複数言語に対応しやすい オンラインプログラミング学習・試験システム track

新田章太 †，小西俊司 †，竹内郁雄 ‡

概要 プログラミング教育の普及に伴い，プログラミング学習環境の構築や改善への期待が高まっている。企業などにおけるプログラミング研修や，新しいプログラミング教育法を開拓している学校では，プログラミング言語の多様化や高度化，アップデートへの対応が必要である。このためには，少ないヒューマンリソースで，進化の激しいプログラミング技術領域における学習教材・試験の編集・改善ができるシステムが望まれる。本研究では，学校や企業におけるプログラミング教育に用いることのできるシステムとして track というオンラインプログラミング学習・試験配信システムを提案する。track は複数のプログラミング言語に対応しやすい実行環境や自動採点機能を搭載した，オンラインのプログラミング教材や問題を誰もが柔軟に作成・編集できるシステムである。本論文は track の研究課題と解決手法を記述し，プログラミング教育の新しい手法として提案する。

キーワード track，オンラインプログラミング教育, プログラミング演習, プログラミング試験，習熟度可視化, スキル可視化, スキル測定

1 はじめに

ビッグデータや人工知能（AI）活用の必要性が高まる中，文系・理系を問わずプログラミングの素養を身につけさせて，IT（情報技術）人材を育成する動きが活発化している。文部科学省は小学校の新学習指導要領において，2020 年からのプログラミング教育の必修化を発表した。大学入試センター試験に代わって導入される「大学入学共通テスト」の科目に，プログラミングや統計の情報科目を導入する検討が進んでいる。

多様な教育現場へのプログラミング教育の普及に伴い，プログラミング学習環境の改善が先行研究でも多く取り上げられてきた [1][2][3][4][5][6][7]。例えば，Bit Arrow に代表されるように，従来の学校におけるプログラミング授業を効率化するために，Web ブラウザを用いたオンライン学習環境の提供がある。これらの研究では，学習者が Web ブラウザ上で JavaScript や C のプログラムを書いて課題に解答できるエディタを用いて学習することで，効率的にプログラミング講習・研修を受けることが可能になった。教員も学習進捗状況の可視化のメリットを受けることができる。

しかし，現実的でアップツーデートなプログラミングを必要する企業におけるプログラミング研修や，新しいプログラミング教育法を開拓している学校では，Java, Ruby，Python など，プログラミング言語の多様化や高度化，言語改訂への対応や採点の自動化が必要になる。つまり，少ないヒューマンリソースで，進化の激しいプログラミング技術領域における学習教材・試験の編集・改善・自動採点ができるシステムが望まれる。

本研究では，学校のみならず企業におけるプログラミング教育に用いることのできる新たなシステムとして **track** という独自のオンラインプログラミング学習・試験配信システムを提案する。track は複数のプログラミング言語に対応した実行環境や自動採点機能を搭載したオンラインのプログラミング教材を誰もが柔軟に作成・編集できるシステムである。本論文は track が解決すべき課題，それを解決するための実装手法，すでに開始されている実践導入などについて述べる。

2 研究課題

この章では我々が課題としたものについて述べる。本論文では節番号で課題を参照する。

2.1 言語改訂への実行環境・教材の更新が面倒

Java は初心者向けのプログラミング教育用の言語としてよく使われる。Java は 1995 年にバージョン 1 が開発されたが，23 年経った現在はバージョン 12（JDK12）に進化している。これ以降も 6 カ月に一度のメジャーアップデートをしていくと ORACLE 社は発表した [8]。

このように，プログラミング言語をはじめとするソフトウェア開発技術は絶え間なく進化し続けているが，進

† 株式会社ギブリー

‡ 東京大学名誉教授，株式会社ギブリー技術顧問

化は API やライブラリが充実するだけに留まらない。例えば，JDK10 では「var を使った型推論」が大きな話題となった。それまで Java で変数を定義する際には，型を明確に指定する記法

```
int num = 1;
String name = "Nitta";
```

を使っていたが，データ型を宣言せず，代入する値でデータ型を自動的に決定してくれる型推論による記法

```
var num = 1;
var name = "Nitta";
```

