Explain Program Risks with Cost of Delay プログラムリスクを遅延コストで説明する

Johanna Rothman ***Mass Bay Chapter*** ジョナサン　ロスマン　マスベイチャプター- July 19, 2018

Johanna Rothman works with companies to improve how they manage their product development. She is the author of Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects, 2nd edition, Agile and Lean Program Management: Scaling Collaboration Across the Organization as well as several other books including the newest: Create Your Successful Agile Project: Collaborate, Measure, Estimate, Deliver. See her blogs and more of her writing at jrothman.com.

ジョナサン　ロスマンは、企業における製品開発の管理方法の改善に従事している。ジョナサンはプロジェクトポートフォリオの管理、能力を向上させて多くのプロジェクトを完了させる（第二版）、アジャイル・リーンプログラムマネジメント、組織間のコラボレーションのスケーリング、その他、最新の著書である、成功するアジャイルプロジェクトを創る、コラボレーション、測定、推測、提出などの著者である。彼女のブログと Jrothman.com の投稿を参照。

Topics: Agile, Cost Management, Risk Management

トピックス：アジャイル、コストマネジメント、リスクマネジメント

Product and feature delays can adversely affect adoption for IT projects and revenue for commercial products. Cost of delay (CoD) is a way to explain and calculate those costs. Here's how one agile program manager used CoD to explain risks.

製品やフィーチャーの遅れは、IT プロジェクトの採用や、商業化製品の収入に悪影響を与え得る。遅延コスト（CoD）は、それらのコストを算出し、説明する一つの方法である。ここでは、あるアジャイルプログラムのマネージャーが、どの様に CoD を使ってリスクを説明したのかを示している。

Cindy, an agile program manager, was concerned. Several trends bothered her. Her program had worked hard on finishing a Very Important Customer demo six weeks ago. Up until about a month before that demo, they'd been proceeding in a fairly predictable way. Each team had been able to finish a small story every day. They'd built the entire product at least once a day and it had worked.

アジャイルプログラムマネージャーのシンディーは心配していた。幾つかのトレンドが彼女を悩ませていた。彼女のプログラムは、とても忙しく働き、六週間前にとても重要な顧客へのデモを完了している。デモの 1 か月ほど前から、彼らはとても予測できる方法で進めてきた。各チームは毎日小さいストーリーを完成することが出来ていた。彼らは、製品全体を少なくとも一日に一度構築しており、その方法は上手く機能していた。

No longer. Now, each team's cycle time was increasing (not by a lot, but by about a day every month). It now took each team at least two days to release a feature. Some teams needed three days. And the cumulative flow measurements were increasing. More teams had more work in

progress (WIP) in all columns—not just in testing (which Cindy had expected), but more WIP in development in the coding columns. All the trends were wrong.

今は、各チームのサイクルタイムは増加しつつある。（大幅な増加ではないが、月に一日の割合で増加している。）今や、一つのフィーチャーをリリースするのに各チームは少なくても二日を費やしている。三日必要とするチームもある。そして、累積フローの測定値は増加しつつある。全てのコラムにおいて、より多くのチームに、より多くの進行中の作業（WIP）がある。それはテストに関してだけではない。（テストに関してなら、シンディーにも経験があった。）しかし、多くの開発進行中の作業が、コーディングのコラムにある。全ての傾向がおかしくなっていた。

Cindy called Ruby, the program architect, to see what the problem was. Ruby said Stan, the program product owner, was no longer a rational human being.

シンディーは、何が問題なのかを知るために、プログラムのアーキテクトのルビーに電話した。ルビーは、そのプログラムのプロダクトオーナーであるスタンは、もはや理性的な人では無いと言った。

Cindy said, “I was just looking at the program data. I’m a bit worried by the teams not making the progress we made before the demo. I was going to find you to see if the problems were technical.”

シンディーは、「先ほどプログラムのデータを見ていたのだけど、チームがデモ以前のような前進をしていないのでは無いかと、少し心配しています。問題が技術的なものなのかどうかを、あなたから聞きたいと思っていたの。」と言った。

“Of course, they’re technical,” Ruby said and shook her head. “We knowingly took on technical debt to make the demo date. We *chose* to do that. Stan promised me we would have time after the demo to replan and fix the debt. But do we? Noooo. Now he wants to keep adding more features—in the legacy part of the code base, mind you—and he won’t listen to reason.”

