# *Vizability: Visualizing Usability Evaluation Data Based on the User Action Framework*

*By*

***Christopher D. Catanzaro***

*Thesis submitted to the faculty of the Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of*

***Master of Science***

*In*

***Computer Science and Applications***

*April 28, 2005*

*Blacksburg, Virginia*

**Advisory Committee:**

Dr. Chris North (Chair)

Dr. H. Rex Hartson

Dr. Manuel A. Pérez-Quiñones

**Keywords:** Visualization, Usability Data Analysis, User Action Framework

* * *

# *Vizability: Visualizing Usability Evaluation Data Based on the User Action Framework*

*By*

*Christopher D. Catanzaro*

## Abstract

Organizations have recognized usability engineering as a needed step in the development process to ensure the success of any product. As is the case in all competitive settings areas for improvement are scouted and always welcomed. In the case of usability engineering a lot of time, money, equipment, and other resources are spent to gather usability data to identify and resolve usability problems in order to improve their product. The usability data gained from the expenditure of resources is often only applied to the development effort at hand and not reused across projects and across different development groups within the organization. More over, the usability data are often used at a level that forces the organization to only apply the data to that specific development effort. However, if usability data can be abstracted from the specific development effort and analyzed in relation to the process that created and identified the data; the data can then be used and applied over multiple development efforts. The User Action Framework (UAF) is a hierarchical framework of usability concepts that ensures consistency through completeness and precision. The UAF by its nature classifies usability problems at a high level. This high level classification affords usability engineers to not only apply the knowledge gained to the current development effort but to apply the knowledge across multiple development efforts. This author presents a mechanism and a process to allow usability engineers to find insights in their usability data to identify both strengths and weaknesses in their process. In return usability practitioners and companies can increase their return on investment by extending the usefulness of usability data over multiple development efforts.

# Dedication

In memory of my grandfather, Harvey Sturgeon, whose strong character, humor, and love for living remains in those who have been fortunate enough to have been a part of his life.

## Acknowledgements

Now at the end of one road about to begin the next I would like to take the time to thank those who have helped along the way.

First, I would like to thank my Family; Mom a constant light and motivation in my life; Dad your words are inspiring, your daily actions even more; and Jean, you are my role model in my life and I am so proud of you.

Coach Gines, thank you for the confidence you instilled in me through the years and always being there for needed advice.

Javier and Libby, thanks for everything. Always helping me and supporting no matter the circumstance.

Pardha Pyla and Jon Howarth, thanks for the constant support and help in the research of this thesis; always there to answer my questions. Jihane Najdi and Amine Chigani, thanks for your time and help in the development of the visualization tool.

And finally my committee, Dr. North thank you for the input and time you have given me over the years; Dr. Perez and Dr. Hartson, thank you for your advice and direction.

# Table of Contents

# Table of Figures

# List of Tables

# 1 Introduction & Problem Statement

In recent times, usability is considered to be a highly important component in the software development process (Hartson & Hix, 1993). We know that good usability practices can be the difference between success and failure of a product. Since usability is an integral component in deciding whether a product fails or succeeds, companies put in a lot of time and resources in order produce a successful product (Bias, Mayhew, & Upmanyu, 2003 ).

Although usability practitioners are trained in finding usability problems, many times usability problems are not documented properly. Keenan states that "the success of a usability engineering program is limited by the quality of the problem descriptions" (Keenan, 1996). If usability problems are not documented reliably and consistently they are likely to be misunderstood, addressed improperly, or not addressed at all. Usability problems that are not documented properly have an inherent danger of confusion while trying to interpret problem descriptions. This often leads to guess work on the part of the developer or even to fixing a problem that does not exist.

The result of such arbitrary decision making wastes valuable resources. Imagine an evaluation of a product or system resulting in a large repository of usability data. The value the organization gets in return on its investment (ROI) in conducting usability evaluations and in collecting usability problems is primarily in the interpretation of that data. Data on its own do not entail or afford insights. In this context, we use the term data as bits and bytes that have no meaning without interpretation. It takes work to transform the data into information. For example, a sequence of numbers in a column is just data. However, the same numbers when shown as a graph with proper labels becomes information. (Ackoff, 1989).

The User Action Framework (UAF) provides a structure by which usability data can be transformed into usability information; information that can be efficiently used to identify

and resolve usability problems. That is, the organization's resources are utilized in a manner that results in the greatest cost-benefit ratio. The usability data analyst is able to perform this transformation by using the UAF to standardize the problem reporting process (Hartson et al., 1999). Usability data classified and documented using the UAF transforms collected data into information with more accuracy (Sridharan, 2001) and less information loss.

The UAF contributes to the usability domain in numerous ways on it own. This thesis documents a way to extend the current functionality of the UAF tools to allow users to apply the usability information gained from an evaluation not only to the development effort at hand but also to the entire process.

Currently most information collected via the UAF or any collection method is only used for the specific development effort at hand and does not feed into the big picture of an organization's overall development process or strategy toward augmenting usability best practices. However, the UAF, in its current form does not lend itself to perform usability information analysis at higher level. It became apparent that the field was in need of a visualization tool that can augment the UAF so that entities can increase the returns of their usability efforts by providing a high level interpretation of the usability information.

## 1.1 Motivation

Information gained in usability evaluations is not applied to the improvement of the process or conveyed in a meaningful form to the stakeholders involved in the process. Usability information gained is only applied to the betterment of the product, and in that particular product release alone. The motivation for this thesis is to enable usability practitioners to find patterns and trends in usability problems so they may improve the overall workings of their development process.

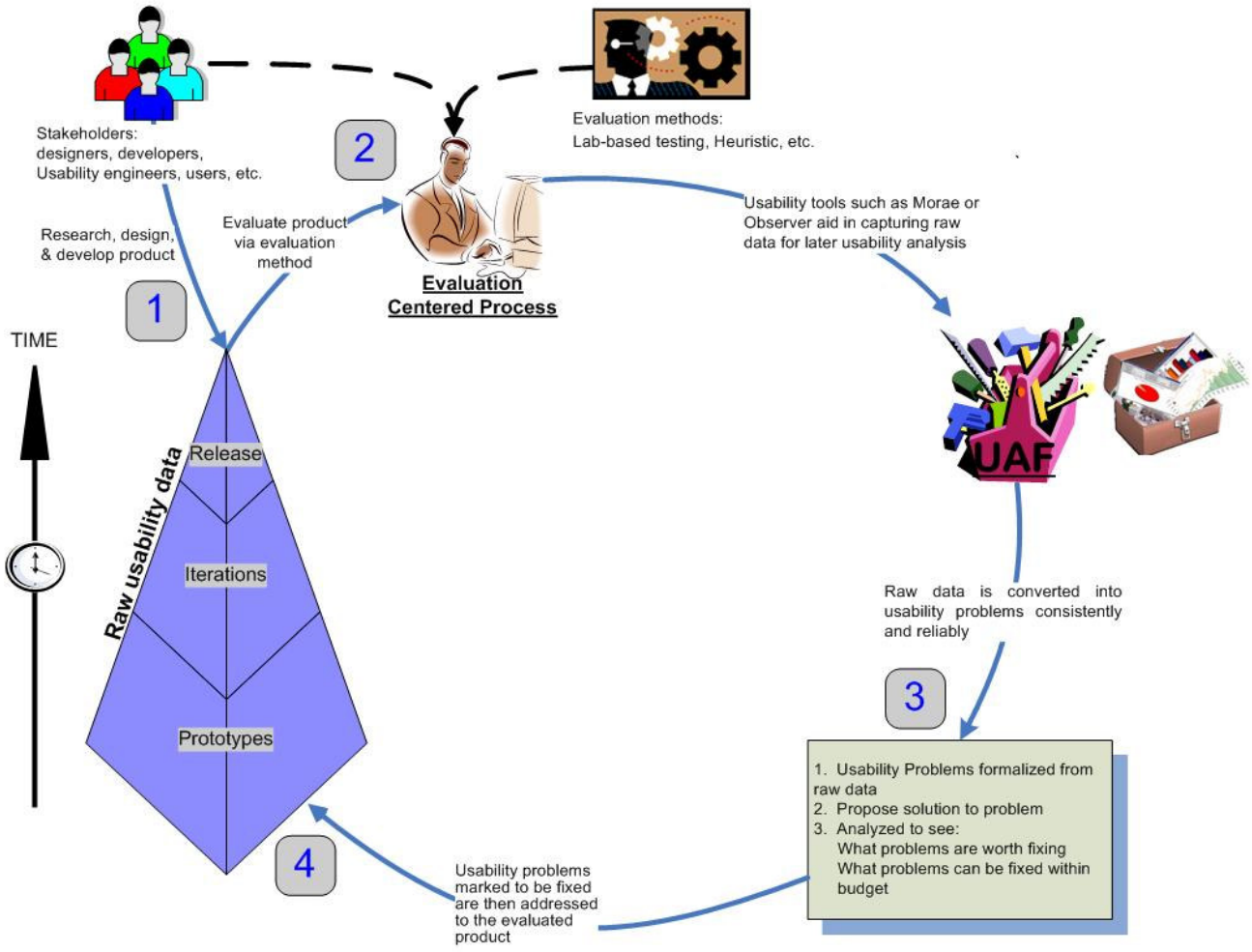



**Figure 1: Generalized product development life cycle**

A lot of time, energy, and resources are dedicated to develop a usable product. When executed correctly those efforts increase the value of the particular product and yet the utility for reuse of that data is often overlooked or undiscovered. In most cases usability assessment efforts are only applied to the specific product being evaluated and filed away from applicability to the development efforts of other products. Figure 1 illustrates the general product life cycle for the development of interactive systems. In stage 1, stakeholders gather requirements and develop prototypes based on those requirements. Stage 2 usability engineers decide upon an appropriate evaluation method depending on the current type of prototype and phase of development. This evaluation often results in

substantial amounts of raw usability data. These data are usually collected using tools such as Morae (TechSmith, 1995). The usability data is then transformed into information by identifying problems and classifying those problems in the UAF structure using DCART. DCART is a tool developed for the UAF to help usability practitioners consistently and reliably classify usability problems. These usability problems could be a direct result of any number of factors. Three such factors are programming language limitations, insufficient user requirements analysis, inadequately trained developers (Hartson, 2004). In stage 3, the documented usability problems are analyzed, solutions are proposed, problem-solution pairs are prioritized, and finally the usability problems that can be fixed in the available budget are short listed. Then in stage 4, the short list of usability problems are passed to the developers with the proposed solution so each usability problem may be corrected in the next prototype version. This cycle continues till the product reaches a level of quality that the stakeholders feel is adequate. This level of acceptability is often documented as usability specifications in the earlier part of the development process.

In this development practice, an area that is often ignored and left unevaluated and uniterated is the overall development process itself. The development process is not explicitly evaluated nor modified to correct areas of deficiencies. For instance, if we look at a sample development team in a particular software company, the team has, on an average, one project per year. It has been found that in the last three projects, the final version delivered to the customer always ran into major problems during the acceptance testing phase. The developed product did not reflect the user's requirements correctly and the team had to change or tweak the product after release. This has consistently resulted in extra costs for the software company.

Often times, a development team has a very low probability of identifying deficiencies in the development process because they do not have a way to interpret trends over the entire process over time. If they did they would have identified their lack of proficiency in elicitation and verification and validation stages of the requirements phase.

The team's current development process might be considered effective in that each round identifies usability problems and important problems from that list are corrected. However, the process is not effective in eliminating recurring usability problems and bad usability practices.

This above scenario depicts only one situation where a tool that enables a company to understand the overall usability problem set, patterns, and trends could curb the cost of the development effort over years of future development efforts. Another important advantage of having this overall analysis mechanism is that one can perform rapid data-mining to identify clusters of problems, and identify areas of strengths and weaknesses.

The UAF categorizes usability problems consistently and reliably at a higher level. The resulting categorization of usability problems can provide critical information not only about the product being evaluated but also critical usability information about the development process as a whole. Therefore, it is the motivation of this author to find a mechanism that can provide a high level analysis and interpretation for usability problem datasets created using the UAF.

## 1.2 Goals

The goals of this thesis are as follows:

- Identify the requirements for a mechanism that can provide the functionality for high level analysis and interpretation of usability data classified using the User Action Framework;
- Develop a tool that implements such a mechanism;
- Perform a formative evaluation of the tool; and
- Identify insights, usability issues, and provide recommendations for improving the tool.

## 1.3 Scope of the work

The scope of the work in this thesis is to design and implement a high-fidelity visualization tool to visualize usability data based on the User Action Framework and to evaluate the tool via a formative evaluation to find areas for improvement.

This visualization tool will integrate into activity "B" of the usability data management cycle as shown in Figure 2 (Howarth, 2004).

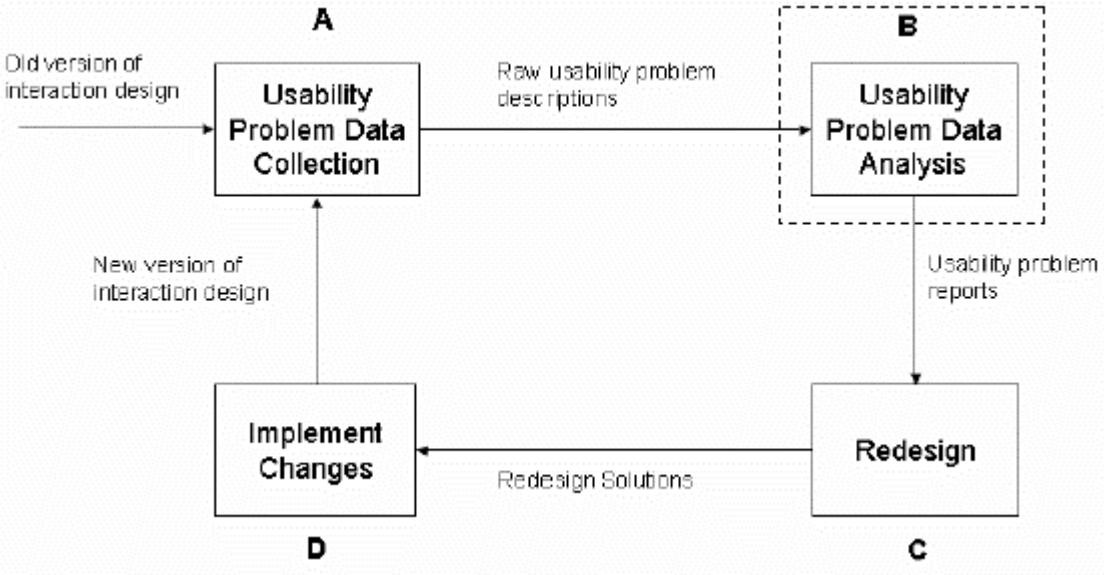



**Figure 2: Usability data management cycle**

# 2 Related work

This chapter will be divided into three parts. The first section will give some background information about the User Action Framework, the tools developed to support it, and how this visualization complements the existing tools. The second section will list some other usability software tools to aid the usability process. In this section we will explain the differences between current usability software usability tools and the UAF software tools. The last section, Hierarchical visualization techniques, will summarize existing visualizations to view hierarchical data structures as is needed for the User Action Framework. We review these visualizations to understand existing methods and how we can use those techniques to design a UAF visualization tool.

## 2.1 User Action Framework

The User Action Framework (UAF) is a conceptual framework of usability concepts (Hartson et al., 1999). It is based on Norman's stages of interaction (Norman, 1988). The framework can be described as a tree structure consisting of approximately 400 nodes. Each node is mapped to a usability concept.

The top level of the UAF is composed of six nodes: Planning, Translation, Assessment, Physical Actions, Outcome and System Functionality, and Overall. Planning is the cognitive process the user has when beginning a task. Translation is how the user must translate his or her intentions (from planning) into interaction steps. Physical actions are the physical actions the user performs in effort to complete the task. The Outcome and System Functionality node deals with the internals of the system(s). It is used to classify errors that occur due to malfunctioning or non-existent system functionality. Assessment is the final stage of interaction and deals with the user's evaluation of the outcome of the interaction; was the task completed successfully or not. The final the top level node is Overall. It is associated with errors that occur across the whole interaction cycle. An example of a usability problem that would be classified to the Overall node could be

software portability; if the system is designed to be portable to multiple platforms and is not, this would be an Overall usability problem.

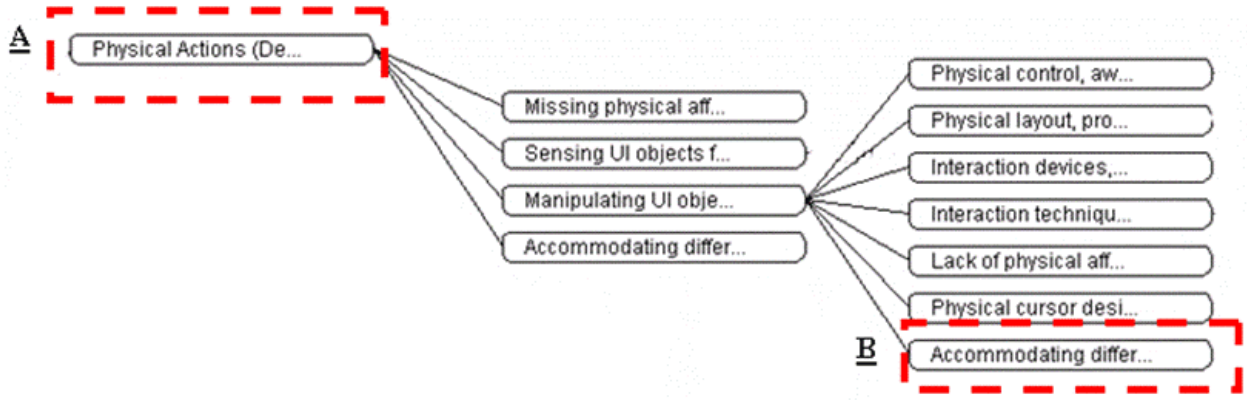



**Figure 3: In this case box 'A' is a more general usability concept than box 'B' therefore usability concepts classified in 'B' are more specific. This illustrates the nodes nested deeper in the tree represent more specific usability concepts than ancestral node.**

These top level usability concepts are then broken down into more detailed usability concepts. This corresponds to the deeper you are in the tree structure the more specific and detailed the usability concept.

The structure by its nature provides a way to group usability problems. The usability concepts in their respective location in the hierarchy can be used to group usability problems consistently and reliably by usability concept. In Figure 3, if usability problems were classified in the 3rd level (same level as 'B') we could group all of the usability problems under the higher level usability concept, "Physical Actions". This would give the practitioner a high level understanding of the problem classifications. But there is another method to categorize usability problems. This can be done by keywords. Keywords are words that are associated with specific usability concepts or nodes distributed across the UAF structure. For example the keyword "labeling" can be associated with nodes in "Assessment" and "Translation", thereby linking problems across the tree by a common usability issue. Multiple keywords may be associated to a single node.

The UAF structure is complex and must be understood in order to benefit from the framework. To help users understand, classify usability problems, and enter usability problem data DCART (Data Capture and Analysis Reporting Tool) has been developed. DCART provides functionality to classify usability problem, document each usability problem, and store those problems in a database. DCART also provides the functionality to view problems in a list format.

## 2.2 Visualization in the usability domain

Visualization has been used in the usability analysis domain but not at the level we propose. Previous efforts have focused on low-level data. Our approach, however, visualizes information that is essentially the result of the abstraction of low-level data.

Previous work with low-level data has focused on visualizing events such as keystrokes and mouse clicks that are performed by users as they interact with an interface (Wesson & Greunen, 2002). Such events certainly have the potential to be useful in evaluating an interface, but they may be difficult to interpret correctly. Ivory discusses the three common activities of usability evaluation: capture, analysis, and critique (Ivory & Hearst, 2001 ). Data capture may be partially or completely automated, but accurate analysis and critique often require a thorough review of the data by a usability engineer. However, the volume of data produced by low-level events, particularly when capture is automated, can be overwhelming. Visualization tools such as those presented in (Gray, Badre, & Guzdial, 1996 ), (Holm, Priglinger, & Stauder, 2002), (Wesson & Greunen, 2002), (Guzdial et al., 1994) are necessary because they help order events, associate user-initiated events with system events, and highlight patterns.

Our work focuses on the visualization of usability problem diagnoses made by usability engineers. A diagnosis involves isolating a distinct usability problem from observational data and documenting that problem according to a predefined method or system. The UAF tree structure provides a way of systematizing the diagnosis process. A usability problem diagnosis represents a higher level of understanding of a problem and is not as prone to the incorrect inferences or contextualization that may complicate usability studies focusing on low-level data (Guzdial et al., 1994; Gray, Badre, & Guzdial, 1996 ).

Diagnoses, particularly when performed by usability engineers using a tree structure, promote consistency. In addition, because they are at a higher level, usability problem diagnoses encourage knowledge sharing and reuse.

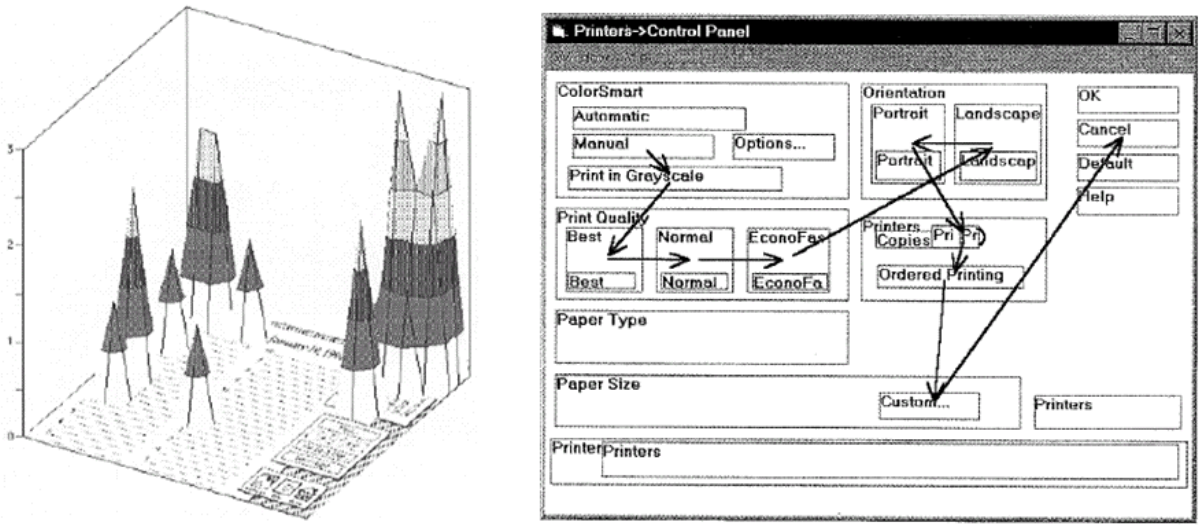



**Figure 4: ON THE RIGHT: Mouse clicks superimposed on a graphical user interface (Guzdial, Santos et al. 1994) illustrate the quantity of mouse clicks. ON THE LEFT: user events arrows showing order in which controls were used (Gray, Badre, & Guzdial, 1996)**

Some software systems have been developed to help usability practitioners collect and analyze usability data. In these systems the focus differs from the UAF in that these tools focus on the recording of the events of usability evaluations; whereas, the UAF tools focus on recording and documenting the actual usability problem. The software systems listed below capture low level data such as mouse clicks or keystrokes and some systems provide screen capture and audio recording capability. This usability data is then later reviewed and analyzed in order to formalize specific usability problems. Listed below are some software packages used to collect and analyze usability data. The list below is reproduced with the written consent of Todd Zazelenchuk (Zazelenchuk, 2003). The list has been adapted for the purposes of this thesis.

