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

信学技報
TECHNICAL REPORT OF IEICE.

可搬性と拡張性を備えたコード品質可視化プラグインの試作

華山　魁生† 　柗本　真佑† 　楠本　真二†

† 大阪大学大学院情報科学研究科 〒 565–0871 大阪府吹田市山田丘 1–5
E-mail: †{k-hanaym,shinsuke,kusumoto}@ist.osaka-u.ac.jp

あらまし　著者らは既存研究において，家庭用ゲームで用いられている「実績」と呼ばれるコンセプトをプログラミング教育に導入し，CI ツールの援用の下に学生にも理解しやすい形でプログラムの品質を提示する教育手法を提案した．またこの教育手法を実現するシステムである Ave を実装し，適用実験において品質に対する学生の意識改善およびプログラムの品質向上に効果があることを確認した．しかし，Ave は実験対象の授業に特化する形で構築されていたため可搬性および拡張性が低く，他の授業にこのシステムを適用することは難しい状況であった．そこで本研究では，既存研究の拡張として CI ツール Jenkins のプラグインである Ape を開発し，Ave と同様の機能を Jenkins に統合する形で実現した．これにより教員は Ape を適用するだけで，実績システムを容易に授業内に導入できる．加えて，Jenkins 内に閉じた環境の中でシステムが実現されているため，テスト結果，品質計測結果および実績達成状況をシームレスに閲覧できる．さらに Ape では，設定ファイルを用いた柔軟な実績項目の定義が可能であるため，特定の授業に特化しない汎用的な利用が可能である．

キーワード　プログラミング教育, プログラム品質, CI ツール, 実績, Jenkins, プラグイン

1. はじめに

効率的かつ効果的なプログラミング教育の手法として，自動テストに基づく手法が数多く提案されている [1] [2]．この教育手法では，学生が提出したプログラムに対してテストを自動で行い，その通過可否で課題達成の判定を行う．以降ではこのような形態で行われる教育を**テストベースの教育**と呼ぶ．テストベースの教育は，学生のプログラミングに対する動機付けや教員の負担軽減に効果があることが知られている [3] [4] [5] [6]．しかし，テストベースの教育ではプログラムの外的振る舞いに主眼を置くことが多く，内的構造の良さすなわちプログラム品質は軽視される傾向にある [7] [8].

著者らの既存研究 [9] では上記の課題を解決するため，テストベースの教育に家庭用ゲームで用いられている「実績」と呼ばれるコンセプトを導入することを提案した．従来のテストベースの教育手法と提案手法の概念図を図 1 に示す．提案手法では品質計測を行いその結果を実績に加工する．従来の品質計測ツールとは異なり，問題のない箇所に注目しそれを実績達成という形で褒めることで，プログラムの品質向上に対する動機付けが可能である．この手法を実現するためのプロトタイプのシステムとして Ave を実装し，適用実験として大阪大学基礎工学部情報科学科の学部 3 年生を対象とした情報科学演習 D(注1)（以降，**演習 D**）に導入した．その結果，プログラム品質に対する受講生の意識向上およびプログラミングに対するモチベーション向上が認められた．

しかし，Ave は演習 D を前提とした実装となっていたため，可搬性(注2)，特に適応性と設置性の観点で課題があった．適応性に対する課題としては，実装がハードコーディングされており実績項目を容易に変更できない点が挙げられる．設置性に対する課題としては，演習環境のログや品質計測結果の処理を行うスクリプトを多数使用しておりそれらが全くパッケージ化されていないため，インストールを容易に行えない点が挙げられる．以上の理由から，実績システムを他授業へ適用するためには，実績項目のカスタマイズとインストールを容易に行える仕組みが必要である．

そこで本研究では，既存研究の拡張として CI ツール Jenkins のプラグインである Ape を開発し，Ave と同様の機能を Jenkins に統合する形で実現した．機能面において Ape は Ave とほぼ同等の機能を実現しており，Ape 使用により上記で述べた実績システムのメリットを享受することができる．実装面においては，Jenkins プラグイン化によって可搬性が大幅に向上し，教員はプラグインを適用するだけで容易に実績システムの導入が可能である．さらに，Ape では設定ファイルによって自由に実績項目を設定可能であるため拡張性が大幅に向上し，実績項目および閾値のカスタマイズを容易に行える．

