Delivering Fast with Confidence

"The Slack Time Pattern"

Joseph W. Yoder, The Refactory, Inc. – USA
Hironori Washizaki, Waseda University – Japan
Ademar Aguiar, FEUP, Universidade do Porto – Portugal

Many software development processes such as agile and lean focus on the delivery of working software that meets the needs of the end users. Many of these development processes help teams respond to unpredictability through incremental, iterative work cadences and through empirical feedback. There is a commitment to quickly deliver reliable working software that has the highest value to the those using or benefiting from the software. A key principle to the long term success of a project is to have confidence that changes will not break important parts of the system. This is only done if there is time taken to build confidence into the process and architecture. To assist with this it is important to allow for "Slack Time" to reflect on things learned and to experiment with ways to make things better. This paper will focus on the "Slack Time" pattern as a key practice to help sustain Fast Delivery with Confidence.

Categories and Subject Descriptors

• Software and its engineering ~ Agile software development • Social and professional topics
• *Software and its engineering ~*

General Terms

Agile, Sustainable Delivery, Patterns,

Additional Keywords and Phrases

Agile Software Development, Innovation, Reliability,

ACM Reference Format:

Yoder, J.W., Washizaki, H., Aguiar A., 2018. Delivering Fast with Confidence "Slack Time". HILLSIDE Proc. of Asian Conf. on Pattern Lang. of Prog. 7 (March), 7 pages.

Author's email address: joe@refactory.com, washizaki@waseda.jp, ademar.aguiar@fe.up.pt

Introduction

Many software development processes such as agile and lean focus on the delivery of working software that meets the needs of the end users. Many of these development processes help teams respond to unpredictability through incremental, iterative work cadences and through empirical feedback. There is a commitment to more quickly deliver reliable working software that has the highest value to the those using or benefiting from the software.

Agile teams often focus on velocity and burndown as a way to measure success. It can become difficult to sustain delivery with a good velocity as the software grows and becomes more complex. Software projects can evolve quickly at first but as the system grows, what once seemed like an easy task can become much more difficult to implement and deploy with confidence, specifically when there are many dependencies within the system.

There are a set of practices that can be used during development to help sustain delivery at a good pace while maintaining confidence in the system. Some of these are related to agile or lean practices such as short delivery cycles, good testing, clean code, and continuous integration. Small delivery size with regular feedback through incremental releases has proven itself over the years and has become the de-facto standard for most agile practices. Delivery smaller pieces getting regular feedbacks so that adjustments can be made of problems noted earlier and fixed. Also keeping your code clean by including refactoring and evolving your architecture has become accepted as a key principle of most agile practices.

However blindly following agile practices isn't sufficient to help sustain delivering quality software at a good pace with confidence. Quite often system qualities, such as reliability, scalability, or performance, are overlooked or simplified until late in the development process, thus causing time delays due to extensive refactoring and rework of the software design required to correct quality flaws. There are other practices that can help with this such as reflection time, finding way to improve, and building quality into the process and product from the start [YWA, YW, YWW].

If we deploy software more frequently, it is important to insure stability and reliability in our systems. The goal is to be able to safely and quickly release our software in a sustainable way. Continuous Delivery and DevOps have focused on both automation and team practices that can help with this. Automation and a good pipeline that gives confidence has proven invaluable to help sustain delivery with confidence.

However to achieve long term stability and confidence, it is necessary to develop ways to minimize risk that changes will not break important parts of the system and if it does happen you know about it before they are released. This is only done if there is time taken to build confidence into the process and architecture. To assist with this it is important to allow for "Slack Time" to reflect on things learned and to experiment with ways to make things better. This paper will focus on the "Slack Time" pattern as a key practice to help sustain Fast Delivery with Confidence.

.

Slack Time

"The purpose of training is to tighten up the slack, toughen the body, and polish the spirit."
— Morihei Ueshiba

Successful software projects grow and become successful to the point where they receive so many requests that often any improvements are put on the back burner. Ultimately the system can evolve to the point where they can become very muddy and hard to maintain. A lot of time can be committed to trying to keep it working. Even small new features can cause things to break which can require a lot of work on putting out fires. The teams would like to innovate or find better ways to build the system, possibly even innovating.

.

People on teams become very busy on projects dealing with everyday tasks of keeping the system going and attempting to add new features. There seems to be no time to deal with problems, reduce waste or find ways to improve. How can teams improve or innovate given these issues?

❖ ❖ ❖

Businesses can get very busy trying to meet the requirements where there is barely time, resources or people to do what is needed to keep a project afloat.

Often there is the thought or culture that any time we are not "focusing on the project" we are not generating value for the business, thus wasting time. This can lead to people feeling guilty if we take time to think about other things or experiment with something that is not immediately apparent to fulfilling the goals of the project. In agile, aren't we only supposed to work on tasks or features that are prioritized by the product owner?

Many organizations don't provide the atmosphere or environment to be creative. This can be a barrier to think and experiment with new ideas.

On the average, average companies get average people. Some people only want to be told what to do and when to do it. They are not interested in improvements but only in what is the minimum they need to do to get the job done.

Some software qualities require significant expertise to make sure they are specified, designed, implemented, and verified appropriately. If the focus is primarily on functionality, qualities can be deemphasized and ignored until late in the development phase. Many qualities require various expertise and this expertise is not always part of the agile team. If not enough focus is emphasized on quality and part of the complete process, trying to add these later can often lead to an unstable architecture that is hard to evolve and maintain.

❖ ❖ ❖

Therefore, allocate "Slack Time" to experiment, build quality into the system, and to try new things. Create an environment to assist with supporting slack time for your teams.

