Building the Argument Player

Rory Duthie
Honours Project
BSc (Hons) Applied Computing
University of Dundee, 2014
Supervisor: Prof. C. Reed

1. Introduction

Argument and debate are the cornerstones of civilised society and intellectual life (Bex et al, 2013). As use of the internet and social networks increases, so does the availability of areas to argue and debate. Argument and debate are no longer constricted by the need of face to face interaction and have therefore expanded to the internet. However this expansion into social networks and the internet has brought to light poor quality and poorly structured debate.

The argument web seeks to improve the quality of debate through the use of software tools like the argument player. There are ways to view analysis of debate, these are only static diagrams. By building the argument player, debate can be analysed using a temporal structure showing how the structure of the argument changed over time. An argument can be played back, paused or fast forward allowing exploration of how contributors spoke and how this changed the overall structure of the debate.

This report will focus on the argument player and how it was implemented to compliment the vision of the Argument Web.

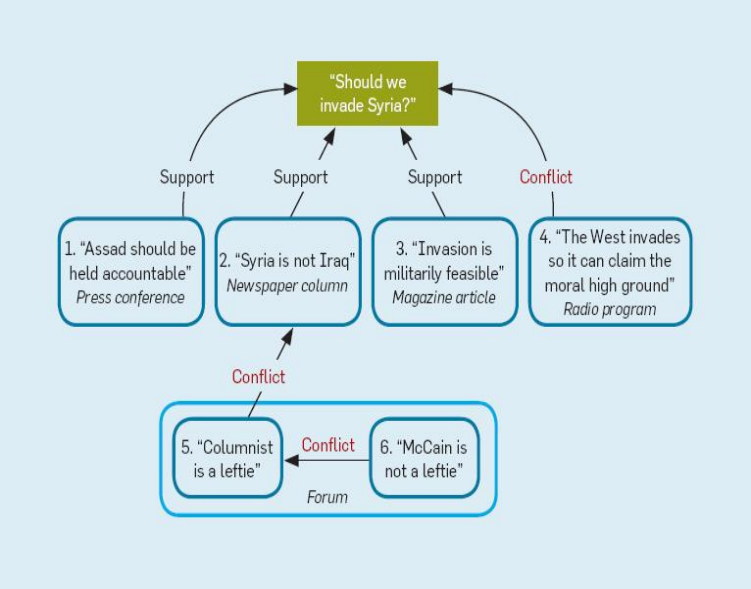

Figure 1: Linked Argument Data (Bex et al, 2013)

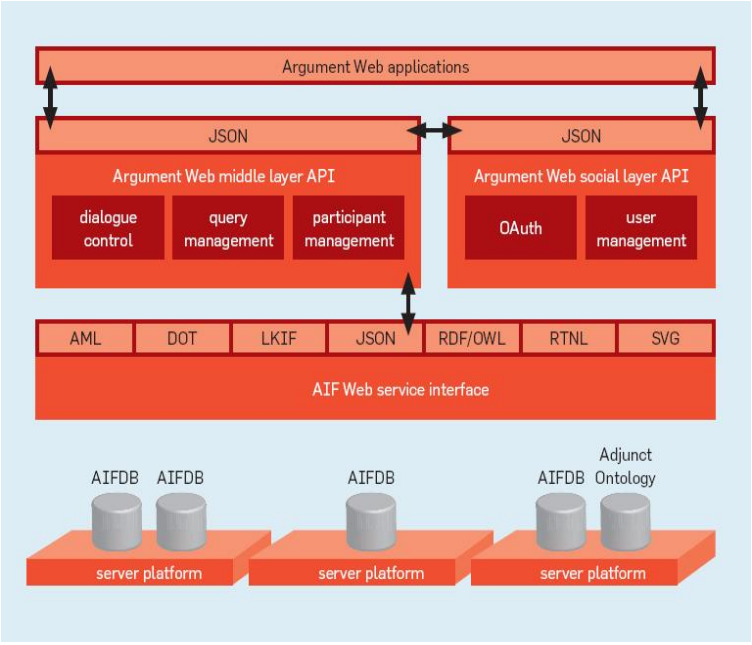

Figure 2: Argument Web Infrastructure (Bex et al, 2013)

2. Background

2.1. What is the Argument Web?

The Argument Web is an infrastructure that allows the storage and automatic retrieval of analysed, linked argument data life (Bex et al, 2013). The Argument Web is formed through the use of different argumentation tools and interfaces, such as AIFdb and the visual interface for AIFdb. These tools allow data from around the internet to be linked. Data can be taken from an argument based in a newspaper forum and linked with an argument or debate in a social network. An example of linked data can be seen in figure 1.