「もちろん、技術的な問題よ。」ルビーは言い、頭を振った。「デモに間に合わせる為に、分かっていて技術的な負債を負ってきていたのよ。私たちは、それを選択したの。スタンは、デモの後には、計画を立て直して負債を返済する時間があると約束したわ。だけど時間はある？いいえ、無いわ。今、スタンは多くのフィーチャーを追加し続けようとしている。コードベースのレガシーパートに。そして、スタンは理由を聞こうともしない。」

Cindy replied, “Well, that makes sense given what I’m seeing in the data. I was thinking of talking with him after I spoke with you.”

シンディーは答えた。「それは、私がデータから理解したことと筋が通っているわ。あなたとの話が終わったら、スタンと話してみようと考えてるところよ。」

“Good luck,” Ruby replied. “I bet he won’t listen to you, either. We’re ‘too technical’ for him. We don’t understand ‘what the business needs,’” using air quotes around the phrases she emphasized.

「頑張ってね。」ルビーは答えた。「スタンはあなたの話も聞かないと思うけれど。スタン曰く、私たちは『技術的すぎる』そうよ。私たちは『ビジネスニーズが何か』理解していないのですって。」強調するところに手で鍵括弧を作りながら言った。

Cindy smiled. “Well, we are pretty technical. I think I can make a case for what we need to do—once I hear what he has to say.”

シンディーは笑った。「そうね。私たちはとても技術的ね。スタンが言いたいことを聞いたら、私たちが何をすべきか例を示せるのでは、と考えているの。」

The two of them spoke about the problems and Ruby’s ideas of the solutions. Cindy called Stan to talk.

二人は問題とルビーの解決案について話した。シンディーはスタンと話すために電話した。

Discover the Root Causes of the Delays
遅延の根本原因を探索する

Cindy started: “Stan, I just spoke with Ruby. She’s not happy with how you’re ranking the technical debt. And I’m seeing data that says the teams have longer cycle time than before. Have you seen that?”

シンディーは話し始めた。「スタン、先ほどルビーと話したのだけど、あなたの技術的な負債のランキングの仕方を、ルビーは快く思っていなかったわ。データを見たのだけど、以前よりチームのサイクルタイムは長くなっているわ。あなたは、気付いている？」

“I have,” he said. He took a deep breath. “Look, we’re having trouble. My boss, the VP of marketing, thinks the only measure of us as product managers and product owners is the number of features we ship. The worst part is he thinks of one feature as an entire feature set. He doesn’t realize what the teams do now.”

「分かっているよ。」スタンは言い、ため息をついた。「見てみてよ。問題があるんだ。私の上司である、マーケティングの VP は、私たちプロダクトマネジャーやプロダクトオーナーを、出したフィーチャーの数だけで評価しようとしているんだ。最悪なのは、全体のフィーチャーセットで一つのフィーチャーと考えていることさ。彼は、チームが今何をしているのか理解していないよ。」

Cindy bit her lip. “Does he realize you folks could workshop the larger feature sets, break them down smaller and release more features faster?”

シンディーは唇を噛んだ。「あなたたちが、長いフィーチャーセットをワークショップして、小さく分けて、より多くのフィーチャーを早くリリースできることを、彼は理解していないの？」

Stan said, “No! I keep telling him that, but it’s as if he’s never heard any of this before. He thinks that making time for ‘refactoring’ and ‘testing’ is code for laziness. He tells me workshopping the stories ‘coddles’ the teams. He doesn’t get it!”

スタンは言った。「全く。彼に説明し続けているが、全く聞いたことが無いような反応さ。彼は、『リファクタリング』や『テスティング』に時間を割くのは、怠惰な証拠だと考えている。彼は、ストーリーをワークショップするのは、チームを甘やかしていると言うのさ。彼は全く分かっていないよ。」

Cindy asked, “Oh, so he’s new to the whole idea of agile projects and programs?”

