特定エディタに依存しない細粒度編集履歴収集ツールの開発とその適用

石田直人・井上研究室・博士前期課程 2年
2020年2月12日

背景

コーディングプロセスの改善

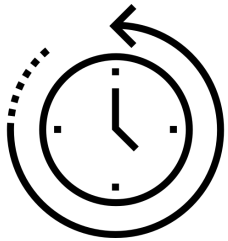
ITが様々な産業で活用される現代において**ソフトウェアの生産性及び信頼性の向上**が望まれている。

ソフトウェアを取り巻く課題の改善方法の一つに**コーディングプロセスの改善**がある。改善するためには**開発中にどのようなことが起きたかを記録・分析・活用することが重要**である。

コーディングプロセスを把握するためにはGit等のバージョン管理システムに記録された情報では不十分である。Negaraらは変更の37%(うち約8割は無視できないコード編集)が追跡不可能になっていることを指摘している[A].

[A] Stas Negara, Mohsen Vakilian, Nicholas Chen, Ralph E Johnson, and Danny Dig. Is it dangerous to use version control histories to study source code evolution? In European Conference on Object-Oriented Programming, pp. 79–103. Springer, 2012.

細粒度開発履歴 (1/2)

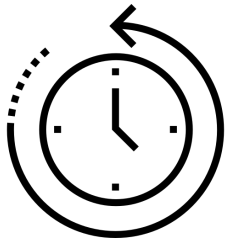


細粒度開発履歴とは**開発者がソースコードを作成する際に発生する様々なイベントを記録したもの**である。開発中にどのようなことが起きたかを詳細に理解することに利用できる。

既存研究では細粒度開発履歴を使用してコミットを適切な単位に自動的に切り分けたり[B]、プログラムの変更内容を自動で付与したりして開発者を支援する[C]ことなどが行われている。

[B] 梅川晃一, 井垣宏, 吉田則裕, 井上克郎. 細粒度作業履歴を用いたtask level commit 支援手法の提案. 電子情報通信学会技術報告, Vol. 113, No. 489, pp. 61–66, 2013.
[C] 木津栄二郎. ソースコードの編集履歴を用いたプログラム変更理解に関する研究 2014.

細粒度開発履歴 (2/2)



既存研究では、エディタ・記録手法・応用手法は依存関係が強く2つの問題がある。

1. **労力をかけて蓄積した細粒度開発履歴データの活用が難しくなる**
2. **利用できるエディタが限定される**

特定エディタに依存しない統一された細粒度開発履歴の記録方法を作り

細粒度開発履歴データが自由に活用されていくような基盤を作る必要がある。

提案手法

KAMAKURA

細粒度編集履歴収集プラットフォーム

概要

エディタから**細粒度編集履歴**を収集して前処理を行いサーバーに送信する。

Language Server Protocol(LSP)[D]を使用する**エディタ依存の少ない設計**であり、エディタ上で動作するプラグインから**共通の仕様でサーバーに細粒度編集履歴を送信できる**仕組みを実現している。

