PD07

Metrics Unleashed II:
Measuring Productivity or Inspiring Artisans

Greg Nelson - ThotWave Technologies, Cary, North Carolina
Mark Matthews, Eli Lilly, Indianapolis, Indiana
Neil Howard –Independent Consultant, Thousand Oaks, California

Abstract

Measuring programming productivity has recently become a hot-button topic in the pharmaceutical industry. Previously, the authors outlined a comprehensive review of metrics and how they can help drive business process improvement (see PhUSE, 2006 – Nelson and Howard). In that paper, we outlined the importance of designing a metrics program and what happens when a poorly implemented measurement (incentive) program drives the wrong behavior.

In this paper, we continue that discussion by focusing specifically in on the metrics that help us understand the speed and quality associated around the production of SAS Tables Figures and Listings (TFLs) and the production of the final output for FDA submission. This paper, however, takes the topic to a more practical level by offering some recommendations on a practical approach to measure programming effectiveness, efficiency and complexity. A detailed example using clinical trial reporting of TFLs on 'what' we should measure, 'how' do we measure and report it, and provide meaningful answers for those who are asking for the metrics will be discussed.

Introduction

Metrics: making data matter

In a previous paper, we outlined a number of criteria that could be used to measure a programmer or more appropriately, a set of programming tasks. These ranged from correctness and completeness to efficient and generalizability. In this paper, we are going to focus on a relative few and demonstrate an application of metrics for clinical programming. We are going to make the basic leap that what we are really interested in is measuring those fundamental few aspects that we wish to make better through measurement.

As a reminder, in our last paper, we outlined a multi-step process for building a metrics process. We'll abbreviate that a bit here and focus on our goals, what data we have available to use and/ or can collect relatively easily and then talk a bit about how we can go after these data points. Obviously, in your setting you wont want to forget about who the consumers of the metrics reporting ought to be nor the ongoing performance improvement aspects of the program.

Performance Goals

In our example used throughout this paper, what we really want to get better at includes these key characteristics:

1. Ability to accurately estimate complexity which is useful in forecasting time and skill of resources

2. Estimate of how long each TFL will take to design, code/ develop and validate (based on complexity and skill level)
3. Some overall estimate of how many and what type of resources would be require to deliver the final set of TFLs

We recognize that there may be lots of other things that people may want to know in an attempt at getting better at managing clinical programming teams. However, for our simplistic example, these are our key focus areas. The other reason that we are interested in these simple measures is that it will require little adaptation of our existing processes to capture the details around these metrics in order to have a fairly robust set of data points that we can use for evaluation.

Metrics Definition

So let's start first with the overall process. We begin with getting the requirements from someone else – usually a statistical analysis plan – complete with data requirements and possibly even a table shell that shows us what it will look like. Since we have lots of experience in TFL programming, we take all of the requirements and begin to assess them in terms of their complexity.

Complexity in clinical programming can be fairly subjective and lead programmers will get better at this over time – especially if actual complexity can be evaluated once coding is complete. Before the SAS programming begins, a good coder would evaluate the entire group of TFLs and best determine how to approach it to what programmatically makes sense. Let's say that there is a formal and robust model that can provide the estimated number of hours it will take to deliver the final package of TFLs based on three tiers of perceived complexity: Simple, Medium, and Complex.

- **Simple:** number of reports with code reusability very high, output format same only data changes, single macro with multiple outputs.
- **Medium:** output formats are different with little code re-usability
- **Complex:** novel or first-time analysis, very less reusability

For practical sense, let's use the assumption that a Simple TFL will take 4 hours to program/validate, Medium takes 8 hours, and Complex takes 16 hours, based on historical averages. For example, if the model suggests that it will take 640 hours to complete this project, and we know this well in advance, then it can be planned and best decided whether to put 2 people on this for 8 weeks, or 8 people on this for 2 weeks. Once you have your group together, then let the project begin.

So at this point we have a few decisions that have been captured, namely:

- Number of TFLs to produce
- For each TFL a measure of their complexity (on a 1 to 3 scale)
- Resourcing decision (what skill level will be applied to each TFL)
- Overall estimate of hours to completion

We may also want to consider historical data here and estimate how many TFLs usually require rework. Rework is defined when the programming and validation has been complete upon initial requirements but the program requires revisiting due to one of two types of conditions.

