

Results from the 2006 Classroom Evaluation of Hackystat-UH

Philip Johnson
Collaborative Software Development Laboratory
Department of Information and Computer Sciences
University of Hawaii
johnson@hawaii.edu

CSDL-07-02

<http://csdl.ics.hawaii.edu/techreports/07-02/07-02.html>

Last update: 12/22/2006 12:42:26 PM

1.0 Abstract

This report presents the results from a classroom evaluation of Hackystat by ICS 413 and ICS 613 students at the end of Fall, 2006. The students had used Hackystat-UH for approximately six weeks at the time of the evaluation. The survey requests their feedback regarding the installation, configuration, overhead of use, usability, utility, and future use of the Hackystat-UH configuration. This classroom evaluation is a semi-replication of an evaluation performed on Hackystat by ICS 413 and 613 students at the end of Fall, 2003, which is reported in [Results from the 2003 Classroom Evaluation of Hackystat-UH](#). As the Hackystat system has changed significantly since 2003, some of the evaluation questions were changed.

The data from this evaluation, in combination with the data from the 2003 evaluation, provides an interesting perspective on the past, present, and possible future of Hackystat. Hackystat has increased significantly in functionality since 2003, which has enabled the 2006 classroom usage to more closely reflect industrial application, and which has resulted in significantly less overhead with respect to client-side installation. On the other hand, results appear to indicate that this increase in functionality has resulted in a decrease in the usability and utility of the system, primarily due to inadequacies in the server-side user interface. Based upon the data, this report proposes a set of user interface enhancements to address the problems raised by the students, including Ajax-based menus and parameters, workflow based organization of the user interface, real-time display for ongoing project monitoring, annotations, and simplified data exploration facilities.

2.0 Methodology

At the end of the Fall 2006 semester, the students in ICS 413 and ICS 613 were contacted by email and asked to respond to the following questionnaire soliciting their opinions regarding the Hackystat-UH configuration. Response was optional, but the students were offered extra credit points for providing their opinions. The students were asked to reply within five days. Fourteen out of the fifteen students contacted provided responses.

The complete questionnaire follows:

In order to better understand the strengths and weaknesses of the current version of Hackystat, and to help guide future improvements, please take a few minutes to answer the following questions. Your identity will be removed before performing any analysis on this data. There are no right or wrong answers: we want to know what your personal experience was.

Most questions ask you to respond with a number from 1 to 5, where 1 indicates the "best" and 5 indicates the "worst". The last question of each section requests your comments as the response.

I. INSTALLATION/CONFIGURATION

Please provide me with your opinions regarding the installation of Hackystat sensors and configuration of the Hackystat server.

1. Installing the Eclipse IDE sensor was:
(Very Easy) 1 2 3 4 5 (Very Difficult)
2. Installing the Ant sensors (JUnit, SCLC, Emma, etc.) were:
(Very Easy) 1 2 3 4 5 (Very Difficult)
3. Configuring Hackystat to track our team's work

(i.e. defining the project, and configuring the workspace roots) was:
(Very Easy) 1 2 3 4 5 (Very Difficult)

4. Please provide any feedback you can on the problems you experienced during sensor installation and server configuration, as well as any suggestions you have to make this easier in future.

II. OVERHEAD OF USE

In this section, I am interested in learning about the "overhead" you experienced with Hackystat--in other words, how much work was required after installation and configuration to gather data and perform analyses:

5. The amount of overhead required to collect Hackystat data was:
(Very Low) 1 2 3 4 5 (Very High)
6. The amount of overhead required to run Hackystat analyses was:
(Very Low) 1 2 3 4 5 (Very High)
7. Please provide any feedback you can on Hackystat overhead after installation and configuration, as well as any suggestions you have to reduce the overhead more in future.

III. USABILITY and UTILITY

This section asks for your opinion on the usability and utility of the primary analyses used to track your progress. I define "usability" to mean the ease of invoking an analysis and understanding what the results mean. I define "utility" to mean the usefulness of the analysis; does the analysis provide information that is actually helpful to you.

8. The TraffoMeterReport telemetry analysis (showing trends in project/process data) was:
(Highly Usable) 1 2 3 4 5 (Not Usable At All)
(Highly Useful) 1 2 3 4 5 (Not Useful At All)
9. The DailyProjectDetails report (showing a single day's process/project data) was:
(Highly Usable) 1 2 3 4 5 (Not Usable At All)
(Highly Useful) 1 2 3 4 5 (Not Useful At All)
10. Please provide any other feedback you can on the usability and utility of these or other Hackystat analyses, as well as any suggestions you have on how we can improve usability and utility in future.

IV. FUTURE USE

In this section, I am interested in learning whether you would consider Hackystat to be feasible (i.e. appropriate, useful, beneficial) for use in a professional software development context.

11. If I was a professional software developer, using Hackystat at my job would be:
(Very Feasible) 1 2 3 4 5 (Not Feasible at All)
12. Please provide any other feedback you can on the feasibility of Hackystat in a professional setting, as well as any suggestions you have on how I could improve its feasibility in future.

Sections I, II, and IV were virtually unchanged from the 2003 evaluation. Section III was changed to reflect the analyses used by students in this semester (the TraffoMeter and DailyProjectDetails reports).

3.0 Results

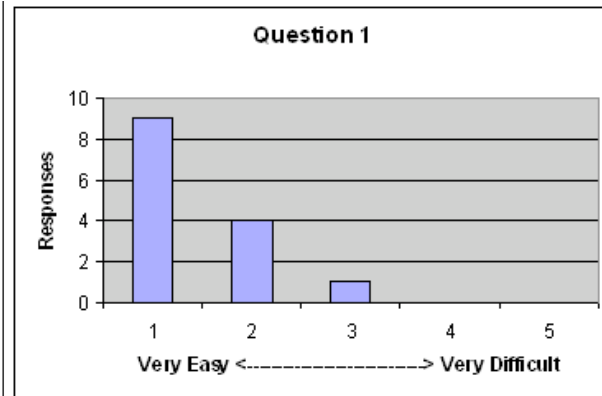
This section presents the responses from the respondents to each of the questions. For the "short answer" questions, I corrected misspellings and minor grammatical errors to improve readability.

3.1 Installation/Configuration

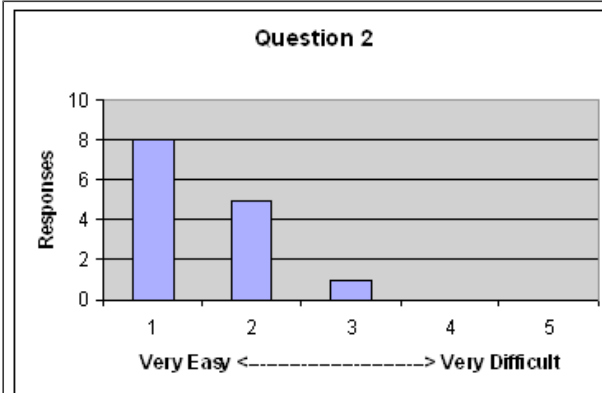
Please provide us with your opinions regarding the installation of Hackystat sensors and configuration of the Hackystat server.

