コンピュータソフトウェア史の現状と課題

吉田晴代

「Mapping the History of Computing ---Software Issues（コンピュータ史研究の現状と課題を探る---ソフトウェア問題を中心に）」というテーマの国際会議が2000年4月にドイツのPaderbornで開催された[1]．最近20年間で目覚しく進歩したコンピュータとその利用に関する歴史研究の中で，コンピュータソフトウェア史研究は立ち遅れている．プログラム言語といった個別分野の歴史研究，1950年代のSAGE防空プロジェクトの回顧といった研究はあっても，最近のソフトウェアの発展を反映し，ソフトウェアの重要性と歴史の全体像が理解できるような研究はない．そこで，ソフトウェア史の研究の現状と今後の研究課題について再検討し，研究の主題と方法を探求することが，会議の目標とされた．コンピュータの一般ユーザーにとっても，ソフトウェアをいかに使いこなすかだけでなく，その本質や意味，歴史についても学ぶ必要があるという点で，研究への社会的要請も強調された．科学史，技術史，社会学，経済史など他分野の方法もとり入れた包括的なソフトウェア史を目指して，1) 科学としてのソフトウェア；2) 工学としてのソフトウェア；3) 信頼できる人工物としてのソフトウェア；4) 労働過程としてのソフトウェア；5) 経済活動としてのソフトウェア；6) 博物館における歴史的ソフトウェアの収集と展示と，6つの分科会が設けられた．コンピュータ科学者やソフトウェア開発の実務家の見方と歴史学者や社会学者の見方双方を総合して，研究体系をつくるという方針から，コンピュータ科学者，企業など現場の実務家とOB，それに歴史家と社会学者が参加した．以下，数学史に関わりが深いと思われる2つの分科会「科学としてのソフトウェア」と「信頼できる人工物としてのソフトウェア」の報告の一部を紹介したい．

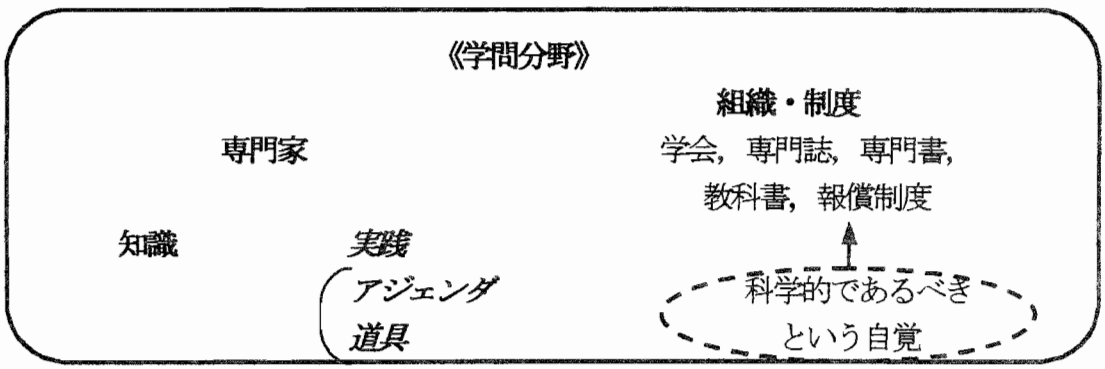
1. 科学としてのソフトウェア

この分科会の報告者マイケル・マホーニイ（プリンストン大学歴史学部）は近世数学史の専門家であり，フェルマー研究の単著がある[2]．科学史研究者の中でコンピュータ史研究のパイオニアの1人として，理論コンピュータ科学の形成や技術史として見たコンピュータなど活発に論文を発表している[3]．「科学としてのソフトウェア---ソフトウェアとしての科学」と題する報告で，3つの問題を論じた．第1に，理論コンピュータ科学（科学としてのソフトウェア）の形成とその研究手段としてのアジェンダ（agendas），第2に，理論コンピュータ科学とソフトウェア工学と

