義務教育段階における設計重視型プログラミング教育の提案

大村 基将[1,a)]　紅林 秀治[2,b)]

概要：義務教育段階の生徒らを対象としたプログラミング教育において，設計を重視するフロントローディング型の開発プロセスの導入を提案する．ソフトウェア開発を含む製品設計では，抽象度の高い概念的な内容から具体的な内容へ落としこむ設計段階を設定する．一方プログラミング教育では，設計は軽視されデバッグによる改善活動が主となる．その設計段階は，設計を正しく実施することで，製品開発では問題の早期発見や段階的な知識・技能の獲得，コストの削減等が果たされている．これらの設計の効果は教育においても問題解決能力の向上や新たな知識・概念獲得の側面から効果が考えられる．そこで初学者向けのプログラム設計プロセスの検討を行い，「目的の明確化」，「全体システム構成の明確化」，「ソフトウェアシステム構成の明確化」，「手順の明確化」の 4 段階とすることを提案する．

Proporsal of Programing Education For students in compulsory education stage

Abstract: In compulsory education stage programming education, we propose the introduction of a development process with an emphasis on design. Designed to be used in product development has set the design stage. In many cases, the design stage is carried out in order to design concrete from design at a higher level of abstraction. On the other hand, in the education programming, debugging activities accounted for much of the course hour, the proportion of design is slight. Design activity reveals the challenge early. In addition, this activity to increase the ability to solve problems. Effect of design activities are also useful in education.

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]．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」を掲げた．中学校技術・家庭（技術分野）では 2012 年より学習内容に「プログラミングによる計測・制御」を必修化するなど，上記に先駆けた対応が行われてきたが，成長戦略の一つに掲げられたことで，小学生から高校生までのプログラミング教育が加速することが予想される．その結果，小学校から高等学校までの授業の中にプログラミング教育が組み込まれていくことが考えられる．その組み込まれ方も，中学校技術・家庭（技術分野）の学習指導要領解説にて，計測・制御システムの各要素において情報の伝達が行えるようにするためにインタフェースが必要であること知る内容を求めるようになった [3] ことからもわかるように，機器を用いて行う物事の情報・理論など無形の要素を検討する内容へと，学習内容が拡大していくと予測される．これは，情報分野の学習において，ものづくりの一連の過程や，ものがつくられるまでに検討すべき内容などが，情報産業と関連していることを考慮しているからである．しかしながら，IT 産業で行われるプログラム開発では設計活動により品質（成果物の妥当性）を上げることを重視する傾向にある現状に対し，中学校や高等学校で扱うプログラミング教育では，設計活動よりも動作確認（プログラムのテスト）により品質を上げていることを重視する傾向にある．技術・家庭（技術分野）だけをとっても，情報以外の「ものづくり」単元では厳密に構想図や設計図を用

[1] 静岡大学大学院
836 0ya Surugaku Shizuoka city, Shizuoka prf. 422–8592, Japan
[2] 静岡大学
836 0ya Surugaku Shizuoka city, Shizuoka prf. 422–8592, Japan
[a)] omura.motomasa.14@shizuoka.ac.jp
[b)] eskureb@ipc.shizuoka.ac.jp

いて設計を行うにもかかわらず，情報に関しては，明確な設計活動や成果物が定義されていない．この状況は，情報産業における「ものづくり」とは乖離した状態であり，同教科が目標とする「ものづくりを支える能力などを一層高める活動」であるとは言いがたい [3]．また初学者への設計活動の検討は，高等教育以降についてはいくつかの実践が報告されている [4][5][6] が，中等教育以前では取り組まれていない現状がある．これらの点から，初学者へのプログラミング教育の在り方を，設計活動の視点から検討する必要を感じた．

本論では，実際のプログラム開発における設計の定義や設計アプローチを整理するとともに，中等教育までの授業におけるソフトウェア設計の位置付けと，課題の検討を行う．なお以後において特別な注釈がない限り，機器類を用いて行う物事の情報・理論など要素の部分をソフトウェア，プログラム言語を用いて処理手順を表現したものをプログラムと称する．

2. 設計

渡辺は設計行動を「目的意識をもって，現存の利用可能なあらゆる知識を駆使し，具体的な手順を用いて最適なものを表現することである」と定義した [8].「目的意識」から「最適なもの表現」に達するまでの設計の過程については，試行錯誤的なプロセスを手順化した設計過程と，有用な設計の過程を分析的な側面から検討した設計過程が考えられる．

試行錯誤的なプロセスを手順化した設計過程として，N.Cross は不完全に定義された課題空間の探求からデザイン案を導き出すまでの一連の設計作業手順を，4 つのプロセスの流れとして示した [9]．N.Cross の示した設計作業手順を図 1 に示す．

