Value Stream Mapping

If you are already familiar with value stream mapping and want to skip this chapter, please jump to the end where I discuss “Where we spend our time”. This includes an important discussion of how what we’ve learned about value stream maps can provide useful insights into what to do.

Understanding the Value Stream

The value stream is essentially all of the work that takes place from the inception of an idea until it has been created and consumed by the customer. For software development, this is illustrated in Figure 1.

![Figure 1. A conceptual diagram of the value stream.](image)

This diagram illustrates the scope of software development and how much of this is often left out in Agile methods. In looking at the diagram we can see that it starts with the inception of an idea that will help the customer. It may be that the customer does not get the idea, but the intended value is focused on the customer. At some point the business discusses this and decides on a set of capabilities that will be built. The development team takes on the project, builds it and delivers it to the customer to be consumed. Management’s role, which is all too often ignored in the Agile space, has the important role of ensuring the work taking place in the value stream is not be impeded.

It is interesting to see where the popular Lean-Agile methods focus on in the value stream. XP concerns itself mostly with the development team and the customer. XP does not intend to provide insights into how projects are started. Rather XP is about building them after they’ve been designated as work to do. Scrum is only slightly different from this. Once a project is started Scrum focuses on the development team with only an indirect impact on the business side of things. This impact will only occur if the team faces an impediment caused by the business side.

Kanban mostly focuses itself on the development through deployment part. However, because Kanban makes its methods visible to business and because it enforces a limited size of input queue, it can have a
stronger influence on the business side than Scrum typically has. In fact, it was seeing this phenomenon – product managers changing their habits of interrupting teams due to the visibility that the teams gave the product managers – that accelerated my Kanban interest.

Lean, of course, covers the entire value stream map. The key aspect of a value stream map is that it creates visibility on the work that is taking place. While some people argue about the effectiveness of metrics in order to manage things, I believe there is no argument that you can’t manage anything if you can’t see it. Value stream maps are first and foremost about making our work visible.

Creating A Value Stream Map

Before starting a value stream map it is important to remember that it is a map about how one, particular, project is being worked on. In other words, we are mapping the flow of work for a particular project. We are not mapping the amount of time people are working on different projects.

There is not one way to create a value stream map. But having led hundreds of folks in creating value stream maps, I’ve seen this six step process work pretty well. The six steps are:

1. Identify the actions that take place
2. Specify the calendar time over which these actions are worked on
3. Specify how much of the time these actions were being worked on actual work was taking place and how much time was spent waiting
4. Specify the amount of time between these steps
5. Look, and denote, any loop backs present in the work flow
6. Total up the average time working on the project

The best way to understand this is, of course, with an example, so let’s go through one.

Step 1: Identifying the actions that take place

Let’s imagine we have a process where the actions that take place go through these steps:


Be clear I am just imagining this example, I am not suggesting this is a good workflow. We are just mapping the value stream – we are not assessing whether it is good or not. We can write down the six steps in boxes to represent the flow as shown in figure 2.1.

![Value Stream Map](image)

---

1 Note: When you map a value stream for real, I suggested that you use stickies on a white board. That way you can move the boxes around really easily.
Improving Collaboration and Planning With Flow Based Development

Step 2: specify the real time from start to finish of the action

After writing down the steps, you should consider how long does each of these take? In other words, from the start of requirements until their end, how many days does it take? There is no requirement to use hours for the times in a value stream map but it’s a lot easier if you consistently use them. I highly recommend using 8 hours a day and 40 hours a week. It makes day to week to month conversions pretty easy. If you change units of measure, that is sometimes use hours and sometimes use days, you’ll find yourself prone to making conversion errors. Figure 3 shows the value stream map after the hours each step has taken is shown.

Figure 3: Step 2: specify the real time from start to finish of the action

Step 3: specify how much of the total time was spent actually working on this

Now comes the tricky part. Lean suggests that we look for delays in our workflow. Sometimes it’s hard to see these delays because everybody is busy working. For example, if someone is working on two projects at the same time then in a sense, when they are working on one, it is delaying the other. This is easier when we have one person working, so let’s take that case first. If a person works 60 hours out of the 160 hours in the requirements step, we’d put down 60 / 100 to denote 60 hours working on this project and 100 hours not working on this project during this step.

What if we had 2 people working on requirements. Say one worked 50 hours and one worked 70 hours. What we want is the amount of time working compared to the amount of time not working. The first person worked on something else for 110 hours while the second person worked on something else for 90 hours. We could write 120 : 200 to denote the totals, but this would not make sense when we tried to compare that to the 120 hours of calendar time. What we want to know during this step is how much time did anyone work on this project and how much time did they do something else. That would be the average of these numbers. Therefore we’d write 60 : 100 as shown in Figure 4.