Question	Response
----------	----------

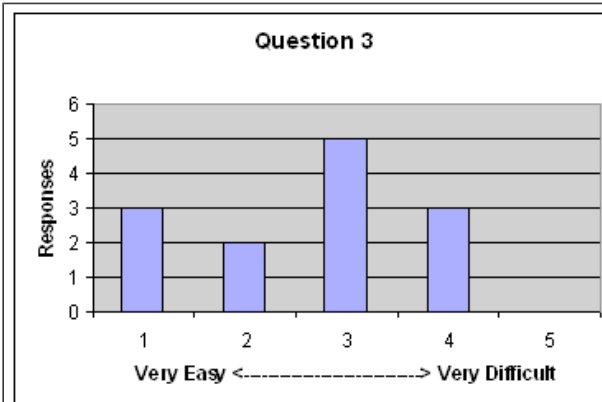
1. Installing the Eclipse IDE sensor was:
(Very Easy) 1 2 3 4 5 (Very Difficult)



2. Installing the Ant sensors (JUnit, SCLC, Emma, etc.) were:
(Very Easy) 1 2 3 4 5 (Very Difficult)



3. Configuring Hackstat to track our team's work (i.e. defining the project, and configuring the workspace roots) was:
(Very Easy) 1 2 3 4 5 (Very Difficult)



4. Please provide any feedback you can on the problems you experienced during sensor installation and server configuration, as well as any suggestions you have to make this easier in future.

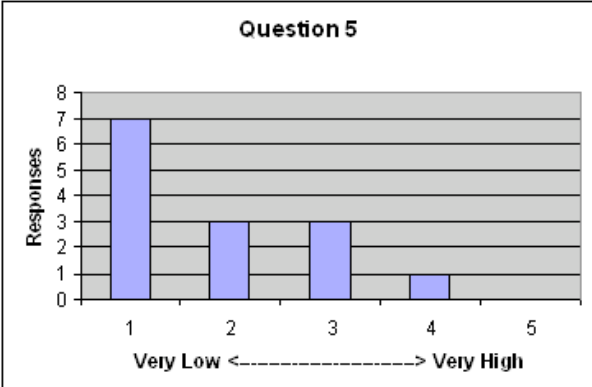
- Overall, there were no problems with the installation process and continuous work with sensors. The only issue was the long time required by the SVN sensor to do its job.
- Overall easy, except that HackyStat installer didn't function correctly on my home computer running Ubuntu linux. Thus, no sensors would install.
- It was easy to overlook the usermaps file. Also, its not very clear if other group members are contributing to the group user data pool. I've also some privacy concerns since I believe to track a team's work each group member has to have all the group members hackystat codes so that the data is separated correctly.
- As mentioned above, HackyInstaller didn't function properly on my Ubuntu linux machine. It could write to the files correctly, but continually lost the information and claimed the sensors were not installed giving some versioning problem. I emailed you previously about this. Maybe Ubuntu is too open-source for your installer ;-).
- Also, installing the sensors in project files is somewhat annoying, but its a one-time cost thing.
- The difficulties we faced were: the unexpected Hackystat project definition page interface that we thought had the correct workspace roots defined but we actually didn't (we called you over to our cubicle for this one). You can have the correct Workspace roots selected, but until you click the save button the change doesn't take effect. Another issue was configuring workspace roots for SVN data collection, since it has to match the path of the person running the svn sensor. One added problem

was that the project was defined by one member, and he wasn't generating any metrics for the new Traffometer-Common project, thus he couldn't configure a new Workspace root for it. I kludged this by temporarily changing my Hackystat key to his and then generated some metrics and switched it back to my own, but that's pretty gross.

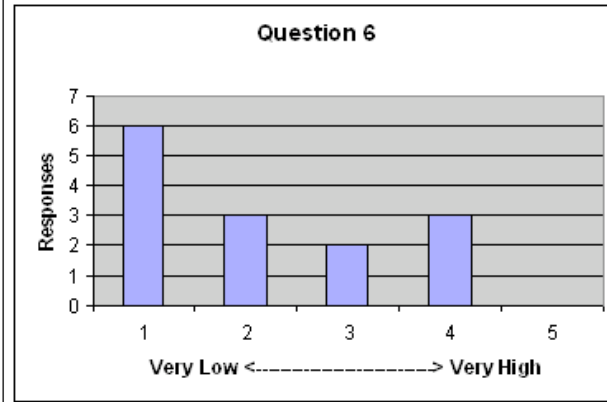
7. The web interface doesn't seem very intuitive. We were confused by some selection boxes that looked like they were displaying current settings, but were actually displaying possible values that we needed to select. (I could elaborate on this if necessary)
8. Is it possible while installing sensor for svn, to add a service in the system so that the subversion commits can be automatically obtained other than by manually submitting to the Hackystat server?
9. Setting up the root space for svn sensor to work is little tricky. In order to show the number of commits from every members of the team, each member must have the workspace of the person who runs the svn.build.xml in his or her workspace root. There should be an option for the users to make their hackystat key so it is easy to remember the key.
10. While configuring hackystat in general was very easy the exception was the subversion sensor setup
11. The GUI enables us to easily install them. But the configuration is not so easy as the installation. I felt like there were too many things on the page. If the page is much more intuitively understandable, it would be nice. If there is a way to make sure that a configuration is appropriate, it would be nice.
12. The sensor installation was easy. I don't remember any particular problems. But maybe it would be good to have a pre-configured installation (configured by a manager).. Say, sensors need to be set for Ant, JUnit and Eclipse. So when a member of a group starts an installer, it detects Ant, JUnit and Eclipse in the system, and says "I'm gonna install sensors for Ant, JUnit and Eclipse". Or if it doesn't find the required tool, it says so. In this case the manager of the group can be sure that all members install all the required sensors.
13. Adding the sensors wasn't too bad thanks to the amount of documentation. Adding the workspace roots was kind of hard because of the interface. Using a file browser might make things easier.
14. I had no problems during installation.
15. Sensor installation consisted of virtually zero overhead and after the hackyInstaller was located took virtually zero time.
16. Having all of the links available on one page would be nice. Some of the [tools] required downloading and installing of other software that was in the read me file of that [tool]. It would be better to know from the beginning that additional components were needed.
17. It would be nice if we could receive instant feedback as to whether sensors are in working order by logging onto the HackyStat server, with the server acknowledging that it has received information.