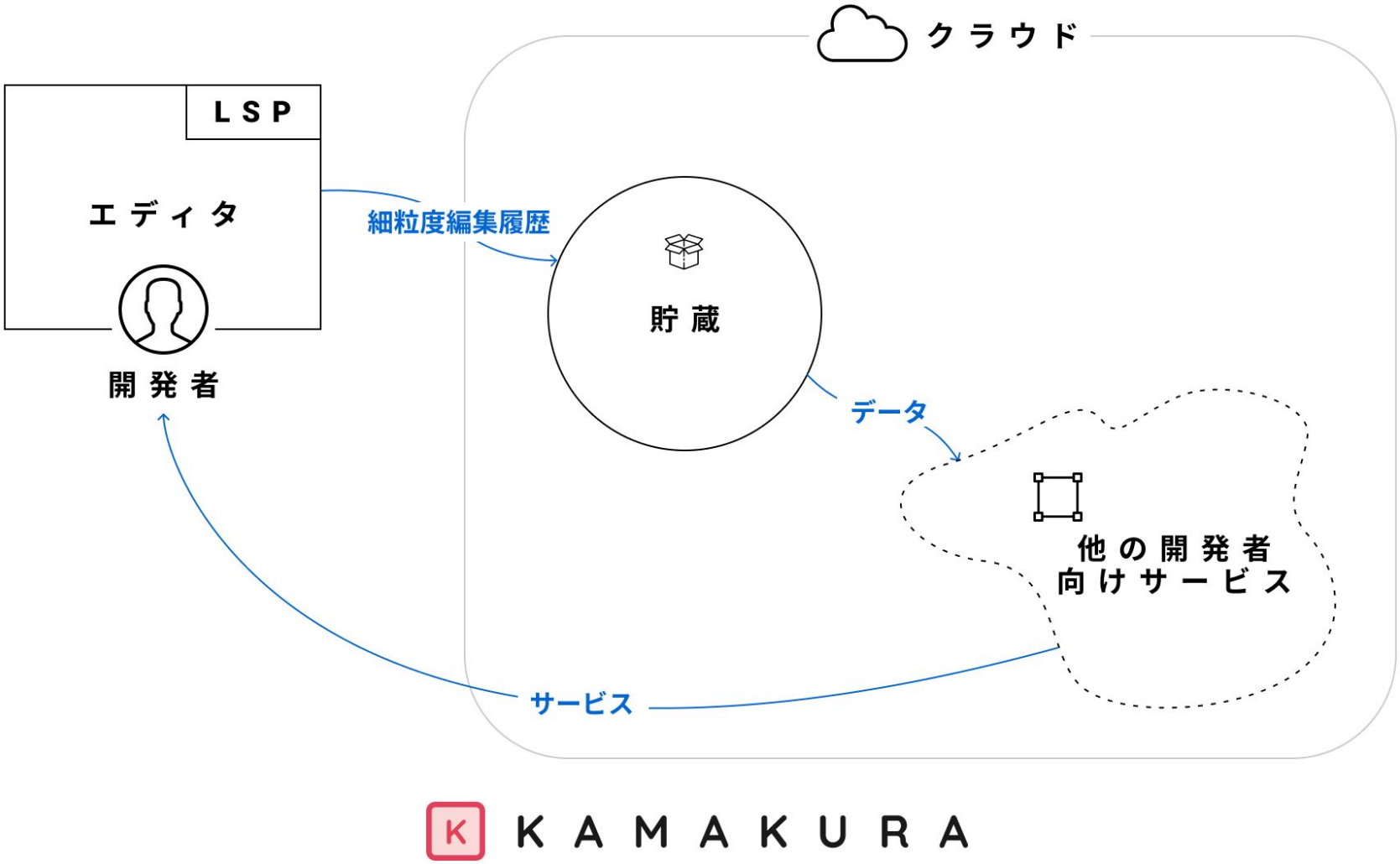
細粒度編集履歴の例。
src/Main.javaを開き
4行目にint型の変数numを初期化している。

```
...
13:07:58.392 <Open> src/Main.java
13:08:03.873 <Add> [L4C0, "int "]
13:08:07.933 <Add> [L4C4, "num"]
13:08:09.201 <Add> [L4C7, "=10;"]
...
```

特定エディタに依存しない細粒度編集履歴の収集が可能になり、開発環境が限定される問題や蓄積された履歴データが活用されないという問題が解決する。

[D] Languageserver.org: A community-driven source of knowledge for language server protocol implementations. https://langserver.org/.

