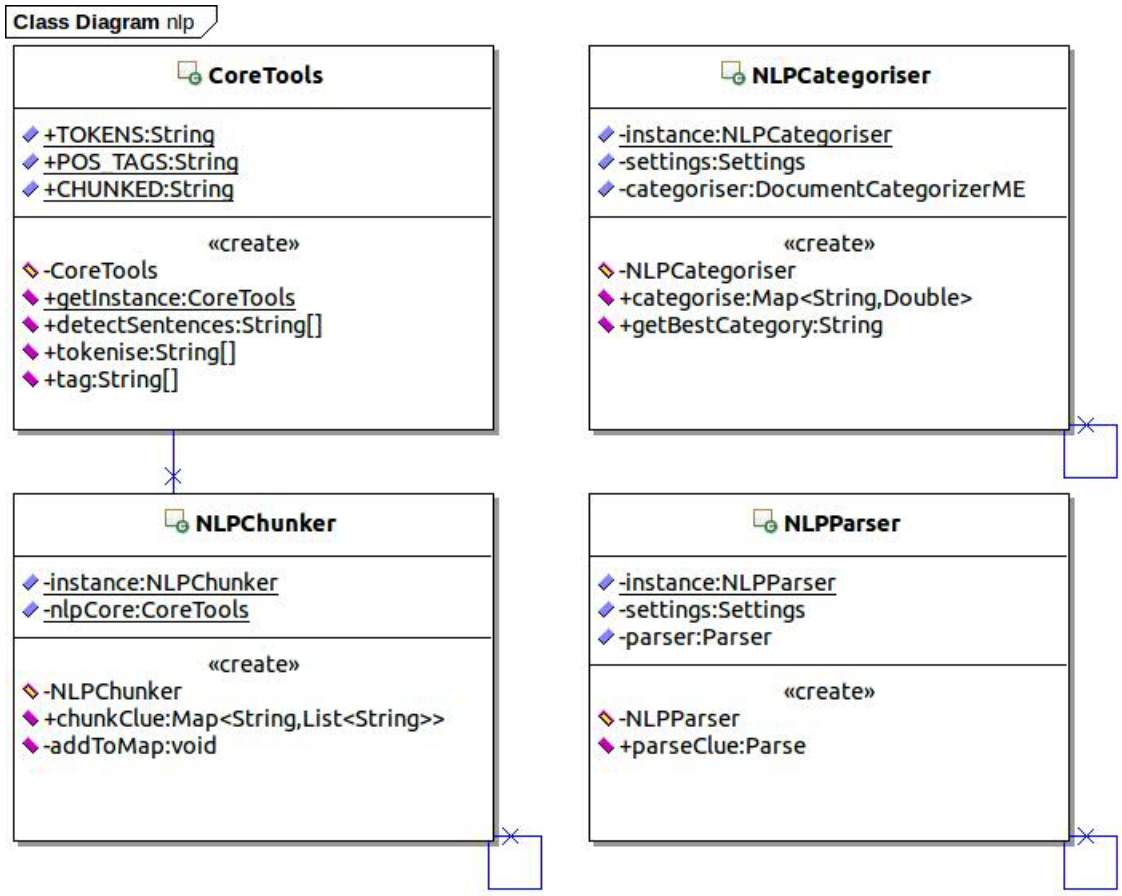

Figure 5.12: NLP package class diagram

5.5.4 Resource

The resource package contains various classes that provide interfaces to the array of resources that the system will use. As with the config package, all classes utilise the Singleton design pattern.

The reason for this is that it is to be expected that these classes will be used multiple times throughout the life of the system. To prevent multiple objects being created and being left

in memory, the singleton design pattern was selected. The pattern will "ensure that a class only has on instance, and provide a global point of access to it" (Gamma, 1995).

The Dictionary, Thesaurus and HomophoneDictionary classes provide the functionality that would be expected. For example the Dictionary has the ability to check to see if a given word is a 'real word'. Where as the Thesaurus class has the ability to find synonyms of a given word.

All of the classes are able to utilise the SolutionPattern class (as found in the core package), which allows for words to be matched upon various known and unknown character combinations.

The Categoriser class will map a given clue and solution combination to various lists of clue type indicators. If an indicator is found, then the confidence rating of the solution will be increased. This will form part of the overall ranking of a solution.

Figure 5.13 illustrates the resource package, containing the various resource classes.

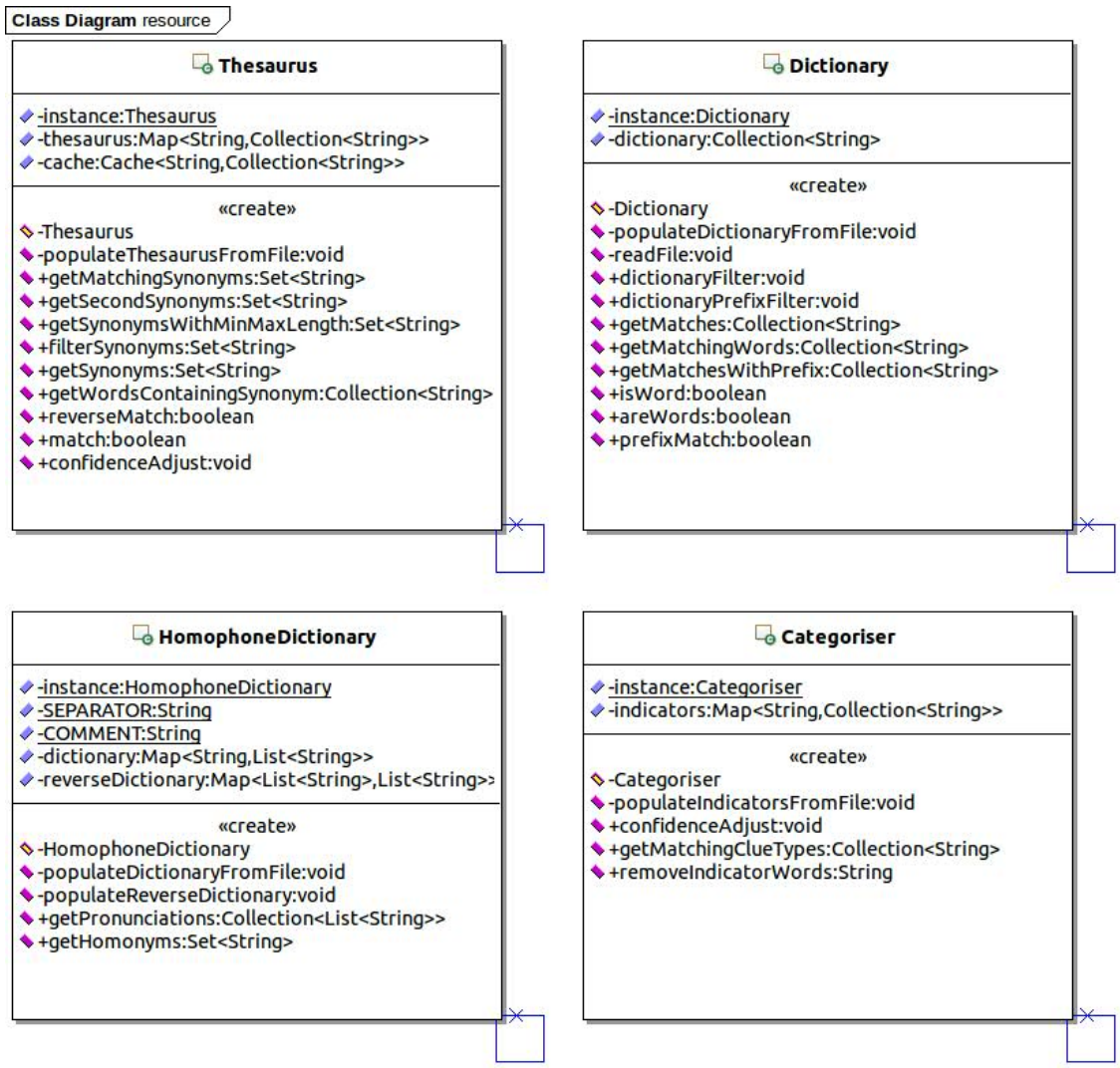

Figure 5.13: Resource package class diagram

5.5.5 Servlet

The servlet package contains two classes that provide the web service interface. Both classes tie into the Tomcat system, which provides a container for the web service to sit in.

The Servlet class is a customised base class that provides various levels of common functionality, that other specific classes will utilise. This functionality includes converting XML into JSON, and determining if a reject was made via AJAX.

The Solver class extends the Servlet class, and provides the functionality required for handling solver related queries. All GET and POST requests to the Solver class made via AJAX will be treated as a request over the web.

However all GET and POST requests that do not pass the AJAX flag, will be treated as if the user intended on viewing a website. All non-AJAX posts will be redirected to the Solver's default HTML page, which will allow users to enter the various parameters via a web form.

It is intended that the solver class will listen upon all requests upon the /solver URL path — for example www.example.com/solver.

Figure 5.14 illustrates the servlet package, containing the public web service related classes.

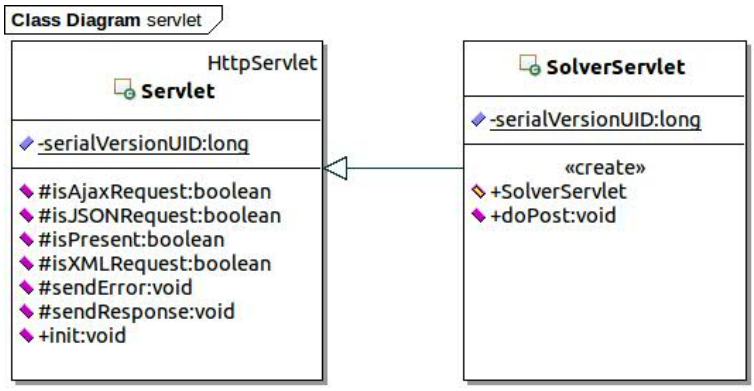

Figure 5.14: Servlet package class diagram

5.5.6 Solver

The solver package contains the all of the solver algorithm classes that are used as part of the over all solving process. Each solver extends a main Solver class which provides a number of base methods, as well as some additional methods that require each solver to override.

The Solver class implements the Callable interface, which provides functionality to run a class in it's own thread. A design decision was taken that the Callable interface was to be used, as once it has finished executing it is able to return an object unlike similar interfaces such as Runnable.

Each of solver classes implement the Abstract Factory design pattern via the base class Solver. This pattern "provides an interface for creating families of related objects without specifying their concrete class" (Gamma, 1995).

Each of the solvers is able to return a concrete product, which in this instance is a a SolutionCollection object. The SolutionCollection will contain all the solutions that the given solver has managed to compute.

Figure 5.15 illustrates the solver package, containing the various solving algorithms used by the system.

