一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
KBSE2023-43,SC2023-26(2023-11)

実験指向プログラミングプラットフォームへの ゲーミフィケーションの導入

長谷　碧[†]　陳　思楠[†]　佐伯　幸郎[††]　中村　匡秀[†,†††]

† 神戸大学　〒657–8501 神戸市灘区六甲台町 1–1
††† 理化学研究所・革新知能統合研究センター〒103-0027 東京都中央区日本橋 1-4-1
†† 高知工科大学 〒782–8502 高知県香美市土佐山田町宮ノ口 185
E-mail: †{ing,chensinan}@ws.cs.kobe-u.ac.jp, ††saiki.sachio@kochi-tech.ac.jp, †††masa-n@cs.kobe-u.ac.jp

あらまし　昨今，データサイエンス系のプログラミングをする際に，JupyterLab や Google Colaboratory といったプラットフォームがしばしば用いられる．しかし，これらのプラットフォームでは，プログラミングが実験的かつ試行錯誤的に行われ，コードの説明が付与されずに放置される問題がある．そこで，本研究では，ゲーミフィケーションの導入によって，開発者のコードの説明の記述を促す仕組みを提案する．ケーススタディとして，JupyterLab 内に提案システムの実装を行い，その利点と限界について考察する．
キーワード　データサイエンス，プログラミング，JupyterLab，ゲーミフィケーション，モチベーション

Introducing Gamification into an Experiment-based Programming Platform

Aoi NAGATANI[†], Sinan CHEN[†], Sachio SAIKI[††], and Masahide NAKAMURA[†,†††]

† Kobe University　Rokkodai-cho 1–1，Nada-ku，Kobe，Hyogo，657–8501 Japan
††† Riken AIP, 1-4-1 Nihon-bashi, Chuo-ku, Tokyo 103-0027
†† Kochi University of Technology, 185 Tosayamadacho Miyanokuchi, Kami, Kochi, 782–8502 Japan
E-mail: †{ing,chensinan}@ws.cs.kobe-u.ac.jp, ††saiki.sachio@kochi-tech.ac.jp, †††masa-n@cs.kobe-u.ac.jp

Abstract　Recently, platforms such as JupyterLab and Google Colaboratory are often used for data science programming. However, these platforms have a problem that programming is done in an experimental and trial-and-error manner, and the code is left unexplained. In this study, we propose a system that encourages developers to write code descriptions by introducing gamification. As a case study, we implement the proposed system in JupyterLab and discuss its advantages and limitations.
Key words　Data Science, Programming, JupyterLab, Gamification, Motivation

1. はじめに

近年，デジタル技術の発展による膨大なデータの収集・蓄積や，計算機の能力の向上により，データマイニングや機械学習といったデータサイエンスの研究が盛んに行われている．データサイエンス系のプログラミングを行う際には，Jupyter Lab [1] や Google Colaboratory [2] といったプラットフォームがしばしば用いられる．これらのプラットフォームではプログラミングが実験的かつ試行錯誤的に行われる．本研究では，これらのプラットフォームを **“実験指向プログラミングプラットフォーム”** と呼び，記述されるソースコードを “実験指向なコード” と呼ぶ．実験志向なコードは以下のような性質を持つ．

C1:　短期間のプログラミング実行過程を他人に共有する
C2:　開発者が永続性を考慮しない

上記の性質から，**実験指向なコードは説明が記述されにくい**と考えられる．実際に，著者の研究室で実験指向プログラミングプラットフォームの利用状況を調査したところ，コードの説明が足りていないことがあることが分かった．よって，この問題を解決するための仕組みが必要である．

本研究では，上記の課題を解決するために，**開発者に実験指向プログラミングプラットフォームにおけるコードの説明の記述を促すこと**を目的とする．この目的を達成するためのキーアイデアとして，**ゲーミフィケーションを用いたシステムを提案する**．具体的には，以下のアプローチに基づいて機能実装を行う．

