ClearCode

概要

須藤功平

株式会社クリアコード

実践リーダブルコード

2015-03-06

今日の流れ - 午前

- ✓ 10:00- アイスブレーク
- ✓ 10:15- 概要と進め方の説明
- ✓ 10:45- 実装
- ✓ 12:15- ランチ

今日の流れ - 昼下がり

- ✓ 13:15- 読み方のデモ
- ✓ 13:30- チェンジして実装
- ✓ 15:00- グループふりかえり
- ✓ 15:30- グループ発表

今日の流れ - 夕方

- ✓ 16:00- まとめ
  - ✓ 次のステップを説明
- ✓ 16:30- 質疑応答
- ✓ 17:30- コードの感想戦
  - ✓ 時間があれば
- ✓ 18:00- （有志で懇親会？）

チューター紹介

- 参加者のサポート係
- 現役エンジニア
- 参加者がわからない
  - →聞くと助けてくれる
  - →モジモジしてると声をかけてくる

チューター紹介

- 結城洋志（ゆうき ひろし）
  - 得意言語: JavaScript
- 沖元謙治（おきもと けんじ）
  - 得意言語: Ruby
- 横山昌史（よこやま まさふみ）
  - 得意言語: Ruby

講師紹介

須藤功平（すとう こうへい）

- ✓ クリアコード代表取締役
- ✓ リーダブルコード（本）の「解説」の著者
- ✓ 進行と全体を気にかける係

講座の目的

- ✓ 自分の開発チームに
- ✓ **リーダブルなコードが当たり前な文化の作り方**を
- ✓ 持ち帰る

→ 「解説」に書いていることの実践方法を学ぶ

サポート

- ✓ 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
- ✓ 無料のフォローアップ面談
  - ✓ チームで実践→悩み
↑の相談に乗る
  - ✓ 受講後3ヶ月以内に1回

そもそもの話

- ✓ リーダブルコードはなぜ必要か

↓を目指すために
チームでの共有は必須

リーダブルなコードが
当たり前な文化

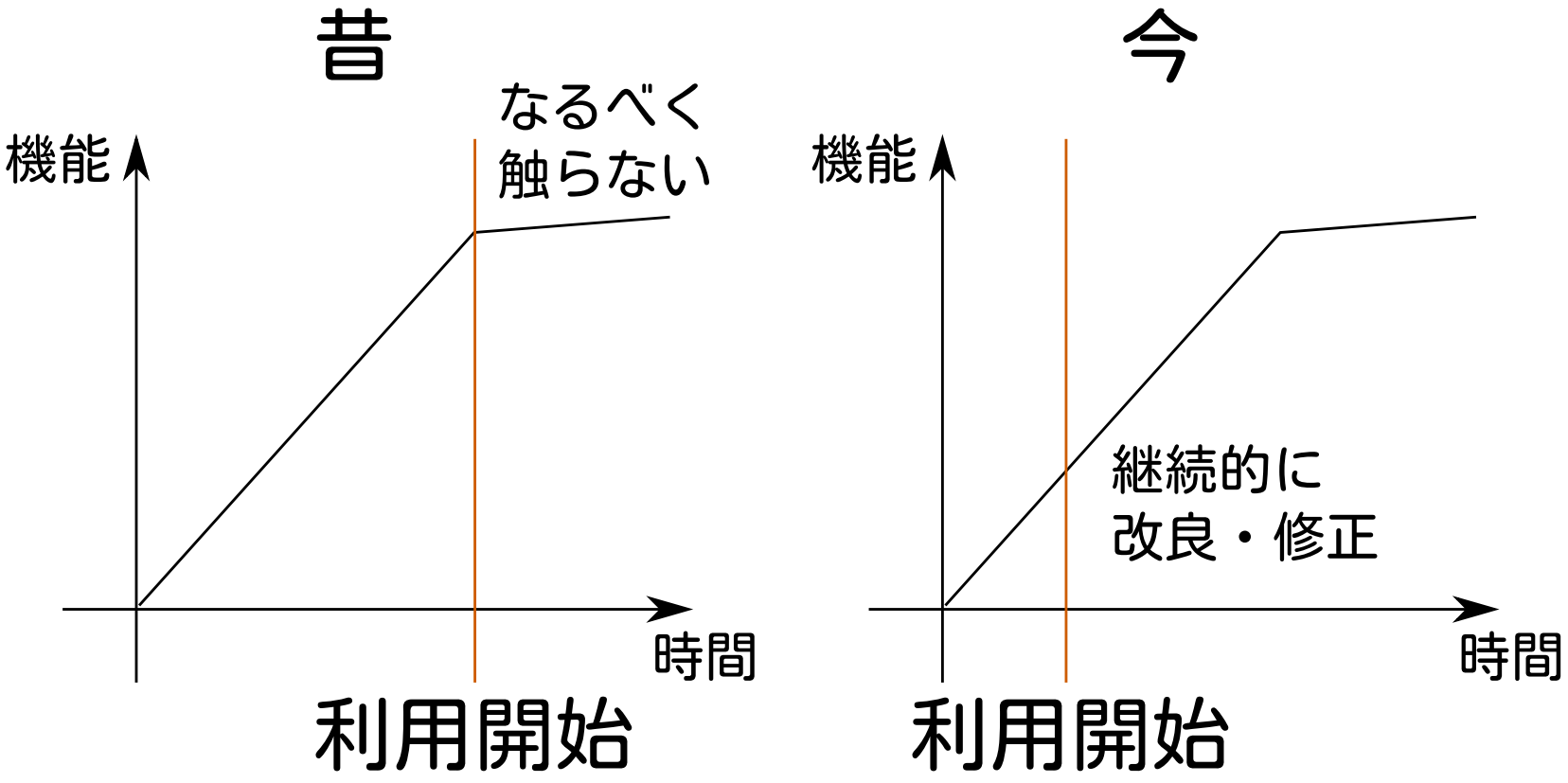
リーダブルコードの必要性 (1)

