ログファイルと Git リポジトリを用いた Ruby on Rails の初学者の躓き要因の分析

高橋圭一[†1]

概要：Ruby on Rails（以降，Rails）は Ruby で書かれたオープンソースの Web アプリケーションフレームワークである．Rails を用いたシステム開発や Rails の機能拡張の提案などは公開直後から研究が進められているが，Rails の学習過程を調査した研究はない．我々はこれまで，筆者が所属する学科の Web アプリケーション開発科目の演習課題として提出されたログファイルを分析し，受講者が躓いたことを示す例外は 9 つあり，そのうち 2 つの例外の発生原因はログファイルだけでは特定が困難であるという結果を得た．本稿では，バージョン管理ソフトウェアの 1 つである Git を用いて例外発生時のソースコードを自動的に保存するスクリプトにより，2 つの例外の発生原因の特定を試みる．本稿では，この 2 つの例外を HIEs（Hard to Identify Exceptions）と呼ぶ．本スクリプトを 2020 年度の授業に適用したところ，33 名から提出されたログファイルから，HIEs が 325 回発生し，Git リポジトリの提出がある場合は，その情報を活用することですべての発生原因を特定できた．

キーワード：Ruby，プログラミング演習，開発フレームワーク

Novice programming mistakes of Ruby on Rails using log files and Git repositories

KEIICHI TAKAHASHI[†1]

Abstract: Ruby on Rails (Rails) is an open-source web application framework developed in Ruby. Immediately after the release of Rails, various studies were conducted, such as the development of application systems using Rails and proposals for extending its functionality. However, there have been no studies investigating the learning process of Rails. In this study, we analyzed the log files submitted as assignments for web application development in our department. The analysis revealed that there were nine exceptions that caused students to make mistakes. Furthermore, we observed that it is hard to identify the causes of two of these exceptions from the log files alone. In this study, we attempt to identify these hard to identify exceptions (HIEs) by having students use a script that automatically saves the source code when an exception occurs using Git, a version control software. When this script was used by the 2020 class, 33 students generated 325 exceptions related to the HIEs. If Git repositories were submitted, we were able to determine the causes of all the exceptions.

Keywords: Ruby, Programming Exercise, Development Frameworks

1. 背景

Rails は Ruby で書かれたオープンソースの Web アプリケーションフレームワークである．公開されてから 10 年以上経過しており，GitHub やクックパッドなど国内外に多くの利用者を抱えるサービスでも採用されている．Ruby の開発者が日本人であるため，日本語で読めるリソースがインターネット上に豊富にあり，国内の初学者が学びやすいことも特徴の 1 つと考えられる．Rails のリリース直後から，様々な研究が進められており，2020 年 9 月に Google Scholar で”Ruby on Rails”を指定してフレーズ検索したところヒット件数は約 11,000 であった．検索結果を分類すると，(a)Rails を用いた応用システム開発[1]，(b)Rails の機能拡張の提案[2]，(c)Rails と他フレームワークの比較[3]，(d)大学における PBL 実施報告[4][5]，の 4 種であった．このうち(d)については，Rails の初学者を対象としているものの，チーム開発やアジャイル開発など総合的なソフトウェア開発の教育メソッドとして計画し実施された報告であり，Rails 学習時の困難さについて具体的に調査した研究ではない．

初学者が Rails を学習する手段としては，Rails チュートリアルや ProGate などのオンラインドキュメント[6][7]，ドットインストールや Udemy などの動画教材[8][9]，書籍，プログラミングスクールがある．いずれも単独もしくは閉じた教育組織での学習であるため，学習者の躓きは共有されていない．

Rails は 2019 年 8 月にバージョン 6 がリリースされたばかりであり，今後も Web アプリケーションを開発する有効な手段として活用されることが期待できる．そこで，公開後 10 年以上経った状況ではあるが，初学者にとって Rails を学びやすくするための知見の蓄積は今後も必要であると考えた．

筆者が所属する学科に 3 年生対象の Web アプリケーション開発の科目がある[10][11]．この科目の演習課題として

†1 近畿大学産業理工学部情報学科
Kindai University

提出された Rails のログファイルを分析したところ，受講者が躓いたことを示す例外は 9 つあり，そのうち，約 8 割の受講者が発生した ActionView::Template::Error と ActionController::RoutingError の例外については，ログファイルのみでは発生原因の特定が困難であり，筆者の指導経験から発生原因を推察するに留まった[12]．この 2 つの例外を HIEs（difficult to identify factors）と呼ぶことにする．