概念図

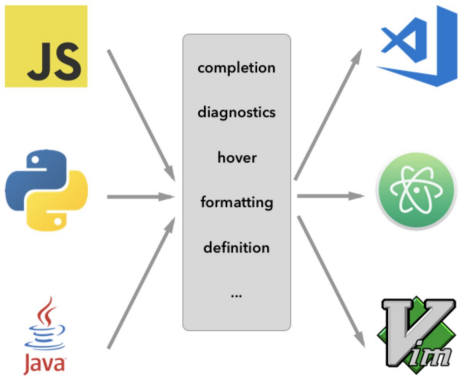


実装

細粒度編集履歴の収集・処理・送信を
担うものを**LSPの言語サーバー**として実装している。

特徴: Language Server Protocol (LSP) の使用

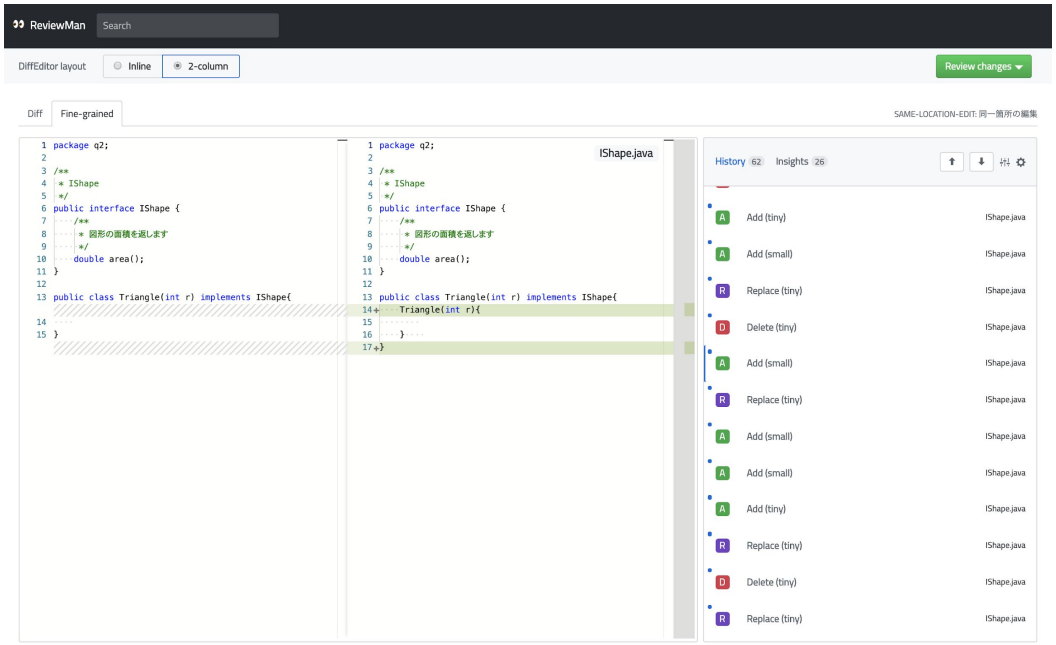
- 本来はエディタと言語サーバーのやり取りを標準化したもの。主要なエディタでサポート済み。
- インクリメンタルにコードの変更を取得する機能が存在。
- 実体はJSON-RPC(Remote Procedure Call)であり拡張可能。