*OvoStudios: URL: http://ovostudios.com/*

*OVO Logger 4.1 comes in three different flavours (freeware, a la carte, fully featured) and provides extensive logging options for both notes and "tapeless"*

*video as well as powerful bookmarking and reporting features designed to optimize the analysis and reporting phase of a study. While the fully featured version may be more than many researchers need or have a budget for, the freeware version may be just the right ticket. OVO Logger is a Windows-only product.*

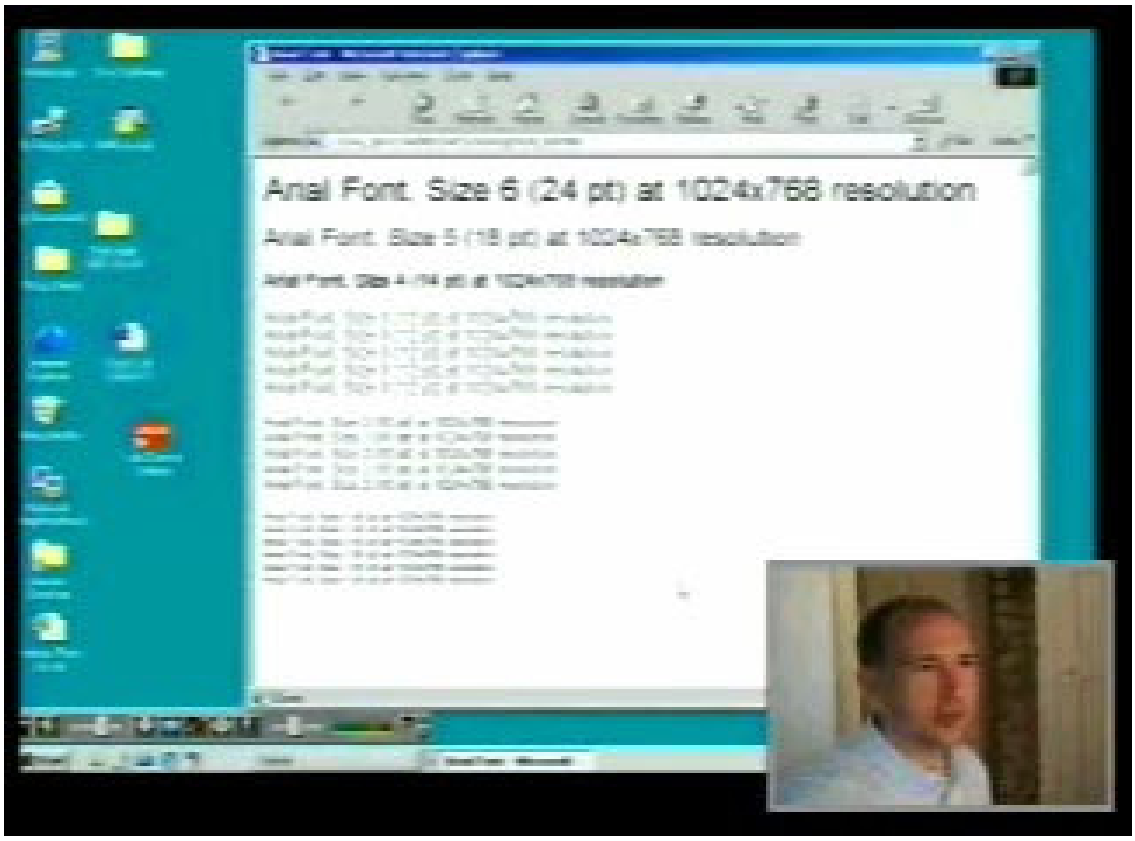



**Figure 5: Screen capture of a user evaluation using OVO Logger 4.1. (http://ovostudios.com)**

*Techsmith: URL: http://www.techsmith.com/*

Morae *is touted as "the only fully integrated, all-digital solution for analyzing human-computer interaction". This application takes advantage of digital video technology to allow researchers to capture, store, locate, and edit their video data from a usability study. Priced at $1298.00 USD, this application may be an attractively priced solution for managing video data. In its current form, the*

*system focuses entirely on the video data and does not include any note taking or reporting functionality. Morae is Windows only.*

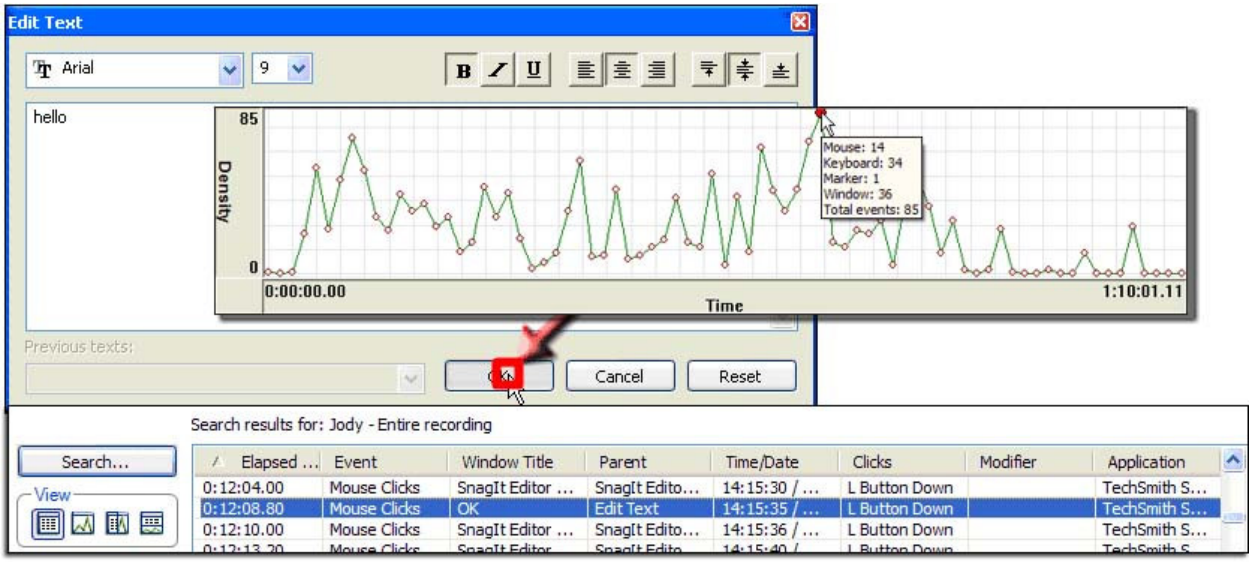



**Figure 6: Morae can create a video of screen captures and can synchronize the video with system events and audio**

*Noldus: URL: http://www.noldus.com/*

*The Noldus Observer is "a professional system for the collection, analysis, presentation and management of observational data." This application is able to accommodate data entry directly from a computer, a handheld device, or a video recorder, and offers extensive coding and analysis options to the researcher. While the Observer is packed with powerful features that may be needed for extensive qualitative research, it may be overkill for many usability researchers. Observer is Windows only.*

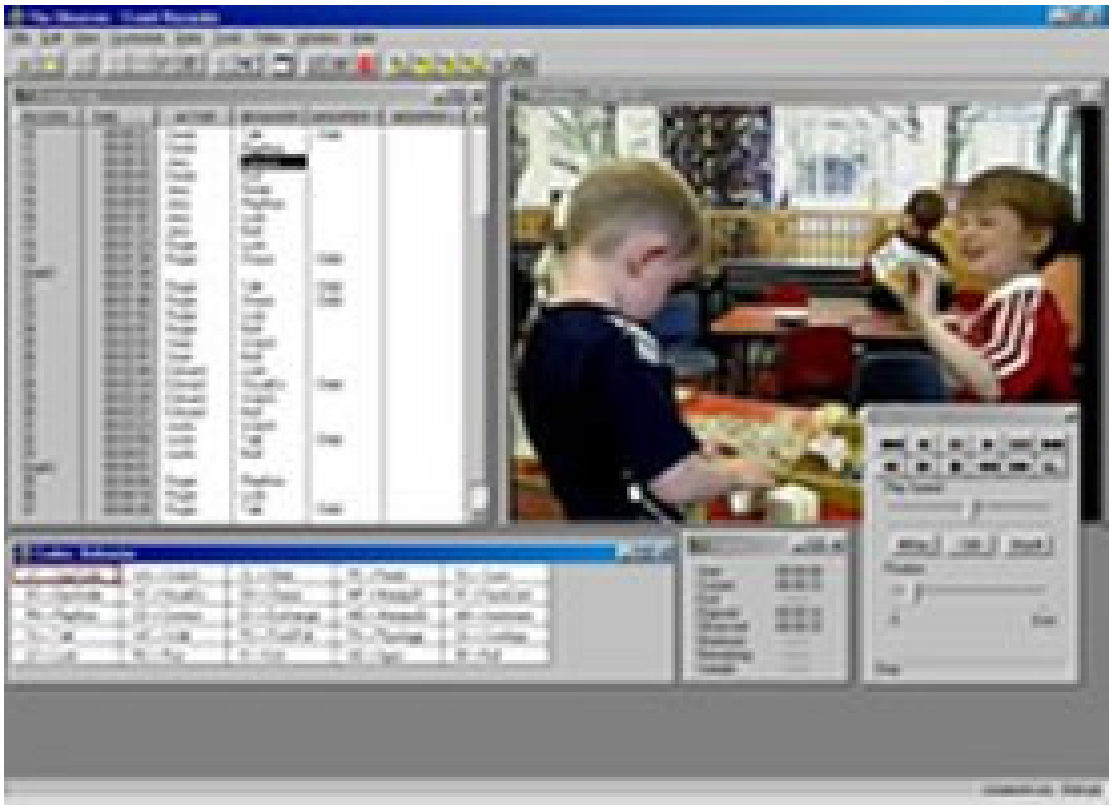


**Figure 7: The Observer by Noldus allows you to record video and synchronizes different system events as well any notes or comments made the evaluator**

*Bit Debris: URL: http://www.bitdebris.com/*

*The Usability Activity Log v2.3 "offers an effective means to easily and unobtrusively document observational data and task performance". This application can be synchronized with existing video equipment so that recorded observations are directly 'linked' to the accompanying video data for easy access by the researcher. The product is a Windows application (NT recommended) and costs $300.00 USD per license.*

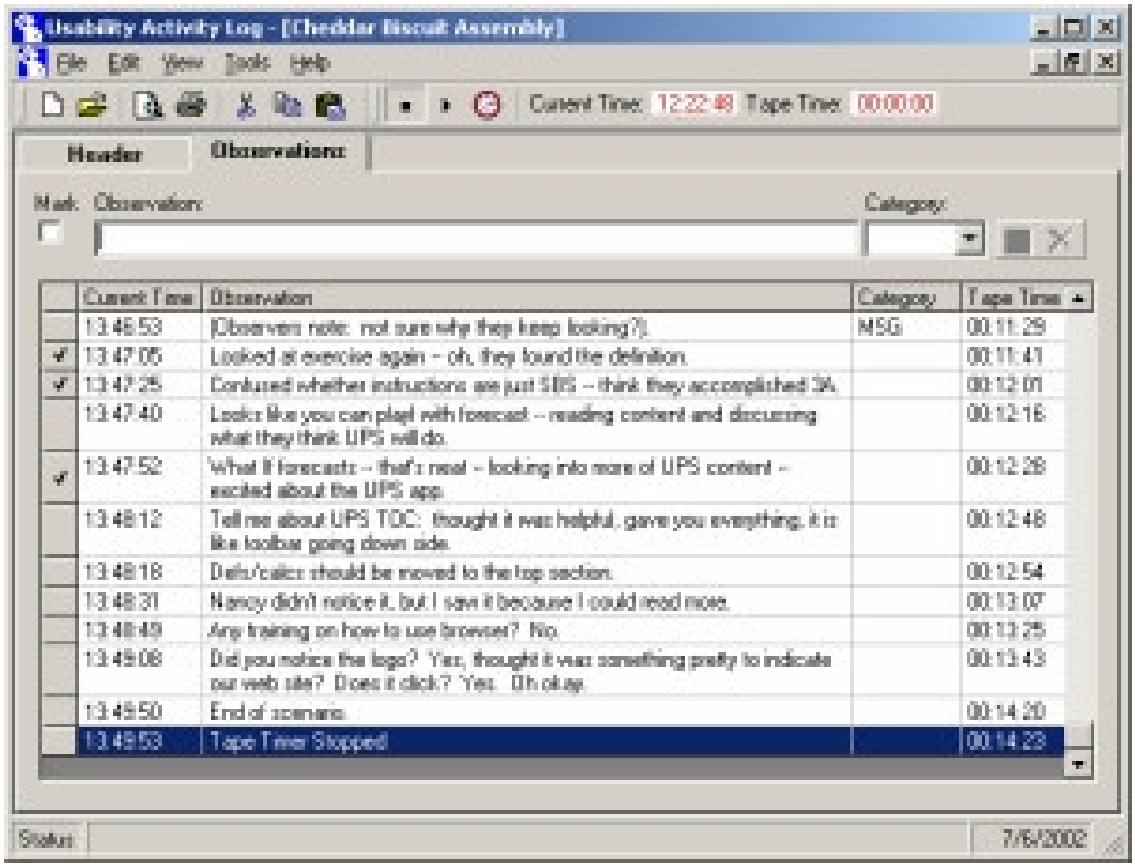



**Figure 8: Bit Debris enables a user to import existing video and link notes and comments to different segments of the video**

*Usability Systems/Alucid: URL: http://www.usabilitysystems.com/*

*UsabilityWare 4.0 is "a single program that can be relied upon as a beginning-to-end tool for all of your data collection, analysis, and final deliverables." This program allows you to enter your recruitment and scheduling details prior to a study, record your observations during the study, analyze the results, and build a report based on the data. The product is a Windows application and costs $4500.00 USD per license.*

Usability Data Logger (http://www.userfocus.co.uk/resources/datalogger.html): The Usability Data Logger developed by Todd Zazelenchuk is based is based on Microsoft Excel. It provides a consistent system to document usability data. It is composed of 20

different worksheets. Each worksheet has its own purpose and provides a template that evaluators can use to enter data specific to the evaluation; tasks, tasks descriptions, pre/post questions, administrative items, and a satisfaction survey. Some of these data are then used to generate the charts shown below.

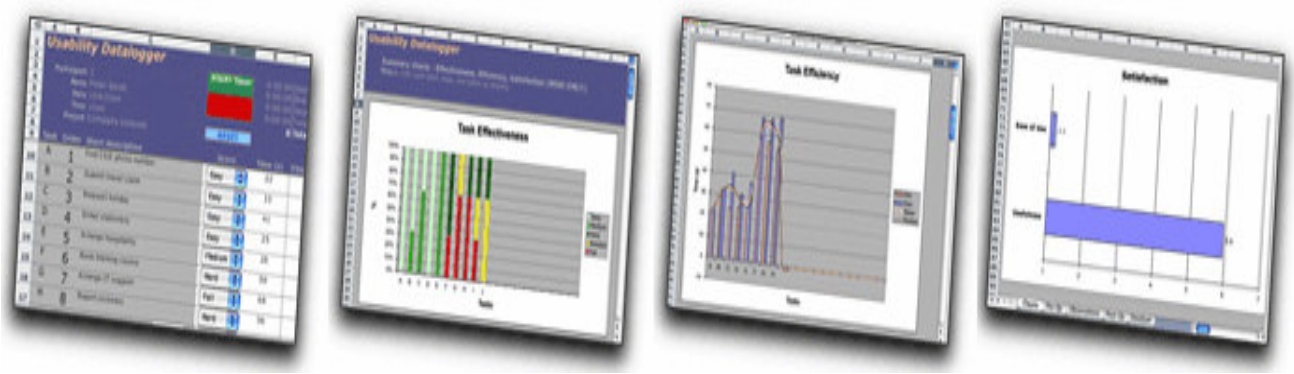


**Figure 9: The Usability Data Logger uses MS Excel to record usability data in a pre--defined template that can automatically generate simple charts based on time and satisfaction ratings**

## 2.3 Hierarchical visualization techniques

In this section we will discuss some techniques that have been created to visualize hierarchical data. We will summarize each tool and make claims as it relates to the UAF visualization. "Claims", as defined by Rosson, "can be seen as examples of tradeoffs, where the tradeoffs are reflected in the claim's upside and downside" (Rosson & Carrol, 2002).

### 2.3.1 Sunburst Visualization

This tool visualizes tree structures in a radial form (Stasko & Zhang, 2000). A visualization example of the Sunburst tool is shown in Figure 10.

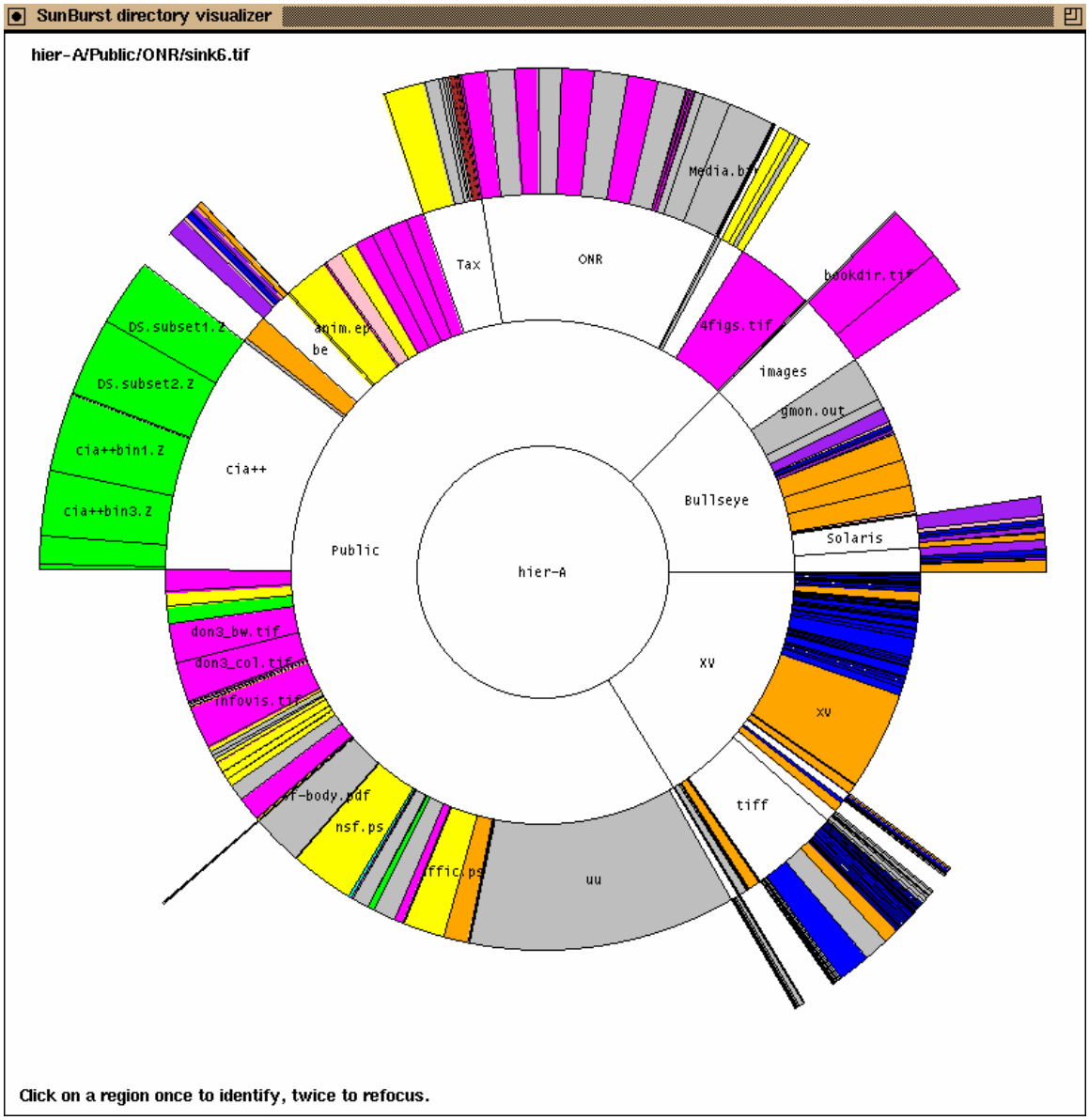



**Figure 10: The Sunburst places the root node in the center and places child nodes in concentric circles around the root node.**

This tool uses a circular hierarchical display where the root is at the center and the children are in the next bigger concentric circles. This tool is effective in showing the attributes like the size and the type of a node, but performs poorly when showing a structure compared to the traditional edge trees. As our application needs to show the hierarchy of usability data in as much detail as possible, this visualization method does

not prove to be helpful to us. Moreover, this tool does not perform very well for comparing two hierarchies at the same time.

| Claims Analysis for the Sunburst visualization as it relates to the User Action Framework | |
|---|---|
| + | Uses screen space efficiently since parent-child nodes are rendered adjacent to each other no screen space is wasted connecting parent-child nodes. Enables user to see the entire structure at one time. |
| - | Difficult to identify specific node attributes whose slices are small. For example the node name, or type. Users have to mouse over the node or slice in order to view the name. |
| - | Doesn't fit the mental model of a conventional hierarchical tree structure putting a higher cognitive load on the users of the system. Users will have to orient themselves to the new presentation of the data. |

### 2.3.2 Pruning Trees

Users often must browse hierarchies with thousands of nodes in search of those that best match their information needs. The PDQ Tree-browser (**P**runing with **D**ynamic **Q**ueries) visualization tool was specified, designed and developed for this purpose (Kumar, Plaisant, & Shneiderman, 1997).

This tool presents trees in two tightly coupled views, one a detailed view and the other an overview. Users can use dynamic queries, a method for rapidly filtering data, to filter nodes at each level of the tree. Dynamic queries allow the user to see immediate results when a query is made; giving the user instant visual feedback as to how the query filtered the data. The dynamic query panels are user-customizable. Sub-trees of unselected nodes are pruned out, leading to compact views of relevant nodes. Usability testing of the PDQ Tree-browser, done with 8 subjects, helped assess strengths and identify possible improvements. The PDQ Tree-browser was used in Network Management (600 nodes) and University Finder (1100 nodes) applications. A controlled experiment, with 24 subjects, showed that pruning significantly improved performance speed and subjective user satisfaction.

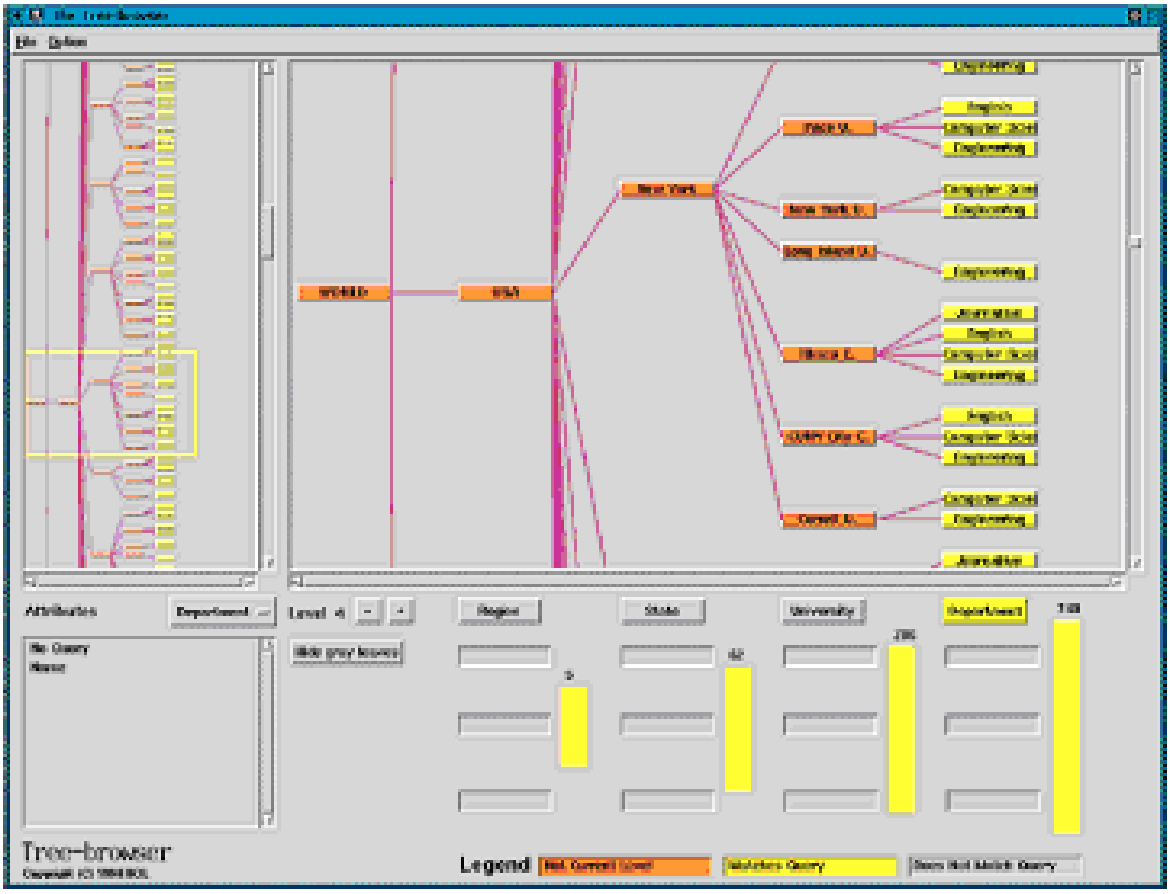



**Figure 11: PDQ Tree**

| Claims Analysis for the PDQ visualization tree as it relates to the User Action Framework | |
|---|---|
| + | This tree allows users to focus on only nodes of interest. Screen-space is not cluttered with nodes that do not match a specific criterion. Users may query specific attributes of nodes. The resulting tree only displays those nodes that fit the users' specified criteria. |
| - | Users may lose context with the complete tree structure when nodes are added and removed based on the user's query. The tree is updated dynamically, repositioning the each node of the tree as nodes are added and removed. This makes it difficult for users to understand the changes made to the tree. |
| - | The tree doesn't allow you to compare nodes that match the query vs. nodes that do not match. User's may want to view nodes of a specific type but still have the ability to view other nodes to see how their query compares to the overall tree structure. |

### 2.3.3 StarTree

While developing applications to navigate large collections of information, one of the biggest obstacles is displaying thousands of objects making up the collections. If the developers decide to use an external tool, they must choose a robust, time-tested tool that is also easy to integrate. The Star Tree™ SDK is a good option in such cases (Lamping & Rao, 1994).

StarTree technology is a proven technique for navigating and visualizing large hierarchies of information developed by Inxight. The StarTree SDKs offer well-designed and tested APIs for programming in either Java or ActiveX environments. And they give access to the full range of functionality of Inxight's underlying, patented StarTree technology for navigating and visualizing large hierarchies of information. Thus, the SDKs are the fastest way to integrate the most powerful navigational tools available today into software applications. StarTree is basically a java-based tool that allows displaying objects as nodes of a tree and allowing the user to browse these objects using the fish-eye technique. It emphasizes the fact that information can be better represented in the form of visualization, instead of the stereotype way of representing it in the form of tables.

| Claims Analysis for the StarTree visualization as it relates to the User Action Framework | |
|---|---|
| + | The StarTee because of its layout can be navigated quickly to find specific nodes. |
| + | The StarTree uses a dynamic layout that adjusts node positions based on the screen space available. This allows the user all the nodes in the structure at one time |
| - | The layout of the tree, because it intends to get the most use out of the screen space available often occludes nodes so they are not readable to the user. |
| - | The tree structure adjusts as the user navigates the tree. This allows the user to navigate deeper in the tree as needed. However, this is a problem when users would like to compare attributes from different sections of the tree. |

### 2.3.4 Tree-Maps

Tree-Maps are a space-filling visualization for hierarchical structures that are extremely effective in showing attributes of leaf nodes by size and color-coding (Shneiderman, 1992 ). Tree-Maps enable users to compare sizes of nodes and of sub-trees, and are especially strong in spotting unusual patterns.

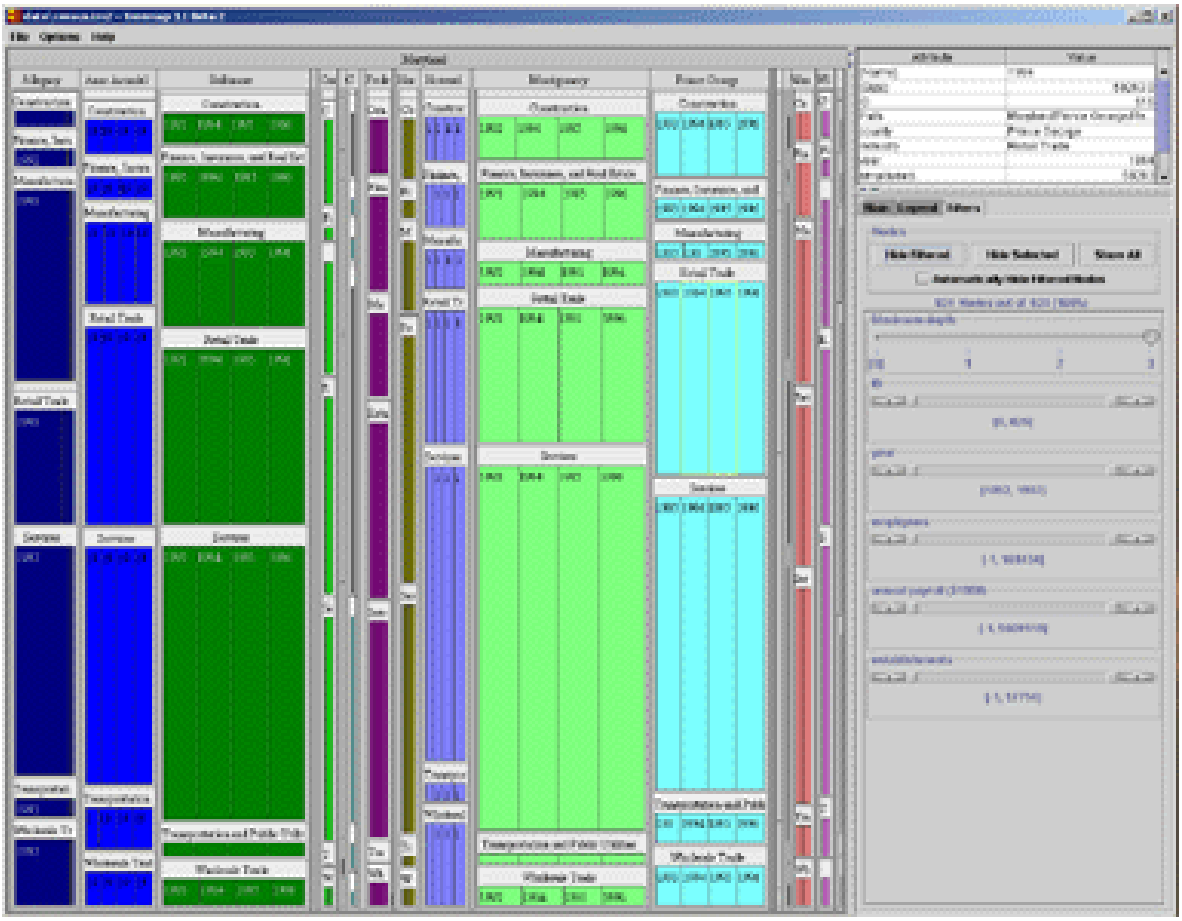


**Figure 12: Tree-Maps**

| Claims Analysis for the Tree-Maps visualization as it relates to the User Action Framework | |
|---|---|
| + | Tree-Maps use 100% of the screen space to display the entire tree structure. Tree-Maps can display in the order of 1000s of nodes. This is an advantage for structures that could contain 1000's of nodes. This allows the user to see the entire structure on the screen at one time. |
| - | Hierarchical relationships are not clear to the user. Users can view children when contained in a "larger" parent however, the relationship the parent has to other nodes is not as obvious. It is very important to be able to identify parent-child relationships in the UAF structure. For this reason Tree-Maps are not the best choice. |
| - | Because the hierarchy is not clear navigation is difficult when trying to find specific nodes. |

## 2.3.5 SpaceTree

Plaisant gave conventional node link trees a new look with the SpaceTree (Plaisant, Grosjean, & Bederson, 2002). The SpaceTree is created for trees with a large number of

nodes. The tree is able to represent the entire tree structure by adding a triangle icon to represent the sub-tree of that particular node. The shading is proportional to the total number of nodes in the sub-tree. The height of the triangle is proportional to the depth of the tree and the base is proportional to the average width of the sub-tree. Using this technique the user can quickly identify nodes not shown by looking at the triangle icon.