3.2 Overhead of Use

In this section, we are interested in learning about the "overhead" you experienced with Hackystat--in other words, how much work was required after installation and configuration to gather data and perform analyses:

Question	Response												
5. The amount of overhead required to collect Hackystat data was: (Very Low) 1 2 3 4 5 (Very High)	<div style="text-align: center;">  <table border="1" style="margin: 10px auto;"> <caption>Question 5 Response Data</caption> <thead> <tr> <th>Rating</th> <th>Number of Responses</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>7</td> </tr> <tr> <td>2</td> <td>3</td> </tr> <tr> <td>3</td> <td>3</td> </tr> <tr> <td>4</td> <td>1</td> </tr> <tr> <td>5</td> <td>0</td> </tr> </tbody> </table> </div>	Rating	Number of Responses	1	7	2	3	3	3	4	1	5	0
Rating	Number of Responses												
1	7												
2	3												
3	3												
4	1												
5	0												

6. The amount of overhead required to run Hackystat analyses was:
(Very Low) 1 2 3 4 5 (Very High)



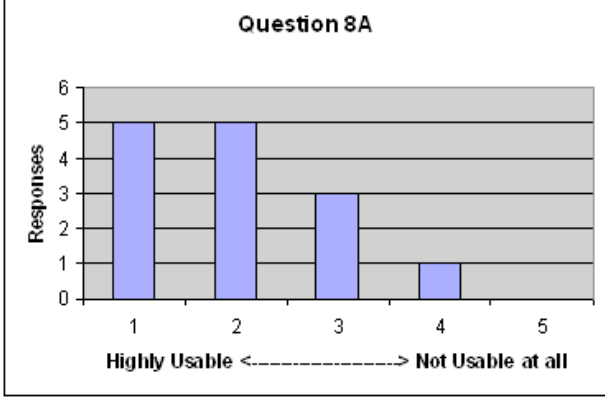
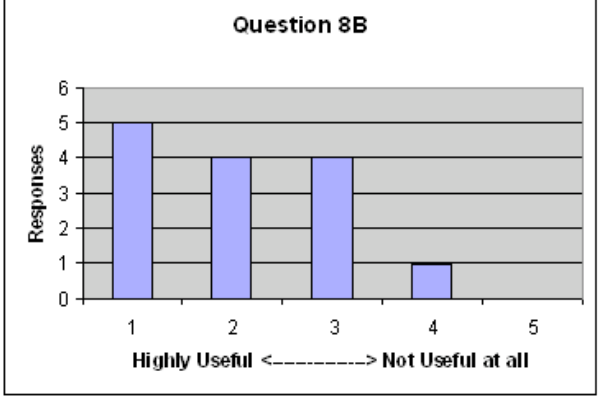
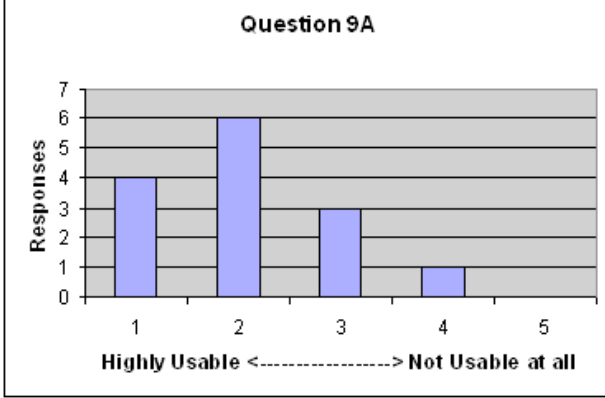
7. Please provide any feedback you can on Hackystat overhead after installation and configuration, as well as any suggestions you have to reduce the overhead more in future.

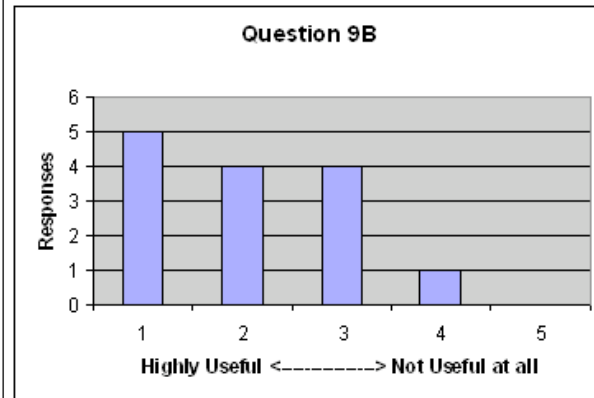
1. To make the process of analyzing the system easier, we designed a script that was scheduled by cron to run and there was no overhead at all, the system was updated and analyzed overnight.
2. For a person that uses the IDE mostly, its rather burdensome to need to launch several terminal windows and run the ant tasks. Many group members neglected to run these tasks on a regular basis leading to incomplete or inconsistent data. We eventually mandated each group member do automated builds which cut down on the overhead. Perhaps it would be nice if there was some automated build extension to hackystat that automated that process. TSR applications can be a bit loathesome, but convenient on occasion.
3. The interface could be more user-friendly. The dates remain fixed. It'd be great if there were an option for from date that started at the project start date as well as an end date that was the current day. If you give a date outside the start date (as if people could actually remember their start date), you get an error. There's also a ton of analysis options, most of which are never used and if you try to use them they produce some wonderful error messages. For the casual user, this huge list of analysis is annoying because they must wade through it to find their one analysis that they use all the time. It's reminiscent of the VAX architecture. You've got every possible analysis for everyone. Perhaps it would be good to have a separation between simple analysis (not requiring input other than dates), and the advanced analysis (requiring advanced input).
4. When doing active development, there is little overhead in collecting metrics. However, to ensure that there are no days without data, someone has to ensure that analyze.build.xml gets run every day (including holidays & weekends) and also the svn sensor needs to be run once daily for each project. I also get lots of hackystat timeouts with the ant sensors. I'd say maybe 20% of the time I get a timeout, which can be 30 seconds or so and is quite painful. Given that the server is up during these periods, and there are no apparent Internet issues at the time, I'm guessing this is some sort of server issue. I've also started to notice problems where any ant command will pause immediately after echoing the name of the build file. Sometimes it hangs so long I just ctrl-c out and try again which usually works. I can't help but think that this is some sort of Hackystat sensor issue (maybe trying to send offline data?). Perhaps this could be tracked down by printing a status message when sending offline data?
5. We needed to run the svn sensor manually and sometimes when offline the hackystat sensor would cause the build to freeze for a few minutes while it tried to connect to the server.
6. Everything was easy once configured.
7. I don't see any problems with this.
8. The time to reflect the sent data to the view takes a while. I think the overhead of Hackystat is trivial in terms of time which it need to respond to the user request. The most important overhead is the time which a user need to learn how to use it and what each data means.
9. I found the Workspace Root configuration somewhat confusing. Especially when in one of the text boxes it was not enough that the folders appear in this box, I also had to select them - and it was not clear to me. Maybe there could be a list of checkboxes or something. And generally the user interface was not very friendly (to me at least). The buttons Analyze, GO etc at the most right of the screen are kinda painful to operate (I didn't even see them at first). And I think that it would be better to separate the Analyses page into pages or tabs Project, Telemetry, Validation.. now the page contains TOO many widgets, which is overwhelming and frightening (to me at least).
10. Once the sensor was added to the build file of the tool, it was really easy to send data to the server. Getting info through the website was easy as well.
11. Sometimes, at the start of the build process, Hackystat sensors try to send the offline, cached data from previous builds to the server, that would just hangs the build (or at least that was what we thought) and we had to CTRL-C to break it and restart.
12. Most of the overhead of running Hackystat analyses was the configuration of the Ant sensors to ensure that data was being passed to the servers, which was initially difficult but progressively easier.
13. It was hard to figure out how to view the data from Hackystat. Figuring out where to go on the webpage and which "analyse" button to press, then the dates where wrong, then the variables were wrong. After finally asking people how to get it to work then

reading the data took some time getting used to. Because i the graph kept climbing i didn't get what it was telling me right away.
 14. It would be nice to have a GUI configuration for the Hackystat sensor for those of us not 100% familiar with XML or with the functionalities of each sensor

3.3. Usability and Utility

This section asks for your opinion on the usability and utility of the three primary analyses used by Hackystat. We define "usability" to mean the ease of invoking an analysis and understanding what the results mean. We define "utility" to mean the usefulness of the analysis; does the analysis provide information that is actually helpful to you.