Linked Data in its simplest form is typed links between data from different sources (Bizer et al, 2009). Linked data can be produced by Argument Web applications and then stored in AIFdb to provide an infrastructure for the data. This allows meaningful links to be made between data.

The Argument Web is based on the Argument Interchange Format (AIF) which distinguishes between information and reasoning that is applied to individual links between data (Bex et al, 2013). The AIF allows a structure to be put in place for the Argument Web which can be seen in Figure 2. At each level of this strucutre the fundamentals of the Argument Web can be seen. At the highest level are the Argument Web applications this is where data is linked between different websites. This data is then passed to the AIF Web Service layer and finally passed to AIFdb. This overall structure allowed the formation of the Argument Web and has helped to promote the main aim of the Argument Web, which is to improve the quality of online argument and debate through the use of online tools.

2.2. What is the AIF?

The AIF (Argument Interchange Format) distinguishes between information and reasoning that is applied to individual links between data (Bex et al, 2013). The main aim of the AIF is to produce a concensus on the tools, concepts and technologies used in the field of argumentation. By creating this it will allow data to be linked from different web sources and allow the production of the Argument Web.

"The AIF project aims to develop a commonly agreed-upon core ontology that specifies the basic concepts used to express arguments and their mutual relations." (Bex et al, 2012). In other words the AIF project aims to create a set of standards, agreed upon by researchers, for mapping argument and debate. The mapping should show the relationships between statements made in an argument, in a clear and concise way.

Part of the AIF ontology is to split statements made in an argument and reltaionships between them into different catagories. A statement made in an argument is an I-node. An I-node cannot be connected to another I-node under the AIF ontology. An I-node can only be connected to a justification of why two I-nodes should be related. Therefore an I-node can connect to a RA-node or a CA-node. An RA-node is used to show a supporting relationship and a CA-node is used to show a conflicting relationship. Figure 3 shows part of the earlier argument visualised in Figure 1, but in the Argument Interchange Format.

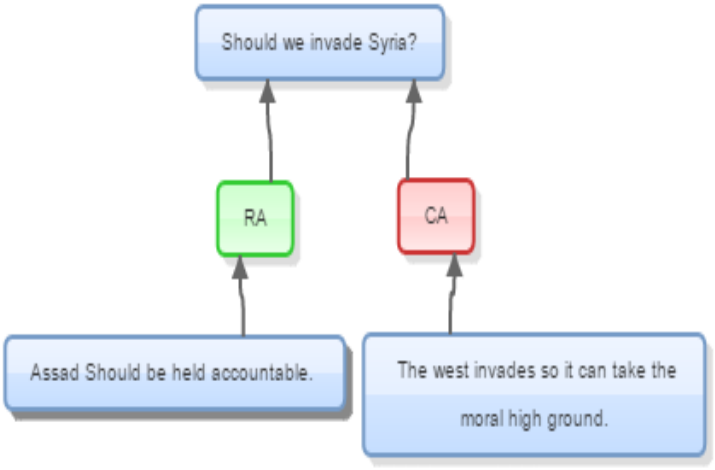

Figure 3: Argument in AIF.

In Figure 3 the statements (I-nodes) are connected by relationships for or against the initial statement. As seen in Figure 3 an RA-node shows support of the initial statement and a CA-node shows a conflict with the initial statement. By creating this argument mapping format, it allows there to be a common way to map arguments. Thus allowing data from different sources to be linked and also allowing argumentation tools to be created such as AIFdb.

2.3. What is AIFdb?

AIFdb is the main infrastructure in the creation of the Argument Web. AIFdb uses the main fundamentals of the AIF. Data is formed in the standards agreed in the AIF and can then be visualised in the same way. Linked Data can also be stored in the database in this format allwoing the data to be analysed and downloaded for use.

AIFdb also makes use of a visual interface allowing users to search for and choose different argument mappings. These mappings are displayed visually by AIFdb and can be seen in Figure 4. The mappings present in AIFdb are from the Argument Web (Bex et al, 2013) and the use of the visualisation interface in AIFdb can allow users to interact with the Argument Web directly.

