計画通りに開発が進まないスクラムチームに関する研究
～チーム内の共通理解って大事～-

A Study on Scrum Teams That Do Not Develop as Planned
-Common understanding within the team is important-

研 究 員：西村　優（東京エレクトロン株式会社）
　　澁澤　良（株式会社ｆｅａｔ）
　　村上 拓也（テックスエンジソリューションズ株式会社）
　　星野 友基（ビー・ユー・ジーDMG 森精機株式会社）
　　澤口 勲（株式会社ビズリーチ）
主　査　：永田 敦（サイボウズ株式会社）
副 主 査：山口 鉄平（freee 株式会社）
　　荻野 恒太郎（株式会社カカクコム）
アドバイザー：細谷 泰夫（三菱電機株式会社）

研究概要
スクラム開発は何を実現すべきかが示されている PBI（Product Backlog Item：プロダクトバックログアイテム）を元に,実際の作業をタスクに分割するスプリント計画を行い,スプリント期間で開発を行う手法である.
しかし，計画した作業が予定していたスプリント期間内で完了せず，開発遅延や，品質低下を招くケースが散見される．これらの開発チームを分析したところ，PBI の内容は共通認識となっていたが，作業タスクへの落とし込みが不十分なケースが見受けられた.
本研究では，開発チームメンバー全員でタスク分割を行う事で，タスク漏れ低減とチームメンバー全員の共通理解にし，手戻りを低減させることを提案する.
全員でタスク分割を行う実験を行った結果，タスク漏れと手戻りの低減により，計画遅延を抑制できた．またメンバー全員がタスクの理解を深めることで，負荷状況から担当タスクの入れ替えが可能となり納期短縮の効果も確認できた.

Abstract
Scrum development is a method to develop in a sprint period by making a sprint plan to divide the actual work into tasks based on the Product Backlog Item (PBI) that shows what should be achieved. However, there were cases where the planned work was not completed within the planned sprint period, resulting in developmental delays and quality deterioration. When we analyzed these development teams, we found that the content of the PBI was commonly understood, but in some cases, it was not sufficiently incorporated into the work tasks. In this study, we propose to reduce task omissions by dividing tasks among all development team members and to reduce rework by making it a common understanding among all team members. As a result of the experiment in which all members divided tasks, we were able to reduce delays in planning by reducing task omissions and rework. In addition, by deepening the understanding of the tasks among all members, it became possible to replace the task in charge based on the load status, and the effect of shortening the delivery time was also confirmed.

1. はじめに

2001 年に「アジャイルソフトウェア開発宣言」が発表され[1],アジャイル開発の採用率は増加傾向にある.アジャイルの中でもスクラムを採用されているケースが多い.

スクラムガイド 2020 には，スクラムとは,複雑な問題に対応する適応型のソリューションを通じて,人々,チーム,組織,が価値を生み出すための軽量級フレームワークである，と書かれている.[2]

スクラムは短いサイクルで開発しリリースを行う事で,従来のウォーターフォール型の開発に比べ,ユーザからのフィードバックを早く製品開発に組み込むことを期待している.これらのフィードバックやビジネス要件,市場動向などを加味し,達成したいゴールや理由等を PBI に表現し,PBI の優先順位を決定する.

開発チームはスプリントと呼ばれる一定期間（１ヶ月以内の長さ）に,実現可能な PBI を選択し開発を行う.開発チームはスプリントプランニングを行う事で実現すべきゴールや価値を理解し,ゴール達成に必要な作業をタスクに分割を行う.

しかしスクラムはフレームワークであるため,プロダクトを構築するプロセス,技法,決定的な方法論は明確でない.

そのため,スクラム開発に精通していない組織がスクラム開発を行った場合，計画通りに開発が進まなくなるケースが散見される.開発作業が計画通りに進まないと遅延が生じ,納期に間に合わせるように無理に開発を行う事で検討不足やテスト不足などによる品質低下に繋がる可能性がある．本研究では,タスク漏れや手戻りにより,計画通りに開発が進まない状態のチームの改善のための研究を行う.

