ソフトウェア設計に基づく初学者のためのプログラミング学習の過程に関する考察

【 論文 】

ソフトウェア設計に基づく初学者のための プログラミング学習の過程に関する考察

大 村 基 将[1]・紅 林 秀 治[2]

[1]静岡大学教育学研究科後期3年博士課程・[2]静岡大学学術院教育学領域

要旨

技術・家庭（技術分野）におけるプログラミング教育にソフトウェア設計の考え方を取り入れた学習の過程を検討する。技術・家庭（技術分野）の学習内容の「プログラムによる計測・制御」の学習は、初学者を対象としたプログラミング学習を実施するものである。しかし、その内容は、簡単なプログラミングの手順を学ぶことには効率的であるが、サンプルプログラムによる演習やプログラムの修正作業が中心となり、プログラムを考えて作り出す、設計の考え方が抜けている。そこで、ソフトウェア設計の過程を、初学者を対象としたプログラミング学習に適用し、要求に応じたプログラムを段階的に作り上げていく学習方法を検討した。検討の結果、「目的の表現」「原理の構築」「原理の具現化」の三段階を有するプログラミングの学習過程を考案した。

キーワード

システム　設計　　技術・家庭（技術分野）　計測・制御

Ⅰ．はじめに

平成25年6月14日「世界最先端IT国家創造宣言」が閣議決定された[1]。その中で、「初等・中等教育段階からプログラミング等のIT教育を、高等教育段階では産業界と教育現場との連携の強化を推進し、継続性を持ってIT人材を育成していく環境の整備と提供に取り組むとともに、」[1]と述べられているように、初等・中等教育段階からのプログラミング教育の必要性が述べられている。

また、世界最先端IT国家創造宣言工程表[2]の中でも「【長期（2019年度～2021年度）】○人材育成を支える環境整備・小・中学校でのプログラミング等のIT教育について、全国への展開を行う。【総務省、文部科学省】」[2]と書かれていることからも初等・中等教育段階でのプログラミング教育が推し進められようとしていることがわかる。

現在、前期中等教育段階でプログラミングを学習内容として扱っているのは、中学校技術・家庭（技術分野）の中の計測・制御の学習である。計測・制御の学習では、あらかじめ準備された制御教材のサンプルプログラムを基に制御プログラムを作成し、そのプログラムの評価を教材の動作で確認し修正をするという流れの学習を行っている[3][4]。この学習方法は、短い授業時間の中で、プログラムによる計測・制御の概要を学ぶためには効果的であるが、目的達成のためにプログラムを作成し、その動作を目的達成の基準に合わせて評価していく設計の過程を取り入れた内容になっていない。そのため、製作品に新たな機能を付加したり、問題を解決するためのプログラミングによる方策を考えたりする学習がほとんど為されていないといえる。

ソフトウェア設計を行っている企業では、プログラム言語に表現する前に、処理方法を何段階かに分けて繰り返し検討している。さらに開発メンバーが複数いるため、設計内容を共有化する工夫も施されている。

筆者らは、企業におけるソフトウェア設計の手法を検討し、中学校技術・家庭（技術分野）のプログラムによる計測・制御学習に設計を取り入れた学習へと昇華させる方法を提案する。

Ⅱ．中学校技術・家庭科におけるプログラミング学習

平成20年9月に告示された学習指導要領において、中学校技術・家庭（技術分野）の学習内容に「プログラムによる計測・制御」が追加された。計測・制御のためのプログラム作成から、計測・制御の基本的な仕組みの理解と簡単なプログラムの作成を図るとともに、情報処理の手順を工夫する能力を育成することを狙いとしている[5]。内容については、計測・制御を実現するシステムの理解と情報処理の手順がポイントとなる[5]。システムの理解では計測･制御を行うシステムの要素や構成、システム内部の処理などを取り扱う。さらに、各要素間

での情報を伝達するためのインターフェースの必要性についても取り扱うことになっている [5]。なお、インターフェースは、二つ以上のものが接続している場所での信号や情報のやり取りの規約（プロトコル）を指す。システムは要素間の関連により作用する。ゆえに、要素の境界のあり方を規定するインターフェースは、互いの要素が連携を行う上の要であり、システムの全体像を把握する上で重要である。

