InVis: An Interactive Visualization Tool for Exploring Interaction Networks

Matthew W. Johnson
University of North Carolina at Charlotte
Charlotte, NC
mjokimoto@gmail.com

Michael Eagle
University of North Carolina at Charlotte
Charlotte, NC
maikuusa@gmail.com

Tiffany Barnes
North Carolina State University
Raleigh, NC
tmbarnes@ncsu.edu

ABSTRACT
We introduce *InVis*, a novel visualization technique and tool for exploring, navigating, and understanding user interaction data. *InVis* creates a interaction network from student-interaction data extracted from large numbers of students using educational systems, and enables instructors to make new insights and discoveries about student learning. Here we present our novel interaction network model and *InVis* tool. We also demonstrate that *InVis* is an effective tool for providing instructors with useful and meaningful insights to how students solve problems.

1. INTRODUCTION
The advancement of personalized learning has been declared a Grand Challenge by the National Academy of Engineers [12]. With increasing use of the web in education and learning management systems the amount of data that can be brought to bear on this important challenge is growing rapidly. For example, the PSLC DataShop, a repository for educational data, has collected logs from over 42,000 students from different tutors with a wide range of topics, from algebra to Chinese [8]. However, such large datasets can be unwieldy, and deciding just how to use them to improve learning is a challenge, as illustrated by the emergence of the new field and conference on Educational Data Mining.

We have developed a visualization tool, called *InVis*, to enable educators to interactively explore our novel interaction network representing the interactions students perform in problem-solving environments. *InVis* displays student behavior across an entire class, enabling educators to develop insights from common strategies and mistakes that groups of students apply in a software tutoring environment. While this work will concentrate on data from computer-aided instructional environments, we have also used *InVis* for exploring user interactions in games and plan to use it for other applications that record sequential interactions.

Educational data mining tools require "good visualization facilities to make their results meaningful to educators and e-learning designers." [15]. If done well, visualizing problem-solving interaction logs can provide insight into how users solve problems and what errors they encounter; and provide more information than a purely summative approach.

To address these issues, we have developed a model that represents the interactions of large groups of users in problem-solving environments, called the Interaction Network model. We developed *InVis* to allow educators to visualize and interactively explore interaction networks to better understand student interactions in problem-solving software tutors. We used three methods of evaluation: 1) a guidelines review, where we compare *InVis* to the commonly accepted visualization guidelines; 2) a set of case-study like success stories with expert users, and 3) a summative usability study, where educators explored data from a university-level logic tutor using guided tasks and completed a validated usability scale, and reflected on their experience through a qualitative survey. Our 'triangulated evaluation' is inspired by Plaisant's aptly named paper, the 'Challenges of Visualization Evaluation', where she described the difficulties of performing evaluations with tools that can "answer questions you didn't know you had." [13] Since there is no single proven technique for visualization evaluation, Plaisant recommends using several complementary methods that can mitigate the weaknesses of single techniques used alone. Our guidelines review shows how *InVis* adheres to commonly accepted visualization guidelines. In the case-studies, educators were able to generate and confirm hypotheses, and discover insights into their data. The results of the usability study showed that educators were able to complete tasks at 85% accuracy with minimal training time.

We show that a Interaction Network is an effective and reasonable description of student interaction data from computer-aided instructional software for problem solving. We also show that *InVis*, an effective interactive visualization designed for visualizing Interaction Networks, can be made and can provide useful and meaningful insights to student behavior in software tutors.

1.1 Related Work
InVis was inspired by work in exploring student data on solving logic proofs. In 2007, Barnes and Stamper created a frequency-annotated behavior graph and a method to convert it into a Markov decision process (MDP) from student logic proof data, and loaded this data into the GraphViz visualization tool to visualize student problem solving sequences [1]. From this exploration, an experienced logic instructor discovered surprising student behaviors and unexpected difficulties with the logic tutor interface. The instructor also found that some students demonstrated expert-like solutions, and that students did not flounder as much as expected while solving problems.

Student tutor-log data sets are large (representing hundreds of problem attempts with hundreds of problem states) and teachers, who are not necessarily savvy with graphs, spreadsheets, or statistics, need support in navigating these large domain models to learn about student behavior. Ben-Naim and colleagues [3] have developed an authoring tool that allows teachers to create simulators for science and explore small graphs of student actions in the simulator. However, this visualization is restricted to teacher-created states, where teachers label a step in the simulator as one of interest. It is not fully derived from student data and does not facilitate exploration of a large, diverse dataset from other tutors.

SpaceTree [14] is software that might enable educators to explore our static interaction networks more interactively than graph visualization tools like GraphViz [7] and Gephi [2]. However, our networks are not always trees and can contain cycles and loops. The Spacetree layout also particularly highlights the children of a node, while we are interested in the full path from start to finish for problem-solving sequence, and in seeing a whole set of student behavior at once, to provide an overview and support pattern-finding.

CourseVis is a visualization tool produces graphical representations of student tracking data collected by a Content Management System, and helps teachers gain an understanding of how students are behaving in their online courses. In CourseVis, the focus is on the behavior of a student over the course of an entire system, where as in our work the focus is more fine grained, as we are interested in the behavior of students in single problems. CourseVis does support some techniques to look at student performance but the focus is on visualizing knowledge components and assessment performance, not problem-solving behavior as in our work [10].