2. 背景

我々の研究チームが関わっている開発チームでは，複数の計画遅延が頻発している開発チームだけでなく，幸いなことに計画遅延の発生率が低い開発チームも存在した．組織や製品の特性の違い等もあるが，実際の開発作業に着目し，比較分析を行った．なお，比較を行う上で，同じ作業内容であっても呼び名が違う場合があったが，実際の内容が同じであれば，呼び名が違っても同じ作業として解釈を行った.

2.1 開発状況比較

研究チームで比較分析を行ったのは，下記の 3 チーム（表 2）である．A チーム，B チーム共に計画通りの開発が進んでいないチームで，C チームは開発遅延が生じていないチームである．A チームと B チームでは問題遅延の発生状況が異なる事から，表 3 の指標を定義し，比較を行った.

表 1　チーム開発状況

	Aチーム	Bチーム	Cチーム
スプリント期間	2週間	2週間	1週間
1スプリントのPBI数	110 PBI/Sprint	2～3 PBI/Sprint	5 PBI/Sprint
計画遅延発生率（未完了率）	66.4%	0.8%	3.4%
手戻り発生確率	－	90%	0%

表 2　チーム構成

		Aチーム	Bチーム	Cチーム
PO（Product Owner：プロダクトオーナー）		1名	1名	1名
SM（Scrum Master：スクラムマスター）		無し	1名	1名
開発チーム		15名	3名	7名
内訳	開発者	14名	2名	6名
	QA	1名	1名	1名

表 3　指標定義

	定義
計画遅延発生率	プランニングにてスプリント期間に計画したPBI数を母数とし、PBIが未完了になったPBIの発生確率。
手戻り発生率	プランニングにてスプリント期間に計画したPBI数を母数とし、伝達ミス、認識不足などによる計画外の追加作業や再検討が発生した確率。手戻りかどうかは判断が難しい所があるが、QAが開発からヒアリングを行い、手戻りかどうかを双方合意のもとで判断する。

2.1.1 A チームの状況

A チームはプランニングで計画した PBI がスプリント期間内に完了しているものが 33.6%と低く，計画通りに開発が完了しない状況だった．手戻り発生率については，過去の実績案件に対する実績データが無いため計測不可だった．

2.1.2 B チームの状況

B チームでは作業の手戻り発生率が 90%と非常に高いが，納期を優先して作業していたため，計画遅延発生率は 0.8%と低い数字になっていた．しかし納期を優先するために一部作業の省略や十分な評価期間を設けないといったケースが発生し，顧客から仕様に対する問い合わせや品質問題が多発していた．

2.1.3 C チームの状況

C チームの計画遅延発生率は 3.4%となっているが，スプリント期間中に PBI の規模が見積もりより大きいことが判明．対策として，再度リファイメントを実施，PBI 分割を行い，再計画となっていたため，手戻りの発生もなく計画遅延も発生していないと判断した．

2.3 開発作業比較

3 チームの開発作業の実態を比較したところ，いずれのチームも大きなスクラムの流れとしてみると同じであったが，プランニングのタスク出し作業について差がある事が判明した（表 4）．

表 4 タスク出し比較

	Aチーム	Bチーム	Cチーム
スプリントプランニングの時間	2時間	1時間	3.5時間
タスク分割	未実施	実施	実施
PBI当たりのタスク数	—	4 タスク/PBI	18.5 タスク/PBI