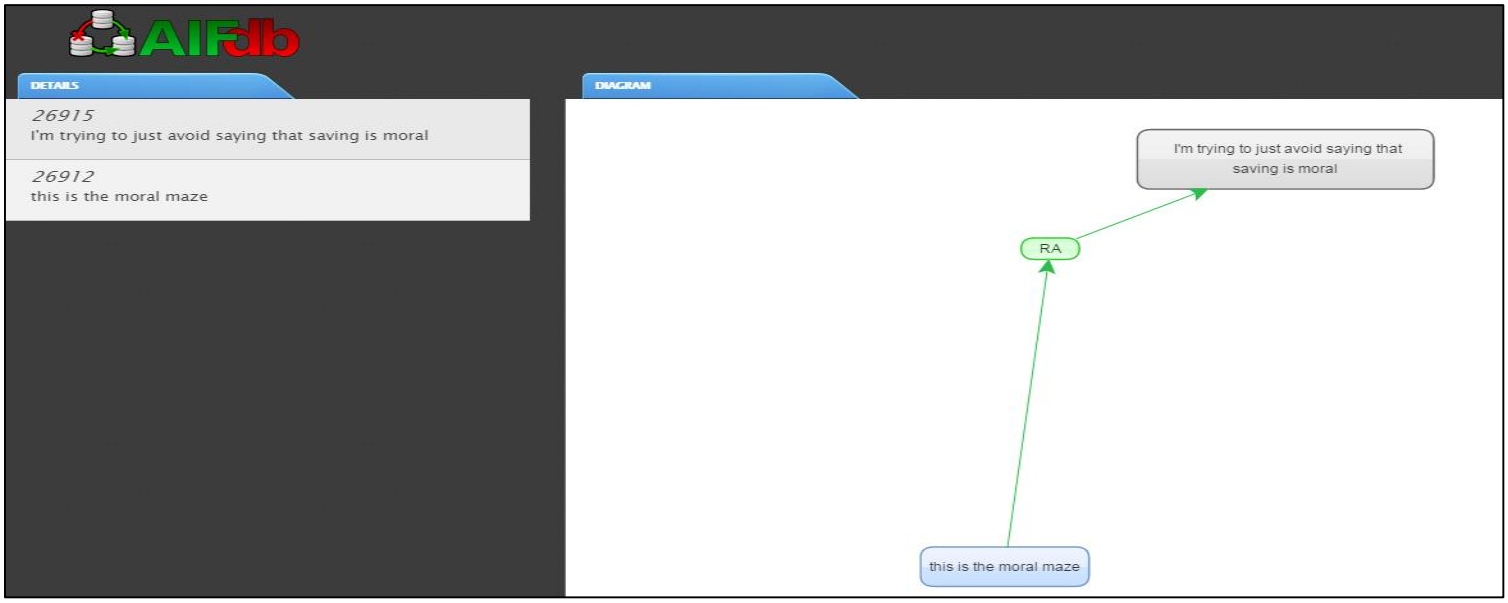

Figure 4: AIFdb Argument Visualisation. (Bex et al, 2013)

2.4. What is Argument Visualisation?

Argument Visualisation is the presentation of reasoning in which the evidential relatonships among claims are made wholly explicit using graphical or other non-verbal techniques (Van Gelder, 2003). To reiterate this Argument Visualisation is to show how a debate or argument is formed in a visual format. This visual format shows the responses to claims made by a person in an argument. In order to portray the argument, it is normally converted from a transcript into a box strucutre. The main conclusion of the argument is outlined at the top of a visualisation with edges appearing from this. The edges can either be statements for or against the main conclusion.

The Argument visualisation allows evidential structures to be created and allow relationships between arguments to be analysed more easily. This first became apparent in 1913 when John Henry Wigmore proposed a Chart Method for analysing a mass of evidence in a legal case (Buckingham Shum, 2003). This case involved the diagramming of evidence to help an analyst reach a conclusion on the case and is one of the first instances of Argument Visualisation.

The argument structure seen today is closer to that of the strucutre created by Stephen Toulmin. Toulmin uses a graphical argument strucutre to portray the relationships between arguments in a simple format which can be read like a story. This structure can be seen in Figure 5 and is the most widely used strucure for Computer Supported Visualisation. However this structure can become over-populated and in large argument maps can become untidy. With the formation of the AIF taking into account this structure and using other formats this can lead to improved Computer Supported Argument Visualisation.

