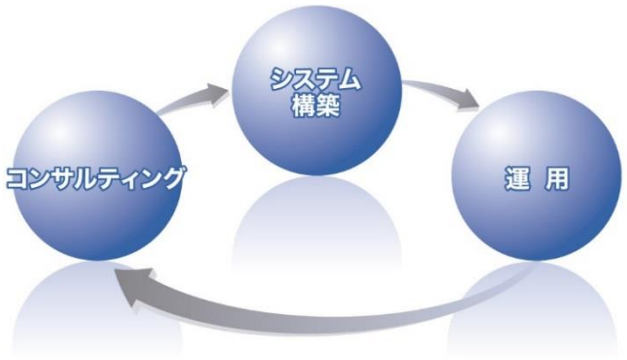
滞留タスク管理のススメ
～動かないモノゴトを追跡する～

（株）SRA
古石　ゆみ
豆腐谷　晶憲

会社紹介（SRAの活動）



製品開発、研究開発、基幹業務開発等、広い分野のスキルを保有し、開発支援ツールを用いた各種開発、業務アプリケーション、大規模Webシステム開発を中心とした受託開発を行っております。

業務システム構築

各分野のリーディングカンパニーに対し、企業の事業戦略を支え、競争力強化に直結するシステムを構築
[構築実績]
システム　●物流システム ●生産管理システム ●販売・在庫管理システム
●顧客管理システム ●Webシステム（B to B、B to C）等

業　　種　●製造 ●通信 ●流通 ●運輸 ●電力、ガス ●薬品 ●食品
●公共 ●情報機器メーカー等

組込ITサービス

日本を代表する家電・車載製品メーカーの高機能製品の組込システムを開発
［具体例］ 高機能ゲーム機器、スマートフォン、カーナビ、
業務用AV機器、複合機、通信機器他

組込開発に求められる「操作性品質」向上という課題にQtフレームワークやインタラクションデザイン方法論など複数の側面からアプローチ

最先端製品開発に研究支援の立場でも参画

コンサルティング

ソフトウェア工学に基づき、実績ある方法論・ツールにより課題を解決

運用サービス

仮想化技術等の普及に伴い、高度化・複雑化する運用業務に対し、システムのメリットを最大限に引き出す「アウトソーシングサービス」を24時間365日体制で提供

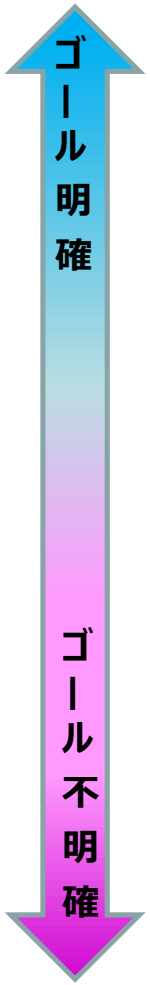
Copyright © Software Research Associates, Inc. All Rights Reserved

はじめに

- 開発現場での不具合管理を行う中で、通常の開発のような「予実管理」をしようとして、うまくいかなかった。
- うまくいかなかった原因とその対策を、データに基づき振り返ってみた。
- 「滞留タスク」に着目し、適切なタイミングで検知する対応を考察してみた。

＜目次＞

タスク管理のアプローチはさまざま



管理のアプローチ	前提条件、制約など	メリット	デメリット
予実管理のアプローチ	✓ 計画ありき ✓ ある程度、ゴール（タスクの全量）が決まっている	✓ 予定通り進まないモノゴトに着目 ✓ 完了日が見通しやすい。もしくは目標が立てやすい。	✓ 計画変更には理由が必要
タスクかんばん的アプローチ	✓ 一定期間（通常2Wくらい）のタスクだけ計画する ✓ 溢れたタスクはバックログへ ✓ 1ターム毎に棚卸し、次のタームの計画を立てる	✓ 受け付けたタスクが定期的に見直される ✓ タスクの全量が変化しても対応できる	✓ 完了日が読めない
チケット的アプローチ	✓ **そもそも計画がない**	✓ 担当者のアサインが早い ✓ 複数の担当者が関わる場合に有効 ✓ 来たものから片づける、スピードが求められる	✓ いつ、何が終わるかわからない（担当者次第） ✓ 放置されるタスクはいつまでも残される ✓ たいていは放置することのリスクが確認されない ✓ どれが放置されているのか気づかない

