ソフトウェアテストを意識したプログラミング教育

しょうなん
千葉県立沼南高等学校
沼崎　拓也

勤務校の概要

千葉県立沼南高等学校

- 柏市と我孫子市にある手賀沼の近く
- 昭和54年4月開校の全日制普通科高校
- ピーク時は10学級規模だが、現在は4学級
- キャッチコピー

「わからない」を「わかる」に！
そして「できる」に！！

背景

2022年　現行の学習指導要領がスタート

本校では
1年:「社会と情報」(必履修)
3年:「情報の科学」(4コース中2コースで必履修)

そのまま情報Ⅰ、情報Ⅱとなった教育課程

勤務校の生徒に向けて

「情報Ⅱ」 寄りの 「情報Ⅰ」

あるいは

「情報Ⅰ」 寄りの 「情報Ⅱ」

などの内容に取り組みたい

情報Ⅱで扱う内容について

情報社会の進展と課題

コミュニケーションと情報デザイン

情報システムとプログラミング

情報通信ネットワークとデータの活用

学習指導要領

(4)情報システムとプログラミング

ア 次のような知識及び技能を身に付けること。

(イ)情報システムの設計を表記する方法, 設計, 実装, テスト, 運用等のソフトウェア開発のプロセスとプロジェクト・マネジメントについて理解すること。

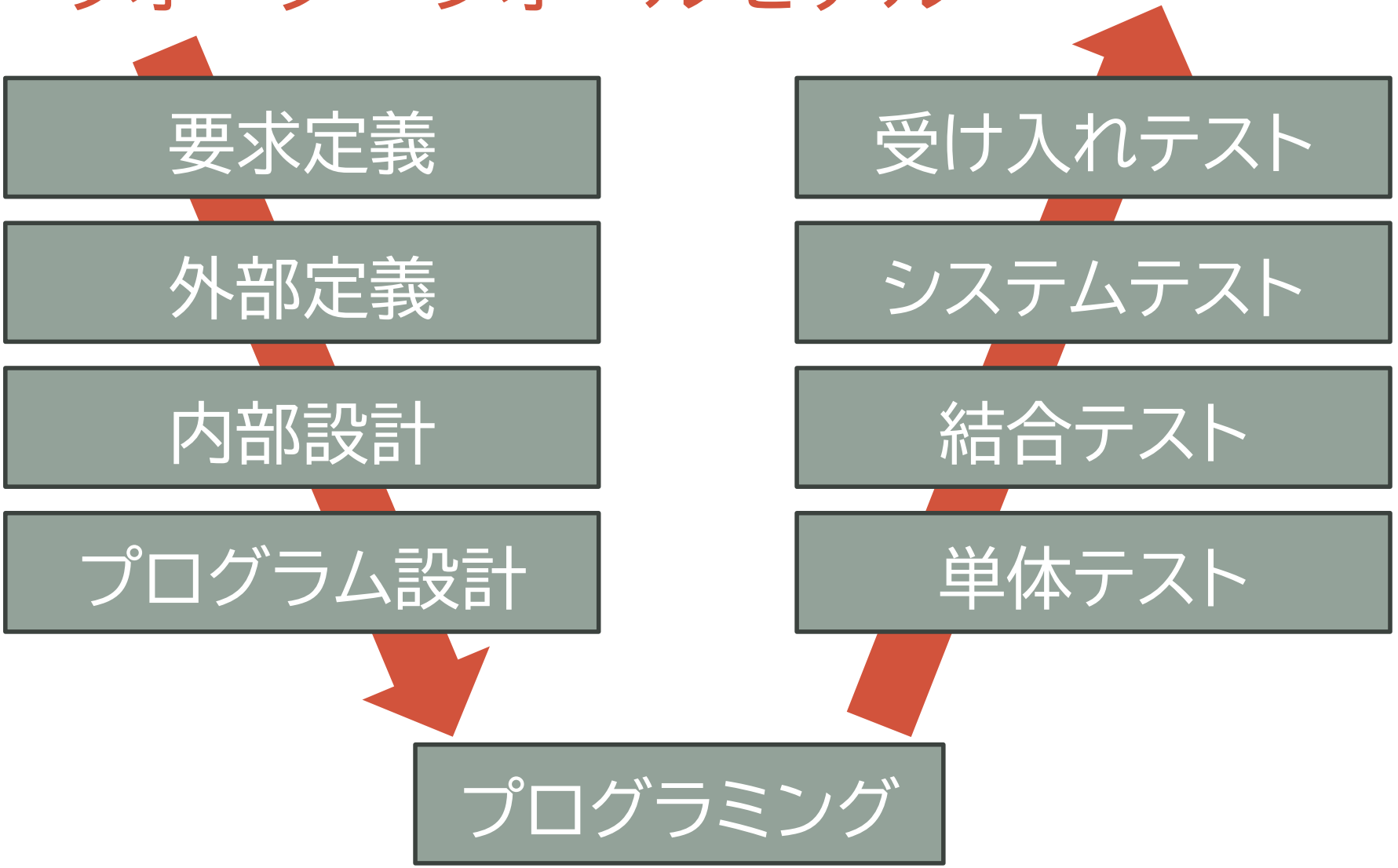
ソフト開発におけるテストの位置づけ

開発プロセス(要求定義～テスト)
「要求定義」
「外部設計」
「内部設計」
「プログラム設計」
「プログラミング」
「テスト」
と進めていく。

情報Ⅱ教科書におけるテストの記述

項目	取り扱われる語句
テストの説明	単体テスト, 結合テスト, システムテスト
単体テスト	命令網羅, 分岐網羅
結合テスト	トップダウンテスト ボトムアップテスト
システムテスト	セキュリティテスト ユーザビリティテスト
テストの方法	ホワイトボックステスト ブラックボックステスト