本稿では，この課題を解決するため，ログファイルを監視し，躓き要因を表すキーワードがログファイルに追加されたときに，変更されたソースコードを自動的に保存するスクリプトを提案する．このスクリプトを前述の Web アプリケーション開発の科目に適用し，HIEs の発生原因の特定の可能性について評価する．

2. Rails 開発と誤りパターン

2.1 Rails の構成

開発者は，Rails が採用している MVC（Model View Controller）アーキテクチャを理解し，rails コマンドなどの各種ツールを使いこなすことで品質の高いソフトウェアを効率的に開発できる．図 1 に Rails の基本的な構成を示す．

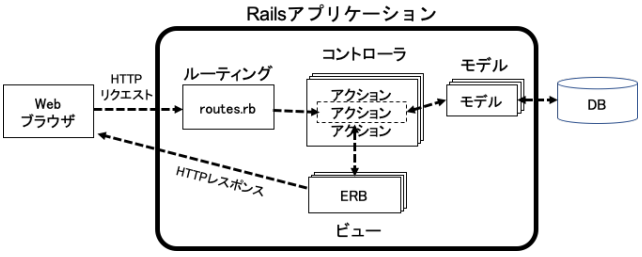

図 1 Rails の基本的な構成

Rails アプリケーションが Web ブラウザから HTTP リクエストを受信すると，ルーティング（routes.rb）の記述内容に従って，コントローラ（Ruby のクラス）内のアクション（Ruby のメソッド）が呼び出される．呼び出されたコントローラはモデル（Ruby のクラス）およびビュー（ERB ファイル[a]）を呼び出し，Web ブラウザに HTTP レスポンスを返す．MVC のそれぞれのファイルは rails コマンドを実行することで雛形を自動生成できるため，開発者はクラス定義など定型的なコードを記述する必要はない．

こうした仕組みは開発者にとっては便利であるが，初学者は Rails の仕組みや規約の理解が不十分であるため，エラーを起こさずにルーティングや MVC のモジュールを組み合わせることは困難である．一方，Rails のような開発用フレームワークは構成要素が多いため，同じエラーメッセージでもその発生原因は様々な状況が考えられる．こうしたデバッグの難しさが初学者の学習を困難にすると考えている．

2.2 Rails 開発手順と初学者の誤りパターン

授業で取り上げた，画像を共有する Rails アプリケーション（myapp）の開発手順を示しながら，筆者がこれまで学生指導で観察した初学者の誤りパターン（**EP1〜11**）について述べる．

① まず，プロジェクトを新規作成する．操作は *rails new myapp* である．この操作により，カレントディレクトリに myapp フォルダが新規作成され，その中に必要なファイル群が生成される．最初に，タイムゾーンや使用言語や使用するライブラリを設定ファイルに追記する．ここで記述を誤ると実行時にエラーが発生する（**EP1**）．

② 次にモデルを作成する．操作は *rails generate model Image title:string file:binary* である．この操作により，String 型の title と Binary 型の file のインスタンス変数をもつ Image クラスと，このモデルがアクセスするテーブル作成用のスクリプトファイルが生成される．*rails db:migrate* という操作で，前述のスクリプトファイルが実行されテーブルが作成される．この操作を忘れて Rails アプリケーションを実行するとエラーが発生する（**EP2**）．また，モデル名やカラム名や型の記述を誤る場合もある（**EP3**）．

③ 次にコントローラを作成する．操作は *rails generate controller images index* である．この操作によって，一覧表示のための index アクションを含んだ ImagesController クラスと index.html.erb というビューファイルが生成される．なお，基本的にモデル名は単数形で，コントローラ名は複数形である．このルールを忘れてコーディングするとエラーが発生するほか（**EP4**），手動でアクションを追加する場合は，ルーティングに対応するアクション名の誤りや，アクションを追加し忘れてエラーが発生する場合もある（**EP5**）．

④ 画像情報を一覧表示するため，開発者は，テーブルから読み込んだデータをインスタンス変数@images にセットするコードを ImagesController の index アクションに追加する（図 2）．この@images を用いて一覧表示する HTML を，ビューである index.html.erb に追加する（図 3）．コントローラとビューの別々のファイルの連携は Rails によって暗黙的に行われるため，以下のような誤りパターンによって様々なエラーに遭遇する場合がある．

