programming 学習初期から理想的な開発様式を提供する仕組みについて

石川高行*1
Email: ishikawa+pcc2024@oiu.jp

*1：大阪国際大学

◎Key Words　　programming，入門者教育，開発環境

1. はじめに

programming の現場では，各種 test に pass していること，code の documentation が一応揃っていること，といった状態を保つための tools がよく利用されるが，programming 教育の場ではこういった tools は殆ど利用されてなく，このことが却って学習者の負担を増やしている側面があるものと思われる。

本発表では，Ruby を例に programming 教育の初期からこうした tools を取り入れる指導とそのための実装を提案する。

2. 単一 file による方法と Bundler による方法

2.1 coding 対象 files

殆どの入門者向け書籍では，coding の最初に file を 1 つだけ用意している。例えば『たのしい開発 スタートアップ Ruby』なら hello.rb, Progate なら index.rb という単一 file のみを用意し，そこに Ruby code を書かせている。

しかし，実際の Ruby programming では後の RubyGem 化を見据えて Bundler による雛型を利用することが殆どである。具体的には，例えば xyz という program を作ろうとするなら `bundle gem xyz` という命令を書くだけで coding を開始することができる。

前者の方法は，coding 対象 file が 1 つしか存在しないという点で確かに分かりやすいかも知れない。しかし，いずれ学習が進んで世の中に存在する Ruby script files (Bundler による雛型から書き始めた files) を見た際にはどこから理解すればよいのかが大変分かりにくい。

後者の方法は，Bundler による雛型に最初から 50 個近くの files が含まれているため一見して分かりにくく感じられるかも知れない。しかしこの点は，教師から学習者へ「大量の files が自動生成されますが，編集すべきは○○という file だけです」と説明すれば済むことである。

2.2 中級者向けの書籍の不足

後述する RuboCop, RSpec, YARD という tools は coding 上で大変便利であるが，Ruby を学習中の中級者がこれらの tools を学ぼうとしても適切な書籍は殆ど存在しない (上記のうち RSpec だけは少々存在する)。結果として中級者はこれらの知識を書籍ではなく web 上で入手しなければならない。書籍と異なり，web 上の情報は誤っていることが多く[1]，誤っている情報に接した中級者は正しい知識を入手するために時間を浪費することとなる。

それならば，Ruby programming 教育の最初の段階から RuboCop, RSpec, YARD が揃っている雛型を用意し，その上で編集対象 file を 1 つだけに制限すればよい。編集対象 file が 1 つだけであれば初学者が迷うことはないし，いずれ RuboCop, RSpec, YARD を使う段階になれば雛型の記述を参考にすることができる。

3. RuboCop

RuboCop[2] は Ruby で最もよく使われる linter/formatter である。

複数人で coding する際に coding style を統一する，という目的で使用されることが多いが，実際には 1 人だけで coding する場面でも大変役立つ。

更には，初学者がでたらめな indent で coding していても RuboCop に整形を任せれば理想的な indent になった状態で出力してくれる。

RuboCop のような formatter がなかった時代は理想的な indent で coding をする技術は programmer に必須の技能であったが，formatter が充実している現在，programming 初学者は indent を完璧にすることを意識せずにその他のことに焦点を当てて学習することができるだろう。

programming 学習者は，RuboCop の詳細を知らなくても「RuboCop を実行して no offences detected と表示される状態が望ましい」ということを知っておくだけで code 整形に意識が向くものと思われる。また `-autocorrect` や `-autocorrect-all` といった options をつけて RuboCop に整形を任せる方法を知れば，ある程度整形されている状態を保ちやすくなる。

ただ残念ながら，Bundler が生成したばかりの雛型でもある。

[1] 例えば、「YARD タグ一覧」という検索語でかつて検索上位に表示されていた site には、実在しない YARD tag が未だに表示されたままである。

[2] https://rubocop.org/

初期設定の RuboCop からいくつかの指摘を受ける[3]ので，初学者に提示するには Bundler が生成しただけの雛型よりも RuboCop による整形済みのものが適しているだろう。