In VisMod students are provided with a visualization tool for representing and interacting with their own student-model allowing students to develop their meta-cognitive skills [19]. The focus of this tool is not the behavior of the students but instead what the students think about their own behavior.

TADA-Ed is a tool designed for mining educational data generated from digital tutors, much like our work. TADA-Ed's focus is on visualizing the results of several data-mining techniques, such as k-means clustering and decision trees applied to educational data [11]. Our work is different in that our focus is on student problem solving behavior.

2. INTERACTION NETWORK MODEL

In this section we describe our novel Interaction Network Model, which we believe can be used across many domains to describe sequences of user interactions in software interfaces. This is the first work to model student problem-solving in a tutoring environment in this way, and to use the Interaction Network model to better understand student tutor-log interactions. As we describe later, we've applied the Interaction Network model to two widely different domains (logic proofs and drawing pictures on a grid) to understand student behavior. The novelty lies in the combination of several ideas: 1) how to define a 'state' and 'action', 2) how to combine multiple sequences into one Interaction Network, and 3) how to represent actions that are valid within the interface (e.g. clicking an active button) but not within the problem-solving context (e.g. clicking the wrong button).

We model a solution attempt as a sequence of states (vertices) and actions (edges). *Case* refers to a single student's solution attempt.

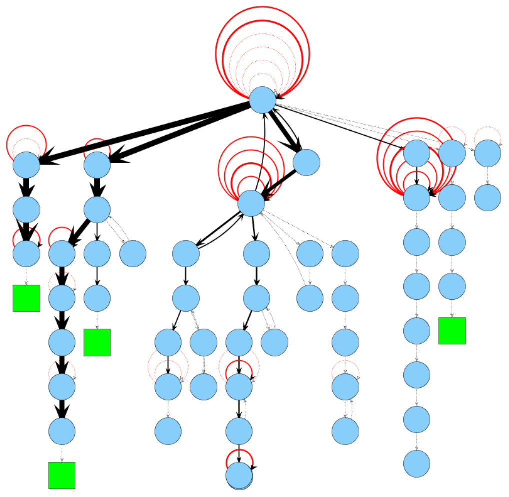


Figure 1: An example of a small Interaction Network.

The interaction network for a problem is the conjoining set of all of its solution attempts. *State* describes the state of the software environment, representing enough information so the program's state could be regenerated in the interface. *Actions* describe user interactions and their relevant parameters, to move a user from one state to the next. We also store the set of all cases who visited any particular state-vertex or action-edge, allowing us to count frequencies and connect case specific information to the interaction network representation. This representation results in a connected, directed, labeled multi-graph with states as vertices, actions as directed-edges to connect the states, and cases that provide additional information about states and edges. (See figure 1.)

This representation allows to us build a interaction network model from any system that logs interactions in state, action, resulting-state tuples. For the intelligent tutoring system community, we encourage the use of logging standards such as the PSLC's Datashop [8]. We recommend preservation of the action's parameters and results, since they are useful for labeling the visualization.

To build the interaction network for a problem we combine the interaction sequences, from each case into a single network, where states are combined when they are considered equal. In different tutors and interfaces, two states could be considered equal as long as the screen looks the same, or all the same actions have been performed, regardless of order, but in other cases, states arrived at by taking the same actions in any order could be considered distinct.

InVis will handle either case, and the logic data in our experiments preserved order. Frequency information and information about which cases have visited, is embedded into the edges and vertices. This results in a network graph which shows the interactions of a large number of users in a relatively small space.

The basic example is whether or not order matters. For example, in the Deep Thought data should the state A,B be distinct from the state B,A? An order-matters approach results in more distinct program states, and thus more vertices. However, preserving the order of premises derived increases the information of the visualization by making the order of steps a student takes more obvious. By contrast, an unordered approach reduces the number of states and reduces the size of the overall graph-layout. While the unordered graph provides a more generalized view, it can be harder to follow the precise steps of an individual case.

2.1 Visual Representation & Graph Layout

A graph is a natural representation for our Interaction Network model. However, there are still a wide variety of graph layouts, and the primary visualization view should be one that is easiest for the novice user to understand. *InVis* uses a tree-like graph layout to present Interaction Networks. The root node is the starting state (the problem 'givens'), and is placed at the top of the view, with student interactions branching downward. This layout makes it easy to follow a student's individual solution-path, as the vertex depth effectively preserves the number of (non-error) steps in the solution-attempt.

State vertices can be labeled with the entire state description, or simply the result of the latest interaction, since reading sequential states should reveal the entire state description. If they exist in the data, final or goal states are represented with a different color and shape, a green rectangle here. Each edge is labeled by its action, but not its parameters, to keep it more readable. Edge thickness is determined by the frequency of observed interactions, with most-frequent edges being thickest. *InVis* uses JUNG [18] to efficiently place nodes in a graph layout.

2.2 Modeling Program States

We have successfully built interaction networks from a variety of sequence oriented interaction data. Here we describe two systems and concentrate on how we modeled the state description.