Ape の有用性を確認することを目的として，大学教員 2 名に

(注1)：https://loki.ics.es.osaka-u.ac.jp/

(注2)：本論文では可搬性という言葉を ISO/IEC 25010:2013 における製品品質モデルの特性の一つ，Portability（移植性とも呼ばれる）の意味で用いる．この Portability は 3 つのサブ特性，適応性（Adaptability），設置性（Installability），置換性（Replaceability）から構成される．

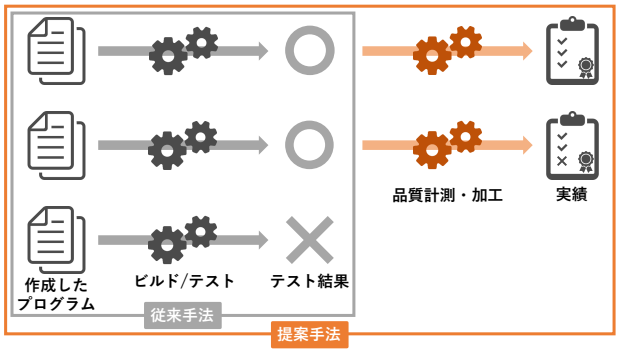

図 1　従来のテストベースの教育手法と提案手法の概念図

Ape を使ってもらい評価実験を行った．その結果，実績システムの有用性，Ape 導入の簡潔さおよび Ape 使用の有効性を確認した．

2. 準　　備

2.1 テストベースの教育と課題点

本研究では，学生が提出したプログラムに対してビルドを自動で行い，教員が事前に定めたテスト通過の可否で評価を行う教育手法を**テストベースの教育**と呼ぶ．テストベースの教育では，プログラムの振る舞いの正しさを自動かつ即座に評価でき，プログラミングに対する動機付けにも一定の効果がある [6] ほか，教員の負担も大きく軽減できる [3] [4] [5]．

テストベースの教育には上記のような利点がある一方，プログラムの外的振る舞いの正しさのみに焦点を当てた二値的なフィードバックしか与えず，プログラムの内的構造の良さを軽視しているという欠点も抱えている [7] [8]．以降では，この内的構造の良さを単に**プログラム品質**と呼ぶ．

PMD や Checkstyle といった品質計測ツールは数多く開発されており，これらのツールを使用することでプログラム品質の客観的な提示が可能である．また Jenkins に代表される CI ツールとの相性も良く，品質計測ツールと CI ツールを組み合わせることで，学生の提出したプログラムのテスト後に自動で品質を計測できる．

しかし，品質計測ツールの計測結果は問題箇所を事細かに指摘した細密かつ膨大なものとなる [10] ため学生はそれを活用することができず，プログラミングに対する苦手意識を植え付けかねない．さらにメトリクスの計測を行う場合，品質の高さに基準を設けるという難しさがある．品質計測結果には基準が定められていないため，学生は自身の作成したプログラム品質がどれほどの水準を達成しているのか判断できない．以上の理由から，品質計測ツールをそのままテストベースの教育に導入することは難しいと言える．

2.2 既存研究：Ave

我々の既存研究 [9] の目的は 2.1 節で述べたテストベースの教育における課題を解決すること，すなわちプログラム品質を取り上げ，学生の作成するプログラムの品質を向上させることであった．我々はこの目的を達成するために，テストベースの教育に「実績」と呼ばれるコンセプトを取り入れることを提案した．実績とは家庭用ゲームで用いられているコンセプトであり，一定の基準を満たすことによって得られる証や業績を示すものである．品質の高さを特徴づける項目を実績として設定しテストベースの教育に取り入れることで，品質の高さに基準を設けて線引きを行い，プログラム品質に関するノウハウを学生にも理解しやすい形で示すことができる．また実績というゲーミフィケーション(注3)の要素をプログラミング教育に導入することで，プログラム品質に対する学生の意識改善とモチベーション向上を促すことができる．

