リーダブルコードの意義と、実践の方法

結城洋志

株式会社クリアコード

実践リーダブルコード

2022-11-02

全体の流れ

- ✓ リーダブルなコードの意義と実践方法の説明
- ✓ 実装
- ✓ 実装を交換、続きを実装
- ✓ 全体で結果を共有

講師紹介

結城洋志（ゆうき ひろし）
aka Piro

- ✓ 株式会社クリアコード所属
- ✓ FirefoxやThunderbirdの法人サポートに従事
  - ✓ トラブルの原因や対策を探るためソースコードを調査することが多い

アジェンダ

- ✓ **講座の目的**を確認
- ✓ リーダブルコードの**必要性**を確認
- ✓ リーダブルコードの**実践方法**を紹介
- ✓ 実践方法を**練習**

講座の目的

- ✓ リーダブルコードを
  日常的に書く上での
  基礎となる考え方
  を実践し、持ち帰る

目的でないこと

- ✓ テクニックをたくさん覚える
- ✓ 難しいプログラムを実装する
- ✓ プログラムを速く実装する
- ✓ 高性能なプログラムを実装する
- ✓ 奇抜な方法で目立つ

ワークショップ後

- ✓ ぜひ実践を！
- ✓ 今日の資料はすべて再利用可能
  - ✓ チーム内で同じ講座を再現できる

そもそもの話

- ✓ リーダブルコードはなぜ必要か
- ✓ 何の役に立つのか？

リーダブルコードが必要な理由

- ✓ 既存のコードを読んで
  素早く内容を把握したい
- ✓ 既存のコードに
  素早く手を加えたい
- ✓ **開発速度**を落としたくない

「速度」？

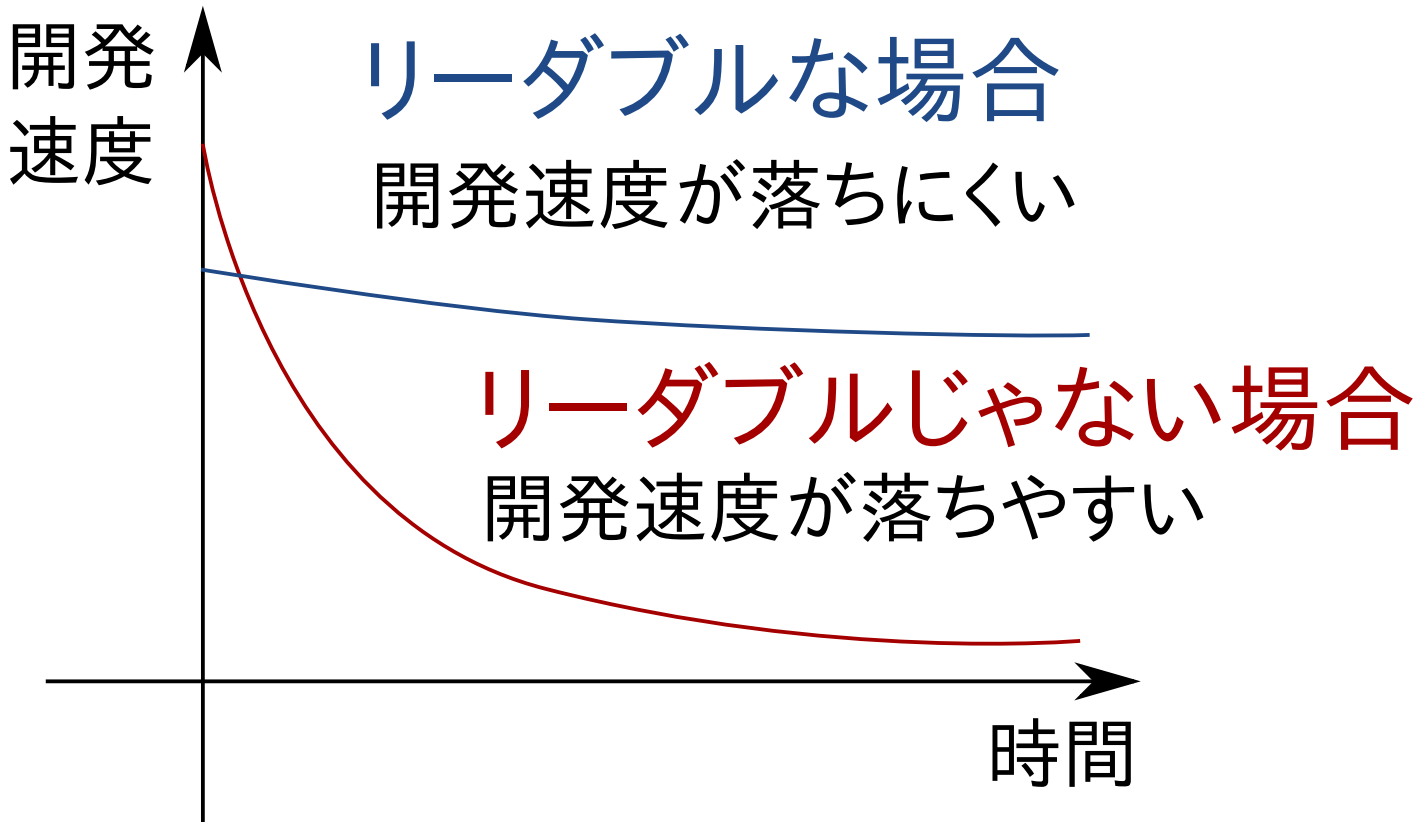
- ✓ 「読みやすさに気をつけたら
  開発スピードが
  落ちそう……」
- ✓ 「読みやすい書き方にしたら
  実行スピードが
  落ちそう……」

むしろ「読みにくいと開発が遅くなる」

- ✓ 既存のコードを理解しにくいと……
  - ✓ 修正・機能追加に時間がかかる
    （理解しないと変更できない）
  - ✓ 後退バグが発生しやすい
    （理解しないまま変更すると問題発生）

