Improving usability and accessibility of Fuzzy Logic software systems with a web-based approach

Submitted MAY 2014 in partial fulfilment of the conditions of the award of the degree MSci (Hons) Computer Science

Craig Knott
cxk01u

With Supervision from Jon Garibaldi

School of Computer Science and Information Technology
University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____/_____/_____

Abstract

This project was both a practical implementation of an online fuzzy logic software system, and a research piece into the optimal design of a fuzzy logic system, and how fuzzy logic can be best introduced to novices of the field. Specifically, I implemented an online system for the creation and evaluation of fuzzy logic sets and systems, using mostly JavaScript and JQuery, with an R backend to deal with the processing requirements of fuzzy logic. In the end, I found out that... (using facts and figures)

Ensure to mention:

1. Changes that Luke makes
2. Friendly errors
3. Things from presentation
4. KeyPress Javascript library
5. Help system is dedicated, but offers links to other, helpful, external resources

Contents

1 Introduction

Fuzzy logic is an ever expanding field, and as such, the tools we are using to work in this field should also be expanding. It is also important that the merits of fuzzy logic are made apparent to those other than the experts of this field, as this would help to produce more advanced control systems in the future.

Many software systems for working with fuzzy logic have already been produced, of which many different approaches have been attempted, and been successful to various degrees. Examples of such systems include: The MATLAB Fuzzy Toolbox[1], An R Package named FuzzyToolkitUoN[2], XFuzzy[3], and fuzzyTECH[4] (a more comphrehensive overview of these systems can be found in section 3.2).

These system are all worthwhile pieces of software, and they fulfil their main objective of allowing for the creation of fuzzy systems. However, whilst researching these systems as part of my second year group project at the University of Nottingham (and actually working on one, in the case of FuzzyToolkitUoN), I noticed that there were two key flaws that the majority of popular fuzzy software systems suffered from: difficulty of use, or difficulty of access (or even both).

The main objective of this project is to produce a software solution for the creation, manipulation, and inferencing of a fuzzy logic system, which is accessible online. With a specific focus on solving the issues that are faced by fuzzy logic software systems that are currently used (difficulty of access and use).

Many different techniques will be employed in solving these fundamental problems, to hopefully create a system that is as easy to use, and as easy to access, as possible. Some of these techniques will include: online access; the ability to work with multiple file types, for cross compatibility; an intuitive design; unrestricted navigation, giving the user complete control and freedom; a dedicated, unobtrusive help system, to offer help to those that need it, but not to bother those that do not; and to build it in a way that allows for future expansions. A comprehensive list of all aspects of the software system can be found in section 4.

It could be argued that *another* fuzzy logic software system is not necessary, as it has been demonstrated that there are already many systems available. However, the currently available software suffers from the key issues identified above, and this project aims to resolve these issues, and attempt to spread the influence of fuzzy logic to those other than experts in the field.

There will, however, be certain areas that this software system will *not* be focusing on, as these are not relevant to the question posed in this research. Namely, this project will not be focusing on higher levels of fuzzy logic; it will only be focused on type-1. This is because the leap in difficulty from type-1 to type-2 fuzzy logic is very large, and type-2 is simply not a concept that is suitable or appropriate to introduce beginners to. More on this topic, including a definition of both terms, can be found in section 8.1.

[1] `http://www.mathworks.co.uk/products/fuzzy-logic/`
[2] `http://cran.r-project.org/web/packages/FuzzyToolkitUoN/index.html`
[3] `http://www2.imse-cnm.csic.es/Xfuzzy/`
[4] `http://www.fuzzytech.com/`

2 Motivation

The motivation of this project is a simple one: to produce a fuzzy logic software system that is easy to access, and easy to use, to help promote the wider adoption of fuzzy logic. The problem with systems currently available is that they suffer from one of the two following pitfalls: difficulty of use or difficulty of access. This means that novices can find it very difficult to get into the field, the software available does not facilitate productive use, and even experts can be held back by the software they are using. Some specific issues include: locating systems to use, complex installation processes, cost to the user, unintuitive user interface, or a requirement of (a considerable amount of) prior knowledge.

As part of my second year group project at the University of Nottingham, I worked on an R Package called FuzzyToolkitUoN. The goal of this system was to expand upon work completed by the Intelligent Modelling and Analysis group[5], to facilitate the use of fuzzy logic within the R programming language [7]. Whilst working on this project, a large amount of research into existing fuzzy logic software systems was conducted, and it was during this research period that the two key commons flaws were noticed. Unfortunately, due to the nature of the R programming language, and the package being produced, FuzzyToolkitUoN also succumbed to one of these pitfalls - difficulty of use. Personally, I was frustrated with this, and that is one of the reasons for the birth of this project - remedying past mistakes.

As have been mentioned, the greater adoption of fuzzy logic would be extremely beneficial, as it adds a new level of reasoning that classical logic simply cannot. As such, another side goal of this project is to make a system that is as easy to use as possible, regardless of the skill of the user in both terms of knowledge of fuzzy logic, and of using computer software in general. This will mean a project that will not only be very easy to access and use, but also help novices to learn about what they are doing, as well as why they are doing it, to help them gain a greater understand of the field of fuzzy logic.

The project detailed in this dissertation will aim to implement a fuzzy logic software system, in a novel format (online), and to specifically avoid the common pitfalls observed of other similar systems. Being online, the system is already on the right path to solving the difficulty of access problem, as the users will be able to access the system from wherever they are, and on what ever platform (as it will not require any plugins, like Java, or Flash). This also means it is more accessible to the novice user, or the computer novice, as they need only navigate to a website to use the system; there is no complex download and installation process.

The project will be heavily influenced by the field of Human Computer Interaction, to ensure that the system is as user-friendly, and easy to pick up as possible. User interaction with a software system is extremely important, and the way systems are designed has a huge impact on how they are received by the user base. Simplicity is important in this design, because studies have shown that users lose more than 40% of their time to frustration, and that in most of these cases, the user ends up angry at themselves, angry at the computer, or feeling a sense of helplessness [3]; which is obviously not ideal for a system that is attempting to help the user learn.

[5] `http://ima.ac.uk/`

3 Background Information & Research

3.1 What is Fuzzy Logic?

Fuzzy logic is a "natural" way of expressing uncertain or qualitative information [1]. It is a form of logic that deals with approximate reasoning, as opposed to fixed, exact values, like those found in classical logic (where we may only have properties being true, or false). Instead of these strict truth values, fuzzy logic systems have a range of truth, between 0 and 1. This makes fuzzy logic much better for handling and sorting data, and is an excellent choice for many control system applications, due to the way it mimics human control logic. Lotfi Zadeh, who formalised fuzzy logic in 1965, states that the key advantages of fuzzy logic are that it allows us to make rational decisions in environments of imprecision, uncertainty, and partiality of truth, and to perform a wide variety of physical and mental tasks, without any measurements or computations [9].

In a classical set, the membership, $\mu_A(x)$ of x, of a set, A, in universe, X, is defined:

$$\mu_A(x) = \begin{cases} 1, & \text{iff x} \in \text{A} \\ 0, & \text{iff x} \notin \text{A} \end{cases}$$

That is, the element is either in the set, or not. In a fuzzy set, however, we have grades of membership, which are real numbers in the interval, $\mu_A(x) \in [0, 1]$. Every member of a set has a membership grade to that set, depicting how true the property represented by that set is, for the given member [8]. The traditional syntax for representing members of a fuzzy set is given below (although a full working knowledge of fuzzy logic theory is not necessary for this project).

$$A = \mu_A(x_1)/x_1 + ... + \mu_A(x_n)/x_n$$

The easiest way to observe the merits of fuzzy logic is to look at terms that we humans use in our everyday life, and attempt to map these as crisp functions. For instance, terms like "hot", "cold", "tall", and "short", are all terms that we understand very well, and use often. However, if we were asked to give *exact* values for tallness, or shortness, we would not be able to do so. At what cut-off point would a person change from being considered short, to being considered tall? Fuzzy logic helps to alleviate these impossible choices, by having varying degrees of membership, for certain properties. The example in figure 1 shows this using three linguistic variables to describe the height of a person. Instead of at one point being either tall, short, or medium height, we, at all times, belong to all properties, to a differing degree.