LSPの概要図

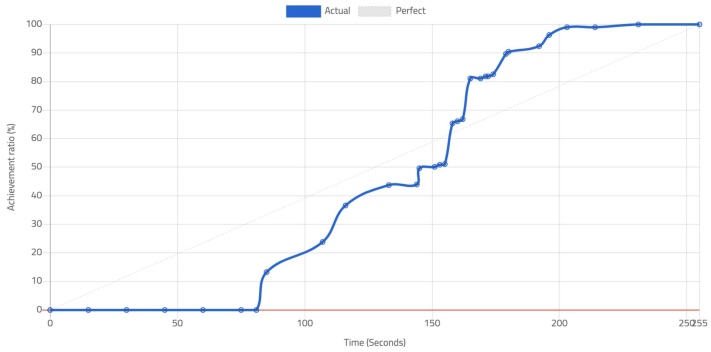
KAMAKURAを使用した2つのアプリケーション

KAMAKURAを使用した2つのアプリケーション



1. ReviewMan

コードレビューツール



2. Kudo

作業進捗可視化手法

ReviewMan コードレビューツール

背景

従来コードレビューの目的は欠陥の発見が主であったが、現代ではコードレビューの目的は広範囲に渡っており特に教育的側面が大きくなってきている[E]。

課題

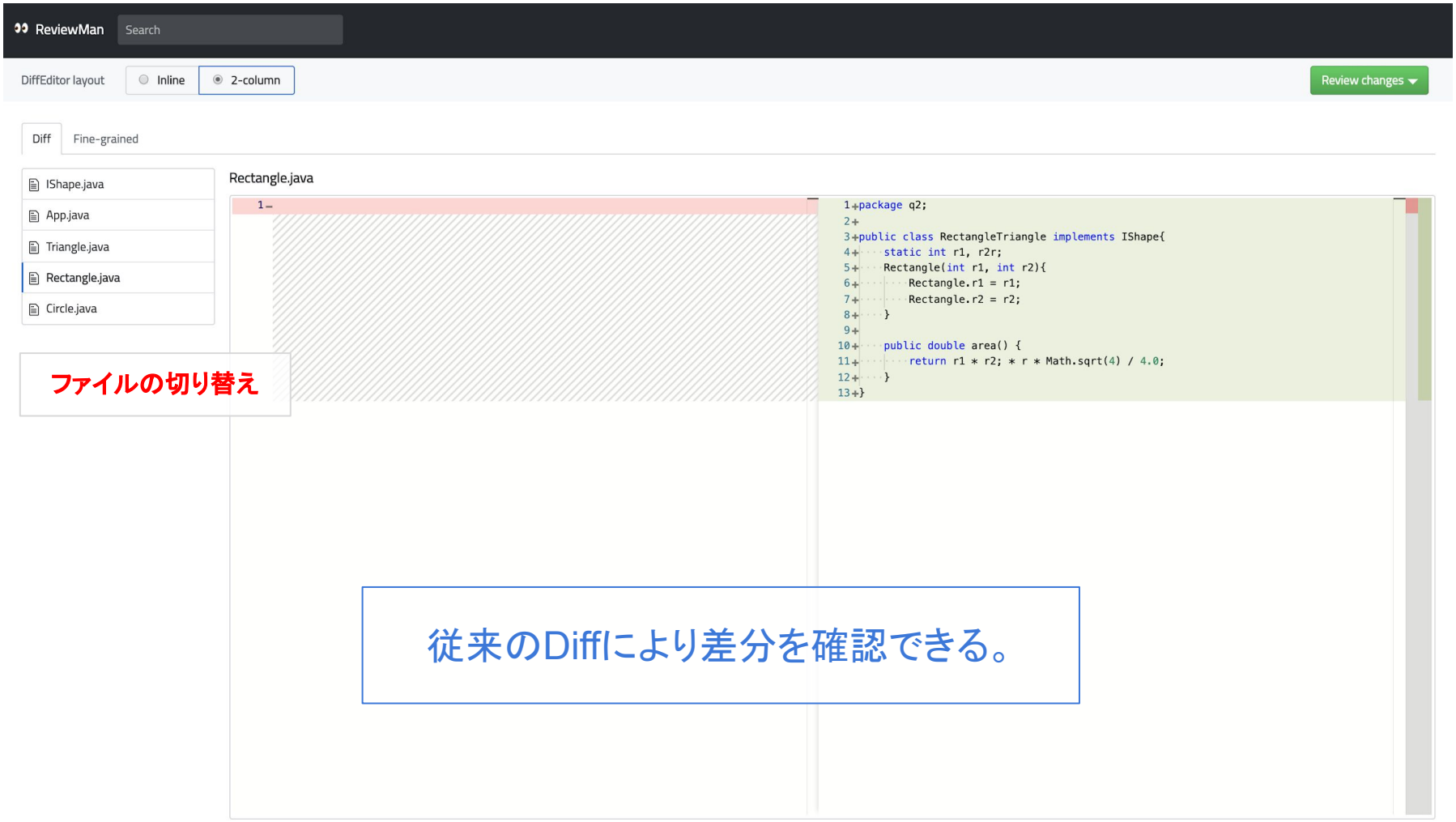
しかしながらバージョン管理システムに記録される履歴には欠損がある。また、どのようにコードが書かれたまでは分からない。よって開発者の意図を十分に理解することが難しく、教育的観点においても好ましくない。

解決

従来のDiffによる環境に加えて、細粒度編集履歴を使用して開発者がどのようにコードを書き進めたのかを再生したり編集履歴から改善パターンを自動的に発見したりできるコードレビュー環境を提供する。

[E] Caitlin Sadowski, Emma S¨oderberg, Luke Church, Michal Sipko, and Alberto Bacchelli. Modern code review: A case study at google. In International Conference on Software Engineering, Software Engineering in Practice track (ICSE SEIP), 2018.

