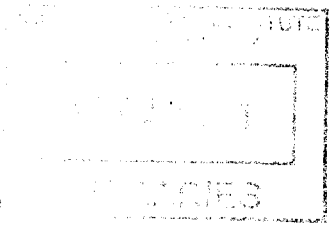
Improving Learnability of High Functionality User Interfaces Through Simplification Without Loss of Functionality



by

Tamara Fleisher

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology
June 2013

Author .....
Department of Electrical Engineering and Computer Science
June 2013

Certified By .....
Henry Lieberman
Professor of Media Arts and Sciences
Thesis Supervisor

Accepted By .....
Professor Dennis M. Freeman
Chairman
Masters of Engineering Thesis Committee

Improving Learnability of High Functionality User Interfaces Through Simplification Without Loss of Functionality

By **Tamara Fleisher**
Submitted to the **Department of Electrical Engineering and Computer Science**
On **May 27, 2013**
In Partial Fulfillment of the Requirements for the Degree of **Master of Engineering in Electrical Engineering and Computer Science**

Abstract

Justify is a high-functionality web application that helps users clearly state arguments and organize them in a meaningful structure, however due to its complicated user interface and abundance of point types, it is extremely difficult and frustrating to learn and use. Research shows that when presented with an abundance of choices, users are less likely to think through their decision rationally and more likely to regret any decision they make. This thesis proposes that reducing the amount of point types initially provided to the user, grouping them by similarity, and then utilizing "just in time" learning to dynamically introduce them to the user with interactive tutorials will dramatically increase the learnability of Justify, and thus also increase user retention and satisfaction. User testing indicated that these changes in Justify made it easier to learn and more enjoyable to use, when compared to both the older version of Justify and to alternative methods of decision making.

Supervisor: Henry Lieberman
Title: Professor of Media Arts and Sciences

Acknowledgements

I would like to thank the following people:

Henry Lieberman for his help and guidance as my thesis supervisor.
Christopher Fry and everyone else who worked on the Justify system.
Lily Berger, Elise Kuo, and Noel Morales for motivating me to get the thesis finished.
My parents for their unconditional love and support in helping me get where I am today.

TABLE OF CONTENTS

List of Figures

INTRODUCTION

High functionality interfaces often overwhelm new users with the amount of information they need to learn in order to begin using the system. This barrier to entry often discourages new users from using the system at all, as they do not want to put in the effort required to learn all aspects of the complicated system before they can properly begin to use it.

There are many psychology studies which show that the more choices a person has, the less likely they are to make any choice at all, even if an option exists that better meets their needs. This contributes to my hypothesis that users who are overwhelmed with the amount of features they must learn are less likely to put in the effort to learn the system, even if these features are ones that would help them in the long run.

Research Question

How can new users be taught to use a complicated and high functionality interface without being overwhelmed, eventually learning to take full advantage of all features of the system as they transition into advanced users?

Hypothesis

Limiting the number of features initially presented to a new user and then slowly revealing more complicated features as required, or as the user becomes more familiar with the system, will increase both the rate of learning and the adoption rate of high functionality interfaces.

Novel Innovation

Determining when to introduce new features, and in what order new features should be introduced, is the main focus of this thesis. This is an area in which very little research has been done. Most high functionality systems do not make much of an effort to simplify the initial user experience and learning process, and even those that do often simplify the interface by hiding the more complicated features, without providing an intelligent way to introduce these features to the user as they become more familiar with the system. This results in "experienced" users who still cannot use the more complicated features of the system (or perhaps don't even know these features exist), thus limiting the user from taking full advantage of all that the high functionality interface has to offer.

Background Research

The number of features initially provided to the user, specifically regarding options they need to choose between, affects how they learn to use the system and how much they enjoy using it. Studies have shown that people do like to have choices; giving people the ability to choose increases their intrinsic motivation, perceived control, task performance, and overall satisfaction and happiness [12]. This indicates that Justify has an advantage over the traditional methods of decision making, such as pro / con lists, email threads, or spreadsheets, as Justify allows them to choose point types from a drop-down menu instead of simply supplying information. However, as the number of options increases, so does the level of complexity of the decision itself. Although people are inherently attracted to having choices, when it comes to actually choosing from among a large number of options, people often find themselves paralyzed and unable to make a decision[11].

Angelika Dimoka, director of the Center for Neural Decision Making at Temple University, conducted a study of decision making by measuring subjects' brain activity with fMRI as they participated in combinatorial auctions. She found that as the information load increased, so did activity in the dorsolateral prefrontal cortex, a region behind the forehead that is responsible for decision making and control of emotions. But as the researchers gave the bidders more and more information, activity in the dorsolateral PFC suddenly fell off, indicating that they were no longer able to process the amount of choices offered to them, and were therefore not able to effectively make decisions any longer. Dimoka refers to this as "cognitive and information overload," stating that people start making stupid mistakes and bad choices because the portion of the brain responsible for smart decision making is essentially turned off. For the same reason, their frustration and anxiety soar, as the brain's emotion regions are no longer held in check by the PFC dorsolateral region. She found that when more information was presented to the user, they were more likely to either choose at random or make no decision at all [11].

Sheena Iyengar, a business professor at Columbia University, came to a similar conclusion, finding that people feel most confident in their decisions when they understand the available options and can comfortably compare and evaluate each one. She concluded that it's easiest to evaluate the options when there are only a few of them, and when they are easily distinguishable from each other. As the number of options increases, the evaluation process

can become overwhelming and intimidating, especially when it feels like making a choice requires expert information or skill [12].

In addition to the cognitive overload of considering an abundance of options, people feel the need to make the right decision and often feel as though they must justify their decisions, both to themselves and to others. In abundant-choice situations, people become unsure of themselves as they grapple with the burden of judging the differences between good and bad choices. People always fear making a wrong decision, which can lead to feelings of regret. People are particularly averse to the experience of regret.

Implications for User Interface Design

Offering extensive choice may work when users are experts in the domain or have extensive knowledge of the system. However, whenever it's possible to limit the number of options, especially for new users, this should be the first plan of action. Selecting an option from a limited choice set leads to better performance than selecting that same option from an extensive choice set.

Research shows that, when choosing from a limited number of options, people feel more confident in choosing and more satisfied with their choice once they make it. Plus, they are subsequently more likely to want to make a choice again. Cognitive research suggests that people are able to keep track of a maximum of only six options at a time, but this number does change depending on a particular audience. Because of this, research and user testing is important to determine the ideal number of options.

When a decision feels too complex, people either don't decide at all, or they employ ways of simplifying their decision-making process. Unfortunately, the simplification strategies they use don't necessarily yield the best decision outcomes. Thus it is the responsibility of the designer to limit choices in such a way that the users will be able to select options without relying on their own simplification strategies.

The Justify Brief

The ultimate goal of Justify is to provide an online environment that promotes rational deliberation by making complex arguments more understandable and manageable. To accomplish this, the system leverages human reasoning capabilities by structuring the points of

a discussion into a hierarchy, aiding users in understanding the issues and reaching valid assessments. It provides a language to isolate and record points, classify them into types, organize them with respect to each other, determine assessments automatically, and summarize results.

What's the "Point"?

A major concept in the Justify language is that every thought in a discussion can be expresses as a "point". Most points have 3 fundamental elements: a title, a type, and an assessment.

Title The title is a description of the point, entered by the user. It is restricted to one line of text, generally too small to contain more than one idea. If the user must elaborate, they can add a description with more text to go along with the title. Or, preferably, they can add clarifying or supporting sub-points. The title of a point helps other users easily scan through a discussion and quickly see the main ideas being presented.

