2020年度修士論文

# 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

指導教員：酒井三四郎

学籍番号
71930055

山縣寿樹

静岡大学総合科学技術研究科
情報学専攻

2021年2月

## 概要

少人数のソフトウェア開発において，ビジネス要求を素早く実現するためにソースコードの追加・修正に追われ，ドキュメントの追加・修正を忘れてしまい，ソフトウェア品質を落としてしまうといった状況が見られる．これは，ソースコードからドキュメントを自動生成する技術を使用したり，ドキュメントとソースコードが乖離したときに開発者に通知を行うことで対応ができる．しかし，ソースコードからドキュメントを自動生成するためには，ライブラリやフレームワークに依存してしまうといった問題点があり，ドキュメントとソースコードが乖離したときに通知を行うためには手間とコストがかかる問題点がある．

以上より，ライブラリやフレームワーク，エディタに依存しない，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．ツールを利用するためには，CI/CD ツール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．また，ツールの有効性を検証するために，Git のコミット履歴を用いたシミュレータの開発および評価実験を行った．Python が使用された OSS のプロジェクトを対象にシミュレーションを行ったところ，乖離リスクのある箇所に対して通知を行うといった結果が得られた．評価実験では，実験協力者 2 名に対し，前半 5 個，後半 5 個の合計 10 個のタスクをこなしてもらい，前後半のいずれかでツールが動作する実験を行った．このとき，ドキュメントとソースコードの乖離が抑制されるのかを評価の対象とした．また，実験終了後にアンケートを実施した．実験の結果，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く，一定の効果が見込めることが判明した．また，アンケート結果から，「ドキュメントの更新を忘れてしまったとき，通知は必要である」といった回答を得ており，ドキュメントとソースコードの乖離抑制に役立つことができたと考えられる．

# 目次

# 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を開発した．これ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，ツール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ツールを使用してもらい，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 [1] 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 ドキュメントとソースコードを対応付ける研究

後藤らのドキュメントとソースコードの対応付けを行う研究 [2] では，ドキュメントを XML 形式で記述したものを扱い，XML のタグとソースコード中の属性値を対応表を用いて対応付けを行う．これによって，ドキュメントとソースコードの整合性を検査できるようになった．しかし，ドキュメントまたはソースコードのどちらかの編集に対して，インタラクティブな機能を提供することができず，例えば，先にドキュメントを作成してからソースコードを記述する方法では不整合となってしまう問題がある．

## 2.3 ドキュメントからソースコードを生成する研究

小池のソースコードからソースコードを生成するフレームワーク SLAF[3] では，ドキュメントとソースコードの乖離の発生は防げないといった考えから，ドキュメントからソースコードを生成することができる．具体的には，プログラミング言語の知識を必要としないドキュメントを記述することで，それに従って動作する Java のソースコードが生成される．これによって，常にドキュメントとソースコードの整合性が維持されることになった．しかし，生成されたソースコードでは性能低下が見られることや，現実的に運用することが難しいといったことから課題も多い．

海老澤らの WebComponents 開発における API ドキュメントとソースコードを自動生成する研究 [4] では，1 つのファイルに，HTML, CSS, JavaScript と API ドキュメントを記述することで，ドキュメントとソースコードを同時に管理することのできるシステムを開発した．これは，既存のシステム開発に組み込むことは難しく，新たなプロジェクトで有効に作用すると考えられる．

## 2.4 ドキュメントを自動生成する技術

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．

### 2.4.1 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[7] が使用されることも多い．Java のソースコードからドキュメントを生成するため，例えば，プロジェクト全体の方針，アーキテクチャなどの概念図，各機能の設計といった内容のドキュメントを記述する場合は不向きである．

### 2.4.2 Spring Boot

Spring Boot[8] 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．RESTful API を実装するときに，Swagger[9] 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．Java で記述するソースコード中に適切な箇所で適切なアノテーションを埋め込むことで，ソースコードから自動でドキュメントを生成することができる．

### 2.4.3 FastAPI

FastAPI[11] 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[10] 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．Spring Boot と同様に，ソースコード中に適切な箇所で適切なアノテーションを埋め込むことで，ソースコードから自動でドキュメントを生成することができる．また，追加のプラグインやライブラリを必要とせずに，元からある機能として組み込まれているため，簡単に利用することができる．しかし，Swagger や ReDoc 以外のドキュメントを生成することができないため，その他ドキュメントに関しては，開発者が意識して，ドキュメントとソースコードの整合性を維持しなくてはならない．

## 2.5 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．また，その他関連研究や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．

また，ドキュメントを自動生成することができるライブラリやフレームワークでは，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

本章では，本研究で提案するツールについて述べる．

## 3.1 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，ソフトウェア開発を行う少人数（3 名〜6 名程度）の開発チームを対象としている．大人数で行うソフトウェア開発では，プロジェクトで使用する技術やツールの自由度が制限されることが多いことや，少人数のソフトウェア開発チームに分割されていくことを踏まえて，提案ツールは少人数のソフトウェア開発に焦点を当てて開発した．

## 3.2 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．

### 3.2.1 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ツールが乖離リスクを検知する仕組みとなっている．

### 3.2.2 C1：ソースコード先行の検知の機能

アノテーションを用いてドキュメントとソースコードの対応付けを行い，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．

具体的な仕組みについて，FastAPI を使用した RESTful API のソースコード 3.1 に，Swagger のドキュメント 3.2 を例にまとめる．これは，GET メソッドで/users に HTTP リクエストを送信すると，ユーザーの一覧取得を行うことのできる RESTful AIP のソースコードとドキュメントである．Swagger 形式のドキュメントは，JSON または YAML で記述することができる．ドキュメント中の operationId はオプション値であり，エンドポイント毎にユニークな値を設定する必要がある．また，機能となる関数の上部 (FastAPI ではデコーレータである@app の上部) に，コメントアウトで@operationId と記述することで，ドキュメントとソースコードの対応付けを簡単に行えるようになる．