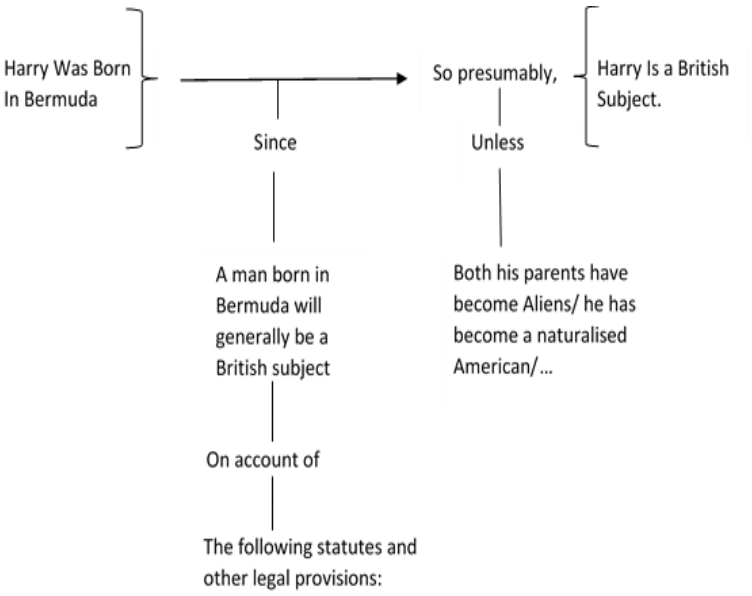

Figure 5: Toulmin's graphical argument strucutre. (Buckingham Shum, 2003)

2.5. What are temporal structures?

A temporal structure is a structure which is influenced by time. As time passes structures change and this is especially prominent in argumentation.

As an argument or debate passes, the shape of argument maps change. Currently within argumentation an argument or debate is analysed and the corresponding argument map is produced for this. The argument map can then be looked over along with audio to try and re-enact the argument again. This uses a temporal structure; however it is not the most intuitive way to do so. Statements made during the debate can be missed when searching through a large visualisation. Currently within the argumentation field this is the only way to view an argument in a temporal way.

By building the Argument Player arguments can be visualised with a temporal structure to allow re-enactment of an argument or debate to as near real time as possible.

3. Preliminary Design Decisions

3.1. Programming Language

There were a number of choices available when deciding on a programming language for the ARG Player. A number of new technologies have been created which allow data visualisation to be implemented more easily. These are: Processing, HTML5 and Google charts. The language used must allow for maintainability of the ARG Player but also allow data visualisation which is easy for a user to make sense of. The programming language used must also allow for audio input to allow the debates to play alongside the argumentation visualisation.

Processing is a relatively new programming language when compared with the more popular languages such as Java. Processing was created in 2001 and is a programming language made for data visualisation (Processing, 2014a). Processing makes use of "Sketches", a blank canvas where anything can be drawn through the use of processing's draw function. Processing also has multiple libraries and can be easily integrated into a website (Sherwood, 2014). The HTML in the website can interact with the processing "sketch" allowing users to have a dynamic experience when using the website. However as processing is a newer, less popular language there are not as many online resources for users when looking for help. Processing's own website does however have a large amount of documentation (Processing, 2014b).

HTML5 is the new version of HTML, which has added extra functionality to make the life of web browsers easier. HTML5 was initially created in 2004 but is still in creation and may not take its final shape until 2022 (Fiedler, 2013). A new feature of HTML5 is the offline capabilities allowing the use of HTML5 in smartphone applications. HTML5 allows the use of a canvas to create visualisations again allowing dynamic content for users. There is a problem with HTML5's compatibility with browsers. Many browsers have yet to shift to the new HTML5 standards and therefore not all content can be processed by the browser.

Google charts is a Google product allowing visualisation of data, on a web base. There are a number of different data visualisation options, word trees being the most useful for the ARG Player. Word trees allow a tree structure of words to be created in a visual form, either using a decision tree or a sequence tree (Google, 2014). The ARG Player will process a large amount of text; this is something which Google word trees can do quickly. The downside to Word Trees is that it is still currently undergoing beta testing and therefore there is very little documentation and help available to developers.

