

スコープが固定された
請負契約でも出来た
アジャイル開発



目次

1. 会社紹介
2. 自己紹介
3. プロジェクトの始まり
4. プロジェクト準備
5. プロダクトバックログアイテムの優先順位付け
6. 日々の作業
7. 品質の確保
8. スプリントレビュー
9. プロジェクト終了
10. 振り返り

中菱エンジニアリング株式会社

本社 名古屋市中村区
社員数 954名(2016年4月現在)
事業内容 航空機・宇宙機器、産業機械、
冷凍・空調機器、制御機器、
電子機器、コンピュータソフト
ウェア等の設計・実験・計測

三菱重工業(株)のグループ会社
三菱重工の名古屋地区の製品に対応



自己紹介

ふるかわ よしひろ

古川 剛啓

中菱エンジニアリング(株)
航空ソフトウェア設計室
グループマネージャ



UMTP L3モデラー

UMTP アジャイル開発部会メンバ

OMG Advanced

認定スクラムプロダクトオーナー

認定スクラムマスター

UMTP
L3 MODELER



プロジェクトの始まり

認定スクラムマスターを取得して3年間
「SCRUM良いよ」といい続けた結果・・・

Boss 「今度の開発、納期が厳しいからSCRUMでやるんで宜しく！」

古川 「えっ！（°д°）」

Boss 「SCRUMでやると早くできるんでしょ？」

古川 「結果、早くなることはありますが・・・」

Boss 「ほら、早くできるじゃん。じゃあ、やって。」

古川 「はあ・・・」

Boss 「あっ！ちなみに、今までだったら3年のプロジェクトだけど、今回は2年しかないから。」



動機がちょっとアレだけど、まっいいか



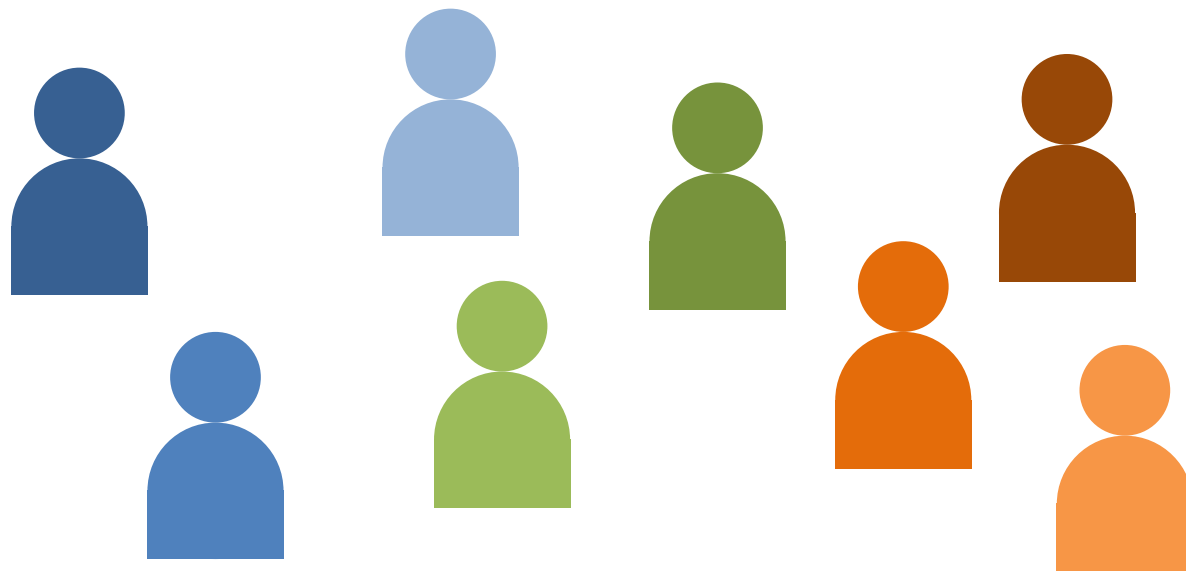
プロダクトの分野としては組込み。
大きなシステムの一部です。
詳細は、察して下さい。

Courtesy of Mitsubishi Heavy Industries, Ltd.

プロジェクト準備

「メンバ招集」と言ってもどこからか人を連れてくるわけではない。グループ間の異動があるくらい。基本、今までのメンバ。メンバ全員が、

SCRUM？なにそれ？美味しいの？



Scrum Boot Camp



古川 剛啓

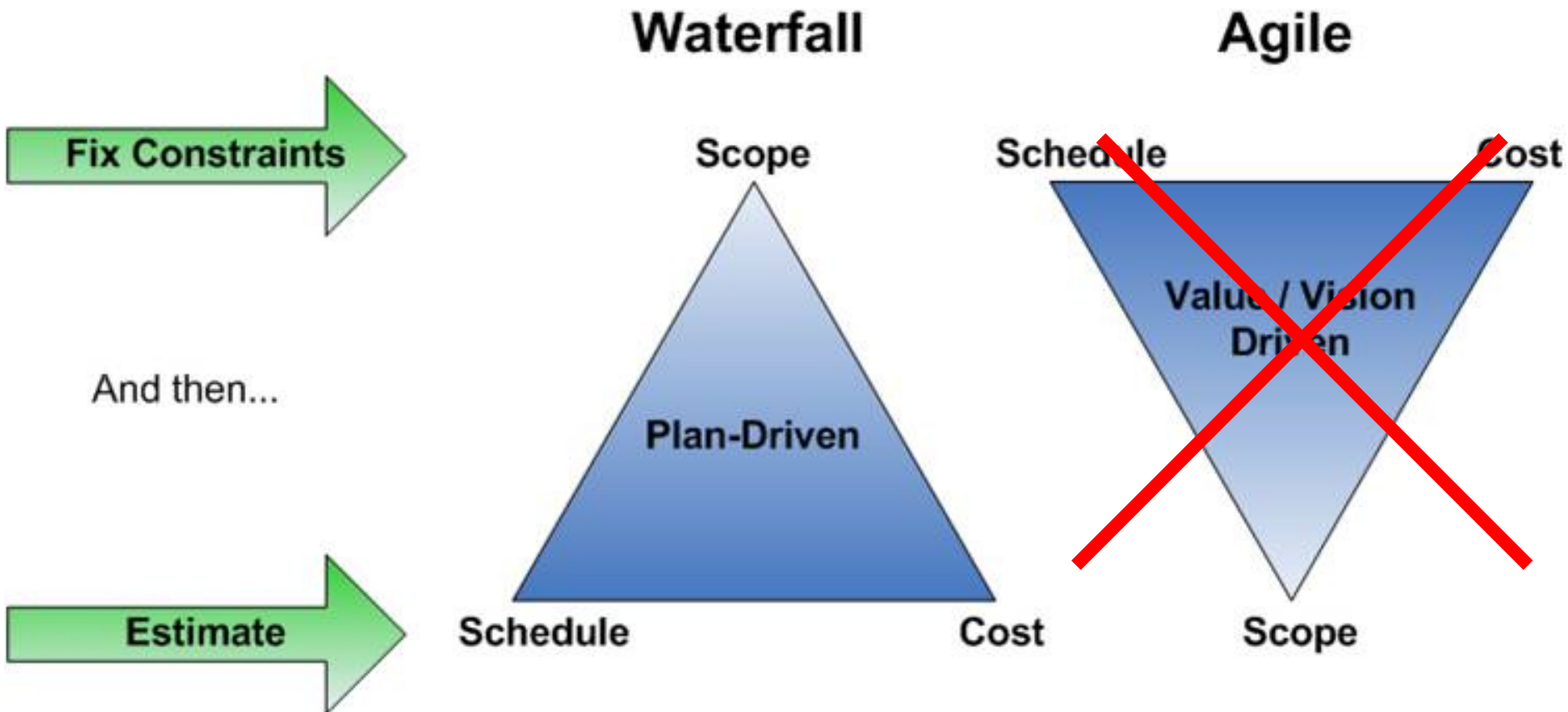
UMTP
L3 MODELER



教えることのポイント

- SCRUMって何？
- リスクは直ぐ解消しろ！
- 目指せ！自己組織化！

通常のAgileとは異なることを明確に伝える



プロジェクトの開始



プロダクトオーナー
製品に対して責任をもち機能に優先順位を付ける



スクラムマスター
スクラムプロセスがうまくいくようにする。
外部からチームを守る



チーム (6±3人)
プロダクトの開発を行う。
製品の成功に向けて最大限の努力をコミットする



ステークホルダー
製品の利用者、出資者、管理職などの利害関係者。鶏と称す



プロダクトバックログ
製品の機能をストーリー形式で記載
プロダクトオーナーが優先順位を付け、プランニングポーカーで相対見積もり。
項目の追加はいつでも自由だが実施有無や優先順位はPOが決める。



