Multiple Simultaneous Projects with One eXtreme Programming Team

Jason Nocks
SourceXtreme, Inc.
201 Exton Commons, Lower Level
Exton, PA 19341 USA
nocksj@sourcextreme.com

Abstract

The group at SourceXtreme feels that it is essential to deal with multiple simultaneous projects with one eXtreme Programming team. Many people feel that too much time would be wasted task-switching, that the team would suffer too much from fractured thinking, or that a sense of team would be lost. It's probable that these outcomes could happen and that focusing on one project might be better in general. This paper describes some of the reasons why the group feels that working on one project wasn't an option at SourceXtreme and what approaches have emerged over the past year to thrive in this situation.

1. Introduction

For many teams, the “Simplest Thing That Could Possibly Work” [1] is to have the team work on one project at a time, with one Customer that is onsite and part of the team. For our team, working on multiple simultaneous projects is financially sustainable. With some effort, we have improved our sense of team by acknowledging that what seemed like many distractions are actually multiple projects that we should account for.

Our team initially tried to shield developers from dealing with more than one project at a time. We also tried to minimize distractions. We realized that we were struggling to meet our commitments. We decided as a team that it didn’t seem like it made sense for us to only focus on one project at a time. We knew that many people felt like large task-switching penalties were inevitable. Our team was willing to experiment to try to figure out what eXtreme Programming might look like with multiple simultaneous projects. Our team has been satisfied with the results we've come up with so far and we thought it would be a good idea to share these results.

2. Background

Jason Nocks founded SourceXtreme with Ian Reinhart Geiser in October, 2002. The company specializes in developing custom cross-platform desktop applications. Projects range from Golf simulators to Diesel engine test systems.

At the time of this writing, SourceXtreme typically has a small number of developers working together as one team while delivering a small number of projects at the same time. Typically the team consists of about 3 or 4 developers. The team scales up or down slightly with independent contractors when it makes sense. The group typically works on one larger project and up to a couple of smaller projects at the same time. For the team, a larger project might be many hundreds or a couple of thousand man-hours. A smaller project might tend to be on the order of a hundred hours.

3. Problems our team faced

Our team found a complete void of information on how others would deal with multiple simultaneous projects. The team had to blaze new territory. This hasn't been easy, but in hindsight wasn't terribly difficult.

3.1 One team

In addition to trying to make the Customer an integral member of the team, team unity among the Developers is essential to us. In the past, when trying to isolate team members from multiple projects, the result was each developer specializing on no more than one project. This meant different team members working on different projects. We often operated more like multiple one-person teams working under the
same umbrella. We were struggling to develop a better sense of team. It felt more like a top-down organizational structure with multiple sub-teams, each consisting of about one person. At least one team member said “We are not working together as an XP team”.

When we've only worked on one project at a time, the whole group would lurch from little to no work, to being very busy and back again. There would also be times when some developers were idle while others were overloaded. This did not feel sustainable.

We sometimes also have gaps between story iterations for some customers while they spend more time working with the delivered functionality. After a delay they are then ready to provide feedback and the next set of stories.

3.2 Working on one project at a time would be simpler

When discussing multiple simultaneous projects, many people feel that too much time would be wasted task-switching. There is an automatic assumption that the team would suffer too much from fractured thinking, or that a sense of team would be lost.

In eXtreme Programming: Explained, Kent Beck writes that “so much time is wasted on task-switching that you can see immediate improvements by grouping the programmers into teams.” [2] Focusing on one project is probably the best general advice. In addition, much of the advice against multiple simultaneous projects seems to be aimed at developers being shared among several teams. That is not the approach we adopted.

It's certainly possible that “fractured thinking” might happen or may be probable in the general case. This might be particularly likely if no attempts are made to counter “fractured thinking”. Our team feels that the approaches described in this paper are some ways to help reduce “fractured thinking”.

Tom Demarco further suggests that there is a built-in task-switching penalty of no less than 15% for knowledge workers. [3] While this is no-doubt true of distractions and splitting one worker among different teams, our team has tried to take a different approach. There is little doubt that distractions cannot be completely eliminated. Our team has focused on evolving techniques for dealing with regularly occurring “distractions” as normal events rather than “distractions”.