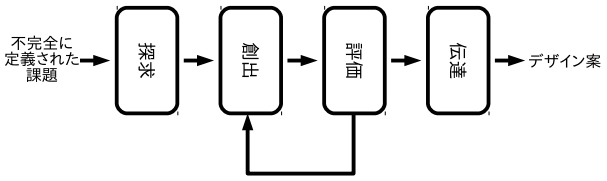

図 1 N.Cross の示した設計作業手順

N.Cross は，試行錯誤的なプロセスを手順化した設計過程では多くの場合に評価段階から伝達へ一度で至ることはなく，満足のいくデザイン案を求め評価から創出段階へとフィードバックループをもどり，新たなデザイン案の創出と評価を繰り返すとした．

試行錯誤的なプロセスを手順したモデルと科学的な設計モデルの中間的な位置づけの設計モデルとしては，向坊が材料・機器・知識といった要素に着目した場合の設計の過程を示している [10]．向坊の示した設計の過程を図 2 に示す．この設計過程では，まず製品の目的に沿う構成要素の

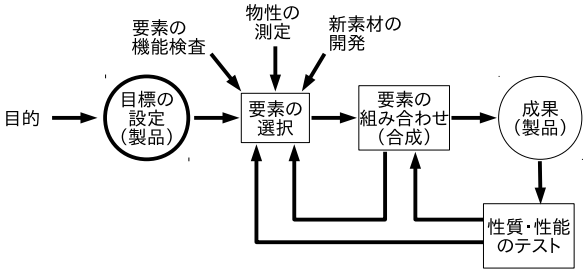

図 2 向坊の設計の過程

性質等を分析・選択する．その後，それらの結合し評価を行うことで課題を明らかにする．これを繰り返すことで，設計の品質を高める．

より科学的な設計モデルとしては，G.Pahl らが示した設計の過程がある [11]．G.Pahl らが示した設計の過程を図 3 に示す．ニーズの詳細化を通して作成した要求仕様により

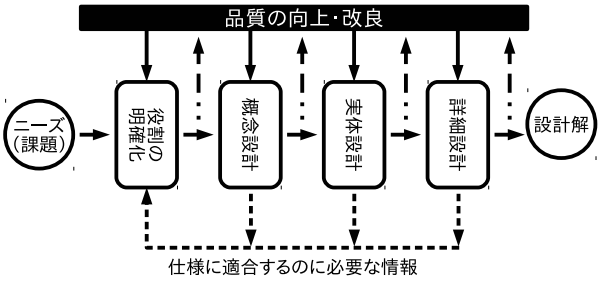

図 3 G.Pahl の設計の過程（概略）

成果物の目的を確認し，それを実現するための機能の構造や機能間の関係性を「概念設計」にて明らかにする．そして，概念設計で示された内容を実現するための具体的な手順を検討することを提言している．また，設計の過程を明確な作業要素として分け，それらの処理内容の問題点や矛盾点が発生した時に，前段階の作業要素へ後戻りをすることで，各作業要素での検討内容を洗練させることを示した．結果として，設計対象を構成する要素全体が最適化されたシステムとして整理され, 製品としての完成度が高まることを目指している．この過程では, 設計段階ごとに明らかにすべき事柄を定義し, 各設計段階が求める範囲で設計対象の要素化または結合を図る活動を再帰的に繰り返す．設計に至るまでの要求や課題，製品を実現するまでの構成要

本文は実際には論文誌ジャーナル編集委員会で作成したものである．

素などの分析を特に重視し，設計概念から具体的な構造・動作までを段階的に検討しながら課題解決を行うことが特徴である．

上記の設計過程から，科学的な設計モデルになるほど以下の特徴が顕著に表れている．

(1) ニーズや課題，目的を実現するための機能・知識・材料など，設計対象を構成する要素の分析を行う．
(2) 要素の関連（結合）を検討し，システムの構造を明らかにする．
(3) 設計の作業要素の段階的繰り返しにより，作業要素の成果を洗練する．
(4) 作業要素間の繰り返しは，複数のプロセスをまたぎ，非連続な関係であっても発生する．

また科学的な設計モデルでは，上記 (1) および (2) より，設計の成果として，要素の関係や構造を示したシステムと，要素の詳細を表すことが必要であることがわかる．また，(4) で示される通り，単独の設計プロセスの中で最適な解が確定したとしても，以降の設計プロセスで問題が発生した場合は見直しを行う必要があることから，設計が目指すものは部分的な最適解ではなく全体を通した最適解であると考えられる．よって本論では，目的を達成するためのシステムを構築し，その要素を明らかにするとともに，目的実現に最適化されたシステムを構築することを設計とする．

