大阪体育大学紀要
第26巻（1995）pp.211–230

コンピューティングの工学化

齊藤了文

Transformation from Mathematical to Engineering Aspects in Computing

Norifumi SAITO

BULLETIN
OF
OSAKA UNIVERSITY OF HEALTH
AND SPORT SCIENCES

大阪体育大学紀要
第26巻（1995）pp.211-230

コンピューティングの工学化

齊藤了文

1995年3月31日受付

Transformation from Mathematical to Engineering Aspects in Computing

Norifumi SAITO

要　約

この小論では，いわゆるコンピュータサイエンスが，徐々に「工学化」を示してきたことを示す資料を収集し，それを整理することをめざす。

それぞれの学問が，自己の営為を反省する場所は2つある。初心者に対する導入と学問の最前線である。従って，ここではまず「カリキュラム」を取り上げ，次いで人工知能研究のリーダー達の発言を見ることにする。そしてその両者において，「工学」的観点が特に注目されるようになってきているという点を概観する。

まず，(1)どういう問題意識をもって「情報」に関する学問を取り上げたかを明らかにする。次に，(2)コンピュータの歴史的由来を探ることによって，その数学的基礎を取り出す。更に，(3)コンピュータサイエンスのカリキュラムを概観することによって，徐々に「工学」であることが意識されてきたことを見る。そして，(4)『カリキュラム88』をめぐるダイクストラを中心とする論争を見ることによって，その論点に入り込む。最後に，(5)人工知能研究者のリーダー達の発言に共通するものの見方を取り上げる。

I. 問題設定

コンピュータサイエンスは新しい学問である。数学，物理学，心理学といった学問は，その起源を遠く古代ギリシアにまで遡れるのに対して，コンピュータ（電子計算機）は第2次世界大戦中に生まれている。

さて，数学はもともとバビロニアやエジプトにおいて，古代ギリシア以前から使われていたことが確認されている。しかし，それを学問として作り上げたのは『ユークリッド原論』に象徴されるように，ギリシア人なのである。彼らは，公理からの演繹という仕方で，数学の定理を導き出した。例えば，「三角形の内角の和が二直角である」という定理を考えよう。この定理を知識として得たということは，3つの角をそれぞれ分度器で計ってその和を出してそれが二直角に等しいということが経験的に分かったということ以上の知識を得たということである。つまり，具体的な測定によっては，「この」三角形に成り立つことが「あの」三角形に成り立つとは分からない。しかし，納得できる公理に従って得られた定理は，それが三角形である限り，どんな三角形でも，その内角の和が

二直角であることを，確実に結論させてくれる。また，具体的な測定においては，測定の誤差は避けられない。それに対して証明された定理は，誤差には依存しない。これが，学問としての数学の普遍性であり，厳密性である。同じように，物理学も古代ギリシアのミレトス学派の自然哲学に由来するとも考えられるが，それが現代のように確実な学問として位置づけられるようになったのは，いわゆる「科学革命」を経てからなのである。そしてこの意味での科学の学問としての成立を特徴づける性質として，物理的現象を「数学的に記述」することと「実験的方法」の使用がよく取り上げられる。まず，物理的なものを定量的に捉えることによって，その限りで予測の正しさが明確に判別できる。単なる思弁によっては決着のつけにくい自然の姿に関する様々な考えを，数量的尺度によってある程度限定できることになる。そして数学的自然科学は，ものを作る場合にも，結局は役立つものとなった。最後に，心理学であるが，これは万学の祖としてのアリストテレスに由来するともいえるが，ある程度学問的な姿をとったのは，フェヒナーあたりの精神物理学あたりからだろう。心理学は学問としてどの程度自律的なものと認められるかは難しいところだが，少なくとも「実験」ができて，その再現可能性が認められることが重要であった。勝手に考えた「心理理論」がその真理性を無闇に主張するのではなく，それが多くの人が納得のいく理論になるためには，他人が調べてもおなじようなことが起こることが確認されねばならない。しかし，心理学の難しいところは，心理というものは，ある意味で人それぞれという面をもっているということである。このため，物理学のように，数学的方法や実験的方法は使ってはいても，再現可能性を主張することが難しく，普遍的で厳密な学問が成立したとはなかなかいえないことになっていた。

このように，ある対象について（例えば，数，物質，心理現象）多くの考察が始まった時期と，それをまとめてその方法論的，学問論的考察が進んだ時期とでは，ある程度相違があることは認められる。しかも，この意味で「科学革命」の起こった時代は，現代から見ると非常に古い時期であり，文献は残ってはおり，それについての研究も進んでいるにしても，学問化の生じた「時代」の理解も難しく，学問化の姿の具体的理解も，いまいち隔靴掻痒の観がある。

それに対して，コンピュータサンエンスはここ数十年に生じた学問である。これは，我々が生きている時代に生じた学問であり，そのため研究資料収集の面で有利であり，現代社会に大きな影響を与えているコンピュータという機械に関わるという意味で重要でもある。

さて，コンピュータサイエンスは，コンピュータという一種類の機械に関わる学問である。これは，数，物質，心理現象といった幅広い対象を扱う学問ではなく，非常に狭い，ある個別的な対象，コンピュータに関する学問である。これは，コンピュータサイエンスを一つの学問とみなすには少し奇妙な状況である[1)]。

さて，あらゆる具体的コンピュータは，チューリングマシンという数理論理的概念の具体化だとも考えられる。するとコンピュータサイエンスはチューリングマシンについての学問といえる。つまり，コンピュータサイエンスの学問的基礎は，結局は数理論理学ということになるだろう。これは，基礎づけをもった，一つの学問の性格を持っているということである。だから例えばコンピュータ化された社会を批判するために，時にこの数理論理学的基礎が論点として使われた。例えば，ゲーデルの不完全性定理によって，コンピュータは人間並みの知能を持つことは無理だといったことが言われたりしてきた。つまり，数学的基礎

の探求によってのみ，コンピュータサイエンスや人工知能，または，コンピュータ化された社会の批判が可能になると考えられてきたのである。

さて，コンピュータサイエンスが数理論理学という基礎から成立した学問であるとすると，これは古来の多くの学問とは違って，最初から成熟した形で成立したことになる[2]。しかし，コンピュータサイエンスの現場では，少し違った理解がされていたようである。

それは，プログラマーの位置づけと情報の多様性に由来する。コンピュータという機械を使える人を作るというだけならば，単に技能を教えるにすぎず，それならば情報工学科とか情報科学科といった大学の学科は，英会話学校のように技能を教えているだけで，学問を教えているとはいえないだろう。ハッカーの方が情報科学の教授よりも，うまくプログラミングすることもよくあるのだから，何も大学まで行って学ぶことはあるのだろうか。そして情報というものも，機械に関わるだけでなく，経済などの社会生活にも関係している。つまり，情報を扱う学問というだけならば，非常にその対象が多様化してきて，そこからみるとこれらの学問の基礎理論は何であるかということがはっきりしなかった。例えば電気工学においてはその基礎理論としてマックスウェルの方程式がある。また機械工学でも材料力学などがある。このように物質やエネルギーを扱う学問は，その基礎となる学問は明確に存在する。それに対して，「情報」を扱う学問はその基礎として何を求めるべきかということは，なかなか難しい問題である。

つまり，コンピュータサイエンスは，数理論理学という基礎をもっていたはずなのに，それが情報を処理する「一種類」の機械であったために，実際上「技能」が重視されてしまって，学問であるという性格が隠れ，また「情報」を処理する機械を扱おうとしたために，情報の多様性を捉える学問をつくることは困難になったのである。