- コントローラでテーブルデータをセットする変数に@をつけ忘れてしまい（ローカル変数になる），ビューで変数を受け取れない（**EP6**）
- コントローラとビューに記述したインタンス変数名が一致しない（**EP7**）

a Ruby コードを埋め込める HTML ファイル

- ビューファイルでの Ruby の記述誤り（**EP8**）

```
class ImagesController < Application
  def index
    @images = Image.all
  end
end
```

図 2 コントローラ記述例

```
<% @images.each do |img| %>
  <p>タイトル: <%= img.title %></p>
<% end %>
```

図 3 ビュー記述例（index.html.erb）

⑤ 最後に，Web ブラウザからアクセスされたパスとコントローラを紐つける定義を routes.rb に追加する（図 4）．ここでは以下のような誤りが考えられる．

- コントローラとビューを追加したが，ルーティングの定義を追加し忘れる（**EP9**）
- ルーティングの定義の記述を間違える（**EP10**）
- コントローラとビューを追加し，ルーティングの定義も正しく追加したが，A タグや FORM タグなどでパス名の記述を誤る（**EP11**）

```
Rails.application.routes.draw do
  get 'images/index', to: 'images#index'
end
```

図 4 ルーティング記述例（rountes.rb）

3. 分析手法

3.1 ログファイルの構成

開発中の Rails アプリケーションの実行ログは development.log というファイルに書き出される．このログファイルは開発者が意図的に削除しない限り，Rails プロジェクト作成直後から現在まですべての情報が記録される．

図 5 にログファイルの例を示す．Web ブラウザからアクセスされると Started で始まる情報が日時とともに記録される．Rails アプリケーションのコントローラやビューを実行したときに例外が発生すると，例外のクラス名とエラー情報がログファイルに出力される．この例では，ActionView::Template::Error という例外名とともに undefined local variable or method というエラーメッセージが出力されている．この例外では，エラー発生箇所のソースコード片が示されているため，ビューファイルで img とすべきところを img0 とタイプミスしたことが例外の発生原因であることがわかる．ここで注意が必要なのは，ログファイルに記載された例外クラス名が同じでも，発生原因となるエラーメッセージには様々な場合があるということである．エラー種別によってはソースコード片が示されない場合もあり，ソースコード片が示されている場合でも，他のファイルで定義した情報に影響を受けている場合は，ログファイルの情報だけでは発生原因の特定が難しい場合があるということである．

```
Started GET "/images/index" for ::1 at 2020-06-29 16:09:00 +0900
（省略）
ActionView::Template::Error (undefined local variable or method `img0'
Did you mean? img):
  1: <% @images.each do |img| %>
  2:   <p><%= img0 %></p>
  3: <% end %>
```

図 5 ログファイル（development.log）の例

3.2 ログファイルを用いた躓き検出方法

本稿では，個々の受講者の躓き状況には立ち入らず，多くの受講者が躓いた要因を特定することを優先する．多くの受講者が躓いたということは，講義や課題や開発環境など受講者に共通する部分に問題がある可能性があり，早急に原因を特定して改善する必要があると考えたからである．

具体的な躓き検出の方法としては，まず，ログファイル中の例外の個数をエラー発生回数とし，その回数分，受講者が躓いたと考える．受講者から提出されたログファイルをもとに例外別にエラー発生回数を求め，エラー発生回数が 1 以上の受講者数を集計し，これを全受講者数で割った受講者率を求める．本稿ではこれを**躓いた受講者率**と呼ぶ．

3.3 Git リポジトリに自動コミットするスクリプト

躓き要因の発生原因を特定するため，躓きが発生した時点のソースコードの変更情報を Git リポジトリに自動的にコミットするスクリプトを図 6 に示す．ログファイルである development.log を tail コマンドで監視し，ログに追加されたメッセージに Error という文字列が含まれているか grep コマンドで検査し，含まれていれば，躓き要因である例外名をコミットメッセージとして自動的にコミットを実行する．本スクリプトを受講者に配布し，各自のパソコンで起動してから課題に着手するように指示する．

```
#!/bin/sh
auto_commit() {
  while read i
  do
    echo $i | grep --line-buffered Error
    if [ $? = "0" ];then
      git add -A
      git commit -m "$i"
    fi
  done
}

touch development.log

tail -n 0 -F development.log | auto_commit
```