3. プログラミング教育におけるプログラム開発過程

教育におけるプログラミング活動の過程を確認するため，技術・家庭（技術分野）で利用されている T 社と K 社の教科書を確認した．

T 社の教科書において計測・制御の学習の中のものづくりの過程を示している．図 4 にその過程を示す [12]．図 4

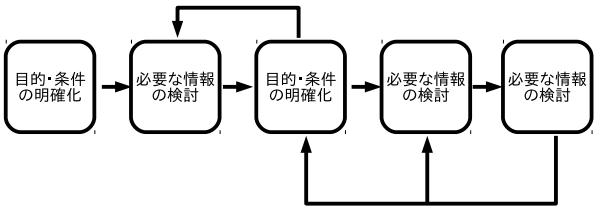

図 4 T 社の計測・制御のものづくり過程

の過程では，設計の要素の一つであるものづくりの目的が明確化している．システム構築については，「必要な情報の検討」において，目的を果たすために必要な情報を確認し，センサー等の物理的な構成要素を選択をしている．このことから，物理的に存在するものを対象としたシステムの構築が行われていると考えられる．しかし，ソフトウェアの検討においては，フローチャートを用いた手順の検討が焦点となっており，ソフトウェア内部における機能などのシステム構成の検討は行われていない．作業の段階的繰り返しによる成果の洗練についても，情報処理手順の検討，プログラムの作成，計測制御の実行において行われている．しかし，繰り返しを行う過程が限定されているため，個別の作業の成果に対する部分的な最適化が発生しているのみで，全体最適に至っていない．

K 社においては，プログラムを利用したものづくり全体の工程について触れておらず，プログラム作成手順が示されるだけである [13]．図 5 に K 社のプログラム作成手順を示す．プログラム作成手順においても，目的の明確化は達

図 5 K 社のプログラム作成手順

成できているもののソフトウェアとしてのシステム構築は行われていない．最適化についても，フローチャートで作成した極めて具体的な処理手順に対するものしか行われていない．ここまでの確認結果から，両社の教科書におけるプログラム開発の過程は，以下のように特徴付けられる．

(1) プログラム開発の目的の明確化を実施している．
(2) ソフトウェアのシステム構成の検討に至っていない．
(3) プロセスの繰り返しは部分的に行われているが，部分最適化にとどまり，全体最適に達していない．
(4) フローチャートを用いた具体的な手順の検討が実施されている．

現在 Web 等で公開されている技術・家庭（技術分野）での計測・制御の実践例や授業案を確認する限りでは，実際の授業においても上記の傾向がみられた．結果として，成果物の品質を上げるため，プログラムコードの作成とデバッグ作業によるトライアンドエラーの作業を重視する傾向が表れている．

4. デバッグ重視で行われるプログラミング教育の背景と問題の推察

中学校および高等学校で実施されているプログラム教育では設計を明確に行わず，プログラム作成と実行エラーの除去（いわゆるデバッグ）を優先する経験主義的なものづくり活動をおこなうケースがある．プログラミング教育に設計が入り込みにくい背景としては，製図などの具体的な

成果物が存在する従来のものづくりの設計と比べ，ソフトウェアは実体を持たない思考結果の産物であると見なされることから，以下の点において授業設計上の困難があると考えられる．

(1) 設計の問題が発覚しにくい．
(2) 初学者に設計を行わせる場合にどのような設計内容を実施すべきかが明確でない．
(3) 設計を行う上で必要な知識・技能の定義が曖昧である．

その結果，設計は曖昧なまますすめ，明確に問題がわかる「プログラムを実行」で確認する方法が最も容易と判断され，現在の状況になったのではないかと推測した．しかし，前提となる知識や技能を与えられずに作ったプログラムは問題が多い．また，曖昧な設計の積み重ねで作られたプログラムは，複数の問題が絡みあうためデバッグ作業は困難になる．さらに，設計がおろそかになれば，ソフトウェアを作るための問題把握や分析の能力や知識，および技能の習得をおこなう機会が失われることを意味する．

5. ソフトウェア開発における設計

一方，製品開発では設計活動での品質の向上を重視する．ソフトウェアは製品開発においてソフトウェア開発を行う場合，一般的には開発プロセスと呼ぶ作業手順を遵守する．開発プロセスは大きく「設計」,「製造 (コーディング)」,「テスト（評価)」のフェーズが存在するが，それぞれのフェーズはさらに細分化されたフェーズによって定義される [14]．詳細なソフトウェア設計の段階は，共通フレームワーク 2013 においては以下の通り示されている．