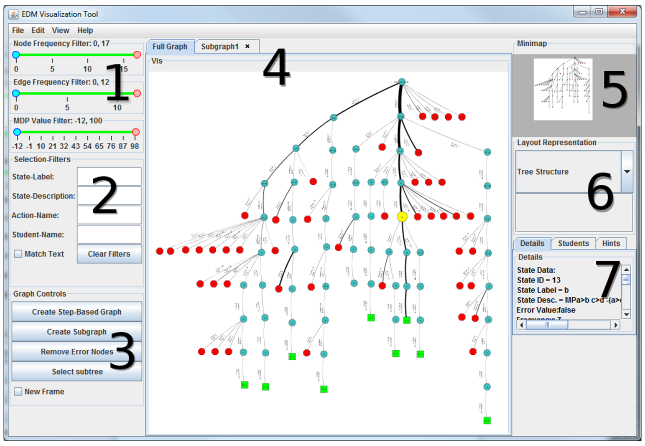
2.2.1 BeadLoom Game

The BeadLoom Game (BLG) [4] is a game extension of the CSDT: Virtual Bead Loom, an educational tool for teaching Cartesian coordinates to middle school students. The BLG added game elements in order to increase motivation and learning [4]. In the BeadLoom Game, players place beads in a 41x41 Cartesian grid using six different tools and an undo command. All actions take a color (12 different options) parameter. The loom starts empty and once beads are added they cannot be removed, aside from the undo action. However, beads can be overwritten by future actions. The goal of the game is to create a target image with the tools available. Figure 4 shows an example from the BeadLoom Game.

To gain a better understanding of the BLG data, we used the *InVis* to explore player solutions. The state representation is a 41x41 array containing the 12 color values. Actions are represented by the six bead-placement tools and their parameters. We also store the set of all cases who visited any particular state-vertex or action-edge. We use an image depicting the player's state as the state label. The BLG does not have error data, meaning users cannot submit invalid parameters as actions. However, users are able to undo previous actions, which we can interpret as an unintended action.

2.2.2 The Deep Thought Tutor

Deep Thought is a propositional logic tutor in which students are tasked with performing first-order logic proofs [6]. Students are given a set of premises and a conclusion; students must use basic logic axioms to prove the conclusion. As the student works through the proof, the tutor records each interaction. We model the application of axioms as the actions. We model the state of the logic tutor as the conjoined set of each premise and derived proposition.



Details
State Data:
State ID = 13
State Label = b
State Desc. = MPa>b c>d -(a>
Error Value:false

Figure 2: An example screenshot of the default view in *InVis*.

For example a student starts at state $A \vee D, A \rightarrow (B \wedge C), \neg D \wedge E$, where each premise is separated by a comma. The student performs the interaction $SIMP(\neg D \wedge E)$, applying the simplification rule of logic to the premise $\neg D \wedge E$ and derives $\neg D$. This leads to the resulting-state of $A \vee D, A \rightarrow (B \wedge C), \neg D \wedge E, \neg D$.

Errors are actions performed by students that are illegal operations of logic and the tutor, so the student cannot exist in that error-state. As a result they are returned to their previous state. In the case where a student makes an error, the action is marked as an error. For example: The student is in state $A \vee D, A \rightarrow (B \wedge C), \neg D \wedge E, \neg D$. The student performs the interaction $SIMP(A \vee D)$ in an attempt to derive A. The resulting-state would remain $A \vee D, A \rightarrow (B \wedge C), \neg D \wedge E, \neg D$, the log-file would mark this edge as an error.

3. DESIGN OF THE VISUALIZATION

Here we address the design of the interactive visualization, *InVis*. The tool was designed with the visual information-seeking mantra [16] as our guide, based on two reasons. First, Schneiderman states from his own extensive experience that an effective visualization contains those elements, and second Cairns has shown the sheer number of examples where it has been used to guide visualization design [5]. Thus we feel this to be a reasonable approach to developing such an interactive visualization. Note our visualization is just one method for designing an interactive tool for exploring Interaction Networks, and later we provide evidence that it is effective at its goal, but many different types of implementations for exploring Interaction Networks are likely to exist.

Referring to figure 2 we will explain the major features of *InVis*.

1. Frequency filters allows the user to filter edges and nodes based on the frequency. The filter removes nodes or edges based on the range. Frequency is a useful metric for investigating the behavior of a large number of users. Nodes and edges with high frequency identify common behaviors, while low frequency nodes represent less common behaviors. Because frequency is an intuitive domain-independent metric we choose to add this filter to the default view.

2. The selection filters, allow users to select states, actions, or cases by entering a search string; the user can select to match with *contains* or *direct match*. This provides an easy measure

to select large numbers of specific states, and is powerful when combined with the subgraph extraction feature.

3. Graph controls allow users to create subgraphs, which are then loaded into a separate tab. The user can also remove the hanging error nodes from the current graph, if they are interested in correct behaviors more than errors. Creating a subgraph, copies all of the currently selected nodes to a new tab (in the interface) and re-applies the graph-layout to the subgraph. This can be used to clear up clutter. By first selecting desired states, edges, or cases with the selection filters, and then generating a subgraph a wide range of options for exploring the data is possible. An example of the subgraph generation is shown in figure 3.