既存研究 [9] ではこの教育手法を実現するシステムである Ave を実装し，演習 D に導入して適用実験を行った．結果として，品質に対する学生の意識改善やプログラミングに対するモチベーション向上への効果が認められた．

2.3 Ave の課題点

Ave は 2.2 節に述べた利点を持つ一方，可搬性の低さと拡張性の低さが問題となっていた．まず前提として演習 D の受講生は，CI ツール Jenkins，Git プラットフォーム GitBucket およびビルドツール Gradle による演習環境を用いて演習を行う．

可搬性の低さとして，実績システムのインストールを容易に行えない点が挙げられる．Ave では演習環境のログや品質計測結果の処理を行うスクリプトを多数使用しており，それらは全くパッケージ化されていない．従って，同様の演習環境が用意されたとしても Ave を全て実装し直さなければ実績システムを導入できない状態であった．拡張性の低さとして，実装がハードコーディングされており実績項目を用意に変更できない点が挙げられる．Ave では演習 D の課題内容に即して実績項目を策定していた．しかし，我々が提案した教育手法の利点を最大限に活かすためには，導入対象の授業の性質や課題の難易度に合わせて実績項目を変更するべきである．

3. 提 案 手 法

3.1 実績システムのプラグイン化

本研究の目的は，2.3 節に示した Ave の課題点を解決し，実績システムの可搬性と拡張性を向上することにある．我々はこの目的を達成するため，実績システムを CI ツール Jenkins のプラグインとして実現することを提案する．このプラグインは機能面においては Ave とほぼ同等の機能を実現する．すなわち品質計測結果を取得しそれを実績に加工して示すことで，学生に理解しやすい形でプログラム品質が提示できる．そして本研究の主たる貢献は，実装面の改善すなわち可搬性と拡張性の向上にある．

可搬性の向上

プラグイン化により Ave と同様の機能を Jenkins に統合する形で実現しており，可搬性が大幅に改善している．Ave は演習 D を前提として実装されていたため，他の授業に実績システムを導入する事はほぼ不可能であった．しかしプラグイン化によって，Jenkins が運用されている授業であればプラグインを

(注3)：コンピュータ・ゲームのなかで特徴的に培われてきたノウハウを現実の社会活動に応用すること [11]

適用するだけで実績システムの導入が可能である．

拡張性の向上

本プラグインでは自由に実績項目を設定可能であり，拡張性が大幅に改善している．Ave で使用していた実績項目は演習 D に合わせて策定されており，さらに Ave の実装内容を書き換えなければ実績項目を変更できなかった．しかし，本プラグインでは設定ファイルによって実績項目および閾値のカスタマイズを容易に行える．

3.2 メトリクスベースの実績とルールベースの実績

既存研究 [9] と同様に，本研究でも実績というコンセプトを取り入れている．実績項目を策定するにあたり，本研究で対象とするプログラムの品質を 2 種類に大別し，以下のような実績項目とした．

メトリクスベースの実績：定量的な計測項目に基づく実績
例：「メソッド当たりの最大コード行数が 30 行以下」「メソッド当たりの最大サイクロマチック数が 10 以下」

ルールベースの実績：二値的な計測項目に基づく実績
例：「空の `if` 文が存在しない」「不必要な変数が存在しない」

メトリクスベースの実績には，閾値の異なる複数の難易度を設定し段階を設ける．例えば「メソッド当たりの最大コード行数が規定値以下」という実績に対し，「Lv.1：50 行以下」，「Lv.2：30 行以下」，「Lv.3：10 行以下」といった段階を設けることで，プログラムの品質に基準を設け理解を深めさせる．また Lv.1 を比較的容易な難易度に設定することで，実装時の品質向上に対する動機付けに役立てる．