図 6 躓きを検知して自動コミットするスクリプト

3.4 ログファイルと Git リポジトリを用いた分析手順

ログファイルと Git リポジトリを組み合わせて HIEs の発生原因を分析する手順を具体例で説明する．

① まず，受講者が提出したログファイル（development.log）をテキストエディタで開き，HIEs を示す例外クラス名である ActionController::RoutingError もしくは ActionView:: Template::Error で

検索する．図 7 は，ある学生のログファイルを開き ActionView::Template::Error を指定して検索し最初にヒットした情報である．例外名の横に記載されているエラーメッセージを読むと「@images が nil::NilClass であるが，nil::NilClass には each メソッドは未定義である」が例外の原因とある．エラーメッセージの下にソースコード片が記録されているが，この情報だけでは@images が nil になった原因はわからない．

② そこで，この例外が発生した時点のソースコードを参照するため，この例外が発生した日時を調べる．例外クラス名の直前にある Started を目視で探したところ，2020 年 7 月 17 日 0 時 31 分 9 秒にアクセスがあったときに例外が発生したことがわかる．

③ Git リポジトリのコミット履歴の一覧を表示し，例外発生日時に追加されたコミット(74cff55)に辿り着く．チェックアウトして例外発生時のソースコードを調べるとコントローラは図 8 の状態であった．インスタンス変数@images はおろか，アクションも未定義であるため例外が発生したことが確かめられた．

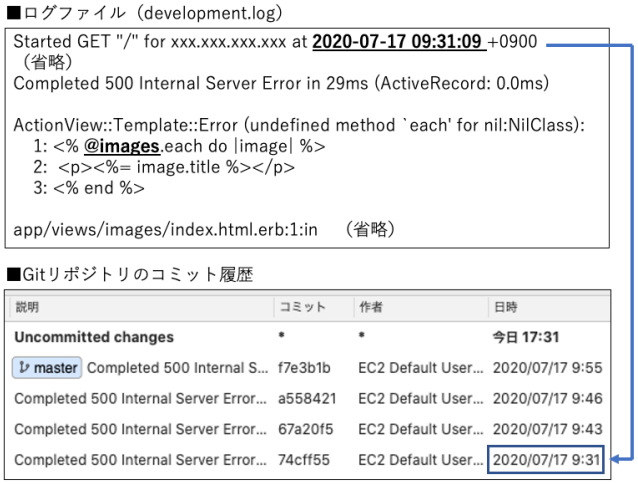

図 7 ログファイルと Git リポジトリの情報の対応付け

```
class ImagesController < ApplicationController
end
```

図 8 Git リポジトリを探索して発見したコントローラ

4. 実験

4.1 実験方法

筆者が所属する学科に 3 年生対象の Web アプリケーション開発の科目がある[10][11]. 講義 1 コマと演習 1 コマが時間割で連続しており，2019 年度は講義で基礎を学んだあと，演習室に移動して課題に取り組んだが，2020 年度は，講義も演習もすべて Zoom を利用したオンライン授業形式で実施した．15 回の演習のうち，第 8 回までは Ruby の応用プログラミングを学び，第 9 回から Rails について学習する．第 9 回と第 10 回に CRUD アプリケーションの作成方法を学習し，第 11 回にファイルアップロード機能をもつ画像共有アプリケーションの作成方法を学習する．開発手順は 2.2 節に示した通りである．3.3 節に示したスクリプトを受講者全員に配布し，課題に取り掛かる前に実行するように依頼した．この第 11 回に受講者が取り組んだ Rails アプリケーションのログファイルと Git リポジトリの情報を収集する．なお，ログファイルも Git リポジトリも Rails プロジェクトを新規作成したときに自動生成される標準的なファイルであり，課題提出のための特別な操作は不要である．課題の提出期間は 1 週間であったが，2020 年度の受講者 57 名のうち 33 名から期限内に課題提出があった．提出された課題はすべてアプリを完成しており，躓いても何らかの方法で自己解決できたとみなせる．