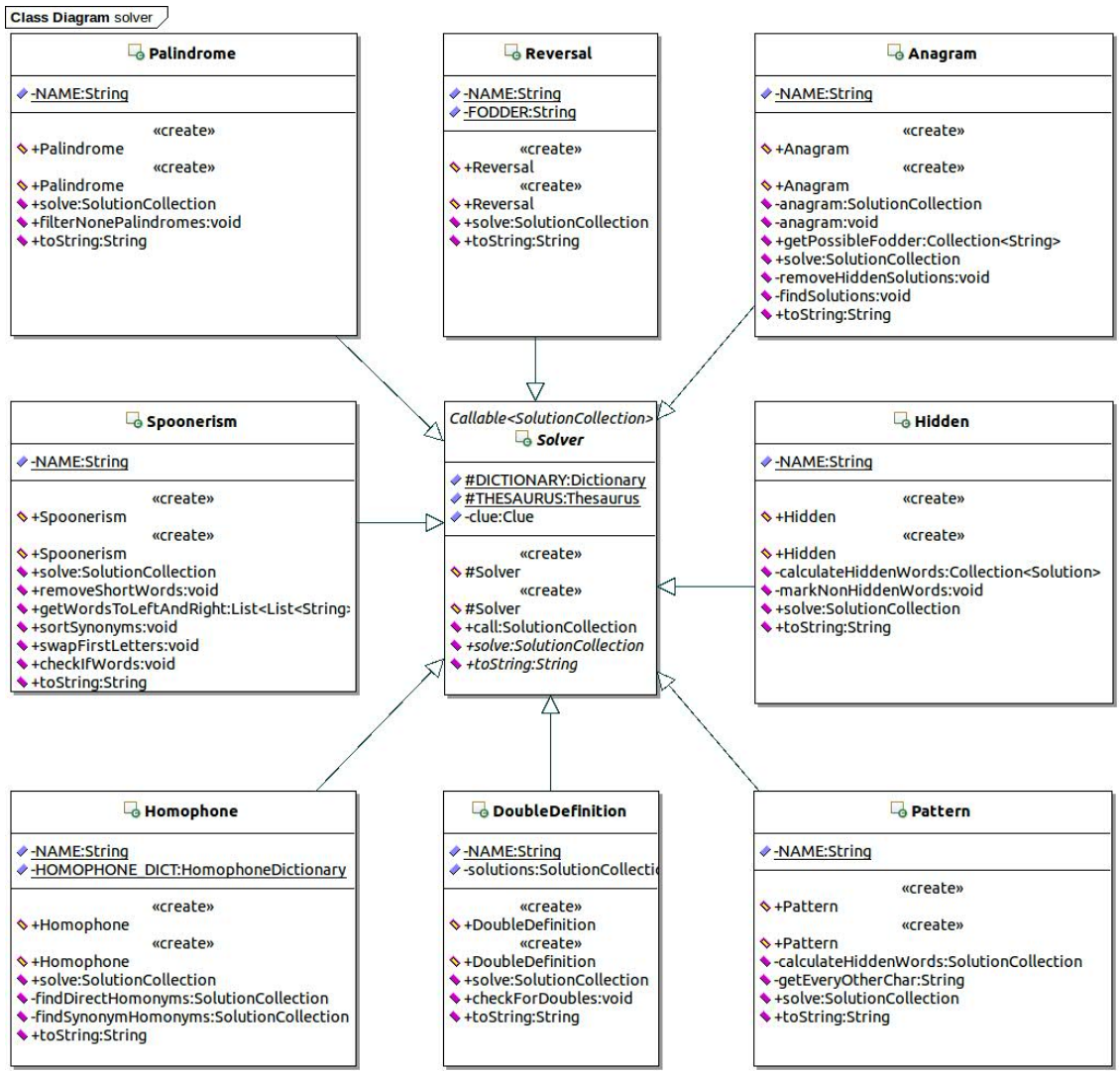

Figure 5.15: Solver package class diagram

5.5.7 Util

The util package contains a number of classes that are unrelated with each other, but provide functionality that is required throughout the system.

The XMLBuilder class provides a data encapsulation layer around an XML library. This class is able to build well-formed XML documents that are suitable for the system's out.

The WordUtils class provides a number of generic methods that are used to manipulate strings as words. This functionality is particular useful, as the system will be dealing with large amounts of natural language (English) sentences.

The confidence class provides a number of rules, that when applied to a solution is able to deduce it's overall 'correctness'. This value is directly reported back to the user as the confidence rating.

Figure 5.16 illustrates the util package, containing the a number of additional utility classes.

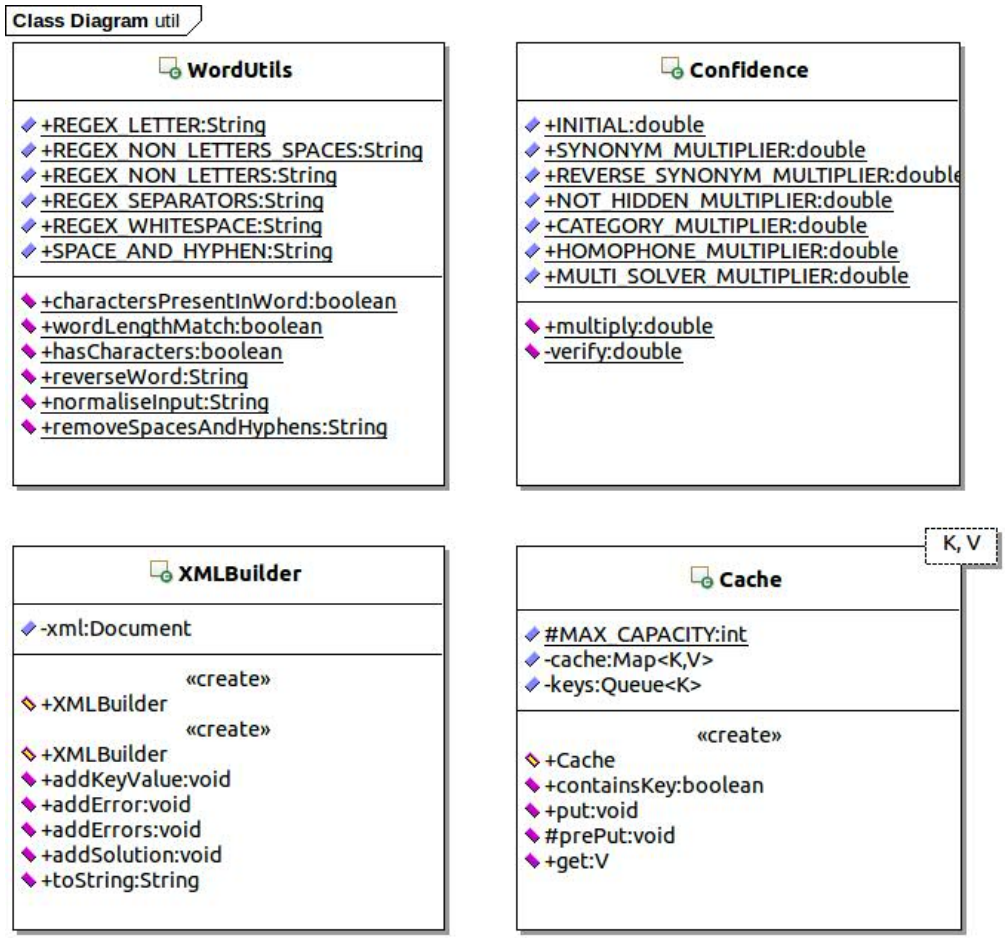

Figure 5.16: Util package class diagram

5.6 Database Design

As part of the testing strategy a list of clues and their solutions are required to be stored. In order to accurately store a large amount of training data, a database will be used.

The database design is of a minimalistic approach as it is not intended to hold a large amount of data, nor will it be used within a production environment. Figure 5.17 shows the proposed database design.

Each record will contain a clue, a solution, and the solutions length. The solutions length has been assigned a VARCHAR data type, because solutions can span across more than one word. For example 3,4 would relate to two words, of three and four characters respectfully. Each of these attributes are required — i.e. not null.

The orientation attribute is an ENUM that contains either 'down' or 'up', indicating the direction that the solution is arranged. The orientation may be useful for certain clue types, such as reversal, where by the clue references the direction — e.g. north-to-south, or east-to-west.

The clue number attribute stores the number of clue, as some clues use the clue number in obtaining the solution answer. Although this is uncommon, it will be taken into consideration.

The clue type attribute is an ENUM that holds the various types of clues, such as 'purely cryptic' or 'anagram'. It is intended that these values will be 'mapped' to the system code, so that clues can be used automatically when training.

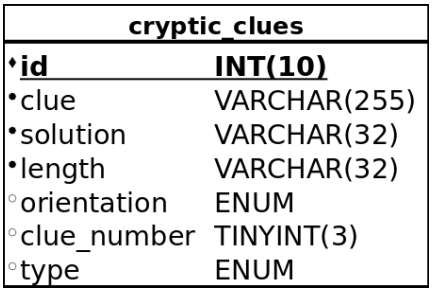

Figure 5.17: Testing database entity-relationship diagram

5.7 User Interface

The previous sections have presented a number of system designs from a programmatic point of view. Within this section, there will be a focus upon user interface designs, and thus how the end user will interact with the system.

Fundamentally there are two aspects to the user interface design of the system — inputting the clue and retrieving the results. Each of these aspects will be discussed in more detail in the following subsections.

5.7.1 Platform Support

One of the main objectives of the project is to develop a system that can be used upon a number of mobile platforms, as well as trying not to neglect 'traditional' desktop users.

In order to complete both of these objectives, a responsive design is required. A responsive design is one that is able to dynamically change based upon the screen size.

This means that there will only be one code based for multiple screen sizes, and thus allows for easier maintainability. All of the designs within this section feature a responsive design. Figure 5.18 illustrates the power of responsive designs.

The left-hand side form shows a 'traditional' desktop experience, whilst the form upon the right-hand side, shows a mobile experience.

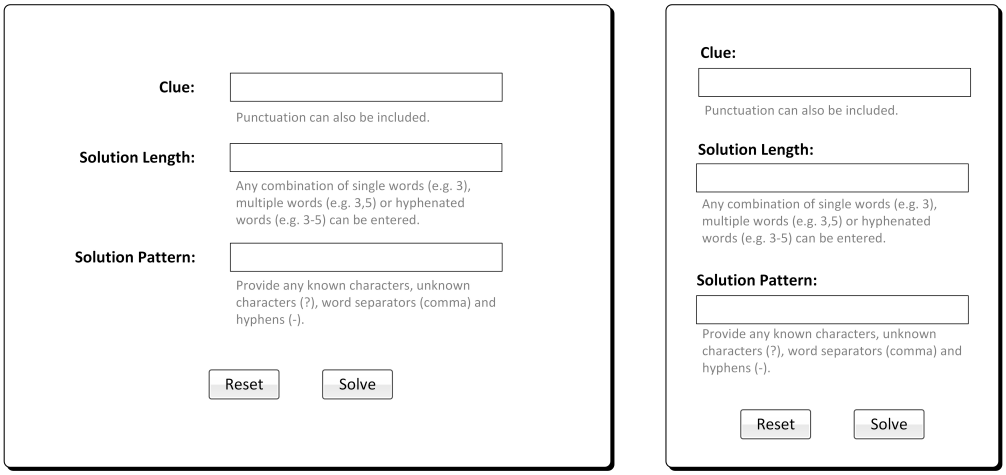

Figure 5.18: The input form to be completed by the end user

5.7.2 User Input

In order for the system to solve the clue, it must first be given the clue, along with additional supporting information. The additional information such as the solution length and solution pattern is vital to the system in order for it to compute the correct answer.

However the intended users of the system are likely to be operating upon some form of mobile device, meaning that a simple and power interface is required. It also means that space will be at a premium, and thus it can not afford to be wasted.

Figure 5.19 illustrates the design of the user input form. One of the most noticeable elements to the form design is how much space has been allocated to the input boxes.

The design of the inputs utilises a bi-column setup using the ratio of 1:3. This means that for every 1 pixel of space allocated to the left-hand labels, there will be 3 pixels of space allocated to the right-hand input boxes. This allows users of mobile devices to quickly select the input box, rather than having to tap a small area several times.

In order to assist the end user additional help blocks will be included, and give useful information to the user, such as describing the solution pattern format required.

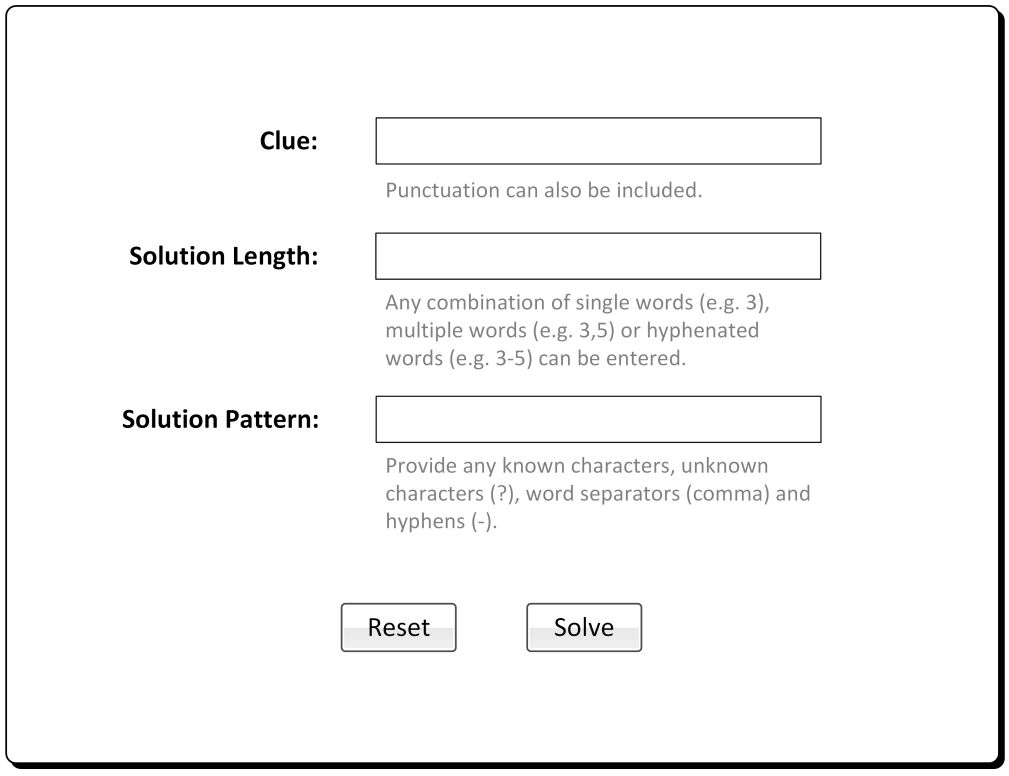

Figure 5.19: The input form to be completed by the end user

As the system is dealing with input from users, it is inevitable that a user will attempt to

input incorrect data. This will require validation to be performed, and if validation has failed then the user should be notified.

