Most Active File Measurement Validation in Hackystat

Hongbing Kou, Xiangli Xu
Collaborative Software Development Laboratory
Department of Information and Computer Sciences
University of Hawai'i
Honolulu HI 96822
hongbing@hawaii.edu

Abstract

Hackystat, an automated metric collection and analysis tool, adopts the "Most Active File" measurement in five-minute time chunks to represent the developers' effort. This measurement is validated internally in this report. The results show that big time chunk sizes are highly linear regressive with the standard time chunk size (1 minute). The percentage of missed effort to total effort is very low with five minutes chunk size. And the relative ranking with respect to the effort of the active files is only slightly changed.

Keyword Hackystat, Most Active File Measurement, Linear Regression

1. Introduction

Hackystat is a sensor-based automated metrics collection tool that is developed at the Collaborative Software Development Laboratory at the University of Hawaii. The sensor, also called plug-in is installed in the IDE to collect the developers' activities. At present the in use sensors are the activity sensors for JBuilder and Emacs. The sensors record the developer's activities by checking the active buffer change after a small period and sends out the new buffer size to the Hackystat server if there is any change. The buffer change can represent the developer's editing effort or work effort because files are the contribution of the developers to the project. The sensor is invoked once every 30 seconds in Hackystat. [1] In addition to the buffer changes the Chidamber-Kemerer object metrics to the active java files are also collected by the sensor at the same time.

The metric data is sent from Hackystat sensors to Hackystat server via SOAP [1] and it is stored in XML files in the Hackystat server repository. With the activity data we can do the analysis and provide useful information for the developers to improve the software development process. However, it is not a simple issue because of the volume of the activity data. Using 30 seconds interval and assuming 6 work hours in a day, the number of items will be 6 * 60 / 0.5 = 720. If the Hackystat sensor is configured to be invoked every 5 seconds or shorter we will have tons of data to be analyzed. It's one issue regarding to the Hackystat analysis. On another hand Hackystat can only detect the developers' editing work effort. However, in the software development process not only the editing work but also the review work like code review, document reference are included So the Hackystat sensor may not function well because it simply thinks only the editing effort is the actual work. These two issues motivated the Hackystat designers and developers to use a conceptual measurement called "Most Active File" measurement. Basically it uses one mostly edited file in a time chunk -- a relatively big period such as 5 minutes, as the representation of the work effort in that period. As long as there is one most edited file the developer will own this time period and that file is called most active file in Hackystat. In current Hackystat system this time chunk size is 5 minutes. The Hackystat analysis modules abstract the developer's daily activity log with 5 minutes chunk size and then conduct the analyses on the abstracted activity log.

However, we don't know whether it is valid to use "Most Active File" measurement and how good the selected time chunk size is. We designed some experiments to validate the "most active file" measurement as a course project for class ICS691[1].

The first experiment is to validate the selected time chunk size in Hackystat and study other possible sizes. In Hackystat we can access to the fine-grained data since the sensor interval (30 second) is very small. However, the "Most Active File Measurement" maps the data to coarse-grained data with 5 minutes time chunk size in Hackystat We believe the bigger the chunk size is the less accurate the data will be. Since 1-minute is small enough we use 1-minute chunk size as the standard to evaluate the effects of big chunk sizes 3, 4, 5, 8, 10 minutes. This

[1] ICS691 represents the class ICS691-1 in fall semester of 2002 at the University of Hawaii in this paper.

study can help us select the good chunk size for Hackystat and have reliable analysis results with the good time chunk size or sizes.

The second experiment is used to evaluate the accuracy of "Most Active File Measurement" with 5 minutes chunk size in Hackystat. The 1 minute time chunk size is used as the standard to evaluate how accurate it is. Since only one mostly edited file is chosen as the most active file in 5 minutes chunk the effort to the less active files might be lost with this measurement. In this experiment the missed effort percentage is calculated to study the accuracy. Also from the statistics point of view the activity density can vary from one developer to another developer and it can also be changed with time going on to one developer. So we designed a standard distribution activity generator program to generate different activity densities to study the effects of density.

The last experiment is to study the relative ranking change of most active files with respect to effort in Hackystat. We ordered the most active files in a day by the effort from the most active to the least active. The ranking of the most active files is an important attribute in a day because it can be used to study how the developers spend their time or other related issues. If the ranking is changed drastically we should give up using this measurement to conduct this kind of studies because the measurement may introduce bias to the experiment data.

In our study we conducted our experiments on one author's Hackystat data and used the results to direct our study direction. In the latter phase of the project we acquired the consents of 10 students of the ICS691 and conducted the experiments on their Hackystat data.