Type A user cannot create a point without specifying a type from a list of predetermined point types. In addition, each point type has possible sub-types, which the user can select to further classify the type of the point. In total, there are approximately 150 different possible point types to chose from, allowing the user to express questions, answers, background information, support, opposition, votes, math, and more. A full list of available points and sub-points can be found in the appendix.

Requiring the user to specify a point type encourages good user-behavior by making it impossible to include two different ideas within a single point, thus simplifying the process of critiquing a point. It also ensures that users are never left confused about the intention of the point; it is clear weather it was meant to support, deny, or clarify a previous statement. In addition to facilitating reasoning for humans, the type of a point aids the Justify program in its auto-summarization and in determining an assessment for the point.

Assessment Each point type has an assessment, specifying what the point concludes to. The type of assessment is determined by the type of the point itself; a pro or con question will have an assessment of 'pro' or 'con', while a number point type will have a number as its assessment. Each point type has a specified algorithm to compute the assessment based on the type of point it is and the assessments of its sub-points.

Assessments may also be 'undecided' if there is not enough evidence to draw a valid conclusion.

Other Information Each point also contains information such as the author of the point, the date it was created, and a unique identification number. In addition, depending on the type of point, there may be 'essential attributes' such as a number, source, or reference point identification number. If applicable, the relevant data must be entered in order to add the point.

Hierarchical Structure

Arguments in Justify are framed in a tree-structure rather than in the typical sequential debate format. Any position in an argument can have its pros and cons, which in turn can be argued for or against; any position can be defended or refuted in a variety of ways. The traditional way of linearizing a tree structure of text is the outliner: each node of the tree is summarized as a line of text. Nodes at greater depths are indented. Each node can be expanded from a single-line summary to the full text of that node. Justify uses the outliner as its core interface metaphor.

This structure requires that each point be inserted in a particular place in the hierarchy. Users cannot make a point with no context, thus in Justify it is always obvious what each point is relevant to. Sub-points can add detail, answer questions, contradict, and / or provide rationale for the original point.

APPLICATION TO JUSTIFY

Justify is currently difficult to learn and inefficient to use. Fundamentally, it is a programming language and not an application. Most users, however, just want a simple interface to interact with and don't particularly care about the underlying language.

During initial user tests, the number one complaint regarding Justify was the difficulty new users faced learning the system. Specifically, users stated that the number of points was overwhelming, and they had difficulty understanding what each point type means and when to use different point types. If my hypothesis is proven correct, this can be applied to Justify by limiting the initial set of point types presented to the user and slowly introducing new point types when deemed appropriate. This will increase the speed at which new users can learn to use Justify and ultimately increase the overall adoption rate of the system.

Design Method

This thesis focusses on how feature set is introduced to new users, providing a solution for the problem of high-functionality, high-complexity interfaces that works for Justify but can also be adapted to other applications outside of the collaborative debate and decision making space.

The first step was to determine which features are absolutely necessary for new users to understand -- the minimum feature set required for the average user to complete common simple tasks in the system. This can be determined through user testing; by observing how users interact with the current system (the high functionality interface with all features revealed) and noting what features are used most often. In addition, by limiting the feature set (ideally by removing the features which are more complicated to learn and the features which are used less frequently) and noting if users have more or less difficulty completing simple tasks, features can either be added or removed from the minimum feature set. The system should be functional with only the minimum feature set, and users should be able to accomplish common simple tasks within the system.

Initial Feature Set

Four users were observed using Justify to determine an initial feature set. The first two users were provided with Justify as it currently stands, with all point types included. Based on the point types that they chose, a trial initial feature set was created. The remaining two users were limited in their point type options to this initial feature set, in order to determine if the limited options would increase the difficulty of the task by being overly restrictive, or if the task would become simpler without all the extra (possibly unnecessary) options to chose from. Each user was given three tasks, chosen because they encompass a relatively wide range of use cases, while not requiring any features too complicated for a new user. Detailed descriptions of each task and the subjects' responses can be found in the 'Evaluation' section below. During testing, all users were asked to explain their thought process aloud, and both their actions and commentary were taken into account. They also completed a post-test survey to provide additional information.

For the first task the users were given five options to decide between, and were provided with a lot of general details and qualitative information to consider, but no specific numbers. When the users entered the title of their first point, something along the lines of 'which job should I accept?', the type was automatically set to 'question', with no subtype selected.

The initial two subjects selected only the points "question.one_answer," "question," "idea," "pro," and "con." However, since the use of "question.one_answer" resulted in much confusion for the one user who tried it, the second two subjects were only given the options of "question," "idea," "pro," and "con," and were able to complete the tasks without any issues.

The second task was left very open ended, to see what users would think to enter if not prompted. This task resulted in the same initial feature set as task 1: "question," "idea," "pro," and "con."

The third task required the users to answer a simple "yes or no" question, but a lot of numbers were provided in the scenario, in order to determine if any of the "math" points should be included in the initial feature set. Despite this, a "math" type point was only used once, by one user, thus is was not included in the initial feature set.

Based on the user tests, and in accordance with the research indicating that six choices is an ideal number in terms of user satisfaction [12], the initial feature set consists of:

- Discussion
- Question
- Question.Pro_or_Con
- Pro
- Con
- Idea

Figure 1.1: List of Initial Point Types

New users are initially presented with only these features, allowing users to begin to interact with the system without being overwhelmed by more complicated features that aren't strictly necessary to use or understand. Once a user becomes familiar with the system, new features can be introduced (and explained), expanding the user's understanding of the system and allowing them to accomplish more with the system. In addition, if a user attempts to accomplish some task which requires a feature included in the system but not yet available to the user, this feature is revealed and explained to the user.

Full Advanced Feature Set

The advanced feature set was determined by discussing the tradeoffs between the complexity of having more point types to learn and the reward of being able to accomplish more tasks during a usability discussion with four new users and three experienced users.

A full advanced feature set was then determined by thinking through the major use cases of the system, and making sure each case would be possible to complete without the need for more point types. The number of points included in the full feature set is fewer than the total number points available in Justify, as the remaining points are so rarely used, and even advanced users found it difficult to deal with the shear number of points available. The remaining points can be accessed through a special menu for the rare very advanced user who needs them, but will not be introduced to the user through the learning process, and will not appear in the main point menu.

Grouping Features for Optimal Learning

The full advanced feature set of Justify points was divided into ten groups, as listed in Figure 2.1 below.

1. If.most (including concept of value)
2. Math.number, Math.[operation]
3. If.[equal, less_than, less_than_or_equal, more_than, more_than_or_equal]
4. Vote, Ballot
5. Folder.agenda
6. Folder.to_do, Idea.action, Idea.action_done
7. Fix.[reason]
8. Moot.[reason]
9. Question.clarification
10. Meta.use_assessment

Figure 2.1: Grouping of Point Types

The point types were then organized into a graph, indicating a path of most common progression through the different point types, as seen in Figure 2.2.