As previously mentioned space is a premium upon a mobile device. Therefore displaying a list of validation error messages at the top of the screen would not be utilising limited amount of space wisely.

In order to combat this issue, validation will be displayed 'inline' with the form input elements, as seen within figure 5.20. By utilising an 'inline' approach it recycles the screen space that has been made available to the form.

For inputs that have passed the validations the outline will change to green, and for validations that have failed the input group will change to red. This will immediately alert the user to the various issues, and thus allows the user to fix the errors in a quicker and more efficient manner.

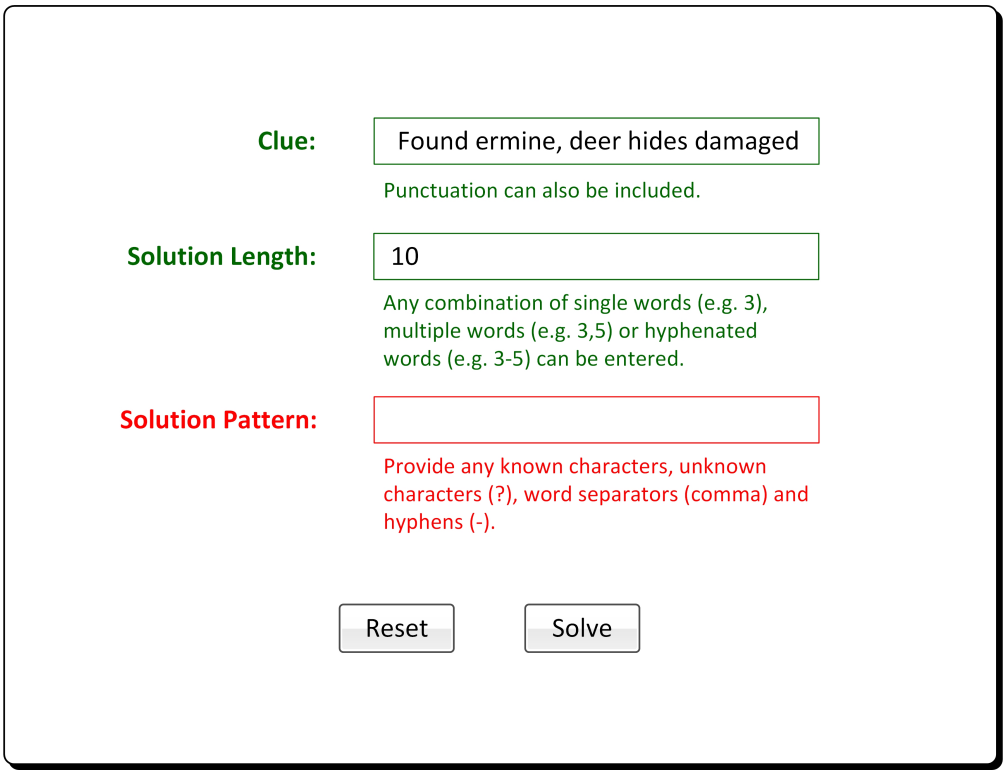

Figure 5.20: The input form indicating a validation error

5.7.3 Results

The displaying of the results follows on from some of the previous design decisions. Each potential solution is rendered into it's own panel, and will contain the confidence rating, and the solver type that managed to deduce the solution (as shown in Figure 5.21).

Within an open panel, additional information is displayed — known as the solution trace. A solution trace provides a step by step account of how the solution was able to be computed. The solution trace may help to teach users how a particular type of clue is solved.

The results are ordered by confidence rating in ascending order, and by default the first panel will be 'open', showing the solution trace. The reason for this is that the top answer is likely to be the answer the end user is looking for

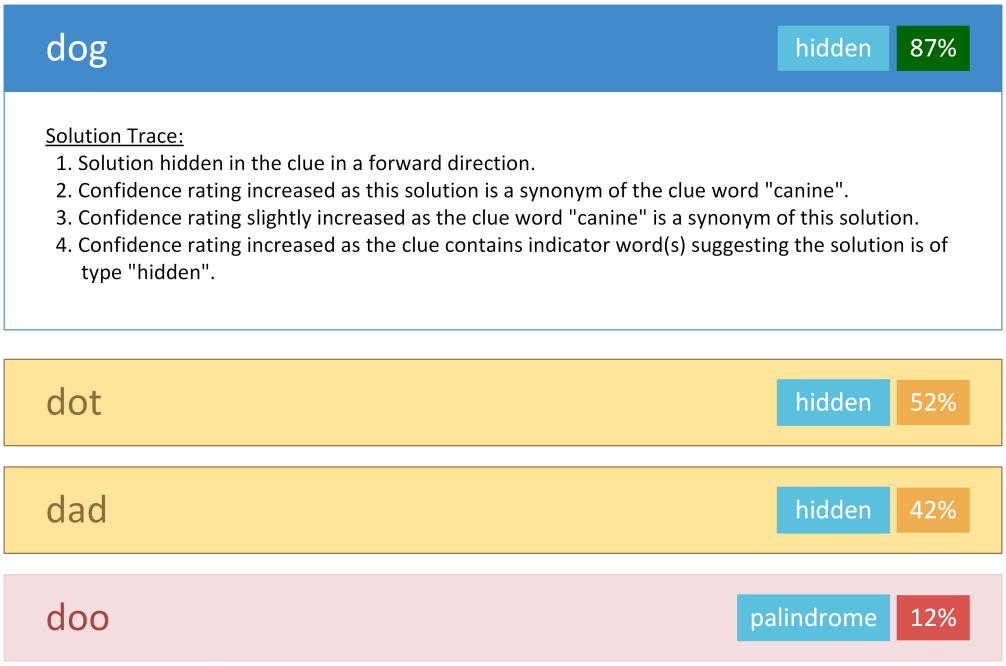

Figure 5.21: Results list displaying the top answer in blue

Each of the panels are "clickable", meaning that if any given panel is selected then the solution trace will be displayed, whilst previously selected panels will be 'closed' (as shown in Figure 5.22). The main reason behind this is to reduce the amount of scrolling a user has to do, especially for those without a mouse.

Each solution is awarded a colour based upon the confidence rating. Blue refers to a top answer, whilst the remaining colours (green, yellow and red) indicate the likely hood of the solution being correct.

This will allow the end user to immediately deduce that 'green' results are more likely to be correct in comparison to 'red results', and that a 'blue' result is likely to be correct.

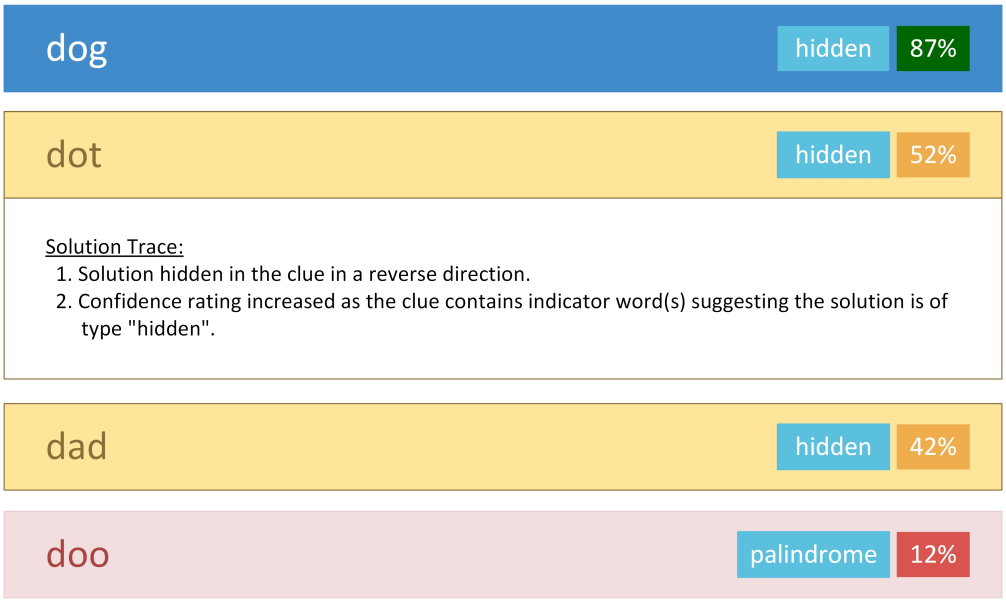

Figure 5.22: Results list displaying alternative solutions

Chapter 6

Development

The development phase was split across three 'iterations'. Each iteration would add and improve upon existing functionality. In order to keep the documentation as compact as possible, this chapter will only focus upon the final development phase. This will ensure that the contents of this chapter is in line with the latest development phase and source code.

This chapter will focus upon the user interface development and how it interacts with the Java Servlets. There will also be significant focus upon the 'back end' system, including how a clue is solved and how the solvers interact with various resources such as dictionaries and thesauri.

Finally the overall system architecture will be discussed, providing an insight into its development and how it has added to the overall product.

6.1 User Interface

In order for users to use the system a simple and powerful user interface was required. The design reasonings behind the user interface were described in the design section.

The user interface has been designed utilising a fall-back system, which is a standard web development approach. The majority of the user interface is delivered by the server utilising JavaServer Pages (JSP) language.

The JSP language essentially extends from XML, and allows for html-like code to be written so that when compiled with Java, a full server-side page is rendered. Within this project an additional JavaServer Pages Standard Tag Library was used. The library – xml – provided additional functionality so that JSP was able to directly utilise XML within its rendering technique.

Figure 6.1 illustrates an example XML parsing snippet from the `solver.jsp` file.

```
<x:choose>
  <x:when select="$solution/trace">
    <p>Solution Trace:</p>
    <ol>
      <x:forEach select="$solution/trace" var="trace">
        <li><x:out select="$trace"/></li>
      </x:forEach>
    </ol>
  </x:when>
  <x:otherwise>
    <p>Solution Trace Unavailable.</p>
  </x:otherwise>
</x:choose>
```

Listing 6.1: isPatternValid deduces if a given solution pattern is valid

The code snippet above shows use of the XML library, as denoted by the '`x`' name space to certain elements. The code is utilising XPATH to find all possible traces that are listed within the trace element (the trace element is an array).

For each of the traces found they will be printed out into the standard HTML output. However if a solution has not been found a simple predefined message will be presented.

The server side rendering that has been described above is known as the fall-back option. This option will work on all browsers and operating systems. The reason for this is that the rendering is controlled by the server, and hence can be fully tested.

Many modern browsers will support JavaScript in some form of fashion. This allows for

various additional functionality to be provided to enhance the user's web browsing experience.

The cryptic crossword website features a JavaScript override that will override the default server-side rendering and provide its own rendering. For example when a user clicks upon the 'solve' button, the website will display a message informing the user that the clue is currently being solved as shown in figure 6.1. Under a non-JavaScript supporting browser this would simply look like a normal page request that is taking its time.

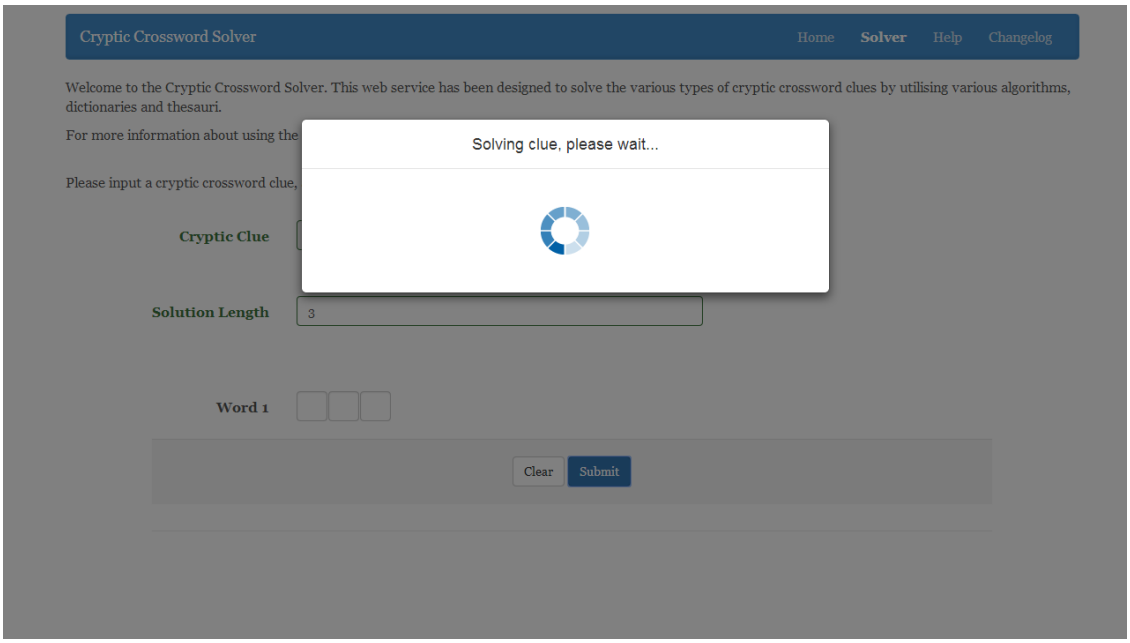

Figure 6.1: Message informing the user that a clue is currently being solved

The JavaScript engine will also provide some basic forms of validation, to prevent the user from waiting a long time simply to find out that there was a validation error. The validation occurs upon a key press, and hence the feedback is instant as shown in figure 6.2.