2. Methods

A new branch is created from Hackystat to conduct all the experiments in this study and we added some new features to Hackystat to facilitate our study.

2.1 Time chunk size selection

2.1.1 Time chunk size configuration

In Hackystat the time chunk size is 5 minutes and it's a constant value. We created a configuration JSP page to configure the chunk size. The configurable time chunk sizes are 1, 2, 3, 4, 5, 8, 10, 15, and 20 minute(s).

2.1.2 "Active File Time" analysis

The "Active File Time" analysis was designed to count the effort of all kinds of chunk sizes in a day or a period. Given a day or a period the analysis will calculate the efforts in that period with different time chunk sizes and the results are used to do linear regression analysis.

2.2.3 Real Hackystat data

The real Hackystat data was used to conduct this experiment. Since our project is a branch of Hackystat we can use the Hackystat data directly in our experiment. We first conducted the experiment on one author's Hackystat data and then on the ICS691 students' Hackystat data.

2.1.4 Linear regression analysis

We did the linear regression analysis between effort of big time chunk size and the effort of standard time chunk size (1 minute in the study). The linear regression analyses between 3 minutes and 1 minute, between 5 minutes and 1 minute, between 8 minutes and 1 minute, and between 10 minutes and 1 minute were conducted on one author's Hackystat data. In the third milestone of the project we did the experiment on 10 ICS691 students' Hackystat data and studied the linear regression model of 5 minutes time chunk size.

2.2 Missed effort analysis

Some efforts could be lost in Hackystat with the "Most Active File" measurement. Any measurement schema has its limitation on accuracy and this measurement has the same problem. For instance, in a 5 minutes chunk, the less active files in that chunk will be lost because the measurement can only reach 5 minutes' accuracy. In this experiment we will study how much effort will be missed with 5 minutes time chunk size.

2.2.1 Algorithm to calculate the missed effort and the missed effort percentage

In Hackystat a most active file is chosen as the representative of a chunk. So the less active file can be lost if there is any. However, the less active files can get their credits back if these files are mostly edited files in another chunk. The effort to some files will be lost with the "Most Active File" measurement. An algorithm is designed to count the effort to these missed files with 5 minutes time chunk size.

Firstly we calculated the most active file set in a day of 1 minute chunk size. This set is used as the standard most active file set. In order to calculate the missed effort we calculated the most active file set of 5 minutes time chunk size and compared this set with the standard set. The effort to the active files that exist in the standard set not in the new set were lost and can not be detected with the "Most Active File" measurement. So the sum of the effort of the lost files is the effort we missed in a day. Dividing this value by the total effort in a day we will get the missed effort percentage. The formula (1) is used to count the total effort miss and formula (2) is used to calculate the effort miss percentage.

$$Missed\ \ effort\ \ ?\ \ ?\ \ effort\ \ (\{S\}\ ?\ \{U\}) \quad (1)$$

{S} is the set of the most active files of 1 minute time chunk in a day
{U} is the set of the most active files of 5 minutes time chunk in a day

$$Missed\ effort\ percentage\ ?\ Missed\ effort\ /\ ?\ \ effort\{S\} \quad (2)$$

2.2.2 Missed effort percentage

A new Hackystat analysis called "Measurement Validation" was created to calculate the effort miss percentage. It calculates over all Hackystat time span of one author and presents the information including missed effort, total effort and missed effort percentage in three layers' structure. The first layer is the accounting information over the entire Hackystat space, the second layer is the accounting information over a month and the last layer is the detail analysis result in a day. In the last layer files and efforts are listed in the descending order for the researcher to observe the results manually.

2.2.3 Missed effort percentage on real Hackystat data

This study is conducted on 10 ICS691 students' Hackystat data over their Hackystat time span and the missed effort percentages were calculated to study the difference of missed percentage to different developers.

2.2.4 Effort missing percentage on simulated data

Since we have only 10 students' Hackystat data for study we are not for sure whether all the conditions are covered so a program was created to generate standard derivation activities of different densities to simulate all possible conditions. A standard derivation activities generator was created to generate different density activities to study the effects of activity density to missed effort percentage. The activities densities are 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%.

2.3 Relative ranking change with respect to effort

As mentioned in the introduction section the relative ranking of the most active file in a day plays an important role on some studies. In this experiment we studied how likely the relative ranking of the most active file of 5 minutes time chunk size in a day is going to be changed. Since the relative ranking of most active files with respect to effort will be changed with the "Most Active File" measurement because of the accuracy the equivalent study is to study the average effort difference caused by the changes of the relative ranking.

