3-2003

A Value Focused Thinking Approach to Software Interface in a Complex Analytical Domain

Christopher M. McGee

A VALUE FOCUSED THINKING APPROACH TO SOFTWARE INTERFACE IN A COMPLEX ANALYTICAL DOMAIN

THESIS

Christopher M. McGee, 2Lt, USAF
AFIT/GOR/ENS/03-16

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT/GOR/ENS/03-16

A VALUE FOCUSED THINKING APPROACH TO SOFTWARE INTERFACE IN A COMPLEX ANALYTICAL DOMAIN

THESIS

Presented to the Faculty

Department of Systems and Engineering Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Operations Research

Christopher M. McGee

2Lt, USAF

March 2003

AFIT/GOR/ENS/03-16

A VALUE FOCUSED THINKING APPROACH TO SOFTWARE INTERFACE IN A COMPLEX ANALYTICAL DOMAIN

Christopher M. McGee
2Lt, USAF

Approved:

Capt. Stephen Chambal, Ph.D. (Advisor) Department of Operations Research	date
Dr. Janet Miller, Ph.D. (Member) Air Force Research Laboratories	date

Acknowledgements

I would like to thank my advisor, Captain Stephen Chambal, for all of his guidance and patience throughout my thesis process. I would also like to thank my sponsor, Dr. Janet Miller, for providing me this thesis to work on as well as for providing insight along the way. My group, consisting of Dr. Patricia Chalmers, Dr. Chris Colliver and Cris McDaniel, proved to be tremendous help throughout the development of hierarchy, and without them my thesis would not be what is. Also, thanks to Brad Bode for all of his help in learning about the software through its development.

I also need to thank my classmates, who gave me support throughout my thesis process. Special thanks goes to Don and Mo, our fearless class leaders, who led by their example, Kris Pruitt, for doing the thesis no one else wanted to do, and Yager, Scott, Justin, Joe and Sarah for wasting countless hours of my time playing Euchre.

Finally, I want to thank my family, especially to my parents who have been there for me throughout my academic career and supported me. A special thanks my wonderful wife who actually married me while at AFIT. Thank you for always being there for me, helping me, and being understanding throughout the past eighteen months.

Table of Contents

LIST OF FIGURES

List of Tables

Abstract

The intelligence community is faced with an extensive amount of data. Software programs are being developed to examine this issue of data overload and to develop solutions. The responsibility of making the final software decision lies on the analyst, therefore, the interface is the key to linking the intelligence data to the processing and results. If the interface is difficult and complex, the software will be less likely to be used. A methodology must be created which can objectively evaluate the effectiveness of the interface. This methodology will also measure the improvements in the interface's effectiveness that result when various changes are made to the original software interface.

Value focused thinking (VFT) is a proven methodology that can be applied to this problem. VFT provides an objective methodology to identify the values of an organization. Its hierarchical structure is well suited for handling multi-objective problems, such as identifying the values of software interfaces. The values can be measured and put to a common scale, allowing their contribution to the overall objective to be evaluated. By assigning quantifiable measurements to the components, the multi-objective goal can be evaluated and insight can be provided to the decision makers involved with the intelligence software.

VFT was applied to determine what is valued in software's interface to members of the intelligence community. With these values identified, a software that is under development was evaluated against the hierarchy. This provided insight into where

improvements could be made to the interface that would provide the greatest benefit. The VFT process also allows for the decision maker to continually reevaluate the software against the hierarchy, enabling continual improvement on the interface while maintaining the values of the intelligence community.

A VALUE FOCUSED THINKING APPROACH TO SOFTWARE INTERFACE IN A COMPLEX ANALYTICAL DOMAIN

Chapter 1. Introduction

1.0 Background

In the aftermath of the September 11 attacks, the intelligence community faced criticism from the public regarding their failure to predict the attacks. As the nation challenged the intelligence community to provide an explanation, the shortcomings in the intelligence community's ability to perform threat assessments became increasingly apparent. The probing questions illuminated one of the most significant obstacles the intelligence community faces: the intelligence community is unable to sufficiently process and analyze the overwhelming amount of data that is being collected with its current resources. Due to this data overload, the unevaluated information undoubtedly contains nuggets of information critical to the intelligence community's core responsibility of composing accurate threat assessments necessary for maintaining our nation's security.

Today, the intelligence community is responsible for collecting and interpreting the information that influences military, economic, and political policy. The range and scope of the information for which the intelligence community is responsible requires the community to work with tremendous varieties and quantities of information. In order to manage this task, the intelligence community continually searches for means of improving its efficiency and effectiveness.

Advances in computer technology, particularly those technologies used in the development of software applications, have assisted the intelligence community in increasing its effectiveness. They make it easier to store data as well look through it. In terms of communication, the internet technology combined with security technologies helps transport valuable data quickly and efficiently through secure channels like the Nipernet, Sipernet, Intelink and other classified connections. Computers have also made it possible to merge multiple data sources, making it possible to process more information more quickly. Programs that can sift through data, process high-tech algorithms that lead to better decision making, and display more data more efficiently are all examples of the capabilities of software applications.

Incorporating a Bayesian belief network approach into software applications is an example of how advances in software technology can assist the intelligence community. Bayesian belief networks use conditional probabilities which allow the software to look for a pattern in the data that would alert the intelligence community to possible problems. Because the intelligence analysts must be able to process the data that the software is running, it is critical that software developers focus on integrating the human aspect with the technical aspect. It is essential that interface of the software that is applying a Bayesian belief network approach serves two functions; it must be able to display the processing done by the software as well as communicate it to the end-user in the intelligence community.

1.1 Problem Statement and Context

Intelligence analysts are experiencing extreme data overload. While the concept of automating processes in theory will address the problem, the specific means in which the automation will take place need to be determined. Software programs are being developed to examine this issue of data overload and to develop solutions. The software being developed can help with this problem by using Bayesian belief networks to search for patterns in the data and determine the likelihood that events will occur. Although software determines the likelihood of the events, the analyst must be able to interpret this information to make a final decision. Since the responsibility of making the final decision lies on the analyst, the interface is the key to finding the data and processing the results. If the interface is not easy to understand, the software will be less likely to be used. The software interface must be commonly understood by the user as well as effectively communicate the data. To ensure that the data is comprehended, a methodology must be created which can objectively test the effectiveness of the interface. This methodology will measure the improvements in the interface's effectiveness that result when various changes are made to the original software interface.

Value focused thinking (VFT) is a proven methodology that can be applied to this situation. The primary benefit that VFT provides is its ability to convert the goals of a project or values of an organization into an objective realm. Its structure lends it to handling multi-objective problems even if the objectives are of a subjective nature. VFT is a type of decision analysis, causing it to have the fundamental characteristic of being a tool to break down an overall objective into smaller values. At this level, the values can be measured and put to a common scale, allowing their contribution to the overall

objective to be evaluated. By assigning quantifiable measurements to the components, the multi-objective goal can be evaluated.

1.2 Research Objectives

The objective of the research project is to provide the intelligence community with a method for evaluating alternatives for improvement to the software interface being developed. The software being evaluated is in the early phase of development which means the there is great potential to adapt the design based on research results. The interface is scored against a set of measurements to determine how effective it is in incorporating components that the intelligence community values in a software interface. The research then identifies where improvements can be made to enhance the value the software interface provides to the intelligence community.

Another important research objective is to create an iterative process so that the interface can be evaluated throughout all stages of its development. This allows the software interface's functionality to evolve with the changing demands of the intelligence community.

1.3 Methodology

The first component necessary to gain a better knowledge of how to improve software interface for the intelligence community is an understanding of software interface itself. A solid understanding of the components of a good software interface provides the basis for improving the software interface overall.

The next step is to examine the types of software specifically used by the intelligence community. This allows for a better understanding of the interfaces that are pertinent to the project. Without understanding the functionality and capabilities of intelligence software, it would be difficult to encompass the values of the intelligence community into a VFT model.

To apply this methodology, the Bayesian Belief network software will be used. The software is a threat assessment software that uses Bayesian networks to analyze threats in its early stages of development. Threat assessment and Bayesian Belief networks will be covered to give a background on the components of the software.

Once there exists an understanding of these components, a VFT hierarchy can be developed. The hierarchy captures the values of the software interface in the intelligence community and is validated by members of the intelligence community to ensure its accuracy.

Upon the validation of the intelligence community, the existing interface is scored against the hierarchy to determine the baseline score. Using the baseline score, value gaps that exist in the existing interface can provide insight as to where improvements to the interface can be made. The VFT process allows the developers to continue to evaluate the interface in the same way as the interface continues to be developed.

Chapter 2. Literature Review

2.0 Overview

The goal of this literature review is to provide appropriate insight into the problem to fully understand the process that is taking place. To do this, it is important to gain a familiarity with a variety of topics that influence the overall situation. A basic understanding of the intelligence community and their responsibility is necessary. Specifically, threat assessment must be explored. Since software interface in intelligence software is the focus of this study, it is important to have a basic level of knowledge about software interface. This study looks at a Bayesian Belief network software named JavaBase, making an understanding of Bayesian Belief networks essential. Finally, an understanding of the VFT process that is used to aid in the development of a user-friendly interface is required.

2.1 Intelligence Background

To get a better understanding of the software, the type of interface that is needed, and the software's applications, a better understanding of the user is needed. Development of the software will be based on the end user's needs and preferences. The intended users for this JavaBase software are the members of the United States Intelligence Community. A better understanding of the demographics of the user—who they are and what they do—will aid in the development of the software. Specifically, since this software is being developed for the Department of Defense, the focus will be on the members of the intelligence community within the Department of Defense.

2.1.1 Who They Are and What They Do

The following is a description of the intelligence community and its users as given by the Director of Central Intelligence of the United States.

> Throughout history, the leaders of nations and armies have sought to be forewarned of dangers and forearmed with information that reduces uncertainty and provides a critical edge for decisions. The effort to meet these fundamental needs of decision makers is what lies behind the practice of intelligence. That practice consists of collecting and interpreting information, overcoming in the process any barriers erected to keep secret the activities, capabilities, and plans of foreign powers and organizations. (DCI, 2002)

Intelligence is necessary for the government of the US to do its day-to-day business. The intelligence agencies provide vital information that gives an advantage to the political, military and economic interest of the nation. These agencies work for the President, his Cabinet, the Congress and the military forces. (DCI, 2002)

Since these agencies are so vital to the nation and our way of life, there is a great deal of responsibility on the intelligence officers in these various agencies. The Director of Central Intelligence says the following about the responsibility of the intelligence officer:

> For intelligence officers, this means maintaining an ability to warn policymakers and military leaders of impending crises, especially those that threaten the immediate interests of the nation or the well-being of US citizens. It also means giving government and military officials advance knowledge of long-term dangers, such as the threats posed by countries that covet weapons of mass destruction. It means helping to safeguard public security by countering threats from terrorists and drug traffickers. It means supporting economic security by uncovering foreign efforts at bribery and other schemes to tilt the playing field of international trade. And it means multiplying the effectiveness of US military forces deployed for operations. (DCI, 2002)

For this reason, the intelligence officer needs as much assistance as possible to monitor all possible potential developing situations. The JavaBase software is intended to help the intelligence personnel follow these potential situations, giving them more time to perform there many other tasks.

The Director of Central Intelligence is the one who oversees the entire intelligence community. The DCI creates the budget for the activities that the intelligence community will manage and implement. Figure 1 graphically represents the members of the intelligence community. The goal of these agencies and of the intelligence community is "to support decision makers with the best possible information, no matter its source." While all agencies work towards common goals, they also have specific responsibilities. The responsibilities of the agencies that make up the Department of Defense will be examined in greater detail.

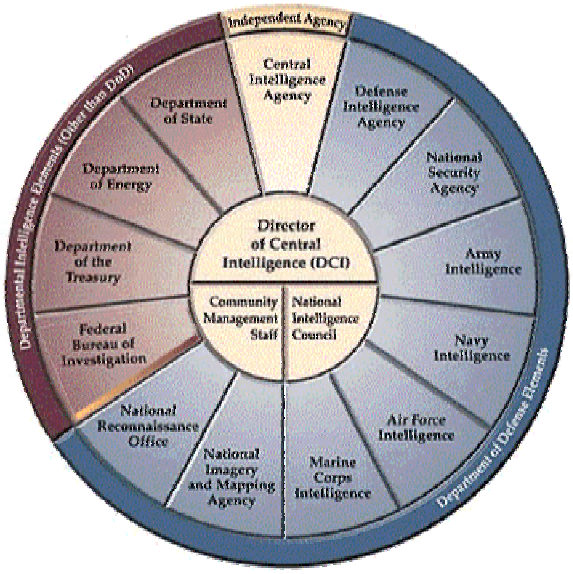

Figure 1: Intelligence Community Members

2.1.2 National Reconnaissance Office

The National Reconnaissance Office (NRO) is the only US program that meets the need for spaceborne reconnaissance. NRO receives it's budget through a program called the National Foreign Intelligence Program and NRO is structured under the DOD. The President appoints the Director of NRO, and Congress must confirm the appointment as the Under Secretary of the Air Force for Space. The existence of NRO was classified until September 18, 1992 when the Deputy Secretary of Defense ordered its existence to be declassified. The mission of the NRO is "to ensure that the US has the technology and spaceborne assets needed to enable US global information superiority." They achieve their mission by researching, developing, acquiring and operating the nations intelligence satellites. They also support as a detection and warning entity— monitoring arms control, military operations and natural disasters. NRO is staffed by the military services, the CIA and the civilian DOD members. (DCI, 2002)

2.1.3 National Imagery and Mapping Agency

According to the DCI, the National Imagery and Mapping Agency (NIMA) "provides timely, relevant, and accurate imagery, imagery intelligence, and geospatial information in support of military, national-level, and civil users."

NIMA was established to support the DOD in combat situations. The DCI asserts that "NIMA is committed to attaining information superiority in the mission space of the next century, as well as to addressing civil issues critical to U.S. national interest, and improving the decision and cycle times for those who make and execute national security

policy. The Agency's focus is on providing high-value information and laying the foundation for the more efficient exchange of data and integration of products and services." Through this DOD hopes to provide the best images and information available to aid in combat situations. (DCI, 2002)

2.1.4 Military Intelligence (Marine, Air Force, Navy, Army)

The chief purpose for each of these intelligence agencies is to aid their respective branch of service to better perform their mission. The following is the mission of each agency as given by the DCI.

Marine Corps Intelligence

> MCIA produces a full range of products to satisfy customer needs in peace, pre-crisis, or contingency situations, and to support service obligations for doctrine development, force structure, training and education, and force modernization. MCIA accomplishes this mission through the integration, development, and application of general military intelligence, technical information, all source production, and open-source materials. (DCI, 2002)

Air Force Intelligence, Surveillance, and Reconnaissance

> The mission of Air Force Intelligence, Surveillance, and Reconnaissance (ISR) is focused on ensuring the US military team - whether in peacetime operations, a crisis, or war - attains information superiority: the ability to collect, control, exploit, and defend information while denying the adversary the ability to do the same. To do this, Air Force ISR, in partnership with the other Military Services and national intelligence agencies, delivers intelligence information when, where, and how it's needed. As a member of the Intelligence Community, the Air Force operates worldwide ground sites and an array of airborne reconnaissance and surveillance platforms such as the U-2, the RC-135, and unmanned aerial vehicles (UAVs) to meet national-level intelligence requirements. As a member of the Joint team, Air Force intelligence professionals work in the Unified Commands' intelligence centers and Air Force personnel and resources are embedded in each Unified Command's air component. To support day-to-day Air Force operations, intelligence professionals at the wing and squadron levels use suites of interoperable analysis tools and dissemination systems to tailor information received from all levels and agencies in the Intelligence Community to meet

specific Air Force requirements. (DCI, 2002) They are supported by National Air Intelligence Center (NAIC) who produce integrated, predictive air and space intelligence to enable military operations, force modernization and policy making.

Naval Intelligence

Naval intelligence products and services support the operating forces, the Department of the Navy, and the maritime intelligence requirements of national level agencies. Office of Naval Intelligence (ONI) is the center of expertise for every major maritime issue--from the analysis of the design and construction of foreign surface ships to the collection and analysis of acoustic information on foreign sensor systems, ocean surveillance systems, submarine platforms and undersea weapons systems. Its analysis of naval air warfare ranges from appraisals of opposition combat tactics to analysis of rival missile signatures, making it the authoritative resource for maritime air issues. Finally, ONI's technical expertise in analyzing naval weapons and systems, combined with the operational expertise of its intelligence and warfare specialists, allows for more effective analysis of the complex questions of contemporary naval capabilities and for a more accurate projection of those capabilities into the future. (DCI, 2002)

Army Intelligence

Army intelligence is prepared to meet the full range of Foreign Ground Force Intelligence requirements generated by commanders at every level across the spectrum of operations. Army intelligence force structure is designed to provide timely, relevant, accurate and synchronized intelligence and electronic warfare support to tactical, operational and strategic level commanders across the range of Joint military operations. (DCI, 2002) The National Ground Intelligence Center (NGIC) support the Army and produce all-source integrated intelligence on foreign ground forces and supporting combat technologies to ensure that US forces and other decision makers will always have a decisive edge on any battlefield. The Army is supported by Missile and Space Intelligence Center who develops and disseminates intelligence concerning threat guided missile systems, directed energy weapons, selected space programs/systems and related command, control, and communication to support operationally deployed forces and the material acquisition process. Develops and distributes digital simulations of threat weapons systems and provide threat simulation support to force developers and operational forces.

2.1.5 National Security Agency

The National Security Agency is a separate agency under the Department of Defense. It was established in 1952 and its role is to plan, coordinate, direct and perform foreign signals intelligence and information securities.

Performing these actions—knowing foreign signals as well as protecting our own—gives the US a competitive advantage. In order to complete these tasks NSA agents must have a tremendous understanding of cryptology, and use this understanding to give them this advantage. NSA must keep ahead in the technological fields to maintain this cryptology advantage. The protection of our information is vital to our national security. Consequently, NSA is a critical member of the intelligence community. (DCI, 2002)

2.1.6 Defense Intelligence Agency

The Defense Intelligence Agency (DIA) is used as a combat support agency for the Intelligence Community. Their mission is to provide all-source intelligence to the different branches in the military. The key areas of emphasis, according to the DCI, include "targeting and battle damage assessment, weapons proliferation, warning of impending crises, support to peacekeeping operations, maintenance of data bases on foreign military organizations and their equipment and, as necessary, support to UN operations and US allies." DIA also supports other members, including the DOD and policymakers, but their main focus is used to support the war fighter. (DCI, 2002)

All of the agencies share the common mission of protecting our nation. Especially in light on recent events, the shared responsibility of threat assessment has

become more important and will be looked at in greater detail. The Armed Forces Medical Intelligence Center (AFMIC) falls under the DIA and produces finished, all-source, medical intelligence in support of the Department of Defense and its components, national policy officials, and other federal agencies. Assessments, forecasts, and databases are prepared on foreign military and civilian health care capabilities and trends, worldwide infectious disease occurrence, global environmental health risks, and militarily significant life science technologies. AFMIC supports all services.

2.2 Threat Assessment

Threat assessment has become an extremely important part of America's way of life. Since the event of September 11, 2001, terrorism has become a bigger issue in America, and Americans want to know where this threat of attack will come from next. President George Bush (2001) said, "The civilized world is rallying to America's side. They understand that if this terror goes unpunished, their own cities, their own citizens may be next. Terror, unanswered, cannot only bring down buildings; it can threaten the stability of legitimate government. We're not going to allow it." To prevent such acts of terror, it is necessary to be able to assess what are legitimate threats to our security. As a result, threat assessment has become a more conspicuous part in defending our country.

According to the Fein (1995), there are three major functions of a threat assessment program. These functions are the identification of a potential perpetrator, assessment of the risks of violence posed by a given individual at a given time, and the management of both the subject and the risks that he or she presents to a given target.

Identifying the perpetrator is obviously an important step, but how does one know who a potential perpetrator is? According to Fein (1995), "Perpetrators of violence often have a traceable history of problems, conflicts, disputes, and failures." This explains why a person may become potential perpetrator, but it does not identify why they act as they do. Perpetrators often act out in response to something that has happened to them in the past. Fein (1995) writes, "Violent behavior may be triggered by these individuals' perception that it provides a means to rectify or avenge an injustice or wrongdoing. Targeted violence can be premeditated or opportunistic when a situation arises that facilitates or permits the violence or does not prevent it from occurring." The Fein (1995) also writes, "Violence is a process, as well as an act. Violent behavior does not occur in a vacuum. Careful analysis shows that violent acts often are the culmination of long-developing, identifiable trails of problems, conflicts, disputes, and failures." It is the identification of these trails that is needed to successfully assess a threat, and hopefully deter it from occurring.

The duty of the investigator is to identify and track these risks and to become aware of these patterns that and individual or a group may pose. This is where the difficulty has come in the intelligence community. Due to information overload, identifying these threats has become very difficult, but tracking the patterns of a group or individual has become even harder. (Stubbing and Goodman, 2002) According to Stubbing and Goodman, "Accurate and timely intelligence is the critical first line of defense against terrorism, America's major national security threat of the 21st century." The challenges associated with this task have been occurring at an increasing rate. According to an ENN Daily Intelligence Report from 1997 that took place in Bosnia, a

task force concluded that critical intelligence was not getting to the necessary U.S. forces while an inordinate amount of invaluable information was reaching and overwhelming their resources.

2.2.1 Intelligence Overload

Since threat assessment has become so important to our national security, there has been increasing pressure in the intelligence community to be able to sift through its intelligence and pick up the patterns and identify threats. Hebert Simon (1981)wrote:

> "The information-processing systems of our contemporary world swim in an exceedingly rich soup of information, of symbols. In a world of this kind, the scarce resource is not the information; it is the processing capacity to attend to information. Attention is the chief bottleneck in organizational activity…"

This is telling us that gathering the information is not the problem, but the ability to pay attention to all of it is the problem. According to Stubbing and Goodman (2002):

> "The CIA and FBI both suffer from organizational overload. The CIA has operational missions to collect human intelligence and conduct covert action. It is also responsible for analysis and publication of national intelligence estimates. The agency cannot perform both well. The FBI also suffers from a bipolar mission. Its traditional law-enforcement mission involves reacting to crimes that have already occurred. Its counterterrorism mission, by contrast, requires a proactive role – fettering out incipient threats to national security."

Although the introduction of technology has helped to alleviate some of the challenges of the intelligence community, it has also brought on many challenges of its own. Development of intelligence gathering technologies has been developed more rapidly than technologies that assist in the processing of data. This disparity within technology development has created significant problems. Technology has led to gathering such a vast amount of information that it is extremely difficult for the existing personnel and technology to sufficiently process the data. (Simon, 1981) The technology can gather the data, but the analytical capacities of technology used in the intelligence

community have not been able to keep up. This creates distinct but interrelated difficulties. Members of the intelligence community spend their time sifting though a large amounts of data rather than investigating a smaller amount of data in greater depth. In an article by Steve Macko (1997), a task force concluded "We need to make sure that we don't saturate the warriors with data while starving him of useful information." The alternative is to examine specific information in detail while ignoring a significant portion of the data that has been gathered. The September 11th attacks are a good example of how serious this problem is. Had there been a way for information to have been more efficiently processed, members of the intelligence community could have potentially spent their time investigating information that could have warned them of the attacks instead of having to sort though a vast amount of information that did not factor in to the attacks.

This example serves to illustrate of serious difficulties data overload creates for the intelligence community. With this in mind, it becomes easy to see the value of JavaBase software that can aid the intelligence community in sifting through the intelligence data and identifying these patterns through the use of Bayesian Belief Nets (to be discussed later). If successfully implemented, the software would be able to sift through the data and alert an agent if a problem area is likely developing.

2.3 Bayesian Belief Networks

Bayesian Networks is a technique that uses conditional probabilities to predict the probability of what event is most likely going to happen in a network. These networks

have been used in a variety of fields, such as medical diagnosis, map learning, language understanding, and heuristic search. (Charniak, 1991)

Bayesian networks are directed acyclic graphs. This means that each event, or node, is connected to another by an arrow, and the arrow implies a causal relation. The node that the arrow starts at is referred to as the parent node, and the node the arrow is pointed to is referred to as the child node. The state of the parent node has an effect on, or is the cause of, the state of the child node. An example of a network is drawn below.

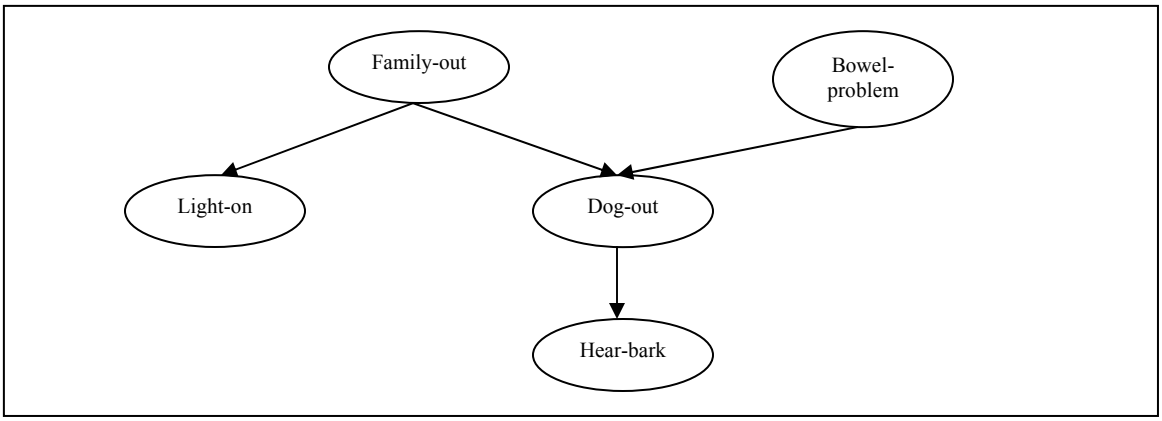

Figure 2: Bayesian Network Example (Charniak, 1991)

In the example above, Family-out, Bowel-problem, Light-on and Dog-out are all parent nodes. Light-on, Dog-out and Hear-bark are all child nodes. It can be seen that a node can be both a child node as well as a parent node. A node that is only a parent is called a root node. Each node has a number of states that it can take. An example of possible states for the node Dog-out from Figure 2 above could be yes or no identifying yes the dog is out or no the dog is not out. The node can take on only one state at a time, either yes or no.

Each node has a probability, or a conditional probability, that it can take on a certain state. A root node only has a probability that it takes on a certain state. For example, let's say the Family-out node has two state, yes the family is out and no the family is not out. The node has probability that the family is out and a probability that the family is not out. These probabilities must add to one. On the contrary, the probability of a node that is not a root node depends on what state the parent node is in. For example, the Lights-on node consists of the states yes the lights are on and no they are not. The probability that the light is on depends if the family is out or not. Therefore, the conditional probabilities for Lights-on are the probability the lights are on given the family is home, probability the lights are on given the family is not home, probability the lights are not on given the family is home, and the probability the lights are not on given the family is not home.

Each node is looked at as a variable, changing at any time. An observation of a node is the state of a node. An event is the state of a node that takes place a certain time. A node stays at the state of an event until it is changed. Once an event has occurred, each node in the network in the network takes on a posterior probability, which is the probability that the node takes on at that time.

This is a rather simplistic look at Bayesian networks (for a more complete look at Bayesian Belief networks, look at Pearl, 1988). These networks are used to see how certain events can affect the entire network. They can also be used to see which event has the greatest impact on another node, or which node has the greatest impact on making a certain event more likely. Many types of software exist that use these types of networks that can calculate these probabilities and update them as events occur, and do

this very quickly. JavaBase will have this capability, with the events being intelligence information that has been gathered. It can also be used to look to potential situations that may develop and the effects they may have. For this JavaBase software to be successful, the analyst must be able to understand it, making the software interface an integral component.

2.4 Software Interface

Before the interface of a software application can be looked at and understood more clearly, we must first define the subject. According to Alison's Head's Design Wise (1999), an interface "is the visible piece of a system that a user sees or hears or touches." The author goes on to say, "Users come into contact with an interface when they use a system often needing to get a task done. Regardless of whether it whirs, spins, speaks, or lights up, an interface exists in one form or another in every system. There are a million different interfaces that are designed by someone for something." This definition appears to be true, since there are countless numbers of software systems that exist, and they are all designed for different types of users. An examination of Microsoft's marketing strategy alone points to this concept; they sell their software applications packaged separately for home use and business purposes knowing that different people have different needs, even if they use the same basic software. The same is true for members of the intelligence community. According to Emily Patterson (1999), "The intelligence analyst rarely directly observes events in the world. Rather, other humans generate reports about events in the world. These reports make up a set of databases whose characteristics are often opaque to the analyst, particularly since the available information is constantly being updated and the information is generally not

indexed." Information is "sampled" from these databases, first by keyword search queries and then by browsing dates and titles through the computer "keyhole," a small CRT screen (Woods and Watts, 1997). It is the interface that provides this keyhole for the user. This observation makes it important for the developers to understand who their end user is and develop their software and its interface with them in mind to be able to provide this "keyhole".

With this definition established, a study how interfaces have been historically designed is able to be started. In the past, according to Head, interface was designed using the "waterfall" model (1999). This model tended to be very time consuming as well as very costly. It is called the waterfall model because the model is made up of several steps that are done independently and consecutively—with each step initiated only when the previous one is completed. The first step was typically the client and the developer deciding what they want developed. Once this was done, engineers would look to see how the developers could accomplish this and what was needed to get this done. The next step, the design step, was used to build the design of the system. The system designers used themselves as the "likely users". The final step was documentation of the system and testing the system by its actual users. (Head, 1999)

The problem with this method, according to Head, is that the feedback from the user that was given during testing was done well after the final design and building of the system had been completed. Basically, this meant that the testing of the product has been done and major changes would not be made and the user's feedback would have no impact. These drawbacks quickly escalated into traits serious enough to cause Head and many like-minded designers to see the need to re-create the process. "User-centered

design" became the focus of both intellectual discussion and real-world application. According to Head:

> User-centered is the mantra of the Human-Computer Interaction (HCI) approach to interface design. When user-centered design principles are applied, end users are involved early on and throughout the development process. The user-centered design process is highly iterative to insure that the design fits users' expectations about functionality and operations. Before they get very far, project developers need to decide who their end users are and how the interface may be used. Once they have prototyped a product, developers conduct user testing that informs them about the effectiveness of their design choices. The user-centered approach is a methodology that includes testing and measurements techniques that are based on design guidelines. (Head, 1999)

The following figure, Figure 3, demonstrates how IBM has integrated this approach and represents the technique IBM requires its designers to employ. It can be seen that the user has a critical role in the development of the design of the software. This model does its best to ensure the software will be developed to the best specifications for the user. (Head, 1999)

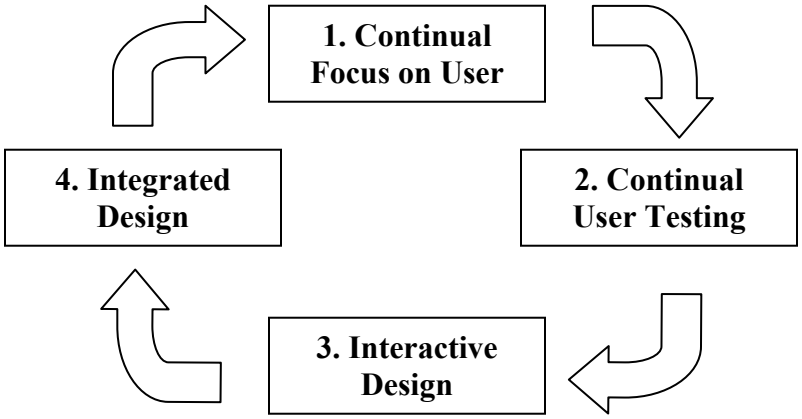

Figure 3: IBM Design Technique

Microsoft uses a very similar technique. Microsoft's design cycle has four phases: designing, prototyping, testing, and iterating. These phases are used to ensure an effective user-centered design. Figure 4 demonstrates the model used by Microsoft. In

the Microsoft model, the design phase is considered one of the most critical phases. This is where designers create the general shape of the software. If there are aspects that are incorrect in the design stage, they will permeate all subsequent stages, making them difficult to fix. The next stage—prototype—allows developers to communicate the design. This helps them visualize the design and where the design is going to go. Testing, the next stage in the cycle, is then done to determine any problems the software may have as well as to compare different alternatives for a particular task. Iteration is the final stage, giving the process its defining characteristic of being a cycle by requiring the repetition of the process. (Microsoft Press, 1995)

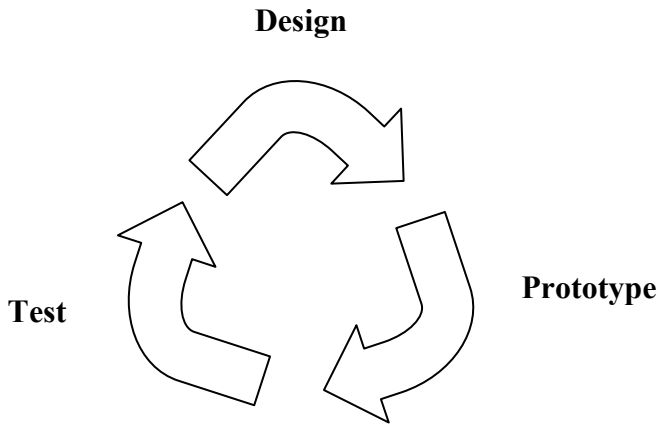

Figure 4: Microsoft Design Technique

2.4.1 Desired Principles

The value of a user-centered design of an interface has been expressed above. According to The Windows Interface Guidelines for Software Design, "a well-designed user interface is built on principles and a development process that centers on users and their tasks." According to this source, the following are desired principles for a

successful user-centered design: user in control, directness, consistency, forgiveness, feedback, aesthetics, and simplicity. A description of each is given as defined in the Windows Interface book. (Microsoft Press, 1995) These principles are similar to much of the literature that is found on interface design, especially the Eight Golden Rules of Dialog Design given by Shneiderman in his book Designing the User Interface (1992).

2.4.1.1 User In Control

It is important that the user should always be in control of the software, as opposed to feeling like he/she must adapt his/her usage to fit the software. This principle manifests itself in a number of ways.

The first way the principle manifests itself is that it makes the assumption that the user is proactive in the interaction rather than reactive. This principle may take shape in an actual application is by allowing the user to control and choose the tasks that are being done, even if the task is automated. An example of this would be a pop-up window which opens before an automated task such as Auto Archive and asks the user if he/she would like the software to perform the task. (Microsoft Press, 1995)

The second manifestation of the principle is that user must be able to customize components of the interface. This allows the user to select the setting that matches his/her abilities and personal taste. Software should allow the user this capability by enabling him/her to customize system settings such as color, font, icons, and other options. An example of this is the user's ability in Microsoft products to select the toolbars that are shown. (Microsoft Press, 1995)

Finally, in order to make the user be in control, the interface should be as interactive as possible, avoiding "modes" whenever possible. A mode, by definition, is a

state that excludes general interaction or otherwise limits the user to specific interactions. If the task does not permit an interactive setting and a mode must be used, the mode should be obvious to the user, highly visible, the clear result of the deliberate steps taken by the user, and very easy to cancel. (Microsoft Press, 1995)

2.4.1.2 Directness

The principle of directness prompts interface designers to create software that allows users to directly manipulate the software representation of information rather than having to interact with the information via a separate step or systems of step. Applied, this principle allows users to drag an object to a trash bin rather than key the command using a program-specific syntax. Users should be able to see how their actions affect objects on the screen. Wherever possible, information should be presented in a visual manner and users should be given choices. These strategies reduce the effort that the user must put forth and studies have shown that users can recognize a command easier than they can recall its syntax. (Microsoft Press, 1995)

Creating "familiar metaphors" (user tasks represented by a word or picture) provide a direct and intuitive interface. Metaphors link user tasks to experiences with which the user has an established familiarity. "By allowing users to transfer their knowledge and experience, metaphors make it easier to predict and learn the behaviors of software-based representations. Metaphors support user recognition rather than recollection. Users remember a meaning associated with a familiar object more easily than they remember the name of a particular command." (Microsoft Press, 1995)

2.4.1.3 Consistency

Consistency is the principle that "allows users to transfer existing knowledge to new tasks, learn new things more quickly, and focus more tasks because they need not spend time trying to remember the differences in interaction." In other words, consistency provides users a framework that is stable and, as a result, familiar and predictable. (Microsoft Press, 1995)

Consistency is important through all aspects of the interface and on many levels. An interface must be consistent in everything from the presentation of text (such as font and size) to the functionality of the application. The Windows Interface Guidelines for Software Design identifies several areas that demand consistency:

- Consistency within a product. Present common functions using a consistent set of commands and interface.
- Consistency within the operating environment. By maintaining a high level of consistency between the interaction and the interface conventions provided by Windows, your software benefits from users' ability to apply interaction skills already learned.
- Consistency with metaphors. If a particular behavior is more characteristic of a different object than its metaphor implies, the user may have difficulty learning to associate that behavior with an object. (Microsoft Press, 1995)

2.4.1.4 Forgiveness

The forgiveness principle recognizes that users often learn a new interface by exploring. This "trial and error" method that users typically employ makes it important

for the interface to prevent them from unintentionally making decisions that can harm the system or data. Software that is designed with the user in mind provides warnings to prevent the user from making these choices, and, in the best case scenario, even allows users the option of reversing their actions or recovering the previous state. (Microsoft Press, 1995)

The forgiveness principle also asks designers to account for the fact that human error is inevitable. Mistakes made by users can be both physical (accidentally pointing to the wrong command or data) and mental (making a wrong decision about which command or data to select). Well-designed interfaces aid in preventing the users from making errors and allowing them to correct those that do occur. (Microsoft Press, 1995)

2.4.1.5 Feedback

This principle stresses the importance of providing the user feedback to his/her actions. Feedback may be visual or audio and serves to confirm with the user that the software is responding to the user's input. Feedback is also used to communicate the details that pertain to the task that is being performed. (Microsoft Press, 1995)

Timing is critical. As stated in The Windows Interface Guidelines for Software Design, "Nothing is more disconcerting than a "dead" screen that is unresponsive to input. A typical user will tolerate only a few seconds of an unresponsive interface. Effective feedback is presented as close to the point of the user's interaction as possible. Even when the computer is processing a particular task, provide the user with information regarding the state of the process and how to cancel that process if that is an option."