Diff モード

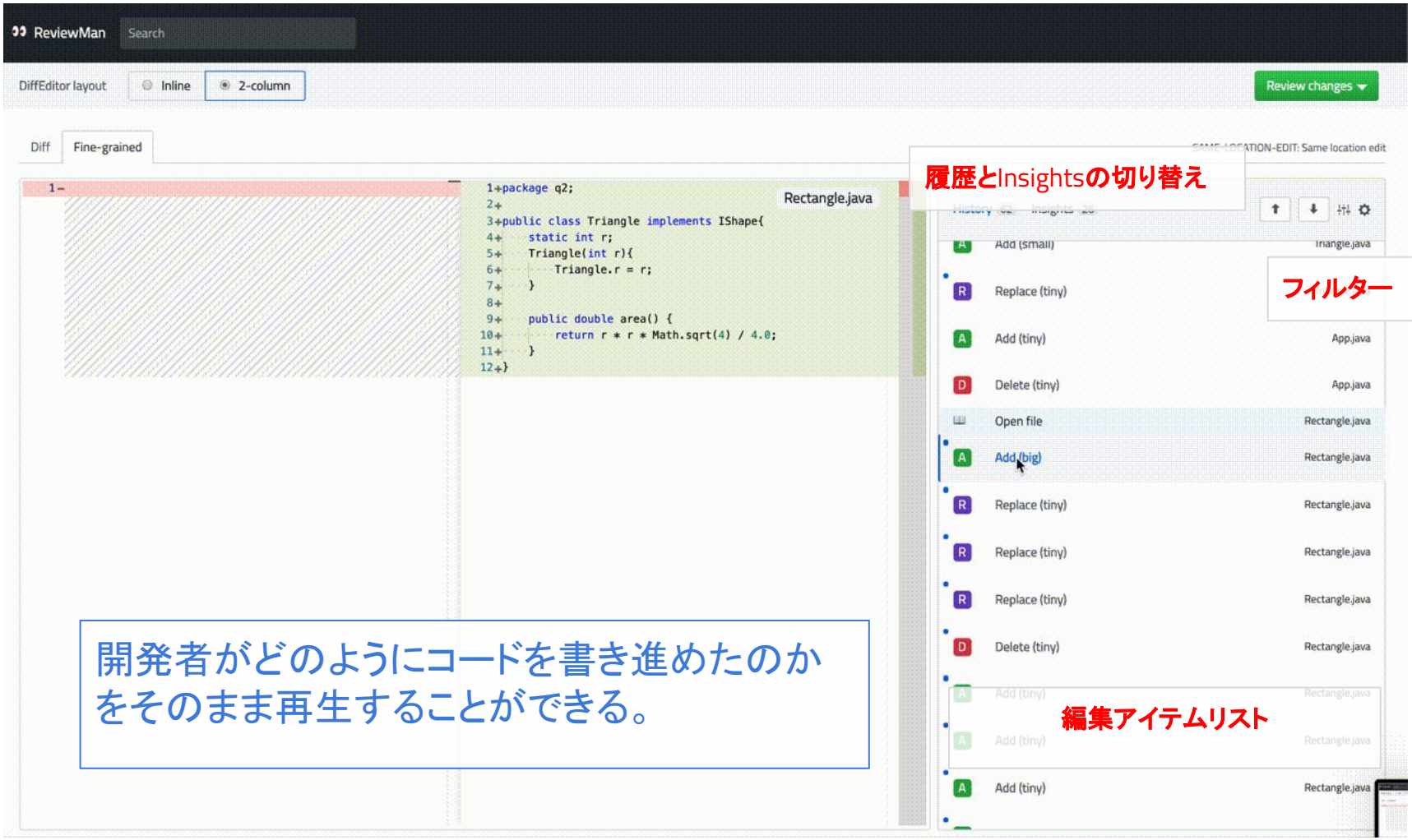


ファイルの切り替え

従来のDiffにより差分を確認できる。

Fine-grained モード

NEW



Insights

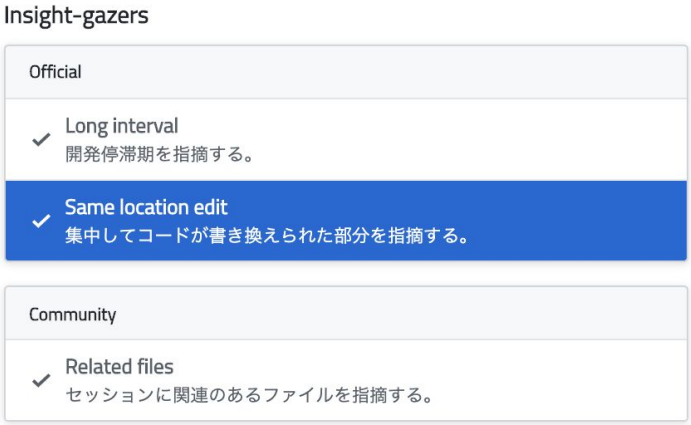
細粒度編集履歴から共通した改善パターンを自動的に発見することで、大量に存在する編集アイテムをレビューする労力を減らす。

Insightを抽出するものをInsight-gazerといい、APIが開放されているので自身で開発・共有が可能である。

Insightの表示例

Insight-gazerの例

Same location edit	集中してコードが書き換えられた部分を指摘する。
Long interval	開発停滞期を指摘する。



Insight-gazerの選択画面

ケーススタディ
(ReviewMan)

概要

プログラミングの問題を解く過程を題材にしてReviewManを使ってコードレビューを行った。各問題で使用するモードは下記のパターンで指定した。

* Diffモード のみ
* Fine-grainedモード のみ
* 両方のモード

ケーススタディ

1. 有効なレビュー項目数
2. レビュー時間
3. フリーコメント

実験設定

被験者数
4

被験者の所属
情報科学研究科の学生

制限時間
~20分

問題数
3

結果・考察 (1/2)

1. 有効なレビュー項目数

仮説

Fine-grainedモードの方がより詳細な開発履歴を見られるため常に多くなる。

結果・考察

特に数の有意な差はみられなかった。