Question	Response																								
<p>8. The TraffoMeterReport telemetry analysis (showing trends in project/process data) was: (Highly Usable) 1 2 3 4 5 (Not Usable At All) (Highly Useful) 1 2 3 4 5 (Not Useful At All)</p>	<div style="text-align: center;">  <p>Question 8A</p> <table border="1"> <tr><th>Rating</th><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><th>Responses</th><td>5</td><td>5</td><td>3</td><td>1</td><td>0</td></tr> </table> </div> <div style="text-align: center;">  <p>Question 8B</p> <table border="1"> <tr><th>Rating</th><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><th>Responses</th><td>5</td><td>4</td><td>4</td><td>1</td><td>0</td></tr> </table> </div>	Rating	1	2	3	4	5	Responses	5	5	3	1	0	Rating	1	2	3	4	5	Responses	5	4	4	1	0
Rating	1	2	3	4	5																				
Responses	5	5	3	1	0																				
Rating	1	2	3	4	5																				
Responses	5	4	4	1	0																				
<p>9. The DailyProjectDetails report (showing a single day's process/project data) was: (Highly Usable) 1 2 3 4 5 (Not Usable At All) (Highly Useful) 1 2 3 4 5 (Not Useful At All)</p>	<div style="text-align: center;">  <p>Question 9A</p> <table border="1"> <tr><th>Rating</th><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td></tr> <tr><th>Responses</th><td>4</td><td>6</td><td>3</td><td>1</td><td>0</td></tr> </table> </div>	Rating	1	2	3	4	5	Responses	4	6	3	1	0												
Rating	1	2	3	4	5																				
Responses	4	6	3	1	0																				



11. Please provide any other feedback you can on the usability and utility of Hackystat analyses, as well as any suggestions you have on how we can improve usability and utility in future.

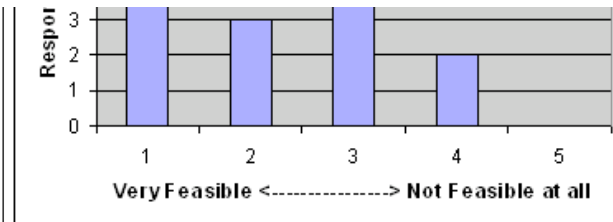
1. Its very difficult to tell what is going on from the data. It's not like we're given any classes or instruction on how to interpret such data. Basically we just look at the pretty squiggles and say, "Gee, wow. Uh, this looks like we're doing a bang-up job on unit-test driven design." Given the wild variation in coding styles and tools, one person may look dreadful on one graph, but may appear to be the group leader on another graph. It might be nice if the graphs supported some sort of code-driven annotations. So, if a person knew something big happened, they could send a message to the hackystat server explaining what happened on that day. Perhaps if you then put your mouse over that day's date you could see the message. Similarly, if you had a graph of SVN commits, it'd be neat if you could see the commit message when you place your mouse over the graph dot representing that commit..
2. The graphs were easy to generate and understand.
3. It was nice to be able to compare the work done each day with the work done the previous day.
4. MemberActiveTime data is not accurate. I have read the documentation and know that the hackystat designers have figured this out.
5. The user is required to learn how to effectively use Hackystat and analyze the data. If the user could more intuitively use it, they would be much more convenient. I learned a lot from Dr. Johnson's explanation of the Hackystat data through ICS 613 group work, not from the document.
6. Both the telemetry analysis and the daily report were very useful. It would be nice if both could list the tools used to collect the displayed data. Also, a quick link to the analysis from the daily report would be nice to quickly access info about the past few days.
7. There are too many types of preconfigured charts without any easily accessible explanation except for the short name. A list of explanation for each of them would be great.
8. Finding and interpreting the chart data from the hackystat website continues to be the most difficult utility in terms of usability. If there were perhaps some way to tailor a specific report output defined individually for each project (this might already be possible i'm not sure if i found a way to do it though), the reporting process could be simplified.
9. The Hackystat website interface could be improved. For example, using a GUI calendar to pick & enter dates instead of using drop-down boxes. It also scares first-time users with the plethora of options & dropdown boxes used on the website
10. The interface doesn't seem very friendly to new users, which might preclude them from ever wanting to become an advanced user. Graph annotations might be helpful in analyzing the data.

3.4 Future Use

In this section, we are interested in learning whether you would consider Hackystat to be feasible (i.e. appropriate, useful, beneficial) for use in a professional software development context.

Question	Response						
12. If I was a professional software developer, using Hackystat at	<p style="text-align: center;">Question 11</p> <table border="1"> <thead> <tr> <th>Rating</th> <th>Uses</th> </tr> </thead> <tbody> <tr> <td>4</td> <td>4</td> </tr> <tr> <td>5</td> <td>5</td> </tr> </tbody> </table>	Rating	Uses	4	4	5	5
Rating	Uses						
4	4						
5	5						

my job would be:
 (Very Feasible) 1 2 3 4 5 (Not Feasible at All)



13. Please provide any other feedback you can on the feasibility of Hackystat in a professional setting, as well as any suggestions you have on how we could improve its feasibility in future.

- In my opinion the whole idea of HackyStat is very nice. There is no overhead in installing the system and doing everyday analysis. The measurements are precise and pictures are very clean and meaningful. Of course the interpretation of gathered data is the main issue here, and if done in proper way it would help in planning the workflow a lot. I started to use this HackyStat in all my projects to monitor activity in overall process in time. I hope in my professional carrier I would be allowed to use the tool.
- The tools seem very language dependent. If someone is not coding in JAVA, I think the usefulness of the hackystat project would be very very low. These tools might be useful if I had to justify my time to someone. However, since the graphs are difficult to interpret, they'd just have to take my word that I say they make me look really good.
- It seems it would be easy to mislead the numbers. If a person's junit invocations were low, just run junit a thousand times. It doesn't matter that the person didn't actually code any of the tests or even care about the results. Similarly, other sensors can be fooled given enough time or ingenuity. It might be nice to see a graph of code percentages broken down by each user. So if every group member contributed about the same percent of code, their lines would be very close together. If one member "owns" 80% of the code, then there might be some explaining to do. Perhaps this information code be obtained through using the SVN blame/praise information. Of course, knowing how the blaime/praise works on line-level changes, it would be easy to deceive this sensor by making changes to every line, and then putting them back. Ah well. However, assuming an honest batch of developers, it might be an interesting graph.
- Privacy and management buy-in would probably be the two biggest issues. While in class environment it might be acceptable to have individually trackable metrics, in industry that seems like an invitation to measurement disfunction. I've even heard people joke in 613 about writing bots to generate Hackystat metrics, but if management in industry had access to the metrics it would probably move past the joking stage.
- I think the biggest thing would be to improve the web application so the interface is more intuitive, even if this also made is a little less flexible.
- Provide more managerial analysis tools. Automatically provide more process control advise according to the data.
- Hackystat is good for java development, eclipse, and ant tasks. But I don't know if works with makefile and microsoft visual studio. If it is, I would recommend it to my friends who use microsoft visual studio or some other IDEs.
- If the Hackystat configuration and data is much easier to understand, I would use it because I understand it is useful. The version with limited functionalities would be better for a beginner.
- Maybe it would be useful to have a feature of sending a developer warnings when the required parameters are not reached (not enough tests executed, not enough hours of activity for the last day) - sounds a little like slavery though.
- Seeing other people's data was interesting in a group. I can imagine that having the data of your subordinates (when in a management setting) could be very useful to track progress.
- I think Hackystat is very useful for project managers, especially for when they need to generate reports. But for team members, it might not be as useful since and it is very possible that they will forget to check or fix if there are problems with sensors.
- Anytime accountability is introduced into the work environment, especially in software development where lots of research is taking place alongside the developmental process, the process that includes a metric recording function soon overshadows the others because the actions one undertakes are now recorded by the hackystat system (files being opened, files being edited, files being compiled). Just as long as the prospective developers implementing this software understand that its to be used as a metrics gathering device about the SOFTWARE and not its EMPLOYEES we can stave off any trends towards a neo-orwellian software developmental environment.