The SpaceTree has an intelligent layout feature that calculates how much screen space is available and then displays as many nodes as will fit in the space available. This calculation is done on the fly dependent upon the user's selection. The SpaceTree is also equipped with layout options such as displaying the tree horizontally or vertically.

The SpaceTree has many features that we need in our visualization tool. It uses screen space wisely as needed for a large number of nodes. It also does not contain overlapping nodes making node labels easy to read and it is able to represent the entire tree with an iconic representation of the sub-trees.

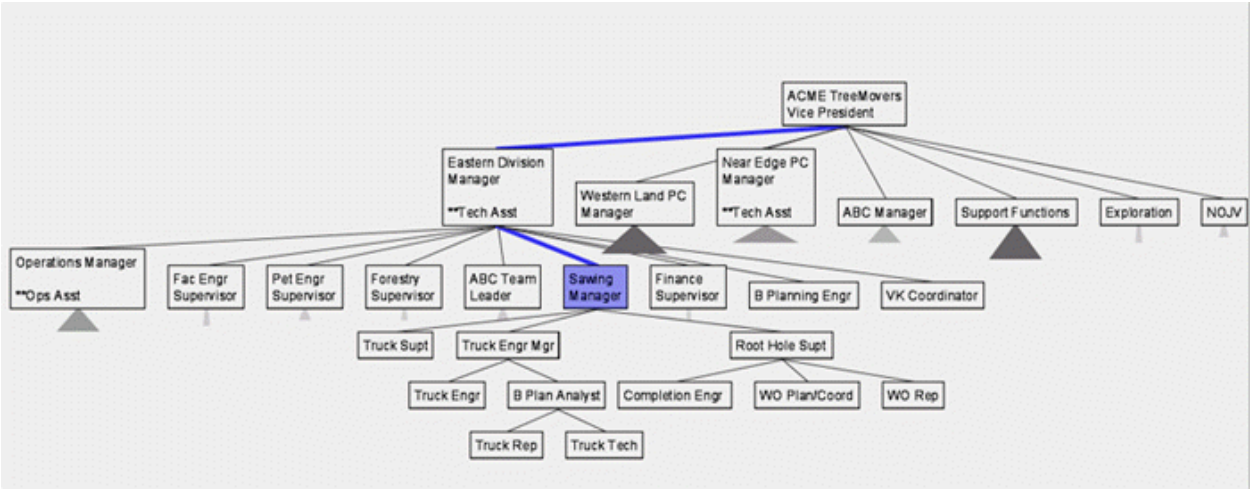



**Figure 13: SpaceTree**

| Claims Analysis for the PDQ visualization tree as it relates to the User Action Framework | |
|---|---|
| + | The SpaceTree can represent the entire tree structure through representative sub-tree icons. Although the all nodes are not visible on an individual bases the sub-tree's make-up is shown so users can identify the attributes of nodes not shown. |
| + | The SpaceTree displays parent-child relationships by a thin line. Making parent- |

| | |
|---|---|
| | child relationships easy to identify |
| + | The nodes are displayed so that they adjust to their node name. This allows users to the ability to read each node label rather than having to mouse over node to view the tool tip. |
| - | The can only display a limited amount of nodes at one time (approximately 80-100 nodes depending on size). However, for the purposes of the UAF displaying the entire tree structure is not as important as being able to view the collective sub-tree. This facilitates the ability for users to see patterns and trends. |
| - | The SpaceTree's iconic representation may not be understood initially. Not understanding the iconic representation lessens the value the SpaceTree offers. |

## 2.4 Summary of related work

Current tools are designed to enhance the capabilities of usability practitioners to convert raw data, data gained from an evaluation, into usability data. The tools help practitioners a great deal. Some tools can record every aspect of the session; voice, screen capture, participants expression/actions, and system events. The raw data obtained from the evaluation is reviewed; usability problems are formalized and documented. The documentation of the usability problems vary by entity and could even vary within the entity from person to person. The tools aid in capturing data so evaluators are capable of identifying and documenting usability problems efficiently and accurately. Now usability practitioners can proceed in the in the process and evaluate the usability problems to see what problems are worth fixing and can they be fixed within budget. The tools help in converting the raw data into usability problems. Generally the information learned is only applied to the specific product being evaluated.

A lot of research has been done on visualizing tree structures. Each technique has its good points and bad. Our purpose was not to create a new tree visualization but to comprehensively analyze the tree visualizations and take the best attributes of each method. After careful analysis the SpaceTree was chosen as the foundation to design and develop the UAF visualization that we will refer to as Vizability.

In the later sections we will describe Vizability, why the SpaceTree was the best choice to base our visualization design upon, and how it has been customized for the purposes of the User Action Framework.

# 3 Approach and Tool Development

Usability evaluations are performed to find usability problems. However, each usability problem on its own provides only a minimum return on investment (ROI) for those parties involved. If the usability problem is addressed as a single entity, as it often is, the benefit in return will be a discrete and nominal value. The resulting ROI will be a single usability problem will be fixed (assuming its important enough to be fixed, and within budget). On the other hand, usability problems viewed as a set can provide a much greater ROI over the entire process.

Usability practitioners are not currently provided many tools to view usability problems at a higher level; to view usability problems as a whole. One such method that usability engineers can use to gain a higher level of abstraction is affinity diagrams. Affinity diagramming and it's purpose can be explained by McQuaid (McQuaid & Bishop, 2001):

> *"Affinity diagramming is a technique that allows people to organize a large number of issues into categories based on each issue's affinity with, or similarity to, other issues. This technique enables us to see patterns in the problems—which problems tend to group together and which seem unrelated. These patterns, in turn, give us a more integrated view of the problem space. Instead of focusing narrowly on each individual problem, we can expand our focus and get a high-level view of all the problem areas.*
>
> *Having a higher-level perspective on the problems is incredibly valuable because it allows us to envision solutions not to each problem, but to whole categories of problems. For example, while evaluating an interface, one evaluator might comment that he had difficulty selecting items from cascading (fly-out) menus, another might have a problem understanding how the navigation areas at the top, left-hand side, and right-hand side of screen relate to one another, while another evaluator might have problems determining where*

*she is in the system. Although we can, and do, recommend solutions to each of these individual problems, we can also see that they are symptoms of a larger problem, namely poorly designed, or undesigned, information architecture(Rosenfeld & Morville, 1998) ."*

The high level abstraction that is gained by the use of affinity diagrams can now be applied to the overall process. As McQuaid and Bishop (2001) explain (see above), the identified usability problems are often apart of a larger problem, in this case information architecture. Using the affinity diagram technique, usability practitioners can not only address the individual problems as needed, but they can also address the item(s) that caused the usability problem, such as better trained developers or more time spent to gather requirements.

Visualization, as noted by (Card, Mackinlay, & Shneiderman, 1999 ), can be seen as a tool to allow persons to view large sets of data to enhance a persons ability to see phenomena in the data. In this case affinity diagrams act as a visualization of the usability problem set that allow usability practitioners to see how the usability problems relate to each other as well as the process that created the usability problems. Without visualization, usability problems can only be viewed as individual pieces of information rather than what they are, a small piece of a larger process.

One may ask, if affinity diagrams have so much to offer, why aren't they an integral part of the process? One reason can be attributed to time constraints. Usability problems are often not documented or organized in a manner that facilitates usability problem groupings, or identifying common issues between problems. Nayak, Mrazek, et al. explain that usability data is often observational and must then be interpreted by the usability engineers as to the exact usability problem (Nayak, Mrazek, & Smith, 1995 ). The translation from a user's observed action to a documented usability problem can take considerable time. To take the additional step to group those usability problems by a common issue is often not feasible.

The User Action Framework, by nature, clusters problems by common issue and DCART enforces proper documentation of all usability problems. With the UAF framework and

DCART integrated into the usability process usability practitioners are able to efficiently integrate a means to visualize usability problems. Figure 14 shows the point in the process where visualization is integrated, 3*.

Figure 14 is modified from Figure 1 to incorporate the visualization step in the process. This new step enables usability practitioners not only fix appropriate usability problems in the current product but also use the usability problems to "see phenomena in the data" (Card, Mackinlay, & Shneiderman, 1999 ). This phenomenon takes the form of usability problem patterns and trends in the UAF structure; abstracting the usability problems to enable practitioners to view problems by common issue or concept.

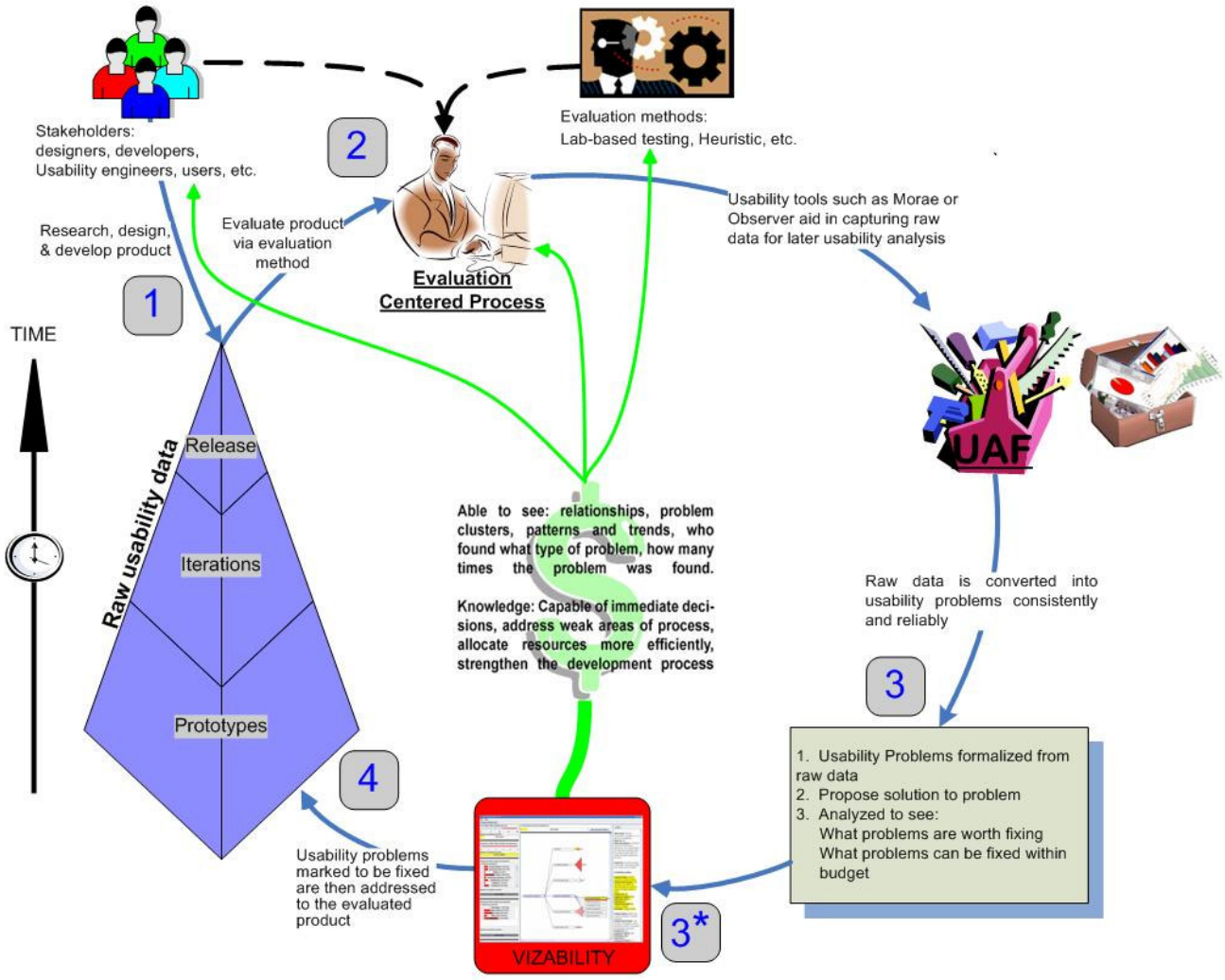



**Figure 14: Generalized product development life cycle integrated with Vizability, a tool that enables practitioners to visualize usability information**

By abstracting the usability problems and viewing the problems as a set, usability practitioners are able to apply the usability information learned to the entire process, generating a larger ROI. This is illustrated by the dollar sign ($) in Figure 14. The dollar sign ($) and respective paths represents the benefits visualization adds to the process. With visualization the practitioner is able to use the usability information to address the stakeholders' strengths and/or weaknesses, allocate resources (money, staff, training, equipment) more efficiently. Further, this visualization allows the practitioner to review the evaluation method and answer the question of: Did the evaluation method identify

enough usability problems for the resources it took; is there a better evaluation method for the type of system being evaluated? Third, the practitioner is able to use the visualization to evaluate the evaluators -- Are the usability experts being used for heuristic walkthroughs identifying usability problems?

These benefits grow as more usability data are collected in the UAF database. This means as time passes practitioners will be able to see trends in usability data over time and over multiple projects. The ability to see this phenomenon in the data is useful for different user classes; it is not only useful for the usability engineer to address usability problems, but it is also useful for the developer to be able to see the usability problems that are being repeated. The administrative staff can also view the trends to see better ways to allocate resources to improve the entire process.

Visualization is needed to see usability problem trends and patterns. The visualization can serve different purposes to the different user classes; each one of the purposes contributes to the overall ROI of usability.

## 3.1 Designing Vizability

In the beginning phases of development the user requirements needed to develop Vizability, the visualization tool, were not clearly identified. Therefore, the approach taken in the development was that of an evolutionary development model. This model can be characterized by systems that are developed iteratively as more user requirements are gathered and identified (Pressman, 2001).

We started the development process by understanding a successful visualization. Figure 15 illustrates the steps that need to be defined in order to develop a successful visualization. The raw data are supplied to us by the product (see the pyramid in Figure 14). The raw data are then analyzed to identify usability problems, and transformed into data tables with the use of DCART. The UAF is setup to use two databases. The first database contains the UAF structure, the descriptions for each node, and the keywords associated with each node. The second database contains the usability problems and all associated data. At the time of this writing the second database did not exist as a finalized

version; it is still being created. So in order for the development of Vizability to continue it became necessary to research the key usability problem attributes in order to create a custom database for the tools use. This step was intended to keep the prototype implementation as close as possible to the final implementation.

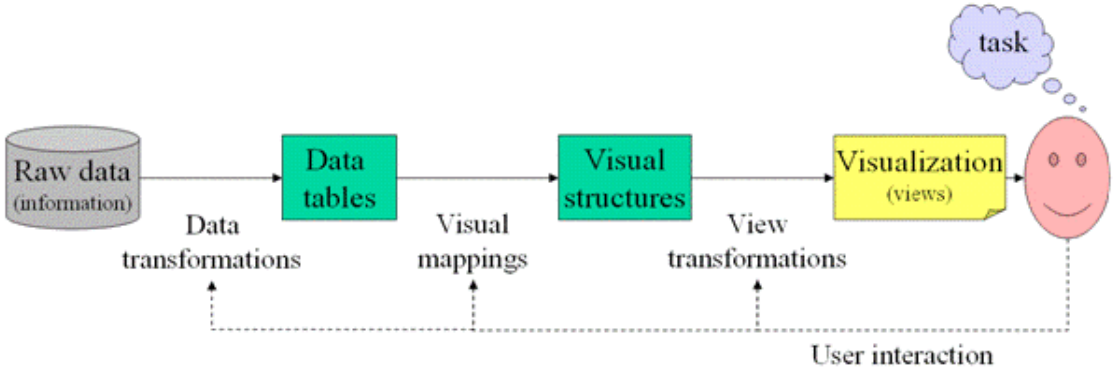



**Figure 15: The model by which raw data are taken and mapped to a visual structure to amplify human cognition (Card, Mackinlay, & Shneiderman, 1999 )**

### 3.1.1 Data to Visualize

The tool, DCART, used to collect usability data at the time of this writing is still being developed. Therefore the exact usability data that will be collected is unknown. For the purposes of thesis four key usability problem attributes were chosen.

The usability problem attributes are:

- Cost: the cost in person hours to fix a usability problem. Cost can be any reasonable real number.
- Importance: the extent to which a usability problem is important to fix. Rated from 1 to 5, with 5 being a 'must fix' usability problem.
- Keyword: a word associated with each node to provide a general theme to those usability problems located in the node
- Evaluator: the evaluator who found the usability problem.

The four usability problem attributes listed above were chosen because they are key attributes of each usability problem. Other problem attributes may prove to be more or just as important to visualize but these four provide a great start in order to better understand the visualization requirements.

Since the four usability problem attributes represented a subset of an unknown number of possible attributes, the visualization was designed to be scalable. The visualization is scalable to visualize *n* number of attributes and *n* number of usability problems. Within the visualization the UAF tree structure itself does not have to be scalable since it is a static structure that is "filled" with problems.

The data tables were retrieved from the UAF database and were then mapped to the data tables (each individual tuple) to a visual structure. Shown below is the database schema for the UAF problem database. This schema was created for the purposes of the prototype and will not be the final version of the UAF database.

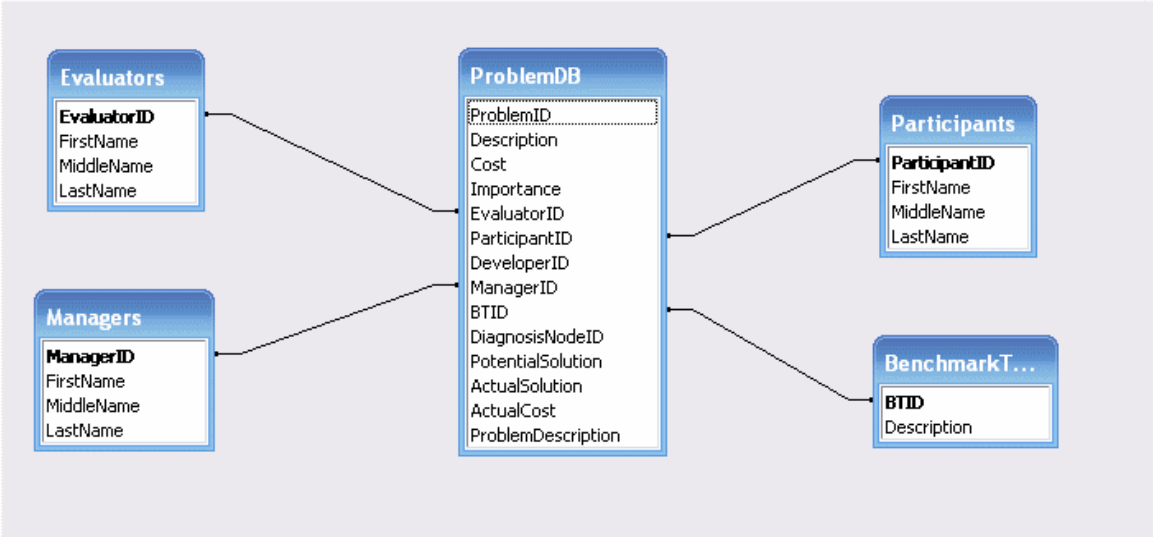



**Figure 16: Database schema for the UAF Usability Problem Database**

The schema (above) also includes participants, managers, and benchmark tasks. These items are not currently displayed in the visualization and are included for future implementations. Including these items in the visualization would give more insights to

the user, by customizing the data for a managers perspective; for example, display the data in terms of time, money, and resources. Benchmark tasks could be used to link the problems against performance measures, time taken to complete the task, or number of errors. Finally participants could help in refining evaluations, picking participants (expert evaluators) who specialize in certain area. The items listed for future versions need to be researched further to see how exactly they could be useful in the visualization.

The following sections will list the requirements of the tool, the visualization mapping, and then summarize each stage of the tool's development and how the visualization and visual mappings evolved.

## 3.2 Visualization Requirements

In order to clearly understand the steps taken to develop the Vizability tool it is first necessary to explain the user requirements gathered to implement Vizability v1.1, the most current version. Those requirements are listed in Table 1. The focus will then turn to a description of the earlier prototypes, the features they lacked and the improvements that came with each new version.

| Vizability Requirements |
|---|
| View usability problem information patterns and trends from a single project as well as multiple projects |
| Filter usability problems based on specific attributes |
| View node names of the User Action Framework |
| View usability problem clusters |
| Provide the functionality to identify the quantity of problems in a node or sub-tree |
| Ability to collapse the UAF structure so users were able to view clusters of usability problems |
| Provide the functionality to view the details of a usability problem or all usability problems that match a user defined query |
| View usability problems as they relate to keywords |
| View usability problems as they relate to evaluators |
| Provide the functionality to view usability problems and their distribution with respect to user defined queries |
| Represent the entire UAF structure in one view |
| Cost and importance filters should be able to select a range of values |

**Table 1: Requirements for visualizing the User Action Framework**

Mahajan who developed one of the earlier versions of DCART stated that being able to visualize the usability data stored in the UAF would help users detect major flaws in the interface (Mahajan, 2003). In its current state, users have a tool they can perform basic searches based on keywords. This type of listing is good when a usability engineer knows what he/she is looking for at the time of the search. However, this type of search is not useful for usability engineers who would like to understand the overall problem set and see the big picture. In order to understand the larger picture one has to be able be able to view the entire problem set in a way that facilitates understandability. It is important that the user be able to see patterns, trends, and weak areas of design as well as strong areas.

### 3.2.1 Vizability: Version 0.9

At the time of the first prototype the User Action Framework was still being formulated. The exact usability data to be stored in the database was unknown, the UAF nodes or usability concepts were being refined, and the database was not created. As a result, the decision was made to gather requirements for an initial prototype by conducting

interviews with Dr. Hartson and reviewing UAF literature (Hartson & Hix, 1993; Keenan, 1996; Hartson et al., 1999; Sridharan, 2001; Mahajan, 2003; Howarth, 2004). These steps provided enough information to design an initial prototype to be used as a tool to aid in identifying future requirements.

The prototype displayed below in Figure 17 is the first attempt at the visualization of the contents of the User Action Framework. This prototype was created by Pardha Pyla for Information Visualization under Dr. Chris North.

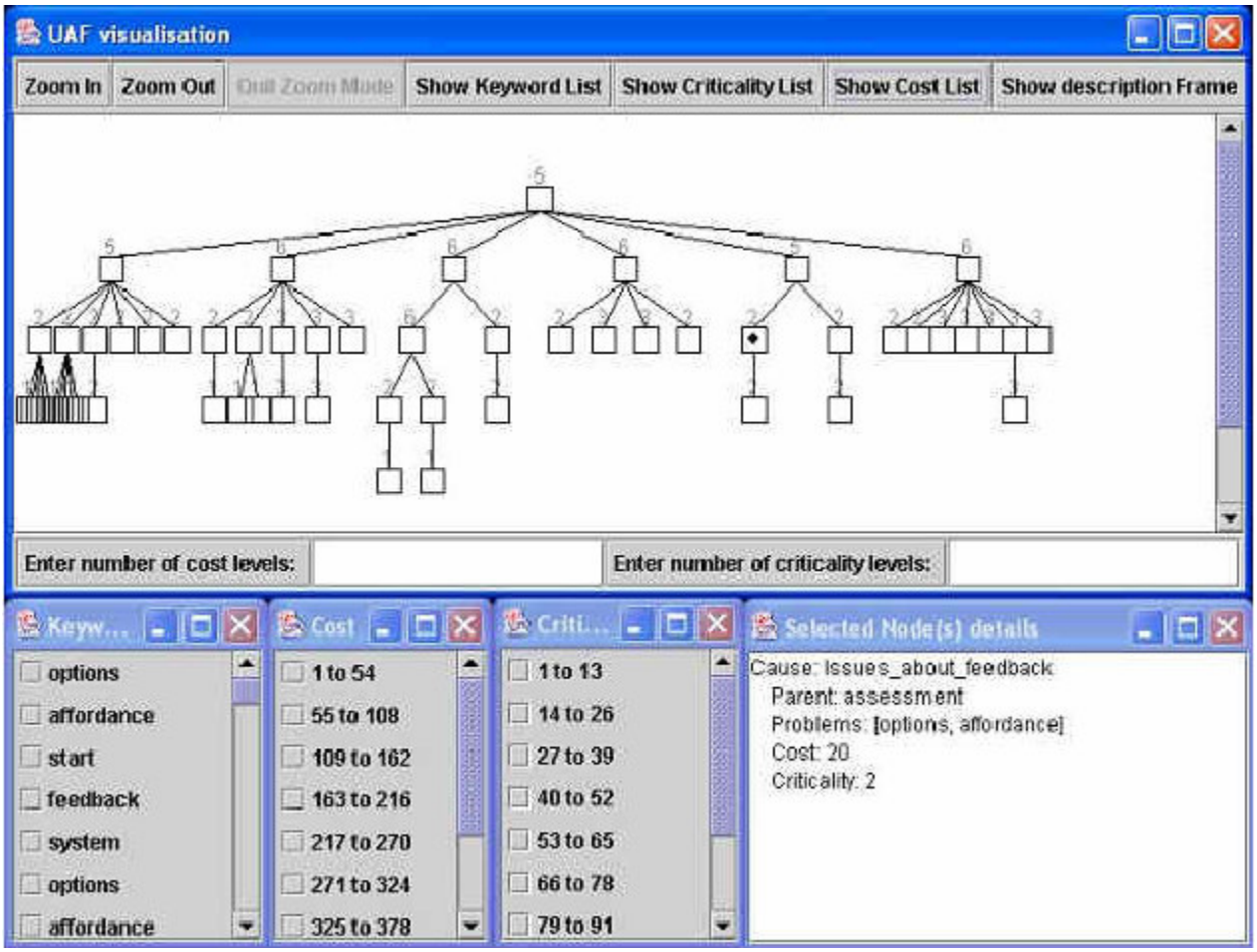



**Figure 17: First prototype of the UAF Visualization, Version 0.9**

In this visualization the information is displayed in multiple windows. Since the database was not created at the time of this prototype, the UAF data and problems were read from

an XML file. This was done as a middle tier. When the database was created, the database could be exported as an XML file to be used by the tool.

The main window displays the UAF structure. Only nodes that contain problems are displayed. This implementation is similar to the pruning dynamic query tree; only nodes of interest are shown (Kumar, Plaisant, & Shneiderman, 1997). The other windows consist of query controls for cost and importance (criticality) and details for selected usability problems. Brushing and linking were used to coordinate user selections and the UAF structure (Ahlberg & Wistrand, 1995). Figure 18 demonstrates how user selections of cost, importance, and keywords were mapped to the main window, the UAF structure.