が導入され，プログラマの間で議論を巻き起こした [9]。

言語の進化はただ機能の充実により便利になるだけではなく，それによってこれまで標準とされていた記法が変わることがある。教材として管理するためには，半年スパンで発生するこのような変化に対応し続けなければならない。

また，教材を更新するだけではなく，受講者の PC すべてで新しいバージョンに対応した実行環境を用意する作業，あるいはネットワーク越しの多様な受講者に対応することが必要である。

2.2 プログラミング言語の多様化への対応

プログラミング言語は進化するだけではなく，多様化も進んでいる。プログラミング言語はそれぞれのコミュニティや管理者を通じて更新されているが，応用分野の変化に応じた新しい技術トレンドが生まれている。

開発者がオープンソースプロジェクトのソースコードを管理する上でよく利用している GitHub が公開した，プログラミング言語別の人気ランキングの 2008 年から 2015 年まで変遷 [10] を見ると，Java はこの 7 年間で著しく人気が増している一方で，授業で利用される定番の C は人気が減少している。また，近年の機械学習や AI のブームを受け，Stack Overflow に登録されている質問閲覧数では Python に注目が急激に集まっている [11]。

このように，社会の現場で利用されている，あるいは注目度や必要性が増加するであろうプログラミング言語を教育に採択しようとすると，時代のトレンドに応じてプログラミング言語を選択する必要がある。教員がこれらをすべてを掌握し，時宜にかなった教育内容を作成するのには多大な工数がかかる。

2.3 採点にかかるヒューマンリソース

これまでのプログラミング授業における演習や試験では，受講者が PC で書いたプログラムを，メールや，授業管理システム，USB などを通じて提出し，出題者あるいは TA がそれらを個別に確認・採点していた。これには莫大な工数と正確性が必要である。

2.4 集合型学習スタイルにおける課題

企業研修の場においては，プログラミング経験の有無にかかわらず，集合的に研修を実施することが多い。しかし，学業でのプログラミング経験がある受講者にとって，一からの研修は退屈なものと感じやすい。習熟度のばらつきがある個々人の能力を把握した上で，能力に最適化した個別の学習環境が求められている。

3 track の概要

3.1 track 開発の経緯

ギブリー（Givery）では 2013 年 8 月からオンラインでプログラムを書いて動かしながら学ぶことができる実践型のプログラミング学習サービス CODEPREP（コードプレップ）を開発し，2014 年 8 月に無料公開した。現在 8 万人が登録する国内有数のオンラインプログラミング学習サービスとして一般に広く受講者がついている。今でも教材をクリアした多くの受講者が気軽に Twitter を始めとした SNS で発信をしている。

しかし，公開している教材を言語のバージョンに合わせて更新する必要や，多言語の実行環境をサポートするために莫大なコストと時間を要することが問題となり，それらを簡単に管理する仕組みが求められた。

そこで，複数言語の実行環境に対応し，教材を簡単に作成・変更ができるエンジンを開発した。この仕組みを用いて，企業が採用や研修時に受験者のプログラミングスキルを把握するためのスキルチェックツールとして，track の前身となる codecheck（コードチェック）を 2015 年 10 月に公開し，50 社近くの企業へ展開してきた。

前述のプログラミング学習における課題を解決し，これまで両軸で展開していた 2 つのサービスを，一つのプラットフォームとして統合開発したのが track である。2018 年 3 月に初版を公開した。

3.2 track が扱えるコンテンツの種類

track は，上に述べたように，複数のプログラミング言語に対応した実行環境や自動採点機能を搭載したオンラインのプログラミング教材を，誰もが柔軟に作成・編集できるシステムである。

授業の管理者は，専用のアプリケーションとテンプレートを用いることで，複数の言語に対応したプログラミング教材を適切な形式で作成したり，更新したりすることが容易にできる。track が扱える**コンテンツ**，すなわち問題や教材は以下のように分類できる。

チャレンジ（問題）は，特定の制限時間の中で解答をすることで点数化することができる問題である。授業後

の演習に活用できる。チャレンジを複数個束ねたものが**試験**で，期末試験などに活用できる。

チャレンジには，コーディング形式と選択式・穴埋め形式がある。

コーディング形式は，与えられた要件を満たすプログラムを書く問題である。受講者が解答に利用したいプログラミング言語を選択すると，言語毎の基本テンプレートが与えられる。問題にはユニットテストを用いた，満たさなければならない要件がテストコードとして指定されており，受講者は要件を満たすプログラムを書くことで問題に答える。一つのユニットテストを多言語にそのまま適用できるのが，track の特徴である。