A1:　コードセルの説明率を可視化する機能の実現
A2:　コードの説明を記述するモチベーションを高められる仕組みの実現

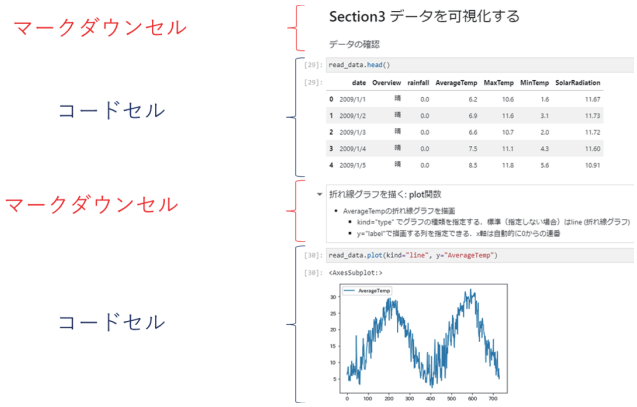

図 1: JupyterLab のインターフェース

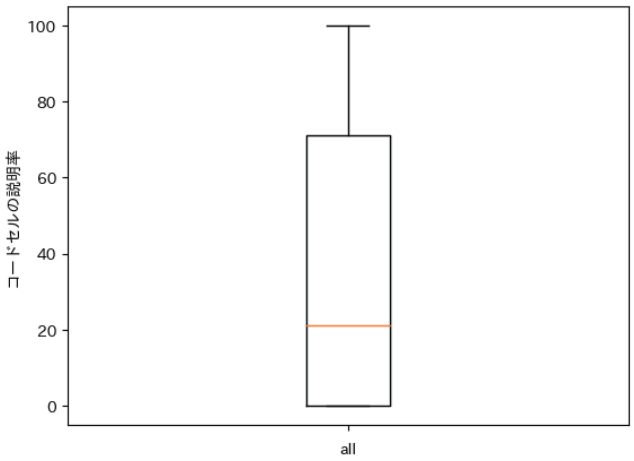

図 2: 全 466 ファイルのコードセルの説明率の箱ひげ図

まず，A1 では説明が付与されているコードセルの割合をグラフや表で可視化することによって，開発者自身にどの程度コードの説明が記述されているかを知ってもらうことを目指す．

次に，A2 では各開発者の説明が付与されているコードセルの割合を参照してゲーム要素を導入することによって，コードの説明を記述するモチベーションを高められる仕組みを実現する．

ケーススタディとして，これらのアプローチを基に JupyterLab 内に提案システムの実装・考察を行う．具体的には，JupyterLab の拡張機能を実装することで，ゲーミフィケーションを導入し，その利点と限界について考察する．

2. 準　　備

2.1 実験指向プログラミングプラットフォーム

昨今，データマイニングや機械学習などのデータサイエンスの研究が盛んに行われている．データサイエンス系のプログラミングを行う際には，Jupyter Lab [1] や Google Colaboratory [2] といったプラットフォームがしばしば用いられる．これらのプラットフォームではプログラミングが実験的かつ試行錯誤的に行われる．そこで本研究では，これらのプラットフォームを **“実験指向プログラミングプラットフォーム”** と呼び，記述されるソースコードを “実験指向なコード” と呼ぶ．

ここで，実験指向プログラミングプラットフォームのインターフェースの説明を行う．例として，図 1 に JupyterLab のインターフェースを示す． 実験指向プログラミングプラットフォームは，2 種類のセルを持つ．1 つ目はコードセルであり，Python，R 等でソースコードを記述する．2 つ目はマークダウンセルであり，マークダウン形式でコードの説明を記述する．これらのセルを駆使して実験ノートのようにプログラミングを行う．

実験志向なコードは以下のような性質を持つ．

C1: 短期間のプログラミング実行過程を他人に共有する

C2: 開発者が永続性を考慮しない

まず C1 について説明する．実験指向プログラミングプラットフォームでは，実験ノートのようなインターフェースでソースコードやその実行の過程を記述することができる．さらに，コード内に記述するコメントとは別に，各コードのひとまとまり毎にマークダウンで説明を付加することができる．よって，実験指向プログラミングプラットフォームで記述される内容は他人に共有することに適している．