The degree of information provided to the user in the feedback is important as well. The level of the feedback must match the task that it is referencing. In some cases,

a pointer change or a status bar message is appropriate. In situations where the feedback necessary is more complex, a message box may be required to convey the type of information that will satisfy the user. (Microsoft Press, 1995)

2.4.1.6 Aesthetics

The principle of aesthetics stresses the importance of the visible appearance of the interface. According to The Windows Interface Guidelines for Software Design, "Visual attributes provide valuable impressions and communicate cues to the interaction behavior of particular objects." In addition to serving these directly functional purposes, the aesthetic appeal of the interface can contribute to the overall impression and experience the user has. Consequently, it is important to involve a graphics or visual designer in the process to make the aesthetic elements as appealing as possible. (Microsoft Press, 1995)

2.4.1.7 Simplicity

The principle of simplicity is centered on the idea that an interface should be simple, easy to learn, and easy to use. The interface should balance functionality while maintaining simplicity.

There are several strategies designers employ to create simplicity. The first of these is to communicate using the minimum level of information necessary to convey the message. This can mean avoiding wordy descriptions for commands and leaving unnecessary information out of the interface. Another strategy to design a simple but useful interface is to use natural mapping and semantics. This contributes to the

simplicity of the interface because the arrangement and presentation of elements affects their meaning and association. (Microsoft Press, 1995)

A means frequently employed to enhance the simplicity of an interface is by using a technique called progressive disclosure. "Progressive disclosure involves careful organization of information so that it is shown only at the appropriate time. By "hiding" information presented to the user, you can reduce the amount of information to process." (Microsoft Press, 1995) A drop down menu that reveals multiple commands when a user clicks on it is an example of progressive disclosure.

2.4.2 Data Presentation

Given the nature of the intelligence community, the need to present a large amount of complex data simultaneously exists. For this reason, the presentation of the data becomes very important. According to Tufte in his book Envisioning Data, "Visual displays rich with data are not only an appropriate and proper complement to human capabilities, but also such designs are frequently optimal." (1991) Humans have the ability to process this vast amount of information. Tufte goes on to say "we thrive in information-thick worlds because of our marvelous and everyday capacities to select, edit, single out…"(1991) But having this vast information in front of us is not all that is needed to process the data; it must also be displayed in a way that humans can comprehend it. How can this be done? Even Tufte says that showing complexity is hard work. (1991)

In displaying complex data, it is important that space is not wasted. This does not mean that data should fill every inch of space, but Tufte says, "Vacant, low-density displays, the dreaded posterization of data spread over pages and pages, requires viewers

to rely on visual memory – a weak skill – to make a contrast, a comparison, a choice." (1991) Keeping the screen filled with the appropriate amount of information is very important to displaying good data.

The use of colors is also important when displaying data. There are many uses for colors that can help display complex data. It can be used to show distinction between objects or to show a change has occurred in an object. The proper use of colors can be very helpful aid when trying to display data; but designers should be aware than it can also be a distraction. The use of too many colors can confuse the user or make the display too complicated to understand. A good balance is needed to properly get the user to comprehend the data. (Tufte, 1991)

Displaying complex data can be complicated in itself. The key to getting the information across to the user is to have that appropriate balance to make sure the display shows enough without overwhelming the user to the point that he/she can't comprehend all the data. Finding what is valued in the interface becomes important to establish this balance. Value-focused thinking provides a methodology to determine the values that make an interface useful to its audience.

2.5 Value-Focused Thinking Principles

The process of making decisions has been around forever. In the past, most people made a "gut" decision based on comparing the alternatives they have been given. In recent years, different philosophies and techniques have been developed to help aid in analyzing these decisions. Value-focused thinking is one of these techniques, and has been proven to work in many different situations. (Kirkwood, 1997) One example where

VFT was proven as an effective methodology was done by Captain David Jurk(2002). He used VFT to select force protection initiatives for evaluation for the Force Protection Battlelabs. Another example where this methodology was used was done by Mark Shoviak (2001) who used it to evaluate integrated solid waste management alternatives for a remote Alaskan air station. These are just two examples that demonstrate VFT is a proven methodology, and they show the variety of topics for which it can be used.

The value-focused thinking approach as proposed by Leon (1999) should provide the following benefits in decision making:

a) Alternatives with more innovative characteristics are included
b) The range of alternatives included becomes wider
c) The future consequences of decisions are taken more into account
d) Alternatives that at first glance would not be considered are integrated
e) More desirable consequences are considered (Leon, 1999)

Value-focused thinking (VFT) concentrates on determining the values at the core of the decision. Consequently, the choice is not between a variety of alternative—each with its own benefits and drawbacks—but rather a selection of the alternative that gives the greatest benefit with regard to what has been determined valuable. VFT emphasizes that values should be the focus for making a good decision. However, most people try to look at all the alternatives and compare them against each other. This presents difficulty if one alterative is extremely better at one aspect of the decision while the other alternative is extremely better at another aspect. This type of decision is called a multi-objective decision, where multiple objectives are desired in the decision. VFT provides a

structure to compare these objectives against each other based on the decision-maker's values. VFT, however, takes more time and requires the decision-maker to give his mind to the exercise, but the benefit of this structure makes it worth the effort. (Keeney, 1992)

Alternative-focused thinking has become the "natural" way for people to make decisions. This way of deciding things has become more of an ingrained habit rather than a true process for making a decision. Alternative-focused thinking is a commonly used way to make decisions quickly, however, it is important to realize that VFT can generate new alternatives by spending more time with the problem and identifying the values behind the decision.

Keeney (1992) writes that, "Values are principles used for evaluation. We use them to evaluate the actual or potential consequences of action and inaction, of proposed alternatives, and of decisions." Hard thinking, according to Keeney, can identify these values. To think of these values in a decision process, the decision must have the following properties: the decision should be a real problem, it should be of great importance, and it should be complex and have no absolute solution. The decision maker should be able to answer the "why is this important" test. If the decision has no real importance the input to the decision will not carry the necessary relevance to make a true decision.

The question that surfaces at this point is how to determine what the decision-maker values. It is important that only values are being pursued and that the decision-maker has no alternatives in mind. Having alternatives already in mind limits the thought process.

The following model show the benefit of thinking about values.

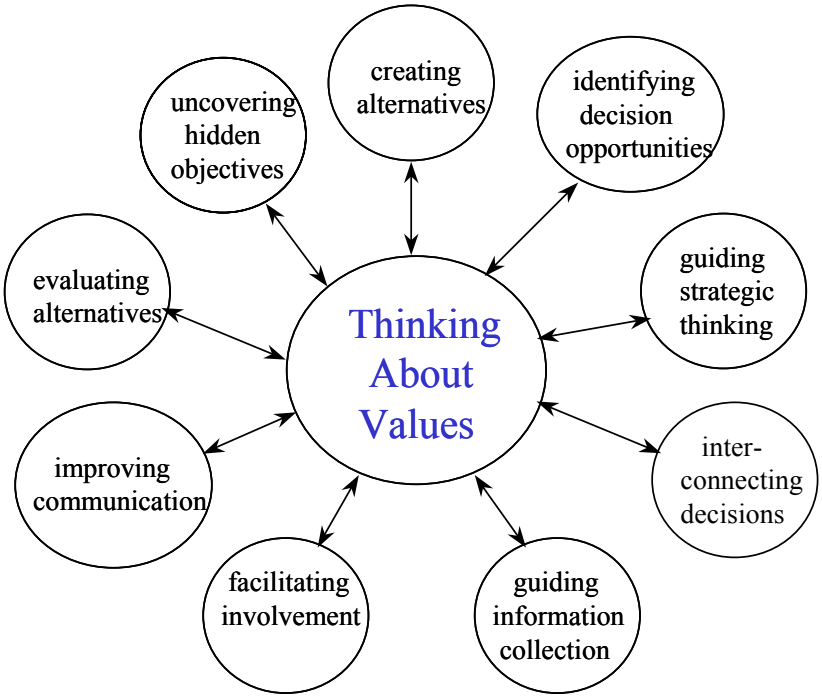

Figure 5: Thinking About Values

The main purpose of VFT is to provide insight and structure to a decision. VFT can be a tricky process, and for this reason is it good to have someone who is experienced in decision analysis and VFT aid the decision maker in the process. Also, if VFT is done correctly, it can be a lengthy process. For this reason it may not be possible for the decision maker to give all the time needed to complete this process. Also, the decision maker may not be experienced in all aspects that make up the decision. In this case subject matter experts (SME's) can be used to aid the decision maker. Ideally in this process the best SME's and most experienced decision analysts would be desired to reach optimal results, but a proxy can be used if the best can not be achieved. (Class Notes)

The following figure, Figure 6, shows the ten-step VFT process, taught by AFIT, that will be used to aid in the decision for which interface would be best for the JavaBase

software. It is important to note that the ten-step VFT process is an iterative process which means that at any point in the process it is possible to return to a prior step if new information or thinking warrants it. To fully understand the ten-step VFT process, each step must be examined ands fully understood. The following is a brief explanation of each step:

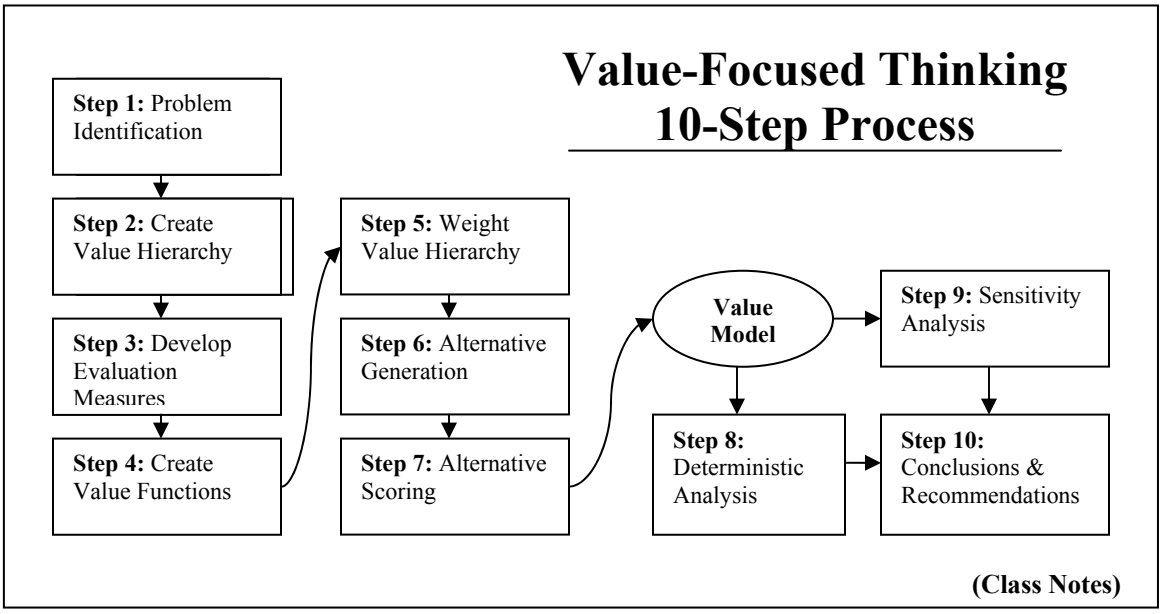

Figure 6: 10-Step VFT Process

Step 1 - Problem Identification: The first step in the ten-step process is to identify the problem. This step is one of the most important steps. A clearly defined/identified problem is needed to drive the remaining nine steps. Having a clearly defined problem can give clarity throughout the process and puts all people involved in the same mind frame.

Step 2 – Create Value Hierarchy: Creating the hierarchy can be a lengthy process. The reason for the length of time is that the hierarchy must be collectively exhaustive and mutually exclusive. Collectively exhaustive, by definition, means that

every possible value that makes up this decision must be encompassed somewhere inside the hierarchy. Even if a value has very little value to the decision, it must be in the hierarchy to make it collectively exhaustive. The mutually exclusive characteristic states that each value must be independent on every other value. The reason this is needed is so no value is counted more than once so that an appropriate weight can be put on it later.

There are two methods typically used in creating a hierarchy, the top-down or bottom-up approach. The top-down approach starts with the problem and breaks the problem down into what is valued in that problem. Then, each value is broken down into what is of importance in that value. If a value can be broken down then the next value is looked at. This process is done until all the values can be broken down no further. In the bottom-up approach, the decision maker and the SMEs typically know the end values that cannot be broken down any further, but are not sure how they should be grouped. These values are then grouped "up" into broader values. All values are grouped "up" until they cannot be grouped "up" anymore. Once this is done, these values should be the values that make up the decision. The following is an example of what a hierarchy may look like:

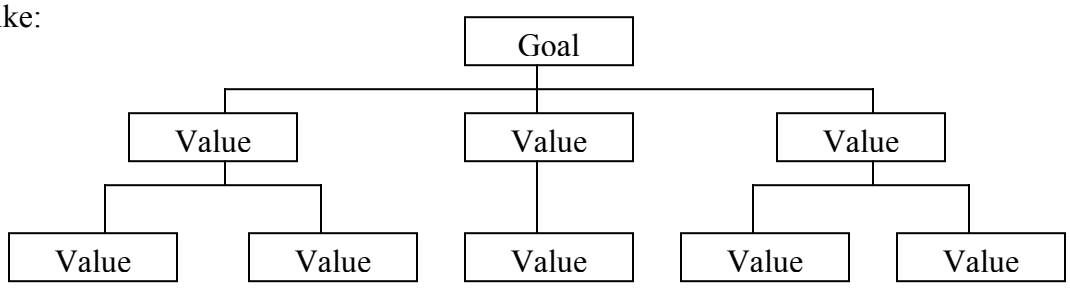

Figure 7: Example Hierarchy

Step 3 – Developing Evaluation Measures: A hierarchy consists of tiers and branches. A tier is how a hierarchy is broken up horizontally and a branch is how it is broken up vertically. (An example of each is shown below.) Once all of the values that

are included in the hierarchy have been found, evaluation measures for the values that are in the last tier must be determined. An evaluation measure is something about the value that can be quantified or be measured. Measures can be assessed either directly of by using a proxy measure scale.

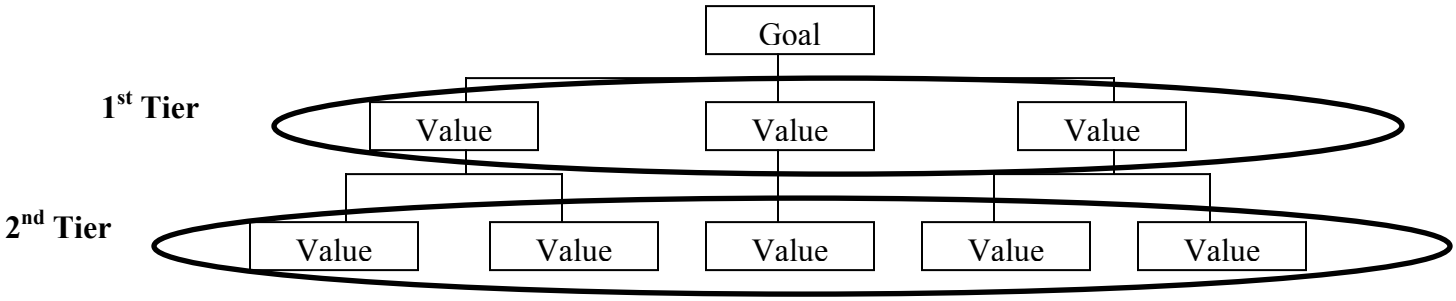

Figure 8: Example of tiers

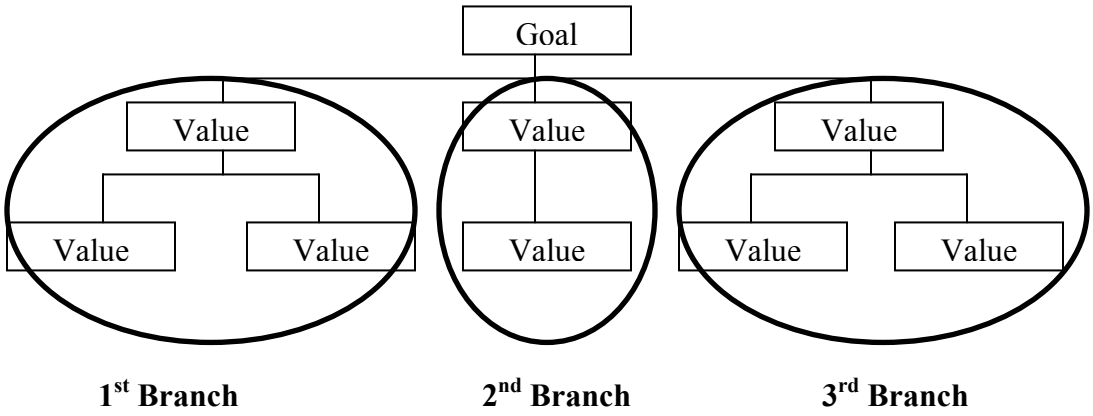

Figure 9: Example of branches

In the example above, the second tier is the lowest tier and needs evaluation measures. According to Kirkwood (1997), a direct measure "directly measures the degree of attainment of an objective." A measure can be assessed directly, but it is not

always possible or may be extremely difficult to assess a measure directly. In this case an approximate measure, or proxy, can be used. Measures can also have a scale that is natural or constructed. A natural scale is one that is common to everyone. When a natural scale does not exist, a scale must then be constructed.

The following shows the hierarchy with evaluation measures:

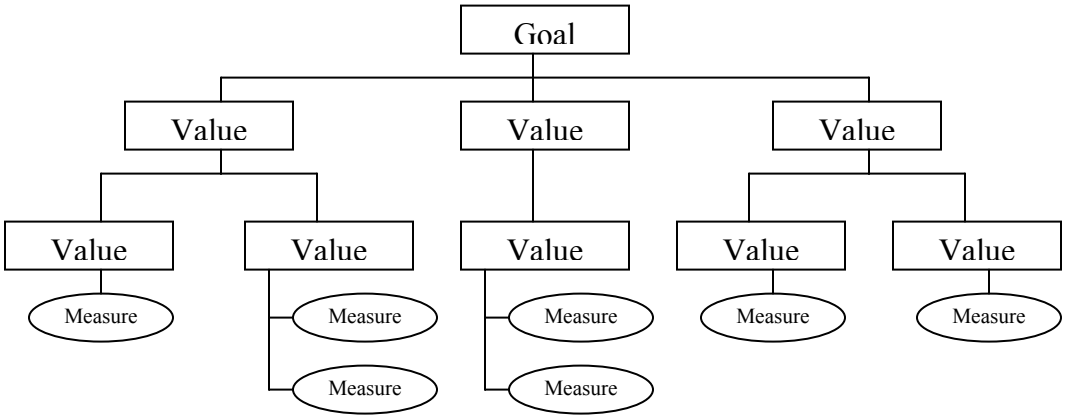

Figure 10: Example of measures

Step 4 – Create Value Functions: Once measures have been created for the values, value functions must be created for the measures. The purpose for a value function is to evaluate all measures on a unit-less scale that ranges from 0 to 1, where 0 is the worst possible outcome and 1 is the best. Once the best and worst cases have been identified for a measure, the decision maker must decide what value every other possible outcome gets in the range of 0 to 1. Also, every function must be monotonically increasing, that is, as the decision goes from the worst case to best case scenario, the value must increase from 0 to 1. This must be done for all measures.

Step 5 – Weight the Value Hierarchy: Once steps 1-4 have been completed, the hierarchy is complete and must be weighted. The purpose of weighting the hierarchy is

to identify the importance the each value contributes to the overall goal or problem. There are two types of weights, local and global. A global weight identifies the overall total value a value has towards the goal. A local weight identifies the total value that a value has towards the value above it. In global weighting, the sum of all the global weights across a tier must sum to one. In local weighting, the sum of the local weights attached to the same parent node must sum to one. The global weight of a value can be calculated be multiplying its local weight by all of the local weight of the values that are connected to it from above. The following is an example of both local and global weighting.

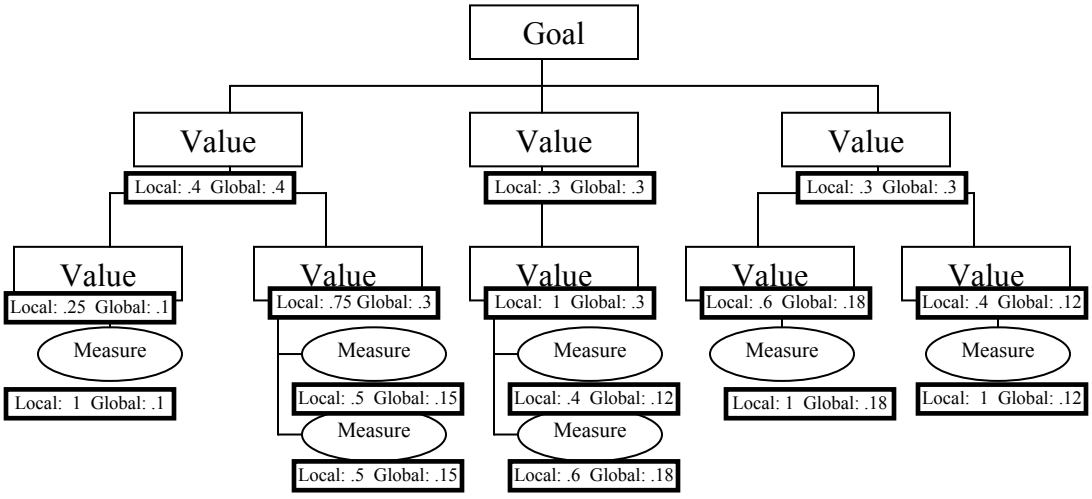

Figure 11: Example of weighting

There are several ways to weight a hierarchy. One method is the "100 balls" method. In this method, the decision maker is asked to divide the 100 balls among a particular tier and branch, each ball signifying a degree of value. In the example above, on the second tier of the first branch, 25 balls would be given to first value where the remaining 75 balls would be given to the second value.

Using another method, the least significant value would be given a value of "x". The remaining values would be given a value in multiple to how they relate to minimum value. All of these values must then sum to 1, so "x" can then be solved for. Using the same example from above, the first value would be given a value of "x" and the second a value of "3x". Since these two values must sum to 1, the following equation can be written: x + 3x = 1. This give 4x = 1. When solving for x it can be seen x = ¼ or .25, which is the value in the Figure 11 above. This also shows the second value gets a weight of 3x or .75 which is also the value from above.

Step 6 – Alternative Generation: Once the hierarchy has been developed and weighted, alternatives must be generated to be scored against the hierarchy to aid the decision maker in the decision process. Keeney states, "…alternatives should be created that best achieve the value specified for the decision situation...." He also says, "Alternatives themselves can trigger thought processes that generate new alternatives. (Keeney, 1992)" Alternatives must be able to be measured using the measures from the developed hierarchy. Although generating alternatives may seem easy, many problems can arise at this step. Kirkwood shows some of these possible problems and possible solution to these problems it the following table.

Table 1:

Alternative Dilemma	Proposed Solutiion(s)
Too many alternatives (Combinatorial problems)	Mathematical programming or optimization routines (e.g., integer linear programming).
Too many alternatives (Data collection problems)	Screening criteria capturing all probable alternatives so the most prefered alternative meets the driteria with ease.
	Strategy generation table to highlight which alternatives make sence and deserve a more detailed look.
Too few alternatives	Strategy generation table to highlight other potential column entries that may result in better alternatives.
	Develop a value hierarchy, if not already accomplished, and think of alternatives to maximize a higher-tier value.
Developing alternatives where there is uncertainty	Hedge against uncertainty by taking the middle ground.
	Allow for sequential decision in the future.
	Share the risk generated by the uncertainty with a partner.

(Kirkwood, 1997)

Step 7 – Alternative Scoring: Once alternatives have been generated, the alternatives must be scored against the measures. Scoring the alternatives is typically done by getting the decision maker and the SMEs to come to a consensus on where each alternative lies on the x-axis for each measure. Having many people come to a consensus is beneficial because it can help eliminate some of the individual bias that occurs naturally.

Step 8 – Deterministic Analysis: Once each alternative is scored for alternative, deterministic analysis must then be performed. Deterministic analysis mathematically combines the results from the alternative scoring of each measure into one final score or measure or as Kirkwood (1997) says, it "combines the multiple evaluation measures into a single measure." This is done by taking the score of each evaluation measure,

multiplying that score by the global weight given to that measure, and then adding all these amounts together.

Mathematically, the equation is as follows:

$$v(x) = \sum_{i=1}^{n} \lambda_i v_i(x_i) \quad (1)$$

where v(x) is the overall value, between 0 and 1, for an alternative, $v_i(x_i)$ is the score that the alternative received on measure i, and λi is the global weight associated with measure i. Once deterministic analysis is done, the final scores for each alternative tells the total amount of value that the alternative has achieved, 0 being the least amount possible and 1 being the alternative is given the most possible value possible towards the decision.

Step 9 –Sensitivity Analysis: Once deterministic analysis is done, the alternatives can then be ranked by there score. Sensitivity analysis can then be performed to determine the "impact on the ranking of alternatives [based on] changes in the modeling assumptions." (Kirkwood, 1997) Sensitivity analysis can be done on any part of the hierarchy, but typically it is done on the weights. Sensitivity analysis shows the decision maker where the weights would have to be changed to impact the decision. When changing the values of the weights it is important that the total sum of the weights still sum to 1. This analysis shows the decision maker the range at which weights can be for an alternative to be chosen. This aids the decision maker in case he/she is uncertain of certain weights or if the group had difficulty coming to a consensus on any weights.

Step 10 – Conclusion and Recommendations: After the deterministic analysis and sensitivity analysis have been concluded, the findings are then presented to the decision maker. This conclusion is not a solution for the decision maker, but is merely a tool that provides insight to the decision maker. If cost is an issue in the decision, cost/benefit analysis can be done to provide additional insight to the decision maker. However, cost should not be included in the hierarchy itself because the value of money stems from the benefits it brings, not the actual dollar value itself. With the findings of the analysis, the decision maker is empowered to make an informed decision.

Chapter 3. Methodology

3.0 Overview

The 10-Step VFT process described in Chapter 2.5 can be applied in focusing on the problem at hand. Using this Ten-Step process taught by AFIT, a group—including three SMEs, the decision maker and DA experts— identified what is valued in software interfaces for members of the intelligence community. Two of the SMEs had expertise in intelligence analysis while the third SME's expertise was in design interface. Through a series of meetings, the first seven steps of the Ten-Step process were completed. This chapter documents the results of those steps.

3.1 Step 1 – Problem Identification

During the initial meeting of the group, the first task at hand was to clearly identify the problem. The decision maker is responsible for ensuring the development of the JavaBase software interface is done in a way that is useful to the intelligence community. This prompted the question of what is valued by the users of the software interface, the intelligence analysts. Thus, as the development of the hierarchy began, the fundamental objective was to identify what is valued in a software interface for a complex analytical domain.

3.2 Step 2 – Construct Value Hierarchy

Once the problem had been clearly identified, the next step was to capture what aspects of the software interface were valuable for efficiency and effectiveness. The analysts proposed three main components: the input process of software, the processing part of software, and the output process of software. These components fit the analysts'

mental model of their work and became the natural breaking points. These natural breaks occurred because separate members of the intelligence community commonly do each part separately and became the first, or top, tier of the hierarchy. Defining these three main components was the first step in breaking the hierarchy down into measurable elements. Each of the three main components then divides into smaller parts that formed the larger category. This breakdown is represented below in Figure 12. The rationale for the breakdown of each component will be discussed later in this chapter.

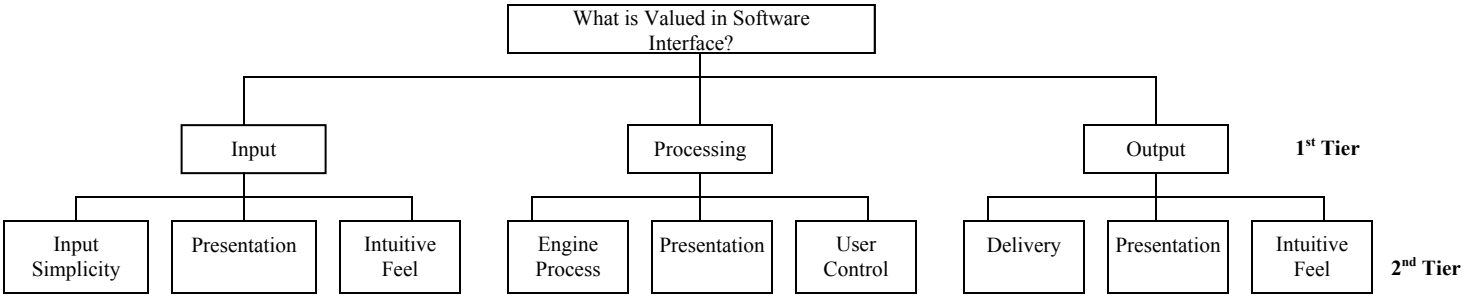

Figure 12: Top tiers in hierarchy development

Values such as presentation and intuitive feel appear in more than one branch giving the appearance of a violation of the need for mutual independence. However, since the Input, Processing, and Output components are mutually independent, the same values can be present in each branch and still remain independent. Therefore "Presentation" in Input is independent of "Presentation" in the Processing and Output branch.

Each of the values in the second tier was then broken down into the values that they encompass. A complete analysis will be conducted by examining the values of each of these branches and their values in greater detail.

3.2.1 The Input component

The Input component of software interface is defined as the way a software interface supports a person in the ability to input data into the software. Input is broken down into the following three values: Input Simplicity, Input Presentation and Input Intuitive Feel. These represent the desirability of an interface to make the input process as easy as possible; valuing an interface that presents the data and feedback in a pleasing way and enables the user to feel as if the software's interface is familiar to use. The Input component is especially important because any difficulty the user has at this stage may prevent him/her from continuing to use the software. The breakdown of this branch is shown in the Figure 13 and defined in Table 2.

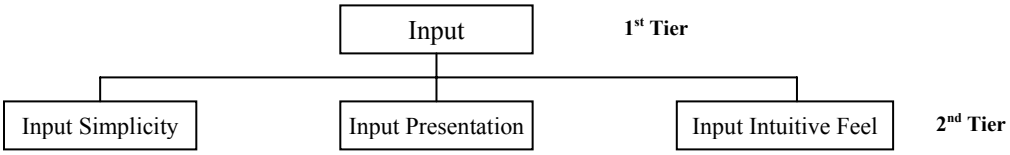

Figure 13: Input top tier values

Table 2: Definition of Input top tier values

Input	The way a software interface aids a person in the ability to enter/import data into the software.
Input Simplicity	The ease of entering data into the software; focuses on supporting the user to ensure accurate and efficient data entry.
Input Presentation	The way the interface displays data and feedback to the user in the input process.
Input Intuitive Feel	How the user to feels as if he/she understands how the interface works and how to input data.

Similarly, each second tier value was broken down further into third and fourth tiers. The breakdown of these values can be seen in below in Figures 14-16 and the definitions of each value can be found in Appendix A, Tables 14-16.

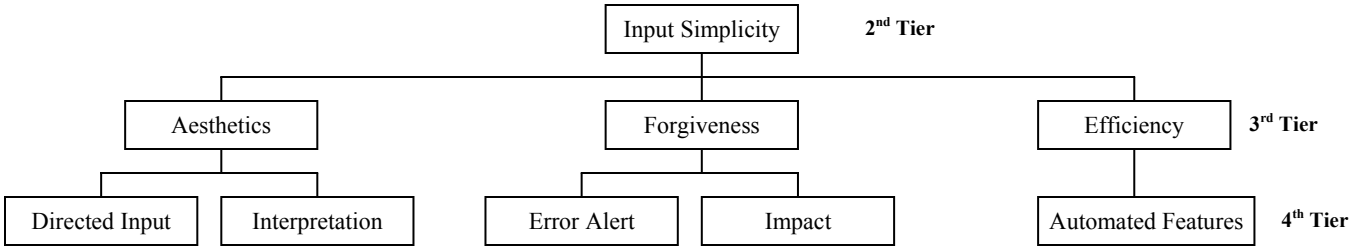

Figure 14: Input Simplicity Breakdown

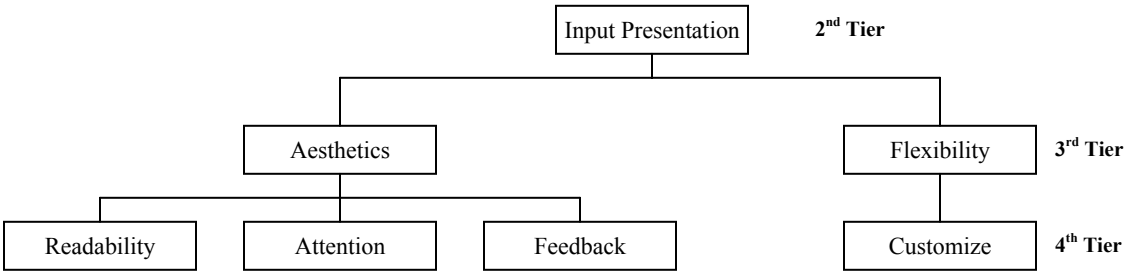

Figure 15: Input Presentation Breakdown

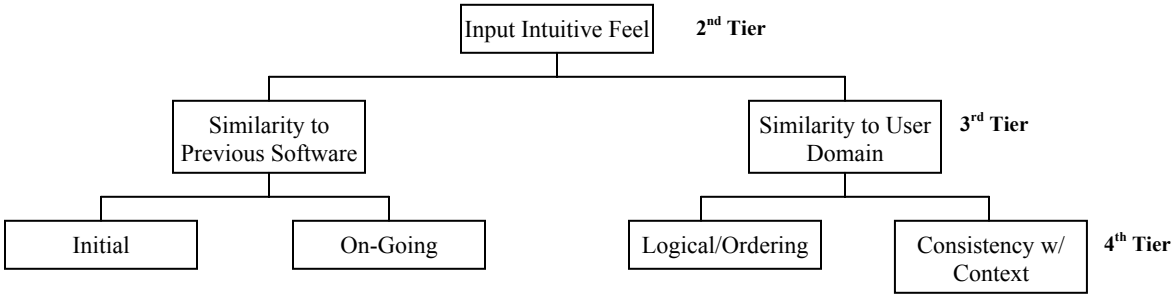

Figure 16: Input Intuitive Feel Breakdown

3.2.2 The Processing Component

The second of the first tier components in the interface hierarchy involves the processing of data in the software. The processing component of software interface is defined as the ability of the software interface to aid the user in his/her ability to process and analyze, e.g. to aid the user to get the answer. The processing component is broken down into the following three values: Engine Process, Processing Presentation and User Control. Engine Process is significant to the intelligence community because the analysts must be able to understand the algorithms being used in the software as well as verify that the data is correct. This ability to understand how the software calculates the data correctly is critical because the intelligence community users need to make important decisions with this software. Presentation continues to be valued for the same reasoning as given in the Input process. Finally, User Control is valued because of the user's ability to select their preferred options.

The breakdown of this branch is shown in the Figure 17 and the values are defined in Table 3.

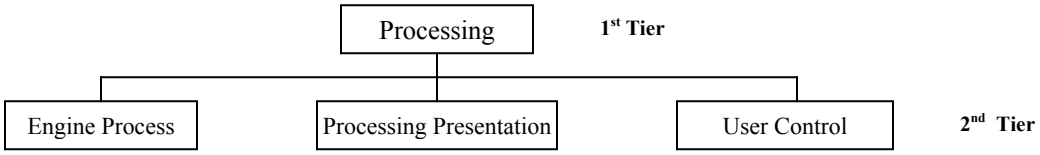

Figure 17: Processing top tier values

Table 3: Definitions of Processing top tier values

Processing	To aid the user in his/her ability to process and analyze, e.g. to aid the user to get the answer
Engine Process	The ability to display what the engine is doing.
Processing Presentation	To display the data in way that makes it easier for the user to process the data.
User Control	Ability to control how the processing is done in a way the suitable to the user. **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**

The second tier values of Engine Processing, Processing Presentation, and User Control were broken down as shown in Figures 18-20 with the definitions of each value in Appendix A, Tables 17-19.

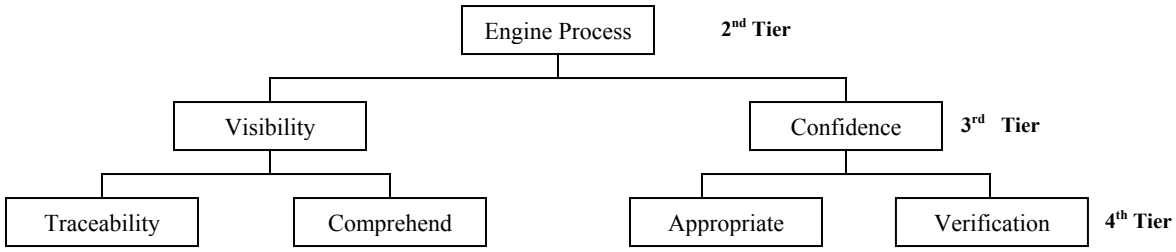

Figure 18: Engine Process Breakdown

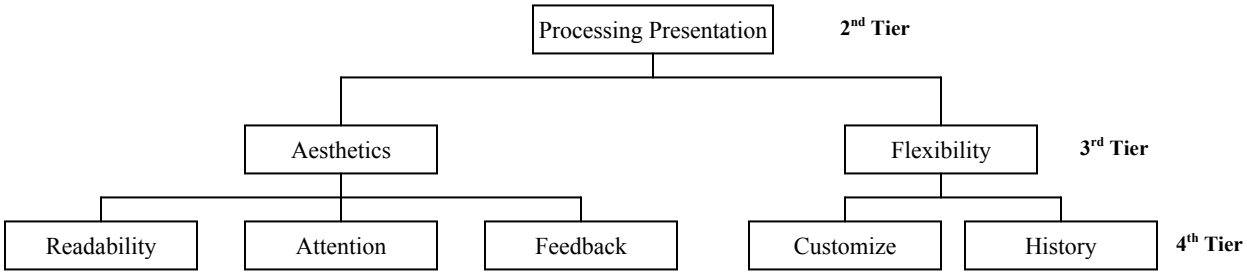

Figure 19: Processing Presentation Breakdown

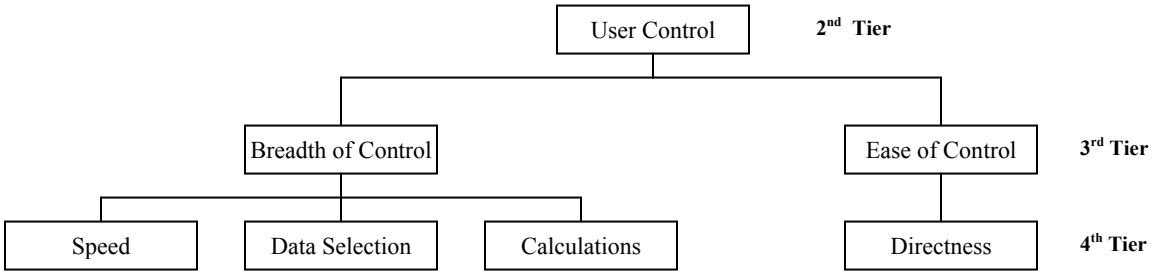

Figure 20: User Control Breakdown

3.3.3 The Output Component

The final first tier component in the interface concerns the outputting of data in the software. Output is defined as the ability of the interface to aid the user in his/her ability to output the data from the software once the processing and analysis is done, e.g. the user has the answer and must now present and/or give it to the customer. The Output component is broken down into the following three values: Delivery, Output Presentation and Output Intuitive Feel (Figure 21). The values of Output Presentation and Output Intuitive Feel parallel the detail for Presentation and Intuitive Feel in the Input section. Users value Delivery, the capability to send the information to the customer in a variety of ways. Output is particularly important because if the user is unable to obtain the computed data in a format that is useful to him, he will not find overall value to the software. The breakdown of this branch is shown in the Figure 21 and the values of each defined in Table 4.