予実管理ではうまくいかないタスクもある

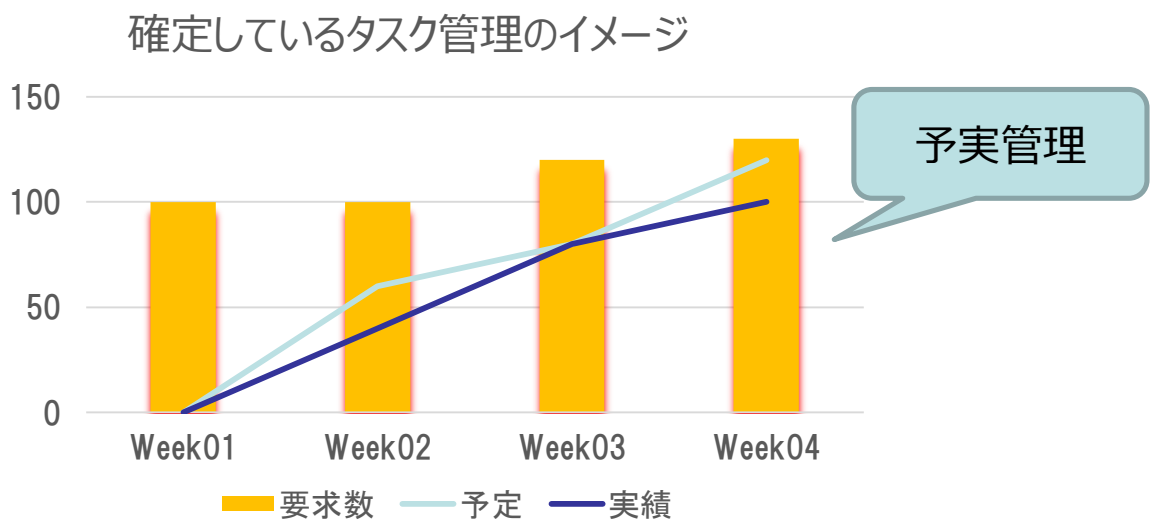
タスクの特徴によって、管理のアプローチを選ぶ

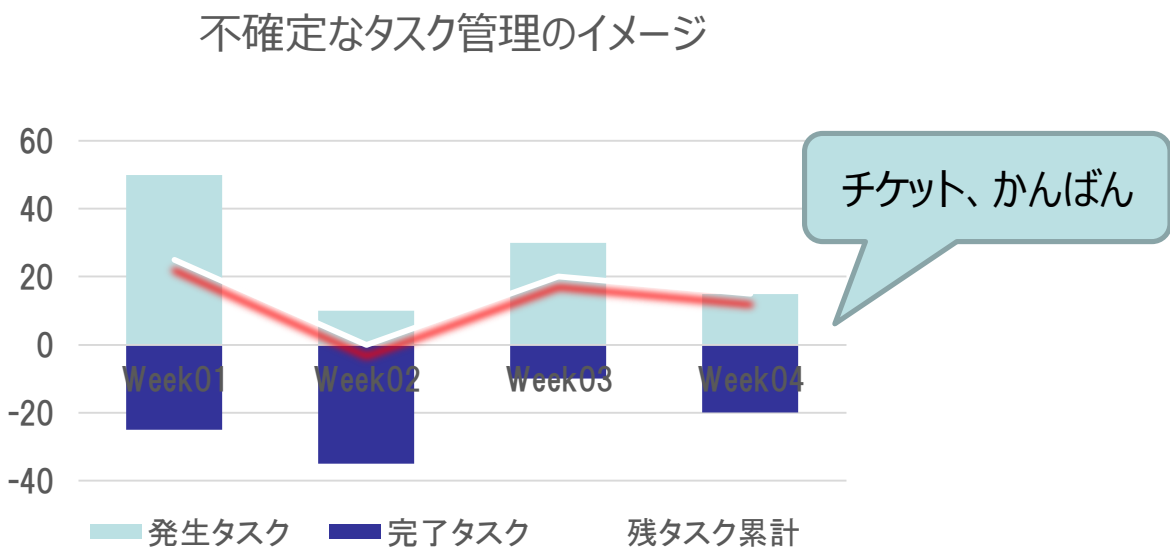
ゴールが明確なタスク

- タスクの量がほぼ決まっている（計画可能）
- 新規開発、仕様変更など。ゴールは「100%」
- 多少の変更は許容。計画変更でキャッチアップ可能
- 予実管理が可能、**予実が乖離することを管理する**