Doneの定義
何をもって「完了」とするかを定義したリスト



デイリースクラム
毎日チームが以下の3つの質問に答える
・昨日やったこと
・今日やること
・困っていること



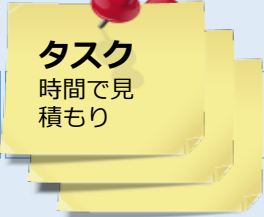
バーンダウンチャート
スプリントタスクの「推定残り時間」を更新してグラフにプロットする



スプリント
最大4週間までのタイムボックス
各スプリントの長さは同一。この間は外部からの変更を受け入れない



スプリント計画会議
プロダクトバックログを再度分析・評価し、そのスプリントで開発するプロダクトバックログアイテムを選択する。また選択した項目をタスクにばらす



タスク
時間で見積もり



毎日の繰り返し



スプリントレビュー
スプリント中の成果である動作するソフトウェアをデモする



ふりかえり
スプリントの中での改善事項を話し合い次に繋げる



出荷可能な製品の増分

スプリントバックログ
そのスプリント期間中に行うタスクのリスト

複数回スプリントを繰り返す



プロダクトオーナー
製品に対して責任をもち機能に優先順位を付ける



スクラムマスター
スクラムプロセスがうまくいくようにする。
外部からチームを守る



チーム (6±3人)
プロダクトの開発を行う。
製品の成功に向けて最大限の努力をコミットする



ステークホルダー
製品の利用者、出資者、管理職などの利害関係者。鶏と称す

プロダクトバックログ
製品の機能をストーリー形式で記載
プロダクトオーナーが優先順位を付け、プランニングポーカーで相対見積もり。
項目の追加はいつでも自由だが実施有無や優先順位はPOが決める。



Doneの定義
何をもって「完了」とするかを定義したリスト



デイリースクラム

毎日チームが以下の3つの質問に答える
・昨日やったこと
・今日やること
・困っていること



バーンダウンチャート

スプリントタスクの「推定残り時間」を更新してグラフにプロットする

スプリント計画会議
プロダクトバックログを再度分析・評価し、そのスプリントで開発するプロダクトバックログアイテムを選択する。また選択した項目をタスクにばらす

スプリント
最大4週間までのタイムボックス
各スプリントの長さは同一。この間は外部からの変更を受け入れない



タスク
時間で見積もり

毎日の繰り返し



スプリントレビュー
スプリント中の成果である動作するソフトウェアをデモする



ふりかえり
スプリントの中での改善事項を話し合い次に繋げる



出荷可能な製品の増分

複数回スプリントを繰り返す

プロダクトバックログアイテムの 優先順位付け

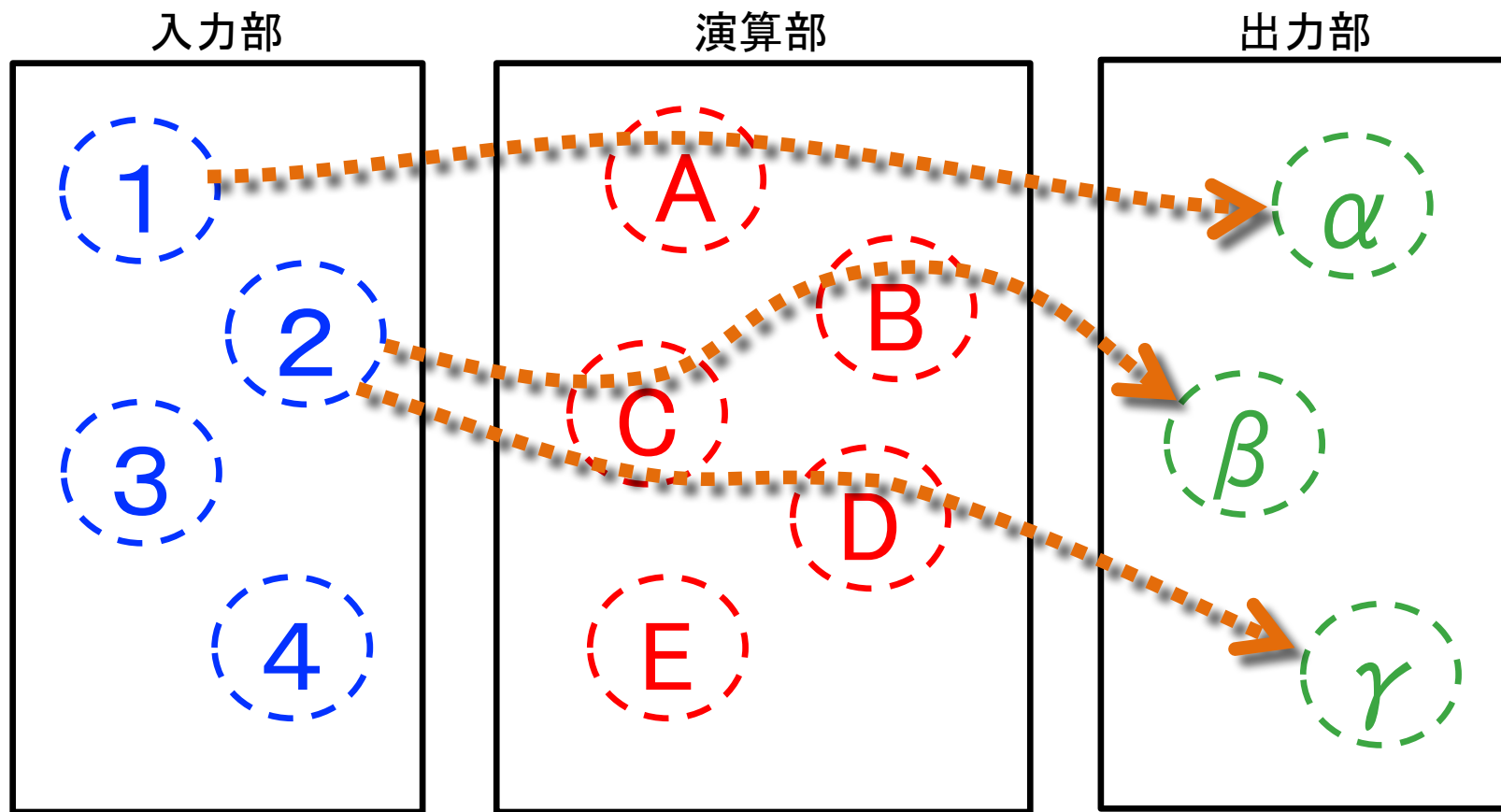
ユーザーストーリーとは、実現したいと思っているフィーチャーを簡潔に示したもので、短い文章として表したものの