ソースコード 3.1　RESTful API

```
# @operationId get_users
@app.get('/users')
def get_users():
    response = []

    for user in get_users_from_db():
        response.append({
            'user_id': user.user_id,
            'name': user.name,
            'age': user.age
        })

    return response
```

ソースコード 3.2　Swagger

```
"paths": {
     "/users": {
         "get": {
            "summary": "Get a list of users.",
            "operationId": "get_users",
            "responses": {
                "200": {
                "description": "Successfully get the list of users."
                }
            }
        }
    }
```

```
13 }
```

### 3.2.3 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 ReDoct 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．

### 3.2.4 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．

### 3.2.5 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

本章では，提案ツールおよび提案ツールを動作するシミュレータについて述べる．

## 4.1 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

## 4.2 提案ツールのアーキテクチャ

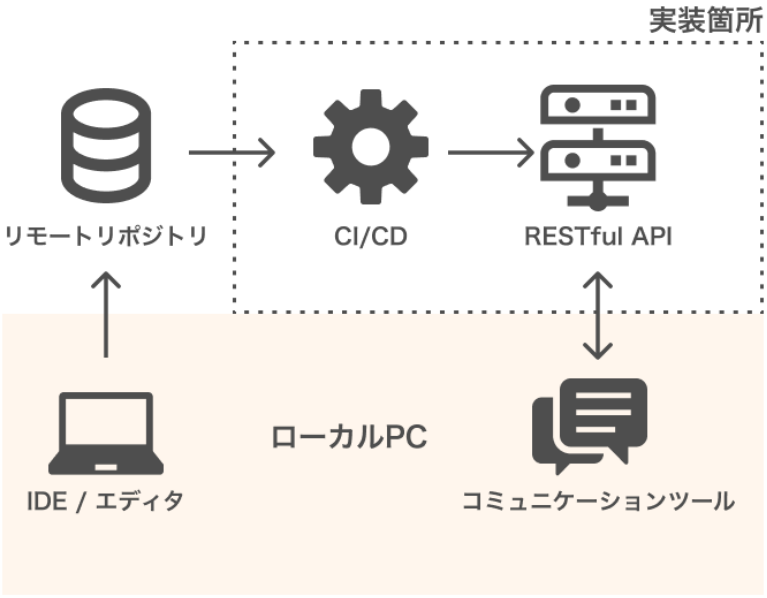



図 4.1 提案ツールのアーキテクチャ

提案ツールのアーキテクチャを図 4.1 に示す．提案ツールは，既存の CI/CD ツール上で動作することを想定しており，本研究では CI/CD ツールに GitHub Actions[6] を用いた．また，CI/CD ツール上で乖離の検知を行うが，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[15] を使用しており，Slack API や CI/CD ツール上で動作する提案ツールは，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．

Slack からに入力したコマンドを RESTful API に反映させるための手順を下記にまとめる．

1. Slack にコマンドを入力する
2. Incoming Webhook より，Slack から RESTful API に入力したコマンド内容が POST で送信される
3. RESTful API では，受け取ったコマンドを解析し，適切な処理を行う
4. 必要があれば，Slack にメッセージを送信する

また，乖離検知を行い，検知した内容を Slack へと送信するための手順を下記にまとめる．

1. GitHub へのプッシュを検知し，CI/CD ツールが起動する
2. CI/CD ツール上で乖離検知を行うために，必要なデータを RESTful API から取得する
3. 取得したデータをもとに，乖離検知を行う
4. 乖離検知をした場合，検知した内容を整形してメッセージとしてまとめ，RESTful API に POST リクエストを行う
5. RESTful API から Slack にメッセージを送信する

## 4.3 RESTful API の実装

RESTful API で利用可能な機能を下記にまとめる．これらの機能はすべて 1 つのファイル (main.py) に集約されており，行数は 162 行である．

GET：/data/{project_code}　乖離検知に必要なデータを取得
POST：/status/update/{project_code}　乖離検知に必要なデータを更新
GET：/git/log/{project_code}　整形した Git のコミット履歴を取得
POST：/git/log/{project_code}　整形した Git のコミット履歴を登録
POST：/message/{project_code}　Slack に通知を送信
POST：/milk　Slack に入力したコマンドを受け取り，適切な処理を実行

また，RESTful API に使用したライブラリとその選定理由を下記にまとめる．

FastAPI　RESTful API を簡易的に構築可能
Uvicorn　FastAPI を動作させるために必要
Slack Bolt　Slack API との通信を簡易的に行うことができる
requests　HTTP リクエストを直感的に扱うことが可能

## 4.4 提案ツールの使用方法

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．4.2 節にまとめたとおり，提案ツールは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に従って実行される．そのため，提案ツールは GitHub Actions を利用可能な状況ではローカル PC に専用の環境を構築せずに利用することができる．ただし，提案ツールを利用するにあたっていくつか準備が必要であるため，それについてまとめる．まず，CI/CD ツール上で動作させるために，プロジェクトのディレクトリに tools ディレクトリを作成し，そのディレクトリの中に Python で作成された提案ツールを配置する．CI/CD ツールの設定ファイルには，プッシュ時に提案ツールを実行するように記述するだけでよい．次に，Slack の API ページより，提