の関係，第3に，ソフトウェアとしての科学（コンピュータを研究手段として利用する科学）から見た理論コンピュータ科学だ．

1-1. 理論コンピュータ科学の形成と「アジェンダ」

科学の本質は知識と実践，つまり科学者の共同体が共通に何を知り何を行なうのかにあると考える．そうすると，科学の形態は，グループが違い，時代や場所が違えば異なることになり，学問的であると同時に社会的な性格を持つ．そこで，科学の実践に焦点を当て，「アジェンダ（agendas）」[4]という概念を用いると，コンピュータ科学のような新生の学問分野の形成をうまく理解できる．アジェンダとは，その学問分野で「何を行なうべきか？」ということであり，「何が問題か？問題の優先順位，問題を解く道具（手段・方法）は何か？何が解か？」といったことから構成される．従って，その学問分野で専門家であるということは，アジェンダを知り，その実行を助け，アジェンダを確立する能力を持つこと，つまり何が問題かを知り，アジェンダを拡張・再形成するような仕方で問題を解いたり，深い問題を提出することだ．アジェンダとアジェンダを実行する道具は不可分であり，道具はアジェンダを体現する．アジェンダに関するマホーニイのこの説明をもとに図式化すると，おおよそ次のようになる．



理論コンピュータ科学は1955年から1975年の間に独立した学問分野として形成された．電子工学から言語学まで様々な分野出身の専門家が関連するアジェンダの小さな集まり（例．オートマトン，形式言語，計算量，形式的意味論）を形成し，その集まりが核となって理論コンピュータ科学を形成した．これらのアジェンダは1955年以前にはなく，専門家のグループが元々属していた専門分野のアジェンダとは異なる新しいアジェンダに合流することで形成された．彼らが出身分野からどんな道具を持ちこみどう適用したか，それによって新しいアジェンダはどのようにして形成されたかを明らかにすることが研究課題となる．こうしたアジェンダへの収束の各点で，新生の分野は親に当たる分野からどんな道具を獲得したか，道具は元の分野で何のためにつくられたか，コンピューティングに利用することで新分野

の形成にどう寄与したか，適用の結果その道具はどう作り直され元の分野における位置に磨きをかけるのにどう役立ったか，といったことが問題になる．理論コンピュータ科学におけるアジェンダはどう形成されたのか，マホーニイは，自分のこれまでの研究に基づいて2つの事例[5]を説明しながら，研究課題を提示した．