3.3 Challenges from additional responsibilities

The team did face some turnover. One developer left to move closer to family. Another left to pursue a specialty in software development that our team doesn't regularly work on. In hindsight, the fact that we now work more closely as a team has lessened the impact of turnover. Still, hiring is a responsibility that impacts the team.

Activities related to sales and lead generation are probably the most asynchronous and seemingly unpredictable in nature. The sales process we use is heavily communication intensive. The team cannot tell prospective Customers that we will get back to them in a few weeks or months. Not factoring in the impact of these types of activities wreaked havoc on any attempts to come up with a group schedule.

4. Approaches we evolved

4.1 Separate internal team task iterations from customer iterations

Some of our Customers have shown great interest in how our team works. We do not want to hide anything we are doing from the customer. In general, however, most of our Customers have no interest in our task-iteration level of planning. We started planning internally focused task-based iterations with the entire onsite team. We would then plan at the story level with the customer, which was mostly the level that the customer cared about anyway. Separating our task-planning from our Story-planning iterations is probably crucial to our ability to cope with multiple simultaneous projects.

4.2 One Day Iterations

For a while, our team did have difficulty coming up with an iteration size that felt like it was working for us. We tried scheduling an entire week. We found that there would be too much change coming from the different projects. We couldn't come up with a schedule that we felt would last more than a day or two before it was obviously wrong.

It occurred to the team that we needed to be able to slip frequent “distractions” to the next iteration. We wanted to absorb critical issues that were small enough to not substantially impact the current iteration. It was apparent that smaller task iterations were needed. With a higher frequency, normally disruptive priority changes can be scheduled within the next iteration. Small enough phone calls or emails can be absorbed during the current iteration when it makes sense.

We started shrinking the iteration size. For the team’s effectiveness, we realized that we needed to have really small task-based iterations consisting of one day. Attempts at larger iterations were disastrous. We even tried half-day task iterations. The team was actually successful at half-day task iterations, but we
did reach a level where planning was consuming almost as much time as development. That is still an issue we try to keep aware of. The team viewed half-day task iterations as a good exercise to improve our ability to plan efficiently and minimally.

We ended up with one day internal task iterations. We do not waste a lot of time task-switching because we have signed up for and planned out the tasks for the day prior to starting to work on any of the tasks. Each developer can just look at the short list of tasks they have signed up for and pull them off the board one after another.

At the customer/story level, we typically end up with one or two week story iterations for each customer. These iterations don't necessarily coincide. Sometimes we also loosely discuss broader groupings of stories at a level of weeks or months. At those times, we've grouped the stories along common themes that are important to the customer. We've contemplated what estimating and measuring at the level of themes might look like, but have not had the need to yet.

4.3 Distractions may be additional projects in disguise

Because our team is also the majority of the workforce for our company, much of our team also has other responsibilities. This includes working on the company website, helping with sales and accounting, hiring, improving our tools, writing papers for conferences, etc. Other people may have similar “distractions”.

Our team decided to treat these distractions as additional projects that our team takes on, where the company is the Customer. We treat creating and improving some of the tools we need in a similar fashion.

The team found that this actually simplifies our daily workflow. It also allows us to make sure that these projects get accounted for and get time needed. Because these customers are “in the room”, we need to make sure that their voices don't drown out the Customer proxies or remote Customers.

4.4 Projects have ebb and flow

Our team feels that supporting past clients, including minor improvements is crucial. Working on multiple simultaneous projects gives us flexibility to schedule these events. We have the flexibility to adapt to minor fluctuations in priorities as parts of the projects have increasing and decreasing demands.

4.5 Accountability

Our team wanted to perform better. The team wanted to be able to commit to doing a certain amount of work in a given amount of time, and meet that commitment. These were commitments we wanted to make to ourselves.

Our team feels that we have been more honest with ourselves when we acknowledge that “distractions” are other projects that make demands for our time. Our schedule is more accurate and we see better results when we account for these other projects.