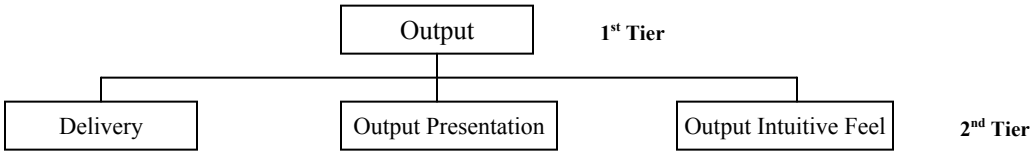

Figure 21: Output top tier values

Table 4: Definitions for Output top tier values

Output	The ability to output the data from the software once the processing and analysis is done, e.g. the user has the answer and must now present and give it to the customer
Delivery	To allow the user to deliver the data to the customer in a particular way
Output Presentation	To display the data in way that makes it easier for the user to present the data to the customer.
Output Intuitive Feel	To permit the user to feel as if he understands how the interface works and how to output data.

The breakdown of these second tier values can be seen in below in Figures 22-24 and the definitions of each value can be found in Appendix A, Tables 20-22.

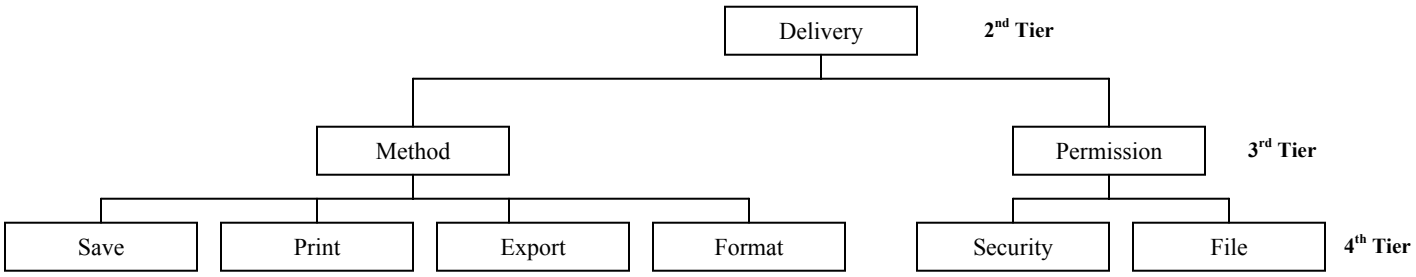

Figure 22: Delivery Breakdown

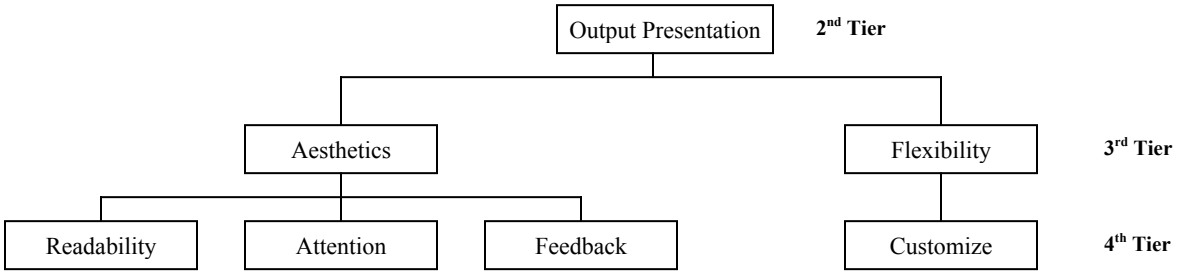

Figure 2: Output Presentation Breakdown

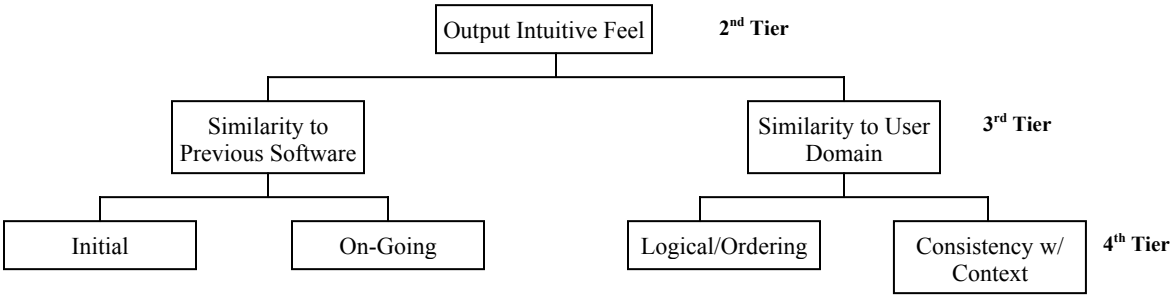

Figure 23: Output Intuitive Feel Breakdown

3.3 Step 3 -Develop Evaluation Measures

With all of the values in the hierarchy developed and defined, a way of evaluating or measuring these values is needed. Because each tier of the hierarchy encompasses the subordinate tiers, measures need to be developed only for the fourth and final tier of the hierarchy. Developed value measures are developed represent the most detailed level of the hierarchy. The measures quantify the values in order to objectively evaluate the alternatives. The following sections break down each measure and identify where each measure has been added to hierarchy. Appendix B defines the definition of each measure in Tables 23-25.

3.3.1 Input Measures

The Input component of the hierarchy was developed to encompass all values that the user would like in an interface while inputting data. The values of this branch were provided in Figures 14-16, but evaluation measures are needed for the fourth-tier values.

Figures 24-26 display the Input branches with the measures added.

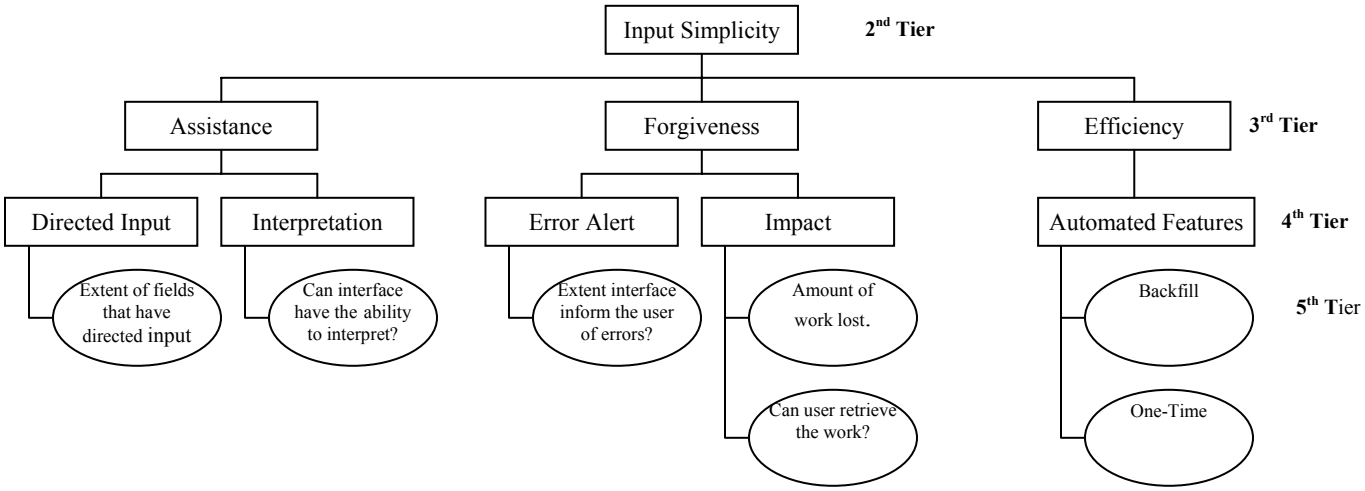

Figure 24: Input Simplicity break down with measures

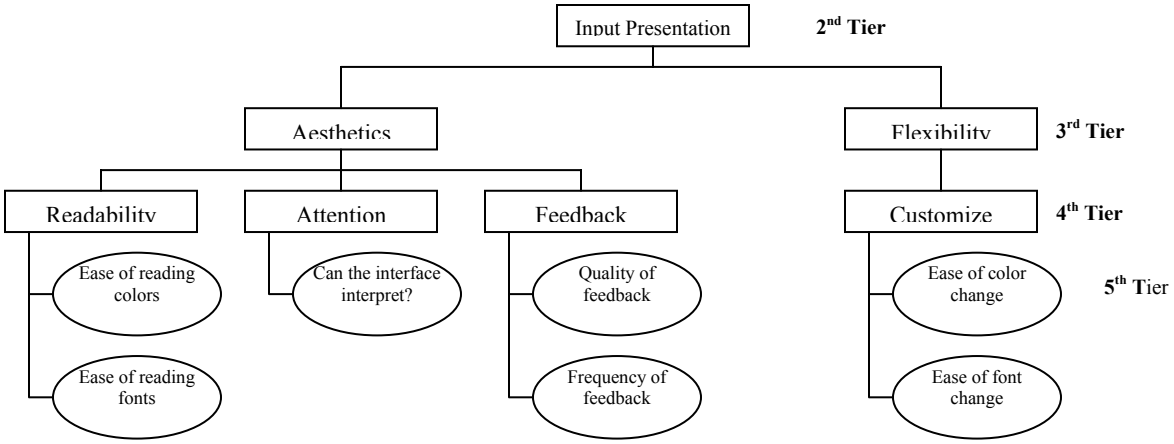

Figure 25: Input Presentation break down with measures

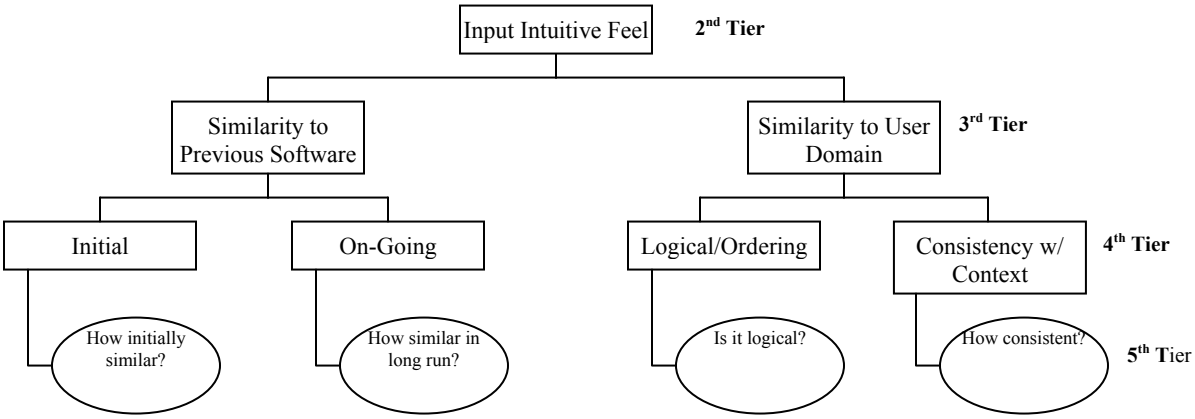

Figure 26: Input Intuitive Feel break down with measures

Table 5 identifies the values in the fourth tier in the Input component, the associated measures to each value, and the limits (the worst and best case) for each measure. The definitions are found in Table 23 of Appendix B.

Table 5: Input fourth tier values and corresponding measures

Fourth-Tier Hierarchy Value	Associated Measure	Lower Bound	Upper Bound
Directed Input	Extent of fields that have directed input	None	Majority
Interpretation	Does the interface have the ability to interpret?	No	Yes
Error Alert	Extent interface inform the user of errors?	None	Majority
Impact	Amount of work lost.	Majority	None
	Can user retrieve the work?	No	Yes
Automated Features	Backfill	No	Yes
	One-Time	No	Yes
Readability	Ease of reading colors.	Very difficult	Clear/Easy
	Ease of reading fonts.	Very difficult	Clear/Easy
Attention	Can the interface emphasize?	No	Yes
Feedback	Quality of Feedback	Vague/Unhelpful	Specific/Helpful
	Frequency of Feedback	None	Majority
Customize	Ease of color change.	Very difficult	Clear/Easy
	Ease of font change.	Very difficult	Clear/Easy
Initial	How initially similar?	Not Similar	Very Similar
On-Going	How similar in long run?	No	Yes
Logical/Ordering	Is it logical?	No	Yes
Consistency w/ Context	How consistent?	Not consistent	Is consistent

3.3.2 Processing Measures

The values of the Processing branch have been identified, but evaluation measures are needed for the values in the fourth-tier. Figures 27-29 display the Processing component of the hierarchy with the measures added.

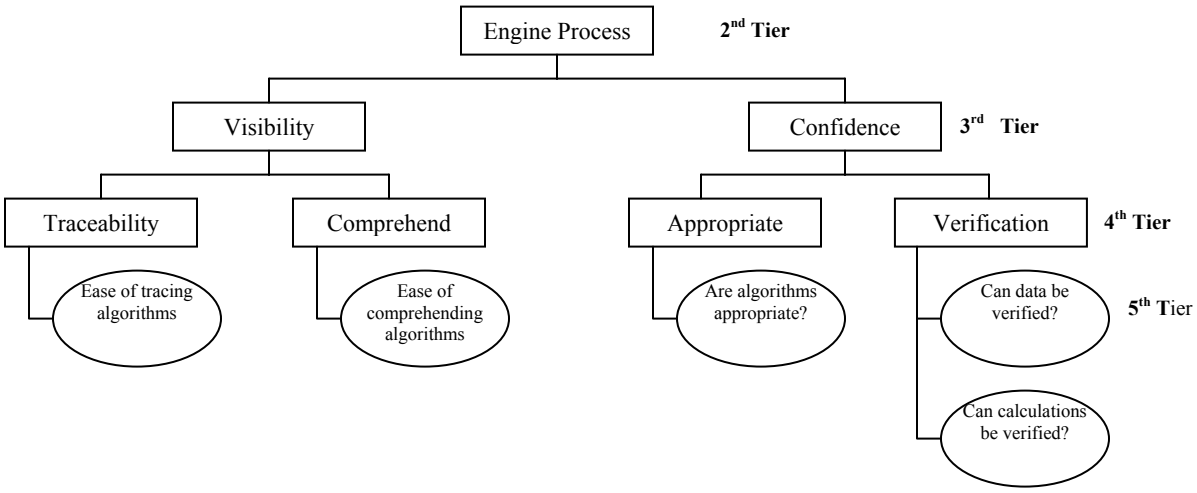

Figure 27: Engine Process break down with measures

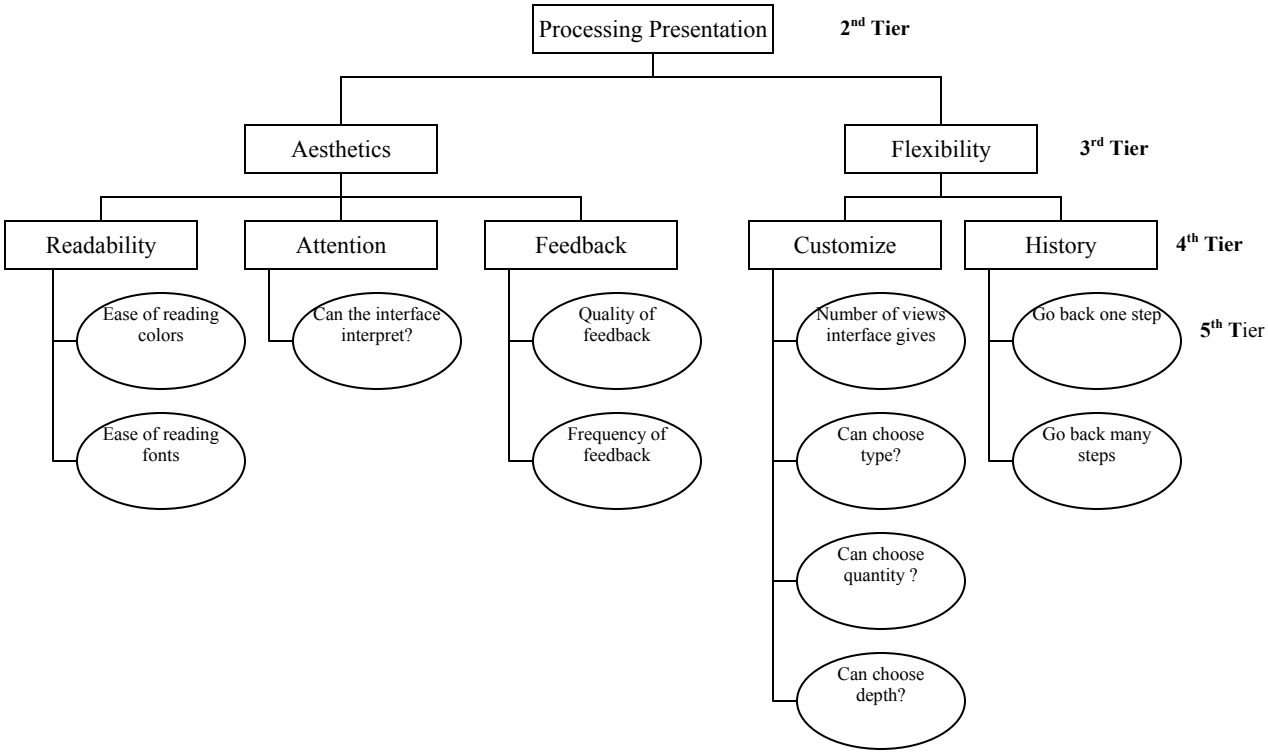

Figure 28: Processing Presentation break down with measures

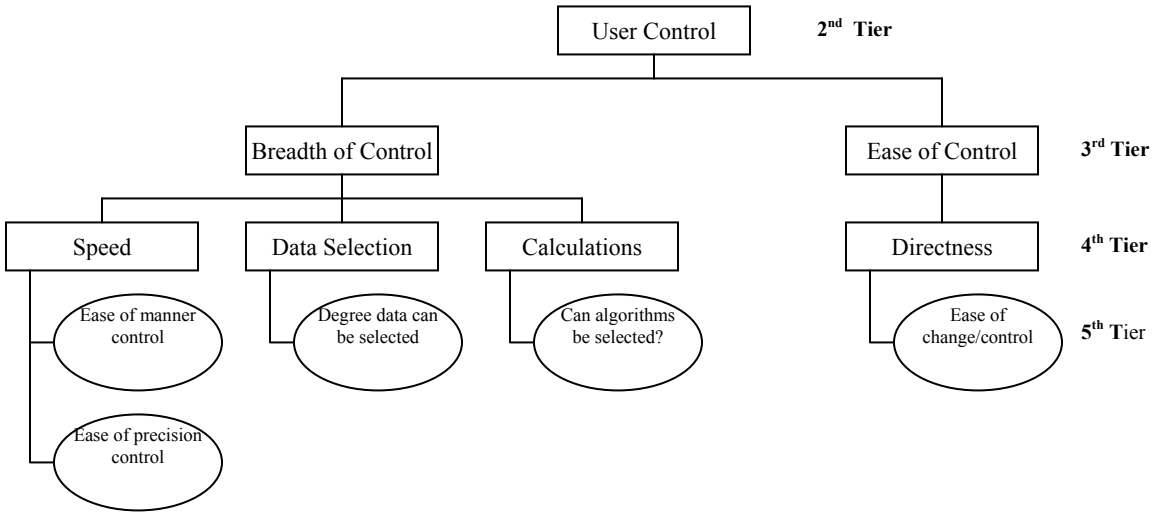

Figure 29: User Control break down with measures

Table 6 identifies the values in the fourth tier, the associated measures to each value, and the worst and best case for each measure. The definitions are found in Table 24 of Appendix B.

Table 6: Processing fourth tier values and corresponding measures

Fourth-Tier Hierarchy Value	Associated Measure	Lower Bound	Upper Bound
Traceability	Ease of tracing algorithms	Can't Trace	Easy to Trace
Comprehendible	Ease of comprehending algorithm	No Explanation	Highly Specific
Appropriate	Are algorithms appropriate	No	Yes
Verification	Can data be verified?	No	Yes
	Can calculations be verified	No	Yes
Readability	Ease of reading colors.	Very difficult	Clear/Easy
	Ease of reading fonts.	Very difficult	Clear/Easy
Attention	Can the interface emphasize?	No	Yes
Feedback	Quality of Feedback	Vague/Unhelpful	Specific/Helpful
	Frequency of Feedback	None	Majority
Customize	Number of views interface gives.	None	Many
	Can choose type?	None	Many
	Can choose quantity?	None	Many
	Can choose depth?	None	Many
History	Go back one step	No	Yes
	Go back many step	No	Yes

Speed	Ease of manner control	No	Yes w/Flexibility
	Ease of precision control	No	Yes w/Flexibility
Data Selection	Degree data can be selected	No	Yes w/Flexibility
Calculations	Can algorithm be selected	No	Yes
Directness	Ease of Changes/Control	Difficult to Control	Easy to Control

3.3.3 Output Measures

The values of the Output branch have been identified, but evaluation measures are needed for the values in the fourth-tier. Figures 30-32 display the Output hierarchy with the measured added.

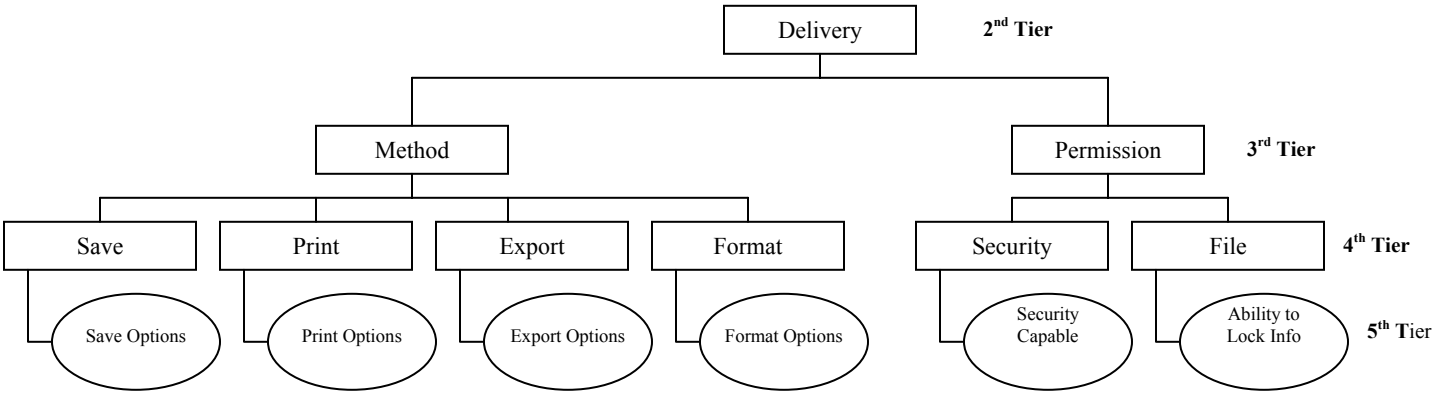

Figure 30: Delivery break down with measures

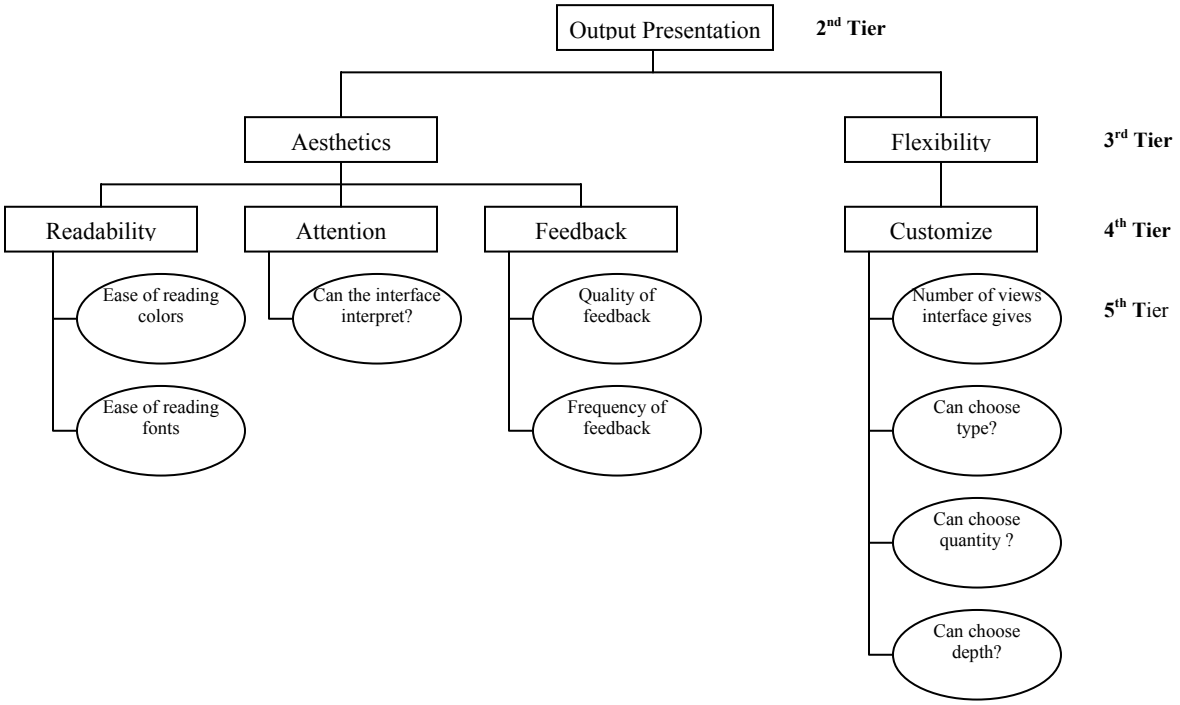

Figure 31: Output Presentation break down with measures

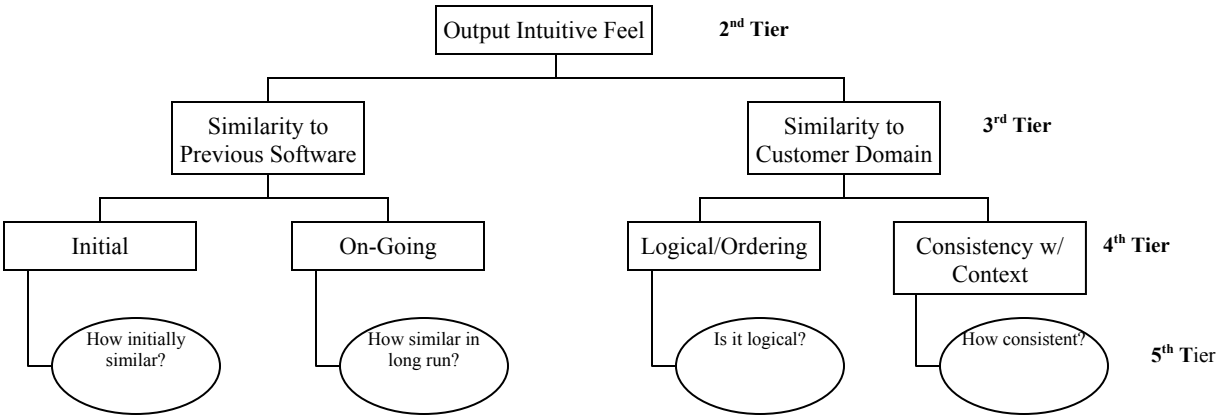

Figure 32: Output Intuitive Feel break down with measures

Table 7 identifies the values in the fourth tier, the associated measures to each value, and the worst and best case for each measure. The definitions are found in Table 25 of Appendix B.

Table 7: Output fourth tier values and corresponding measures

Fourth-Tier Hierarchy Value	Associated Measure	Lower Bound	Upper Bound
Save	Save options	Doesn't meet standards	Meets standards
Print	Print options	Doesn't meet standards	Meets standards
Export	Export options	Doesn't meet standards	Meets standards
Format	Format options	Doesn't meet standards	Meets standards
Security	Security Capable	No	Yes
File	Ability to lock info?	No	Yes w/Flexibility
Readability	Ease of reading colors.	Very difficult	Clear/Easy
	Ease of reading fonts.	Very difficult	Clear/Easy
Attention	Can the interface emphasize?	No	Yes
Feedback	Quality of Feedback	Vague/Unhelpful	Specific/Helpful
	Frequency of Feedback	None	Majority
Customize	Number of views interface gives.	None	Many
	Can choose type?	None	Many
	Can choose quantity?	None	Many
	Can choose depth?	None	Many
Initial	How initially similar?	Not Similar	Very Similar
On-Going	How similar in long run?	No	Yes
Logical/Ordering	Is it logical?	No	Yes
Consistency w/ Context	How consistent?	Not consistent	Is consistent

3.4 Step 4 – Create Value Functions

Once the evaluation measures were established, single-dimension value functions (SDVFs) were developed in face-to-face meetings with the SMEs. The SDVF measures the degree of value that the given measure provides towards the end goal. Each SDVF gives a value from 0 to 1. All of the measures have a categorical SDVF, meaning the scale is not continuous, but grouped into categories. Each SDVF is monotonically increasing, as can be seen in the sample SDVF, Figure 33. This figure illustrates the desirability of having directed input in the interface. The x-axis for the SDVF identifies the amount of fields that have directed input in the Input component. The categories show that having a greater number of fields with directed input is increasingly desirable. The y-axis for the SDVF identifies the value each category receives with a 0-1 scale. The SDVF shows that the greatest increase in value to the user comes when an interfaces goes from having Little to Some directed input as evidenced by the weight assigned to each category. In contrast, an increase from None to Little directed input has a relatively small increase in value.

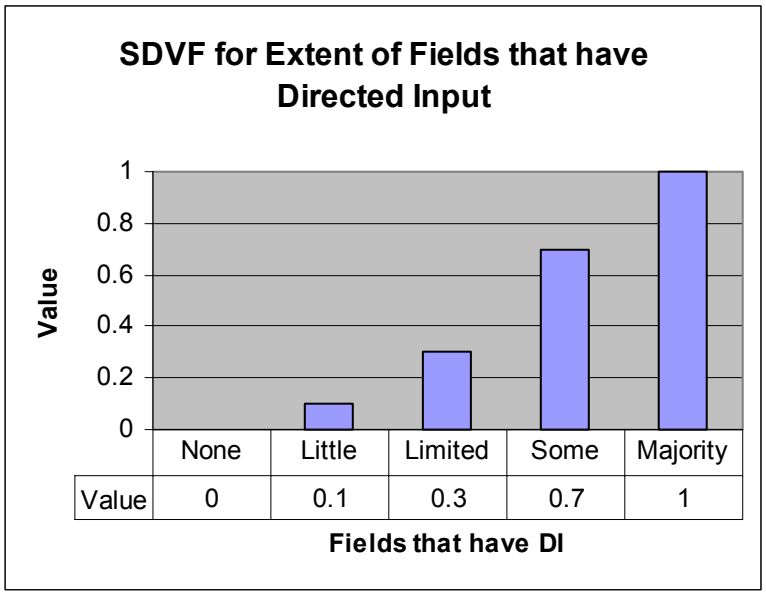

Figure 33: SDVF for Extent of fields that have Directed Input

Table 8 gives the definition of each category that is found on the x-axis of Figure 33. These definitions provide a quantitative set of standards to capture the qualitative categories. The SDVFs for the 58 measures are given in Figures 45-102 and their corresponding definitions for their categories are given in Tables 26-83 in Appendix C.

Table 8: Definitions of categories for Extent of fields that have Directed Input

Category	Definition
None	Having no directed input.
Little	Having some directed input, but less than or equal to 10% of the fields having it.
Limited	Having greater the 10% directed input, but less than or equal to 50% of the fields having it.
Some	Having greater the 50% directed input, but less than or equal to 80% of the fields having it.
Majority	Having greater the 80% directed input.

3.5 Step 5 – Weight the Value Hierarchy

The complete hierarchy must now be weighted because each value does not have equal weight throughout the hierarchy. Ideally, the weighting of this hierarchy would be done by its intended user, but since this is not known, the decision maker and the SMEs served as proxy users. The intelligence SMEs were primarily used in the weighting because they provided the closest approximation of the intended user's value system. The weighting was done locally, that is, done within every branch on each tier of the hierarchy. Figure 34 shows the local weighting in the top two tiers with the global weights in parentheses next to the local weight. The figure shows that Input and Output

are equally important and are each slightly more important than Processing. This finding may at first seem surprising when considering how important the Processing component is to the analyst and his work. This is not to suggest that the analyst does not find the Processing component important, but demonstrates how critical the ability to easily input data and the capability to present the data to a customer is to the overall success of a software and its interface.

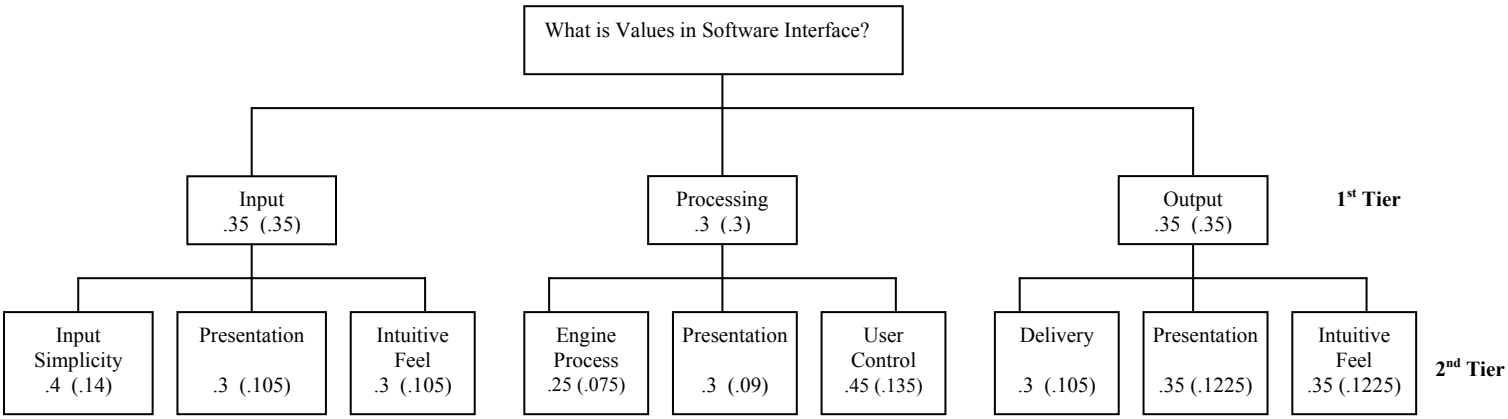

Figure 34: Top tiers weighting

After weighting the first two tiers, weighting was done down each branch on the second tier values. The weight given to the engine process in the second tier is surprising since the intelligence SMEs stressed the importance of an interface having the ability to show the analyst what the software is doing, yet it has the smallest global of all the second tier values. However, the analysts commented that once they achieved confidence in the Engine Process the value they placed on it in comparison to other values decreased. Input Simplicity was identified as the most important of the second tier values which reinforces the fact that a software interface that is difficult to use initially

has little overall value. The results of the weighting of the remaining hierarchy can be seen in Figures 103-111 in Appendix D.

3.6 Step 6/Step 7 – Alternative Generation/Alternative Scoring

Step six of the ten-step process is alternative generation. However, in this case there are no alternatives to be generated as only one prototype exists. The purpose of this study was to more generically identify what is valued in software interface to members of the intelligence community. With the identification of these values, the developers of the JavaBase program will then be able to identify what improvements need to be made to their existing interface. Since no alternatives can be generated, only the existing software interface was scored against the hierarchy's measures. The interface will be scored against the measures in the Input, Processing, and Output components. It is also important to note that the existing prototype was in its early stages and was not very developed.

3.6.1 Input Scoring

Scoring of the baseline interface was done against the measures of the Input, Processing and Output components. Table 9 identifies the measures, the associated possible categories of each measure, and which category was selected along the x-axis in the Input component.

Measure	Selected
Extent of fields that have directed input	
None	
Little	X
Limited	
Some	
Majority	
Does the interface have the ability to interpret?	
No	X
Yes	
Extent interface inform the user of errors?	
None	
Little	X
Limited	
Some	
Majority	
Amount of work lost.	
Majority	X
Some	
Limited	
Little	
None	
Can user retrieve the work?	
No	X
Yes	
Backfill	
No	X
Yes	
One-Time	
No	X
Yes	
Ease of reading colors.	
Very Difficult	
Somewhat Difficult	
Somewhat Easy	X
Clear/Easy	
Ease of reading fonts.	
Very Difficult	
Somewhat Difficult	
Somewhat Easy	X
Clear/Easy	

Measure	Selected
Can the interface emphasize?	
No	X
Yes	
Quality of Feedback	
Vague/Unhelpful	X
Somewhat Vague	
Somewhat Specific	
Specific/Helpful	
Frequency of Feedback	
None	
Little	
Limited	
Some	X
Majority	
Ease of color change.	
Very Difficult	X
Somewhat Difficult	
Somewhat Easy	
Clear/Easy	
Ease of font change.	
Very Difficult	X
Somewhat Difficult	
Somewhat Easy	
Clear/Easy	
How initially similar?	
Not Similar	
Limited Similarity	
Some Similarity	X
Very Similar	
How similar in long run?	
No	X
Yes	
Is it logical?	
No	X
Yes	
How consistent?	
Not Consistent	
Some Consistency	X
Is Consistent	

Table 9: Input Scoring

3.6.2 Processing Scoring

With the scoring of the Input component done, the Processing component was then scored. The Table 10 identifies the measures, the associated possible categories of each measure, and which category was selected along the x-axis in the Processing component.