3.3 設定ファイルによるカスタマイズ

本プラグインでは，実績項目を設定ファイルで定義することが可能である．品質計測ツールが提供するルールセットの中から「メソッド当たりのコード行数が規定値以下」「不必要な変数が存在しない」など使用するルールを選別していくことで，品質計測内容を自由に編集することができ，それを実績に加工できる．すなわち品質計測ツールが計測できる品質項目は，本プラグインにより全て実績に加工可能である．さらにメトリクスベースの実績の場合は閾値も自由に変更可能なため，授業の性質や課題の内容に合わせて実績の難易度を容易に調整できる．

3.4 プリセットの実績

本プラグインでは，使用するルールセットと閾値を事前に定義済みの実績（以降，**プリセットの実績**）を用意している．現時点で用意されているプリセットの実績を表 1 に示す．3.3 節で述べたように，本プラグインでは実績項目を容易にカスタマイズできる．これは拡張性が改善し自由度が向上している一方で，教員が実績項目を一から設計しなければならないという欠点も抱えている．プリセットの実績を用いることで，既存研究 [9] により学生の意識改善やプログラムの品質向上における有用性が既に示されている実績項目を，簡単に使用することが可能である．あるいは実績項目を策定する際の設計例として，プリセットの実績を用いることもできる．

(注4)：クラス名にパスカルケースを用いていないなど，慣習に反する命名

4. 実装プラグイン：Ape

4.1 概　　要

我々は 3 章で述べた提案手法を実現するためのプロトタイプとして **Ape** を実装した．Ape とは，“Ave Plugin Extention” の略称であり，Ave で実現した実績システムを CI ツール Jenkins のプラグインとして実装したものである．既存システム Ave と提案プラグイン Ape によるテストベースの教育の流れを図 2 および図 3 に示す．以下，順にテストベースの教育の流れについて説明する．

1. 教員が課題となるプログラムの仕様を策定し，テストコードとルールセットを保存
2. 教員が Jenkins 上でプログラムの振る舞いをチェックするテストタスクを登録
3. Ave は実績の登録が不可能であるが，Ape は実績設定ファイルにより実績を登録可能
4. 学生が仕様を満たすプログラムを作成し，コードを提出
5. Jenkins が作業用ディレクトリにコードを Pull
6. Jenkins がビルドツールを用いてプログラムをビルドおよびテスト
7. Jenkins が品質計測ツールを用いてプログラム品質を計測
8. 学生が品質計測結果を確認
9. Ave は実績の処理に多数のスクリプトを実行する必要があったが，Ape は Jenkins と連携しているため簡潔に処理を実施可能
10. Ave は実績達成状況を確認するために Jenkins と異なる Web サイトを閲覧する必要があったが，Ape ではテスト結果，品質計測結果および実績達成状況を全て Jenkins 上で閲覧可能

4.2 Ape の画面

Ape における実績達成状況の確認画面を図 4 に示す．この画面では，各課題の達成状況を確認することができる．チェックマーク（“✓”）が付いているアイコンは達成済みの実績，エクスクラメーションマーク（“！”）が付いているアイコンは未達成の実績を表している．このように，実績の達成状況を可視化し一目で判断できるようにすることで，プログラム品質の高さを端的に示せる．画面の上半分がメトリクスベースの実績，下

表 1　プリセットの実績一覧

名前	達成条件
Lines of Code	各メソッドの 最大コード行数が規定値以下
Nest	`for` 文，`if` 文，`try` ブロックの 各ネストの深さが規定値以下
Cyclomatic Number	各メソッドの最大サイクロマチック数が 規定値以下
Naming Conventions	不適切な命名(注4)を行っている クラス，メソッド，変数の個数が 規定値以下
Import	適切にインポートを行っている
Magic Number	ソースコード内に マジックナンバーがない

