Abstract

Little’s Law deals with averages. It can help us calculate the average waiting time of an item in the system or the average Lead Time for a work item. In project development we break the project delivery into a batch of work items. We are interested in how much time it will take for the batch to be processed by the system. But we need a relationship between the average Lead Time for a work item and the finite time period over which the project will be delivered. A formula is presented that gives us the relationship between the average lead time for an item and the finite time period over which the project will be delivered.

This paper is about planning a project by applying Little’s Law and statistical data from kanban systems. Also covered is how to manage the inherent uncertainty associated with planning and executing a project.

Introduction

Clients come to us with an idea for a product and they always ask the questions - how long will it take and how much will it cost us to deliver? They need a delivery date and a budget estimate.

What is a project?

In a kanban system a project is a batch of work items each one representing independent customer value that must be delivered at or before a certain date.

Why project definition speaks about “work items each delivering independent customer value”? Because using a kanban system we have statistical data for work items each delivering independent customer value.

A project is just a batch taken out from the flow of work. Say the flow processes thousands of work items. We can name a batch of some 100 work items “a project” but we have statistical data for the flow itself and we use that data!

Little’s law

Let’s recall what the Little’s law for production systems is:
Where \( \overline{TH} \) is the average output of the production process per unit time, \( \overline{WIP} \) is the average inventory between the start and end points of the production process per the same unit of time, and \( \overline{LT} \) is the average time from starting a job at the beginning of the production process until it reaches the end of the production process (that is, the time the job spends as WIP) in the same unit of time. Note that lead time is also referred to as flow time, throughput time, and sojourn time, depending on the context.

Notice that our formula is exact, but after the fact. This is not a complaint. It merely says that we are observing and measuring not forecasting.

Little’s Law applies to any system, and most importantly, it applies to systems within systems. So in a bank, the queue in front of the tellers might be one subsystem, and each of the tellers another subsystem, and Little’s Law could be applied to each one, as well as the whole thing.

**Little’s Law holds in case of a project**

Now, we ask what conditions are necessary for Little’s Law to hold:

- At a minimum we must have conservation of flow. Thus, the average output or departure rate \( \overline{TH} \) equals the average input or arrival rate \( \lambda \). Furthermore, we need to assume that all work items that enter the system will eventually be completed and will exit the system; there are no work items that get lost or never depart from the system.
- Little’s Law holds exactly between any two time instances at which the system is empty.

For a project both conditions are true hence Little’s Law holds in case of a project.

**Project in a kanban system**

Let’s see how a project is handled in a kanban system. Fig. 1 shows a kanban board that consists of the columns:

- Project backlog
- Input Queue
- Three stages of development process
- Deployed column

We have kanban system within a kanban system or a two tiered board. First kanban system is the Project. Inside it we have the Development system. Project system’s columns Project Backlog and Deployed are highlighted in blue. Development system’s columns Input Queue, Development, Test, QA, Deployed are highlighted in green.

It is visible that Deployed column belongs to both Project and Development systems and is in fact shared by them. That is why Deployed column is highlighted in blue and green.
Little’s Law holds only for Dev system. That is what we measure and have historical data about. Hence it is safe to discard a work item from the Project system and break conservation of flow requirement on the project system level.

Because the Project system and Development system share the same output namely Deployed then the average throughput of the project system equals the average throughput of the development system.

**Kanban system as a Queuing system**

On Fig. 2 the kanban system from Fig. 1 is presented as a queuing system comprised of queuing system within a queuing system:

- Project System – it processes the project scope or the batch of N work items each one representing independent customer value.
- Dev System – it processes work items suitable for the development organization. User stories are just an example of a suitable work item type.

What we have done is map:
- Project Backlog to Project Queue
- Input Queue to Dev Queue
- Deployed column to Delivered work

Project and development systems are connected in a way that they share the same output. Output from the Development system is flowing into the output of the Project system. Hence the average throughput of the Project system is equal to the average throughput of the Development system or $\bar{TH}_p = \bar{TH}_f$

The project will be delivered in the time T. We break down the project into N work items. All the items enter Project Queue. The size of the individual work items doesn’t matter. When breaking a project down we do not consider how many steps the workflow of server S2 consists of. Important thing is that we have two queues - Project and Dev. Project queue has the size of N and Dev queue could be of any size in the range [0,N]. Server S1 could have service time of more than 0 if for instance what it does is to decide which would be the next item to leave Project queue and enter Dev queue.
Where:

- $T$ is the time period over which the project will be delivered (lead time).
- $N$ is the number of items to be delivered or the total arrivals in $[0, T]$.
- $\overline{LT}_p$ is the average lead time at the project level or what the client can see.
- $\overline{WIP}_p$ is the average work in process at the project level.
- $\overline{WIP}_t$ is the average work in process at the development organization level.
- $\overline{LT}_t$ is the average lead time (cycle time) at the development organization level.
- $\overline{TH}_p$ is the average throughput at the project level.
- $\overline{TH}_t$ is the average throughput at the development organization level.
- $\overline{TT}_p$ is the average Takt Time at the project level.
- $\overline{TT}_t$ is the average Takt Time at the development organization level.

It is visible that $\overline{TH}_p$ of the Project System comes from the Dev System hence $\overline{TH}_p$ cannot be bigger than $\overline{TH}_t$ and at the same time cannot be smaller. Hence $\overline{TH}_p$ and $\overline{TH}_t$ are equal!

**How we measure Lead Time?**

For the Project System we measure Customer Lead Time defined as the time interval between Request Accepted and Delivery to the Customer.

For the Dev System we don’t measure the time between Commitment point and first Infinite Queue. We measure Lead time defined as the time interval between Commitment point and Delivery to the Customer.
And that is for a reason because Project System and Dev System share the same output which is Delivery to the Customer!

**Why we need a two tiered kanban board?**

One may think about using kanban board with only one Queue and moving all work items from Project Backlog into Dev queue and hence having \( \overline{WTP}_t = WTP_t + N \).

That will be incorrect because the average throughput \( \overline{TH}_t \) of Dev system was achieved using a certain WIP limit – say 5 work items of which 2 in Dev Queue.

We don’t know what average \( \overline{TH}_t \) of Dev System will be if we increase \( \overline{WTP}_t \) e.g. making the WIP limit for the Dev Queue to be \( N \) work items.

We have the Dev System in place and we try to accommodate the Project System to current Dev System capabilities and not the other way around!

**Anderson’s Formula**

We can calculate the lead time for the project using the formula:

\[
T = N \frac{\overline{LT}_t}{\overline{WIP}_t} = \frac{N}{\overline{TH}_t} = N \overline{TT}_t \quad (1)
\]

We can calculate the number of developers we will need to deliver for a particular lead time using the formula:

\[
\overline{WIP}_t = \overline{LT}_t \frac{N}{T} \quad (2)
\]
Note: Of course $\overline{TH}_t$ is the average of a random variable modeled by a distribution of some type. If the departure is a Poisson process with intensity $\overline{TH}_t$, then the required time for N-th work item to exit the system (be delivered) is modeled by a Gamma distribution Gamma(a,b) with a=N and b=1/$\overline{TH}_t$.

**Proof of Anderson’s formula**

We apply Little’s law at the client level:

$$\overline{TH}_p = \frac{\overline{WIP}_p}{\overline{LT}_p}$$

Then we apply Little’s law at the development organization level:

$$\overline{TH}_t = \frac{\overline{WIP}_t}{\overline{LT}_t}$$

Consider Fig.2 where the project is presented as a queuing system consisting of two queuing systems in tandem. Client and development systems are connected in a way that they share the same output. Output from the development system is flowing into the output of the client system. Hence average throughput of the client system is equal to the average throughput of the development system or $\overline{TH}_p = \overline{TH}_t$. For both systems we have to work with the same time interval for the throughput time and throughput rate i.e. minutes, hours, days, weeks, etc. Also we have to have the same work item type – it is not possible average throughput at the project level to be measured in sites, and average throughput at the development organization level to be measured in web pages.

By noting that

$$\overline{TH}_p = \overline{TH}_t$$

$$\frac{\overline{WIP}_p}{\overline{LT}_p} = \frac{\overline{WIP}_t}{\overline{LT}_t}$$

$$\overline{TH}_p = \frac{N}{T}$$

It follows that

$$T = \frac{N}{\overline{TH}_p} = \frac{N}{\overline{TH}_t} = \frac{N}{\overline{WIP}_t} = \frac{N\overline{LT}_t}{\overline{WIP}_t}$$

As well as that

$$\overline{WIP}_t = \overline{LT}_t \overline{TH}_t = \overline{LT}_t \overline{TH}_p = \overline{LT}_t \frac{N}{T}$$

or

$$\overline{WIP}_t = \frac{\overline{LT}_t}{\overline{LT}_p} \overline{WIP}_p$$

**Examples**

*Original example from David Anderson*
We have a major project with 2200 user stories to be delivered. The business needs the project delivered in 10 months. The questions to answer are: How much money do we need? How many people do we need?