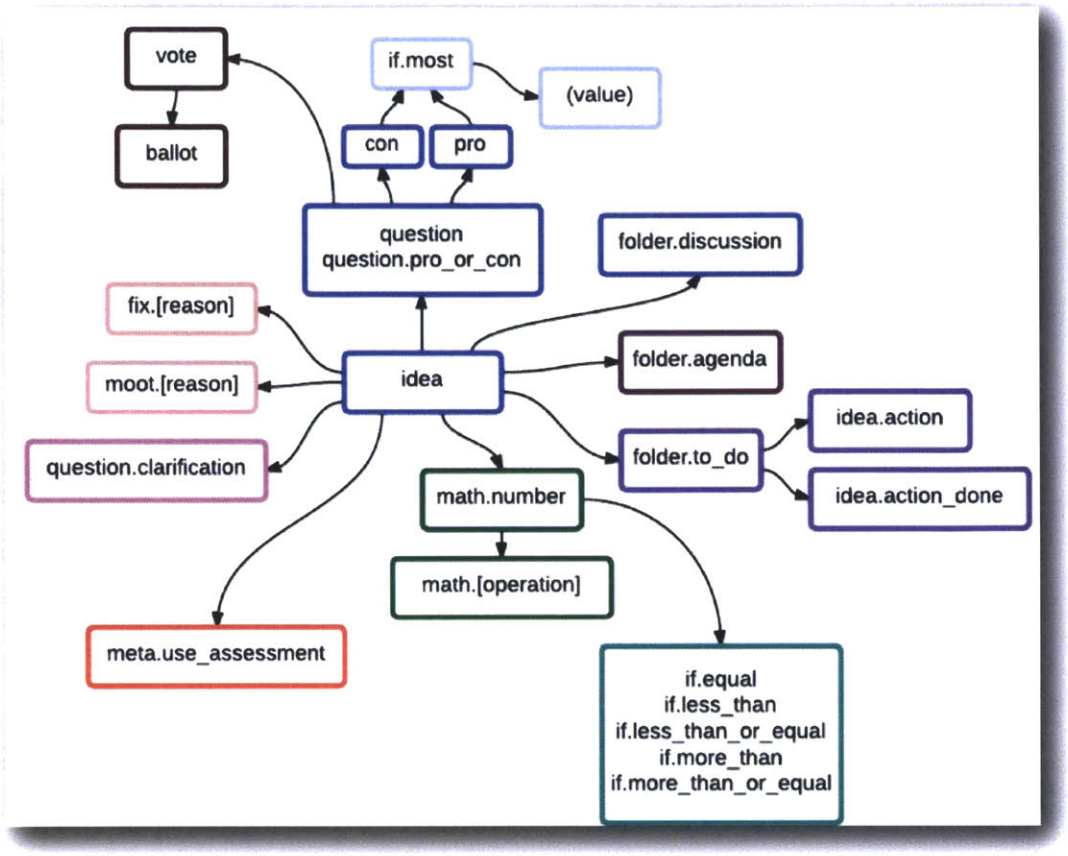


Figure 2.2: Point Relation Graph

In this graph, the center blue points represent the initial feature set. The coloring represents groups of point types that should be introduced together. The graph details how features relate to each other, with connections between features that are often used in similar scenarios, where the direction of the connection indicates what previous knowledge is assumed before a new feature can be introduced.

If a user ever indicates that they are moving in the direction of one of these groupings, all point types in the grouping will be added to the user's personalized point menu at once. For example, if a user enters the title "how much money should I be saving for retirement?", the "how much" will alert the system that the user is likely going to need "math" point types, and thus all the point types related to 'math' (the green points in the graph) will be introduced. Related points are introduced at once to ensure that the user never gets frustrated by the lack of available point types in their personalized menu, since the use of one point type in a grouping indicated they will likely need to use other point types in the grouping in the near future.

This method teaches users about the new point types as they learn the system, utilizing "just in time" learning without overwhelming them from the beginning. When a new point type is added to the user's personalized point type menu, a very quick tutorial appears to inform them of its existence of the point type and to teach them what the point type means and how to use it.

Introducing New Features

Determining when a user is ready to learn to use a new features is key in making this method work correctly. It is done by analyzing how the user is currently using the system; by keeping a tally of which point types they are using, how often they replace the suggested point type provided by the system, how often they click on the help menu to find more information about points already available to them, and how often they click on the help menu to find information regarding points not yet in their personalized menu. This information is kept in a new class, as detailed in Figure 3.1 below.

```
structure learning_info
    set string available_points
    map string:integer point_types_used
    integer total_points_made
    integer point_replacements_made
    map string:integer available_point_help
    integer total_available_point_help
    map string:integer new_point_help
    integer total_new_point_help
```

Figure 3.1: Storing Learning Information

The information stored in the `learning_info` structure is used to determine when new point types should be introduces, as well as if the tutorial about features already introduced should be re-displayed. The `available_points` set is simply a set of all point types currently included in the user's personalized menu. The map is a mapping of point type to the amount of times it has been used. The integers `total_points_made`, `point_replacements_made`, `total_available_point_help`, and `total_new_point_help`, as their names suggest, are simply counts of all points entered by the user, the number of times they've replaced the suggested point type with one of their own choosing, and the number of times they've accessed the help menu to search for points included in their personalized menu and points not yet accessible to

them, respectively. Although these integers can be computed in other ways, it is easier to simply store their values in the `learning_info` structure, as they are used so often.

If the user is taking advantage of all features offered to them without difficulty, as determined by the amount of points from their `available_points` set covered by the mapping of `point_types_used`, as well as the ratio of `total_points_made` to `total_available_point_help`, they have likely managed to learn all points in the current feature set and might be ready to be introduced to additional features. In addition, if they are attempting to accomplish some task that requires a hidden point type, as determined by applying basic natural language processing to their point titles as well as keeping track of the point types in `new_point_help`, it is likely that the new feature should be presented to them.

If the user is looking for information about features they currently have access to, as determined by `total_available_point_help`, they are likely not ready to be introduced to new points, and could probably benefit from another tutorial about the features they are having trouble with. Determining which specific tutorial should be displayed is done by iterating through `available_point_help` to see which specific points they're having trouble with.

Features should not be revealed too slowly to users, as this may frustrate them by unnecessarily limiting what they can accomplish with the system. Thus it is important to predict when a user should be introduced to a new feature before they realize they need it. The features that they will likely need next must be predicted by taking into account the specific point types, and their respective counts, found in both `point_types_used` and `new_point_help`. By comparing them with the graph of how features relate to each other, the system can predict the direction through the graph that the user is likely heading and provide the relevant point types before the user themselves even know they are going to need them.

By analyzing how quickly the specific user learns new features (measured by the number of times they access the help menu for information about new features and the frequency with which they utilize these feature correctly), we can either slow down or speed up the rate at which new features are presented.

This allows users to eventually learn to use the entire high functionality interface, including all complicated features, without being initially overwhelmed by the amount to learn. This leads to a quicker and more streamlined learning process, and a higher engagement and retention rate for the system overall.

Alternative Approaches to Simplification

There are a number of tools that utilize strategies similar to Justify in order to support large-scale online collaborative debate and decision making. While they all attempt to solve a problem similar to Justify, they use different interface patterns to do so, with varying degrees of success regarding usability and learnability.

Examples of these interfaces include Debatepedia[10], Debategraph[8], Cohere[7], Truthmapping[9], and Parmenides[2], ArgDF[15], gIBIS[3], and MAgtALO[18]. The major design decisions made in these applications which relate to Justify include the use of wiki-based technology, the utilization of structured argument constructs, the hierarchical structure of debates, the use of various methods to visualize arguments or debates, and the existence of search functionality.

Collaborative Environment

Most tools utilize 'wiki' technology to allow for easy collaboration between users. Debategraph even includes an activity stream to involve participation between users. As web applications, a major goal of these tools is to engage as many users as possible, thus in most cases any user can begin a debate, add arguments to existing debates, or simply participate by reading through posted discussions.