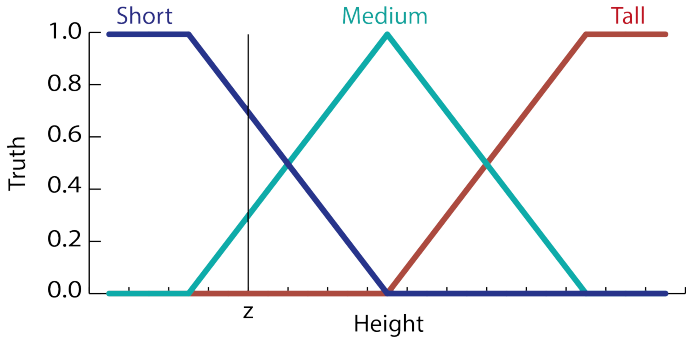

Figure 1: A fuzzy set depicting "height"

For instance, at the point labelled z, in the sets in figure 1, the membership to the "Short" set is 0.7, the membership to the "Medium" set is 0.3, and membership to the "Tall" set, is 0.0. This is, naturally, much more precise than simply saying we are "Short", "Medium", or "Tall".

3.2 Existing Systems

Fuzzy logic has been around for almost 50 years now, and, with the rising age of the computer, it would be alarming if no software systems for its usage were in circulation. Luckily, this is not the case, and there are many examples of software systems focusing on the use of fuzzy logic, of which many different approaches have been attempted, to varying degrees of success. In this section, a number of these software systems will be evaluated, to discern their positive and negative qualities, to help improve the design of the project presented in this report.

MATLAB Fuzzy Toolbox
The first system to be explored is MATLAB's fuzzy toolbox, an add-on for the MATLAB software suite, to work with fuzzy sets and systems. This toolbox provides everything required to create type-1 fuzzy sets and systems, with relative ease. The main advantage it has over most other systems is that it has a graphical user interface, which makes a task like working with fuzzy sets (that require a lot of visualisation and updating in real time) much simpler. There is also an extensive library of documentation and tutorials available for both MATLAB, and this specific toolbox, that help novices to get acquainted with the system. These things both help to make the system very easy to use, and novice friendly.

Unfortunately, these positives do not outweigh the major disadvantage of MATLAB, and it's fuzzy toolbox; which is that are pieces of proprietary software. This means that a novice to the field of fuzzy logic would have to invest a considerable sum of money, before they could even begin using the software. Whilst the system does have extensive documentation, and the user would be able to understand and use the system with relative ease, a piece of software does not require a large price tag to achieve this level of functionality and support. Another disadvantage of the MATLAB fuzzy toolbox is that is it not a dedicated piece of software, and is instead a limited subsection of the greater software of MATLAB. This means that the potential for extensibility is much less likely, as updates to the encompassing software would be deemed more important. It could even be argued that the installation of the MATLAB software, and then the installation of further software could be confusing to some novice users, which further alienates them. A final issue that has been observed is the excessive number of separate windows that are opened whilst working with the MATLAB fuzzy toolbox, as a new window is opened for each individual task that is being worked on (for instance, one for membership functions, one for rules, one for evaluations, which clutters the user's workspace rather rapidly).

FuzzyToolkitUoN
FuzzyToolkitUoN is an R-Package, produced by the Intelligent Modelling and Analysis group, at the University of Nottingham, that I personally worked on as part of my Second Year Group Project, at the aforementioned University. It provides functionality for the R Programming language to allow for work with fuzzy sets and fuzzy systems, including their evaluation. A major milestone for the project was it's official acceptance onto CRAN (The Comprehensive R Archive Network, an online library for R packages), in 2013. Being written in R, the package has access to the very powerful R processing tools, and graphics drawing capabilities, which help the user to visualise the system they are creating. Due to being hosted on CRAN, there is extensive documentation for every function in the package, including example usages, and explanations of all their parameters. This makes the usage of the package much simpler, because the user can easily access the documentation for any function they require, and there is a fully worked example for them to follow. Another advantage of the CRAN hosting is that any user with an R interpreter installed can access the package with only a few simple commands, helping the system to be much more accessible.

Unfortunately, there are downsides to working with the R language, the most prominent of which is the command line interface. Whilst the users have the features and functionality to plot the sets they are drawing, it can still be a cumbersome task, and does not promote ease of use. In a graphical user interface, the updating of graphs would be automatic, and the user could see their changes in real time, instead of having to modify a set, and then check what it looked like separately. Another issue with the package is that the descriptions of the functions, and the system itself, are specified so that a novice to fuzzy logic is not given the support they require; the system assumes that the user's knowledge of fuzzy logic is already somewhat sufficient.

XFuzzy (3.0)
Developed by the Institute of Microelectronics in Seville, Spain[6], XFuzzy (3.0), is a set of several tools, that cover the different stages of the creation of a fuzzy system. It allows for the construction of complex systems, whilst also offering flexibility. Each of the tools can be executed as an independent program, or, using the XFuzzy environment, can be integrated together as a graphical user interface, to ease the design process. As mentioned in the evaluation of the MATLAB fuzzy toolbox, a graphical user interface is a huge aid to the user in constructing a fuzzy system, as it allows them to see what changes they are making, and what effect these changes have. The software also runs on Java, which means it can be used on any operating system, as long as the Java Runtime Environment is installed, making it relatively easy to access.

There are, however, some issues with this software. These being that it is relatively unknown, is not updated frequently (the last update for version 3.0 was in 2012), and without using the graphical user interface, you are stuck using a command line interface, and must learn the system's bespoke language, in order to complete any tasks. The final disadvantage (which is also listed as an advantage, from a different view point), is the necessity for Java to be installed. This is an additional piece of software that this product is dependant on, and could further confuse the novice user with extra installations necessary.

fuzzyTECH
The final system that was observed was fuzzyTECH, another proprietary software, produced by INFORM[7]. It comes with a graphical user interface, making the interaction with the system much simpler than other software available. It also boasts that the application programmer does not require an understanding of fuzzy logic *or* programming to be able to use the system adequately - a feature that the software detailed in this report also hopes to boast. Another advantage of fuzzyTECH is that it produces source code that can be used on various hardware platforms (for instance, PCs and micro-controllers).

The main disadvantage of fuzzyTECH, just as with the MATLAB fuzzy toolbox, is that it is a pay-to-use piece of software. This greatly alienates the novice user, as they may not be willing to spend a large sum of money on a software they have never used, for a purpose they have never experienced before. There are also many different versions of fuzzyTECH, which would further confuse the novice user, as they may not necessary be aware of their specific needs, and thus be unsure as to which specific version would suit them best. In addition to this, fuzzyTECH has not been updated since early 2010, meaning the support that the novice user requires, may not be there - despite the systems extensive support functions.

[6] `http://www.imse-cnm.csic.es/`
[7] `http://www.inform-software.com/`

3.3 Platforms and Tools

This section introduces some potential platforms and tools that could have been used in the project, along with justifications for and against. The system itself consisted of two major parts, the web front end, and potentially some server back end (at the research stage of the project it had not yet been decided whether the application was to be entirely client-side or not). Both front end tools (including design frameworks and programming languages) and back end tools were explored and evaluated during this process to discern which would be the most appropriate for this project. Further to this, there was some research into graphing tools to be used on the website, as drawing fuzzy sets as graphs is an extremely powerful and intuitive way to represent them. A full list of the final decisions of tools to use, including their justifications, can be found in section 6.1.

3.3.1 System Back End

R-Node

R-Node[8] is a web front end that allows access to an R instance, providing full access to the R language, and the tools it provides. This means that there could be direct interaction with the FuzzyToolkitUoN package, alleviating a lot work on the back-end side of the project. This would greatly simplify the production process, as a brand new inference engine would not need to be produced. A particularly useful tool that FuzzyToolkitUoN provides is the drawing of graphs, which would be a huge usability boost to the system (as visualising fuzzy sets as graphics is extremely useful). FuzzyToolkitUoN is also currently under further development, so more features could easily be integrated into this project, if it were used for the back end. Unfortunately, the website for R-Node has been unresponsive for some time now, so finding documentation, or an actual download link is impossible. This shows that the tool is not very well maintained, and any issues discovered whilst using it would not likely be resolved in a reasonable time frame.