メリット

- ✓ 作成に時間が掛からない
- ✓ フィーチャーの本質を捉える事ができる
- ✓ ユースケースの粒度によらない



<http://www.flickr.com/photos/psd/8591351239/>



 フィーチャー

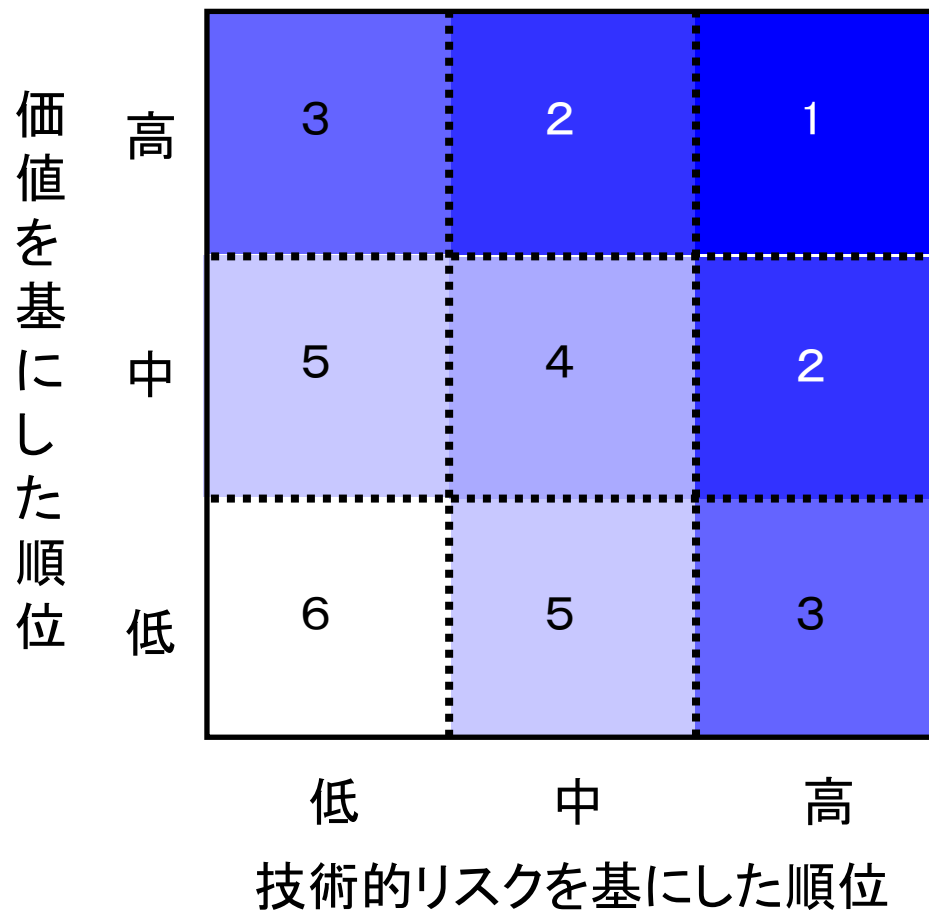
リスクの洗い出し



優先順位付け

- ✓ 技術的リスクを考慮する
 - 開発チームの意見
- ✓ ユースケースの価値を考える
 - ユーザーストーリーマッピング
 - 狩野モデル
- ✓ フィーチャーの関連を考慮する
 - ユースケース図

価値を基にした順位と技術的リスクをマッピングする



マスの中の数字は優先度を表す
数字が小さいほど優先度が高い

フロアマットを活用

フィーチャーを書いた紙をフロア上に
並べて優先順位付け

日々の作業



プロダクトオーナー
製品に対して責任をもち機能に優先順位を付ける



スクラムマスター
スクラムプロセスがうまくいくようにする。
外部からチームを守る



チーム (6±3人)
製品の開発を行う。
製品の成功に向けて最大限の努力をコミットする



ステークホルダー
製品の利用者、出資者、管理職などの利害関係者。鶏と称す



プロダクトバックログ
製品の機能をストーリー形式で記載
プロダクトオーナーが優先順位を付け、プランニングポーカーで相対見積もり。
項目の追加はいつでも自由だが実施有無や優先順位はPOが決める。