Structured Argument Constructs

In Debatepedia, arguments are made in free-text form. There is no specific structure the user must follow, but each point must be classified as a pro or con. Other tools provide a more structured environment by explicitly representing different argument constructs. gIBIS utilizes the Issue Based Information System (IBIS) model of decision making, as outlined in Figure 4.1 below.

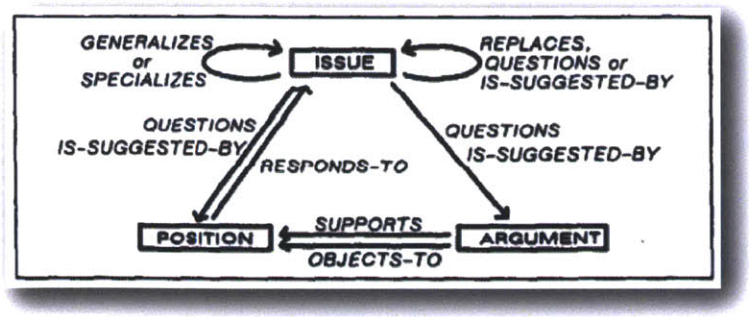


Figure 4.1: Argument Structure of IBIS [3]

Debategraph allows users to enter issues open to debate; the position taken; and arguments attacking or supporting these positions. In Cohere, users create ideas, which play roles within an association. Ideas linked by different types of connections, and each connection is classified as either positive or negative. Truthmapping exhibits a fairly advanced argument structure, distinguishing between the premises and conclusions of each argument, allowing users to critique any argument, and allowing the creator of an argument to add rebuttals to critiques, and distinguishing between deductive and inductive arguments. Arguments can also contain hyperlinks to other websites or to premises or conclusions of other arguments in the system. ArgDF utilizes the Argument Interchange Format (AIF), while also allowing users to extend the underlying ontology by adding new argument schemes.

Visualization Tools

There are multiple methods of visualizing arguments and debates, and many existing tools utilize a combination of methods for users to visualize and navigate the debates.

Discussion View MAgtALO uses the familiar view of a chat interface to display arguments. Users are able to simply enter arguments by typing text into the chat box.

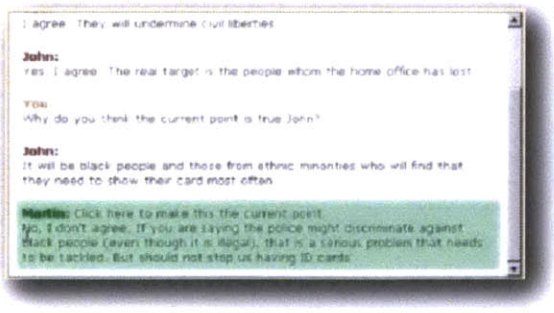


Figure 5.1.1: Screenshot of MAgtALO's Discussion View

While this is simple and easy to learn, it does not take advantage of the non-linear structure of debates, nor does it help users visualize the debate by viewing points within their context of the hierarchy. Thus the added simplification comes at the expense of usability, which was found unacceptable for Justify.

Outliner Interface Although offline debates taking place on the internet are typically conducted in a linear fashion, most of the observed applications structure arguments into a hierarchy, taking advantage of the non-linear structure that is afforded by the computer. Because of this, the outliner interface is a common tool used to visualize arguments and navigate through various different responses. This method was found to not add complexity to the system, and is thus employed by Justify.

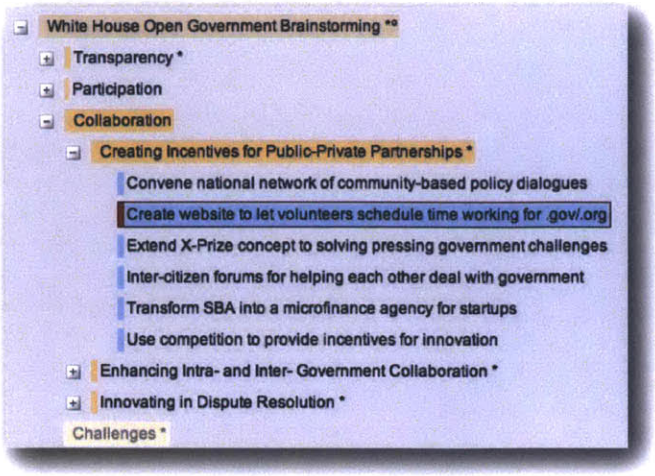


Figure 5.2.1: Screenshot of Debategraph's Outliner View

Details Pane Many tools offer a details pane to easily view a summary of the arguments provided in plain text format. For some sites, such Debatepedia, this is the primary way to visualize the debate. For others, the details pane is an optional visualization method, in addition some other method.

Argument Network Structural argument constructs can be related to each other through an argument network, allowing users to easily visualize elements of the argument and their relation to each other.

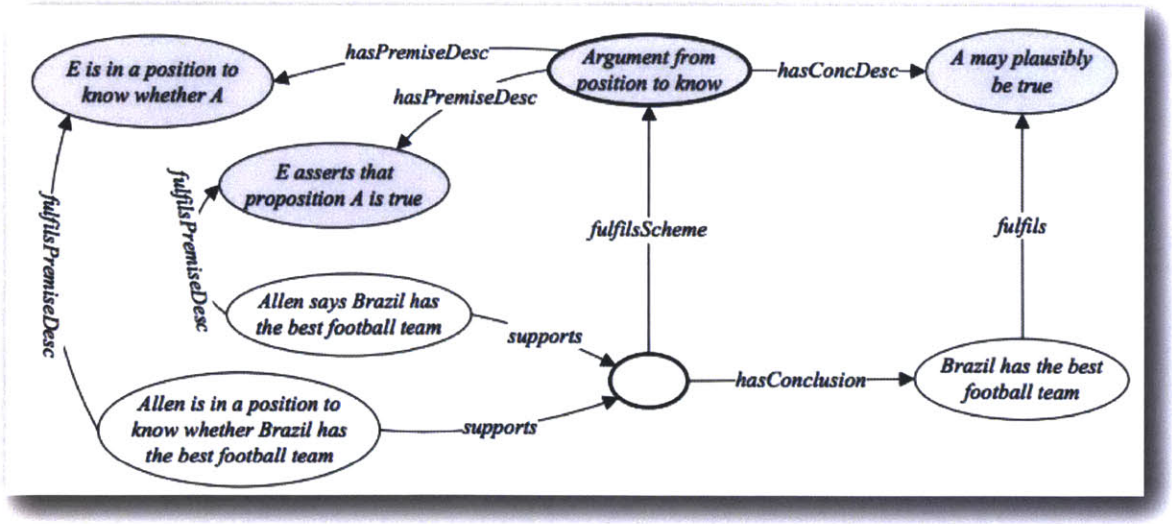


Figure 5.3.1: Argument Network of AIF Abstract Model [17]

Figure 5.3.1 emphasizes the relationship between instances of arguments and scheme components in AIF in the form of an argument network visualization, as used by the ArgDF system.

Figure 5.3.2 below is an example of the argument network implemented in Debategraph's user interface, allowing users of the system to visualize debates, with a focus on the relationship between the issues, positions, arguments, and repertoire.