R-Shiny

R-Shiny[9] is an easy way to create web applications, using the R programming language. Like R-Node above, the advantages of using R would be the use of the FuzzyToolitUoN package, allowing for direct access to tools for the manipulation of fuzzy sets and systems. R-Shiny is currently in development, which means it lacks a lot of important features that would make its use much simpler, however, the development team are very active both in regards to responding to bug reports, and answering questions on their discussion board. R-Shiny allows for the creation of entire web pages using entirely R, or the use of just R as a back end, allowing for a much more dynamic web front end (created with HTML). Unfortunately, despite this more dynamic web front end, R-Shiny still does not provide an adequate way to create fully dynamic websites, which could be an issue for this project.

Node.js

Node.js[10] is a platform built on JavaScript that allows for the construction of network applications with ease. It is quick and easy to set up a basic server, however, can become quite confusing and difficult to set up a more complex one. Luckily, there is a lot of support available for Node.js, as it is becoming more and more popular among developers. This means that any issues that were encountered would hopefully be solvable within a reasonable time frame, and would not hinder the project too greatly. However, unlike the other back ends discussed, the use of Node.js would mean the construction of a new, bespoke, JavaScript based fuzzy inference system, which would be a significant increase to the work required.

[8] `http://squirelove.net/r-node/`

[9] `http://www.rstudio.com/shiny/`

[10] `http://nodejs.org`

3.3.2 Front End Programming Language

Naturally, the website would be built using HTML 5 and CSS 3.0, as they are the de-facto standard languages for the construction and design of websites. However, the most appropriate language to provide the actual functionality to the website required some research.

JavaScript
JavaScript is an obvious choice when looking to provide functionality to a website, due to it's high level of integration with HTML. It is a client-sided programming language used to alter the content of a displayed document, or in this case, website. The main advantage of JavaScript is it's ability to directly and easily manipulate the content of a web page, by accessing the document object model. The basic semantics and syntax of JavaScript are very similar to many other languages, meaning adopting this language is very easy to both expert and novice programmers. It also provides advanced features and functionality, that help to produce extremely powerful systems with relative ease. Using JavaScript also allows access to other JavaScript tools, such as JQuery. The disadvantages of JavaScript are that there are security issues (although this is not an issue with the system proposed in this project), and there is potential that JavaScript can be rendered differently depending on layout engine, causing inconsistencies.

Adobe Flash (ActionScript)
Adobe Flash is a tool for the creation of graphics, animation, games, and rich internet applications, which require the Adobe Flash Player to be viewed. This allows for the creation of websites that are both aesthetically pleasing, and are interactive for the user, making the entire experience of using the website much more pleasing. However, HTML 5 is now at the stage where it can almost perform the same functionality as Flash, negating it's necessity. It also requires that the user have Adobe Flash Player installed, which, whilst free, is an extra unnecessary download that could stand to make the system more difficult to access (for optimal accessibility, the project would require no additional downloads).

3.3.3 Front End Design Framework

A front end framework, in the context of web development, is a standardised set of concepts, practices and criteria for dealing with the production of the front end of a website. There are many advantages to using pre-made front end frameworks for developing web applications, the most prominent of which are that: they generally look more aesthetically appealing than a single individual would be able to produce, they provide access to a large selection of easy to implement dynamic user interface elements, and many of them deal with browser cross-compatibility issues automatically.

Semantic
Semantic[11] is a web front end development system produced by Jack Lukic. The main advantages of semantic are that is it tag agnostic (meaning any tag with UI elements can be used), and that it has a variety of elements with real-time debug output, allowing the application programmer to see what effect their code has. It also has the advantage that it isn't Bootstrap, spoken about below, which is often cited as being overused; this would give the project a slightly more unique feel. Unfortunately, as this is a less well known, and less well developed framework, there are much fewer UI elements to choose from, in comparison to a framework like Bootstrap, and the entire framework is less documented, and there is less online help (due to it's lower adoption rate).

[11] `http://semantic-ui.com`

Twitter's Bootstrap
Twitter's Bootstrap[12] is an extremely popular web development front end (so popular in fact, that some individuals[13] are beginning to complain about it's overuse), which helps to produce responsive websites, that function both on desktop and mobile devices. It is designed so that anyone of any skill level, beginner to expert, can pick up the framework and begin creating websites with relative ease. It provides a framework that produces websites that easily and efficiently scale for a multitude of devices, such as phones, tablets, and desktops, helping to make them much more accessible (although, not entirely relevant for this project, as there are many logistical issues with attempting to create a fuzzy logic system from a mobile device). There is also extensive and comprehensive documentation available for Bootstrap, including live working examples, and ready-to-use code samples, making its adoption into any project extremely simple. As mentioned above, however, Bootstrap is so popular nowadays that many people find it's usage off putting, as a large number of websites are beginning to look like clones of one another. For this reason, if Bootstrap were to be used in this project, it would be important to ensure that some customisations to the basic styles were applied, so that the website looked unique.

HTML KickStart
HTML KickStart[14] was the final web front end that was evaluated. Like Bootstrap, this was a responsive and scalable framework that meant it was suitable for devices ranging from desktops, to mobiles (even if the application itself wasn't). It also comes with over 240 icons, which can be used to help the design and usability of elements on the website (as people generally associate certain icons with certain tasks), and the entire framework can be included with just two files (instead of Bootstrap that requires some level of configuration to ensure all possible features are included). On the other hand, this framework provided much fewer elements than other packages (like Bootstrap), and the documentation presented on the framework's web page was confusing and laid out poorly (which is ironic, as the website was built using HTML KickStart), which made searching for specific tools much more troublesome.

3.3.4 Graphing Tools

The drawing of fuzzy sets is extremely intuitive and powerful in a graphical format. For this reason, a suitable tool for the drawing of graphs on a website was required for this project. This would help to make the system considerably easier to use (as the user would be able to see their work as they were completing it), would take advantage of the graphical interface, and help to improve the general aesthetics of the system.

Directly from R
As mentioned in the back end section, there was potential to use the R language as the back end for the system. This meant that any R libraries could be imported and used for the front end. In the case of this system, the use of FuzzyToolkitUoN would allow for the work with fuzzy logic and fuzzy sets, without the need to write an entirely new client-sided inference system from scratch. Using FuzzyToolkitUoN to draw the graphs would promote uniformity between the back end and front end (if the back end did end up being R based), and would mean the implementation of drawing graphs would not need to be considered (as this would be handled by the R package). However, transferring graphical data from an R back end to a web front is a difficult task, and the graphs themselves would be simple, static images, where dynamic or interactive graph would be much preferred.

[12] `http://getbootstrap.com/`
[13] `http://css.dzone.com/articles/please-stop-using-twitter`
[14] `http://www.99lime.com/elements`

Google Charts
Google Charts[15] is a tool for creating and displaying highly customisable graphs and charts on a web page. This package provides a large range of graphs and charts, all of which have interactive elements (helping to improve usability by a considerable margin, and reduce screen clutter). As this service is provided by Google, there is a large library of support available, and it can be assumed that the service is of a high quality. The only negatives of this tool are that the API isn't as simple to use as it potentially could be, and the returning of graphs can sometimes be slow.

Flot
Flot[16] was the final graphing tool that was evaluated. It is written, and accessed, through jQuery, which means any project using JavaScript can easily incorporate it. The project is in constant development, and bug reports filed by the users are taken into account, as well as their general suggestions. However, the package itself can be difficult to get started with, and there are much fewer graph types available, compared to other services.

4 System Specification

As has been mentioned multiple times, the system to be produced is an on-line fuzzy logic inferencing and visualisation system, with the specific goal to be as user friendly, and as accessible as possible. In this section, the functional requirements (goals that the software must achieve), and non-functional requirements (constraints placed upon the system) of the system have been enumerated. This allows for the reader to see what functionality and features the system will have, but also provides a list of test criteria for later stages of the project.

4.1 Functional Requirements

In this section, the functional requirements of the project have been enumerated, so that they can be referred to at the evaluation stages of the project.

