Deadlines are elastic by value

> "Nothing in life is to be feared, it is only to be understood. Now is the time to understand more, so that we may fear less." Marie Curie

The question, the ultimate question, the tyrannical question:

> When will it be ready?

I'm sure every reader is familiar with this enquiry. But actually, it is the wrong question, a better question is:

> When do you need it by?

When you think about it "when" is just another parameter that accompanies "what." The available time is one of the constrains software engineers work under. Since there are many potentially different solutions to the "what" questions "when" is simply one of the constraints to selecting, and designing, a solution.

Rather than frame the discussion in terms of "How long will it take?" the discussion can be framed in terms of: "What can be built within the available time?" Not only does this side-step the question of estimates, and the accuracy of estimates, but it allows for a fuller discussion of value and options.

Once you start thinking along these lines it becomes clear that delivery on different dates results in different values. Most of the time delivering something sooner generates more value, not always but usually. However delivering something sooner may mean delivering something with less functionality and therefore something that delivers less value.

The original question changes once again:

> What is the difference in value from different delivery dates?

The discussion now focused on whether less product delivered sooner captures more value then more product delivered later.

There is nothing wrong with deadlines but deadlines should come from business need, not effort estimates which are magically transformed into deadlines. When deadlines are business driven they come from the need to capture value and the value which gets captured varies over time.

Different "end" dates result in different values. Understanding the trade-offs allow engineers and non-engineers to consider options and design solutions.

There is no single "right" design or approach, many are possible, and knowing the constraints allows one to choose appropriately.

Worked example

Imagine you run an software development team for a toy business. Two internal customers come to you and ask for similar but different products to be developed. Both customers, Alex and Si, present you with a story and an estimated value:

> Alex requests: As a toy retailer I want an app that allows kids to select from my special range so that their parents can buy. Estimated value: $355,000.
>
> Si requests: As a toy retailer I want an app that allows kids to make lists of toys they want so their parents can buy. Estimated value: $1,060,000.

Although these two stories look similar lets assume there is very little overlap and therefore no opportunity to combine any of the work. (Issues around intellectual property rights can prevent outsourced development teams from combining very similar client requirements from different clients.)

Following frequently recommended Agile practice Si's request would be prioritised above Alex's and work would begin because it has a higher value. Now supposes Alex complains:

> "But my request is smaller, my story has a better cost-benefit payoff, it gives a bigger bang for the buck."

So your team take some time and analyse the work required for both apps. Leave aside for the moment that undertaking analysis would itself take time and would delay the start of work, leave aside too the question of whether such estimates are accurate. Your team come back and say:

> Alex's request will take four weeks to build.
>
> Si's request will take six weeks to build.

On this basis:

> Alex's payoff: $355,000 / 4 = $88,750 per week
>
> Si's payoff: $1,060,000 / 6 = $176,666 per week

Si's story represents the best value for money.

Alex doesn't give up easily and next points out the company doesn't need to choose between these two applications it can have both! He points out that his request is aimed at the Halloween market so the development team could do his request and then do Si's. Of course Si points out that his application is aimed at Christmas and is also time critical.

It is now September 1st and both customers have modified their stories:

> Alex's request: As a toy retailer I want an app that allows kids to select from my special HALLOWEEN range so that their parents can buy. Value: $355,000. Time to build: 4 weeks.
>
> Si's request: As a toy retailer I want an app that allows kids to make lists of toys they want SANTA TO BRING so their parents can buy. Value: $1,060,000. Time to build 6 weeks.

Options

At this point the portfolio board have a range of options they could choose from:

A. Do Halloween & forget Santa

B. Do Santa & forget Halloween

In revenue terms, and cost-benefit terms, if the company is to choose only one story to build then Si's Santa App is the one to build.

C. Do Halloween & then Santa

D. Do Santa & then Halloween

Potentially these options allow both Si and Alex to get what they want. But both options require more analysis.

E. Change the estimates

As silly as it sound people not only suggest this but do this when faced with such problems in real life. Changing the estimates may remove the immediate problem but more likely than not it will create more problems when one or both applications fails to ship in time.

Given the inaccuracies common in the estimation process this strategy might not be as irrational as it initially sounds. Even so, changing estimates to fit

the time available without corresponding flexibility in elsewhere may simply increase risk.

F. Do both and pray

Again, sticking ones head in the sand and hoping for the best is a common strategy but one which rarely brings success.

G. Add more people

Obviously adding more people would increase costs so more analysis would be needed. But before that analysis is undertaken there is a more practical question: could more people be recruited in time? and would they become productive enough quickly enough to make a difference?