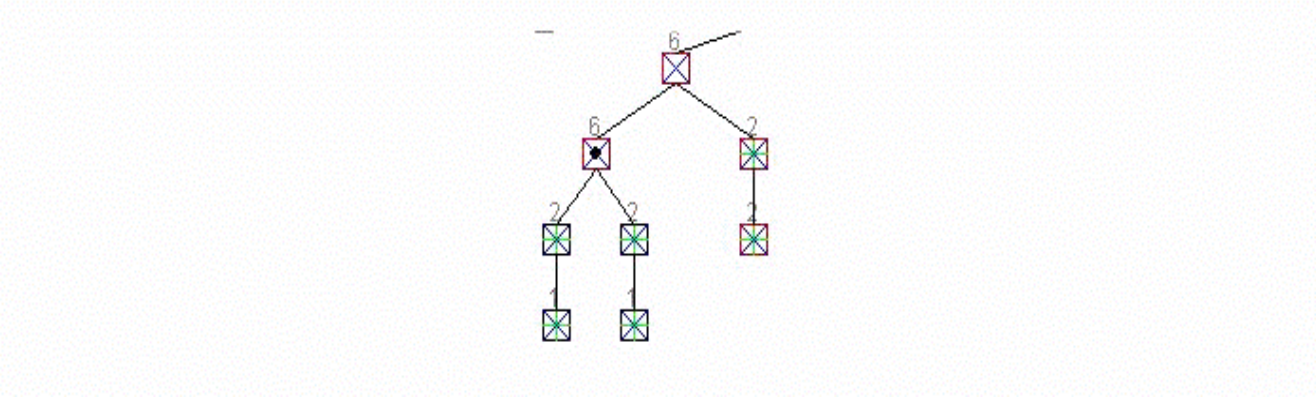



**Figure 18: Technique used to encode multiple attributes placed a high cognitive load on the user. Red border to show the nodes with selected keyword occurrence, blue cross inside the node to show the nodes falling within a particular cost range, green plus marks for nodes with importance in the user selected range and selected nodes were shown using a black circle inside all the nodes**

As one can see, a lot is happening in the visualization; different colors are being used, as well as multiple marks. This particular visualization proved to be useful to build upon but was not useful to any users.

Usability issues and requirements were identified by conducting informal interviews. A sample of the information gathered is listed below. The list below is not comprehensive but highlights some of the major issues in the visualization and design.

| Version 0.9 Usability Problems & Requirements |
|---|
| Visualization technique used to visualize attributes resulted in a high cognitive load place on the user |
| Users wanted to see usability problem patterns and trends (Table 1, Req. 1). Usability problem patterns were not obvious to users |
| There was no text used in the visualization (Table 1, Req. 3). Users could not identify node unless on mouse over |
| Only those nodes with usability problems were displayed. Since the nodes lacked labels it was hard for users to understand what nodes were being displayed |
| The visualization read from an XML file rather than the actual database. Due to XML limitations (e.g., no spaces in child/sub-node names) it caused problems when displaying text. This can be seen in Figure 17 in the details area. Underscores were used in place of spaces. |
| Users wanted to see usability problem clusters (Table 1, Req. 4). Nodes with problems were denoted by a number representing the quantity of problems in that specific node. This meant the user had to read each individual number rather than being able to visually scan the diagram. |
| Users could not collapse the tree to focus on a particular section of the tree structure |
| Details area was not very readable due to formatting, layout, and XML issues |
| Separate query windows were irritating, often being hidden by overlapping windows |

**Table 2: List of usability problems and requirements for Version 0.9**

### 3.2.2 Vizability: Version 1.0

Improvements were made to version 0.9 and were implemented in version 1.0. This version was a considerable improvement. Usability problems were indicated visually rather than with numbers, the tree structure was collapsible, query windows were integrated into one window, and color mapping was given more affordance by providing context; green represented cost, red represented importance, and blue represented keywords.

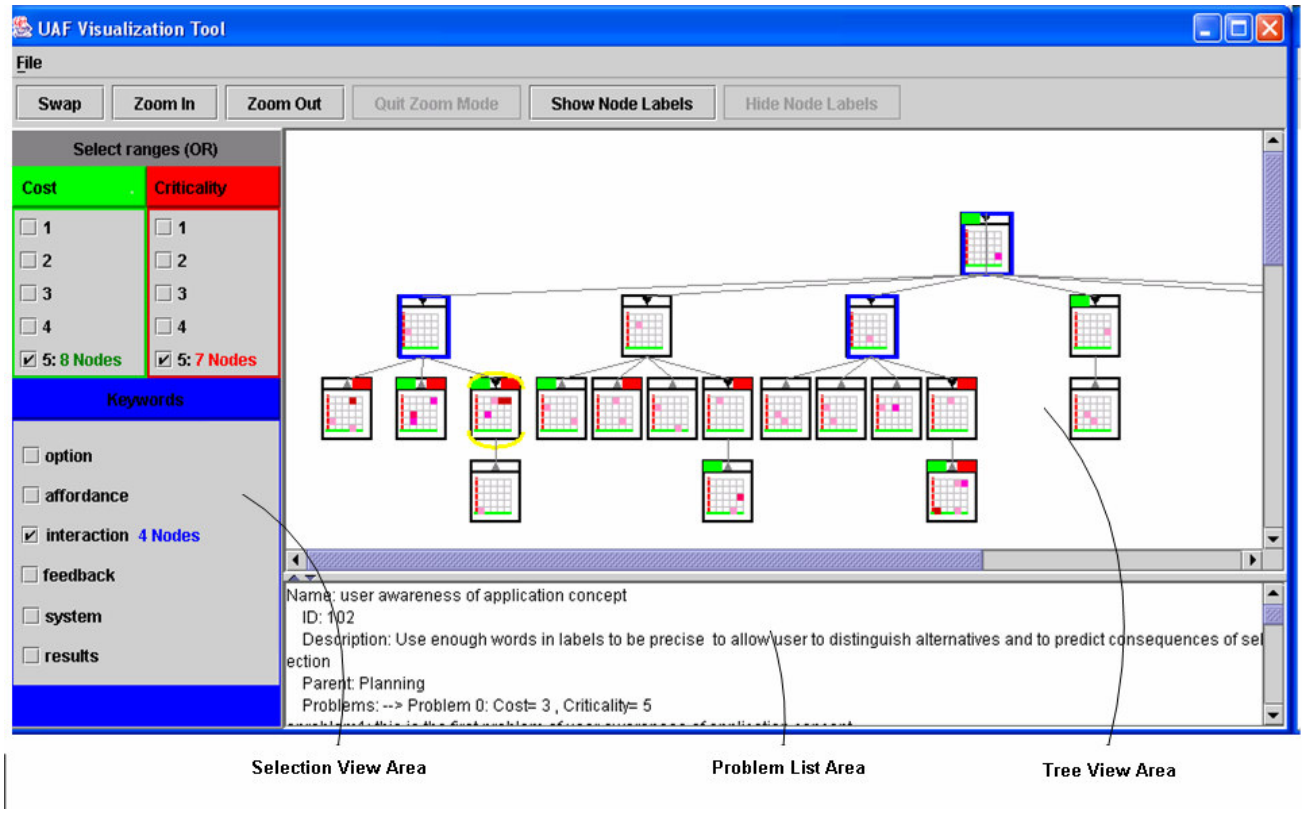



**Figure 19: Version 1.0**

This version integrated some of the requirements that were identified in version 0.9. Figure 20 illustrates the ability for the user to show node labels. As seen in the figure, the node labels are shown but in almost all cases are not readable due to layout problems.

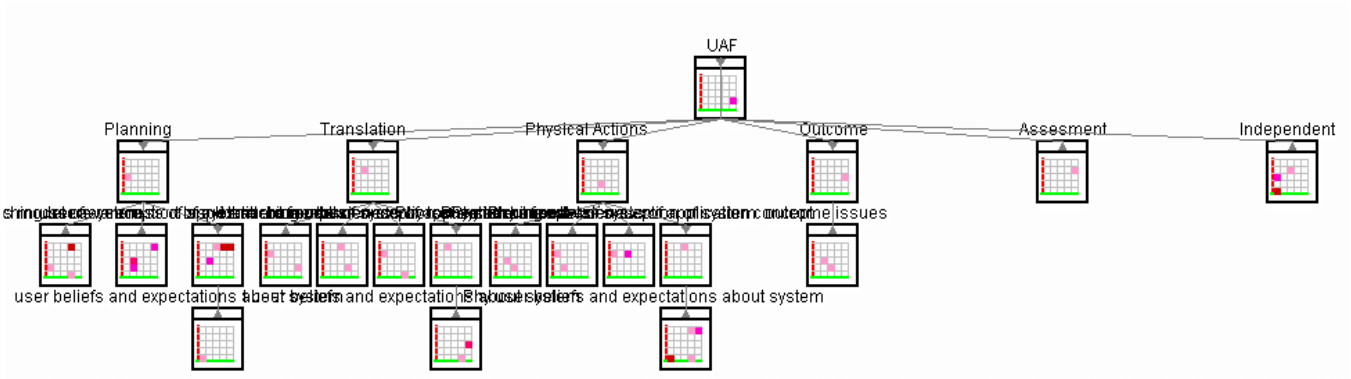



**Figure 20: Show node labels in the horizontal view**

Figure 21 shows how the layout problem was addressed by constructing a new vertical layout so the text has more space to occupy. However, due to the node style and the long node names this still posed a problem.

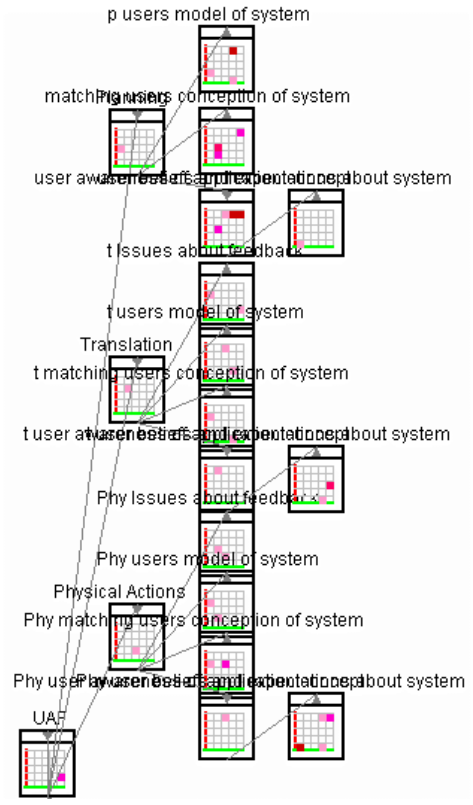



**Figure 21: Show node labels in the vertical view**

The nodes in this visualization were developed to help the user understand usability problems patterns and trends as well as problem clusters (requirements 0 and 0). By looking at each individual node one can see the number of usability problems located in the node along with each problem's respective cost and importance rating. This visualization allowed user to easily see relationships and patterns within each individual node. This implementation satisfied the requirement but at a low level. A later evaluation of this visualization revealed that users want to see clusters, patterns, and trends in the overall UAF structure and not be limited to the node. This implementation might be useful as an additional feature for users, comparing a large number of usability problems at a more detailed level. The nodes would be very useful in discovering usability problem patterns within individual nodes but not the overall UAF structure.

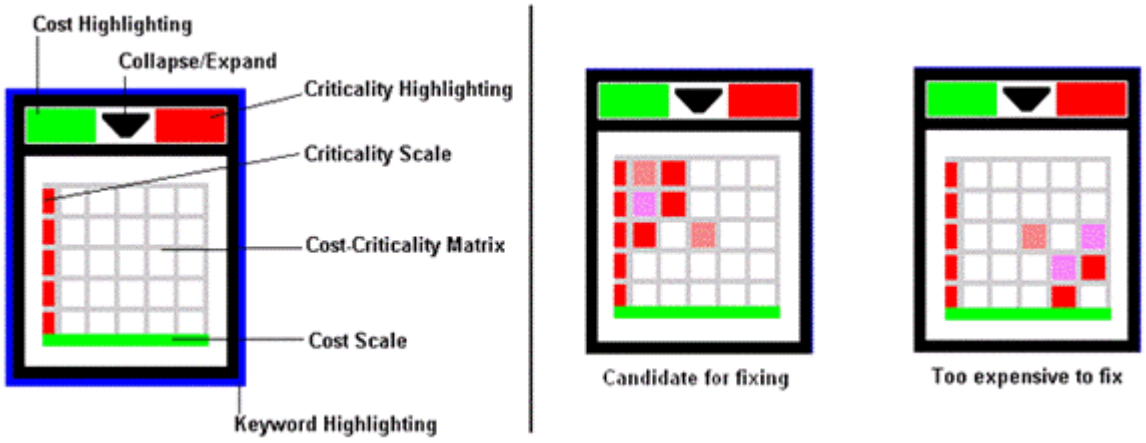



**Figure 22: Left side: The components of each node. Right side: A problem summary of the node. Each dot represents one or multiple problems in the respective cost/importance matrix. The more intense the color the more problems have the respective cost/importance value.**

Users were also able to collapse nodes in order to focus on specific sections of the structure. This satisfied requirement 0 but collapsing the node hides other information that users may still be interested, such as the number of usability problems in the sub-tree of the collapsed node.

The learning curve was high and still lacked the fundamental purpose of visualization, amplify human cognition by abstraction and aggregation of information (Card, Mackinlay, & Shneiderman, 1999 ).

Version 1.0 was implemented to satisfy the requirements know at the time. It also helped to refine the requirements in more detail and investigate additional user needs. The investigation proved to be valuable in that it contributed to a more comprehensive list of requirements for the next design iteration.

### 3.2.3 Vizability: Version 1.1

Earlier prototypes were a necessity in order to identify and generate a solid foundation of requirements. As Vizability tool is in its early stages of development, it is understandable that the requirements stated in this document will be modified based on additional findings as the UAF tools develop and more uses are realized. To meet this anticipated need, the tool has been designed to be scalable as more requirements are defined. The initial requirements for the Vizability tool are in Table 1.

After identifying these requirements, the next step was to re-evaluate existing visualization methods based on the extent to which a given method met the gathered requirements. The SpaceTree (Plaisant, Grosjean, & Bederson, 2002) was deemed to best fit to these user requirements in that the SpaceTree alone satisfies requirements 0, 0, and 0 in Table 1. The design of the space tree is conducive to conveying nodes and sub-trees within a limited space. The SpaceTree when collapsed represents the collapsed section of the tree with a triangle. “The shading of the triangle is proportional to the total number of nodes in the sub-tree. The height of the triangle represents the depth of the sub-tree and the base is proportional to the average width (i.e., number of items divided by the depth); (Plaisant, Grosjean, & Bederson, 2002).” Each representation of the triangle’s attributes was modified to satisfy requirements 0, 0, and 0. The nature of how the triangles have been modified will be described in more detail later in the Description section. Due to the similarities of identified user requirements and the existing features of the SpaceTree, it seemed reasonable to use SpaceTree as a foundation and customize it to fit our needs.

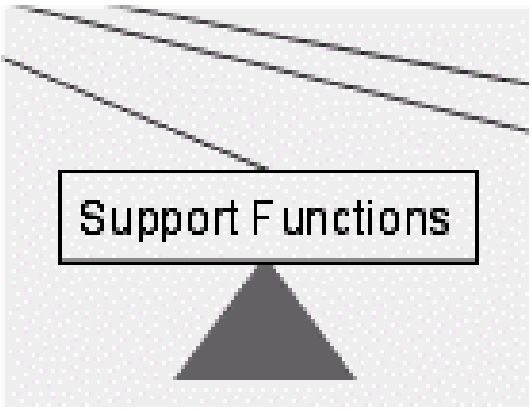



**Figure 23: The triangle under the node "Support Functions" represents the nodes under it (sub-tree) as a group. Shading represents total number of nodes, height represents the depth of the sub-tree, and the base represents the average width of the sub-tree.**

The transition from Version 1.0 to 1.1 was marked by significant changes. The design was first prototyped with paper in order to refine the design. The paper prototypes helped clearly convey the proposed design and ensure it satisfied all known requirements.

Some slight modifications were made from the paper prototype during the implementation. The most notable changes are: tabs were used to separate keywords and developers, the keyword bank was implemented as a JSplitpane to give users a better perception of the bank/keyword or bank/developers relationship depending on the current tab, and a statistics area was added to show query statistics.

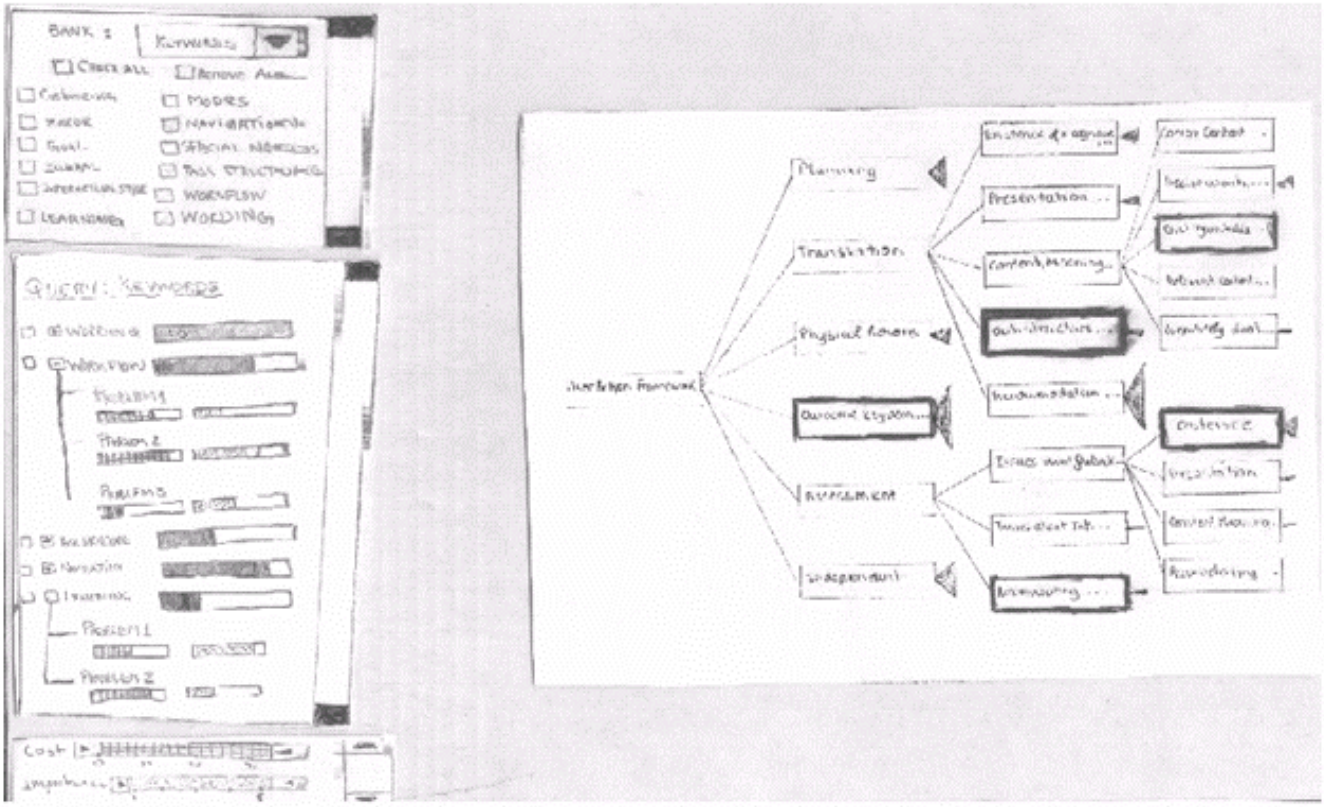



**Figure 24: Paper prototype**

In this version all requirements were satisfied (Table 1). Furthermore, this version also introduced interface enhancements to provide multiple-views of the data. Multiple views provide utility by separating the dimensions of data into simpler views of the multi-dimensional data (Baldonado, Woodruff, & Kuchinsky, 2000; North, Conklin, & Saini, 2002). The multiple views will allow users to decompose the usability data into simpler data chunks. The back-end of the tool was improved to read from two separate databases; the actual UAF database, now implemented, and a separate database containing the actual usability problem data. This allowed the tool the ability to display usability problems in more detail, including complete problem descriptions, proposed solutions, and all other problem attributes. The interaction techniques of the cost and importance attributes have been changed from checkboxes to a more dynamic method using double knobbed sliders. The usability data that can be queried was modified to include the evaluator who discovered the usability problem. The current state of the tool

will be discussed in more detail in the next section and in the Vizability 1.1 Re-designed section.

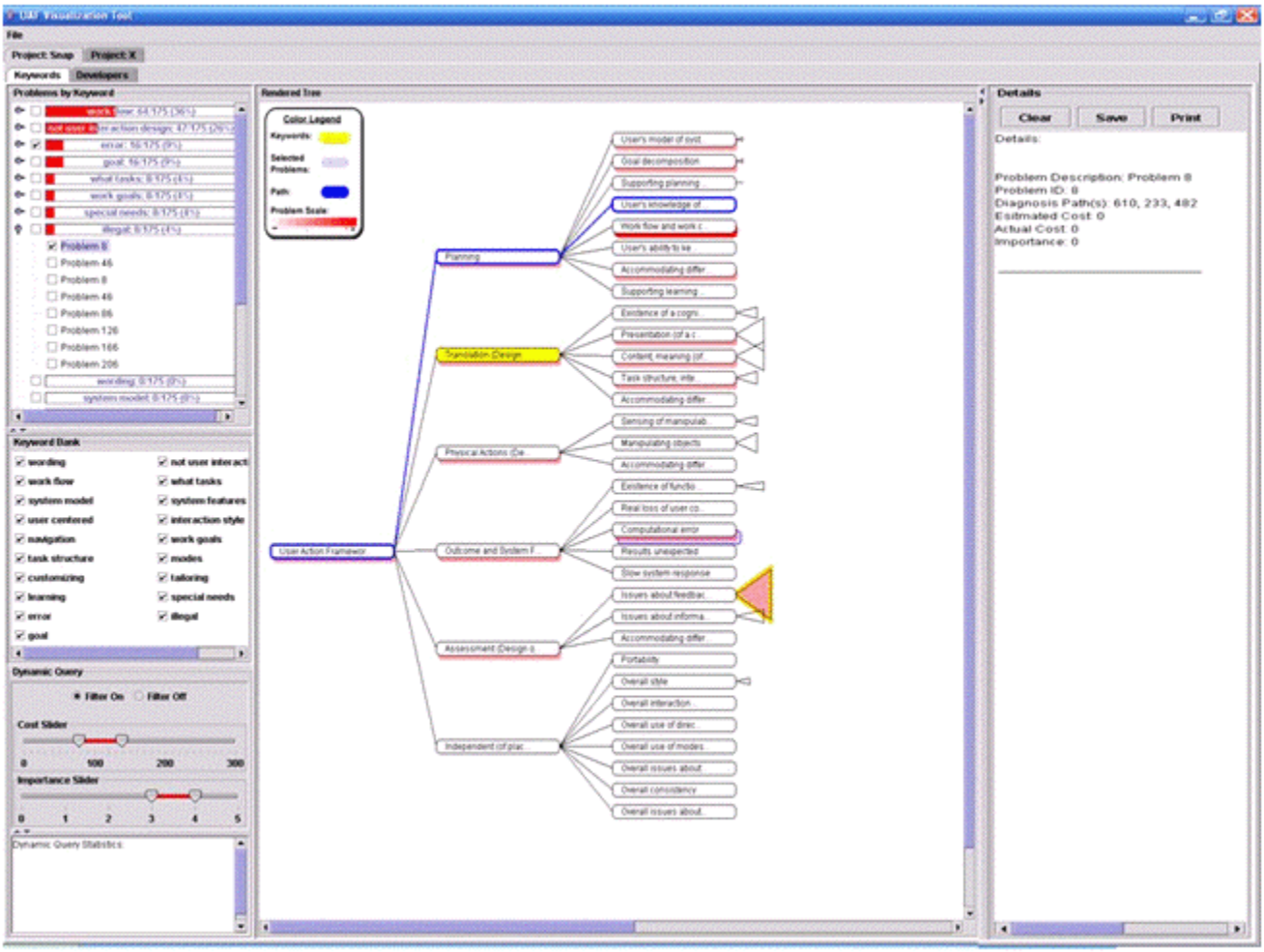



**Figure 25: Version 1.1**

## 3.3 Description of Vizability 1.1

Vizability is a complex tool that gives the users different ways to visualize and organize the usability data based on four key attributes; cost, importance, keywords, and evaluator. The capabilities of the visualization will be enhanced as development proceeds further. For instance; mapping usability problems to usability problems to show related problems will help usability engineers to strategize their usability efforts. Doing so would lessen the time and resources by allowing the development team to see that, by addressing one

problem, it will lessen the importance rating of 4 other related problems. The table below describes pairs of attributes and their respective purposes.

| Attribute 1 | Attribute 2 | Reason/Purpose |
|---|---|---|
| Cost | Usability Problems | Allows users to view the number of usability problems and see patterns and trends associated with cost in the UAF. This will help to solve/avoid problems that occur in more costly areas of the UAF |
| Importance | Usability Problems | Allows users to view the number of usability problems and see patterns and trends associated with importance in the UAF. This will help to solve/avoid problems that occur in more important areas of the UAF |
| Cost | Importance | Allow users to view the cost and importance ratio of a usability effort or multiple efforts. |
| Evaluator | Usability Problems | Allow the user to see what evaluator finds the most problems. Could be due to more experience, different evaluation method. Enables easy identification of the strengths of each evaluator. |
| Evaluator | Keywords | Allow the manager to see trends in the type of problems an evaluator finds. This could be linked later to the type of evaluation method used. |
| Evaluator | Importance | Allow users to view each of the usability problems found and the problem's importance. |
| Evaluator | Cost | Allow users to view each the usability problems found and the problem's cost. |
| Keyword | Usability Problems | Allows the user to view the number of usability problems associated with each keyword and see how the problems are distributed in the UAF |
| Keywords | Importance | Find out what keywords are usually associated with a higher importance rating. Teams could put more importance to avoid such problems |
| Keywords | Cost | View trends in the relationship to keywords and cost |

**Table 3: This table shows pairs of attributes that are able be visualized and their purpose.**

### 3.3.1 Overview of tool

Vizability has been designed in compliance with the mantra: overview first, zoom and filter, then details on demand (Shneiderman, 1996). The tool is divided into three main

parts that we will describe in detail in the following sections. The center area is the Overview Area. The Overview Area represents the big picture and contains the UAF structure where problems are classified. The left side is the Problem Selection Area. The Problem Selection Area allows the user to build a query based on usability problem attributes. The third part of the tool is the Details Area. The Details Area allows the user to view all the usability about the queried or selected usability problems.