ここでは，情報に関する学問のすべての基礎になるものは何かということを考えるのではなく（そのため後者の問題設定に関わる学問性はとりあげない），コンピュータサイエンスに話を限り，それの学問的基礎は何かということを概観することにする。もうすこし具体的に述べる。『学問としての計算機分野』Computing as a Discipline[3] においてデニング等は3つのパラダイムを提起している。第一は，理論であり，数学のように首尾一貫することをめざす。第二は，抽象化であり，実験科学のように仮説の検証をめざす。第三は，設計であり，工学が行っているように要求使用に基づいてシステムを製作することをめざす。このうち，理論は数学の学問性に関わり，抽象化は自然科学の学問性に関わる。その意味で，この両者は学問の成立に対して重要な意味をもつことがある程度確立している。問題は，設計である。これを学問成立の要因とすることがどういう意味で可能なのか。これが問題である。工学はどういう意味でコンピュータサイエンスの学問的基礎になったのか。ただし，物質やエネルギーに関する工学ではなく，情報についての工学であるという点に注目することが重要である。コンピュータという金物を作るために，シリコンの物性がどうなっているかの研究，物理的基礎を求める研究は，差し当たりコンピュータサイエンスの問題ではない[4]。コンピュータサイエンスは，コンピュータという機械を使う技能を教えるというよりも，情報に関する工学的な学問だと考えられる。「情報」に関して工学がありうるということ，そしてそれが学問でありうるということ，その考えの変遷をこの小論で同定したい。

この節を終えるにあたって情報科学に関わる言葉遣いを規定する。

情報科学 computer and information science

とは,「情報の生成, 伝達, 変換, 認識, 利用などの観点からその性質, 構造, 論理を探究する学問, およびその具体化を行う計算機を中心とする情報機械のハードウェア, ソフトウェアの理論と実際に関する学問の分野をいう」[5]と規定されている。また, 情報工学 information engineering は, 日本だけで用いられる語であって,「情報工学と情報科学との間にあまり差がなく, ほとんど同義的に使われる」とも述べられている。また, 計算機科学 computer science は,「情報科学の一分野。計算機に関する理論, ハードウェアおよびソフトウェアに重点をおいた分野をいう。情報そのものではなく, 情報を収集し, 伝達, 蓄積, 加工する枠組みとしての情報機械を研究対象の中心とする。」[6]ただし, 計算機科学と情報科学との差は縮まりつつある。また, コンピュータサイエンス, コンピュータエンジニアリング, 情報学その他の類似する学問をまとめて, コンピューティング（計算機分野と訳されることもある）と呼ぶこともある[7]。この論文の表題として用いた「コンピューティング」という言葉は, この用法に従った広い意味の学問を指す。

II. コンピュータの歴史的由来

コンピュータサイエンスは, コンピュータ（計算機）を扱う学問である。計算機械の源流としてまず理解されるのはソロバンである。これは非常に簡単な道具であるが, 今日の電卓のもつ機能を備えている。(1)記憶, すなわち何桁かの数をいつまでも保持する機能, (2)一桁の数の加減算の機能, (3)数の表示, すなわち記憶された内容を人が容易に読み取れること, である[8]。「はじめて計算機械, それも加減算だけでなく乗除算までできる機械をつくったのは, ドイツ, Tübingen 大学の教授, W. Schickard (1592-1636) だといわれているが, 彼のつくった計算機は現存していない。」[9]現存している最古の計算機は, Blaise Pascal (1623-1662)が作った加減算を行う簡単な機械である。Gottfried Wilhelm Leibnitz (1646-1716) は, Leibnitz の歯車として知られている装置を発明することによって, 加減算だけでなく, 乗除算も行える計算機を作った。さて,「計算機の歴史の中にもっとも大きな足跡を残した人はイギリス人 Charles Babbage (1791-1871) だといえよう。」[10]彼は最初「階差機関」をつくった。それは6次の多項式を階差表を計算することによって計算するものであった。この研究を進めている途中で, 彼は「解析機関」の着想を得た。彼は Jacquard 機械から得たアイディアをもとにして,「加減乗除の演算の切替えや, 数値データの転送などを, カードに穿孔したプログラムに従って順序正しく実行させることによって, 長い複雑な計算を人手が介入することなしに全く自動的に進める機械－汎用の自動計算機－がつくれることに気づいたのであった。」[11]「彼の解析機関の計画書には, 初期の（ディジタル）電子計算機に盛り込まれていた着想のほとんどすべてが既に含まれていたのであった。」[12]

この後, 1944 年に Harvard 大学の計算研究所で, H. H. Aikin のチームによって, Babbage の構想が, Harvard MarkI という自動逐次制御計算機として実現された。しかし, この機械は,「電子」計算機ではなく, 電気機械的な装置を用いた「電気計算機」であった。電子計算機の最初は ENIAC だとされる。これは 1946 年2月に公式に完成式が行われた。そして, ENIAC の重要性は, John von Neumann (1903-1957) の発案と言われている「プログラム内蔵方式」（記憶装置にプログラムも数値データも同じように記憶する方式）の導入にある。それこそが,「電子計算機というものを単なる数値計算の道具から, 真に無限の可能性を内蔵した汎用の情報処理機械に発展させた鍵だったのである。」[13] ENIAC をつくるときに, 各

種の演算制御機能を実行しうるために回路を組み立てる必要があった。多くの人々は，その電気技術的な側面に集中していた。その問題を，はじめて論理的な側面から取り扱ったのはvon Neumann であった[14]。von Neumann は A. M. Turing の論文 “On Computable numbers, with an Application to the Entscheidungsproblem” (1936) を研究していた。この論文で Turing は今日，Turing Machine と呼ばれる仮想的な計算機を提案している。この機械は，幾つかの異なる「内部状態」をとる本体と，無限に長いテープからなる。本体には，テープ上のます目を読み，そこに字を書き込むことのできるヘッドがついている。そしてテープを一ます分だけ前または後ろに送ることができる。そして，現在の内部状態とテープから読んだ文字によって，テープの前後運動とそこに書き込む文字の決定(出力関数)，およびその後の内部状態の決定 (状態遷移関数) が行われる。たったこれだけのことができる機械が実現できれば，人間が計算できることはすべてできる。このことが重要である。さて，出力関数と状態遷移関数を決めることによって計算ができるということは，この関数が特定のプログラムを意味することになって，それに従う計算をするということである。しかし，Turing は更にこれをすすめて，万能 Turing Machine を考える。つまり，「外から(テープにあらかじめ書き込んだ) 任意のプログラムを与えることによって，そのプログラムを実行させられるような万能機械がつくれることを指摘するのである。」[15]

その他関連する多くの人々に言及することはやめて，結局大筋このような経緯によって，コンピュータは単に計算をする機械ではなく，プログラムによって，何でもできる情報処理機械として理解されることになったのである。このとき，特に機械の素子が真空管であろうとシリコンチップであろうと，関係なくなってしまった。機械を作る場合に通常は重要となっていた「物理学的」基礎とは独立して，また電子工学や機械工学とは独立して，コンピュータに関する学問，情報に関する学問が提起されてきた。Turing が数学基礎論の中で提出した Turing Machine の構想が，万能の情報処理機械としてのコンピュータの基礎になっている[16]。これがコンピュータの歴史的由来である。

III. カリキュラムの分析

ACM (Association for Computing Machinary) はこれまで『カリキュラム 68』『カリキュラム 78』『カリキュラム 88：プロトタイプカリキュラム』『カリキュラム 91』と，三度にわたる (88 の本格的カリキュラムが 91 にあたる) コンピュータサイエンスのカリキュラムを出版した。(後ろにつく数字は，それぞれの出版年度を示している。)「ACM カリキュラムの発展は，そのままコンピュータサイエンスの学問体系の発展の歴史的記録となっている。」[17]

ここで見ていきたいのは，同じくコンピュータサイエンスを学問として理由づけているにもかかわらず，『カリキュラム 68』から『カリキュラム 91』へと移行するにつれて，どこに学問の本質を見ているかが変化してきているということである。それは，結局は大きな流れとして，数学的根拠づけから，工学的根拠づけへの移行としてとらえられるであろう。この論点を，資料に基づいて具体的に示したい。

カリキュラム 68

まず，『カリキュラム 68』の初級科目は，次の 4 つである。

B1. コンピューティング入門
B2. コンピュータとプログラミング
B3. 離散構造入門

B4. 数値計算

中級科目は，次の9つである。

I1. データ構造
I2. プログラミング言語
I3. コンピュータ構成
I4. システムプログラミング
I5. コンパイラ構成法
I6. スイッチング理論
I7. 順序機械
I8. 数値解析I
I9. 数値解析II

上級科目は，次の9つである。

A1. 形式言語と構文解析
A2. 上級コンピュータ構成
A3. アナログおよびハイブリッドコンピューティング
A4. システムシミュレーション
A5. 情報組織と検索
A6. コンピュータグラフィックス
A7. 計算可能性の理論
A8. 大規模情報処理システム
A9. 人工知能と発見的プログラミング