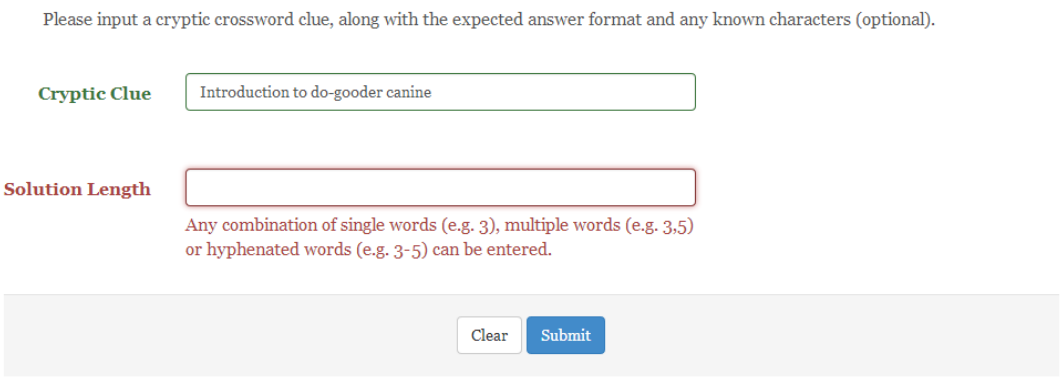

Figure 6.2: Live validation feedback ensures users are entering correct data

The JavaScript engine will also make POST requests to the server. This reduces the total amount of work the server will be required to do, as it only has to return the requests (rather than render HTML).

The reason for the fall-back system that was previously described is that if for some reason the JavaScript engine fails to load, or is incompatible with the device, then the JavaScript will not load. However, the functionality of the site will still continue to work, as the browser will 'fall-back' to the standard server-side validation and rendering.

6.2 Web Service & Servlets

The core system is wrapped around a RESTful web service, that allows users from various devices to submit clues to be solved. Within this section both the web service and the servlet implementations will be discussed and presented.

6.2.1 Web Service

During the project analysis phase the decision was taken that the system's functionality will be delivered via a web service. The web service was developed using Java Enterprise Edition (Java EE) and Tomcat 7.

The main reason for using Java is that the web service could easily make use of the various packages that are provided by the chosen natural language processing library — Apache OpenNLP written in Java.

The web service has solely been produced using the Java EE platform and does not use any additional frameworks or libraries such as Apache Axis. The reason being is that the Java EE platform will run on any machine that is capable of running the Java virtual machine without any additional configuration.

Although Apache Axis (for example) provides additional functionality in configuring the web service, it was decided that the project was to focus upon the solving of a clue. Therefore a 'standard web service' setup would easily meet the requirements of the project.

Another design decision was taken to ensure that the web service followed a RESTful style of communication. The main reason for this is that some of the target devices (i.e. mobile and tablet platforms) do not support SOAP based communication without additional plug-ins. RESTful web services also provide a number of advantages over their SOAP-based counter parts, as was highlighted within the research section.

6.2.2 Servlets

The servlet design has been split across two classes — `Servlet` and `Solver`.

The `Servlet` class extends the standard `HttpServlet` class and provides common functionality. The `Servlet` class provides a base for all system Servlets to use the common functionality.

The `Servlet` class is able to if a given request is from a JSON, XML or Ajax background. For example, if a client was to make a request to the web service through a web browser utilising an Ajax request, then the `isAjaxRequest` method would return `true`.

For illustrative purposes the `isAjaxRequest` method is shown below in listing 6.3.

```
protected boolean isAjaxRequest(HttpServletRequest request) {
  String ajax = request.getHeader("x-requested-with");
  return ajax != null && ajax.toLowerCase().contains("
     xmlhttprequest");
}
```

Listing 6.2: isAjaxMethod deduces if a request was made by AJAX

The servlet also contains two customised methods – one to handle errors, and the other to handle a good response – that are able to send a response back to the requesting client based upon a number of factors.

For example the methods are able to convert the return data into either XML or JSON depending upon what the client has asked for. The `sendError` method will also set the HTTP status code correctly, allowing the client to correctly authenticate the response.

Finally the `Servlet` class overrides the `init` method, which is automatically called as part of the object construction. This method will initialise resources at servlet creation (i.e. first run-time within tomcat) rather than during the first call to the servlet.

In doing this, Tomcat will take more time initially starting up, however the user will notice that their queries are dealt with much quicker. In this project the `init` method has been used to initialise the various in-memory dictionaries and thesauri.

The second class is the `Solver` servlet and handles all requests that are specifically for solving a given clue. The `Solver` servlet accepts both GET and POST requests, with each requiring the clue, the length of the solution and the solution pattern.

The `Solver` servlet upon receiving a request will validate the input parameters, based upon a number of criteria including presence checks and regular expressions. The code snippet below is an example validation rule that will validate the solution pattern against a regular expression.

```
private boolean isPatternValid(String pattern) {
   // Pattern string regular expression
   final String regex = "[0-9A-Za-z?]+((,|-)[0-9A-Za-z?]+)*";
   boolean match = Pattern.matches(regex, pattern);

   // Pattern String must be present and of a valid format
   return isPresent(pattern) && match;
}
```

Listing 6.3: isPatternValid deduces if a given solution pattern is valid

In order for validation to pass, the solution pattern must not be empty and must match to the regular expression stated in the method.

The `Solver` servlet class will initialise the solving of a clue if the three inputs are deemed to be valid. The `solveClue` method will utilise the Clue manager class, that will handle the distributing of the clue to the various solvers.

This has been designed so that the servlet and the solving processes are upon separate threads. This prevents Tomcat from freezing and allows it to handle requests from users.

Once all the solvers have finished executing, the `Solver` servlet will produce an XML document based upon the various elements. Once the XML document has been created, it will be sent back to the client as either XML or JSON.

6.3 Core

Within this section the core package will be presented. The core package is the package that forms the heart of the system, and provides various base classes that when instantiated will represent Clues, Solutions and Solution Patterns.

6.3.1 Clue

The clue class represents an individual cryptic crossword clue, and will maintain a list of possible solutions that have been computed. The Clue class is a basic class, that essentially provides references to other aspects of solving a clue, such as the solution pattern (see subsection 6.3.4).

A clue is able to have a number of solutions, and hence each clue will house a SolutionsCollection, which is described in more detail in subsection 6.3.3.

A clue will also have a solution pattern – described in subsection 6.3.4 – which allows for potential solutions to be matched against an expected pattern.

6.3.2 Solution

The solution class implements the Comparable interface, which is a standard Java interface that imposes ordering upon the object that implements it. It was decided that the solution class should implement the interface, so that solutions can be compared.

The comparing of solutions is perhaps one of the most common pieces of functionality that the system will be required to do. An example use case would be comparing any number of solutions to deduce which is more likely to be the correct answer.

Listing 6.4 shows the implemented compareTo() method found within the Solution class.

```
public int compareTo(Solution o) {
  int solutionCompare = solution.compareTo(o.getSolution());
  int confidenceCompare = -1
      * Double.compare(confidence, o.getConfidence());

  if (solutionCompare == 0) {
    // compare the actual solution text.
    return 0;
  } else if (confidenceCompare == 0) {
    // return a comparison of the solution text.
    return solutionCompare;
  } else {
```

```
      // compare them based on their confidence.
      return confidenceCompare;
    }
  }
```

Listing 6.4: compareTo() compares two solutions

The above code will try to deduce which of the two solutions are closest to being correct. It does this by comparing their confidence ratings, to which every solution will have. If the solution is the same the 0 is returned, whist -1 or +1 returned depending upon which solution is closest to being correct.

As well as housing the confidence rating, the solution class also houses the solution trace. The solution trace is a list of steps that were taken in order to compute the solution. It is intended that the solution traces will help to teach the user how to complete similar clues in the future.

6.3.3 SolutionCollection

The SolutionCollection class extends the standard Java HashSet class utilising the Solution class as the element type. Although a HashSet can not guarantee the order of the set, it can guarantee that no duplicates will be added to the set. This decision was taken to ensure that the system is not dealing with large amounts of repetitive datasets that will only harm the performance of the system as whole.

In order to get around the fact that the ordering of the set has no guarantee, the system makes use of the fact that the element type – Solution – implements the Comparable interface. This means that a copy of the current collection can be retuned in a sorted order if possible. This is illustrated in listing 6.5.

```
  public Set<Solution> sortSolutions() {
    return new TreeSet<>(this);
  }
```

Listing 6.5: Method returns a new sorted collection

As both the TreeSet class and the HashSet both share the same parent class Set, it is possible to cast the result of this method back to a SolutionCollection.

The SolutionCollection class overrides basic methods found within the HashSet class, such as `contains`, `add`, `addAll`, whilst providing additional functionalities, such as returning all solutions whose confidences are greater than (or less than) a given value.

6.3.4 Solution Pattern

The SolutionPattern class models the solution to a corresponding clue. For example if the clue is nine letters long, then the SolutionPattern class would provide information about the pattern of that solution using any given known characters.

The solution pattern is able to split a well-formed pattern and represent it as an in-memory object allowing for a faster clue matching process, in comparison to constantly trying to match a string.

A good example of the level of functionality available in this class is shown in listing 6.6.

```
public void filterSolutions(Set<Solution> solutions) {
  Collection<Solution> toRemove = new ArrayList<>();
  // For each proposed solution
  for (Solution solution : solutions) {
    // If it doesn't match the pattern, throw it out
    if (!match(solution.getSolution())) {
      toRemove.add(solution);
    }
  }
  solutions.removeAll(toRemove);
}
```

Listing 6.6: Method returns collection of matched solutions

The code snippet will match the given set of solutions — often a SolutionCollection — to the current object. A match can simply be described as matching a solution pattern to a possible solution, for example 'd??k' could match to 'duck', 'deck' or even 'dork'.

This method is often used within the filtering down of potential solutions, and thus ensures that the application is not dealing with too much data. In effect this helps the application become more efficient when solving solutions.

Obviously the efficiency of the matching process is directly linked to the number of known characters. If a large number of known characters is given by the user, then the total 'search space' is dramatically reduced.

As an example there are about 308 million different letter combinations within a six letter word (26^6) – this refers to the arrangement of letters, and not 'actual' words. If just one of those letters is known, this would be reduced down to 11 million different letter combinations (26^5), and knowing two letters would reduce it down to around half a million (26^4).

6.3.5 Manager

The Manager class manages the process of solving the given clue. The manager class heavily utilises the standard Java future interface, which is designed to represent the result of asynchronous computation.

Listing 6.7 illustrates the distribution functionality that distributes a copy of all the necessary resources — such as the clue and its solution pattern – and starts the computation upon various new threads (if available upon the system).

Each of the solvers are obtained dynamically via the plug and play system to which more information can be found within section 6.4 on page 118. Once the solvers are obtained they are initialised. The initialisation process simply creates a new object instance of the solvers, and starts them solving the clue.

Once all solvers have finished, their computed solutions are added to an overall solutions collection, which will ignore duplicate solutions as explained in section 6.3.3.