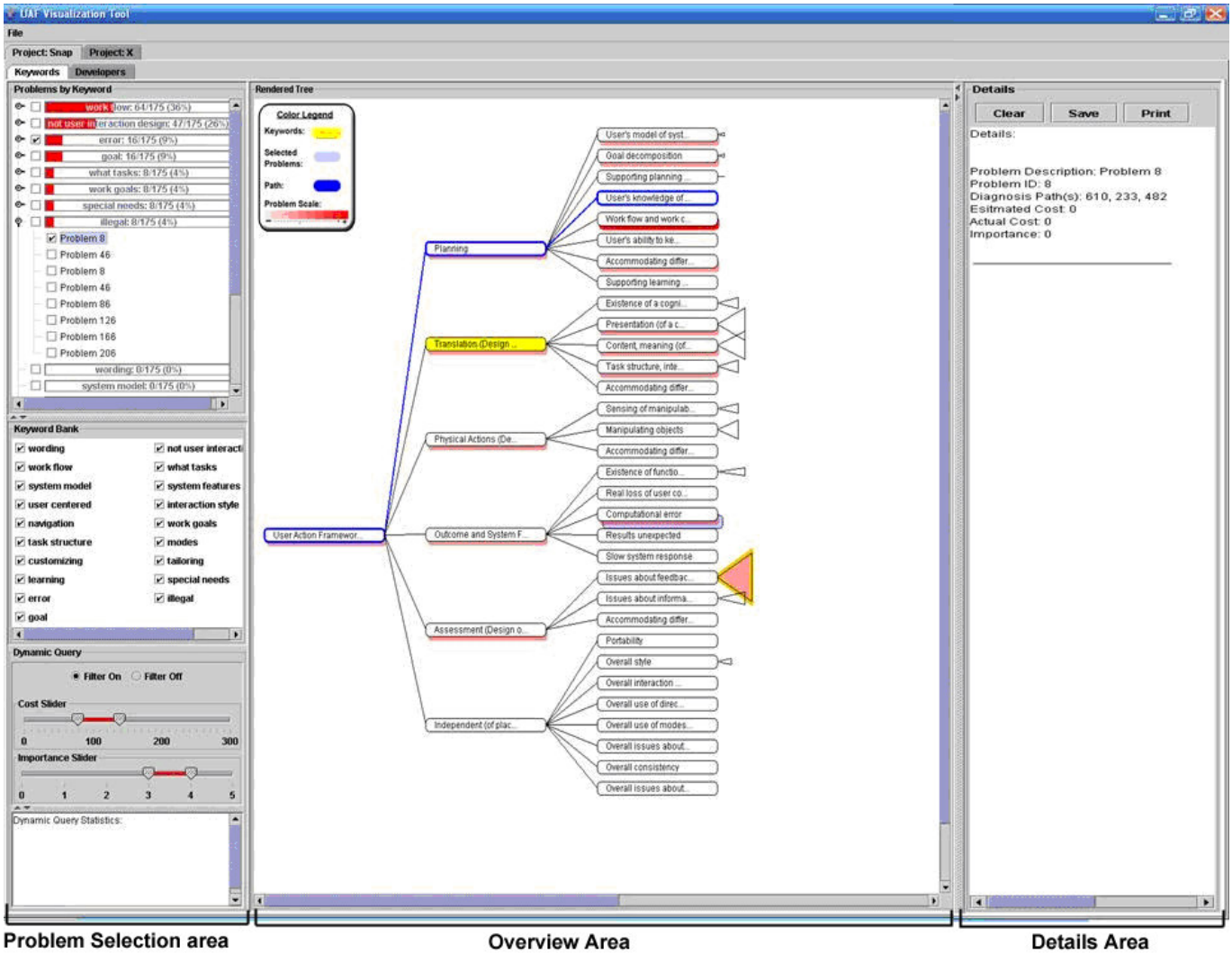



**Figure 26: Full screenshot of version 1.0**

## 3.3.2 Overview area

The overview area is a visualization of the UAF structure; each node is represented by a rounded rectangle and parent-child nodes are connected with black lines. The tree has

been implemented to display the first three levels of the UAF in the initial set-up. This allows the user to get a good idea of problem distribution and types of problems in the initial glance. Each node of the tree can be clicked on to expand or collapse the children. Such a feature is useful for focusing on a particular aspect of the tree.

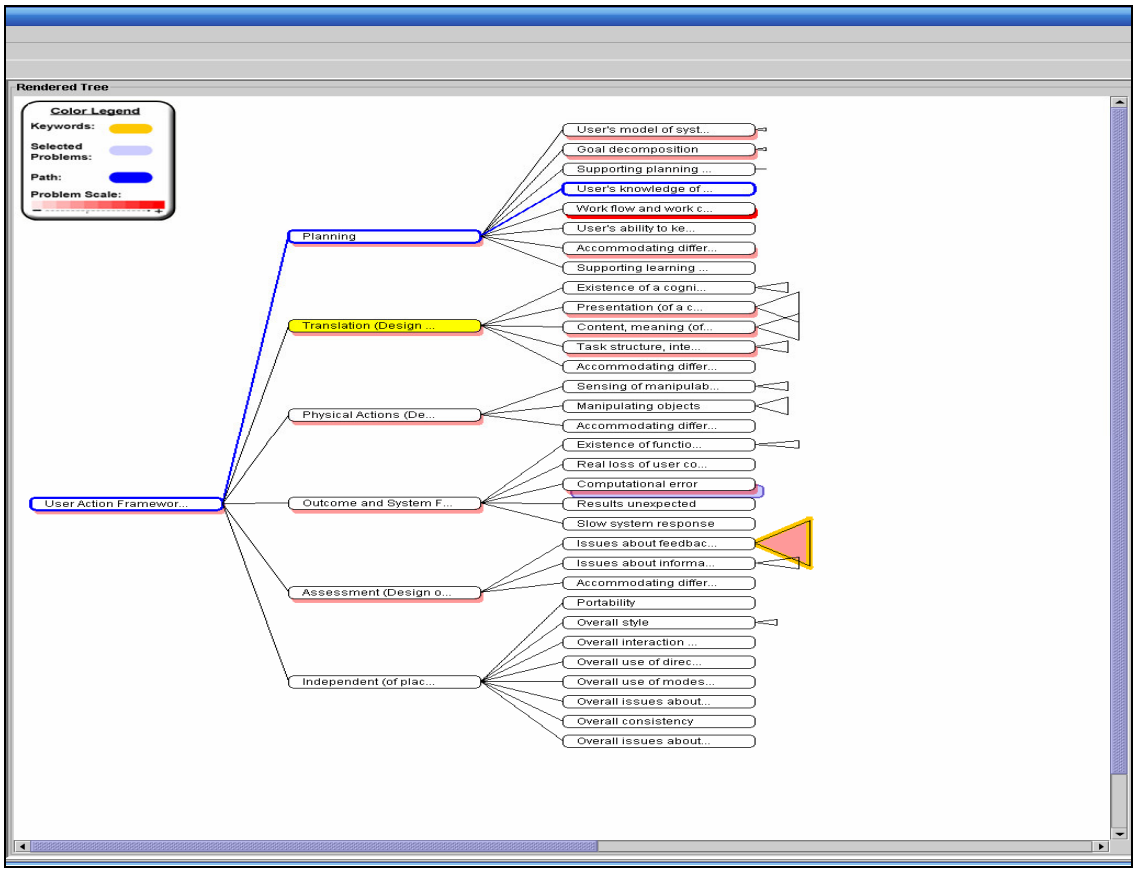



**Figure 27: Overview area**

When nodes are collapsed a triangle appears to represent child nodes that are not currently displayed (discussed later). Due to screen space the user may only have one open branch of the tree past the third level. For example, if the user has un-collapsed the planning branch of the UAF to the forth level then would like to explore a different section of the tree. The initial planning branch will collapse automatically so the user may explore the new section. This allows nodes to be easily seen without over crowding the screen. However, if the user chooses to return to the planning section of the tree the

user's last point of interest, in this case the fourth level will be shown upon clicking the node allowing the user to continue his or her work flow.

Since screen space is a major concern all nodes are not shown fully at one time. Triangles are used to represent the nodes not shown. Our tree representation is strongly influenced by the SpaceTree (Plaisant, Grosjean, & Bederson, 2002). The alternate representation of nodes using the triangle has been adapted from The SpaceTree. Plaisant's SpaceTree uses the triangles attributes size, height and base, and color to map to specific attributes of the tree. Similarly, we have used the triangles attributes to map to multiple attributes as shown in Figure 29. The base of the triangle maps to the total number of nodes in the sub-tree and the height represents the depth of the sub-tree. Finally, the intensity of the red shading is proportional to the number of problems associated with the sub-tree. This feature allows the user to quickly summarize the sub-trees in one glance without the need to scan or query each individual node.

The node itself is a rounded rectangle labeled with its respective usability concept. The node may have a shadow of varying red shades corresponding to problems located in that specific node. If there is no red shadow, it means that no problems are associated with that node. The red color signifies problems. Giving the red color a consistent meaning throughout the interface

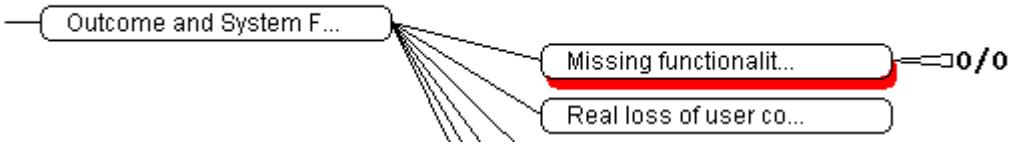



**Figure 28: The red shading in the node "Missing Functionality" means that usability problems exist in that node**

The path of the node the user is currently focusing on can be seen by the blue outline of the nodes and blue connecting lines.

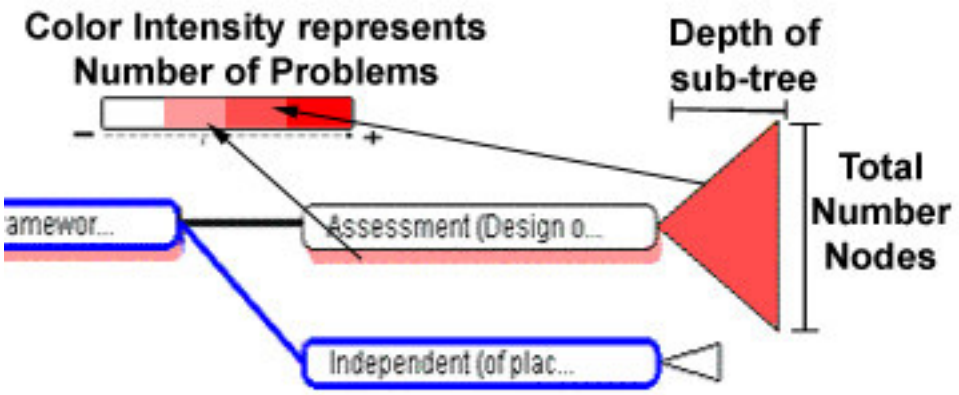



**Figure 29: The attributes height, base, and color represent the sub-tree. The height represents the depth of the sub-tree, the base represents the total number of nodes in the sub-tree, and the color intensity represents the number of usability problems in the sub-tree. Notice the node's shading represents the number of usability problems in that specific node.**

Since screen real estate is an issue the space allotted to the overview area is adjustable in cases were more space may be needed. You may adjust the screen space by dragging the dividers on either side of the overview area.

### 3.3.3 Problem Selection Area

The problem selection area acts as a way for the user to view the details about the data set. It is based on dynamic queries so user's can immediately see query results (Ahlberg & Shneiderman, 1994). The problem selection serves multiple purposes. The first purpose is it allows users to build custom queries in order to find specific problems of interest. The second purpose is that it provides a new view into the data. Users may see the relationships of specific problems to keywords or to the developer. The view is useful when users are focused on problem/keyword relationship or problem developer relationship. Another detail is users can quickly see the cost range of usability problems. The cost slider is dynamic based on the usability problem set. Therefore the highest number on the cost slider represents the largest problem to fix in person hours.

The problem selection area can be divided into 2 main parts. The top section allows users to group and/or sort problems according by either keywords or developer. In the initial view users are presented problems that have been grouped by keyword and sorted by the number of problems associated with each keyword. For example, in Figure 30 the user can clearly see that the keyword "work flow" is associated with the most usability problems.

Keywords are supplied by the usability engineer and represent another way of understanding the documented usability data. Keywords are associated with nodes. They are a new dimension by which the user can view the UAF structure. Examples of keywords include wording, layout, and system model. Keywords allow the users to drill down to specific problems of interest. The problem selection view also provides the ability to group usability problems by developer for cases where the user might want to see developers with the most experience in a certain area.

Another feature provided in the problems selection area is a bank. The bank allows users to add/remove items such as keywords or developers from the list. This feature was implemented so users can easily focus on specific items. Screen space is limited and in cases where there are numerous keywords the keywords of focus could be in two different sections of the list, having only one keyword in view at a time. Allowing users to remove keywords will allow keywords or developers in focus to be in view to the user.

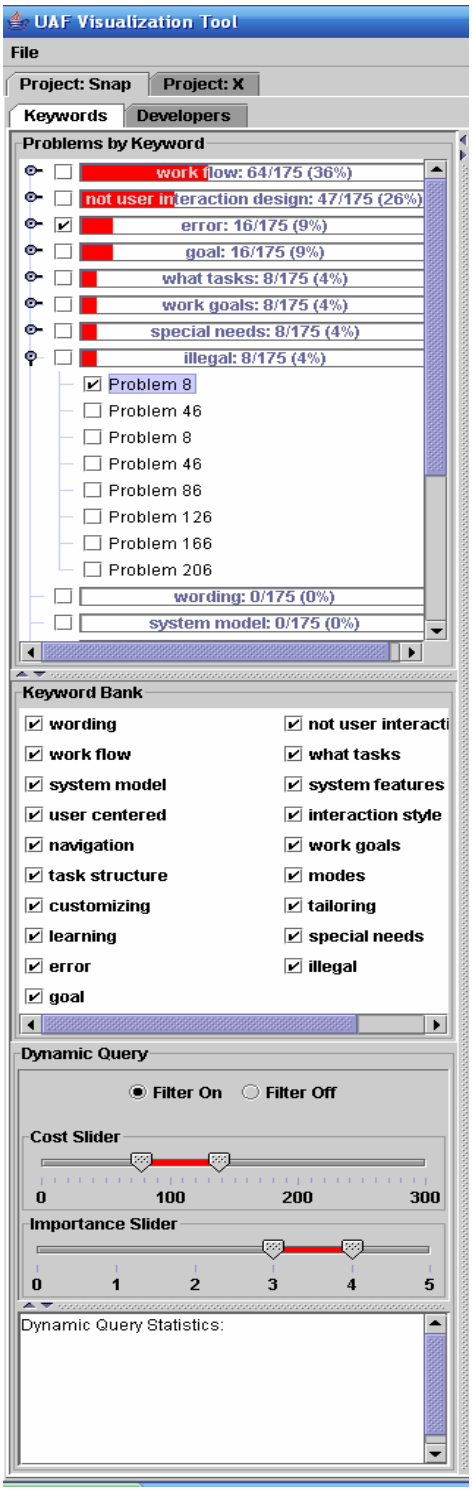



**Figure 30: Problem selection area**

The checkboxes preceding each item allow the user to brush and link to the overview area; allowing the user to see an area of interest in the overall context of the UAF. The corresponding checked items will be highlighted in the overview with a yellow color if they are keywords/developers (depending on the view) and a highlight of purple to denote selected usability problems.

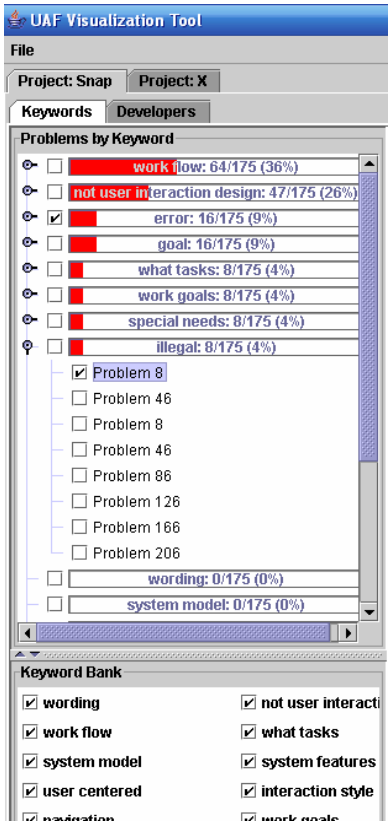



**Figure 31: Keywords sorted in order by number of problems**

Part 2 of the problem selection area is the bottom half. The bottom half contains two sliders to allow users to dynamically query problems based on cost and/or importance. The cost and importance ratings require communication between software developers and usability engineers. In particular, software developers familiar with the design and implementation of the system are responsible for supplying the cost, and usability engineers supply the importance.

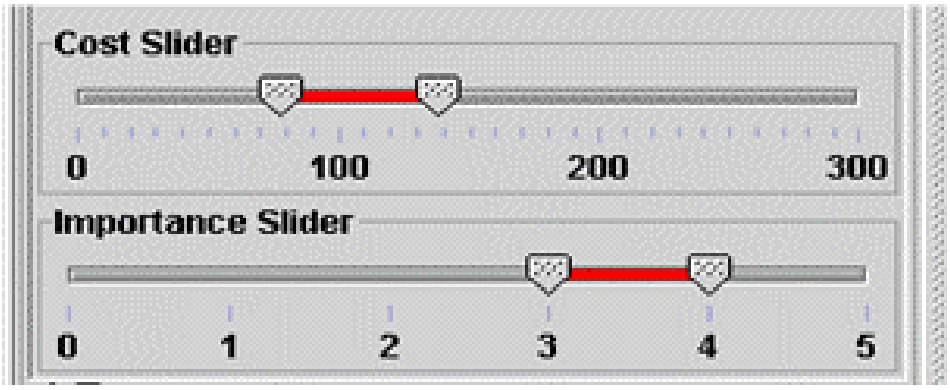



**Figure 32: The double knobbed sliders allow users to dynamically filter usability problems based on cost and/or importance**

### 3.3.4 Details Area

The details area displays text descriptions of all pertinent information about the selected problems. The tool displays three levels of detail. All details are currently accessed through a pop-up menu on right-click. The first level is a specific problems detail. If the user in the top-portion of the problem selection area selects the detail of a usability problem the detail area will display the problem description, problem id, cost, importance and the diagnosis path of the problem (location of node). If the user selects a keyword/developer the detail area will display the nodes associated with all of the keywords/developers in the UAF structure and displays the details of all associated usability problems. Finally, in the overview area the user may find the details of a node. The details of a node include: the node id, keywords associated with the node, and all problems currently located in the specific node.

The detail area is useful for understanding a collection of problems at a lower, less abstract level. Information about the diagnoses path and the actual node is essential for an a "big picture" understanding of deficiencies in a given effort, but the understanding the problem details is essential for understanding errors at the level of detail necessary to fix them.

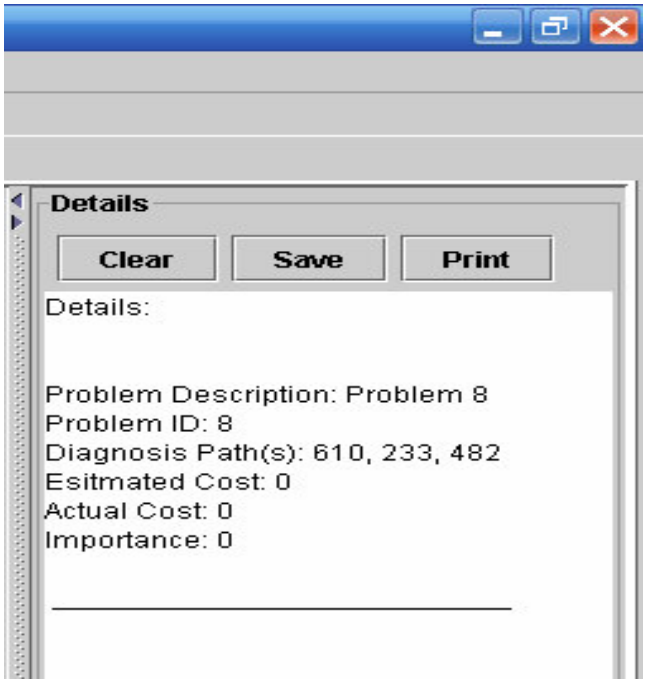



**Figure 33: The details of a specific problem**

## 3.4 Vizability 1.1 Re-designed

Upon completion of Vizability one more iterative design phase was implemented before the formative evaluation. This was done by asking two people to do walkthroughs and conducting informal interviews. Both persons had advanced knowledge of the UAF and offered many useful comments and suggestions. The suggestions and information gathered from the interviews are listed in the table below. The suggestions were placed in one of two categories: 1. Within budget to be implemented before the formative evaluation or 2. Out of budget and to be documented for later implementation. The out of budget category refers to suggestions that would result in major changes to the code, costly to implement.

| Modifications and Action taken for Version 1.1 | |
|---|---|
| Functionality | Implemented (Y/N) |
| Integrate the developer attributes and the keyword attributes so user may create a more detailed query | Y |
| Leave out the bank as it takes up valuable screen space | Y |
| Format details for better readability | Y |
| Include more usability problem attributes in the details area | Y |

| | |
|---|---|
| Highlight usability problems in the details area that are selected by the user in cases were multiple problems exist in the node | Y |
| Interface is too clickable referring to right-clicking for details. The details should be automatically updated when the users clicks a node | Y |
| Functionality to be implemented in the future such as print and save were removed from the interface | Y |
| Statistics area was removed. | Y |
| Individual result bars were added for each individual attribute. This enabled users to see individual results of their query as well as the combined results | Y |
| An option to view all selected problems at one time. At the time users could only see those problems selected on node by node basis. | Y |
| The ability to select individual problems in the keyword or developer area. | Y |
| The developer area should be re-named as the evaluator. The evaluator who found the usability problem. | Y |
| Overview area should only show top two levels of the UAF structure to allow users to see problem distribution clearly. | Y |
| Query attributes such as cost, importance, should be dynamic. Users should be able to add and remove all attributes that exist | N |
| Usability problems should be able to be associated with multiple evaluators | N |
| The tree layout should be dynamic in order to use space more efficiently. | N |
| Ability to zoom-in/out in the overview area. | N |

**Table 4: Suggested changes for Version 1.1**

All the suggested changes in Table 4 were listed because they were deemed to be valuable additions or usability issues that needed to be addressed before the formative evaluation was conducted.

The results of the changes are illustrated in Figure 34 and described briefly in the following sections.

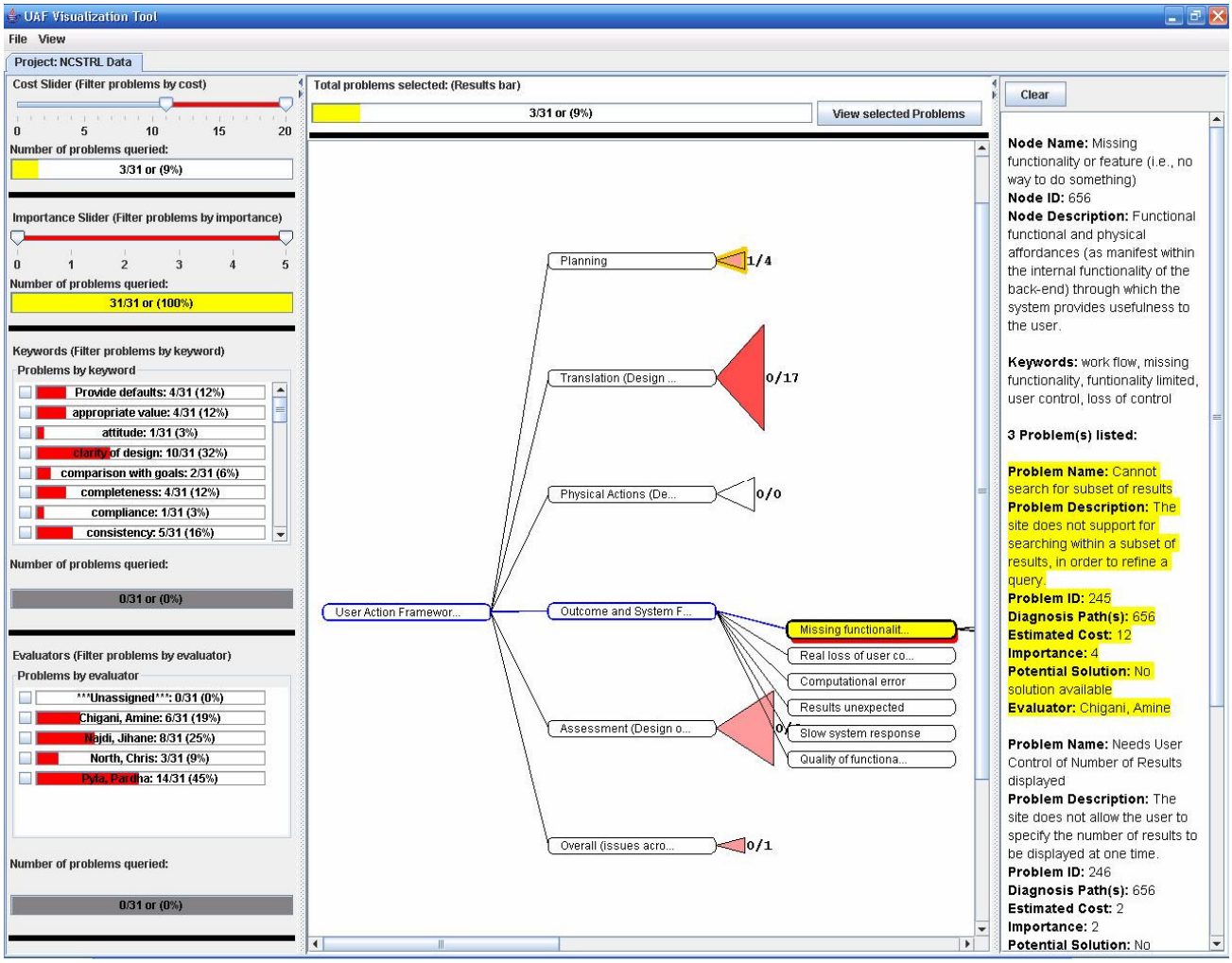



**Figure 34: Screen shot of Vizability with the suggested changes implemented**

### 3.4.1 Modifications to the Overview Area

The Overview Area gives the user an initial view of the usability data so the presentation of this datum collection is very important. This area was refined in several ways. The first way was the layout. The initial layout of the UAF was reduced from 3 levels to 2 levels (Figure 35). This was done so users could get a better feel of problem distribution under the 6 main nodes.

Next, the Overview Area was improved to have a more solid color mapping. The previous iteration we implemented using four different colors: blue represented the user's current path, purple represented specific selected usability problems, yellow represented

selected nodes, and we used seven shades of red to represent usability problem groupings (in the node or sub-tree).