なお，RuboCop の整形項目には初学者に適さないものもある。例えば 1 つの method での行数制限[4]などは refactoring に属することであり，学習の初期段階では気にする必要がないと考える。

4. RSpec

RSpec[5] は恐らく Ruby で最もよく使われている test 環境である。

RSpec で大切なことは全ての tests に pass する状態を維持することであり，初学者に分かりやすく表現すれば RSpec 実行時に赤字が表示されないことである。

RSpec の tests (spec files) を書く技能を身につけるにはそれなりの学習量が必要であるが，RSpec を実行して赤字が表示されないことを確認する方法は簡単に身につけることができる。

教育の場では，教師が tests (spec files) を用意し，学習者はその tests (spec files) に pass する program を書く，という手法もあるだろう。RSpec を実行しても赤字が表示されないことは学習者にとってかなりの爽快感を得られることであり，学習の動機づけとなることが期待できる。

5. YARD

YARD[6] は恐らく Ruby で最もよく使われている documentation tool である。RuboCop や RSpec と異なり，文書化率が 100% でなくても大きな問題とはされない。それでも，存在する classes くらいは説明がなるべく 100% 付与されていることが望ましいと言えるだろう。

code の文書化は，実際に coding している最中はそこそこやっかいな作業である。「新しい class を作ろう」「新しい method を作ろう」と思った場合，その文書を書くよりも先に coding したくなるが，文書化を後回しにすると文書化率が低いままとなることがよくある。

また programming 教育の場であれば作るべき classes や methods の指示は教師から学習者へ与えられていることも多いだろう。この場合，学習者には classes や methods を文書化する動機が希薄となるので，coding はされていないが文書化は済んでいる雛型を教師から学習者へ渡すことが望ましいと言えるのではないか。

6. .gemspec

実は，Bundler が生成する雛型に含まれている `.gemspec` file はそのままでは利用できない。例えば RubyGem としての Ruby version と RuboCop としての Ruby Version を合わせないといけない。しかし，`.gemspec` file の編集は初学者には難しいので，望ましいものを教師側で用意する必要がある。

7. 理想的な雛型

以上の点から，Ruby programming 教育では教師から学習者へ以下の点を満たす雛型を渡すことが重要であると考える。

- Bundler が自動生成した雛型が元であり，`gem build` によって RubyGem を容易に生成できる
- RuboCop parameters は学習向けに調整済みである
- RuboCop によって offences が検出されない (RuboCop 整形済みである)
- YARD による文書化率が 100% である
- 以上のことを簡易に実行できる Rakefile を備えている
- 学習者が手作業で `.gemspec` file を編集する必要がない

ここまでの考察に従い，上記全てを満たす雛型を生成する仕組みを作成した。具体的には下記のような YAML file を与えることで条件を満たす雛型を生成するものである。詳細は発表時に実演する。

```
name: stmanage
summary: 学生管理
description: 学生を管理する system
classes:
  - name: student
    docstring: 学生を表す。
    methods:
      - name: insert
        docstring: 学生を追加する。
  - name: teacher
    docstring: 教員を表す。
```

長くなるためここには記していないが，必要事項を追記することで RSpec の files も自動生成する。学習者は RSpec を実行して赤字が出なくなれば課題を達成した状態であると判断することができる。

また，RubyGem 化する全ての要素を備えているため，この雛型に慣れることで学習者は他の Ruby program の構造を理解しやすくなることが期待できる。

8. 終わりに

本発表では Ruby を例として取り上げたが，こうした programming 学習に必要な要素を持つ雛型の提供は他言語でも必要であろう。とりわけ，RSpec のような test 環境による課題達成判断材料を教師から学習者へ提供することは，学習者の学習動機づけに役立つと考える。

[3] 原稿執筆時点で Bundler 2.5.14, RuboCop 1.64.1 である。
[4] `Metrics/MethodLength`。
[5] `https://rspec.info/`
[6] `https://yardoc.org/`