このように、現在の学習指導要領においても「システム」が大きく取り扱われ、社会で行われているソフトウェア設計に近い内容が求められるようになってきたといえよう。

一方、情報処理の手順については、課題解決のための処理手順の検討を重点と置き、計測・制御技術の目的を意識した指導が求められている [5]。情報処理の手順の検討については、処理の手順を図示させ、可視化する指導を取り入れたり、手順の工夫については、目的に対してそれを実現する入力と出力の制御を考えさせたりするなど、適切なシステムを考えさせている。したがって、中学校技術・家庭（技術分野）における「プログラムによる計測・制御」では、プログラミングとシステムの検討を通して技術的課題を解決していくことを学習することが期待されているといえよう。これはすなわち、理論や設計に基づき人工物を製作することに他ならない。しかし、現実に実施されている「プログラムによる計測・制御」の授業を始めとするプログラミング教育にこれが適応されているとは言い難い。なぜなら、既存の教科書の内容を見ると、サンプルプログラムの入力作業をもって、プログラミング体験とするものや、問題と期待動作が明確に提示された上ですでに示されている処理内容を確認する内容、目的のないままプログラムを作成し見つかった不具合を修正（デバッグ）していくような内容が多数占めているからである [5]。これらの内容はプログラミングを体験させることを目的とする、あるいは、特定のアルゴリズムを習得させることを目的とするならば、決して否定されるものではない。ただし、このような内容では、解決に必要な情報がほぼ明らかにされている中で技術的問題解決方法を検討するという非常に特殊な状況を作っている。この状況はプログラム作成、ひいてはものづくりの実情には即しているとはいえない。多くの場合、制作対象に求められる要求、課題、期待動作、および理想的な構造について開発初期から明確であることは稀である。したがって、このような内容に基づく学習方法では、目的から具体的な目標を実現するまでの過程で生じる技術的な問題解決を体験できないといえる。また、先に述べた特殊な状況では、問題を解決すべき対象を単独の要素だけに絞ることになってしまい、制作対象の理想的なあり方や、要素間の関係を検討する機会を奪うことになる。それは、学習者にシステム概念の形成を育む機会を失わせることにもなる。このような学習方法は、現行の学習指導要領でプログラミング教育に期待されている内容からも乖離しているといえよう。

この乖離している問題を解決するためにも、筆者らは実際のソフトウェア設計における設計の過程を参考にしようと考えた。なぜなら、設計の過程には、必ず解決すべき技術的な問題が出てきて、それらをどのように解決するのか議論したり、効率よく解決するための段階が設けられたりしているからである。

Ⅲ．ソフトウェア設計における設計の過程

ニーズや要求といった漠然とした目的から、プログラムが作成できるだけの論理的方法や手順が確立するまでをソフトウェア設計としたとき、その過程には、技術的問題解決を想定した段階が含まれている。さらに、その解決方法は、システム概念に基づいている。当然ながら企業活動におけるソフトウェア設計には、工学的、経済学的、あるいは統計学的な検討や評価がなされている。本章では、ソフトウェア設計における設計の過程を検討する。

Ⅲ－1　設計について

理想的な設計の過程の一つに、Pahl らにより定義されたものがある。Pahl らにより定義された設計の過程を図1に示す。Pahl らは設計の各段階における思考内容や達成すべき成果の理想的な流れや内容を示した。これは製造業ではスタンダードなものである。この設計の過程では、分析的段階と検討段階を区分している。これは、現代の工業的分野における理想的なモデルである [7]。

Pahl らが示す設計の過程を図1を基に説明する。顧客のニーズから設計解を得るまでの段階を Pahl らは、「要求仕様の明確化」「概念設計」「実体設計」「詳細設計」の各段階に分けている。

図1の「課題の明確化」では、ニーズを分析することで製作物に期待される要求を整理し、設計解が持つべき特性を明らかにする活動が行われる。この成果として設計仕様書が作られる。目的に合致する設計解を得るためには目的や要求が曖昧なままでは、その設計解が正しいのか判断することはできない。このため、ニーズを詳細に収集、分析し、要求の矛盾や実現不可能なものがないかを確認する必要がある。場合によっては、要求に近いものに変更したり、要求を実現するための項目を減らしたりする、要求の制御が行われる。これにより、設計の妥当性を常に目的と合致しているかという観点で確認できるようになる。

