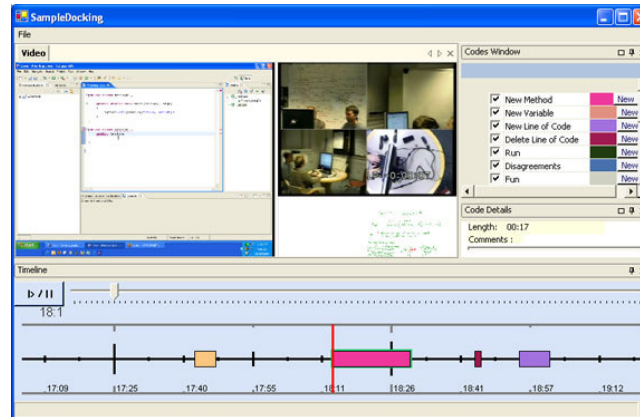


---

# VACA: A Tool for Qualitative Video Analysis

## Brandon Burr

Stanford University  
353 Serra Mall, Room 160  
Stanford, CA 94305 USA  
bburr@stanford.edu



---

Copyright is held by the author/owner(s).  
CHI 2006, April 22–27, 2006, Montreal, Canada.  
ACM 1-xxxxxx

## Abstract

In experimental research the job of analyzing data is an extremely slow and laborious process. In particular, video and audio data of human behavior are difficult to analyze, as this type of information does not lend itself to automation. Here we present VACA, an open source tool for qualitative video analysis. VACA presents video annotations on a timeline interface and integrates external sensor data to improve the rate at which analysis can be performed. A comparative study is run against commonly used video analysis tools, and results are reported.

## Keywords

Video analysis, annotation, behavioral research.

## Introduction

Most disciplines of behavioral study require a significant degree of human observation, either in a lab or in the field. Many of these studies use video as their data medium, as video is perhaps the richest of the recording media. Because the data is very rich, it requires a large amount of time to analyze the qualitative content. Usability and human behavioral researchers analyze video data by watching videos on

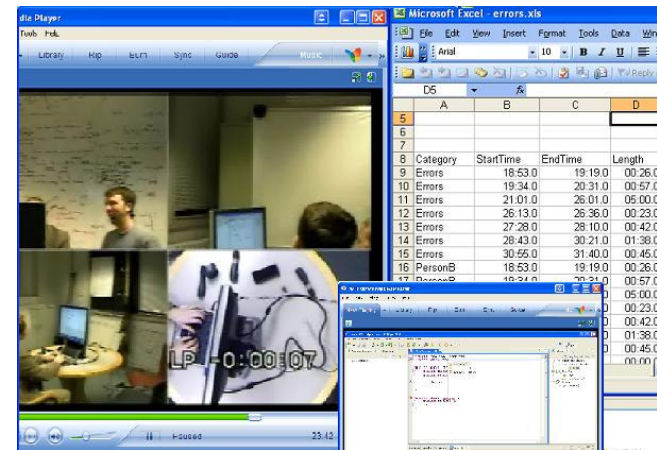
one or more monitors while using a separate program to code the information. This is commonly done using basic video player and spreadsheet applications (Figure 1). Analysis in this way demands approximately four hours to analyze a single hour of video. Much time is spent simply switching back and forth between programs, resulting in loss of attention on the subject matter, and requiring multiple passes on a significant amount of the video data. With experiments generating hundreds to thousands of hours of video footage, the time to analyze video data quickly begins to dominate the period of a typical behavioral study, and often becomes a major labor cost of such studies.

VACA addresses several significant problems with this type of video analysis. It combines the video viewing and the annotation into one system to eliminate the switching costs. A timeline view of the events facilitates drawing qualitative conclusions, and external sensor data can be imported into the system to act as an additional set of annotations.

The timeline, which affords a direct manipulation interface to the video data, combined with the ability to import external data, provides a unique means for constructing an automatic index to the videos. Such an index has the potential to increase the efficiency of video analysis tasks.

#### *Definition of Terms*

Before describing the system, it is helpful to have definitions for common terms in usability analysis.