Each of the solutions will have their confidences adjusted based upon how 'correct' the solution is.

```
public SolutionCollection distributeAndSolveClue(Clue clue) {
    // This will hold the solvers to be run at runtime
    Collection<Solver> solvers = getSolversFromClasses(clue);
    // This will hold the returned data from the solvers
    Collection<Future<SolutionCollection>> solutions =
       initiateSolvers(solvers);
    // This will hold all solutions that have been returned
    SolutionCollection allSolutions = new SolutionCollection();
    // Now we need to 'unpack' the SolutionCollections
    for (Future<SolutionCollection> future : solutions) {
      try {
        allSolutions.addAll(future.get());
      } catch (InterruptedException | ExecutionException e) {
        e.printStackTrace();
      }
    }
    // Adjust confidence scores based on category matches
    Categoriser.getInstance().confidenceAdjust(clue,
       allSolutions);

    return allSolutions;
  }
```

Listing 6.7: Method to distrbute the clue out to all solvers

6.4 Plug and Play Architecture

One of the major aspects of the design and development of the system was the underlying architecture. A well designed architecture would not only improve the overall efficiency of the system, it would also aid in reducing development time and improve the maintainability of the system.

It was decided that a 'plug and play' architecture would be adopted. A plug and play architecture allows for external components to be attached to a main system, and thus allows for that system to be able to detect that component and use it (plu, 2006).

The plug and play architecture is a common model found within computing, an example usage would be a network. A computing device is able to connect (either via a wire or wireless) to a network and it will automatically be able to access the network resources without having to reconfigure the computer, the network or other devices connected to that network.

The plug and play architecture would aim to solve the problem of having an unknown number of solvers working within the overall system. By implementing the architecture, any number of solvers could be added and removed to the system at any time (compile time or run time). Thus allowing the system to be as flexible as possible when it comes to handling the solvers. As a bi-product of this, it would also enable any future developers to be able to add their own solver without having to re-factor any core application code.

The architecture uses an 'includes' file which lists all solvers that are intended to be included within the system. The file also supports the ability to ignore files which enables solvers to not be included in the system at run time. Listing 6.8 illustrates an example snippet. Lines that start with '#' are files that are to be ignored — i.e. the class will not be imported at run time. This is illustrated in listing 6.8.

```
uk.ac.hud.cryptic.solver.Acrostic
uk.ac.hud.cryptic.solver.Anagram
#uk.ac.hud.cryptic.solver.Homophone
#uk.ac.hud.cryptic.solver.DoubleDefinition
uk.ac.hud.cryptic.solver.Pattern
```

Listing 6.8: plug and play solver import file

Listing 6.8 illustrates that the Acrostic, Anagram and Pattern solvers will be 'loaded' in to the system's list of available solvers. Whilst the Homophone and DoubleDefinition solvers will not be loaded. It must be stated that currently the system only supports the loading of solvers at the initial start up of the application. Solvers can not be added or removed during the use of the application, but due to the flexibility of the architecture this could be easily added in a future revision.

The loading and parsing of the classes are handled within the manager class found within the core package, which was described in more in detail in section 6.3.5 on page 117.

Each of the required classes added to the properties file are imported by utilising Reflection. Reflection allows for the solver classes to be 'loaded' at run time, by importing the Java class and then instantiating it at run time. Listing 6.9 illustrates the snippet of code that 'imports' and initialises the various solvers.

```
try {
  // Name of the solver read from the propertes file
  String solverName;
  // Load the class
  Class<?> cls = Class.forName(solverName);
  Constructor<?> c = cls.getDeclaredConstructor(Clue.class);
  // Instantiate the given solver class
  Solver solver = (Solver) c.newInstance(clue);
  // Add to the list of solvers
  solvers.add(solver);
} catch (Exception e) {
  // Error handling omitted
}
```

Listing 6.9: run time class import using reflection

Although reflection may provide a number of issues such as security implications and performance issues in this instance it was deemed to be a benefit, as it allows for extensible features (solvers) to be defined at run time, rather than at compile time.

As long as all additional solvers extend the main solver class, then the above reflection practise will work for all future solvers.

6.5 Solvers

A Feasibility Study was created to attempt to predict the difficulty of specific clue types and their regularity in cryptic crosswords. All seventeen clues types were analysed in order to plan which would be implemented in each iteration of the implementation process. The first iteration involved Hidden, Anagram, Acrostic and Pattern as they all had a low difficulty. The next iteration involved Homophones, Palindromes, Double Definition and Spoonerisms to step up the difficulty of the algorithms. Finally, the last solvers to be implemented were Charades, Deletions, Containers and Reversals as they were seen as the most beneficial to implement in the time left for the project.

Therefore, with the additional reason of a time limit, Purely Cryptic and & lit clue types were not implemented due to their high difficulty and Substitutions, Shifting and Exchange clue types were not implemented due to their rarity.

6.5.1 Hidden

When investigating the Hidden clue type, it was found that the answer to the clue would be within the clue itself. For example:

Creamy cheese used in apricot tart.

Answer: RICOTTA

In the clue above the answer 'ricotta' is hidden within the two words 'apricot tart'. The algorithm takes the clue as a whole, without spaces, and then uses the substring method within Java to find all possible hidden words (as the Hidden clue type can also hide words in reverse, the clue is also passed in reverse). It does this by taking the index of the for loop and length of the solution as boundaries.

Listing 6.10 illustrates how all possible solutions are found (without additional solution trace functionality):

```
int index;
for (index = 0; index <= limit; index++) {
    Solution s = new Solution(clue.substring(index, index +
        totalLength), NAME);
    solutions.add(s);
}
```

Listing 6.10: Retrieving all possible solutions using the substring method in Java

For the above clue, where limit is equal to seven because 'ricotta' is seven letters long, the first five iterations will add the following solutions to the list; 'creamyc', 'reamych', 'eamyche',

'amychee', 'mychees'. Eventually, the loop will pick up 'ricotta' and add it to the solution list. From the output gained in the first five iterations it is apparent that a lot of invalid solutions are added to the list, therefore a number of steps are also implemented to eliminate invalid solutions.

Once all possible solutions have been found, they are all checked to make sure they are not words from the original clue. For example, 'apricot' is seven letters long and will have been picked up through the algorithm, however it is not a hidden word and therefore not a valid solution.

Next, the solutions are checked against the pattern provided. This means if the user has input known letters or there are spaces in the end solution but the solution being checked does not match these requirements, these solutions are removed.

Finally, all solutions are checked against the dictionary to determine whether they are valid words. The solutions that are left are then returned to the user.

6.5.2 Anagram

Within a clue in the form of an 'Anagram' the end solution is placed within the clue itself. For example:

A sportsman spinning the tale

Answer: ATHLETE

The answer comes from rearranging the letters in 'the tale' to get athlete.

The algorithm first checks whether it is possible to be an algorithm clue type. This is done by checking the length of the clue is greater than the solution length. The solution should be composed of all the characters from one (or more) words contained within the clue. In other words, the anagram solution cannot be formed from characters that have been picked and chosen across all the words of the clue. The next step creates a list of possible substrings of the entire clue, which match up with the length of the solution.

Once all the substrings have been collected from the clue, for each substring a new thread is created to speed up the algorithm as finding all possible combinations of letters within a clue on one thread is slow. Within each thread, the combination is passed to a recursive method which attempts to build up more combinations which could potentially be the solution. Dictionary filters and pattern filters are handled within the recursive method meaning once all possible solutions have been found, they are returned to the user.

6.5.3 Acrostic

The Acrostic clue type is another of the types which have the end solution placed within the clue itself which can be seen in the following clue:

What's seen at start of any road running one way?

Answer: ARROW

The answer to the above clue can be found by taking the first letter from each word in the phrase 'any road running one way'. The algorithm splits the clue into an array of words to allow each word to be looked at individually. For each word, the first letter is taken using the substring method in Java. The first letter of each word is stored in a string to use the substring method once again to search through possible solutions within the string.

To find the possible solutions from the string, it is done within a loop with the same number of iterations as the length of the end solution. This means the loop will pick up the following possible solutions for the above clue; 'wsaso', 'sasoa', 'asoar', 'soarr', 'oarro', 'arrow'.

As with previous algorithms (e.g. Hidden), this method adds various invalid potential solutions into the list. To eliminate invalid solutions, the potential solutions are run through the dictionary to determine whether they are valid English words. Finally, the potential solutions will be run through the pattern given by the user and the solution itself to ensure the requirements for the end solution match up. Once all this has been completed, the results are returned to the user.

6.5.4 Pattern

The Pattern clue type is one which holds the end solution within the clue itself. For example:

Beasts in tree sinned, we hear - nothing odd there

Answer: REINDEER

From the phrase 'tree sinned, we hear', each odd letter spells 'reindeer'.

The algorithm firstly checks that the pattern can potentially be of type Pattern. This is done by taking the total length of the clue without any punctuation and dividing it by two. By doing this, the maximum length of any solution found by this algorithm is determined. This means if the end solution is longer, it cannot be a clue of this type and therefore the algorithm ends.

The next step is to look for words in even positions. To do this a string of letters to find potential solutions within it needs to be generated. Listing 6.11 illustrates how every other character is found whether it is for even or odd letters:

```
private String getEveryOtherChar(Clue c, boolean even) {
    final String text = c.getClueNoPunctuation(true);
    String newString = "";
    int i;
    for (i = even ? 0 : 1; i < text.length(); i += 2) {
        newString += text.charAt(i);
    }
    return newString;
}
```

Listing 6.11: Retrieving every other character from a string

The method above gets the clue as a string with no punctuation or spaces. It then takes the letter at an even index or an odd index depending on the value of the boolean passed in. For finding letters in even positions, the value will be true and for the example clue above the following string will be generated; 'batitesnewhantigdtee'.

With the string generated, substrings are found within a loop using the length of the end solution as a boundary. This means with the string generated above, the first five iterations within the loop will find the following possible solutions; 'batitesn', atitesne', 'titesnew', 'itesnewh', 'tesnewha'.

When all possible even solutions have been found, the algorithm then repeats the same steps to find all possible odd solutions. The final steps are to filter the solutions on their length and whether they match the known letters, if any, that have been input by the user as well as checking whether the words are valid dictionary words. Once all these steps have been completed, the potential solutions are passed back to the user.

6.5.5 Homophone

A 'Homophone' clue solution is a word which sounds the same as a word in the clue or a synonym of a word in the clue. For example:

Animal said to have connections

Answer: LINKS

The answer comes from an animal called 'lynx' which has the same pronunciation as the solution 'links'.

The algorithm uses a homonym dictionary to retrieve pronunciations for clue types and synonyms. Each word of the clue is taken and firstly run through a method which gets all homonyms of that word. Then, the algorithm retrieves all the synonyms for each clue word and in turn finds all the homonyms of each of the synonyms.

Once all the homonyms have been found the algorithm checks to make sure none of the clue words themselves have been added as potential solutions and removes them if so. They are then checked to determine that they are valid dictionary words and match the pattern given by the user. Finally, all potential solutions are passed back to the user.

6.5.6 Palindrome

The Palindrome clue type gets it's end solution by taking a word from the clue itself and taking a synonym from it. This synonym must be spelt the same whether the word is reversed or in it's normal state. An example of a palindrome is 'noon' because when the word is reversed it still retains it's original spelling. Below is an example Palindrome clue:

Look both ways

Answer: PEEP

As with 'noon', 'peep' can be reversed and still retains it's original spelling.

The algorithm for the Palindrome clue type first takes each word from the clue and places them into a list. Each word is then run through the thesaurus to find their synonyms. Once this has been completed a filtering mechanism is needed to find all synonyms which follow the rules of palindromes.

Listing 6.12 illustrates how all the synonyms that have been found are filtered by the algorithm:

```
private void filterNonePalindromes(Collection<String>
   solutions) {
   for (Iterator<String> it = solutions.iterator(); it.hasNext
      ();) {
       String normal = WordUtils.removeSpacesAndHyphens(it.
          next());
       String reverse = new StringBuilder(normal).reverse().
          toString();

       // If the word isn't "symmetrical" from both sides,
          remove it
       if (!normal.equals(reverse)) {
           it.remove();
       }
   }
}
```

Listing 6.12: Checking to see if a word is a palindrome

The method to remove synonyms that are not palindromes uses an iterator to loop through each one. For each one, spaces and hyphens are removed to make the comparison possible. This step is essential for palindromes such as 'put up' because with the space present the algorithm would not see 'pu tup' as equal. Once all punctuation and spaces have been removed from the synonym, the alogorithm then uses the reverse method within Java to retrieve the reversed synonym. For example, if the word 'cryptic' was passed to the reverse method it would return 'citpyrc'. This means when the normal word 'cryptic' and the reverse word 'citpyrc' are passed to the if statement to check whether they were equal, it would be false and therefore the synonym 'cryptic' would be removed as a potential solution.

Once the synonyms have been filtered, the algorithm then does further filtering to make sure the potential solutions fit the pattern and known letters of the end solution. When the filtering has taken place, the potential solutions are passed to the user.