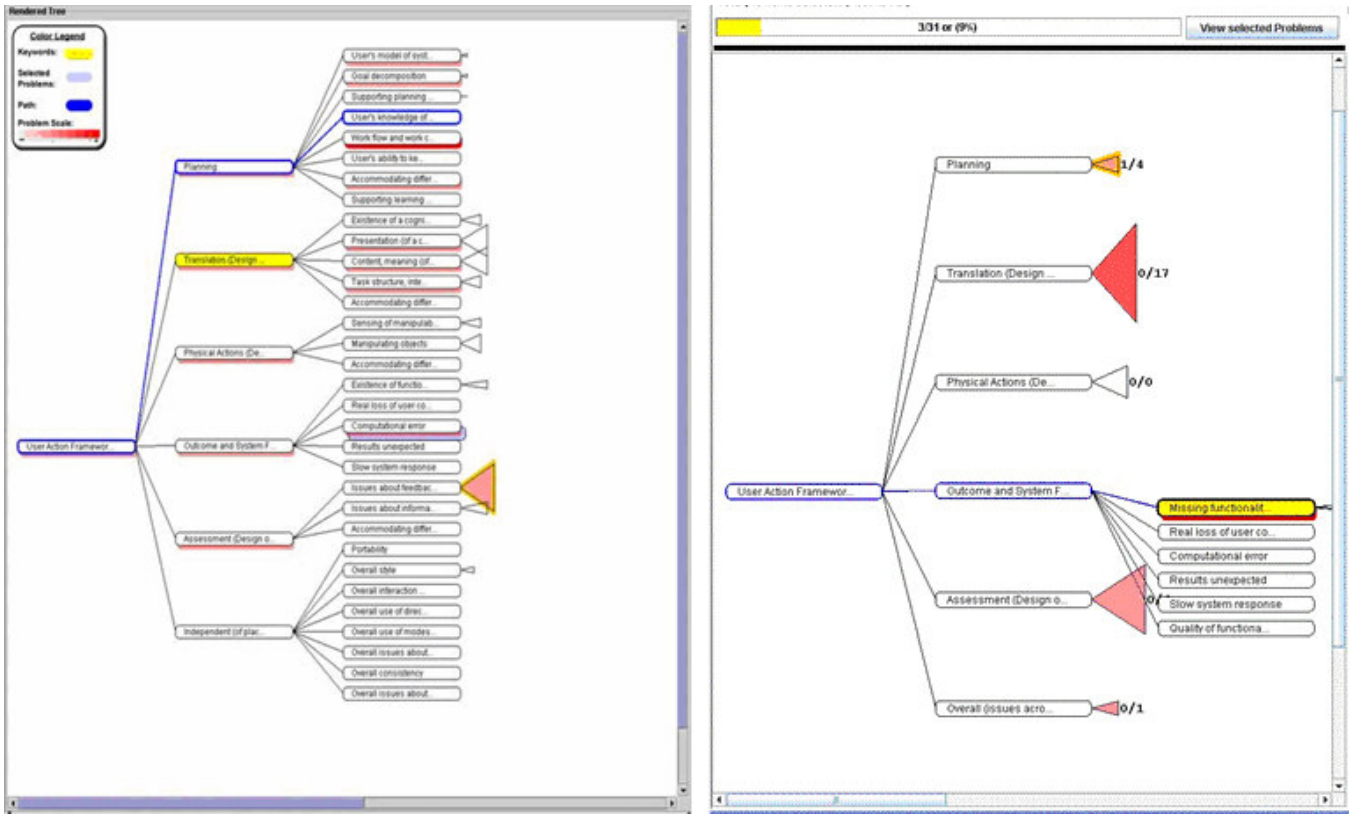



**Figure 35: New simplified color mapping made it easier for users to identify necessary usability problems and nodes in the visualization. The numbers at the base of each triangle enables users to get the exact number of selected problems in the sub-tree as well the total number of problems in the sub-tree.**

The new Overview Area uses blue, yellow, and red (reduced to four shades). Yellow and red still represent highlighted items and usability problems respectively. This color mapping is consistent across the whole tool. Blue represents the current path of the user. The blue path is closely coupled to the Details Area because with every click the details are updated with the information of the node in the path of user. This means that the blue highlighted node's details appear in the details area.

The third area of improvement is the numbers. The triangle shading allows for users to quickly scan the tree and pick out usability problem clusters but for users who want to view parts of the tree in more detail numbers were added at the base of each triangle.

These numbers were added for users who wanted an exact number of problems in that particular sub-tree. The number are written in the form "M/N" where "M" is the number of usability problems selected in the sub-tree and "N" is the total number of usability problems in the sub-tree

Finally, the functionality to view all selected problems at one time was implemented via the "View Problems Selected" button. Previously users could select problems but then could only view those problems one at a time. With the new functionality implemented users can select usability problems and have easy access to the list of problems that matches their selection. Now a user can copy/paste their custom list of problems into an email or report.

The results bar was also added to the Overview Area. It represents the number of usability problems that match the user's query. The results bar will be discussed later in the Problem Selection Area.

### 3.4.2 Modifications to the Problem Selection Area

The Problem Selection Area was simplified to one section. Previously, as shown in Figure 36, the Problem Selection Area was separated into two tabs. The features on the tabs were the same except one tab had a developer section (later changed to represent evaluators), and the other tab had a keyword section. The integration of the two tabs gives the user more "query power" by allowing the user to query both keywords and the evaluator. The integration also simplifies the overall usability of the tool by combining similar function components.

The integration of the tabs forced the need to remove the banks that allowed users to filter keywords and/or evaluators. The reason they had to be removed was due to screen space. It is recommended that the banks be implemented in future version as an advanced user feature.

The other notable change adding an attribute results bar to each attribute. In Figure 36, all the results bars are yellow. The results bar indicates the number of usability problems selected for that particular attributes current criteria. For example, the importance

attribute in Figure 36 is currently set to an importance rating of 4 and the results bar for the importance attributes shows there are 7 usability problems that have an importance rating of 4. Note that the other result bars represent the number of usability problems.


**Figure 36: The Problem Selection Area was simplified by integrating the two tabs into one. Also the result bars were added to give the user more insight into the details of their query.**

### 3.4.3 Modifications to the Details Area

The Detail Area was modified slightly but the modification provided huge gains in user satisfaction and understandability. The first improvement was to increase the font size so users could see the text more readily. The second enhancement was to change font to Helvetica to improve readability; the headings such as Node Id were also formatted in bold in order to separate the information.

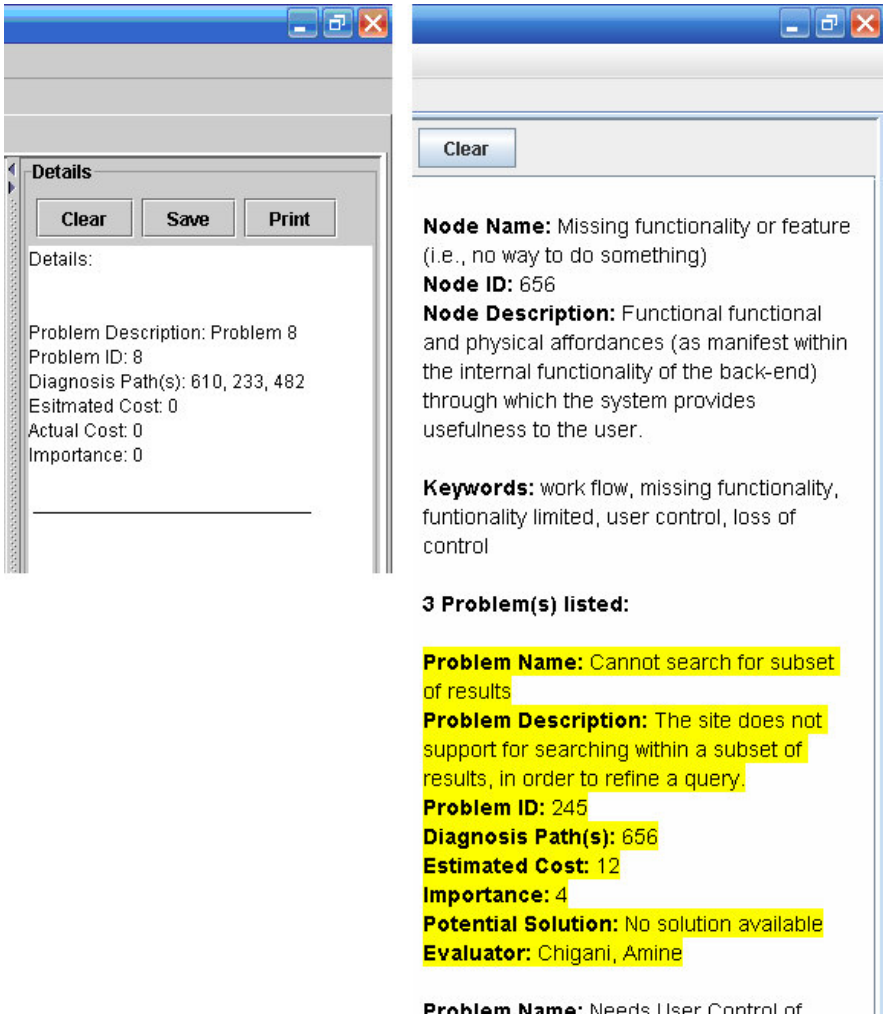



**Figure 37: The Details Area was enhanced for easier readability and understandability. Usability problems in the details area that match the user criteria is highlighted in yellow to distinguish the problem from other problems that may be listed.**

The Details Area was also implemented as to highlight the details of usability problems that match the user's query. This was done in cases were the user is browsing the UAF

structure and views the details of a highlighted node with multiple usability problems. The node in the Overview Area is implemented to be highlighted if only one of the usability problems matches the user's selection. So in order to clearly display the selected usability problem in the Details Area the text of the usability problem is highlighted in yellow to distinguish it from other usability problems that may appear in the Details Area.

## 3.5 Tool Architecture

Vizability is a Java 1.5 application. It is composed of approximately 5,000 lines of code and is implemented according to the Model View Controller (MVC) architecture. Because of the MVC architecture Vizability is scalable to accommodate future needs. The model as seen in Figure 38 is UAFData. UAFData maintains all the data such as usability problems, keywords, and evaluators. These data are then referenced by the DetailsPanel, JTreeTwoLevels, and the GraphicalTreePanel classes. All these classes display the data in a different way so the user may transform the data into more useful information. Vizability 1.1 is the third and most current prototype that visualizes the User Action Framework.

Vizability is implemented to read the usability data from two different MS Access databases. The first database is the User Action Framework structure. The second database contains usability problem information that is entered through DCART. When Vizability is executed the first database containing the UAF structure is loaded into Java Vectors. These data includes the node names, node descriptions, the node's respective child order, the node ID of the respective node's parent, and the keyword(s) associated with each node. These data are used to render the hierarchical structure that can be seen in the Overview Area (Figure 34).

The second database that is loaded contains the usability problems. The usability problems are the main object in our visualization and are represented by the java class, ProblemModel. Vizability is implemented so the ProblemModel stores all information about itself. The problem stores its associated nodes, keywords, as well as all problem attributes such as, cost, importance, problem description, etc. (the complete listing of

problem attributes can be seen in Figure 16). All the attributes data associated with a specific problem are used to create a ProblemModel object. These objects are then stored in a Java Vector. It is noted that this implementation differs from the User Action Framework concept in that usability problems are not associated with keywords. Keywords are only associated with nodes.

All usability data are read from the database only on the initial startup. Once the data are processed it is loaded into Java Vectors and the interface is constructed. Vizability is a Dynamic Query based visualization (Ahlberg & Shneiderman, 1994). Users are able to query any of the four attributes in the visualization and updates to the interface are immediately seen. This is done by communication to the model represented by the java class UAFData. When the user performs a query, the criterion of the query is communicated to the model. The model then searches the usability problem vector marking usability problems that match the criteria as *selected*. Once usability problems are marked as *selected*, the model then communicates the UAF structure in the Overview Area to update itself. In the process of updating itself each node has a list of usability problems it contains. If the nodes contains a usability problem that is *selected* it is highlighted in yellow to distinguish itself from other nodes.

Vizability is a complex application however; the performance of Vizability has not been optimized since it is only a prototype. Currently it is being executed on a desktop with a Pentium 4, 512 MB ram, and an on board video card and there are no significant performance issues. As usability problem sets increase in size and system overhead grows efficiency must become a priority.

Vizability is a prototype that will be integrated into the UAF suite of tools. It can operate as an integrated tool or as a stand alone application. The only requirement is it must have access to the problem database. Minor changes have to be implemented in order for Vizability to read from a user chosen database; currently the databases are hard coded.

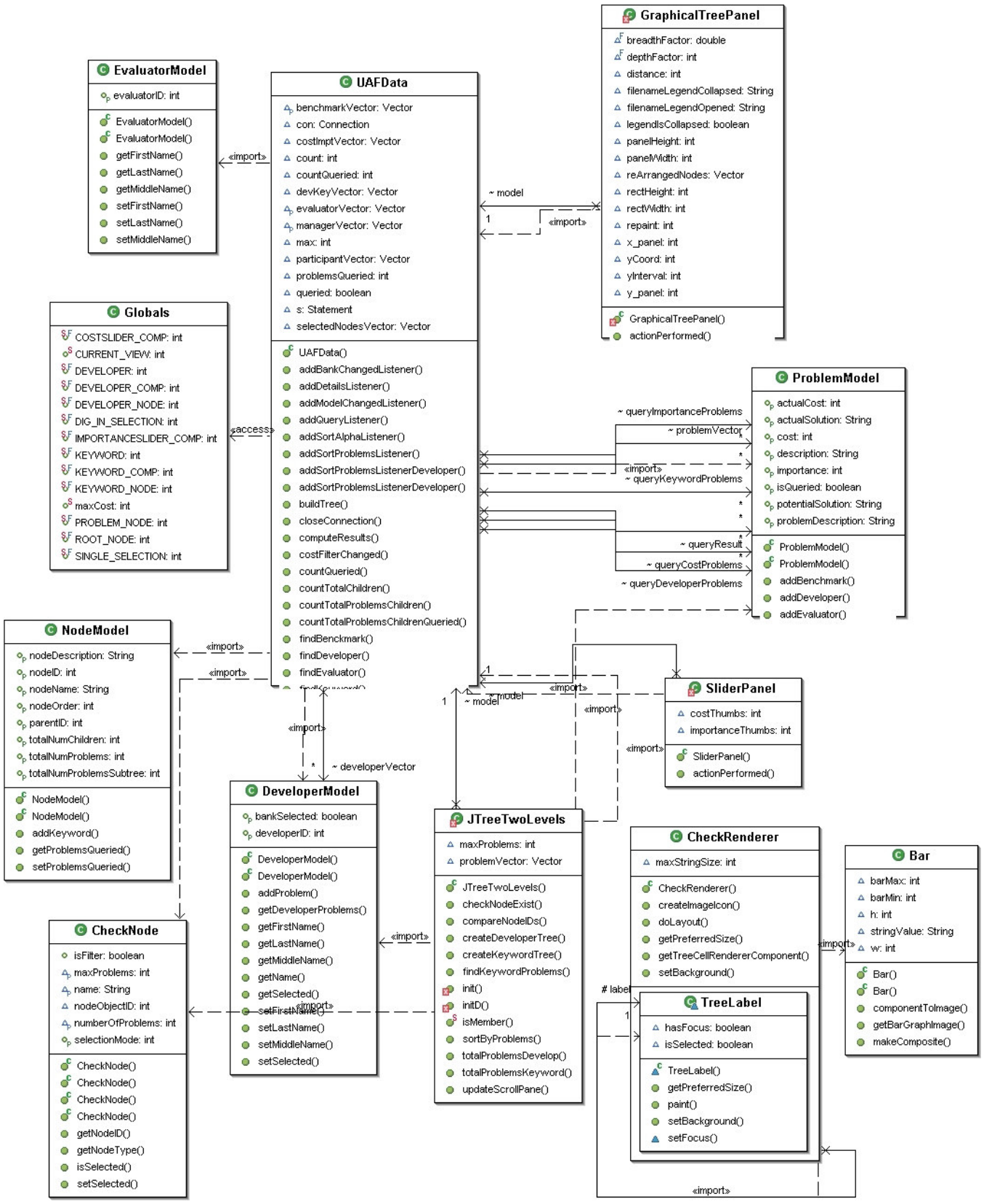



**Figure 38: Class diagram representing the major classes of Vizability**

## 3.6 Tool Usage

This section describes some possible uses and user classes for the tool as well how the tool relates to the other usability tools.

### 3.6.1 Usage Scenarios

The following usage scenario describes and illustrates how Vizability can facilitate the development effort.

**User Class: Usability Engineer**

A software development company is under pressure to release their new product. The project manager has stated they have one week's time and can devote two full time developers to the project. This means they have 80 hours to fix usability problems (40 hours per week for each developer). There are currently 31 usability problems that have been identified. The usability engineer in charge of the project uses Vizability to query low cost problems so they may be fixed quickly. The engineer finds that there are 26 problems of a cost 5 or less.

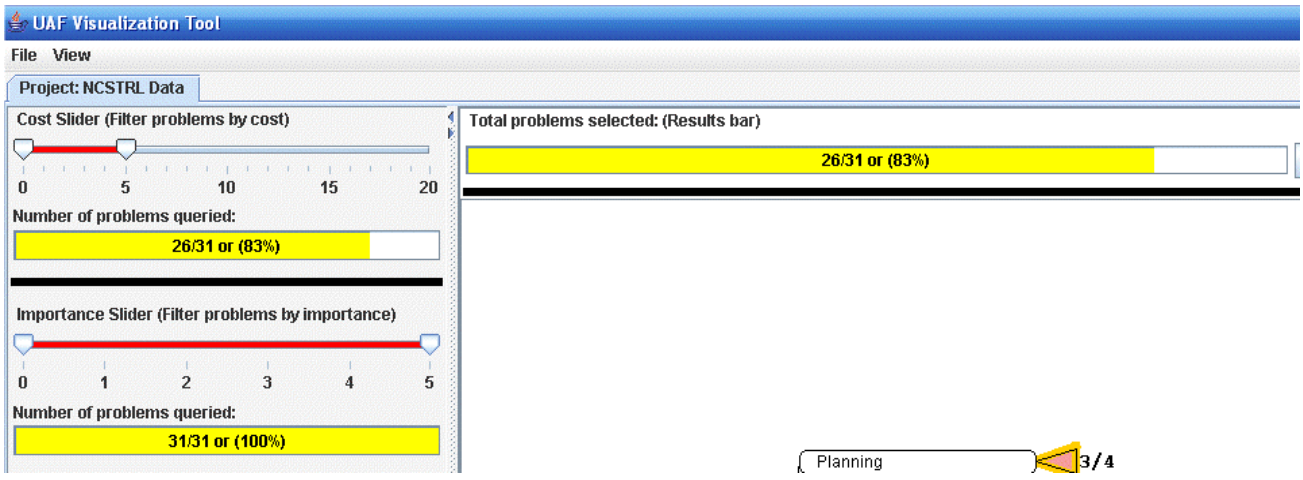



**Figure 39: The results bar on the top of the screen shows the user that there are 26 usability problems that fit the cost criteria of 0-5 person hours.**

To quickly estimate the time needed to resolve the 26 usability problems the engineer multiplies the number of usability problems (26) by the maximum cost (5) and finds 130

hours are needed to fix all the problems. The company doesn't have the luxury of being able to fix all of the low cost problems. The engineer then uses the importance slider to filter out the less important usability problems in order to get the most out of the available time.

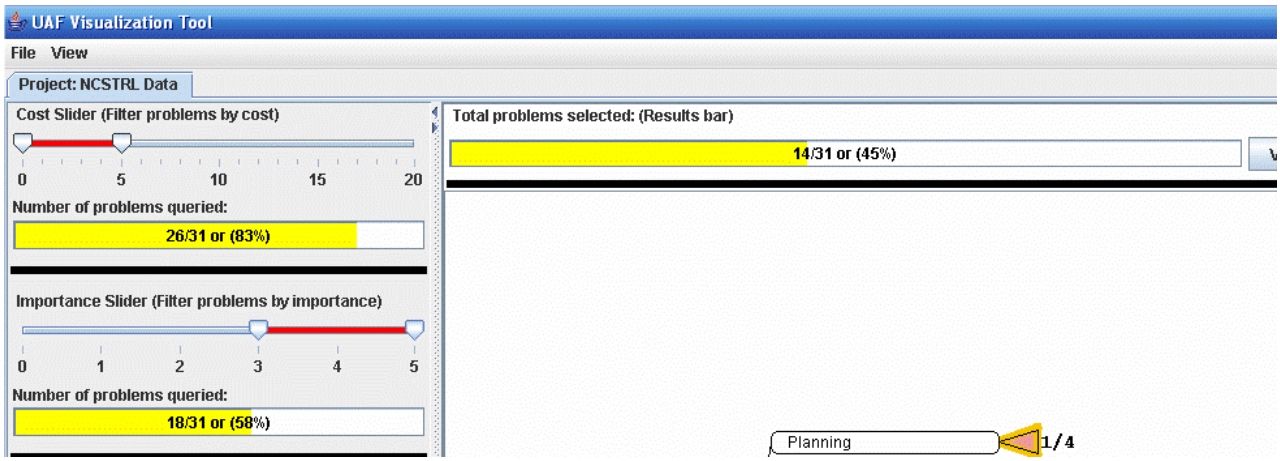



**Figure 40: The results bar on the top of the screen shows the new query of using both the cost and importance attributes. The user is still able to see that there are 26 usability problems that fit the cost criteria of 0-5 person hours by viewing the yellow bar underneath the cost slider. Added to the query is importance, the user queries usability problems from 3-5 resulting in 18 usability problems. However shown in the results bar on top there is 14 usability problems that match both cost and importance criteria.**

The new query, a cost between 0-5 and importance of 3-5 results reduces the problem list to 14 usability problems. The engineer estimates the hours it will take to fix the resulting 14 problems again and multiplies 14 usability problems by the maximum cost (5) to get 70 hours. The resulting set of usability problems fit into the allotted time with some time to possibly fix more problems. The engineer gets the usability problem set by clicking on the "View Problems Selected" button and emails the list of problems generated by the query to the two developers.

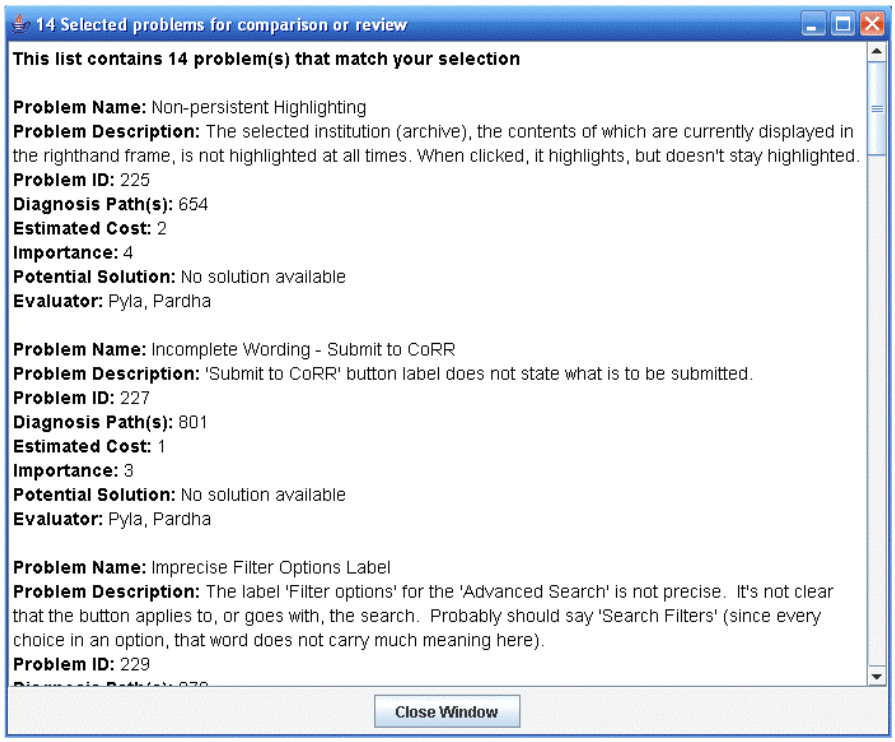



**Figure 41: The list of all usability problems that match the user defined query. The user may copy/paste or customize the list in the window as needed.**

After the usability engineer passes the usability problem set to the developers they can start fixing the problems. The usability engineer then spends some time and investigates the complete usability problem set in more detail. The investigation reveals that 58% of the problems are associated with wording and labeling.


**Figure 42: The results bar at the top of the screen show 18/31 or 58% of the usability problems deal with the keywords checked on left side (Keyword area). The keywords checked all deal with wording. This insight gives the user the ability to improve the process rather than just the individual usability problems.**

Realizing eliminating 58% of the user interface problems can substantially improve any system. The engineer is able to drill down to the problem in the process that contributed to the usability problems. The contributing factor turned out to be deficit requirements analysis. In the next company meeting the engineer is able to confidently bring this to the attention of the usability engineering team. The team was later able to discuss the issue as a group to improve their requirements gathering practices.

The following sections describe additional scenarios that demonstrate important uses for Vizability and different user classes of the tool.

**User Class: Usability Engineer**

Chris is a usability consultant with 10 years experience. He has been hired by CuttingEdge Inc. to help resolve usability problems found in the new product the company is developing. Upon being hired Chris uses his expertise to do a heuristic evaluation of the software. The evaluation generates many issues that he documents using DCART. Once the data are loaded he opens Vizability and loads the problem data into the tool. He immediately sees how the usability problems are distributed over the UAF structure. Without the visualization this would have to be done by spreadsheet, or by hand; grouping similar usability problems together. He analyzes the visualization and begins to write up obvious issues he can infer from the problem distribution. Chris then asks to review past usability evaluations. He loads some of the usability problems into the visualization. The visualization now allows him to view multiple usability efforts over time. He finds that some usability problems occur across multiple products. And those problems are caused because they are not conducting proper requirements analysis. This is able to be seen because most of the reoccurring problems are due to a lack of functionality. If requirements were properly conducted this functionality would have been included in the final product release. Upon finding this out he is able to recommend solutions for the new product he was hired to evaluate and also is able to formulate a recommendation to the company for the companies overall usability and development process. The report he is able to generate list potential solutions to fix current usability problems as well as some suggestions to stop usability problems from re-occurring in the future. Vizability enables him to save valuable time by allowing him to drill down on

individual usability problems, usability problem areas, and apply past knowledge to current product developments. Without Vizability the information would have existed but would not have been able to be used as effectively. Chris is able to give a comprehensive report to CuttingEdge representatives within 3 days.

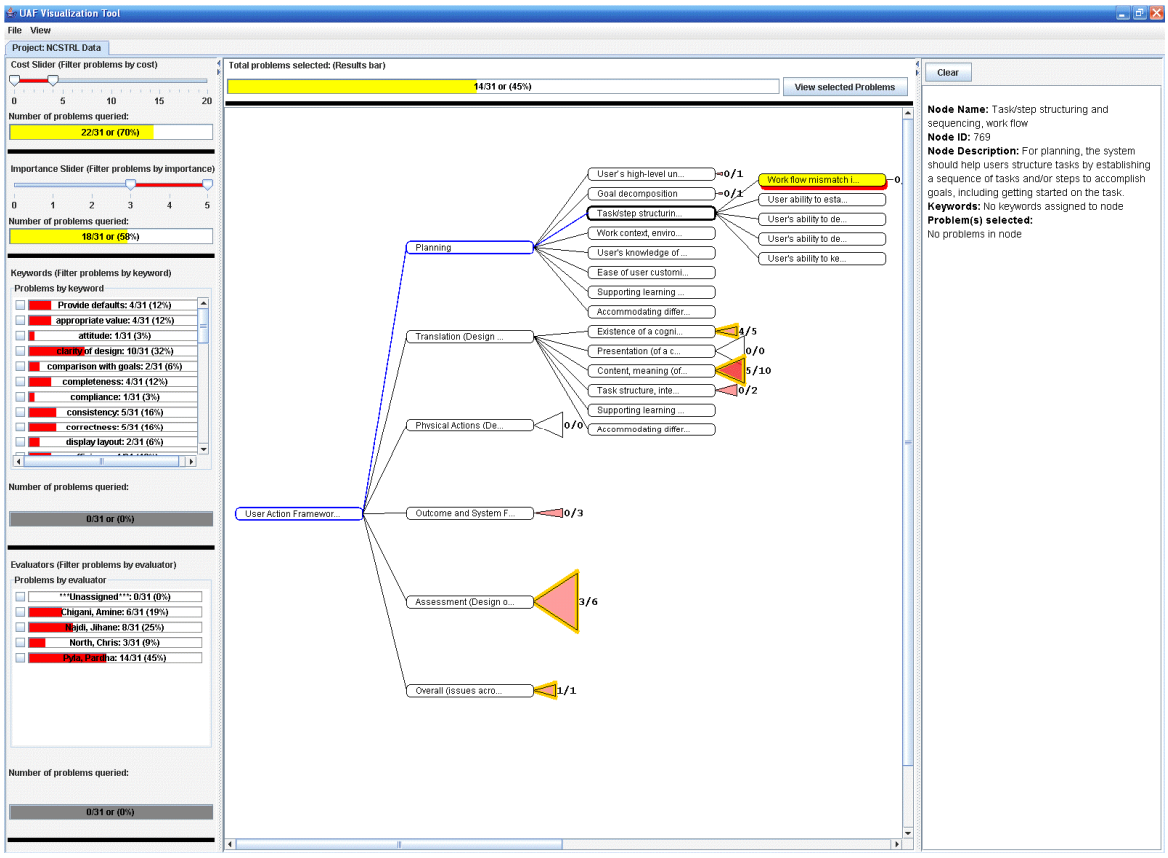



**Figure 43: Chris is able to see the distribution of usability problems and make recommendations to the company based on his findings. Later he loads more data representing multiple projects in to find reoccurring usability problems.**

## User Class: Developer

Greg is a programmer currently developing WebConference for Tech Co. He has been an independent programmer for 9 years hired by companies on a project basis. He loves the job because the freedom it gives him to travel and be with his family. However, what he dislikes the most is reviewing usability evaluations. He usually emailed a PDF file or MS Word document asking him to fix as many usability problems as he can by a certain deadline. The deadline is usually not too long after he receives the document. A major

portion of the allotted time given to him to fix the problems he must dedicate to understand the report.