2.3.1 A チーム

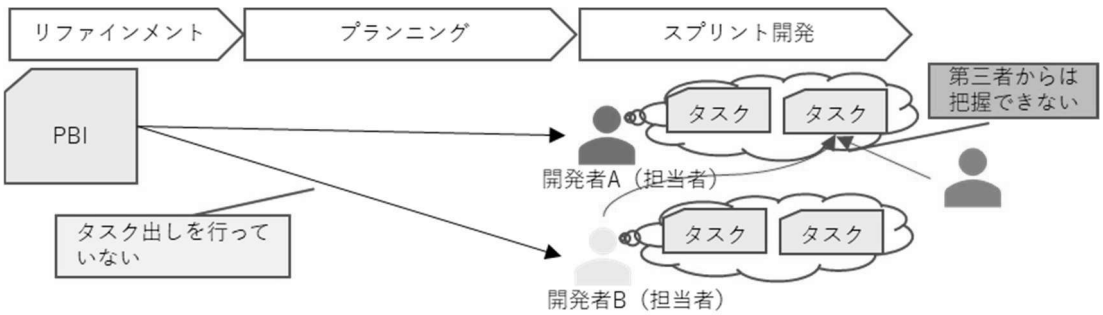

図 1　A チームにおけるタスク出しフロー

図１に示す通り，A チームではタスク出しを行っておらず，タスクは担当者の頭の中に漠然と存在する状態であり,担当者以外がタスク内容を把握するのが難しい状況であった．タスク自体が開発者の頭の中にあるため，担当者が気付いていないタスクの見落としの可能性がある．

2.3.2 B チーム

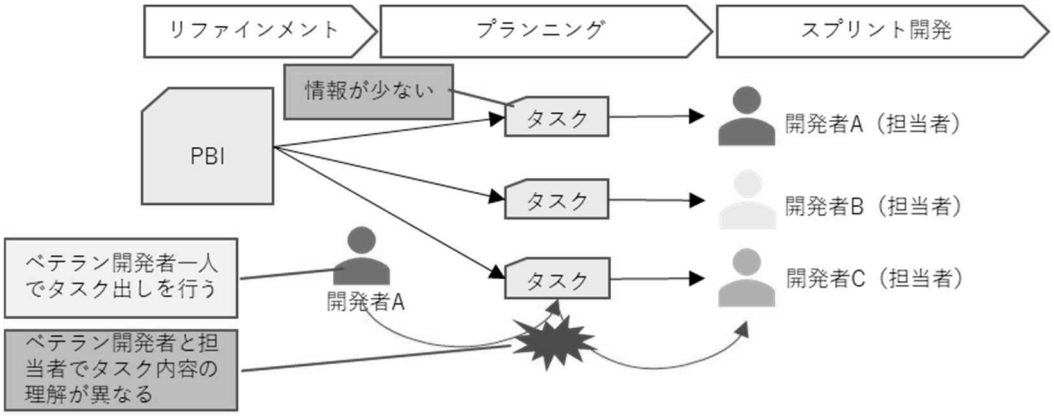

図２　B チームにおけるタスク出しフロー

図２に示す通りタスク出しはベテラン開発者が一人で行っており，洗い出されたタスクのチケットには,「調査」「実装」「評価」と書かれているだけとなっていたため，担当者をはじめとする第三者が詳細を把握するのが難しい状況であった．ベテラン開発者から実担当者への口頭説明が十分とは言えず，実担当者が内容を完全に理解できているとは言えない状況であった．そのため，ベテラン開発者が想定している内容と実担当者が実施している内容にずれが生じ，修正や追加作業といった手戻り作業が発生していた．