4. The interaction network is displayed here. This panel has mouse controls for panning, zooming, and selecting, controlled by right-click, mouse-wheel, and left-click respectively. Multiple graphs or subgraphs can be loaded at once, each placed in a separate tab.

5. A Mini-map, helps users stay oriented even in large data sets and provides context for the user. The mini-map provides a white box which represents the current view of the main visualization panel described above, and updates as the user pans and zooms.

6. In figure 2 the interaction network is shown using the default graph layout; however, there are several layouts available via this drop down menu.

7. Additional information about the states, actions, and cases are available here. Users are able to view the complete state of the node and a list of all case IDs who visited the state or edge. Other information, such as tutor hint messages or test scores can also be displayed here.

3.1 Guidelines Review

Our visualization was designed with the visual information seeking mantra in mind. As such the tool supports functionality for overview, zooming and filtering, details on demand, view relationships, and extraction. We are going to describe why the element is important to viewing interaction networks, how each element was included and supported, and improvements which can be made. Craft and Cairns state that many other developers cite the mantra as their guiding source for the development of their visualization but often forgo explaining how and where they used it [5]. We use this evaluation to find strengths and weaknesses in our approach, and confirm we have an implementation based on these principles.

3.1.1 Overview First

The hierarchical graph offers an overview representation of the students' behavior as they work through a single problem. Combined with the edge width representing frequency we provide a quick understanding of student behavior trends. In addition the mini-map consistently orients the user within the context they are working, always providing an overview for the user to reference as they navigate the Interaction Network graph. A possible improvement would be to apply other visual components to other variables, future studies will show the affects of these changes.

3.1.2 Zoom and Filter

Zooming and filtering on the interaction network means users can focus on specific behaviors they are interested in. Zooming is not only physically zooming in on the graph-layout, but also zooming in on students who performed a specific action, or visited a particular state. Zoom and filter are supported through a variety of filter and selection tools which allow manipulation based on states, student-IDs and actions. Additionally selection combined with creating subgraphs allows for alternative approaches for filtering and zooming, while the mouse wheel allows the user to zoom on the graph layout. Frequency based filters allow users to focus on either common trends or atypical behaviors exhibited by the students. Identifying common sequences in the interaction network could help identify sub-goals to problems and is one type of improvement that could be made to filtering and selection.

3.1.3 Detail on Demand

The details tab is where the user can find specific information regarding a state, action or student. Details are available in a set of tabs, displaying text information about the selected node. Including students who visited the node, the frequency of the state, actions leading from the node, whether the state is a goal or error state and the description of the state. These details are the finest granularity we can provide from the log data which was read in. One improvement for details is to provide aggregate user statistics regarding the current graph, for example number of error states, total number of states, number of actions, average actions per student, and more.

3.1.4 View Relationships

To relate in our tool is to compare behaviors between students and to compare sub-graphs of the interaction network. Students can be compared to other students by selecting their nodes via the student selection tool and generating sub-graphs. Sub-graphs allow users to compare, at once, all the times a specific action was used from all the students. Viewing the sub-graphs of two frequent paths let a user view how two different strategies were applied to solving the problem. View relationships is supported through the selection and filtering tools combined with creating new sub-graphs in separate tabs. An example of this type of comparison is shown in figure 3. Multiple problems can be loaded into separate tabs and problems can be compared, as well as Interaction Networks for groups of students. Much of the Interaction Network maintains a hierarchical structure, so it is possible to do slight comparison between similar approaches, but a desirable improvement for comparing could be to allow users to more easily compare strategies between students, more on this in the discussion section.

3.1.5 History

Shneiderman notes that History is the most often ignored element due to its high cost of development and is rare in prototypes [16]. In our prototype interface this feature is weakly supported. In *InVis*, when users edit the main graph in significant ways, such as filtering vertices, the new graph is placed in a new tab. Users are able to keep track of each in the tabbed interface. This allows some measure of preserving the history of the user actions. However, this could be improved further by allowing users to undo and redo actions such as selection even when they do not generate new subgraphs.

3.1.6 Extract

Sharing ones findings about student behavior is important, particularly for teachers, so challenging issues for students can be addressed. In *InVis* users can save the image of the visualization panel so that it can be shared. Teachers can show the sequence of steps to their students and highlight situations where errors were common.

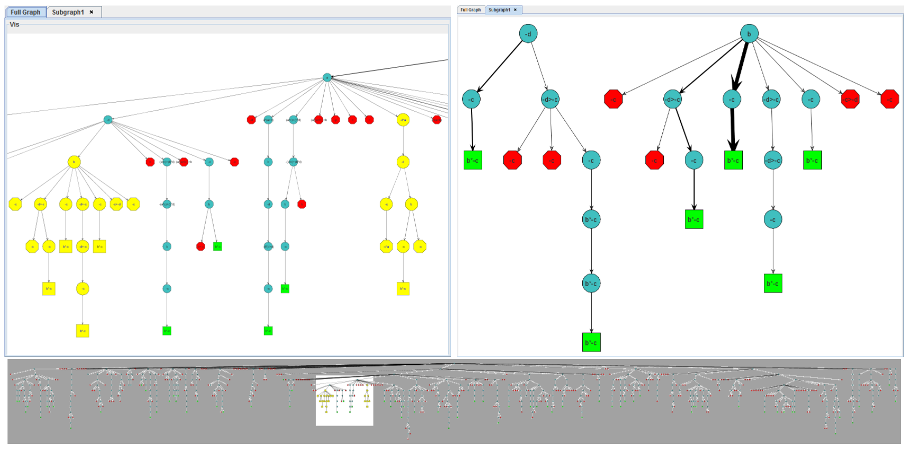


Figure 3: Left) The user has selected several nodes. By constructing a sub-graph, *InVis* presents the selected nodes in a new tab, shown on the Right. Below is an example of the mini-map, with the white box representing the view frame of the left hand image. Note the size of the graph when visualizing 170 students.