1. Users can manipulate membership functions
    - 1.1. Users will be able to create membership functions, of type...
        - 1.1.1. Gaussian
        - 1.1.2. 2-Part Gaussian
        - 1.1.3. Trapezoidal
        - 1.1.4. Triangular
    - 1.2. Users will be able to add membership functions to variables
    - 1.3. Users will be able to edit membership functions, including:
        - 1.3.1. Membership function names
        - 1.3.2. Any function parameters
    - 1.4. Users will be able to delete membership functions from variables
    - 1.5. Users will be able to access help on how to create membership functions
    - 1.6. Users will be able to see a plot of their membership functions
2. Users will be able to manipulate linguistic variables
    - 2.1. Users will be able to create linguistic variables

[15] `https://developers.google.com/chart/`
[16] `http://www.flotcharts.org`

    2.1.1. Users will be able to create input variables
    2.1.2. Users will be able to create output variables
  2.2. Users will be able to edit the range of linguistic variables
  2.3. Users will be able to delete variables
  2.4. Users will be able to rename variables
  2.5. Users will be able to access help on how to create variables
3. Users will be able to manipulate system rules
  3.1. Users will be able to create rules for the system
    3.1.1. Users will be able to specify rule terms
    3.1.2. Users will be able to negate certain terms in a rule
    3.1.3. Users will be able to change the weight of a rule
    3.1.4. Users will be able to specify the connective to be used in the rule
  3.2. Users will be able to edit any previously constructed rules
  3.3. Users will be able to delete any previously constructed rules
  3.4. Users can access help on how to create rules
4. Users will be able to manipulate system-wide parameters
  4.1. Users will be able to edit the system-wide parameters
    4.1.1. Name of the system
    4.1.2. Type of evaluation to use
    4.1.3. "And" method to use
    4.1.4. "Or" method to use
    4.1.5. Aggregation method to use
    4.1.6. Implication method to use
    4.1.7. Defuzzification method to use
  4.2. Users will be able to access help on what affect these changes make
5. Users will be able to perform file input/output on the system
  5.1. Users will be able to export their system to various formats, including
    5.1.1. MATLAB .fis file
    5.1.2. FuzzyToolkitUoN .fis file
    5.1.3. JSON Object file
  5.2. Users can access help on how to export files and what is supported
  5.3. Users will be able to import previously made systems of various formats, including
    5.3.1. MATLAB .fis file
    5.3.2. FuzzyToolkitUoN .fis file
    5.3.3. JSON Object file
  5.4. Users can access help on how to import files and what is supported
6. Users will be able to evaluate their system
  6.1. Users can provide a value for each input, and receive the output value
  6.2. Users can access help on how the evaluation process works

4.2 Non-Functional Requirements

In this section, a list of the non-functional requirements, that place constraints on the system, is provided.

Accessibility
This is, naturally, a huge goal for the system, as it was one of the reasons for it's conception. The proposed system is to be made available entirely on-line, allowing anyone with internet access, and a computer, to use the system. The system's accessibility will be further increased by it's lack of client-side dependencies, meaning the user is not required to download or install any additional software if they wish to use the software. The only potential issue with accessibility is if the server that is hosting the website was to stop functioning. This is, however, a potential issue that does not have a solution.

Usability and Operability
Due to the large range of potential users, the system will be designed in a way that is easy to navigate and use, regardless of the skill of the user. There will be a dedicated help system present in the system, that will allow the user to get help, whenever they need it, without having the leave the system and check some external documentation. The graphical user interface of the system helps to promote the ease of use, as the user is not expected to memorise a collection of commands, and instead need only press a few buttons to construct the system they wish to construct.

Maintainability
One of the eventual goals of the project is that the inference engine, that will initially be a bespoke JavaScript implementation, or using the FuzzyToolkitUoN implementation, can be swapped out, and any compatible engine be used. This means that this system would evolve into a web-front end, for any Fuzzy Logic back end; greatly expanding it's usefulness. As a result of this, and a result of good software engineering practices in general, the code will be kept as readable and modular as possible. The large quantity of JavaScript code that is required will be split into separate files, so that the maintaining programmer can easily identify issues. Each function will be commented in a JavaScript equivalent of JavaDoc, so that automated documentation can be produced, if necessary. This will hopefully ensure that new back-ends can be easily incorporated into the system, and any additions necessary will be quick and easy for the maintaining developers.

Quality
As this system is to be used externally, and will be a representation of both myself, and the University of Nottingham, there are several quality issues that must be addressed. The system must be built so that it is robust, and works as the user expects, but it must also contain as few bugs as possible. Any bugs that are identified should be reportable to the maintainer, and be fixed as soon as possible.

The quality of the code must also be considered. In this regard, the code will be written in as modular a way as possible, and useful comments will be provided to highlight the purpose of each function, and to illustrate any particularly complex code.

Resource Requirements and Constraints
As this system is aimed at any users of any skill level, no assumptions can be made on the level of hardware that the users will possess. For this reason, the system will be designed to use as few resources as possible. Fortunately, due to being a web-based system, the load of the system would be fairly minimal anyway, as it is mostly loading only JavaScript and HTML. The only true computation takes place when the system is evaluated, but this could take place on the server side, and thus would not be a concern for the user.

The ability to load and save files potentially causes a problem, but only if the user has very little hard drive space, and attempts to save an extremely large file from the system. Unfortunately, there is nothing that can be done about this, although even very large systems will have a relatively small file size.

Cross Platform Compatibility
Due to the project being a web-based system, it is difficult for it not to be cross platform compatible. However, there are still a few issues that may have to be dealt with, especially when looking at different browsers that the user may be using. For this reason, research into how the system performs on different browsers will be important, so that it can be assured that any user using any browser will have full access to the system, as it was meant to be.

Security
The issue of security is not one that need be discussed in great detail, as there are no security issues with the system to be produced. In a web-based system, there is potential for many security issues to be present, such as the storing of cookies without the users permission, or the tracking of what a user may be doing. However, the system being produced here is simply a tool for the users to create fuzzy systems in, and there is no useful information that tracking their movements would return. There is also no need to save cookies, as the system will have full file loading and saving functionality.

Reliability and Robustness
As the system is to be used by experts and novices alike, the system needs to be as robust and as reliable as possible. This is to ensure that the expert users can work uninterrupted, and that the novice users do not get confused, if something unexpected happens. This is why there will be rigorous testing of the system, by both novices, and by experts (both in regards to fuzzy logic, and in regards to the use of a PC), so that the system can be thoroughly bug-checked, and feedback can be gained on the usability of the system.

Documentation
The system will be fully documented through comments in the JavaScript code. This will be done using a style similar to JavaDoc, so that documentation pages can be generated easily. The system itself will have the extensive help system present on every page, so that any user confused with the system can be guided in the right direction. For these reasons, there will be no external documentation for the project produced (which actually helps the "ease of use" objective, as the user does not need to look through a large external document to find help).

Disaster Recovery
All of the files for the project are stored in a GitHub repository (which is, for the time being, private), so any issues involving loss of code are not a concern. There is potential for the system server to go down, and in this case, due care and attention will be dedicated to resolving this issue. The only other issue that requires disaster recovery is if the user enters their system, and closes the browser without saving. This is, however, out of my control, although a pop-up warning the user may be implemented.

5 System Designs

In this section, all of the design aspects of this system have been detailed and justified.

5.1 UI Design

As the purpose of this project was to aid in the accessibility and usability of fuzzy logic, to users of all skill levels, the design of the user interface was extremely important. The difference between a good piece of software, and a great piece of software, can easily be the interface they provide. In fact, the entire reason that FuzzyToolkitUoN is an inadequate piece of software for specifying fuzzy sets, is it's interface, and this is the exact thing this project aims to overcome. There were two main iterations to the user interface design, the first was a simple attempt to include all information on the page, in an easily viewable format. The second iteration was a refinement of the first stage, in which design principles were applied, and feedback was gathered from potential end users.

It is worth noting that having all the different tasks of constructing a fuzzy system (specifying inputs, specifying outputs, specifying rules, evaluating the system, and file input and output) are all distinct tasks, and there is no reason for them to be together at any point. This is why, regardless of design, these tasks are all separated and are on different pages, or tabs, of the website. Further to this, the designs presented below cover only the input variable creation page and the rule creation page, as the input creation page is an exact replica of the output variable creation page, and the remaining pages or tabs of the website do not require much level of design, as they are relatively small.