このカリキュラムでは，それ以降のカリキュラムと同様，「学問としての」コンピュータサイエンスであることが注意されている。

まず，ここで気づくことは，数学や論理学と結びつく科目が多いことである。B3. 離散構造入門は，集合論と命題論理をその中心的内容とするものであり，B4. 数値計算は，その名の通り，科学計算の基本的な数値アルゴリズム，例えば常微分方程式の数値解法といったものを扱っている。I7は有限オートマトンを扱う，数学的論理学的内容のものであり，I8，I9は明らかに数値アルゴリズムを扱うものである。またI5も数学的技法の紹介が中心となっている。またA7は数学基礎論に関わる内容を中心にしている。

また，これらの科目以外に，数学科目として，微積分や確率や線型代数などの学習を奨励している。

『カリキュラム68』においては，「コンピュータサイエンスは数学的思考や手法に大きく依存しているので，コンピュータサイエンスの大学課程は数学にその基礎をおくべきである」[18]とされる。

つまり，プログラミングに関わる教育を，その数学的論理的構造のみに注目して教えようとしている。『カリキュラム68』の提案は，「既存の学問分野に対してコンピュータサイエンスが独立した学問分野であり，これを専門に教育研究する学科を独立して作りそこで教える内容がどのようなものであるかを明らかにしたものである。それから20年たった現在の時点でこれを眺めれば，数学的な面が豊富なのに対して，計算機分野については不十分の感があるのはやむを得ない。」[19]

カリキュラム78

『カリキュラム78』は，すべてのコンピュータサイエンス大学学部課程に共通するコアカリキュラム8つと，上級レベルで開講する選択科目10科目に分かれる。8つのコアカリキュラムのうち，5つは初級レベルであり，3つは中級レベルである。

5つの初級レベルは次の通りである。

CS1. コンピュータプログラミングI
CS2. コンピュータプログラミングII
CS3. コンピュータシステム入門
CS4. コンピュータ組織論入門
CS5. ファイル処理入門

以下3つは中級レベルのコア科目である。

CS6. オペレーティングシステムとコンピュータアーキテクチャI
CS7. データ構造とアルゴリズムの解析
CS8. プログラミング言語の構成

以下は上級レベルの選択科目である。

CS9. コンピュータと社会

CS10. オペレーティングシステムとコンピュータアーキテクチャII
CS11. データベース管理システム設計
CS12. 人工知能
CS13. アルゴリズム
CS14. ソフトウェアの設計と開発
CS15. プログラミング言語理論
CS16. オートマトン，計算可能性と形式言語
CS17. 計算数学：解析
CS18. 計算数学：線形代数

『カリキュラム 78』においても，数学はこれらの科目と深い関連をもつと考えられている。だから，微積分学や確率や線形代数を学ぶことが奨励されている。しかし，前提科目を見てみると，数学の科目とコンピュータサイエンスの科目は，ほぼ独立に学習できるようになっている。例外はCS16であり，これは『離散構造』という数学の科目を予め学習することが前提されている。少なくとも，コア科目のCS1からCS8までの科目は，他の分野の科目に依存しないという意味で，独立した学問の成立を示しているように思える。いわばコンピュータサイエンスの学問としての独立宣言が行われている。

もちろん，「ほとんどのコンピュータサイエンス科目では前提として特定の科目をあげてはいないものの，数学はコンピュータサイエンスのカリキュラムにおいて不可欠なものである。」[20]しかし，「課程内のコンピュータサイエンス関連部分を学ぶにつれ，学生はごく実践的なレベルから始め，次第に概念的，理論的な内容へと進んでいく」[21]というような仕方で，カリキュラムを組むことができるようになったのが，進歩であろう。

さて，『カリキュラム 68』と『カリキュラム 78』を比べる[22]と，数学的論理学的な部分が表面に出ることが徐々に少なくなってきている。そのため，B3. 離散構造入門，B4. 数値計算，I6. スイッチング理論などは，ひとまとまりの科目として『カリキュラム 78』に現れることはなくなった。これらは，もともと科目としてある程度発展しまとまった分野になっていたのであるが，コンピュータサイエンスの中の独立した科目とは認められなくなったのであろう。I7. 順序機械は，有限オートマトンを扱う科目であるが，『カリキュラム 78』では，コア科目からはずれることになった。また，A4. システムシミュレーションとA6. コンピュータグラフィックスは，『カリキュラム 78』では，「特別トピック」という，学科に余力があるときに開講するものとなっている。結局，数学を指向する科目が減ったことが特徴といえる。

また，『カリキュラム 78』に新しく現れてきた科目は，CS9. コンピュータと社会がある。そして，CS14. ソフトウェアの設計と開発は，A8. 大規模情報処理システムと一見似た面を含む。しかし，CS14は大規模ソフトウェアを扱い，しかもチームプロジェクトを行うという面でA8とは相違する。「ソフトウェア工学は，ACM カリキュラム '68 が発表された当時には存在せず，ACM カリキュラム '78 が発表された後になって研究が進んだ。」[23] なお，付言しておくとCS5. ファイル処理入門は，IBM System/360 という特定の機械を念頭においている。そのため，CS5は，CS4とCS6に解体すべきだという見解が存する[24]。

結局，『カリキュラム 78』に至って，実践的レベルからの教育という観点がとられ，大規模ソフトウェアという複雑な対象を扱うにつれて，数学的な科目の比重が減ってきたことは見て取れるだろう。

カリキュラム 91

『カリキュラム 91』は，9つの専門分野と，社会的，倫理的，職業的問題，およびオプションとしてプログラミング言語入門から成る。

AL　アルゴリズムとデータ構造
AR　アーキテクチャ
AI　人工知能とロボティクス
DB　データベースと情報検索
HU　人間とコンピュータの情報伝達
NU　数値計算と記号計算
OS　オペレーティングシステム
PL　プログラミング言語
SE　ソフトウェア方法論とエンジニアリング
SP　社会的，倫理的，職業的問題
PR　プログラミング言語入門

まず，『カリキュラム 91』において初めて出てきた科目は，HU. 人間とコンピュータの情報伝達，というヒューマンインタフェースに関わる学問である。これはユーザインタフェースとコンピュータグラフィックスを含む。後者は，実は『カリキュラム 68』の A6 という科目と同じ題名である。ただし，A6 は，応用システムプログラミングと科学的応用プログラミングを専攻する人のための上級レベルの選択科目として用意されていた。それに対して，『カリキュラム 91』においては，HU は，共通必修科目と見なされている。コンピュータを人間に分かりやすく使いやすくする方法であるヒューマンインタフェースが，必修科目の一つとしてこのように重視されてきたのは，注目に値する。また，AI. 人工知能とロボティクスや SE. ソフトウェア方法論とエンジニアリングの両者は，『カリキュラム 78』において既にそれに対応する科目は存するが（人工知能は『カリキュラム 68』に既に上級科目として含まれていた），『カリキュラム 91』において初めて，共通必修科目としてコア科目の位置を得ることになった。つまり，これらの科目は学問としてのコンピューティングを専攻する人はすべて学ぶべき科目という位置づけを得ることになった。

また，NU. 数値計算と記号計算において，「数値計算」がカリキュラムのコア科目となったのは，『カリキュラム 68』以来のことである。しかしそこで費やす時間は，『カリキュラム 68』では，それに並列する初級のコア科目と同じかやや多いくらいの時間である。それに対して，『カリキュラム 91』では，例えば，AR の 8 分の 1 程度の時間をかけるにすぎない。（実は，NU は共通必修科目の中でも一番時間数が少ないのであるが，それより少し多い程度の時間を使う科目としては，HU, DB, AI, SP がある。これらの科目に比べて残りの科目は，3〜8 倍の時間を使って教育している。）数学については，『カリキュラム 91』では，「数学と科学の理解はコンピューティングを中心とする学生にとって重要である」[25]と一言述べられているに過ぎない。応用や問題解決に「科学」を学ぶことが役立つということと並べて，数学はコンピューティングのいくつかの基礎的な話題を習得するために本質的であると言われている。数学の役割は，『カリキュラム 68』に比べて，（基礎的役割を果たすことはもちろんであるが）相対的に低くなっているようである。