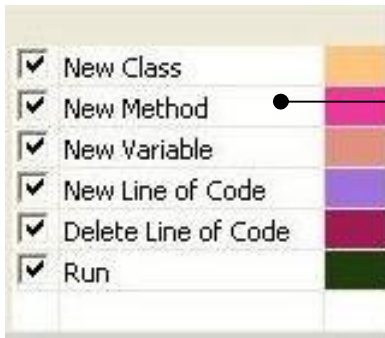
**Figure 1.** Traditional tools for video analysis. Consists of one or more video players along with a spreadsheet application. Not particularly easy to manage.

**CODE:** Also called a code category. Refers to a type of behavior that a researcher is interested in observing. For example, a researcher analyzing videos of pair programmers might want to note occurrences of disagreements between the programmers. One of her code categories would be “disagreements”.

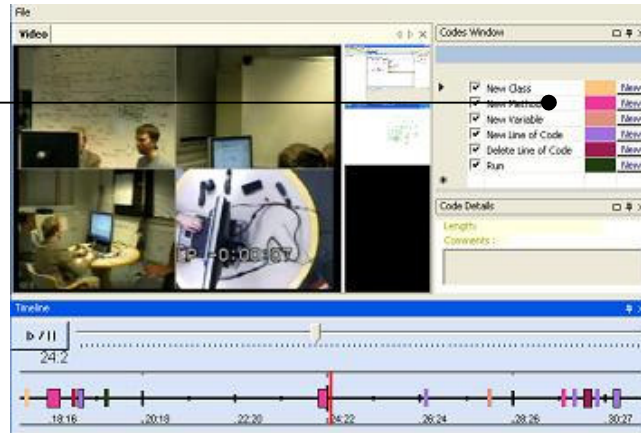
**EVENT:** An instance of a particular code. Usually associated with a time interval during which the event occurred. In the example above, there would be a “disagreement” event for every disagreement the pair had.

#### **Related Work**

The timeline interface in VACA drew inspiration from the Silver video editor [1]; however, this system was not designed for analysis, annotation, or multiple video



VACA can import external sensor data. This data appears as an additional set of codes in the Codes Window. Here, logging events from an Eclipse programming session have been imported and are shown.



**Figure 2.** The VACA program. Videos are on the top left, the timeline is at the bottom, and the codes are on the right.

streams. The Anvil analysis tool [2] also has a timeline and supports annotation, but not multiple streams. Observer [4] and Diver [5] are video analysis tools that support annotation, but not a timeline visualization of that annotation. None of these systems support timeline-based indexing of video with external data.

### Design

With the VACA interface (Figure 2), users designate one stream as the focus stream. This focus video is displayed as large as possible, with the other videos shown as thumbnails off to the side. Clicking a thumbnail switches that video to the focus. The size of the thumbnails can be adjusted if close attention is needed to two or more videos simultaneously.

A right-hand pane shows a list of the codes that have been created for the video streams. From this pane the

user can instantiate new events for particular codes, and can annotate those events freely.

The bottom pane shows the timeline. The timeline reflects the current temporal location of video playback. All events are displayed on the timeline in a color corresponding to their code category. Events can be shown or hidden on the timeline, facilitating correlation between various codes, and allowing the user to obtain a big-picture view of her codes.

External data can be imported into VACA, and shows up in the main list of codes as additional categories. In general, any external sensor data in the correct xml format can be imported and used to index the videos. The study we conducted for this project used a set of videos from a pair programming session. In this case, the external data from the event is the log of Eclipse events from this programming session. Figure 2 shows VACA as it appeared in the user study. The codes shown were imported from the Eclipse log.

### Methods

We conducted a preliminary comparative evaluation with 9 undergraduate students. Four tasks were performed by each participant using both the VACA system and existing commonly used tools (Windows Media Player and Excel). 4 of the participants completed the tasks first using the VACA system, and then using the common tools. The other 5 used the common tools first, and VACA second. Each user was given a 5 minute demonstration of the VACA system before using it.

A 30 minute clip of a pair programming session was used as the target for the users to analyze. This

included a video recording of the programmers taken from four different angles, a video of the screen capture from their computer, and a video of the whiteboard capture. Only the video recording of the programmers contained audio. Also, for the common tools setup the videos were not synchronized. Instead, the users were given the timing offsets that would be required to synchronize the videos. This is consistent with common practice. In the VACA system, however, the videos were synchronized, as this was one of the design goals. Finally, users were given the Eclipse logging data only for the VACA system.