4.0 Interpretation of the 2006 data

4.1 Experimental Limitations

Before drawing any conclusions from this data, it is important to recognize the limitations of this study, which are quite similar to the limitations associated with the 2003 study.

First, this data is drawn from a limited sample size of 14 students in software engineering classes at the University of Hawaii. The subjects therefore have a relatively narrow and homogeneous background in software development.

Second, the context in which they used the system was a course project. Course projects tend to be smaller, narrower in scope, and with less pressure on the developers than an industrial context. It is one thing to get a poor grade for doing a poor job, it is another thing to lose your job for doing a poor job. In addition, students are not working full-time on the system; the development project is just one assignment among several.

Third, the administration of the questionnaire was performed by the designer of the system under study, who was also the instructor for the class. In addition, responses were not provided anonymously, but rather emailed back to the instructor/designer. This raises the question of whether the responses are biased, either consciously or unconsciously, in order to "please" the instructor/designer who would presumably be gratified by positive responses to the questionnaire.

These are all major limitations on the external validity of the responses. They do not make the results meaningless, but rather help provide a perspective on how to gain additional evidence in future that would confirm/disconfirm these initial findings. For example, it would be helpful to deploy Hackstat-UH in a classroom setting in another University, and then gather data anonymously from the participants using someone other than the instructor in order to avoid the potential for bias present in this study. Other insights into future research directions will be covered in an upcoming section.

4.2 Conclusions regarding Installation/Configuration

The data from this section indicates that client-side sensors were easy to install, with the exception of the Subversion sensor, but that configuration on the server side (particularly of the Workspaces) was more difficult.

Our principal conclusion is that the hackyInstaller GUI has made sensor installation relatively straightforward, but that project and workspace configuration is still difficult.

4.3 Conclusions regarding overhead of use

The data from this section indicates that the overhead of use was small to very small for most of the users. The comments indicate that the complexity of the user interface on the server side was a primary contributor to overhead. Students also indicated that an automated daily build mechanism would be useful to eliminate the need to manually invoke certain sensors (such as the size counting sensor or the SVN sensor) on a daily basis.

Our principal conclusion is the server-side user interface is the most significant contributor to the overhead of use.

4.4. Conclusions regarding usability and utility

Recall that the survey defined "usability" to mean the ease of invoking an analysis and understanding what the results mean. The survey defined "utility" to mean the usefulness of the analysis: does the analysis provide information that is actually helpful to you. Although the Hackstat-UH configuration actually provides many analyses, there were only two analyses that were actively discussed in class and which the students were expected to use. These two analyses were the subject of the questions in this section.

Numerically, the two analyses scored well with respect to both usability and utility. The comments indicate that the user interface continues to be a barrier, and that further support in interpreting the data would be helpful.

4.5 Feasibility in a professional software development context

The numerical data indicates that most students think Hackstat would be at least somewhat feasible in a professional setting. A variety of concerns were raised: privacy concerns, the difficulty in interpreting the metrics, and the complexity of the user interface were all viewed as obstacles to professional adoption.

5.0 Comparison of the 2003 results to the 2006 results

5.1 Hackstat in 2003 vs. 2006

To usefully compare the data from 2003 to the data from 2006, it is necessary to understand the changes that have been made to the Hackstat system in the past three years.

First, the number of sensors, tools, and metrics used by the students has grown significantly since 2003. In 2003, they collected four distinct measures from four tools: Active Time (from Eclipse), Unit Tests (from JUnit), Coverage (from JBlanket), and size (from LOCC). In 2006, three additional measures and five additional tools were added: Build (from Ant), Commit (from Subversion), and Code Issues (from PMD, Checkstyle, and FindBugs). Thus, in 2006 they collected a total of seven distinct measures from nine tools. In

addition, the "ActiveTime" measure was replaced by the more sophisticated "DevTime" measure, which more accurately tracked developer effort than ActiveTime by utilizing data collected from multiple tools.

Second, the analyses used by students has changed since 2003. In 2003, the system was oriented around a set of "Course" analyses that were tailored to an educational setting. These analyses presented summaries of the individual team project metric data to-date in tabular form, and also presented comparisons of all of the course projects. The data from 2003 indicated mixed results for these course analyses with respect to utility and usability. In 2006, the Software Project Telemetry system was used to present the data. This interface is somewhat more complicated to use and interpret, but is also a more general interface that would be equally applicable to an industrial setting.

Third, in 2003 the students had to manually install the sensors. In 2006, they used hackyInstaller, a GUI system that we implemented to simplify client-side installation.

Fourth, in 2003, the principal analysis provided a tabular representation of "to date" values for one or more of the course projects, as illustrated below:

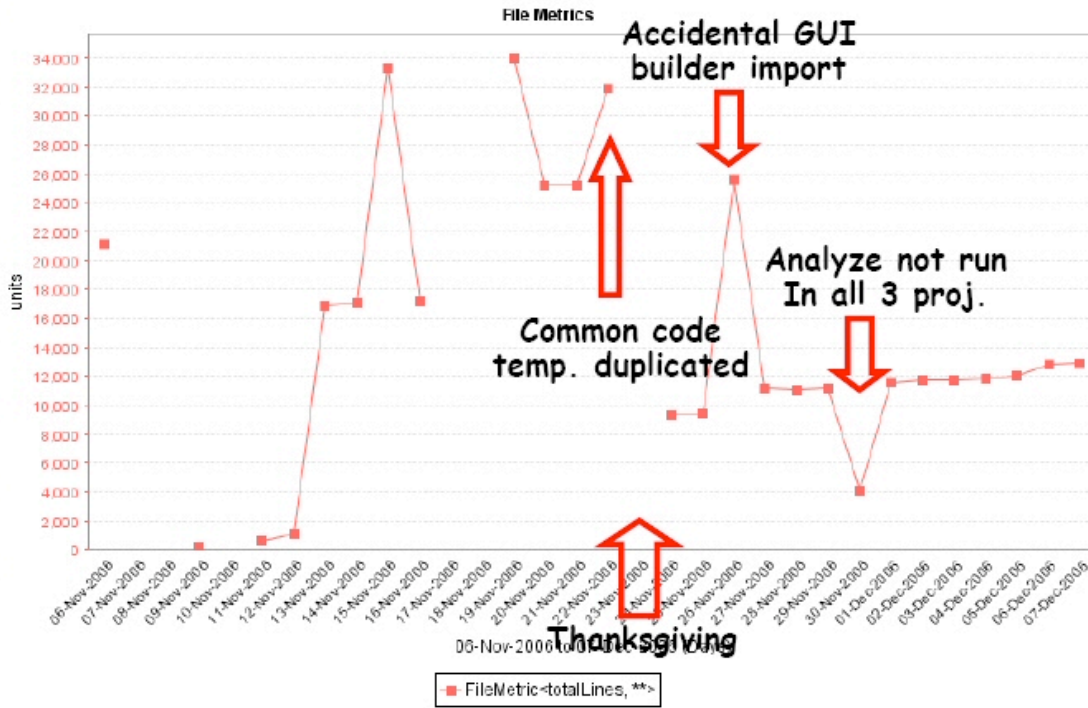
The screenshot shows the HackyStat web application in Microsoft Internet Explorer. The page title is "HackyStat - Microsoft Internet Explorer". The browser's menu bar includes File, Edit, View, Favorites, Tools, and Help. The page content includes the HackyStat logo, the user email "takuyay@hawaii.edu", and the title "Course Project Analysis". Below the title is a navigation menu with links for "analyses", "preferences", "alerts", "extras", and "help". The main content area is titled "Course Project Analysis: Comparative analyses for classroom projects (more...)" and features an "Analyze" button. The form includes the following fields:

- Course: ics413-613
- Project prefix: Sitewatch
- Comparison: Project To Date
- Start Day: 01, November, 2003
- End Day: 08, December, 2003

Below the form is a table with the following data:

Project	Active Time (hrs)	Classes	Methods	LOC	Tests	Coverage
	163.4	62	263	2371	30	94%
	95.7	41	157	1456	18	98%
	128.2	57	220	1948	23	36%
sitewatch-ewalu	117.4	79	322	2438	32	86%
	28.4	22	83	884	9	81%
	97.2	54	189	1632	19	99%
	118.7	63	237	2190	25	84%
	169.7	53	227	2191	23	72%
	95.2	49	205	1698	20	31%

In 2006, the principal analysis was based upon software project telemetry, which would show trends over time. The following image shows one of approximately a dozen different charts that the students would use to analyze and interpret their collected data:



This screenshot was extracted from a Powerpoint presentation from a group to the class. Note how this group annotated the image with information to help explain changes in metric values. The telemetry analysis contains much more information than the "snapshot" course analysis, but is also significantly more difficult to interpret.

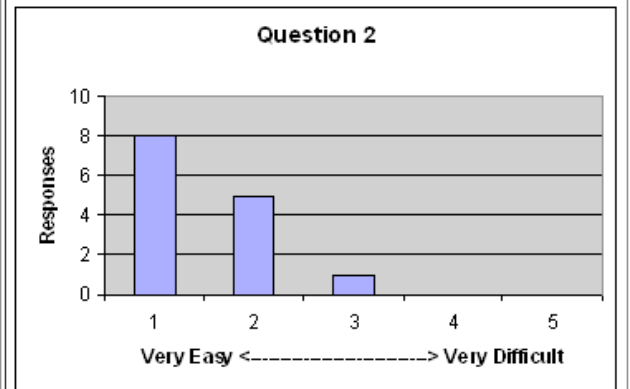
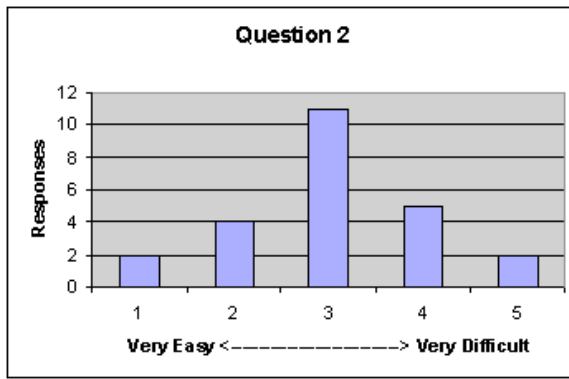
5.2 Comparison of the empirical results

The next section presents a comparison of the histogram data from 2003 to the histogram data from 2006. At this time, no statistical tests for significant difference have been performed. Thus, any differences claimed between the data sets based upon the "shape" of the histograms are tentative and await statistical confirmation.

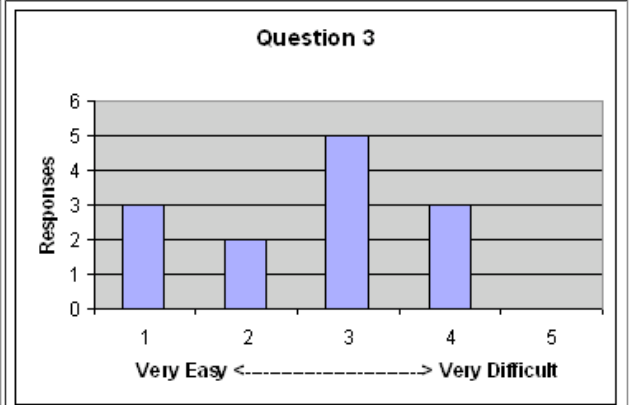
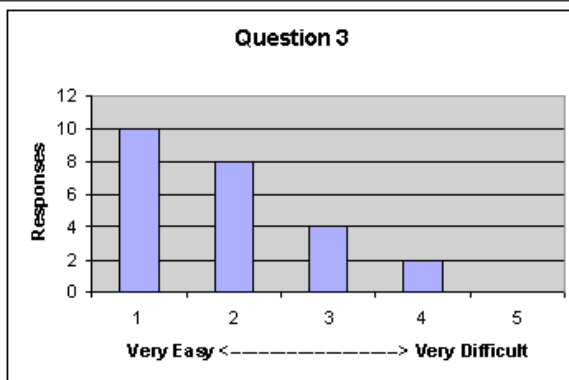
5.2.1 Installation/Configuration

Question	2003 Response	2006 Response
1. Installing the Eclipse IDE sensor was: (Very Easy) 1 2 3 4 5 (Very Difficult)		

2. Installing the Ant sensors (JUnit, SCLC, Emma, etc.) were:
 (Very Easy) 1 2 3 4 5 (Very Difficult)



3. Configuring Hackstat to track our team's work (i.e. defining the project, and configuring the workspace roots) was:
 (Very Easy) 1 2 3 4 5 (Very Difficult)