→**コストがかかる**

時間が経つほど影響大



（注意：グラフではなく概念図です）

つまり

- ✓ 現実的なコストの範囲で
- ✓ 既存のコードを継続的に、
  無理なく改良・修正したい

→なので、リーダブルコード

既存のコードに手を加える場面

- ✓ 新機能を追加するとき
- ✓ 不具合を修正するとき

業務での開発は、全く0からの新規開発でない限り既存のコードに手を加えることの方が多い

手を加えられないコードは業務上のリスク

- ✓ 分かる人が1人しかいない→危険
  - ✓ 実装者が抜けたら詰む
    - ✓ 変更できてもものすごくコストがかかる

↓

チームで開発し、
チームの誰でも作業を引き継げる
状態にしておきたい

リーダブルコードの実践

どうすれば無理なく実践できる？

リーダブルコードの実践

コードを**読む**
習慣を作る

読む？書くじゃないの？

- リーダブルコードを書くには
  コードを読むことが欠かせない
  - なぜ？

コードを読むのが大事な理由

- ✓ 何が「読みやすい」か
  読む側でないと
  分からないから
- ✓ 書いている最中は
  読みやすさ・読みにくさ
  を実感しにくいから

何が「読みやすい」かは受け手の状況次第

- ✓ しょっぱいのが好きな人と甘いのが好きな人で「おいしい」の基準は違う
  - ✓ 自分はどんな味が好みか？
- ✓ 日常の運転とレースでは「乗りやすい」の基準は違う
  - ✓ 自分はどんな乗り方をするのか？

リーダブルコードも同じ

- ✓ チームの状況によって
  「リーダブル」の基準が違う

リーダブルコード

「読む人」が
読みやすいなら
リーダブル

チーム内で「リーダブル」の基準を共有しよう

- ✓ どんなコードをどんな理由で「リーダブル」と感じたのか自己分析してみる
- ✓ 本で紹介されている書き方が本当に自分達にとっても「リーダブル」なのか確かめる