- **Type N1** is when the program contradicts the requirements and a necessary change to the code was needed to meet the intended analysis. (e.g. the programmer and validation overlooked the required covariate in the model.)
- **Type N2** is when the program does not contradict the requirements, but a scientifically necessary change to the code was needed to meet the intended analysis. (e.g. gender specific processing on female pregnancy events only did not clearly define to eliminate post menopausal female subjects from the analysis).

We also have some context in our data – namely, those related to the therapeutic area, compound, skills profile of the coders, how many coders, estimates of complexity and level of effort, some forecast of rework and we are ready to begin data collection.

Data Collection

For ongoing data collection, we use a simple time tracking application, which captures who (programmer), what (task or activity – e.g., which TFL), which aspect (design, coding, validation, production generation or rework), and how long the activity has taken. Each day (or more often!), the programmers enter their time in the tracking system.

Once programming begins, imagine that the programmer enters in the number of minutes (say to a 15 minute precision) toward Project_X at the end of each day. At strategic milestones, or at project completion, additional data can be collected that highlights the actual complexity as well as how much time was actually spent on each TFL and how much rework was done. Here we have a summary of the data we have collected on this project.

Therapeutic Area: Oncology

Compound: ABCD

Indication: metastatic melanoma

Study: Project_X

Start Date: 03_MAR_2008

Stop Date: 15_MAY_2008

Programming group: Outsource_A

Number of involved individuals: 4

Planned effort (hours): 640

Initial Number of TFLs: 100

Perceived # simple: 60

Perceived # medium: 30

Perceived # complex: 10

Hours Spent on programming: 500

Hours Spent on validating: 400

Total Hours spent on programming and validation: 900

Number of Type N1 handbacks: 6

Number of Type N2 handbacks: 8

Hours Spent reworking N1 handbacks: 12

Hours Spent reworking N2 handbacks: 40

Total Hours spent on total rework (N1 + N2): 52

Initial Number of TFLs: 110

Actual # simple: 50

Actual # medium: 40

Actual # complex: 20

This is a simplified version of our data but the coders only spent about a few minutes each day capturing their time and a one-time assessment of the complexity distribution around the TFL programming before and after the project. It is not too much effort to do this, however the relevance around this simple entry from the programmers can make quite an impact.

Reporting and Continuous Improvement

There is a lot to analyze within this single record. Why did we expect it would take 640 hours to perform the work when it actually took 900 hours? Why were there 8 handbacks from, say, unclear requirements? What is the root cause of the 6 N1 handbacks? Does it suggest that it takes more than twice as long to 'repair' N2 handbacks than that of N1? Why were there an additional 10 TFLs brought into the project? Why did the complexity shift toward a more complex project at the end? Can we further analyze the perceived effort versus actual effort and make our future estimation of capacity model more robust? Is the time validating with respect to actual programming excessive?

These questions can be asked and should not go unnoticed unless it is not desired to improve with the upcoming projects. To analyze this on a project specific level is good, but when this same data is captured over time and across multiple projects, it is possible to unlock hidden trends and variation around the bigger picture.

Now imagine that this is 1 of 75 studies in a database that consists of 5 therapeutic areas, 12 compounds and 20 indications over a 3-year period that consists of over 7,000 TFLs. A simple database such as this allows for a further analysis and understanding of the business. How does compound ABCD quality and speed compare to the other compounds? Has compound ABCD been improving in speed and quality over time? What has been the overall rework rate of all projects and is it time to take corrective actions on this unnecessary waste?

Advanced Analytics for Metrics

So as we begin to populate our database with lots of data from lots of protocols, we wanted to take a moment and talk about measuring quality and speed and that can be turned into practical resourcing decisions.

A few fundamental equations- Measuring Quality and Speed:

Consider the following equation: Quality

$$(\text{Time})^{-1} \times (\text{Factor\_A}) \times (\text{Factor\_B}) \times \ldots (\text{Factor\_i}) \sim \% \text{ handbacks}$$

This fundamental equation may be applied to any statistical programming. It is only suggesting that time is inversely proportional to the % handbacks should all the other factors remain constant, and thus, the quality improving. For example, let us consider that Factor_A were to represent a quantification of "People experience" and Factor_B were to represent a quantification of "Scope of work" and there were no other factors in this equation. Then, if we were to repeatedly give the same work to same people over and over again, then (people experience) and (Scope of work) remain constant (people experience remains constant because they do not learn anything new with the same work) and thus, theoretically, over time, the number of handbacks will approach zero. However, when the people change or if a new scope of work is presented, then this offsets the $(\text{Time})^{-1}$ factor and can result in a increases in % handbacks. What is the result in quality when both (new people) and (increase scope) are added? Let's measure it and understand