The well known Brook's Law states:

> "adding manpower to a late software project makes it later" Fred Brooks[1]

Even trying to recruit more people will create delay because technical software recruitment usually requires existing technical staff to review resumes and conduct some interviews; and such work takes them away from cutting code and developing product.

Since identifying and recruiting suitable staff members is not guaranteed such a strategy will increase risk: existing developer capacity will be surrendered in the hope of adding more capacity.

H. Parallel work

Unless more people are available to undertake then running the work as two parallel streams does not offer any immediate advantage. Indeed such an approach will make for more management work and may create bottlenecks were two work streams require the same resources.

Some parallel work may be beneficial but again more analysis would be required.

I. Reuse code

Where the development team to be an outsource provider there would be questions of intellectual property rights which might prevent this option if Alex and Si were different from different clients. However in the scenario described this is not an issue.

[1] *Mythical Man Month*, Brooks, 1975

But again more analysis is really needed to understand: how much code of the Halloween app could potentially be reused form Santa? And how much of the Santa app would still need developing from scratch? More importantly, how much extra time would be spent building Alex's Halloween app for reuse? The effort estimates need to be revisited.

More importantly, linking the apps through a reuse strategy significantly increases risk. While the apps are considered separately or built in sequence the second can proceed even if the first fails. However, where reuse enters the picture the second app is dependent on the first. Failure to complete the first app places the second at high risk.

Many of these options require more analysis. Such analysis takes time and will delay development work. And all analysis is based on assumptions and available data, should assumptions or existing data prove wrong then the analyst will be flawed.

What is needed light weight analysis technique which can be used with incomplete data. One such tool is a time-value profile which builds on the idea of *cost of delay*.

Time-Value profile

The first time-value profile graph shows the lifetime value of a product as it varies by delivery date. From this graph it can be seen that the stated $1,060,000 of value can only be recognised if the app can be delivered to market before September 15. At any date after this date the revenue generated will be less.

To borrow from food retailing, one may think of this time-value profile as having a "Best Before" date and a "Use By" date. In food retailing some food is marked with a "Best Before" date: consuming the food before this date delivers maximum taste.

Food marked with a "Use By" date will not only taste bad but may actually be harmful - eating shellfish after the "use by" date may well give tummy ache!

From the graph it is clear that if work on the Alex's Santa begins immediately the app will be delivered before the critical "best before" date and most - although not all - the anticipated revenue will be realised.