（実践パートでやってみよう）

何が「読みやすい」か分かっていても…

- ✓ **コードを書いている最中**は読みやすさ・読みにくさを実感しにくい

実際のコードで考えてみる

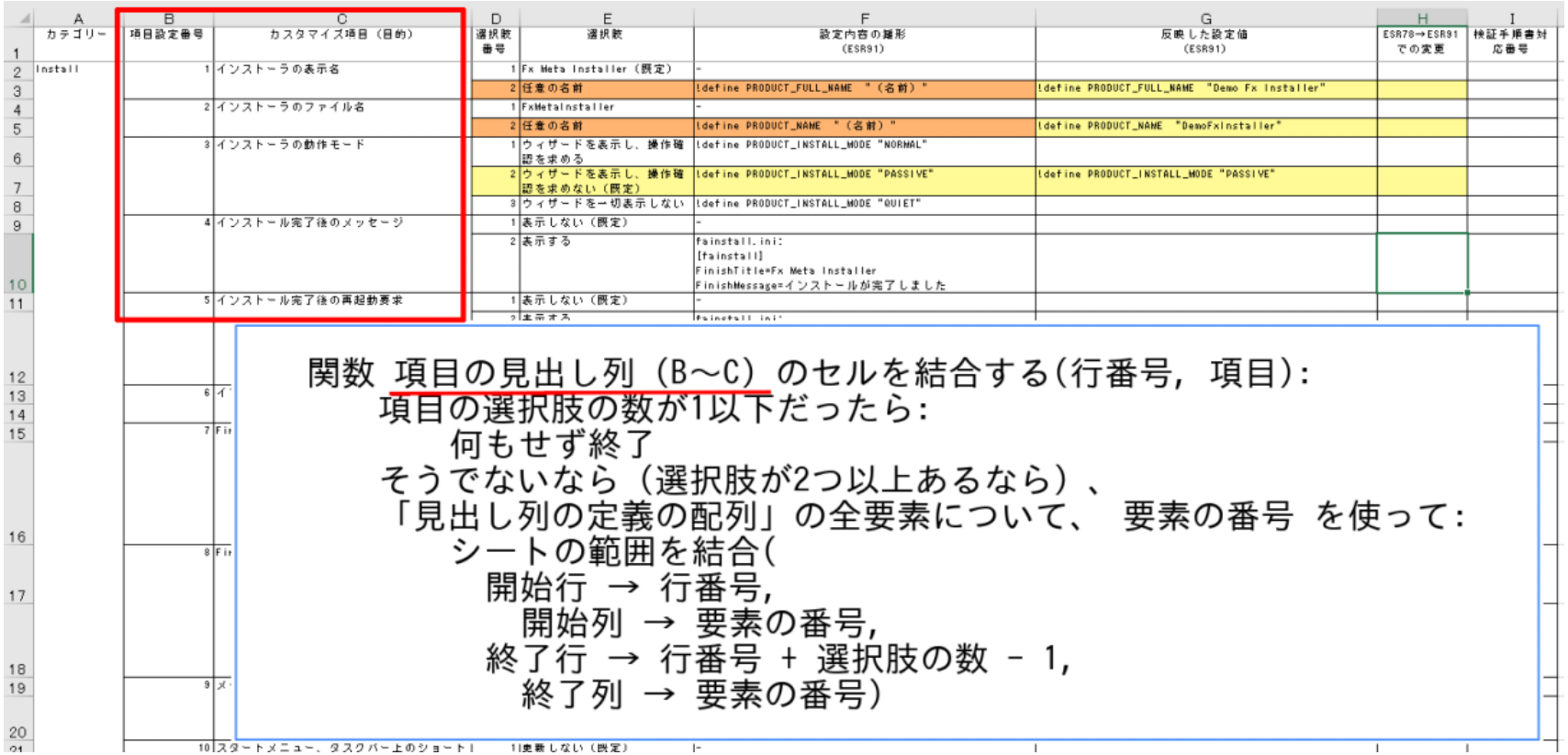
- ✓ Excelワークシートの生成で見出しセルを結合する関数

	A	B	C	D	E	F	G	H	I
1	カテゴリー	項目設定番号	カスタマイズ項目（目的）	選択肢番号	選択肢	設定内容の雛形（ESR91）	反映した設定値（ESR91）	ESR78→ESR91での変更	検証手順書対応番号
2	Install	1	インストーラの表示名	1	Fx Meta Installer（既定）	-			
3				2	任意の名前	!define PRODUCT_FULL_NAME "（名前）"	!define PRODUCT_FULL_NAME "Demo Fx Installer"		
4		2	インストーラのファイル名	1	FxMetaInstaller	-			
5				2	任意の名前	!define PRODUCT_NAME "（名前）"	!define PRODUCT_NAME "DemoFxInstaller"		
6		3	インストーラの動作モード	1	ウィザードを表示し、操作確認を求める	!define PRODUCT_INSTALL_MODE "NORMAL"			
7				2	ウィザードを表示し、操作確認を求めない（既定）	!define PRODUCT_INSTALL_MODE "PASSIVE"	!define PRODUCT_INSTALL_MODE "PASSIVE"		
8				3	ウィザードを一切表示しない	!define PRODUCT_INSTALL_MODE "QUIET"			
9		4	インストール完了後のメッセージ	1	表示しない（既定）	-			
10				2	表示する	fainstall.ini: [fainstall] FinishTitle=Fx Meta Installer FinishMessage=インストールが完了しました			
11		5	インストール完了後の再起動要求	1	表示しない（既定）	-			
12				2	表示する	fainstall.ini: [fainstall] ConfirmRestartTitle=Fx Meta Installer ConfirmRestartMessage=今すぐコンピュータを再起動しますか？			
13		6	インストーラのウィザードの表示言語	1	日本語（既定）	!define PRODUCT_LANGUAGE "Japanese"			
14				2	英語	!define PRODUCT_LANGUAGE "English"			
15		7	Firefox/Thunderbirdの同梱	1	同梱しない（既定）	-			
16				2	指定バージョンを同梱する	fainstall.ini: [fainstall] AppMinVersion=91.0 AppMaxVersion=91.99 ApplsESR=true Appls64bit=true ;ApplsDevEdition=true	fainstall.ini: [fainstall] AppMinVersion=78.0 AppMaxVersion=78.99 ApplsESR=true Appls64bit=true ;ApplsDevEdition=true		
17		8	Firefox/Thunderbirdのインストール先	1	既定のインストール先（既定）	Firefox-setup.ini/Thunderbird-setup.ini: [Install] InstallDirectoryName=Mozilla Firefox InstallDirectoryPath=			
18				2	任意のインストール先	Firefox-setup.ini/Thunderbird-setup.ini: [Install] InstallDirectoryName= InstallDirectoryPath=	Firefox-setup.ini: [Install] InstallDirectoryName= InstallDirectoryPath=C:¥Program Files (x86)¥Mozilla Firefox		
19		9	メタインストーラの表示バージョン	1	更新しない（既定）	-			
20				2	指定する	fainstall.ini: [fainstall] DisplayVersion=バージョン番号	fainstall.ini: [fainstall] DisplayVersion=78.0		
21		10	スタートメニュー、タスクバー上のショート	1	更新しない（既定）	-			

やりたいことを言語化してみる

```
関数 項目の見出し列（B～C）のセルを結合する(行番号, 項目):
    項目の選択肢の数が1以下だったら:
        何もせず終了
    そうでないなら（選択肢が2つ以上あるなら）、
    「見出し列の定義の配列」の全要素について、 要素の番号 を使って:
        シートの範囲を結合(
          開始行 → 行番号,
            開始列 → 要素の番号,
          終了行 → 行番号 + 選択肢の数 - 1,
            終了列 → 要素の番号)
```