2.3.1 Algorithm to calculate the average effort difference

Table 1 is an example of the list of most active files of 1 minute time chunk size and 5 minutes time chunk size in a day. The second column is the most active file list of 1 minute time chunk size ordered by the effort from the biggest to the least. The second column is the most active file list of 5 minutes time chunk size ordered by the effort from the biggest to the least. The number in the bracket after the file name is the total effort to that file in that day.

Table 1 List of most active files in a day in descending order of effort

	1 minute (effort)	5 minutes (effort)
1	Foo.java (12)	Bar.java (30)
2	Bar.java (9)	Foo.java (25)
3	Foo.jsp (8)	Foo.jsp (10)
4	Bar.jsp (4)	Foo.html (10)
5	Foo.html (3)	Bar.html (5)
6	Bar.html (2)	Bar.jsp (5)

We can see from the above table that the ranking of most active files are inconsistent. The formula to calculate the difference of effort is as following

|12(Foo.java)-9(Bar.java)| + |9(Foo.java)-12(Bar.java)| + |8(Foo.java) – 8(Foo.java)| + |4(Bar.jsp) – 3(Foo.html) | + |3(Foo.html) – 2(Bar.html)| + |2(Bar.html) – 4(Bar.jsp)|
= 3 + 3 + 0 + 1 + 1 + 2
= 10 minutes

Note: All the numbers are from the second column because the effort of 1 minute time chunk size is used as the standard for the study.

The total effort difference is 10 minutes and the average effort difference is 10 / 6 = 1.67 minutes because there are 6 most active files in this day.

The general formula to calculate the average effort difference is as following.

$$AED\ \ ?\ \ ?\ \ \sum_{i?1}^{M} \left| E_i\ \ ?\ \ E_{(index\ (U_i))} \right|\ \ /\ M \quad (3)$$

AED standards for Average Effort Difference

$\boldsymbol{U_i}$ is the *i*th item of the most active file list of 5 minutes time chunk size in the descending order by effort

$\boldsymbol{E_i}$ is the effort of the *i*th most active file of the standard effort list of 1-minute time chunk size

Index () takes a file name and looks up the most active file list of 1 minute time chunk size for the index of that file

M is the size of the most active files list of 5 minutes time chunk size

2.3.2 Average effort difference on real data

We conducted the experiment on 10 ICS691 students' Hackystat data and calculate the average effort difference over their Hackystat time span.

3. Results

3.1 Time chunk size selection study

This study was conducted on one author's Hackystat data from April 10 to December 2, 2002. Figure 1 gives the linear regression mode between 5 minutes chunk size and the standard chunk size.

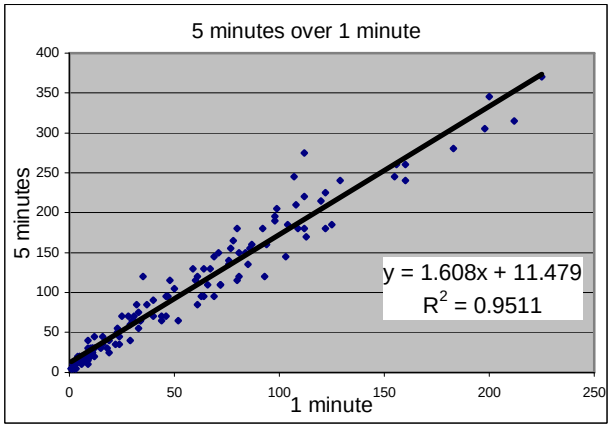

Figure 1 Linear Regression model of 5 minutes to 1 minute of one author's Hackystat data

We can see that the effort of 5 minutes' chunk size and the effort of 1 minute's chunk size are linearly regressive. The R-square value is 0.9511 so we can predicate the standard effort accurately by the effort of 5 minutes chunk size. We also conduct the linear regression analysis between time chunk size 3 minutes and standard time chunk size, between 8 minute time chunk size and standard time chunk size and between 10 minutes time chunk size and standard time chunk size. The following table gives the linear regressive models of different time chunk sizes to the same person's Hackystat data.

Table 2 Linear regression models of time chunk sizes 3 minutes, 5 minutes, 8 minutes and 10 minutes of one author' Hackystat data

Chunk size	Linear approximation	R-Square
3 minutes	Y = 1.3928X + 5.9342	0.9771
5 minutes	Y = 1.608X + 11.479	0.9511
8 minutes	Y = 1.8553X + 20.574	0.9174
10 minutes	Y = 1.954X + 24.623	0.8965

From this table we can see that the linear regression model fits well to all time chunk sizes. The R-Square is big enough so that we can choose any one of them as the chunk size in Hackystat.