【設計】
(1) システム要件定義
(2) システム方式設計
(3) ソフトウェア要件定義
(4) ソフトウェア方式設計
(5) ソフトウェア詳細設計
【製造】
(6) ソフトウェア構築
　(プログラミング・単体テスト)
【試験（評価)】
(7) ソフトウェア結合
(8) ソフトウェア適格性確認テスト
(9) システムシステム結合
(10)システム適格性確認テスト

ソフトウェアの開発目的は，手動で行われている仕事の一部分をコンピュータによって自動化することである．このため，コンピュータとそれに関連する人や機器を要素とした上位のシステム構成と，コンピュータの内部処理（ソフトウェア）の機能構成を主とする下位のシステム構成を検討する必要がある．

開発プロセスではこれらの検討段階を個別に分けている．一般的に，前者をシステム設計・システム方式設計で検討し，後者をソフトウェア設計で実現する．これにより，開発するソフトウェアの運用を含めた全体像が明確になると共に，下位のシステム構成に致命的なミスが発生した場合であっても，上位のシステム構成に立ち戻り検討ができるように配慮されている．上記のシステム検討からもうかがえるように，開発プロセスにおいて，顧客のニーズを把握しソフトウェアの目的を明確にする要求分析の段階以降は，検討内容を全体から段階的に具体化する流れがとられる．この流れは G.Pahl らが示した設計の過程と一致する．

検討内容を全体から段階的に具体化していく手法をとることは，各設計の終了条件が明確になる点で利点がある．すなわち,「各設計段階の完了条件は（設計段階が要求する粒度で）前段階の検討結果を満たす仕組みが実現できること」を達成すれば次の設計段階に進むという明快な基準が生まれるのである．同時に，作業中の設計段階で解決不可能な問題に直面した時は，より上位の設計の見直しを行えばよいことを示している．これは，問題解決の方法を示しているともいえる．

開発プロセスでは，品質評価（試験）の段階についても定義がある．品質評価の段階においては，個々の要素の妥当性からシステムの妥当性の評価へと段階的に進めることを定義しており，設計とは逆に具体から全体へと確認の幅を広げていく．これにより設計の各段階と，テストの各段階は対応関係が発生する．この関係は一般に V 字モデルとして表現される．V 字モデルの代表例を図 6 に示す．V

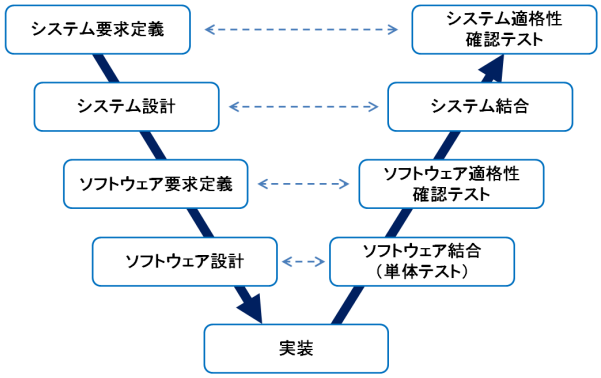

図 6　ソフトウェアプロセスの V 字モデル

字モデルでは，プロセスの流れを矢印で示し，対応関係は行にて示している．設計とテストを対応付けたことで，テストにより確認すべき内容が明らかになると同時に，テスト段階での問題確認時に，是正を行うべき設計の検討範囲が限定されるため， G.Pahl より効率的に修正を行うことができるようになる．なお，ニーズや要求を整理する要求分析，ソフトウェアを機能等の単位で要素化しシステムの構成を検討する基本設計，ソフトウェアの各要素の処理を

明らかにする設計活動を詳細設計とする開発プロセスも開発現場では多く用いられる．この開発プロセスを図 7 に示す．この開発プロセスにおいて，G.Pahl らの設計過程におけるニーズの明確化が要求分析に相当し，概念設計が基本設計，実態設計および詳細設計がソフトウェア開発における詳細設計に相当するプロセスであると考えられ，より G.Pahl らの設計に近似したものになる．上記までのとお

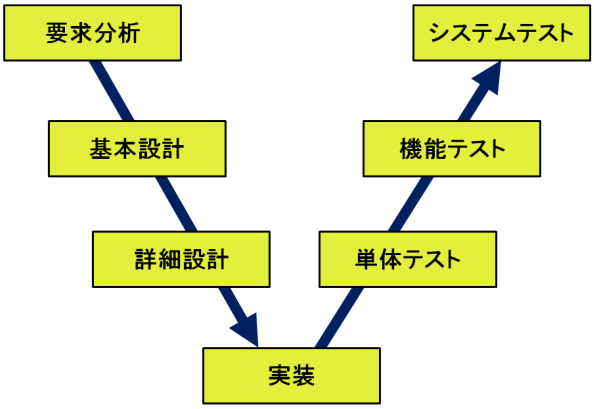

図 7　ソフトウェア開発プロセスの汎化例

り，ソフトウェア開発における設計では，目的を達成するためのシステムを構築し，その要素を明らかにするとともに，目的実現に最適化されたシステムを構築するという，設計活動をすべて満たしたもので実施されていることがわかる．

