Vol.2013-CE-121 No.6
2013/10/12

ソフトウェア製作を通した情報教育初学者における,各製作プロセスと教育効果の考察

大村 基将[†1]

現在,初・中・高等学校の情報教育として,ソフトウェア製作を題材に授業が行われている.
本稿では,ソフトウェア開発における「設計」がどのように定義されているか,および設計活動のアプローチを整理する.
また,初学者へのソフトウェア製作を題材とした先行研究等を元に,各プロセスの実施の現状と,ソフトウェア開発を教育するにあたっての課題を整理する.

1. はじめに

2006 年以降,情報通信産業（ＩＴ産業）の市場規模（実質 GDP)は全産業の中でトップを維持しており,日本における主要な産業を担っている [1].
2012 年より,中学校技術・家庭の学習内容に「プログラミングによる計測・制御」が加わるなど,教育におけるプログラミングの重要度は高くなっている. また,安倍政権は 2013 年 6 月 14 日に「日本再興戦略-JAPAN is BACK-」を策定した[2].
その中で,「産業競争力の源泉２. 設計プロセスが曖昧なプログラミング教育となるハイレベルな IT 人材の育成・確保」を目指すための方策として
「義務教育段階からのプログラミング教育等の IT 教育を推進」を掲げていた.
これをうけ,小学生や中学生へのプログラミング教育が注目されはじめている.
「産業競争力の源泉となるハイレベルな IT 人材の育成・確保」の観点を踏まえたプログラミング教育を実施していくにあたり,プログラム開発プロセスと設計の観点を特に注視していく必要があるのではないかと考えた.
初学者へのソフトウェア開発において設計の観点を検討するあたり,本論では,実際のソフトウェア開発における,開発プロセスや設計の定義および,設計アプローチ法を整理すると共に,中等教育までの授業におけるソフトウエア設計の位置づけと,課題の検討をおこなう.

2. ものづくりにおける設計

中学校技術・家庭（技術分野）にて扱われる「ものづくり」の授業では, 作成に至るまでに,大きく
「設計」→「制作」→「確認（テスト)」
のフェーズが設けられている[3].
多くの展開では
『構想図の作成・製図→工作→工作精度・実用性の確認』
というフローがみられる[3].
このとき,各フェーズの成果物,観点および技術的要素などは明確になっており,それをレビュー観点としてインクリメンタル型のプロセスをたぐるケースが多い.

3. ソフトウェア開発現場における設計

実際のソフトウェア開発において行われる設計活動を見なおした.
ソフトウェア開発における開発プロセスは
共通フレームワーク 2007(SLCP-JCF2007)においては,以下のとおり定義されている[4].

(1) **要件定義**
(2) **システム設計（システム外部設計）**
(3) **システム方式設計（システム内部設計）**
(4) **ソフトウェア設計(ソフトウェア外部設計・内部設計)**
(5) **プログラミング**
(6) **ソフトウェア結合テスト**
(7) **システム結合テスト**
(8) **システム適格性確認テスト**

それぞれの概要を以下に示す.

(1) **要件定義**
システムの利用者が,そのシステムを利用して何がしたいかを整理し,その要求に対してどのような機能や性能が必要であるかを検討する.

†1 静岡大学大学院
836 Oya ,Surugaku,Shizuoka-city, Shizuoka 422-8529,Japan.

(2) **システム設計（システム外部設計）**
システム化の対象となる業務について，ユーザからのヒヤリングにより，システム化する範囲や対象業務を明確にする.
この結果から,新システムに必要な機能を検討する.

(3) **システム方式設計（システム内部設計）**
要件定義を満たす機能の実現に必要なシステムの構成を決定する.
ハードウェアの構成，システムの処理方式などを決定する.

(4) **ソフトウェア要件定義（外部設計）**
業務の手順を整理して，システムで扱うデータや処理の流れ，ソフトウェアの内容を決める.また，画面（GUI）など,ユーザーインタフェース部分の仕様を検討する.

(5) **ソフトウェア方式設計（アーキテクチャ設計）**
ソフトウェア要件定義にて決定した機能ごとにコンポーネント（サブシステム）に分割して機能を定義するとともに,コンポーネント間のインタフェースの仕様などを設計する.

