Creating and Interpreting Cumulative Flow Diagrams

Cumulative Flow Diagrams (CFDs) are valuable tools for tracking and forecasting agile projects. Today we will look at creating CFDs and using them to gain insights into project issues, cycle times, and likely completion dates.

In Microsoft Excel a CFD can be created using the "Area Graph" option. The attached file "Example CFD.XLS" contains the data used to create the CFDs in this article, including the one shown below.

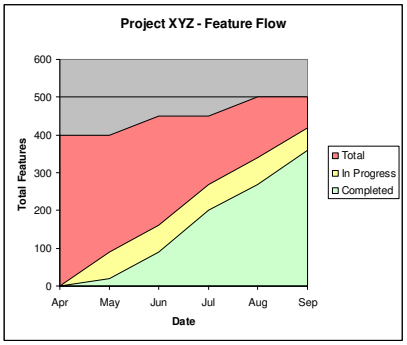

Figure 1 – Sample Cumulative Flow Diagram

Interpreting CFDs

Figure 1 shows the features completed verses the features remaining for a fictional project that is still in progress. The red area represents all the planned features to be built, this number has risen from 400 to 450 in June and then 500 in August as additional features were added to the project. The yellow area plots the work in progress, and the green area shows the total number of features completed.

Little's Law

In Donald Reinertsen's "Managing the Design Factory" published in 1997, Little's Law is introduced as a way to analyze queues from CFDs,

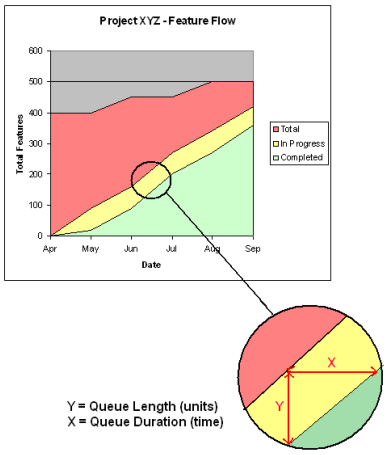

Figure 3 – Examining Work in progress

The yellow area representing work in progress (WIP) is our queue, it is the work started that has not completed yet. We can determine how many items are in the queue by looking at the vertical (Y) distance and we can determine how long they will likely take to complete by examining the horizontal (X) distance.

You may recall from my post about Mary Poppendieck's presentation on Lean Development, that cycle times are important metrics for creating lean production systems. Measuring the horizontal X distance on a CFD gives you your cycle time, which is a prediction of when all the work currently in progress will be done. We should aim to keep WIP and cycle times as low as possible as it represents sunk investment costs with no business benefits yet and, if a problem is encountered, the amount of potential scrap we have in the system.

To recap, Mary reminds us that we should be monitoring cycle times, but Little tells us that cycle times and queue length are proportional, so if monitoring cycle times is too onerous, we can just keep an eye on queue lengths, which is an easier metric to monitor.

Looking for Bottlenecks and the Theory Of Constraints

Eli Goldratt introduced the Theory of Constrains (TOC) as a tool for optimizing a production system. He observed that "*Changes to most of the variables in an organization usually have only small impacts on global performance. There are few variables (perhaps only one) for which a significant change in local performance will effect a significant change in global performance.*" So create the greatest benefits we should find these constraints (bottlenecks in the system) and focus on improving these issues.

Question 3 from the Daily Stand-up meeting (Scrum meeting) that asks for any impediments (blockers) to making progress is an adoption of TOC thinking. We are looking for roadblocks and then removing them.

Constraints also manifest themselves as throughput capacity restrictions. Perhaps our database folks can not keep up with the changes coming from the development team, or perhaps the customer proxy can not keep up with questions around validation rules for a new screen we are designing. In both cases the database group and the customer proxy are constraints, but as a project manager, these bottlenecks are not always easy to spot. We usually have a feeling where the bottlenecks are, but short of standing over people with a stopwatch and counting how many features that can process a day, how can activity throughput and bottlenecks be objectively measured? The answer of course is by using CFDs.

If instead of lumping all the work in progress as a single measure, we break it out by activity and plot the flow of this work, we can find bottlenecks with CFDs.

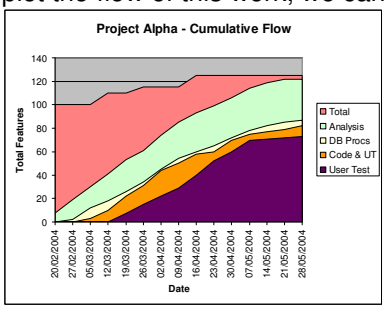

Figure 3 – Detailed CFD

In Figure 3 the work in progress has been broken out by activity. We have "Analysis", "DB Procs", "Code and Unit Test", and "User Testing" work being done on the project by different groups. When examining CFDs for bottlenecks we are looking for widening areas before the final activity ("User Test" which we want to see widening to show the growing completion of work). Here's the math part, a widening area is created when one line is followed by another line of shallower gradient. Since the line gradient indicates the rate of progress for an activity (features over time) a widening area is created above an activity that is progressing at a slower rate.

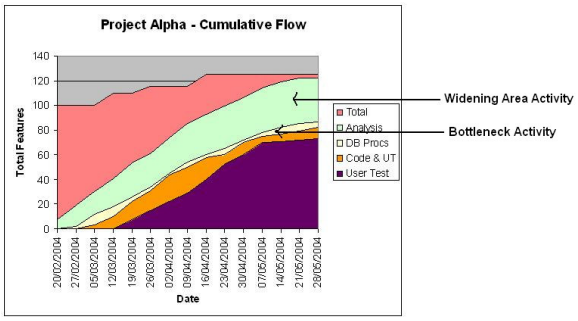

Figure 4 – Detailed CFD with Bottleneck Identified

If that made sense or not, all you need to remember is that the bottleneck is the activity **below** the widening band. The widening band is the feeding activity not the problem activity, in our example "Analysis" is going OK, but "DB Procs" (creating the database stored procedures) is below the widening band, indicating a slower rate of progress, and a warning to us that it is our bottleneck in the system.

So, we do not need to micro-manage team throughput, by using CFDs to track progress we can identify bottlenecks unobtrusively. Then once we know where the problem is we can start addressing the issue by applying the TOC actions items from step 2 onwards:

1. Identify the constraint
2. Exploit the constraint
2. Subordinate all other processes to exploit the constraint
3. If after #2 and #3 more capacity is needed to meet demand, Elevate the constraint.
4. Go back to #1, but don't let inertia (complacency) become the system's constraint.

Agile methods inherently reduce the likelihood of activity based bottlenecks by promoting multidisciplined teams. By avoiding role specialization people are able to move between roles more effectively and share the workload. However, while fully multidisciplined teams are the goal, in practice some role specialization and workflow management is the norm on projects. Hopefully this overview introduces the creation and interpretation of Cumulative Flow Diagrams. They are a valuable tool for agile projects that are simple to create, but extremely powerful.

Mike Griffiths is an independent consultant specializing in effective project management. Mike was involved in the creation of DSDM in 1994 and has been using agile methods (Scrum, FDD, XP, DSDM) for the last 13 years. He serves on the board of the Agile Alliance and the Agile Project Leadership Network (APLN). He maintains a leadership and agile project management blog at www.LeadingAnswers.com.