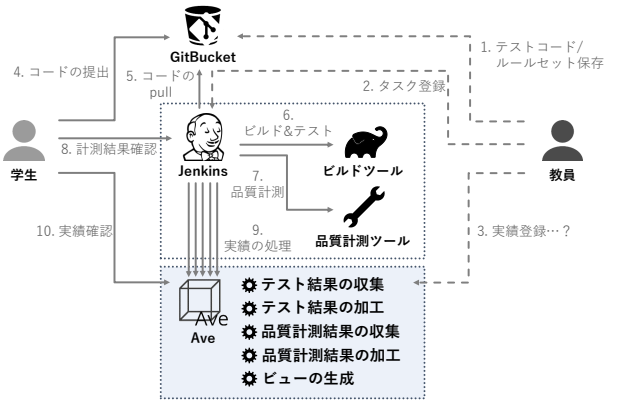

図 2　既存手法 Ave によるテストベースの教育の流れ

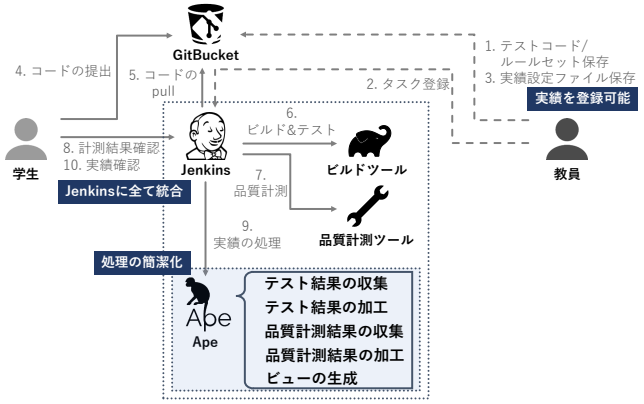

図 3　提案手法 Ape によるテストベースの教育の流れ

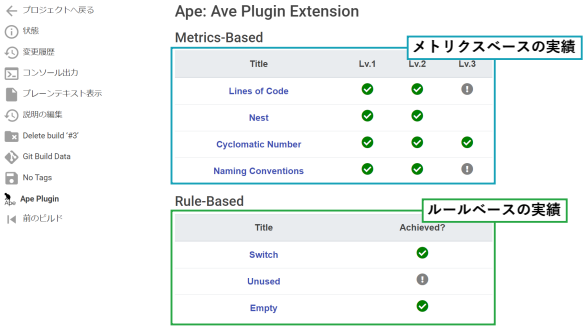

図 4　実績達成状況の確認画面

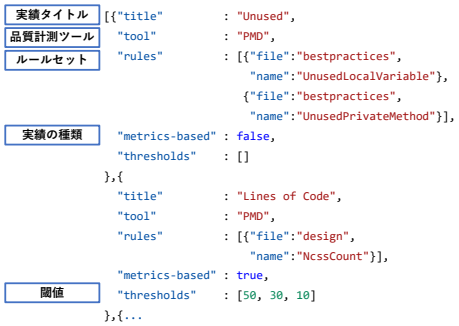

図 5　実績設定ファイルの記載例

半分がルールベースの実績を表している．メトリクスベースの実績は複数の段階が設けられており，各行が「各メソッドの最大コード行数が規定値以下」といった実績項目，各列が左から Lv.1，Lv.2，Lv.3 の段階を表している．また “Nest” の実績に示されているように，設定ファイルによって 1 段階あるいは 2 段階の実績とすることも可能である．ルールベースの実績は段階のない実績であり，各行にあるアイコンが 1 つの実績項目を表している．

プリセットの実績の場合は，実績のタイトル部分をクリックすると詳細が確認できる API ドキュメントにジャンプする．この API ドキュメントは Ape の実装と並行して我々が執筆したものであり，使用しているツールの説明，コード例，実績達成による利点，プログラミングにまつわるコラムなどを閲覧できる．これにより，プログラム品質に関する様々な知識を提供でき，学生の品質向上に対する意識付けが可能となる．

4.3　設定ファイルとジェネレータ

Ape では，実績項目を JSON 形式の設定ファイルにより定義することが可能である．以降，この設定ファイルを**実績設定ファイル**と呼ぶ．実績設定ファイルの記載例を図 5 に示す．実績設定ファイルには，実績項目のタイトル，使用する品質計測ツール，使用するルールセット，メトリクスベースの実績かルールベースの実績かを示す実績の種類，そしてメトリクスベースの実績の場合はその閾値を記載する．実績設定ファイルの編集後，Jenkins と連携しているリポジトリ内に保存することで，記載内容に従った実績の表示が行われる．このように，Ape では実績設定ファイルにより柔軟な実績項目の設定が可能であり，実績システムを様々な授業に適用可能である．