When he received an email from Tech Co. He realized it was time to fix problems. Not looking forward to the email he was intrigued to find a link to a tool they use to visualize their usability problem data. The link was customized to his view, only the usability problems in the component of the tool. After reviewing a short illustrated description of the visualization tool, explaining how it is used. He clicked the link opening a web-based visualization tool, Vizability. His initial look gave him an overall picture of the all the usability problems. Although he was not familiar with the UAF tree structure of the visualization he knew where the problems were located. He went straight to the node with red shading and found the details. He was impressed with the details supplied to him, as they were in a standardized format (provided by DCART), giving him a proposed solutions as well as past solutions for similar problems.

This new visualization allows Greg to spend the majority of his time fixing problems rather than reading and understanding documents. The visualization allowed Greg to see the overall usability problem patterns and relationship. A "big picture" he was not able to see over the past 9 years he has been programming. He noted that some of the usability problems he has been repeating for many of his past projects. These reoccurring usability problems not only cost him time to fix them but also the companies are also taxed resources.

**User Class: Manager/Administrator**

Bob is the Manager responsible for company cost. His company has developed a CRM tool. The tool is marketed to sales companies of small to medium size. To do this he starts with support cost knowing cutting the need for support will provide long term gains for the company. He gathers phone support records and finds 30% of callers are calling about creating new customer account. An essential component to the software as its job is to track customer accounts. With this new found information he quickly looks into the usability data collected during evaluations. He loads all the usability problems into Vizability. He views the problems by software component. He selects to view only the

problems dealing with the "New Customer" component. He finds some problems of high importance that were not addressed due to their cost. The importance ratings of all the problems were 4s and 5s. These usability problems were not addressed because they had such a high cost. But after learning this he realizes that the cost of the usability problem may insignificant when compared to the continuous cost of supporting the problem. Bob knows he must make a decision now that he has good detailed facts of a critical situation. Vizability enabled Bob to view the problem set as a whole in order to see relationships that are not easily identifiable using convention reports or spreadsheets.

## 3.7 Impact on process

The scenarios above illustrate the usefulness of the tool and how the tool can help to eliminate usability problems from conception; impacting the total process (Figure 14) significantly eliminating usability problems and reducing the time needed to complete a quality product. However, the visualization tool presented in this thesis can be improved upon further to help users of the tool gain more insights into the data.

The current tool is able to visualize a database of collected usability problems. The usability problem database that is loaded into the tool is currently static and once loaded there is no mechanism to filter specific usability problems. The database viewed may contain usability problems from a single evaluation of a product, usability problems from multiple evaluations of the same product, or even usability problems from multiple products. This enables the users to see patterns and trends in usability problems in a single development effort or across multiple development efforts. Also, to aid users in comparing multiple development efforts tabbed views were thought to help users compare multiple development efforts in order to view problem clusters and trends. Users could then load different usability problem sets in different tabs and then by switching between tabs the user can compare the different datasets.

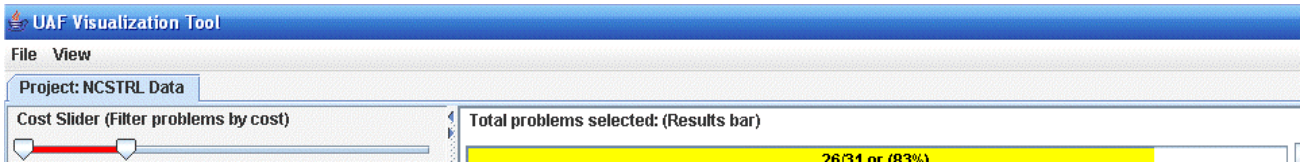



**Figure 44: Tabs can be used to view different sets of usability problems. Currently there is only one tab: "Project NCSTRL Data". By using multiple tabs users and loading different usability problem sets users can compare development efforts by switching between tabs.**

These methods are effective as they are right now; however they could be improved upon by giving the user a dynamic query mechanism to interact with the time attribute of the usability problem data (Ahlberg & Shneiderman, 1994). For example, time could correspond to the version of the product and the associated usability problems found during the evaluation of that version. In both cases allowing the user to interact with the time attribute would allow the user to evaluate the usability process over time in a more dynamic and effective way.

One method to give the user more dynamic interactions with the time attribute can be to incorporate an additional double knobbed slider into the interface whose ranges can be dynamically created based on the time attributes of the loaded usability problem dataset. This would give the user the ability to then view usability problems in a specific time range.

This more direct interaction with the data can be better illustrated in the following short scenario:

> Bill works for Tech Company and is the project leader. Bill is currently in charge of the development of a web-based system. Currently, the team is on their 3rd iteration of the product and has performed evaluations at each stage, identifying usability problems. Bill wants to ensure the system is maturing and becoming more stable. To do this he uses Vizability with an integrated time slider. He knows that most of the problems identified in the beginning stages of the web-based system should deal with high-level usability problems such as planning or translation. However, as the system matures the usability problems that are identified should move into the leaf nodes of the UAF representing more detailed and specific low-level problems. Having this knowledge Bill first filters out the later usability problems identified in the later versions (versions 2 and 3). When

he does this he notices that indeed the problems still deal with the high level concepts, raising a red flag in his mind. His next step is to filter out the usability problems identified in the first version and then see only those in the most recent version. Bill expects to see the problems of the first version fixed and perhaps see newly identified problems dealing with more specific usability issues. However, when he filters the first version's usability problems (so only the usability problems from the most recent version are being displayed) he realizes that many of the same problems that should have been addressed still remained. So as the project leader he knew he needed to understand why the usability problems where not being addressed across versions. Further investigation into this issue identified by the visualization allowed him to pin-point the issue. It turns out the programmers tried to address the issues themselves without the collaboration of the usability engineering team. He found that the current development process didn't implement integrated meetings of the two groups (usability engineers and programmers) regularly enough and in a timely fashion so issues could be addressed properly. Bill recognizing this oversight in their process now schedules regular meetings between the programmers and the usability engineers.

The current visualization tool is a great start but it is not final. There is more work that can be done to understand how the current tool can be improved upon to provide users more capabilities when needing to evaluate the development process over time.

# 4 Formative Evaluation

A formative evaluation of the tool was performed to accomplish three goals:

- Find out if the tool enables users to find insights documented in the requirements (Table 1)
- Evaluate the usability of the visualization tool
- Identify future needs for the tool: features, design, suggestions, etc.

These goals were accomplished via the following actions:

- Facilitated insights: Task 2 and task 10 instructed the participants to explore the tool freely commenting on any insights, suggestions, or issues.
- Usability of tool: Task 1 and Tasks 3-9 tested the functionality of the tool. The results of these tasks will be discussed in the Formative Evaluation section.
- Identify future needs: Participants were instructed to talk freely throughout the experiment. All relevant comments, suggestions, or issues were documented. Participants were also asked to fill out an exit questionnaire to gain more knowledge future possibilities of the tool.

## 4.1 The usability information

The usability information used in the evaluation included 31 usability problems. The usability problems are based on the findings by Hartson, Shivakumar, et al. in there case study of a digital library system (Hartson, Shivakumar, & Pérez-Quiñones 2004). The problems included the following usability problem attributes: UAF classification, usability problem name, and usability problem description. Some problems also included proposed solutions. The data, however, did not include a cost rating, importance rating, or evaluator (the person who found the problem). Since these are main attributes of Vizability those attributes (cost, importance, and an evaluator) were amended to the usability problem data set. The importance ratings were assigned based on the problem descriptions. The cost value was based on the proposed solution and best guess

estimation in person hours to implement the solution. If no proposed solution was provided a best guess estimation of the cost to fix the problem was based solely on the problem description.

All UAF nodes that contained problems were assigned a keyword. The keywords are used to identify usability problems by a different dimension than the UAF. For example, usability problems distributed in different locations of the UAF could all be associated with the keyword "missing functionality." The use of keywords in the UAF can be analogous to affinity diagrams.

## 4.2 Benchmark tasks criteria

Benchmark tasks were developed to test the usability issues of tool. The benchmark tasks are listed in the appendix. The tasks were created according to the following guidelines (Hartson & Hix, 1993):

- The tasks are specific and state what the user should do rather than how the user should do it.
- The tasks were developed to test the performance of the tool and not the participant.

The tasks were designed to test the majority of the tool's features: overall result bar, individual results bar, triangles shading and yellow highlighting, numbers, navigation, and the construction of queries.

The participants were asked to perform 10 tasks. In 8 of the tasks participants were asked to perform a specific activity and 2 of the tasks were free use task. Free use tasks can be valuable for revealing participant and system behavior that might not have been predicted by the evaluator (Hartson & Hix, 1993).

## 4.3 Selection of participants

The participants were carefully selected to represent "real" users of the visualization tool. Real users of the system will have usability engineering experience and be familiar with the User Action Framework. This criterion was met by asking graduate students who

have taken the Usability Engineering course (CS5714) taught by Dr. Rex Hartson. All the students in the course were introduced to the User Action Framework and have experience with usability engineering.

The number of participants selected for the evaluation was based on Neilson (Nielsen & Molich, 1990). Our evaluation included six participants; 5 males and 1 female, ages 23-33 years old.

## 4.4 Usability lab setup

The usability lab was setup so the participant and the evaluator were located in separate rooms; McBryde 102A and McBryde 102B. The rooms are adjacent to each other connected by one-way glass. This allows the evaluator to observe the participant from the other room and remain incognito while the participant performs the tasks.

Communication between the participant and the evaluator is a necessity in cases where the participant may have questions or comments about the task or system. The lab is set-up so the participant may talk freely and the sound is wired through a television so the evaluator may hear the participant. The evaluator may communicate with the participant via an intercom. This allows the evaluator to hear all comments made the participant while the evaluator can remain silent unless explicit actions are taken to talk to the participant.

Screen capture and voice recording was used to capture all data for later reference. This was done using the Morae Recorder (TechSmith, 1995). Morae Recorder is an application created to aid usability practitioners in usability evaluations. This application ran on the participants' machine while the Morae Viewer ran on the evaluator's machine. The Morae Viewer allowed the evaluator to view the screen of the participant in real time.

## 4.5 Formative Evaluation Procedure

Prior to arriving for the evaluation all participants were asked to view a 12 minute video tutorial of the visualization tool. The purpose of the tutorial was to help the participant

understand the purpose of the tool as well as the functionality. Since the tool is not trivial we wanted the participants to have some familiarity with the tool before beginning the evaluation. The tutorial proved to be successful in helping the users understand Vizability.

Upon arriving participants to the evaluation location, 102A, we explained the usability lab setup and approximate time of the evaluation. Then we gave the participants the informed consent form to read and sign if willing, an entrance questionnaire, and written instructions of the formative evaluation. If the participant had no questions about the evaluation the Morae Recorder was started, the participant was left alone in McBryde 102A, and prompted to begin the benchmark tasks.

During the evaluation participants made comments and suggestions about the tools features, "coolness", and possible improvements. Usability issues were noted and will be discussed in the Results section.

After the evaluation participants were asked to fill out an exit questionnaire to gather the users' subjective comments about the visualization tool.

## 4.6 Results

The results section will be divided in 3 parts. Each part will describe the result in relation to each goal of the study. The evaluation of Vizability provided great feedback. Some participants made the following comments during the evaluation:

"This makes me want to go through the formal process"

"…takes away the hassle of evaluating usability data making it easy and appealing to use."

"I can extrapolate a lot of information from it"

"Excellent visual understanding of data-makes it easier to see overall trends to address during development"

Vizability is an excellent start to the UAF visualization. The visualization was designed for usability practitioners however, while designing the visualization it was realized that another use could also be for administration and developers. The visualization could be extended to display specific information dependent upon the role of the individual user. By customizing the information the visualization could act as a communication medium between software engineers and usability engineers and even administrative roles as well.

### 4.6.1 Evaluation Goal 1: Open-ended Insights

Open ended insights were found by users in the initial task. Users were given five minutes to comment on any *insights*, comments or suggestions for the tool. The term *insights* is described by Saraiya, North, et al. as a "critical breakthrough that leads to discovery: suddenly seeing something that previously passed unnoticed, or seeing something familiar in a new light. (Saraiya, North, & Duca, 2004)" Insights are important because they are the facts that will help usability engineers and other users of Vizability to make both local project decisions and global company level decisions. In this section we will point out the insights that were able to be seen in the first 5 minutes of using the tool and viewing the data.

Insights:

- Keyword and usability problem relationship: One participant was able to immediately identify usability problems of type "clarity of design" and the distribution of those problems across the UAF structure
- Evaluator and usability problem relationship: User found what evaluator found the most problems, evaluating his effectiveness.
- Usability problems and type of problem: participant found that translation had the most low cost problems; mostly dealing with the keyword "word choice."
- Number of usability problems: could quickly identify the exact number of problems distributed across the UAF structure.

- Usability problems and cost: found problems with the highest cost were located in the Planning and Outcome and System node
- Usability problems and importance: found usability problems of the highest importance (5) distributed across Planning, Translation, and Assessment.

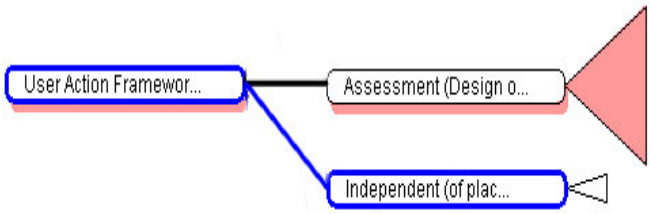
These insights are useful and needed to understand the big picture. Vizability enables users to interpret high level effects of the low level decision (i.e. requirements analysis, evaluations, prototyping, etc.) in the development process. Currently without the use of Vizability these insights often on unnoticed resulting in reoccurring usability problems and process weaknesses or deficiencies.

### 4.6.2 Evaluation Goal 2: Identifying Usability Issues

Listed in the table below are the usability issues that arose during the evaluation and a proposed solution. The solutions have not been implemented for this work.

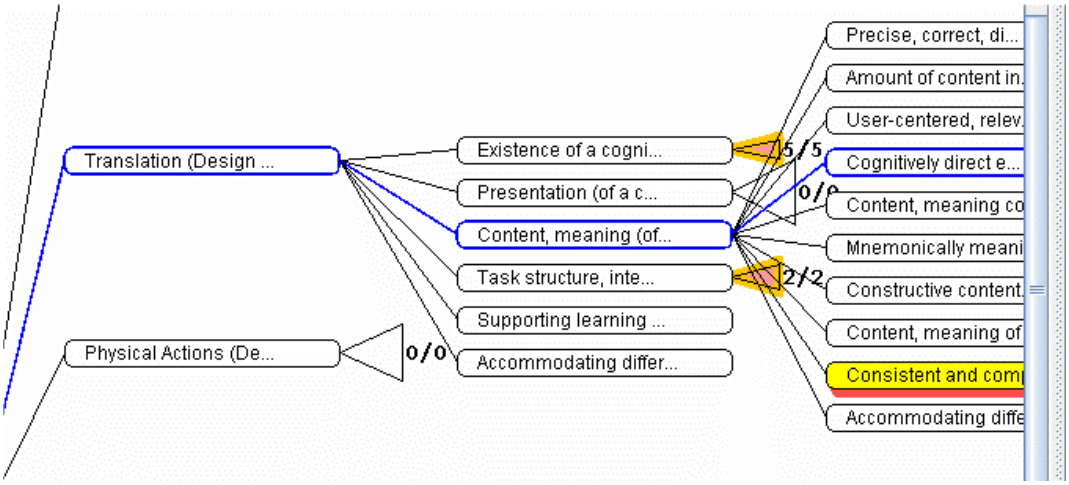
Usability Problems

1. The triangle attribute (height, width) were not intuitive for first time users. However, once introduced to users seemed to have very little trouble with the meaning of the attributes



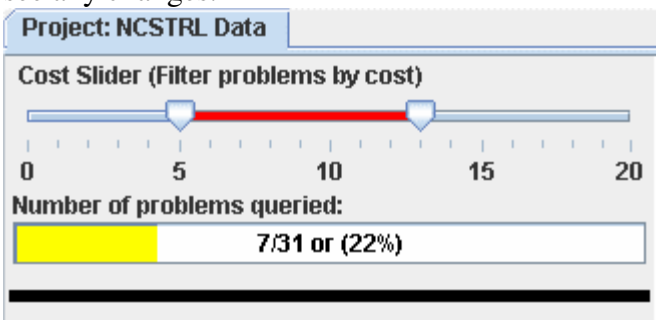

Proposed Solution:
Provide a legend on the screen that users may reference if they do not understand the meaning. The legend must be made apparent for users so they know there is a difference in the triangles. Therefore the legend option should be a preference that can be turned off for advanced users.
Estimated Importance Rating: 2 Estimated Cost Rating: 1

2. When users navigate through the tree and the tree opens, the tree location of focus extends past the viewable area of the screen; forcing users to scroll the 'x' scrollbar to view the tree structure



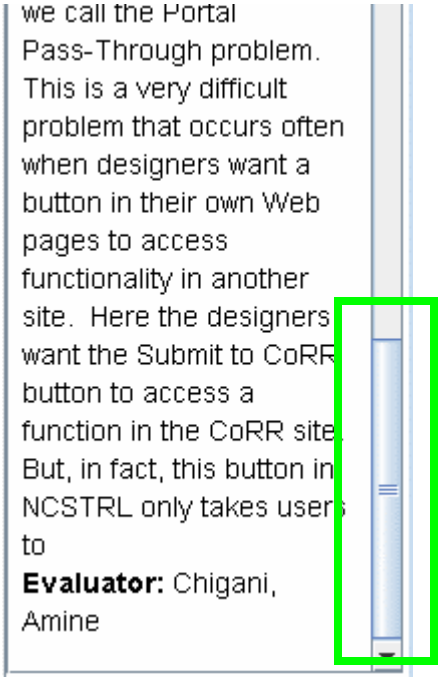

Proposed Solution:
Automatically scroll the 'x' scrollbar to the newly opened section of the tree.
Estimated Importance Rating: 3 Estimated Cost Rating: 5

3. The cost slider and the importance slider must be released by the user before the user may see any changes.



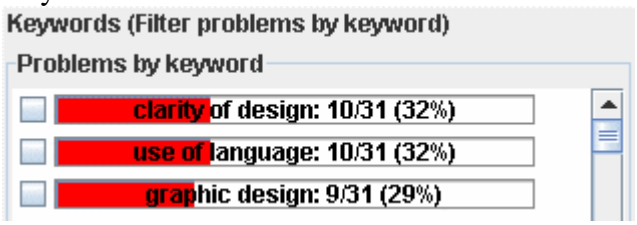

Proposed Solution:
Change the implementation of the slider bars so that moving the slider knobs are reflected if any change is made to the slider
Estimated Importance Rating: 3 Estimated Cost Rating: 1

4. The scrollbar starts scrolled to the bottom of the details area so users start reading at the end of the details rather than the beginning.



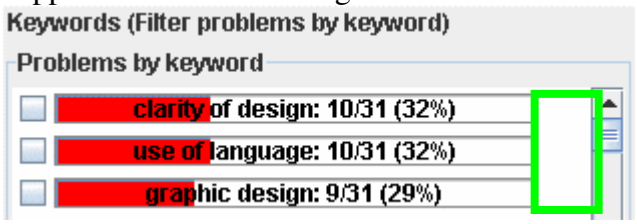

Proposed Solution:
Start the scrollbar at the top of the page.
Estimated Importance Rating: 1 Estimated Cost Rating: 1

5. Keyword and evaluator bar red with black text is hard to see.



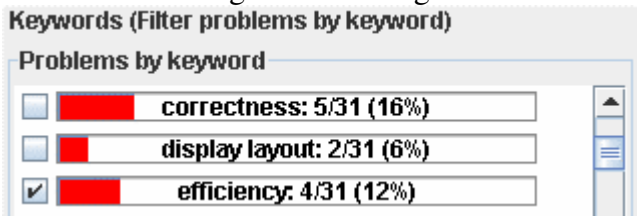

Proposed Solution:
One solution may be to dim the red color some so the black color is able to be more readable.
Estimated Importance Rating: 1 Estimated Cost Rating: 1

6. Right-clicking on white area (inside the green box) results in no action. The pop-up menu is only displayed when you right-click on a bar. Users were confused when nothing happened and could not figure out how to sort.



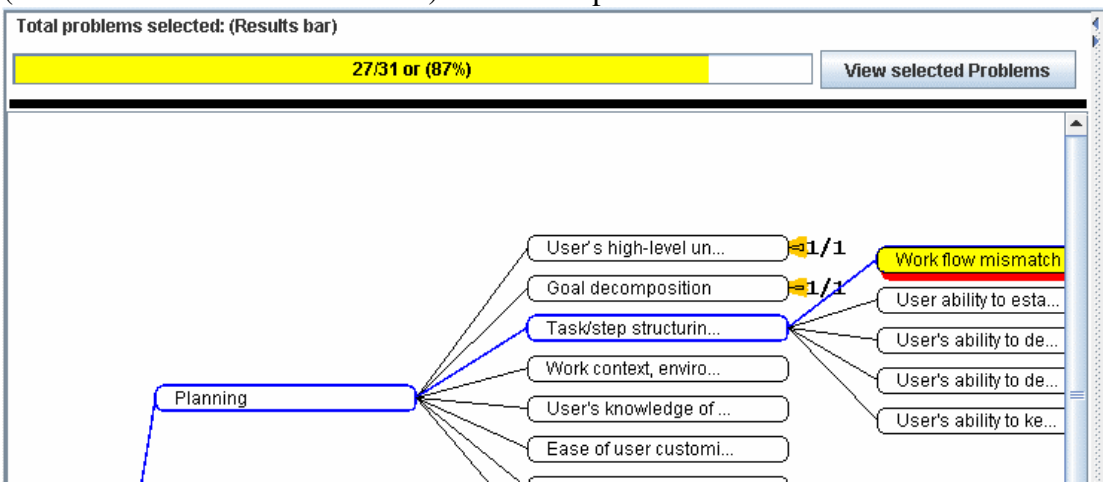

Proposed Solution:
Allow users clicking in the white area to view the pop-menu.
Estimated Importance Rating: 4 Estimated Cost Rating: 2

7. Clicking anywhere on the bar checks/un-checks the bar. Because of the bar length some users did not recognize the change in the bars status



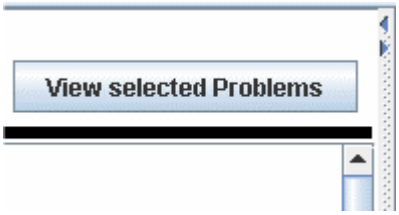

Proposed Solution:
Highlight the entire bar when the user clicks. This will give the user feedback when a change to the bar's status is made.
Estimated Importance Rating: 1 Estimated Cost Rating: 20

8. Some users felt the location of the Results bar should be in the problem selection area (underneath the evaluator section) rather than placed over the Overview area.



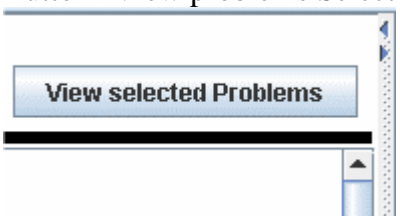

Proposed Solution:
This was a dependent upon the user and was a matter of preference. To address this issue an option should be given to the user as to where he/she would like the results bar to appear.
Estimated Importance Rating: 3 Estimated Cost Rating: 15

9. "View problems selected" wording is confusing since you are not selecting problems by the common use of the word "selected" via clicking a problem.