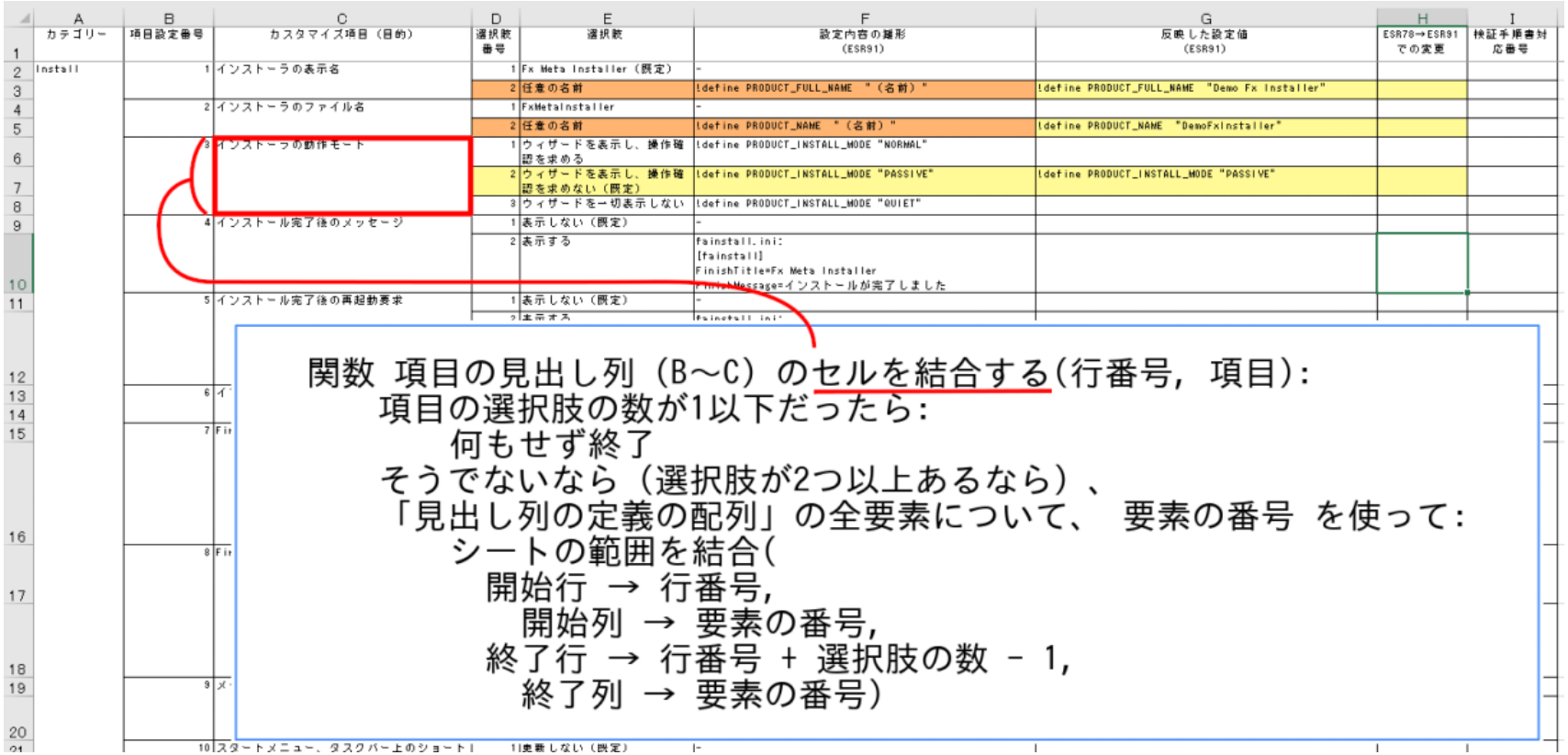
項目の見出し列の……



	A	B	C	D	E	F	G	H	I
1	カテゴリー	項目設定番号	カスタマイズ項目（目的）	選択肢番号	選択肢	設定内容の雛形（ESR91）	反映した設定値（ESR91）	ESR78→ESR91での変更	検証手順書対応番号
2	Install	1	インストーラの表示名	1	Fx Meta Installer（既定）	-			
3				2	任意の名前	!define PRODUCT_FULL_NAME "（名前）"	!define PRODUCT_FULL_NAME "Demo Fx Installer"		
4		2	インストーラのファイル名	1	FxMetaInstaller	-			
5				2	任意の名前	!define PRODUCT_NAME "（名前）"	!define PRODUCT_NAME "DemoFxInstaller"		
6		3	インストーラの動作モード	1	ウィザードを表示し、操作確認を求める	!define PRODUCT_INSTALL_MODE "NORMAL"			
7				2	ウィザードを表示し、操作確認を求めない（既定）	!define PRODUCT_INSTALL_MODE "PASSIVE"	!define PRODUCT_INSTALL_MODE "PASSIVE"		
8				3	ウィザードを一切表示しない	!define PRODUCT_INSTALL_MODE "QUIET"			
9		4	インストール完了後のメッセージ	1	表示しない（既定）	-			
10				2	表示する	!finstall.ini: [finstall] FinishTitle=Fx Meta Installer FinishMessage=インストールが完了しました			
11		5	インストール完了後の再起動要求	1	表示しない（既定）	-			
12				2	表示する	!finstall.ini:			
13		6	イ						
14									
15		7	Fi						
16									
17									
18		8	Fi						
19		9	メ						
20									
21		10	スタートメニュー、タスクバー上のショートカ	1	表示しない（既定）	-			

関数 項目の見出し列（B～C）のセルを結合する(行番号, 項目):
　項目の選択肢の数が1以下だったら:
　　何もせず終了
　そうでないなら（選択肢が2つ以上あるなら）、
　「見出し列の定義の配列」の全要素について、 要素の番号 を使って:
　　シートの範囲を結合(
　　開始行 → 行番号,
　　開始列 → 要素の番号,
　　終了行 → 行番号 + 選択肢の数 - 1,
　　終了列 → 要素の番号)