選択式・穴埋め形式は，ラジオボタンやチェックボックスで 4 択の問題に解答する，あるいは空欄を文字や文字列で穴埋めすることで，正誤が判定される形式の問題である。

教材（ブック）は，受講者が複数の章と節から構成される演習問題集を，問題文を読みながら解答していく。プログラムを書いて実行しながら体系的にプログラミング言語に習熟するのに適している。教材を複数個，系統的に束ねたものを**コース**と呼ぶ。

教材はドリル教材と演習教材に分かれる。

ドリル（選択式・穴埋め）教材は，問題文とプログラムの一部が空欄で与えられ，適切な解答で穴埋めすることで，プログラムが実行され，次に進むことができる形式の教材である。節の中に文章や図のみで構成されている補足テキストを追加して，リーディング教材のようにすることも可能である。

演習（実装）教材は，問題文とプログラムが与えられ，適切にプログラムを編集して実行し，すべてのテストケースを通すことで，次に進むことができる形式の教材である。

3.3 track の 3 つの主要機能

前節のような教材やチャレンジを作成し，コースや試験を合成するほかに，受講者がそれらを享受できるようにすること，また採点やレポート作成をすることも重要な機能である。

(1) コンテンツ作成機能

track のブック（教材）やチャレンジ（問題）は，GitHub 上に管理されている形式ごとにまとめられているテンプレートをベースに，テキストや問題文，模範解答コード（コーディングテストの場合），テストコード，詳細設定を編集して作成する。作成したら，管理アプリケーションから必要項目を指定して教材や問題として登録する。

登録された教材と問題は，管理アプリケーションから一覧することができ，それらを組み合わせて，目的に合わせたコースや試験を作成できる。試験には，提出締切や問題それぞれの制限時間を設定することができる。

(2) コース受講・受験機能

作成したコースや試験は受講者のメールアドレス宛に配信するか，専用の URL を受講者がアクセスすることで届ける。受講者はそれぞれの方法によって受け取った URL に Web ブラウザからアクセスして，コースや試験を受ける。

試験の場合，受講者は Web ブラウザエディタを利用して提出締切までに問題を解く。コーディング形式の問題の解答中は，プログラムを実行してみて，正しく実装ができているかを確かめ，エラーコードを読みながら修正することができる。編集が終わったら，アプリケーション上の提出ボタンを押して，解答を提出する。このとき，受講者が利用したいプログラミング言語を選択して解答できるのが track の特徴である。

試験以外では，受講者と track のインタラクションはもっと単純である。

(3) 採点・レポート機能

コースの進捗や，提出された解答と採点は，全体のレポートと，受講者ごとの個別レポートとしてまとめられる。解答データや点数を可視化することで，企業研修の効果測定や授業の習熟度チェック，受講者の評価に利用することができる。詳細に関しては本論文では割愛する。

4 track の実装技術

本章では，課題を実装面でどのように解決したかについて述べる。コンテンツを作るための情報を含む。

4.1 track コア

track を構成する様々なコンポーネントのうち，

- コンパイラ (trackc)
- 専用エディタ
- コード実行サーバ (orca)

をまとめて，特に track コアと呼ぶ（図 4.1）。これは 2 章で述べた課題を解決するためのアイデアを実現するためのフレームワークである。

(1) コンパイラ (**trackc**)

trackc はコンテンツのソースとなるテキストをエディタで実行可能な形式に変換するモジュールである。track におけるコンテンツのソースはチャレンジ（問題），ブック（教材）ともにテキスト形式である。マークダウンから参照する画像等を含めることができる。