さらに，我々は GUI で実績設定ファイルを生成できるジェネレータ（以降，**JsonGen**）を作成した．JsonGen の画面を図 6 に示す．各行はそれぞれ 1 つの実績項目に対応している．左上の “Add row” ボタンから行の追加すなわち実績項目の追加が可能であり，“Add preset achievement” ボタンから表 1 に示したプリセットの実績の追加が可能である．各列は左から順に，実績のタイトル，使用する品質計測ツール，使用する品質計測ツールのルールセット，メトリクスベースの実績かルールベースの実績かを選択するチェックマーク，そしてメトリクスベースの実績の場合はその閾値である．これらの内容は図 5 に示した実績設定ファイルの内容と一致している．右の “Delete” ボタンをクリックするとその行の内容を削除する．左下の “Show settings!” と書かれたボタンをクリックすると，入力した値に基づいて図 5 のような実績設定ファイルの内容を表示する．その後，“Copy” ボタンをクリックするとその内容をクリップボードにコピーできる．そして右下の “Download!” と書かれたボタンで，実績設定ファイルをダウンロードできる．このように JsonGen では GUI による簡潔な操作で実績設定ファイルを生成できるため，教員の使用性向上が可能である．

5.　評 価 実 験

5.1　実 験 設 計

Ape の有用性の確認を目的として，被験者実験を行った．既存研究 [9] では学生側の観点から実験を行ったが，本研究では教員側の観点から実験を行うことを指針とした．そのため，大学教員である被験者 2 名に Ape を使用してもらったのち，Ape を使用した印象，良かった点や今後改善を望む点などについて

図 6　JsonGen の画面

コメントを収集し分析した．

実験を行うため，図 3 に示されたテストベースの教育を模した実験環境を構築した．ただし初期環境では品質計測と Ape の導入は行われておらず，被験者には現状の確認後に，品質計測の導入および Ape による実績システムの導入を行ってもらった．

5.2　被　験　者

被験者 2 名はいずれも情報科学の分野に所属する大学教員であり，大阪大学に所属する教員 1 名（以降，**被験者 A**）と大阪工業大学に所属する教員 1 名（以降，**被験者 B**）である．実験に先立ち，被験者には基本情報として教員歴，プログラミングを教える授業およびテストベースの教育の担当経験を尋ねた．またテストベースの教育に関連するツールおよびシステム，すなわち Gradle などのビルドツール，Jenkins などの CI ツール，PMD などの品質計測ツール，Git などの版管理システムの使用経験を尋ねた．その結果を表 2 に示す．なお，いずれの被験者も既存研究 [9] について実験前に知っていた．

5.3　実 験 結 果

5.3.1　概　　要

以降では，実験の結果を「実績システム」「Ape の導入」「Ape の使用」の 3 つの観点から分析し述べる．「実績システム」について，本実験は Ape のプラグインとしての有用性を試すために行ったものであるが，実績システムと密接に関連している．既存研究では主に学生の視点から実績システムの評価を行ったため，本実験では教員の視点から評価を行ってもらった．また「Ape の導入」は Ape を授業に取り入れる際，「Ape の使用」は Ape を使用する際に予想される利点と欠点の評価である．

表 2　被験者の基本情報とツールおよびシステムの使用経験

	被験者 A	被験者 B
教員歴	約 2 年	約 12 年
プログラミングを教える授業の担当経験	あり	あり
テストベースの教育の担当経験	なし	あり
Gradle などのビルドツール	日常的に使用	日常的に使用
Jenkins などの CI ツール	使用経験なし	日常的に使用
PMD などの品質計測ツール	使用経験なし	何回か使用
Git などの版管理システム	日常的に使用	日常的に使用