シンディーは聞いた。「それは、彼はアジャイルプロジェクトやプログラムの考え方に馴染みがないのかしら？」

Stan sighed. “Yes. It’s terrible. It doesn’t matter what I say. I don’t get anywhere with our conversations.”

スタンはため息をついた。「そうさ。酷いものさ。何と言っても変わらないよ。彼との会話で何も得られないよ。」

“Okay,” Cindy said. “I think I have an idea.” She hung up the phone.

「分かったわ。」シンディーはいいました。「私はアイディアがあるの」と言い、電話を切った。

Cindy had met people like this VP in the past. He was stuck in a mindset of thinking that the teams had to do “all” of a feature set instead of realizing the value of frequent deployment of small features. Too often, those people also didn’t understand the value and the costs of not keeping the code base and unit tests clean.

シンディーは以前に、この VP のような人に会ったことがあったのだ。その人は、小さいフィーチャーを短期間で開発する価値を実現するのではなく、チームは『全ての』フィーチャーセットを開発しなくてはならない、という考え方に凝り固まっていた。しばしば、このような人々は、コードベースとユニットテストを綺麗な状態に維持しないことの負荷と価値を理解していない。

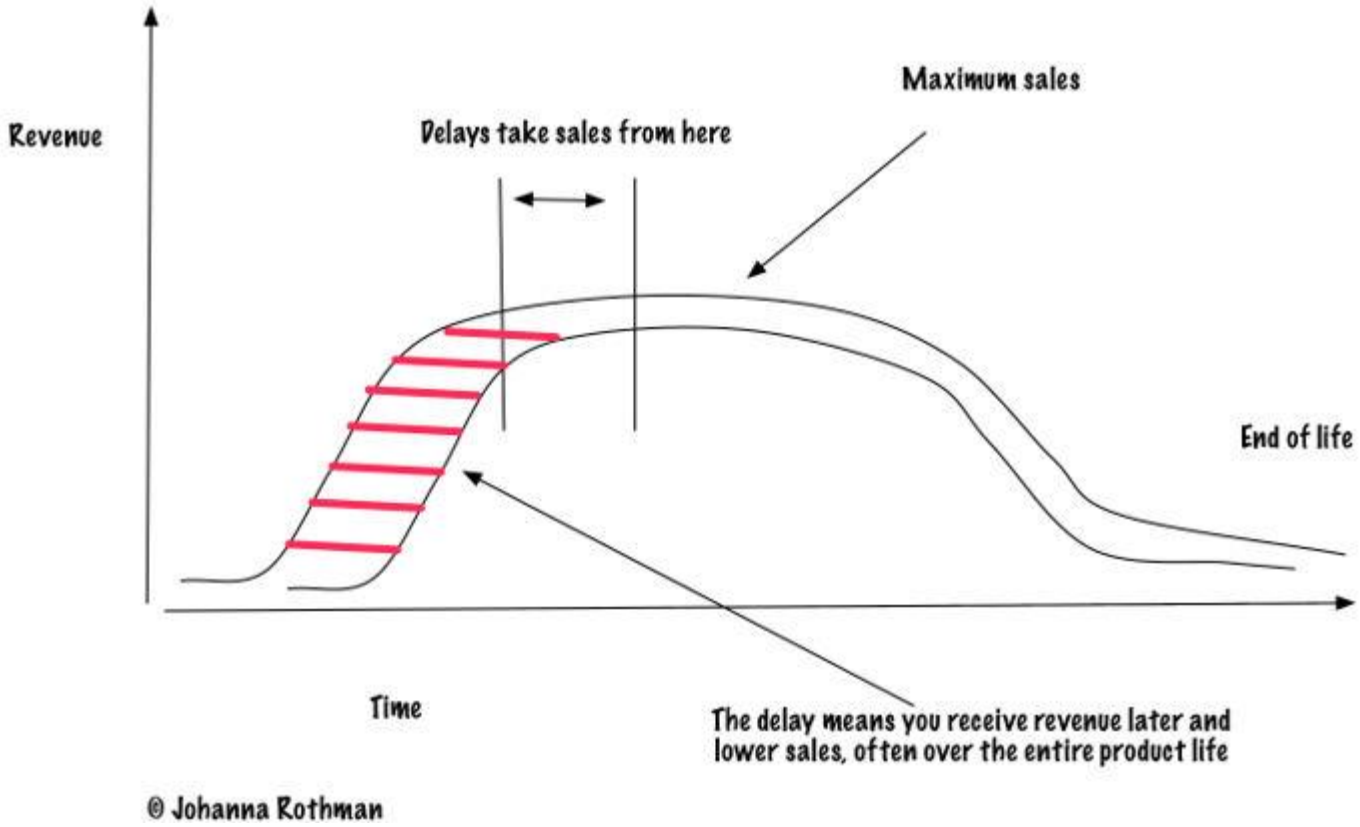
She decided to use cost of delay CoD to see how to decide what work to do—and when.