実験と実習について

『カリキュラム 91』（『カリキュラム 88』も含めて）において特徴的なことは，実験，実習について特に取り上げて論じていることである。それ以前のカリキュラムにおいては，実験や実習に必要な時間数の指摘はあったが，それの意義については特にふれていない。どのカリキュラムも「学問」としてのコンピューティングを目指していたのだが，実験や実習を含めて「学問」として位置づけようとしたのは，『カリキュラム 88』（これは，『カリキュラム 91』のプロトタイプである）の提案である。以下，ACM の提案を少し離れて，実験と実習の役割を見ることにしよう。

情報処理にかかわる実験については，大学教育で行われる化学実験や電子工学実験や医学部の解

剖実習などの実験と違った性質をもつと言われている[26]。つまり，これらの実験においては，「すでに知られたことの再現，過去の科学者が行った実験の再体験」[27]が行われている。つまり，忠実に指導書に従って実験が行われる。いわば碁や将棋において定石を学んだり，絵画において手本を模写するようなものである。これによって，代表的な測定器具の使用法を習得したり，データの整理法を学び，実験結果を報告書としてまとめることを通じて実験の論理を学ぶ。

それでは，マニュアル指導型の実験をすることがなぜ重要なのか。恐らく，囲碁や絵画を考えても分かるように，具体的な例では非常に多様性が生じる。物理学の実験においても，同じ実験をしているはずなのに，得られたデータは非常に相違することがある。様々のエラーやノイズが生じる可能性がある。ここに実験技術の習得ということの意義があるのだろう。つまり，実験手順を厳密に同じにするということが，実験をやるうえで非常に重要になるだろう。「再現可能性」ということが，普通の科学の実験の客観性を保証するのである。

さて，情報分野での実験を考えると，ハードウェアの実験は，電子工学科の実験と共通するものが多い。しかし，アーキテクチャ関係の実験，プログラムの演習は，それとは違う側面をもつ。「「簡単な計算機を設計して，プロトボード（かロジックトレーナ）で試作し，動作を確認せよ」というような課題が与えられる。その指導書には，その計算機をどう作ればよいかは書いてない。ここが，上にのべた実験と違ってくるところである。伝統的な実験では指導書には「どうするか」がことこまかく指定してある。そういう手順を理解し，覚えることも目的に含まれるからである。しかし，いま述べた情報関係での実験では，「何をすべきか」という仕様はあっても，どう作ればよいかは指定されていない。ここに大きな違いがあるといえよう。これはプログラミングやソフトウェアになるとより顕著になる。プログラミングの課題は，こういうプログラムを作成せよという，仕様しか与えられていないのが普通である。」[28]

これに似たものは，建築での設計，都市計画，工業デザイン，機械設計，美術の課題制作，音楽の課題作曲のようなものである。

これらに共通する点が6つ挙げられている[29]。

(1) 課題とか，設計目標を指定するがその具体的な実現部分は学生に委ねられる。

(2) いずれも講義などでだいたいの作り方などの指導を受けているが，その課題に対してどのようにやればよいかを書いた指導書があるわけではない。

(3) 過去の実験の再現，追体験ではなく，かりに結果的には類似性のあるものになるにせよ，教官側も学生側も新規性や独自性，独創性が生まれうるものであることを意識している。

(4) 実験は理論との照合をしてうまくいったと判断することがあるが，ここであげたような場合唯一の基準解があるわけでなく，解は無数にある。

(5) 一般に結果の評価は単純ではない。

(6) 一般に教育手法はあまり確立されていない。

つまり，一つの尺度だけでは判断できない「総合的」な課題に関わっている。これは，さきほど述べた科学実験とは異なり，設計と基本的に類似するいわば工学的実験だといえるだろう。プログラミングにおいては，「再現可能性」はある意味では自明なことになる。誰がプログラムしても，同じやり方をすれば，同じ結論が得られる。この意味で，実験をやるときにノイズやエラーは基本的には考えられない。(例えば，自然科学では自然環境そのものがエラーを生じる原因となる。そのため，実験の方法をすべて教えておいても，実験結

果がうまく得られるとは限らない。)情報処理実習では，要求を仕様にまとめあげ，それを具体的にプログラムに書くことが問題である。過去の手法を知ることは重要であっても，そこから先のことが実験には要求される[30]。その意味で「総合的な価値判断のできる能力を身につける」ことが，実習を通して要求されているのである。この意味で，工学的な実験をやっているのである。

『カリキュラム88』における基本的考え

ここでは，コンピューティングを学問として成り立たせるための3つのパラダイムが提案されている。

「そのパラダイムの第1は，理論（Theory）である。その源泉は数学にあり，次の4段階を経て，首尾一貫し，筋のとおった理論へと導くものである。

(1) 研究の対象を特徴づける（定義）
(2) 対象間に存在すると思われる関係について仮説を立てる（定理）
(3) それらの関係が確かに存在するかどうか確かめる（証明）
(4) 結果を解釈する

これが，(誤りや整合性の欠如が発見されたような場合には）繰り返される。

第2のパラダイムは抽象化（abstraction）である。その源泉は実験科学の方法に見いだされ，次の4段階を経て，現象の解明に導くものである。

(1) 仮説を形成する
(2) モデルを構成し，予測を立てる
(3) 実験を設計し，データを集める
(4) 結果を解析する

これが（モデルの予測が実験結果と合わないような場合には）繰り返される。モデリングとか実験とか呼んだほうが適切である可能性もなるが，ここでは計算機分野ではふつうそういわれているという理由によって，「抽象化」と呼んでおくことにする。

第3のパラダイムである設計(design)は，工学に源泉があり，次の4段階を経て与えられた問題を解くシステムないし装置の構築に導くものである。

(1) 要求を述べる
(2) 仕様を述べる
(3) システムを設計，製作する
(4) システムをテストする

これが（テストの結果，システムが要求を充分に満たしていないことが知れたような場合には）繰り返される。」[31]

ここでは，3つのパラダイムがあるということが重要である。「理論は数理科学の基盤である。応用数学者はみな，これなくしては科学の進歩はありえないと信じている。抽象化（モデリング）は自然科学の基盤である。科学者はみな，科学の進歩は主として仮説を立て，モデル化のプロセスを組織的に遂行し，それらを検証，評価することによって起こると信じている。同様に設計は，工学の基盤である。技術者はみな，進歩の源は問題を設定し，設計のプロセスを組織的に遂行することによって，それを解くシステムを作り上げることにあると信じている。」[32] そのそれぞれのパラダイムが，数学，科学，工学の基本的な方法論を示している。そして，コンピューティングでは，この3つのプロセスが密接に絡み合っており，そのためにどれか一つだけを基本的なものとして取り上げることはできない，とされる[33]。

抽象化と設計というパラダイムを取り上げたことが，実験や実習を特に取り上げた理由になっているのだろう。そしてもちろん，ここで「工学」とはいっても，通常の物理法則にかかわる，シリコンチップの物性などは問題にされない。あくまでも，情報の処理にかかわる限りでの工学が問題になっている。

この小論では，コンピューティングという学問において，工学という側面が徐々に意識されてきたということを確認することが目標であった。しかし，もちろん哲学的な問題はここから生じる。つまり，数学や科学においては，学問というものが，伝統的に認められていた。その意味で理論と抽象化に関しては，これらの方法によって学問が成立してくることは恐らく認められるだろう。問題は，「設計」にある。この方法論を使うことが，学問ということにどういう意義をもつのか，その詳細な究明は，今後の課題になる。

ただし，「ソフトウェア工学」がどう位置づけの変化を被ってきたかを見ると，一つの手掛かりが得られる。さて，「1960年代に入って，多人数を要する大規模なソフトウェア開発が行われるようになってきたけれども，開発の当初に意図したようにはシステムが実現できず，その性能，信頼性，納期，費用などの点で多くの問題が顕在化しつつあった。」[34]こういう問題状況の中から，ソフトウェアの開発や保守に関する技術や管理の方法の考察が始まり，ソフトウェア工学が提唱されるようになった。問題は，一人で作ることのできない規模のソフトウェアの開発に関わる。大規模な複雑性の処理が問題である。まず，発注者が頭の中に描いているものと，受注者が発注者の要求として理解したものとが一致していなければならない。大規模なソフトで開発に携わる人が多い場合には，これは単純な問題ではない。またプログラムの検証の理論も，プログラムの規模が大きくなるとほとんど無力であると言われている[35]。そのため，検証とテストも，大規模なプログラムの場合には難しい。このように，数学や論理学においては，あまり問題にされなかったことが，大規模な複雑性に直面することによって，問題として取り上げられて，ソフトウェア工学が成立したように思われる。