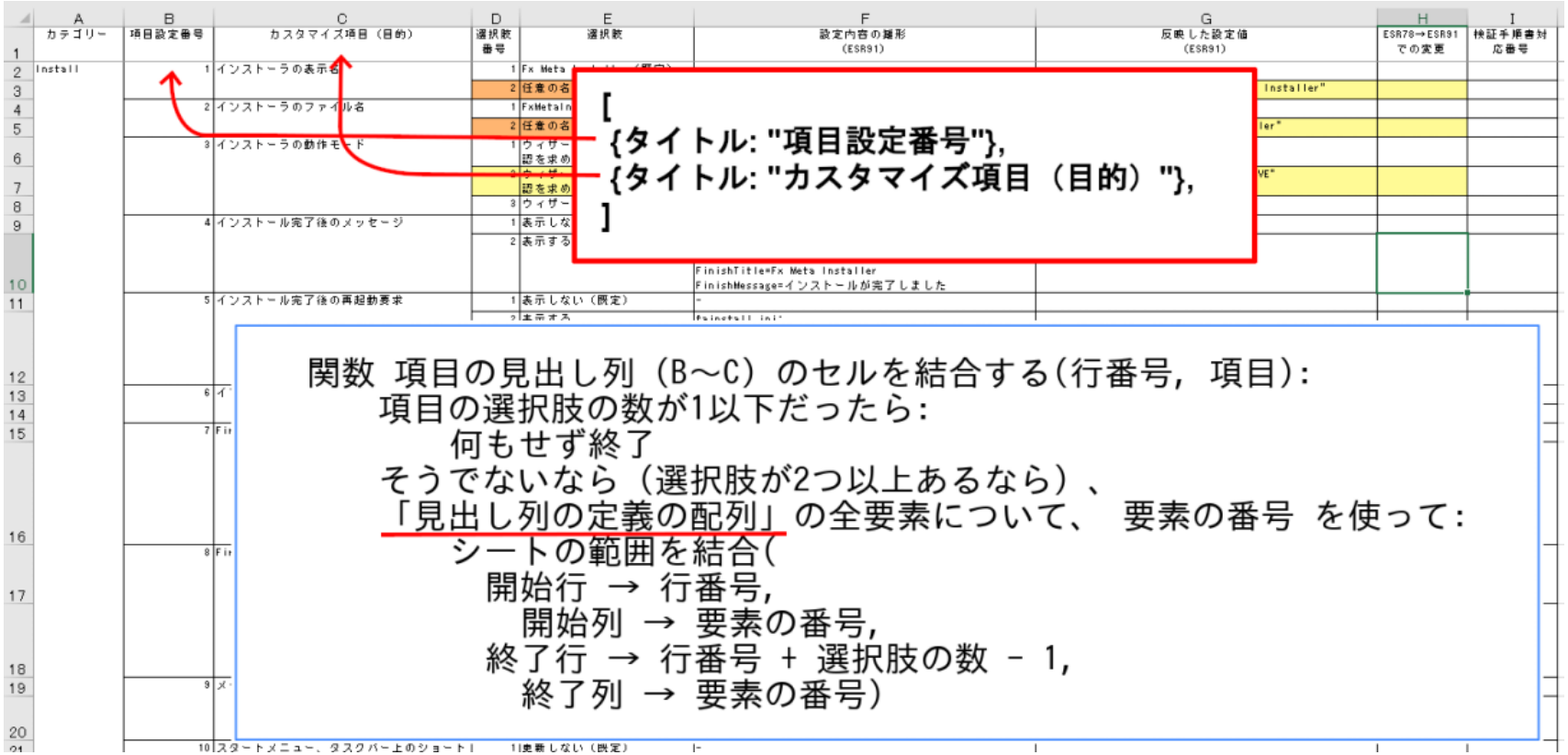
セルを結合する



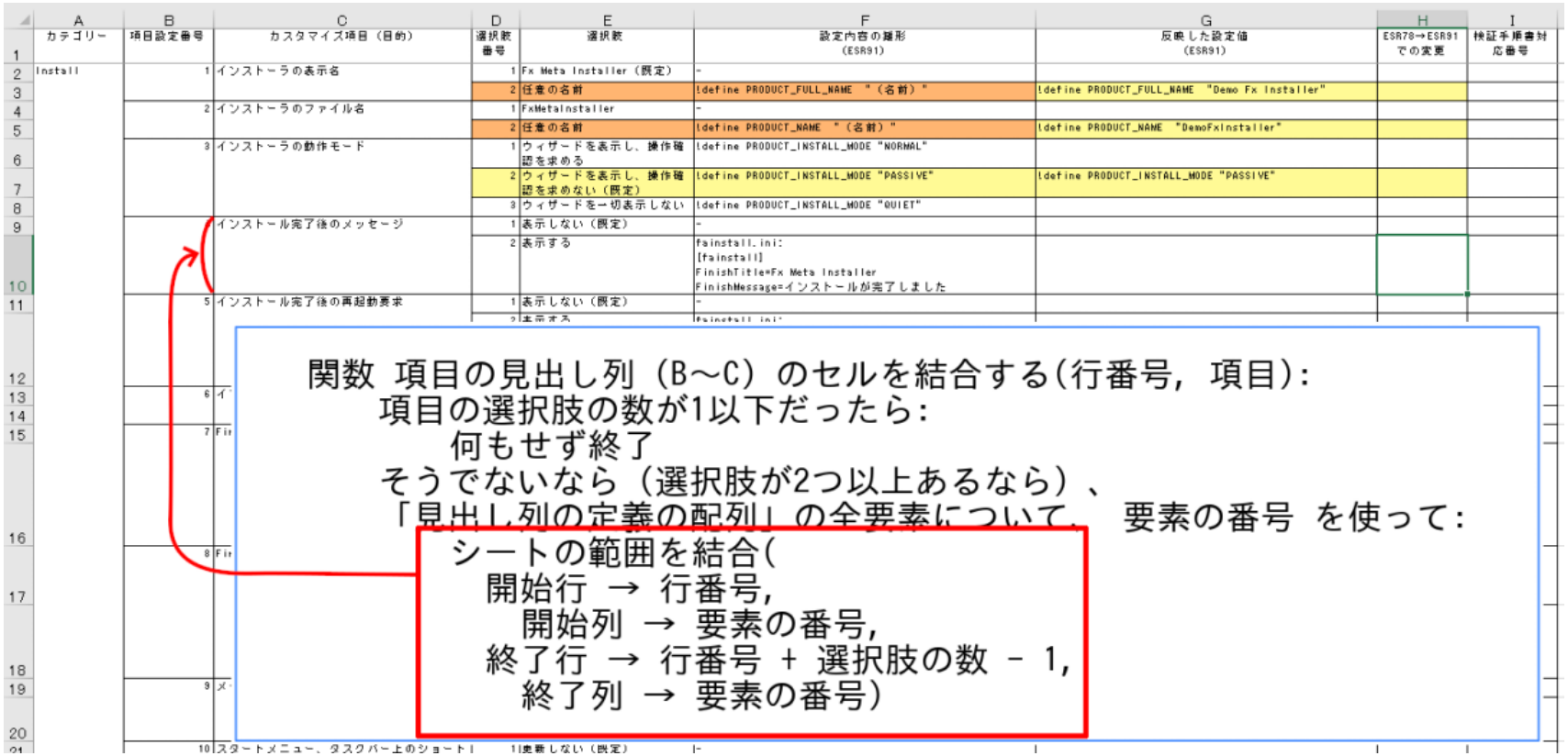
	A	B	C	D	E	F	G	H	I
1	カテゴリー	項目設定番号	カスタマイズ項目（目的）	選択肢番号	選択肢	設定内容の雛形（ESR91）	反映した設定値（ESR91）	ESR78→ESR91での変更	検証手順書対応番号
2	Install	1	インストーラの表示名	1	Fx Meta Installer（既定）	-			
3				2	任意の名前	!define PRODUCT_FULL_NAME "（名前）"	!define PRODUCT_FULL_NAME "Demo Fx Installer"		
4		2	インストーラのファイル名	1	FxMetaInstaller	-			
5				2	任意の名前	!define PRODUCT_NAME "（名前）"	!define PRODUCT_NAME "DemoFxInstaller"		
6		3	インストーラの動作モード	1	ウィザードを表示し、操作確認を求める	!define PRODUCT_INSTALL_MODE "NORMAL"			
7				2	ウィザードを表示し、操作確認を求めない（既定）	!define PRODUCT_INSTALL_MODE "PASSIVE"	!define PRODUCT_INSTALL_MODE "PASSIVE"		
8				3	ウィザードを一切表示しない	!define PRODUCT_INSTALL_MODE "QUIET"			
9		4	インストール完了後のメッセージ	1	表示しない（既定）	-			
10				2	表示する	!finstall.ini: [finstall] FinishTitle=Fx Meta Installer FinishMessage=インストールが完了しました			
11		5	インストール完了後の再起動要求	1	表示しない（既定）	-			
12				2	表示する	!finstall.ini:			
13		6	イ						
14									
15		7	Fi						
16									
17		8	Fi						
18									
19		9	メ						
20									
21		10	スタートメニュー、タスクバー上のショートカ	1	表示しない（既定）	-			

```
関数 項目の見出し列（B～C）のセルを結合する(行番号, 項目):
    項目の選択肢の数が1以下だったら:
        何もせず終了
    そうでないなら（選択肢が2つ以上あるなら）、
    「見出し列の定義の配列」の全要素について、 要素の番号 を使って:
        シートの範囲を結合(
            開始行 → 行番号,
                開始列 → 要素の番号,
            終了行 → 行番号 + 選択肢の数 - 1,
                終了列 → 要素の番号)
```