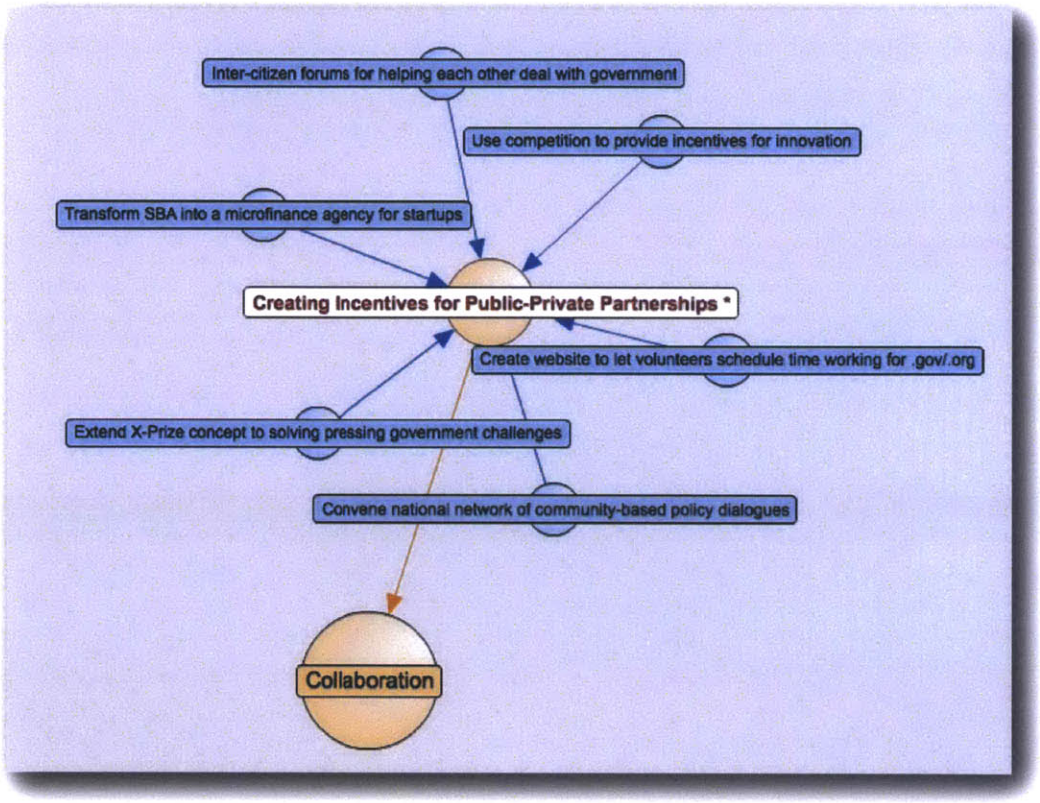


Figure 5.3.2: Screenshot of Debagraph's Argument Network View

Hierarchical Structure

Although debates are typically conducted in a linear fashion, most of the observed applications structure arguments in a hierarchical fashion. This aids in creating a structure that is easier for users to read and understand.

Search Functionality

Debatepedia, Truthmapping, and ArgDF provide basic keyword search functionality over all arguments, however none of the existing tools have more complex search features.

While Justify does employ many of these design decisions, none of them are able to adequately solve the problem of complexity in the system. While they manage to make improvements, they are still too complicated for the average user to easily learn and use. None use the approach of partitioning the command space and dynamically introducing new features using the "just in time" learning approach, as outlined in this thesis. Because of this, all the alternative systems suffer from the same problem of complexity creating a system that is too complicated for the average user.

EVALUATION

A new user's ability to learn how to use Justify was measured by observing how quickly and accurately they were able to input a discussion and by considering their responses to the post-test questionnaire. Once a user had completed at least three tasks, they were considered a novice user and were given a slightly more difficult task to accomplish; one which required (or was made easier by) point types not included in the initial feature set. This triggered a 'learning event' where a new set of features was introduced, allowing them to complete the task.

Since this thesis is attempting to improve upon a system currently in use (Justify), the testing process used Justify in its previous form (referred to as Justify 1.0) as the control, and tested it against a version of Justify with the changes implemented (referred to as Justify 2.0). Depending on the task, email, spreadsheets, and pro/con lists were also used as controls, to determine the effectiveness and ease of use of methods currently employed by users compared to Justify. During testing, all users were asked to explain their thought process aloud, and both their actions and commentary were taken into account. They also completed a post-test survey to provide additional information.

Many different parameters were adjusted based on the results, such as what the initial feature set is, how quickly new features are revealed, and what actions lead to which features being revealed. Thus there were multiple tests done throughout the process in an attempt to find the ideal combination of all these parameters.

Determining the Initial Feature Set

The first set of user tests, involving four users, were conducted in order to determine an initial feature set for Justify. The first two users were provided with Justify 1.0, and based on the point types that they chose, a trial initial feature set was created. The remaining two

users were limited in their point type options to this initial feature set, in order to determine if the limited options would increase the difficulty of the task by being overly restrictive, or if the task would become simpler without all the extra (possibly unnecessary) options to chose from. Each user was given three tasks, chosen because they encompass a relatively wide range of use cases, while not requiring any features too complicated for a new user.

Task 1 - Picking a Job

> *"You are about to graduate from MIT with a degree in computer science , and are are trying to decide what to do next year. Your family lives in NYC, and while you'd like to live near them, its not a deal breaker to move across the country. You have interned at Google in Mountain View as a software engineer for the past two summers and really enjoyed it. You've always been interested in startups, but you find the idea of starting your own company a little scary. You've been offered a software engineering position at Google, an associate product management position at Google, a software engineering position at Apple, and a position as the tenth engineer at a new startup in Boston. While Google and Apple are offering more money, you would get equity in the startup and there is the potential of becoming very rich. You've also been approached by a friend about co-founding a company with him. Use Justify to figure out which job you should take."*

The first user attempted to select a subtype by reading through the available options, and initially settled on "question.one_answer," but decided against it after looking the help pane, stating that "the explanation here is really long and confusing... I don't think this is what I want." After spending an extended period of time looking through different point types and attempting to read the documentation, getting more and more frustrated and complaining that "there's too much jargon. I don't understand, why can't I just answer the question. I don't know how these points work... this is too confusing...", he finally returned to "question.one_answer" and moved on. After entering all his points, however, he was frustrated that it didn't give him a clear conclusion, saying that he "wanted one answer, why does it still say undecided? The only option that has all pro sub-points is Google software engineer, shouldn't that be the assessment? Does it not even tell me what to pick??" This example can be seen in Figure 6.1.1 below.

Figure 6.1.1: Justify Discussion - User 1, Task 1

The second user, on the other hand, simply accepted the auto-completion of type "question," with no subtype. Though it still didn't provide the user with the answer once completed, as shown in Figure 6.1.2, he was less frustrated because he hadn't spent as much time deciding which point type to use.

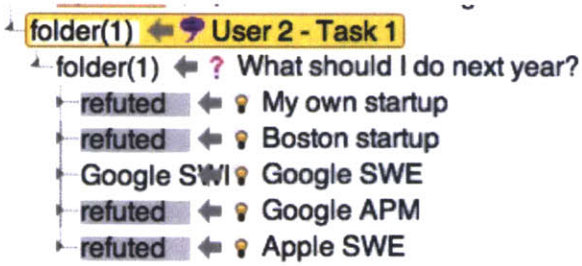


Figure 6.1.2: Justify Discussion - User 2, Task 1

Based on the frustration expressed with "question.one_answer" by the first user and the second user's ability to complete the task easily without a subtype, "question.one_answer" was not included in the trial initial feature set. The other types selected by the first two users were "idea," "pro," and "con." When the remaining two users were given the limited initial feature set they were able to complete the task in less time, and reported higher overall satisfaction with the system. None of them mentioned having problems with the lack of options available, either during the test or in the post-test questionnaire.