そして，大規模な複雑性に対処するための学問を作り上げることが，『カリキュラム91』を特徴づけるものだと思える。

IV. ダイクストラをめぐる議論

数学から工学へという移行をめぐって，その論点を深めるための興味深い議論が行われている。

1989年2月，ダイクストラ（Edsger Dijkstra）は，『計算科学を本当に教えることの残酷さについて』という講演を行い，ACMの『カリキュラム88』に対して問題を投げ掛けた。これに対して何人かの同僚と討論することになった[36]。ダイクストラは，数学的証明を中心にして，コンピュータサイエンスを理解しようとしている。しかし，それに対して何人かが反論している。この論点の理解は，『カリキュラム88』の論点を明確にするためにも重要である。

ダイクストラの問題状況の把握は「計算機は我々の歴史において過激な新規性（radical novelty）を表している」という論点である。我々は，問題を解決しようとして，普通は，過去の経験から知ったものを使い，さらにそれとの類似を探ろうとする。「自分の過去と集積した経験とその中で形成された習慣」を使った問題解決は，しかしながら，「過激な新規性」に出会うとその効力を失う。「コンピュータは過激な新規性を表す」，この観点に立って，計算科学教育をダイクストラは考えようとする。コンピュータは2つの過激な新規性を表す。我々は大きくて複雑なものに対処する方法を知っている。分割統治の方法である。つまり全体を部分の集まりと見て，各部分を別々に扱う。そして部分が大きすぎれば，この手順を繰り返す。ここに分解によって生じる階層の深さは，全体の寸法と究極の最小部分との比によって理解される。そしてプログラマは，「巨大な比率が一つの技術によって橋渡しされている唯一の規範と専門を

もった独特の職である。」1ビットと数百メガバイトの比10^9を，ハイハイする赤ん坊と超音速のジェット機の比10^3と比べると，「巨大な比率」が実感できる。

「第二の過激な新規性は，コンピュータが我々には初めての大規模なディジタル装置であるということだ。」力学的法則に従う機械は，たいていの場合，少し力を加えると，少し働き，更に力を加えると，更に余分に働く。連続関数に従って働くという意味で，通常の機械は基本的にはアナログ装置とみなすことができる。それに対して，「ディジタル装置」は，1ビット変化しても（＋と－の変化など），劇的な結果を引き起こすことがある。

要するに，「コンピュータが我々にできることのだだ一つのことは，記号を操作して，このような操作の結果を作り出すことである。我々の先の観測から，これは離散的な世界であり，さらには，関連する記号の数として実行される操作の量は我々の想像よりもオーダにして何倍も大きいものであることを思い出さねばならない。」

さて，プログラムを実行するのが，機械としてのコンピュータの役割である。そのとき，プログラムを式と見ることが重要である。つまり，計算科学を形式数学と応用論理の方向に位置づけるべきだと，ダイクストラは主張する。ただし，計算科学は，形式的方法の効果的使用に関心があり，しかも普通の数学の式よりもずっと大規模な式を扱うという点に注意しなければならない。ダイクストラは，計算科学をVLSAL（Very Large Scale Application of Logic: 大規模応用論理学）と見なそうとする。

基本的なアイディアは，大きな集合について何かを示そうとするときには，その要素を個別的に扱って，しらみつぶしの探索をするよりも，一般化された定理を証明する方がいいということである。「形式的な構文規則と意味論を定義する証明規則をもったプログラミング言語は，形式的なシステムであり，それに対してプログラムの実行が一つのモデルを与えるだけである。」こういう意味で，プログラマはプログラムをかくときに，形式的な証明を与えることもやるべきである。これが，プログラミングコースが形式数学のコースの一つだというダイクストラの主張である。

この主張に対して，パルナス（David L. Parnas）は，工学教育において，設計と設計の証明に数学的方法が使用されていることを述べる。「設計を導き，検証するのに形式的方法を用いる能力は技術者の本質的な特徴の一つである。」このように，パルナスはダイクストラの主張を認めつつ，更に次の言葉を付け加える。「しかし，工学教育は同時に，学生に数学的方法の限界も理解するように教える。」もうすこし詳細にパルナスの論点を見てみよう。「我々は，厳密な数学的モデルでは手に負えないような問題を扱う方法を教えられた。最も大事なことは，(a) 数学的な解析のチェックを行うこと，(b) 数学的モデルが調べている実際の装置に十分なモデルであることをテストすることの重要性を教えられた。だれしも，数学を使っていて間違いをしでかすし，装置が数学的モデルを反映しないような特性をもつことを見いだすことも珍しいことではない。」

もちろん，このテストによってエラーがないということは決定できない。しかし，「適切に設計された統計的テストは，コード中に残るエラーの確立に関する情報，あるいは，使用中に起こる故障の確率の情報を与えることができる。我々が住む不完全な世界では，このようなデータはたいへん重要である。」テストの代わりに数学的検証がとってかわるのではなく，両者は補完的であり，両方とも必要である。

シャーリス（W. L. Scherlis）は，問題解決においては，多様な手段をもち，どの手段が状況に合っ

ているかを選択することが重要だとする。この意味で，非形式的な操作的直観を用いることもありうる。例えば，大きなシステムの設計者は，より高いレベルのスケールで一種の連続性「粗い連続性」を追求する。この意味で，ソフトウェアシステムを不連続なものとして管理しないようにする。また，ダイクストラの言うように，形式的方法は重要ではあっても，それは記号操作だけが問題ではない。形式化，「この世の中から実際の計算問題のために記号的に表現して構造を与えること」が，問題である。

エムデン（M. H. van Emden）は，「仮に我々が常にプログラムを形式的に規定してそれらが正しいことを証明する段階に到達したとしよう。その場合，我々はソフトウェアが抱える問題の小さな部分を消去したにすぎない。大きな問題は，我々が何を望んでいるか正確に言うのが難しいことである」とのべ，ダイクストラの形式的方法の限界を指摘する。複雑なシステムにおいては，「我々が望んでいることを本当には知らない」。それを発見する一つの方法は，プログラムを走らせることである。

ハミング（R. W. Hamming）は，「ダイクストラが，プログラミングは数学に似ており，数学とはユークリッド流のものであると信じている」ことを最大の問題だと考えている。「プログラムを実行する前に正しいことを「証明」するという彼の考えは，ペーパーマシン上のペーパープログラムには適用できても現実の機械にはできない。」「工学においては，設計の提案と現状の実践で何ができるかの間には，「もちつもたれつ（give and take）」の関係がある。」そのために，書かれたプログラムの正確な記述から始まることはできない。「ソフトウェア保守」というのは，故障した部品を修理するという意味ではなく，現在のソフトウェアを変化する要求と環境に合わせることなのだ。

リチャード・カープ（Richard M. Karp）も，ダイクストラの形式的方法を問題にする。まず，ダイクストラは数学的推論における形式論理の役割を強調するが，それは「数学的真理が発見される方法あるいは実際にコミュニケートされる方法とほとんど関係がない。」また，プログラムは，形式的な式にすぎず，その実行は形式的変換規則の応用に過ぎないとダイクストラは言う。しかし，形式的証明は，ある特別なプログラムにおいては不可欠の役割を演ずることはあるが，大多数のプログラミング問題ではそんなことはない。この論点を，大規模ソフトウェアシステムの設計で経験を積んだ同僚のことばを引いて示している。彼らは，大規模プログラミングが事前に規定済みのアルゴリズムの発見ではなく，プログラムの振る舞いについて我々が予測できないことがあると論じている。