コンテンツ作成は，課題 2.2 に対処するため，テキストエディタのみで可能とし，trackc を通してエディタで実行可能な形式にパッケージングされるようにした。

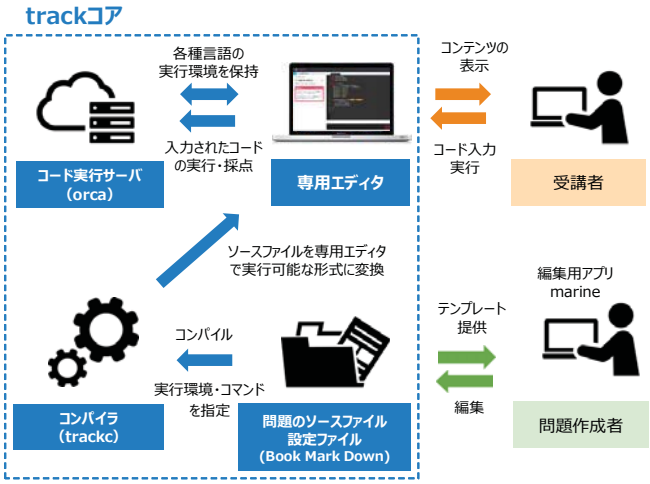

図 4.1 track コア

(2) 専用エディタ

専用エディタは trackc が生成した問題をブラウザ上で表示し，実行するフロントエンドアプリケーションである（図 4.2)。このモジュールには

- 問題がどのように供給されるか？
- 受講者の解答がどのように管理されるか？

という機能は含まれておらず，純粋に問題を表示・実行する機能だけを受け持つ。専用エディタ上で編集したソースコードはコード実行サーバに送られ，その実行結果（標準出力）が画面上に表示される。通常，専用エディタは外側のシステムから iframe として実行され，問題供給を行うのは外側のシステムの役割となる。

(3) コード実行サーバ (**orca**)

orca は受講者が記述したプログラムを実際に実行する，バックエンドサーバの一つである。orca は

- 受講者の記述したソースコード
- 問題作成者が提供するユニットテストで使用するファイル群
- 実行するコマンド

を受け取り，その都度 Docker コンテナを起動してその中でコマンドを実行する。コマンドはコンテナ内で実行されるので，仮に受講者が悪意のあるコードや不用意な無限ループを実行したとしてもその影響範囲はコンテナ内のみである。また，Docker イメージを追加するだけで容易に新しい言語をサポートすることが可能となっている。これは課題 2.1 および課題 2.2 を解決するものである。

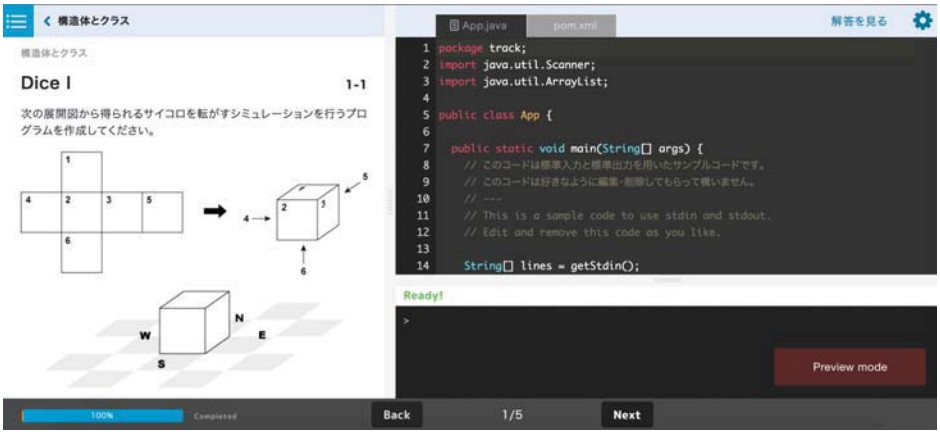

図 4.2 専用エディタの画面。左が問題。右上にコードを書き，その下に結果が表示される。

4.2 track

track は実は track コア上のアプリケーションである。つまり，track コアに

- コンテンツ作成者の作成したチャレンジやブックを管理する機能
- 受講者の解答コードを管理する機能
- 受講者の解答コードを orca で実行し，その採点を管理する機能

などを加えたものである。原則，track コアの変更は track 本体には影響を与えない。これは逆に言えば track コアを利用した別のアプリケーションが作成可能であることを意味する。その例には，track の前に開発され，実用に供された codecheck がある。