In Hackystat the chunk size of the analysis is 5 minutes. We did the linear regression analysis of time chunk size 5 minutes on 10 ICS691 students' Hackystat data. The following table lists the r-square values of the linear regression models to everybody's Hackystat data. Nearly all of them are bigger than 0.9 and there is only one r-square value is less than 0.9.

Table 3 R-square values of linear regression model of time chunk size 5 minutes to all students' Hackystat data.

No.	R-Square
1	0.9511
2	0.95
3	0.9678
4	0.8312
5	0.9496
6	0.9341
7	0.9705
8	0.9527
9	0.9876
10	0.9581

From the above table we can conclude that the linear regression model can be applied to all students' Hackystat data with time chunk size 5 minutes. So the 5 minutes time chunk size is appropriate in Hackystat to all users by our study.

3.2 Missed effort analysis

3.2.1 Missed effort on real Hackystat data

Table 4 Missed effort percentage to 10 ICS691 students' Hackystat data of time chunk size 5 minutes

No.	Total effort	Missed percentage%
1	7070	5
2	4703	4.1
3	2259	2.5
4	72	9.7
5	8953	4.5
6	13001	4.3
7	5858	0.9
8	377	1.3
9	226	2.2
10	2443	1.7

The above table is the missed effort percentage of 5 minutes time chunk size to 10 ICS691 students' Hackystat data by applying the algorithm described in section 2.2.1. The missed percentages are less than 5% to most students except for one student who has only little Hackystat work effort. (It is less than 72 minutes) On the average the missed effort percentage is around 5% but to some students it is only 2% or less. Clearly the missed effort is trivial with respect to the effort students spent on the projects with 5 minutes time chunk size.

3.2.2 Missed effort percentage on different effort densities

The standard derivation activity generator generated the activities according to experiment plan of table 5. The first column is the month with activities and the second column is the activity density in that month.

Table 5 Experiment plan between month and the density

Month	Density
1/2002	10%
2/2002	20%
3/2002	30%
4/2002	40%
5/2002	50%
6/2002	60%
7/2002	70%
8/2002	80%
9/2002	90%
10/2002	100%

100% percent density means that the developer works every minute and there is no break. 10% density means that the developer spends 10% time on the project. The results we got are shown on figure 2. The effort miss percentage is 1.3% to 10% activity density and 12.5% to 100% activity density. The average effort miss percentage is 6.3%.

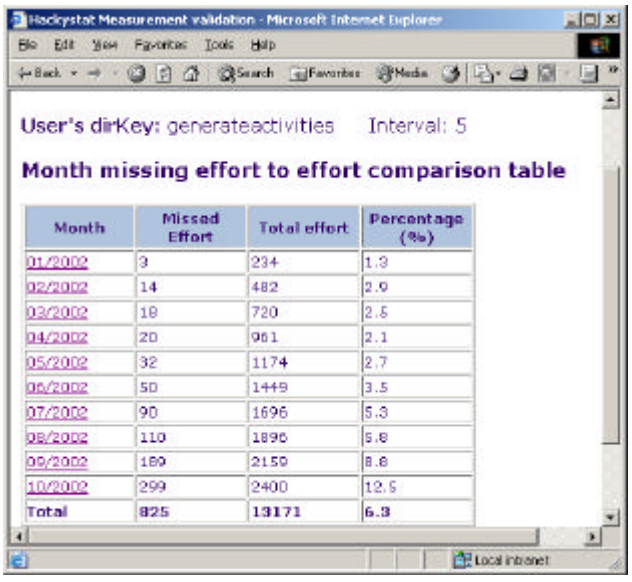

Figure 2 Missed effort percentage on different activity densities

3.2.3 Relative ranking change with respect to effort

With the algorithm described on section 2.2.3 we calculated the average effort differences caused by the relative ranking change of the most active files with respect to effort. Table 6 lists the average effort difference of 5 minutes time chunk size.

Table 6 Average effort differences of 5 minutes time chunk size of one author's Hackystat data

Month	Average out of order effort difference (min)
4/02	1.4
5/02	1.3
6/02	0.9
8/02	0.6
9/02	0.6
10/02	0.8
Average	0.93 << 5

The value in the table is the effort difference with respect to the relative ranking changes. The average effort different is only 0.93 minute of 5 minutes time chunk size. It means that the order of most active files with respect to effort will not be changed if the effort difference between two files is bigger than 0.93 minute. Because 0.93 minute is a small period compared with the time spent on the files the result is acceptable with 5 minutes time chunk size.