Measure	Selected
Ease of tracing algorithms	
Can't Trace	X
Difficult to Trace	
Easy to Trace	
Ease of comprehending algorithm	
No Explanation	X
Limited Explanation	
Some Explanation	
Highly Specific	
Are algorithms appropriate	
No	X
Yes	
Can data be verified?	
No	X
Yes	
Can calculations be verified	
No	X
Yes	
Ease of reading colors.	
Very Difficult	
Somewhat Difficult	X
Somewhat Easy	
Clear/Easy	
Ease of reading fonts.	
Very Difficult	
Somewhat Difficult	
Somewhat Easy	X
Clear/Easy	
Can the interface emphasize?	
No	X
Yes	
Quality of Feedback	
Vague/Unhelpful	
Somewhat Vague	X
Somewhat Specific	
Specific/Helpful	
Frequency of Feedback	
None	
Little	

Measure	Selected
Number of views interface gives.	
None	X
Some	
Many	
Can choose type?	
None	X
Some	
Many	
Can choose quantity?	
None	X
Some	
Many	
Can choose depth?	
None	X
Some	
Many	
Go back one step	
No	X
Yes	
Go back many step	
No	X
Yes w/No Flexibility	
Yes w/Flexibility	
Ease of manner control	
No	X
Yes w/No Flexibility	
Yes w/Flexibility	
Ease of precision control	
No	X
Yes w/No Flexibility	
Yes w/Flexibility	
Degree data can be selected	
No	X
Yes w/No Flexibility	
Yes w/Flexibility	
Can algorithm be selected	
No	
Yes	X
Ease of Changes/Control	

Limited	X
Some	
Majority	

Very Difficult	
Somewhat Difficult	
Somewhat Easy	X
Clear/Easy	

Table 10: Processing Scoring

3.6.3 Output Scoring

With the scoring of the Input component done, the Processing component was then scored. The Table 11 identifies the measures, the associated possible categories of each measure, and which category was selected along the x-axis in the Processing component.

Measure	Selected
Save options	
Doesn't meet Standards	X
Meets Standards	
Print options	
Doesn't meet Standards	X
Meets Standards	
Export options	
Doesn't meet Standards	X
Meets Standards	
Format options	
Doesn't meet Standards	X
Meets Standards	
Security Capable	
No	X
Yes	
Ability to lock info?	
No	X
Yes w/No Flexibility	
Yes w/Flexibility	
Ease of reading colors.	
Very Difficult	X
Somewhat Difficult	
Somewhat Easy	
Clear/Easy	
Ease of reading fonts.	
Very Difficult	X
Somewhat Difficult	
Somewhat Easy	

Measure	Selected
Frequency of Feedback	
None	X
Little	
Limited	
Some	
Majority	
Number of views interface gives.	
None	X
Some	
Many	
Can choose type?	
None	X
Some	
Many	
Can choose quantity?	
None	X
Some	
Many	
Can choose depth?	
None	X
Some	
Many	
How initially similar?	
Not Similar	X
Limited Similarity	
Some Similarity	
Very Similar	
How similar in long run?	

Clear/Easy	
Can the interface emphasize?	
No	X
Yes	
Quality of Feedback	
Vague/Unhelpful	X
Somewhat Vague	
Somewhat Specific	
Specific/Helpful	

No	X
Yes	
Is it logical?	
No	X
Yes	
How consistent?	
Not Consistent	X
Some Consistency	
Is Consistent	

Table 11: Output Scoring

3.7 Summary

This chapter demonstrated the extensive process that was undertaken to identify the values, a hierarchical structure, and the measures that were developed. Each of these values was defined, as well as an explanation given for each developed SDVF that was developed. The definitions support the hierarchy and have the attributes of mutual exclusivity and collective exhaustiveness. Additional details regarding the results may be found in Appendices A-D. With steps one through seven completed, deterministic analysis was performed.

Chapter 4. Results and Analysis

4.0 Overview

This chapter contains the results of the deterministic analysis. Since the software interface that was evaluated in its early stages of development, only one alternative—the baseline—was scored. Without different alternatives, sensitivity analysis will not be performed. However, in order to fulfill the research objective, analysis to discover where enhancements to the interface can be made was performed by identifying value gaps. A value gap is the amount of possible improvement a measurement can give to the overall value of the problem.

4.1 Step 8 – Deterministic Analysis

With the hierarchy built and the baseline scored, the deterministic analysis was performed by taking the SDVF related to the score of each measure and multiplying it by the measure's global weight. These products were then summed to produce the total value or overall alternative score, as shown in Equation 1. A breakdown of the value each component gave towards the final goal as well as the combined, overall value the baseline received is given in Appendix E. The total score of each component is given in the following section.

4.1.1 Total Value Baseline Received

Table 12 gives a breakdown of the value each component contributed to the overall goal. The breakdown shows the total possible value each component can contribute to the goal as well the value that the baseline actually received.

Table 12: Total Baseline Scoring

	Total possible value component	Value received component
Input	.35	0.05
Processing	.3	0.08
Output	.35	0.0
Total value of baseline	**1**	**0.13**

The score of 0.13 received by the baseline seems to be very small, indicating that the baseline is very weak. However, this prototype software is in the early development stages and a great deal of work remains to be done before completion of the interface. Specifically the Output component has not been developed at all.

4.2 Identification of Value Gaps

The deterministic analysis identified the value that each measure contributed to the overall goal as well the value each component of hierarchy contributed to the overall goal. With this information, the gaps in the value for each measure were identified. The value gap is defined as the potential improvement in score for each measure. The identification of these gaps can aid the software developers to identify where to focus the development of the interface. Since many of the measures did not score the full value of one, many of the measures have these gaps in value. The gap is computed by taking the maximum possible score the value can receive, which is one, and subtracting the score the measure received. Since multiple measures have value gaps, they were prioritized by

the global weights, which allows for ranking of the importance of the value gap of each measure with respect to each other. Multiplying the global weight by the value gap shows the possible improvement by a measure. The possible improvement can be given by the mathematical equation that follows:

$$\lambda_i \cdot \left[1 - v_i \cdot \left(x_i \right) \right] \tag{2}$$

where $v_i\left(x_i\right)$ is the value that the alternative received on measure i, λ_i is the global weight associated with measure i, and $\left[1 - v_i \cdot \left(x_i \right) \right]$ defines the value gap. The importance of the measures will be broken down in each component, and the value gaps of each measure will be identified. Table 13 shows the top seven measures with the most possible improvement in value. For the list of all 58 value gaps, their rank, and their potential improvement in value see Tables 87-89 in Appendix F.

Table 13: Top measures with most possible improvement

$\lambda_i \cdot \left[1 - v_i \cdot \left(x_i \right) \right]$ **Possible improvement**	Measure (Branch)
0.047775	Is it logical? (Output)
0.04095	Is it logical? (Input)
0.03185	How initially similar? (Output)
0.0294	One-Time (Input)
0.02646	Extent of fields that have directed input (Input)
0.02625	Security Capable (Output)
0.02625	Ability to lock info? (Output)

It is important to note that six of the seven measures shown in Table 13 received a score of 0, while the other received a score of 0.1. The majority of these measures are part of Output which had not been fully developed. Decreasing the value gap of these measures at the top would increase the overall value the most, with diminishing value as the list is gone down. These measures, if improved, will have the greatest increase in the

overall value given to improving the interface. The top two values in Table 13 demonstrate how important it is for a software interface to be structured logically in order to increase its value.

4.3 Summary

This chapter provided the deterministic analysis for the baseline of the interface, finding the score to be a 0.13 out of a possible 1. The value gaps of each evaluation measure were also identified and demonstrated the need for an output component to the software. Although this analysis objectively evaluated the interface to be very lacking, the interface is only a prototype, so the results are not unexpected.

Chapter 5. Discussion

5.0 Overview

Chapter 5 culminates this thesis and provides conclusions of the VFT process that was used in the interface application. The initial objective of this thesis will be addressed, as well as insights and recommendations for the process. Also, future research suggestions will be made before final conclusions are reached.

5.1 Initial Objectives

The initial objective of this research was to provide the decision maker insight into what improvements were needed in the prototype JavaBase software. In order to provide this insight, the ten-step value-focus thinking process, as taught by AFIT, was used determine what the interface requires to be useful to the members of the intelligence community. Using a group that included the decision maker, subject matter experts and a decision analysis expert, a hierarchy was developed that portrayed the values in an interface for members of the intelligence community. This hierarchy was used to score the baseline as well as to determine where improvements in the interface can be made as evidenced through value gaps. This score as well as the value gaps provide essential information to aid the decision maker in the improvement of the JavaBase software interface.

5.2 Insights and Recommendations

As the software was in the early stages of development, the interface was also at an early stage of development. This early stage became apparent in the scoring of the baseline, scoring only 0.13 out of a possible 1.00. The lack of maturity became even

more apparent in Output because the SMEs determined there was no Output component. Therefore the whole Output component received a score of 0 out of 0.35.

The SMEs who scored the baseline had minimal experience with the software prior to scoring it. They were provided a brief demo that lasted approximately two hours, where they could ask questions until they felt comfortable enough to score the interface. This unfamiliarity with the software may also have affected the low score of the baseline. The SMEs scored the interface based on their knowledge of the software and its interface, however, if they were more familiar with it that may have prompted different scores. A recommendation may be to rescore the baseline after the SMEs have a better understanding of the software.

5.3 Future Research

With the hierarchy created, there are many ways to explore for the future. One possibility for future consideration would be to aid the developers in creating alternatives once they continue developing the software. When this is done and the alternatives have been created, the alternatives can then be scored against the software interface hierarchy. Then sensitivity analysis can be performed on the hierarchy.

Another possible option for consideration could be to train the SMEs in the JavaBase software and to determine if this has any effect on the overall score for the software. This would be particularly interesting once the end users of the software are determined by having the end users score the software's interface before and after they gain familiarity with the software.

Since this process was done specifically for software being developed for the intelligence community, taking a step back in developing the hierarchy for general software interfaces may be interesting. The hierarchy built specifically for the intelligence community had some functionality issues that do not necessarily apply to all interface situations. Using the literature from Microsoft, a possible "gold standard" hierarchy was developed in the next section.

5.3.1 Gold Standard Hierarchy

An extended area to explore is the creation of a hierarchy that can be used for software interface in general. This approach is referred to as a gold standard and follows pure doctrine. In it, the values will be taken from the literature to create the hierarchy, in particular from The Windows Interface Guidelines for Software Design. (Microsoft Press, 1995)

The principles used in Microsoft publications (user in control, directness, simplicity, feedback, forgiveness, consistency, and aesthetics) are similar to many of the values in the intelligence interface hierarchy developed in Chapter 3. In the Gold Standard Hierarchy, the principles are broken down into two parts, usability (user in control, directness, simplicity, feedback, forgiveness) and presentation (consistency and aesthetics). The top tier of this Gold Standard Hierarchy consists of Usability and Presentation. Under the Usability component are the values User In Control, Directness, Simplicity, Feedback, and Forgiveness. Under the Presentation component are the values Consistency and Aesthetics. Each of these values can then be broken down into similar values that were used in the interface hierarchy. This hierarchy could be used to evaluate the intelligence software interface by developing multiple evaluation measures

based on Input, Processing and Output components. An example of this is under the Readability of Fonts in Figure 35, the measures could be Readability of Input fonts, Readability of Processing fonts, and Readability of Output fonts. Figure 35 shows what this hierarchy may look like.

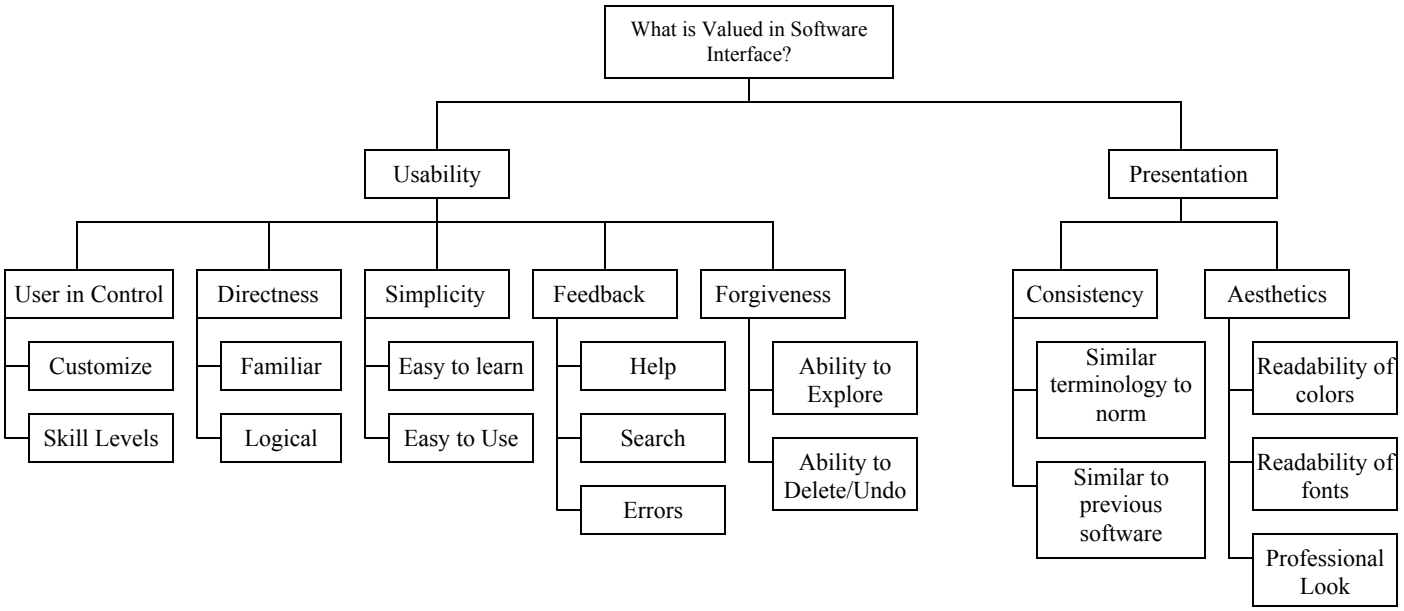

Figure 35: Gold Standard Hierarchy

The hierarchy above was created by merging the information given in the literature with the ideology behind this research.

5.4 Conclusions

The VFT process used is an appropriate method to evaluate the interface as well as to provide insight into possible enhancement of the interface. Although this methodology has both strengths and weaknesses, it has provided insight into the baseline interface of the JavaBase. This process can be used throughout stages of the software

development, providing insight to the decision maker throughout software development lifecycle. This insight can aid the decision maker to provide an interface that presents the key needed to make the JavaBase software a useful tool to members of the intelligence community.

Appendix A

Appendix A provides a breakdown of the second tier values, as well as the definition of each value.

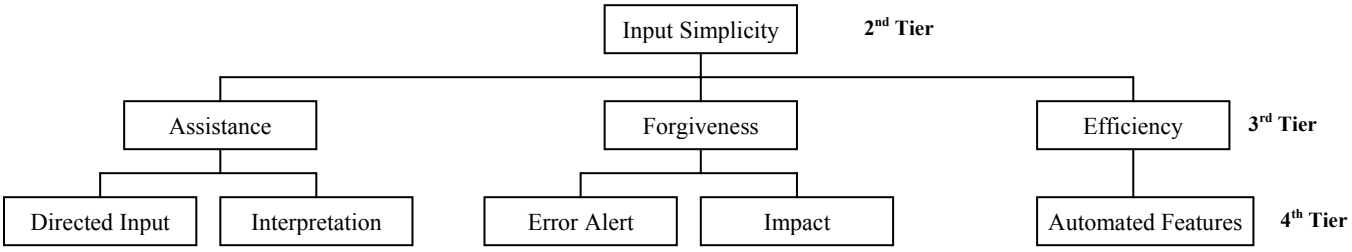

Figure 36: Input Simplicity Breakdown

Table 14: Definition for Input Simplicity Values

Input Simplicity	The ease of entering data into the software, focuses on supporting the user to ensure accurate and efficient data entry. **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**
Assistance	To aid in the data entry, providing the user helpful features to simplify the input process.
Directed Input	To inform or direct the user the correct way, or correct format to input the data, e.g. the use of drop down boxes to enter the organization.
Interpretation	The ability to interpret or recognize similar inputs that define the same thing as the same thing, e.g. MIG29 and Mig-29 would both be interpreted as MIG-29.
Forgiveness	The ability to "forgive" the user if the user makes an error and aid them in correcting the error. **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**
Error Alert	The ability to inform the user of the error that they have made in an effort to make the user aware of the error and adjust their mistakes.
Impact	The ability to minimize the effects of an error once an error has occurred, as well as the ability to enable the user to recover as much inputted data as possible.
Efficiency	The ability to recognize the user and past entries/preferences and recalling these entries/preferences in the current processing/analysis.

Automated Features	The ability able to recognize past entries as similar to current entries and offer this information to aid the user in inputting the data based on the user.

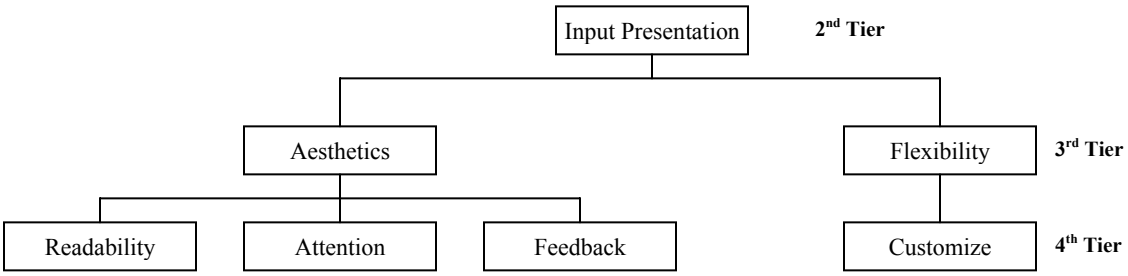

Figure 37: Input Presentation breakdown

Table 15: Definition for Input Presentation Values

Input Presentation	Defined as the way the interface displays data and feedback to the user during the input process.
Aesthetics	The ability to present the data in a way that is pleasing to the eye to increase the user's ability to input the data. **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**
Readability	The ability to present the data in a way that enables the user to read and input data with ease. This may include the colors, fonts, format and overall look of the interface.
Attention	To inform the user where they are in the document as well as to direct the user to the important/required items that are needed to be inputted, e.g. highlight required items.
Feedback	To provide information that would aid the user and give more information where needed, e.g. a help option
Flexibility	The ability to be adapted to enhance the appeal and efficiency for the user.
Customize	The ability to be modified by the user so that the look and feel of the software allows them to input data in an easier, more comfortable fashion, e.g. to be able to change the colors and fonts of the interface.

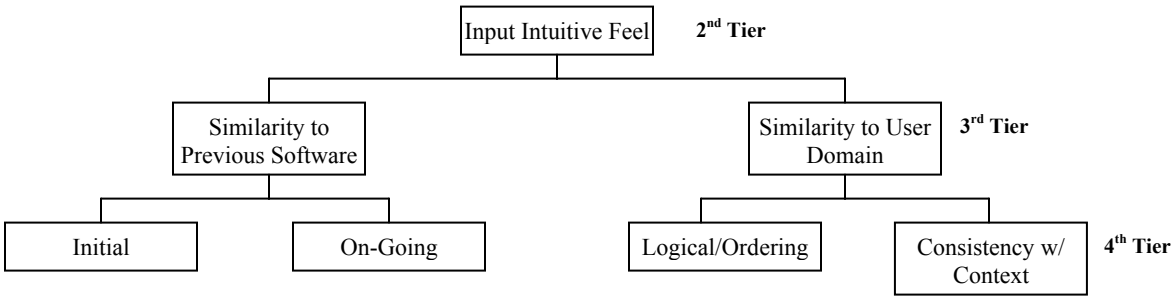

Figure 38: Input Intuitive Feel breakdown

Table 16: Definitions for Input Intuitive Feel Values

Input Intuitive Feel	Defined as the ability of the interface to permit the user to feel as if he/she understands how the interface works and how to input data.
Similarity to Previous Software	The ability to reduce the user's learning curve by mimicking interfaces with which the user is already familiar.
Initial	The ability to provide efficiency immediately, helping the user learn how to manipulate the software quickly and early on in the learning process.
On-Going	The ability to provide efficiency through out the use of the interface by creating consistency among interfaces throughout the use of the software.
Similarity to User Domain	The ability to resemble contexts with which the user has experience., e.g. the context of the intelligence community.
Logical/Ordering	To remain logical and to be in a logical order, e.g. if the user is filling out an address, the logical order would be name, street, city, state, zip.
Consistent w/ Context	To maintain consistency in terminology within the context of the outside realm and within the context of the software **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**

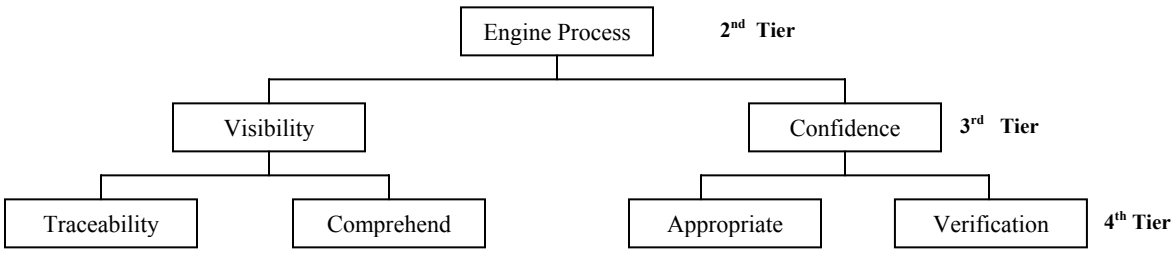

Figure 39: Engine Process breakdown

Table 17: Definitions for Engine Process Values

Engine Process	To be able to display what the engine is doing.
Visibility	The ability to show the algorithms that the engine is using, to see them, and to be able to step through each step of the algorithm.
Traceability	The ability to trace where the algorithm comes from and see where it is used in the processing.
Comprehendible	The ability to explain the algorithm and its uses in and understandable way.
Confidence	The ability to see that the algorithm is being used correctly in the software to provide confidence in the software's processing ability.
Appropriate	To show the engine is using the appropriate or the correct algorithm
Verification	The ability to show that is the working at the right level, being done correctly, and using the right data and calculating it the right way.

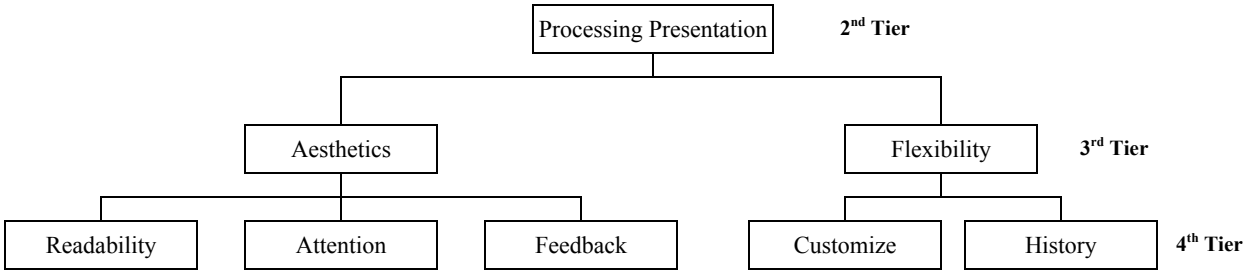

Figure 40: Processing Presentation breakdown

Table 18: Definitions for Processing Presentation Values

Processing Presentation	The ability to display the data in way that makes it easier for the user to process the data.
Aesthetics	The ability to present the data in a way that is pleasing to the eye to increase the user's ability process and analyze the data **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**
Readability	The ability to present the data in a way that enables the user to read and process data with ease. This may include the colors, fonts, format and overall look of the interface.
Attention	To inform the user of aspects of the processing and analysis that may be of importance to the user, e.g. where the output is, highlight important data
Feedback	To provide information that would aid the user and give more information where needed, e.g. a help option
Flexibility	Defined as the ability of the interface to be adapted to enhance the appeal and efficiency for the user.
Customize	The ability to be modified by the user so that the look and feel of the software allows them to process and analyze data in an easier, more comfortable fashion, e.g. to be able to change the visual options of the interface.
History	The ability to go back and see past steps that have been done

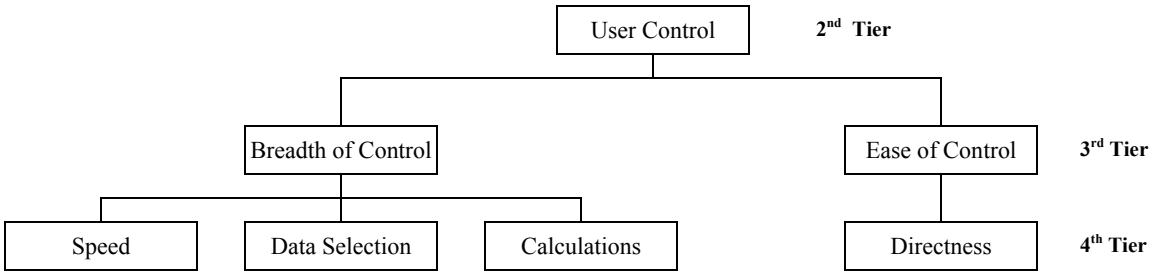

Figure 41: User Control breakdown

Table 19: Definitions for User Control values

User Control	To allow the user the ability to control how the processing is done in a way the suitable to the user. **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**
Breadth of Control	Defined as how many different aspects the user can take control of.
Speed	The ability to give the user options that that would increase the processing speed
Select Data	The ability to allow the user to decide to include/exclude data that is/is not wanted.
Calculation	The ability to give the user the option to choose the algorithm which the engine uses to process the data and perform the analysis.
Ease of Control	How easy it is for the user to implement his/her control in the processing
Directness	To be able to implement your control with as few amount of steps as possible **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**

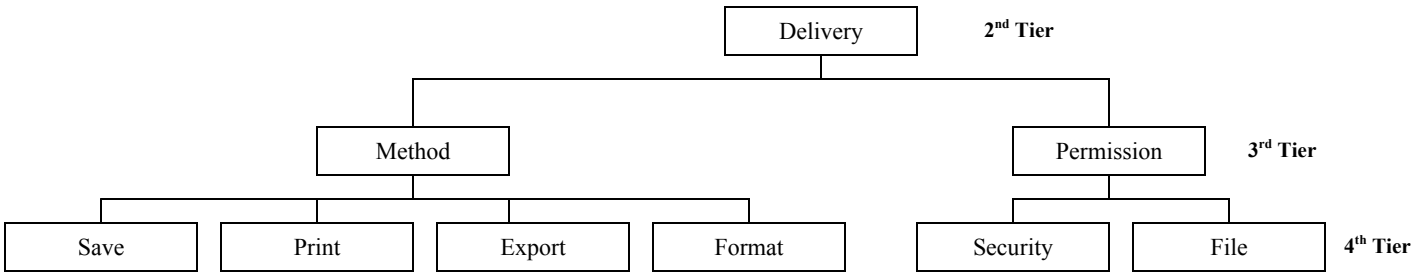

Figure 42: Delivery breakdown

Table 20: Definitions for Delivery breakdown

Delivery	Defined as the ability of the interface to allow the user to deliver the data to the customer in a particular way
Method	The method used to deliver the data to the customer
Save	The ability to save the data to a particular place, e.g. Hard drive, disk, CD
Print	The ability interface to print the data in a particular way, e.g. Black and White, Color, horizontal, vertical
Export	The ability to export the data to other programs and retain its look
Format	To be able to save the data in a particular format, e.g. text
Permission	The ability to put restrictions on the data when sent to the customer
Security	To put security restrictions on the data, e.g. Unclassified, Secret, Top Secret
File	The ability to put file restrictions on the data, e.g. Read-only, lock certain parts, no restrictions.

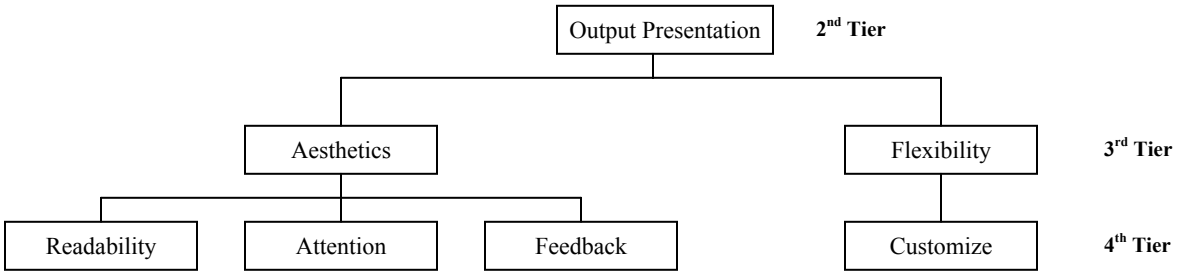

Figure 43: Output Presentation breakdown

Table 21: Definitions for Output Presentation Values

Output Presentation	Defined as the ability of the interface to display the data in way that makes it easier for the user to present the data to the customer.
Aesthetics	To present the data in a way that is pleasing to the eye to the customer or end user **PLEASE NOTE: This value corresponds with the literature found in The Windows Interface Guidelines for Software Design.**
Readability	The ability to present the data in a way that enables the customer to read and process data with ease. This may include the colors, fonts, format and overall look of the interface.
Attention	To allow the user to inform the customer of aspects of the processing and analysis that may be of importance to the customer, e.g.. where the output is, highlight important data
Feedback	The ability to provide information that would aid the user and give more information where needed, e.g. a help option
Flexibility	Defined as the ability of the interface to be adapted by the user to enhance the appeal and efficiency for the customer.
Customize	To be modified by the user so that the look and feel of the software allows them to output data in an easier, more comfortable fashion for the customer, e.g.. to be able to change the visual options of the interface.

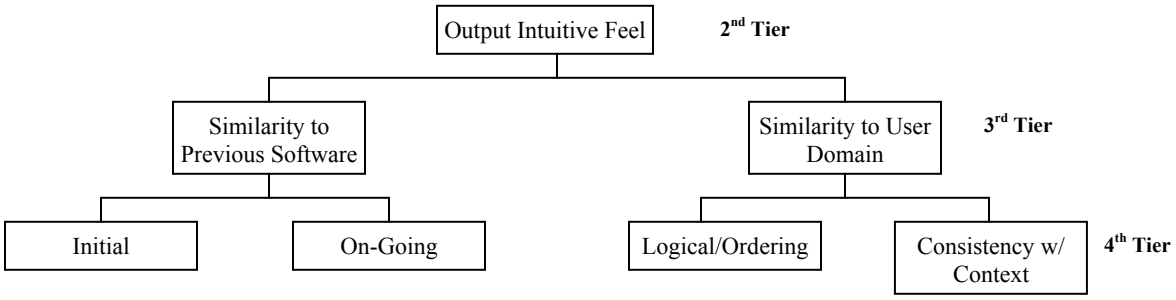

Figure 44: Output Intuitive Feel breakdown

Table 22: Definitions for Output Intuitive Feel Values

Output Intuitive Feel	To permit the user to feel as if he/she understands how the interface works and how to output data.
Similarity to Previous Software	The ability to reduce the user's learning curve by mimicking interfaces with which the user is already familiar.
Initial	The ability to provide efficiency immediately, helping the user learn how to manipulate the software quickly and early on in the learning process.
On-Going	Defined as the ability of the interface to provide efficiency through out the use of the interface by creating consistency among interfaces throughout the use of the software.
Similarity to Customer Domain	To allow the user to output the data in a way that is comfortable to the customer
Logical/Ordering	To allow the user to display the output in a logical order for the customer, e.g. to be able to arrange the order
Consistent w/ Context	To maintain consistency in terminology within the context of the outside realm and within the context of the software for the customer

Appendix B

Appendix B gives the definitions for the measures of each component.

Table 23: Definitions for Input measures

Measure	Definition
Extent of fields that have directed input	The extent that the interface has directed input in the fields that are used in the input component
Does the interface have the ability to interpret?	Whether or not the interface has the capability to interpret.
Extent interface inform the user of errors?	How often does the interface alert the user of an error when an error has occurred in the input process.
Amount of work lost.	How much of the inputted work is lost if an error occurs or the system crashes.
Can user retrieve the work?	Assuming the work can be retrieved, can the user retrieve it themselves or must they get help from a trained computer person.
Backfill	Does the interface immediately fill in the data as that previous inputted item, e.g. the user previously entered Kroger as a place the user shops at, so when the user types Kr- the remaining –oger is given to him/her with the option to automatically fill in it in is given.
One-Time	Can the interface connect input fields so that the entry of identical data must only be made once, and be able to keep the preferences of the user every time the software is used.
Ease of reading colors.	How easy are the colors to read when inputting the data for the user.
Ease of reading fonts.	How easy are the fonts to read when inputting the data for the user.
Can the interface emphasize?	Does the interface draw attention to the appropriate fields that need to be in the input process, e.g. can you highlight important items
Quality of Feedback	How helpful is the feedback to the user, e.g. does the user get “ERROR 56” or a good definition of what needs to be done
Frequency of Feedback	How often is feedback given to the user.
Ease of color change.	Of the items that allow the colors to be changed, how easy it to apply this change.
Ease of font change.	Of the items that allow the fonts to be changed, how easy it to apply this change.
How initially similar?	When the interface is looked at initially, how does it compares to the similarity of other software programs, e.g. how similar does it look to Microsoft products
How similar in long run?	Once the user has become familiar with the program, does the interface have a similar feel to other software programs, e.g. does it have same hot keys as Microsoft products
Is it logical?	Is the interface provided in a way that is logical to its intended user? e.g. is the file menu on the top-left
How consistent?	Is the interface provided in a way that is consistent with the business rules of its intended user?

Table 24: Definition for Processing measures

Measure	Definition
Ease of tracing algorithms	The ease for the user to trace the algorithm that is being used in the software.
Ease of comprehending algorithm	How good is the explanation that is given of the algorithms that are used it the software.
Are algorithms appropriate	Can the user look to see if the appropriate algorithms are being used.
Can data be verified?	Does the user have the ability to see if the correct data is being inputted into the algorithm.
Can calculations be verified	Does the user have the ability to see if the correct are being done by the algorithm.
Ease of reading colors.	How easy are the colors to read when processing the data for the user.
Ease of reading fonts.	How easy are the fonts to read when processing the data for the user.
Can the interface emphasize?	Does the interface draw attention to the appropriate fields that need to be in the processing, e.g. can you highlight important items
Quality of Feedback	When the interface provides feedback to the user, how helpful is the feedback to the user, e.g. does the user get "ERROR 56" or a good definition of what needs to be done
Frequency of Feedback	When feedback can be useful, how often is feedback given to the user.
Number of views interface gives.	Defined as the number of additional views for the, e.g. Web like/Normal/Print view
Can choose type?	Defined as the ability of the interface to be able to control the type of data that is shown, e.g. show a graph or bar chart
Can choose quantity?	Defined as the ability of the interface to be able to control how much data is shown at a time, e.g. The top 5 results displayed
Can choose depth?	Defined as the ability of the interface to be able to control how much is displayed, e.g. display the name, or the name address and phone #
Go back one step	Defined as the ability of the interface to go back one step at a time
Go back many step	Defined as the ability of the interface to be able to take the user back to a given point that the user has previously been
Ease of manner control	Defined as the ability of the interface to give the user control over the manner of which the processing is done, e.g. with pictures or without
Ease of precision control	Defined as the ability of the interface to give the user control over the amount of precision the user wants in the algorithm, e.g. 5% precision, 1% precision, .1% precision.

Degree data can be selected	The ability to select which data is used when processing the data
Can algorithm be selected	Defined as the ability of the interface to check to see the calculations and if they are done correctly.
Ease of Changes/Control	Of the items the user can control, how easy is it to implement the control over those items.

Table 25: Definitions for Output measures

Measure	Definition
Save options	Do the save options in the software meet the standards that are typical in most software packages, e.g. are the save options the same as those in Microsoft packages.
Print options	Do the print options in the software meet the standards that are typical in most software packages, e.g. are the print options the same as those in Microsoft packages.
Export options	Do the export options in the software meet the standards that are typical in most software packages, e.g. are the export options the same as those in Microsoft packages.
Format options	Do the format options in the software meet the standards that are typical in most software packages, e.g. are the format options the same as those in Microsoft packages.
Security Capable	Does the interface allow for the user to put any security clearance options on the program when saving it.
Ability to lock info?	Does the interface allow for the user to put any lock information options on the program when saving it so the user can prevent others who look at the program change the work that the user has done.
Ease of reading colors.	How easy are the colors to read when outputting the data for the user.
Ease of reading fonts.	How easy are the fonts to read when outputting the data for the user.
Can the interface emphasize?	Does the interface draw attention to the appropriate fields that need to be in the output process, e.g. can you highlight important items
Quality of Feedback	How helpful is the feedback to the user, e.g. does the user get “ERROR 56” or a good definition of what needs to be done
Frequency of Feedback	How often is feedback given to the user.
Number of views interface gives.	Defined as the number of additional views the interface is able allow the user to view the way the output is viewed, e.g. Web like/Normal/Print view
Can choose type?	Defined as the ability of the interface to be able to control the type of data that is shown, e.g. show a graph or bar chart
Can choose quantity?	Defined as the ability of the interface to be able to control how much data is shown at a time, e.g. The top 5 results displayed
Can choose depth?	Defined as the ability of the interface to be able to control how much is displayed, e.g. display the name, or the name address and phone #

How initially similar?	When the output interface is looked at initially, how does it compares to the similarity of other software programs, e.g. how similar does it look to Microsoft products
How similar in long run?	Once the user has become familiar with the output of the program, does the interface have a similar feel to other software programs, e.g. does it have same hot keys as Microsoft products
Is it logical?	Is the output interface provided in a way that is logical to its intended user?, e.g. is the file menu on the top-left
How consistent?	Is the interface provided in a way that is consistent with the business rules of its intended customer?

Appendix C

The following are the SDVFs for the measures of the interface. Along with the SDVF is given the x-axis for the SDVF, the categories each is broken into, as well as the definition for each category.