Improved sharing between colleagues could be supported by saving the current layouts, subgraphs, and graph annotations, so they can be stored and shared with others easily and they too can interact with the data via *InVis*.

4. CASE STUDIES AND SUCCESS STORIES

Plaisant comments on the usefulness of case studies and success stories in her work, 'The Challenge of Information Visualization Evaluation' [13]. These methods are common ways to aid in the evaluation of visualizations. These evaluations provide useful information from experts and can help address whether a user was able to find answers to questions they did not know they had. As mentioned by Plaisant, evaluation of tools meant to discover features which you did not know existed is difficult. Tory and Moller offer some support for ways in which this can be done [17]; and argue that expert reviews are useful because they can identify aspects of systems which non-experts may not recognize. In order for tools like *InVis*, to be adopted it is important to have success stories so that the early majority will adopt the tool [13], for this reason we provide four *InVis* success stories.

We met with the developers of two separate propositional logic tutors. Both developers are University professors of logic and have extensive knowledge about how their tutors and their logs. We modeled their logic tutor data and provided each with *InVis*. We observed as they used the visualization tool to explore their tutor log-data. We also met with the developers of an educational game and visualized student-player data in a similar way.

4.1 Case 1: Deep Thought

We visualized data from Deep Thought [6] and interviewed the professor responsible for its development. We met for one hour and had him explore tutor data and inform us of different insights and hypothesis he was able to discover or confirm from using *InVis*.

We prepared a data set of thirty students, representing a classroom of students. The professor noticed a student had performed addition rather than conjunction in order to derive $A \wedge B$, which is an incorrect application of the rule. After he recognized this, he mentioned that it was a common mistake made by students; this was his hypothesis. He then used the action selector and entered ADD which selected all instances of students performing the ADD action in his logic tutor. Next he built a sub-graph, moving all those actions and their corresponding nodes to a new tab. Last, he was able to confirm his hypothesis; the data showed that eight of nine applications of the addition rule (ADD) were errors, five of which would have been correct had the student performed conjunction instead, the action the student actually needed. With the same hypothesis in mind, a larger separate data set of Deep Thought tutor data was loaded, this time 170 students.

Again all addition actions were selected, a sub-graph generated, and the hypothesis confirmed. This time, 16 out of 18 applications of the addition rule were in fact errors, and 8 of the 16 would have been correct had the students used conjunction instead. Within minutes, our user was able to identify a hypothesis, check the data using *InVis* and confirm the hypothesis. In the second data set, the task certainly would have been time prohibitive had he scoured the 170 student logs of the data.

4.2 Case 2: Proof Solver

In this example, the professor was interested in the general behavioral trends of students. By including frequencies on nodes and edges, we are able to identify strategies that students perform in order to solve problems. In this case the hypothesis was that students who change all the implications into 'ORs' likely had no true strategy for completely solving the problem. The reason for this is over the years the professor has recognized students who employ this strategy often have difficulty actually solving the problem and thus students are explicitly instructed in class not to use this approach.

After loading that data into *InVis*, the professor looks for the two main strategies performed by the students. The strategies being the two most frequent sets of steps performed. The first strategy, having the highest frequency, is the strategy which she teaches to her students in class, we call the professor's strategy, the first node in this strategy has 74 students. The next most common first step has 29 students, and is the start of the prohibited strategy, that is to change an implication into an OR. Next the professor selected the first node of a strategy and performed the select sub-tree action, which selected all states derived after the current state, effectively selecting all the different variations of the professor's strategy. Then she created a sub-graph. The same was done for the prohibited strategy. Next she selected all of the goal nodes of each sub-graph in turn and looked at the combined frequency of the goal nodes for each sub-graph. For the 74 students who applied the professor's strategy, 55 of those students arrived at the goal, giving a 74% success rate. For the prohibited strategy approach, the sub-graph has a combined goal node frequency of 17 out of the 29 students, resulting in a 59% success rate which is noticeably lower. In total there are 174 students, and the two strategies highlighted above are the most common. The next two most common strategies have frequencies of 11 with 9 and 7 students successfully solving the problem with their respective strategies. Again suggesting the prohibited strategy approach has a particularly low success rate.

