PharmaSUG 2019 - Paper LD-174

It's a wonderful day in this neighbourhood – managing a large virtual programming team

Victoria Holloway, Covance Inc.

ABSTRACT

Programming can be lonely, isolated work within a large virtual team. Whether in the office or at home, everyone needs to be part of something bigger. Creating a virtual neighborhood-style community center enables the programmers to interact with each other as neighbors. Community is the first step for building trust and respect through experience and successes. Important public works projects including infrastructure, such as electricity, water, phone, and internet are essential to the community and can be likened to training, code sharing, work breaks and process improvements. Nothing happens unless there are funds to support these endeavors, so linking the team to the financial aspects is also key to running successful projects. Counting the beans includes how to track the work and project management tools. This presentation will share my experience as a new manager where neighborhood concepts helped ensure that the work was done on time, within budget and by a competent team happy and able to work together. Mr. Rogers had it right: creating "a wonderful day in the neighborhood" allows for successful projects even under difficult circumstances.

INTRODUCTION:

I will begin by describing the scope of this project and then I will describe my experiences as a virtual first-time manager of a virtual SDTM programming team.

SCOPE OF THE PROJECT

This project involved analyzing over 70 studies in many different therapeutic areas, in multiple phases and over many drugs. Many of the studies were in oncology. To submit to the FDA, the client needed submission-ready SDTMs and documentation.

An SDTM team was created. I was put in charge of programming (14 people) and my co-manager (also brand new) was responsible for the specifications team (six people). We both managed the submissions team (three people), which was going to start up later in the project. And we had a Sr Manager to help guide us.

The SDTM team delivered to three groups: the internal ADaM safety analysis team, the client's data cleaning group, and the client's ADaM efficacy analysis team.

As well as SDTM creation, the client required "refreshed/revalidated" SDTMs for each significant milestone within a study (interim analyses, etc.), under very tight timelines.

THE CHALLENGES

As a new manager, I was excited, but nervous. I had programmed some domains for a few months, and I had written some of the specifications, so I knew the work. But this was my first virtual programmer job, so I was still adjusting to that.

In the first weeks, three issues became of utmost concern: 1) how do I manage people (both new people and people I had worked with as the programming lead), 2) how do I manage from afar (virtually and from another country), and 3) how do I track all the work being completed and translate that into productivity numbers and revenue.

I had very little idea how to approach the management of this large virtual project. This presentation describes how I survived this situation – and even thrived.

As I reviewed the SDTM team, I noticed that programmers were communicating infrequently with each other. I saw that spec writers and programmers were often annoyed with each other. It appeared that the

team did not act like a team – they appeared to be just a bunch of people working on the same project. For example, during weekly team meetings, no one raised questions or concerns, and when prompted for feedback, everyone said “It’s all fine”. I knew the staff were very skilled at SDTM work, but they seemed unwilling to provide feedback on the work out loud or even to initiate any discussions.

Working virtually for the very first time, I had begun to feel very isolated from people. In an office environment, there are always spontaneous conversations about the work, and the non-verbal communication you see shows the unspoken feelings people have about issues. While we used Skype with video for some meetings, many of these non-verbal ticks did not get conveyed in all their nuances. I thought: maybe everyone else is also feeling isolated from each other, and from me their manager?

To achieve a dynamic team state which I knew the team was capable of, something had to be done. But what to do?

BUILDING THE NEIGHBOURHOOD

I conceptualized the SDTM team as a high-tech neighborhood, where each person represented a different dwelling. People were connected to each other through high-tech communication tools like Skype, email, phones, meeting tools, etc. Just like in a physical neighbourhood, people could step outside and chat as neighbours. The infrastructure provided us with data and requirements. We had sewers for processes and code and expectations that were no longer needed. We had roads to other neighbourhoods, where people could visit or come from, or even move to if they didn’t want to stay. We would have safety, company, tools, infrastructure, room to grow, learn and develop. If I could help people see this and work to this, we could have a pretty nice neighbourhood to live in.

First, I spent some time thinking about my previous managers – figuring out what I liked about each and what I didn’t, and what I totally disagreed with. I liked managers who treated their staff as competent skilled people and did not ask me about every little detail. I liked having a specific time (weekly or bi-weekly) when I had my manager’s undivided attention to talk about the work, my career, and the issues I was facing. Together, these things gave me a direction for my personal guidelines as a manager.

MEETING THE NEIGHBOURS

Secondly, I realized that I didn’t really know any of my neighbours. I would need to meet them, learn about them, and understand them. I wanted to provide my team with lots of access to me since we were separated geographically, so I chose to meet with each programmer weekly. I wanted them to get to know me and I wanted to see who they were. I wanted to encourage open and honest communication. I wanted to alleviate their fears of these organizational changes, and to bring out their needs and ideas.

In our first few meetings, I asked a lot of questions. I assured them that while I was a new manager, I had 20 years’ experience in the industry and many in leading projects, and that I would always let them know what was happening in the project and in the company. I told them that my “virtual door” was open.