The Input SDVFs

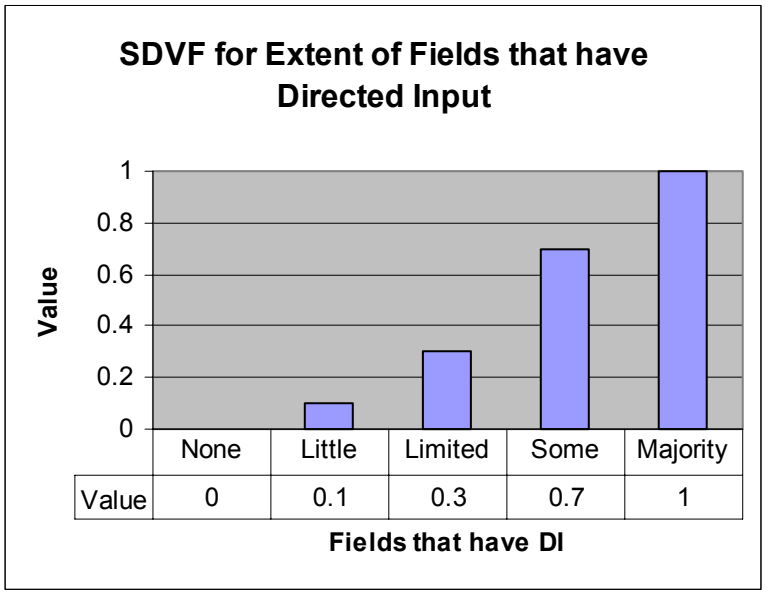

Figure 45: SDVF for Extent of Fields that have Directed Input

Table 26: Definitions for categories of Extent of Fields that have Directed Input

Category	Definition
None	Having no directed input.
Little	Having some directed input, but less than or equal to 10% of the fields having it.
Limited	Having greater the 10% directed input, but less than or equal to 50% of the fields having it.
Some	Having greater the 50% directed input, but less than or equal to 80% of the fields having it.
Majority	Having greater the 80% directed input.

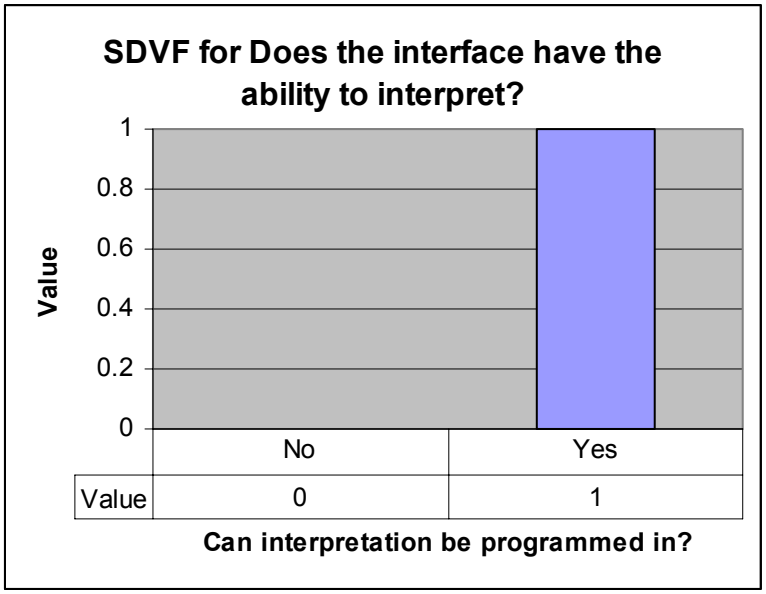

Figure 46: SDVF for Does the interface have the ability to interpret?

Table 27: Definitions for categories of Does the interface have the ability to interpret?

Category	Definition
No	Having no ability to interpret
Yes	Having ability to interpret.

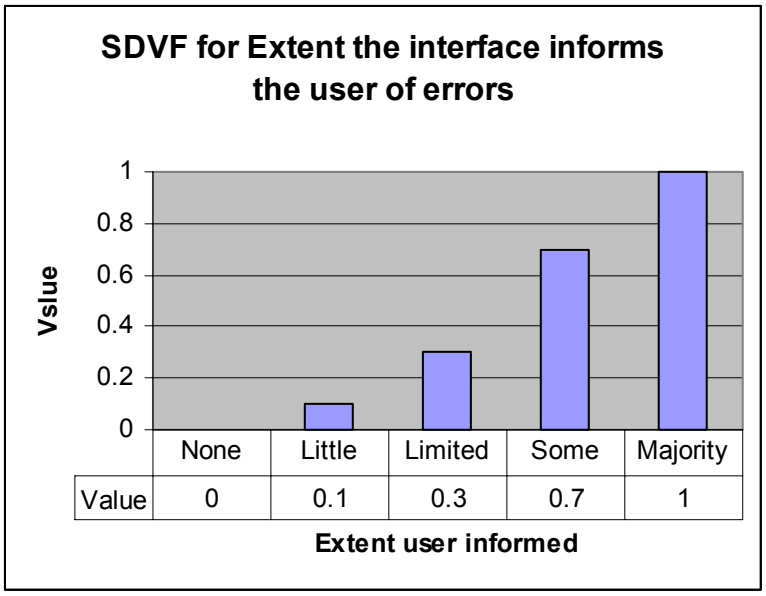

Figure 47: SDVF for Extent the interface informs the user of errors

Table 28: Definitions for categories of Extent the interface informs the user of errors

Category	Definition
None	Defined as not informing the user of errors.
Little	Having some error informing, but less than or equal to 10% of the time having it.
Limited	Having greater the 10% error informing, but less than or equal to 50% of the time having it.
Some	Having greater the 50% error informing, but less than or equal to 80% of the time having it.
Majority	Having greater the 80% error informing.

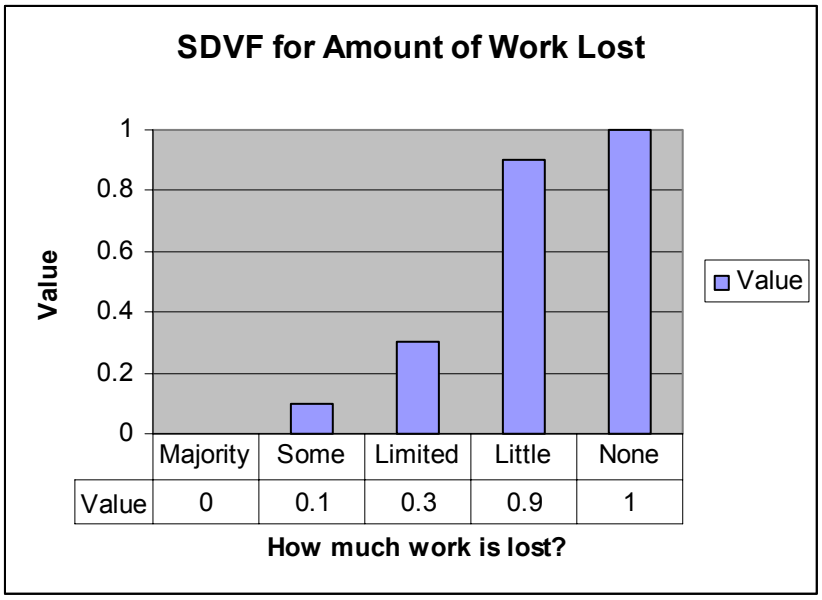

Figure 48: SDVF for Amount of Work Lost

Table 29: Definitions for categories of Amount of Work Lost

Category	Definition
Majority	Having greater the 80% work lost.
Some	Having greater the 50% work lost, but less than or equal to 80% of it lost.
Limited	Having greater the 10% work lost, but less than or equal to 50% of it lost.
Little	Having some work lost, but less than or equal to 10% of it lost.
None	Defined as not having work lost.

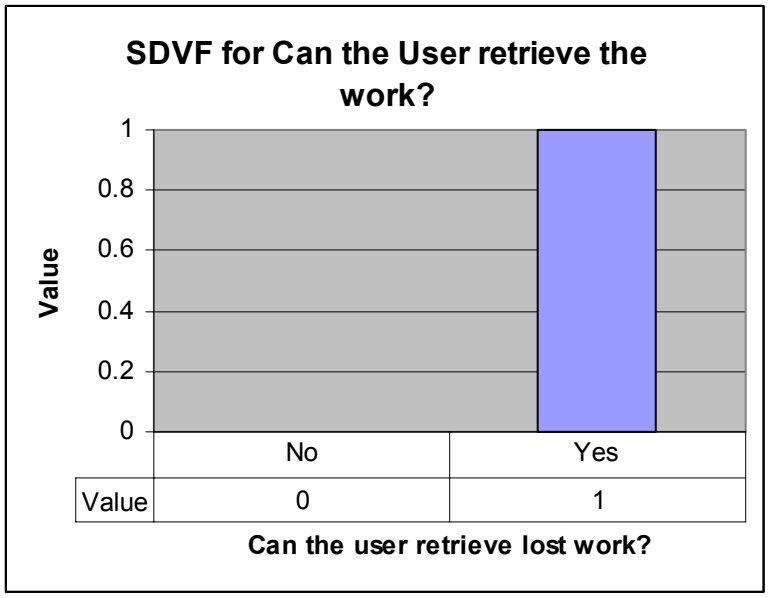

Figure 49: SDVF for Can the User retrieve the work?

Table 30: Definitions for categories of Can the User retrieve the work?

Category	Definition
No	Having no ability to retrieve work.
Yes	Having ability to retrieve work.

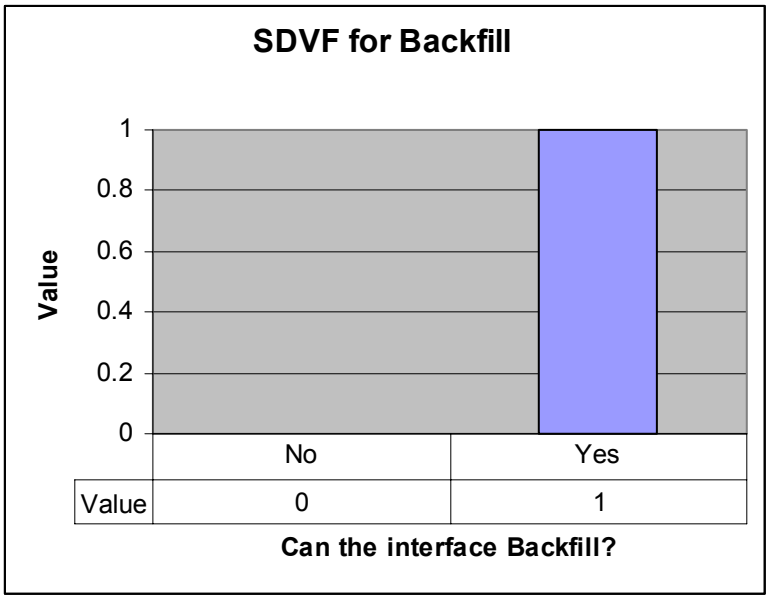

Figure 50: SDVF for Backfill

Table 31: Definitions for categories of Backfill

Category	Definition
No	Having no ability to backfill.
Yes	Having ability to backfill.

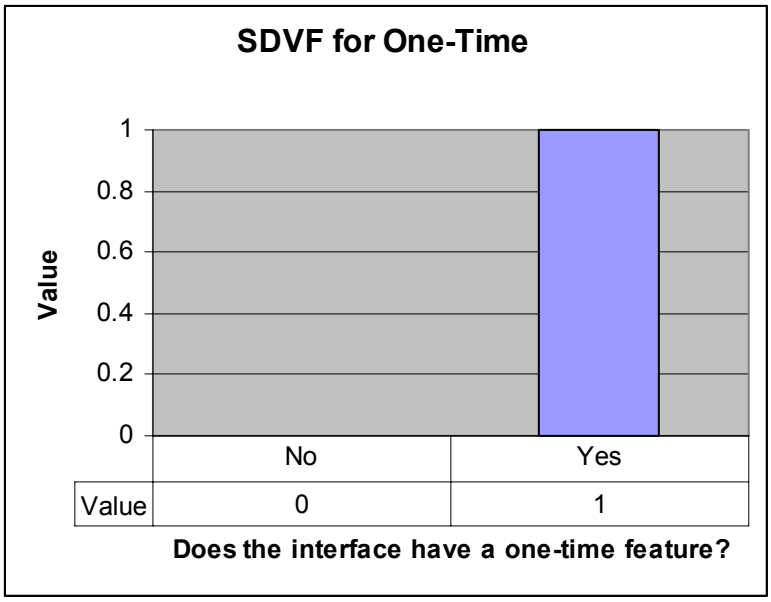

Figure 51: SDVF for One-Time

Table 32: Definitions for categories of One-Time

Category	Definition
No	Having no one-time feature.
Yes	Having a one-time feature.

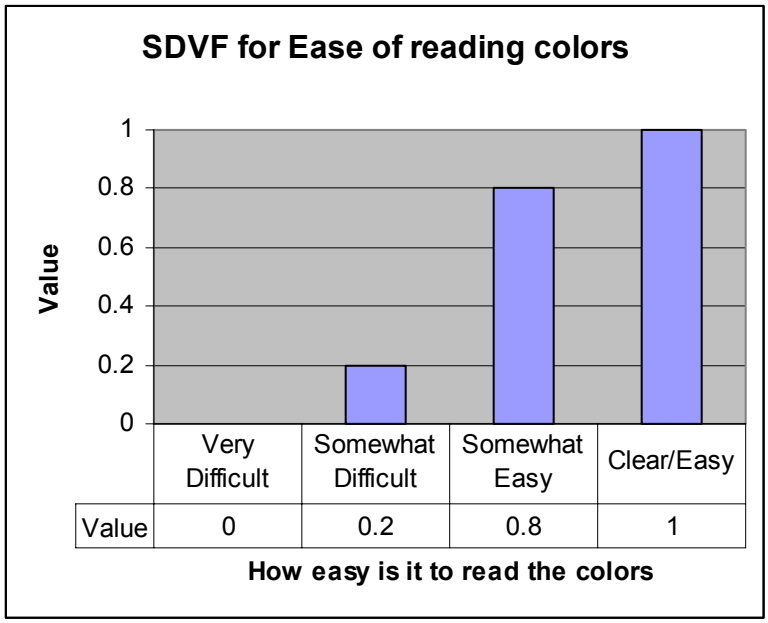

Figure 52: SDVF for Ease of Reading Colors

Table 33: Definitions for categories of Ease of Reading Colors

Category	Definition
Very Difficult	Very difficult to read
Somewhat Difficult	Somewhat difficult to read.
Somewhat Easy	Somewhat easy to read.
Clear/Easy	Easy to read.

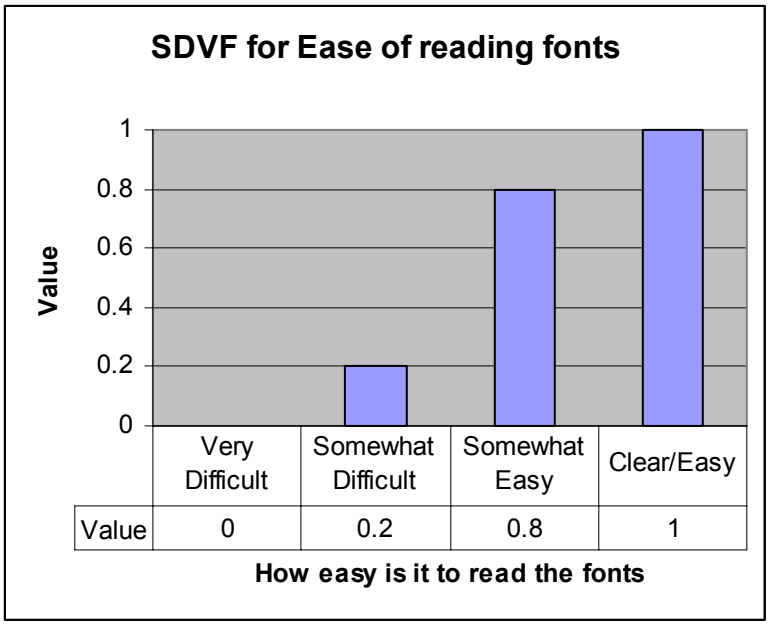

Figure 53: SDVF for Ease of reading fonts

Table 34: Definitions for categories of Ease of reading fonts

Category	Definition
Very Difficult	Very difficult to read
Somewhat Difficult	Somewhat difficult to read.
Somewhat Easy	Somewhat easy to read.
Clear/Easy	Easy to read.

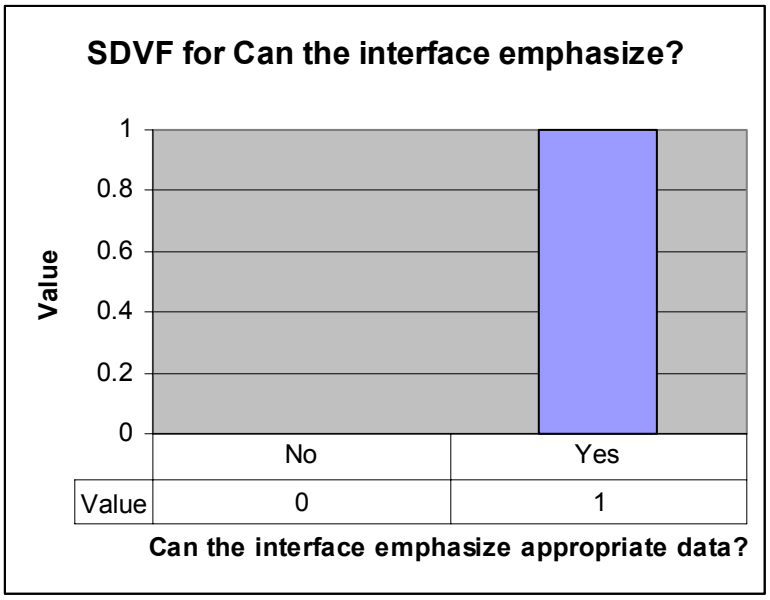

Figure 54: SDVF for Can the interface emphasize?

Table 35: Definitions for categories of Can the interface emphasize?

Category	Definition
No	Having no ability to emphasize data.
Yes	Having ability to emphasize data..

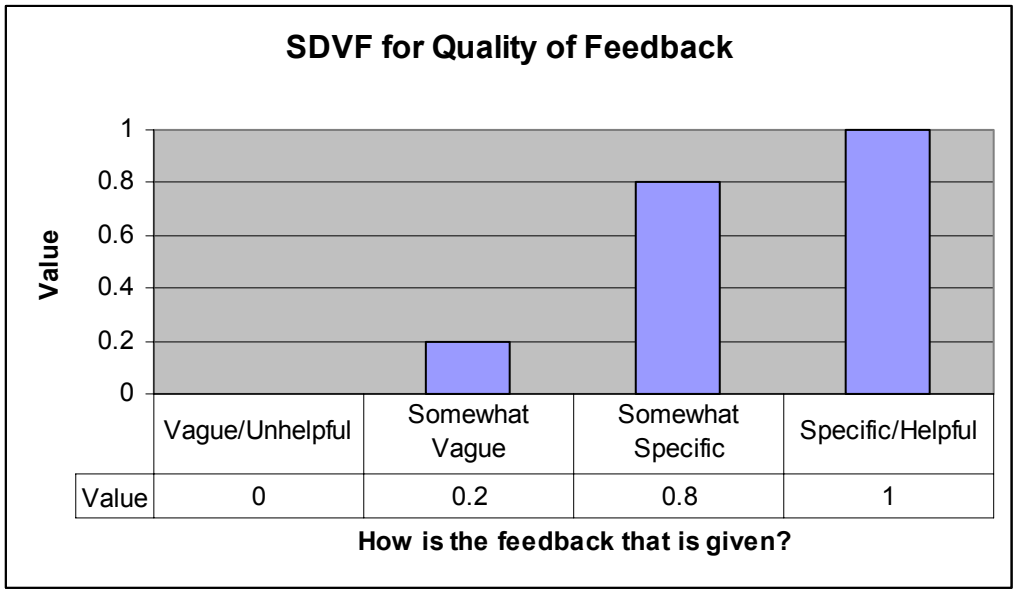

Figure 55: SDVF for Quality of Feedback

Table 36: Definitions for categories of Quality of Feedback

Category	Definition
Vague/Unhelpful	Feedback that is vague and unhelpful
Somewhat Vague	Feedback that can be understood somewhat, but is somewhat vague and is difficult to interpret.
Somewhat Specific	Feedback that can be understood and is somewhat specific, but can still be up to interpretation.
Specific/Helpful	Feedback that can be understood and is specific in its details.

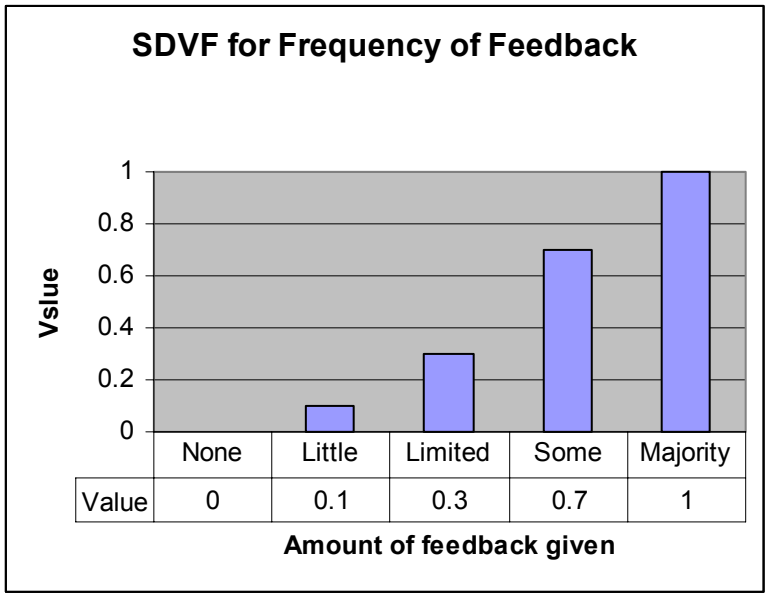

Figure 56: SDVF for Frequency of Feedback

Table 37: Definitions for categories of Frequency of Feedback

Category	Definition
None	Having no feedback.
Little	Having some feedback given, but less than or equal to 10% of the time having it.
Limited	Having greater the 10% feedback given, but less than or equal to 50% of the time having it.
Some	Having greater the 50% feedback given, but less than or equal to 80% of the time having it.
Majority	Having greater the 80% feedback given.

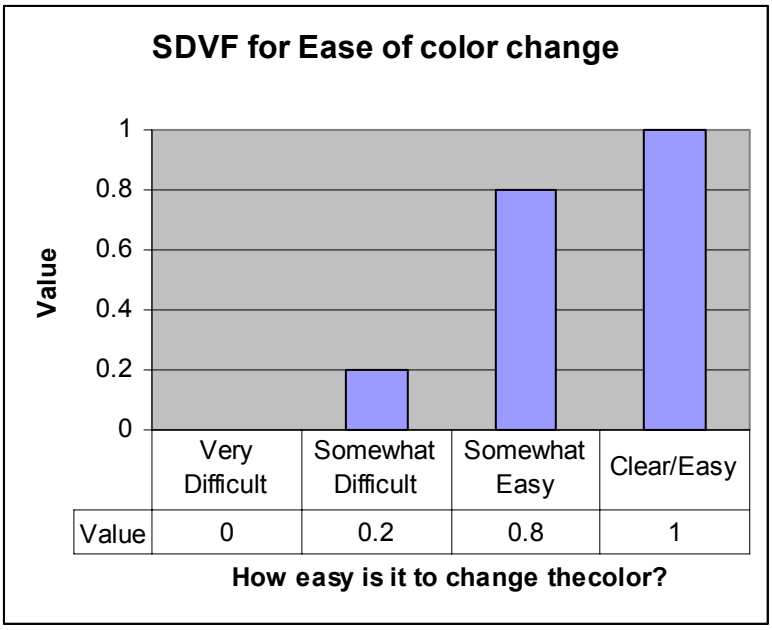

Figure 57: SDVF for Ease of color change

Table 38: Definitions for categories of Ease of color change

Category	Definition
Very Difficult	Very difficult to change
Somewhat Difficult	Somewhat difficult to change.
Somewhat Easy	Somewhat easy to change.
Clear/Easy	Easy to change.

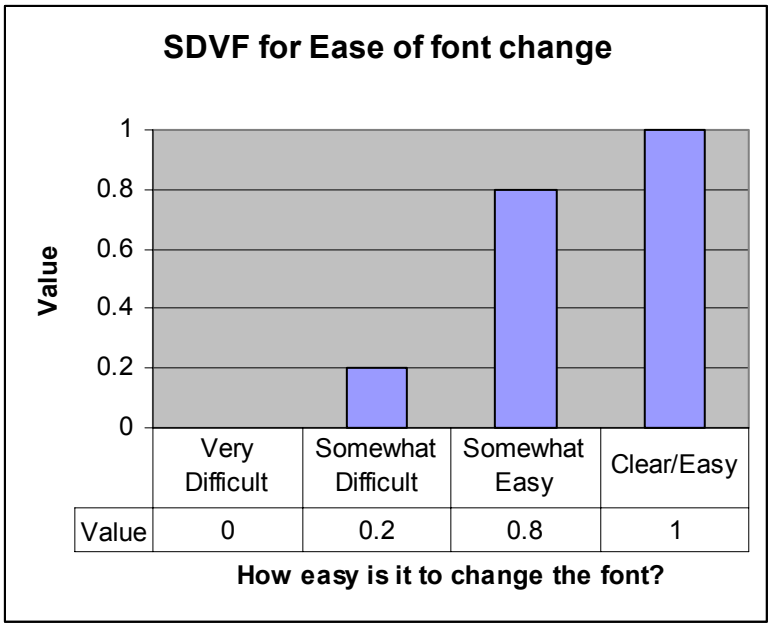

Figure 58: SDVF for Ease of font change

Table 39: Definitions for categories of Ease of font change

Category	Definition
Very Difficult	Very difficult to change
Somewhat Difficult	Somewhat difficult to change.
Somewhat Easy	Somewhat easy to change.
Clear/Easy	Easy to change.

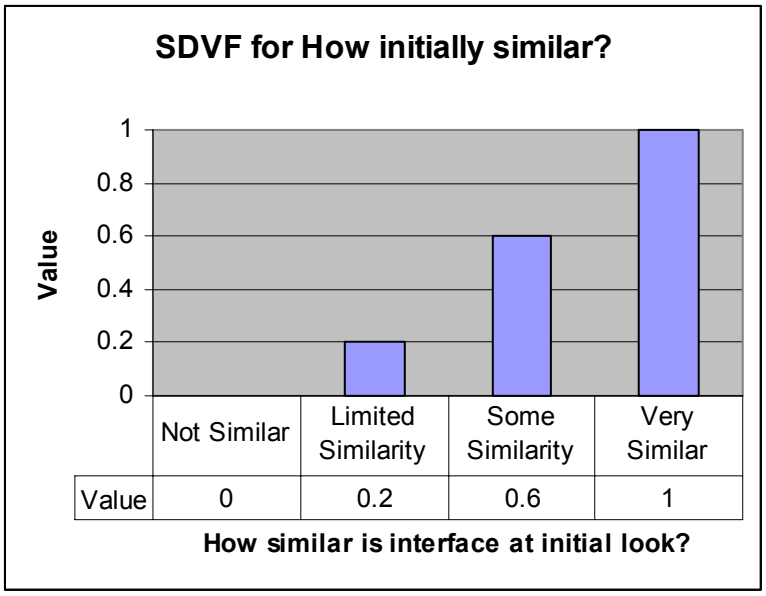

Figure 59: SDVF for How initially similar?

Table 40: Definitions for categories of How initially similar?

Category	Definition
Not Similar	Defined as the interface not being similar to other software when initially looked at.
Limited Similarity	Defined as the interface having few similarities to other software when initially looked at.
Some Similarity	Defined as the interface having more than a few similarities to other software when initially looked at.
Very Similar	Defined as the interface being very similar to other software when initially looked at.

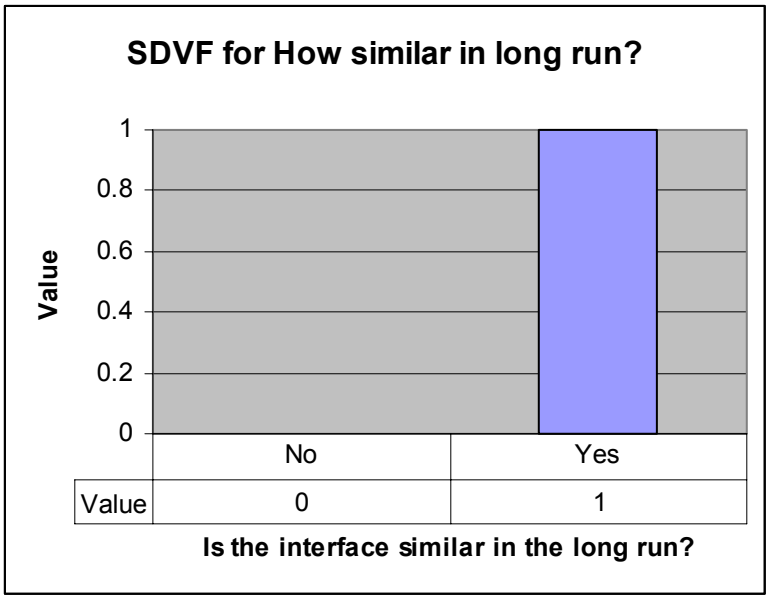

Figure 60: SDVF for How similar in long run?

Table 41: Definitions for categories of How similar in long run?

Category	Definition
No	Having no similarity to other software in the long run.
Yes	Having similarity to other software in the long run.

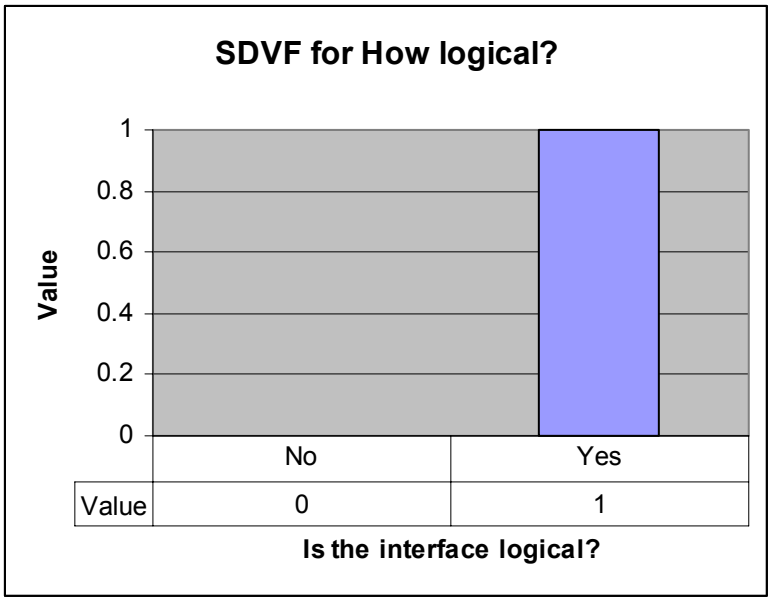

Figure 61: SDVF for How logical?

Table 42: Definitions for categories of How logical?

Category	Definition
No	Not being logical
Yes	Being logical.

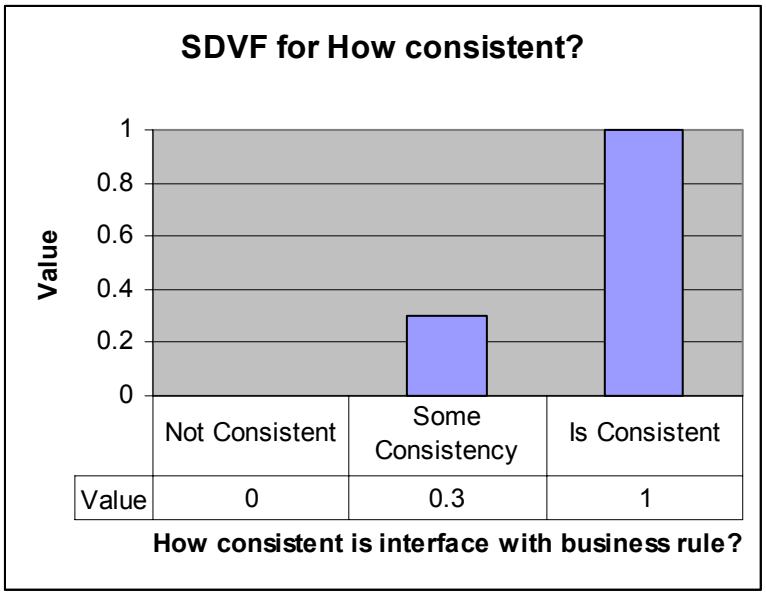

Figure 62: SDVF for How consistent?

Table 43: Definitions for categories of How consistent?

Category	Definition
Not Consistent	Defined as not having consistency with the context of the user.
Some Consistency	Having some consistency with the context of the user, but not total.
Is consistent	Having consistency with the context of the user.

The Processing SDVFs

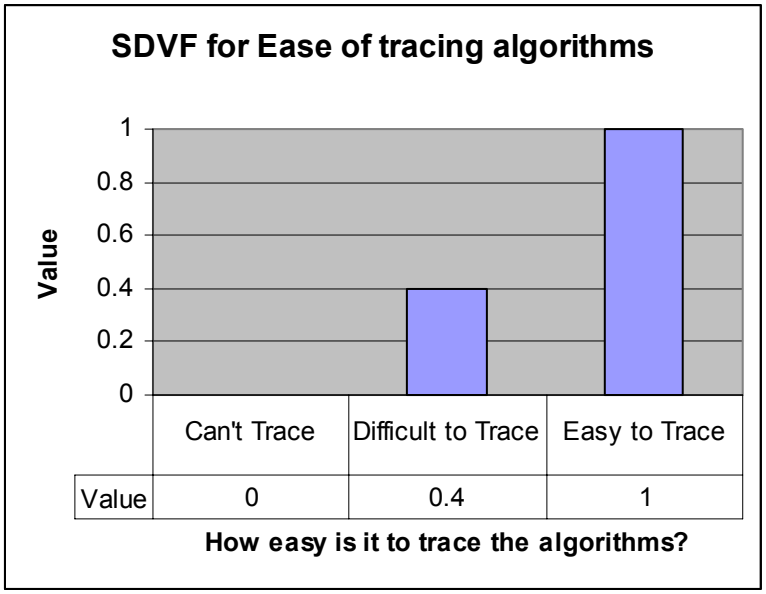

Figure 63: SDVF for Ease of tracing algorithms

Table 44: Definitions for categories of Ease of tracing algorithms

Category	Definition
Can't Trace	Defined as not having the ability to trace the algorithms.
Difficult to Trace	Defined as being able to trace the algorithms, but having difficulty in doing so.
Easy to Trace	Defined as being able to easily trace the algorithms.

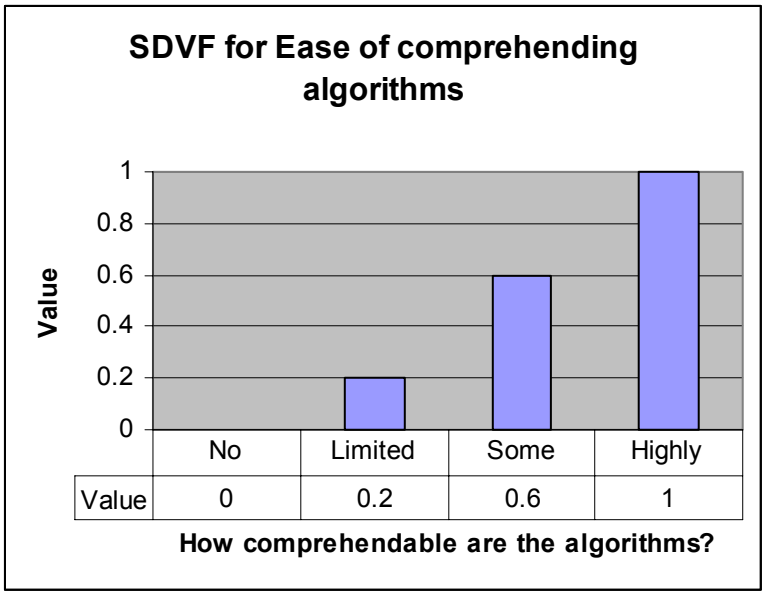

Figure 64: SDVF for Ease of comprehending algorithms

Table 45: Definitions for categories of Ease of comprehending algorithms

Category	Definition
No	Having no ability to comprehend the algorithms.
Limited	Defined as the interface defining the algorithm but with limited info
Some	Defined as the interface defining the algorithm with some information about the algorithm provided
Highly	Defined as the interface defining the algorithm with a very good explanation of it.

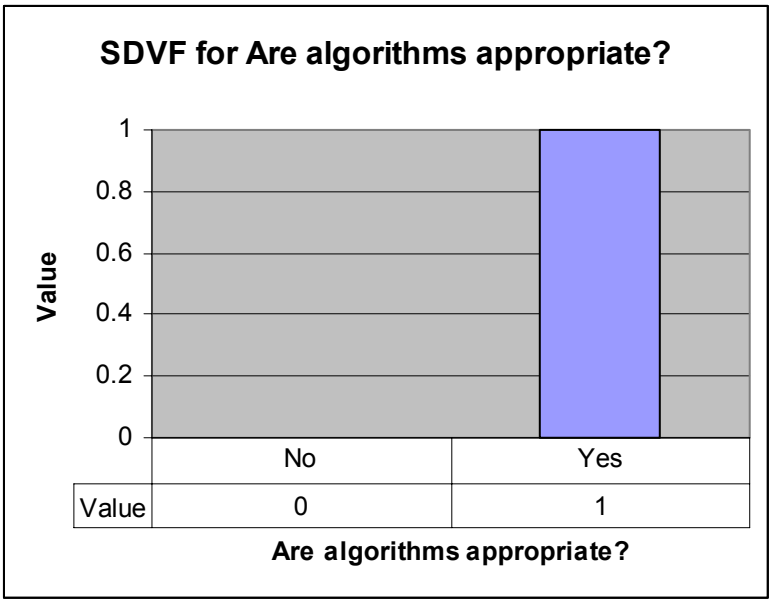

Figure 65: SDVF for Are algorithms appropriate?

Table 46: Definitions for categories of Are algorithms appropriate?

Category	Definition
No	Having no ability to see if the algorithms are appropriate.
Yes	Having ability to see if the algorithms are appropriate.