Doneの定義
何をもって「完了」とするかを定義したリスト



デイリースクラム

毎日チームが以下の3つの質問に答える
・昨日やったこと
・今日やること
・困っていること



バーンダウンチャート

スプリントタスクの「推定残り時間」を更新してグラフにプロットする



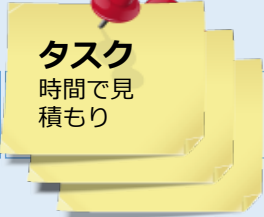
スプリント

最大4週間までのタイムボックス
各スプリントの長さは同一。この間は外部からの変更を受け入れない



スプリント計画会議

プロダクトバックログを再度分析・評価し、そのスプリントで開発するプロダクトバックログアイテムを選択する。また選択した項目をタスクにばらす



タスク
時間で見積もり



毎日の繰り返し



スプリントレビュー

スプリント中の成果である動作するソフトウェアをデモする



ふりかえり

スプリントの中での改善事項を話し合い次に繋げる



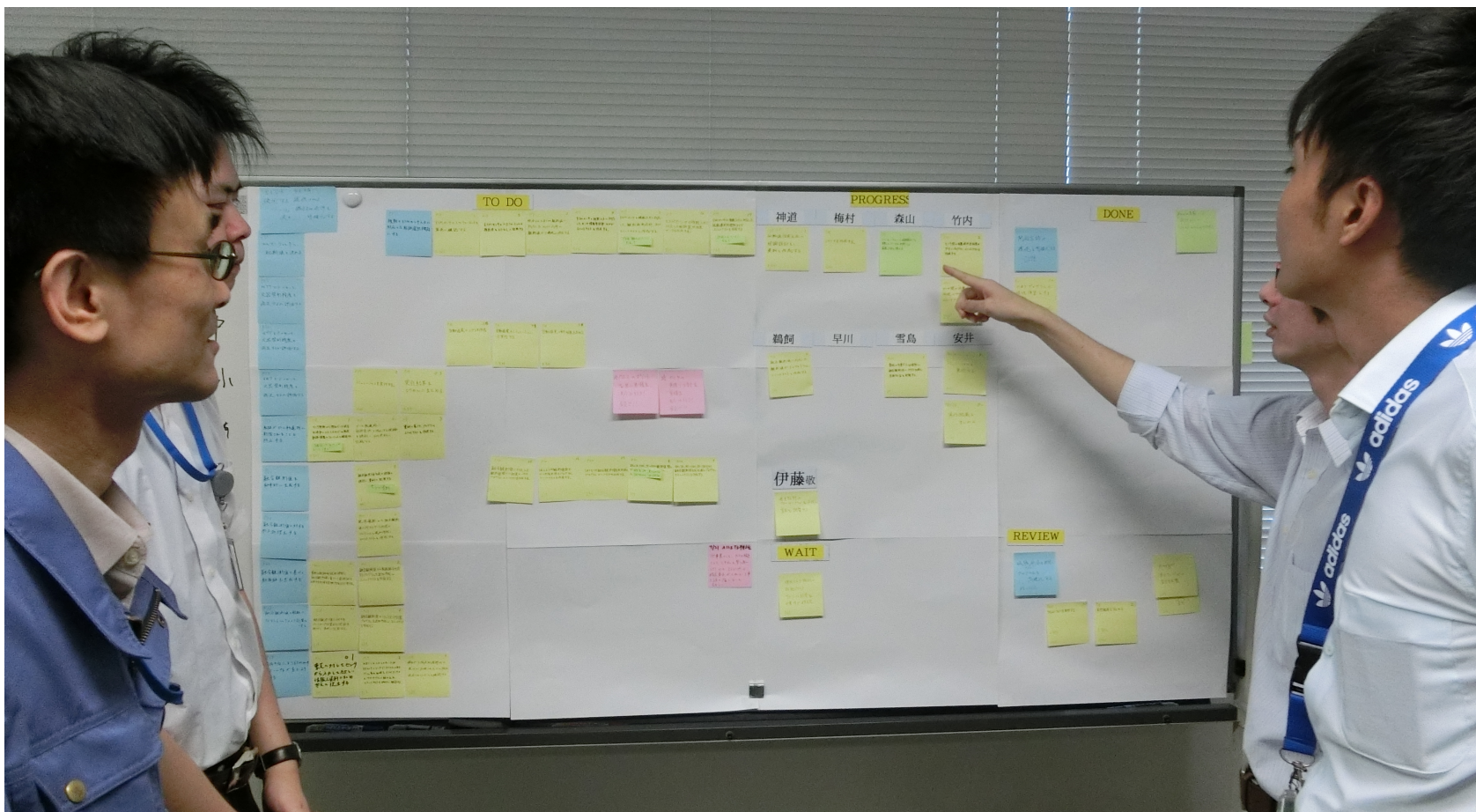
出荷可能な製品の増分

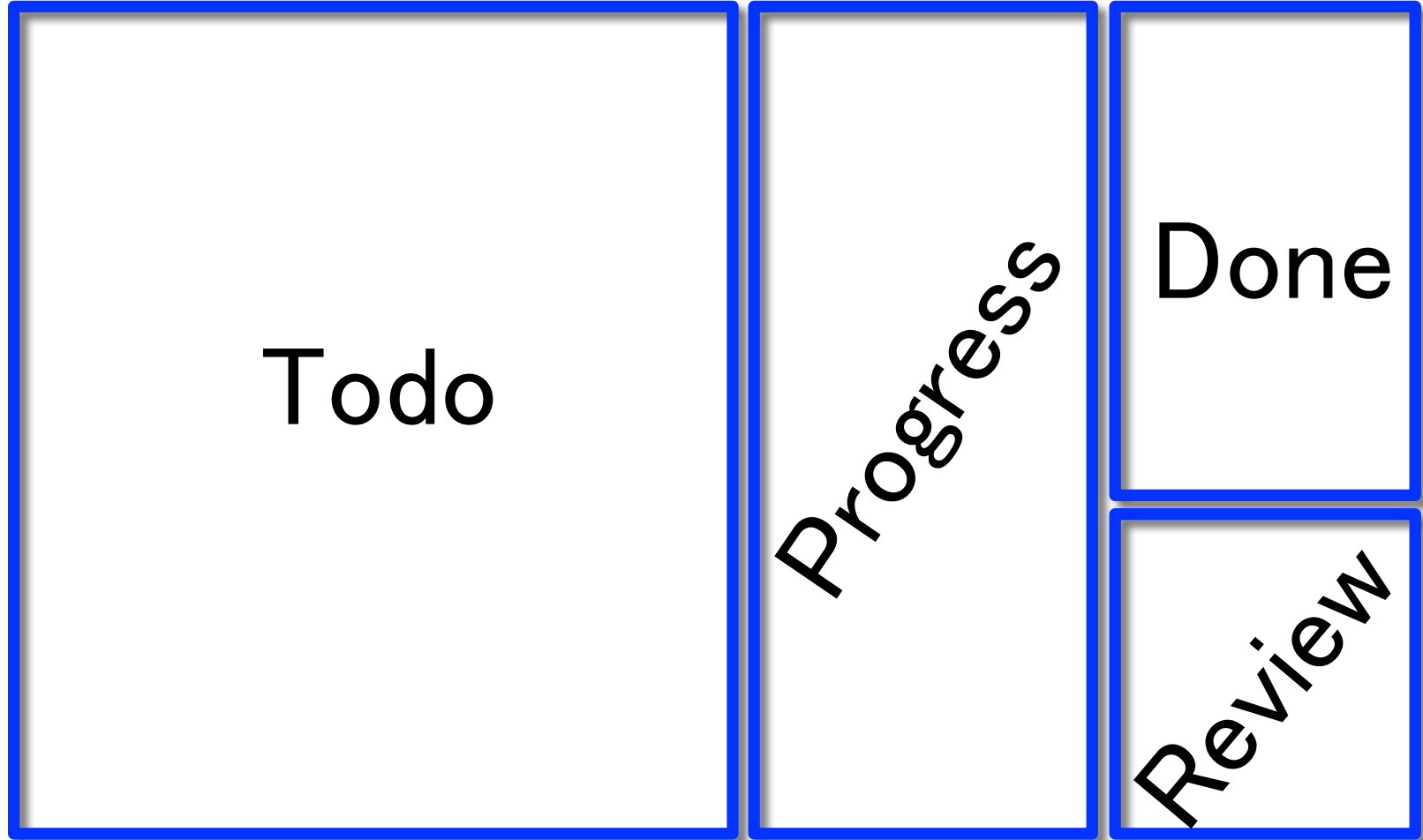
スプリントバックログ

そのスプリント期間中に行うタスクのリスト

複数回スプリントを繰り返す

タスクボードでタスクを見える化 タスクボードの前でデイリースクラム





進捗管理と言ったらやっぱり・・・

スコープもスケジュールも
決まっているので

[Redacted]

[Redacted]

[Redacted]

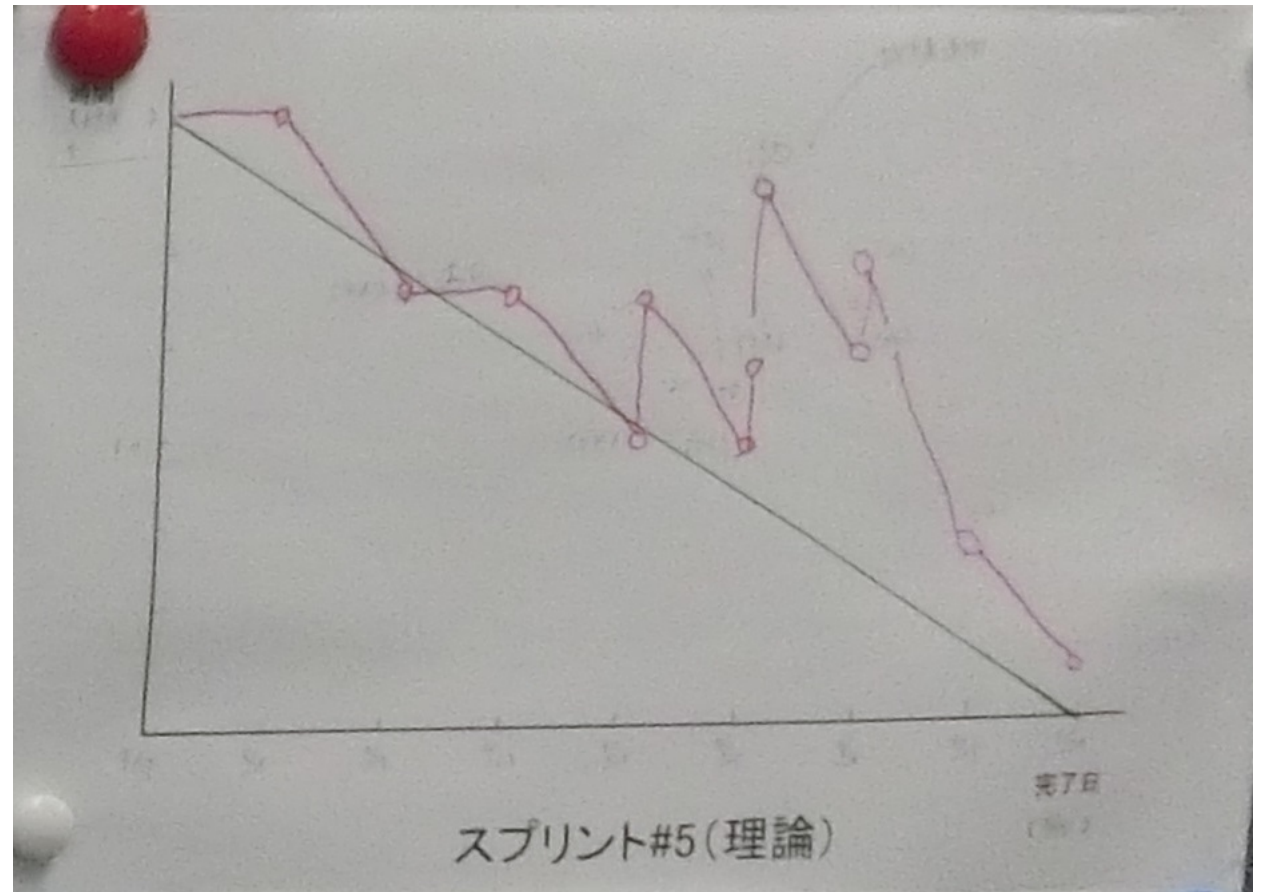
[Redacted]

[Redacted]

特許権

そんなわけないがや！

やっぱり、バーンダウンチャート



教科書にあるような事をやってきたところ・・・

予定通り失敗した(笑)