見出し列の定義の配列


[
{タイトル: "項目設定番号"},
{タイトル: "カスタマイズ項目（目的）"},
]
関数 項目の見出し列（B〜C）のセルを結合する(行番号, 項目):
項目の選択肢の数が1以下だったら:
何もせず終了
そうでないなら（選択肢が2つ以上あるなら）、
「見出し列の定義の配列」の全要素について、 要素の番号 を使って:
シートの範囲を結合(
開始行 → 行番号,
開始列 → 要素の番号,
終了行 → 行番号 + 選択肢の数 - 1,
終了列 → 要素の番号)

シートの範囲を指定して結合



実際のコード

```
def m(s, n, i):
    if len(i['o']) <= 1:
        return
    s2 = s.s
    for i2, c in enumerate(hcol):
        s2.merge_range(n, i2,
                       n + len(i['o']) - 1, i2,
                       '')
```

書いた本人の視界

```
関数 項目の見出し列（B～C）のセルを結合する(行番号, 項目):
    項目の選択肢の数が1以下だったら:
        何もせず終了
    そうでないなら（選択肢が2つ以上あるなら）、
    「見出し列の定義の配列」の全要素について、
      要素の番号 を使って:
        シートの範囲を結合(開始行番号, 開始列番号,
                          開始行番号 + 選択肢の数 - 1, 終了列番号)
```

```
def m(s, n, i):
    if len(i['o']) <= 1:
        return
    s2 = s.s
    for i2, c in enumerate(hcol):
        s2.merge_range(n, i2,
                       n + len(i['o']) - 1, i2,
                       '')
```

第三者の視界

```
def m(s, n, i):
    if len(i['o']) <= 1:
        return
    s2 = s.s
    for i2, c in enumerate(hcol):
        s2.merge_range(n, i2,
                       n + len(i['o']) - 1, i2, '')
```

```
関数 何か(何か, 何か, 何か):
    何かの数が1以下だったら:
        何もせず終了
    そうでないなら、
    何か = 何かが保持している何か
    何かについて、 何か を使って:
        何かの範囲を結合(何か, 何か,
                         何か + 何か - 1, 何か)
```

前情報無しだと、意味ある情報を読み取るのは困難……

リーダブルなコードの場合

```
def try_merge_item_heading(self, row, item):
    if len(item['options']) <= 1:
        return
    sheet = self.sheet
    for index, _column in enumerate(HEADING_COLUMNS):
        sheet.merge_range(row, index,
                          row + len(item['options']) - 1, index,
                          '')
```

```
------------------------------------------------------------
関数 項目の見出しセルを結合してみる(自分, 行, 項目):
    項目の選択肢の数が1以下だったら:
        何もせず終了
    そうでないなら、
    シートは自分が保持している
    見出しの列 の全項目について、 番号 を使って:
        シートの、範囲を結合( 行, 番号,
                              行 + 項目の選択肢の数 - 1, 番号,
                              '')
```

先の例よりは意味のある情報を読み取れるはず

よく聞く話

- ✓ 手紙を書いたら一晩寝かせろ！
- ✓ 試験の回答文は、最後に
  もう一回読み返せ！

前提を知らない読み手
の気持ちになろう

実際のコードを読んでみよう

Excelのワークシートを自動生成するPythonスクリプト

	A	B	C	D	E	F	G	H	I
1	カテゴリー	項目設定番号	カスタマイズ項目（目的）	選択肢番号	選択肢	設定内容の雛形 (ESR91)	反映した設定値 (ESR91)	ESR78→ESR91での変更	検証手順書対応番号
2	Install	1	インストーラの表示名	1	Fx Meta Installer（既定）	-			
3				2	任意の名前	!define PRODUCT_FULL_NAME "（名前）"	!define PRODUCT_FULL_NAME "Demo Fx Installer"		
4		2	インストーラのファイル名	1	FxMetaInstaller	-			
5				2	任意の名前	!define PRODUCT_NAME "（名前）"	!define PRODUCT_NAME "DemoFxInstaller"		
6		3	インストーラの動作モード	1	ウィザードを表示し、操作確認を求める	!define PRODUCT_INSTALL_MODE "NORMAL"			
7				2	ウィザードを表示し、操作確認を求めない（既定）	!define PRODUCT_INSTALL_MODE "PASSIVE"	!define PRODUCT_INSTALL_MODE "PASSIVE"		
8				3	ウィザードを一切表示しない	!define PRODUCT_INSTALL_MODE "QUIET"			
9		4	インストール完了後のメッセージ	1	表示しない（既定）	-			
10				2	表示する	fainstall.ini: [fainstall] FinishTitle=Fx Meta Installer FinishMessage=インストールが完了しました			
11		5	インストール完了後の再起動要求	1	表示しない（既定）	-			
12				2	表示する	fainstall.ini: [fainstall] ConfirmRestartTitle=Fx Meta Installer ConfirmRestartMessage=今すぐコンピュータを再起動しますか？			
13		6	インストーラのウィザードの表示言語	1	日本語（既定）	!define PRODUCT_LANGUAGE "Japanese"			
14				2	英語	!define PRODUCT_LANGUAGE "English"			
15		7	Firefox/Thunderbirdの同梱	1	同梱しない（既定）	-			
16				2	指定バージョンを同梱する	fainstall.ini: [fainstall] AppMinVersion=91.0 AppMaxVersion=91.99 AppIsESR=true AppIs64bit=true ;AppIsDevEdition=true	fainstall.ini: [fainstall] AppMinVersion=78.0 AppMaxVersion=78.99 AppIsESR=true AppIs64bit=true ;AppIsDevEdition=true		
17		8	Firefox/Thunderbirdのインストール先	1	既定のインストール先（既定）	Firefox-setup.ini/Thunderbird-setup.ini: [Install] InstallDirectoryName=Mozilla Firefox InstallDirectoryPath=			
18				2	任意のインストール先	Firefox-setup.ini/Thunderbird-setup.ini: [Install] InstallDirectoryName= InstallDirectoryPath=	Firefox-setup.ini: [Install] InstallDirectoryName= InstallDirectoryPath=C:¥Program Files (x86)¥Mozilla Firefox		
19		9	メタインストーラの表示バージョン	1	更新しない（既定）	-			
20				2	指定する	fainstall.ini: [fainstall] DisplayVersion=バージョン番号	fainstall.ini: [fainstall] DisplayVersion=78.0		
21		10	スタートメニュー、タスクバー上のショート	1	更新しない（既定）	-			