4.2 講義及び課題の内容

第 9 回と第 10 回で基本的な手順は学習済みのため，第 11 回の講義では，開発手順の定着を促すため，入力すべきコマンドやプログラムを空欄にした資料を配布し，講義中に解説しながら受講者に記入させた(図 9)．演習課題は「講義資料をもとに画像共有 Web アプリを完成させなさい」であり，講義中の解説をもとに受講者が資料を正確に完成していれば，課題完成までに各自で試行錯誤を必要としない．2020 年度では，前述のとおり講義も演習もすべて Zoom を利用したオンライン授業形式で実施し，講義終了後には，録画した講義動画を受講者に公開したため，受講者は必要であれば繰り返し視聴して手順を確認できる環境であった．

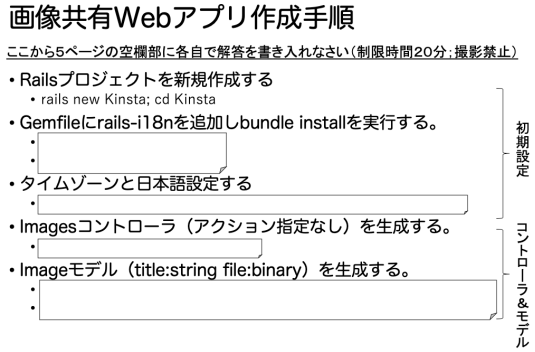

図 9 講義スライド例

5. 実験結果

5.1 ログファイルから得られた躓き状況

全ログファイルから抽出した例外は 7 種類であった．3.2 節に示した方法で**躓いた受講者率**を求めた結果と例外の発生回数を表 1 に示す．比較のため 2019 年度に調査した結果を併記した[12]．全体的に躓いた受講者率は減少傾向にあるが，HIEs の 1 つである ActionController::RoutingError は 0.75 から 0.39 と大幅に減少している．講義スライドと演習課題は両年度でほぼ同じであるが，前述のとおり 2020 年度はオンライン授業であったため，受講者は講義の解説を視聴しながら自宅パソコンを操作できた．そのため，手順誤りや記述箇所の間違いが減少したと推察される．一方，

もう 1 つの HIEs である ActionView::Template::Error は昨年と同様に約 8 割の受講者が躓いたことがわかる. HIEs の発生回数は 129+196 で 325 であった．なお，2019 年度の発生回数は 200+171 で 371 であった.

表 1 ログファイルから得られた例外ごとの躓き状況

例外クラス名	躓いた受講者率		発生回数
	2020年度	2019年度	
ActionView::Template::Error	**0.85**	**0.79**	**129**
ActionController::RoutingError	**0.39**	**0.75**	**196**
SyntaxError	0.39	0.67	52
NoMethodError	0.45	0.53	54
NameError	0.39	0.40	26
ActiveRecord::PendingMigrationError	0.00	0.13	0
ActiveModel::UnknownAttributeError	0.03	0.03	4
Encoding::UndefinedConversionError	0.00	0.03	0
LoadError	0.03	0.03	2

5.2 Git リポジトリを用いた HIEs の発生回数

3.4 節の方法で HIEs の発生原因を分析した結果を表 2 に示す．HIEs の列は，ActionView::Template::Error と ActionController::RoutingError の例外発生回数を合計である. HIEs の発生回数は 325 であり, そのうちログファイルのみで発生原因が特定できたのは 106 であり，発生原因の特定に Git リポジトリの情報が必要なのは 148+71 であった. Git リポジトリの提出があった 148 の例外については，Git リポジトリのソースコードを参照することですべて発生原因を特定することができた. 一方, Git リポジトリの提出がない受講者が 6 名いたため，その受講者が発生した 71 の HIEs の原因の特定はできなかった. HIEs の内訳をみると，ActionView::Template::Error はビューに関する例外であるため，ログファイルのみで原因特定ができる場合が多いようである．一方，ActionController::RoutingError はルーティングに関する例外ではあるが, 発生原因は様々な場合があり，ログファイルのみでは特定が難しいことがわかる.

Git リポジトリの提出なしを除くと，今回の実験で得られたログファイル中の HIEs のうち 79%（33%+46%）の発生原因が特定できたことになる．次節では，この情報を用いて発生原因を誤りパターンを用いて整理する.

表 2 HIEs の例外発生回数と原因特定に必要な情報源

発生原因の特定に必要な情報源	**HIEs (回)**	HIEs (%)	Template Error (回)	Routing Error (回)
ログファイル	**106**	33%	76	30
ログファイル＋Git（提出あり）	**148**	46%	40	108
ログファイル＋Git（提出なし）	**71**	22%	13	58
合計	**325**	100%	129	196

5.3 HIEs の発生原因となった誤りパターンの発生割合

特定できた HIEs の発生原因を 2.2 節に挙げた誤りパターンを用いて類別し，それぞれの発生回数が全体に占める割合を表 3 と表 4 に示す.