テリー・ウィノグラード（Terry Winograd）は，計算機やプログラマや技術者がやっていることをダイクストラが誤解していると論じる。プログラマの仕事は，「彼の意図するプログラムが等価な形式機能仕様に一致する形式的証明を与える」ことだとダイクストラは言う。しかしこのためには，手近なその仕事に適切な形式機能仕様を作るという仕事を誰かが終えていなければならない。そしてまた，「工学的規範は，機能を果たすための信頼できる装置を作成することに関係している。」「技術者は，適度な費用で適度な努力で適度の信頼できる装置を作ることを可能にするような，たくさんの技術・道具・過去の設計を用いて設計作業に取り組んでいる。」「健全な技術は，完全な結果を保証するものではない。」つまり，非常に多くの複雑なプログラムが，信じられないほどの多様な状況下で，形式的証明の恩恵なしに動いているという事実を忘れてはいけない。

このような彼らの問題点の指摘に対して，ダイクストラは，数学の将来像の可能性について，非常にオプティミスティクな返答をしている。

以上の議論を私なりに評価すると，ダイクストラに対する「反論」こそが，工学化に向かう傾向を明確にしている，と言うことができる。もう少し詳細に反論者の主張を理解しよう。

例えば，パルナスの重要な論点は，我々が不完全な世界に住んでいることの確認である。この状況認識は，数学的証明という問題解決の方法以外に，テストという経験的な問題解決の方法も必要だということを示している。

また，シャーリスの論点を少し敷衍する。形式化をどのようにすればいいかという問題は，形式化が完了した上で形式をどう操作して証明するかという問題とは全く違っているということである。数学的な記号操作とは違う，形式化の問題が重要になってくる。

エムデンの論点は，シャーリスの論点と似ているが，シャーリスの方は仕様からプログラムの設計の段階を扱っているのに対して，エムデンは要求から仕様の段階への移行を問題にしている。

ハミングの論点の前提にあるのは，設計がアイディアの単純な実現とはいえないということだろう。つまり，現実に合わせて修正を加えることが，「設計」の中心課題だという考えである。これは，エムデンの「我々が何を望んでいるか正確に言うのが難しい」という論点と結びつき，モデルの変更，仕様の変更をどのようにしてうまくやるかということが，形式的証明以上に，解決を要する大きな問題なのである[37]。

V．人工知能研究が示す工学的観点

「ほかの自然科学が外界の客観的・観測可能な対象，あるいはそこに存在する理論的構造を対象とするのに対して，情報科学は人間の頭脳の中での抽象的概念と思考を対象とするところに大きな特徴がある。情報科学は，究極的には人間の脳の働きを理解し，その機能を機械によって実現することを目指すといえる。このような情報の扱いは通常の機械では難しく，シンボル操作ができる計算機の出現により初めて可能となった。」[38] この意味で，情報科学は物質やエネルギーではなく特に「情報」を扱おうとする限り，シンボル操作のできるコンピュータと関わり，より一般的には人間の脳の働きと結びつく機能に関わる。つまり，コンピータサイエンスにおいても，人工知能という分野は，その意味で焦点を当てて考察すべき分野だと言える。

さて，コンピューティングの学問性が意識されるのは，一方では初心者を導くためのカリキュラムにおいてであり，他方では研究の最先端であろう。この節では，日本の人工知能研究の最先端の研究を概観するために，ここでは『AI マップ』を取り上げる。これは，社団法人　日本人工知能学会の学会誌である『人工知能学会誌』においてなされた，その指導的研究者による人工知能の全体像を示そうとする企画である。特に，「研究の結果よりも，プロセスと背後の思想」[39] に重点を置いた論述を第一級の研究者が行っているところに意義がある。

人工知能は，人間の知性を心理的，哲学的に理解するのではなく，いわば科学的に理解する試みだと考えられている。しかし，特に「科学的」というよりも，「工学的」であることが重要だと私は考える。

恐らく，1980 代初頭から始まる第五世代コンピュータのプロジェクトと同期して生じた人工知能フィーバーが一段落した 1990 年あたりから，研究に対する反省や見つめ直しは起こってきたのであろう。例えば，1991 年に堀浩一は，アインシュタインとエジソンという典型例のうち，人工知能

研究者は，エジソンであるべきで，「偽物小物アインシュタイン」になるべきでないと述べている[40]。

以下，『AIマップ』のシリーズの中で，多くの研究者の強調する論点を取り上げることにする。全般的に見て，人工知能がトイワールドでのみ成功して，現実の問題ではうまくいっていなかった，ということが問題意識の中心にあると思われる。

大須賀節夫は，『AIマップ－AI研究者のあり方』で，次のように述べている。

「萌芽期におけるAI研究は，認知科学的な立場から観察された人間の知的能力を研究対象とし，それを記号表現として定式化することによって機械化を図るというアプローチをとってきた。要素研究の常道として，ここで必要なことは，定式化する範囲をうまく限定し，それをできるだけ単純な数学的あるいは論理的モデルで表すことであった。このようにして相互にあまり関連をもたない多数の研究成果が蓄積されてきた。現実の問題がたまたま要素研究の範囲に入る場合は別にして，これまでのAIは実問題をこのレベルにまでブレークダウンすることは含めていない。これは今日のAI研究の多くが，実用性については十分に考慮していないままに進められてきたことを意味する。なぜなら，実用性の観点から最も重要なことは，現実に生じる極めて複雑な泥くさい問題を適切に整理し，個々の問題を適正規模に分割していくことであり，これが成功すればその中で問題を処理することは相対的には容易なことだからである。もし問題を分割することによって個々の問題の規模を十分小さなものにしてしまうことができたなら，その中ではあらゆる可能性の組み合わせを作っていくという原始的な方法でも解は求まる。今日のAIはより重要な分割問題には触れず，問題は分割され整形されていることを前提として内部の処理方法を議論する研究が多い。これは一見実用化を目的としているように見えるシステムにおいても同様である。例えば，自然言語処理や定理証明，あるいは分野を限定したエキスパートシステムなどは（解決法の難易とは別に）問題としては認識しやすく，したがってAI研究の対象となってきた。しかし，今日，指摘されるのは，現実問題に対したときのこれらの成果のもろさである。」[41]

大須賀は，トイワールドを扱うのでなく，実用性を考えるために，問題の分割に焦点をあてて考えようとする。「実用性」を考慮するためには，範囲をうまく限定することが重要だと考える。それをうまく限定しさえすれば，特に数学的，論理的モデル化をしなくても，問題が解決できる。その意味で，実用性を求める工学は，理論化とは違った側面の考察をする必要がある。彼の論点は，このように理解できる。

『AIにおける科学革命』という論文で，北野宏明は次のように述べている。

「古典的人工知能論は，いくつかの理想化された前提に基づいている。それらは，以下のようなものである。

・問題解決をするのに十分な知識を専門家が知っている。
・それらの知識は，シンボルとして表現可能である。

また，古典的人工知能の手法は，完全な（Complete, Consistent, Correct）データと入力を前提としており，不完全なデータに関しては非常にもろいものである。不完全なデータとは以下のようなものである。

・情報が欠落している（Incomplete）
・一貫性がない（Inconsistent）
・誤りを含む（Incorrect）

しかし，実世界では，ほとんどの情報は不完全であり，理想化された世界で動作するシステムが，実世界でも動作する保証はない。むしろ，ほとん

どはうまくいかないのである。さらに，必要とする知識が，記号として表現可能であったり，専門家が明示化できる知識として認識している保証もないのである。このため，古典的人工知能の枠内で何か実証的研究を行おうとすると，古典的記号体系で取り扱えるようなモデル化をする必要に迫られてくる。トイワールドの誕生である。現在までの人工知能研究の主流の方法論は，極めて限定されたドメインで，非常に複雑な処理を行い，それが一定の成功をおさめたら，ドメインを広げていこうというものである。例えば，WinogradのSHURDRUやYale時代のRoger Schank一派の研究のように，限定された世界での言語理解を達成したら，その方法を実世界に応用して広い領域にかんする自由な自然言語入力を理解するシステムを作ろうというものである。このような方法は，現在でも主流である。しかし，この方法の問題点は，限定された世界から実世界に移行する際のスケールアップの難しさ，さらに，情報の不確定さの劇的増大に対応できるかというところにあり，限定されたドメインで検証された方法が根本的に役に立たない場合が多いのである。……このように，理想化された世界と情報の完全性を前提とした古典的人工知能は，破綻をきたしている。」[42]