4.3 Case 3: Debugging Tutors

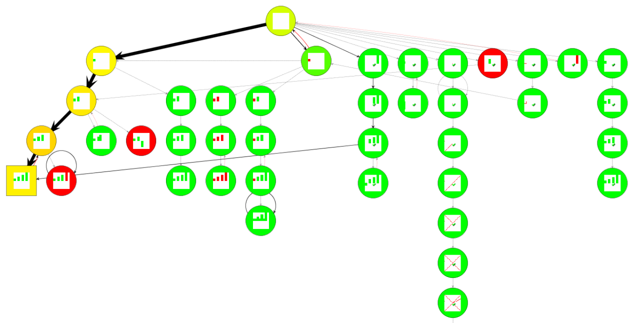
One interesting application of *InVis* is found in the debugging of tutors. In the previous examples, the Interaction Network uncovered bugs in the tutor systems; that is, places where the recorded interactions should not have been legal actions. This is interesting, as both of these tutors have been used for many years and their data have been subject to extensive analysis. However, these bugs were not discovered until their data were visualized with *InVis*. Viewing the entire group of user behaviors at once improves the ability to spot 'peculiar' behaviors. In *ProofSolver* several solutions were noticeably shorter than the average, or skipped to the goal in 'strange' and invalid ways. After examining the series of actions these students performed, the professor confirmed that the interactions were illegal and should not have been permitted.

In the case of *Deep Thought*, some students were able to reach the goal by repeatedly performing the same action. In this case, the students were able to use the instantiation-action inappropriately to add any proposition to the proof. As a result of this, students could simply add items directly to the proof rather than use the axioms, allowing them to game the system and illegally solve the problem.

4.4 Case 4: BeadLoom Game

We collected game log-files from a study performed on the BeadLoom Game (BLG) in 2010. Data came from a total of 6 classes, ranging from 6th to 8th grade; for a total of four sessions. There were 132 students, and 2,438 game-log files. The students were split into two groups (called A-day and B-day) and were presented with BLG features in different orders. The A-Day students were given access to custom puzzles (a free play option,) while B-Day students were given a competitive game element in the form of a leader board. Due to differences in student time lines some B-Day classes missed session three. These students followed an abbreviated A-Day schedule during session four. In order to investigate whether or not there were different problem solving patterns between the groups, we colored vertices based on the percentage of students who visited from each group. The values were normalized from green (A-Day) to red (B-Day.) We loaded the data into *InVis* and presented it to the BeadLoom Game developers.

Next we met with the BeadLoom Game developers and asked them to explore their log data using our prototype visualization tool. In figure 4 we have a set of students who worked on the same problem on two different days, the first and third day of the study. By looking at the number of states we can see a more diverse set of attempts on the first day. As mentioned before, edge width represents frequency, green vertices are from one set of students and red vertices are from another set based on how the study was run, the goal has a square vertex. At the start of our investigation we colored the


(a) Students on Day One

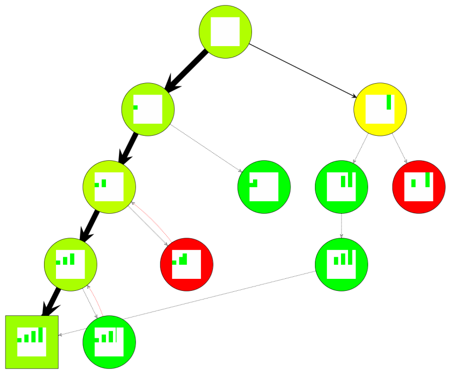

(b) Students on Day Three

Figure 4: This image shows the students attempt on the first day on the top, and their third day attempt on the bottom. This image suggests that as students become more familiar with the tool they are better at solving the problem and make fewer mistakes, thus fewer states are visited.

vertices to see if we could discover differences but it does not seem the classes had any significant differences between them. It is possible that the change in the number of states over time is the visual representation of learning, which figure 4 might suggest, additional research will be necessary to determine if that is so. The designers were able to identify a variety of design changes they would like to make to the game after spending roughly 20 minutes using *InVis*. The most surprising detail the developers were interested in was the number of students who seemed to participate in off-task behavior. Off-task behavior is easily identified as student solution-paths with low frequency and unusual length. For example, a student may opt to draw a picture rather than solve the puzzle. This will result in a path visually jutting out of the interaction network.

5. USABILITY STUDY

One goal of *InVis* is to make complex interaction data accessible to non-experts in the field of user-modeling. To test the usability of *InVis* we created a quantitative task-based test as a measure of summative usability testing. We developed 15 tasks based on the propositional logic tutor interaction data. These questions were designed based on common use-cases, and cover the range of features in the tool. Noting the inherent difficulties involved with evaluating a visualization tool, we ran a user study consisting of three sections, a quantitative portion, usability portion and qualitative portion, each supporting different types of evidence that the tool is effective at allowing users to make discoveries about their data.

5.1 Methods and Materials

For the design of the quantitative portion we created 15 questions which were atomic and had a set of correct responses, we determined how well users were able to answer the questions in this portion of the survey. Table 1 highlights 10 of the 15 questions as a representative sample of the nature of the tasks.

For the usability portion, we used a validated survey written by James Lewis at IBM [9]. Questions 9, 10 and 11 from his survey CSUQ were removed because they were deemed irrelevant to *InVis* because it does not contain any error messages. One difference between Lewis' survey and ours was in our CSUQ, the score has high scores being preferable rather than low scores; Strongly Agree is equal to seven instead of one. Our overall CSUQ score was 4.65. CSUQ stands for The Computer System Usability Questionnaire, and is divided into four scores, an Overall CSUQ score, a SYSUSE, INFOQUAL, and INTERQUAL scores. SYSUSE stands for system use, INFOQUAL is information quality and INTERQUAL is interface quality. The qualitative portion of the survey allowed users to provide open ended responses directed towards functionality that they would like to see in future versions of *InVis*.

