目次

第 1 章　序論 2

第 2 章　関連研究 4
2.1　ドキュメントとソースコードの一元管理 4
2.2　ドキュメント自動生成 4
2.3　本研究との相違点 5

第 3 章　提案手法 6
3.1　提案ツール 6

第 4 章　設計と実装 8
4.1　提案ツールの設計 8
4.2　開発環境 8
4.3　実装規模 9
4.4　提案ツールのアーキテクチャ 9
4.5　提案ツールの使用方法 10
4.6　コミュニケーションツールで利用可能なコマンド 10

第 5 章　評価実験 12
5.1　仮説 12
5.2　実験計画 12

第 6 章　結果と考察 14
6.1　実験結果と考察 14
6.2　アンケート結果と考察 14

第 7 章　結論 15

第 1 章

序論

近年のソフトウェア開発では，ソフトウェア開発における複雑性が増し，少人数の開発チームへと分割されていく中で，開発者は短期間でより多くのを作業をしなくてはらならない状況にある．そのため，分割されたチーム間で円滑な連携を行うことや，ソフトウェア品質を保つためにドキュメントを整備することが重要視されている．また，2020 年初旬からの新型コロナウィルス感染症拡大により，ソフトウェア開発の現場ではリモートワークを中心としたテキストベースでのやり取りも増えたため，より一層ドキュメントを充実させる必要性が高まった．このような状況において，開発スピードを求められるソフトウェア開発の現場では，ドキュメントとソースコードの整合性を維持することが困難となってきた．

ソフトウェア開発の現場では，バージョン管理システムを使用したソフトウェアのバージョン管理，CI/CD(継続的インテグレーション/継続的デリバリー) を活用したソフトウェアのビルド・テスト・デプロイなどの自動化，円滑なやりとりを行うためのチャットツールの導入など，開発スピードの向上に力を入れている．また，同じ開発チームであったとしても，フロントエンドの開発，バックエンドの開発，モバイルアプリケーションの開発，デザインの調整など開発者の役割は様々であり，開発チームの内外問わずに円滑なやりとりを行う必要がある．このため，ソフトウェア開発で使用するドキュメントは常に最新のものであることが求められている．

開発者は，ビジネス要求を素早く実現するために，機能の追加や修正を優先してしまい，ドキュメントの更新を後回しにしてしまう事例は少なくない．例えば，誤って古いドキュメントを参照して開発を進めた場合，後の段階で手戻りが発生していしまうため，開発スピードを大幅に落としてしまう原因となってしまう．また，普段と異なる開発環境では開発者は自身の能力を最大限発揮することは難しい．したがって，ライブラリやフレームワーク，エディタに依存せずに，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ツールを作成する必要があると考えた．

以上を踏まえた上で，ドキュメントとソースコードが乖離している可能性のある箇所を特定および開発者に通知をすることができる機能をもった支援ツール milk を開発した．milk はライブラリやフレームワーク，エディタに依存しないツールである．同時に，milk の有用性を検証するために，Git のコミット履歴を使用し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．シミュレーションで検証できなかった機能については，実際にプログラミング経験が豊富な実験協力者に milk を使用してもらい，その有用性を確かめる評価実験を行った．

本論文の構成は以下のとおりである．2 章で，本研究の関連研究について示し，本研究との相違点を述べる．3 章では本研究で提案するツールについて，4 章ではその設計と実装について述べる．5 章では，提案ツールの有用性検証するためのシミュレーションおよび評価実験について述べる．6 章では，評価実験の結果と考察について述べ，7 章で本研究の結論を述べる．

第 2 章

関連研究

本章では，本研究に関連する研究および技術と，本研究との相違点について述べる．

2.1 ドキュメントとソースコードの一元管理

赤石らの提案するドキュメントとソースコードの一元管理ツールは，開発者が XML 形式のドキュメントを作成した後，それに従ってソースコードを作成することで，ドキュメントとソースコードの整合性を維持することを期待する．作成したドキュメントとソースコードは，ビュワーを通じて閲覧を行うことができる．ビュワーではドキュメントとソースコードが混在した形で近い場所に表示されるため，開発者はドキュメントとソースコードの乖離に気づきやすい．しかし，ドキュメントとソースコードの整合性を維持するためには，赤石らの提案するエディタおよびビュワーを使用しなくてはいけないため，開発者の学習コストや開発スピードの低下などが懸念される．また，ドキュメントとソースコードの対応付けを行う方法や，提案ツールの評価，ビュワーの見た目などが未実施である．

2.2 ドキュメント自動生成

ソースコード中に適切なアノテーションやコメント文を挿入することで，ドキュメントを自動生成することのできるライブラリやフレームワークが存在する．それぞれのライブラリやフレームワークでドキュメントを自動生成するための書き方は異なるが，いずれも適切な箇所で適切なフォーマットでコーディングする必要がある．以下では，ドキュメントを自動生成することができるライブラリおよびフレームワークを紹介する．