図1の「概念設計」では、要求を定義し、基本機能や原理を明確にするなどして、最終的な設計解に至るため

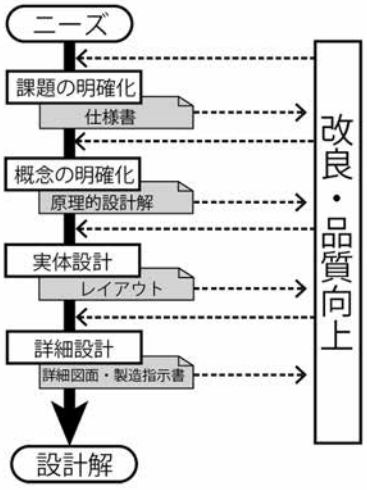

図１　Pahl の設計の過程

の道筋となる原理的設計解を作る。この成果は概略図や縮尺図などにより表現される。原理的設計解を得るために、まず設計仕様を解析し設計上の問題の本質を明らかにする。そして、特殊なものや付帯的なものを除いた本質的な問題を得る。この本質的な問題について解決策となる機能構造と動作構造を検討し原理的設計解を得る。特殊なものや付帯的なものを除くことによって設計対象の全体像としての機能構造や制約などを確認することができる。

図１の「実体設計」では、概念設計で得られた原理的設計解にしたがい、主たる機能を実現するための要素を検討する。全体のレイアウトや形状を作成し、最適化されたシステムやシステムを構成する要素を確定してく。

図１の「詳細設計」では、あらゆる部品の配置やサイズ、材料などのすべての特性が定められる。コストや製造可能性の評価を行い、最終的な製造対象の設計が確定する。実際に製造を行うために必要な情報を明確化するために、詳細図面や製造手順などの製造指示書が作られる。

Pahl らが示した設計の過程では、仮に顧客のニーズが曖昧なものであっても、その要求を分析し明確化することで、解決方法を案としてまとめあげ（概念設計）その後、その案をもとにした実物を試作（実体設計）し始める。試作を基にさらに詳細化していく。この過程の特徴として、各段階で必ず評価を行い、それぞれの評価規準に基づき、前段階へ戻ったり、次の段階へ移行したりしいる。

このように設計の過程では、要求の明確化の作業から始まり、その解決案を構想し、具体的な試作へという順で流れていく。そしてそれぞれの段階における評価規準が定められており、その規準にしたがって、設計解を得るためにステップアップしていく流れを踏んでいる。

これまでに、様々な設計の過程が定義されてきた [6][8][9]。設計者のもつ経験や知識が適切な設計解を導くことを前提に定義された設計の過程は、その活動が試行錯誤的であり、特に工業分野において正しい設計解が導き出されることは稀である。段階的にステップアップしていく過程を無視した試行錯誤的な設計は、設計に必要な情報も思考の過程も個人の経験に依存してしまい、問題点の把握や検討すべき内容を共有できない。このため、工業的な生産活動においては、思考の流れを含め最適な解を得るための理想的な設計の過程を適用することで問題を共有しながら解決できるようにしているのである。

Pahl らの設計の過程は、製作物で実現すべき要求を整理し、要求に応じた機能の構造や機能間の関係性といったシステム構造の検討から、実体設計での具体的（物理的）要素による実現を検討するという段階を経て、詳細設計により実体の各要素のあり方を定める。そして、その結果の総体が設計の目的を満たす完全な設計解となる。概念設計は機能を生み出すために様々な要素を有機的に結合していくシステム構築の原案を作る作業であり , これを具現化する方策を段階的に進める中で、設計者は、人工物のあり方をシステムの観点から把握するようになると言えるだろう。

Ⅲ－２　ソフトウェア設計について

ソフトウェアを作る本来の目的は、人間のなんらかの活動の一部について、コンピュータを中核としたシステムに置き換え、活動全体の簡便化、正確化、高速化などを図ることである。ソフトウェアはコンピュータに割り当てられた役割を実現するための手続きをプログラムとして表現したものといえる。ゆえにソフトウェアは機械要素のように具体的な実体をもたない。現実世界に影響を与えるためには当然物理的なハードウェアの存在が前提となる。コンピュータはこのハードウェアによるシステムの一つの要素である。その要素は、ソフトウェアというサブシステムを持つと考えることができる。この階層構造のイメージを図２に示す。ソフトウェア設計において、これらの事実は以下に述べる二つの問題を生む。