2.3.3 C チーム

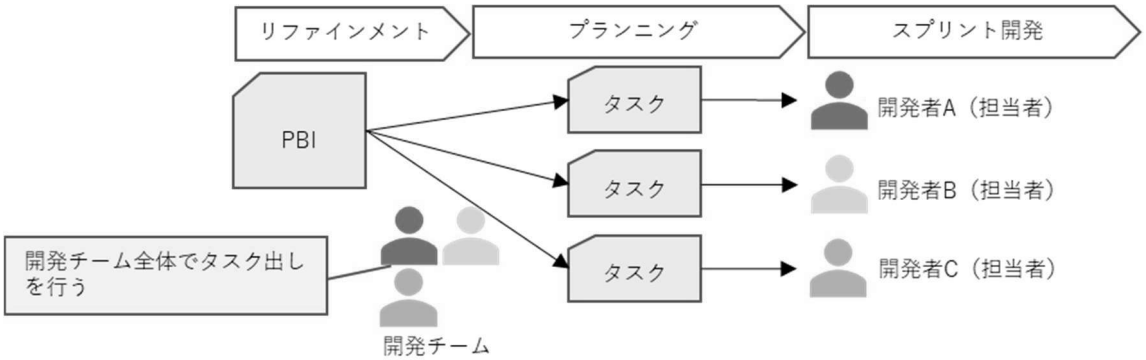

図３　C チームにおけるタスク出しフロー

図３に示す通り C チームの場合は，タスク出し自体が全員参加で行われており，実担当者もタスク内容は十分に理解できている状況であったため，タスク作業に関する認識ずれなどによる手戻り作業は発生していなかった．

表５　Ｃチームのタスク名(一例)

Cチーム（カテゴリ）	Cチーム（タスク）
実装	○○クラスに△△の処理を追加
	○○クラスのUnitTest実装
	○○テーブルに△△カラムを追加
	○○クラスで△△している処理を修正
評価	結合テスト作成、実施
	デザインレビュー
	コードレビュー
	負荷テスト実施

表５に示す通り，Ｃチームのタスク出しでは実装から評価までの具体的なタスクが洗い出されていた．

3. 提案

3.1 複数人でのタスクの洗い出し

Ａチームの計画遅延発生率が 63%となっており，Ｂチーム，Ｃチームと比較すると未完了な PBI が多い．Ａチームと，Ｂチームおよび Ｃチームとの相違点から次のことが実施できれば，PBI 未完了率を大きく削減できると仮説を立てた．

・スプリントプランニングでタスク出しまで行うこと
・タスク出しをチーム内の複数人（可能ならチーム内の全員）で実施し，チーム全体の共通認識にすること

3.2 タスク出し方法の改善

Ｂチームは手戻り発生率が 90%となっていたが，2.3.2 章に記載した通り開発チーム内での認識ずれが手戻りの要因となっていた．開発期間に入ってから手戻りが発生していることから，開発期間の作業内容となるタスクに着目し，問題が生じていない Ｃチームとタスク出しについて比較を行った．

表６　タスク出しの状況比較

	Bチーム	Cチーム
タスク出しを行う人	ベテラン１名	開発チーム全員、SM
タスク内容の合意形成	×	○

タスク出しの実施形態の違いに着目し，タスク出し方法の見直しにより，手戻り発生を抑制できると仮説を立てた．(表 6)

・タスク出しを開発チーム全員参加で行う
・開発チーム全員の共通理解になるようにタスク内容の共有を行う

4. 実験と評価

4.1 複数人でのタスクの洗い出し

Ａ チームでは今まで明示的なタスク出しを行っていなかったが，複数人でタスク出しを行う実験を行う．本実験により複数人でタスク出しを行う事による効果や変化について，検証を行う．

4.1.1 実験手順

・担当者(若手開発者)がタスクを事前に洗い出す
・複数人（担当者，担当外の開発者，ベテラン開発者）でタスク出しを実施する

・プランニングで担当者が事前に洗い出したタスクを説明し，参加者から質問を受ける
・参加者全員で検討を行い作業が必要と合意した物については，タスクを追加する

4.1.2 実験結果

担当者（若手開発者）が事前に洗い出したタスクは 5 個に対し，別の開発者やベテラン開発者を交えたタスク出しを行った結果，6 個のタスクが追加され 11 個のタスクに増えた（表 7）．若手開発者が事前に洗い出していたタスクを見ると開発作業については洗い出されていたが，部外との調整やソフト開発後の現場への展開といった開発作業以外のタスクが漏れていた．