(1) ローカル環境での受験 (**track CLI**)

track でのチャレンジ，ブックの解答は通常，track コ

アの専用エディタ上で行うが，コーディング問題に関しては track CLI というコマンドラインアプリケーションを使用することで，受講者のローカル環境で使い慣れたエディタを利用して解答することも可能である（図 4.3）。

これを実現するのが track CLI である。track CLI はコマンドラインアプリケーションで，Windows，Mac，Linux のすべての環境で動作する。これは 2.1 節の最後の課題に対応している。

track CLI は track のバックエンドサーバ及び orca と通信し，以下を行う。

- 解答用テンプレートなどの取得（問題文自体は二次利用を防ぐためダウンロードしない）
- 受講者の解答コードの orca での実行
- 受講者の解答の提出

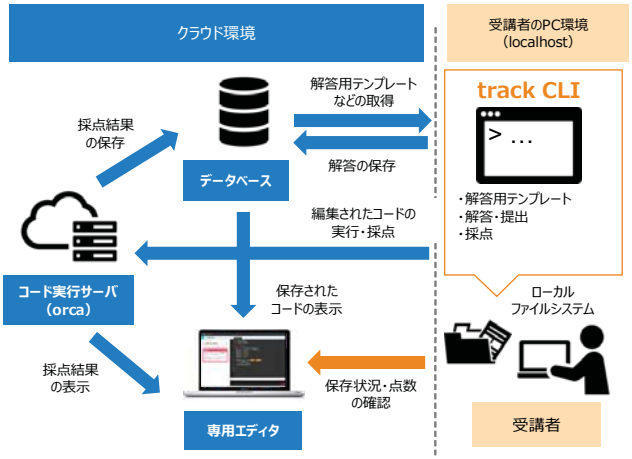

図 4.3 ローカル環境での受験

(2) 採点モジュール

これは課題 2.3 に対応するモジュールである。チャレンジのユニットテストは専用エディタ，あるいは track CLI から実行される以外に track 本体の採点モジュールからも実行される。この際に実行されるユニットテストには受講者に不可視のものも含まれる。

つまり，受講者視点からは 10 個のユニットテストがあって，そのすべてにパスしているように見えても，実はその裏に隠されたエッジケースのユニットテストが存在して，そこで採点に差がつくことがあり得る。

(3) **marine**

marine はコンテンツ作成者に提供されるチャレンジ，ブック作成用アプリケーションである。

marine は Docker イメージとして提供されており，コンテンツ作成者のローカル環境で Docker を使用して実行する。Docker コンテナには trackc と専用エディタを含む（図 4.4）。コンテンツ用のディレクトリは，ホスト側のディレクトリとリンクしておりコンテンツ作成者はローカル環境上でのコンテンツの編集が可能となっている。

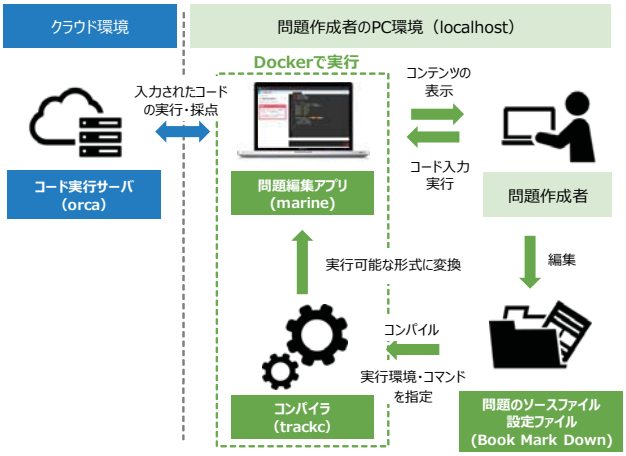

図 4.4 問題作成の仕組み

marine では

- コンテンツディレクトリから一連のソースファイルを読み込み，
- それを trackc でコンパイルして，
- コンパイル結果を専用エディタで表示

としているので，コンテンツ作成者は実際の画面での表示を確認しながら，コンテンツを修正できる。専用エディタ上で変更した内容を orca で実行することも可能だが，編集したコード自体はどこにも保存されない。これはコンテンツ作成での実験を容易にする。