一つ目は、作成するソフトウェアそのものを直接表現

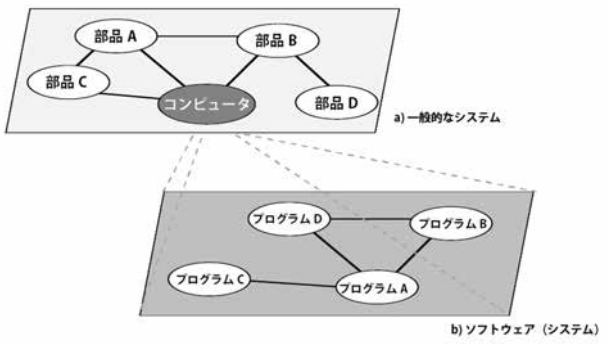

図２　ソフトウェア検討で発生するシステムの二重構造

できないことである。物理的な実体を持つものであれば、プロトタイプや類似するものを目視することで、設計対象の全体像や仕組みを設計者あるいは第三者に理解させることができる。つまりは、システムの実物教示が可能なのである。しかし、ソフトウェアは、実行しても処理の結果だけが示され、内部の動きを確認することが難しく、他者にシステムを実物教示できないのである。

二つ目は、ソフトウェア設計の複雑化である。コンピュータの性能向上により、従来は専用のハードウェア部品で実現していた機能を、ソフトウェアによりコンピュータ上で実現することが可能となった。ハードウェアを利用するよりもコスト的にも利点が多いことから、利用が増えている。その結果、ソフトウェア設計で取り扱う要素の数も増加し複雑化したのである。

これらの問題に対し、ソフトウェア設計では以下の対策が実施されている。

第１番目は、設計の段階において、システムと外部とのやりとりや、ユーザに提供する機能という視点で検討する「機能設計」と、実際のプログラムの機能や構成を決める「インターフェース設計」を分離させ、独自の段階として設定したことである。また、これらの段階においても、エビデンスとして UML[10] や DFD[11] などの図表による表現を利用し可視化している。さらに、それらの図表で示される内容は、横断的な関係性を示すことも行われている。加えて、段階ごとに生み出される機能同士の関連性を追跡できる仕組みを設けるなど、実物教示できない設計対象を可能な限り具体的に示す工夫がなされいる。

第２番目に作業の分散化である。ソフトウェア設計においては、設計を複数に分けて実施する分散開発や、既存のライブラリ（単一機能のプログラム要素）を利用することで、ソフトウェア設計の複雑化に対応している。一人当たりの担当する量を小さくすることで、設計に必要な専門的知識の範囲を狭めることができる。それにより、同時並行で行う共同作業が進めやすくなり、作業効率を上げることができる。この方法をとることで、設計者は自分の担当する要素以外の知識が必要なくなるが、自分が設計する要素と関連する要素および、インターフェースの概要を把握する必要がある。このため、自身が担当しない要素は、ブラックボックスのまま役割だけを明確にする。この要素を使ってシステムを形成することで、インターフェースの合意をとる。これにより、設計者がシステムを構成する各要素をすべて把握しなくても、全体として整合性の取れた設計を実現できる。また、このような設計を行うことで、作成した設計要素が別の開発にも利用可能なものとなる。

第３番目は、問題に共通した対応として、エビデンスの明確な定義にある。設計書として設計中の検討すべき内容を定義し、形式に則って記述する。それにより、他者にも伝達可能な設計書になる。設計書の妥当性を評価することで設計内容が共有され、設計者および他者が客観的な評価を下すことができる。

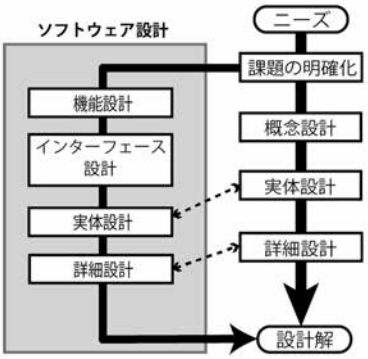

図３　システムの設計の過程

ソフトウェア設計は従来の設計方法に加え、これら「設計視点の明確な定義」、「作業の分散化」、「エビデンスの定義」の３点を検討内容に加える必要がある。Pahl らの設計の過程に対して、ソフトウェア設計に対応するよう修正を加えた設計の過程を図３に示す。図３には、ここまでに示したソフトウェア設計の特徴である機能設計、インターフェース設計、を追加した。ソフトウェア設計の過程の詳細を図４に示す。図４の機能設計では、従来の設計の過程にシステム外とのやりとりに着目し概念形成をおこなう。

インターフェース設計では、外部設計の結果を実現するソフトウェアを小さなプログラムの集合（以後、モジュール）と考え、そのプログラム同士のインターフェース設計を決める。モジュールの実現方法は考えず、機能や役割だけを割り当てる。これらを検討内容に加えることで、ソフトウェアという実体を伴わず理論のみにより構成された複雑極まりない成果物を、高効率かつ高品質で生み出すことができるようになったのである。