6.5.7 Double Defintion

The solution for a Double Defintion clue comes from taking two words within the clue that could be the definition of the solution. For example:

Car plant

Answer: LOTUS

A 'lotus' can be either a type of car or a type of plant, hence why the clue is of type Double Definition.

To explain the algorithm for this clue type it is necessary to also explain the functionality that comes with using the thesaurus. The thesaurus used for the project is a text file where each line starts with a word. Each of the following words on that line are synonyms of the first word. From implementing the solvers different ways to access synonyms were found, these different methods provide a range of results which vary from vague to limited synoynms being found. The Double Definition algorithm uses two of these methods, one which takes a clue word and finds it in the thesaurus where it is the first word and adds all the synonyms to a list, this method can be known as getting 'first level synonyms'. The second method does not only look for the clue word as the first word of a row, it looks for the clue word in any position in any row in the thesaurus, therefore returning a wider range of results. This second method can be known as getting 'second level synoynms'.

The algorithm itself first splits the clue into an array so each word can be looked at individually. For each word in the clue, both the methods explained above are used. This means two maps are created, one which contains the clue word along with it's first level synonyms and a second which also contains the clue word but along with it's second level synoynms.

Now the algorithm compares each of these maps to find different words in the clue with the same synonym. This is firstly done by comparing each word and it's synonyms for the first

level synonyms. Next, it is done by comparing first level synonyms with second level synonyms. If it occurs, the synonym that has been found for two words in the clue, the potential solution is added to a list. As the thesaurus methods check for invalid solutions (potential solutions which do not match the length of the end solution etc.), once all synonyms have been matched up and checked, the results can be returned to the user.

Second level synonyms are not compared with second level synonyms. This is because second level synonyms can be extremely vague so the chances of finding a synonym which links to two words in the clue is more likely and less likely to be the correct answer. Another reason is that the likelihood of this algorithm returning results for a clue that is not in fact of Double Definition type is highly more likely if second synonyms for one word are compared with second synoyms for another. Although, the chances of finding the answer for all double defintion clues by comparing two sets of second level synonyms is highly beneficial, the downfalls outweigh the benefits and therefore that is why this design decision was made.

6.5.8 Spoonerism

A 'Spoonerism' clue type involves swapping the first one or two letters from two words to retrieve a new phrase. For example:

In which to immerse the beasts of Spooner's ocean liner

Answer: SHEEP DIP

A synonym for 'ocean' is deep and a synonym for 'liner' is ship. When the first letter from one of these synonyms is swapped for the first two letters from the other synonym, the phrase 'sheep dip' is created.

The first task the algorithm completes involves removing all words within the clue that have less than or a length of two letters. This is because the swapping of two letters would not be possible on a word of two letters meaning it should potentially also speed up the algorithm by removing extra words to complete functionality on. The next step checks to see if the clue contains a word which starts with 'spoon' to determine whether the clue is of this type. This step is necessary as all 'Spoonerism' clues have an indicator like 'Spooner' or 'Spooner's' meaning the algorithm can terminate if it is not found.

It is also common for the words that have their synonyms manipulated are close to the indicator either to the left or the right so the algorithm takes two words to the left of the indicator and two words to the right. Then, the minimum and maximum length of synonyms to find are then calculated if the solution is one word or multiple words.

With the words found around the indicator and the minimum and maximum lengths of synonyms retrieved, synonyms are found for each of the words and put into a list. Next, each synonym is paired up with every other synonym and they are passed in as a pair

to another method which swaps their first letter or letters around. This method does the following possible combinations to find the potential solutions; for ‘deep’ and ‘ship’, the first two letters (‘seep’, ‘dhip’), two letters with one letter (‘sheep’,‘dip’), one letter with two letters (‘sep’,‘dehip’), two letters with two letters (‘shep’,‘deip’).

Each combination is checked against the pattern and the dictionary. If they are valid they are added to a list of possible solutions which are returned to the user.

6.5.9 Charade

The ’Charade’ clue type can take a range of different aspects of a clue to get to the end solution. The example below shows a clue which uses all possible aspects which can be used:

Quiet bird has a sign on a strange occurrence

Answer: PHENOMENON

In the clue above, abbreviations, synonyms and clue words themselves are used (it is also possible for substrings of clue words to be used). To get to the answer; ‘P’ is an abbreviation of ‘quiet’, ‘HEN’ is a synonym of ‘bird’, ‘OMEN’ is a synonym of ‘sign’ and ‘ON’ is a full word taken from the clue itself. All these appended together make up the end solution ‘phenomenon’.

The first step for the algorithm is to take each of the clue words and get synonyms, construct substrings and retrieve abbreviations for each word. It then attempts to reduce all the data retrieved by comparing to the pattern input by the user.

With all the possible abbreviations, substrings and synonyms possible solutions are generated. To do this a set is created first. Charades can be constructed from any sequential components from the clue. For example, if the clue words are ”one two three”, then a solution may be constructed by using components for clue words ”one, two, three”, ”one, three”, ”one, two” or ”two, three”. A method in the algorithm generates these combinations of clue words which will be processed one by one in an attempt to find the solution.

Once these combinations have been found, they are processed one by one to attempt to find solutions. This is done using recursion with a base case of the potential solution being longer or of equal length to the end solution or running out of data to append to the solution. The recursive method takes an abbreviation or a substring or a synonym for a word in the combination passed in and appends another abbreviation, substring or synonym from another word featured in the combination passed in and carries on until it hits the base case. This generates a string which could be a potential solution. Once this has been achieved, possible solutions are checked against the dictionary and if they are valid they are returned to the user.

Retrieving all possible combinations makes the algorithm slow in speed. This means the solver has not been deployed on to the Cryptic Crossword Solver and will only be used for demonstration purposes.

6.5.10 Deletion

The solution for a 'Deletion' clue type comes from taking the first letter, last letter or both from a word. For example:

Dog beheaded bird

Answer: EAGLE

From taking the first letter from the word 'beagle' (a type of dog) the solution 'eagle' is found. 'Beheaded' is the indicator word for the example clue which tells the solver that the first letter must be taken. For the algorithm to determine whether the first letter, last letter or both must be taken from a word, an indicator file was created. The indicator file was split into three lists, one to determine the first letter being taken, another for the last and a final list for both letters to be taken.

The first task the algorithm completes is to read in the indicator file and compare it against the clue to determine which letter or letters need to be removed. Once this has been determined, the algorithm then finds synonym for each of these words using the thesaurus. All the synonyms are then filtered to remove the ones that will be longer than the end solution length when letters are removed.

When all the valid synonyms have been found, the algorithm then uses the substring method within Java to remove certain letters.

Listing 6.13 illustrates how the synonyms have letters removed depending on the position which has been determined by the indicator file:

```
// Remove head/head edge of a word
if (position == Position.HEAD || position == Position.EDGE) {
    solution = solution.substring(1);
}

// Remove tail/tail edge of a word
if (position == Position.TAIL || position == Position.EDGE) {
    solution = solution.substring(0, solution.length() - 1);
}
```

Listing 6.13: Removing letters from a word depending on the indicator

The code above compares the position found with the positions stored in the enum within the class then the letters are removed. If the synonym requires both the first and last letters to be removed, it will fall into both of the if statements and remove both.

After the correct letters have been removed, the algorithm checks if the word is still a valid dicationary word and matches the pattern provided by the user. If they are valid, they are added to a list and finally this list of potential solutions is returned to the user.

6.5.11 Container

The Container clue type involves putting a word inside another word. For example:

In appearance everyone is lacking in depth

Answer: SHALLOW

With the clue above, the synonym 'show' for appearance is found and the synonym for everyone, 'all', is found. Placing all inside show gives the solution 'shallow'.

A clue word itself can be put inside or around the outside of another clue word or synonym of clue word. Another possibility is a synonym can be placed inside or around the outside of another clue word or synonym of a clue word. These possibilities make the need for an indicator of which possibility it is, necessary.

For the Container clue type, an indicator file was created which determined which word went inside another. The indicators were seperated into two lists, one where the indicators mean the word in the left of the clue goes in the word in the right of the clue, and the other where the indicators mean the word in the right of the clue goes in the word to the left of the clue.

Firstly, the algorithm reads in the indicator file and finds the word in the clue which is an indicator. A variable is then set to determine whether the word to the left goes in the word to the right or vice versa.

The algorithm then finds synonyms for each of the words in the clue and adds them to a map. The synonyms are then filtered to make sure they are the correct length for the end solution. Once this has been achieved the synonyms are then matched up. This involves taking two synonyms for two words in the clue and putting one inside the other. This is done within a loop where the position of the word being contained is moved within the word acting as a container. For example, for the clue above, if the synonyms 'all' and 'show' are being matched, the loop will output the following potential solutions; 'sallhow', 'shallow', 'shoallw'. Once these potential solutions have been found, they are passed to the dictionary to make sure they are valid words.

When all the potential solutions have been found, they are then filtered to make sure any known letters input by the user match the potential solutions. Finally, they are all passed

back to the user.

6.5.12 Reversal

The reversal algorithm is the only algorithm to make some use of the natural language processing library (NLP) — Apache OpenNLP. Although the algorithm does not fully use the all capabilities of the NLP it does use it enough to ensure that the algorithms are not being over worked.

The algorithm will attempt to get a list of possible fodders that can be used throughout the solving process. From a initial review it was clear that the majority of the fodders within a reversal clue are a either a singular or plural noun.

Apache OpenNLP has a number of methods that can tokenise a string, meaning that each word in a sentence is aligned to it's type of word. For example if the algorithm was given the clue:

Secure weapons turned over (4)

Answer: SUNG (reversal of guns)

Then the only noun (or fodder) that would be returned is weapons. Although the algorithm can handle multiple fodders it can not handle the rare clues in which the answer can not be deduced from a noun.

Listing 6.14 illustrates the simply way in which the algorithm utilises Apache OpenNLP's tokenising system to be able to pick out all nouns.

```
for (int i = 0; i < tags.length; i++) {
  // Obtain words that are a singular or plural noun
  if (tags[i].equals("NN") || tags[i].equals("NNS")) {
    fodders.add(words[i]);
  }
}
```

Listing 6.14: Deducing all singular or plural nouns within the clue

Once a list of fodders have been computed each of the fodders will have their synonyms computed and reversed. If the newly generated 'word' can be found in the dictionary, and can be matched to the given patter then it is marked as a potential solution.

Listing 6.15 indicates the simplicity of the algorithm and allows for an efficient approach to solving the clue.

```
// Get all synonyms that match the reversed pattern
Set<String> synonyms = THESAURUS.getSecondSynonyms(fodder,
   pattern, true);
```

```
// Reverse all synonyms to try to create another word
for (String synonym : synonyms) {
  // Reverse the synonym
  String reversedWord = WordUtils.reverseWord(synonym);
  // Add as a solution if the reversed word is a real word
  if (DICTIONARY.isWord(reversedWord)) {
      // Add the solution to all possible solutions
      collection.add(new Solution(reversedWord, NAME));
  }
}
```

Listing 6.15: Core reversal algorthim deducing possbile solutions

However it must be stated that the algorithm (as with all other algorithms) does not have any understanding of the words it generates. For example it is unaware that Alpha and Beta are both types of characters. This can cause many issues for the reversal algorithm as it is depended upon additional/external 'knowledge'.

6.6 Resources

There are a number of resources which were needed to aid with solving the various clue types. As a lot of the solvers shared most of the resources a design decision was made to keep the resources in their own package for all solvers to access them as and when they need to.

6.6.1 Abbreviations

Some clue types, for example the 'Charade' clue type, require the knowledge of the different abbreviations that come with certain words. To provide this knowledge a file was found with a list of words and abbreviations in JSON format (JavaScript Object Notation).

Listing 6.16 illustrates the way in which the abbreviations for 'quiet' are displayed in the file:

```
"quiet": [
 "p",
 "pp",
 "sh",
 "mum"
],
```

Listing 6.16: A sample of the abbreviations file

The Abbreviations class reads in all the possible abbreivations from the file at run time and stores them within a map which includes the word to get the abbreviations for and it's abbreviations as a set.

There are two ways in which abbreviations could be needed to solve a clue, one way is to retrieve abbreivations for one word and the other is to retrieve abbreviations for a phrase or a whole clue.

The method to retrieve abbreviations for one single word simply returns the set of abbreviations for a given word (or key) in the map if it exists within the file.

The method to get abbreviations for a phrase or a whole clue gets the abbreviations for as many words as possible in the given clue. For example, "help the medic" will contain seven abbreviations for the word medic. "medal for the medic" will contain four abbreviations for medal and seven for medic. However, the clue "master of ceremonies" will return one abbreviation which matches the entire clue (i.e. "master of ceremonies"). The algorithm is greedy, and will attempt to match the biggest String possible in the given clue. This means it will match all of the String "master of ceremonies" before matching abbreviations for "master".

