University of Huddersfield

M.Eng Group Project

Cryptic Crossword Solver

Authors:
Mohammad Rahman
Leanne Butcher
Stuart Leader
Luke Hackett

Supervisor:
Dr. Gary Allen

Examiner:
Dr. Sotirios Batsakis

Friday, 26^{th} April 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

1 Problem Analysis **4**
- 1.1 Defining the Environment . . . 4
- 1.2 Defining the Problem . . . 4
- 1.3 Research Areas . . . 5
  - 1.3.1 Initial Survey . . . 5
  - 1.3.2 Cryptic Crosswords . . . 5
  - 1.3.3 Natural Language Processing . . . 6
  - 1.3.4 Application Platform . . . 7

2 Research **8**
- 2.1 Web Services . . . 9
  - 2.1.1 What are Web Services? . . . 10
  - 2.1.2 Web Service Categories . . . 11
  - 2.1.3 Clients . . . 13

3 Development Methodologies **16**
- 3.1 Waterfall . . . 17
  - 3.1.1 Advantages . . . 17
  - 3.1.2 Disadvantages . . . 17
- 3.2 Spiral . . . 18
  - 3.2.1 Advantages . . . 18
  - 3.2.2 Disadvantages . . . 18
- 3.3 Agile . . . 19
  - 3.3.1 Advantages . . . 19
  - 3.3.2 Disadvantages . . . 19
- 3.4 Rapid Application Development . . . 20
  - 3.4.1 Advantages . . . 20
  - 3.4.2 Disadvantages . . . 20
- 3.5 Summary . . . 21

List of Figures

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 Defining the Environment

Cryptic crosswords are a popular type of puzzles found in many parts of the world. Most commonwealth 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.

1.2 Defining the Problem

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.

By using some form of natural language processing and one or more cryptic algorithms, it should be possible to generate an answer to a given clue.

Once a clue has been correctly "guessed" it can simply be returned to be user. It is the "guessing" of the answer that this project will primarily focus upon.

1.3 Research Areas

1.3.1 Initial Survey

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.

1.3.2 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 the cryptic crosswords are the same style.

Each clue is categorised as being either 'across' or 'down' with its corresponding grid number. Each clue will also contain 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 Guardians 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 looked at 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 come to 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.

1.3.3 Natural Language Processing

In order to correctly solve a clue, some form of natural language processing will be required. It is the natural language processing that will 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 its 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%.

1.3.4 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. As many people own a smartphone and/or a 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 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 (Distributed Computing Environment) (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 to 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 upon a private network, such as a companys internal Intranet.

2.1.1 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 programmed in the same language. Many of the 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 last service, or use a completely new language (Kalin, 2013).

2.1.2 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 e.g. 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 2.1 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 2.1: HTTP verbs mapped to the associated CRUD operation.

There are additional verbs, such as HEAD, TRACE, CONNECT, OPTIONS and INFO. Some of the additional verbs 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 the same/similar specification. The real difference between the two technologies is that SOAP utilises it's own application protocol by extending current protocols — namely HTTP.

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 PUT 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).

2.1.3 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.

Bibliography

James Cadle, Malcolm Eva, Keith Hindle, Debra Paul, Craig Rollaston, Dot Tudor, and Donald Yeates. *Business Analysis*. British Computer Society, second edition, 2010.

Crossword Tools. Clue solver, October 2013. URL `http://www.crosswordtools.com/cm/`.

Cryptic Solver. Solver, October 2013. URL `http://cryptic-solver.appspot.com/`.

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

Octavian DOSPINESCU and Marian PERCA. Web services in mobile applications. *Informatica Economica*, 17(2):17 – 26, 2013. ISSN 14531305.

Prof Geoffrey Elliott. *Global Business Information Technology: An Integrated Systems Approach*. Addison Wesley, first edition, 2004.

Gary Gershon. Web services. *e-Doc Magazine*, 2004.

ISTQB Exam Certification. What is rad model- advantages, disadvantages and when to use it?, October 2013. URL `http://istqbexamcertification.com/what-is-rad-model-advantages-disadvantages-and-when-to-use-it/`.

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

R.P. Knott and R.J. Dawson. *Software Project Management*. Group D Publications Ltd, 1999.

One Across. Crossword puzzle help, October 2013. URL `http://www.oneacross.com/`.

Tom Sullivan. Web services. *InfoWorld*, 23(11):38, 2001. ISSN 01996649.