しかしながら、**すべての被験者がFine-grainedモードでないと気づけない点を指摘することができていた。(例: あるクラスをコピー・ペーストして他のクラスに作り変えている部分でシンボル名を変更する時はエディタの一括置換機能を使った方がよい。)**

結果・考察 (2/2)

2. レビュー時間

仮説

Diff モード < 両方 < Fine-grained モード

結果・考察

制限時間いっぱい使用した被験者2名を除けば仮説通りになった。時間差は難易度が高い問題ほど小さくなる傾向が見られた。Fine-grainedモードの方が開発者の意図が分かりやすいという利点が活かせるためと考えられる。

3. フリーコメント

肯定的なコメント

- 全体の変更を把握するときのDiffモードと詳細に調べたいときのFine-grained モードという**使い分けが有効に感じた。**
- **より効率の良い開発スタイル**につながるのではないかと思った。

否定的なコメント

- Fine-grainedモードだけでレビューすると強いストレスを感じた。

まとめと今後の課題

まとめと今後の課題

まとめ

1. 細粒度編集履歴収集プラットフォームKAMAKURAの提案・開発を行った
2. KAMAKURAをコードレビューツールのReviewManと作業進捗可視化手法のKudoに適用した。
3. ケーススタディにおいて、ReviewManではFine-grainedモードをDiffモードと併用することでより質の高いレビューを支援すること、Kudoでは作業進捗状況をよく表現できることを確認した。

今後の課題

- KAMAKURAの統一された記録手法を活かし、既存の細粒度編集履歴関連ツールをKAMAKURAに移植できることを示す。
- KAMAKURAを使用して細粒度編集履歴を大量に収集し、ソフトウェア進化等の研究に貢献する。