5.1.1 First Iteration

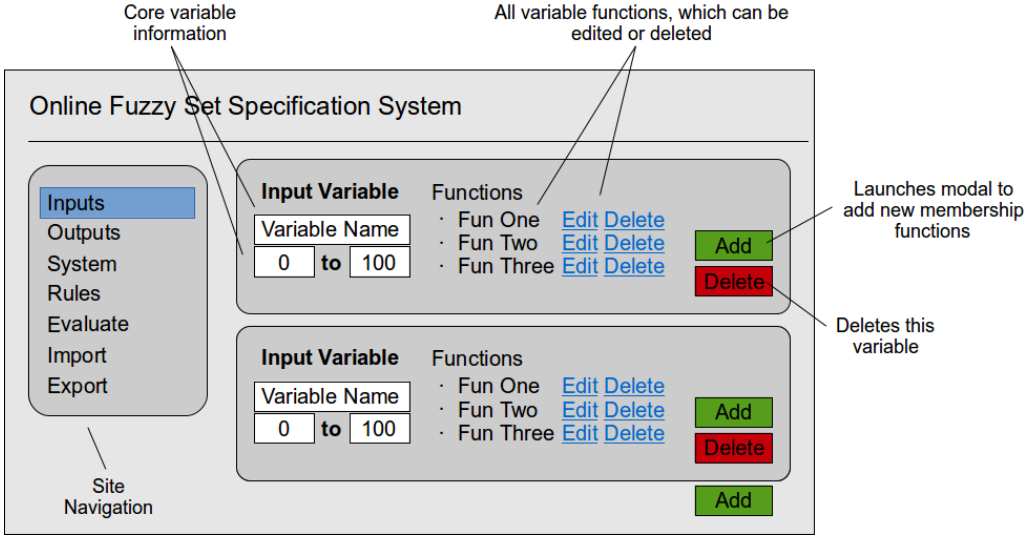

Figure 2: The first iteration of the design of the input creation page

The first iteration of the input creation page listed the navigation to the left hand side of the page, as this is a standard positioning for navigation, and the users would, most likely, be accustomed to this.

The separation of different tasks to different sections of the website is very important for the construction of a fuzzy system, as there are many individual elements, and a lack of separation of these would cause a great deal of confusion for the user, as there would be a huge number of elements on the screen at one time.

The design displays the core information of the variable on the far left (including name and range), as this is where the user's eye would fall first. The next set of information (the functions contained within the variable), are displayed further to the right, as the user would look to here after reading the initial information. The users have the option to edit or delete any function that they have created, granting them complete freedom over the system and everything they do within it. They also have the option to add a new membership function (which is a green button, to represent creation, which brings up a modal window to be used to construct the membership function), or delete the current variable (which is a red button, to represent destruction).

This design uses long horizontal representations of the variables, so a large number of them can be on the screen at the same time, as they take up little vertical space. This means the users can view many of their variables at the same time, and make edits as necessary. Colour is used sparingly throughout the page, so that it can be used effectively for highlighting important information or elements, so they are easily visible to the user.

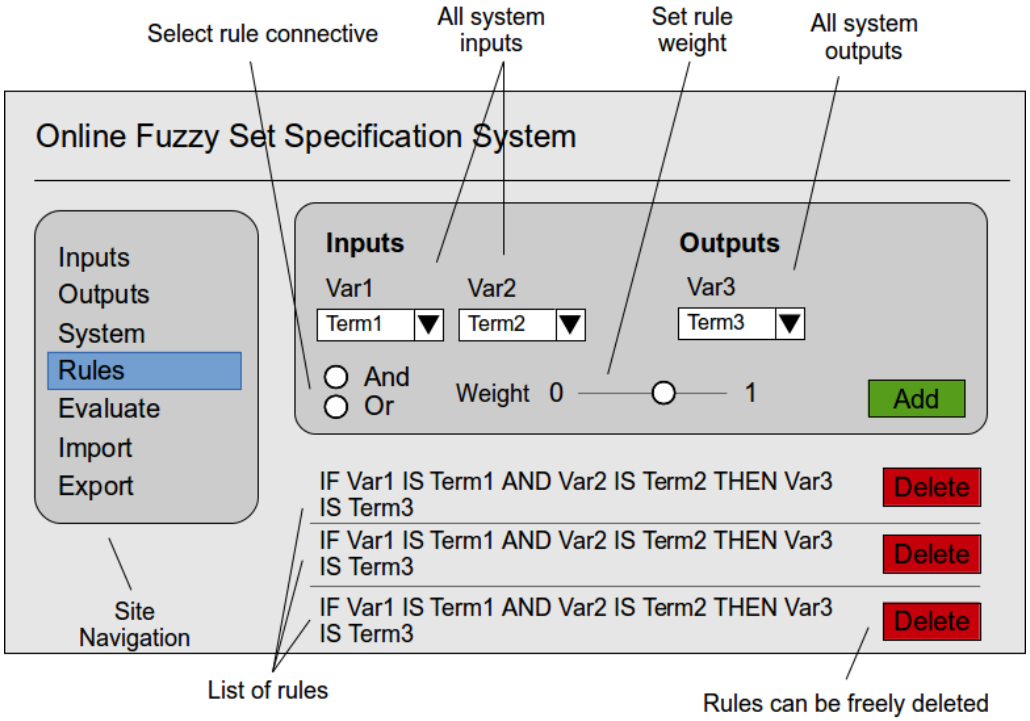

Figure 3: The first iteration of the design of the rule creation page

The rule creation page also follows a horizontal theme, which is especially effective for the rules, as there could potentially be a long list of them. The creation of the rules, and their displaying, are two clearly distinct segments to the page, which means there is no confusion between the two, and the process remains a simple one.

In keeping with the standards of the previous page, the "Add" button is coloured green, to represent creation, and the "Delete" buttons present by each rule is coloured red, to represent destruction. These clear colour separations are useful to the user, as they do not even need to read a button to gain an understanding of what it will do, and they can be more careful to not make mistakes.

As can be seen, the navigation and header of the rule creation page are the same as

those on the input creation page, which promotes a consistent style amongst the pages, and helps to remind the user they are still within the same system, even if they are completing a different task.