ActionView::Template::Error はビューに関する例外であるから，ビューに誤りがあることを示すエラーメッセージが表示される．表 3 によると EP8 の発生割合は 66%であることから，この例外に関してはまずビューに誤りがあると考えて確認する必要がある．一方，EP5～7 の発生割合は 34%あることから，ビューを参照して誤りが見つからない場合はコントローラの誤りも確認する必要があることがわかる.

一方，ActionController::RoutingError はルーティングに関する例外であるが，今回の実験で発生した例外のエラーメッセージを読むと，すべてビューに誤りがあることを示していた. 表 4 によると EP11 の発生割合は 38%であるから，エラーメッセージに従ってビューを確かめたあと，問題なければ関係するルーティング（routes.rb）の記述内容を確認すればよいことがわかる.

表 3 ActionView::Template::Error の誤りパターンの発生割合

誤りパターン	発生割合
コントローラ名を複数形にしてない（EP4）	0%
コントローラのアクション定義忘れ（EP5）	24%
ビューに渡す変数に@のつけ忘れ（EP6）	4%
コントローラ・ビューの変数名の不一致（EP7）	6%
ビューで文法誤り（EP8）	66%

表 4 ActionController::RoutingError の誤りパターンの発生割合

誤りパターン	発生割合
ルーティング定義忘れ（EP9）	13%
ルーティング定義の記述誤り（EP10）	49%
Aタグなどのパス名の記述誤り（EP11）	38%

6. 考察

6.1 分析方法（スクリプト）の効果

実験の結果から，ログファイルのみを用いた場合，発生した HIEs 全体の 33%しか発生原因が特定できなかった(表 2). 一方, 3.3 節に示したスクリプトを利用することで HIEs 全体の 79%（33%+46%）の発生原因を特定できたことになる．これまでの研究では，ログファイルに記載された情報をもとに発生原因を分析し，情報が不足する場合には指導経験をもとに推察するに留まった．本スクリプトを使用することで例外が発生するたびに変更されたソースコードが Git リポジトリに保存されるため，前回の例外発生時点から今回までに受講者がどういった変更を加えたのかを確実に読み取ることができるようになった．例外発生時のソースコードが明らかになったことから，本講義の内容には限定されるものの，HIEs の発生原因は 2.2 節に挙げた誤りパターン以外の誤りは存在しないことが確認できた．今後，誤りパターンをもとに表 3 や表 4 に示した発生割合を考慮したデバッグ手順を開発し，受講者の指導に活用する道筋が開けたと考えている.

6.2 分析方法（スクリプト）の問題点

Git リポジトリの提出がない受講者が 6 名おり，その受講者が発生した 71（全体の 22%）の HIEs の発生原因はわからなかった．本稿では簡単のため受講者の開発環境について触れなかったが，演習では Amazon 社の AWS Cloud9 というクラウドベースの Linux 環境を前提としていため，3.3 節に示したスクリプトは Linux 版のみ用意した．実際には確認していないが，学生によっては MS-Windows 上に Rails の開発環境を構築して課題に取り組んだ場合も考えられる．この場合は Git リポジトリの提出は不可能である．また，Git リポジトリの提出はあったもののログファイルに記載された例外と対応しない場合も「Git リポジトリ提出なし」としてカウントした．恐らく，スクリプトを少なくとも一度は実行したものの，何らかの理由でスクリプトが停止してコミットが実行されない状況があったのではと考える．対策案としては，Rails アプリケーションの動作を確認するときには，*rails server* というコマンドを入力して開発用の Web サーバーを起動する．この操作時にスクリプトが停止していれば起動する機能を追加することで，スクリプトの起動忘れを防止できるかもしれない．