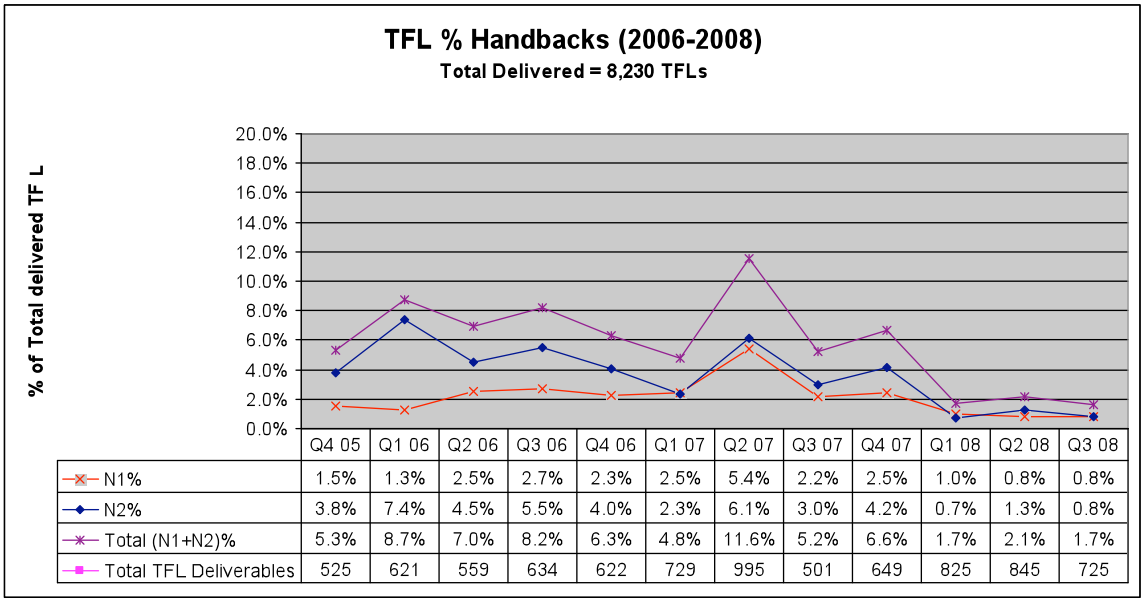
Consider the next following equation: Speed

$$(\text{Time})^{-1} \times (\text{Factor\_A}) \times (\text{Factor\_B}) \times \ldots (\text{Factor\_i}) \sim \text{cycle time}$$

Again, it is only suggesting that time is inversely proportional to the cycle time, and thus improving speed. Is it suggesting that the cycle time will approach zero? Maybe. With the computational power ever so increasing in conjunction with the right use of metadata (enter once and use many times) and standards, a click of a mouse just might produce all tables, graphs and listings instantaneously. The equation is only suggesting that over time, the cycle time will eventually reach some horizontal asymptotic line until something in your equation changes and either improves or worsens the process. For example, let Factor_A be a quantification of migrating your computing environment from one platform to the next, or another factor is the rollout of a series of reusable code against a new set of standards, or another factor is the use of different outsourcing. Do these factors increase or decrease your cycle time. So let's measure it and further understand by selectively choosing the factors that affect both quality and speed that interest your business.

An example- Measuring Quality:

Let us take a look at how data, if captured in the above structure, can be graphically analyzed and displayed. In this case, the actual result is not real and we generated bogus data for illustration/example purposes only.



	Q4 05	Q1 06	Q2 06	Q3 06	Q4 06	Q1 07	Q2 07	Q3 07	Q4 07	Q1 08	Q2 08	Q3 08
N1%	1.5%	1.3%	2.5%	2.7%	2.3%	2.5%	5.4%	2.2%	2.5%	1.0%	0.8%	0.8%
N2%	3.8%	7.4%	4.5%	5.5%	4.0%	2.3%	6.1%	3.0%	4.2%	0.7%	1.3%	0.8%
Total (N1+N2)%	5.3%	8.7%	7.0%	8.2%	6.3%	4.8%	11.6%	5.2%	6.6%	1.7%	2.1%	1.7%
Total TFL Deliverables	525	621	559	634	622	729	995	501	649	825	845	725

We can see that from Q4 of 2005 through Q4 of 2007 the quality has not been improving as theoretically should happened, and it can improve. So why was the rate of handbacks? What was offsetting that time factor? Go back the fundamental equation and it just may be explained by the information that exists in the database. A further slice and dice of the data has been analyzed and the result came up with 2 target areas of improvement.