Task 2 - Selecting a Final Project

> *"You're currently taking 6.170 (laboratory in software engineering), and have just been assigned your final project. You are to design and develop a website; the theme is education, but you can do whatever you want as long as it relates to the theme. You have one month to complete the project. The final product must not only work, but must also be easy to use and look good. You will also have to convince the professor that you could attract users. Use Justify to figure out what the focus of your site will be.*
>
> *[Note: If you can't think of ideas, consider existing sites related to education: Khan Academy, EdX, Piazza, Stellar, Wikipedia]"*

The results of this task were similar to those of task 1, though on average fewer points were added (likely because it was more difficult to think of points without being given as much information about the scenario). Thus the initial feature set was not changed by this task.

Task 3 - Funding a Startup

> *"You are a venture capitalist working at a major VC firm in Silicon Valley, and are trying to decide if you should fund a startup called Google, a new search engine developed by two PhD students from Stanford. Google is 6 months old, and currently has 25 employees. While they have no profits, they do have over 100,000 users, and are trying to figure out how to make money. They are asking for $500,000 for a 20% stake in the company, thus valuing the company at $2.5 million. While the employees all seem very smart, they are also very young and inexperienced. You're not sure if they can win over the market while also figuring out how to monetize their site. Use Justify to decide if you should invest or not."*

After entering a question title along the lines of "Should I fund Google," Justify automatically selected the "question.pro_or_con" point, and all users simply accepted that.

Despite the amount of numbers in the task description, the first user only used the "math.number" type once. In addition, he did not use any of the math operations, thus his point could have been made just as well by simply labeling the point as "pro" or "con" instead of "math.number," as depicted in Figure 6.1.3 below.

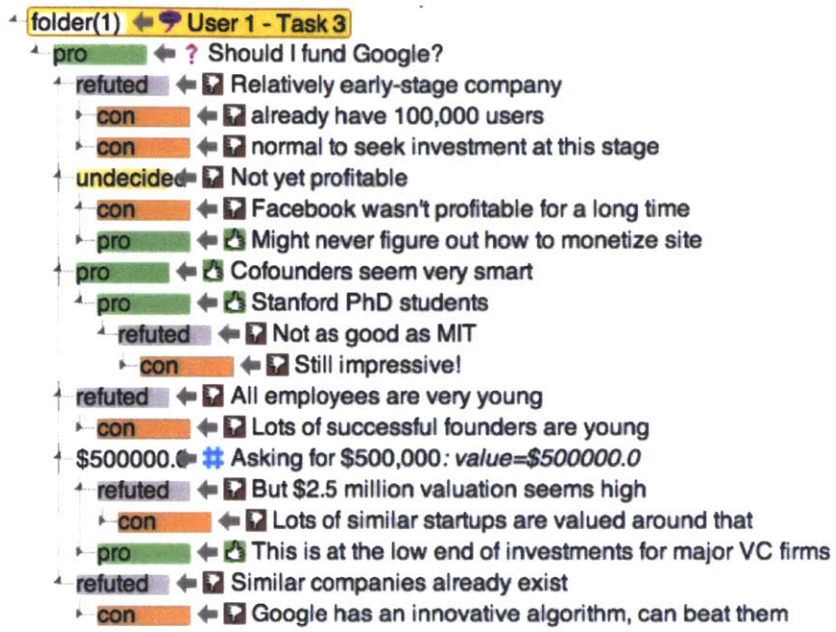


Figure 6.1.3: Justify Discussion - User 1, Task 3

He did spend a lot of time looking through the available type options each time he went to enter a new point, however, and this seemed to increase his frustration each time he attempted to enter a new point. In the post-test questionnaire he noted that "there are way too many types that aren't necessary. It took me too long to figure out what types and sub-types I should pick, and the documentation for most of them was really confusing". The second user, on the other hand, didn't even look at the 'math' type and was satisfied simply adding 'pro' and 'con' points. Because of the first users's overall frustration and the lack of benefits seen by using the 'math.number' point type, 'math' was not included in the feature set provided to the remaining two users. Even without the 'math' points, they had no trouble entering their arguments, and again spent less time and were more satisfied with the system overall.

Multiple users expressed satisfaction that an answer to their initial question in Task 3 was clearly displayed, as seen above in Figure 6.1.3 contrasting it with the more confusing assessments of "undecided" or "folder(2)" from the earlier tasks. This behavior was due to the subtype "pro_or_con" being selected in task 3, whereas in the earlier tasks most users had selected "question" with no subtype. It seemed as though the subtype "pro_or_con" added value without adding much (if any) complexity, therefore it was included in the initial feature set.

General Usability Testing

After all parameters were finalized, user tests were performed with eight new users. All users were given a general briefing on Justify, how it works, and what it is used for, but had not seen the actual system user interface until the test began. They were all given a written copy of the exact same tasks, as detailed below.

The users tested were all MIT students, with various amounts of programming knowledge and experience with different methods of decision making Figure 6.2.1 below details their basic information.

	Sex	Major	Year	Programming Experience	Usual Method of Decision Making
User 1	Male	Computer Science	4	5 years Python 3 years Java	Pro / Con list
User 2	Female	Brain and Cognitive Science	3	None	Spreadsheet
User 3	Female	Environmental Engineering	4	1 semester Java	None
User 4	Male	Chemical Engineering	2	1 year Matlab	Discuss with friends
User 5	Male	Biological Engineering	3	1 year Python	Pro / Con list
User 6	Female	Mechanical Engineering	4	1 semester Java 1 year Assembly	None
User 7	Male	Electrical Engineering	3	4 years C 2 years Python	Discuss with friends
User 8	Female	Architecture	3	None	Pro / Con list

Figure 6.2.1: User Demographics

The first four users completed all tasks using Justify 1.0, and the second four completed all tasks using Justify 2.0 (they were divided so that each group would have approximately equal programming experience. All users completed the tasks in the same order, such that the more difficult tasks were attempted by more experienced users. For the group discussion, users using the same interface were grouped together. All eight subjects

completed the alternative method in each case, and it was randomly determined if they would use Justify or the other method first. After each user completed a task they were given a post-test questionnaire to complete, where they were asked to answer each question with a ranking of 1 through 5, representing the options "Strongly Agree," "Agree," "Neutral," "Disagree," and "Strongly Disagree." Tasks 1 through 3 were identical to the tasks described in the "Determining an Initial Feature Set" subsection above. Figures 6.3.1 through 6.3.6 detail the average response to each question, for each method tested.

Task 1 - Picking a Job

	Justify 1.0	Justify 2.0	Pro / Con List
I reached an ideal conclusion	3.75	2	4
It was easy to enter all the information I wanted	5	2	1
I was confused by the interface I was using	1	3	5
Task would have been easier with additional functionality	5	4.25	2.5
It is easy to see how I reached the conclusion	2.5	2	2.5
I would spend more time learning how to use this method	4.25	2	5
I would use this method in the future for a similar decision	5	2	2

Figure 6.3.1: Rating Summary - Task 1

This task was given to new users. The results show that subjects using Justify 2.0 had much less difficulty using the system and were much more likely to use it in the future, when compared to subjects using Justify 1.0. Subjects found the pro / con list easiest in terms of entering information and found it least confusing, however they were less confident about their conclusions, demonstrating the potential benefit of using Justify. The most important question, would they use it again in the future, found that users were strongly against using Justify 1.0 again, but were equally as likely to use Justify 2.0 as they

were to use a simple pro / con list. This suggests that Justify 2.0 is an improvement over Justify 1.0, but perhaps for such a simple task neither are truly necessary. This task did not trigger "learning events" for any subjects using Justify 2.0, meaning no new point types were introduced by the system, yet overall they did not find that functionality was lacking. Subjects using Justify 1.0 only ended up using point types from the initial feature set, yet it took them, on average, longer to determine which point type to use. No subject using Justify 2.0 accessed the help menu at all during this task, while two of the four subjects using Justify 1.0 spent a significant amount of time reading through the help menu or trying to figure out the differences between different point types, one user finally stating that "I the documentation is way too long, I don't want to have to read through all of this."