6.　ソフトウェア開発における設計の利点

基本設計ソフトウェア開発において，設計フェーズの重要性は開発効率において，顕著にみることができる．

池田は製品開発プロセスの全体に関わる大きな課題の一つとして，後戻りの発生をあげている [15]．単純な作業ミスを起因とする後戻りから必要機能の考慮不足などにともなう後戻りなどを様々な例の後戻りが発生することをあげた．池田があげた後戻り例を図 8 に示す．

図 8 に示すように，様々な原因により後戻りの発生が考えられる．後戻りによる影響の大きさは，後戻りの要因が発生したプロセスとその原因が発生したプロセスの工程上のそれぞれの位置と相互プロセスの距離によって変わる．

一般に開発プロセスの V 字カーブの検証フェーズ側（右側）から，設計フェーズ（左側）への後戻りが発生すると，その影響度は大きくなる（図 9）ことから，池田らは開発プロセスの“ V 字カーブの谷越え後戻り ”を避ける工夫が極めて重要であると結論付けている．

池田は“ V 字カーブの谷越え後戻り ”を避ける工夫として，設計段階を重視し，同段階の負荷を高める開発をあげた．開発中のソフトウェアに変更を加える場合，設計段階であれば設計書の改定を行った後にプログラミングをおこ

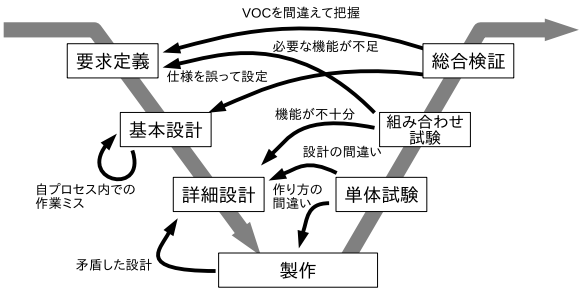

図 8　後戻りの原因 [15]

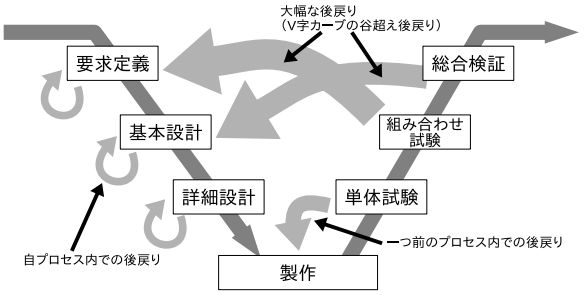

図 9　後戻りの距離と影響の大きさ [15]

なえばよい．しかし，テスト段階に達している場合は設計書の改定からやり直し，再試験を実施する必要がある．このように，ソフトウェアへの変更は，プロセスの進行に応じて，ロールバックすべきプロセスが増え，費用的にも時間的にもコスト増となる．つまり，設計初期段階での作りこみを重視することで，コストを最小限にとどめたまま，十分な性能や品質の検討ができることを意味する．このように，設計初期の段階に負荷をかけ（ローディング），上流工程で性能や品質を作りこむことをフロントローディングと呼ぶ．開発フェーズとライフサイクルコスト，開発コスト，変更の容易さの関係を図 10 のように示す．

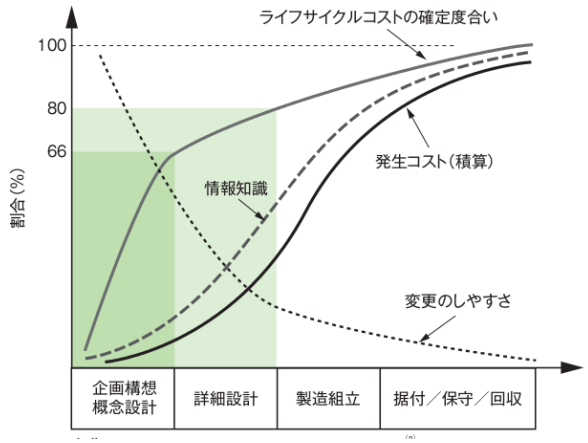

図 10　製品開発における設計の重要性 [15]