作って終わり
じゃないから

ソフトウェアの生涯

- 昔
  - 作って終わり
  - 作ったらできるだけ触らない
- 今
  - 機能は少なくてもまず動くものを
  - 動いてからも継続的に改良・修正

ソフトウェアの生涯



注：同じものができていそうな図だが、できあがるものは違う。
←は利用開始前に想定していたものができる。
→は利用したフィードバックを反映したものができる。

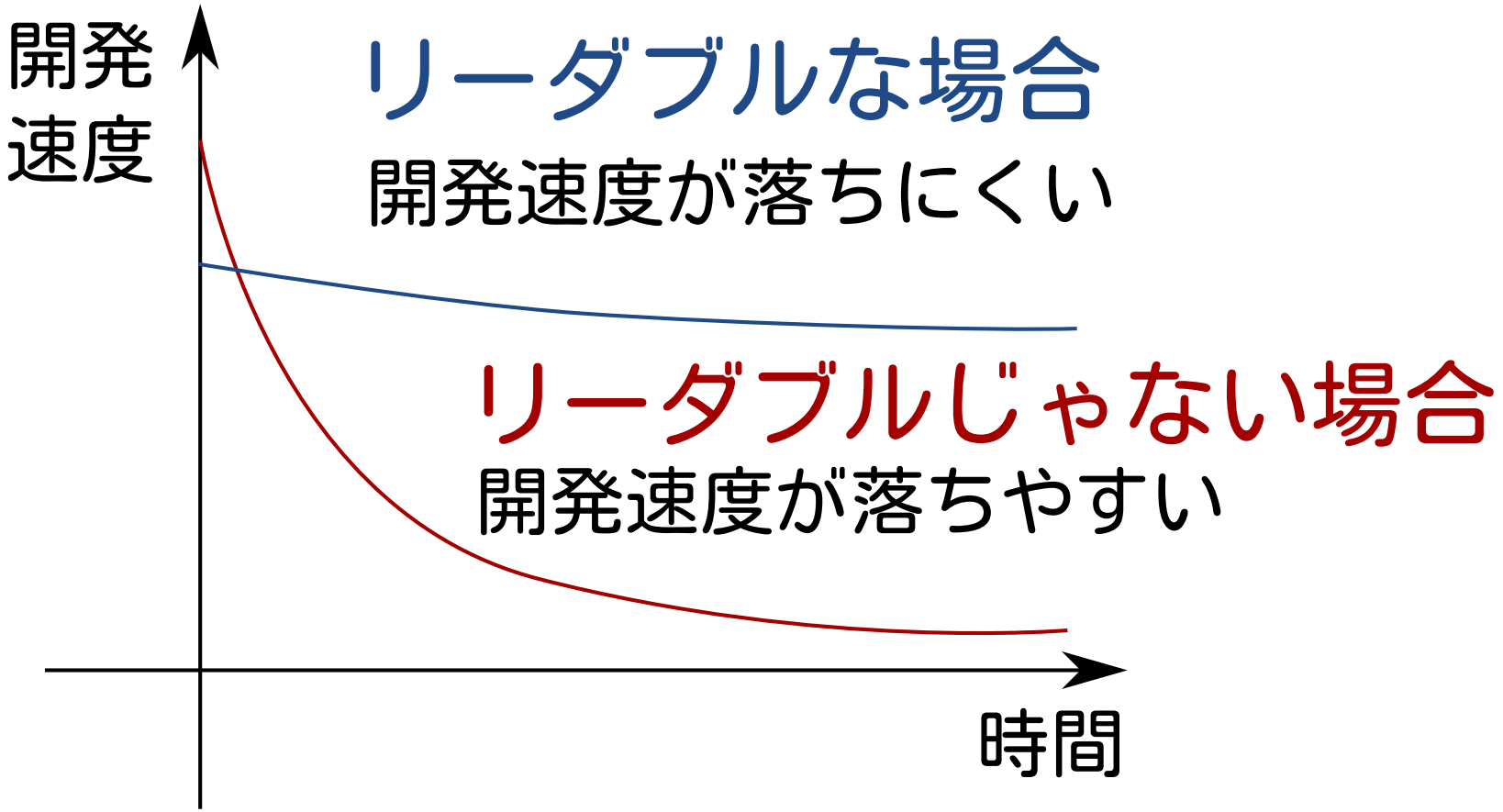
継続的に改良・修正

- ✓ 既存の機能の改良
  - a. 既存機能の理解→
  - b. 改良
- ✓ 既存の機能の修正
  - a. 既存機能の理解→
  - b. 修正

既存機能の理解のため

リーダブル
コード

時間が経つほど効果がでる



リーダブルコードの必要性(2)

1人だけの開発じゃないから

チームでの開発

- 昔
  - 1つのモジュールに1人の担当者
  - →1人しか触れないモジュール
  - →チームとして改良・修正できない
- 今
  - 1つのモジュールに複数担当者
  - →複数人が触れるモジュール
  - →チームとして改良・修正できる

複数人が触れる

触れる

↓

既存機能を
理解できる

既存機能の理解のため

リーダブル
コード

リーダブルコードの必要性

- ✓ 継続的に改良・修正したい
- ✓ チームとして改良・修正したい

↑をチームで共有することが
最初にやること

必要性を共有した後

コードを読む
文化を作る

読む？書くじゃないの？

- リーダブルコードを書くには
  コードを読まないといけない
  - なぜ？
  - →リーダブルコードは
    チーム毎に違うから

リーダブルコード

「読む人」が
読みやすいなら
リーダブル

読む人

- ✓ 多くの場合、いない
  - ✓ チームのコードを読んでいますか？
- ✓ 読む人（チームメンバー）毎にリーダブルの基準は違う
  - ✓ 背景が違うので当たり前
    （背景：使ってきた言語とか）

チームでのリーダブル

- ✓ 1つずつ見つけていくしかない
  - ✓ 各メンバーの読んだ感覚を
    チームで共有
  - ✓ 既存の基準をベースにするのはアリ
    （基準：本の内容やコーディングスタイルなど）

チームでのリーダブルコードは
育てていくもの

リーダブルの基準の育て方

- ✓ コードを読む文化を作る
  （最初の難関）
- ✓ チームのコードの中から
  リーダブルなコードを見つける
- ✓ リーダブルなコードを