### *Tasks*

It was important to choose tasks that would be representative of common analysis patterns of behavioral researchers. Also, as the users were undergraduates, not professionals, the tasks were framed inside of a narrative. This was done to give them insight into how an expert might think about this kind of analysis. The relevant parts of the narrative are reproduced below, edited for length. The four tasks the users performed are displayed in bold.

Welcome! Today you will be analyzing a 30 minute clip of a pair programming session. Your goal for this analysis is to try to get a sense of how productive the pair was, and what factors contributed to their productivity (or lack thereof). To start off, you decide to use 'lines of code produced' as a rough metric to measure their performance.

**Task 1: Determine how many lines of code were produced by the end of this session.**

Ok, so they ended up with only 6 lines of code after a 30 minute session. That's pretty low for the simple task they are trying to

accomplish. It is possible that they wrote a lot of code, but did a lot of editing towards the end of the session. This warrants a closer look.

**Task 2: Determine how many lines of code total were written during this session.**

Hmm... they wrote 8 lines total. So they really were being quite unproductive. Were they having issues with the design, or problems with the programming?

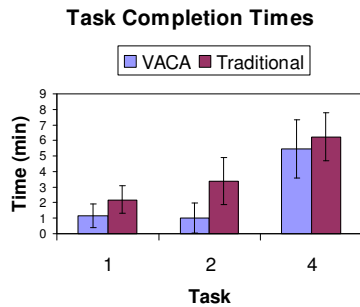
**Task 3: Create a code for "Programming Problems". This will represent problems that the pair ran into while programming. Code the occurrences of "Programming Problem" events for the entire video session. The start time for each event should be when the problem occurs, and the end time should be when it is resolved.**

Interesting. It looks like they spent most of their time dealing with programming problems. No wonder they didn't get much written. Why are they running into so many problems? Is one person encountering more issues than the other, or is it a shared struggle?

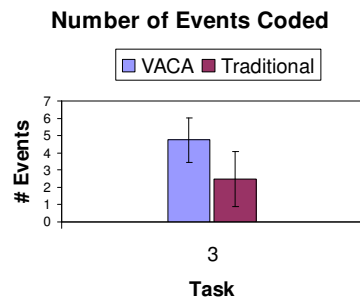
**Task 4: Go back over the events you coded for Task 3, and recode them into the following 3 subcategories: "Person A Problem", "Person B Problem", "Both Problem".**

Ah! I see now. Person A is the one running into all the problems. It appears he is a beginner. Person B is spending most of the time explaining things to Person A. Now it makes sense...

As the users performed the tasks, completion times were recorded. There was also a time limit imposed for each task, to keep the user on track. The limit was 3 minutes for Task 1, 5 minutes for Task 2, 10 minutes for Task 3, and 8 minutes for Task 4, chosen as



**Figure 3.** VACA outperformed the common tools setup for all of the tasks. While only marginally better for task 4, tasks 1 and 2 were completed in about half the time or better using VACA.



**Figure 4.** For task 3, users produced almost twice as many events in VACA as compared to the common tools setup.

reasonable bounds on completion times. If the user did not successfully complete the task in the allotted time, they were stopped, and they proceeded with the next task. Finally, because Task 3 was particularly time consuming, all users were cutoff at the 10 minute limit, and the number of 'programming problems' successfully coded was used for comparison.

Our goals for the study were to

1. Determine how the VACA system performs on common analysis tasks as compared to the common tools setup. Additionally, understanding which kinds of tasks are well suited to VACA and which aren't will help inform further design.
2. Assess the benefit of the Eclipse logging data to the video analysis. We wanted to see if having this form of external sensor data would be useful when performing common analysis tasks.

## Results

We present both quantitative and qualitative results from our evaluation.

### Quantitative Results

When using the VACA system, the average task completion times for tasks 1, 2, and 4 was 1.17 minutes, 1.01 minutes, and 5.46 minutes, respectively. For the common tools setup the times were 2.18 minutes, 3.38 minutes, and 6.24 minutes (Figure 3). The VACA times, as a percent of the common setup times, are 54%, 30%, and 90%. For Task 3, the average number of problems coded was 2.50 for the common setup and 4.75 for the VACA system, or 1.9x the number coded in the common setup (Figure 4).