案ツールを登録する必要がある．最後に RESTful API の動作環境を用意する必要がある．例えば GCP(Google Cloud Platform)[12] や Heroku[13] などの PaaS(Platform as aa Service) を利用することで，RESTful API を動作する環境を用意することができる．筆者が作成した RESTful API を SaaS として展開し，数多くのプロジェクトで利用することもできるが，現状ではセキュリティ上の懸念から各プロジェクトで RESTful API を構築するのが望ましい．いずれも，各 Web サービスの登録手順に従って進めていくだけでよく，PaaS をよく理解していれば RESTful API の実行環境を簡単に構築することができる．

## 4.5 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 c3 param

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，現在設定しているバージョンを扱うファイルを確認することができるコマンドである．

### /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．

## 4.6 シミュレータの設計

シミュレーションには，CI/CD ツール上で動作する乖離検知機能の一部を利用した．シミュレーションの実行結果をグラフに描画するにあたり，Python のグラフ描画ライブラリである Matplotlib を使用した．以下に，シミュレーションを行う流れを示す．

1. シミュレーションを行いたいリポジトリから Git コマンドを使用してコミット履歴を指定件数取得する
2. 取得した文字列のコミット履歴を扱いやすいようにリストに整形する
3. コミットごとに C1〜C4 の機能を検証し，乖離リスクを検知したときに記録する

4. 記録したデータをもとに，Matplotlib でグラフの描画を行う

# 第 5 章

# シミュレーションと評価実験

本章では，提案ツールの有効性を検証するために行ったシミュレーションおよび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 シミュレーション

シミュレーションでは，GitHub 上にある OSS(オープンソースソフトウェア) のプロジェクトを利用する．対象とするプロジェクトは Python のパッケージ管理および仮想環境を構築可能な Pipenv[16] と Python の画像処理ライブラリである Pillow[17] 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，C2 C4 の機能について当時の乖離リスクの検証を行う．C1 の機能については，適切な検証を行うことができるプロジェクトが存在しなかったため除外する．また，今回のシミュレーションで使用するパラメータを下記にまとめる．

**C2：一定時間経過後のリマインダー** 7 日間ドキュメントが更新されなかったときに通知を行う

**C4：ソースコード変更量の検知の機能** ソースコードを 10000 行更新する間にドキュメントが更新されなかったときに通知を行う

## 5.3 実験計画

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．

### 5.3.1 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．

### 5.3.2 実験で使用したプロジェクト

実験では，RESTful API の開発およびドキュメントの更新を実験協力者に行ってもらう．このとき，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．下記に前後半のタスクの概要をまとめる．各タスクの詳細については付録 A に示す．

前半タスクの内容は以下のとおりである．

**タスク 1**　既存の機能に新たなバリデーションを追加する
**タスク 2**　新たな機能を作成する
**タスク 3**　既存の機能に新たなバリデーションを追加する
**タスク 4**　新たな機能を作成する
**タスク 5**　既存の機能に新たなバリデーションを追加する

後半タスクの内容は以下のとおりである．

**タスク 6**　タスク 5 で作成したバリデーションを修正する
**タスク 7**　新たな機能を作成する
**タスク 8**　既存のバリデーションを修正する
**タスク 9**　新たな機能を作成する
**タスク 10**　タスク 1 で作成したバリデーションを修正する

### 5.3.3 実験の流れ

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 (30 分)

2. 前半タスク (提案ツールを使用) (60 分)
3. 後半タスク (提案ツールを使用しない) (60 分)
4. 事後アンケート (10 分)

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 (30 分)
2. 前半タスク (提案ツールを使用しない) (60 分)
3. 後半タスク (提案ツールを使用) (60 分)
4. 事後アンケート (10 分)

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があったときは，いつでも質問してよいこととした．

# 第 6 章

# 結果と考察

本章では，シミュレーション，評価実験，アンケートの結果とその考察について述べる．

## 6.1 シミュレーション結果と考察

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 回呼び出され，Pipenv 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 451 回呼び出されることがわかった．Pipenv のシミュレーション結果を図 6.1，Pillow のシミュレーション結果を図 6.2 に示す．水色の折れ線がドキュメントのコミット数，オレンジ色の折れ線がソースコードのコミット数，紫色の点が C2 の呼び出し，緑色の点が C4 の呼び出しを表す．C3 については呼び出しがなかった．また，シミュレーション結果を図に表す時，各機能の呼び出しを日単位でまとめているため，実際の呼び出し回数より少なく表示されている．