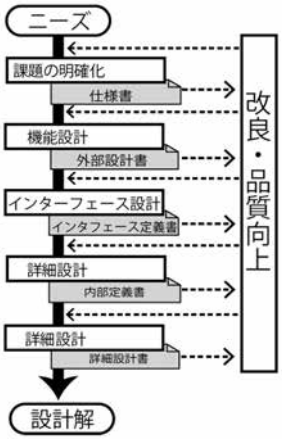

図４　ソフトウェア設計の過程

Ⅳ．ソフトウェア設計をプログラムによる計測・制御学習に取り入れるための方法

設計の過程は、一定の専門教育を受けたり、職業人としてのトレーニングを積んだりした、いわゆる専門家を対象とした内容である。本章では、３章で示した設計の過程を初学者を対象とする普通教育に適用するための方法を検討する。

筆者らは、Pahl らの設計の過程にソフトウェア設計の特殊性を鑑みた設計の過程を加えた。この設計の過程は、物理的に存在するハードウェアと、物理的な実態はないものの、ハードウェアの動作を決めるソフトウェアの両方をシステムとして考え、設計解は、要素の組み合わせで構成されていると捉えているところが特徴である。

ハードウェアもソフトウェアもシステムと捉えることで、設計解を求めるまでに必要する機能を実現する段階で、各要素の仕組みを詳しく知らなくても、接続の関連性や接続方法を知ることで、要求にかなうものを示すことができるようになる。

しかし、この過程で大事なことは、試行錯誤的に要素を組み合わせるのではなく、組み合わせ方を他者と共有しながら検討を重ねることである。

筆者らは、設計解をシステムとして捉えることと、設計解を導くまでの検討を共有しながら導くという原則が、初学者にも可能なソフトウェア設計に基づく、プログラミングの学習方法になるのではないかと考えた。初学者向けのソフトウェア設計に基づく、設計の過程を図５に示す。

図５ は、初学者自身がニーズから設計解を、システムの検討を通して導く過程を示した。この設計の過程は、「目的の表現」、「原理の構築」、「原理の具現化」の３つの過程から構成される。

「目的の表現」は、設計者が正しい設計内容の評価基準と、設計開始時に設計対象の全体像を把握するための糸口となる情報を獲得させるための段階である。ニーズから設計の目的や目標など設計解が持つべき特性（以後、仕様）を明らかにする。これには初学者に対して２つの利点がある。

一つは矛盾のない仕様を多数見つけ出すことで、検討するシステムの要素が抽出しやすくなることである。二つ目は、設計で達成したい最終的な到達点を先に示すことで、フィードバックが繰り返される設計においても、その時点の設計の達成度が定量的に評価できることである。

「原理の構築」では、すべての仕様を実現するプログラムを作るための原理を、構成要素と要素間の関連により検討する。つまり、システムを原理として設計を進めていく。システムは設計の全体像を示す。全体像は自分

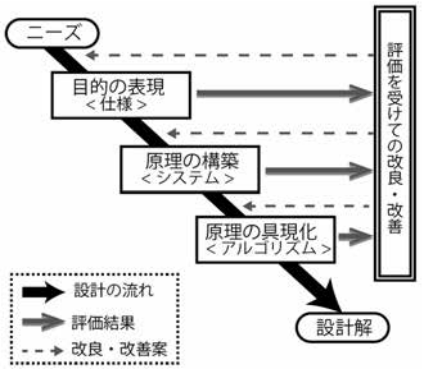

図５　初学者向けの設計の過程

の作業の位置付けを明らかにする。また、要素と要素の関係を検討することは、要素間の境界のあり方を示す。これにより、要素のインターフェースが明らかになり、既存の設計要素などと置き換えることができる。複雑な要素を事前に用意することで、設計者の技量が未熟であっても、要素の選択により解決ができる。また、技量の範囲で設計を進められるよう難易度を調整することができるようになる。

「原理の具現化」では、ソフトウェアの仕様を実現する原理をアルゴリズムで表現する。具体化すべき内容は原理の構築により要素として独立している。このため、検討内容は最も単純化した状態で、実現方法の検討を行うことができる。

それぞれの各設計段階では他者との思考の共有のため、仕様、システム、アルゴリズムを図表または、文書として表現を行う。これにより、他者による設計内容の評価を実現できるようになり、設計者の経験や知識の不足による設計内容の偏りを補うことができるようになる。また、設計段階ごとにエビデンスが得られるため、問題点を早期に発見できるようになる。これは、設計の効率化へとつながる。