5.1.2 Second Iteration

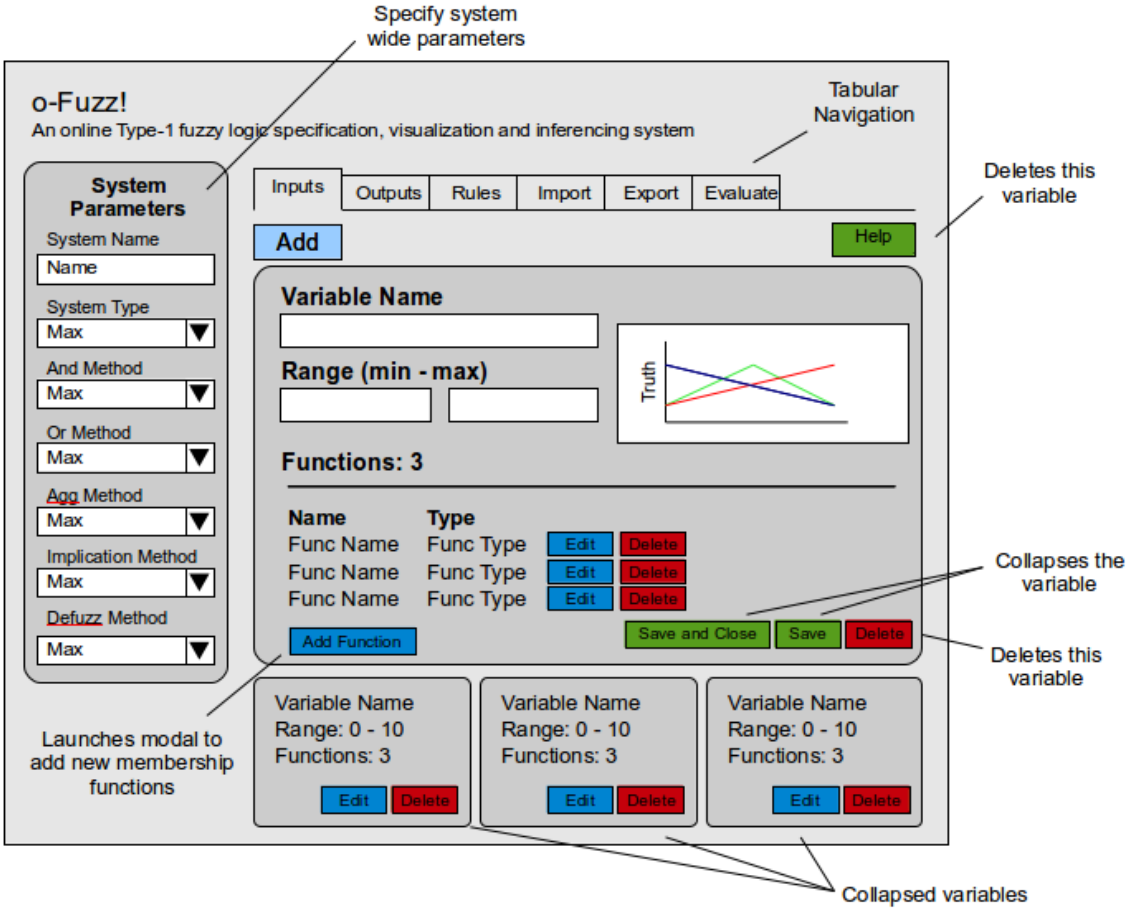

Figure 4: The second iteration of the design of the input creation page

The second iteration of the input creation page design includes many improvements over the first iteration. Specifically, this second iteration dealt with the issues the first iteration presented, and had well known and well documented usability heuristics and design principles applied to it, to ensure optimality.

One of the largest changes to the design was the re-working of the navigation to be entirely tabular, and to replace the old navigation area with system wide parameters segment. This tabbing helps to split the different tasks of the system in an intuitive manner, as most users are familiar with the concept of tabs, due to their popularity within internet browsers [**?**]. The system wide parameters are now displayed in place of the old navigation, which means they can be changed, regardless of the tab the user is on. This is especially convenient when the user comes to evaluate the system, as it makes tuning much quicker, and all permutations of parameters can be evaluated with ease.

The overall structure and displaying of the variables has also been drastically changed in this second iteration of design. The reason for this was that, whilst the old design promoted a large number of variables on the screen at any one time, this would severely increase the cognitive load on the user, and they could quickly become confused.

The new design works by having expandable and collapsible variables. When in the collapsed view, the variable takes up a much smaller space, displaying only key information, and in the expanded view, all of the information of the variable is present (like in the previous design). The ideal number of elements on a page is 7±2 [4], which can easily be exceeded with these new collapsed variables. However, the reason this design works, is because when the variable is collapsed, the user feels like it is completed, and they no longer need to concern themselves with it. So whilst this new design may look more cluttered, it significantly helps to reduce the cognitive load of the user, as they can create a variable, and then essentially forget about it.

To fit with the consistency of the other pages, the buttons for the editing and deletion of membership functions have also been changed. The edit button is now blue, which is the system wide colour for editing, and the delete buttons are now red, the system wide colour for deletion. This keeps the website consistent, and makes the functionality of these buttons easily identifiable.

A graphical representation of each variable is now also displayed in the expanded view, allowing the user to quickly interpret how their system is progressing, and whether it looks as they expected. This is a very powerful tool, as users will be able to spot errors much sooner than if this were not present.

Some other smaller changes include the moving of the "Add" button from the bottom right of the page, to the top left. The reason for this is that, in the first iteration of design, the add button would constantly move down the page as variables were added, and a moving button can be frustrating and confusing for some users. Another change is the addition of a short name for the software, with a longer tag-line underneath this, at the top of the page. This shortened title allows for users to refer to system with ease, allowing them to search for help for the system if necessary, and simply discuss the system with other potential users. The longer tag-line allows for a brief description of the system, so the user knows whether it will be able to accomplish the tasks they have set out to achieve.

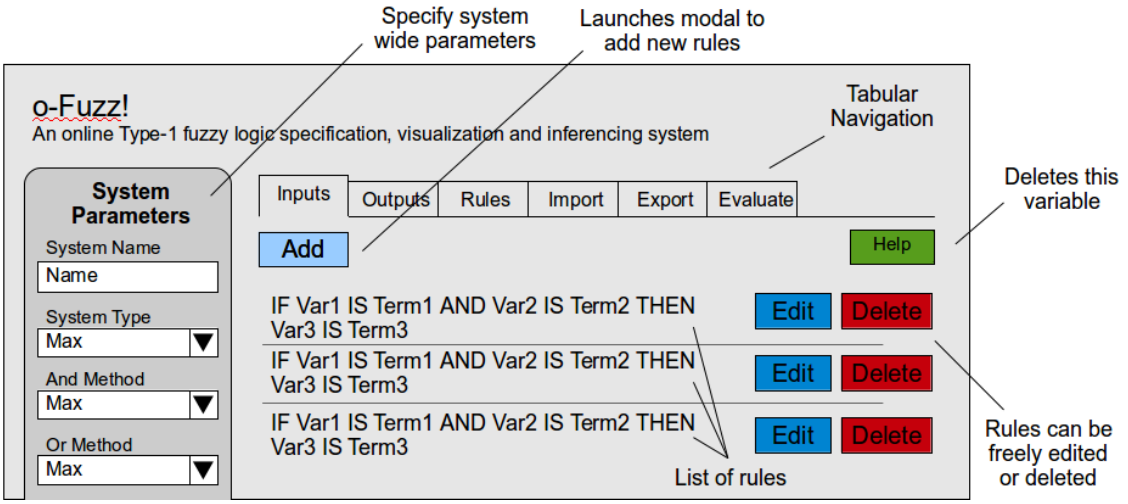

Figure 5: The second iteration of the design of the rule creation page

The rules creation page also received heavy changes in the second iteration. The first of

these was the inclusion of certain features that were not present in the first design, but were necessary for the functioning of the program. For instance, the ability to edit rules once they had been created, the ability to create negated terms (for instance, "If food is *not* good") and the addition of the help button to the page.

However, the biggest change is the removal of the ability to create rules on this initial page. Instead, as with the creation of membership functions, this has been moved into a separate window that is launched when the user clicked the "Add" button. This reduces the clutter on the page, and means the rule creation functionality is only accessed when the user requires it (helping to reduce cognitive load on the user).

In addition, an extra feature was included in the form of a rule table that could be displayed if the user was using a system with two inputs, and one output (a relatively common set up for a fuzzy system). If this were the case, the table would be displayed, mapping the inputs to the outputs, in an intuitive graphical manner, an example of which can be seen in figure 6.

AND		Temperature		
		Low	Medium	High
Heart Rate	Low	Critical	Rising	Critical
	Medium	Critical	No Threat	Critical
	High	Critical	Rising	Critical

Figure 6: An example of a rule table that would be displayed if the user had two inputs (heart rate and temperature) and a single output (urgency)

5.1.3 Heuristic Evaluation of Second Iteration

The new design also went through a heuristic evaluation [5], using the Usability Heuristics laid out by Jakob Nielson [**?**], and the Golden Rules for Design, laid out by Ben Shneiderman [6]. Both of whom are extremely well known and very influential in the field of Human Computer Interaction, and their heuristics define some of the most basic, but most important properties a user interface should possess.

1. Visibility of system status/Offering of informative feedback

   It is important that the users of a system are always updated as to what is happening within the system. This will be implemented through the use of JavaScript alert messages to alert the user to any errors they have made, or any important changes they have taken place.

2. User control and freedom/Support internal locus of control
   The user should be in full control of their experience of the system at all times, and thus a fluid and flexible navigation system is important. This is the purpose of the tabular layout of the website, as the user is free to travel to whichever pages they wish, in whichever order. More details on the navigation of the system can be found in section 5.2.

3. Consistency and standards/Strive for consistency
   A good interface should also be consistent in it's design, and follow platform standards. As you can see from the design of the input creator and the rule creator pages, the tabular layout is consistent throughout the website, and colours such as blue, green, and red have clearly defined purposes, which stay consistent throughout the as well.

4. Error prevention/ Help users recognize, diagnose, and recover from errors
   Unfortunately, with a system that is entirely dictated by user input, it is very difficult to prevent the user from making errors. However, measures will be put in place to ensure these errors do not affect the system. For instance, there is no way to enforce the user of a system to enter a number, but an informative error message will be displayed, telling the user this is not valid, and what should be done to rectify the issue (and, of course, not accepting the value into the system).