ゴールが不明確なタスク

- タスクの量が予測できない（計画不可能）
- 不具合対応、問い合わせ対応など。
- 「何等かの収束」が起きるまで、日々のタスクが増えていく。一旦0%になっても、また増える。
- 予実管理が不可能。**何を管理すればよい？**





ゴールが不明確なタスク管理の具体例

（実際の不具合管理で直面した事例です。）
状況：現時点の残不具合179件。
課題：あと1か月で、不具合を収束させる。

<対策>

時間がかかりそうなタスクを識別し、先に終わらせる。

その他のタスクについて、チームのタスク消化率を測り、消化率の目標値を設定する（13件/1日）

日々のタスク消化件数と残タスク件数をモニタリングする。

<結果>

時間がかかりそうなタスクは早めに終わらせることができた。 ✔

設定した目標値で、ほぼ消化することができた。 ✔

タスク消化のペースは変わらなかったが、**新しい不具合が発生し**、1か月では収束できなかった。 課題

収束させるために何をしたか？

状況：**新しい不具合が発生し**、1か月では収束できなかった。
課題：期日までに、不具合を収束させる。

<対策>

優先度を見直し、「低」の対応を後回しにした。「高」「中」のものから着手した。

他チームが原因で止まっているタスク（再現待ち、QA待ちなど）について、エスカレーションパスを明確にしてリマインダを出した。

特定の担当者に作業負荷が集中して着手できなかったタスクを、他のメンバーに割り振り直した。

<結果>

単なる残件数のモニタリングでは拾えないリスクだった。
もっと早く手を打っていれば。

✔

✔

だけど手遅れ？

✔

だけど手遅れ？

「滞留タスク」
- ✓ 意図する／しないに関わらず「後回し」になり長期間放置されているタスク
- ✓ 「他チームの対応待ち」などで、「積極的にリマインド」されないことが多い

「滞留タスク」の背景、要因

<外的要因>自分が待っている

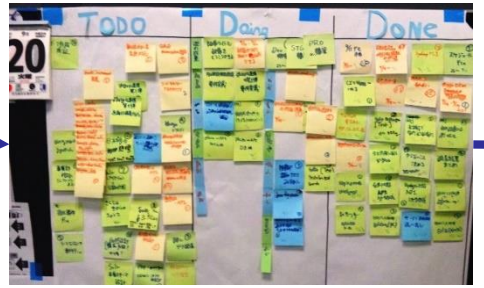
1. チーム視点の問題
   - そもそもさばき切れないタスクを受けてしまった
   - 「早急に対応せよ」の外圧
   - 受けたタスクをとりあえず担当者に振ってしまう
   - 振ったあとは担当者の対応待ち

2. 個人視点の問題
   - 割込みが入る
     - 優先順位が入れ替わる
     - 仕掛中のタスクが後回しになる
   - 他チーム等へ依頼、対応待ちのタスクがある
     - リマインダを出しても反応がない

<内的要因>自分が待たせている

3. 先送りの心理
   - 優先順位は自分で決める、楽なものから着手
   - 後回しになったタスクを忘れてしまう
   - 仕掛中のタスクが中断されるとストレスが溜まる
   - 「待ち」のものは、返ってきたら面倒。面倒なものは後回しにしたい