次に C2 について説明する．実験指向プログラミングプラットフォームでは実験的かつ試行錯誤的にプログラミングが行われ，頻繁にコードが変更・改良される．よって，開発者はコードの永続性を考慮せず，ソリッドな設計を行わないと考えられる．

2.2 課　　題

章 2.1 で述べた実験指向プログラミングプラットフォームの性質により，**実験志向なコードは説明が記述されにくい**と考えられる．頻繁な変更のたびにコードの説明を更新することは開発者に多大な労力を要するためである．コードの説明が少ないことによって以下の問題が生じると考えられる．

- 開発者以外の人間がコードを理解することが困難になり，コードの再利用性が低下する
- 開発者当人も，コードの処理内容を把握しにくくなるため，コードの保守性が低下する

実際に，著者の研究室で実験指向プログラミングプラットフォームの利用状況を調査した例を報告する．そのために，コードの説明をどれだけ記述しているかの指標として，“コードセルの説明率” を定義する．コードセルの説明率とは，全てのコードセルに対して，直前のマークダウンセルによって説明がなされているコードセルの割合である．例えば，図 1 で示した状況であれば，コードセルが 2 つに対して，直前のマークダウンセルによって説明がなされているコードセルが 2 つであるため，コードセルの説明率は 100% である．全てのコードセルに対して説明が記述されることが望ましいと考えられるため，目指すべきコードセルの説明率は 100% である．

利用状況の調査は，著者の研究室の Jupyter Lab で記述された 466 ファイルを対象に行う．調査対象のファイルを記述したとされる研究室メンバーの人数は 32 人である．

まず，全 466 ファイルのコードセルの説明率の分析結果を報告する．全ファイルのコードセル数の平均値 22.67 個に対して，コードセルの説明率の平均値は 34.80% であった．また，図 2 に全 466 ファイルのコードセルの説明率の箱ひげ図を示す．図 2 において，縦軸はコードセルの説明率を表す．

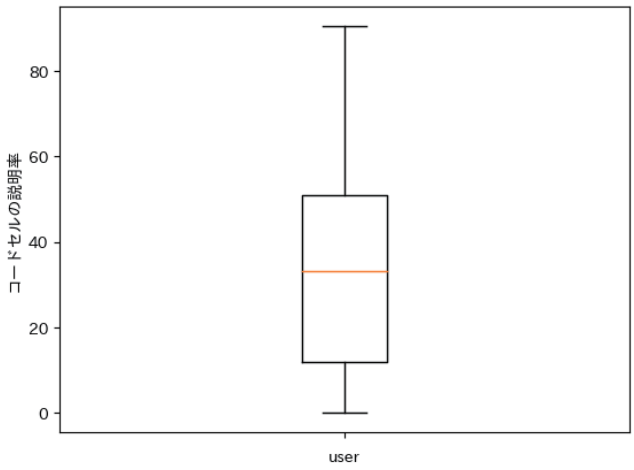

図 3: 研究室メンバー毎のコードセルの説明率の平均値の箱ひげ図

次に，ファイルを記述した研究室メンバー毎にまとめて，コードセルの説明率を分析した結果を報告する．図 3 に研究室メンバー毎のコードセルの説明率の平均値の箱ひげ図を示す．図 3 において，縦軸はコードセルの説明率を表す．また，各コードセルの説明率の平均値は，ファイルを記述した研究室のメンバー毎にファイルをまとめ，平均値を算出している．

以上のように，コードセルの説明率が理想とする 100% に比べて大きく低く，コードに対する説明が足りていないことがある．よって，この問題を解決するための仕組みが必要である．

2.3 ゲーミフィケーション

章 2.2 で述べた課題を解決するために，**“ゲーミフィケーション”** [3] [4] を用いることを考える．ゲーミフィケーションとは，ゲーム的な要素を導入することによってサービスをゲーム化し，ユーザの好奇心を刺激したり，行動を活性化させたり，ユーザに良い利益をもたらす等の効果を期待する手法である [5]．以降，サービスをゲーム化するための機能のことを “ゲーミフィケーション機能” と呼ぶ．