全員で失敗から学ぶ

- ✓ 自分達の実力
- ✓ 失敗の原因
- ✓ 誰が何をできるのか / 出来ないのか

- ✓ タスクの粒度が荒い
- ✓ 誰が何時間働けるか分からない
- ✓ タスクの進捗状況が分かり難い

8時間のタスクは氷山のようなもの

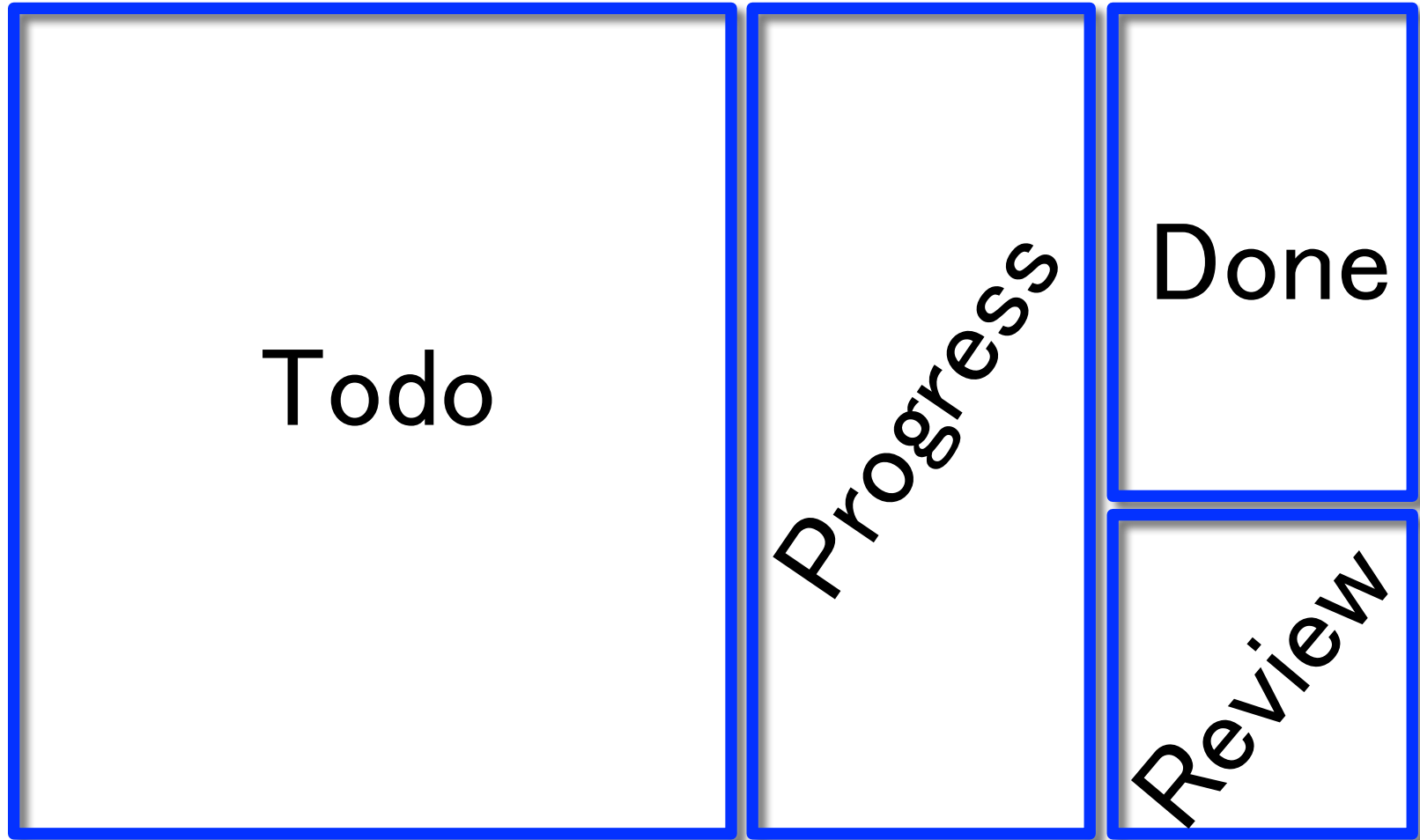


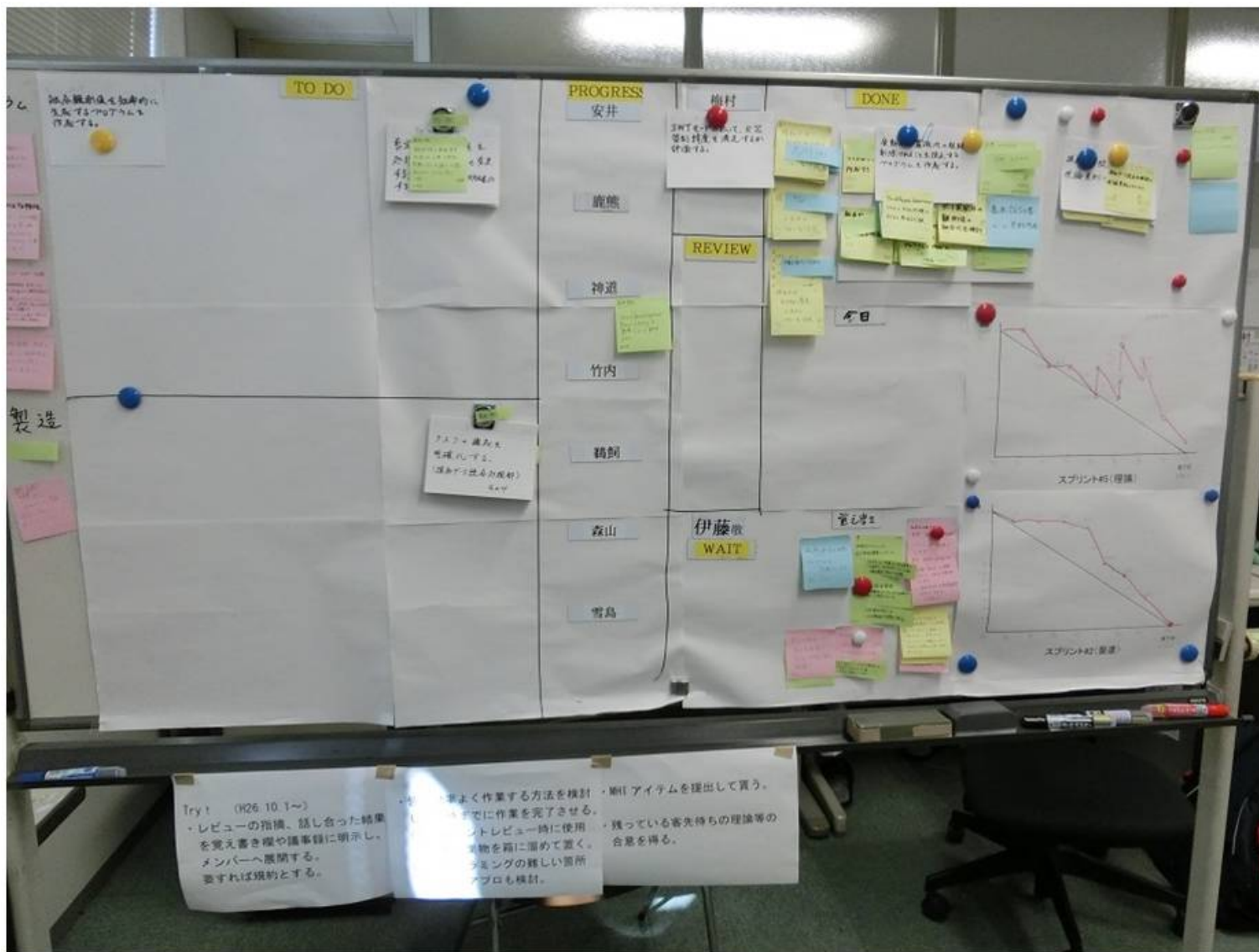
A scenic landscape featuring snow-capped mountains in the background, a bright sun setting over a body of water, and large, white icebergs scattered on a dark, rocky shore in the foreground. The sky is filled with soft, colorful clouds.