- XX月XX日までに、対応してもらわないと！
- リスクはすぐに上げて欲しい。



- 早急に対応しなくては！
- 担当者をアサインしなくては！

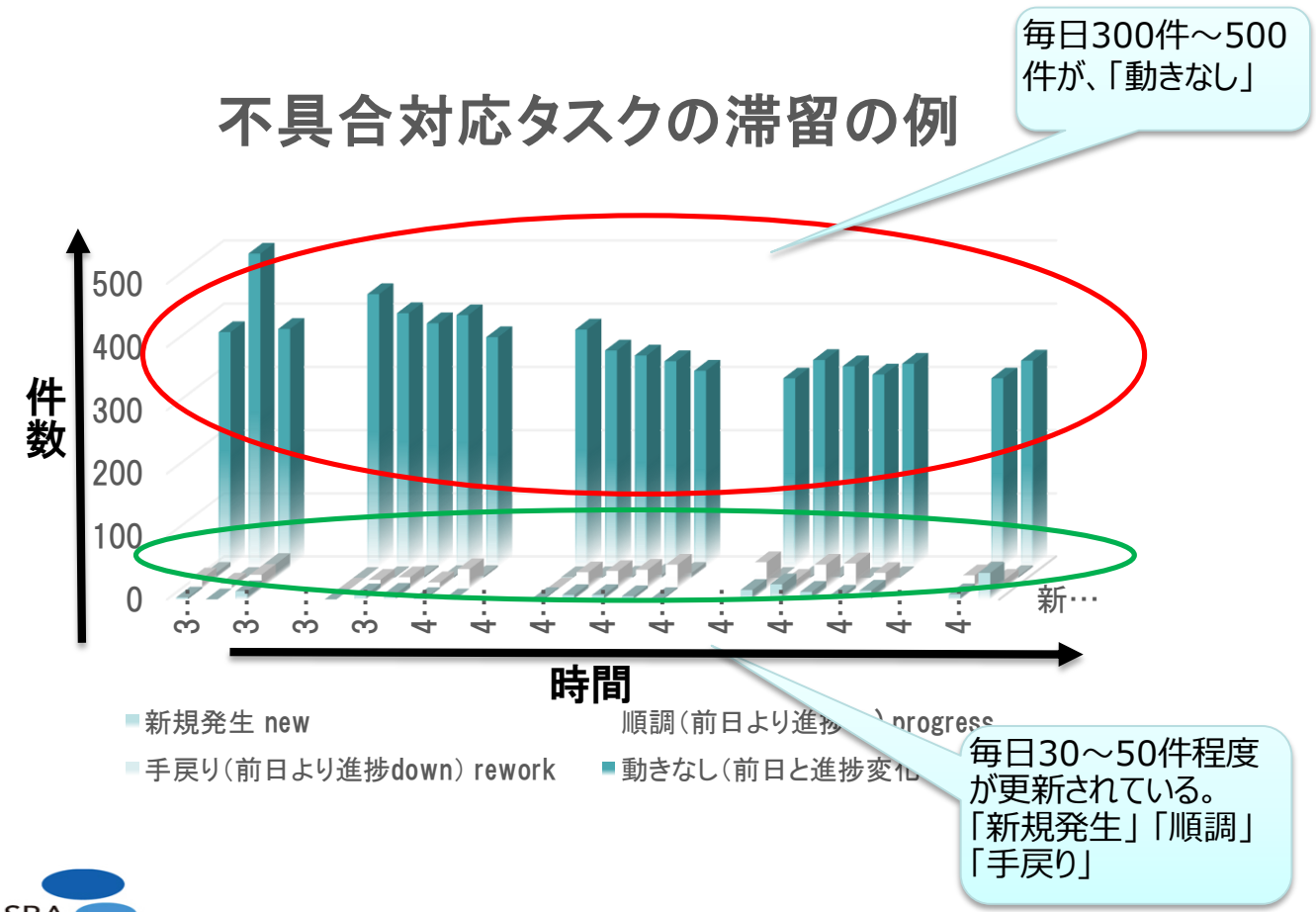
仕掛中のタスクがあったのに、後回しかよ。。。再開するときに忘れちゃうよなぁ。

QA回答が返ってこないと進められないなぁ。。。でも、今さら返ってきても面倒かも。

どれからやってもいいなら、面倒なのは後回しにしよう。。。

「滞留タスクによるリスク」をタイミングよく拾うには？

処理ペースの10倍のタスクが「滞留」している状況。
いつからこうなったのか？**事前に、手は打てなかったのか？？？**



こんなこと↓ができれば良かった。

1. 定期的にモニタリングを実施

- ✓ パラメータを監視

2. 絶妙なタイミングで

- ✓ パラメータが一定の値になったら

3. 手を打つ！

- ✓ タスクの棚卸し
- ✓ エスカレーション
- ✓ 担当者の負荷分散

滞留の度合を測るメトリクスを作ってみた

測定の目的（Goal）	滞留タスクを監視し、適切なタイミングで処置したい

何を知りたいか（Question）	タスクがどのくらい滞留しているのか？