シンディーは、どの仕事を何時するかをどの様に決めるか分かるように、遅延コスト（CoD）を使うことにした。

Use Cost of Delay to Explain the Effect of Project and Program Problems
プロジェクトとプログラムの問題への効果を説明するために遅延コストを使う

Cost of delay is the effect of a delay on expected revenue:

遅延コストとは、期待される価値への遅延の影響である。



図中

Revenue：収益
Time：時間
End of life：製品寿命
Maximum sales：最大売上げ
Delays take sales from here: 遅延により売上げが失われる期間
The delay means you receive revenue later and lower sales, often over the entire product life：遅延は売上げ時期の遅れと売上額の減少を意味する。売上額の低下はしばしば製品寿命期間全てに渡り生じる。

- The company misses the potential sales from the introduction delay.
- The delay introduces lower maximum sales.
- The delayed introduction creates less overall demand, so the feature has less value in total.
- With less demand, the end of life might be earlier with a delay. Often, this is a function of not being able to capture the market at the optimum time. Since you don't have the customers, you have an earlier end of life.
- 企業は製品投入時期の遅れにより、潜在的な売り上げを損失する
- 遅延により、最大売上高が低下する
- 遅延により、需要全体の創出量が減少し、全体としてフィーチャーの価値が減少する
- 需要の減少に伴い、遅延によって製品寿命が短縮するかも知れない。しばしば、これは最適条件では捉えることができない機能である。顧客がいなくなるので、製品寿命が短くなる。

As a program, they were only six weeks into this downturn, so Cindy was pretty sure Ruby and Stan could address the problem in the program.

プログラムとして、停滞状態に陥ってから 6 週間しか経っていないため、シンディーはルビーとスタンはこのプログラムの問題に対応できると確信していた。

The real issue was the conversation with Stan's boss. Cindy decided to calculate the CoD for one of the delayed feature sets. That would take both the technical debt costs and how to address a legacy feature set without planning for sufficient automated tests.

本当の問題はスタンの上司との会話だった。シンディーは遅延したフィーチャーセット一つについて、CoD を計算することにした。技術的な負債コストと充分な自動化されたテストの計画無しに開発されたレガシーフィーチャーに対応する為のものだ。

Calculate the Possible Costs of Delay for the Project or Program

考えられるプロジェクト又はプログラムの遅延コストを計算する

Cindy calculated the costs of delay this way: She took just the three teams most affected and their run rate. She used 16 people, at an average weekly rate of $,2000/week. They had a six-week delay right now in fixing the technical debt, so the product would be more robust and work according to the customers' expectations:

シンディーは遅延コストを次のように計算した。シンディーは最も影響を受ける三チームのみを選び、それらのチームのランレートを計算した。16 人、賃金は$2,000/週を用いた。現在、技術的な負債を修復するのに 6 週間の遅延を抱えている。修復すれば、製品はより頑健になり、顧客の期待に添うように機能するようになる。

16 (people) x $2,000 (salary/week) x 6 (weeks) = $192,000 total cost

16 (人) x $2,000 (賃金/週) x 6 (週間) = $192,000 トータルコスト

She called Ruby and Stan and asked how long—as a ballpark estimate—it would take these teams to fix the technical debt, create smaller stories, and create automated unit and system tests for the problematic feature sets.

