2K-01

少人数開発向けのドキュメントとソースコードの軽量な乖離抑制手法

山縣 寿樹† 酒井 三四郎‡

静岡大学大学院総合科学技術研究科† 静岡大学情報学部‡

1. はじめに

近年のソフトウェア開発では，ソフトウェア品質を保つために，ドキュメントを作成することが重要視されている．しかし少人数のソフトウェア開発では，ビジネス要求を素早く実現するために開発スピードが強く求められており，ドキュメントとソースコードが乖離しやすく，ソフトウェア品質の低下に繋がりかねない．

少人数のソフトウェア開発の特徴として，開発手法にスクラム開発を採用し，短期間単位での機能の開発および修正を行うことが多い．ソフトウェアは Git やそのホスティング先に GitHub などを活用しバージョン管理を行い，コミュニケーションツールに Slack などを利用し，その他 CI/CD ツールなどを活用することで開発スピードの向上に力を入れている．

本研究では，少人数のソフトウェア開発において開発スピードを落とすことなくドキュメントとソースコードが乖離するのを防ぐことを目的とする．そのために，乖離の可能性（乖離リスク）のある箇所の特定および通知を行うことのできるソフトウェアの開発を行った．また，バージョン管理システムである Git のコミット履歴を用いたシミュレータの開発を行い，実際のソフトウェア開発を想定したシミュレーションを行った．

2. 関連研究

ドキュメントとソースコードの乖離を抑制するための研究として，ドキュメントとソースコードを一元管理する研究が行われている[1]．

関連研究と本研究との相違点に，提案ツールは，ライブラリやフレームワークに依存しないため，既存のソフトウェア開発プロジェクトに無理なく組み込むことができるという点が挙げられる．また，提案ツールを利用するためには，CI/CD の設定ファイルに数行追加するだけでよいため，非常に軽量である．加えて，利用者は手に馴染んだエディタを使用することができ，開発環境を大幅に変更する必要がない．

Light-Weight Divergence Suppression Method of Documentation and Source Code for Small Team Development

†Kazuki YAMAGATA, Graduate School of Integrated Science and Technology, Shizuoka University

‡Sanshiro SAKAI, Faculty of Informatics, Shizuoka University

3. 提案手法

本研究では，ドキュメントとソースコードの乖離を抑制するために，乖離リスクのある箇所を特定し，コミュニケーションツールを使用してユーザーに通知を行うことができるツールを提案する．これにより通知を受け取ったユーザーは通知内容をもとに，乖離リスクのあるドキュメントまたはソースコードを修正することを期待する．

提案ツールのアーキテクチャを図 1 に示す．提案手法を実現するために，CI/CD 上で乖離の検証を行うプログラムとプロジェクトの状態を管理およびコミュニケーションツールへの通知を行う RESTful API を開発した．また，実験を簡易的に行うために，ローカル PC 上で動作させた RESTful API を HTTPS として外部公開させることができるツールの ngrok を使用した．

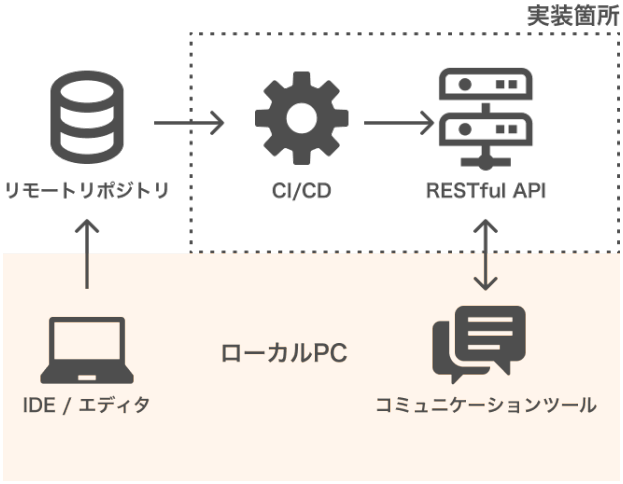

図 1: 提案手法のアーキテクチャ図

提案手法を使用したときの開発の流れは下記のとおりである．通常のソフトウェア開発と何ら変わりないことがわかる．

(1) ローカル PC でコーディングやドキュメントの作成・修正を行う
(2) 変更したファイルをコミットする
(3) リモートリポジトリに変更を書き込む
(4) CI/CD 上で乖離リスクのチェックが行われる
(5) 乖離リスク特定時、コミュニケーションツールに通知が飛ぶ