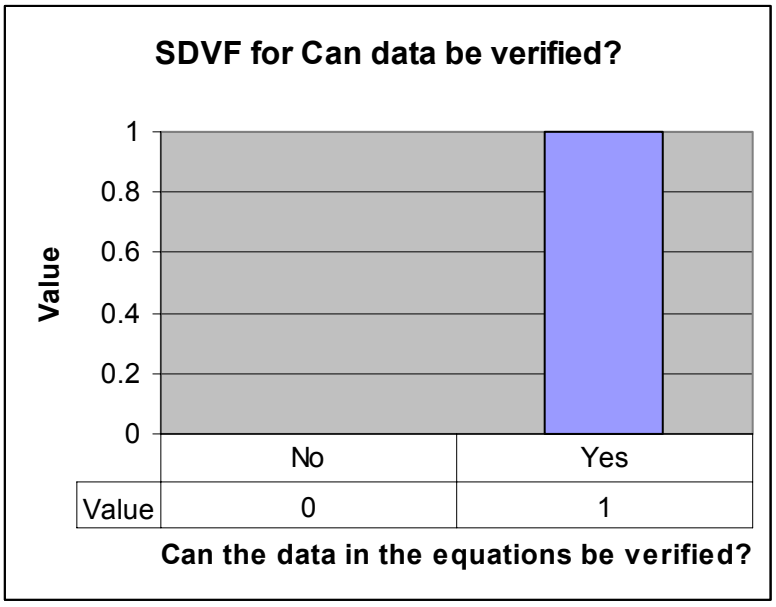

Figure 66: SDVF for Can data be verified?

Table 47: Definitions for categories of Can data be verified?

Category	Definition
No	Having no ability to verify the data in the equation.
Yes	Having ability to verify the data in the equation.

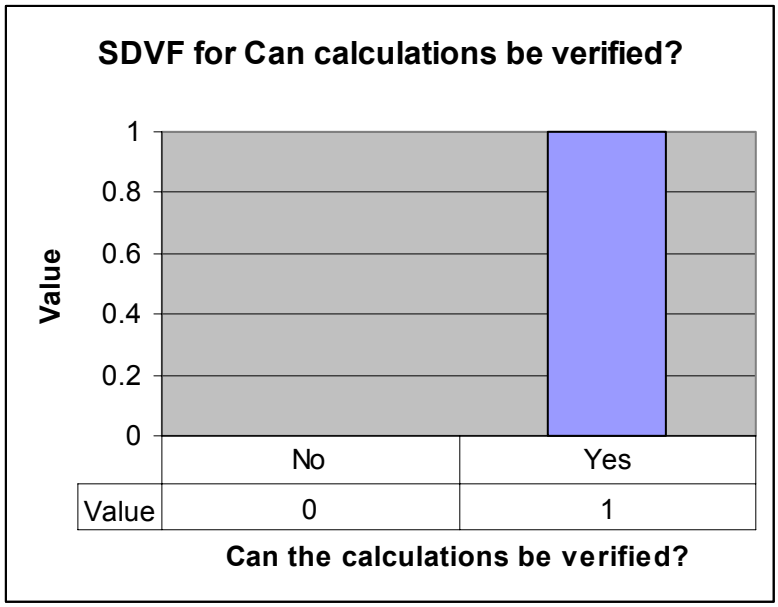

Figure 67: SDVF for Can calculations be verified?

Table 48: Definitions for categories of Can calculations be verified?

Category	Definition
No	Having no ability to verify the calculations.
Yes	Having ability to verify the calculations.

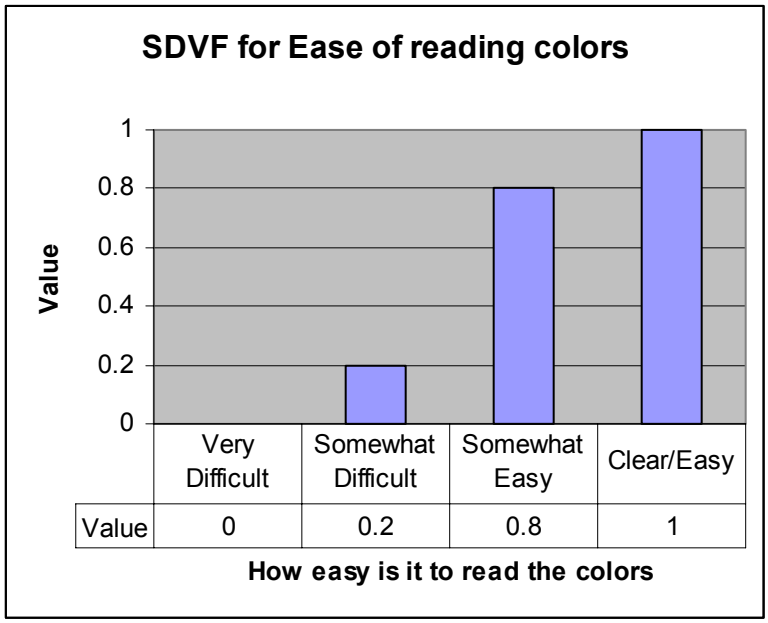

Figure 68: SDVF for Ease of reading colors

Table 49: Definitions for categories of Ease of reading colors

Category	Definition
Very Difficult	Defined as being very difficult to read
Somewhat Difficult	Defined as being somewhat difficult to read.
Somewhat Easy	Defined as somewhat easy to read.
Clear/Easy	Defined as being easy to read.

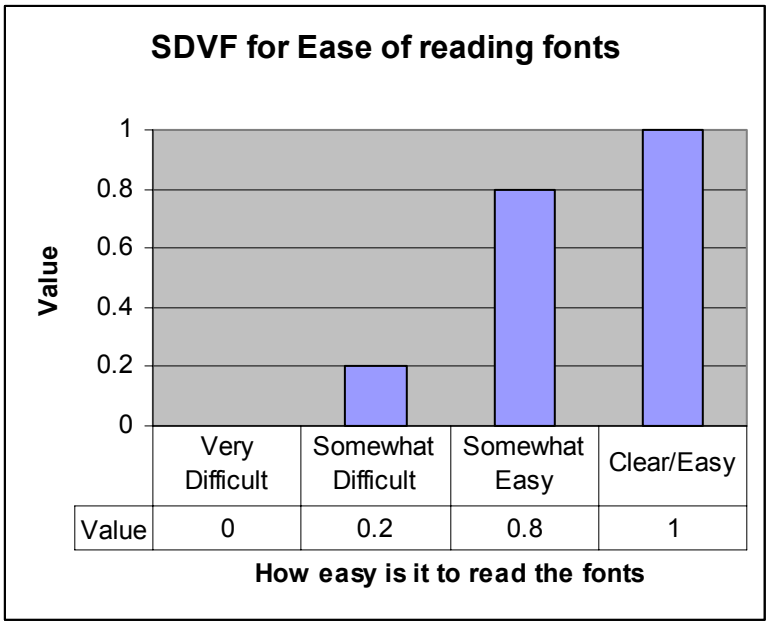

Figure 69: SDVF for Ease of reading fonts

Table 50: Definitions for categories of Ease of reading fonts

Category	Definition
Very Difficult	Defined as being very difficult to read
Somewhat Difficult	Defined as being somewhat difficult to read.
Somewhat Easy	Defined as somewhat easy to read.
Clear/Easy	Defined as being easy to read.

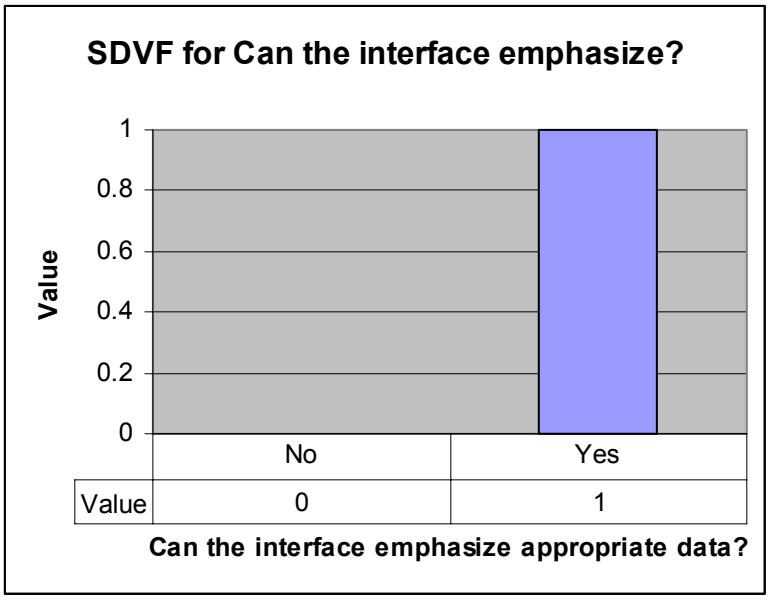

Figure 70: SDVF for Can the interface emphasize?

Table 51: Definitions for categories of Can the interface emphasize?

Category	Definition
No	Having no ability to emphasize data.
Yes	Having ability to emphasize data.

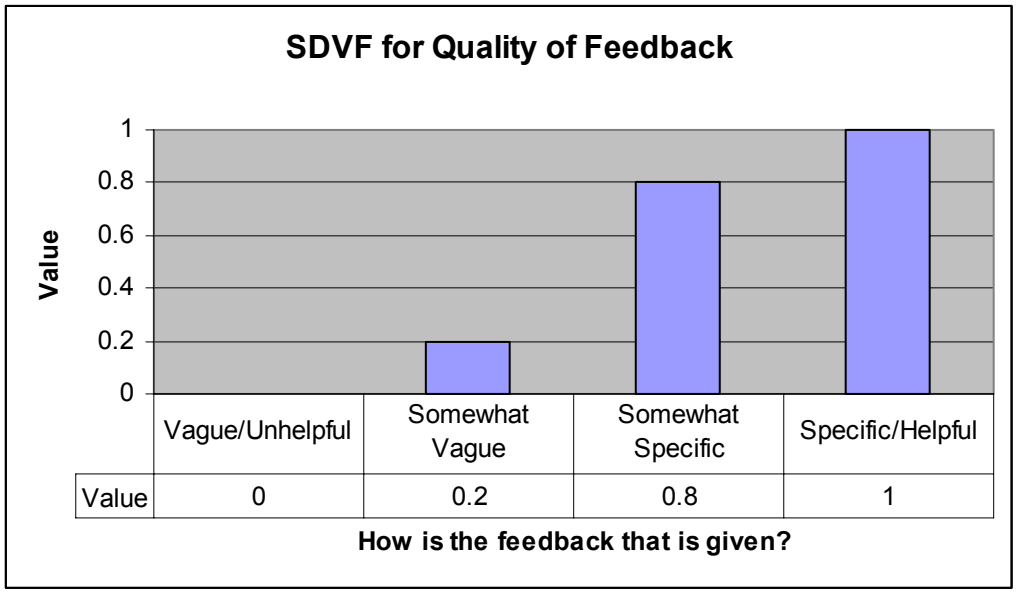

Figure 71: SDVF for Quality of Feedback

Table 52: Definitions for categories of Quality of Feedback

Category	Definition
Vague/Unhelpful	Defined as being feedback that is vague and unhelpful
Somewhat Vague	Defined as feedback that can be understood somewhat, but is somewhat vague and is difficult to interpret.
Somewhat Specific	Defined as feedback that can be understood and is somewhat specific, but can still be up to interpretation.
Specific/Helpful	Defined as feedback that can be understood and is specific in its details.

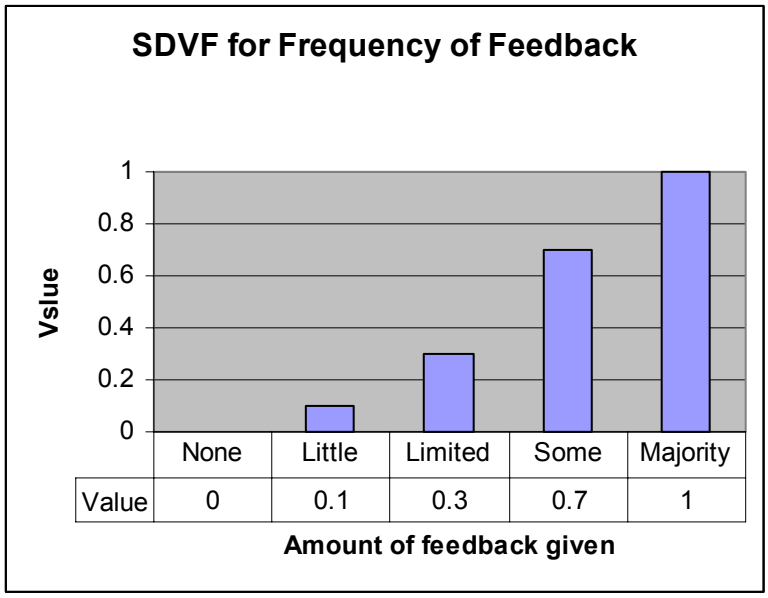

Figure 72: SDVF for Frequency of Feedback

Table 53: Definitions for categories of Quality of Feedback

Category	Definition
None	Having no feedback.
Little	Having some feedback given, but less than or equal to 10% of the time having it.
Limited	Having greater the 10% feedback given, but less than or equal to 50% of the time having it.
Some	Having greater the 50% feedback given, but less than or equal to 80% of the time having it.
Majority	Having greater the 80% feedback given.

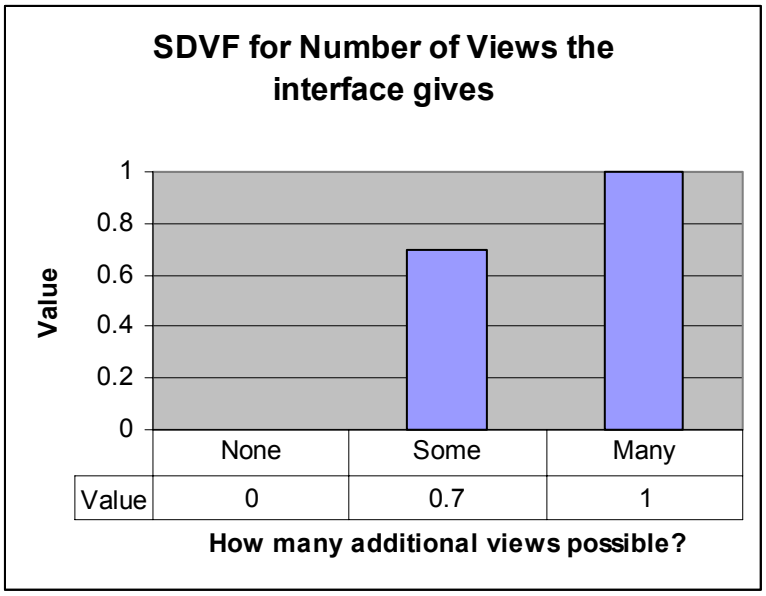

Figure 73: SDVF for Number of Views the interface gives

Table 54: Definitions for categories of Number of Views the interface gives

Category	Definition
None	Defined as not having no more options available
Some	Having some options available
Many	Having many options available

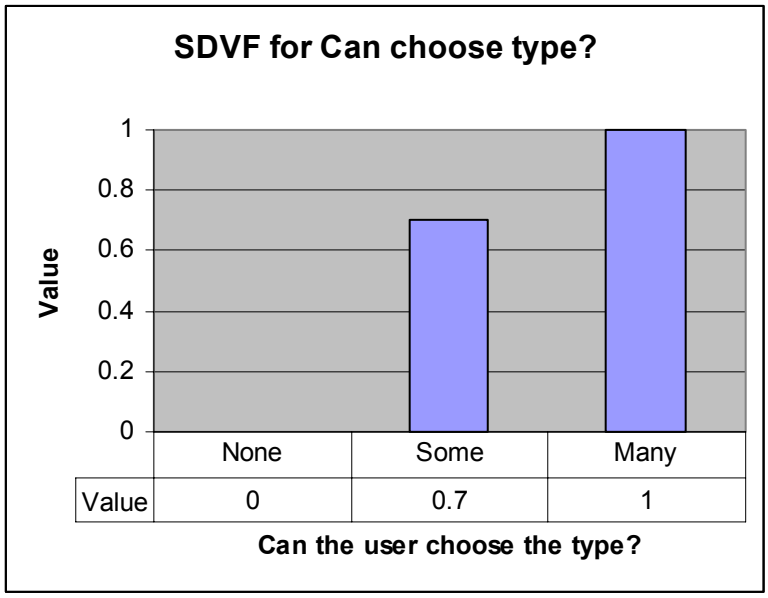

Figure 74: SDVF for Can choose type?

Table 55: Definitions for categories of Can choose type?

Category	Definition
None	Defined as not having no more options available
Some	Having some options available
Many	Having many options available

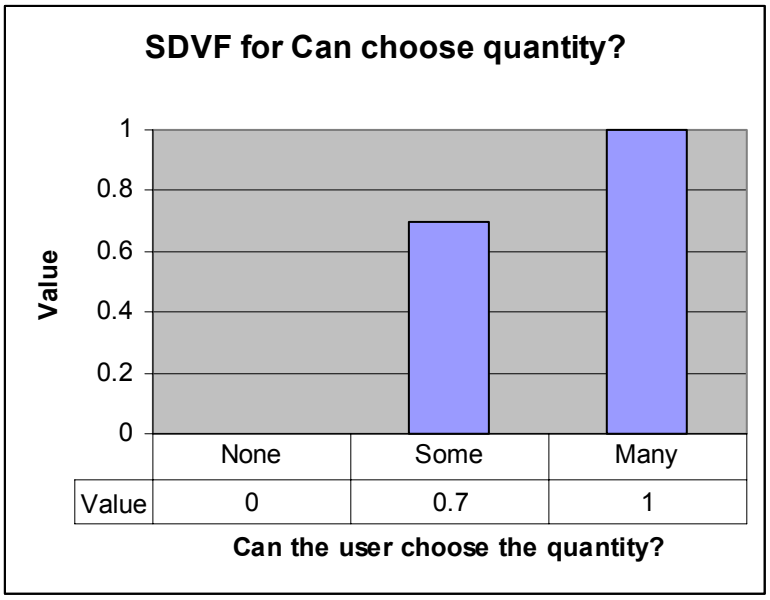

Figure 75: SDVF for Can choose quantity?

Table 56: Definitions for categories of Can choose quantity?

Category	Definition
None	Defined as not having no more options available
Some	Having some options available
Many	Having many options available

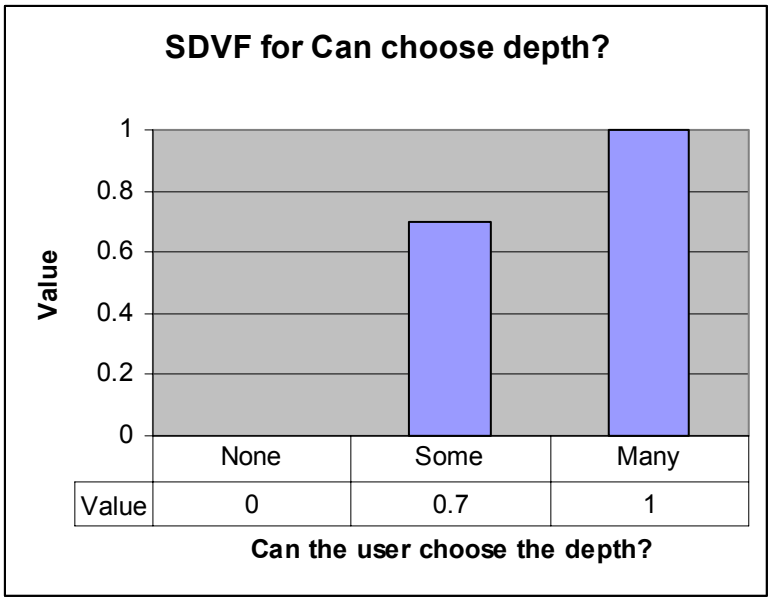

Figure 76: SDVF for Can choose depth?

Table 57: Definitions for categories of Can choose depth?

Category	Definition
None	Defined as not having no more options available
Some	Having some options available
Many	Having many options available

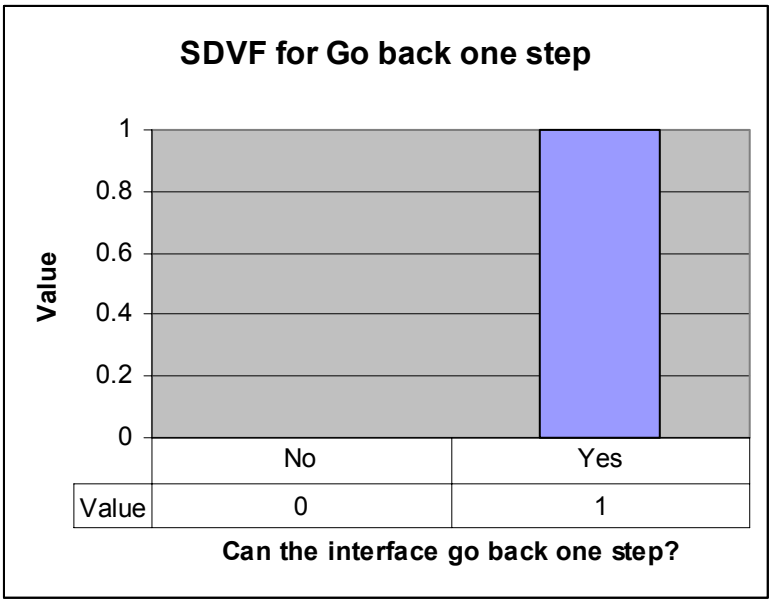

Figure 77: SDVF for Go back one step

Table 58: Definitions for categories of Go back one step

Category	Definition
No	Having no ability to go back one step at a time.
Yes	Having ability to go back one step at a time.

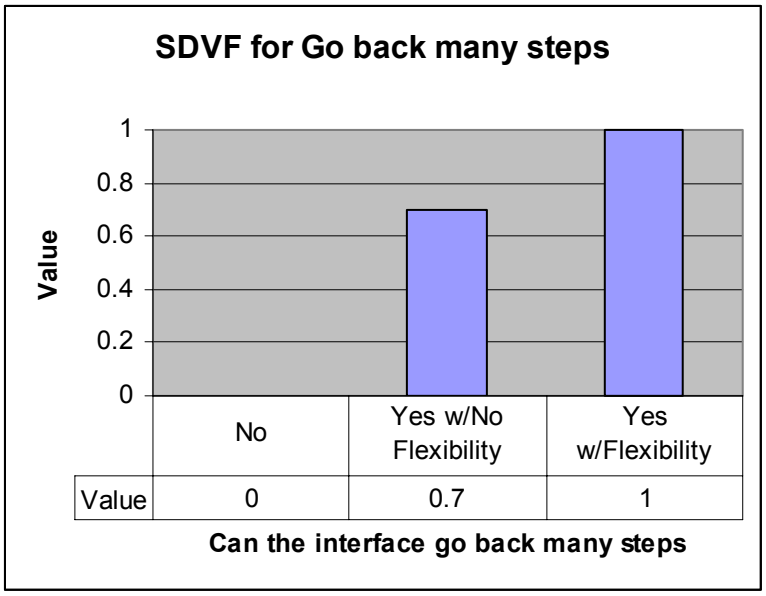

Figure 78: SDVF for Go back many steps

Table 59: Definitions for categories of Go back many steps

Category	Definition
No	Defined as not being able to go back many steps
Yes w/No Flexibility	Defined as being able to go back a fixed amount of many steps
Yes w/Flexibility	Defined as being able to choose how many steps to go back

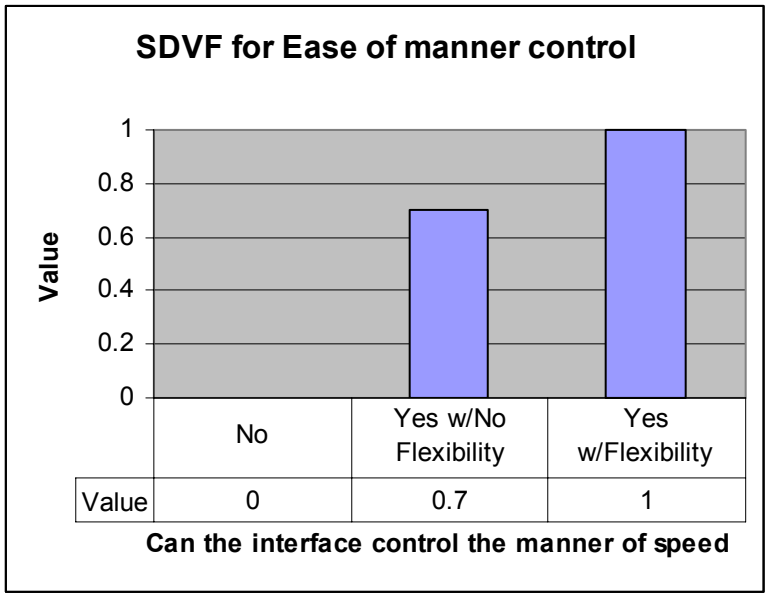

Figure 79: SDVF for Ease of manner control

Table 60: Definitions for categories of Ease of manner control

Category	Definition
No	Defined as not being able to go control the manner of speed
Yes w/No Flexibility	Defined as being able to control the manner of speed, but not being able to control how to do it.
Yes w/Flexibility	Defined as being able to control the manner of speed and being able to control how to do it.

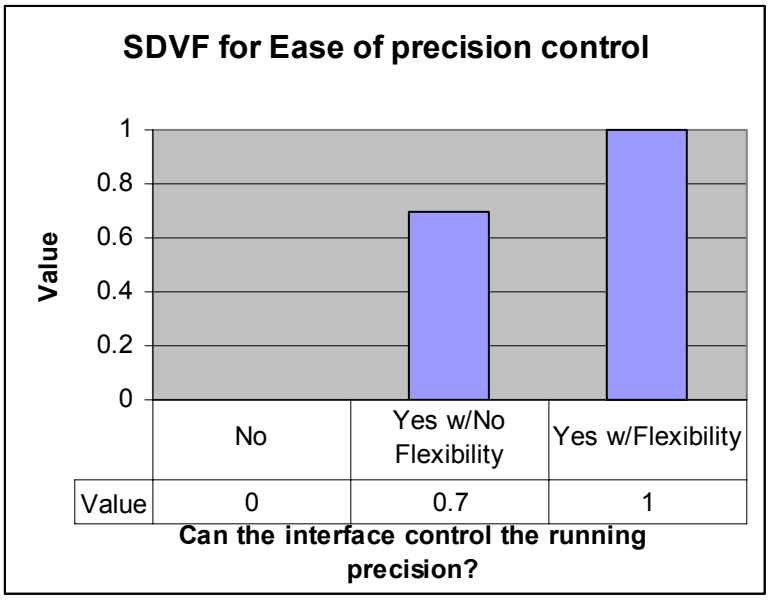

Figure 80: SDVF for Ease of precision control

Table 61: Definitions for categories of Ease of precision control

Category	Definition
No	Defined as not being able to go control the running precision
Yes w/No Flexibility	Defined as being able to control the running precision, but not being able to control how to do it.
Yes w/Flexibility	Defined as being able to control the running precision and being able to control how to do it.

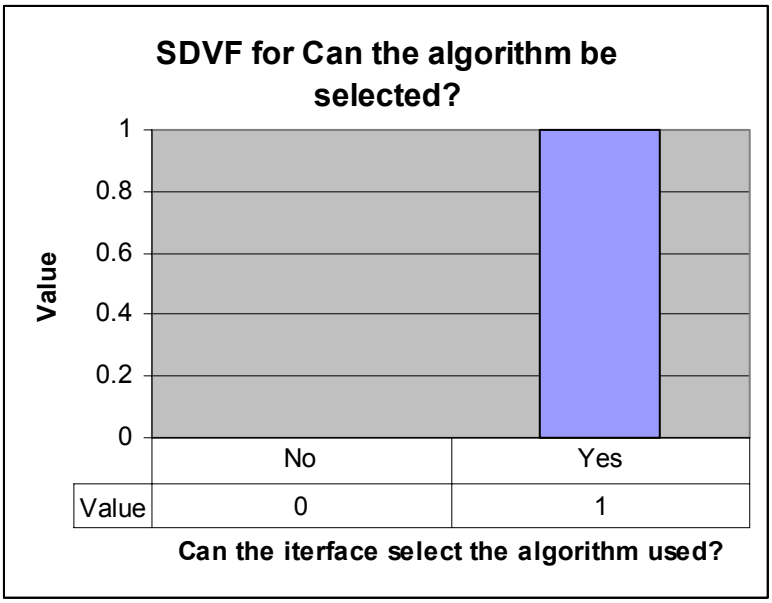

Figure 81: SDVF for Can the algorithm be selected?

Table 62: Definitions for categories of Can the algorithm be selected?

Category	Definition
No	Having no ability to select the algorithm that is used.
Yes	Having ability to select the algorithm that is used.

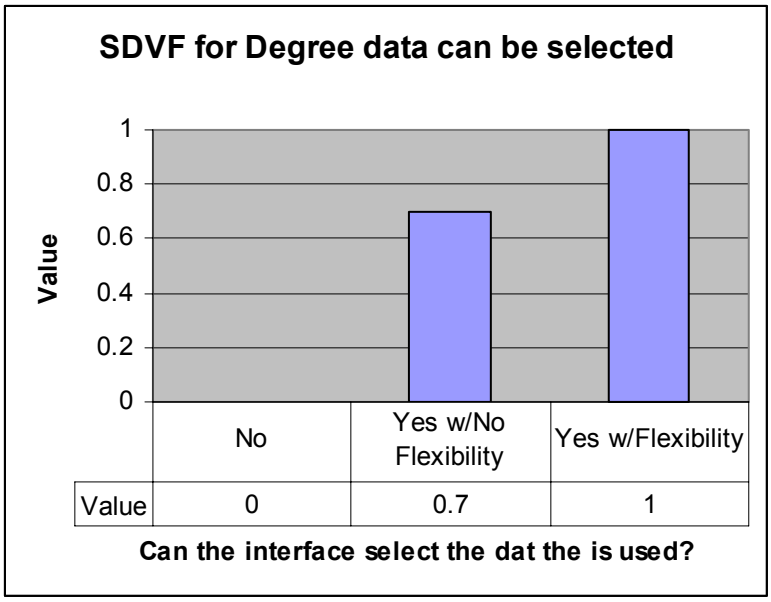

Figure 82: SDVF for Degree data can be selected

Table 63: Definitions for categories of Degree data can be selected

Category	Definition
No	Defined as not being able to go control the data selected
Yes w/No Flexibility	Defined as being able to control certain data selected, but not being able to choose from all of the data
Yes w/Flexibility	Defined as being able to select from any of the data

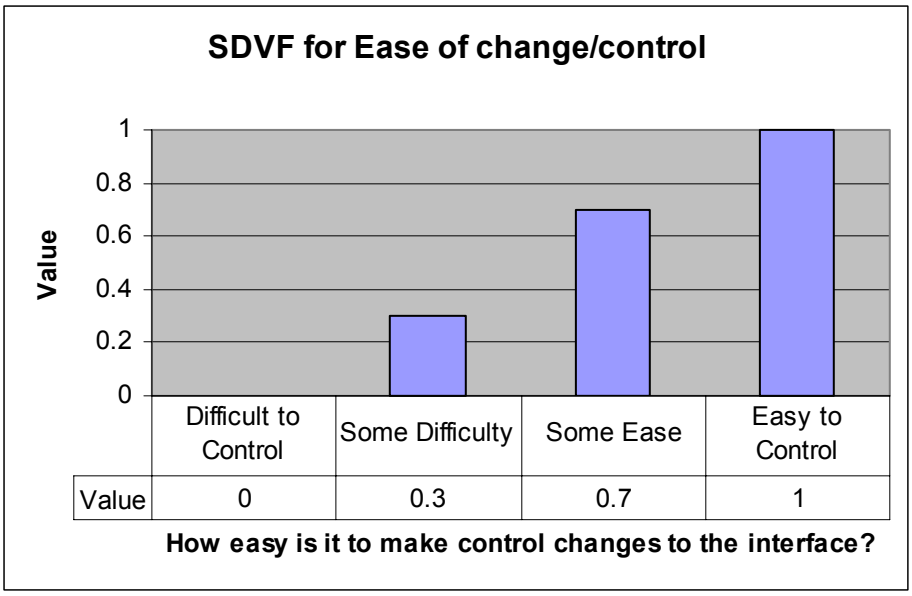

Figure 83: SDVF for Ease of change/control

Table 64: Definitions for categories of Ease of change/control

Category	Definition
Difficult to Control	Defined as being very difficult to change
Some Difficulty	Defined as being somewhat difficult to change.
Some Ease	Defined as somewhat easy to change.
Easy to Control	Defined as being easy to change.

The Output SDVFs

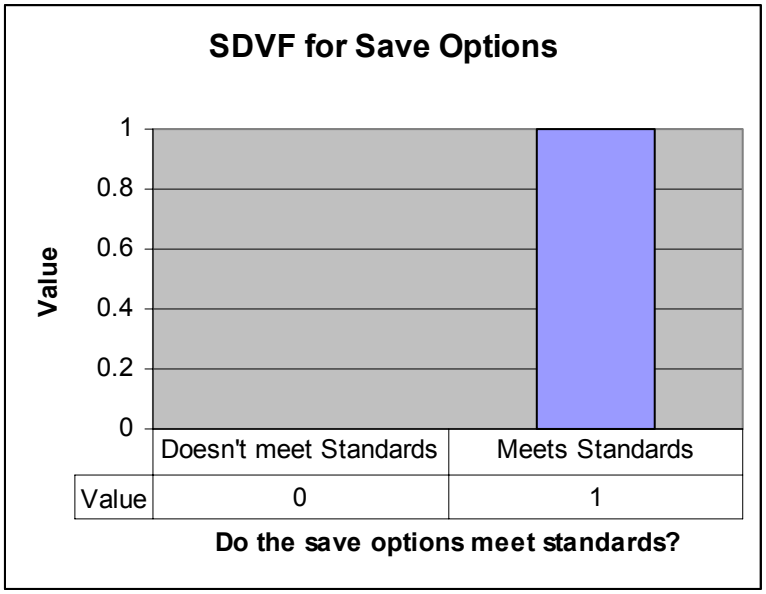

Figure 84: SDVF for Save Options

Table 65: Definitions for categories of Save Options

Category	Definition
No	Not meeting save option standards.
Yes	Meeting save options standards.

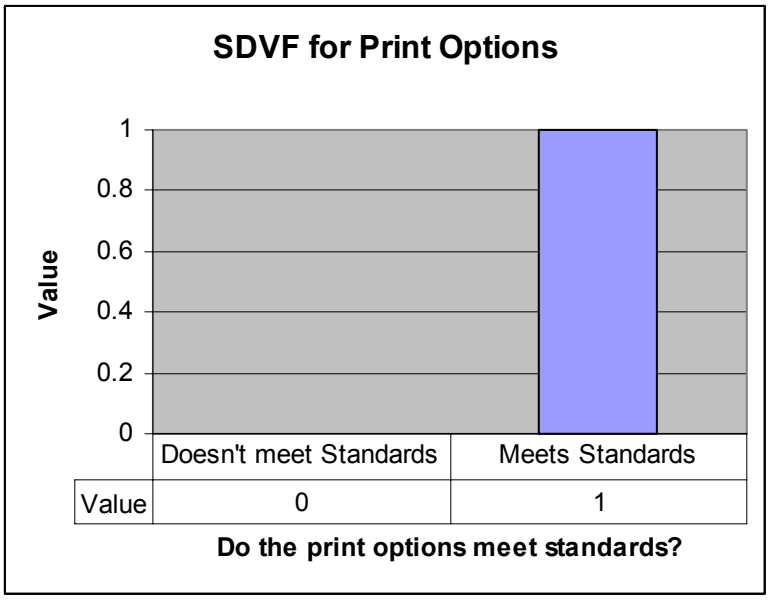

Figure 85: SDVF for Print Options

Table 66: Definitions for categories of Print Options

Category	Definition
No	Not meeting print option standards.
Yes	Meeting print options standards.

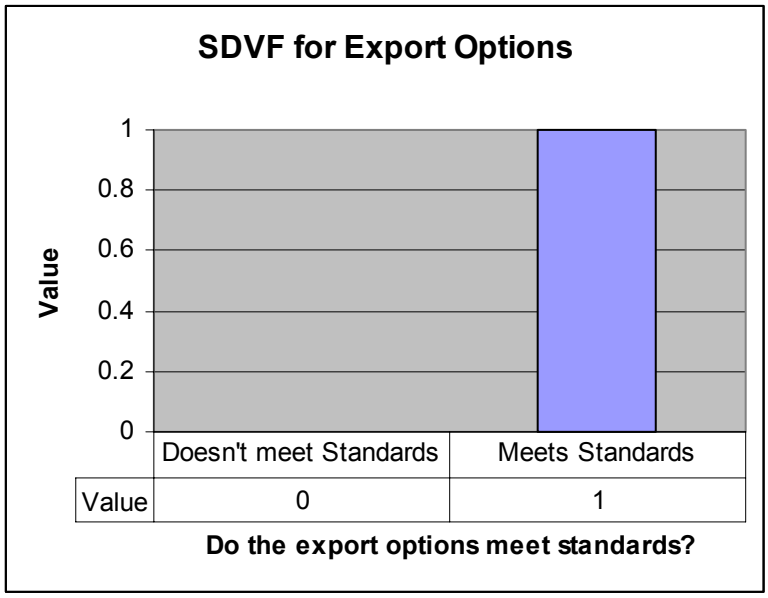

Figure 86: SDVF for Export Options

Table 67: Definitions for categories of Export Options

Category	Definition
No	Not meeting export option standards.
Yes	Meeting export options standards.

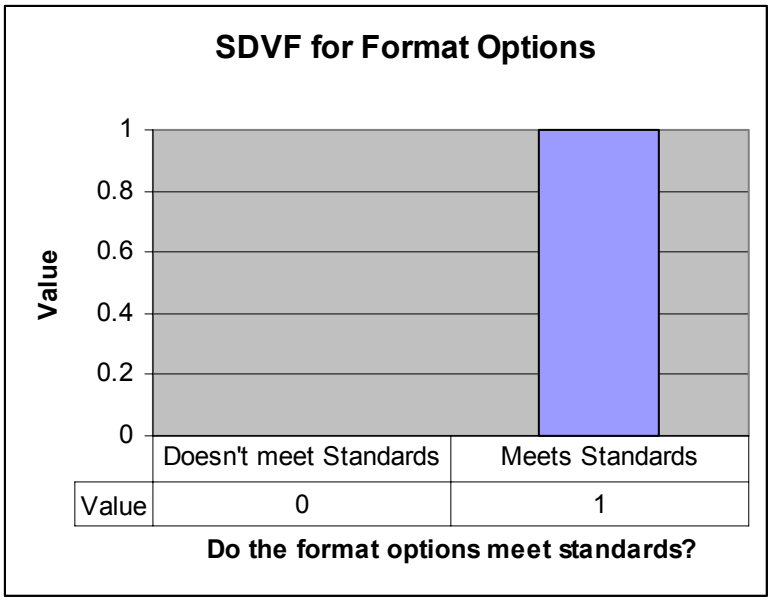

Figure 87: SDVF for Format Options

Table 68: Definitions for categories of Format Options

Category	Definition
No	Not meeting format option standards.
Yes	Meeting format options standards.