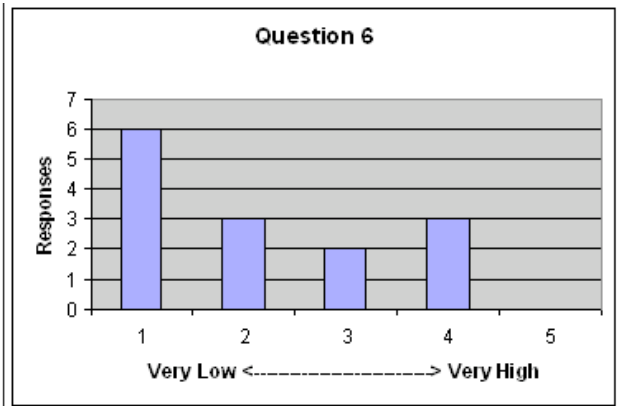
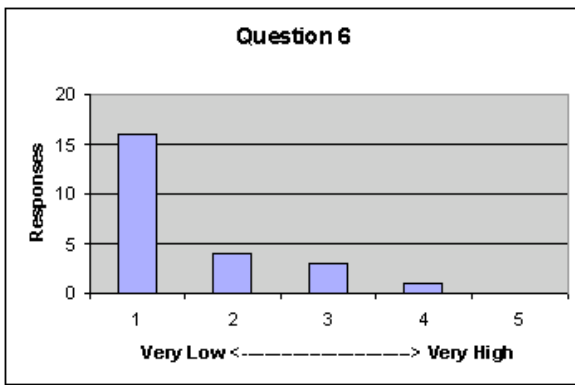


The empirical data indicates little change in student opinion regarding Eclipse sensor installation. On the other hand, the data appears to indicate that Ant sensor installation is easier than it was in 2003. Finally, it appears to indicate that server-side configuration has actually gotten harder since 2003.

5.2.2 Overhead of Use

Question	2003 Response	2006 Response																								
5. The amount of overhead required to collect Hackstat data was: (Very Low) 1 2 3 4 5 (Very High)	<table border="1"> <caption>Question 5 (2003) Responses</caption> <thead> <tr> <th>Overhead Rating</th> <th>Number of Responses</th> </tr> </thead> <tbody> <tr> <td>1 (Very Low)</td> <td>17</td> </tr> <tr> <td>2</td> <td>3</td> </tr> <tr> <td>3</td> <td>3</td> </tr> <tr> <td>4</td> <td>0</td> </tr> <tr> <td>5 (Very High)</td> <td>1</td> </tr> </tbody> </table>	Overhead Rating	Number of Responses	1 (Very Low)	17	2	3	3	3	4	0	5 (Very High)	1	<table border="1"> <caption>Question 5 (2006) Responses</caption> <thead> <tr> <th>Overhead Rating</th> <th>Number of Responses</th> </tr> </thead> <tbody> <tr> <td>1 (Very Low)</td> <td>7</td> </tr> <tr> <td>2</td> <td>3</td> </tr> <tr> <td>3</td> <td>3</td> </tr> <tr> <td>4</td> <td>1</td> </tr> <tr> <td>5 (Very High)</td> <td>0</td> </tr> </tbody> </table>	Overhead Rating	Number of Responses	1 (Very Low)	7	2	3	3	3	4	1	5 (Very High)	0
Overhead Rating	Number of Responses																									
1 (Very Low)	17																									
2	3																									
3	3																									
4	0																									
5 (Very High)	1																									
Overhead Rating	Number of Responses																									
1 (Very Low)	7																									
2	3																									
3	3																									
4	1																									
5 (Very High)	0																									

6. The amount of overhead required to run Hackstat analyses was:
 (Very Low) 1 2 3 4 5 (Very High)



The empirical data seems to indicate that, while still low, the overhead of use of Hackstat has increased to some extent since 2003.

5.2.3 Usability/Utility

The data from this section is difficult to compare, since the 2003 system used different analyses than in 2006. For illustrative purposes, we will show one analysis from 2003 and one from 2006:

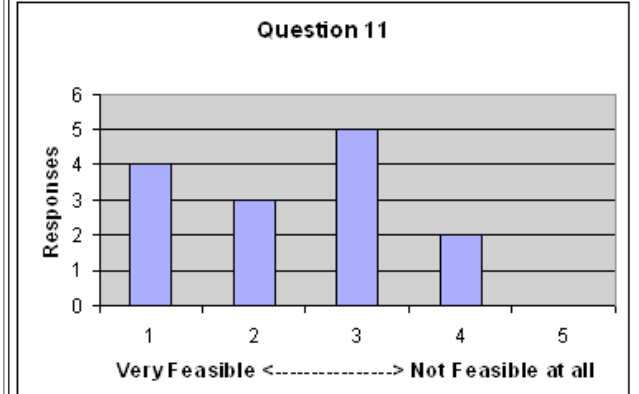
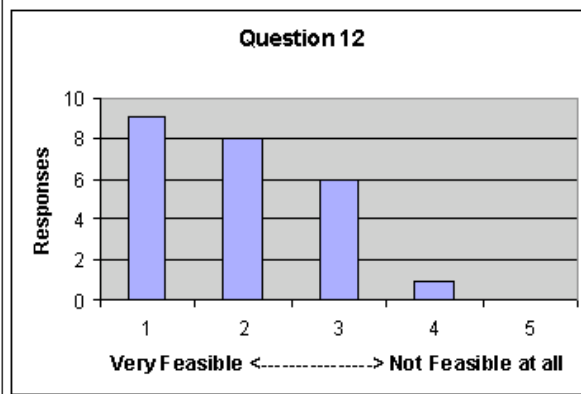
Questions	2003 (Course Project analysis)	2006 Response (TraffoMeter Report Analysis)																								
2003: The Course Project Analysis (showing a comparison of project data between all of the SiteWatch projects) was: (Highly Usable) 1 2 3 4 5 (Not Usable At All) (Highly Useful) 1 2 3 4 5 (Not Useful At All)	<table border="1"> <caption>Question 10A Data</caption> <thead> <tr> <th>Response</th> <th>Count</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>8</td> </tr> <tr> <td>2</td> <td>10</td> </tr> <tr> <td>3</td> <td>6</td> </tr> <tr> <td>4</td> <td>0</td> </tr> <tr> <td>5</td> <td>0</td> </tr> </tbody> </table>	Response	Count	1	8	2	10	3	6	4	0	5	0	<table border="1"> <caption>Question 8A Data</caption> <thead> <tr> <th>Response</th> <th>Count</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>5</td> </tr> <tr> <td>2</td> <td>5</td> </tr> <tr> <td>3</td> <td>3</td> </tr> <tr> <td>4</td> <td>1</td> </tr> <tr> <td>5</td> <td>0</td> </tr> </tbody> </table>	Response	Count	1	5	2	5	3	3	4	1	5	0
Response	Count																									
1	8																									
2	10																									
3	6																									
4	0																									
5	0																									
Response	Count																									
1	5																									
2	5																									
3	3																									
4	1																									
5	0																									
2006: The TraffoMeterReport telemetry analysis (showing trends in project/process data) was: (Highly Usable) 1 2 3 4 5 (Not Usable At All) (Highly Useful) 1 2 3 4 5 (Not Useful At All)	<table border="1"> <caption>Question 10B Data</caption> <thead> <tr> <th>Response</th> <th>Count</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>5</td> </tr> <tr> <td>2</td> <td>7</td> </tr> <tr> <td>3</td> <td>5</td> </tr> <tr> <td>4</td> <td>4</td> </tr> <tr> <td>5</td> <td>3</td> </tr> </tbody> </table>	Response	Count	1	5	2	7	3	5	4	4	5	3	<table border="1"> <caption>Question 8B Data</caption> <thead> <tr> <th>Response</th> <th>Count</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>5</td> </tr> <tr> <td>2</td> <td>4</td> </tr> <tr> <td>3</td> <td>4</td> </tr> <tr> <td>4</td> <td>1</td> </tr> <tr> <td>5</td> <td>0</td> </tr> </tbody> </table>	Response	Count	1	5	2	4	3	4	4	1	5	0
Response	Count																									
1	5																									
2	7																									
3	5																									
4	4																									
5	3																									
Response	Count																									
1	5																									
2	4																									
3	4																									
4	1																									
5	0																									