関連研究として，プログラミング学習の継続的なモチベーションを維持するために，JupyterLab にゲーミフィケーションを導入した例 [6] [7] を説明する．この研究では，プログラミングの初学者にとってのコーディングのハードルの高さや，プログラミングの講座の落第率の高さを問題として挙げている．そこで，この問題を解決し，初学者が継続的にプログラミング学習を続けられるために，JupyterLab にゲーミフィケーションを導入している．主なゲーミフィケーション機能としては，JupyterLab 内の行動によって得られる経験値機能や，連続ログイン機能，あらゆる達成項目に対応したバッジ機能等が実装されている．図 4 に，この研究で実装されたゲーミフィケーション機能の例を示す．

以上のように，関連した研究 [8] [9] は盛んに行われており，ゲーミフィケーションはユーザーの行動を促す効果があることが分かっている．

図 4: 関連研究で実装されたゲーミフィケーション機能の例 [6]

3. 提 案 手 法

3.1 目的とキーアイデア

章 2.2 の課題を解決するために本研究では，**開発者に実験指向プログラミングプラットフォームにおけるコードの説明の記述を促すこと**を目的とする．この目的を達成するためのキーアイデアとして，章 2.3 で述べた**ゲーミフィケーションを用いたシステムを提案する．**

3.2 アプローチ

章 3.1 で述べた本研究の目的を達成するためのシステムの機能実装を以下のアプローチを基に行う．

A1: コードセルの説明率を可視化する機能の実現

A2: コードの説明を記述するモチベーションを高められる仕組みの実現

これらのアプローチに基づいてシステムの機能実装を行うことで，開発者に実験指向プログラミングプラットフォームにおけるコードの説明の記述を促す仕組みを実現する．

3.3 A1：コードの説明率を可視化する機能の実現

このアプローチでは，コードセルの説明率をグラフや表で可視化することによって，開発者自身にどの程度コードの説明が記述されているかを知ってもらうことを目指す．データを可視化することもゲーミフィケーションの一種である [10]．開発者にコードセルの説明率を自覚してもらうことで，コードの説明が足りていない現状を認識し，その必要性について考えるきっかけを与えられると考えている．

具体的には，以下のような機能を実装する．

- 開発者ごとの全ファイルにおけるコードセルの説明率を表示する機能
- 開発者ごとの全ファイルにおけるコードセルと説明が記述されているコードセルの割合をグラフで可視化する機能
- 直近で作成されたファイルについて，コードセルと説明が記述されているコードセルの割合をグラフで可視化する機能

直近で作成されたファイルについて可視化する機能を取り入れたのは，開発者にとって古いファイルよりも新しいファイルの方がコードの説明を改めて記述しやすいと考えたためである．

これらの機能によって，開発者に自身のコードの説明率を知ってもらうことを目指す．

3.4 A2：コードの説明を記述するモチベーションを高められる仕組みの実現

このアプローチでは，各開発者のコードの説明率を参照してゲーム要素を導入することによって，コードの説明を記述するモチベーションを高められる仕組みを実現する．

具体的には，以下のような機能を実装する．

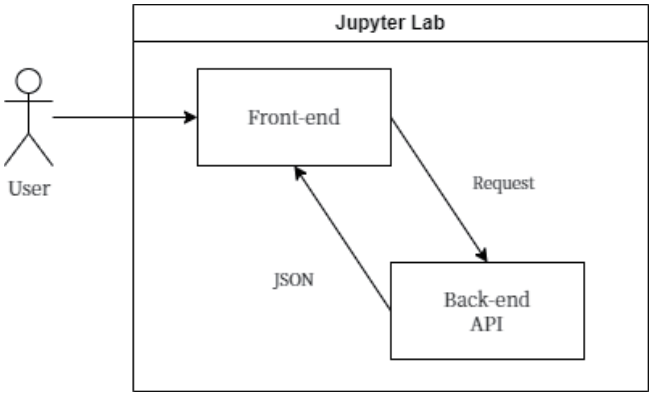

図 5: 提案システムのアーキテクチャ

- コードセルの説明率によって変動する画像を表示する機能
- コードセルの説明率の現状に対するアドバイスを表示する機能

これらの機能によって，開発者のコードの説明を記述するモチベーションを高めることを目指す．

4. ケーススタディ：JupyterLab へのゲーミフィケーションの導入

本研究では，章 3.2 で述べたアプローチを基に，ケーススタディとして JupyterLab 内に提案システムの実装を行う．

4.1 アーキテクチャ

提案システムの実装は JupyterLab の拡張機能を開発することによって行う．拡張機能を用いる利点として，開発者が実験指向なコードを記述しているインターフェースと同じ場所に機能を実装できることが挙げられる．