2.2.1 Javadoc

Javadoc は，JDK に標準搭載された機能で，適切なフォーマットに従って作成したコメント文やソースコードから HTML 形式のドキュメントを自動生成するツールである．プログラミング言語に Java を使用したソフトウェア開発では，ソースコードの API 仕様書としてに Javadoc が使

用されることも多い．

2.2.2 Spring Boot

Spring Boot は，Web アプリケーション開発を行うことのできる Java の Web フレームワークである．REST API を実装するときに，Swagger と呼ばれるドキュメントを自動生成する機能が存在する．

2.2.3 FastAPI

FastAPI は，API サーバーを構築することのできる Python の Web フレームワークである．FastAPI には Swagger と ReDoc と呼ばれるドキュメントを自動生成する機能が存在する．

2.3 本研究との相違点

赤石らの研究で提案されたドキュメントとソースコードの一元管理を行うツールでは，目的は同じであるものの，プログラミング言語やエディタが限定されるのに対し，本研究では，ライブラリやフレームワーク，エディタに依存せずに，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った点で異なる．ドキュメントを自動生成するライブラリやフレームワークでは，技術選定の段階で制限されてしまうのに対し，本研究では，現在進行中のプロジェクトに無理なく組み込むことができるといった点で異なる．

ソースコードに直接関係のないドキュメント，例えば，開発プロジェクトのアーキテクチャの概念図やプロジェクトの方向性，補助的な説明などを記載するドキュメントは，通常ソースコード中には記載せずに，別途ドキュメント専用のファイルなどに記載するため，自動生成がすべてをカバーできないことに留意する．これらのドキュメントも，プロジェクトが進み古くなってしまうと，ドキュメントとソースコードが乖離してしまう可能性がある．

本研究ではドキュメントとソースコードが乖離している可能性のある箇所を特定および開発者に通知をすることができる機能をもったツールの開発を行う．

第 3 章

提案手法

本性では，本研究で提案するツール「milk」とシミュレータについて述べる．

3.0.1　想定利用者

1 章で述べたとおり，ソフトウェア開発のチームでドキュメントとソースコードの乖離を抑制することが目的なため，

3.1　提案ツール

本節では，提案ツールの目的，要求仕様について述べる．

3.1.1　提案ツールの目的と要求仕様

提案ツールの役割は，ドキュメントとソースコードの乖離リスクのある箇所または原因を特定し，開発者への通知を行ったときに，通知内容をもとに開発者がドキュメントまたはソースコードを修正することで，乖離を抑制することを期待する．

これを実現するためにツールには以下の機能をもたせる．このとき，それぞれの機能に C1〜C4 と名付けた．いずれの機能も，リモートリポジトリである GitHub 上にプッシュされた際に，CI/CD ツール上で動作する milk が乖離リスクを検知する仕組みとなっている．

3.1.2　C1：ソースコード先行の検知の機能

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，乖離リスクとして開発者に通知する．本機能は，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．開発者は，この通知を受け取ったときに，ドキュメントに適切なものを追加するか，誤ったアノテーションを修正し，ドキュメントとソースコードの乖離を抑制す

ることを期待する．

3.1.3　C2：一定時間経過後のリマインダーの機能

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，乖離リスクとして開発者に通知する．本機能は，ドキュメントとソースコードが同じリポジトリで管理されることを想定している．また，Markdown 形式ファイルや reStructuredText 形式ファイル，単純なテキストファイルなどをドキュメントとして利用する必要がある．開発者は，この通知を受け取ったときに，ドキュメントが古くなっていないかを確認し，古くなった箇所を修正するか，または現状維持でよいのかを判断し，ドキュメントとソースコードの乖離を抑制することを期待する．

3.1.4　C3：リリース時ドキュメント更新有無の検知の機能

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，乖離リスクとして開発者に通知する．本機能は，一定時間経過後のリマインダーの機能と同様に，ドキュメントとソースコードが同じリポジトリで管理されていることを想定している．開発者は，この通知を受け取ったときに，前回のバージョンリリース時との変更点をドキュメントに記述するか，または現状維持でよいのかを判断し，ドキュメントとソースコードの乖離を抑制することを期待する．

3.1.5　C4：ソースコード変更量の検知の機能

新たなファイルを作成，既存のファイルを変更・削除したとき，ある一定量の変更が行われた場合に，乖離リスクとして開発者に通知する．本機能は，一定時間経過後のリマインダーの機能と同様に，ドキュメントとソースコードが同じリポジトリで管理されていることを想定している．開発者は，この通知を受け取ったときに，ドキュメントが古くなっていないかを確認し，古くなった箇所を修正するか，または現状維持でよいのかを判断し，ドキュメントとソースコードの乖離を抑制することを期待する．