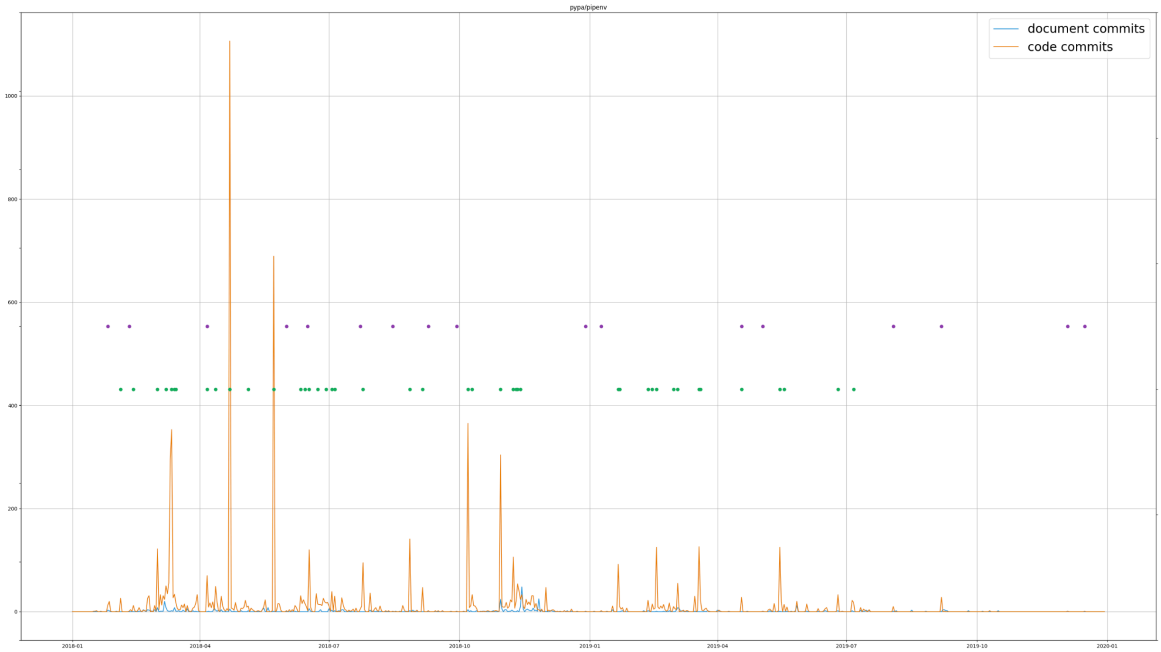



図 6.1　Pipenv のシミュレーション結果

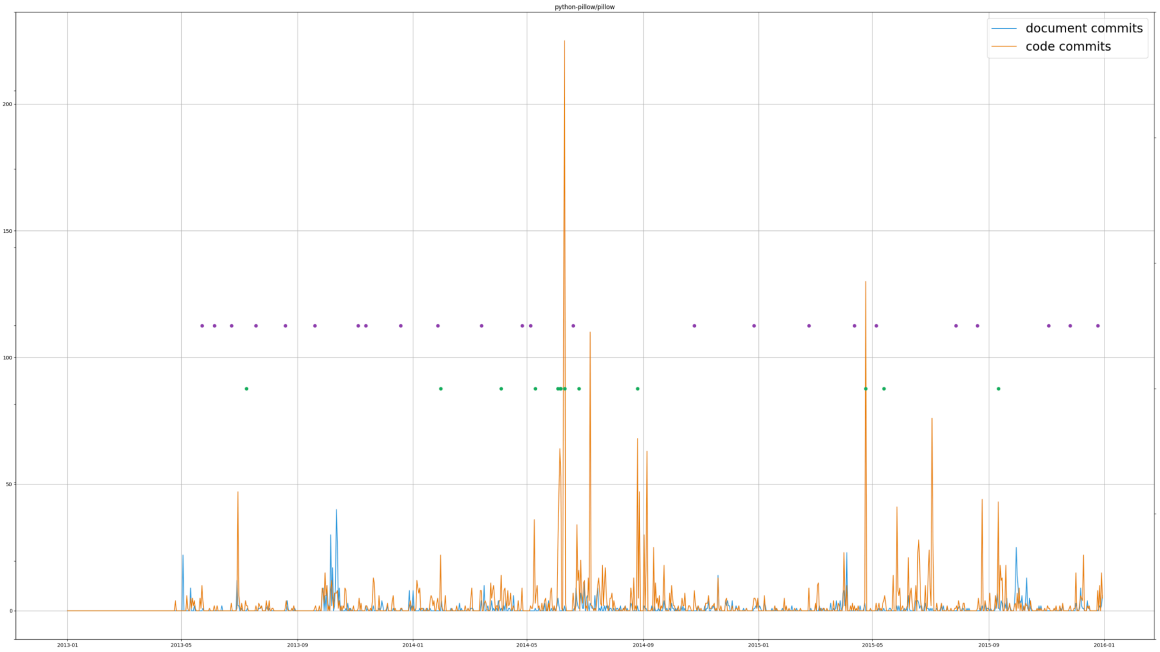



図 6.2　Pillow のシミュレーション結果

図 6.1 より，Pipenv では 2019 年以降に積極的な開発が行われていないため，ドキュメントとソースコードのコミット数が減少傾向にあることがわかる．そのため，ドキュメントとソースコードの乖離が起こることは少ないが，それでも通知を出してしまっている結果となっているため，ド

キュメントやソースコードで機能の追加や修正に全く関係のないコミットが通知を出さないように改良する必要があることが判明した．

以上のシミュレーション結果より，ドキュメントとソースコードの乖離リスクが検知されたときに通知を出すことで，乖離の抑制に役立つ可能性があることが判明した．一方で，精度の高い検知を行うためには，さらなる改善を加えないといけないことがわかった．また，様々なプロジェクトでシミュレーションを行うことのできるシミュレータの改善や，プロジェクト規模に合わせた各種パラメータの自動調整などをする必要があると考える．

## 6.2　実験結果と考察

5.3 節で示した実験計画に従って評価実験を行った．以下では，実験協力者 2 名について，それぞれのタスクの取り組みや，ドキュメントとソースコードの乖離検知をしたときの行動を詳細にまとめる．

