

Quality Oversight Measured By Complexity

Hui-Ping Chen, Eli Lilly and Company, Indianapolis, IN
 Mark Matthews, Eli Lilly and Company, Indianapolis, IN

ABSTRACT

As SAS programmers, we have the responsibility of coordinating our own projects and account for our deliverables to be on-time based on the availability of resources and project timelines. We would be interested in knowing how many hours it will take to generate assigned reports or how many days it will take to deliver based on the individuals that are assigned to the project. In addition, it may be important to minimize the future rework and measure our quality based on what we learn from the previous project work. This paper provides an example of how you can go about estimating the number of hours (resources) that are needed to complete the programming and validation (QC or result checking, for example) based on the complexity of reports.

HOW TO DEFINE COMPLEXITY OF REPORTS:

The definition of report complexity can be based on the combination of the following: 1) what statistical methods are used to generate the report, 2) the dichotomy of whether or not the report layout is simple and 3) if this report is generated by a single program or a macro. The time unit associated with the complexity of the report in this example is Day. This time unit can be established based on prior programming experiences in a known environment.

Table of definition of complexity and associated time units (days)

Complexity	Definition	Time Unit
L	Duplicate from or modify an existing program	0.5
M1	Create a new program with simple statistics methods and generate a simple output	1.0
M2	Create a new program with simple statistics methods and generate a complicated output	1.5
M3	Create a new macro with simple statistics methods and generate several outputs	2.0
H1	Create a new program with complicated statistics methods and generate a simple output	1.5
H2	Create a new program with complicated statistics methods and generate a complicated output	2.0
H3	Create a new macro with complicated statistics methods and generate several outputs	2.5

The definition of complexity is shown as the combination of characters and numbers. The characters (L, M, and H) pertain to the complexity of statistical methods used in the programs, which stand for Low, Median, and High. The numbers (1, 2, and 3) reflect the complexity of generating outputs (Tables, Graphs or Listings). The lower the number, the easier it is to generate the layout of the report. Thus we have established 7 tiers of complexity with a time unit associated with it.

Level L suggests that you generate reports by simply adopting or modifying an existing program to meet the requirements. Therefore, generalizing reports with level L is equivalent to taking approximately 0.5 day as defined in the table above. Let's say that it may take you substantial time to modify an existing program to meet the new requirements, then in that case, you may want to consider to rate the complexity as "M" instead of "L". Or, if you are not familiar to the statistical methods described in the requirement and you need time to learn about it, you could rate "H" for this report. In this "H" case, if multiple reports are generated from a macro, you can rate the first report as H3 and L for the rest of reports.

The complexity of a report is different from person to person and it is not necessary that there is a relationship between the programmer and quality controller's complexity category. For example, the quality controller may not need to focus on how to display the reports, thus, a complexity rating might be M1 to the quality controller but M2 to programmer. Another example is that if the quality controller is more experienced, he/she may not need to research the statistical methods or seek other programming references before implementing his/her task. In other words, the complexity association between programmer and quality controller is not expected nor a factor.

HOW TO CATAGORIZE CAUSES OF REWORK

Table of rework reason

Rework Reason Environment	(1) Coding Error	(2) Requirement Updated	(3) Typo or Format Error	(4) Data Issue
TEST	T1	T2	T3	T4
PRODUCTION	P1	P2	P3	P4

The table above contains suggested categories, or reasons, that cause rework. For example, if 'requirement updated' contributes to a few hours of reworking your program, then we should capture both the reason for rework and the time spent on that rework. If a coding error only takes the programmer a couple of minutes to fix it, we can collect the reason but not necessarily capture the time associated with that rework. It is up to the individuals to understand the threshold of time relative to the task's complexity category. The reasons that cause rework provides the opportunity for a deeper look into reducing the rework load for future projects and perhaps look for any correlations between rework and complexity. It will provide us another variable in our data base that can be used to further analyze why time is being spent in those categories and perhaps target process improvement and yield a higher first time right.

WHAT INFORMATION SHOULD BE COLLECTED

The following information collected should provide all of the information we need in order to build a data base that provides information about our capacity needs and measure our performance.

- 1) Program Name
- 2) Report Name
- 3) (CP) Complexity Rating by Programmer
- 4) (PH) Programming Hours
- 5) (PR) Programming Rework Reason
- 6) (PRH) Programming Rework Hours
- 7) (CQC) Complexity Rating by Quality Controller
- 8) (CQH) Quality Control Hours
- 9) (CQR) Quality Control Rework Reason
- 10) (CQRH) Quality Control Rework Hours

Based on the results of compiling the complexity of each report, we may be able to make business decisions such as better understanding the capacity needs and execution planning of the project so we can deliver the reports to our customers.

WHEN SHOULD THE INFORMATION BE CAPTURED

The project coordinator, the one who owns the project tracking data base, collects or assigns the perceived complexity from the involved individuals for each report prior to the project start with respect to their role. This will give a predicted amount of time it will require to execute the project. Then throughout the project, the programmer and quality controller will input the working hours, rework reasons, and rework hours after they implement each task or report.