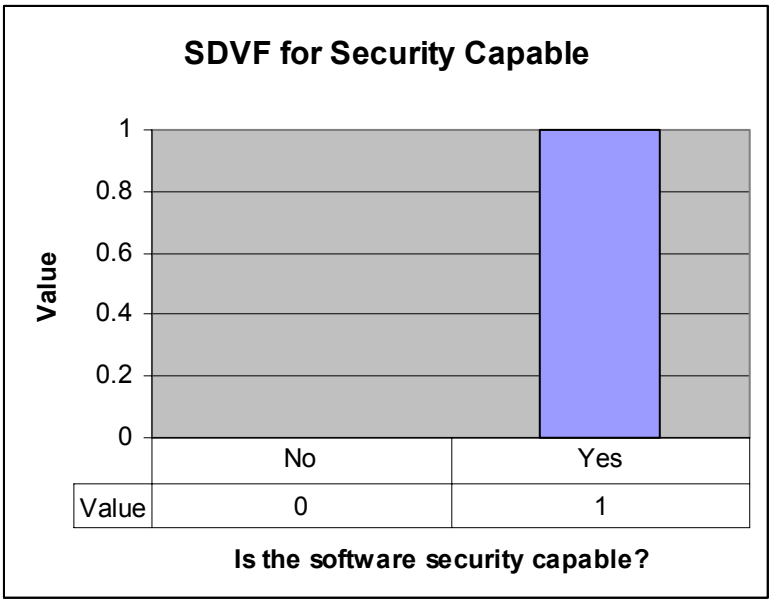

Figure 88: SDVF for Security Capable

Table 69: Definitions for categories of Security Capable

Category	Definition
No	Having no security capabilities.
Yes	Having security capabilities.

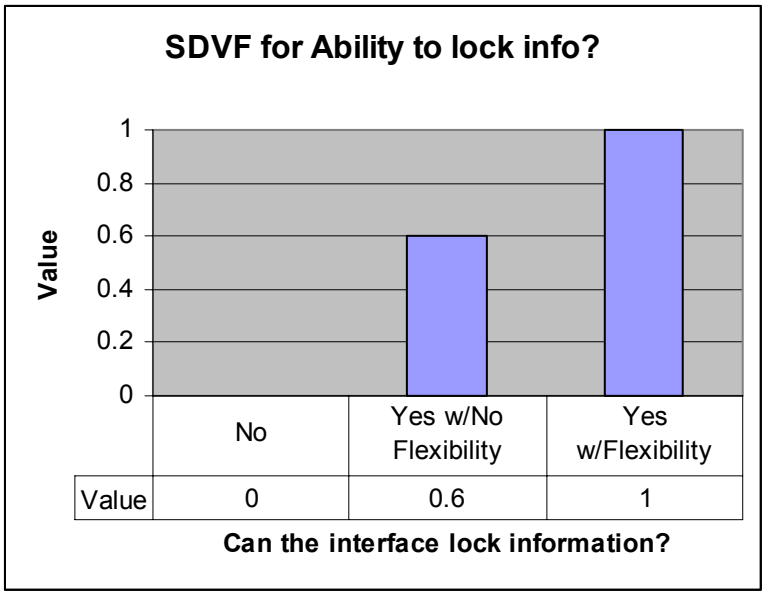

Figure 89: SDVF for Ability to lock info?

Table 70: Definitions for categories of Ability to lock info?

Category	Definition
No	Not being able to lock info
Yes w/No Flexibility	Able to lock the entire program from outside use.
Yes w/Flexibility	Ability to choose which information is locked to outside users and which is not.

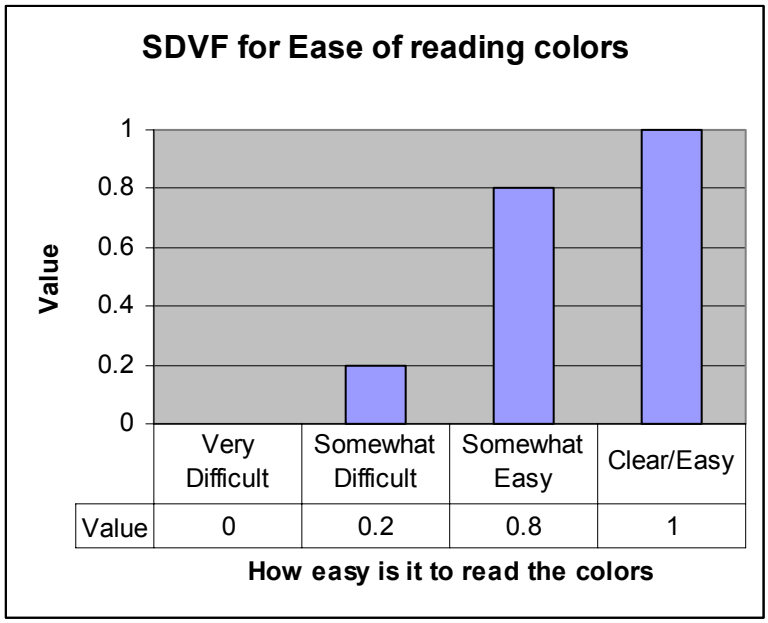

Figure 90: SDVF for Ease of reading colors

Table 71: Definitions for categories of Ease of reading colors

Category	Definition
Very Difficult	Defined as being very difficult to read
Somewhat Difficult	Defined as being somewhat difficult to read.
Somewhat Easy	Defined as somewhat easy to read.
Clear/Easy	Defined as being easy to read.

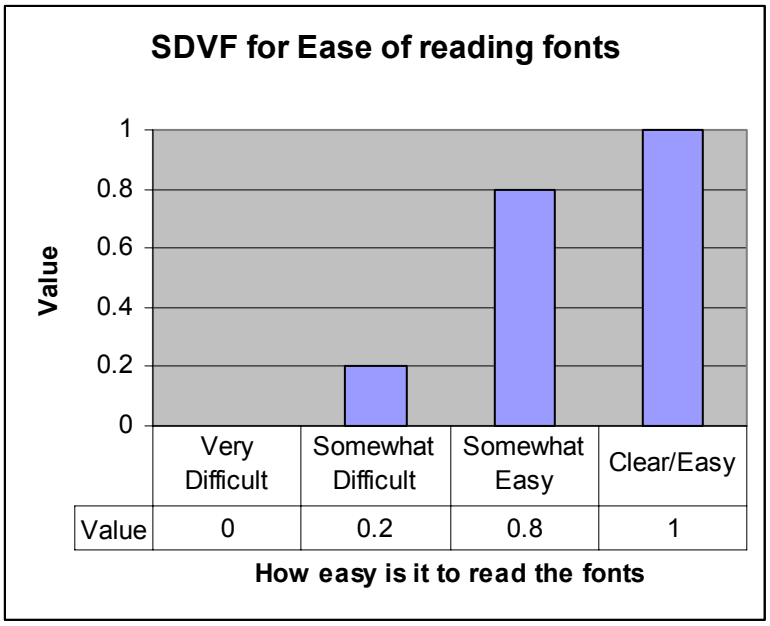

Figure 91: SDVF for Ease of reading fonts

Table 72: Definitions for categories of Ease of reading fonts

Category	Definition
Very Difficult	Defined as being very difficult to read
Somewhat Difficult	Defined as being somewhat difficult to read.
Somewhat Easy	Defined as somewhat easy to read.
Clear/Easy	Defined as being easy to read.

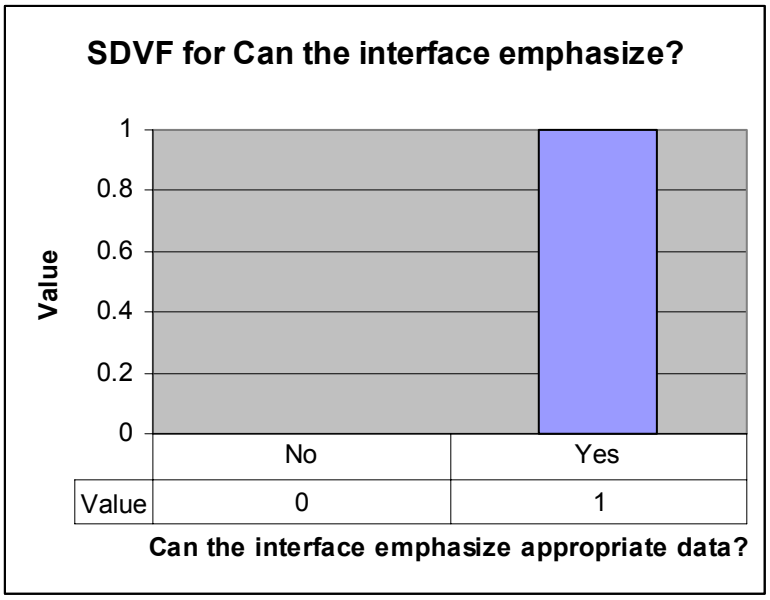

Figure 92: SDVF for Can the interface emphasize?

Table 73: Definitions for categories of Can the interface emphasize?

Category	Definition
No	Not having the ability to emphasize data.
Yes	Having the ability to emphasize data.

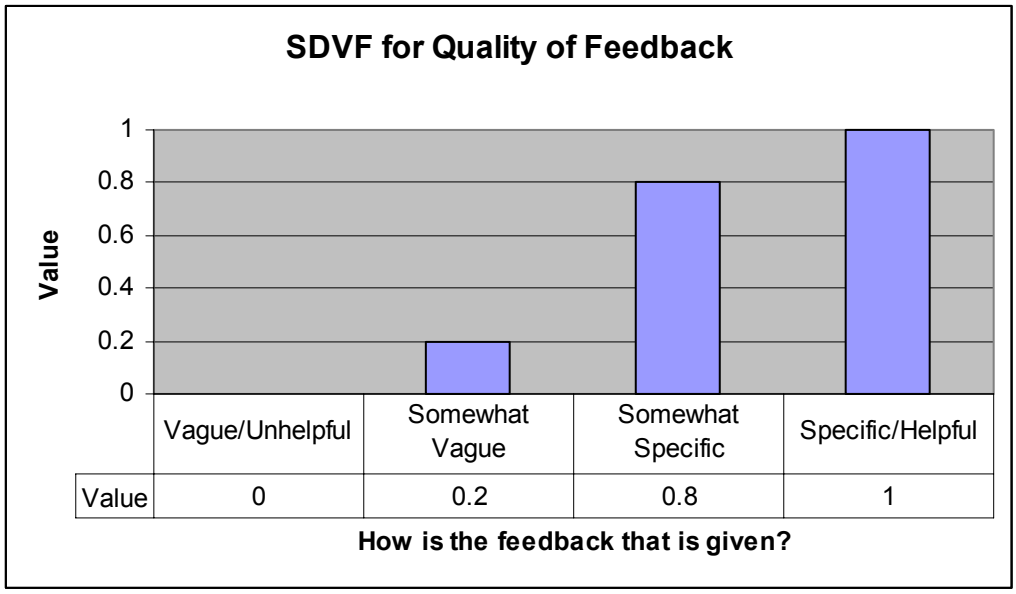

Figure 93: SDVF for Quality of Feedback

Table 74: Definitions for categories of Quality of Feedback

Category	Definition
Vague/Unhelpful	Feedback that is vague and unhelpful
Somewhat Vague	Feedback that can be understood somewhat, but is somewhat vague and is difficult to interpret.
Somewhat Specific	Feedback that can be understood and is somewhat specific, but can still be up to interpretation.
Specific/Helpful	Feedback that can be understood and is specific in its details.

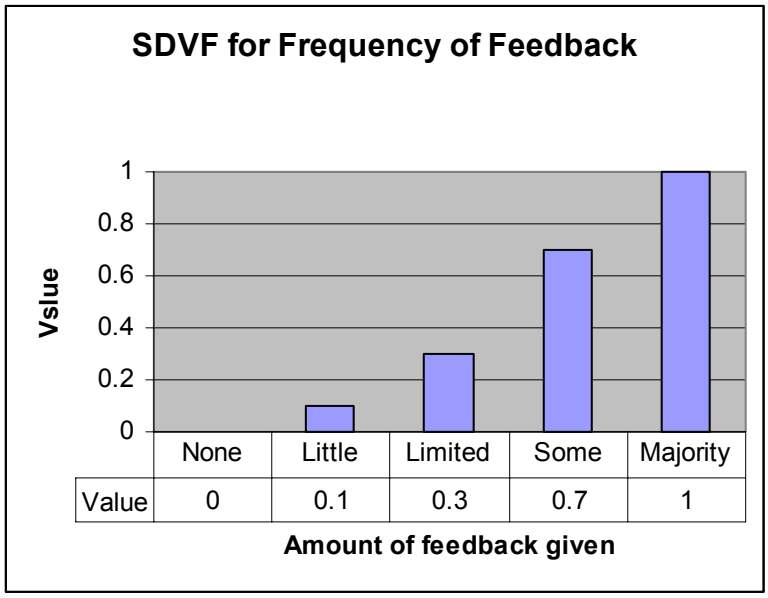

Figure 94: SDVF for Frequency of feedback

Table 75: Definitions for categories of Frequency of feedback

Category	Definition
None	Having no feedback.
Little	Having some feedback given, but less than or equal to 10% of the time having it.
Limited	Having greater the 10% feedback given, but less than or equal to 50% of the time having it.
Some	Having greater the 50% feedback given, but less than or equal to 80% of the time having it.
Majority	Having greater the 80% feedback given.

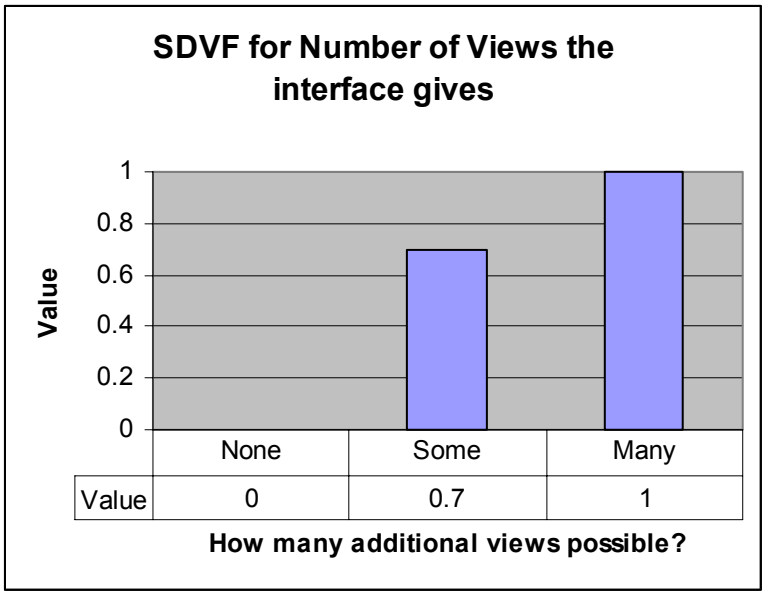

Figure 95: SDVF for Number of Views the interface gives

Table 76: Definitions for categories of Number of Views the interface gives

Category	Definition
None	Having no options available
Some	Having some options available
Many	Having many options available

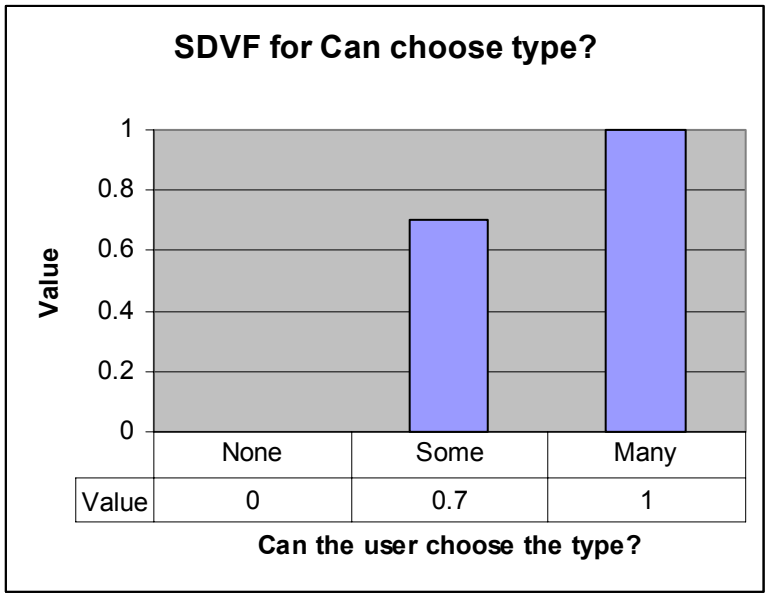

Figure 96: SDVF for Can choose type?

Table 77: Definitions for categories of Can choose type?

Category	Definition
None	Having no more options available
Some	Having some options available
Many	Having many options available

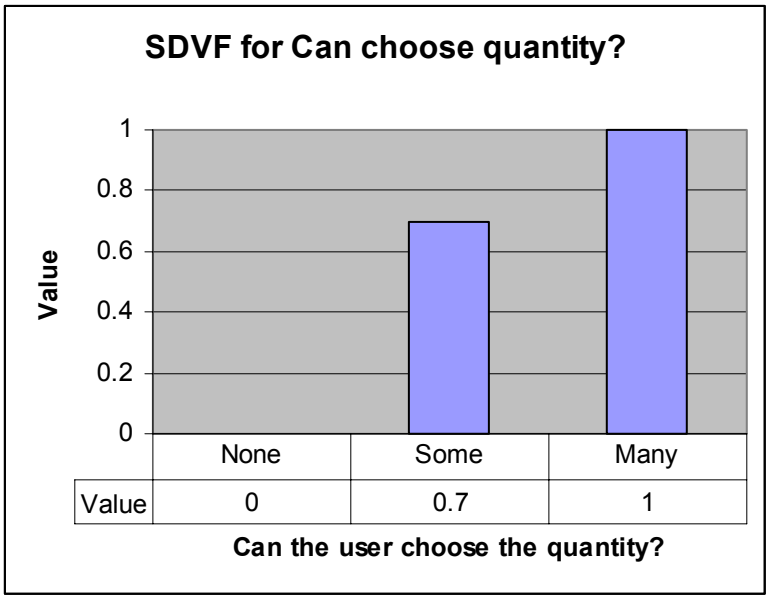

Figure 97: SDVF for Can choose quantity?

Table 78: Definitions for categories of Can choose quantity?

Category	Definition
None	Having no more options available
Some	Having some options available
Many	Having many options available

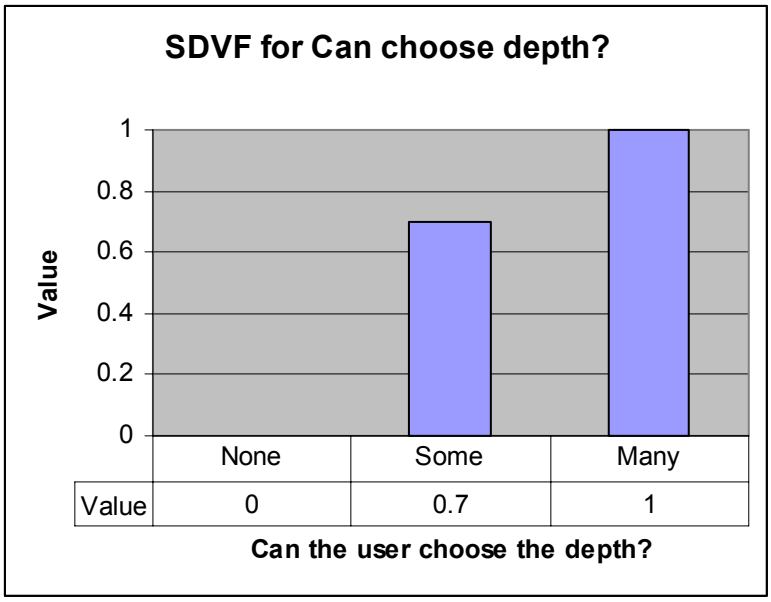

Figure 98: SDVF for Can choose depth?

Table 79: Definitions for categories of Can choose depth?

Category	Definition
None	Having no more options available
Some	Having some options available
Many	Having many options available

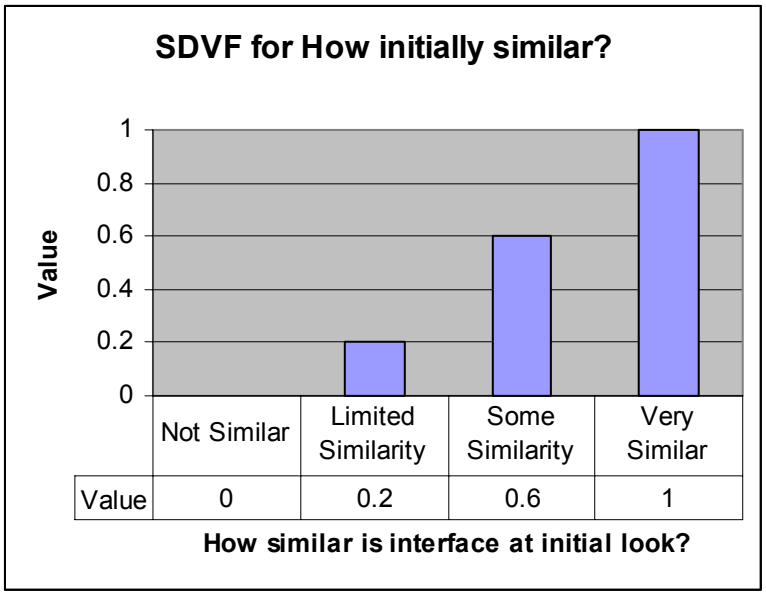

Figure 99: SDVF for How initially similar?

Table 80: Definitions for categories of How initially similar?

Category	Definition
Not Similar	Defined as the interface not being similar to other software when initially looked at.
Limited Similarity	Defined as the interface having few similarities to other software when initially looked at.
Some Similarity	Defined as the interface having more than a few similarities to other software when initially looked at.
Very Similar	Defined as the interface being very similar to other software when initially looked at.

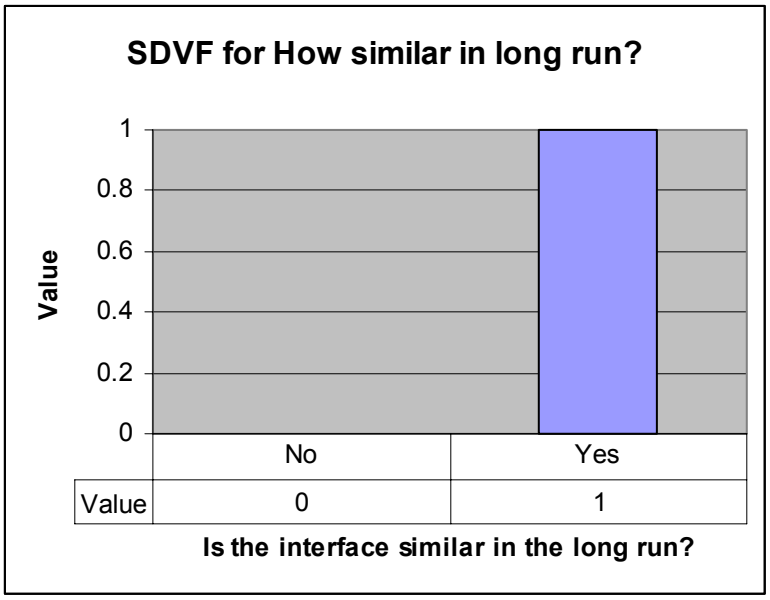

Figure 100: SDVF for How similar in long run?

Table 81: Definitions for categories of How similar in long run?

Category	Definition
No	Not being similar to other software in the long run.
Yes	Being similar to other software in the long run.

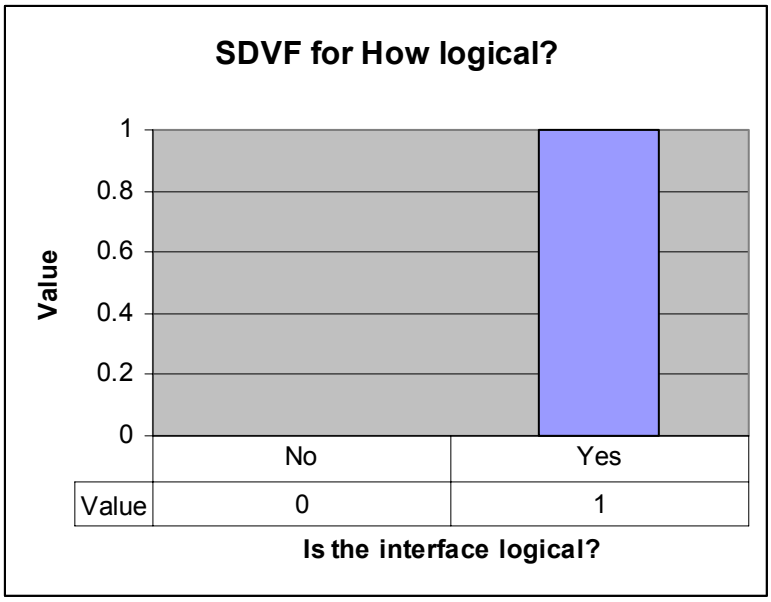

Figure 101: SDVF for How logical?

Table 82: Definitions for categories of How logical?

Category	Definition
No	Not being logical.
Yes	Being logical.

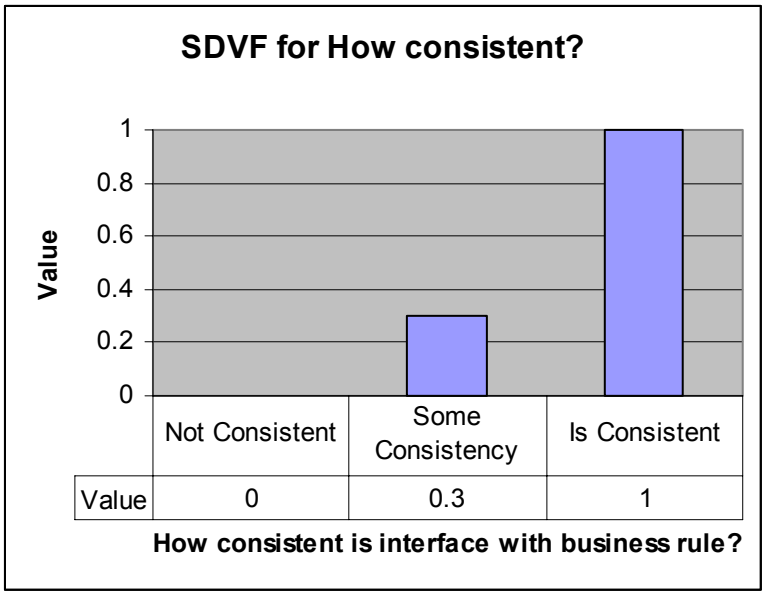

Figure 102: SDVF for How consistent?

Table 83: Definitions for categories of How consistent?

Category	Definition
Not Consistent	Not having consistency with the context of the user.
Some Consistency	Having some consistency with the context of the user, but not total.
Is consistent	Having consistency with the context of the user.

Appendix D

The following are the results from weighting of the hierarchy. The figures below show the local weighting of each branch with the global weights in parentheses next to the local weights.