初学者向けの設計の過程では、システムの検討を導入することにより、各段階で比較的小さな要素による検討ができるようになる。各段階にエビデンスを言葉や図で表すことで、他者の意見をとりいれることができるようになる。また、要素同士の関連をインターフェースとして捉えることにより、全要素の実現方法を把握せずとも、全体として矛盾のない設計ができるようになる。したがって、要素の設計の一部を他者と分担することや、既存の設計成果を取り入れることが容易になる。

これらの利点は、初学者を対象とした授業において、非常に有効なものと考えられる。

Ⅴ．設計を取り入れるための授業展開と教材案

初学者向けの設計の過程の検討により、設計の各段階で実現すべき内容が明らかとなった。ここでは、授業に

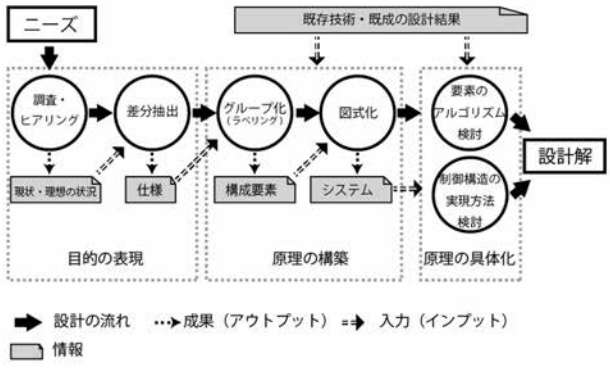

図６　設計の過程を取り入れた授業の展開案

おいて各設計段階を効果的に実施するために必要な指導上のポイントや授業展開、および、配慮すべき事項を検討する。

初学者向けの設計の過程は、一つ前の設計結果は常に正しく、必要十分の情報が得られていると考えることを原則とする。このため、各段階で明らかにする設計内容中に誤りがあれば、以降の設計結果はすべて誤りを含むことになる。問題の修正に必要な時間は、段階を経るほど検討範囲が拡大するため長くなる。また、設計結果に不足がある場合、不足に気づくまで設計活動が停滞してしまう可能性を含む。ゆえに、初学者向けの設計過程では、段階的に設計情報を取得できる仕組みや、早期に設計上の問題を発見するための細やかなフィードバックなどの仕組みを有する。

初学者向けの設計の過程を取り入れた授業においても、必要な情報の段階的な取得や、フィードバックを生かした授業の展開が望ましい。設計の過程の各段階だけ見ても取り扱う情報は多く、不慣れなうちは設計結果の導き方に迷う場合もある。また、一つの段階を一巡するだけでも時間を要する場合もありうる。授業日数が少ない「技術･家庭」のような教科では、授業の間隔が長く、検討が長期にわたって途切れてしまうこともありえる。このため、初学者向けの設計の過程を授業展開に反映する場合は、各段階をさらに細分化し、よりスモール・ステップでの授業展開とすることが有効であると考えた。初学者向けの設計の過程の各段階をさらに詳細化し、小段階ごとに活動内容を割り当てた授業の展開案を図６に示す。

図６の「目的の表現」の段階では、「作ろうとしているものが、やらなくてはいけないことは何か」を学習者が検討する活動をおこなう。このとき、現状と理想とする状況との差を埋めるために「やらなくてはいけないこと」を最も簡潔に表現したものが仕様であり、この段階で得るべき成果となる。仕様を得るためには、ニーズが発生した原因たる現状と期待（理想とする状況）を明らかにした上で、両者の差を検討する必要がある。このため、本段階では、現状や、理想の状況といった情報を取得する「調査・ヒアリング」と、調査結果をもとに現状と理想の状況の差を明らかにする「差分抽出」の小段階を設定した。なお、「調査・ヒアリング」での調査活動時は、表現方法にこだわらず、できるだけ多く現状への問題や不満、理想とする状況に関する情報を集めることに集中することが望ましい。「差分抽出」では、収集した情報の差分を取りやすくするために短文で記述し直す。図形表現などに含まれていた仕様や複数の内容が混在していた文章なども、簡潔な短文へと変換する。その上で、現状と期待の差分を検討する。得られた差分の内容についても、箇条書きや短文で表現する。これにより、差分の抽出漏れや曖昧さの混入を回避できる。