図 10 では，設計段階の利点や重要性を以下のように示している．

(1) 製品ライフサイクルコストの 80%が設計段階で確定

する．
(2) 問題発生時の変更の容易さは，設計フェーズで行うほど容易であり，テストフェーズになるほど困難になる．
(3) 試験フェーズになるほど，作業にかかるコストは高くなる．
(4) テスト段階前までに，全体の 80%の情報知識が明らかになる．

教育においてコストに相当するものを時間ととらえると，設計を軽視しテスト重視で行うプログラミング教育には以下の問題が考えられる．
(1) 製品ライフサイクルコストの 80%が設計段階で確定する．
(2) 限られた時間内での問題解決回数は，設計での問題解決を行う場合に比べ活動の回数が少なくなる．
(3) 設計段階で明らかになるはずの情報知識を獲得する機会が失われる．

7. フロントローディング型情報教育の提案

普通教育におけるプログラミング教育の目的は，中学校技術・家庭では「目的や条件に応じて, 情報処理の手順を工夫する能力を育成」[3] を，高等学校情報では「(アルゴリズムとプログラミング及びコンピュータを活用した問題解決で得た）知識と技術を問題を解決するための活動などにおいて実際に活用することができる能力と態度を育成」[7] を目指している．これは一種の問題解決能力の獲得と言える．

しかしながらこれまでの調査のとおり，現在行われている経験主義教育の側面をもつテスト重視型のプログラミング教育には，成果物をシステム（要素の関係）でとらえるという設計の重要な思考を失わせるだけでなく，問題解決という学習機会を減少させるという負の側面が存在する．現状の問題の解消（＝製作目的）を達成する最適解が成果物という面でも，問題解決の対象が常に具体的問題のみに着目してしまう．そのため，部分最適に陥りやすく，全体最適に達していない不完全な設計目的を果たせないものとなりやすい．一方，設計には，目的を達成するための全体最適された解を得るために必要な課題分析，知識・技能習得，それらを適切に統合する段階が存在する．渡辺は創造とは「それが行われる前は取るに足らないと思われていた概念を正確にすることであるといえる.」とした [8]．この意味においても,「ものづくり」のなかに創造性を見出す活動を行う場合は，曖昧な概念に着目する必要がある．知識や要素間の整合性を検証を行うことを目的の一つとする設計には, 創造性を育む要素ともなりうると考えられる．

これらのことをふまえ，プログラミング教育の初学者に対しても，設計活動を重視し負荷をかけるフロントローディング型の教育が望ましいと考える．プログラミング初学者に対するフロントローディング型のプログラミング教育では，設計段階を明確に定義し段階的な繰り返し活動を行うことを提案する．提案する初学者向け設計段階のプロセスを図 11 に示す．

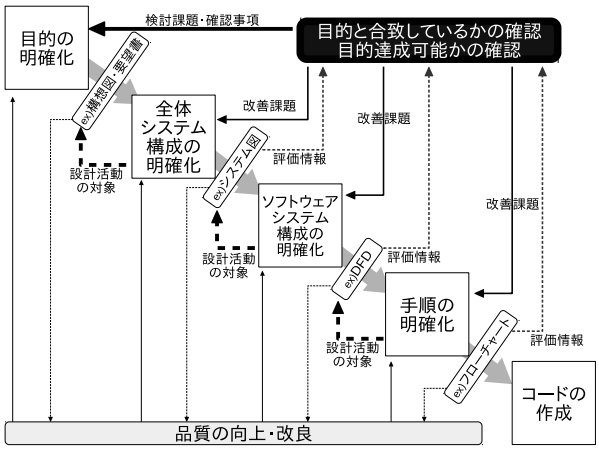

図 11 初学者向けソフトウェア設計段階

提案する設計活動では，設計段階を以下の 4 段階に定義した．
(1) 目的の明確化
(2) 全体システム構成の明確化
(3) ソフトウェアシステム構成の明確化
(4) 手順の明確化
本設計活動は，(1) により明確にした仕様を，(2)〜(4) にて全体像の把握から具体的操作へと段階的に検討し設計をすることを意図している．

設計では既有の知識を組み合わせて目的の達成を目指す．このとき，設計開始時点で設計に必要とされる知識がすべて既有であるとは限らない．また，既有であっても目的を達成できる組み合わせをみつけることは難しい．このため，抽象的な検討から具体的検討へと段階を設けることで知識の組み合わせにより得たい結果を明らかにし，必要となる知識（技術）を自分で導出できる（または，必要となる知識・技術の条件を明らかにできる）ことを期待した．

各設計段階は設計成果としてのドキュメント（以後，設計ドキュメント）を定義する．設計ドキュメントを定義することで，検討内容や進捗状況を可視化すると共に，自己と他者との製作対象に関する意識共有や問題討議などがやりやすい環境を作る．各設計段階にあたっては，前設計段階の設計ドキュメントが設計インプットとなる．前設計段階の設計ドキュメントを実現する設計ドキュメントができた時点で設計段階の終了となる．