Most were happy to talk about themselves and their work. They offered comments and complaints about the daily work. Some programmers indicated they were not happy in SDTM-only programming and wished they could get back to ADaM programming. Others noted they were happy to have such a large bolus of work in front of them.

This set the stage for open and free discussions, which allowed me to establish a positive working relationship with each programmer.

DEVELOPING THE COMMUNITY CENTRE

Thirdly, I realized that we did not have a “community centre” in our neighbourhood – a place where we could meet, relax, learn something new and talk to each other.

Previously the team had been having a quick weekly 5-10-minute meeting where the next studies for the week were presented. There was no planned time for learning, or questions or comments. When I attended this meeting as a programmer, I did like to know what was coming up, but I did not feel like I was part of a team.

I restructured the team meeting. My purpose was to build a unified team, where both spec authors and programmers understood everyone else's role in the creation of quality and on-time SDTMs. I wanted people to change their mindset from individual to group, from single home to a neighbourhood.

I made the team meeting longer. I certainly kept the upcoming work as one of the first agenda items, but I also added introductions that would help neighbours get to know each other better (a brief work history and information on their current role). I also set up time in the agenda for discussion of specific specs and programming issues. This discussion helped each person to see and understand problems their neighbours were dealing with. It allowed everyone together to come up with a solution, or at least a path to a solution. I even designated one person to tell a "joke of the week" – this turned out to be a happily-shared task. And of course, I encouraged questions and comments.

The first team meeting had quite a lot of silence. People did not share any issues or voice any opinions on the work. But over time, with lots of encouragement and examples and humour from me, they began to speak up more often.

It became clear there was a definite rift between two "streets" in the neighbourhood: between the spec authors and the programmers. Programmers were annoyed with two items: the frequent typos in the specs, and with the lack of detail in the description of merges between source datasets. In turn spec authors were tired of fixing minor typos, and they didn't understand why programmers needed more information on a merge. The phone lines were down – each side glared at the other. I helped both sides see that while typos are minor, they are annoying, and we agreed they should be corrected since all specifications were to be reviewed by the client. On the merge issue, several examples of merges that were described clearly and worked correctly were shown, and those merges that were unclear and lead to many interpretations of the merge, and therefore didn't work well were also shown. This demonstrated why more merge details were needed. Communication was key – the phone lines were back up.

Our neighbourhood contained quite a variety of homes: we had both spec writers and programmers, who varied in levels and experience from new to mid-level to senior. As a neighbourhood, we needed to understand that a person may have very good abilities in one area and not so much in another. I believe that everyone has something to offer, be it excellence on a particularly gruesome domain, or a nice code snippet to catch all source variables, or flexibility on which domains they could programme or author. Not everyone can do everything. As an example, we had a spec writer who was quite slow. They were offered programming work, and it turned out that they were quite good at programming!

In the team, I made it very clear that the timely completion of the task was paramount and that this was the responsibility of the whole team. If someone's assignments were completed ahead of the rest but the study was not yet delivered, that person should jump in to help get the whole task done. I encouraged a helping mentality within the team to create flexibility - there is always a "hot priority" task that comes up unexpectedly, and we needed to be agile.

ERROR? WHAT ERROR?

I have worked to actively change the meaning and use of the concept of error. Error is often used as a sledgehammer to scare people into working harder and longer hours. But if error is used as a tool to improve productivity and processes, it can be a wonderfully dynamic thing.

I encouraged both sides of the programmer duo (production and QC) to TALK to each other, rather than blame each other for coding differences. There will always be differences and mistakes in the production of datasets, otherwise we would not need to QC. Error is not something to be feared – it should be used as way to grow. Error is a key part of learning – without it, we won't learn anything new and are doomed to repeat the same mistakes again and again. While I don't know how many people fully embraced error, the team certainly relaxed when they realized I was not using error as a penalty against them, but as a tool to improve productivity, processes and personal abilities.

There are errors that not particularly significant. It is more like a small pothole in the road that does not impede the flow of traffic, but it is of a size that may take some effort to fix. However, one must be aware of potholes because sometimes, in different circumstances, they can become so numerous or so deep that the road is unpassable.

There is absolutely no way to be error-free. If that is happening in your project, then you are not seeing the full picture: the error is there, you have just not identified it yet.

You may find that after errors are found and you have applied fixes, updated processes and/or created new checking macros, there are still some discrepancies that keep coming up. An analysis of these errors may point out that an individual may not be up to company standards, or there is a bias in a macro (for example). Reviewing error allows you to see if someone (or some process) is weak but has the potential to develop (or be easily fixed), or if someone should not be part of your team (or a new macro is needed). Identifying such things is necessary for a team to succeed.

COUNTING THE BEANS

I found data showing programmers and spec writers dates and domains created during a study. Now I had a count of each team member's unique domains by study completed by week.

I gathered more data on the studies. The study planning spreadsheet was expanded to include much more information: re-work requested by the sponsor and ADaM teams by domain and reason, specifications dates (expected and due), and type of deliverable.