なお，trackc とバックエンドは Scala，実行サーバは Go，track CLI は Rust で実装した。

4.3 採点の仕組み

採点モジュールについて少し詳しく述べる。

(1) ユニットテスト

チャレンジの採点は基本的にはユニットテストによって行われる。このため問題作成者がチャレンジ作成時に行う最も重要な作業はユニットテストの作成である。

ユニットテストは

- Java → JUnit
- Ruby → RSpec
- Node.js → mocha

のように，それぞれの言語における代表的なフレームワークを使用して記述することができる。それぞれのテ

ストフレームワークの出力をTAP形式[1]に揃えることによってユニットテストの成否を透過的にアプリケーションが取得できる。これは課題2.2で述べた多言語対応のための重要な手段である。

ユニットテストをベースとした試験方式はTDD (Test Driven Development) の考え方を応用したもので，実際の業務との親和性が高い。

(2) CLIによる言語非依存のユニットテスト

課題2.2，および4.2節 (1) にも関連するが，各種言語のテストフレームワークを用いてユニットテストを記述する方式には，受験可能な言語が限定されるというデメリットがある。例えばRSpecでユニットテストを書いた場合は受講者はRuby以外の言語で解答を提出することができない。しかし，分割統治や動的計画のようなアルゴリズムを問うような問題の場合は受講者が任意の言語を選択できることが望ましい。これを実現するために標準入出力をハンドルする独自のテストフレームワークを用意した。

このフレームワークを用いた場合，

- 受講者は，標準入力からパラメータを受け取り，標準出力に結果を出力するCLIアプリケーションを作成する
  - 使用する言語は何でもよい
  - 言語ごとに標準入出力を扱うテンプレートが用意されているので，受講者はアルゴリズムの実装に集中できる
- 問題作成者は入力パラメータと対応する出力値を用意することでテストケースとする
  - あるいは，専用判定アプリケーションを作成することもできる
  - フレームワークはNode.jsで実装されており，mocha上で動作するためカスタマイズが可能

となる。この方式で作成されたチャレンジをアルゴリズムチャレンジと呼ぶ。

4.4 システムの拡張性

4.1節 (3) でも述べたが，orcaはDockerをベースに構築されている。このため，Dockerイメージさえ準備すれば様々な拡張が可能となる。ここではその詳細を述べる。

(1) 対象言語の新規追加

trackは現時点でHaskell，Rustなどの言語をサポートしていない。これらを追加するための作業は，例えば

- Haskell用のDockerイメージを用意してコード実行サーバに含める
- アルゴリズムチャレンジ用のテンプレートを用意する

だけである。そのためサポート言語が次々と追加できる。

(2) 対象言語・フレームワークのバージョンアップ

2.1節でも述べたように，Javaは半年に一度，Rubyはほぼ年に一度，言語自体のバージョンアップが行われている。他の言語もそれに近い周期でバージョンアップがされることが一般的である。また，mochaやRSpecのようなテストフレームワーク自体がバージョンアップされることもある。

これに対応する方法は，対応するDockerイメージの差し替えである。trackにおいて受講者が書くコード量はそれほど大きくないので，互換性を重視する言語であれば，旧バージョンを新しいバージョンに置き換えてもほとんど問題にならない。ただし，厳密に過去の実行環境を残す必要がある場合は，置き換えではなく新しいDockerイメージの追加で対応することになる。

(3) 言語固有のライブラリの使用

Ruby用のDockerイメージにはbundlerがプリインストールされており，Python用のイメージにはpipがプリインストールされている。これらを使用して受講者が自分の使いたいライブラリをインストールすることが可能であり，それらは受験をしている間，コンテナ中で維持される。

また，別のパッケージ管理ツールや特殊なパッケージをプリインストールしたDockerイメージを作成して使用することも可能である。

4.5 コンテンツの作成

4.2節 (3) marineについて少し詳しく述べる。

(1) 各種言語でのチャレンジの作成

trackのチャレンジは以下のように進む。

- 問題作成者がユニットテストを書く
- 受講者はそのユニットテストにパスするコードを解答として記述する

問題作成者の主たる作業は，ユニットテストを作成することと，問題文 (README.md) を記述することである。受講者の使用する言語が固定であるならその言語で一般的なテストフレームワークを使うことができる。例えば，Rubyなら RSpec, Javaなら JUnitなど。