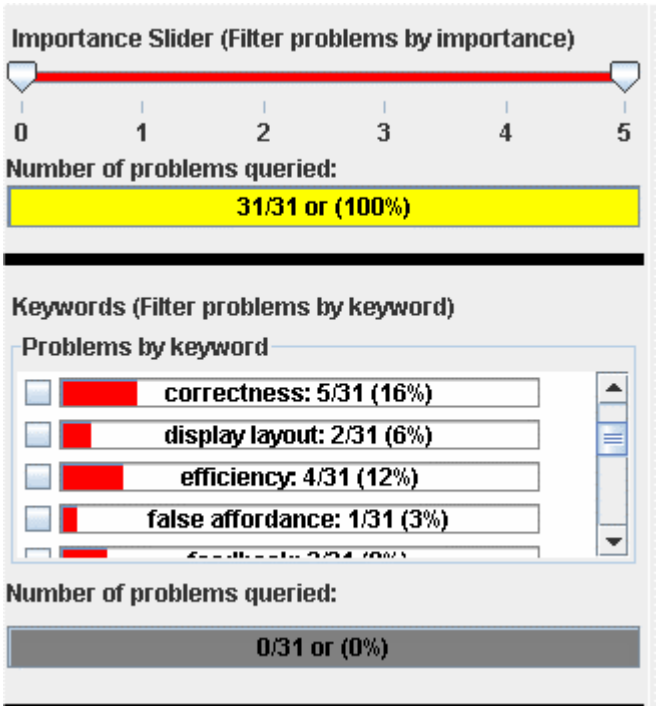

Proposed Solution:
The button should be worded as "View highlighted problems." This wording is in context with the rest of the visualization. As problems are highlighted if they are selected
Estimated Importance Rating: 2 Estimated Cost Rating: 1

10. Button “View problems Selected” is mistyped.



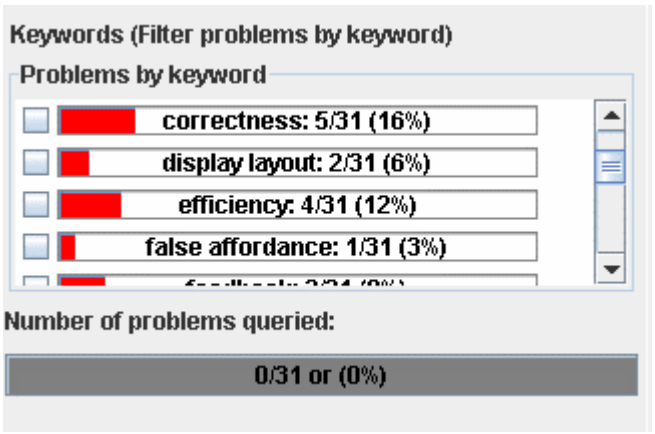

Proposed Solution:
Use all capital letters to begin each word: “View Selected Problems”
Estimated Importance Rating: 2  Estimated Cost Rating: 1

11. In the problem selection area there was no way for users to reset the query settings back to the default value.




Proposed Solution:
Provide users with a button to that can reset all attributes automatically.
Estimated Importance Rating: 2  Estimated Cost Rating: 10

12. In the checkbox areas (keyword and evaluator) there is no method to deselect or select all checkboxes



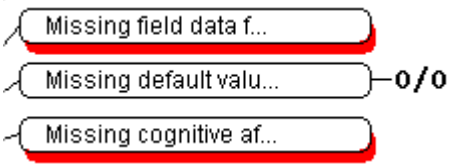

Proposed Solution:
In the title bar, "Problems by keyword" provide a checkbox that is labeled "check all" to give users the functionality to select/deselect all items
Estimated Importance Rating: 3 Estimated Cost Rating: 5

13. Some of the node's details appeared as HTML. The HTML is not understandable by the user.

**Node Name:** Mnemonically meaningful cognitive affordances to support human memory limits
**Node ID:** 405
**Node Description:** Use labels with cognitively direct meanings, rather than arbitrary encodings, to support <a href="glossary.asp#memory">human memory</a> limitations.
**Keywords:** No keywords assigned to node
**Problem(s) selected:**
No problems in node

Proposed Solution:
The details area should be implemented to display HTML. Future versions can then incorporate links to screen shots or video if need be.
Estimated Importance Rating: 2 Estimated Cost Rating: 5

14. Some triangles were too small to be identified if the sub-tree was small (e.g. consisted of only one node)




Proposed Solution:
Implement a minimum size of triangles. This will allow triangles to remain in context with one another and allow users to see the triangle as a triangle rather than a thin line.
Estimated Importance Rating: 2 Estimated Cost Rating: 1

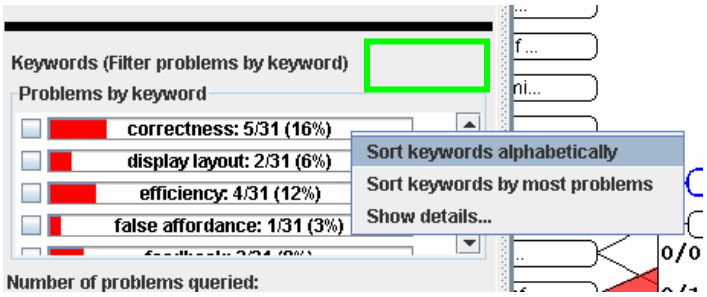
15. The details section can be hard to read if there are many problems located in the node

**Problem Name:** Hidden Default for Search Criteria
**Problem Description:** The default search (searching without specifying any search criteria) returns the entire contents in the archive of the first institution - an unlikely candidate even for a default. But nowhere does the user interface describe the default behavior, either before or after the search.
**Problem ID:** 243
**Diagnosis Path(s):** 649
**Estimated Cost:** 1
**Importance:** 1
**Potential Solution:** Make the default explicit by including a selection of an institution (archive) as part of the query with 'All' as a visible default. The 'Search for' field should also have 'All' as a visible default and have it selected (highlighted) so that any typed in
**Evaluator:** Pyla, Pardha

**Problem Name:** Default search is hidden
**Problem Description:** Searching without specifying any search criteria yields the entire contents in the archive of the first institution, but nowhere does the user interface describe the default behavior, either before or after the search. The default is hidden and it is not a good choice for a default, anyway.
**Problem ID:** 1053
**Diagnosis Path(s):** 649
**Estimated Cost:** 1
**Importance:** 4

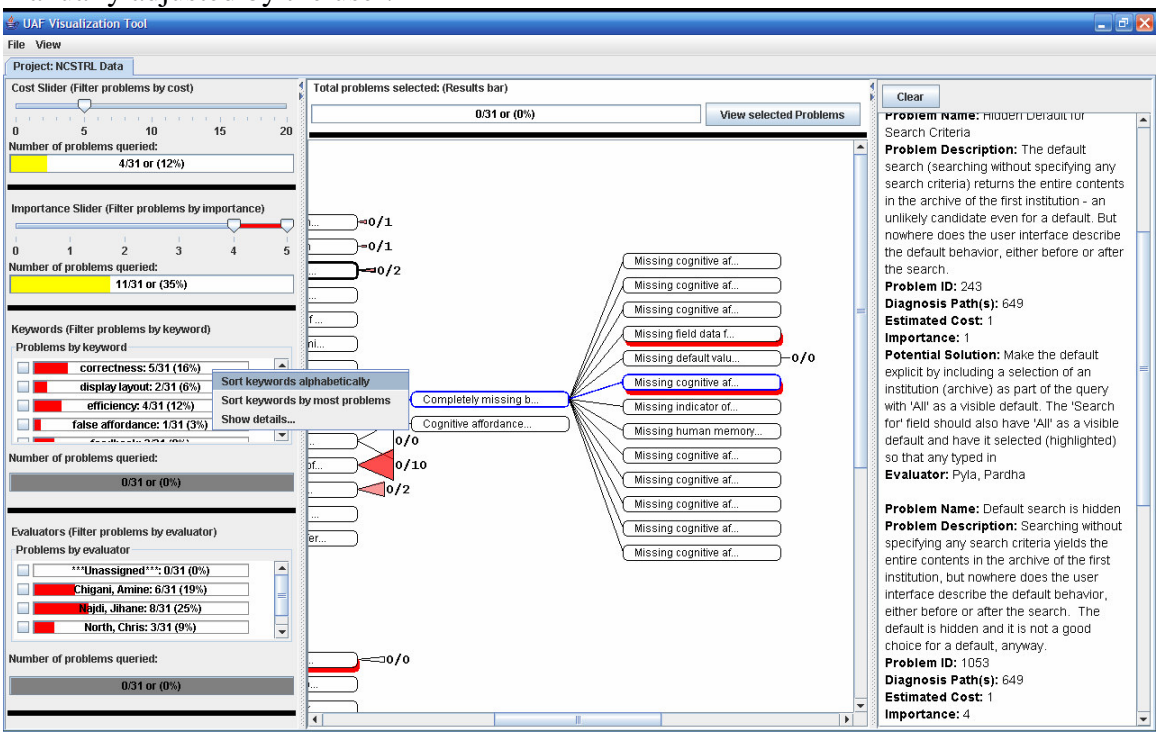
Proposed Solution:
The details area should be collapsible by section much like window explorer (+/-). A section for the node details, problems (if any), and selected problems (if any). This would allow the user to understand the details being viewed initially.
Estimated Importance Rating: 2 Estimated Cost Rating: 10

16. Lack of affordance for sorting




Proposed Solution:
Provide a drop down box in the area of the green box above containing the functionality available. This will give the user a visual component to illustrate the functionality
Estimated Importance Rating: 5 Estimated Cost Rating: 15

17. The screen space was not used efficiently. Screen space can be dynamically allocated based on the users' need. The adjustable dividers help considerable but they must be manually adjusted by the user.




Proposed Solution:
The UAF structure should be implemented to in order to show the maximum number of nodes as screen space will allow and change focus of the window ("x" scroll). This involves dynamic calculations of screen space available and nodes to be shown. Problem selection area attributes should be to "float" or "dock" this would allow users to adjust screen space more freely and allow for more attributes if need be.
Estimated Importance Rating: 4 Estimated Cost Rating: 50

**Table 5: Usability issues identified in the formative evaluation**

### 4.6.3 Evaluation Goal 3: Future needs, suggestions, etc.

The final goal of the formative evaluation was to solicit recommendations about features to be implemented, uses of the tool, and general comments. This section will list those items the users expressed:

- Allow a user to input a budget and integrate a prioritization table into the tool
- Provide a mini-version of the tree structure. This will allow users to compare the problems that have been selected vs. the original tree structure
- Save different filters so users may quickly view specific queries
- Allow for dynamic attributes. For example if a user does not wish to view "cost" the user may remove it from the Problem Selection Area.
- Allow users to search for a usability problem using a word(s) from the usability problem name or description
- Display problem details in text sections. Allow users to collapse sections of text.
- List node usability problems in the tool tip along with the node's description.
- Implement a more dynamic layout so the UAF structure uses screen space more efficiently.
- Have the option for an apply button. So you can set your attributes accordingly then click "Apply" and those results will be shown in the overview area.
- Use a table form to list the usability problems in the detail area
- Highlight the path to nodes that contain selected usability problems in yellow
- Visually relating problems to one another (e.g. Problem A is related to problem B and fixing one of the problems may fix both problems. Then this relationship should be visualized.)
- Have the ability to group keywords that are often used in combination so that users may check only one grouping rather than 2+ keywords.

Some of the suggestions were preferences of the user and should be integrated into Vizability as user preference. These items include:

- Change the red intensity of the triangles based on the user defined query.

- Allow the results bar to be positioned in a user defined location i.e. being able to float

# 5 Contributions, Key Findings & Conclusions

The contributions of this thesis are as follows:

- Identified a process that integrates visualization and the UAF to enhance to overall return on investment in a usability engineering process.
- Identified requirements needed for a visualization tool to facilitate and enhance the development process
- Designed and conceptualized a prototype that satisfies *ALL* gathered requirements
- Architected a high fidelity multiple-view visualization in Java 1.5 that allows users to find trend and patterns in usability information over a single or multiple usability efforts
- Evaluated the visualization to find insights, usability issues, and future suggestions

Significance of contributions:

The contributions: identifying requirements, conceptualizing, architecting, and then implementing Vizability; enhances the development process for both sides of the product: the developer, and the user.

The developer is able to benefit from the visualization in the following ways:

- Ability to gather a high level view of the usability deficiencies of a product by being able to see usability problem clusters, trends, and distributions.
- Knowledge accumulation helps novice developers to re-use concepts and experiences from past usability efforts.
- Process improvement over time by being able to see areas of weakness in the development process and allocate resources more effectively to save time and money

This also benefits the user by having access to a better and possibly less costly product.

## 5.1 Key Findings

The key findings presented in this document are summarized below.

The development and evaluation of Vizability has suggested that visualization can be a valuable resource to the UAF suite of tools.

Key Findings:

- Visualization of usability problems can be used for multiple user classes to help refine different parts of the development process. For example, the developer can have easy access to usability problems that need to be fixed, the usability practitioner can address deficit areas of the process, and the project manager can allocate resources more efficiently by capitalizing on the strengths of each evaluator.
- Vizability can be used as a communication aid between the software engineer and the usability practitioner. The visualization provides an easy and efficient way to exchange usability problem descriptions and corresponding costs to fix those problems. This effectively provides a communication mechanism to facilitate gaining insights from the software engineering life cycle to the usability engineer life cycle
- Vizability can lessen the need to exchange lengthy reports as it appears to be a viable substitute
- Vizability can be used to facilitate knowledge gained over multiple development efforts and direct that knowledge towards any one development effort so usability may be used early in the development process.
- Vizability can be used as a training aid for UAF users
- Vizability can be used as reference guide for users to lookup specific usability concepts and find associated usability problems. It can also be used a way to prove usability problems are indeed problems as sometime needed in real practice

## 5.2 Conclusions

Vizability has been in development for 3 years at the time of this writing. It has been through numerous small and several major modifications. The current version as described in this work is based on solid requirements and can be enhanced as described in the Results and Key Finding sections.

Making the suggested improvements to Vizability and further research can enhance the tool significantly. Vizability is an initial start to what can be a very powerful and useful tool for different user classes associated with system development.

# 6 References


Ackoff, R. L. (1989). "From Data to Wisdom." Journal of Applied Systems Analysis 16: 3-9.

Ahlberg, C. and B. Shneiderman (1994). Visual information seeking: tight coupling of dynamic query filters with starfield displays. Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence, Boston, Massachusetts, United States, ACM Press.

Ahlberg, C. and E. Wistrand (1995). IVEE: an Information Visualization and Exploration Environment. Proceedings of the 1995 IEEE Symposium on Information Visualization, Atlanta, Georgia, IEEE Computer Society.

Baldonado, M. Q. W., A. Woodruff, et al. (2000). Guidelines for using multiple views in information visualization. Proceedings of the working conference on Advanced visual interfaces, Palermo, Italy, ACM Press.

Bias, R. G., D. J. Mayhew, et al. (2003 ). Cost justification The human-computer interaction handbook: fundamentals, evolving technologies and emerging applications Lawrence Erlbaum Associates, Inc.: 1202-1212

Card, S. K., J. D. Mackinlay, et al. (1999 ). Information visualization Readings in information visualization: using vision to think Morgan Kaufmann Publishers Inc.: 1-34

Gray, M., A. Badre, et al. (1996 ). Visualizing usability log data Proceedings of the 1996 IEEE Symposium on Information Visualization (INFOVIS '96) IEEE Computer Society: 93

Guzdial, M., P. Santos, et al. (1994). Analyzing and visualizing log files: A computational science of usability. Presented at HCI Consortium Workshop.

Hartson, H. R. (2004). Visualization discussion. C. Catanzaro. Blacksburg.

Hartson, H. R., T. S. Andre, et al. (1999). The User Action Framework: A Theory-Based Foundation for Inspection and Classification of Usability Problems. Proceedings of HCI International (the 8th International Conference on Human-Computer Interaction) on Human-Computer Interaction: Ergonomics and User Interfaces-Volume I - Volume I, Lawrence Erlbaum Associates, Inc.

Hartson, H. R. and D. Hix (1993). Developing User Interfaces, Ensuring Usability Through Product & Process. New York, John Wiley & Sons, Inc.

Hartson, H. R., P. Shivakumar, et al. (2004). "Case Study of a Digital Library Usability Inspection." Special Issue of Journal of Digital Libraries on Usability of Digital Libraries 4(2): 108-123.

Holm, R., M. Priglinger, et al. (2002). Automatic Data Acquisition and Visualization for Usability Evaluation of Virtual Reality Systems. Eurographics, Saarbrucken, Germany.

Howarth, J. R. (2004). An Infrastructure to Support Usability Problem Data Analysis. Computer Science. Blacksburg, Virginia Polytechnic Institue and State University: 104.

Ivory, M. Y. and M. A. Hearst (2001 ). "The state of the art in automating usability evaluation of user interfaces " ACM Comput. Surv. 33 (4 ): 470-516

Keenan, S. (1996). Product Usability and Process Improvement Based on Usability Problem Classification. Computer Science. Blacksburg, Virginia Polytechnic Institute and State University: 242.

Kumar, H. P., C. Plaisant, et al. (1997). "Browsing Hierarchial Data with Multi-Level Dynamic Queries and Pruning." International Journal of Human Computer Studies 46(1): 103-124.

Lamping, J. and R. Rao (1994). Laying out and visualizing large trees using a hyperbolic space. Proceedings of the 7th annual ACM symposium on User interface software and technology, Marina del Rey, California, United States, ACM Press.

Mahajan, R. (2003). A Usability Problem Diagnosis Tool: Development and Formative Evaluation. Computer Science. Blacksburg, Virginia Polytechnic Institute and Stat University: 77.

McQuaid, H. L. and D. Bishop (2001). An Integrated Method for Evaluating Interfaces. Usability Professionals' Association 2001 Conference Proceedings, Lake Las Vegas, Nevada.

Nayak, N. P., D. Mrazek, et al. (1995 ). "Analyzing and communicating usability data: now that you have the data what do you do? a CHI'94 workshop " SIGCHI Bull. 27 (1): 22-30

Nielsen, J. and R. Molich (1990). Heuristic evaluation of user interfaces. Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people, Seattle, Washington, United States, ACM Press.

Norman, D. (1988). The Design of Everyday Things. New York, Doubleday.

North, C., N. Conklin, et al. (2002). "Visualization Schemas for Flexible Information Visualization." Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02): 15.

Plaisant, C., J. Grosjean, et al. (2002). SpaceTree: Supporting Exploration in Large Node Link Tree, Design Evolution and Empirical Evaluation. Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02), IEEE Computer Society.

Pressman, R. S. (2001). Software engineering: a practitioner's approach, Thomas Casson.

Rosenfeld, L. and P. Morville (1998). Information Architecture for the World Wide Web. Sebastopol, California, O'Reilly & Associates, Inc.

Rosson, M. B. and J. M. Carrol (2002). Usability Engineering: Scenario-Based Development of Human-Computer Interaction. San Francisco, Morgan Kaufmann Publishers.

Saraiya, P., C. North, et al. (2004). "An Insight-based Methodology for Evaluating Bioinformatics Visualizations." IEEE Transactions Visualization and Computer Graphics.

Shneiderman, B. (1992 ). "Tree visualization with tree-maps: 2-d space-filling approach " ACM Trans. Graph. 11 (1 ): 92-99

Shneiderman, B. (1996). The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. Proceedings of the 1996 IEEE Symposium on Visual Languages, IEEE Computer Society.

Sridharan, S. (2001). Usability and Reliability of the User Action Framework: A theoretical Foundation for Usability Engineering Activities. Industrial and Systems Engineering. Blacksburg, Virginia Polytechnic Institute and State University: 87.

Stasko, J. and E. Zhang (2000). Focus+Context Display and Navigation Techniques for Enhancing Radial, Space-Filling Hierarchy Visualizations. Proceedings of the IEEE Symposium on Information Vizualization 2000, IEEE Computer Society.

TechSmith (1995). The Digital Solution for Usability Analysis. 2005: A Complete Usability Testing Solution for Web Sites and Software (http://www.techsmith.com/products/morae/default.asp).

Wesson, J. and D. V. Greunen (2002). Visualisation of usability data: measuring task efficiency. Proceedings of the 2002 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology, Port Elizabeth, South Africa, South African Institute for Computer Scientists and Information Technologists.

Zazelenchuk, T. (2003). Using Microsoft Excel to Collect Usability Data. 2005 (http://www.userfocus.co.uk/resources/datalogger.pdf).

# 7 Appendix

## 7.1 Benchmark Tasks

1. You are looking at a screenshot of the tool's overview area, only the UAF structure. From this screenshot:

   What interaction cycle (second level node) has the most problems in its sub-tree?

   What interaction cycle (second level node) has the most nodes in its sub-tree?

2. You are the usability expert beginning to analyze the collected usability data displayed by the tool. The goal of your analysis is to gather enough information so you can make informed decisions about the usability problems; what problems should be fixed, future steps in development, pattern or trends in the usability data, etc.

   Please take 5 minutes to explore the data. Talk out loud while you explore the data. You are looking for any patterns, trends, generalities, insights* in the data that would help in your job. Please do not take any breaks as we are timing you in this task.

   * *An insight for our purposes is defined as an individual observation about the data.*

3. You have collected a number of usability problems. Each usability problem has a problem name, a description, a cost, and an importance. The cost is determined by how many person hours the problem will consume to fix. Cost can be rated from 1 to any realistic number of person hours. Importance is rated on a scale of 1-5, 1 being a minor problem and a rating of 5 is a must fix problem.

   How many problems have a cost of 5 or less and importance rating of 3 or above?

   How many problems have a cost of 5 or less?

How many problems have an importance rating of 3 or above?

---

Follow-up question for task 3:

---

You have found 14 problems matching the criteria. Where are the majority of these problems located in the overview area of the visualization tool?

4. You have just completed a small session of lab based usability testing with 3 participants, who found 31 different problems. Each problem has been diagnosed by associating it with a UAF node. The UAF node is associated to one or more descriptive keywords. This results in each problem having an associated keyword or multiple keywords.

   What keyword(s) is associated with the most usability problems?

---

Follow-up question for task 4:

---

You have said the Keywords "clarity of design" and "use of language" both have 10 problems. Now, what keyword has more important problems, a rating of 3-5 on the importance scale?

5. For this task only: assume that you are viewing problems that have not been assigned an importance rating. Your purpose it to assign those problems to one evaluator who is most familiar with the problems, the one who discovered the majority of those problems.

   You are looking to find problems associated with "functionality" (There could be multiple keywords that deal with "functionality" so use your best judgment). Then assign those problems to the evaluator who discovered the majority of the "functionality" problems.

What is the name of the evaluator you would assign the functionality problems?

6. What stage or node in the interaction cycle (second level of UAF) has the largest grouping of problems with an importance of 4 or higher?

   How many total problems exist in this node's sub-tree?

   How many selected problems (problems with an importance of 4 or higher) exist in this node's sub-tree?

7. In this task your job is to find problems dealing with the keywords: "use of language", "precise wording", "labeling", "word choice", and "jargon".

   You want to find all problems associated with the listed keywords and those that have an importance rating of 4-5.

   How many usability problems fit into the criteria described above?

8. How many problems have a cost of 20?

   What is the name(s) of the node(s) where this problem is located?

   What is the name(s) and problem ID(s) of the problem(s)?

9. Please check the keyword "user centeredness". Can you tell me the name of the node (not the problem name) where this problem is located?

10. Are there any more insights that you can think of after going through the tasks? If so please take the time to demonstrate the other insights you can find using the tool?

## 7.2 Questionnaires

### Entrance Questionnaire for Demographic and Background Information

Thank you for participating in our study. Please let us know if you have any questions at any time. All information will be kept confidential and used for purposes of the study only.

Please answer the following questions:

1. **Age:** _____

2. **Gender:** (please circle) Male Female

3. **Did you watch the video tutorial of the tool?** (please circle) Yes No

**If you answered <u>yes</u> to question 3, please answer the following questions.**

**3 a. Approximately how many hours have pasted since you viewed the video?**

(Please circle) 0-6 hours 6-12 hours 24 - 36 hours 36+ hours

**3 b. Your overall understanding of the tutorial**

| Tutorial not understandable | | | | | | | | Tutorial very understandable | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |

**3 c. How attentive were you while watching the tutorial**

| Not attentive | | | | | | | | Very attentive | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |

4. **Level of usability knowledge:** (please circle) Advanced Intermediate Beginner

5. **Have you ever had the task to analyze collected usability data whether it is in an academic environment or industry?** (please circle) Yes No

6. **Are familiar with Don Norman's stages of interaction (Gulf of execution/evaluation)?** (Please circle) Yes No

7. **Are you familiar with the User Action Framework?** (please circle) Yes No

8. **If you circled "yes" to question 7:**

   **8.a. How were you introduced to the User Action Framework (i.e. Usability Engineering, research, etc.)**: **What was your task(s) associated with the UAF; class assignments, research, usability analysis, etc?**

   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____
   _____

   **8.b. What would you evaluate you UAF knowledge to be:**
   (Please circle) Advanced Intermediate Beginner

9. **Thinking about your usability engineering experience what might be some characteristics of the usability data you will be looking to find during analysis? For example, distribution of problems, many problems of the same type (re-occurring problems), problem prioritization, etc.**

   ***If necessary use back of page to answer.***

## Exit Questionnaire

**Instructions of tasks are clear**

Strongly disagree
Strongly agree
0 1 2 3 4 5 6 7 8 9 10 NA

**Data presented in the tool was overwhelming**

Strongly disagree
Strongly agree
0 1 2 3 4 5 6 7 8 9 10 NA

**Data presented in the tool was understandable. (e.g. there was no confusion with the presentation of the data)**

Strongly disagree
Strongly agree

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Satisfaction of visualization tool**

| Unsatisfied | | | | | | | | | | Very satisfied | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |

**Visualization tool was pleasing to the eye**

Strong disagree
Strongly agree

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Visualization tool encouraged exploration of the usability data**

Strong disagree
Strongly agree

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |
|---|---|---|---|---|---|---|---|---|---|---|---|

**Overall reaction of the visualization tool**

Terrible
Wonderful

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |
|---|---|---|---|---|---|---|---|---|---|---|---|

| Frustrating | | | | | | | | | | Satisfying | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |

| Uninteresting | | | | | | | | | | Interesting | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |

| Dull | | | | | | | | | | Stimulating | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |

Difficult
Easy

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NA |
|---|---|---|---|---|---|---|---|---|---|---|---|

**What did you like most about the tool?**

**What did you dislike about the tool?**

**Which tasks did you find especially easy or difficult? Why?**

**Which task/features did you find especially useful or irrelevant? Why?**

**What in your opinion are the bad points of this visualization?**

**What in your opinion are the good points of this visualization?**

**Would this tool encourage you to attempt to do more usability data analysis as opposed to other analysis methods? Why? And what methods are you referring to if any?**

**Other comments about tool: ideas, improvements/desired features, uses, likes/dislikes, or general suggestions?**

## 7.3 Informed consent

**Informed Consent for Participant of Investigative Project**
Virginia Polytechnic Institute and State University

Title of Project: User Action Framework Visualization

Principal Investigator: Chris Catanzaro

**I. THE PURPOSE OF THIS RESEARCH/PROJECT**

You are invited to participate in a study to evaluate a usability visualization tool. This research will aid in a comparison study between standard usability analysis methods and a usability visualization tool. Your involvement in this study will guide us towards more insightful methods to evaluate usability data.

## II. PROCEDURES

You will be asked to perform a set of tasks using an everyday desktop system. These tasks will consist of exploring usability data. Your role in these tasks is that of usability engineer analyzing usability problems and plotting how to fix these problems. We are not evaluating you or your performance in any way; you are helping us to evaluate our visualization tool. All information that you help us attain will remain anonymous. The time you take to do each task and other aspects of your interaction with the system will be measured. You will be asked to think out loud while you are performing each task. You may be asked questions during and after the evaluation, in order to clarify our understanding of your evaluation.

The session will last about one hour. You may terminate your participation at any time, for any reason.

Because we are measuring a range of abilities, some tasks are more difficult than other tasks. It is important that you understand the instructions before beginning each task. If anything is unclear, be sure to ask any questions you may have.

## III. RISKS

There is no more than minimal risk.

## IV. BENEFITS OF THIS PROJECT

Your participation in this project will provide information that will be used to improve the usability design process. No guarantee of benefits has been made to encourage you to participate. You may receive a synopsis summarizing this research when completed. Please leave a self-addressed envelope with the experimenter and a copy of the results will be sent to you.

You are requested to refrain from discussing the evaluation with other people who might be in the candidate pool from which other participants might be drawn. For this experiment, please don't talk to other people about the experiment until permission is granted by the experimenters.

## V. EXTENT OF ANONYMITY AND CONFIDENTIALITY

The results of this study will be kept strictly confidential. Your written consent is required for the researchers to release any data identified with you as an individual to anyone other than personnel working on the project. The information you provide will have your name removed and only a subject number will identify you during analyses and any written reports of the research.

This experiment will be digitally audio recorded. The audio files will be stored securely, viewed only by the experimenters (Chris Catanzaro, Amine Chigani, and Jihane Najdi), and erased after 3 months.

## VI. COMPENSATION

Your participation is voluntary and unpaid.

## VII. FREEDOM TO WITHDRAW

You are free to withdraw from this study at any time for any reason.

## VIII. APPROVAL OF RESEARCH

This research has been approved, as required, by the Institutional Review Board for projects involving human subjects at Virginia Polytechnic Institute and State University, and by the Department of Computer Science.

## IX. SUBJECT'S RESPONSIBILITIES AND PERMISSION

I voluntarily agree to participate in this study, and I know of no reason I cannot participate. I have read and understand the informed consent and conditions of this project. I have had all my questions answered. I hereby acknowledge the above and give my voluntary consent for participation in this project. If I participate, I may withdraw at any time without penalty. I agree to abide by the rules of this project

_____ _____

Signature Date

_____ _____

Name (please print) Contact: phone or address or

_____

Email address (OPTIONAL)

Should I have any questions about this research or its conduct, I may contact:

Investigator: Chris Catanzaro Phone (540) 231-3986
Graduate Student, Computer Science Department
Email: ccatanza@vt.edu

Review Board: David M. Moore, Office of Research Compliance, CVM
Phase II (0442) 231-4991

Instructor: Dr. Chris North Phone (540) 231-2058
Professor, Computer Science Department (231-2458)

Email: north@cs.vt.edu

cc: the participant, Dr. Chris North; Chris Catanzaro