5. Recognition rather than recall/Reduce short-term memory load
   As mentioned multiple times already, the system is designed to reduce cognitive load on the user as much as possible. This is done by reducing the number of elements on the page at any one time, and by using modal windows for the creation of new membership functions and rules, so their creation and their display are distinct.

6. Aesthetic and minimalist design
   To reduce any possible distractions for the user, the system is designed in a minimalistic fashion, using colour very sparingly, and sticking to neutral shades for background elements. The only colour used in the system is on the graphs drawn, and on the important buttons the user will be pressing. These buttons are essentially colour coded so the user is aware of their functionality, without even having to read them. Further to this, the use of modal windows to essentially hide functionality greatly helps to reduce clutter on the pages of the website, giving it a much cleaner look.

7. Help and documentation
   Due to the goal of being as easy as use as possible, the system is to be designed with a dedicated help system, built in. The advantage of this is that the user is able to access help without having to leave the application itself, meaning a minimised distraction time. Help is accessed via the large green help button present on every page, which is easily visible, and provides concise and helpful information for the user.

As the final stage of the heuristic evaluation process, the "sins" of New Media Design [2] were also evaluated, to ensure an optimal design had been produced. The second iteration of design has taken these sins into account and has avoided those that were appropriate. Specifically, the website features no bulky borders, a generous use of margins, left alignment of elements (as opposed to centring), and an avoidance of a busy background, opting instead for a very simple white background. All of the aforementioned design choices could have easily caused distractions for the user, and damaged their enjoyment of the system.

5.2 Navigation/Control Flow Design

As of the second iteration of the design, the website has six main sections, enumerated below;

1. Inputs; The page used to construct input variables of the system

2. Outputs; The page used to construct output variables of the system
3. Rules; The page used to construct rules of the system
4. Evaluate; The page used to evaluate the system
5. Import; The page used to import files into the system
6. Export; The page used to export file from the system

The generally *expected* path for a user of the system to take, would be to launch the system to the input creation screen (the landing page), on which they would spent some time constructing their input variables. They would then move to the output variables, construct those, and then move onto the rules, and construct those. At this point, the path splits between evaluating the system, exporting it for later, or a mixture of the two.

Due to the ability to import files, an entirely new path is also available throughout this process, in which the user begins by importing a file, and then either editing it, or going straight to evaluating it.

The expected path through the system (beginning at the inputs page, as this is the landing page of the website), is represented in figure 7

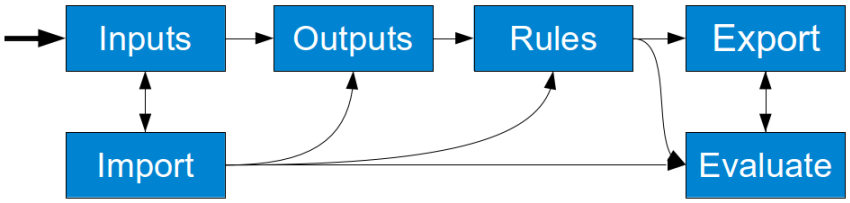

Figure 7: Expected path a user would take through the system

Of course, whilst the general path through the system has been determined, this is not the only path through the system, and there is no guarantee that the user will follow this path. In fact, the user has the freedom to traverse the system in any path they please, as shown in figure 8. This is important for promoting freedom and a sense of control within the system. The user is free to decide how they wish to travel through the system, and this means they are free to traverse forward or backward through the system to make any alterations, or fix any mistakes they may have made.

5.3 Internal Design

The user interface of a system is a huge part of the design process, and is extremely important, as this will be how the user will actually be interacting with it. However, this is not the only part of the system that requires designing; as the internal workings of the system also require a lot of thought.

For this project, a web-interface was to be used to interact with a back end that could deal the processing of a fuzzy logic system. By this point in the project's development, the software's implementation decisions had been made (which are detailed in section 6.1), and the decision of the back end to be used was the FuzzyToolkitUoN R Package, using the R-Shiny service, to interact between the front end and back end. The diagram in figure 9 shows how the interaction between these two systems is handled.

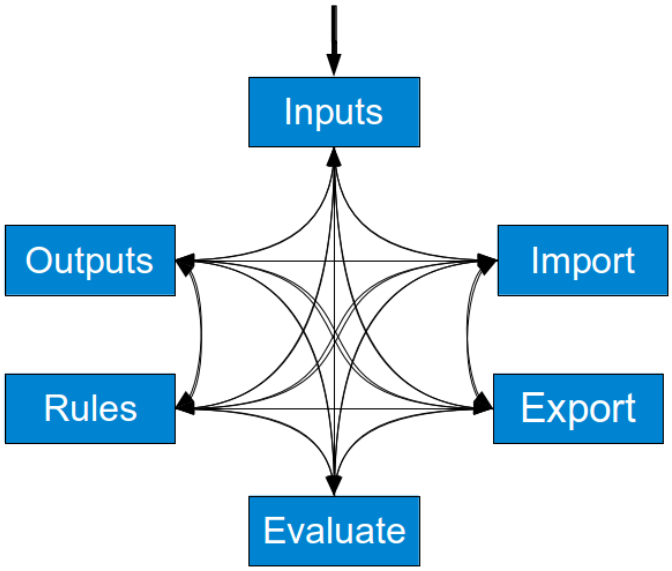

Figure 8: Possible paths through the system

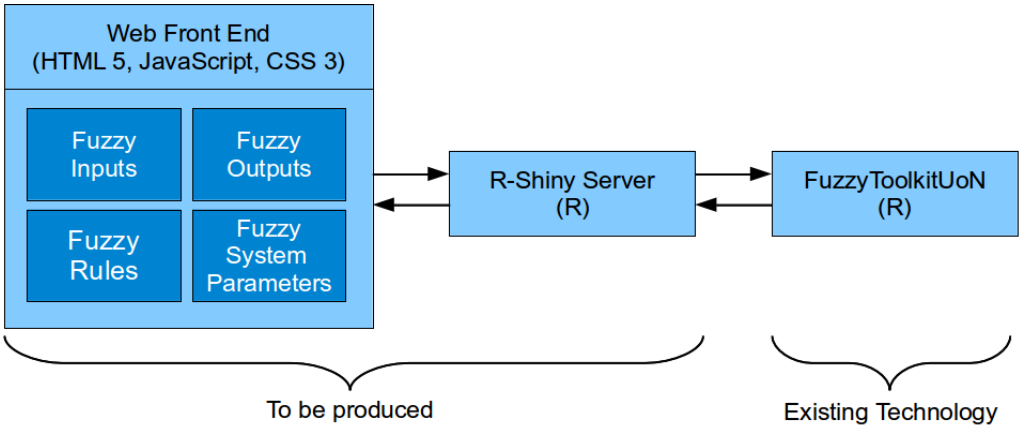

Figure 9: Interaction between the back end and front end of the system

As you can see, the web front end will be constructed using HTML 5, JavaScript, and CSS 3, and will allow for the creation and specification of fuzzy inputs, outputs, rules, and system parameters. Once a system has been specified, and the user chooses to evaluate the system, the information of the system is passed to the R-Shiny server. This then takes all the form elements from the web front end, extracts their values, and constructs a FIS file, within R. This can then have functions from the FuzzyToolkitUoN package applied to it, and the results of these functions can then be passed back to the web front end, for display to the user.

6 Software Implementation

1. simple introduction to the program, what it does and who it is for
2. technical solution adopted, what technical solution has been implemented, whether it is ideal, whether an alternative exists
3. software engineering information, program design, structure, definition language,

test plans

4. development approach used (evolutionary, build and fix)
5. Problems encountered (bugs, errors, uncompleted sections of code)
6. hardware/ software requirements

6.1 Key Implementation Decisions

A list of the tools and platforms that were used in the project, including justification for their inclusion. (To cover: Languages used, Web technologies used, Shiny, Bootstrap, JQuery) - using an old version of bootstrap because i was more familiar with it and work had already begun

6.2 Detailed Description of the Implemented System

A detailed description of the system, including how the individual sections were implemented, how they interact, strengths and weaknesses of individual components. Talk about the system as a whole as well, covering how the parts interacted and how well they did so.