Processing will be used to create the ARG Player due to the ease of creating the visual transitions that need to be displayed when showing how an argument or debate has changed. This is easily done with processing's draw function. The finished ARG Player can then be embedded in a web page where users can interact with it. HTML5 will not be used due to the complex nature, when compared to Processing, needed to create visual images. Processing and HTML5 both offer ways to add audio and can both be used or be deployed to applications. Google word trees will also not be used due to the nature in which it displays the words added to the tree. The ARG Player will need to display a large amount of text. Google word trees are suited to processing this text but not suited in displaying large sentences as it displays them in a linear layout.

3.2. Software Development Methodology

There are a number of software design methodologies available to choose from when embarking on a software project three of these are: the Waterfall approach, the Agile method and an Iterative software design. Each have their own advantages and disadvantages and are suited for different projects.

The waterfall approach (Figure 6 below) is perfect for projects which must have a defined start and finish and need to have a full project plan in place to keep to this rigid structure. The Agile method comes under many different names, its main basis however is on a more flexible project experience. User stories are more flexible and can be changed at any time within the project and are completed in short periods known as sprints. Finally an Iterative approach is a mixture of both Agile and Waterfall. The project has a more rigid project plan with a set of requirements. However the full design process is completed a number of times meaning that requirements can be altered and user feedback sought at different stages of the project.

The waterfall approach is the traditional approach to software engineering. It has a number of strengths and weaknesses. Some of the strengths are that it allows rigid project plans to be made for a project, meaning that there is a defined start and end. This allows money to be budgeted for a project as the end is a known date. A waterfall approach uses a lot of documentation; this means that the software process has clear goals to work towards.

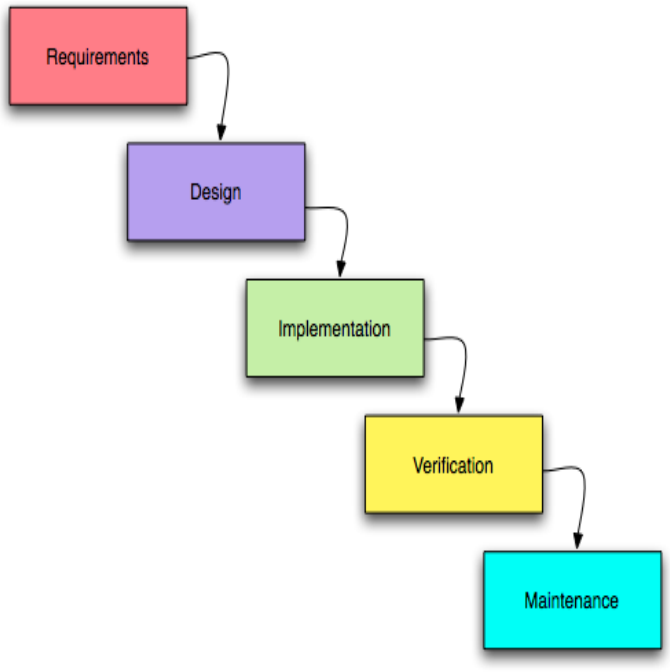

Figure 6: Waterfall Approach (Hughey, 2009)

Some drawbacks to the waterfall approach are that any changes that need to be made within the project can be problematic due to its rigid structure. If a requirement were to change then all the documentation from this requirement would have to change with it. Any implementation would have to change and any tests would also have to change. This leads to large delays in the project.

The agile methodology (Figure 7 below) employs a flexible model, all user stories created in the product backlog must be flexible as they can be changed at any point. There is also a need for constant feedback with users and the customer. This can be easily done with Agile due to the short sprints of development that are undertaken. Once a sprint is complete prototypes can be examined and feedback given to a development team. Daily meetings are also used to talk about the progress of the sprint and to communicate any problems. Post sprint review meetings are also used to discuss how the previous sprint went. Any suggestions for improvement can be made in this meeting.