5.3.2　実績システム

被験者 A からは学生に付加的な学習コンテンツを与えられる点や，客観的なプログラム品質の評価を行える点が評価された．被験者 A による評価を以下に要約する．

> プログラミングを教える授業では，学生の作成したプログラムがテストを通過したところまでは評価できるが，それ以上の付加価値を認めることは難しい．この実績システムを用いることで，意欲的な学生に付加的な学習コンテンツを与えモチベーション向上に繋げられる．教員としてはぜひ導入してみたい．
>
> ただし，難しすぎる実績が多すぎるとモチベーション向上には繋がらない．難易度調整は一つの課題であり，Ape がそれを克服しているわけではない．学生のレベルに応じて実績項目を変えるなどの工夫があれば，効果的な実績提示が可能かもしれない．

被験者 B からはプログラム品質を学生に学ばせる契機となる点が評価された．被験者 B による評価を以下に要約する．

> 授業内でテストだけを用意すると，テスト通過を一番の目標として開発を進めてしまい，品質を考慮しないことがある．実績達成によりどのようなメリットが得られるのか，出典をきちんと示しながら学生に提示すればプログラム品質について考えるきっかけになる．どのような実績を採用するか考える難しさはあるが，実績達成状況を学生にフィードバックすることは非常に有意義であると感じた．

このように実績システムは両被験者から良好な評価を得た．一方，実績の内容や難易度調整については熟考する必要があり，Ape でそれをサポートする必要がある．

5.3.3　Ape の導入

被験者 A は表 2 に示した通り CI ツールの使用経験がなかったが，Ape の導入は分かりやすいと評価した．被験者 A による評価を以下に要約する．

> Ape の導入に関して問題点は特になかった．Jenkins を使ったことは無かったが，流れは分かりやすかった．CI ツールを導入しているプログラミングの授業がどれほどあるかは分からないが，Jenkins を既に導入している授業であれば Ape 導入によるデメリットはほとんどない．Ape を導入することによって学生の動機付けに繋がるのであれば，教員が CI ツールを授業に導入するモチベーションにもなる．

被験者 B は Ape 導入に必要なファイルが実績設定ファイルのみであり，手軽に導入できる点を評価した．被験者 B による評価を以下に要約する．

> Ape の導入は間違いなく簡単で，実績設定ファイルをリポジトリに置いておくだけで使えるというのも非常に分かりやすい．必要なファイルも 1 つだけなので

煩わしさも感じない．

ただし，JsonGen については改善点がある．品質計測ツールに精通していない教員のことを考え，閾値と使用するルールセットを設定する際に参考となるデータが必要である．必要な設定ファイルは 1 つだけであり，その可搬性を活かして設定ファイルを収集および共有することで，各授業で使用している閾値とルールセットを記した事例集を作成してはどうか．

さらに，既に Jenkins を導入している授業が少ないので，現状では Ape を適用できる授業が限られている．テストベースの教育として学生にリポジトリとビルドツールが用意されているのであれば，そこに Jenkins のような CI ツールを導入するのは難しくない．そこをサポートできれば，Ape を導入することについては何のハードルもない．

このように両被験者から Ape 導入の簡潔さを評価された一方，閾値やルールセットの設定が難しいとの指摘も得た．これについては JsonGen を改善することで，教員により多くの情報を提示し実績項目を策定しやすくする必要がある．また Jenkins を既に導入している授業が少ないため適用対象が狭いことが指摘されているが，被験者 A からは Ape 導入により教員が新しく CI ツールを導入するモチベーションに繋がるという評価も得ている．CI ツール導入へのサポートを強化することで，適用対象は拡大できると考えられる．

5.3.4 Ape の使用

被験者 A は Ape が実績達成状況の画面で品質計測の結果を分かりやすく学生に提示できるため，学生の品質改善に対するモチベーション向上に繋がると評価した．被験者 A による評価を以下に要約する．

> プラグインでコンパクトに実績システムを実現できているため使いやすいと感じた．実績達成状況の画面も UI が明快で分かりやすい．品質計測が即座に行われ実績という形で表示されることで，計測結果をそのまま示すよりも品質改善への動機付けに繋がる．
>
> 閾値が画面上で確認できないためその点は改善が必要．さらに，他学生との比較を行えるとより良い．クラス内の何人が各実績を達成しているのか確認できれば，より実績を達成しようという気になるはず．

