


# Dark matter, failure demand and S-Curve – project planning with Little’s Law and adding a project buffer to anticipate risks

 [ontheagilepath.net/2015/08/dark-matter-failure-demand-and-s-curve-project-planning-with-littles-law-and-adding-a-project-buffer-to-anticipate-risks.html](https://ontheagilepath.net/2015/08/dark-matter-failure-demand-and-s-curve-project-planning-with-littles-law-and-adding-a-project-buffer-to-anticipate-risks.html)

By Sebastian Radics

Based on my previous post where I explained how you can apply Little’s Law for Release Planning this post describes how to add a necessary buffer to consider that project delivery rates follow a fairly predictable S-Curve and that Little’s law can just be applied to the middle section of this curve (if at all).

The following description is based on the book Scrumban [R]Evolution – Getting the most out of Agile, Scrum and Lean, Kanban by Ajay Reddy. (I highly recommend reading this book to get a wide range overview about Scrumban!)

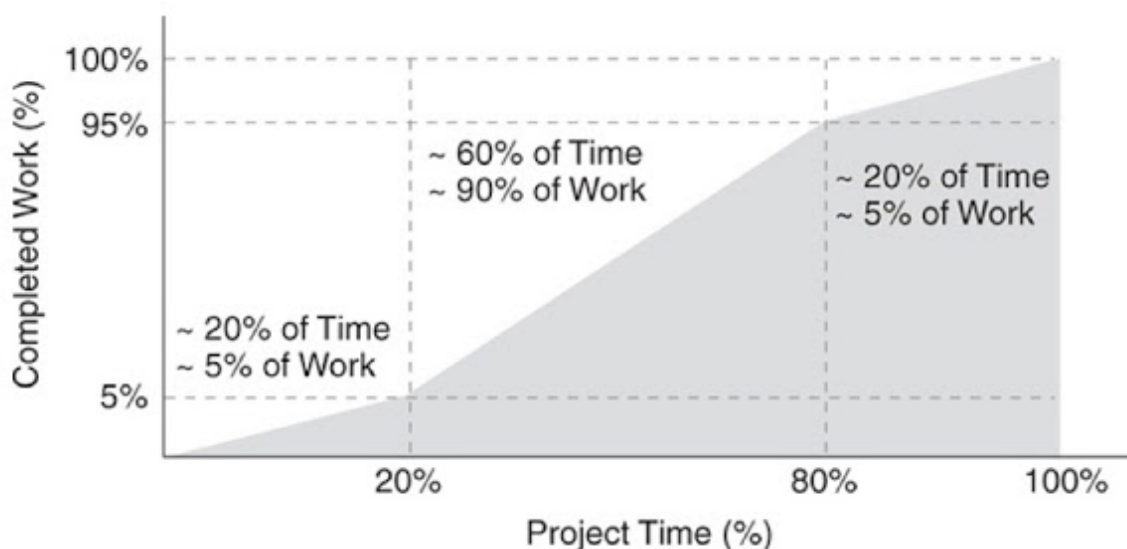
As a short recap from the previous post lets shortly have a look on the basic formulas explained:

Average WIP (The amount of developers,resources,... needed) = Average Lead Time  
number of user stories Total project time

(Project/Release) lead time (Duration) = number of user stories Average Throughput

Duration = number of stories average lead time average WIP

Now lets consider an example S-Curve for project delivery rates:



Source: Scrumban [R]Evolution – Getting the most out of Agile, Scrum and Lean, Kanban by Ajay Reddy.

It shows that work flows more slowly during the early stages of new projects as understandings are refined and basic infrastructure set up takes place.

It also flows more slowly toward the end of project cycles because of more testing overhead, bug fixes and regression effects (although we try to minimize this effect by test driven approaches, zero bug policies and using agile approaches with iterative and incremental deliveries).

The described formulas can only be applied to the middle portion of the S-Curve for most projects (in the example around 60% of the projects time).

In addition project duration is influenced by expansion of work (as we learn more about the true customer requirements during the project and incremental delivery) and failure demand (like for instance ensuring high quality by removing technical debt).

We need a way to include the mentioned project influences (40% of the S-Curve, expansion of work and failure demand). This can be done by calculating a project buffer like described in [Project Planning Using Little's Law](#) by [dimitar.bakardzhiev](#).

Project Buffer = S-Curve coefficient x 1 + average failure demand + average expansion of work x number of stories average throughput  
All variables remain best estimates based on historical observation.

The **S-Curve coefficient** is expressed as percentage and accounts for the uncertainty associated with the slower first and third legs of the S-Curve. Values vary depending on the system but are usually between 25-50%.

The **average failure demand**, expressed as percentage, accounts for the typical rate of failures (defects, rework, technical debt and others) the system experiences across development efforts.

The **average dark matter**, expressed as percentage, accounts for the typical rate of work expansion (e.g through knowledge discovery). It can range from 20% to as much as 100% for novice teams.

Lets assume a failure demand of 20% of your time (measured during your project) and your expansion of work is around 80%. In addition we assume an S-Curve coefficient of 40% (0,4).

Our average lead time is 0,9 weeks per story and average WIP is 24 (meaning the average throughput is 26 stories per week)

$$\text{Project Buffer} = 0.4 \times 1 + 0.2 + 0.8 \times 675 / 26 \approx 21 \text{ weeks}$$

## How to use the buffer

---

As items are delivered you can compare the consumption rate of the buffer against the rate of progress of the project as a whole.