ビジネス上の要件

- ✓ Firefoxを法人運用向けに設定したい
  - ✓ 設定を一覧で管理したい
    - ✓ 「自動更新：有効/無効」など
  - ✓ 顧客ごとに設定内容を変えたい
  - ✓ バージョンごとの変化を見たい
    - ✓ Firefox 78からFirefox 91の間で追加された設定、廃止された設定

元々はExcelワークシートを手作りしていた

- ✓ Gitでバージョン管理しにくい
- ✓ 同じ変更内容を複数顧客のワークシートに反映するのに手間がかかる

どうにかしたかった

自動化を図った

- 資料自体を
  バージョン管理しやすく
  - プレーンテキスト形式のソースから
    Excelワークシートを自動生成
    1. 設定項目の定義
    2. 前バージョンのFirefox用の設定情報
    3. 今バージョンのFirefox用の設定情報
- 初版はPythonに慣れた人が実装

読んでみよう1

build-xlsx-1.py

ざっと見て概要を掴んでみよう

ポイント

- ✓ 少数の短い関数
- ✓ 分かりやすい名前付け
- ✓ 列番号はべた書き

試しに探してみよう1

- ✓ 「...→...での変更」の列に「新規」と出力する条件はどこで判定している？

試しに読んでみよう2

build-xlsx-2.py

第三者が手を加えた物

改修の経緯

- 要件が増えた
  - 比較対象の数が可変になった
    - 改修前：「Firefox 78」と
      「Firefox 91」
    - 改修後：「Firefox 78」と
      「Firefox 91(デスクトップPC)」
      「Firefox 91(ノートPC)」...
- 行数は約1.2倍に増加

試しに探してみよう2

- ✓ 「検証手順書対応番号」の列に出力する内容はどこで決まっている？

さっきより大変なはず

リーダブルだった書き方がアンリーダブルに……

- ✓ 関数の数が少ない
  - ✓ 個々の関数が肥大化し、全体像の把握が困難に
- ✓ if-elif/elseでの条件分岐
  - ✓ 階層が複雑化して処理の実行条件を追いにくい

試しに読んでみよう3

build-xlsx-3.py

大幅に改修した物

改修の要旨

- ✓ 複雑化した問題に合わせて「リーダブル」の基準を変えた
  何がリーダブルかは状況に依存する
  - ✓ 定数を使う部分を増やした
  - ✓ メソッド・変数のスコープを小さく
- ✓ コードの行数は約1.5倍に増加

試しに探してみよう3

- ✓ 「検証済み」という列を
「検証手順書対応番号」
の列の右隣に追加したい
（セルの内容は空でよい）が、
どこに手を入れればいい？

試しに探してみよう3

- 「検証済み」という列を
  「検証手順書対応番号」
  の列の右隣に追加したい
  （セルの内容は空でよい）が、
  どこに手を入れればいい？
- →**探し方のコツ**がある
  - 改修前：**全体**を読んで覚えて探す
  - 改修後：**必要な部分だけ**読んで探す

つまり、ここでの「リーダブル」とは

- ✓ 全体を一望するのが難しい複雑なコードについて
- ✓ 一度に読むことは諦めて
- ✓ 必要な部分だけ読む

という前提での「読みやすい」

ふつうのOSS開発者の日常

- ✓ × イメージ
  - ✓ コード全体を詳細・完璧に把握
  - ✓ ノールックで修正
- ✓ ○ 実態
  - ✓ 全体像はボンヤリ把握
  - ✓ 都度必要な部分を読み直して調べ直しながら修正

自分で書いたコードでも、覚えてないのが当たり前！

- ✓ **覚えておかなくていいようにする**のがリーダブルコード
- ✓ 書籍「リーダブルコード」巻末の「解説」も読んでみて！

リーダブルコードの実践

- ✓ 別の切り口でも考えてみよう
- ✓ なぜアンリーダブル化するのか？

アンリーダブル化する原因

考えてみよう

アンリーダブル化する原因（例）

- ✓ コードを**書き換えない**から
  アンリーダブル化する

理想的な変更の流れ

1. 要件に従って設計する
2. 設計通り実装する
3. 要件に従って設計する
4. 設計通り実装する
5. ...

悪くない変更の流れ

1. とりあえず動くようにする
2. 設計を見直して整理する
3. とりあえず動くようにする
4. 設計を見直して整理する
5. ...

アンリーダブル化しやすい流れ

1. とりあえず動くようにする
2. とりあえず動くようにする
3. とりあえず動くようにする
4. ...

設計は随時見直さないといけない

- 設計が**状況の変化**に追従できていない
  - コードの規模や前提は**徐々に変化**する
    - 気付かないうちに進行する！

状況の変化に気付けても、追従できない……

- ✓ 既存のコードを**書き直せない**
  - ✓ 書き直して動かなくなるのが怖い
  - ✓ 「とりあえず動くように」で焦って「書き足す」一辺倒になる
    - ✓ 冷静に見直す余裕がない

→良くない兆候を見て見ぬフリ

ベテランはどうする？

- ✓ 悪い**兆候**を見逃さない
- ✓ 既存のコードを
  書き直すのをためらわない
  - ✓ 書き直さない方が、
    後で痛い目を見ると知っている

悪い兆候に敏感になろう

- ✓ **マメに読み返している**と、
  アンリーダブルのなり始めに
  すぐ気付ける
- ✓ 「アンリーダブルに
  なり始めてる……？」
  →書き直しを考えるタイミング
  後回しにしない！
  （すぐやらないなら、せめてコメントを残す）

ためらわずに書き直せるようにするために(1)

- ✓ わけが分からないままにしない
  - ✓ **自分が今何をしているか**を正しく理解するよう努めてる
  - ✓ 不安が少しでも生じたら立ち止まって考える