The average effort differences to all 10 students' Hackystat data are listed in table 7.

Table 7 Average effort differences of 5 minutes' chunk size

No.	Average Effort difference
1	0.83
2	1.18
3	0.93
4	0.3
5	1.25
6	0.99
7	1.1
8	0.2
9	0
10	0.91

All of the average effort difference values are much smaller than the 5 minutes time chunk size. From the results we can tell the relative ranking with respect to the effort will not be changed if the work effort to two files are not too close. In another word the relative ranking

with respect to effort will be only slightly changed if we choose 5 minutes as the time chunk size.

4.Conclusion and Discussion

The effort of big chunk size is linearly related with the standard effort (of 1 minute time chunk size). From the effort of 5 minutes' chunk size we can predicate the efforts accurately with linear regression model according to our results. From the time chunk selection analysis we can conclude that we can use any size from 3 minutes to 10 minutes as the time chunk size in Hackystat and the 5 minutes chunk size is a good value according to our study. With our results of the study of linear regression model of 5 minutes time chunk size on 10 ICS691 students' Hackystat data the "Most Active File" measurement can be applied to everybody.

The missed effort percentage in Hackystat is trivial with 5 minutes time chunk size. The missed percentage is around 5% to most students and it's much less than 5% to some students. And our analyses on all kinds of activity densities also suggest that 5 minutes chunk size is a good value because the average missed effort percentage is only 6.3%.

The relative ranking change of most active files with respect to effort is only slightly changed by our results. The average effort difference is only 0.93 minute to 5 minutes time chunk size.

So we can conclude that the "Most Active File" measurement is applicable in Hackystat and the data quality is still very high with 5 minutes chunk size.

However, as a sensor-based automatic metrics collection tool, Hackystat can only collect the data that represents the programmer's editing activities. It cannot represent the programmer's other activities in the software development process, for example, reading document, thinking about the structure of the program, discussing with other colleagues, reviewing the code etc. It's the weakness of Hackystat compared with PSP and Leap Toolkit [3], which are relatively flexible on choosing what kind of data to be collected.

In the project proposal discussion session Philip M. Johnson, the instructor of class ICS691, and some other classmates opposed this study because they thought we could not validate the Hackystat measurement by Hackystat itself. They said you could not say the effort with "Most Active File" measurement is validate by this study. They proposed the external validation to conduct the research, which views the "most active file" measurement from another point of view. Since other kinds of work like documenting are interleaved with the editing work it's possible that the "Most Active File Measurement" can include the work effort except for the editing work. As we can see from linear regression model the effort we got with 5 minutes time chunk size is much bigger than the effort with 1 minute time chunk size. If the "Most Active File" measurement can measure the developer's actual work effort including programming, referring to other parts of the code, reading the documentation, reviewing the code etc it will be a great feature of this measurement. The only way to verify this is to do the external validation.

In our study we concluded that we can choose any time chunk size from 3 minutes to 10 minute. Any value can help us to have reliable analysis results. With external validation we can find which size can summarize the developer' work effort best and to study whether there is one universal time chunk size applicable to all developers. So the external validation could be the future research direction. The Ginger2 system[4] designed by Koji Torri etc can be used to conduct this research. In the study we conducted the experiments with time chunk sizes from 3 minutes to 10 minutes but it is not the upper bound. So another research can be conducted to study the upper bound of the chunk size in the future.

5. Acknowledgments

This project used the real Hackystat data provided by some students in ICS691 of fall semester of 2002 at University of Hawaii. Dr. Philip M. Johnson spent a lots time on discussing with us about this project and gave us a lot helpful suggestions on experiment design. The classmates of class ICS691 also gave a lot good suggestions and comments on the research and technique report. Also thanks for CSDL for providing the CVS repository and Hackystat data for us to conduct this research.

6. References

[1] Philip M. Johnson, etc. "Beyond the Personal Software Process: Metrics collection and analysis for the differently disciplined", http://csdl.ics.hawaii.edu/techreports/02-07/02-07.pdf July, 2002

[2] Watts S. Humphrey *Introduction to the Personal Software Process* Addison-Wesley Long Inc, Berkeley, California, 1997

[3] Carleton A. Moore "Investigating Individual Software Development: An Evaluation of the Leap Toolkit", Ph. D thesis, August 2002

[4] Koji Torri etc "Ginger2: An Environment for Computer-Aided Empirical Software Engineering", IEEE Transaction on Software Engineering, Vol. 25, No. 4, July/August 1999