この設計ドキュメントを設計目的と品質の向上の両視点で評価し，問題の発生した設計段階に立ち戻りながら設計活動を進める．評価にあたっては，第三者にも共通の理解を得られること（曖昧さがないこと），製作目的に合致していること，記載内容に対応する内容が前段階の設計内容に存在すること，前段階の設計内容がすべて説明されている

ことに着目して確認する．

以下に，「バッテリーチェッカー（簡易電圧計）」を例に挙げながら各設計段階の概要について示す．

7.1 目的の明確化

目的の明確化では，コンピュータを用いて実現したい課題や要求を明らかにすることを目的とする．提示された課題や要望から，性能・物理特性・動作環境を含む機能や要求される能力や，想定する運用（利用）状況などを明らかにして文書化する．「バッテリーチェッカー」の製作では，以下のような内容を検討することに相当する．

(1) どんな電池を対象にするか
(2) 電池の残量に応じてどのような警告（動作）をするか
(3) どんな材料を利用することができるか・利用可能な材料の特性はどのようなものか
(4) どこで，誰が利用するのか
(5) 電池を逆につないでしまう場合の考慮

この段階では，製作物のイメージを形にする自由記述の構想図や，調査・要望の聞き取り結果を文書でまとめたワークシートなどをアウトプットとして設定する．

7.2 全体システム構成の明確化

全体システム構成の明確化では，コンピュータの行う処理の範囲を明らかにすることを目的とする．7.1 節の設計ドキュメントを満たすために必要な物理的な要素とシステムを明確にすることで，機械（回路）・コンピュータ（ソフトウェア）・人間がそれぞれ行うべき仕事を明らかにする．

「バッテリーチェッカー」の製作では，以下のような内容を検討することに相当する．

(1) 警告を行うためには，どのような材料（装置）が必要か．（ex: LCD，ブザー，LED）
(2) 各装置にどのように命令（操作）をしたらよいか
(3) 登場物（外部装置．コンピュータ，人間）がどのように連携すれば目的を達成できるか

この段階では，製作に関連する登場物とその関連を明確にしたシステム図などをアウトプットとして設定する．

7.3 ソフトウェアシステム構成の明確化

ソフトウェアシステム構成の明確化では，コンピュータで行う処理の機能を要素化し関連を明らかにすることを目的とする．7.3 節の設計成果のうちコンピュータで行う仕事を，プロセス（機能）・状態・オブジェクトなどを単位とした要素に分解し，各要素の責務と各要素間のつながり（関連）を検討することで，目指すべきシステムを明らかにする．

「バッテリーチェッカー」の製作では，以下のような内容を検討することに相当する．

(1) 電圧の計測結果から LCD に表示するまでにどのようなプロセスが必要か検討する
ex:)
”計測結果から電圧を計算 ”→ ”電池の残量判定 ”→ ”判定結果の表示 ”
(2) ”電池の残量判定 ” の分岐条件を明らかにする
(3) ”計測結果から電圧を計算 ” をするのための計算式を検討する
(4) ”計測結果から電圧を計算 ” プロセスを実施するために必要な情報（input）と，プロセス実施結果（output）情報を明確にする

この段階では，システム構成要素とその関係を示すため，分析方法に応じて DFD やクラス図，シーケンス図などをアウトプットとして設定する．

7.4 手順の明確化

手順の明確化では，要素の現方法を明らかにすることを目的とする．7.2 節の設計成果である各要素について，求められている責務を果たすためのアルゴリズムを明らかにする．

「バッテリーチェッカー」の製作では，以下のような内容を検討することに相当する．

(1) ”電池の残量判定 ” を行うための処理手順を整理する
(2) 8bit の計算範囲の制限のなかで，どのように 16bit 範囲の値を計算するか

この段階では，プログラムの命令実行順序を含めて処理手順を示すため，フローチャートなどをアウトプットとして設定する．

7.5 設計の評価

各設計活動に対する評価は，各設計段階の設計ドキュメントに対して以下の観点で行う．

(1) 前設計段階の設計ドキュメントにかかれている内容がすべて実現できているか
(2) 前設計段階の設計ドキュメントに書かれていないことが含まれていないか
(3) 設計ドキュメントの内容は，目的を実現できるものか
(4) 表現や図に曖昧なままの部分がないか