Task 2 - Selecting a Final Project

	Justify 1.0	Justify 2.0	Email
I reached an ideal conclusion	3.75	2.5	4.25
It was easy to enter all the information I wanted	4	1.5	1.25
I was confused by the interface I was using	1.5	3.5	5
Task would have been easier with additional functionality	5	4.5	1.25
It is easy to see how I reached the conclusion	2	2	5
I would spend more time learning how to use this method	4	1.5	4.75
I would use this method in the future for a similar decision	4	2	2

Figure 6.3.2: Rating Summary - Task 2

The results from this task were very similar to those from task 1. Overall satisfaction for both Justify 1.0 and Justify 2.0 increased, though this is likely due to the fact that it was the subjects second time interacting with the system, thus they had more experience and knew

what to expect. Again, no "learning events" were triggered, and subjects using Justify 1.0 only used point types in the initial feature set. However, one subject using Justify 1.0 spent much more time on this task, reading through the different point types in the help menu, and finally stating that "there are too many points here, and I don't understand what most of them do... I guess if the decision was more complicated I would have to use them, but for now I'll ignore them... hopefully I don't need them."

Task 3 - Funding a Startup

	Justify 1.0	Justify 2.0	Spreadsheet
I reached an ideal conclusion	3	2	4
It was easy to enter all the information I wanted	3.5	2	4.25
I was confused by the interface I was using	2	4	3
Task would have been easier with additional functionality	5	5	1.5
It is easy to see how I reached the conclusion	1.75	1.5	4.5
I would spend more time learning how to use this method	3	1.5	4
I would use this method in the future for a similar decision	4	1.25	4.25

Figure 6.3.3: Rating Summary - Task 3

Task 4 - Balancing a Budget

"You are a recently married 35-year-old, trying to figure out if your spouse needs to get a job, or if your salary alone is enough to support both of you. You make $6,000 / month in your current job, which means you pay 30% income tax. You own a $600,000 house with a mortgage of $300,000 (lets assume no interest, for simplicity) and neither of you have any student debt. You do not own a car and are able to walk most places, but occasionally take public transportation or a taxi if required. You hope to retire when you're 65, but so far you have not put any money towards retirement savings (again, assume no interest on savings and no inflation). Luckily, your company just announced that they will match 50% of whatever you contribute to your (and your spouse's) retirement saving plan each month.

Ideally, you'd also like to donate some money to MIT, your alma matter. You cook most of your meals at home, but enjoy going to restaurants occasionally with friends. You also have a bad habit of buying the latest and most expensive tech gadgets. Use Justify to create a monthly budget and determine if your spouse needs to get a job."

	Justify 1.0	Justify 2.0	Spreadsheet
I reached an ideal conclusion	1.5	1	4.5
It was easy to enter all the information I wanted	1.5	1.5	2.75
I was confused by the interface I was using	3	4	2
Task would have been easier with additional functionality	5	4.75	2.5
It is easy to see how I reached the conclusion	2	1.5	2.5
I would spend more time learning how to use this method	2.75	1.5	4.5
I would use this method in the future for a similar decision	2.75	1.25	2.5

Figure 6.3.4: Rating Summary - Task 4

Task 5 - Product Analysis

"You work at Apple as the product development team lead for the new iPhone 6, and Tim Cook wants to know how much profit you expect to make off of the release of the new iPhone 6 this September. To figure this out, you need to decide how much you plan to charge for the phone, how many units you expect to sell, and what the cost of production will be. Determining the cost of production will require you to finalize the list of features that the phone will include and what the cost of each feature will be (don't be afraid to consider different options here). Use Justify to determine the expected profit."

	Justify 1.0	Justify 2.0	Spreadsheet
I reached an ideal conclusion	3	2	3.75
It was easy to enter all the information I wanted	2	1	4

	Justify 1.0	Justify 2.0	Spreadsheet
I was confused by the interface I was using	3	4.5	3.5
Task would have been easier with additional functionality	5	5	2
It is easy to see how I reached the conclusion	2	2	2.5
I would spend more time learning how to use this method	2.5	1	4
I would use this method in the future for a similar decision	2.5	1	3

Figure 6.3.5: Rating Summary - Task 5

Task 6 - EMT Elections

"The four of you are members of MIT EMS, and three of you (as determined by me) are part of the executive team. You need to select a new chief of operations; any executive team member can be elected and you must all vote. To be considered, the candidate needs to be nominated by at least one person (candidates may nominate themselves). A nominated candidate may also reject a nomination if they wish. The chief of operations is responsible for maintaining the state of the ambulance, handling any repairs required, and making sure it remains in good condition throughout the semester. This position usually involves a commitment of at least 10 hours / week. Once nominations have been made, consider carefully who is best qualified for the position, using Justify to share your thoughts with the rest of the group."

	Justify 1.0	Justify 2.0	Email
I reached an ideal conclusion	1.5	1	2
It was easy to enter all the information I wanted	2	1.5	1
I was confused by the interface I was using	2	4.25	5
Task would have been easier with additional functionality	2.5	5	2

	Justify 1.0	Justify 2.0	Email
It is easy to see how I reached the conclusion	1	1	4.75
I would spend more time learning how to use this method	3	1	5
I would use this method in the future for a similar decision	2.5	1	4.25

Figure 6.3.6: Rating Summary - Task 6

Discussion

In order to more easily compare how the different methods compared to each other, the rating system for questions three and four was then reversed, such that a high number always reflects a negative response, and a low number always reflects a positive response to the system. Figure 6.4.1 shows the average rating for each task, for all three methods.