第 4 章

設計と実装

本性では，提案ツールを開発するときのアプローチを述べる．

4.1 提案ツールの設計

4.2 開発環境

提案ツールの開発環境を下記に示す．

- OS：macOS Big Sur 11.1
- プログラミング言語：Python 3.7
- 統合開発環境：PyCharm 2020.3.2

4.3　実装規模

4.4　提案ツールのアーキテクチャ

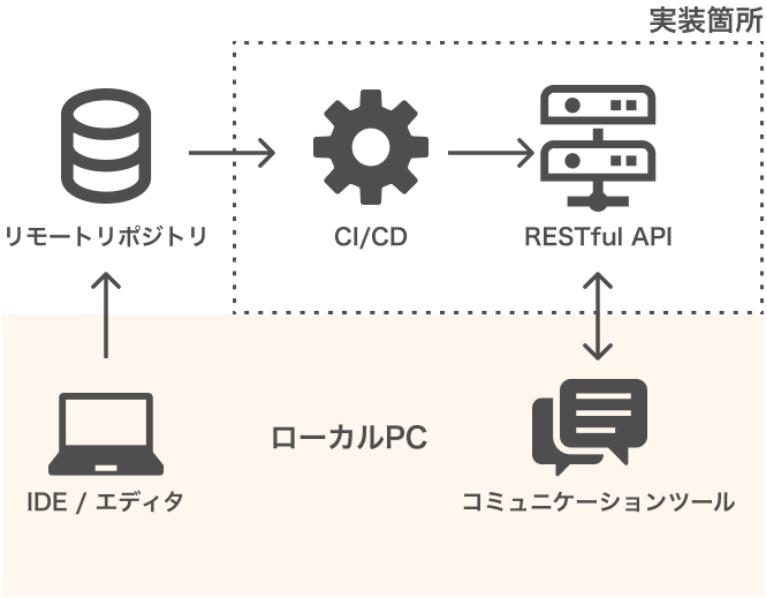

図 4.1　提案ツールのアーキテクチャ

提案ツールのアーキテクチャを図 4.1 に示す．提案ツールの milk は，既存の CI/CD ツール上で動作することを想定しており，CI/CD ツールには GitHub Actions を用いた．CI/CD ツール上でできることが限られているため，同時に，Slack とのやりとりや，開発プロジェクトの状態の管理を行うことができる RESTful API の開発も行った．RESTful API には FastAPI を使用し，各種データの保持は JSON ファイルの I/O にて管理を行っている．また，RESTful API は筆者のローカル PC 上で動作させているが，HTTPS として外部公開させることができるツールの ngrok を使用しており，Slack API や CI/CD 上で動作する milk はローカル PC 上で動作する RESTful API を使用できるようになっている．

RESTful API には，下記の機能をもたせているが，乖離検知の処理はすべて milk で行う

- 各機能のステータスを更新する機能
- Git のコミットの差分量を追加する機能
- Slack の API を呼び出してメッセージを送受信する機能

開発者は，適切なタイミングで変更を記録するためにコミットを実行し，リモートリポジトリに

プッシュすることを期待している．リモートリポジトリにプッシュされたタイミングで，CI/CD ツールが起動し，CI/CD ツールの設定ファイルに定義されたコマンドが実行される．このとき，Python で作成した milk コマンドを実行するために，CI/CD ツール上で Python 環境を構築している．乖離リスクの検知を行うために，まず RESTful API から開発プロジェクトの現状態を取得する．その後，C1 C4 について乖離リスクの検知を行い，検知したときに RESTful API を通じて Slack へとメッセージを送信する．

Slack に milk コマンドが入力されると，Slack から RESTful API に HTTP リクエストを行われるため，これを適切にハンドリングすることで JSON ファイルに値を書き込むことができる．

4.5 提案ツールの使用方法

4.6 コミュニケーションツールで利用可能なコマンド

milk では，Slack のスラッシュコマンドを活用することで，各種パラメータの設定を行うことができるようになっている．以下では，milk で使用可能なコマンドをまとめる．

/milk c1 doc [file type]

ソースコード先行検知の機能において，開発プロジェクトで使用する Swagger ドキュメントのファイル形式を指定することができるコマンドである．本コマンドは，開発プロジェクトの初期段階で Swagger ドキュメントのファイル形式を指定する場合や，開発プロジェクトの途中でファイル形式を変更したい場合に利用されることを想定している．

/milk c1 code [file type]

ソースコード先行検知の機能において，開発プロジェクトで使用するプログラミング言語を指定することができるコマンドである．現在では，プログラミング言語に Python のみを指定可能である．