4.6 Fewer themes to not lose focus

There have been days when team members have struggled to get their tasks done. The team works to maintain a supportive, constructive retrospective. Through this, we've found that reducing the number of themes any individual team member is responsible for during the day greatly improves our ability to accomplish the day's tasks.

For example, we encourage team members to only sign up for tasks that have a consistent, coherent theme that ties them together. These tasks may be on the same or different projects.

5. Results

The team typically tracks hours worked for billing purposes. In addition, it’s important that time be tracked according to the correct project and customer. However, we try not to spend too much time measuring. Prior to January 2006, “distractions” would often not get tracked at all.

Starting in January, the team made a concerted effort to track hours for all of the projects and “distractions” that are material. An internal tracking tool was also enhanced at that time to simplify tracking to make sure that we captured the data more regularly.

Fairly recently, the team started running more reports to see if we were spending large amounts of time due to task-switching overhead. Because we sign up for tasks at the beginning of each day, we would expect this overhead to show up as increased time spent in our planning and stand-up meetings. We gathered about 3 months of: hours worked per week, number of projects worked for those weeks, and the amount of time spent in planning and stand-up meetings for those weeks. A couple of weeks with no time worked were removed. Figure 1 is a scatter plot of the % time in planning and stand-up meetings for a week versus the number of projects worked that week.
The data is far from conclusive. There are weeks that our team felt like the planning and stand-up meetings went on too long. This is reflected in the data points above 20%. However, there does not appear to be an obvious directly proportional link between the number of projects worked on during that week and the percentage of time we spent in the planning meetings.

The average was about 11.25%. In a forty hour work week that is about four hours per week. Furthermore, this is time spent discussing priorities, signing up for tasks, etc. This is not “wasted time” due to “fractured thinking”.

Our team will continue to gather more data and look for better metrics. The team does feel that the data matches with the general feeling on the team that we do not generally experience a task-switching penalty directly proportional to the number of projects we work on during a week.

6. Lessons learned and areas for improvement

During retrospectives, our team has discussed some ways we'd like to try to improve planning.

We're looking for additional ways to incorporate Slack. When focusing on only one project at a time, there will be natural downtimes waiting for the Customer team, etc. These downtimes can be used to handle “important, non-urgent tasks”. [4] Our team needs to be careful that we don't lose Slack entirely when these downtimes are filled in by other project demands that arise to fill these voids. With no Slack, we lose flexibility and capacity for improvement. [5]

We learned that some of the things that we thought were distractions were actually other projects competing for our attention. We learned that we needed to account for some of these “distractions”, as we noticed how frequently they occurred.

The biggest area we need to improve is estimating earlier and giving customers better numbers earlier in the project. This sometimes has the feel of fixed cost or fixed bid projects, which we are eager to avoid. We've stumbled a bit in this area and are looking for ways to improve. However, this is outside the scope of multiple simultaneous projects.

7. Conclusion

Our group has improved the way we deal with multiple simultaneous projects, a situation we face just about every day. We initially struggled with our situation, including denying that we even needed to come up with a way to deal with multiple simultaneous projects. During this denial period we felt like a fractured team. We struggled to come up with team plans that we could meet with some consistency. As we developed some simple approaches to our situation by thinking about the problem a little differently, we've met with greater success. We've accepted that multiple simultaneous projects feels like a reality that we need to get better at dealing with rather than a problem we hope will go away or that we need to eliminate. The team welcomes additional suggestions and will continue to look for areas we can improve.

8. Acknowledgments

Thanks to the team members of SourceXtreme, and Ian Geiser in particular. Thank you to Monica Yap for reviewing this paper and helping with clarity, focus and flow issues, in addition to the mundane typos, etc. I also need to thank my family for their ongoing encouragement, support, and catching typos. Thank you in particular to Bill Wake, a few attendees of XP
Day DC 2005, and the employees of Digital Focus. A number of thought provoking discussions with these people encouraged our team to more fully explore and share how we've tried to work with multiple simultaneous projects.

9. References

[5] Slack by Tom Demarco, page 34