Then I added the revenue information for each task, so our management team could quickly figure out the revenues for upcoming deliverables. This took several months, and the spreadsheet eventually became the key source of information for billing completed SDTM work.

PATTERN RECOGNITION

So now I had an objective picture of who was doing how much every week. I tracked it for the next number of weeks and began to see patterns that I found very interesting.

- Pattern 1: When the team was mostly focused on delivering a brand-new study, the average number of domains per person per week was steady. When delivering refreshed/revalidated studies, the total counts grew by quite a lot. Were we comparing apples to oranges here?
- Pattern 2: Experienced programmers could do two to three times more domains than less experienced / newer programmers. To see the breadth of that range was enlightening.
- Pattern 3: The number of finalized specs was less than the number of finalized programmes. Were the spec writers too slow or was the balance between programmers and spec writers wrong?

The patterns I saw showed me very specific features of this project. It forced me to examine functional task ratios, people's skill sets, performance levels and certain assumptions of our data.

I spend days looking the data over, trying to see correlations to help make the team more productive and streamlined. As time passed, I began to remove data that did not contribute to increased productivity or did not help with predicting events.

Examining data (quantitatively and qualitatively) is akin to improving the unseen infrastructure of the neighbourhood. This allowed me to highlight and clarify issues in the team meetings, in order to improve productivity and encourage team dynamism.

One can get bogged down in too much data. Only over time, as you use the data, can you begin to see what is essential and what is less so and what is not useful at all. Let the data guide you.

COUNTING MORE BEANS

So now I knew the relative productivity numbers. But I didn't know if those numbers were in line with the contract, or even if the work was profitable. Because I didn't have all this information, I had great difficulty forecasting the next month's work and delivering to that forecast.

Senior management provided us with the contract for the project, together with the value of each task and other assumptions.

They also provided us with the company's standard calculator for hours per task. This provided us a baseline for comparing our team's work to the whole company.

I crunched those numbers and came up with the project-specific hours per domain, for both specifications and programmes. Then I compared those hours to the costs we were incurring. This allowed me to see how our team was performing against the contract expectations as well. Now I knew we were on track.

GAINING TRUST AND RESPECT THROUGH HONESTY

I decided to discuss the domain counts report with each of my direct reports, all of whom were programmers. At first, I was leery of letting people know I was "watching" them like big brother. However, my instinct was to share so that everyone would have an equal chance to improve.

I explained how a person's totals were counted, and I indicated where they were in the spectrum (high, low, mid-level). I took the opportunity to find out exactly how they felt about their domain assignments, the difficulties they faced in programming them, and the type of domains they were good at and those they weren't. I was never judgmental – the focus for the below average programmers was solely on bringing up their numbers to the average for four weeks in a row. For those who did above average, I asked how they were so successful, and brought some of that information back to the below-average programmers. We then planned the next week's numbers by discussing the upcoming planned work. The following week, we reviewed the numbers to see if they were met and why/why not.

After the first weeks of this, I asked if they saw this report as useful. The overwhelming response was YES! Without any prompting, most of them told me that they never knew what the standard was. It had always been push, push, push, with no feedback when they did well or not so well.

Sometimes it took a while for a below average person to bring their numbers up (up to eight weeks in one case, but most took only a couple of weeks).

Buoyed by this success, I began to share the programmers' numbers with each other. Then I began to share all numbers between both spec writers and programmers. That truly made the point that it was possible to do "that many domains" every week. This created an open and friendly competition to do more in the same amount of time.

As well, I made a point of not cancelling the weekly 1-on-1 meetings, or the team meetings, nor cut them short. Since we did not have the opportunity to see each other very much given that there were no hallways or water coolers or lunchrooms in our virtual world, the time we spent in meetings and chats seemed relatively small. I believe that the frequency of contact and the openness of the communication was the key determinant in developing the neighbourhood into a productive place.

Over time, I found that a person's commitment to the neighbourhood reflected their trust in their colleagues and their manager. A neighbourhood must be open to improvements and growth – this will allow both individuals and the team to grow and learn and overall increase productivity.

In 3 months, our billable work value doubled, then in six months doubled again. We were now a significant financial part of the whole project.

CONCLUSION: NEIGHBOURHOOD CHANGES

The character of a neighbourhood is continuously changing. New communication devices are introduced as the tech evolves. As a project grows, new homes are needed, new plumbing and connections are developed. The community centre will need to be expanded, and more green space added. Similarly, when a project is winding down, less infrastructure and even homes are necessary. Or if a new project is to be created, some of the existing infrastructure may need to be removed, including processes that worked for one project that won't work in the new one. The neighbourhood needs to meet the needs of the project.

And if everyone keeps their lawns tidy, talks to each other frequently, helps each other out, and is aware of the state of the infrastructure, the neighbourhood will be a really nice place to live.

As Mr. Rogers says, "It's a beautiful day in this neighbourhood".

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Victoria Holloway
Covance
Victoria.Holloway@Covance.com