ためらわずに書き直せるようにするために(2)

- ✓ 便利な道具で補う
  - ✓ 自動テストで**結果の同値性**を保証
    自動テストは大事！
  - ✓ Gitで**いつでも巻き戻せる**安心感
    バージョン管理システムは大事！
  - ✓ **コードフォーマッター**で安全にコードを整形
    「リーダブルにする」はある程度自動化できる

早め早めのリカバリー

- ✓ なるべく早くリーダブルに直す
  - ✓ 書く→直す のサイクルを回す

参考：良い分割の仕方(1/2)

- "良いコードとは何か - エンジニア新卒研修 スライド公開"
  - https://note.com/cyberz_cto/n/n26f535d6c575#E0aBe
    - サイバーエージェント社の新卒研修の資料
  - 「凝集度と結合度」が参考になる

参考：良い分割の仕方(2/2)

- 良いコード/悪いコードで学ぶ設計入門
  ―保守しやすい 成長し続けるコードの書き方
  - 著：仙塲 大也
  - 刊：技術評論社
  - 通称「ミノ駆動本」

リーダブルコードの練習

- ✓ 読まれる前提で
  コードを書いてみよう
- ✓ コードを書く→読む→直す
  のサイクルを体験しよう

体験学習

- 11:55-14:15 課題を実装
  - リーダブルコードを書く
- 14:30-15:45 実装チェンジ
  →開発継続
  - 「まず自分が読み始める」
  - 「リーダブルコードを探す」
- 16:00- グループふりかえり
  - 「他のメンバーと共有」

おさらい

- ✓ 講座の目的？
- ✓ リーダブルコードの必要性？
- ✓ 講座でやること？

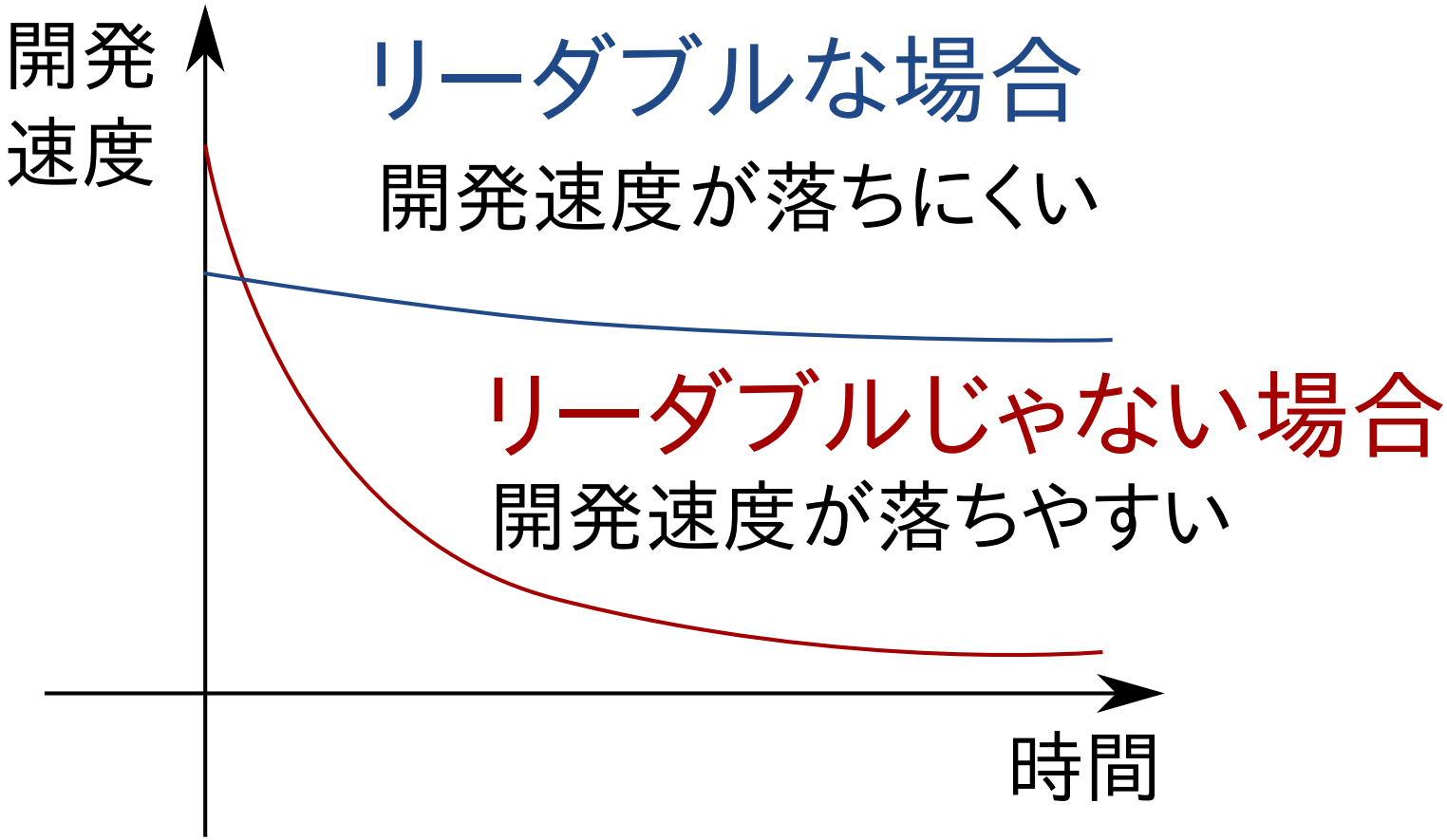
講座の目的

- ✓ リーダブルコードを
  日常的に書く上での
  基礎となる考え方
  を実践し、持ち帰る

リーダブルコードが必要な理由

- ✓ 既存のコードを読んで
  素早く内容を把握したい
- ✓ 既存のコードに
  素早く手を加えたい
- ✓ **開発速度**を落としたくない

変更コストと開発速度



講座でやること

- ✓ コードを読む文化作りの体験
  - ✓ 自分がコードを読んでみる
  - ✓ 読まれる前提でコードを書いてみる
  - ✓ 他の人とリーダブルコードの基準を共有する

ここまでの説明

腑に落ちましたか？