We ran a user study with seven participants to determine the level of usability of our design of *InVis* for understanding interaction networks. In our study we used the logic domain as our target audience, so we collected data from how undergraduate students simplified problems in the logic domain using a computerized logic tutor. All teaching material was conducted in the classroom, and the logic tutor is strictly used for conducting homework. The students were given a set of premises, $[(A \rightarrow B),(C \rightarrow D),\neg(A \rightarrow D)]$, and were tasked with generating a first-order logic proof for the conclusion of $B \wedge \neg C$. Next we duplicated or removed students to ensure each task-question had a single correct answer.

Of the seven participants, we met with four individually and they used our computer with *InVis* and target dataset loaded. We gave them a brief overview of how the tool worked, how to zoom, pan and select. We also demonstrated how to generate a subgraph (a GUI button), and how to use the selection tools (text boxes). The demonstration lasted 5–10 minutes. Due to logistical issues, the other three participants were emailed a 2.5 page description of how the interactive elements of the tool works. This document served as the resource for the 5–10 minute demonstration. These three participants conducted the study by themselves via the Internet. Participants were instructed to contact us if any issues arose that they felt were unintended or prevented them from conducting the study. If any task took more than five minutes, they were asked to stop and move to the next one. Participants were instructed not to ask how to complete any of the tasks. After the quantitative section was complete they were asked to do the usability survey then the qualitative survey. Notably, a sample size of seven is low, but we recruited educators who have taught a course which uses the Deep Thought logic tutor, so we are limited by aspect. This decision is based on the assumption that a person who is unfamiliar with a domain, and related tutor, would not understand the logs of the tutor environment well enough to recognize meaningful tutor-based student behaviors.

Table 1: Quantitative Questions/Tasks

Q1	Find the shortest correct solution-path to this problem.
Q3	Find the most frequent solution-path to this problem.
Q4	Find the error(s) with the highest frequency and write the State ID(s).
Q7	Write the action-label and the corresponding final state ID for the last action of student X?
Q9	After filtering nodes and edges to frequency 5 and greater, how many complete solution paths exist?
Q11	What is the student ID of the student(s) with longest solution to the goal?
Q12	Who are the students on the node with the following node label: –a*-d (note the label is: minus minus a * minus d)
Q14	Highlight the node with node-label: $\neg\neg A \wedge \neg D$ and select the sub-tree, then build a sub-graph. How many error nodes exist in this sub graph?
Q15	Answer yes or no, did student 81 find a goal solution?

5.2 Results

In the quantitative portion of the study the group of participants completed 85% of the tasks successfully ($M = 12.71, SD = 2.66$). From the questions in table 1 the most commonly missed questions were Q4, 64% success and Q6, 71% success. Participants spent an average of about 43 minutes on the quantitative survey, with a SD of about 20 minutes, from survey start and end time stamps. However, participants reported the tasks taking an average of 23 minutes, with a SD of 13 minutes.

To evaluate the relationship between the quantitative skills test and the usability survey we submitted the results to a bivariate correlation analysis. The quantitative skills test strongly correlated($M = 12.71, SD = 2.70$), $r = 0.87, n = 7, p = 0.01$, with the overall usability survey($M = 4.65, SD = 1.89$). With an *r-squared* of 0.76, which means that 76% of the variance in the quantitative skills test-scores is accounted for by variance in usability scores. We examined the three subsections of the usability survey. The quantitative skills test strongly correlated, $r = 0.78, n = 7, p = 0.04$, with the Sysuse score($M = 4.84, SD = 2.04$); it also strongly correlated, $r = 0.90, n = 7, p < 0.01$, with the Infoqual score($M = 4.54, SD = 1.98$); and it strongly correlated, $r = 0.96, n = 7, p < 0.01$, with the Interqual score($M = 4.39, SD = 1.70$).

This result provides evidence for the validity of using the quantitative test as a measure of the visualization quality. This provides insight into what functionalities the developers should focus on in future development of *InVis*. We cannot make strong casual inferences between the quantitative test and the usability scale, i.e., were participants able to complete the skills test because *InVis* was usable, or did they report that it was usable because they were able to complete the tasks? The fact that 85% of the tasks were correctly completed, with little time spent on training, provides evidence that our technique is usable by our target population. The questions that were most commonly missed, were tasks related to understanding the most frequent error, and the state from which the most *unique* errors were made. This highlights a potential problem with the current error-state representation; which seems to make differentiation between unique errors and frequent errors difficult to separate. It is difficult to interpret the results of the CSUQ survey, however this score is useful for comparing to future versions of *InVis*.

5.2.1 Qualitative

In the qualitative section of the study we asked participants about specific ways to improve *InVis*. We mention some of the most important suggestions we received from their experiences. These suggestions are from the conclusions we can make from their qualitative survey results and the comments made while using *InVis*.