被験者 B による評価を以下に要約する．

> 実績達成状況の画面を見せることで品質改善へのモチベーションが向上するのであれば，使用する価値はある．ただし，実績達成状況の画面で閾値を学生に示していないのは気になる．API ドキュメントでも閾値を示していないため，設定ファイルから閾値の情報を取り込んで動的にドキュメントを生成するべき．実績達成状況の画面でアイコンにマウスオーバーすると，閾値が表示されるといった改善も考えられる．

実績達成状況の画面で閾値を学生に提示できていない点は今後の課題と言える．他学生との比較は Ave で実現できていたが，プラグイン化したことによって Jenkins に閉じた環境となり現状では実現できていない．しかし被験者 A に指摘された通り，他学生との比較は競争心をあおりモチベーション向上に繋がると考えられるため今後の課題とする．

6. おわりに

本研究では，既存研究で提案した実績システムを Jenkins プラグイン Ape として実装し，可搬性と拡張性を向上させた．評価実験として大学教員 2 名に Ape を使用してもらい，実績システムの有用性，Ape 導入の簡潔さおよび Ape 使用の有効性を確認した．本研究の今後の課題としては，JsonGen を改善することによって実績設定ファイルの作成をより容易にすることや，実績達成状況の画面を改善して使用感の向上および学生の更なるモチベーション向上を狙うことが考えられる．

謝辞　実験にご協力頂いた教員の方々に感謝する．本研究の一部は，日本学術振興会科学研究費補助金基盤研究（B）（課題番号：18H03222）の助成を得て行われた．

文　献

[1] P. Ihantola, T. Ahoniemi, V. Karavirta, and O. Seppälä, “Review of recent systems for automatic assessment of programming assignments,” Turkish Journal of Gastroenterology, vol.18, no.4, pp.265–267, 2007.

[2] K. Ala-Mutka, “A survey of automated assessment approaches for programming assignments,” Computer Science Education, vol.15, no.2, pp.83–102, 2005.

[3] J.A. Harris, E.S. Adams, and N.L. Harris, “Making program grading easier: but not totally automatic,” Journal of Computing Sciences in Colleges, vol.20, no.1, pp.248–261, 2004.

[4] G. Tremblay and E. Labonte, “Semi–automatic marking of java programs using junit,” Proc. International Conference on Education and Information Systems: Technologies and Applications, pp.42–47, 2003.

[5] J. Carter, J. English, K. Ala-Mutka, M. Dick, W. Fone, U. Fuller, and J. Sheard, “How shall we assess this?,” Proc. Working group reports from ITiCSE on Innovation and technology in computer science education, vol.35, pp.107–123, 2003.

[6] A. Kyrilov and D.C. Noelle, “Do students need detailed feedback on programming exercises and can automated assessment systems provide it?,” Journal of Computing Sciences in Colleges, vol.31, no.4, pp.115–121, 2016.

[7] S.H. Edwards, “Rethinking computer science education from a test–first perspective,” Proc. Conference on Object–Oriented Programming, Systems, Languages, and Applications, pp.148–155, 2003.

[8] J. Jansen, A. Oprescu, and M. Bruntink, “The impact of automated code quality feedback in programming education,” Proc. CEUR Workshop, vol.2070, pp.1–19, 2017.

[9] 華山魁生，柗本真佑，肥後芳樹，楠本真二，“プログラミング教育における実績制度を用いたコード品質可視化システムの試作，”電子情報通信学会技術研究報告，pp.145–150，2019.

[10] P. Tomas, M.J. Escalona, and M. Mejias, “Open source tools for measuring the internal quality of java software products. a survey,” Computer Standards and Interfaces, vol.36, no.1, pp.244–255, 2013.

[11] 井上明人，ゲーミフィケーション – <ゲーム>がビジネスを変える，NHK 出版，2012.