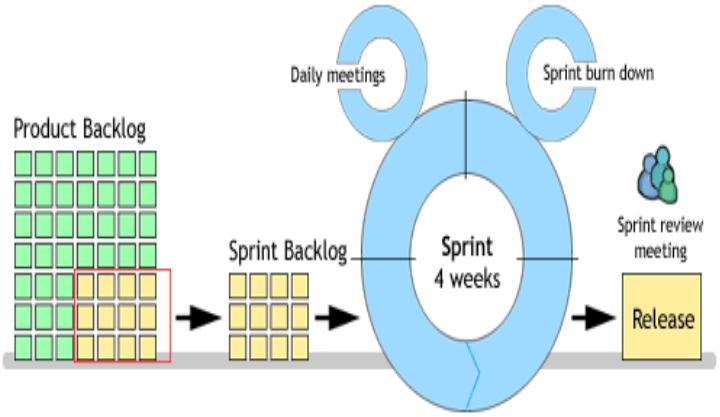

Figure 7: Agile Methodology (Agile Tools, 2013)

A weakness with the Agile approach is that the project has a whole can be hard to predict. There is no defined end date and therefore projects could run well over any predictions made. Another issue is that users and customers need to be involved in nearly every stage of development this is not always achievable and can then lead to delays in the project.

The iterative approach deploys some of the actions of both methodologies. It has the same rigid plan time as the waterfall approach and the same steps but it deploys the Agile methods flexibility in that every stage of development is tested and then commented upon by users and customers. The documentation such as requirements and project plans are created at the start like the Waterfall approach but cycles of the development are planned within this, allowing changes to be made to development.

An Iterative approach (Figure 8) can also be very costly however. This arises when there are large

issues in development. There is also no overlap within cycles this means that in each cycle there must be periods in the plan to allow for issues.

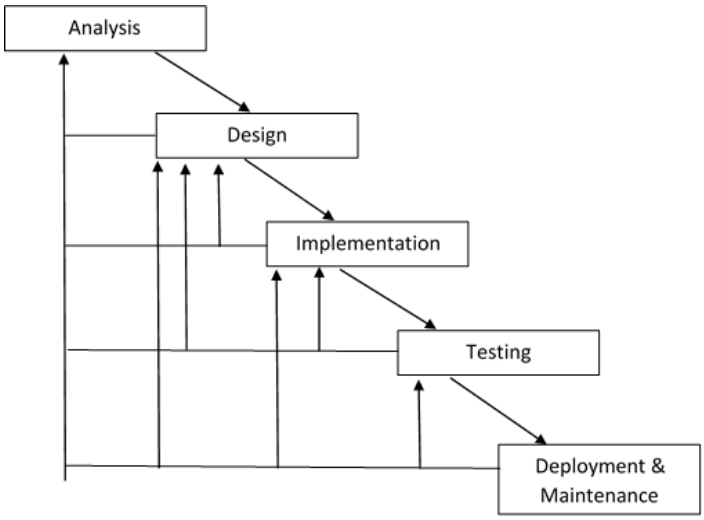

Figure 8: Iterative Waterfall Cycle (Dey, 2014)

As building the Argument Player is a large scale project that has a defined set of requirements the Iterative approach will be used. This will allow the main outlines of the project to be set and any smaller requirements can be merged and changed as the project moves forward. Having defined cycles will allow for more user feedback and will allow the project to seamlessly connect with a user centred design process. Thus allowing the project to be defined by users and help with the goals of the users. The waterfall approach would be too rigid for this project as feedback from users could mean slight changes in requirements which the waterfall approach wouldn't allow for. The Agile approach would be hard to achieve due to how hard it can be to predict the final project dates and final software release dates.

4. Requirements

The requirements created in the requirements document used themes based on the popular BBC IPlayer (BBC, 2014). The IPlayer contains many functions that would be desirable to have in the Argument Player such as a slider to move an argument visualisation backwards and forwards, volume altering controls and pause and play features. Inspiration for further features comes from general music and video players such as Windows Media Player and VLC Media Player. Each of these also gives fast forward and rewind buttons which may be desirable within the Argument Player. However the need for fast forward and rewind buttons can be debated. By having a dedicated slider on the Argument Player like the BBC IPlayer uses it renders the fast forward and rewind buttons useless. To have fast forward and rewind buttons it would have to alter the User Experience of the Argument Player for it to be worthwhile. By implementing both buttons it can make the system easier to use as both of these buttons are well known features which have featured on physical devices like the VCR. Therefore they are familiar to users and can make them feel comfortable using the Argument Player because of this enhancing their overall experience of the system.

Below are featured some of the highest priority requirements:

R1 Data Import

The system shall import all node sets from AIFdb for a chosen program at one time.