(6) **ソフトウェア詳細設計**
コンポーネントをさらに小さな単位のモジュール（ユニット）に分割する.
各モジュールが担当する機能や実現方法（用いるアルゴリズム）,モジュール間のインタフェースなどについて定義する.

(7) **プログラミング**
a)コーディング
システム設計に従い，プログラム言語を用いてモジュールごとにプログラム作成を行う.
b)単体テスト
モジュールごとにプログラムの誤りがないかを検証する.

(8) **ソフトウェア結合テスト**
ソフトウェア方式設計にて決定した機能要件どおり動作するかどうかを検証する .
単体テストをクリアしたモジュールを結合して実施する.

(9) **システム結合テスト**
ソフトウェア，ハードウェア，それ以外（人間による作業など）のシステムのすべてを結合し,システムがシステム方式設計の通り動作し,要件を満たしているかどうかを確認する.

(10) **システム適格性確認テスト**
実際に業務で使うデータを用いて,構築したシステムが,要求を満たすものであるかを検証する.

上記のプロセスが示す通り,ソフトウェア開発の大半は,設計と試験より構成されているといえる.
具体的なソフトウェアの製造工程にコーディングが該当する.
開発言語に起因する範囲のソフトウェア設計をコードとして記載していると捉えると,物理的なソフトウェア製造はコンパイラ（インタプリタ）が施しているとも言え,ソフトウェア開発における開発者の担当領域は設計と試験のみと考えることもできる.
　ウォーターフォール型の開発に代表されるように,一般的にソフトウェア設計はより上位（前フェーズ）の設計成果を用いて進めていくため,設計に対しては十分な検討が必要である.
また,ソフトウェアの試験は,各設計フェーズでの設計内容をもとに実施されることから,ソフトウェア開発において設計の重要性が高いことが示唆できる.

4. ソフトウェア設計アプローチ

ソフトウェア設計を行う場合,大きく以下のアプローチが用いられる.

(1) **プロセス中心アプローチ**
(2) **データ中心アプローチ**
(3) **オブジェクト指向アプローチ**

各アプローチは着目点が異なるものの,いずれの場合においても,課題の範囲に対して以下のようなモデル構築が行われる.

「現実のモデル化」
　→「現実の抽象的モデル化」
　　→「理想的な抽象的モデル構築」
　　　→「理想的な現実のモデル構築」

　これにより,ソフトウェアの目的である「仕事の自動化」の適用範囲や,有効性の検証を行なっている.
　以下に,各アプローチの概略を示す.

(1) **プロセス中心アプローチ**
「システムはある機能を実現するプロセス（処理,手続き）の集まりである」ととらえる.
そのプロセスの具体的な内容は何か,そのプロセス実行のためにはどのようなデータが必要か,という考え方に基づ

いて設計をおこなう方法論である.

(2) **データ中心アプローチ**
プロセス中心アプローチが時間経過と共に変更されやすいプロセスに着目するのに対して,より安定したデータに着目する設計手法である.
データ中心アプローチでは,要求分析の結果を元に開発対象となるシステム全体のデータモデリングを重視し,データを標準化することによって,システム全体としてのデータの重複を排除することが可能となる.

(3) **オブジェクト指向アプローチ**
手続きとデータを行った帰化したオブジェクトを対象として分析・設計を行う方法論である.

5. 設計プロセスを意識した教育実践の有無

ソフトウェア開発初学者への設計の提案については,草野ら[5]による,大学生へのソフトウェア開発のアルゴリズム設計の支援としてデータ指向の開発アプローチである JSP 法の利用の提案や,影山[6]による,大学生を対象とした段階的 UML 設計手法の提案などが報告されている.
また,井戸坂ら[7]は,中学校,技術・家庭において,状態遷移図を取り入れた学習の提案をおこなった.
ソフトウェア開発の初学者への設計法の提案の多くは,大学生以上の段階にあるものを対象とすることが多く,中学生以下を対象とするケースは非常に少ない.

発達段階に応じてソフトウェア開発教育の変化が見られるかを確認するため,一般に公開されている中学校技術・家庭における計測・制御学習の指導案を対象に,ソフトウェア開発プロセスの設計を意識した授業を行うケースは調査中である.
（学習序盤～中盤において,ローチャートを学習するケースがいくつか確認できたが,実際に生徒のみでソフトウェアを開発する発展学習においては,指導案で示される評価観点に含まれているケースが見当たらなかった.）
技術・家庭の授業では,設計を明確化することよりも,コーディングと動作を重視する傾向が見て取れる.