Importantly the graph also shows two more things. Firstly it is not possible

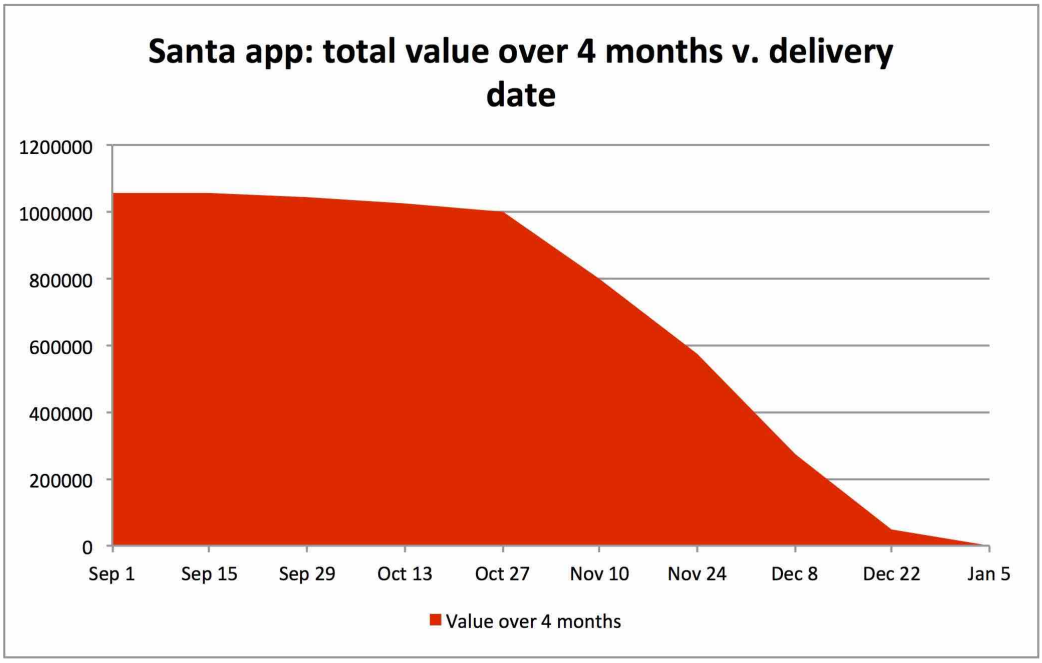

Figure 1: Time-Value profile of the Santa Claus app as of September 1

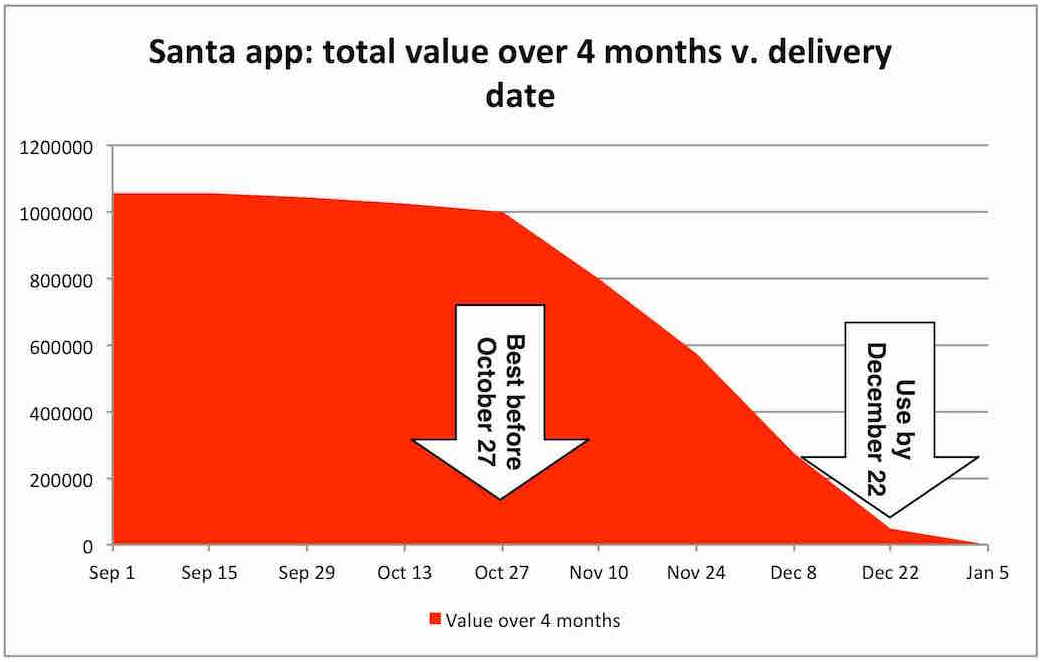

Figure 2: Time-Value profile reveals best before and use by dates

to recognise all the anticipated revenue unless it ships in the next two weeks. Therefore this figure needs to be revised downwards in light of the anticipated development time. Secondly, and more importantly, the graph shows that delivering the Santa App a little later it still worthwhile.

Time-Value profile for Halloween

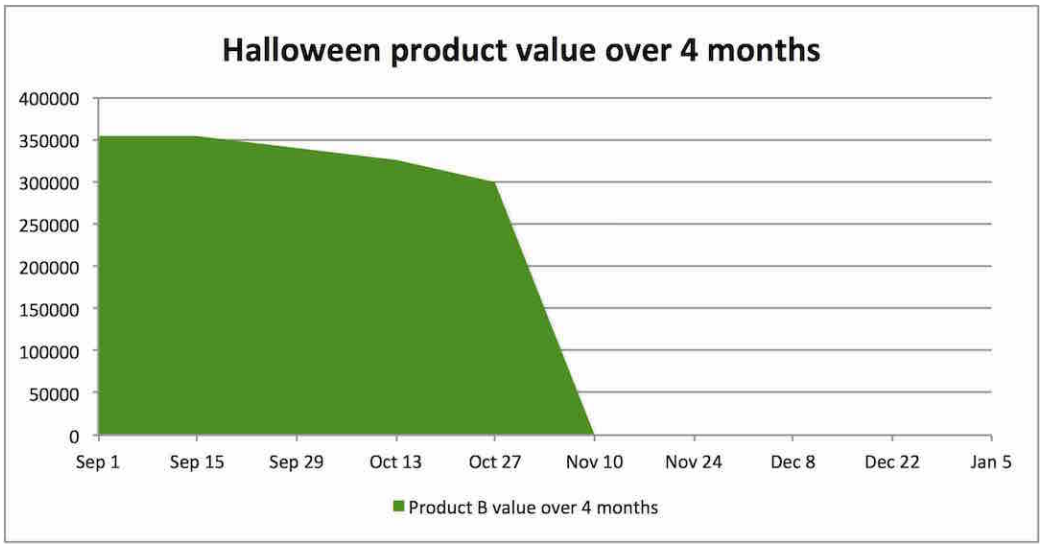

Figure 3: Halloween app time-value profile

Looking now at the time-value profile of the Halloween app we see a similar, but more dramatic, pattern. Again "Best Before" and "Use By" dates can be imposed but in this case there is little difference because the value decline is that much steeper.