At least when comparing these two analyses to each other, the data seems to indicate that usability might have decreased slightly, while utility (usefulness) might have increased slightly.

5.2.4 Future Use

Question	2003 Response	2006 Response
----------	---------------	---------------

12. If I was a professional software developer, using Hackstat at my job would be:
(Very Feasible)
1 2 3 4 5 (Not Feasible at All)



The empirical data seems to indicate that student feelings toward "professional feasibility" seem to have dropped a bit since 2003.

5.3 Interpretation

The comparison of the data from 2003 to 2006 is extremely instructive, in light of the changes that occurred in the system since 2003. I was quite excited to introduce Hackstat to my software engineering classes in 2006, because I felt that a much broader range of data could now be collected, and that the availability of HackyInstaller would enable the installation of sensors for this data with lower overhead. I was also excited to transition students away from a classroom-specific set of analyses and into Telemetry-based analyses, which would correspond more directly to the kind of analyses that they would be able to use in a professional setting.

What the data reveals is that the increase in capability of the system since 2003 has come at the cost of a decrease in usability. In 2003, the customized nature of the analyses, and the relatively small number of different measures and tools appeared to ease the understanding and interpretation of the data by the students. In 2006, the increase in the range of data and the more general nature of the analyses caused some students to feel frustrated by the lack of guidance in data interpretation. In 2003, no sensor data was collected from the configuration management tool, and thus no students were required to set up the usermaps file. In 2006, they were required to manage this sensor installation by themselves, which added new overhead to the installation and daily collection activities.

In 2003, the empirical data and the associated comments indicated that a major obstacle to the adoption and use of Hackstat was the lack of support for client-side sensor installation. In 2006, the data indicates that the availability of HackyInstaller appears to have addressed that obstacle. However, the data from 2006 indicates that the current most significant obstacle to the adoption and use of Hackstat is the server-side user interface, which introduces difficulties with respect to configuration, display, and interpretation of the collected data.

The next section presents some preliminary proposals for user interface improvements intended to address the issues revealed by this study.

6.0 Future Directions

As the previous section indicates, the current server-side user interface appears to create significant barriers to the usability and utility of Hackstat. Here are some preliminary proposals for a "next generation" Hackstat interface that might address some of the problems that have been identified.

6.1 Ajax-based menus and parameters

Several students reported problems with the form-based interface to Hackstat commands, including:

- Not understanding that after selecting a menu item, one must press "Submit" to store the selection.
- Not understanding what parameters were required for a given chart or report.
- Selecting incorrect dates for a Project-based command.

Many of these kinds of problems could be avoided by moving to an "Ajax" based interface. In this style of interface, selections could be communicated to the server immediately, avoiding the need to explicitly "store" them. Once a chart or report is selected, this information can be communicated to the server immediately and used to generate context-specific input for the appropriate parameters. Finally, once a project is selected, this information can be communicated to the server and used to generate date input selectors that are specific to the range of dates appropriate for the Project.

6.2 Workflow-based organization of the user interface

A significant usability problem with the current interface is the organization of the Hackystat commands into the "Analyses", "Preferences", "Alerts", "Extras", and "Help" pages, requiring users to move back and forth among several different pages, invoking different commands at different times to accomplish various tasks. An alternative approach would be based upon "workflow", and organize the user interface according to a particular task, such as:

- Initial sensor setup and validation
- Initial project setup
- Ongoing project monitoring
- Project reporting

Depending upon the workflow task indicated by the user, the interface could dynamically generate the set of commands required to deal with that task.

6.3 Real-time data display for ongoing project monitoring

After initial setup of sensors and project definitions, the next type of workflow for a user is ongoing project monitoring. Project monitoring involves two principle activities: (1) performing telemetry-based (or other) project analyses to monitor project productivity and quality, and (2) ensuring that sensor data is being generated, sent to the server, and associated with the project correctly.

An improved user interface to accomplish this workflow task might consist of a Project-specific "Home Page" which would group together a set of charts showing the telemetry data appropriate to that project, along with "sensor validation" information. This could show, for example, whether or not sensor data of a given type was sent by a given project member for the past several days. In addition, the page could display a scrolling window updated in real-time with information about sensor data received by the current user (and/or other members of this project) at the server.

Currently, project monitoring activities requires manually invoking various commands located on different pages of the server. The current interface makes it not only difficult to understand how to accomplish project monitoring tasks, but quite tedious even when understanding has been obtained. A much better solution might be the proposed user interface which provides "live", "real time" information about sensor data transmission to the server along with "current" charts, all grouped together on a single Home Page.

6.4 Annotations

Correct interpretation of the empirical data often requires supplemental information about events occurring during project development. The FileMetrics telemetry chart presented in the Powerpoint presentation shown in Section 5.1 illustrates one way to effectively use annotations. The TimeLine project <<http://simile.mit.edu/timeline/>> shows another way to incorporate annotations.

One way to facilitate the interpretation of the data would be to enable group members to attach annotations to the data that would appear in reports.

6.5 Simplified data exploration facilities

In 2003, providing new forms of data analysis in Hackystat was extremely complex and time-consuming, since it involved designing and implementing a new user interface command. This was clearly beyond the realm of possibility for students in the course. In 2006, we made significant improvements to support for data exploration through the creation of the Software Project Telemetry language, which enables users to define new Charts and Reports without extending the user interface. However, the telemetry language was not discussed in the course, and with only one or two exceptions, students relied upon the "shrinkwrapped" data analyses provided by the TraffoMeter Telemetry Report.

An intriguing question is whether there are simpler alternatives to the Software Project Telemetry language for supporting data exploration in Hackystat. For example, the Swivel web site <<http://www.swivel.com/>> provides a public repository for data along with high level primitives for simple juxtaposition of data series. Could similar user interface techniques be adapted to Hackystat in order to enable students to explore their data sets? If so, would students actually take advantage of the opportunity? Finally, would this create interpretation issues involving the fact that [correlation does not imply causation](#), or in the worst case leading some students to believe that [correlation is causation](#)?

7.0 Want to participate further?

If you have any other thoughts or inspirations upon reading this technical report, please don't hesitate to send me email at

johnson@hawaii.edu.