Use the buffer consumption over a defined threshold as a warning signal for your project. Calculating the *percentage of your project completed* in addition to the buffer consumption enables tracking your buffer consumption over time.

Percentage of project completed = number of stories delivered / number of total stories in your project

Lets assume we have after 14 weeks in a project you have delivered 85 out of 675 stories.

$$\text{Percentage of project completed} = \frac{85}{675} \approx 12,6 \%$$

Projected lead time = average lead time average WIP x number of stories completed  
With an average lead time of 0.9 weeks per item and WIP of 24 (see values from previous post) this results in:

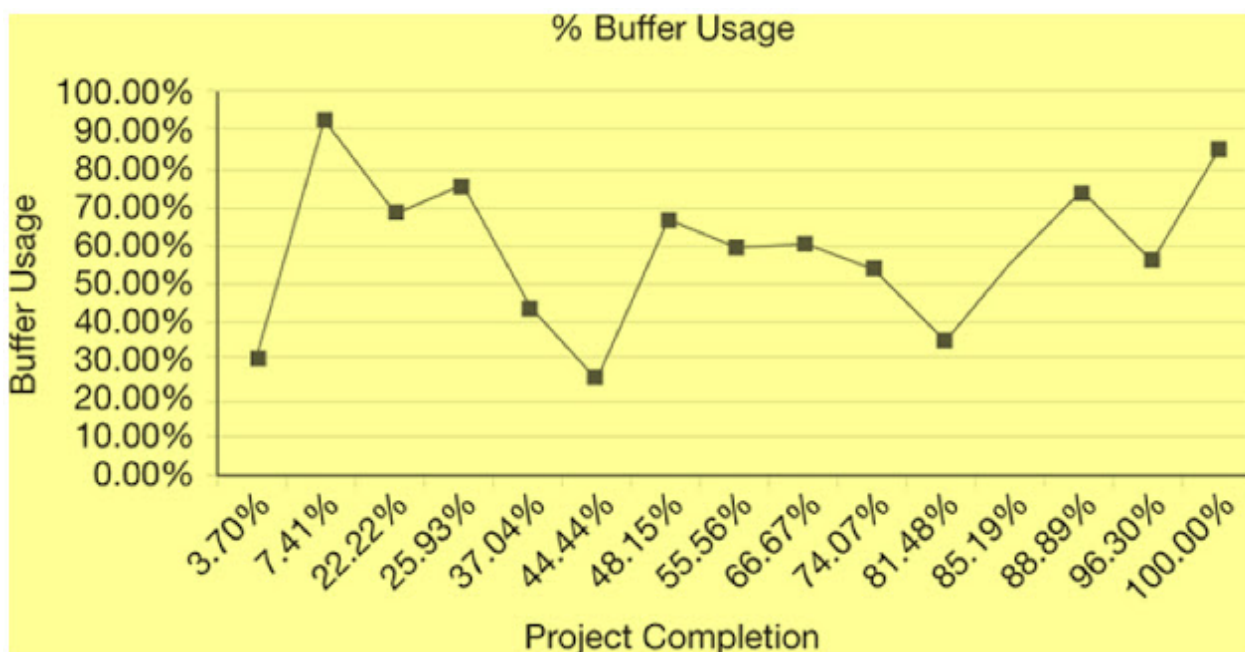
$$\text{Projected lead time} = 0.9 \times 24 \times 85 \approx 3.2 \text{ weeks}$$

Now we can calculate the percentage of buffer consumed after 14 weeks:

Percentage of buffer consumed = actual lead time – projected lead time / project buffer  
Assuming a buffer of 21 weeks we calculate the buffer consumption:

$$\text{Percentage of buffer consumed} = \frac{14 - 3.2}{21} = 51 \%$$

Using the percentage of project completed and the calculated buffer consumption you can plot this information over time and visualize your buffer usage.



Source: [Scrumban \[R\]Evolution – Getting the most out of Agile, Scrum and Lean, Kanban by Ajay Reddy](#)

## Disclaimer

While reading through various sources and considering a great comment by [Gerrit Beine](#) I'm still struggling if its right to apply Little's Law in software projects or not at all. Having all the limitations in mind (please read e.g. my [summary about Actionable Agile Metrics for predictability](#)):

- The average Arrival Rate (Lambda) should equal the average Departure Rate (TH)  
= we will only start new work at about the same rate that we finish old work

- Needs a more late binding (commitment) approach.
- Monitor policies around the order in which we pull items through the system – so that work items do not sit and age unnecessarily
- all work started will be completed and exit the system
- WIP should be roughly the same in the chosen interval
- average WIP is neither increasing nor decreasing
- CT,WIP,TH are measured using consistent units

I still need to figure out whether it's correct or not. For the moment I think it's another indicator (e.g. in addition to the [Release burndown chart](#)) to foster discussions but should be used carefully.

Please share your opinions – I would like to learn more deeply what to consider and if it makes sense at all? I'll keep you updated with more results of my journey.

### **Further readings**

---

[Project Planning Using Little's Law](#) by [dimitar.bakardzhiev](#)

[Scrumban \[R\]Evolution – Getting the most out of Agile, Scrum and Lean, Kanban](#) by [Ajay Reddy](#).

[Whats wrong with Little's Law predictability.](#)

[Actionable Agile Metrics for predictability.](#)

[Release burndown chart](#)

[Little's Law for Release Planning](#)