When considering best before and use by dates for the Halloween app the placement of the "best before" date is a little subjective. September 15 is arguably the best before date because this is the last date on which the full value could be realised. However as a most value is retained until end of October a later date might also be considered the "best before."

Whatever best before date is used the important thing is that the profile is understood and a date is agreed.

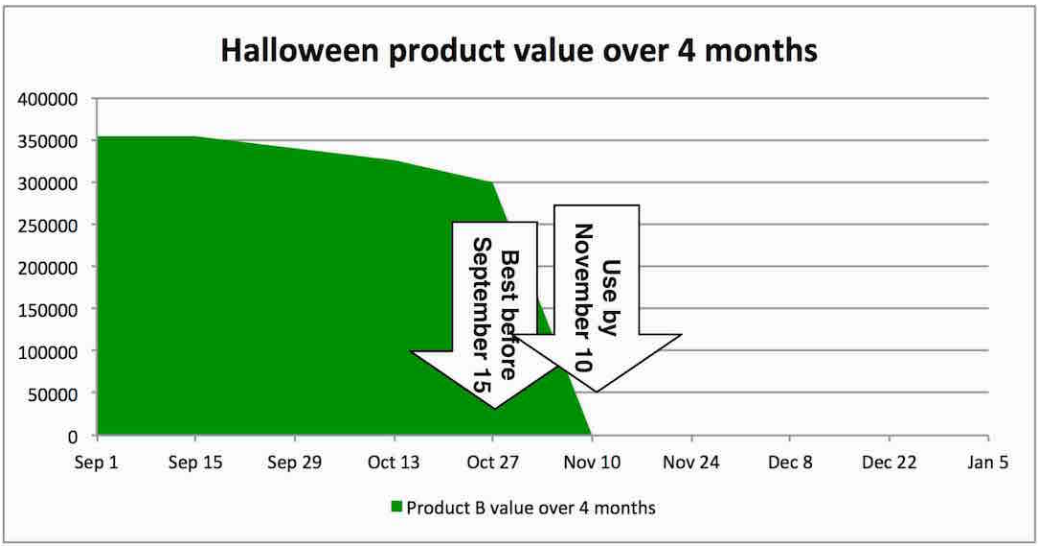

Figure 4: Halloween app best dates

The answer is...

Using these graphs one can calculate what the value maximising options are.

Building Si's Santa app first - options B and D - will result in $1,025,000 of revenue. Even with earliest possible delivery date the full potential cannot be realised. However most of the value can still be realised.

Unfortunately, once the Santa app is delivered the Halloween app cannot be delivered for at least another four weeks: November 9th. By which time it would be valueless. (Options B and D are actually equivalent.)

Reversing the build order - options A and C - would see the Halloween app delivered on September 29 and although it will not make the originally promised $355,000 of revenue it will capture $340,000.

With Halloween done work on the Santa app can commence (option C) and deliver should occur by November 10. By this date $225,000 of potential revenue will be foregone but the app should still generate $800,000 when delivered.

Together the apps will generate $1,140,000, or $115,000 more than the second best option of only building the Santa app.

Hence one can conclude:

Deadlines are elastic by value: Different delivery dates generate

different value.

Parallel build

Having established a model several other options and consequences can be considered. For example: what is Halloween and Santa were built in parallel?

Again one needs to make some assumptions, lets assume that the development team is divided into two equal parts and as a result both development efforts take twice as long:

- The Halloween app now takes eight weeks and be delivered on October 22. This would allow it to capture about $312,000.
- The Santa app would now take 12 weeks and deliver by November 24 capturing $575,000.

In total parallel build approach would capture $882,000 which is still less than the best option considered so far. Arguably this approach reduces risk because both efforts proceed independently and the Santa app is no longer dependent on the Halloween app finishing on schedule.

One might also argue the counter case, that with smaller team both apps are at greater risk because they are less able to absorb problems. For example, inaccurate effort estimates, technical complications or disruption (e.g. illness).

Again one can make different assumptions and calculate different values and different risk profiles. For example, if after completing Halloween the team merged with the Santa team and competed the remaining work in two weeks instead of four the Santa app could realise $800,000 making $1,112,000, although again the risk profile would change.