北野は，『実世界における知能—「AIマップ」へのコメントに対する回答』[43]においてAIの対象とする問題を分類する。1. 線形分離可能問題（対象問題が記述要素の線形結合で記述可能な問題），2. 線形近似可能問題（問題は非線形であるが，実用的には，要素の線形結合で近似可能な問題），3. 非線形問題（本質的に非線形性が高く，線形近似では実用的にならない問題），4. 非平衡環境問題（システムが運用される環境が非常に不安定であり，単純なパラメータ最適化では，対応できない問題）。北野の言う「実世界」は工学的にこのような問題のすべてに関わる分野を含むものと捉えられている。しかも，「実世界ではほとんどの情報は不完全」であるという特徴をもっている。このような世界は「古典的記号体系で取り扱えるようなモデル化」をすることが困難である。この意味で，通常のモデル化，理論化のできない複雑な世界を工学（としてのAI）は扱わねばならないのである。

さて，『AIマップ—自然言語へのアプローチ』での長尾真の発言にも注目すべきである。

「抽象化することが学問であり，抽象化するとその適用範囲が広がり，学問的にも理解しやすく，理論としての体裁をなしているように一見感じられ，皆そちらのほうに流れ，そちらの肩を持ちたくなるが，実は現実を詳しく記述する能力は落ちているのである。本質論的説明，あるいは悪い言葉でいえば第1次近似のレベルで対象を記述し，説明するにはそれでよいかもしれないが，工学的に具体的な個々の対象を取り扱おうとすると，これではまったく役に立たず，第3次近似的な世界を築かねばならないのである。理学的立場は気楽であるが，それでは現実に対して責任をとる工学的立場とはなれないのである。そういった意味で，理学より工学は難しいのである。」[44]

さらに，『コメントに対する回答』で，長尾は彼の論点を敷衍している。

「文科系科学における理論はいわば対象の第1次近似のようなものであるのに対し，工学が要求するのは第2次近似，第3次近似の世界なのです。言語だけでなくパターン認識の世界においても，ある理論でやれば（ほとんど，どの理論を選択するかにかかわらず）まず70～80%の正答率まではいくのですが（第1次近似の世界），それを90～95%にまでもっていくにはその理論をきたなく修正したり，他の原理を持ち込んで複合的にするか，狭い個別場面ごとに理論を詳細化しなければなりません（第2次近似の世界）。ところがさらに99%

を狙おうとすると(第3次近似の世界)，これはまさに大変なことで，人間の持つあらゆる知識を総動員しなければならないといったことになります。」[45]

長尾の主張の注目すべきところは，抽象化し学問的に理解しやすくなった「理論」は，工学的には役に立たないという主張にある。つまり，このような理論は，「現実」を詳しく記述する能力は落ちているのである。工学は具体的な個々の対象を取り扱い，現実に対して責任を取ろうとする。それに対応する理論は，たとえあったとしても，「きたない」泥臭いものである。工学は，この意味で複雑な世界を扱い，それに対処するために第3次近似的な世界を築かねばならない。少なくとも，理学とは違った意味で理論が評価されていることは，理解される。

以上3人の論者[46]の主張をまとめてみると，人工知能において，ある意味で，理論的，数学的な形式化の問題とは違ったところが，基本的に研究しなければならない焦点だと言っている。全く表面的ではあるが，それが彼らに共通する論点と言えるだろう。いわば，工学的な問題設定を意識することこそが，人工知能研究の要点になると言われているのである。

VI. 最 後 に

以上概観したように，カリキュラムの分析とそれに続くダイクストラに対する反論からは，コンピューティングが取り扱わねばならない問題が，数学的論理学的問題とは違ってきていることが理解されるであろう。つまり，形式化が完了した上での問題設定である数学的な問題設定の枠組みでは扱えないタイプの問題こそが，解決さるべき問題の中心をなしているのである。どのように形式化すべきかとか，我々自身が計算間違い等のミスを犯す可能性があるということ，またモデルの修正が必ず必要になるためにその修正をうまくやるにはどうすればいいかといったことが，実際上解決しなければならない問題の核心になる。

また，人工知能研究者は，理論的に分析がうまくいくトイワールドを扱うのではなく，現実の世界を扱わねばならないと主張する。つまり，本質を説明する理論を求めようとするのではなく，きわめて複雑な泥臭い問題を解く必要がある。結局，科学的なスッキリした説明で満足することはできない。工学的に解決すべき問題は，「現実」という複雑な対象を，「我々人間」という誤りを避けられない者が，できるだけうまく解決しなければならないものなのである。これこそ，理学とは異なる工学の（泥臭いとも言われる）試みであろう。

「計算機分野は，数学および工学に深く根を張っている。数学は解析を，工学は設計を，そこにもたらした。計算機分野はそれ自身の理論と実験方法と工学をもっている。この点計算機分野は，たいていの物理的科学において，科学とその発見を応用する工学が（たとえば化学と化学工学，というように）別物とされているのと対照を示している。この分野において科学と工学が不可分なのは，この分野の科学的および工学的パラダイムが，根本的な相互関係をもっていることによる。」[47]

設計することがどういう意味で学問と結びつくのか。また，エレガントさや論理的正当化にあまりコミットせず，しかも学問的であることが，どうして可能なのか。論理的正当化に関与する度合いは，数学，自然科学，工学と，徐々に減っていくように見える。「実用性」「実世界」「現実」に関わろうとする工学は，自然科学の合理性に満足することはできない。つまり，自然現象や人工現象を「うまく説明できる」理論を合理性の典型とみなすことはできないのである。にもかかわらず，全くのカンや個人的技能といった非合理的な側面を強調することも，工学の進歩を目の当たりにし

ている現代人としては納得できないところがある。すると最後に問題になるのは，工学的根拠づけがどのような意味で，学問的と言えるかということである。この問題の解明は，科学を典型としてきた伝統的な知識観に代わる，新たな知識観を示すことになると期待される[48]。

注

1) 「振子時計やテンプ時計の発明者で，波動力学でも功績のあるオランダの物理学者ホイヘンス（Huygens）は，実はchronology（時計学）で学位を得ているが，時を計るということを中心として成立していた時計学は，その後ばらばらに分解されてほかの学問分野に吸収されてしまった。また，今世紀最大の産業の一つを形成する工業活動の生産製品として自動車があるが，自動車学科（department of automobiles）といったものを保有する大学としては，アメリカ内では知名度の低い大学に二つあるだけだということである。このように，役には立つが，しかしながらコンピュータに関連するということが唯一の統一原理で集められたcomputer scienceの諸科目は，早晩ほかの学科に吸収されてしまうであろう」p. 318『アメリカの大学における情報処理教育』山田尚勇 in『bit別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版 1993年1月

2) 60年代のアメリカでも，コンピュータサイエンスは数学の一種であり，大学院生でなければ教えられないという風潮があった。p. 476『創造科学としてのコンピューティング：障壁を取り除く』アンドリース・ヴァンダム in『bit』Vol. 22, No. 5 1990/5 を参照

3) Computing as a Discipline,by Peter J. Denning (Chairman), Douglas E. Comer, David Gries, Michael C. Mulder, Allen Tucker, A. Joe Turner, and Paul R. Young, Comm. ACM, Vol. 32, No. 1 (Jan. 1989), pp. 9-23『学問としての計算機分野』ピーター・デニング等　木村　泉訳　『情報処理』Vol. 31, No. 10 (Oct. 1990), pp. 1351-1372

4) 「情報科学は，物理化学的な法則，つまり現実の世界に存在する個々の物質の性質には束縛されないで，純粋に抽象化された概念，抽象的なモデルについて論ずる抽象的な学問であり，その点で，実在の世界を扱う物理学とは一線を画するものである。」p. 211『岩波講座　情報科学－1　情報科学の歩み』高橋秀俊：「かつて「コンピュータはハードウェアとソフトウェアとから成る」と言われ，コンピュータに関する学問分野をハードウェアを主とするものとソフトウェアを主とするものに分ける試みがあったが，今日の学界では，アルゴリズムと計算機アーキテクチャの概念を欠いた学問やハードウェアそのものを対象とする学問は計算機分野とは見なされていない」p. 10『大学等における情報処理教育のための調査研究報告書』社団法人　情報処理学会　大学等における情報処理教育検討委員会　平成3年3月