6.6.2 Dictionary

The dictionary is an essential resource for the solving of clues to programmatically determine whether a string of letters is a valid word. A text file was found with a list of words within it which is read into the Dictionary class when an instance is created.

Listing 6.17 illustrates simply how the words in the dictionary file are displayed:

```
abaci
aback
abacs
abactinal
abactinally
abactor
abactors
abacus
```

Listing 6.17: A sample of the dictionary file

Once the file has been read in, there are a custom list of words to exclude from the dictionary and a list of words that need to be added found from solving particular clues.

One of the methods used regularly is the filtering method which removes any words from a given collection that are not present in the dictionary. This is an effective way to remove words that have being constructed by a solver algorithm which are essentially just an assortment of letters which hold no identified meaning. A similar method is used to filter any prefixes passed in within a collection that are also not held in the dictionary.

When potential solutions have been found by a solver, it is necessary to ensure the algorithm has returned solutions that fit the end solutions pattern. Requirements such as the length input by the user and any known letters input by the user must be adhered to. In the dictionary class there is a method which gets all word matches within the dictionary for a given solution pattern. As with filtering solutions, there is also a method to match up all words in the dictionary that have the same prefix as a prefix passed in.

There are also simple methods solvers can used to identify whether a single word or a phrase is contained within the dictionary simply by checking whether the collection the dictionary file has been read into contains the word or not.

6.6.3 Thesaurus

For clue types which do not have the answer itself nested within it, it is usually necessary to take the clue words and find synonyms for them. This is where the necessity for a thesaurus applies to aid the algorithms in solving clues. A thesaurus file was found which holds a vast

number of entries where each word after the first word in an entry is a synonym of the first word.

Listing 6.18 illustrates how the words in the thesaurus file are displayed with the example word 'dank':

```
dank,boggy,damp,dampish,dewy,fenny,humid,marshy,moist,muggy,
     rainy,roric,roriferous,sticky,swampy,tacky,undried,wet,
     wettish
```

Listing 6.18: A sample of the thesaurus file for the word 'dank'

As with the other resources the file is read in and stored within a collection, this time in the form of a map where the first word is stored within an entry along with it's synonyms.

There are a wide range of different methods that can be used to retrieve a different array of synonyms from vague to specific. Below is a list of functionality that has been written to retrieve synoynms:

- Obtain a list of "synonyms of a word's synonyms" to increase the chances of finding the correct solution. These must match against a supplied pattern.
- Obtain a list of "synonyms of a word's synonyms" to increase the chances of finding the correct solution. These must match a minimum and maximum length passed in for the synonym.
- Retrieve all single word synonyms of a given word with a maximum and minimum length (because some synonyms can be more than one word long)
- Retrieve all synonyms of a given word with no pattern or minimum or maximum length
- Get the synonyms for as many words as possible in the given clue. For example, in the clue "help the medic", look for synonyms of "help the medic", "help the", "the medic", "help", "the", "medic".
- Retrieve all synonyms in the same entry in the thesaurus as a given word. This means to not only look at the first word of an entry for synonyms, look through all entries and return every entry which contains a given word.
- Retrieve all synonyms in the same entry in the thesaurus as a given word which match against the given pattern.
- Check if a given potential solution found by a solver matches as a synonym against any of the words present in the clue.

6.6.4 Homonym Dictionary

The 'Homophone' clue type requires a resource for retrieving homonyms for words. A file was found which lists words with a string representation of the words pronunciation and as with the other resources, the file is read in and stored within a collection holding the word and an list of the pronunciation split into chunks.

An additional collection is formed which is essentially a reverse of the first collection created. This allows pronunciations to be looked up faster. For example, looking up the pronunciation "HH AH0 L OW1" will return "hello". This saves having to iterate the entire homophone map to search for words with the same pronunciation.

Listing 6.19 illustrates how the words in the homonym dictionary file are displayed with the example word 'hello':

```
HELLO  HH AH0 L OW1
```

Listing 6.19: A sample of the homonym dictionary file for the word 'hello'

The class allows the algorithm to retrieve the pronunciation of a given word as well as get words which share the same pronunciation as the supplied word. This only works for words which share the exact same pronunciation.

6.6.5 Categoriser

Some algorithms for certain clue types require indicators to determine how to generate the end solution. For example, the 'Container' clue type requires an indicator to identify which word is the container word and which word is to be contained. For the purpose of providing a confidence score for an end solution, indicators can also be used. As the clue is passed to all solvers, it is possible that two solvers may return the same answer back to the user. By using the indicator file associated with the clue type it is possible to assign a higher rating to a solution from a solver if the clue has an indicator within it from the associated file.

To provide faster access to the indicators for each clue type, the files listing the indicators are read in and stored within a map in the Categoriser class with the name of the clue type and a collection of the indicators.

One use of the Categoriser class is to retrieve the indicators simply by passing in a clue type. Another, is the functionality to remove the indicator from a clue to potentially speed up a solver algorithm by removing extra unnecessary checking on an indicator word.

Listing 6.20 illustrates how the indicator is removed from the clue:

```
public String removeIndicatorWords(String c, String type) {
    // Remove any punctuation
    String clue = WordUtils.normaliseInput(c, false);
```

```
    // The indicator words for the given type have to be
       present
    if (indicators.containsKey(type)) {
        for (String i : indicators.get(type)) {
            if (clue.contains(i)) {
                // Only remove the first match
                clue = clue.replace(i, "");
                break;
            }
        }
    }
    // Remove any double spaces, etc...
    return WordUtils.normaliseInput(clue, false);
}
```

Listing 6.20: Removing the indicator from the clue

The code above uses another class to remove any punctuation from the clue itself, then it checks to see if the clue type has an entry within the map storing all the indicators. If a word in the clue matches an indicator in the indicator file, it is removed and the modified clue is returned.

As mentioned, the Categoriser class can also boost the confidence of a solution if it contains an indicator within the clue that is featured in the indicator file.

Listing 6.21 illustrates how the confidence is boosted using indicators:

```
public void confidenceAdjust(Clue c, SolutionCollection
   solutions) {
    Collection<String> matchingTypes = getMatchingClueTypes(c);
    for (String clueType : matchingTypes) {
        for (Solution s : solutions) {
            if (clueType.equals(s.getSolverType())) {
                double confidence = Confidence.multiply(s.
                   getConfidence(),
                        Confidence.CATEGORY_MULTIPLIER);
                s.setConfidence(confidence);
                s.addToTrace("Confidence rating increased as
                   the clue contains indicator
                       word(s) suggesting the solution is of
                          type \"" +
                       clueType + "\".");
            }
        }
    }
```

```
}
```

Listing 6.21: Boosting the confidence of a solution

The first line of code in the method above calls another method to retrieve a list of clue types that have an indicator in the associated indicator file which matches a word within the clue passed in. Then, all the solutions found from all the solvers for the clue passed in are matched against the clue types that have been found and if the solver the solution has come from matches a clue type found, the confidence is boosted using a method in the Confidence class.

6.7 Utilities

The utilities package (named utils) is a package that contains various classes that cannot be grouped together in their own package, and provide generic helper methods in the aiding the system as a whole.

Within this section two commonly used classes — Confidence and WordUtils — will be discussed in more detail.

6.7.1 Confidence

The confidence class will manage the various functions that are associated with calculating and assigning confidence scores. The calculating of any given confidence score can be easily achieved across multiple solvers by focusing upon the 'common elements' that are found between all solvers. Once the 'common elements' are calculated then specifics ratings can be applied on top.

Each generated solution will start with an initial confidence of 42. The confidence rating will then be increased or deceased based upon certain features that are found within the clue, solution and the solver that managed to generate the solution.

As previously mentioned there are a number of features, and to describe all of them would be laborious, however three example features will be presented below:

- if a solution is a synonym of the clue's definition word then the confidence is increased by 50%
- if a solution has a synonym of the clue's definition word then the confidence is increased by 10%
- if a double-definition with two first-level synonyms is found then the confidence rating is increased by 40%

It is possible for more than one confidence increase to be applied to one solution, however it is not possible for a confidence to be less than 0 or greater than 100.

The confidence class is a 'bear bones' class that simply stores the value of each type of confidence increase, and by how much that increase should be. The confidences adjustments are often made by the various resources as shown in the Resources section.

6.7.2 WordUtils

The WordUtils class is a collection of helper methods relating to the manipulation of words and sentences. The class is a static class, and has been set as static due to the fact that the

WordUtil functionality is used extensively throughout the system.

An example method is shown in listing 6.22. This method will deduce if a word can be 'created' using the supplied string of characters. This method is often used in conjunction with the dictionary and thesaurus classes when trying to find hidden words.

```
public static boolean hasCharacters(String targetWord, String
   characters) {
  // This list will be reduced as characters are consumed
  Collection<Character> remaining = new ArrayList<>();
  for (char c : characters.toCharArray()) {
    remaining.add(c);
  }
  // Loop over each of the target word's characters
  for (char c : targetWord.toCharArray()) {
    // Continue if the char is available in the character pool
    if (!remaining.remove(c)) {
      return false;
    }
  }
  // the target word can be built
  return true;
}
```

Listing 6.22: deduces if a word can be made with a given set of unilearncharacters

6.8 Plug-ins

Section 6.4 on page 118 described the overall system's plug and play architecture and how it has helped the overall project development phase. Within this section an in depth description and small example will be presented to help future developers create a new solver that can be plugged into the system.

All solvers must be placed within the `uk.ac.hud.cryptic.solver` package. This package will be the base package in which the run time class importer will attempt to import the various solver classes.

Once developed, the new solver class must be added (with it's full package name) to the properties file which was described in section 6.4.

Each solver must extend the `Solver` base abstract class. The newly extended class can have various methods but must implement the `solve` and `toString` methods.

The `toString` method should return a String representation of the type of solver that the solver is, for example "container" or "anagram". The `solve` method will be automatically called by the system, and the core algorithm must be placed within this method (or utilise class specific methods).

Listing 6.23 illustrates a basic acrostic solver template.

```
public class MyAcrosticSolver extends Solver {

  public MyAcrosticSolver() {
    super();
  }

  public MyAcrosticSolver(Clue clue) {
    super(clue);
  }

  @Override
  public SolutionCollection solve(Clue c) {
    // Add the algorithm here
  }

  @Override
  public String toString() {
    return "acrostic";
  }

}
```

Listing 6.23: An example user-defined solver

As shown in listing 6.23 the solve method returns a SolutionCollection which is added to an overall collection to deduce the likely correct solution to the clue. This was described within sub-section 6.3.5 on page 117.

Chapter 7

Testing

7.1 Unit Testing

7.2 Walk-through

The previous section focused upon automated unit tests that were able to programmatically deduce the correctness of output. Although unit testing is a major part of the overall testing strategy it is not the only strategy.

Within this section a number of objective based software walk-throughs will be conducted. A software walk-through is an analysis technique that requires all interested parties to ask questions and make comments about possible anomalies, violation of development standards, and other problems (IEEE, 2008).

As described within the development section, the user interface makes use of JavaScript, and hence there will be two walk-throughs, one with JavaScript enabled and another with JavaScript disabled.

Non-JavaScript

As part of the testing process, the user interface's default view will be tested through a series of walk-throughs. These tests are designed to test to ensure that a user is unable to break the user interface.

All of the tests shown in table 7.1 were conducted in a browser that had JavaScript disabled.

Test Number	Test Data	Reason For Test	Expected Outcome	Actual Outcome	Corrective Action Key	Notes
Clue Input						
1	cryptic clue = nil solution length = nil known characters = nil	To ensure that the server rejects incomplete submitted data.	When the page is reloaded, error messages are shown informing the user that required fields have not been given.	As expected.	None.	None.
2	cryptic clue = “We hear twins shave” solution length = nil known characters = nil	To ensure that the server rejects incomplete submitted data.	When the page is reloaded, error messages are shown informing the user that required fields have not been given.	As expected.	None.	None.
3	cryptic clue = nil solution length = “4” known characters = nil	To ensure that the server rejects incomplete submitted data.	When the page is reloaded, three error messages are shown informing that required fields have not been given.	As expected.	None.	None.