Since users repeat the same tasks on both systems, it's expected that their times would improve for the second system they use. For those that used the common setup first, the VACA/common setup ratio for the four tasks was 63%, 37%, 2.9x, and 69%. For those that used the VACA system first, these ratios are 41%, 25%, 1.2x, and 117%. Thus, even when the common setup was used after the VACA system, it still performed worse for tasks 1-3. However, this was not the case for task 4.

### Qualitative Results

After the study each user was questioned about their experiences using both systems. All 9 users preferred using the VACA system over the common setup. Opinions ranged from "it's better for some tasks, but maybe not others" to "I don't see how anyone could ever do this stuff in [Excel and Windows Media Player]!" Interestingly, one user commented that VACA was "a lot less stressful than the other way. Trying to switch between videos and Excel to input numbers, while still trying to pay attention to what's happening in the videos – it's just too much."

Not surprisingly, most participants used the Eclipse logging events to complete tasks 1 and 2 in VACA. The two that didn't use the Eclipse events had the longest completion times for these tasks. More interestingly, however, 4 of the 9 participants used the Eclipse events to complete task 3 – the task to code occurrences of "programming problems". They reasoned that problems would occur in proximity to basic programming events, and they used the Eclipse events to jump around and hone in on these problems more rapidly. Unfortunately, this did have an adverse effect for 2 of the users, who

skipped over some “programming problem” events due to this kind of seeking.

### Conclusions

Users were more efficient at completing common video analysis tasks using VACA than using the common setup of Windows Media Player and Excel. 3 of the 4 tasks were close or better to a factor of 2 improvement.

Interestingly, the last task – that of refactoring a code into a subset of codes – was only marginally better in VACA, and in some cases even worse. It appears that the VACA interface doesn’t provide much benefit over the common tools system for tasks that involve a known location in the video. For example, the 4<sup>th</sup> task mainly involved seeking to specific locations in the video file, and then watching the video from there. This task was accomplished in Windows Media Player just as easily as in VACA. The benefit seems to come with tasks that require some degree of searching through the video. In these situations, helpful context in VACA is provided by other codes or events, the Eclipse logging data, or simply the location of events on the timeline. And, as with the 3<sup>rd</sup> task, this context seems to be useful in speeding up the search task.

Users were frustrated by the lack of editing capabilities in VACA. If they made a mistake, or wanted to change a start or end time for a particular event, it wasn’t easy to do. One user pointed this out as a major benefit of having data in a numerical, spreadsheet format. It’s very simple to change the data to your liking. While VACA does have editing capabilities, none of the participants made use of them, leading us to conclude that the current editing features are unusable.

### Future Work

Qualitatively, it appears that the Eclipse logging data was useful in completing the given tasks. Next we plan to perform a quantitative study to measure the amount to which the logging data improves video analysis.

As was suggested by this study, for the next iteration we plan to integrate a simple spreadsheet view of the codes to provide easier editing capabilities.

So far we have only studied how beginners interact with VACA. While the results have been very encouraging with novices, and they have served to inform some issues of the design, the best feedback we can get is from a longer study with actual behavioral researchers, using the system for their real world analysis tasks. To this end, we are planning to have VACA used as the analysis tool for a pair programming research project being conducted here at Stanford.

### Acknowledgements

We would like to thank Jan Chong for her help in designing the experiment tasks and Scott Klemmer for his tireless advice and many helpful comments.

### References

- [1] Casares, J., *et al.* Simplifying Video Editing Using Metadata, in *Proc. DIS 2002*, p. 159-166.
- [2] Kipp, Michael. Anvil video annotation system. <http://www.dfki.de/~kipp/anvil>.
- [3] Noldus, L.P., *et al.* The Observer Video-Pro, in *Behav Res Methods, Instrum Comput 2000*, 32, p. 197-206.
- [4] Pea, R., *et al.* The DIVER Project: Interactive Digital Video Repurposing. *IEEE Multimedia*, 11(1), p. 54-61