JupyterLab は拡張機能を開発するためのテンプレート [11] を提供しているため，これを利用する．標準の JupyterLab の拡張機能はバックエンドに Python のフレームワーク “Tornado”，フロントエンドに TypeScript を用いて実装される．本研究では，それらを拡張して以下のような言語・技術で開発を行う．

- バックエンド
  - Python
  - Tornado
- フロントエンド
  - TypeScript
  - React

React を用いる理由としては，ユーザービリティが高いインターフェースを比較的容易に実装できるためである．

図 5 に提案システムのアーキテクチャを示す．データの流れを以下に説明する．

Step1: ユーザーがデータを必要とする操作を行うと，フロントエンドはバックエンドに対してデータを要求する．

Step2: バックエンドは，JupyterLab 内にあるファイルを各ユーザーごとにデータを計算・整形し，フロントエンドに返す．

Step3: フロントエンドは，バックエンドから受け取ったデータをもとに，データをグラフ等で画面に表示する．


```
[
  {
    "userName": "○○",
    "numAllDescribedCodeCells": 688,
    "numAllCodeCells": 742,
    "files": [
      {
        "fileName": "△△.ipynb",
        "numDescribedCodeCells": 123,
        "numCodeCells": 123
      },
      {
        "fileName": "□□.ipynb",
        "numDescribedCodeCells": 26,
        "numCodeCells": 27
      },
      {
        "fileName": "× ×.ipynb",
        "numDescribedCodeCells": 56,
        "numCodeCells": 57
      }
    ]
  },
  ...
]
```


図 6: バックエンドからフロントエンドに送信する JSON データの一例

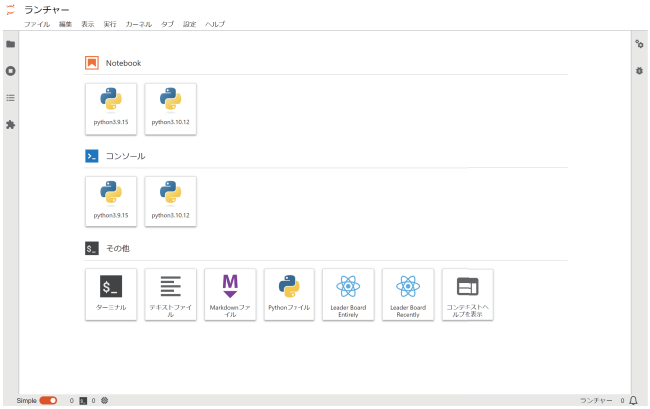

図 7: JupyterLab のトップページ

図 6 にバックエンドからフロントエンドに送信する JSON データの一例を示す．

4.2 機　　能

章 3.3 と章 3.4 で述べたアプローチを基に機能実装を行う．まず，全期間のファイルと任意の期間のファイルを対象とする 2 つのリーダーボードを実装し，そのコンテンツとして各種機能を実装する．2 つのリーダーボードは対象とするデータの期間の違い以外は同じ機能を持つ．ユーザーは図 7 に示す JupyterLab のトップページの “Leader Board Entirely” もしくは，“Leader Board Recentry” をクリックすることで，2 つのリーダーボードのいずれかを選択できる．