An important issue to address is graph layouts, two issues regarding the layout were raised. First the layout would be more intuitive if it were possible to order the layout in some manner along the breadth (x-axis), either based on frequency, or other metric. By applying a more informed layout to the graph, we could order the states in some manner along the X-axis, for example making the most frequent path on the left, and the least frequent on the right.

The second problem is in regard to strategies, sub-strategies and ordered states. Participants mentioned they would like the layout to group or cluster approaches based on similar strategies. If two students each have nine identical steps, but the first step in each approach is different, then the layout does not necessarily put the states from these two approaches close to one another. When looking at 100 plus students, this makes understanding the number of strategies difficult to understand. A graph layout which places similar paths next to each other could provide a more intuitive visualization of the interaction network.

All participants reported a positive response for whether or not they would use the tool to augment their understanding of current classes' behavior and learning. This suggests a need for these types of tools for exploring, and understanding student behaviors in software tutors. We will conclude with a quote from one user who said, 'The tool provides a sense of how broadly varying students are in their approaches, how many get stuck, and how many make similar mistakes.' Which we feel is a good representation of the kinds of insights *InVis* was designed for detecting.

6. CONCLUSION

The main contribution of this work is the discovery and implementation of visualization techniques for user-interaction data from educational systems. This led to new insights into problem-solving in the deep thought logic tutoring environment, for example the conclusions drawn in the case studies. The use of interactive visualization techniques combined with a interaction network model in *InVis* allows users to explore and gain insight from interaction-log data. We performed a user study on *InVis* to show that users can successfully complete relevant tasks, and paired these results with a standardized method for testing the usability of a software tool. Users were able to explore an entire class' set of interactions and were able to confirm some of the hypothesis they had about students, which was a primary goal. This suggests that our technique is effective at allowing users to explore and learn information from the data. This is the first step in creating a domain independent visualization tool for understanding student behavior in software tutors, and our initial results seem promising for the future development of *InVis*.

7. REFERENCES

[1] T. Barnes and J. Stamper. Toward the extraction of production rules for solving logic proofs. In *Proceedings of the 13th International Conference on Artificial Intelligence in Education, Educational Data Mining Workshop (AIED2007)*, pages 11–20, 2007.

[2] M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009.

[3] D. Ben-Naim, M. Bain, and N. Marcus. A User-Driven and Data-Driven Approach for Supporting Teachers in Reflection and Adaptation of Adaptive Tutorials, 2009.

[4] A. Boyce and T. Barnes. Beadloom game: using game elements to increase motivation and learning. In *Proceedings of the Fifth International Conference on the Foundations of Digital Games*, FDG '10, pages 25–31, New York, NY, USA, 2010. ACM.

[5] B. Craft and P. Cairns. Beyond guidelines: What can we learn from the visual information seeking mantra? In *Proceedings of the Ninth International Conference on Information Visualisation*, pages 110–118, Washington, DC, USA, 2005. IEEE Computer Society.

[6] M. J. Croy. Graphic interface design and deductive proof construction. *J. Comput. Math. Sci. Teach.*, 18:371–385, December 1999.

[7] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. Graphviz - Open Source Graph Drawing Tools. *Graph Drawing*, pages 483–484, 2001.

[8] K. Koedinger, R. Baker, K. Cunningham, A. Skogsholm, B. Leber, and J. Stamper. *A Data Repository for the EDM community: The PSLC DataShop*. Boca Raton, FL: CRC Press, 2010.

[9] J. R. Lewis. Ibm computer usability satisfaction questionnaires: psychometric evaluation and instructions for use. *Int. J. Hum.-Comput. Interact.*, 7:57–78, January 1995.

[10] R. Mazza and V. Dimitrova. Visualising student tracking data to support instructors in web-based distance education. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, WWW Alt. '04, pages 154–161, New York, NY, USA, 2004. ACM.

[11] A. Merceron and K. Yacef. Tada-ed for educational data mining. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 7(1):267–287, 2005.

[12] N. A. of Engineering. Grand Challenges for Engineering, 2008.

[13] C. Plaisant. The challenge of information visualization evaluation. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '04, pages 109–116, New York, NY, USA, 2004. ACM.

[14] C. Plaisant, J. Grosjean, and B. B. Bederson. Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In *Proceedings of the IEEE Symposium on Information Visualization (InfoVis'02)*, INFOVIS '02, pages 57–, Washington, DC, USA, 2002. IEEE Computer Society.

[15] C. Romero and S. Ventura. Educational data mining: A survey from 1995 to 2005. *Expert Syst. Appl.*, 33:135–146, July 2007.

[16] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. *Visual Languages, IEEE Symposium on*, 0:336, 1996.

[17] M. Tory, I. Lab, and T. Moller. Evaluating visualizations: Do expert reviews work. *IEEE Computer Graphics and Applications*, 25:8–11, 2005.

[18] S. White, D. Fisher, P. Smyth, S. White, and Y. B. Boey. Analysis and visualization of network data using jung. *Journal Of Statistical Software*, VV(Ii):1–35, 2005.

[19] D. Zapata-Rivera and J. E. Greer. Exploring various guidance mechanisms to support interaction with inspectable learner models. In *Proceedings of the 6th International Conference on Intelligent Tutoring Systems*, ITS '02, pages 442–452, London, UK, UK, 2002. Springer-Verlag.