Pre-work investigation

A more radical approach, and one which might be effective where effort estimates are unreliable would be to spend some initial time in investigation or prototyping. The problem with this approach normally is that such planning or research has a habit of continuing too long. Using these graphs one can see how much value needs to be traded for time.

For example, suppose the first iteration was used in prototyping and research. Lets assume a worst case scenario where this work did not produce any time saving or revenue enhancing result.

As a consequence the Halloween app would not start build until September 15 and would complete on October 13 with the resultant reduction of $25,000 in value.

Similarly the Santa app is delayed two weeks, ships on November 24 and captures $225,000 less value.

A two week delay at the start of the work results $250,000 less value being captured. The work is still worth doing - one can see that further delays would have more significant effects. Depending on ones approach to risk, and the potential upside from prototyping one might feel this is an acceptable option.

Costs

Note that this analysis has taken place with minimal discussion of costs. While costs traditionally get a great deal of attention value receives far less. Reversing this position demonstrates that costs should be considered after a proper discussion of value.

In fact, even the estimated development times could be dispensed with. Rather than asking the technical team "How long will this take to build?" it would be equally possible to ask the technical team: "How much of X would we get in time T?" Or "Given we have four weeks to build something like X could we do it?". Such questions are not only less confrontational but are also open to discussion.

When time-value profiles are available one can also examine the effect of inaccurate estimates. The Halloween app in this example is expected to take four weeks to build. Even if it overruns by another four weeks it will still deliver the lions share of the forecast value. However this will seriously impact the Santa app.

Armed with time-value profile it because possible work backwards to establish a development schedule and examine engineering trade-offs. As discussed earlier, one options could be to release a *smaller* product to capture a small part of a large potential. Alternatively, launching a *larger product* later could capture a large part of a smaller potential value.

Accuracy of Time-Value profiles

It is worth noting that Time-Value profiles do not actually need to be particularly accurate. What is important is the shape of the graph and the dates where value changes.

Consider the first Santa Claus Time-Value profile from above. It is unimportant whether the value that would recognised by releasing on September 1st is actually $1,060,000 or more, or less. It could be $2,000,000 or $500,000. While the actual numbers are useful for comparison with the Halloween app what is more important is the "Best Before Date", in this case October 27.

Between the initial date and October 27 the value changes little, yes there is a slight loss over the month of October but it is nothing compared with what happens during November when each day of delay reduces revenue significantly.

Similarly, once past December 22 it matters little what date deliver occurs on - the Orthodox market is small and for analysis purposes meaningless.

Given that all estimates, whether they are of value or effort, are rough approximations based largely on judgement and therefore, almost by definition, not accurate means caution needs to be exercised. What the graph does show is some key dates.

By analysing how the value changes over time important dates have been revealed.

What is also shown is that the original value estimate of $1,060,000 is itself a time dependent figure and one which is highly unlikely to be realised.

Assumptions and sensitivity

During the discussion above a number of simplifying assumptions have been made - as is common practice with economists! Making these assumptions has allowed an analysis model to be constructed.

The reader is now free to relax any of the assumptions and rebuild the model. I advise relaxing only one assumption at a time!

In doing so the model will remain but the range of outcomes will increase. Replacing my assumptions with your own assumptions will generate an alternative scenario with different graphs and numbers. As the assumptions are relaxed further the potential outcomes will increase.

This is called sensitivity analysis: plug a range of different assumptions, with different numbers, into the model and see how the outcome varies.

Using sensitivity analysis one will find that some assumptions and variables make far more difference than others. Simply knowing which variables cause the greatest variation vastly improves the analysis.

Finally

As the old saying says "Time is money."

While most people are familiar with the idea that more time leads to higher costs fewer think how more time often - although not always - results in lower revenues. In my experience expression "cost of delay" leads people to think about the additional costs incurred by late delivery: the use of express shipping, increased staffing costs, perhaps temporary staff, fines, regulatory charges and so on.

Both extra costs and revenue foregone are known as *cost of delay*. Don Reinertsen discusses cost of delay in depth in his book *Principles of Product Development*[2].

Similarly most people see a deadline as a binary thing: the deadline is met or missed. In fact deadlines are analogue. Different delivery dates and deadlines result in different value.

And again, although the discussion in this chapter is couched in terms of value as money, value may be defined in other terms: children able to read, patients receiving treatment or families accessing clean drinking water.

Constructing time-value profiles and undertaking sensitivity analysis creates a framework within which engineers can work to produce a solution. After all, that is what engineers do best: solve problems within constraints.

[2] *Principles of Product Development*, Reinertsen, 2009.