Input weighting

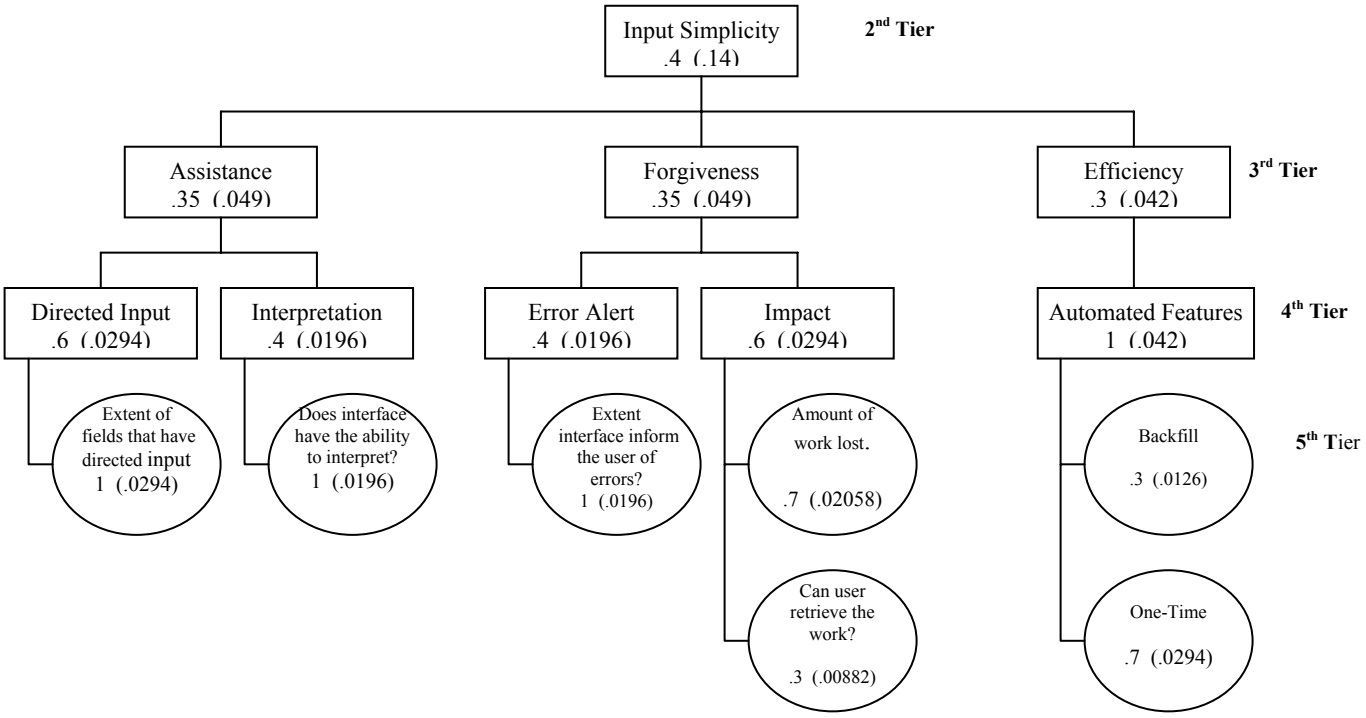

Figure 103: Weighting of the Input Simplicity branch

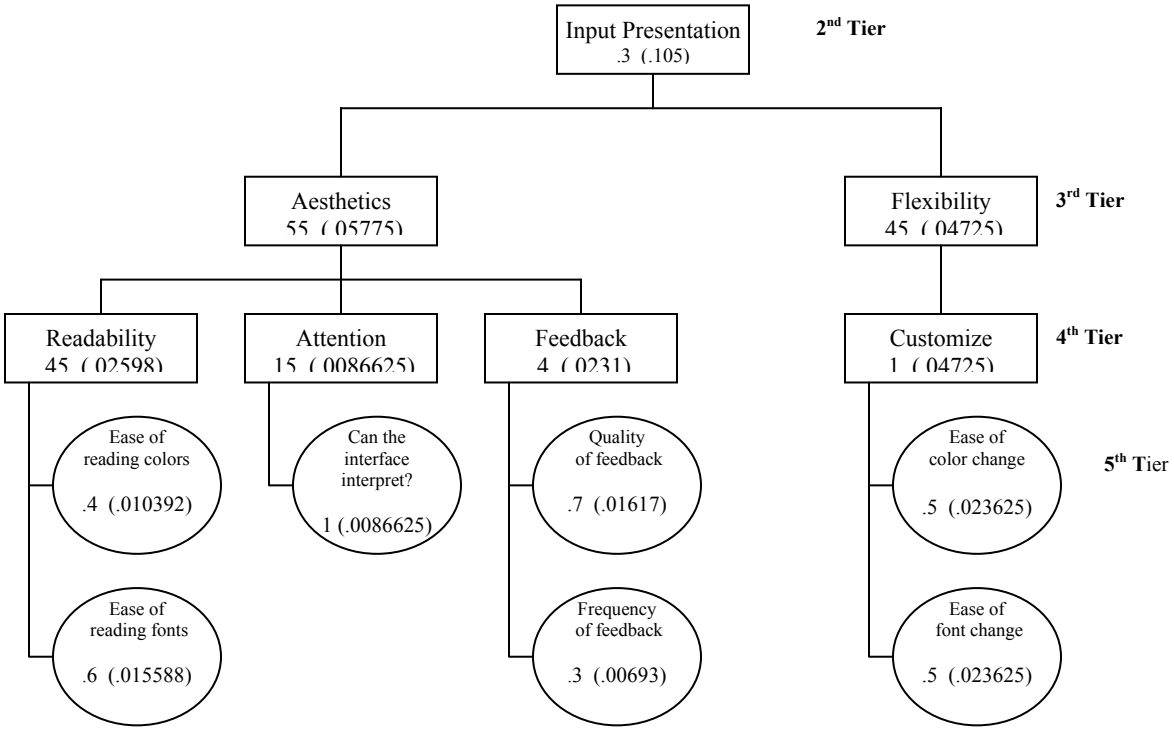

Figure 104: Weighting of the Input Presentation branch

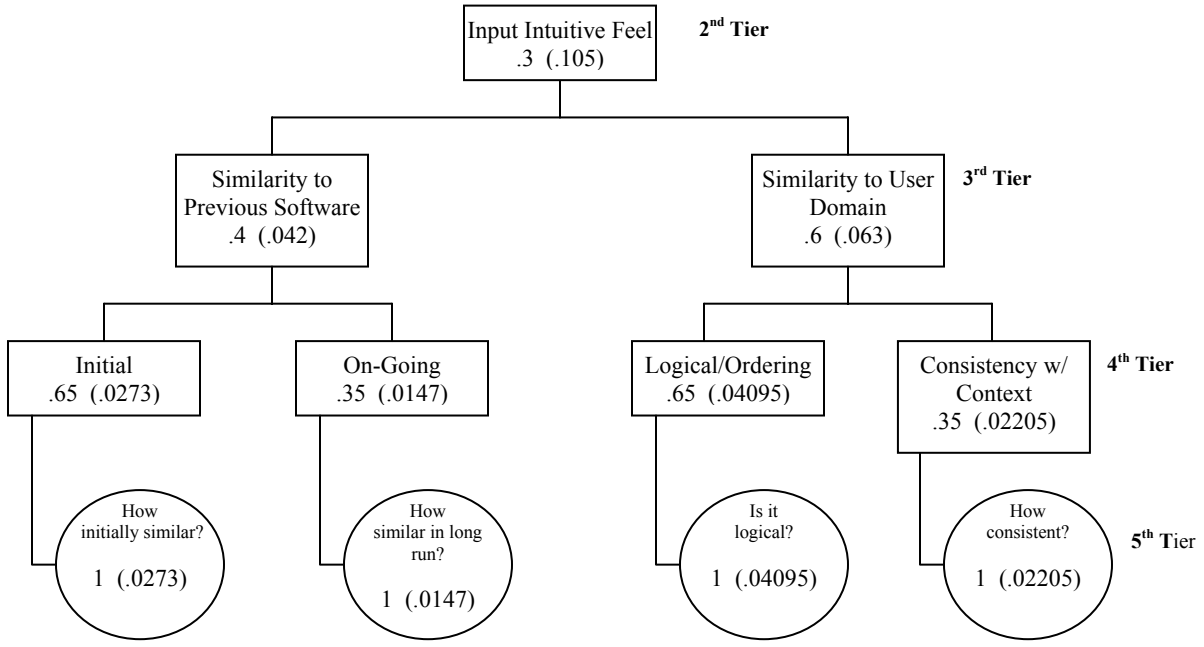

Figure 105: Weighting of the Input Intuitive Feel branch

Processing weighting

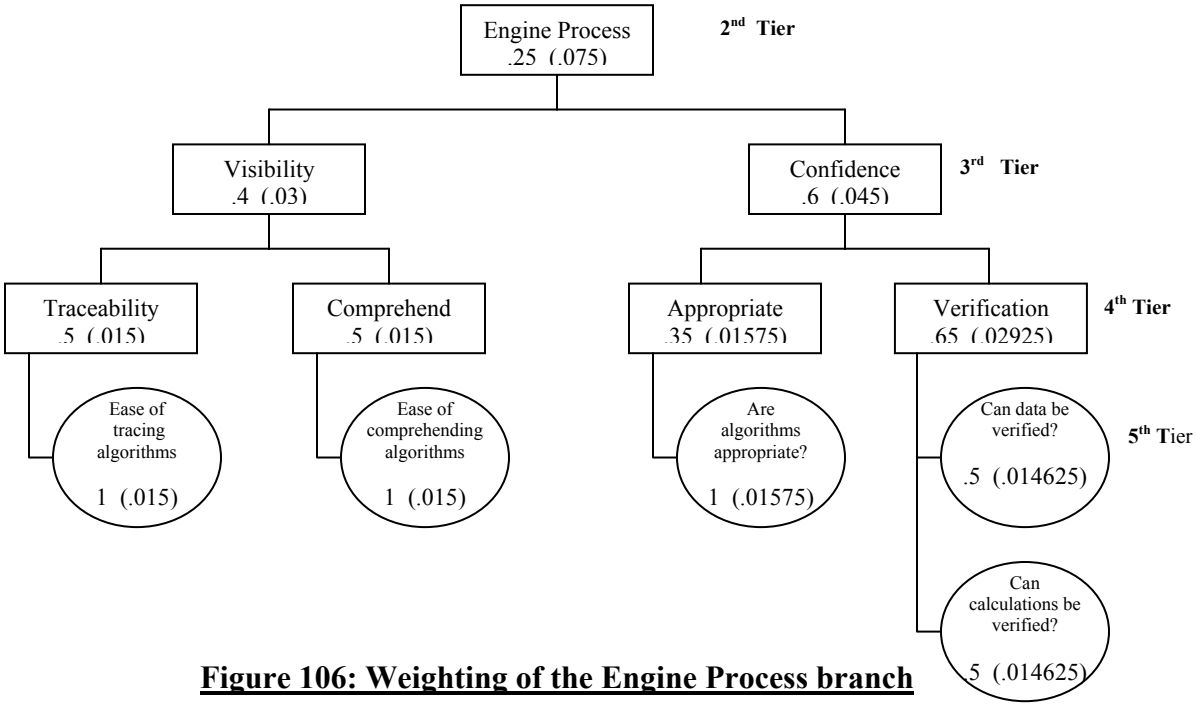

Figure 106: Weighting of the Engine Process branch

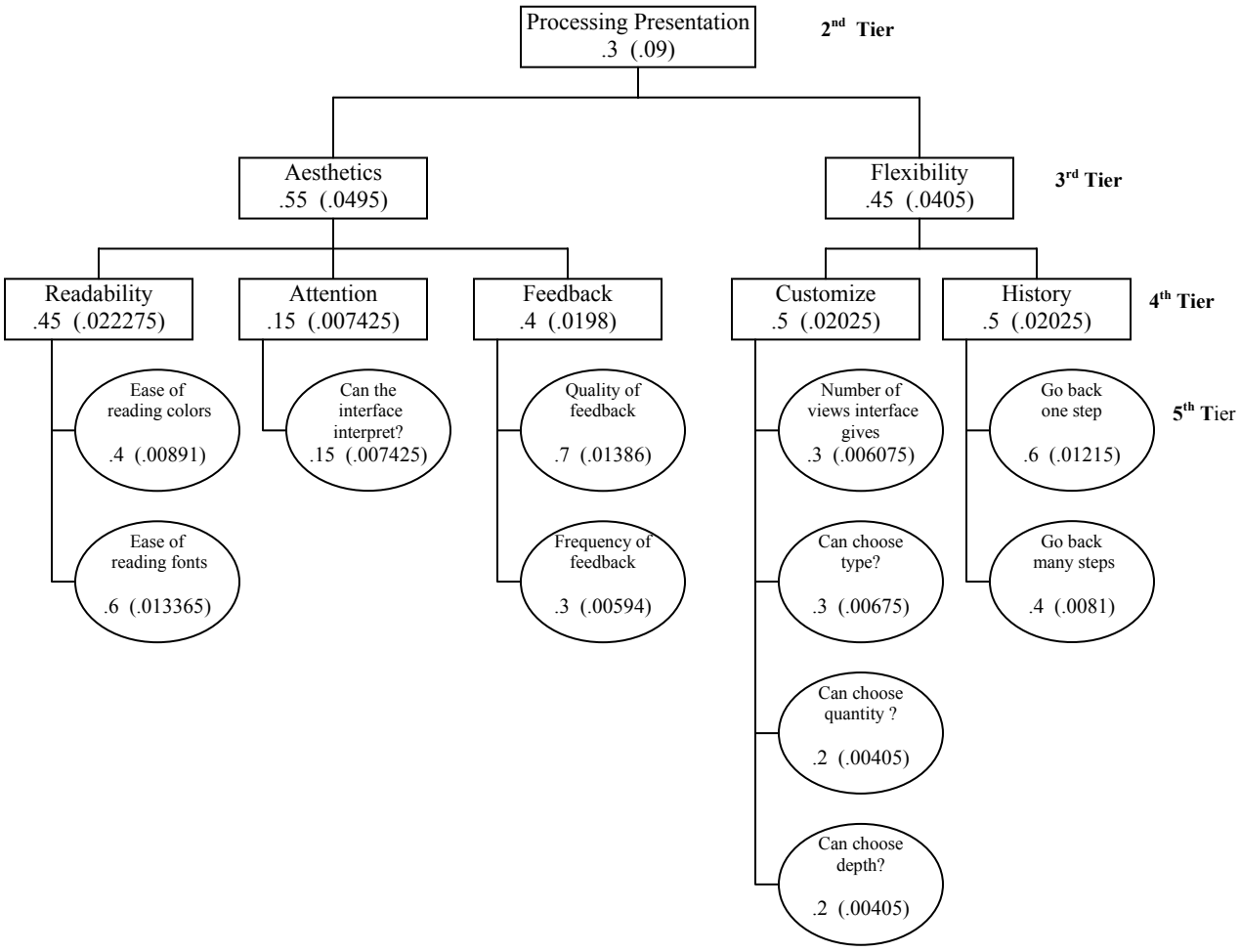

Figure 107: Weighting of the Processing Presentation branch

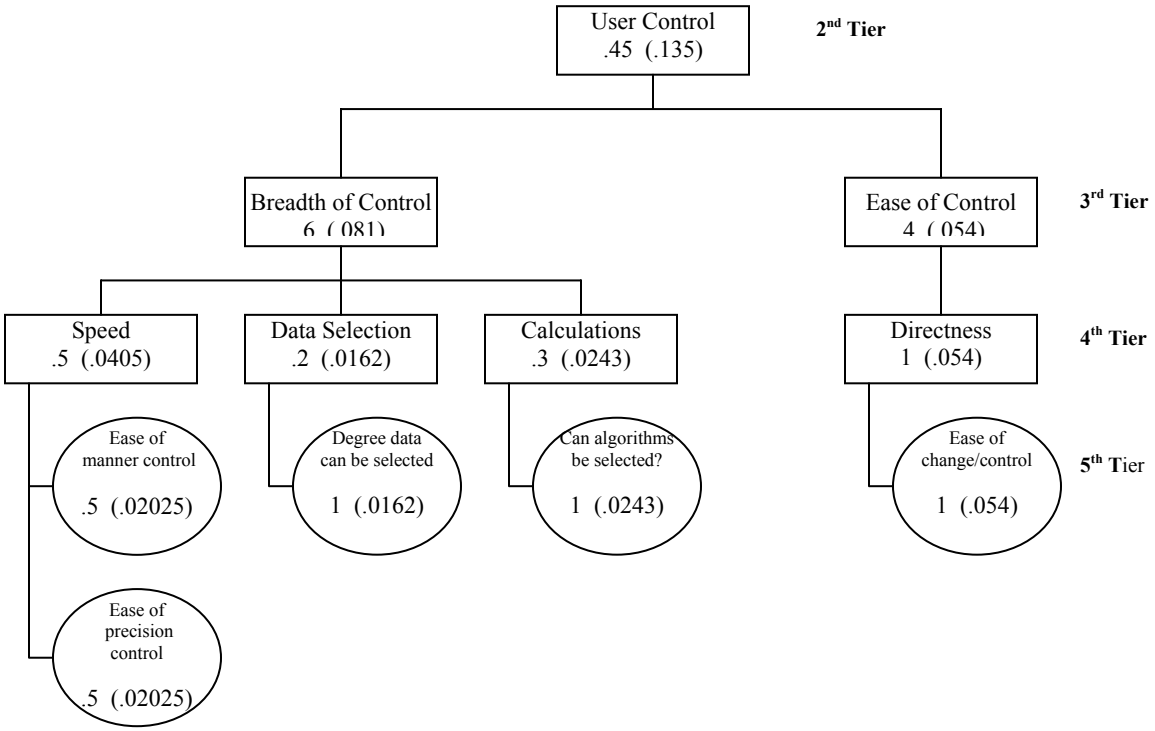

Figure 108: Weighting of the User Control branch

Processing weighting

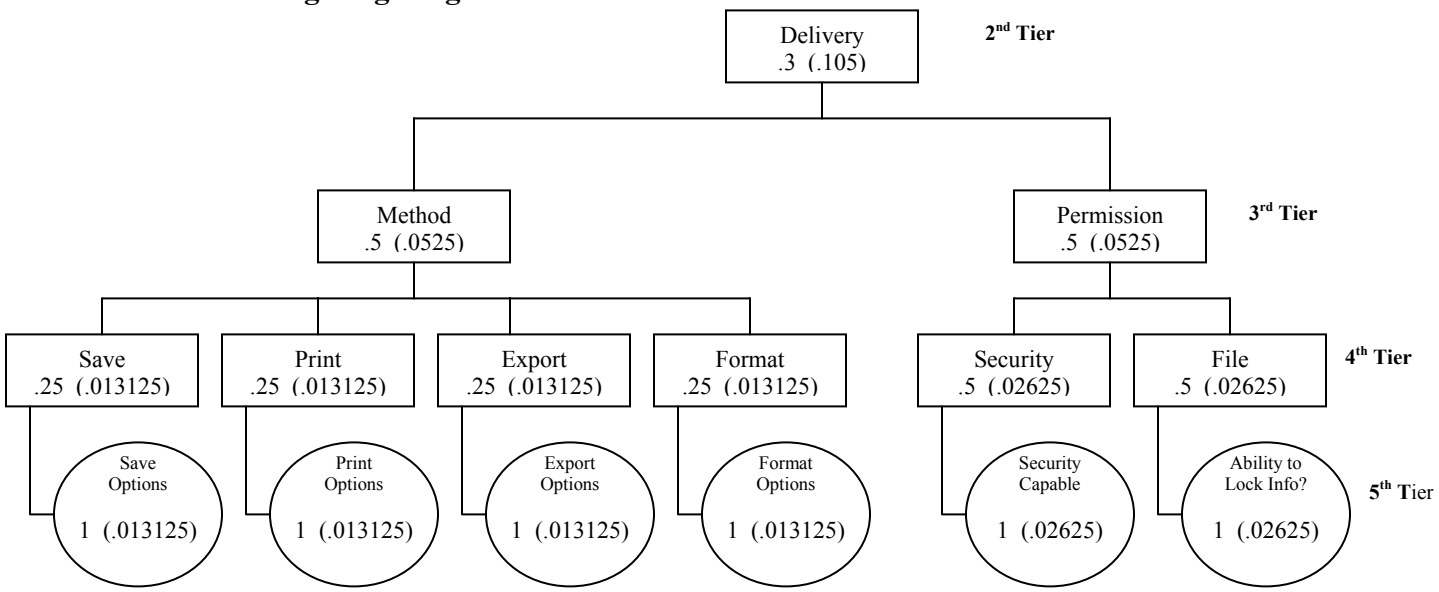

Figure 109: Weighting of the Delivery branch

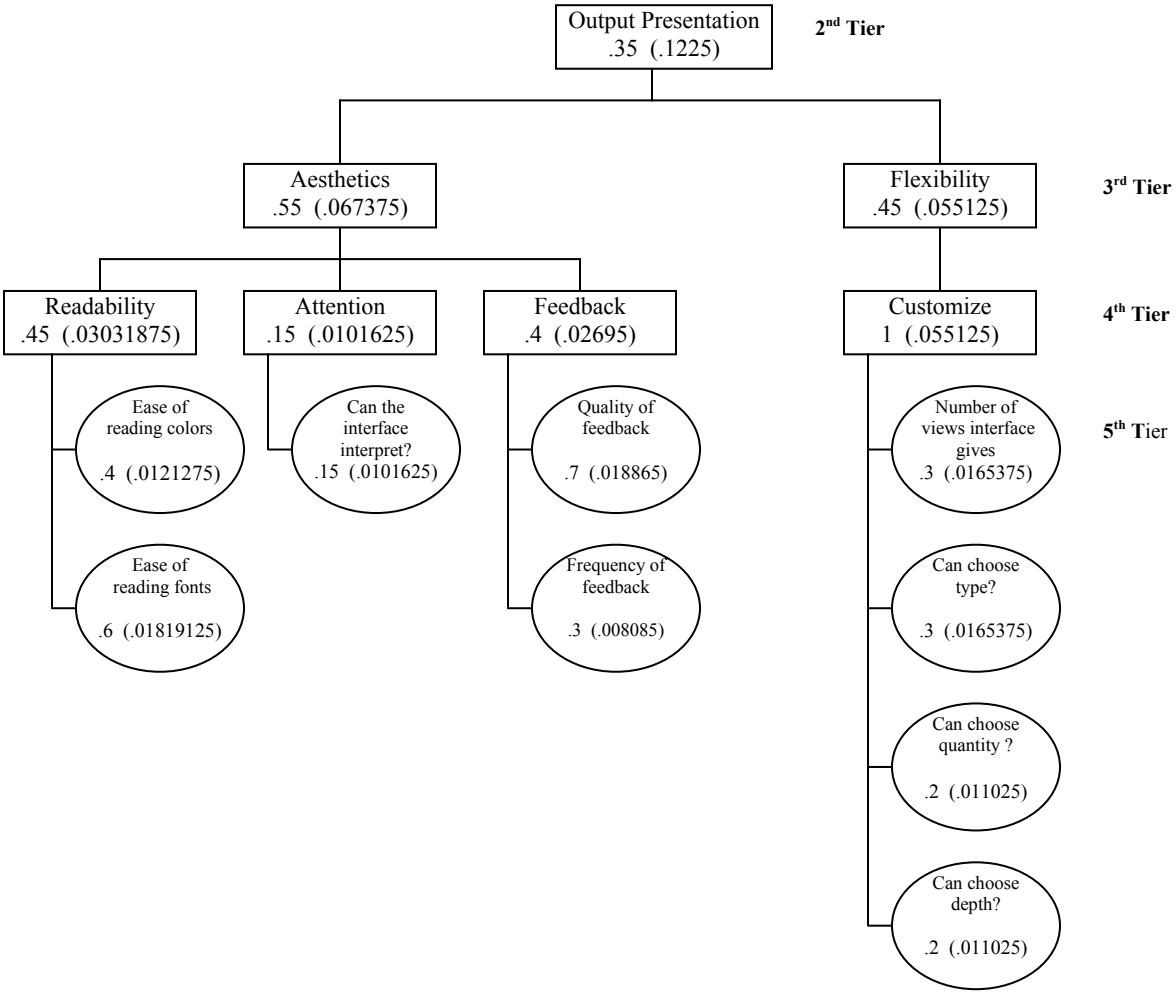

Figure 110: Weighting of the Output Presentation branch

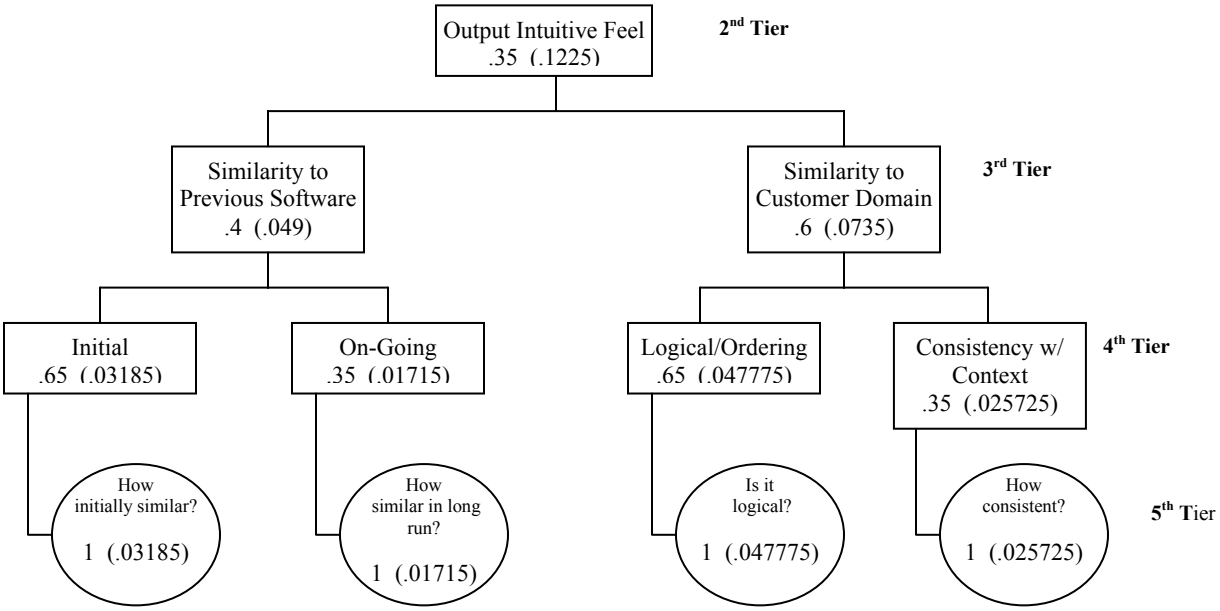

Figure 111: Weighting of the Output Intuitive Feel branch

Appendix E

Given below is a breakdown of the measures that comprise the hierarchy. The table demonstrates the value of each measure and how it contributes to the overall goal. For each measure, the breakdown shows the category the baseline fell into, the global weight of the measure (as determined by the group) and the overall value for that measure (as found by multiplying the score and the global weight). Given at the bottom of the table is the total possible value the component can contribute to the goal as well the value that the baseline actually received. This is given in the figures below.

Value Input Received

Table 84: Input Scoring

Measure	Global	Value
Extent of fields that have directed input	0.0294	0.00294
Does the interface have the ability to interpret?	0.0196	0
Extent interface inform the user of errors?	0.0196	0.00196
Amount of work lost.	0.02058	0
Can user retrieve the work?	0.00882	0
Backfill	0.0126	0
One-Time	0.0294	0
Ease of reading colors.	0.010392	0.0083136
Ease of reading fonts.	0.015588	0.0124704
Can the interface emphasize?	0.008663	0
Quality of Feedback	0.01617	0
Frequency of Feedback	0.00693	0.004851
Ease of color change.	0.023625	0
Ease of font change.	0.023625	0
How initially similar?	0.0273	0.01638
How similar in long run?	0.0147	0
Is it logical?	0.04095	0
How consistent?	0.02205	0.006615

Total possible value from Input	0.35	
Value received from Input		0.05

Value Processing Received

Table 85: Processing Scoring

Measure	Global	Value
Ease of tracing algorithms	0.015	0
Ease of comprehending algorithm	0.015	0
Are algorithms appropriate	0.01575	0
Can data be verified?	0.014625	0
Can calculations be verified	0.014625	0
Ease of reading colors.	0.00891	0.001782
Ease of reading fonts.	0.013365	0.010692
Can the interface emphasize?	0.007425	0
Quality of Feedback	0.01386	0.002772
Frequency of Feedback	0.00594	0.001782
Number of views interface gives.	0.006075	0
Can choose type?	0.006075	0
Can choose quantity?	0.00405	0
Can choose depth?	0.00405	0
Go back one step	0.01215	0
Go back many step	0.0081	0
Ease of manner control	0.02025	0
Ease of precision control	0.02025	0
Degree data can be selected	0.0162	0
Can algorithm be selected	0.0243	0.0243
Ease of Changes/Control	0.054	0.0378
Total possible value from Processing	0.30	
Value received from Processing		0.08

Value Output Received

Table 86: Output Scoring

Measure	Global	Value
Save options	0.013125	0
Print options	0.013125	0
Export options	0.013125	0
Format options	0.013125	0
Security Capable	0.02625	0
Ability to lock info?	0.02625	0
Ease of reading colors.	0.0121275	0
Ease of reading fonts.	0.01819125	0
Can the interface emphasize?	0.0101625	0
Quality of Feedback	0.018865	0
Frequency of Feedback	0.008085	0
Number of views interface gives.	0.0165375	0
Can choose type?	0.0165375	0
Can choose quantity?	0.011025	0
Can choose depth?	0.011025	0
How initially similar?	0.03185	0
How similar in long run?	0.01715	0
Is it logical?	0.047775	0
How consistent?	0.025725	0
Total possible value from Output	0.35	
Value received from Output		0.00

Appendix F

To identify the relative importance of each measure in the each component, the measures of this component must be ranked by their possible improvement. The ranking of the each component's measures by possible improvement is given in the tables X-Y. The measures highlighted are the top sever measures that were used in Chapter 4. Figures X-Y identify the value gaps the measures have to fill.

Input Value Gaps

Table 87: Rank of Input possible improvements

Possible gain in Value	Measure
0.04095	Is it logical?
0.0294	One-Time
0.02646	Extent of fields that have directed input
0.023625	Ease of font change.
0.023625	Ease of color change.
0.02058	Amount of work lost.
0.0196	Does the interface have the ability to interpret?
0.01764	Extent interface inform the user of errors?
0.01617	Quality of Feedback
0.015435	How consistent?
0.0147	How similar in long run?
0.0126	Backfill
0.01092	How initially similar?
0.00882	Can user retrieve the work?
0.0086625	Can the interface emphasize?
0.0031176	Ease of reading fonts.
0.002079	Frequency of Feedback
0.0020784	Ease of reading colors.

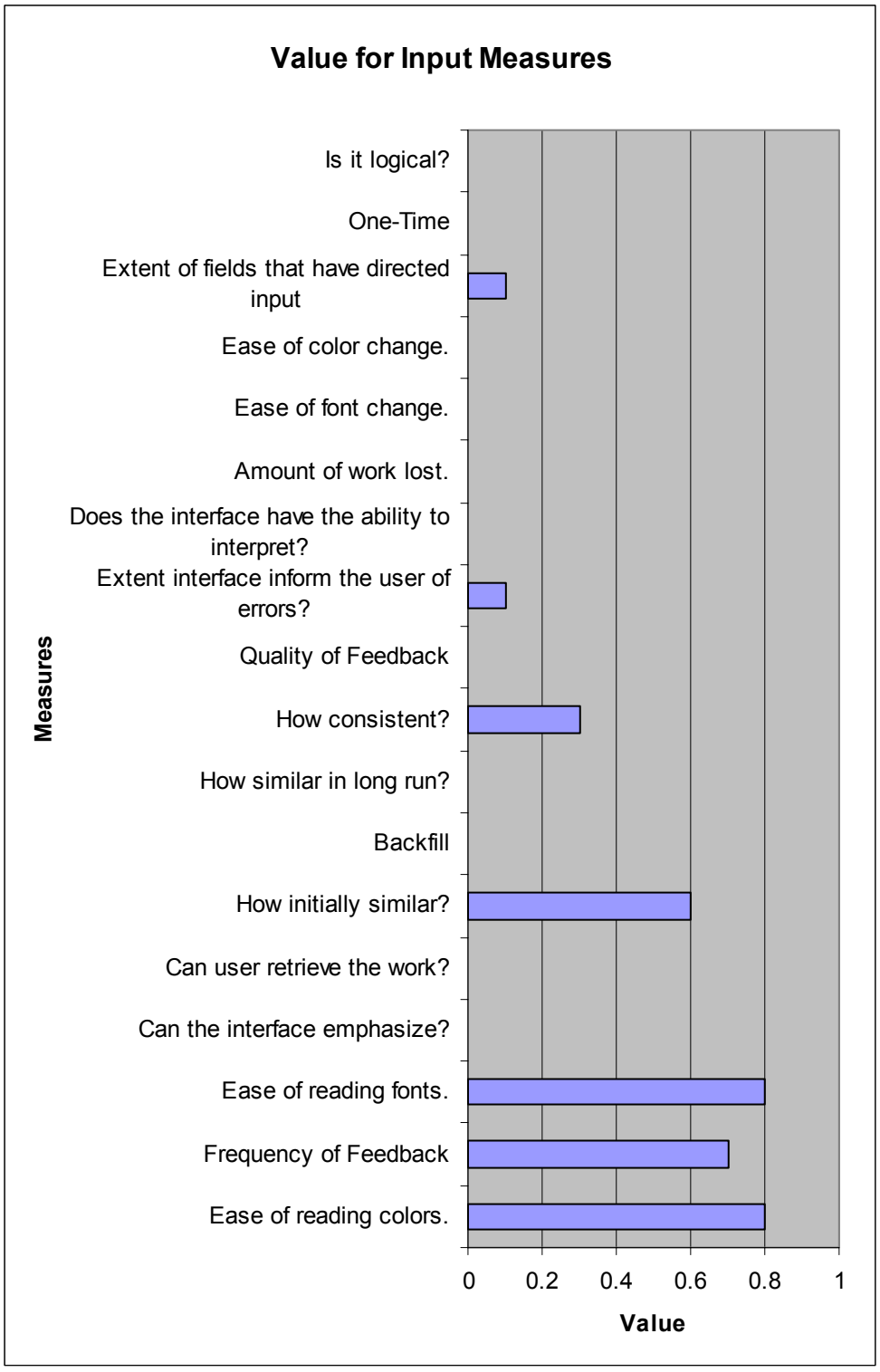

Figure 112: Input Value Gaps

Processing Value Gaps

Table 88: Rank of Processing possible improvements

Possible gain in Value	Measure
0.02025	Ease of manner control
0.02025	Ease of precision control
0.0162	Ease of Changes/Control
0.0162	Degree data can be selected
0.01575	Are algorithms appropriate
0.015	Ease of tracing algorithms
0.015	Ease of comprehending algorithm
0.014625	Can data be verified?
0.014625	Can calculations be verified
0.01215	Go back one step
0.011088	Quality of Feedback
0.0081	Go back many step
0.007425	Can the interface emphasize?
0.007128	Ease of reading colors.
0.006075	Number of views interface gives.
0.006075	Can choose type?
0.004158	Frequency of Feedback
0.00405	Can choose quantity?
0.00405	Can choose depth?
0.002673	Ease of reading fonts.
0	Can algorithm be selected

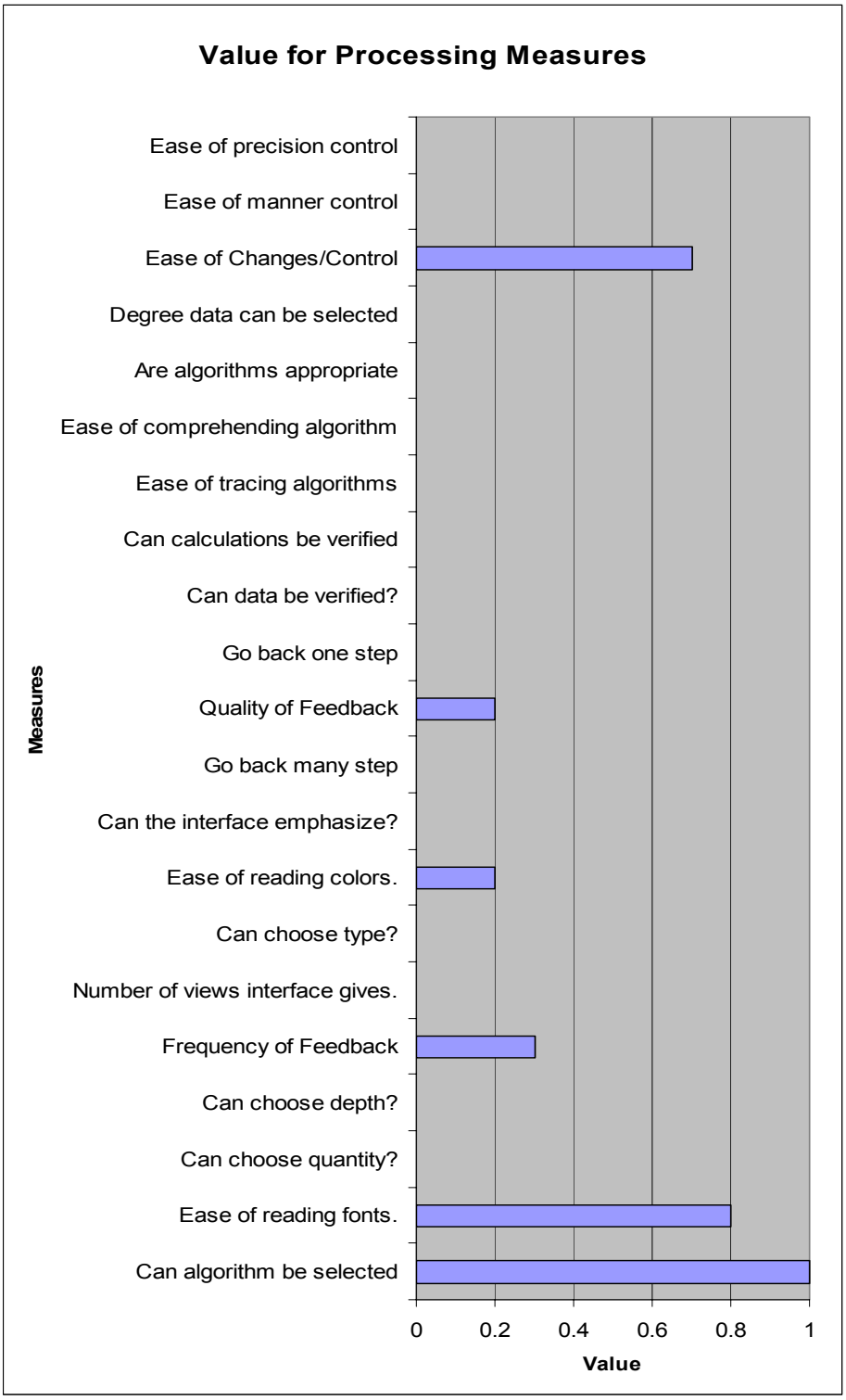

Figure 113: Processing Value Gaps

Output Value Gaps

Table 89: Rank of Output possible improvements

Possible Gain in Value	Measure
0.047775	Is it logical?
0.03185	How initially similar?
0.02625	Security Capable
0.02625	Ability to lock info?
0.025725	How consistent?
0.018865	Quality of Feedback
0.01819125	Ease of reading fonts.
0.01715	How similar in long run?
0.0165375	Number of views interface gives.
0.0165375	Can choose type?
0.013125	Save options
0.013125	Print options
0.013125	Export options
0.013125	Format options
0.0121275	Ease of reading colors.
0.011025	Can choose quantity?
0.011025	Can choose depth?
0.0101625	Can the interface emphasize?
0.008085	Frequency of Feedback

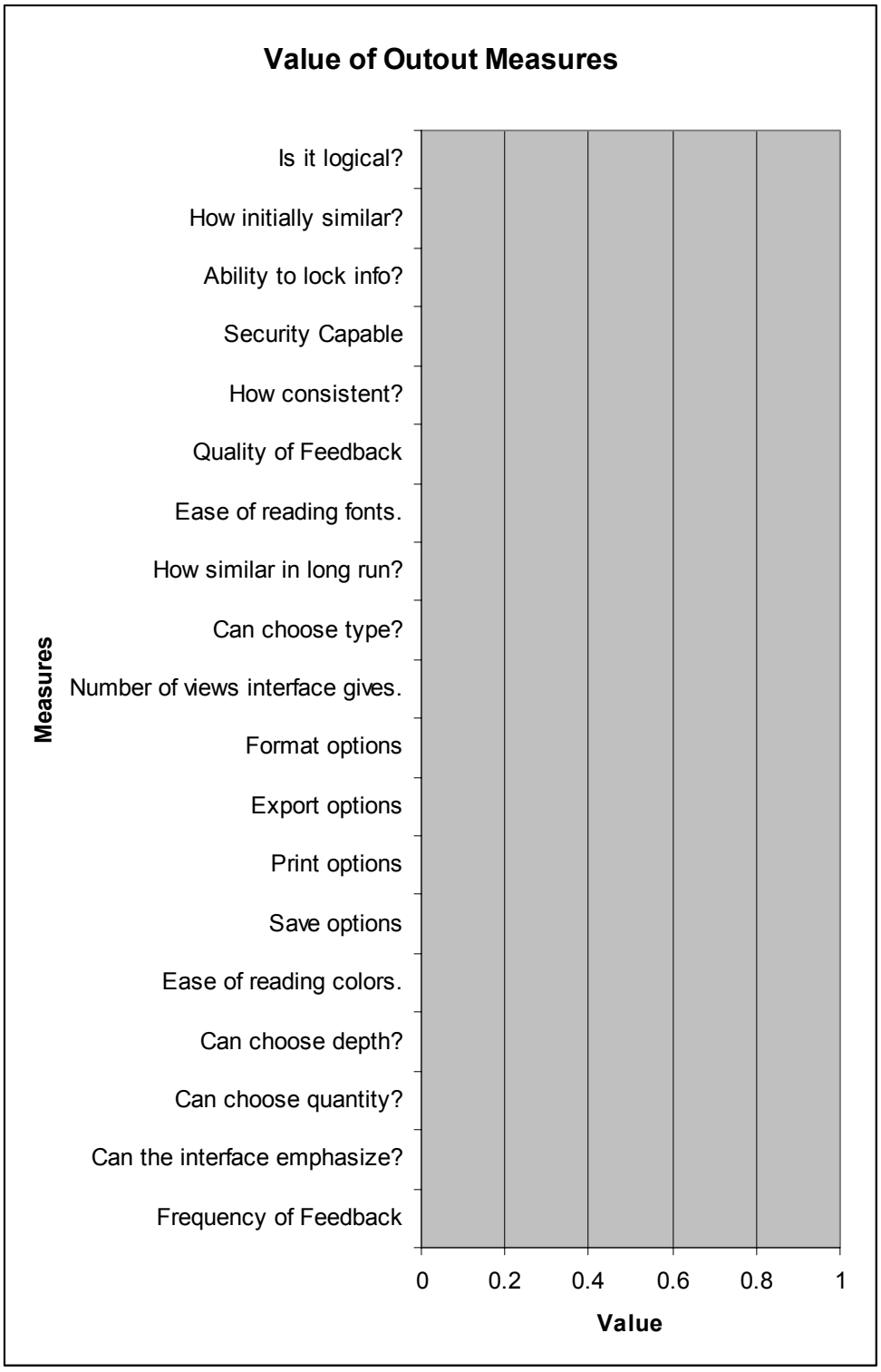

Figure 114: Output Value Gaps

It is apparent from the graph above that the whole Output component has no value and every measure has a complete gap. The reason for this is that the group felt that the software did not have an output component to it and therefore did not have an output component to its interface. For this reason the every measure in the Output component received a score of zero.

Bibliography

Bush, George W. "Terrorism: Threat Assessment Countermeasures and Policy", *Electronic Journal of the Department of State,* Vol 6, No. 3, November 2001

Chambal, Stephen. Class notes, OPER 643, Advanced Decision Analysis. School of Systems and Engineering Management, Air Force Institute of Technology, Wright-Patterson AFB OH, Summer Quarter 2002

Charniak, Eugene. "Bayesian Networks without Tears," *AI Magazine,* 50-63 (Winter 1991)

Director of Central Intelligence. www.intelligence.gov. September, 2002

Fein, Robert A. & Bryan Vossekuil & Gwen A. Holden, "Threat Assessment: An Approach to Prevent Targeted Violence", *National Institute of Justice: Research in Action*, 1-6, July 1995

Head, Allison J. *Design Wise: A guide for evaluating the interface design of information resources.* Medford NJ: Information Today, Inc., 1999

Jurk, David M. *Decision Analysis with Value Focused Thinking as a Methodology To Select Force Protection Initiatives for Evaluation.* MS thesis, AFIT/GEE/ENV/02M-05. School of Systems and Engineering Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2002.

Keeney, Ralph L. *Value-Focused Thinking: A Path to Creative Decisionmaking*. Cambridge MA: Harvard University Press, 1992.

Keeney, Ralph L. "Creativity in Decision Making with Value-Focused Thinking," *Sloan Management Review*, 35: 33-41 (Summer 1994)

Kirkwood, Craig W. *Strategic Decision Making, Multiobjective Decision Analysis with Spreadsheets*. Belmont: Wadsworth Publishing Company, 1997.

Leon, Orfelio G. "Value-Focused Thinking versus Alternative-Focused Thinking: Effect on Generation of Objectives," *Organizational Behavior and Human Decision Processes,* Vol. 80, No. 3:213-227, December, 1999

Macko, Steve. "Military Forces in Bosnia; Intelligence Overload…" *ENN Daily Intelligence Report,* Vol 3 – 102, April 12, 1997

Microsoft Press. *The Windows Interface Guidelines for Software Design.* Redmond WA: Microsoft Press, 1995

Patterson, Emily S. & David D Woods,.. *Aiding the Intelligence Analyst: From Problem Definition To Design Concept Exploration.* Cognitive Systems Engineering Laboratory, Institute for Ergonomics, The Ohio State University, Columbus Oh, March 2001

Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Interference.* San Mateo CA: Morgan Kauffmann, 1988.

Shneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Reading Massachusetts: Addison-Wesley Publishing Company, 1992

Shoviak, Mark J. *Decision Analysis Methodology to Evaluate Integrated Solid Waste Management Alternatives for A Remote Alaskan Air Station*. MS thesis, AFIT/GEE/ENV/01M-20. School of Systems and Engineering Management, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, March 2001.

Simon, H.A., *Evidence and interference for the intelligence analyst, Volume I.* Lanham, MD: University Press of America, 1981

Stubbing, Richard A. & Melvin A. Goodman, "How to Fix US Intelligence," *The Christian Science Monitor,* July 26, 2002

Tufte, Edward R. *Envisioning Information.* Chesire, Connecticut: Graphics Press, 1990

Tufte, Edward R. *The Visual Display of Quantitative Information.* Chesire Connecticut: Graphics Press, 1983

Woods, D.D. & J.C. Watts, "How not to have to navigate through too many displays." *Handbook of Human-Computer Interaction, Second Edition.* North-Holland: Elsevier Science Publishers, B.V., 1997

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
03-25-2002	**Master's Thesis**	Mar 2002 – Mar 2003

4. TITLE AND SUBTITLE

A VALUE FOCUSED THINKING APPROACH TO SOFTWARE INTERFACE IN A COMPLEX ANALYTICAL DOMAIN

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

McGee, Christopher, M., 2Lt, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)

Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 P Street, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT/GOR/ENS/03-16

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AFRL
Attn: Janet Miller
2255 H Street
WPAFB OH 45433-7765
DSN: 786-4847
e-mail: Janet.Miller.3@wpafb.af.mil

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The intelligence community is faced with an extensive amount of data. Software programs are being developed to examine this issue of data overload and to develop solutions. The responsibility of making the final software decision lies on the analyst, therefore, the interface is the key to linking the intelligence data to the processing and results. If the interface is difficult and complex, the software will be less likely to be used. A methodology must be created which can objectively evaluate the effectiveness of the interface. This methodology will also measure the improvements in the interface's effectiveness that result when various changes are made to the original software interface.

Value focused thinking (VFT) is a proven methodology that can be applied to this problem. VFT provides an objective methodology to identify the values or an organization. Its hierarchical structure is well suited for handling multi-objective problems, such as identifying the values of software interfaces. The values can be measured and put to a common scale, allowing their contribution to the overall objective to be evaluated. By assigning quantifiable measurements to the components, the multi-objective goal can be evaluated and insight can be provided to the decision makers involved with the intelligence software.

VFT was applied to determine what is valued in software's interface to members of the intelligence community. With these values identified, a software that is under development was evaluated against the hierarchy. This provided insight into where improvements could be made to the interface that would provide the greatest benefit. The VFT process also allows for the decision maker to continually reevaluate the software against the hierarchy, enabling continual improvement on the interface while maintaining the values of the intelligence community.

15. SUBJECT TERMS

Operations Research, Decision Analysis, Value Focused Thinking, Software Interface

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Stephen P. Chambal, Capt, USAF (ENS)
U	U	U	UU	183	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, ext 4314; e-mail: Stephen.Chambal@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18