ウォーターフォールモデル



アジャイルモデルでの開発

全て設計してから開発、テストと進むのではなく、小さな単位での設計、開発、テストを行う

開発とテストを並行して進むこともある

テストを作成して、テストが通るように開発する
TDD(テスト駆動開発)というアプローチも

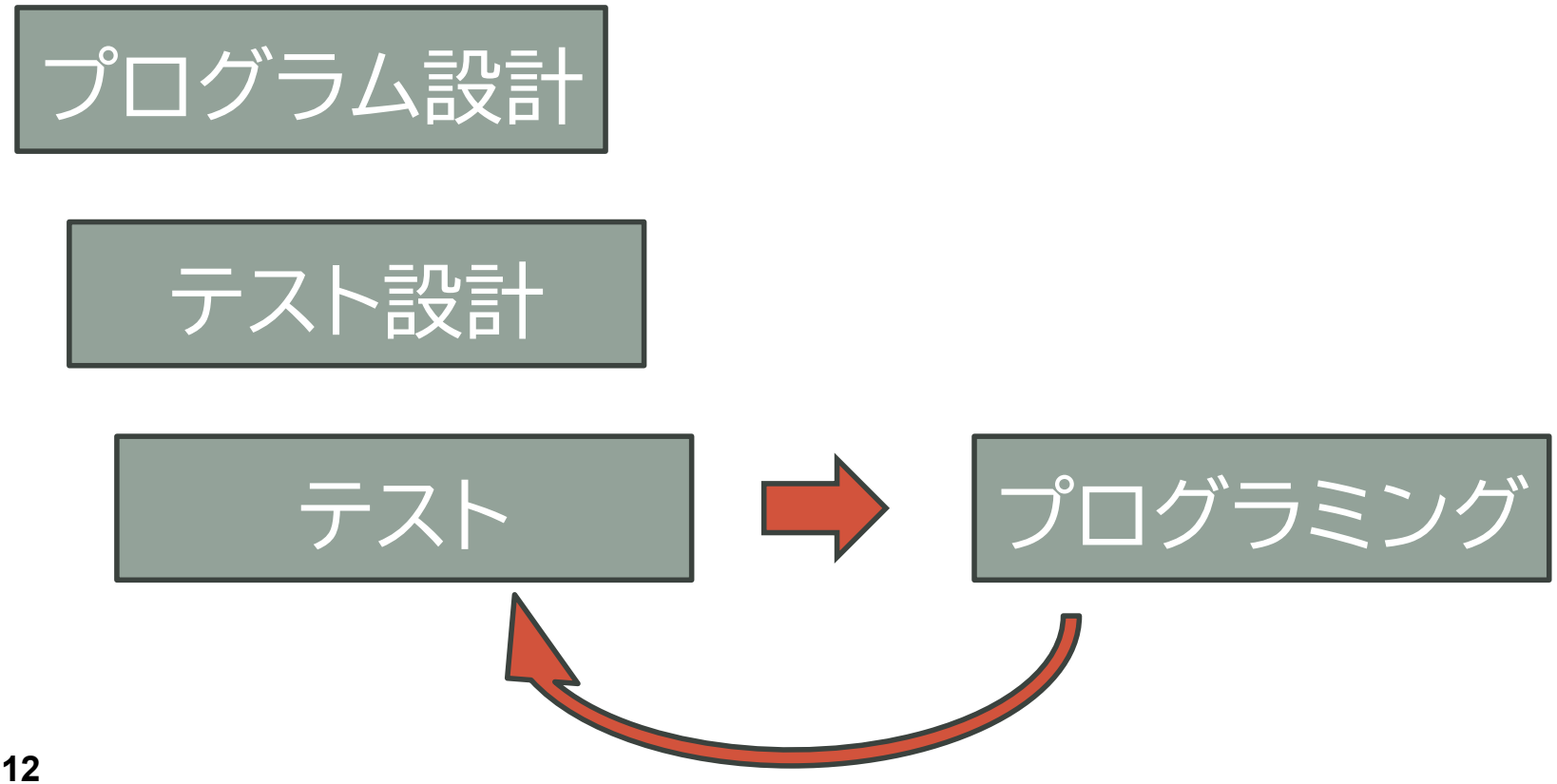
テスト駆動型開発(TDD)の考え方

・要求定義に基づき、プログラミングより先にテストを作成する
・テストをパスするようにプログラミングをする

テストが通る状態になってからプログラム改善する
リファクタリングや機能追加を実施

情報Ⅰでの導入可能性

TDD(テスト駆動開発)の考え方なら情報Ⅰにも
関連させやすいのではないか



テストのイメージ

AtCoderや日本情報オリンピックでの
採点のイメージ

与えられた全てのテストデータ(ケース)
に対して、想定の正しい結果を
返すことができればテストをパスした

取り組むべきテストについて

情報Ⅰのプログラミングに適用しやすいと思われる
単体テスト

Pythonの環境導入が用意な
Google Colaboratory

を対象に、Pytestの単純な事例のみ説明

Pytestの基本

Pythonで作成したモジュールを呼び出して、テストデータを渡して検証する

コマンドとしてのpytestを実行すると、ファイル名の先頭にtest_とつくスクリプトについてテストを実行する

Pytestのコード例その1(足し算)

```
!pip install pytest

%%writefile test_add.py
import   pytest

def   add(a, b):
    return   a + b

def   test_add():
    assert   add(1, 2) == 3
```

pytest実行(テスト成功例)

!pytest

```
============================= test session starts ==============================
```

(略)collected 1 item test_add.py . [100%]

```
======== 1 passed in 0.01s ======
```

Pytestのコード例2(足し算)

```
!pip install pytest

%%writefile test_add.py
import pytest

def add(a, b):
    return a - b

def test_add():
    assert add(1, 2) == 3
```

pytest実行例(テスト失敗)

```
!pytest

 1 collected 1 item test_add.py F [100%]
===FAILURES===============
```

```
E  assert -1 == 3
E       +  where -1 = add(1, 2)
```

```
test_add.py:7: AssertionError
```

Pytestを利用した問題例

nが偶数ならばTrue　奇数ならばFalseを返す

```