表 7　実験前後のタスク内容（A チーム）

Aチーム（実験前）	Aチーム（実験後）
パラメータ変更の実装	検証日程の外注とのすり合わせ
パッチの作成	工場への指示票提出
インストーラの作成	工場での確認項目連絡
変更箇所のテスト項目の作成	対象機械が2台、もう1台への提供
検証環境準備	工場での組み立て日程確認
	出荷日と顧客への日程連絡

4.1.3 実験の考察

今回の実験から，複数人でのタスク出しを実施することでタスク漏れ防止に大きく改善効果がある事が確認できた．今回追加で見つかったタスクは，今までの開発ではスプリント開発期間中に追加作業として発生し，開発遅延に繋がっていたと考えられ，複数人でのタスク出しは計画遅延抑制の効果があると考えられる．

追加されたタスク内容を見ると，若手開発者だと経験が不足していて見落としがちな作業が追加されている．経験が浅い若手開発者にとって経験が豊富なベテラン開発者を交えたタスク出しが，気づきやタスク漏れ防止に対する効果が高いと考えられる．

またタスク漏れ防止以外の効果として，「メンバー間で状況に応じてタスクを入れ替えることもしやすくなる」の意見もあった．

4.2 タスク出し方法の改善

B チームではベテラン開発者が一人でタスク出しを行い若手開発者へ口頭で説明を行っていた．本実験ではファシリテーターが会議の雰囲気作りを行い，タスク内容が開発チーム内で共通理解を得られるようなタスク出しを行う．タスク内容をチーム内の共通理解とすることで，手戻り削減と開発遅延が解消する事を確認する．

4.2.1 実験手順

本実験では全員参加でタスク出しを行う予定だったが，実際にはベテラン開発者以外は知見が不足していたため，以下のような形でタスク出しを行った．

・チーム全員参加で PBI を議論し，ベテラン開発者が必要タスクを洗い出す
・ファシリテーターがベテラン開発者にタスク内容を質問し，説明を求める
・ファシリテーターが参加者全員の理解状況を確認する
・参加者全員で合意したタスクの内容をチケット化する

今までは若手開発者と認識ずれが生じていることから，経験や立場の違いも考慮し，タスク出しの雰囲気作りを行い，共通理解が得られるように以下の点に心がけた．

・若手開発者の代わりにファシリテーターがベテランエンジニアに質問を行う

・初歩的な質問も交えることで若手開発者が発言しやすい雰囲気作りを行う
・ファシリテーターが若手開発者の雰囲気等を観察し，理解度合いを推測する
・理解不足が推測される場合は若手開発者に問いかけを行い，質問しやすい雰囲気を作る

4.2.2 実験結果

今までは 1PBI につき 0.9 回の手戻りが発生していたが，今回の実験では手戻りは発生していない．（表 8）また，納期についても実験開始 1 スプリント目は 2 日前倒し，2 スプリント目は 1 日前倒しで開発を完了することができた．

表 8　実験前後のタスク数と手戻り発生数（チーム B）

	Bチーム（実験前）	Bチーム（実験後）
1スプリントのPBI数	2～3 PBI/Sprint	1～3 PBI/Sprint
PBI当たりのタスク数	4 タスク/PBI	10 タスク/PBI
手戻り発生件数	0.9 件/PBI	0 件/PBI

表 9　実験前後のタスク名（一例）

Bチーム（実験前）	Bチーム（実験後）
調査	使用できるOSSの調査
	○○仕様書の調査
実装	○○クラスの実装
	○○のエラーケースの実装
評価	連動確認評価
	デグレード確認評価

今までのタスクは「調査」「実装」といった抽象的な内容で若手開発者が理解できる状態ではなかった．今回の実験では，ファシリテーターがタスク出しを行う中で参加者が全員理解できるように質問などを行い，参加者全員が理解できていると判断し，タスク出しを完了した．結果として今までと比較し，具体的なタスク内容でタスク数も増える結果となった．（表 9）