図６の原理の構築では、仕様を満たすために製作物に必要な要素と、その関係を学習者が明らかにする活動を行う。

このため、一定のルールのもとに、仕様をまとめる「グループ化（ラベリング）」と、グループ間の関係の検討する「図式化」の小段階を設けた。「グループ化（ラベリング）」では、仕様には他と似通った内容や、要素間で時間的な順番が存在するものについて、グループ化と名前がつけられないかを検討する。うまく名前がつかない場合はグループ化を諦め、別のグループが作れないかを検討する。ここでのグループがシステムの要素となる。ここでのグループ化の視点は機能や状態とするとわかりやすい。「図式化」では、グループをいくつか見つかったところで、それらのグループの関係性を検討する。すべてのグループが何らかの関係を持つまで検討を行う。このとき、システムの要素のうち学習者の能力では詳細化が困難と判断されるものなどは、あらかじめ要素を設計の部品として提示することも有効な手立てである。検討結果はそれぞれの視点にあった図示をおこなう。ここで扱う図は作図のルールが単純で作図方法の理解に時間を必要とせず修正も容易な物が適している。ある程度関係が見つかった時点で、他者と検討結果と比較し、検討内容にフィードバックを繰り返すことで偏りのないシステムが構築される。

図６の「原理の具現化」の段階では、システムの具体的な実現方法について検討する。このため、本段階では、システムを実現する各要素の機能実現方法を検討する「要素のアルゴリズム検討」と、システムの関連の実現方法を検討する「制御構造の実現方法検討」の活動を行う。両小段の入力情報は、共に原理の構築までの設計結果である。このため、実施順序は問われず、並行して実施することができる。「要素のアルゴリズム検討」では、期待されているシステム要素の振る舞いを実現するための手順を、フローチャートなどを使いながら明らかにする。特定のプログラム言語の命令でどのように表現すれ

ば良いかについては、実際にプログラムを作成するプログラミング段階の検討事項である。このため、ここでの検討は、コンピュータがどのように動いたらシステムの振る舞いを実現できるかについて考える。「制御構造の実現方法検討」では、「ボタンを押したら機能が切り替わる」というような機能の関係性を実現する手順を検討する。検討が完了したら原理の構築の成果であるシステムの表現図やフローチャートを指でなぞるなどしながら動作のシミュレーションを行うことで、妥当性を検証する。また、他者にも検討漏れの有無や、実現可能であるかを評価してもらうことで、より客観的な評価を行えるようにする。

ここまでにあげた設計の過程および授業の展開を実現する方法のひとつに、状態遷移図を教材とする方法が考えられる。ある状態から別の状態に変化することを遷移と呼ぶ。状態の遷移を図示したものを状態遷移図 [12] という。状態遷移図の記述例を図７に示す。状態遷移図は、状態を円で、状態が遷移する方向と条件を矢印と文字で示すシンプルな図である。状態遷移図はあらゆるものの状態を表現することができる。このため、「目的の表現」の段階における現状と理想の状況の分析活動において、表現として利用することが可能である。ごく単純な表現方法であるため､現状と理想の状況を図示した時、変化点を明らかにすることが容易である。これは「目的の表現」の分析にとって大きな利点となる。さらに、状態を要素、遷移のルールを関係性と考えることで、状態遷移図はシステムの表現方法と考えることが可能になる。これは、状態遷移図を「原理の構築」段階におけるシステムの表現としても利用することができることを意味する。さらに､「目的の表現」で理想の状況を状態遷移図で記述すれば、その図を「原理の構築」段階のシステム案として利用することが可能な場合が多い。「目的の表現」から「原理の構築」までの表現を、同じ図法や考え方で表現できることは、学習の効率の点において非常に有利である。表記が単純であることに加え、状態という考え方が初学者にとっても日常的に利用するものでもあるため、状態遷移図の理解や習得は他の図表表現に比べても容易である。これは、個々の図表の理解の進度差が出にくいともいえ、他者との認識共有のツールとしても有用であることを示唆する。少ない時間内で設計の過程を取り入れた授業を展開する場合には、状態遷移図は強力な教材となることが期待される。

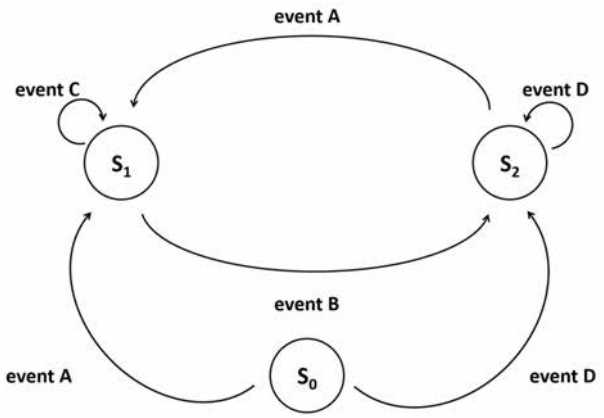

図７　状態遷移図例

Ⅵ．まとめ

中学校技術・家庭科（技術分野）における「プログラミングによる計測・制御」では、プログラミングとシステムの検討を通して技術的課題を解決していくことを学習することが必要となる。しかし、現行の同授業では、これらの活動が正しく実施されていない現状を示した。