/milk c1 param

ソースコード先行検知の機能において，現在設定している Swagger ドキュメントの拡張子および使用しているプログラミング言語の拡張子を確認することができるコマンドである．

/milk c2 set [days]

一定時間経過後のリマインダーの機能において，ドキュメントが更新されずソースコードのみが更新されている場合に，ドキュメントを追加・修正するためのリマインダーを行う間隔を指定することができるコマンドである．

/milk c2 param

一定時間経過後のリマインダーの機能において，現在設定しているリマインダーを行う間隔を確認することができるコマンドである．

/milk c3 set [version file]

リリース時ドキュメント更新有無の検知の機能において，バージョンを扱うファイルを指定することができるコマンドである．例えば，Python では setup.py と呼ばれるファイルにソフトウェアのバージョンを記述している．現在では，バージョンを扱うファイルに setup.py のみを指定可能である．

/milk c4 set [number]

ソースコード変更量の検知の機能において，ドキュメントが更新されずソースコードのみが更新されている場合に，ソースコードの追加・削除・修正が行われた行数を指定することができるコマンドである．指定した行数を超過してソースコードのみを記述し続けたときに通知がでる．

/milk c4 param

ソースコード変更量の検知の機能において，現在設定している変更量を確認することができるコマンドである．

第 5 章

評価実験

本章では，提案ツールの有効性を検証するために行った実験について述べる．

5.1 仮説

本実験では以下の仮説を検討する．

仮説 1 提案ツールを使用することで，ドキュメントとソースコードの乖離を抑制できる

5.2 実験計画

本実験では，実験用に用意したプロジェクトに実験協力者が途中から参加し，ドキュメントとソースコードの追加・変更を行ってもらう．このとき，実験協力者は前半 5 個，後半 5 個の合計 10 個のタスクをこなし，前半で提案ツールを使用する郡と後半で提案ツールを使用する郡に分けた．また，実験後にはアンケートに回答してもらう．ドキュメントとソースコードが乖離，アンケート結果，コミット履歴から提案ツールの有効性の検証を行う．

5.2.1 実験協力者

本実験での実験協力者は，静岡大学総合科学技術研究科情報学専攻の学生 2 名で，両名とも Python と Git を利用することができる．評価実験は，1 名は 2020 年 12 月 26 日と 12 月 27 日に実験を行い，もう 1 名は 1 月 6 日と 1 月 7 日に実施した．本実験では，前半に提案ツールを使用し，後半に提案ツールをしない 1 名と，前半に提案ツールを使用せず，後半に提案ツールを使用する 1 名に振り分けた．

5.2.2 実験で使用したプロジェクト

本実験で使用したプロジェクトを付録 A に示す．実験では，前半 5 個，後半 5 個の合計 10 個のタスクを用意しており，各タスクではソースコードとドキュメントの追加や修正を行う必要がある．また，前後半で難易度に大きな差がでないように調整した．

5.2.3 実験の流れ

評価実験は下記の流れで行った．実験協力者は，各タスクを終えるたびに変更履歴をコミットし，リモートリポジトリである GitHub にプッシュする．提案ツールを使用している場合，ドキュメントとソースコードで乖離を検知すると Slack から通知を受け取ることができる．また，実験では実際のソフトウェア開発現場で途中から参加した開発者であることを想定し，実験協力者がタスクに取り組むにあたって技術的にわからないことはいつでも質問してよいこととした．

一人目の実験協力者は下記の流れで実験を行った．

1. 本実験の内容の説明
2. 前半タスク (提案ツールを使用)
3. 後半タスク (提案ツールを使用しない)
4. 事後アンケート

また，二人目の実験協力者は下記の流れで実験を行った．

1. 本実験の内容の説明
2. 前半タスク (提案ツールを使用しない)
3. 後半タスク (提案ツールを使用)
4. 事後アンケート

第 6 章

結果と考察

6.1　実験結果と考察

6.2　アンケート結果と考察

第 7 章

結論

本研究では，〜〜ために，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の開発を行った．ドキュメントとソースコードが乖離してしまう原因は，〜〜であると考えた．

ライブラリやフレームワーク，エディタに依存せずに，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必要があると考えた．

今回提案したツールの効果を検証するために，Git のコミット履歴を用いたシミュレーションおよび実験協力者に提案ツールを使用してもらい実験を行った．その結果，〜〜アンケート結果から〜〜

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．

謝辞

本研究の全過程を通じて，丁寧なご指導をいただいた静岡大学情報学部情報科学科の酒井三四郎教授に深く感謝するとともに，厚く御礼申し上げます．また，本研究の期間中にご協力いただいた酒井研究室の皆様，ならびに実験に協力していただいた皆様に心から感謝いたします．