Slack time is crucial to innovate or to make any critical long-term improvements. Of course this is easier to say than do. Often teams want and say they will allow more time in the next sprints or after the next release to do that needed refactoring or improvement, However there are many forces than often go up against teams finding slack time without making tricky trade-off decisions and often teams get stuck.

Successful innovators often take time to work on tasks that are less exciting than their great ideas. It's the mundane, less exciting jobs that really help make a project successful. Slack time may give you the opportunity to do those mundane, tasks that are still key to the success of a project. These can often be little experiments trying new ways to optimize the process or ways to build confidence about what is being work on.

If you primarily focused on just efficiency to get as many tasks done as possible, or there is a lot of complaining about the delay of the last release, you have no room to experiment. Efficiency can be very seductive as you believe you are finding the least expensive way to do something. Tom DeMarco in his book "Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency", points out that there is a danger in focusing too much on efficiency which can end up stifling the creative talent and reduce overall effectiveness [DeMarco]. You might end up micro-optimizing the ways you do tasks but could be missing on the creative way that could be much better. This is similar to what Daniel Kahneman talks about in his book "Thinking, Fast and Slow" [Kahneman]. There is value in both kinds of thinking and slack time definitely allows for some "Slow Thinking" that can be quickly applied.

Those who work towards steady improvement know and understand the importance of having time to think and experiment. It helps when you have time to reflect on experiences and think about experiments to make things better. Unfortunately many are not always capable of learning from our experiences. Often this is because we do not have time for allowing a new process to mature and for new ideas to germinate. Slack time provides an environment and free time that allows us to perform activities that had not been scheduled, thus allowing ideas to germinate and to experiment with new ideas. It can also be used to address important qualities that should be addressed to help with confidence for the system.

It can be useful to have slack spaces. This can help provide the environment and motivation to spend time in activities we enjoy and that we can share with our colleagues. This space can also help create the space or attitude for creativity.

Finding good slack time or when to have our slack spaces can be a challenge in itself. Some like to schedule it at the beginning of the day. Others like to schedule it at the end of the day, with a secondary goal of helping to make progress in tasks that have been put aside and finish the day in a more relaxed way.

Some think it a good option to plan what they are going to do in their slack time and space. For example, I know of one team that started their day with "Refactoring Exercises" doing a little cleanup before they got into the hectic part of the day. This is similar to stretch exercises before you run or morning meditation that can help with the rest of the day.

There are many ways that a team can create slack time. Create slack time through:

- Dojos
- Training
- Morning exercises
- Open space
- Hackathons
- Google 20% time
- Retrospective with time to reflect...but take action.

When done right, slack time can lead a positive consequence such as innovation, and building safety and confidence into the process and system. Also it gives room to try things that normally would never be attempted. Additionally slack time can be used to validate what you are doing or building, as well as find good ways to build quality into the the system and process.

However, there is also negative consequences associated with slack time. For example, in the short term it may appear you are not as productive. As time is spent to experiment or try different things, some may believe that time was wasted on not releasing important features. Also, some might take advantage of slack time. They might use it as time to goof off or to procrastinate. Possible it becomes a catch up time for email or browsing the internet. They might use it to put off needed tasks they just do not want to do. When this is done, you will not receive the benefits of slack time. It is important to not have too much slack.

Slack time can sometime be seen as being lazy or not getting important tasks done. However this can often be seen as something similar to what some of us call the “lazy programmer”. I often see lazy programmers as some the best programmers. Not lazy in terms of doing the right thing, rather a lazy programmer will avoid writing monotonous, repetitive code—thus avoiding redundancy which is the enemy of software maintenance and flexible refactoring. They often find a way to do the same thing that an average programmer can be done in a fraction of the time, thus allowing more time to validate what they are doing and experiment with better ways to solve the problem with confidence in the solution.

Summary

This paper outlines…We'll finish this later, although it may not require a summary. We are looking for feedback from our Writers' Workshop first.

Acknowledgements

I'd (Joe) like to personally acknowledge Harada Kiro for all of the slack time and great discussions we've had on this topic I had while visiting Japan and Shéhérazade Benzerga for helping me outline my ideas into pattern form while on the train during a recent visit to Germany. We'd also like to thank our shepherd Yung-Pin Cheng for his valuable comments and feedback during the Asian PLoP 2018 shepherding process. Finally we'd like to thank our 2018 Asian PLoP Writers Workshop Group, XXX, YYY, and ZZZ for their valuable comments and suggestions.

References

[DeMarco] DeMarco T., *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency.* New York: Broadway Books a division of Random House, 2002.

[Kahneman] Kahneman, D., *Thinking, Fast and Slow.* New York: Farrar, Straus and Giroux, 2013.

[YWA] Yoder J., Wirfs-Brock R., and Aguilar A., "QA to AQ: Patterns about transitioning from Quality Assurance to Agile Quality," 3rd Asian Conference on Patterns of Programming Languages (AsianPLoP), Tokyo, Japan, 2014.

[YW] Yoder J. and Wirfs-Brock R., "QA to AQ Part Two: Shifting from Quality Assurance to Agile Quality," 21st Conference on Patterns of Programming Language (PLoP 2014), Monticello, Illinois, USA, 2014.

[YWW] Yoder J., Wirfs-Brock R. and Washizaki H., "QA to AQ Part Three: Shifting from Quality Assurance to Agile Quality: Tearing Down the Walls," 10th Latin American Conference on Patterns of Programming Language (SugarLoafPLoP 2014), Ilha Bela, São Paulo, Brazil, 2014.