いずれかを満たさない場合は，設計段階をやり直す．このとき，前段階の設計に問題を発見した場合は，その設計段階に戻って設計（設計ドキュメント）を修正し，以降の設計段階に影響がないかを順番に確認する．設計段階を無視しないようにすることで，全体の把握を意識させながら問題解決を図る．設計途中に改良案が思い浮かぶ場合もあり，(2) のケースが発生することも考えられる．この場合も，設計ドキュメントの内容が各設計段階のどれに当たるかを追跡できる状態を維持するとともに，不要な最適化に力を入れて部分最適になることを防ぐために許可しない．

7.6 設計の改良

ここでの改良とは問題の除去や解決ではなく性能や機能向上のための活動を指す．設計中に改良案を思いついた場合は，改善の要否や影響を判断する活動を行う．この活動により改良の効果があるのかや修正によりほかの要素に影響を与えることがないかを検討させ，常に目指すものが部分最適ではなく全体最適であることを意識させる．改良の意義があるものについては，改良の影響のある設計段階のうち最も上位の段階から設計をやりなおす．

「バッテリーチェッカー」の製作では，回路上の制限から絶対にありえない電圧の計測結果を対象とした改良などがこれにあたる．

7.7 設計の教育的価値

各設計段階では，具体物（目的を実現するに至った現状）からシステムの検討を行い，システムの改善を通して，目的を達成する理想的な成果物を検討できるように考慮する．ものづくりを支える能力などを一層高める．具体物から成果物に至るまでの検討の流れを図 12 で示す．

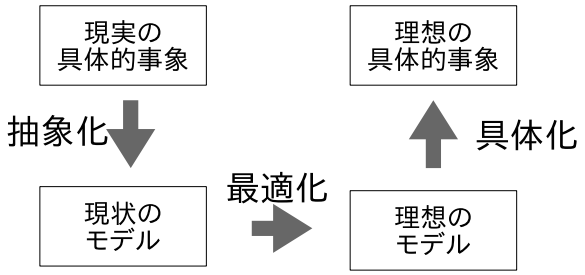

図 12 具体物から成果に至るまでの改善の流れ

8. 今後の課題

コーディングベースで行われている初学者向けプログラミング教育の現状および問題から，プログラミング学習における設計内容と到達点を明確にした上で設計プロセスを重視した学習をデザインすることは，効果検証をすべき学習方法と考えられる．このため，今後は以下の検証・検討を実施していきたい．

(1) ソフトウェア開発プロセスにおける，各設計プロセスの実現ために必要な前提条件と完了条件を既存研究から明確化する．
(2) 各設計プロセスで行われる思考内容をモデル化する
(3) 発達段階別に実施可能な設計プロセスと，各プロセスの成果物 (習得すべき内容) を明確化する
(4) 上記までの検討内容から，初学者の発達段階に応じた設計プロセスを実施可能な教材を策定し指導案を作成すると共に，想定される理想的な成果物の検討，作成を行う
(5) 初学者に対し授業実践を実施し，初学者が各フェーズにて作成した成果物・表れから，検討したプロセスの妥当性と，学びの内容を評価する

参考文献

[1] 総務省：情報通信の現況と政策動向,
入手先 ⟨http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h2410.html⟩ (2013.09.17 確認).
[2] 日本経済再生本部 ：日閣議決定:日本再興戦略-JAPAN is BACK-,
入手先 ⟨http://www.kantei.go.jp/jp/singi/keizaisaisei/⟩ (2014.05.28 確認).
[3] 文部科学省：中学校学習指導要領解説 (技術・家庭編), 教育図書 (2008/09).
[4] 荒木恵, 松澤芳昭, 杉浦学, 大岩元：プログラミング授業の導入としての「お絵かきプログラム開発演習」, 日本教育工学会研究報告集言語力を育む授業づくり,pp.111-117 (2008).
[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)
[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)
[7] 文部科学省：高等学校学習指導要領解説（情報編）, 教育図書 (2010/5).
[8] 渡辺　茂：設計論, 岩波書店 (1968).
[9] Nigel Cross：エンジニアリングデザイン [製品設計のための考え方], 培風館 (2008/07)
[10] 基礎工学概説, 向坊 隆：岩波書店 (1968).
[11] G.Pahl and W. Beitz: 工学設計一体系的アプローチ, 培風館 (1995/02).
[12] 加藤 幸一ほか：あたらしい技術・家庭 技術分野, 東京書籍 (2012).
[13] 門田泰弘ほか：技術・家庭 技術分野, 開隆堂 (2012).
[14] 独立行政法人情報処理推進機構（IPA） 技術本部 ソフトウェア高信頼化センター（SEC）：共通フレーム 2013〜経営者，業務部門とともに取組む「使える」システムの実現〜, 独立行政法人情報処理推進機構（IPA）(2013).
[15] 池田義雄：フロントローディングによる上流設計力強化, 東芝レビュー　Vol.62 No.9 ,pp3-8(2007).