システムの検討や技術的課題の開発の過程や方法論は工学的設計手法として一定の研究がなされてている。Pahl らの設計の過程は、製作物で実現すべき要求を整理し、要求に応じた機能の構造や機能間の関係性といったシステム構造の検討から、実体設計での具体的要素による実現を検討するという段階を経て、詳細設計により実体の各要素のあり方を定める。そして、その結果の総体が設計の目的を満たす完全な設計解を得るものであった。概念設計は機能を生み出すために様々な要素を有機的に結合していくシステム構築の原案を作る作業であり、これを具現化する方策を段階的に進める中で、設計者は、人工物のあり方をシステムの観点から把握するようになると言える

ソフトウェア設計は従来の設計方法に加え「設計視点の明確な定義」､「作業の分散化」､「エビデンスの定義」の３点を特徴とする。これにより、ソフトウェアという実体を伴わず理論のみにより構成された複雑極まりない成果物の設計を可能とした。しかし、これらの設計の過程は専門教育や職業活動により一定の技能を有した熟達者が対象である。

このため、システム形成による設計解の導出という設計の本質を維持しつつ、設計経験やプログラミング経験の乏しい初学者に適応するものとして「目的の表現」､「原理の構築」､「原理の具現化」の３段階による設計の過程を提案した。また、この設計の過程を効果的に授業に導入するために、各段階における指導上のポイントを示した。

今後は、この初学者向け設計の過程についてより具体的な授業方法および教材の検討と、それらを用いて提案した設計過程の妥当性と教育効果の評価をおこなう。

参考文献

[1] 内閣 : 世界最先端 IT 国家創造宣言（平成 27 年 6 月 30 日）

（https://www.kantei.go.jp/jp/singi/it2/kettei/pdf/20150630/siryou1.pdf）（20150911 確認）
[2] 内閣 : 世界最先端 IT 国家創造宣言 工程表（平成 27 年 6 月 30 日）
（https://www.kantei.go.jp/jp/singi/it2/kettei/pdf/20150630/siryou3.pdf）（20150911 確認）
[3] 門田泰弘ほか : 技術・家庭技術分野，開隆堂（2012）. 庭技術分野，東京書籍（2012）.
[4] 加藤幸一ほか：あたらしい技術・家庭技術分野，東京書籍（2012）.
[5] 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）
[6] L. Bruce Archer:Systematic Method for Designers, Council of Industrial Design（1965）
[7] G.Pahl, W. Beitz: 工学設計－体系的アプローチ，培風館（1995/02）.
[8] Nigel Cross:Engineering Design Methods: Strategies for Product Design, Wiley（2001）
[9] 渡辺　茂：設計論，岩波書店（1968）.
[10] イヴァー ヤコブソン，グラディブーチ，ジェームズランボー：UML による統一ソフトウェア開発プロセス—オブジェクト指向開発方法論，翔泳社（2000/03）
[11] トム デマルコ：構造化分析とシステム仕様，日経 BP 社（1994/9/20）
[12] JISX0131:1995. ソフトウェアの状態遷移の構成及びその表記方法 .

【連絡先　大村　基将
E-mail: omura.motomasa.14@shizuoka.ac.jp】

Consideration of a Learning Programming Process based on Software Design for Beginners

Motomasa Omura[1], Shuji Kurebayashi[2]

[1]*Cooperative Doctoral Course in Subject Development in the Graduate School of Education, Aichi University of Education & Shizuoka University*

[2]*Academic Institute College of Education, Shizuoka University*

Abstract

We considered a learning programming process based on software design for technology education. Lessons of computer program-aided measurement and control are for beginners to learn programming. These lessons are efficient to learn the step of programming, but the main of the lessons are works of typing the sample programming and debugging. Therefore, these lessons have a fundamental lack of the concept of design. Then we considered learning processes of programming and applied the process of software design to lessons for beginners (junior high school students) to learn programming to meet needs in a systematic manner. As a result, we propose the process of programming, which constructed three steps couching aim, construction of rules and realization of rules.

Keywords

system, design, Technial course, Measurement and Control