So we have \( N = 2200 \) user stories, \( T = 10 \) months = 40 weeks. We also have historical data that average lead time for the development organization is 0.4 weeks. Using the Anderson’s formula (2) we have:

\[
\bar{WIP}_t = \bar{LT}_t \frac{N}{T} = 0.4 \frac{2200}{40} = 22 \text{ user stories}
\]

\( \bar{WIP}_t \) is the sum of the number of developers times the allowed number of work items per developer plus the size of all the queues.

That formula is to be used very carefully because there is no linear correlation between \( \bar{WIP}_t \) and \( \bar{LT}_t \).

Development organization has the policy of 1 work item per person hence we will require 22 persons. That assumes the dev queue has finite size of 0 work items. If the dev queue has finite size of 2 user stories we will need 20 persons.

The result should be used for the second leg of the Z-curve, but that is another topic.

**Calculating delivery date example**

Let’s have a similar project with major project with 2200 user stories to be delivered. The business wants to know when we will be able to deliver. There are no options for adding more developers to the development organization so we know the average work in process which is 22 user stories. We also have historical data that average lead time for the development organization is 0.4 weeks per user story. Using the Anderson’s formula (1) we have:

\[
T = N \frac{\bar{LT}_t}{\bar{WIP}_t} = 2200 \frac{0.4}{22} = 40 \text{ weeks}
\]

The result should be used for the second leg of the Z-curve.

**Managing uncertainty when planning a project**

**Sources of uncertainty**

Uncertainty is reduced by aggregation. When calculating Project lead time we need to account for the uncertainty in terms of:

- Account for the first and third legs of the Z-curve
- Account for Dark Matter
- Account for Failure load

In order to manage the high uncertainty and risks in the projects we use a safety time interval to protect the project due date promised to the customer from variation in Throughput. This improves the reliability of the overall due date as well. Such a buffer is known as Project buffer. Its size could vary from as little as 15% to as much as 200% of the initially calculated duration of the second leg of the Z-curve (T).
The promised delivery date of the project is then the sum of the initially calculated duration of the second leg of the Z-curve (T) plus the Project Buffer.

Calculated T should be used only for the second leg of the Z-curve

Will we have the average TH on the first week of the project? No! On the second week? May be! If we know we’ll going to go slow at the beginning let’s add that into the plan! Also most of the large projects tend to go slow towards the end. That’s because may be we have regression effects in our code, we have testing overhead and bug fixing. So let’s add that into the plan too!

**Cumulative Flow Diagram**

It turns out that the delivery rate (TH) follows an S-curve pattern as shown on Fig.3 as the blue line. The red line is the Z-curve. Slow in the beginning and in the end and fast in the middle.
There is a little bit of empirical evidence shown on Fig. 4 that for approximate 20% of the time the delivery rate will be slow. Then for approximate 60% of the time we’ll going to go fast or “the hyper productivity” period and for approximate 20% till the end we’ll go slow.

Numbers may vary depending on the context but the basic principle about the three sections is correct.

**Dark matter**

The number of work items will grow through natural expansion as we investigate them. It is possible some features will be broken into 2 or more once you get started on them. That is why we add work items to compensate for the unknown work or dark matter.

My experience is that I would buffer at least 20% for this and with novice teams on a new product (especially if it was a highly innovative new product where we had no prior knowledge or experience) then I might go as high as 100%.

**Failure load**

We add work items to compensate for the expected failure load in terms of defects, rework, and technical debt.

Failure load tracks how many work items the Kanban system processes due to poor quality. That includes production defects (bugs) in software and new features requested by the users because of a poor usability or a failure to anticipate properly user needs.

Defects represent opportunity cost and affect the lead time and throughput of the Kanban system. The count of defects is a good indicator if the organization is improving or not.