また，図 8 に全期間のファイルを対象とするリーダーボード，図 9 に任意の期間のファイルを対象とするリーダーボードを示す．

ここから各種ゲーミフィケーション機能について説明を行うに先立って，図 10 に図 8 で示したリーダーボードのあるユー

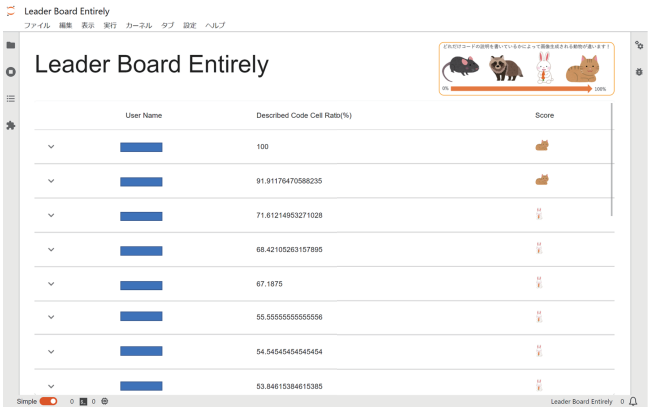

図 8: 全期間のファイルを対象とするリーダーボード

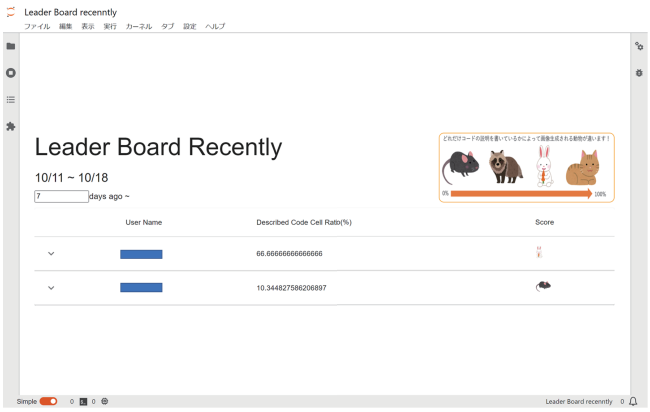

図 9: 任意の期間のファイルを対象とするリーダーボード

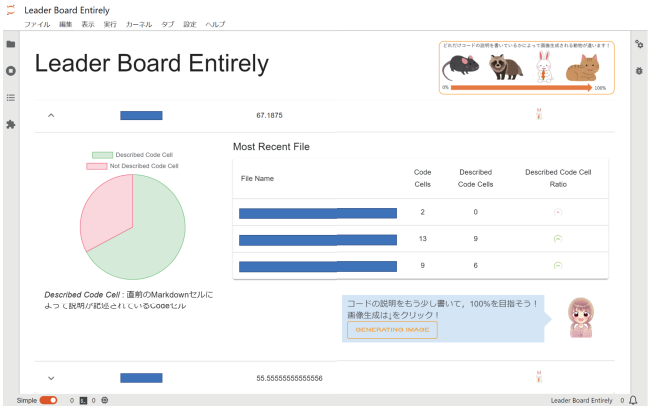

図 10: リーダーボードのあるユーザーについて展開した画面

ザーについて展開した画面を示す．ユーザーは，リーダーボードの各ユーザーネームに対応した行をクリックすることで，図 10 のような画面を展開することができる．弊研究室では，ユーザーごとに作業ディレクトリが用意されているため，ディレクトリ名をユーザーネームとしている．

4.2.1 A1：コードの説明率を可視化する機能の実装

ここでは，章 3.3 で述べた 3 つの機能実装を行う．

まず，開発者ごとの各ファイルにおけるコードセルの説明率を表示する機能について説明する．この機能は，図 8 と図 9 のように，リーダーボードの各ユーザーに対応する “Described Code Cell Ratio(%)” の列に表示する．コードセルの説明率の高い順に並べることで，実質的にランキング形式になるような仕様にしている．

次に，開発者ごとの全ファイルにおけるコードセルと説明が記述されているコードセルの割合をグラフで可視化する機能について説明する．図 10 の画面の左側に表示されている円グラフがこの機能に該当する．この機能では，説明が記述されているコードを緑色，記述されていないコードを警告色の赤色で表示する．記述されていないコードの割合を警告色で表示することで，コードの説明が足りていない場合は改善が必要であることを開発者に直観的に認知してもらうことを狙いとしている．