Figure 4: Step 3: specify how much of the total time was spent actually working on this
Improving Collaboration and Planning With Flow Based Development

Figure 4: Step 3: specify how much of the total time was spent actually working on this

Step 4: Specify the time between the actions
The rest is now pretty straightforward. One action does not start immediately after the prior action ends. We need to record that as well. It is shown in Figure 5.

![Diagram showing the time between actions](image)

Figure 5: Step 4: specify the time between the actions

Step 5: Identify any loop backs required
The last act of mapping is to look for any loop backs and how often they occur. This is shown in Figure 6.

![Diagram showing loop backs in the process](image)

Figure 6: Step 5: identify any loop backs required

Finally, we have to calculate the process cycle efficiency. This will be a measure of how much time was worked contrasted with how much time was available. The time available is actually the calendar time from start to end. This can be computed by summing up all of the boxes and the times between the boxes including the loop-backs. If this does not match your recollection of the real calendar time this took then your mapping is off a bit. The calendar time for the project should be the time grayed out in Figure 7.

Step 6a: Calculate the cycle time efficiency – calendar time
The actual time (calendar time) of the project should simply be the sum of the times we’re working plus the times between the steps including any times for loop backs. We show this with the shaded times in Figure 7.

Copyright ©, 2011 Net Objectives
Figure 7. Step 6a: calculate the calendar time of the project

The time worked without the loops backs would be:
\[0.5 + 320 + 8 + 80 + 160 + 320 + 8 + 80 + 100 + 80 + 120 + 160 + 2 + 80 + 280 + 80 + 240 + 80 + 8 = 2206.5\]

The loop back times would be computed as: \[1 \times 20\% \times (120 + 160 + 2)\] and \[3 \times 65\% \times (280 + 80 + 240) = 56.4 + 1170\].
Adding all these numbers up gives us 3433.

**Step 6b: Calculate the cycle time efficient – time worked**

Figure 8 shows us what numbers to use for the time actually worked. Note that we’re including the timed worked during loop backs. Technically, lean would say not to include these times as they represent re-work, but our point will be made by just discussing time worked contrasted with total time – I don’t need to have a conversation about whether work is valuable or not.

Figure 8: Step 6b: calculate the time that the project was being worked on

Doing this for the time worked gets us:
\[0.5 + 0.1 + 60 + 1 + 40 + 40 + 2 + 80 + 40 + 3 = 266.6\] and
\[1 \times 20\% \times (40 + 2)\] and \[3 \times 0.65 \times (80 + 40) = 8.4 \text{ and } 234\].

Add these together and you get 509 hours.

The process cycle efficiency is therefore 509 hours / 3433 or 14.9%

This doesn’t mean that people are working only about 1 our of 7 hours. Rather, it means that they are working only about 1 out of 7 hours on this project.
Is this a typical number? That somewhat depends on your process. When I first started doing value stream mapping about 7 years ago, most of the companies I worked with did not do any sort of Agile method. In these cases, anywhere from 5 to 20% were common. Maintenance projects with 6 month release cycles could be as low as .1% (2 hours work waiting for a 1000 hours release time would be .2%). When teams do some sort of Agile this percentage rises to between 50-70%. Mostly this is due to people working on one project at a time with some (smaller) level of interruption.

Where We Spend Our Time
The value stream shows us that we spend most of our time waiting for someone or something else. This should be no surprise. Imagine how many times you send an email off and then go off and do something else until you get a response? Most of our time in non-co-located teams is spent waiting to get information from somewhere else. If your organization is siloed people are likely working on multiple projects. When one considers how delays literally create waste (remember Why Looking At Time Is So Important) all of this delay may be causing more extra work than we might first think.

Consider the following two questions. Let’s say this project we just mapped had been for some very important function. Imagine going to the business stakeholder who was promoting it and asked them would they be willing to increase the cost of the project by 20% if they could get the time down from 1.75 years (the 3433 hours) to one year (about 2000 hours). When I ask this, most people say “almost certainly.” The time of earlier delivery is well worth more than a 20% increase in cost if the project is important. But now let me ask another question. What would happen if the time of the project were reduced to a year from almost 2 years? Clearly the only way to do this would be to eliminate the delays between the steps and to reduce the amount of distracting work while working on this project. But if we lower delays, how will that affect our work to be done? It’ll lower it as well (since raising delays causes additional work, if we eliminate these delays this additional work will not be created). In any event, it certainly shouldn’t go up.

This is the “magic” of lean flow. By focusing on the most important value to be built, we improve both time to market and we do it with less work.

Which Gives a Better Return?
Analyzing the value stream gives us the insight that we need to focus on delays more in removing the delays within our workflow than in improving the steps themselves. Of course, this doesn’t mean ignore making the steps better. Ironically, when we start looking at delays, many new methods (ATDD, TDD, design patterns) start proving to be good ways to eliminate these delays.