5) 『岩波情報科学辞典』長尾真等編　「情報科学」の項　1990年

6) 『岩波情報科学辞典』長尾真等編　「計算機科学」の項　1990年

7) 『カリキュラム91』p. 154 in『bit別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版　1993年1月

8) p. 1『岩波講座　情報科学－1』情報科学の歩み』高橋秀俊

9) p. 2『岩波講座　情報科学－1』高橋秀俊

10) p. 3『岩波講座　情報科学－1』高橋秀俊

11) p. 4『岩波講座　情報科学－1』高橋秀俊

12) p. 6『岩波講座　情報科学－1』高橋秀俊

13) p. 17f.『岩波講座　情報科学－1』高橋秀俊

14) p. 220『計算機の歴史　パスカルからノイマンまで』ハーマン　H. ゴールドスタイン　共立出版　参照

15) p. 76『岩波講座　情報科学－1』高橋秀俊

16) 「計算機科学ないし工学は，情報を記述し変換するアルゴリズム的プロセス（その理論，解析，設計，効率，実現，および応用を含む）の，系統的な研究を内容としている。計算機分野全体の背後に横たわる基本的な問いは，（効率よく）自動化できるものとはどんなものかというものである。この分野は1940年代初頭に，アルゴリズム理論，数理論理，およびプログラム内蔵計算機の発明の3者が結びついて生まれたものである。」p. 1360『学問としての計算機分野』ピーター・デニング等　木村泉訳　『情報処理』Vol. 31, No. 10 (Oct. 1990)

17) p. iv『プロローグ』國井利泰 in『bit別冊　コン

ピュータサイエンスのカリキュラム』國井利泰編　共立出版 1993 年 1 月
18) p. 17『カリキュラム 68』in『bit 別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版　1993 年 1 月
19) p. 31『大学等における情報処理教育のための調査研究報告書』社団法人　情報処理学会　大学等における情報処理教育検討委員会　平成 3 年 3 月
20) p. 114『カリキュラム 78』in『bit 別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版　1993 年 1 月
21) p. 113『カリキュラム 78』
22) 「カリキュラム 68 は計算機科学というものを学長とか，学部長とか，評議員とかいった人々に売り込むためのものだったのに対して，カリキュラム 78 は，今や（米国では）当たり前のものとなった計算機科学科のカリキュラムの最大公約数を求めるためのものであった」とも言われている。p. 204『計算機科学の発想』紀華彦　日本評論社 1981/5
23) p. 105『大学等における情報処理教育のための調査研究報告書』
24) p. 44ff.『大学等における情報処理教育のための調査研究報告書』
25) p. 172『カリキュラム 91』in『bit 別冊　コンピュータサイエンスのカリキュラム』國井利泰編　共立出版
26) 『情報処理における実験・演習』都倉信樹 in『情報処理』Vol. 32, No. 10, 1991/10 参照
27) p. 1102『情報処理における実験・演習』都倉信樹
28) p. 1103『情報処理における実験・演習』都倉信樹
29) p. 1103『情報処理における実験・演習』都倉信樹
30) 「数学や物理学のコースでは，典型的な一つか二つのやり方しか，問題解決のし方がありません。そして問題解決のし方は文字どおり，以前の何百万人の学生と同じかたちで解決していくわけです。しかしコンピュータ・サイエンスにおいては，二人の人が同じ問題を，同じやり方で解決するということはありません。自分のアイディアを使ってどのように問題を構造化し，解決していくかを決めていくわけです。問題は大ざっぱに記されているだけです。細部には言及されません。ですから，学生たちは問題解決の前に，問題を具体的なものにしていかなければなりません。それがコンピュータ産業の特徴なわけです。」p. 482『創造科学としてのコンピューティング：障壁を取り除く』アンドリース・ヴァンダム in『bit』Vol. 22, No. 5 1990/5
31) p. 1352f.『学問としての計算機分野』ピーター・デニング等　木村泉訳　『情報処理』Vol. 31, No. 10 (Oct. 1990)
32) p. 1353『学問としての計算機分野』ピーター・デニング等
33) p. 1353『学問としての計算機分野』ピーター・デニング等
34) p. 105f.『大学等における情報処理教育のための調査研究報告書』
35) p. 108『大学等における情報処理教育のための調査研究報告書』
36) Edsger W. Dijkstra, David L. Parnas, William L. Scherlis, M. H. van Emden, Jacques Cohen, Richard W. Hamming, Richard M. Karp, Terry Winograd: "A DEBATE on teaching Computing Science", Communications of the ACM, Vol. 32, No,12, 1989, pp. 1397-1414,『計算科学教育に関する討論』(前編) エズガー・ダイクストラの問題提起『bit』1991/4 pp. 772-783,『計算科学教育に関する討論』(後編) ダイクストラの主張への同僚の応答『bit』1991/5 pp. 903-916　この節の引用は，特に断らない限り，この邦訳からのものである。
37) 「抽象数学に成果をあげている大数学者が工学における難問に解答を与えてくれるには，数学以前の工学的問題を理解してかからなければならないが，これは大変な努力と時間を必要としよう。逆に工学の難問を解こうとしている人々が近代数学を理解して何かの手がかりにしようとするのも望み薄であろう。それは工学上の難問は数学的には汚い問題で，数学も線形の問題なら素晴らしい数学的成果があるが，非線形の問題はたいていどうにもならないのに似ている。」p. 77『情報工学教育のゆくえ』室賀三郎 in『bit 臨時増刊　情報工学の教育・研究』1980/12
38) 『岩波情報科学辞典』長尾真　等編　「情報科学」の項　1990 年

39) p. 443『AIマップについて』西田豊明，奥乃博，寺野隆雄，堀　浩一，田中穂積 in『人工知能学会誌』1992/5 Vol. 7 No. 3

40) p. 785『編集後記』堀　浩一 in『人工知能学会誌』1991/9 Vol. 6 No. 5

41) p. 797『AIマップ－AI研究のあり方』大須賀節雄 in『人工知能学会誌』1992/9 Vol. 7 No. 5

42) p. 744f.『AIにおける科学革命』北野宏明 in『人工知能学会誌』1993/11 Vol. 8 No. 6

43) p. 74『実世界における知能－「AIマップ」へのコメントに対する回答－』北野宏明 in『人工知能学会誌』1994/1 Vol. 9 No. 1

44) p. 533『AIマップ－自然言語へのアプローチ』長尾　眞 in『人工知能学会誌』1994/7 Vol. 9 No. 4

45) p. 681『コメントに対する回答』長尾　眞 in『人工知能学会誌』1994/9 Vol. 9 No. 5

46) その他「AIマップ」には，『AI辺縁における諸問題』福村晃夫 (1992/5)，『AIマップ－ロボットから見たAI』辻　三郎 (1993/7)，『ロジックプログラミング』淵　一博 (1994/3)，がある。このうち，福村と辻は，本文で論じたのと同様の「現実」意識を示している。また，『AIマップ－辻　三郎先生へのコメント』松原　仁 (1993/9) は，フレーム問題をキーワードとして設計の原理（単なる説明と対比される）について深く考察することを通じて，「実世界」こそがターゲットにされねばならないことを説得的に述べている。ただし，淵は第五世代コンピュータに関わる問題を主題にしているため違った論点になっている。しかし，淵の論文に対するコメントの中で，雨宮真人は，同様の「現実」意識を提示している。

47) p. 1360『学問としての計算機分野』ピーター・デニング等

48) "The Foundations of Artificial Intelligence" eds. by Derek Partridge and Yorick Wilks Cambridge U. P. (1990) での多くの論考は，哲学的な議論（機械は知能を持てるか）に陥らず，AIにおけるプログラムの役割，プログラムと理論との関係，合理的再構成といった地に足のついた問題の解明を行っている。例えば，ニーダムの「AIは何ら特殊な方法論をもっていない」という論点がある ("Is there anything special about AI?" Roger M. Needham)。彼は，AIと化学工学とを比較し，実際上の相違はないとする。そして，我々が本質的に工学的なものに進むべきであり，基礎に悩むのをやめるべきだ，と主張する。この論点がどれほど納得のいくものであるかはともかくとして，プログラミングを基本的な手段とする人工知能研究が，どういう根拠に基づいているかを更に究明することは，興味をそそる問題である。