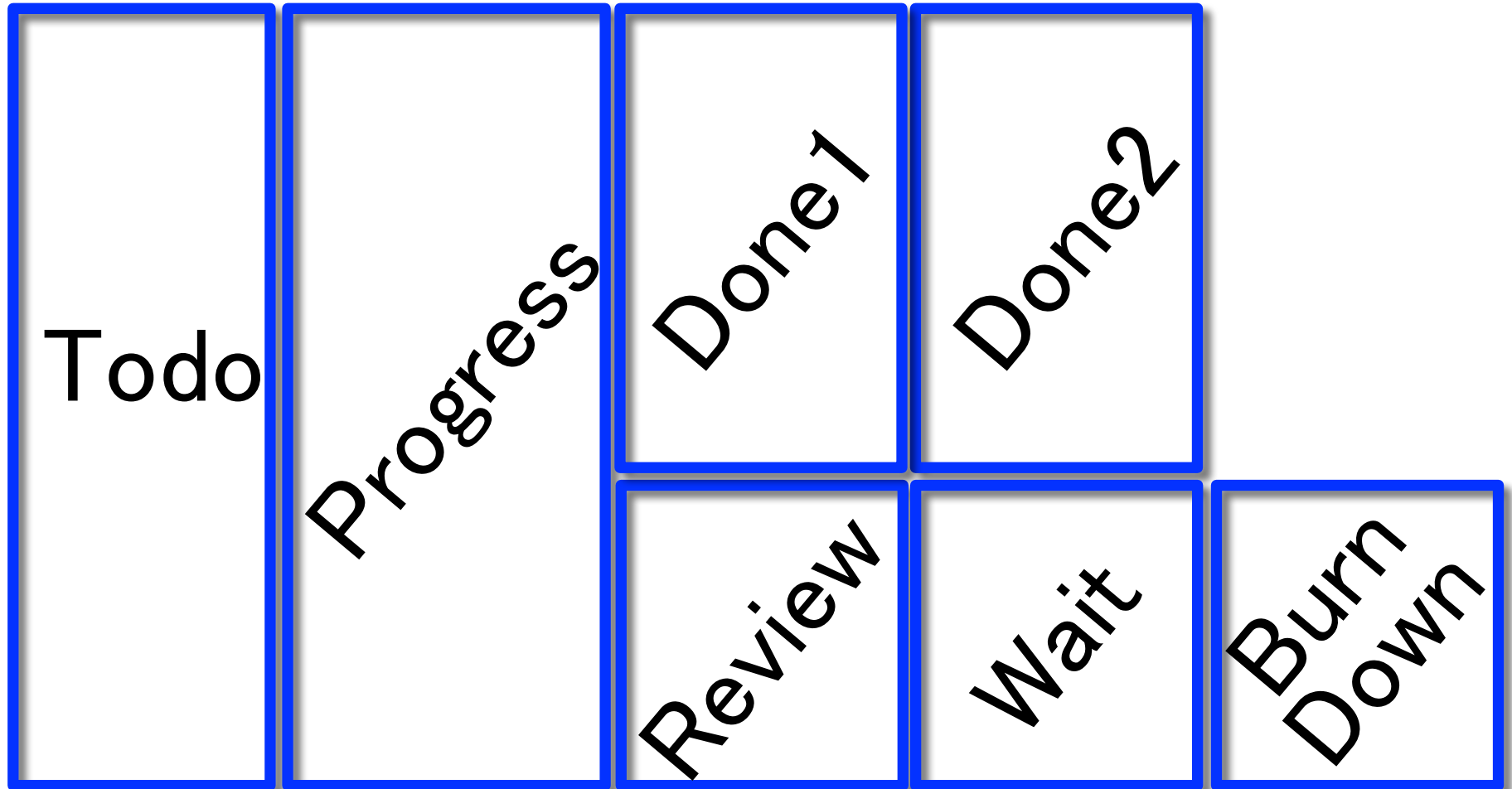
全体が見える粒度へ細分化

チームに貢献できる時間を見える化

氏名	1	2	3	4	...	10
A	6	7	6	休	...	8
B	6	8	休	8	...	7
C	5	8	7	7	...	6
D	7	6	4	7	...	7







デイリースクラムで共有された問題点や要確認事項も
タスクとして見える化。
直ぐ解消する様にルール決め。

解決していない問題がありますね。どうしますか？

客先のフォローはしましたか？

ガミガミ・・・ガミガミ・・・



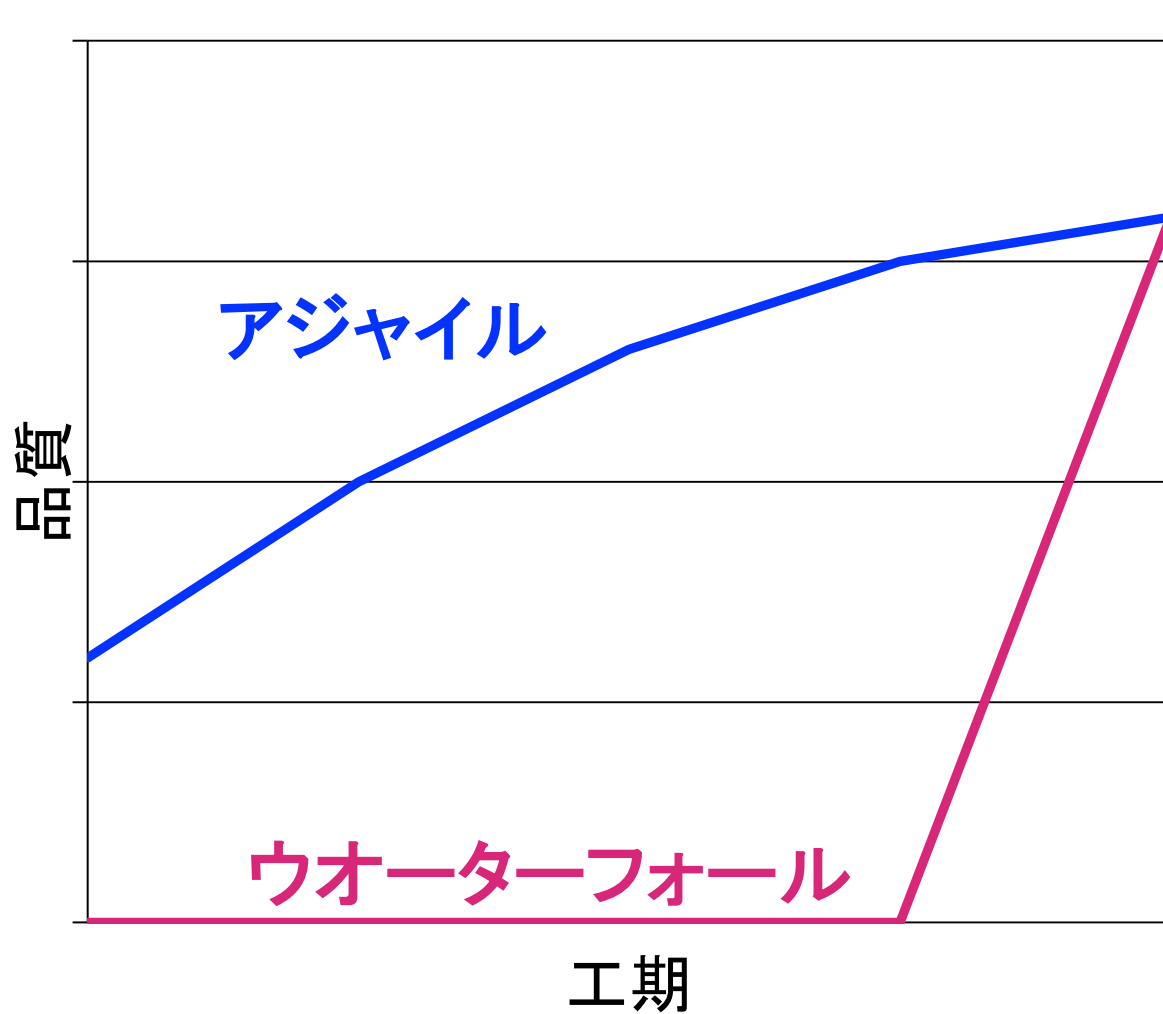
スクラムマスター

デイリースタラムで助けを求めるようになった
ナレジットランスファーにペアプロを始めた



salãodocarro.com

品質の確保



継続的なテスト
による品質の
積み上げ

品質の確定は
プロジェクト
後期になる

- グローバル変数の禁止
 - ✓ ヘッダファイルには、変数を定義しない
- publicな関数と、privateな関数を明確に分ける
 - ✓ publicな関数を定義したヘッダファイルとprivateな関数(static関数)を定義したヘッダファイルに分ける

TEST FIRST

先天下驅動開苑

働き者のJenkinsさん



お任せください

- ✓ ユニットテスト
- ✓ テストカバレッジ
- ✓ 静的解析
- ✓ 重複コードチェック
- ✓ 未解決タスクチェック
- ✓ ソフトウェア結合テスト
- ✓ 動的解析
- ✓ ビルド

プロジェクト XXXXXXXXXX

絶対プロジェクト名: XXXXXXXXXX



ワークスペース



最新成功ビルドの成果物

XXXXXXXXXX.outTest.xml 9.86 KB 参照



変更履歴



最新のテスト結果 (全て成功)



最新のテスト結果 (全て成功)