何をどのように測るか（Metric）	滞留率 (1日あたりの全タスク毎に) ◆ タスクが更新されていない日数 ◆ 4日以上更新されていないタスクの件数とその滞留日数

何を知りたいか (Qestion)	なぜ、タスクが滞留しているのか？
何をどのように測るか（Metric）	（１件のタスク毎に） ◆ 滞留期間 ◆ 滞留の理由 ◆ 処置方法 ◆ タスクを放置した場合のリスク

<滞留率の算出方法>
- タスクが更新されていない日数を4段階に分ける。
  - ✓ A:3日以内、B:4日～1週間、C:1週間～2週間、D:2週間以上
- **滞留率＝（Bの件数×0.5+Cの件数×0.7+Dの件数）／（A+B+C+Dの件数）**
  - ✓ 取りうる値は0(0%)～1（100%）

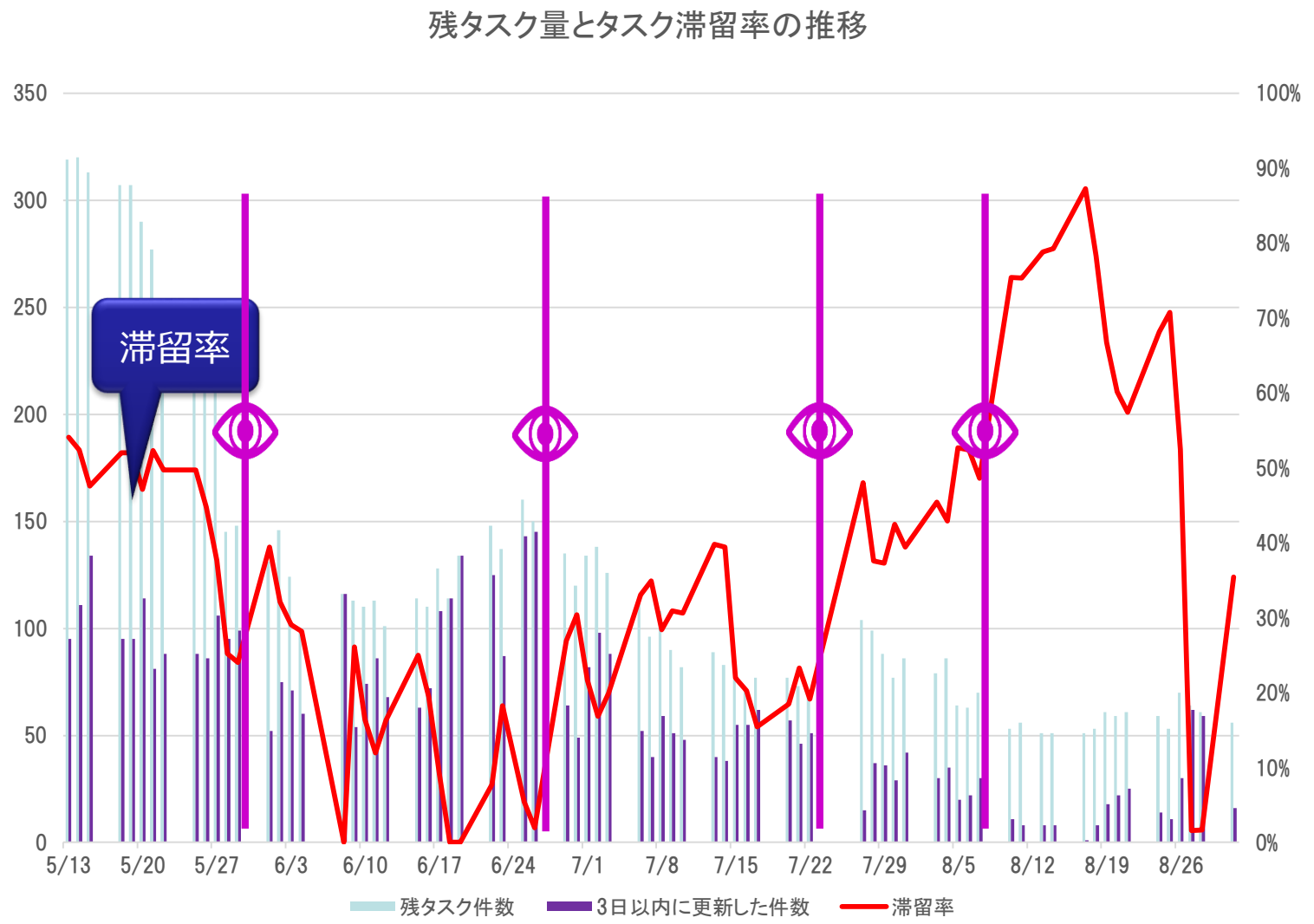
#	「滞留状況」のケース	Aの件数	Bの件数	Cの件数	Dの件数	合計	計算式	滞留率
1	全てのタスクが3日以内に更新	100	0	0	0	100	(0*0.5+0*0.7+0)/100	0%
2	タスクの半数が4日以上放置	50	24	10	16	100	(24*0.5+10*0.7+16)/100	35%
3	全てのタスクが4日以上放置	0	0	0	100	100	(0+0+100)/100	100%