現状のスクリプトの問題に，コミットメッセージが例外名と一致しないことがある．図 7 のコミット履歴のコミットメッセージはすべて Completed 500 Internal Server Error…となっているが，本来は例外クラス名と関連するエラーメッセージがセットされる予定だった．例外発生時にログファイルに追加される情報を調べると，例外によっては Error を含む行が複数存在する場合があることがわかった（図 10）．今回の実験ではコミットの実行漏れはなかったが，今後，同様のスクリプトを使用する場合は Completed 500 Internal Server Error を含む行を無視するよう改善する．

```
Started GET "/" for xxx.xxx.xxx.xxx at 2020-07-17 00:31:09 +0000
（省略）
Completed 500 Internal Server Error in 29ms (ActiveRecord: 0.0ms)


ActionView::Template::Error (undefined method `each' for
nil:NilClass):
   1: <% @images.each do |image| %>
   2:  <p><%= image.title %></p>
   3: <% end %>
（省略）
```

図 10 例外発生時のログファイルの追加情報

6.3 分析方法（スクリプト）の動作環境

スクリプトを動作させるためには Git リポジトリの初期化が必要であるが，Rails は標準で Git に対応しているため，Rails プロジェクトを新規作成するときに Git リポジトリも新規作成される．そのため，操作を忘れて記録漏れが発生することはない．動作環境としては，ローカル環境はもちろん，我々が演習で利用している AWS Cloud9 のようなクラウド環境や Docker を利用した仮想環境でも利用できる．実用的な開発用フレームワークであれば，Rails の development.log のようなログ機能を具備しているため，スクリプトを用いて同様の分析が可能である．本稿で提案したスクリプトは極めて単純な方法ではあるが，ソースコードなどテキストファイルの変更履歴を自動的に蓄積し，分析する手段の 1 つとして，様々な場面で利用できると考えている．

7. まとめ

本稿では，Rails の初学者の躓き要因を分析するため，ログファイルに加えて，例外発生時のソースコードを自動的に保存するためのスクリプトを提案した．本スクリプトを 2020 年度の授業に適用したところ，Git リポジトリの提出があった HIEs の発生原因はすべて特定でき，その発生原因は筆者が指導経験から整理した誤りパターンに当てはまることがわかった．

これまでの研究で，講義で扱っている範囲ではあるが，Rails 開発時の躓き要因とその発生原因が明らかになった．我々は初学者が上達するためには躓きは必要な経験と考えている．しかし，現状の演習では躓いたときに講師や TA が支援するに留まり，デバッグ技術を積極的に指導するまでには至っていない．今後，これまで得られた情報をもとにデバッグ手順を開発し，講義や演習でその効果を評価していきたい．また，デバッグ手順を他の初学者向けの教育方法[6][7][8][9]に適用して，一般化が可能か評価していきたい．

参考文献

[1] Chunling, Cao : Construction of the Individualized College English Learning Management System Using Ruby on Rails, International Conference on Service Science, pp.160-163, 2015.

[2] An, Jong-hoon, Avik Chaudhuri, and Jeffrey S. Foster : Static typing for Ruby on Rails. IEEE/ACM International Conference on Automated Software Engineering, pp.590-594. 2009.

[3] Crawford, Tyler, and Tauqeer Hussain. A comparison of server side scripting technologies. Proceedings of the International Conference on Software Engineering Research and Practice, pp.69-76, 2017.

[4] 井上明，金田重郎：実システム開発を通じた社会連携型 PBL の提案と評価，情報処理学会論文誌, Vol.49, No.2, pp.930-943, 2008.

[5] Chishiro, Hiroyuki, Yoshihide Chubachi, and Morikazu Nakamura, Co-education of master course students and business people, Proceedings of the 6th International Conference on Information and Education Technology, pp.89-96, 2018.

[6] Rails チュートリアル, https://railstutorial.jp（2020-10-1 参照）

[7] Progate Ruby on Rails5, https://prog-8.com/languages/rails5（2020-10-1 参照）

[8] ドットインストール Ruby on Rails5 入門, https://dotinstall.com/lessons/basic_rails_v3（2020-10-1 参照）

[9] Udemy, https://www.udemy.com（2020-10-1 参照）

[10] 高橋圭一, Ruby on Rails による Web アプリ開発の授業実践, 情報処理学会九州支部火の国シンポジウム 2018, 2019.

[11] 高橋圭一，Ruby on Rails によるチーム開発の授業実践，情報処理学会 情報教育シンポジウム 2019, pp.10-16, 2019.

[12] 高橋圭一，Ruby on Rails の初学者の躓き要因分析，ソフトウェア工学の基礎 XXVII，日本ソフトウェア科学会（FOSE2020），2020 年 11 月（in press）．