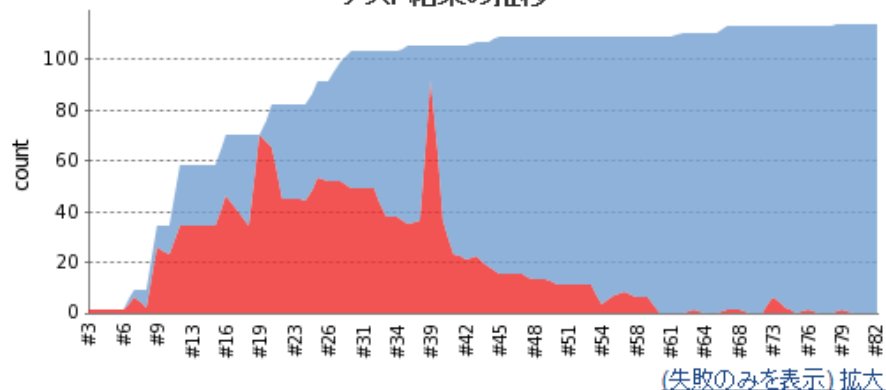
永続リンク

- [最新のビルド \(#82\), 5ヶ月 16 日前](#)
- [最新の安定ビルド \(#82\), 5ヶ月 16 日前](#)
- [最新の成功ビルド \(#82\), 5ヶ月 16 日前](#)
- [最新の失敗ビルド \(#79\), 9ヶ月 21 日前](#)
- [最新の不成功ビルド \(#79\), 9ヶ月 21 日前](#)

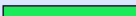








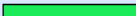





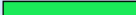



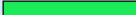





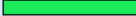



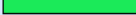



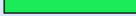
説明を記入

プロジェクトの無効化

テスト結果の推移



lcovによるテストカバレッジの計測

Filename	Line Coverage		Functions		
[Redacted]		100.0 %	13 / 13	100.0 %	2 / 2
[Redacted]		100.0 %	3 / 3	100.0 %	1 / 1
[Redacted]		100.0 %	226 / 226	100.0 %	38 / 38
[Redacted]		100.0 %	25 / 25	100.0 %	4 / 4
[Redacted]		100.0 %	5 / 5	100.0 %	1 / 1
[Redacted]		100.0 %	84 / 84	100.0 %	8 / 8
[Redacted]		100.0 %	20 / 20	100.0 %	3 / 3
[Redacted]		100.0 %	18 / 18	100.0 %	2 / 2
[Redacted]		100.0 %	22 / 22	100.0 %	2 / 2
[Redacted]		100.0 %	10 / 10	100.0 %	1 / 1
[Redacted]		100.0 %	4 / 4	100.0 %	1 / 1
[Redacted]		100.0 %	11 / 11	100.0 %	1 / 1
[Redacted]		100.0 %	73 / 73	100.0 %	7 / 7
[Redacted]		100.0 %	69 / 69	100.0 %	6 / 6
[Redacted]		100.0 %	162 / 162	100.0 %	5 / 5
[Redacted]		100.0 %	148 / 148	100.0 %	15 / 15
[Redacted]		100.0 %	6 / 6	100.0 %	1 / 1
[Redacted]		100.0 %	21 / 21	100.0 %	1 / 1
[Redacted]		100.0 %	149 / 149	100.0 %	23 / 23
[Redacted]		100.0 %	4 / 4	100.0 %	1 / 1
[Redacted]		100.0 %	26 / 26	100.0 %	7 / 7
[Redacted]		100.0 %	55 / 55	100.0 %	9 / 9
[Redacted]		100.0 %	140 / 140	100.0 %	47 / 47
[Redacted]		100.0 %	20 / 20	100.0 %	2 / 2
[Redacted]		100.0 %	135 / 135	100.0 %	11 / 11
[Redacted]		100.0 %	249 / 249	100.0 %	19 / 19
[Redacted]		100.0 %	47 / 47	100.0 %	4 / 4
[Redacted]		100.0 %	4 / 4	100.0 %	1 / 1
[Redacted]		100.0 %	15 / 15	100.0 %	6 / 6
[Redacted]		100.0 %	43 / 43	100.0 %	15 / 15
[Redacted]		100.0 %	90 / 90	100.0 %	7 / 7
[Redacted]		100.0 %	12 / 12	100.0 %	1 / 1
[Redacted]		100.0 %	28 / 28	100.0 %	3 / 3
[Redacted]		100.0 %	56 / 56	100.0 %	3 / 3

重複コード

警告の推移

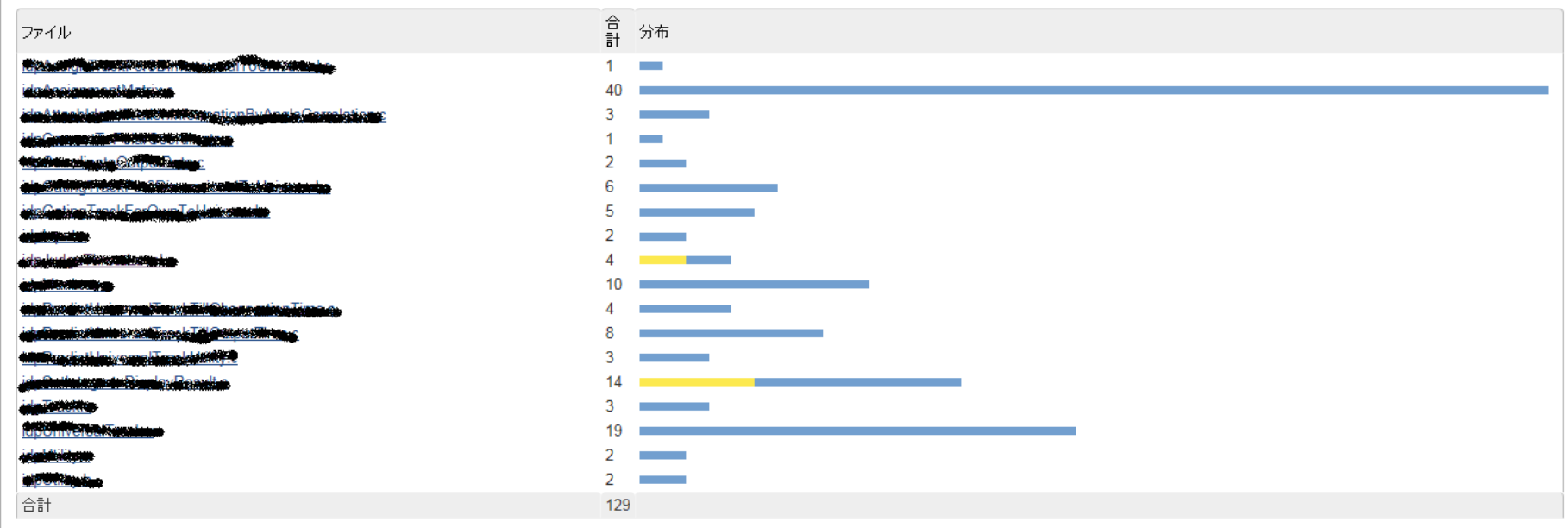
警告の合計	新規の警告	修正された警告
129	129	0

要約

合計	重要度 High	重要度 Normal	重要度 Low
129	0	1	122

Details

[ファイル](#)
[警告](#)
[詳細](#)
[新規](#)
[Normal](#)
[Low](#)



未解決タスク

未解決タスクの推移





未解決タスクの合計	新規の未解決タスク	修正された未解決タスク
181	0	0

要約

合計	重要度 High	重要度 Normal	重要度 Low
181	1	26	154

Details

フォルダー	ファイル	タイプ	警告	詳細	High	Normal	Low
ソースフォルダー							
...							
...							
...							
...							
合計							

ソースフォルダー	合計	分布
...	17	
...	59	
...	64	
...	41	
合計	181	

プロジェクト █████_cppcheck

絶対プロジェクト名: █████_cppcheck



[ワークスペース](#)



[変更履歴](#)



[Cppcheck Results](#)

Severity	Count	Delta
Error	0	
Warning	0	
Style	123	
Performance	0	
Portability	0	
Information	1	
No category	0	
Total	124	

- ✓ 正しく作られた部品を組み合わせる
- ✓ 製品が完成するまで継続的にチェックする





プロダクトオーナー
製品に対して責任をもち機能に優先順位を付ける



スクラムマスター
スクラムプロセスがうまくいくようにする。
外部からチームを守る



チーム (6±3人)
製品の開発を行う。
製品の成功に向けて最大限の努力をコミットする



ステークホルダー
製品の利用者、出資者、管理職などの利害関係者。鶏と称す



プロダクトバックログ
製品の機能をストーリー形式で記載
プロダクトオーナーが優先順位を付け、プランニングポーカーで相対見積もり。
項目の追加はいつでも自由だが実施有無や優先順位はPOが決める。