シンディーはルビーとスタンに電話し、概算見積りとして、チームが技術的な負債に対応して、より小さなストーリーを作り、問題のあるフィーチャーセットに対する、自動化されたユニットテストとシステムテストを開発するのに、どのくらいの期間を要するか質問した。

Ruby said that if the teams only worked on fixes, she was pretty sure the teams could complete them in two weeks. In addition, the teams wouldn't be digging themselves further into a hole of difficult code and tests. Why was she so sure? The teams had left notes on what they hadn't done before.

ルビーは、もしチームが問題の修復のみに専念するのであれば、2 週間で完成させることができると確信している、と言った。更に、チームは困難なコードやテストについて深堀りしていくことは無いだろうと言った。何故、ルビーはその様な確信を持っていたのだろうか？それは、チームが何をやり残していたか、記録を残していたからだった。

The cost of fixing the technical debt was two weeks multiplied by the 16 people and $2,000/week, a total of $64,000. That was much less cost than what they were spending now: $192,000 over the previous six weeks (and only growing).

技術的な負債を修復する為のコストは、2 週間に、16 人と$2,000/週を掛け合わせて、計$64,000 となった。現在費やしているコストは、過去 6 週間に渡って費やした$192,000 と増加し続けているものであり、修復コストはそれに比較して、ずっと低いものになっている。

Stan said, “I’d like to have a feature set workshop day with the product owners from each of those teams, and then I suspect we would have small-enough stories that the teams could pretty easily create automated tests.”

スタンは言った。「各チームのプロジェクトオーナーとフィーチャーセットのワークショップの日を一日取りたい。そうすれば、充分に小さいストーリーを作ることが出来、チームがかなり簡単に自動化テストを開発することができるようになる。」

Cindy thought that might be a little optimistic, so she did the calculation two ways: once for a day and once for a week. She decided to make the teams’ cycle time two days instead of the normal one day for the newer stories to account for the lack of tests. That meant that the time to workshop the stories was four people (Stan and the other three product owners) times one day, which was $2,000. If it took Stan and the other three people a week, that would be $8,000 to fully understand the new stories in the legacy area.

シンディーは、それは少し楽観的なのではと考え、一日に一回と週に一回の、二通りの計算をした。シンディーは、チームのサイクルタイムを、これまでの新しいストーリーを一日で開発するサイクルから、二日とすることに決めた。これまでのサイクルは、テストを実施しない原因となっていた。ストーリーをワークショップする時間は、四人（スタンと三人のプロダクトオーナー）掛ける一日で$2,000 となることを意味していた。もしスタンと他の三人が一週間費やしたら、それは$8,000 となり、レガシーエリアの新しいストーリーを完全に理解できるようになる。

Explain So People Understand　人々が理解できるように説明する
Cindy spoke with the VP and explained these ideas:

シンディーは VP と話し、このアイディアを説明した。

- Aside from the teams’ disappointment, it had cost the program $192,000 to work like this. For a $64,000 investment, the teams would be able to proceed at their previous pace.
- And, for between $2,000 and a maximum of $8,000, the teams would be able to work with the legacy code faster.
- チームの失望は別にして、この様な仕事の仕方は、プログラムに$192,000 のコスト負荷を与えている。$64,000 の投資により、チームは以前のペースで仕事を進められるようになる。
- 更に$2,000 から最大$8,000 の投資で、チームはレガシーコードを速く開発できるようになる。

The VP would see throughput go up and keep the customers. He grumbled, but agreed. He and Cindy decided to meet again in four weeks to make sure the teams continued to produce faster.

VP は処理能力が向上し、顧客を維持することを確認するであろう。VP は不平を言ったが、了承した。シンディーと VP は、四週間後に再び会い、チームが速い生産を維持しているかを確認することを約束した。

If your project or program “suddenly” slows down, first, discover the cause(s). Once you know what causes the delays, see if you can explain the slowdown with cost of delay. It’s one more tool in your toolbox.

もしプロジェクトやプログラムが『突然』遅くなったならば、まず、その原因を探ってみよう。そして、遅延させている原因が分かったら、遅延コストを使いその遅れを説明できそうか、見てみよう。遅延コストは、みなさんのツール箱に追加するツールです。