最後に，直近で作成されたファイルについて，コードセルと説明が記述されているコードセルの割合をグラフで可視化する機能について説明する．図 10 の画面の右側に表示されている表がこの機能に該当する．この機能では，直近で作成された 3 ファイルについて，コードセルの説明率をゲージで表示する．ゲージの色は，コードセルの説明率が 100% に近づくほど緑色，遠ざかるほど赤色で表示する．この機能によって，開発者は直近で作成されたファイルについて，コードの説明が足りているかを直感的に認識できると考えている．さらに，直近で作成したファイルであるため，開発者が直ちにコードセルの説明率を改善しやすいと考えられる．

これらの機能によって，開発者自身にどの程度コードの説明が記述されているかを知ってもらうことを目指す．

4.2.2 A2：コードの説明を記述するモチベーションを高められる仕組みの実装

ここでは，章 3.4 で述べた 2 つの機能実装を行う．

まず，コードセルの説明率によって変動する画像を表示する機能について説明する．図 10 の画面の右下に表示されている画像がこの機能に該当する．この機能では，コードセルの説明率によって変動する画像を表示する．コードセルの説明率が 100% に近づくほど，画像の表情が明るくなり，100% から遠ざかるほど，画像の表情が暗くなるように設定する．図 10 の画像は，コードセルの説明率が 67% と比較的に高いため，画像の表情が明るくなっている．図 11 にコードセルの説明率が比較的に低い場合の画面を示す．さらに，それぞれ 4 種類の表情パターンを持つ画像を 3 種類用意し，JupyterLab を読み込むたびに，ランダムで画像が表示される仕様にしている．また，Stable Diffusion を用いて，画像をランダムで生成する機能を実装する．ユーザーは図 10 の画面の右下の吹き出し内のボタンをクリックすることで，画像を生成することができる．ここで生成する画像のモチーフは，コードセルの説明率によって変動し，モチーフは開発者の好みによって設定できることを想定している．これら機能によって，開発者は画像の状況を良くするために，コードの説明を記述しようというモチベーションが高まると考えている．

次に，コードセルの説明率の現状に対するアドバイスを表示する機能について説明する．図 10 の画面の右下に表示されている吹き出しの 1 行目がこの機能に該当する．この機能では，コードセルの説明率が高い開発者に対しては，現状を維持する旨のアドバイスを表示し，コードセルの説明率が低い開発者に

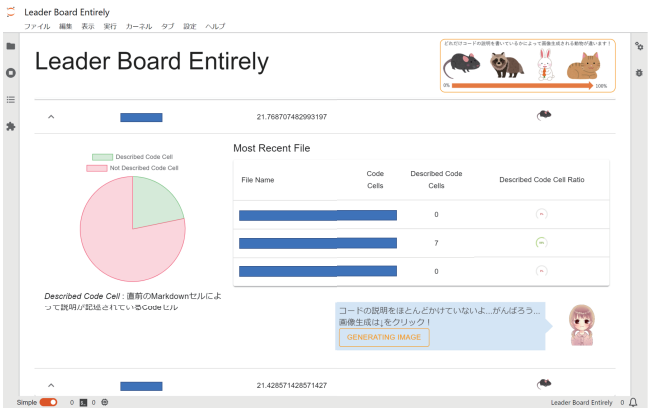

図 11: コードセルの説明率が比較的に低い場合の画面

対しては，現状を改善する旨のアドバイスを表示する．アドバイスはコードセルの説明率によって 4 パターン用意している．この機能によって，開発者に現状からどのように改善すればよいかを提示することで，コードの説明の現状を改善する意識を持ってもらうことを目指す．

これらの機能によって，開発者のコードの説明を記述するモチベーションを高めることを目指す．

4.3 考　　察

ケーススタディで実装した提案システムの利点と限界について考察する．

まず提案システムの利点については，ユーザーにとって目に触れやすい場所にゲーミフィケーション機能が存在することである．ゲーミフィケーションの効果を最大限に発揮するには，以下の点が重要であると考えている．