4.2.3 実験の考察

今回の実験では 1PBI あたりのタスク数が 4 個から 10 個に増え，具体的な内容となっている．その中でファシリテーターの質問により実装箇所が「○○クラスの実装」として明確になったタスクがある．複数の実装手段がある中で「○○クラス」をベテラン開発者は暗黙知として選択していたが，今までは「実装」としか書かれておらず，「○○クラス」を採用する事が若手開発者には伝わっていなかった．

今回はなぜ「○○クラス」を選択したのかを更に質問し，複数の実装手段があること，実装手段による違いや影響範囲の違いを共有し，○○クラスを選択した理由などを聞き出すことができた．単にベテラン開発者が想定して「○○クラス」で実現するというだけでなく，ベテラン開発者が気にしているポイントも共通認識になることで，若手開発者が実際の実装時に詳細まで配慮して手戻りなく開発を行う事ができた．

タスク出しの際に，チーム全員で理解するまでコミュニケーションを取りやすい環境を提供し，チーム全員の共通認識になるまで情報を引き出し，対話を行いチーム内の共通理解を深め，チーム全員で開発を行う意識付けを行ったことによる効果と考えられる．

表 10　アンケート結果（チーム B）

No.	今回の施策についての感想
A	チケットの記載である程度作業の目星はつくようになった、がまだ何かしらの成果が上がっているようには見受けられない。 継続が必要かもしれない。
B	要求が書いてある点は理解できてよかった。 Goalやどこまでやればよいかの指標はあるとわかりやすい。
C	チケットを取る人に読んでもらうことでその人が迷うわないように作業できるか考えるため、頭の整理になる。 初めて作業する担当がチケットだけで作業を1つ終えれたのは満足だった。
D	チームの作業効率化の一環として取り組んで1回目としてはまずまずだったと個人的に思う。 今後も継続してPBI、TASKチケットの情報の充実を図っていきたい 全員がチケットを切れるようになれば自立したチームにまた一歩つながる気がする
E	他のチームと違い、チームとしての改善を図ろうという意思が感じられる。 継続してほしい。

表 10 は実験後に取ったアンケート結果である．回答者 A は少し懐疑的なコメントを書いているが，実験後も自主的にタスクチケットに詳細な作業内容を書いており，一定の効果が実感できていると推測できる．今回の実験では表 10 のアンケート結果の通り，開発チームメンバー全員から概ね好評であった．

今回の実際の作業分担を見ると，今まではベテラン開発者が一人で行っていたタスクを複数の開発者で分担して平行作業で進めることができていた．一人の優秀なベテラン開発者中心で作業していたものが，若手開発者中心のチーム全員で分担して作業を進めることで納期短縮に繋がっている．

5　まとめ

本研究でチーム全員でのタスク洗い出しを実施しチームの共通理解を深めた結果，計画遅延の解消が確認できた．

遅延の発生しているチームでは明示的なタスク出しを行っていなかったため，タスク漏れによる計画遅延が生じていた．複数人でのタスク出しにより，特に若手開発者に経験が無いためこれまで出せなかった開発後工程のようなタスクが洗い出せるようになり計画遅延が防げるようになった．

一方で，全員で集まってタスク出しを行っているチームでも，タスクの内容が共通理解になっていないケースがあった．ベテラン開発者が理解できる粒度でタスク出しを行っていたため，若手開発者からみるとタスクの抽象度が高く，質問しづらい雰囲気もあり理解不足のまま開発を行っていた．今回，開発経験が浅い開発者の理解促進を行うために，質問しやすい雰囲気作りやファシリテーターがフォローすることで，対話が活発に行えるようになり，共通理解が実現できたと考える．これまで受け身だった若手開発者が実験後も積極的に質問をようになり，有効性を感じることができた．

6　参考文献

[1]アジャイルソフトウェア開発宣言
https://agilemanifesto.org/iso/ja/manifesto.html

[2]スクラムガイド 2020
https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-Japanese.pdf