def is_even(n):
    return
```

ここを考える

```
def test_is_even():
    assert is_even(2) == True
    assert is_even(3) == False
      (略)
```

Pytestコード例

偶数判定関数のテストコード

```
def is_even(n):
    return n % 2 == 0


def test_is_even():
    assert is_even(2) == True
    assert is_even(3) == False
      (略)
```

テストで考慮すべき事項

乱数を用いた処理のテストの難しさ

(1)乱数を用いて実装した処理が想定した確率となっているか

(2)乱数の値と対応した処理(グー, チョキ, パーの決定等)がなされているか

プログラミングコンテストの効果

- AtCoder
- 日本情報オリンピック

アルゴリズムをしっかりと考えなければ、
解くことができない問題のレベルとなっていく

テストの効果

教科書の例題程度の難しさ
プログラミングなので、唯一の正解例はない

テストコードが通れば良しとする問題ならば

「解けた！」という体験

教育現場での生成AI活用の展望

- Microsoft 365 Copilot
- Gemini for
  Google Workspace Education

教科「情報」においては、汎用的な使い方と共に
プログラムのコード作成という面で影響が大きい

生成AIとテストの関係

情報Ⅰで扱う範囲のプログラムについては
生成AIでも十分な質のコード

実際の開発でも生成AIを活用する比率は
高まっていると思われる

生成AIとテストの関係

1. 生成AIによる生成コードの質

2. テスト設計での生成AIの支援

情報Ⅱ教科書におけるテストの記述

項目	取り扱われる語句
テストの説明	単体テスト，結合テスト， システムテスト
単体テスト	命令網羅，分岐網羅
結合テスト	トップダウンテスト ボトムアップテスト
システムテスト	セキュリティテスト ユーザビリティテスト
テストの方法	ホワイトボックステスト ブラックボックステスト

テストケースの生成AIへの相談

6歳までは子供、6歳以上は小中学生、16歳以上は大人、65歳以上は高齢者となる判断のプログラムのテストケースは？

生成AIでのテストケース提案

入力(年齢)	期待	狙い
0	子供	下限境界
6	子供	子供の上限
15	小中学生	小中の上限
16	大人	クラス境界 (小中→大人)
64	大人	大人の上限
65	高齢者	クラス境界 (大人→高齢者)

まとめ

情報Ⅱの「情報システムとプログラミング」、
テストをプログラミング教育について考察

Pytestのようなテストツールを使い、テスト駆動開発の単体テストを情報Ⅰでも展開する提案

生成AIの発展により、プログラミング以外の設計やテストについて、活用する流れも考えられる

参考文献

- 『テスト駆動Python 第2版』(翔泳社, 2022)
- 『ソフトウェア開発にChatGPTは使えるのか？』(技術評論社, 2023)
- Pytest公式ドキュメント(参照日:2025年6月20日)