- ゲーミフィケーション機能がユーザーの目に触れやすいところにあること
- ゲーミフィケーション機能を有することをユーザーが知っていること

そこで，本研究では，外部サービスではなく，JupyterLab のシステム内に機能を実装できる拡張機能を用いてゲーミフィケーション要素を導入した．また，実際にこのシステムを開発者に利用してもらう際には，ゲーミフィケーション機能について深く理解してもらうために，詳細な説明が必要であると考えられる．

次に，提案システムの限界については，開発者のモチベーションを継続的に高めることの困難さが挙げられる．本研究のゲーミフィケーション機能によって一時的にモチベーションを高めることは可能であるが，継続的にモチベーションを高めることは困難であると考えられる．モチベーションを維持するためには，コンテンツの更新や，ゲーミフィケーション機能のアップデートといった，継続的な運営・メンテナンスが必要であると考えられる．

5. ま と め

本研究では，Jupyter Lab や Google Colaboratory といった “実験指向プログラミングプラットフォーム” で記述されるコードの説明が足りていないことを問題として挙げた．そこで，この問題を解決するために，開発者に実験指向プログラミングプラットフォームにおけるコードの説明の記述を促すことを目的とし，ゲーミフィケーションを用いたシステムを提案した．また，ケーススタディとして，JupyterLab 内に提案システムの実装を行い，その利点と限界について考察した．

今後の展望としては，今回実装した JupyterLab の拡張機能によさらに効果的なゲーミフィケーション機能を追加実装することが挙げられる．具体的には，開発中のユーザーの目に触れやすいコードの編集画面でのゲーミフィケーション機能の実装である．また，実装した JupyterLab の拡張機能を導入した JupyterLab 環境を実際に開発者に利用してもらい，ゲーミフィケーション機能の効果を検証することが挙げられる．

謝辞　本研究の一部は JSPS 科研費 JP19H01138, JP20H05706, JP20H04014, JP20K11059, JP22H03699, JP19K02973, 若手研究 23K17006 の助成を受けて行われている．

文　　献

[1] “Project jupyter,” https://jupyter.org/. visited on 2023-10-21.
[2] “Google colaboratory,” https://colab.research.google.com/. visited on 2023-10-21.
[3] I. Hungerbuehler, K. Daley, K. Cavanagh, H.G. Claro, and M. Kapps, “Chatbot-based assessment of employees’ mental health: Design process and pilot implementation,” JMIR Formative Research, vol.5, no.4, p.e21678, 2021.
[4] L. Moldon, M. Strohmaier, and J. Wachs, “How gamification affects software developers: Cautionary evidence from a natural experiment on github,” 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)IEEE, pp.549–561 2021.
[5] A. Nagatani, S. Chen, M. Nakamura, and S. Saiki, “Exploiting motivation subscales for gamification of lifelogging application,” International Journal of Software Innovation (IJSI), vol.10, p.27, Dec. 2022. DOI: 10.4018/IJSI.313445.
[6] A. Brocker, S. Judel, R. Roepke, N. Mihailovska, and U. Schroeder, “Gamifying jupyterlab to encourage continuous interaction in programming education,” International Conference on Games and Learning AllianceSpringer, pp.316–322 2022.
[7] A. Brocker, S. Judel, and U. Schroeder, “Integration of gamification and learning analytics in jupyter,” eleed, vol.14, no.2, 2022.
[8] A.C.T. Klock, A.N. Ogawa, I. Gasparini, and M.S. Pimenta, “Integration of learning analytics techniques and gamification: An experimental study,” 2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT)IEEE, pp.133–137 2018.
[9] J. Agapito, M.M. Rodrigo, and B.-P. Lorenzo, “Identifying meaningful gamification-based elements beneficial to novice programmers,” Proceedings of the 26th International Conference on Computers in Education, pp.619–624, 2018.
[10] “[gamification] ゲーミフィケーションでプロジェクトを改善するには? (1/5) [デスマーチ] | oshiire*blog,” https://oshiire.to/archives/3421. visited on 2023-10-21.
[11] “Develop extensions — jupyterlab 4.0.7 documentation,” https://jupyterlab.readthedocs.io/en/stable/extension/extension_dev.html. visited on 2023-10-21.