メトリクスを使ってシミュレーションしてみた

実際のデータで、振り返ってみた

	報告日	残タスク件数	滞留率	3日以内に更新した件数	B：4日～1週間以内にチケット更新	C：2週間以内にチケット更新	D：3週間以上放	仕掛中チケット件数	平均効率
2	5/13	319	54%	95	66	62	96	319	
3	5/14	320	52%	111	63	33	113	320	
4	5/15	313	48%	134	37	39	103	313	
5	5/18	307	52%	95	86	31	95	307	
6	5/19	307	52%	95	86	31	95	307	
7	5/20	290	47%	114	60	31	85	290	
8	5/21	277	52%	81	76	44	76	277	
9	5/22	246	50%	88	32	66	60	246	
10	5/25	246	50%	88	32	66	60	246	

- 残タスク件数や、更新件数（および滞留件数）だけではキャッチできない
- 滞留率が跳ね上がるタイミングがいくつかある。
- 日々の滞留率をモニタリングし、「跳ね上がる」タイミングをキャッチすることで、タスクの滞留リスクに対して早めに手を打つことができる。
- 残タスク件数との関連も見ながら、適切なタイミングを選ぶとよい。



まとめ　～動かすべきモノゴトを識別し、追跡する～

- 滞留タスクには２つのパターンがある
  - ◆<外的要因>自分が「待っている」タスク
    →回りを巻き込み早く手をつけるべき
    「クローズしてしまえ！」の判断も、時には必要
  - ◆<内的要因>自分が「待たせている」タスク
    →自律的にコントロールするしかない
    「先送りの心理」に立ち向かう

■　チーム外で滞留していて協力要請できるもの
- ✓ QA回答待ちで進まないタスク
- ✓ 判断待ちでクローズできないタスク
- ✓ 内部で悶々としているよりも、聞けば早いもの
- ✓ エスカレーションパスが分からないもの

「滞留タスク」のうち「自分が待っているタスク」は、早めに手を打つことができる。
状況をモニタリングし、適切なタイミングで識別し、対処することが重要

（おわり）

ご清聴ありがとうございました。

お願い

- 同じような苦労をされた方、いらっしゃいませんか？
- 滞留タスクが発生するような場面に出会うことはありませんでしたか？
- 滞留タスクをどのように処理されていますか？
- ゴールが不明確なタスクを管理するための、もっといいアイデアはありませんか？