[1] https://testanything.org/

設定ファイルにはファイルの配置，実行コマンド，ユニットテストの公開/非公開などを記述する。

(2) アルゴリズムチャレンジの作成

アルゴリズムチャレンジのユニットテストは，受講者が自分の得意な言語を使用して作成した CLI アプリケーションの入出力を検証する。アルゴリズムチャレンジ用のテストフレームワークは標準で用意されている。問題作成者の主たる作業はユニットテスト用の入出力パターンを用意することと，問題文 (README.md) を記述することである。入力に対する出力が一意に決まらない問題に対応するためにカスタム判定アプリケーションを作成することも可能である。設定ファイルにはファイルの配置や実行コマンドを記述する。

(3) ブックの作成

ブックはマークダウンを拡張した専用のブック記述言語 (BMD, Book Mark Down) で記述する。専用言語といっても，マークダウンの記述時にいくつかのルールを決めているだけで特別な知識は必要としない。

4.6 コンテンツ作成の容易性

前述したように，コンテンツのソースはすべてテキストファイルであるため，作成者は使い慣れたエディタを使用してコンテンツを作成することができる。

初期は，少し修正しては marine で表示を確認，というコンテキストスイッチを繰り返すことになるが，このサイクルは慣れるに従って長くなり，作業がスムーズになる。また，テキストベースであるため GitHub 等の既存の VCS（Version Control System）でソースを管理できることもメリットである。

以下に Java のブックのサンプルを示す。`${…}` の部分がエディタ上では穴埋めとなり，受講者が解答を埋めた際には自動的に orca でそのコードが実行される。

````
## 標準出力に"Hello World"を出力する
標準出力に文字列"Hello World"を出力してみましょう。

標準出力への出力には`System.out.println`というメソッドを
使用します。

### main(Main.java)
```
public class Main {

  public static void main(String[] args) {
    System.out.${println}("Hello World");
  }
}
```

### remote

- build:  javac Main.java
- command:  java Main
````

アプリケーションの作成法を解説するようなブックでは，前節の内容を少しだけ変更したり，付け足したりするという内容のページが多く発生する。こうしたブックの作成を容易にするための組み込み関数をブック記述言語 BMD には多数用意した。

チャレンジ作成者はユニットテストを記述することになるが，正答コードが実際にすべてのユニットテストをパスするかを検証する仕組みが marine に組み込まれている。

4.7 Bit arrow との比較

track は汎用的な track コアをベースに開発されているため，任意の言語に対して実行可能な環境を用意し，コマンドラインアプリケーションを生成したり，テスト実行，評価することがスムーズに行える。

track は，実行環境をサポートしているバックエンドのプログラミング言語が，C，C++，C#，Node.js，Java，Perl，Ruby，Go，Python2，3 系，Scala，Kotlin，Swift（最後の 2 つは 7 月に追加）と Bit Arrow よりも多く，リクエストがあれば簡単に増やしていける。実際，Kotlin，Swift はそれぞれ正味 1 人週で追加できた。

また，track では，フロントエンドで使用される HTML と Javascript に加え，CSS 言語のブックを用意している。このようなフロントエンド言語の場合は受講者の記述が即時に専用エディタ上のプレビューに反映される。

5 研修における実践導入事例

4 月から，セゾン情報システムズの新入社員 25 名を対象に，track を用いて研修前のスキルチェックによる習熟度の把握，及びオンライン学習の提供による習熟度の差異に対応したセルフラーニング環境を提供し，研修の効果検証を実施した（写真 5.1）。

5.1 研修でのプログラミングスキル測定

まず，研修が始まる前に track を用いて IT 基礎知識問題，プログラミング読解問題，データベース実装（SQL 記述）問題，アルゴリズム実装問題を試験として配信し，研修受講者のスキルアセスメントを実施した。習熟度の高いメンバーと低いメンバーでペアを組み，相談しながらプログラムを学んでいくなど，カリキュラムの調整を実施した。課題 2.4 については当面運用で解決するが，自動化を将来の課題としたい。