本研究では，以下の4つの機能を実装した．各機能はプロジェクトごとに許容するリスクや性質が異なるため，コミュニケーションツールを通じて各パラメータを調整することができるように実装した．

3.1. ソースコード先行の検知

ドキュメントとソースコードの対応付けを，アノテーションを用いて行い，ドキュメント中に記述されていない機能をソースコードに実装した際に，乖離リスクとしてユーザーに通知する．この機能RESTful API開発に利用可能であり，プログラミング言語にPython，ドキュメントにSwaggerを選択する必要がある．

3.2. 一定時間経過後のリマインダー

ドキュメントが作成され一定時間経過した後，ソースコードは更新され続けているがドキュメントは更新されていない場合に，ユーザーに通知する．

3.3. リリース時ドキュメント更新有無の検知

新たなバージョンがリリースされたとき，前回のバージョンリリース時よりドキュメントが更新されていない場合に，ユーザーに通知する．

3.4. ソースコード変更量の検知

新たなファイルを作成，既存のファイルを変更・削除をしたとき，ある一定量の変更が行われていた場合に，ユーザーに通知する．

4. シミュレーション

提案ツールの有用性を検証するため，Gitのコミット履歴を用いたシミュレータの開発を行った．シミュレーションの対象として，GitHub上にあるOSS(オープンソースソフトフェア) のプロジェクトを利用した．対象としたプロジェクトはPythonのパッケージ管理および仮想環境を構築可能なPipenvとPythonの画像処理ライブラリであるPillowである.シミュレーションでは，1万件のコミット履歴をもとに3節にまとめた機能について当時の乖離リスクを検証する．ただし，ソースコード先行の検知機能については，適切な評価をすることができるプロジェクトが存在しなかったため除外した．本機能は評価実験にて有用性を検証した．

シミュレーションの結果，Pillowでは一定時間経過後のリマインダーが24回，ソースコード変更量の検知が32回，Pipenvでは一定時間経過後のリマインダーが17回，ソースコード変更量の検知が451回呼び出されることがわかった．またリリース時ドキュメント更新有無の検知機能については呼び出されることがなく，その他機能については一定の効果が見込めることが判明した．

5. 評価実験と考察

ソースコード先行の検知機能の有用性を検証するために，実験協力者2名に前半5個，後半5個の合計10個のタスクをこなしてもらった．各タスクにはAPIサーバーからリソースを取得およびリソースを登録するための機能追加や，入力値のバリデーションの実装・修正をするといった指示が書かれている．実験協力者は前後半のどちらかで提案ツールを使用するが，実験協力者にはどちらで使用するかは明示しなかった．また評価実験では，ドキュメントとソースコードが乖離していないかを評価の対象とした．

評価実験の結果，提案ツールを前半に使用した1名の実験協力者は，前半タスクを2つこなした後に，提案ツールが乖離リスクを検知し通知を行った．このとき，実験協力者がドキュメントの修正を行ったため，前半タスク終了時にはドキュメントとソースコードの乖離が抑制された．提案ツールを後半に使用した実験協力者は，前半タスク終了時にはドキュメントを記述することはなかったが，後半タスク開始時に乖離検知の通知を受け取った後，ドキュメントの修正を行った．実験協力者2名とも，実験終了時にはドキュメントとソースコードが乖離することはなかった．

またアンケートより，通知を受け取ったことで，忘れてしまっていたドキュメントの修正を行うことができたといった結果を得た．

シミュレーションおよび評価実験の結果を踏まえると，提案ツールを使用したソフトウェア開発では，乖離リスクを抑制するのに一定の効果があると期待できることが判明した．

6. おわりに

本研究では，ドキュメントとソースコードの乖離の可能性のある箇所の特定および通知を行うことのできるツールを開発した．また提案ツールの有効性を検証するためにシミュレータの開発し，OSSのプロジェクトを対象にシミュレーションを行った．今後の課題として，様々なプログラミング言語の対応や，より正確な乖離リスクの特定を行う必要がある．

参考文献

[1] 赤石裕里花，坂井麻里恵，奥野拓，伊藤恵: 整合性維持に着目したソースコードとドキュメントの一元管理環境の提案，日本ソフトウェア科学会第30回大会,2013