Example Project AA: Project Tracking Data Base

Program	Report	CP	PH	PR	PRH	CQC	QCH	QCR	QCRH
AE1	AE11	L	3			M1	5		
AE1	AE12	L				L			
AE4	AE41	M1	8	T1	1	M1	8	T1	1.5
DEM1	DEM11	L	1			L	2		
LAB3	LAB31	M3	16	T4		M1	6	T4	2
LAB3	LAB32	L				L			
LAB3	LAB33	L				L			
LIST1	LIST11	L	2			L	1		
LIST2	LIST21	M2	8			L	3		
GRAPH1	GRAPH11	H2	14	P2	4.5	M1	5	P2	2
GRAPH1	GRAPH11			P3					

ANALYSIS RESULTS

Prior to project start we can now use the Project Tracking Data Base in order to make capacity estimations from the time units assigned to the 7 tiers of rated complexity.

Summary of Project AA Prior to Project Start

Project	Number of Programs	Number of Output
Project AA	7	10

Group	Complexity	Frequency	Unit	Predict
Program	H2	1	2.0	16.0
Program	L	6	0.5	24.0
Program	M1	1	1.0	8.0
Program	M2	1	1.5	12.0
Program	M3	1	2.0	16.0
QC	L	6	0.5	24.0
QC	M1	4	1.0	32.0

Resource prediction based on perceived complexity

Predict = Frequency * Unit * 8

Group	Predict Hours
Program	76.0
QC	56.0

Given that an individual only spends 6 hours per day on project AA, a scenario could be that 4 programmers are able to finish programming in just over 3.16 days and 2 quality controllers are able to finish their tasks in about 4.6 days. We would expect quality controllers will not start their task until the first report is ready. Thus, if the quality controller will start working one day later, then, the projected duration of this project would be at least $(4.6 + 1) = 5.6$ (days). Likewise, if only 1 programmer and 1 quality controller is assigned to this project, and if the assigned complexity levels remain the same, then the projected duration would be at least 12.6 working days even if the quality controller begins their task up to 3 days after the first report is ready. Either way, this total project is expected to take 132 hours to complete. It was computed by assigning a perceived complexity rating for each report.

Association between Programmer and Quality Controller

H₀: No Association
Fisher's exact: 0.1190

A test for association between the programmer and quality controller's perceived complexity rating shows that there is no significance. Again, this is not expected nor does it mean anything because the QC and programming roles have different tasks. However, if in the situation where an individual, such as a project manager, who is not taking on the programming or QC role would assign complexity to the reports, then ongoing tests for association between the project manager and each role should be assessed in order to understand any differences between the perceived versus actual complexity. If a strong association exists, then this is an indication that a non-programmer can accurately assign and predict the amount of time it will take to complete the project.

Summary of Project AA Post Project Completion

Relationship between Actual and Predicted Hours Based on Complexity Rating

Predict = Frequency * Unit * 8

Group	Complexity	Frequency	Hours	Unit	Predict	Rework	Rework_Count
Program	H2	1	14.0	2.0	16.0	4.5	2
Program	L	6	6.0	0.5	24.0	0.0	0
Program	M1	1	8.0	1.0	8.0	1.0	1
Program	M2	1	8.0	1.5	12.0	0.0	0
Program	M3	1	16.0	2.0	16.0	0.0	1
QC	L	6	6.0	0.5	24.0	0.0	0
QC	M1	4	24.0	1.0	32.0	5.5	3

Group	Actual Hours	Predict Hours
Program	57.5	76.0
QC	35.5	56.0

As we can see from this example, the data tells us that on the grand total, the actual time spent on this project was less than what was predicted. It only took 93 hours (including rework) to complete rather than the expected 132 hours.

Take a closer look at some of the discrepancies; say the reports at complexity level L. On the average, from both roles, it actually took 1 hour per report. This is equivalent to 0.125 days in relation to the time unit. This may be an indication that you should adjust your time unit from 0.5 to a lower unit, such as 0.125 for future project work in this environment. If this were the case, then the original 132 expected hours would have been 100 expected hours which is much closer than what actually happened.

Let's take a closer look at the rework rates for this particular example. The ideal and unlikely situation is to have zero rework. However, in reality, we know that it exists. The data shows that 11.8% of the total effort around this project was associated with rework. Ask yourself if it exceeds your defined tolerance, and if so, then the data is there to analyze the frequency of causes of rework that may lead to corrective actions and process improvements around the quality of deliverables.

The scenarios above hopefully provided a conceptual representation of how a strategic design of operational data can be collected and analyzed and further measure our programming and project performance. During our presentation, we will share with you some additional analysis that can be done with the above collection of project data that may be meaningful to the programmers and measure the value of our work.

CONCLUSION

Delivering all project reports on time with high quality is always a priority. In other words, quality and speed will be appreciated by our customers. The example above provides 2 components of project oversight: 1) understanding the report cycle time and maximize the availability of resources, 2) measuring quality and rework to measure our first time right performance. When a strategic design of project data is collected and analyzed over time, it can be used to unlock further trends and variations that can explain the quality and speed of our work.

REFERENCES

SAS® Institute (1999), SAS/STAT® Software: Reference, Version 9.1.3,
Copyright 1999 by SAS® Institute Inc., Cary, NC, USA

CONTACT INFORMATION

Your comments and questions are valued and appreciated. Authors can be reached at:

Hui-Ping Chen
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, IN 46285 U.S.A.
Email: huiping@lilly.com

Mark Matthews
Eli Lilly and Company
Lilly Corporate Center
Indianapolis, IN 46285 U.S.A.
Email: matthews_mark_r@lilly.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.
Other brand and product names are trademarks of their respective companies.