5.2 習熟度に応じた学習環境

また，集合研修的な講義以外に，受講者にはプログラミング教養，プログラミング基礎，アルゴリズム基

礎，Java 基礎，SQL 基礎，Web フロントエンド基礎（HTML/CSS/JavaScript）などのオンライン教材を配布し，それぞれが自分のペースで自習できる環境を提供した。

5.3 簡単な考察

まだ，過去の研修との習熟度向上の差異，効果検証をするための対照実験にまでは至れていないが，研修半ばに実施した試験では，全体の平均点が 1 カ月の間で，33 点から 51 点まで向上した（実践経験の少ない受講者が多かった）。受験者のスキルを把握した上での学習カリキュラムを提供することで，効果的なプログラミング研修カリキュラムを提供できたのではないかと考える。これらについては別途報告する。

写真 5.1 研修の様子

6 終わりに

本論文では現在のプログラミング教育環境における課題の定義から，多言語や言語アップデートに対応しやすい track のシステム構成，及びシステムを研修に応用した実践導入について述べた。 今後，今回の実験における受講者の主観的な満足度調査や，導入前後での能力向上を検証することで，オンラインでのプログラミング学習環境の提供及び試験によるスキルの可視化がエンジニア育成に効果的であることを確認していきたい。

また，現在システム及び問題の英訳を進めている。国際的な受講者のプログラミング問題解答データも収集・分析して，多様な解答データからプログラミング能力に関するソースコードメトリクスを提案したい。解答データを活用したこのような研究を深め，エンジニアのよりよい評価・育成につなげていく。

謝辞 本研究は株式会社セゾン情報システムズに実践導入していただくとともに被験者の解答データを提供していただいた。また，会津大学渡部有隆上級准教授から，Aizu Online Judge（AOJ）に掲載しているオンライン学習教材を提供していただいた。

[参考文献]

[1] 井垣 宏，齊藤 俊，井上亮文，中村亮太，楠本真二: プログラミング演習における進捗状況把握のためのコーディング過程可視化システム C3PV の提案，情報処理学会論文誌 54(1), 330-339, 2013-01-15.

[2] 長島和平，本多佑希，長 慎也，間辺広樹，兼宗 進，並木美太郎: オンラインで複数言語を扱うことができるプログラミング授業支援環境，情報教育シンポジウム 2016 論文集 2016, 137-140, 2016-08-15.

[3] 長島和平，長 慎也，間辺広樹，兼宗 進，並木美太郎: Web ブラウザを用いたプログラミング学習支援環境 Bit Arrow の設計と評価，研究報告コンピュータと教育（CE），2017-CE-138, 2, 1-8, 2017-02-04.

[4] 間辺広樹，長島和平，並木 美太郎，長 慎也，兼宗 進: オンラインプログラミング学習環境 BitArrow を用いた JavaScript の授業実践報告，研究報告コンピュータと教育（CE），2017-CE-138, 3, 1-8, 2017-02-04.

[5] 齋藤宏太郎，豊田哲也，大原剛三: オンラインプログラミング学習システムのための適応型出題モデルの提案，第 79 回全国大会講演論文集，2017, 1, 685-686, 2017-03-16.

[6] 間辺広樹，長島和平，並木美太郎，長 慎也，兼宗 進: 自宅で行うオリジナル作品制作の学習効果と問題点 〜オンラインプログラミング学習環境を用いて〜 情報教育シンポジウム論文集，2017, 15, 101-109, 2017-08-10.

[7] 岩田麻暉，長島和平，長 慎也，兼宗 進，並木美太郎: プログラミング学習環境 Bit Arrow における JavaScript によるデータベース API の設計と実装，研究報告コンピュータと教育（CE），2018-CE-143, 3, 1-9, 2018-02-10.

[8] Donald Smith: Faster and Easier Use and Redistribution of Java SE, https://blogs.oracle.com/java-platform-group/faster-and-easier-use-and-redistribution-of-java-se

[9] Raoul-Gabriel Urma and Richard Warburton: Java 10 Local Variable Type Inference, https://developer.oracle.com/java/jdk-10-local-variable-type-inference

[10] Alyson La: Language Trends on GitHub, https://github.blog/2015-08-19-language-trends-on-github/

[11] David Robinson: The Incredible Growth of Python, https://stackoverflow.blog/2017/09/06/incredible-growth-python/