Rationale: The data for all node sets is stored on AIFdb and will be imported at one time in order to keep the processing time and loading time of the system down.

Risk: High

Priority: High

R2 Node Layout

The system shall place nodes in a hierarchical structure.

Rationale: Nodes will be displayed in a hierarchical structure in order for all nodes to be clearly legible and so the nodes have clear relationships.

Risk: High

Priority: High

R3 Node Edges

The system shall show edges between nodes.

Rationale: Node edges will be displayed between nodes to show the node relationships.

Risk: High

Priority: High

R4 Data Layout Format
The system shall display the nodes in the AIF format.

Rationale: Nodes will be displayed in AIF format in order to show clear relationship meanings between nodes.

Risk: High

Priority: High

R6 Location Import
The system shall import each node location in .dot format.

Rationale: Node location will be imported in .dot format in order to allow for accurate layout of nodes.

Risk: High

Priority: High

R7 Data Import
The system shall import each node set in JSON format.

Rationale: Data import will be in JSON format to allow for easy import and layout of nodes.

Risk: High

Priority: High

The above requirements are a small feature of the whole requirements document which can be viewed in Appendix A.

The requirements featured above are the main requirements to construct a partially functional system. All have a high priority because of this and all have a high risk because of this. Requirements 6 and 7 in particular pull data from a database over the internet and therefore have a high risk due to this as well. All of the above requirements are functional requirements of the Argument Player. A sample form the document of a non-functional requirement can be viewed below:

R29 Internet Connection
The system shall have an internet connection.

Rationale: The system will have an internet connection so that data can be downloaded from AIFdb.

Risk: High

Priority: High

The above non-functional requirement states that the user must have an internet connection. Again this has been given high priority and high risk as it is an integral part of the Argument Player as without an internet connection the argument player would not be accessible.

5. Iteration 1

5.1. Design

5.2. Implementation

5.3. Testing & Evaluation

6. Iteration 2

6.1. Design

6.2. Implementation

6.3. Testing & Evaluation

7. Conclusion

References

Bex, F., Lawrence, J., Snaith, M., & Reed, C. (2013). *Implementing the Argument Web. Communications of the ACM*. Page 66.

Bizer, C., Heath, T., and Berners-Lee, T. (2009). *Linked data: The Story So Far. International Journal on Semantic Web and Information Systems*. Page 1–22.

Bex, F., Gordom, T., Lawrence, J. & Reed, C. (2012). *Interchanging arguments between Carneades and AIF - Theory and Practice in Proceedings of the 4th International Conference on Computational Models of Argument (COMMA 2012)*, IOS Press, Vienna.

Van Gelder, T. (2003). *Enhancing Deliberation Through Computer Supported Argument Visualisation*. Page 98 – 100.

Buckingham Shum, S. (2003) *The Roots of Computer Supported Argument Visualization*. Page 4- 9.

Fiedler, D. (2013). *Web Developer Basics: Learning about HTML5.* Available at: http://www.htmlgoodies.com/html5/Web-Developer-Basics-Learning-About-HTML5-3915046.htm#fbid=KzevKI4pQg9 [Last Accessed on: October 20th, 2014]

Google. (2014). *Word Trees.* Available at: https://developers.google.com/chart/interactive/docs/gallery/wordtree#Styling [Last Accessed on: October 20th, 2014]

Processing. (2014a). *Cover*. Available at: http://www.processing.org/ [Last Accessed on: October 20th, 2014]

Processing. (2014b). *Reference*. Available at: https://www.processing.org/reference/ [Last Accessed on: October 20th, 2014]

Sherwood, A. (2014). *Using Processing In The Web.* Available at: http://aaron-sherwood.com/processingjs/ [Last Accessed on: October 20th, 2014]

Dey, P. P. (2014). *Notes on Software Development Process.* Available at: http://asethome.org/soft/soft_Process2.html [Last Accessed on : 17/11/14]

Hughey, D. (2009) *The Traditional Waterfall Approach* Available at: http://www.umsl.edu/~hugheyd/is6840/waterfall.html [Last Accessed on: 17/11/14]

Agile Tools. (2013). *Agile Glossary* Available at: http://www.agile-tools.net/scrum.asp [Last Accessed on: 17/11/14]

BBC (2014). *BBC IPlayer* Available at: http://www.bbc.co.uk/iplayer [Last Accessed on: 16/12/14]