Doneの定義
何をもって「完了」とするかを定義したリスト



デイリースクラム
毎日チームが以下の3つの質問に答える
・昨日やったこと
・今日やること
・困っていること

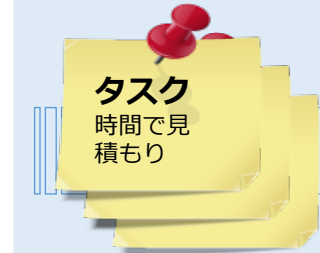


バーンダウンチャート
スプリントタスクの「推定残り時間」を更新してグラフにプロットする



スプリント
最大4週間までのタイムボックス
各スプリントの長さは同一。この間は外部からの変更を受け入れない

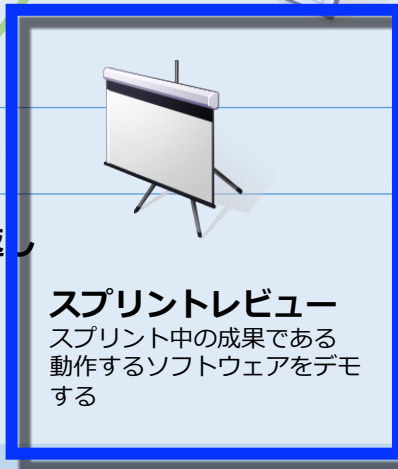
スプリント計画会議
プロダクトバックログを再度分析・評価し、そのスプリントで開発するプロダクトバックログアイテムを選択する。また選択した項目をタスクにばらす



タスク
時間で見積もり



毎日の繰り返し



スプリントレビュー
スプリント中の成果である動作するソフトウェアをデモする



ふりかえり
スプリントの中での改善事項を話し合い次に繋げる



出荷可能な製品の増分

複数回スプリントを繰り返す

スプリントレビュー

大きなシステムの一部となる組込みプログラムなので製品を動作を確認するレビューは不可能。

プログラムは、開発環境上で実行。

入出力の期待値と実行結果を比較して動作を確認。

入出力の期待値は、事前に客先に確認してもらう。

スプリントレビューには、客先も同席。

レビュー済みの機能も含め、全ての受け入れテストがパスしていることを確認。

自動化された受け入れテスト

プロジェクト UAT

絶対プロジェクト名: ~~XXXXXXXXXX~~

 説明を記入

プロジェクトの無効化



ワークスペース



変更履歴



最新のテスト結果 (全て成功)



最新のテスト結果 (全て成功)

上流プロジェクト

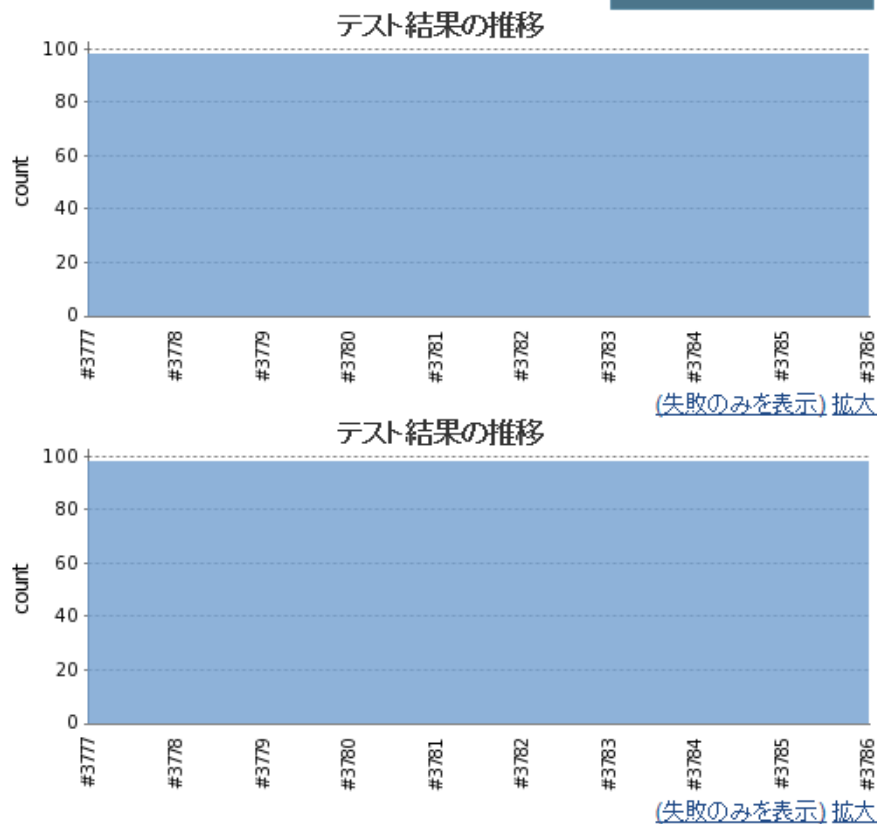
 ~~XXXXXXXXXX~~

下流プロジェクト

 ~~XXXXXXXXXX~~

永続リンク

- 最新のビルド (#3786). 9ヶ月 8日 前
- 最新の安定ビルド (#3786). 9ヶ月 8日 前
- 最新の成功ビルド (#3786). 9ヶ月 8日 前





プロダクトオーナー
製品に対して責任をもち機能に優先順位を付ける



スクラムマスター
スクラムプロセスがうまくいくようにする。
外部からチームを守る



チーム (6±3人)
製品の開発を行う。
製品の成功に向けて最大限の努力をコミットする



ステークホルダー
製品の利用者、出資者、管理職などの利害関係者。鶏と称す



プロダクトバックログ
製品の機能をストーリー形式で記載
プロダクトオーナーが優先順位を付け、プランニングポーカーで相対見積もり。
項目の追加はいつでも自由だが実施有無や優先順位はPOが決める。



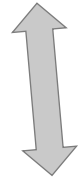
デイリースクラム
毎日チームが以下の3つの質問に答える
・昨日やったこと
・今日やること
・困っていること



バーンダウンチャート
スプリントタスクの「推定残り時間」を更新してグラフにプロットする



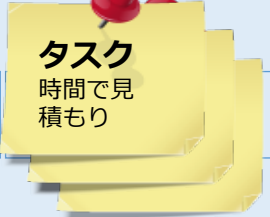
Doneの定義
何をもって「完了」とするかを定義したリスト



スプリント計画会議
プロダクトバックログを再度分析・評価し、そのスプリントで開発するプロダクトバックログアイテムを選択する。また選択した項目をタスクにばらす



スプリント
最大4週間までのタイムボックス
各スプリントの長さは同一。この間は外部からの変更を受け入れない



タスク
時間で見積もり



毎日の繰り返し



スプリントレビュー
スプリント中の成果である動作するソフトウェアをデモする



ふりかえり
スプリントの中での改善事項を話し合い次に繋げる



出荷可能な製品の増分

スプリントバックログ
そのスプリント期間中に行うタスクのリスト

複数回スプリントを繰り返す

振り返り時に、初めてSCRUMに取り組んだチームは、生産性を2倍にすることも可能なことを聞いたことをチームに伝える。生産性を2倍(タスクに掛かる時間を半分)にすることをチームに要求し、その方法をチームに考えさせる。

このチームはタスクの見積り時間を半分にした。

→ 流石に全てを半分にはできなかったが、それでも多くのタスクが半分の時間でできるようになった。

プロジェクト終了

従来の2／3の開発期間で完了
コストも問題なし
品質も今まで以上に高品質

振り返り

1. プロジェクトが大きかった

期間が長いために、失敗しても取り戻せるチャンスが多い。
気持ち的に余裕がある。

SCRUMを始めてから慣れるまで3ヶ月は掛かる。

2. メンバが良かった

優秀な人材を集めたわけではない。むしろ、若手ばかり。
上手く行った理由は、基本みんな喋りたがり。
人に教えるを請うことも、教えることも好き。

3. 改善意欲満載

良くも悪くも自分達がやりやすいように変えようとする。
SCRUMのコンテキストから外れなければ許可。外れるようなら話し合い。SMがコントロール

1. プロジェクトが小さい

初めてのSCRUMなのにプロジェクト規模が3ヶ月以内。

2. 初めにメンバに対しSCRUMの説明がない

3. 担当を固定する

You vs Me

自己組織化出来ない

4. 設計しない

今回は契約上、基本設計フェーズがあったので設計をした。

別プロジェクトでは、設計が不十分でチームの認識が揃わず手戻りが多くなる羽目に。

アジャイルだから設計しなくても良いわけではない。

Not Do Agile
Be Agile

MOVE THE WORLD FORWARD

**MITSUBISHI
HEAVY
INDUSTRIES
GROUP**