6.3 Problems Encountered

A description of any major problems encountered during the implementation of the project, causes of these, and their resolutions - if there were any. Talk about the impact of the problems on the project as a whole, and how this potentially affected it (like changes to designs, etc) - SHINY BEING A MASSIVE CUNT, but fixing it! confusion due to such a large code base

7 Evaluation of the Project

7.1 Functionality Testing

check the software fits the functional requirements. Tests that look at the functionality of the system, including the individual components and the communication between the different parts. Give a list of tests, and the results of each of these tests, to ensure each functional requirement has been met.

7.2 User Feedback Testing

Comparisons against other software, and checking against non-functional requirements. General explanation about the tests, and the participants, and what we are looking for (non functional, usability, accessibility). talk about using disseration to do fuz coursework. “Considering I wrote both FTU and oFuzz, i found oFuzz much easier to use. Having the fuz coursework gave me a unique experience to test using both systems, as a user, instead of as the developer.”

An important part of evaluating the usability and accessibility of my system was to have real world users attempt to actual use not only my system, but similar systems, so that they could make comparative comments. In order to do this, I set up several sessions in which I would invite users of various skills levels, both in terms of computers, and fuzzy logic, to complete a list of tasks using all three software systems, after which I would ask them to give their feedback and opinions on all the systems. The three software systems that were evaluated were: my project, the MATLAB fuzzy toolbox, and FuzzyToolkitUoN, within the standard R environment.

I split the participants into four main categories, based on their skill levels in terms of using computers, and knowledge of fuzzy logic. There were a total of 23 participants in these studies, and the distribution of skill levels is displayed in figure 10. The reason for this split was so that these distinctive groups could be evaluated individually, and their specific requirements could be observed. For instance, a participant skilled in computers, but not in fuzzy logic, would not struggle in navigating a system, but could potentially struggle understanding some of the fuzzy terminology.

		Computer Skill	
		Low	High
Fuzzy	Low	7	5
Logic Skill	High	3	8
	Total	23	

Figure 10: Distribution of participant skill levels

The task assigned to the participants was designed to use as much of the different systems as possible, but focused mainly on cross-compatible parts of the systems, so they could be easily compared. The test itself was to construct the fuzzy tipper example, using service and food as inputs, to produce a number for the tip to leave (the full set of instructions can be found in appendix A).

7.2.1 FuzzyToolkitUoN Evaluation

Good things, bad things, statistics to back this up, talk about non funcs, and mention each type of user

7.2.2 MATLAB Fuzzy Toolbox Evaluation

As above, but comparisons with above. specificaly mention that MATLAB has milliions of windows to open which make it very confusing!!!

7.2.3 My Project, O-Fuzz, Evaluation

As above, but comparison with above and above above

7.2.4 Summary

Overall results and comparative statistics. The main two factors that were observed whilst the participants were completing the tasks were the speed at which they could do so, and the ease. Generally a faster completion meant either a high level of understanding, or an easier piece of software to use. The data collected strongly suggests that completion of the task using FuzzyToolkitUoN was the most difficult, which after speaking to the participants was the result of a poor user interface, and a very steep learning curve (especially for those of a novice computer skill level). The graph in figure 11 shows box plots of the completion times of each of the tasks, for each of the different groups. Each of these plots shows that FuzzyToolkitUoN was the most time consuming task to complete (taking on average 100 seconds), and that the graphical systems were much easier to use (with MATLAB on average, taking 40 % less time, and my new system taking 10 % less time than that).

some graph yo

Figure 11: Time taken to complete the tasks in the different software systems

Whilst the time taken to complete the tasks was a strong indicator of the success of the software system, it was also important to ask the participants which system they enjoyed using the most. The results for this were conclusive, which 100% of participants (across all categories) claiming FuzzyToolkitUoN was the piece of software they enjoyed using the least. This is because.... The piece of software that the participants enjoyed using the most was x % in favour of my produced software, over MATLAB (and the majority of those that said they preferred MATLAB said so as they were already very familiar with MATLAB). The results for most favoured, software system can be seen in figure 12, organised by category of participant. The main reasons for an attraction to MATLAB were... and o-fuzz

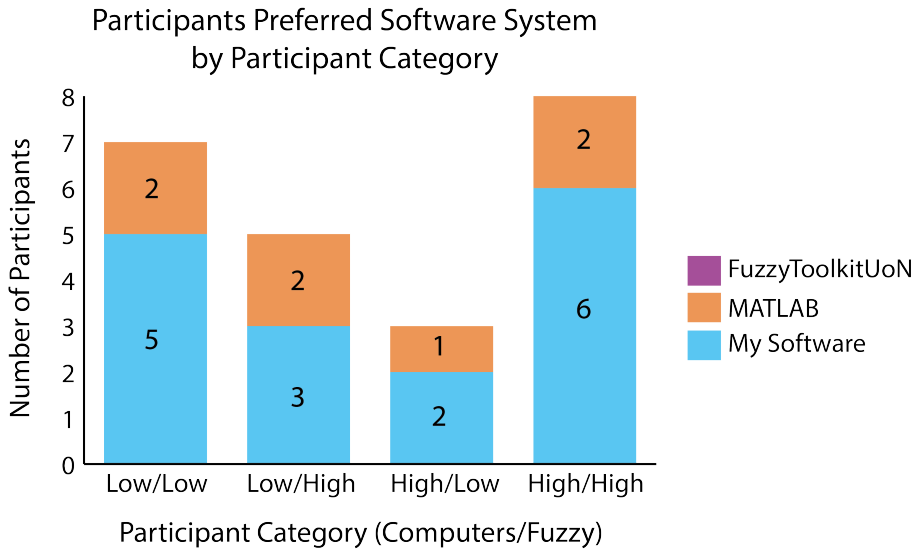

Figure 12: Favoured/Least Favoured software system, by participant category

7.3 Successes and Limitations of the Project

Explain what went well with the project, what didn't go so well, and what could be done better for next time (with concrete goals). issues with FTU as the backend (some options in params disabled because they don't work in FTU - not my fault!).

8 Further Work

A list of things that could be done to expand the system in the future / things I did not have time to complete

8.1 Type-2 Fuzzy Logic

To include a briewf explanation of what type 2 fuzzy logic is, and why it is different to type 1. Also to include why it was not included.

8.2 Backend Interoperability

Talk about how the majority of the system is written in JavaScript, so that the system could be ported to different inference engine backends with little work.

8.3 Customisations

Talk about potential for a site that is more customisable for the user, as this would help them feel more at home, and make the system more effective for them. But also mention how basic customisations (like changing the background colour), wouldn't really make that

much of a difference

9 Summary

This section will include a summary of the project as a whole, including my personal reflections on the experience of working on it, and a critical review of how the project went

Words in text | 4,717

Calculated with the TeXCount web service
`http://app.uio.no/ifi/texcount/online.php`

References

[1] Pedro Albertos and Antonio Sala. Fuzzy logic controllers, advantages and drawbacks. *IEEE transactions on control system technology*, 1998.

[2] Kim Golombisky and Rebecca Hagen. *White Space is Not Your Enemy: A Beginner's Guide to Communicating Visually Through Graphic, Web & Multimedia Design*. CRC Press, 2013.

[3] Jonathan Lazar, Adam Jones, and Ben Shneiderman. Workplace user frustration with computers: An exploratory investigation of the causes and severity. *Behaviour & Information Technology*, 25(03):239–251, 2006.

[4] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[5] Jakob Nielsen and Rolf Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256. ACM, 1990.

[6] Shneiderman Ben Shneiderman and Catherine Plaisant. Designing the user interface 4 th edition. *ed: Pearson Addison Wesley, USA*, 2005.

[7] Christian Wagner, Simon Miller, and Jonathan M Garibaldi. A fuzzy toolbox for the r programming language. In *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, pages 1185–1192. IEEE, 2011.

[8] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[9] Lotfi A Zadeh. From computing with numbers to computing with words. from manipulation of measurements to manipulation of perceptions. *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, 46(1):105–119, 1999.

A User Evaluation Test Instructions

The purpose of this test is to construct the fuzzy logic tipper example, in three different software systems, so that the strengths and weaknesses of each of these systems can be identified. The tipper example is a simple fuzzy system that uses food quality (rated from 0 to 10) and service quality (rated from 0 to 10) as inputs, to determine how much of a tip should be left (between 0 and 30 percent).

Please follow the instructions below, and do not hesitate to ask for help if you are stuck (this is not a test!)

1. test