### 6.2.1　前半に提案ツールを使用した実験協力者

前半に提案ツールを使用した実験協力者（以下，実験協力者 A と呼ぶ）の行動と乖離検知の流れを図 6.3 に示す．

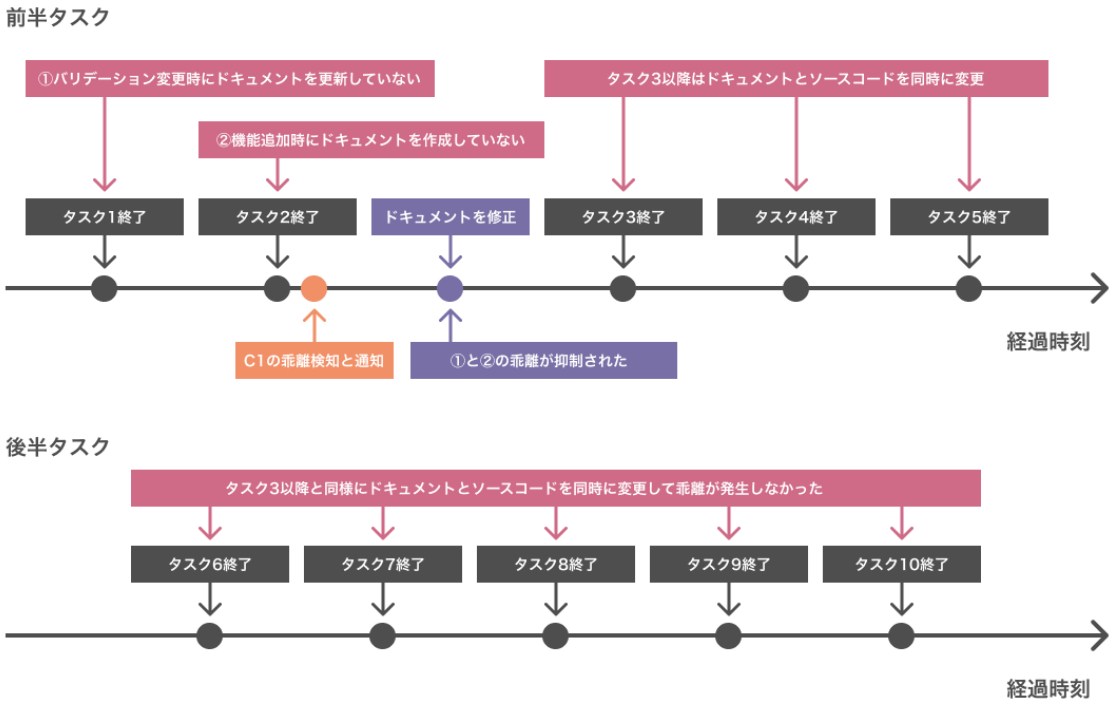



図 6.3　前半に提案ツールを使用した実験協力者の行動と乖離検知の流れ

実験協力者 A は，既存の機能に新たなバリデーションを追加するタスク 1 では，ドキュメントを更新せずにソースコードの追加のみを行っていた．このとき，提案ツールでは 3.2.2 で示したとおり，ドキュメントとソースコードの operationId のみを用いて乖離検知を行っているため，既存の機能に新たなバリデーションを追加し，新規エラーメッセージを返すバリデーションについては乖離検知を行うことができない．そのため，タスク 1 で変更されることのなかったドキュメントに対して，通知を送ることはなかった．次に，新たな機能を作成するタスク 2 を終えたときに，提案ツールが乖離検知を行ったため，実験協力者 A に「[C1]: ドキュメントに無い機能である get_users を作成しています」と通知を送った．図 6.4 は実際に実験協力者 A に通知を行った時のメッセージである．実験協力者 A は，通知を受け取った後，タスク 1 とタスク 2 で生じたドキュメントとソースコードの乖離を修正したため，乖離が抑制された．

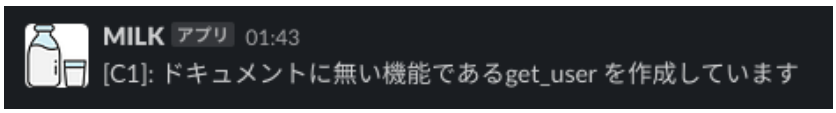



図 6.4　乖離検知をしたときに実験協力者 A に通知を送ったときのメッセージ

その後の後半タスクでは，前半に乖離検知の通知を受け取ったことを受けて，タスクに取り組むときは，ドキュメントとソースコードの両方を追加・修正しており，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．