**Why don't we use calculated average TH for all three Z-curve legs?**

Only the second Z-curve leg is representative for the Dev System capability. It shows the common cause variation specific to each Dev System. First and third Z-curve legs are project specific and show special cause variations.
We protect the project delivery date from special cause variation using a project buffer.

**Project Buffer sizing**
The size of the project buffer must be based on the uncertainty involved in the project. The uncertainty must be accessed in terms of technology, people, subject matter, architecture, maturity of the organization and suppliers.

\[ PB = ZC \frac{(1 + FL + DM)N}{TH} \]

Where:

- \( PB \) is the project buffer in units of time used
- \( TH \) is the average throughput for the second leg of the Z-curve
- \( ZC \) is the coefficient for accounting for the first and third legs of the Z-curve. Usually is is between 25-50% i.e. 0.25-0.5 as the quotient.
- \( DM \) is the average percentage of dark matter calculated by appropriate dark matter quotient (based on historical observation). If we don't have history we use 40-60% expansion i.e. 0.4-0.6 as the quotient.
- \( FL \) is the average percentage of Failure load.

The value of \((FL+DM)*N\) is in number of work items.

Project buffer should be sized aggressively so that at the end of the project the consumption is more than 80%. An overestimated buffer size could lead to incompeting proposals and consequently loss of business.

**Project Buffer Management**
In order to provide focus and be proactive during project progress, the buffer is monitored to ensure that project due date is protected. This mechanism is called Buffer Management and is the key to managing and tracking of project performance. As project execution proceeds, if a work item takes longer than estimated, it consumes the project buffer.

The buffer helps management to act proactively. Buffer Management identifies potential problems much earlier than they would ordinarily be detected using traditional project management techniques. Building explicit buffering into the model would also help communicate the degree of anticipated risk in the work and demonstrate that we are mitigating against it from day one.

Upon delivering a work item Buffer Management compares the consumption rate of the buffer to the project progress rate. The two essential measurements of project performance in buffer management are the percentage of the project completed and the amount of the project buffer consumed. This relationship is the signal to management for appropriate action. Project review meetings focus on whether the completion of the project is at a pace for completion without consuming the project buffer.

**Focus on the median lead time**
When using kanban systems, the role of the project manager shifts from a focus on all tasks to monitoring Average Lead time \( \bar{LT} \) because if WIP is limited then Throughput depends only on Lead Time.
Human beings do not understand the average – we have to calculate it. What people understand is the median – every other one. When we say “On average each work item should take 2 days” what people subliminally hear is “every other work items should take 2 days or less and the odd ones should be more that 2 days”. And that is the median and not the mean. And since Lead Time is not normally but Weibull distributed the median and the mean are not the same. The median $M$ is always smaller than the mean. Every team member will understand by the message given to the team – every other ticket should take $M$ days or less. The odd ones can take longer than $M$ but if they take longer than two times the $M$ alert your manager. Because we don’t want the tail on the Lead Time distribution getting really long which will mean our delivery rate is going down and we are not hitting the target!

**Project buffer usage**

Project buffer usage determines whether or not the delivery date is affected. Buffer usage should be monitored as a percentage used.

We are visualizing the cumulative buffer usage just like we would visualize a bank account overdraft usage.

\[
P_C(t) = \frac{\sum_{i=0}^{t} N_i}{N} \times 100\%
\]

\[
PBU(t) = \frac{(ALT(t) - PLT(t))}{PB} \times 100\%
\]

Where

\[
ALT(t) = T_t - T_0
\]

\[
PLT(t) = \frac{LT}{WIP} \sum_{i=0}^{t} N_i
\]

$T_0$ is the project start date
\( T_t \) is the date at time \( t \)

\( N_i \) is the number of completed work items at time \( i \)

\( \overline{TT} \) is the planned average Takt Time for the project

\( PC_t \) is the percent delivered out of total work items to be delivered at time \( t \)

\( PBU_t \) is the percent of the Project Buffer consumed at time \( t \)

\( ALT_t \) is the Actual Lead Time at time \( t \)

\( PLT_t \) is the Planned Lead Time at time \( t \)

As long as there is some predetermined proportion of buffer remaining, the project progress is assumed to go well. If the project execution consumes a buffer by a certain amount, a warning signal is raised. If the buffer consumption rate is high so that whole of the buffer are expected to be consumed before completing of the project, corrective action should be taken.

**References**