- There is a measurable and strong correlation between the numbers of handbacks with respect to the duration of the project. A further look at the start and stop dates suggest that the longer to project duration, the more handbacks.
- There are measurable spikes in handbacks that pertains to Outsourcing_B. It can be shown that Outsourcing_B has a significant and quantifiable degree of higher handback rate.

So what was done to make a measurable increase in quality in Q3 of 2007? Was a full-blown six-sigma project launched to find out why the duration has an effect? Was Outsourcing_B eliminated from all future projects? Not necessarily, because there was a more simple, effective, and more immediate approach.

After a root-cause analysis on outsourcing_B intercontinental operations, it was found that the communication flow was at an imbalance. The preferred method of clarifying requirements from a 12-hour time zone difference was through voicemail iterations in which there was excessive communication loss from verbal requirements. Instead of eliminating outsourcing_B from future work, a collaboration space was set up on a secure website which allowed for written clarification of the requirements.

Also, beginning in Q4 2005, there was a tendency to put 2 people on the project for 8 weeks as opposed to 8 people on it for 2 weeks for a job that needs 640 hours to complete. The data was telling us something and it was explained by the simple fact that the customers of the TFLs and those who defined the requirements were not prepared to address any requirement ambiguities and provide sufficient requirement detail that far in advance; perhaps because more attention was placed on projects that needed to be completed sooner. Thus, beginning with Q1 2008, an action was to lean toward the tendency of condensing the amount of programming time closer to the final deliverable date.

These 2 factors in conjunction with each other could have been offsetting the time factor because the next year's worth of data suggests that the handback rate is approaching zero and all else remained constant.

What other worthwhile areas do you think could be analyzed? A similar analysis can be conducted for speed/cycle time as suggested in the above section, but we will leave it to the readers to craft their own strategic analysis of the data and provide it to the consumers of the metrics.

Summary

There is no question that measuring quality and productivity in the context of statistical programming is a key differentiator for any company spending its share of the multi-billion dollar drug market. We have to produce higher quality software in shorter time periods. We are faced with off-shoring and out-sourcing decisions all of the time. Justifying team composition and head count is part of our everyday world – as is balancing work load. It is not a question of if we should measure, but how.

The scenarios above hopefully provided a conceptual representation of how a strategic design of operational data can be collected and provide a significant increase in the knowledge of our business with little additional effort from those who are actually doing the work.

This was only for operational metrics on clinical trial TFL reporting. Considered how it can be extended into SAS programming on the development of reporting databases, non-trial TFLs such as regulatory responses, ad-hoc or integrated summary of efficacy/safety, programming on electronic submissions, etc.

Let's pretend the world is flat and every CRO and Pharma/biotech company participated in a centralized version of this. Take the database above and add the following elements to it: Company (ThotWave Technologies, Amgen or Eli Lilly) and Geography (US or Japan) and Platform (SAS Drug Development or Unix). Can you imagine the information we can draw from this? Perhaps we could see consistent demonstration of Company_Z in Europe having a much lower cycle time on programming efforts with higher quality (all quantified as to exactly 'how much' and 'how high') explained by their portability across platforms because they use SDD. We could actually measure and identify the areas of top performance and share best practices and reduce cycle time and improve quality of clinical reporting across the globe, and thus, contribute to the ever-increasing demand from our patients to lower medication costs. It would bring the SAS programmer to a new level of renowned value and truly inspire our artisan coders.

Acknowledgments

The authors would like to thank Jack Shostak and Kyle McBride for their contributions to the panel discussion on metrics at PharmaSUG 2006 in Bonita Springs, FL which served as a springboard for this paper. We also would like to thank Jim Baker at Amgen and Richard Phillips at ThotWave for their "thotful" review.

Author Contact Information:

Greg Nelson, President and CEO of ThotWave Technologies, LLC.

ThotWave Technologies, LLC.
2054 Kildaire Farm Rd #322
Cary, NC 27511
800-584-2819
greg@thotwave.com

Mark Matthews

Eli Lilly and Company
Lilly Corporate Center
Indianapolis, Indiana 46285 USA
317.433.6521
matthews_mark_r@lilly.com

Neil Howard

Amgen Inc.
One Amgen Center Drive, B24-2-C
Thousand Oaks, CA
805-447-5713
neil.howard@amgen.com