### 6.2.2　後半に提案ツールを使用した実験協力者

後半に提案ツールを使用した実験協力者（以下，実験協力者 B と呼ぶ）の行動と乖離検知の流れを図 6.5 に示す．

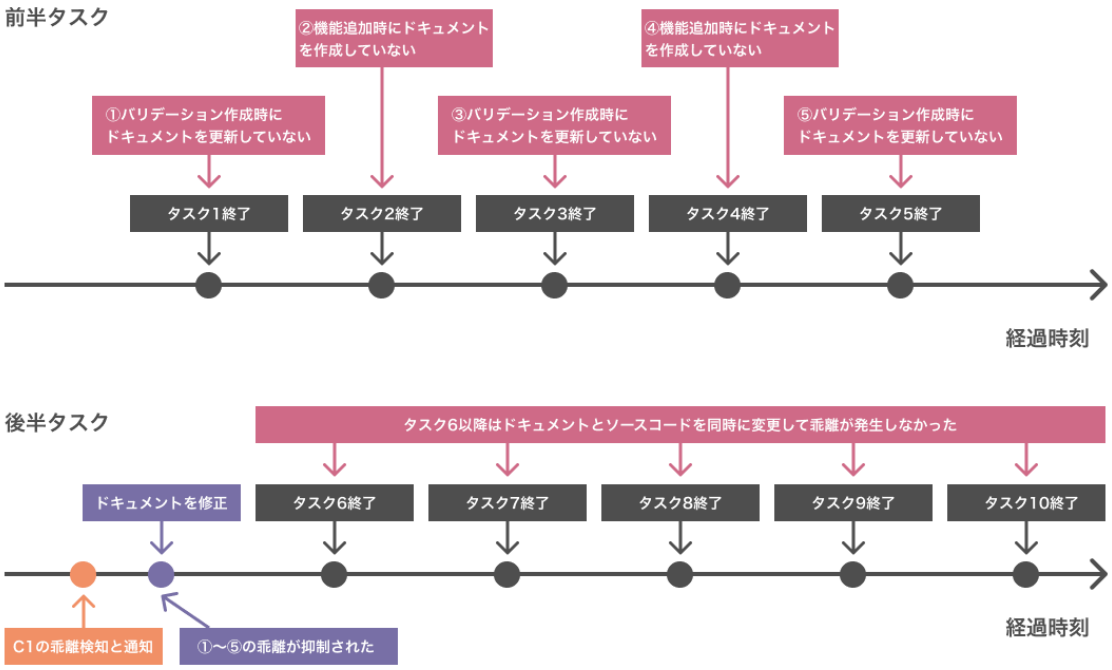



図 6.5　後半に提案ツールを使用した実験協力者の行動と乖離検知の流れ

実験協力者 B は，前半タスクをこなす間，一度もドキュメントに触れることなくソースコードの追加を行っていた．後半タスク開始時に，ドキュメントとソースコードの乖離検知が行われたため，実験協力者 B に「[C1]: ドキュメントに無い機能である get_user get_users_child を作成しています」と通知を送った．図 6.6 は実際に実験協力者 B に通知を行った時のメッセージである．


図 6.6　乖離検知をしたときに実験協力者 A に通知を送ったときのメッセージ

ここで，タスク 6 を開始する前に，実験協力者 B はドキュメントの更新を行ったため，タスク 1 〜タスク 5 で生じたドキュメントとソースコードの乖離が抑制された．その後，後半タスクが終了するまでドキュメントとソースコードの変更を同時に行った．

### 6.2.3　実験結果の考察

上記では実験協力者 2 名について，それぞれのタスクの取り組みや，ドキュメントとソースコードの乖離検知をしたときの行動を詳細にまとめた．2 名とも，乖離検知の通知を受け取った直後に，ドキュメントの修正を行っており，全タスク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．このことから，RESTful API の開発において，提案ツールはドキュメントとソースコードの乖離抑制に一定の効果が見込めることが判明した．一方で，実験協力者 A のタスク 1 において，バリデーションを修正したがドキュメントを更新しなかったことについて，乖離を検知できなかったため，提案ツールの性能を向上させる必要があると考えられる．

## 6.3 アンケート結果と考察

実施したアンケートを付録 B に示す．実験協力者 2 名に対し，Slack からの通知を受け取ったことで，ドキュメントを更新しないといけないと感じたかという質問に対し，1 名が「とても思った」，1 名が「やや思った」と回答した．回答結果を図 6.7 に示す．

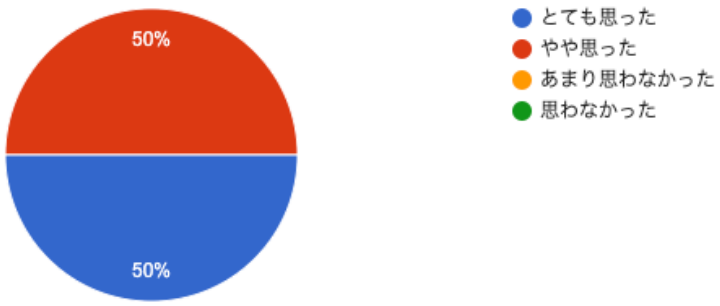



図 6.7 アンケート結果: Slack からの通知を受け取ったことで，ドキュメントを更新しないといけないと感じたか

また，ドキュメントの更新を忘れてしまったとき，通知は必要だと感じるかという質問に対し，2 名が必要であると回答した．回答結果を図 6.8 に示す．

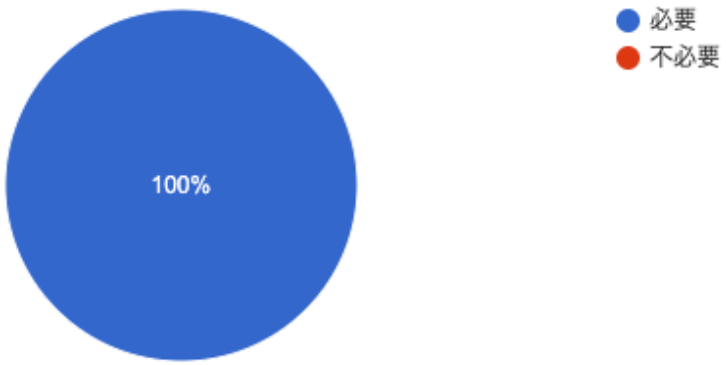



図 6.8　アンケート結果: ドキュメントの更新を忘れてしまったとき，通知は必要だと感じるか

Slack から受け取った通知内容で改善して欲しいところはあるかという質問に対して，以下のような回答を得た．

- 通知内容が簡素に感じた
- もっと詳細を教えて欲しい

以上のアンケート結果より，提案ツールはドキュメントとソースコードの乖離の抑制に効果を感じていることがわかった．一方で，通知内容を改善し，開発者に親しみやすいように改良を加えることで，より使いやすいツールになることが判明した．

# 第 7 章

# 結論

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，ビジネス要求を素早く実現することが求めらるソフトウェア開発現場において，ソースコードの追加・修正に追われてしまい，ドキュメントの追加・修正を忘れてしまうことにあると考えた．そこで，ドキュメントとソフトウェアの乖離の可能性のある箇所を特定および通知を行い，ドキュメントの追加・修正を開発者に促す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と考えた．また，開発スピードを落とさないためにも，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．シミュレーションの結果から，ドキュメントとソースコードの乖離リスクが検知されたときに通知を出すことで，乖離の抑制に役立つ可能性があることが判明した．一方で，精度の高い検知を行うためには，さらなる改善を加えないといけないことがわかった．