6. 教育現場は実はアジャイル型なのか？

学習指導案から確認した,授業構成は以下の特徴が見受けられた.
(4) **数人のグループにて活動が行われる.**
(5) **チーム内コミュニケーションを重視する**
（問題はコミュニケーションにより共有し,解決方法もチーム内での相談により解決する）.
(6) **ペア（複数人）プログラミングにてコーディングする.**
(7) **設計書よりコーディングを重視する.**
(8) **途中で,複数チームのコードを比較する時間を設け,自チームのコードのリファクタリングを促す.**
(9) **実機によるトライ＆エラーによるバグ修正される.**
（このとき,テストケースは設計よりも明確である場合が多い.）
(10) **授業は機能単位の単元で構成されることが多く,開発は反復的に行われる.**

この特徴はアジャイル型のソフトウェア開発の特徴によく似ている.
アジャイル型ソフトウェア開発では,開発対象を小さな機能単位に分割し作業を行い,ひとつの反復でひとつの機能を実装していく.実際の動くプログラムを進行尺度とするため機能実現の状況がわかりやすく,教育においても,画面上あるいは実機上で状況を確認しやすい利点がある.
また,文書化よりもコミュニケーションに重きを置く考え方から,協調学習的な効果が得られる可能性が考えられる.
一方で,アジャイル型開発は以下の批判を受けることが多い.

(1) **開発チーム内に,熟練した開発者（＝ソフトウェア開発の知識を持つ者）を複数人必要とする.**
(2) **ソフトウェア設計が不十分になりやすい.**

明確に設計フェーズを定義し,アウトプットとしての設計書の作成と検証をおこなう開発手法と異なり,
機能実現とコードを優先することから,本来必要な考慮が抜けてしまいやすく,そのフォローに有識者の知識と経験によるフォローを必要とするためと考えられる.
初学者へのソフトウェア開発教育においては,技術的に未熟であることを加味し,ソフトウェア設計の場と評価を明示的に行う必要があるのではないだろうか.

7. まとめと今後の研究課題

本論では,実際のソフトウェア開発現場における設計の定義とプロセスおよびアプローチ法を調査結果と,現段階までの調査で確認できた,中学校技術科・家庭を中心としたソフトウェア開発教育の現状を報告した.
今後,ソフトウェア開発の初学者に対してソフトウェア設計に注目したアプローチの必要性を検討するにあたり,以下を課題として検討を進めていく必要があると考える.

(1) **ソフトウェアプロセスで定義された各設計フェーズを実現するにあたっての,前提条件（知識・発達段階）などの素養など）の明確化**
(2) **各設計フェーズを実施することによる教育効果（獲得する技術的素養）の調査**
(3) **発達段階に応じた,ソフトウェア開発における理想的なプロセスの考察・定義**

参考文献

1) 総務省:情報通信の現況と政策動向
http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h24/html/nc241110.html(2013.0917 確認)
2) 平成 25 年 6 月 14 日閣議決定:日本再興戦略-JAPAN is BACK-
http://www.kantei.go.jp/jp/singi/keizaisaisei/pdf/saikou_jpn.pdf
(2013.0917 確認)
3) 門田泰弘ほか：技術・家庭　技術分野、開隆堂 p.32(2012)
4) 情報処理推進機構ソフトウェアエンジニアリングセンター:共通フレーム 2007―経営者、業務部門が参画するシステム開発および取引のために (SEC BOOKS),オーム社
5) 草野 翔,橋浦 弘明,古宮 誠一:事務処理ソフトウェアの設計法学習支援システム,電子情報通信学会技術研究報告. KBSE, 知能ソフトウェア工学 110(158), pp.1-6, 2010-07-22
6) 影山 智一,上田 賀一:初学者を対象とした段階的UML設計手法の提案,情報処理学会研究報告. ソフトウェア工学研究会報告 2010-SE-167(26), 1-8, 2010-03-11
7) 井戸坂 幸男,青木 浩幸,李 元揆,久野 靖,兼宗 進:状態遷移概念を利用した制御プログラミングの学習効果,日本産業技術教育学会誌 53(3), pp.179-187, 2011-09-26