  チームで共有
- ✓ ↑の繰り返しで基準を増やす

コードを読む文化を作る

- ✓ まず自分が読み始める
  - ✓ 仲間がいると心強い
- ✓ リーダブルなコードを探す
  - ✓ 読みにくいコードは今は置いておく
    （チームにコードを読む文化ができてから！）
  - ✓ 見つけたリーダブルなコードは…

リーダブルなコードは…

- ✓ 他のメンバーに教える
  （例：話しかける。チャットに書く。Wikiにまとめる。）
  - ✓ 「○○さんの△△という書き方、リーダブルでしたよー」

↓

読みやすさの基準を共有
コードが読まれているという自覚

読むことを「当たり前」に

- ✓ 「あいつはコードを読むやつ」という認識を広める
- ✓ 自分だけからチームへ

…続きはセミナーの最後に

ワークショップ内容

改良するために
他の人のコードを読む

- ✓「まず自分が読み始める」
- ✓「リーダブルコードを探す」
  （読みにくいコードは今は置いておく）
- ✓「リーダブルの基準を共有」

注意：やらないこと

リーダブルコードを書くための
テクニックをたくさん伝授

テクニック伝授は範囲外

- ✓ 順番が違う
- ✓ まず読む文化を作ること
  - ✓ 今日は↑がメイン
- ✓ テクニックはその後
  - ✓ 時間があれば「コードの感想戦」でフォロー

やること

読む文化作りの体験

読む文化作り

- ✓ まず自分が読み始める
- ✓ リーダブルコードを探す
- ✓ 他のメンバーに教える

読む文化作りの体験

- 10:45- 課題を実装
  - リーダブルコードを書く
- 13:30- 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」
- 15:30- グループ発表
  - 「他のメンバーに教える」

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

- ✓ 自分の開発チームに
- ✓ **リーダブルなコードが当たり前な文化の作り方**を
- ✓ 持ち帰る

文化作りの前

- リーダブルコードの必要性をチームで共有
  - 必須
  - これを飛ばすと頓挫する

リーダブルコードの必要性

- ✓ 継続的に改良・修正したい
- ✓ チームとして改良・修正したい

↑

既存機能の理解が重要なら必要

講座でやること

- ✓ コードを読む文化作りの体験
  - ✓ まず自分が読み始める
  - ✓ リーダブルコードを探す
  - ✓ 他のメンバーに教える