実験では，前半 5 個，後半 5 個の合計 10 個のタスクを 2 名の実験協力者にこなしてもらい，前後半のいずれかで提案ツールが動作するように設定した．実験の結果，前半に提案ツールを使用した実験協力者は，前半のタスクを 2 つこなした後に，提案ツールから乖離検知の通知を受け取った．この後，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．また，提案ツールを使用しない後半タスク終了時には，前半タスクで乖離したことを踏まえて作業を行ったため，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．後半に提案ツールを使用した実験協力者は，前半タスク終了時までにドキュメントを追加・修正することがなかった．しかし，後半タスク開始時に乖離検知の通知を受け取った後に，ドキュメントの修正を行い，後半タスク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．実験協力者 2 名の実験終了後にドキュメントとソースコードが乖離していなかったことから，提案ツールはドキュメントとソースコードの乖離を抑制するのに一定の効果があることが判明した．

今後の課題として，様々なプログラミング言語の対応や，汎用性の向上，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．

# 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

[2] 後藤英斗，大久保弘崇，粕谷英人，山本晋一郎：文脈に基づいたソースプログラムとドキュメント間の識別子対応付け手法，商法処理学会研究報告，2005

[3] 小池晃弘：ソースコードを記述しないアプリケーションフレームワークの提案，情報処理学会第 76 回全国大会，2014

[4] 海老澤雄太，丸山一貴，寺田実：Web Components 開発におけるドキュメント同時生成手法の提案，第 57 回プログラミング・シンポジウム，2016

[5] Slack, https://slack.com/intl/ja-jp/, (最終閲覧日：2021 年 1 月 18 日)

[6] GitHub Actions, https://github.co.jp/features/actions, (最終閲覧日：2021 年 1 月 18 日)

[7] Javadoc, https://docs.oracle.com/javase/jp/8/docs/technotes/tools/windows/javadoc.html, (最終閲覧日：2021 年 1 月 18 日)

[8] Spring Boot, https://spring.io/projects/spring-boot, (最終閲覧日：2021 年 1 月 18 日)

[9] Swagger, https://swagger.io/, (最終閲覧日：2021 年 1 月 18 日)

[10] ReDoc, https://github.com/Redocly/redoc, (最終閲覧日：2021 年 1 月 18 日)

[11] FastAPI, https://fastapi.tiangolo.com/, (最終閲覧日：2021 年 1 月 18 日)

[12] Google Cloud Platform, https://console.cloud.google.com/, (最終閲覧日：2021 年 1 月 18 日)

[13] Heroku, https://jp.heroku.com/, (最終閲覧日：2021 年 1 月 18 日)

[14] Matplotlib, https://matplotlib.org/, (最終閲覧日：2021 年 1 月 18 日)

[15] ngrok, https://ngrok.com/, (最終閲覧日：2021 年 1 月 18 日)

[16] Pipenv, https://github.com/pypa/pipenv, (最終閲覧日：2021 年 1 月 18 日)

[17] Pillow, https://github.com/python-pillow/Pillow, (最終閲覧日：2021 年 1 月 18 日)

# 付録 A

# 実験で使用したタスク

## A.1　前半タスク

### 1：ユーザー登録時に年齢をチェックする

- 既に「ユーザー登録時に名前が15文字を超過した場合にエラー」とする機能があります。
- **1～100歳以外の場合はエラーとしてエラーメッセージを返す機能を追加しましょう。**
- エラー時のレスポンスは
  **raise HTTPException(status_code=400, detail="年齢は1～100歳")**とします。
- Userクラスにある “check_name” メソッドを参考に実装することができます。

### 2：特定のユーザーを取得する機能を作成

- ユーザーIDを元にユーザーを取得する機能を作成しましょう。
- **URL：/user/{user_id}**
  **リクエストメソッド：GET**
- 例えば、/user/100 であればユーザーIDが100のユーザーを返します。
  /user/200であれば、ユーザーIDが200のユーザーを返します。
- ユーザーIDがJSONファイルに存在しない場合のレスポンスは
  **raise HTTPException(status_code=400, detail="ユーザーIDが見つかりません")**
  とします。

### 3：ユーザーIDを100の倍数とする

- 既存の「ユーザーを登録するAPI」において、user_idのチェックを行う機能を作成します。
- **ユーザーIDが100の倍数でなかったときにエラーとする機能を作成しましょう。**
- 例えば、user_idが201であった場合は、100の倍数ではないため、レスポンスは
  **raise HTTPException(status_code=400, detail="ユーザーIDが不適切です")** とします。
- 適切なユーザーIDの場合は、2で作成した機能どおり、ユーザー情報を返却します。

### 4：子供のユーザー一覧を取得する機能を作成

- 子供は1～12歳の年齢とします。
- ユーザー一覧取得の機能とは別に、**新しく1~12歳のユーザー一覧を取得する機能を作成しましょう。**
- **URL：/users/child**
  **リクエストメソッド：GET**
- レスポンスはユーザー一覧取得と同じ内容にします。

### 5：ユーザーIDの上限を100,000とする

- 既存の「ユーザー登録をするAPI」において、user_idのチェックを行う機能をさらに作成します。
- **ユーザー登録時に、ユーザーIDが100,000を超過したときにエラーとする機能を作成しましょう。**
- 例えば、user_idが100,001であった場合は、100,000を超過しているため、レスポンスは**raise HTTPException(status_code=400, detail="ユーザーIDが大きすぎます")** とします。
- 適切なユーザーIDの場合は、2で作成した機能どおり、ユーザー情報を返却します。
- **ただし、100の倍数でないときは3と同様のレスポンスを返しましょう。**

# A.2　後半タスク

## 6：ユーザーIDの上限を100,000→1,000,000

- **ユーザー登録時、ユーザーIDの上限を100,000から1,000,000に変更します。**
- **「ユーザー登録」が対象です。**
- エラー内容も適切なものに変更します。