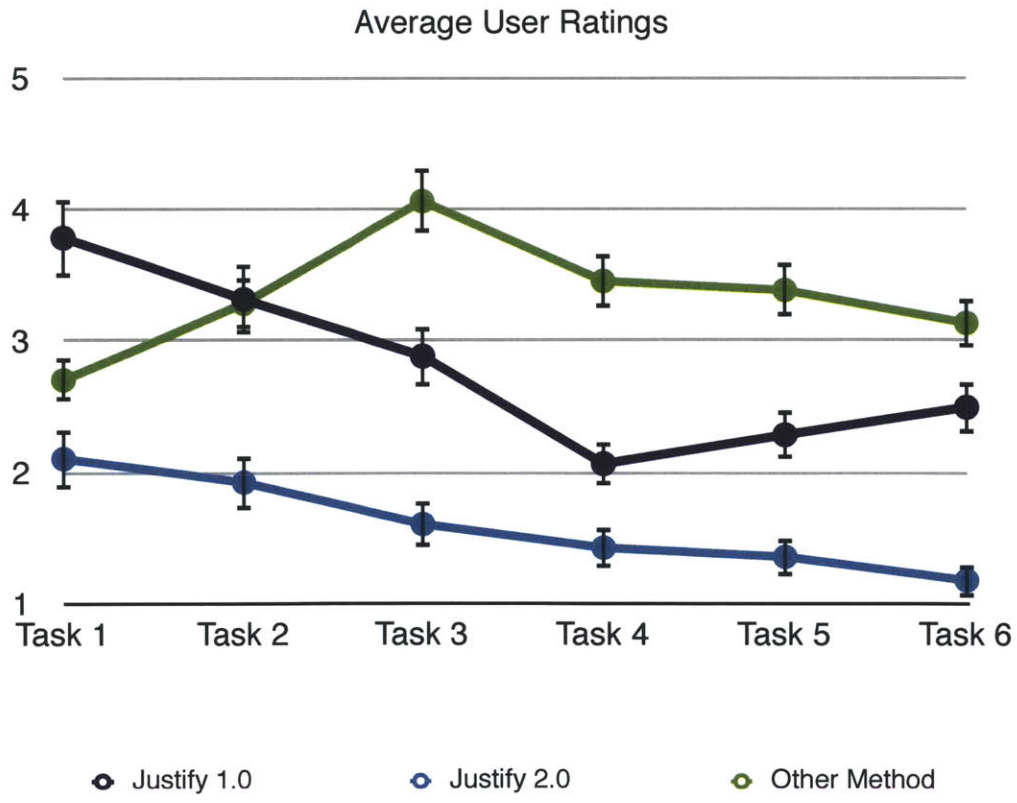


Figure 6.4.1: Average Ratings

As hypothesized, Justify 2.0 was consistently rated better than Justify 1.0, indicating that users appreciated the simplicity that the system afforded, and did not miss the hidden features. In the first few tasks, the alternative method was rated better than Justify 1.0, likely because the simplicity of the task did not require the use of Justify, thus users felt as though the effort of learning the new system was not worth the payoff received from the benefits the system provided. However, as tasks became more difficult, the rating of the other method dropped well below the rating of both Justify 1.0 and Justify 2.0, indicating that Justify does contribute positively overall to the decision making process, especially for more complicated decisions.

As new users continued to interact with the system, they became more comfortable and learned more about it, thus the ratings tended to improve. It is interesting to note that for task four and five, Justify 1.0's ratings became worse, while Justify 2.0's ratings improved. This is likely due to the fact that those tasks required new features that had not previously been used. In the case of Justify 2.0, when the system detected the need for these features (based on the title entered by the user, or the sections of the help menu that were accessed), a group of the necessary features were introduced and explained. However, in Justify 1.0, though the features were already accessible to the users, they had not previously used them and had therefore not learned how they worked, thus these tasks required much more effort as the users needed to look through the list of all the features, as they had no extra knowledge of which ones would be best for this task, without the help of the interactive tutorial introducing the new set of features.

In addition, while the first four tasks indicate a higher rate of learning for Justify 2.0 than for Justify 1.0, it doesn't take into account that many users would have stopped using the system before they were able to get the benefit of this learning had they not been participating in this trial. Since the initial ratings for Justify 1.0 were so poor, many users would have simply given up on the system after one trial, thus the fact that Justify 2.0 has an initial average rating of over two and a half points better is extremely significant.

Testing Limitations

Since all users progressed through the tasks in the same order, new users were always given simple tasks to complete, and only experienced users were introduced to the more complicated applications. Because of this, the tests do not show how new users would react to attempting complicated tasks without first having the benefit of learning how to use

the system while completing the simpler tasks. This is something that can be addressed in a more in-depth study of the system.

In addition, all participants in the study were MIT students, and most had at least some knowledge of programming, thus these responses do not necessarily reflect those of the general population. It is likely that the average user would have more difficult with the system than those tested in this study.

Conclusion

This thesis attempted to provide a method for improving the learnability of high-functionality interfaces without through simplification without loss of functionality. It proposed a method in which the amount of features initially provided to the user was limited. All features were then grouped by similarity, and "just in time" learning was utilized to dynamically introduce new features to the user as required, through simple tutorials. This method was applied to Justify, a high-functionality user interface with hundreds of different point types for users to learn. It was found that the new method dramatically increased the learnability and usability of the system, and thus increased user retention and satisfaction. These changes made Justify significantly outperform its old version as well as alternative decision making methods during user testing.

Future Work

In the future, it would be interesting to apply the ideas outlined in this thesis to other high-functionality user interfaces, to confirm the ability to universally apply the techniques for positive results. In addition, the process of determining when to introduce new features can be improved by implementing more advanced natural language processing as well as applying commonsense reasoning to the point titles entered by users. This will hopefully result in an even smoother and more intuitive introduction of new points, greatly eliminating the need for the user to ever search for point types in the help menu.

REFERENCES

1. Arango Aramburo, S., Castañeda Acevedo, J. A. and Olaya Morales, Y. (2012), Laboratory experiments in the system dynamics field. Syst. Dyn. Rev., 28: 94–106. doi: 10.1002/sdr.472
2. Atkinson, K., Bench-Capon, T. J. M. and McBurney, P.: 2006, PARMENIDES: facilitating deliberation in democracies, Artificial Intelligence and Law 14(4), 261–275.
3. Conklin, Jeff and Begeman, Michael L. 1988. gIBIS: A hypertext tool for exploratory policy discussion. In Proceedings of the 1988 ACM conference on Computer-supported cooperative work (CSCW '88). ACM, New York, NY, USA, 140-152.
4. Conklin, J., Selvin, A., Buckingham Shum, S. and Sierhuis, M. (2003) Facilitated Hypertext for Collective Sensemaking: 15 Years on from gIBIS. Keynote Address, Proceedings LAP'03: 8th International Working Conference on the Language-Action Perspective on Communication Modelling, (Eds.) H. Weigand, G. Goldkuhl and A. de Moor. Tilburg, The Netherlands, July 1-2, 2003. [www.uvt.nl/lap2003]
5. Doyle, J. A Truth Maintenance System. AI. Vol. 12. No 3, pp. 251–272. 1979.
6. Greg Newman, Don Zimmerman, Alycia Crall, Melinda Laituri, Jim Graham & Linda Stapel, (2010) User-friendly web mapping: lessons from a citizen science website. International Journal of Geographical Information Science 24:12, pages 1851-1869.
7. http://cohere.open.ac.uk
8. http://debategraph.org
9. http://truthmapping.com
10. http://wiki.idebate.org/index.php
11. Iyengar, Sheena, and Mark Lepper. "When Choice Is Demotivating: Can One Desire Too Much of a Good Thing?" Journal of Personality and Social Psychology, 2000, Vol. 79, No. 6.
12. Levav, Jonathan, Mark Heitmann, Andreas Herrmann, and Sheena Iyengar. "Order in Product Customization Decisions: Evidence from Field Experiments." Journal of Political Economy, 2010, Vol. 118, No. 2.
13. Lieberman, Henry and Fry, Christopher (2012), Justify: A Web-Based Tool for Consensus Decision Making. http://web.media.mit.edu/~cfry/.
14. MacEachren, A.M. 2005. Enabling collaborative geoinformation access and decision-making through a natural, multimodal interface. International Journal of Geographical Information Science, 19: 293–317.
15. Nielsen, J., and Molich, R. (1990). Heuristic evaluation of user interfaces, Proc. ACM CHI'90 Conf. (Seattle, WA, 1-5 April), 249-256.
16. Nielsen, J. (1994a). Enhancing the explanatory power of usability heuristics. Proc. ACM CHI'94 Conf. (Boston, MA, April 24-28), 152-158.

17. Rahwan, I. Banihashemi, Reed, Walton and Abdallah (2011). Representing and Classifying Arguments on the Semantic Web. The Knowledge Engineering Review. Volume 26, Issue 4, pp 487-511.

18. Wells, Simon and Reed, Chris. MAgtALO: Using Agents, Arguments, and the Web to Explore Complex Debates.