4	cryptic clue = “We hear twins shave” solution length = “4” known characters = nil	To ensure that the server rejects incomplete submitted data.	When the page is reloaded, error messages are shown informing the user that required fields have not been given.	As expected.	None.	None.
5	cryptic clue = “We hear twins shave” solution length = “4” known characters = “pr”	To ensure that the server rejects incomplete submitted data.	When the page is reloaded, an error message informing the user that the solution length and known characters do not match.	As expected.	None.	None.
6	cryptic clue = “We hear twins shave” solution length = “4” known characters = “p??r”	To ensure that the server accepts correctly submitted data.	When the page is reloaded, a number of results are displayed.	As expected.	None.	None.
7	cryptic clue = “We hear twins shave” solution length = “4” known characters = “????”	To ensure that the server accepts correctly submitted data.	When the page is reloaded, a number of results are displayed.	As expected.	None.	None.
Solution Output						

8	cryptic clue = “We hear twins shave” solution length = “4” known characters = “????”	To ensure that the correct result is returned upon the submission of a valid form with a solvable clue.	A number of solutions should be returned, with the correct solution – “pair” – holding the highest confidence rating.	As expected.	None.	41 solutions were returned.
9	cryptic clue = “We hear twins shave” solution length = “4” known characters = “p??r”	To ensure that the correct result is returned upon the submission of a valid form with a solvable clue.	A reduced result set in comparison to test #8, however the correct solution – “pair” – should have the highest confidence rating.	As expected.	None.	2 solutions were returned.
10	cryptic clue = “We hear twins shave” solution length = “4” known characters = “????”	To ensure that the all results as shown, with their solution trace’s open, and viewable.	All solution traces are open, and viewable by default.	As expected.	None.	None.

Table 7.1: Non-JavaScript enabled walk-through test results

JavaScript

As described within the development section, the user interface by default will use JavaScript to reduce the amount of input the user is required to do. It also adds to the user experience, by removing the requirement for some pages to be 'reloaded'.

Within this subsection, a number of software walk-throughs will be conducted with JavaScript enabled. This will test the additional layer of functionality that is brought by using JavaScript. Table 7.2 outlines the results of each of the tests.

Test Number	Test Data	Reason For Test	Expected Outcome	Actual Outcome	Corrective Action Key	Notes
Clue Input						
11	cryptic clue = nil solution length = nil known characters = nil	To ensure that the JavaScript prevents submission of an invalid form.	The 'cryptic clue' and the 'solution length' inputs should become red. The 'known characters' input fields should not appear.	As expected.	None.	None.
12	cryptic clue = "We hear twins shave" solution length = nil known characters = nil	To ensure that the JavaScript prevents submission of an invalid form.	The 'cryptic clue' input should become green, the 'solution length' input should become red. The 'known characters' input fields should not appear.	As expected.	None.	None.
13	cryptic clue = nil solution length = "4" known characters = nil	To ensure that the JavaScript prevents submission of an invalid form.	The 'solution length' input should become green, the 'cryptic clue' input should become red. The 'known characters' input fields should not appear.	As expected.	None.	None.

14	cryptic clue = “We hear twins shave” solution length = “4” known characters = nil	To ensure that the JavaScript allows the submission of a valid form.	The ‘cryptic clue’ and the ‘solution length’ inputs should become green, and a “Solving Clue” message dialogue appears.	As expected.	None.	None.
15	cryptic clue = “We hear twins shave” solution length = “4” known characters = “p”, “”, “”, “r”	To ensure that the JavaScript allows the submission of a valid form.	The ‘cryptic clue’ and the ‘solution length’ inputs should become green, and a “Solving Clue” message dialogue appears.	As expected.	None.	None.
Solution Output						
16	cryptic clue = “We hear twins shave” solution length = “4” known characters = nil	To ensure that the correct result is returned upon the submission of a valid form with a solvable clue.	A number of solutions should be returned, with the correct solution – “pair” – holding the highest confidence rating.	As expected.	None.	41 solutions were returned.

17	cryptic clue = “We hear twins shave” solution length = “4” known characters = “p”, “”, “”, “r”	To ensure that the correct result is returned upon the submission of a valid form with a solvable clue.	A reduced result set in comparison to test #14, however the correct solution – “pair” – should have the highest confidence rating.	As expected.	None.	2 solutions were returned.
18	cryptic clue = “We hear twins shave” solution length = “4” known characters = nil	To ensure that the results are paginated to 10 solutions per page.	41 solutions are expected, and therefore there should be 5 pages of results with no more than 10 solutions per page	As expected.	None.	None.
19	cryptic clue = “We hear twins shave” solution length = “4” known characters = nil	To ensure that the top solution’s trace is opened, and viewable by default.	The top solution’s solution’s trace is viewable by default.	As expected.	None.	None.
20	cryptic clue = “We hear twins shave” solution length = “4” known characters = nil	To ensure that other solution traces can be viewed when clicked upon, and any open solution traces are closed.	Clicking upon a second solution opens the associated trace and closes the original trace.	As expected.	None.	None.

Table 7.2: JavaScript enabled walk-through test results

Glossary of Terms

The following section contains a glossary with the meanings of all names, acronyms, and abbreviations used by the stakeholders.

Term/Acronym	**Definition**
The Guardian	A newspaper with a website featuring cryptic crosswords
Blackberry	A mobile phone platform by Blackberry
iOS	A mobile phone platform by Apple
Android	A mobile phone platform by Google
NLP	Natural Language Processing
SRS	Software Requirements Specification
App	Short for application

Bibliography

(2006). *High definition : an A to Z guide to personal technology*. Houghton Mifflin, Boston.

Akman, O. (2013). The smartphone revolution. Retrieved from http://mail2web.com/blog/2011/05/smartphone-revolution-growth-smartphones-exchange-activesync/.

Apache (2013). Apache opennlp developer documentation. Retrieved from http://opennlp.apache.org/documentation/1.5.3/manual/opennlp.html#tools.sentdetect.

Apple (2007). Macworld san francisco 2007 keynote address. Retrieved from https://itunes.apple.com/gb/podcast/apple-keynotes/id275834665.

Apple (2013). App store tops 40 billion downloads with almost half in 2012. Retrieved from http://www.apple.com/pr/library/2013/01/07App-Store-Tops-40-Billion-Downloads-with-Almost-Half-in-2012.html.

Bird, S. (2009). *Natural Language Processing with Python*. O'Reilley Media, Inc.

Businessline (2012). Microsoft to release windows 8 on oct 26. Retrieved from http://search.proquest.com/docview/1321630361?accountid=11526.

Cadle, J., Eva, M., Hindle, K., Paul, D., Rollaston, C., Tudor, D., and Yeates, D. (2010). *Business Analysis*. British Computer Society, second edition.

Cha, B. (2009). Rim store crowned blackberry app world. Retrieved from http://news.cnet.com/8301-17938_105-10188400-1.html?tag=mncol;txt.

Collins, J. (2004). Languages, natural and formal. Retrieved from http://www.uea.ac.uk/~j108/languages.htm.

Connor, A. (2012a). Cryptic crosswords for beginners: palindromes. Retrieved from http://www.theguardian.com/crosswords/crossword-blog/2012/nov/01/cryptic-crosswords-beginners-palindromes.

Connor, A. (2012b). Cryptic crosswords for beginners: spoonerisms. Retrieved from http://www.theguardian.com/crosswords/crossword-blog/2012/mar/01/cryptic-crosswords-beginners-spoonerisms.

Crossword Tools (2013). Clue solver. Retrieved from http://www.crosswordtools.com/cm/.

Cryptic Solver (2013). Solver. Retrieved from http://cryptic-solver.appspot.com/.

Dawson, C. W. (2009). *Projects in Computing and Information Systems: A Student's Guide*. Addison Wesley, second edition.

Dictionaries, O. (2011). *Concise Oxford English Dictionary*. Oxford University Press, twelth edition.

DOSPINESCU, O. and PERCA, M. (2013). Web services in mobile applications. *Informatica Economica*, 17(2):17 – 26.

Elliott, P. G. (2004). *Global Business Information Technology: An Integrated Systems Approach*. Addison Wesley, first edition.

Friedlander, K. and Fine, P. (2009). Expertise in cryptic crossword performance. In *An exploratory survey*, Auckland, New Zealand. International Symposium on Performance Science.

Gamma, E. (1995). *Design patterns: elements of reusable object-oriented software*. Addison-Wesley.

Gershon, G. (2004). Web services. *e-Doc Magazine*.

Gordius (2003). Cryptic crossword no 22,717. Retrieved from http://www.theguardian.com/crosswords/cryptic/22717.

Halpern, J. (2013). Cryptic crossword no 26,091. Retrieved from http://www.theguardian.com/crosswords/cryptic/26091.

IDC (2013). Android and ios combine for 91.1% of the worldwide smartphone os market in 4q12 and 87.6% for the year, according to idc. Retrieved from http://www.idc.com/getdoc.jsp?containerId=prUS23946013.

IEEE (2008). IEEE Standard for Software Reviews and Audits. *IEEE STD 1028-2008*, pages 1–52.

ISTQB Exam Certification (2013). What is rad model- advantages, disadvantages and when to use it? Retrieved from http://istqbexamcertification.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/.

Jiang, T., Li, M., Ravikumar, B., and Regan, K. W. (2010). *Algorithms and theory of computation handbook*. Chapman & Hall.

Jones, C. (2013). Idc: Windows phone is the second most popular mobile platform in latin america. Retrieved from http://www.forbes.com/sites/chuckjones/2013/09/30/iphones-market-share-down-prior-to-5s-5c-launch-with-windows-almost-double-digits-in-europe/.

Kalin, M. (2013). *Java Web Services: Up and Running*. O'Reilly Media, second edition.

Knott, R. and Dawson, R. (1999). *Software Project Management*. Group D Publications Ltd.

Litman, D. (2003). Cs 1573: Artificial intelligence application development (spring 2003). Retrieved from `http://people.cs.pitt.edu/~litman/courses/cs1573s03/lec/chunking.ppt`.

Lunn, K. (2003). *Software development with UML*. Palgrave, first edition.

McCluskey, T. L. (1999). Cha 2555 - artificial intelligence. Retrieved from `http://helios.hud.ac.uk/scomtlm/cha2555/lecture13_18.ppt`.

McManus, J. (2003). *Risk Management in Software Development Projects (Computer Weekly Professional)*. Routledge.

NLTK Project (2012). Nltk 2.0 documentation. Retrieved from `http://nltk.org/data.html`.

One Across (2013). Crossword puzzle help. Retrieved from `http://www.oneacross.com/`.

Robertson, J. and Robertson, S. (2013). *Mastering the requirements process*. Addison-Wesley, third edition.

Sabri, S. (2013). Idc: Windows phone is the second most popular mobile platform in latin america. Retrieved from `http://www.wpcentral.com/windows-phone-second-most-popular-mobile-platform-latin-america`.

Stoeller, W. (2003). What is risk management? *Globalization Insider*, XII(3.7).

Sullivan, T. (2001). Web services. *InfoWorld*, 23(11):38.

Upadhyay, S. (2008a). Acrostics. Retrieved from `http://www.crosswordunclued.com/2008/09/first-and-last-letters.html`.

Upadhyay, S. (2008b). Charades. Retrieved from `http://www.crosswordunclued.com/2008/11/charades.html`.

Upadhyay, S. (2008c). Cryptic definitions. Retrieved from `http://www.crosswordunclued.com/2008/12/cryptic-definitions.html`.

Upadhyay, S. (2008d). Decoding double definitions. Retrieved from `http://www.crosswordunclued.com/2008/10/decoding-double-definitions.html`.

Upadhyay, S. (2008e). Digging out hidden words. Retrieved from `http://www.crosswordunclued.com/2008/08/digging-out-hidden-words.html`.

Upadhyay, S. (2008f). How to spot anagrams. Retrieved from `http://www.crosswordunclued.com/2008/08/how-to-spot-anagram.html`.

Upadhyay, S. (2008g). &lit, literally so. Retrieved from http://www.crosswordunclued.com/2008/08/and-literally-so.html.

Upadhyay, S. (2008h). Reversals. Retrieved from http://www.crosswordunclued.com/2008/11/reversals.html.

Upadhyay, S. (2008i). Substitutions simplified. Retrieved from http://www.crosswordunclued.com/2008/12/substitutions.html.

Upadhyay, S. (2008j). Tune in to homophones. Retrieved from http://www.crosswordunclued.com/2008/10/homophones.html.

Upadhyay, S. (2009a). Containers. Retrieved from http://www.crosswordunclued.com/2009/02/containers.html.

Upadhyay, S. (2009b). Deletions. Retrieved from http://www.crosswordunclued.com/2009/03/deletions.html.

Upadhyay, S. (2009c). Letter exchange. Retrieved from http://www.crosswordunclued.com/2009/12/letter-exchange.html.

Upadhyay, S. (2009d). Letter picking. Retrieved from http://www.crosswordunclued.com/2009/04/letter-sequences.html.

Upadhyay, S. (2009e). Letter shifting. Retrieved from http://www.crosswordunclued.com/2009/12/letter-shifting.html.