## 7：子供のユーザー登録をする機能を作成

- 既存のユーザー登録をする機能(POST: /user) とは別に、**新しく子供のユーザー登録をする機能を作成しましょう。ただし、ユーザーIDは100の倍数とします。ユーザーIDが100の倍数でないときは、同様のエラー文を返しましょう。**
- URL：/user/child
  リクエストメソッド：POST
- リクエストで受け取った年齢が1～12歳でない場合はレスポンスは、
  **raise HTTPException(status_code=400, detail="子供ではありません")** とします。
- レスポンスはユーザー登録をする機能と同じにします

## 8：名前の制限を15文字→30文字にする

- **ユーザー登録時、名前の制限を15文字から30文字に変更しましょう。**
- **「ユーザー登録」と「子供のユーザー登録」が対象です。**
- エラー内容も適切なものに変更します。

## 9：最後に追加したユーザーを取得する

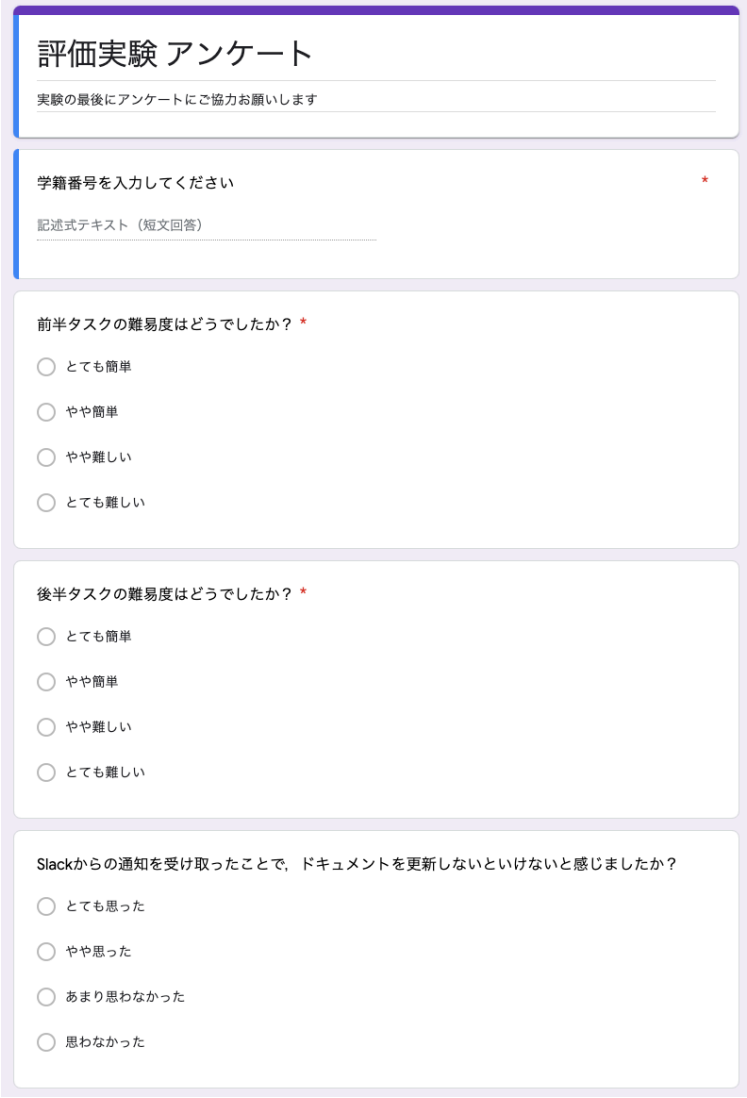
- **最後にユーザー登録したユーザーを取得する機能を作成しましょう。**
- 登録されているユーザー件数が0件の場合は考慮する必要はありません。
- URL：/user/one/latest
  リクエストメソッド：GET
- レスポンスはユーザーの情報(user_id, name, age)をJSONで返します。
  例：{"user_id": 100, "name": "Alice", "age": 18}

## 10：年齢の制限を120歳までに緩和する

- **ユーザー登録時、年齢の制限を「100歳まで」から「120歳まで」に変更しましょう。**
- **「ユーザー登録」が対象です。**
- エラー内容も適切なものに変更します。

# 付録 B

# 実験で用いたアンケート



評価実験 アンケート

実験の最後にアンケートにご協力お願いします

学籍番号を入力してください *

記述式テキスト（短文回答）

前半タスクの難易度はどうでしたか？ *

- とても簡単
- やや簡単
- やや難しい
- とても難しい

後半タスクの難易度はどうでしたか？ *

- とても簡単
- やや簡単
- やや難しい
- とても難しい

Slackからの通知を受け取ったことで，ドキュメントを更新しないといけないと感じましたか？

- とても思った
- やや思った
- あまり思わなかった
- 思わなかった

Slackからの通知を受け取ったことで，ドキュメントを更新しないといけないと感じましたか？

- ○ とても思った
- ○ やや思った
- ○ あまり思わなかった
- ○ 思わなかった

ドキュメントの更新を忘れてしまったとき，通知は必要だと思いますか？

- ○ 必要
- ○ 不必要

タスク2，タスク4，タスク7，タスク9などで新たな機能を追加するとき、ドキュメントとソースコードどちらから書き始めましたか？

- ○ ドキュメント
- ○ ソースコード
- ○ 特に意識はしていない

Slackから受け取った通知内容で改善してしてほしい点があれば教えて下さ

記述式テキスト（短文回答）

全体を通して気になる点や質問などあればお願いします

記述式テキスト（短文回答）