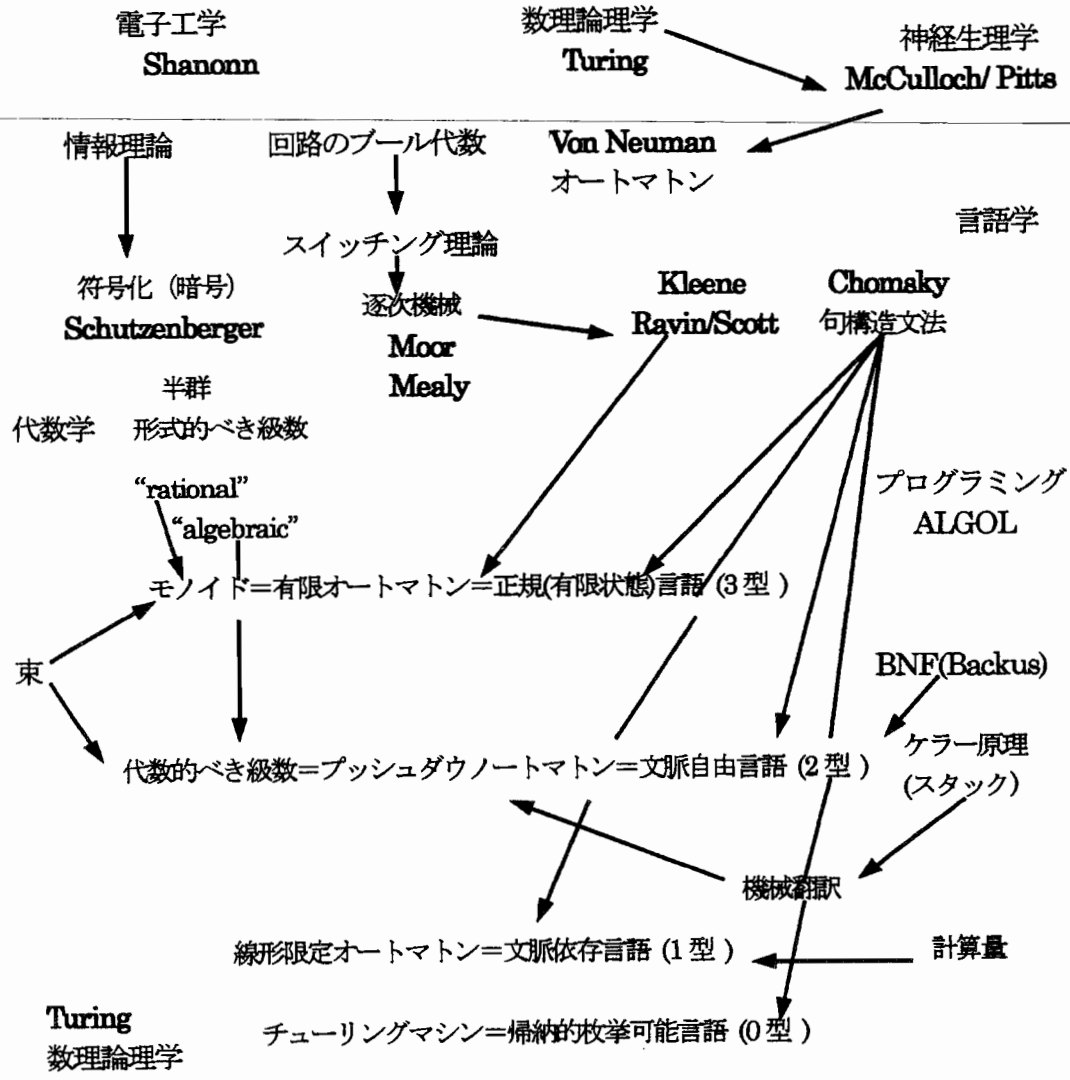


図1．オートマトンと形式言語のアジェンダ（実行計画）

《例1．　オートマトンと形式言語のアジェンダの形成》（図1）　コンピュータ科学は独自のアジェンダを持たずに出発した．出発点は科学理論ではなく物理装置としてのコンピュータだった．コンピュータ科学にとって親にあたる数理論理学は計算可能性の問題（すなわち，時間・空間が無限でもコンピュータに何ができないか）を明らかにし，スイッチング理論は基本演算のために回路の総合・分析の手段

を提供した．だが，有限なランダムアクセス式メモリーを持つ有限なマシンに何ができるか，それを実行するにはどうしたら良いかを明らかにする理論はなく，それが必要とされた．重要なアジェンダの収束の例として，形式言語とオートマトンの理論の形成がある．文法能力の数学的理論という言語学者チョムスキーのアジェンダと，代数学と数論を出発点とするブルバキ派数学者シュッツェンベルガーの数学的アジェンダがあった．ブルバキ数学の半群のような数学的基礎構造が道具として符号の数学的解析に役立ち，符号の問題は有限オートマトンと関係することが明らかになると，シュッツェンベルガーは数学的一般化を追求すると同時に，句構造文法では自然言語の研究に不十分なことを理解したチョムスキーと共同研究を推進した．その結果，チョムスキーの言語学のアジェンダ，機械翻訳のアジェンダ，代数的プログラミング言語のアジェンダが合流し，代数的べき級数とプッシュダウンオートマトンと文脈自由言語の等価性が明らかになった．この等価性は，Algol60 が文脈自由言語を構成するとわかると，コンピュータ科学の基礎となった．アジェンダはいろいろな場所で形成された．形式的べき級数とプッシュダウオートマトンと文脈自由言語を同一視することで，パリのシュッツェンベルガーの代数的符号理論からミュンヘンの F ・バウアー達の逐次形式翻訳に至るアジェンダが MIT において合流し，自動プログラミングの取組みに発展するという具合にだ．

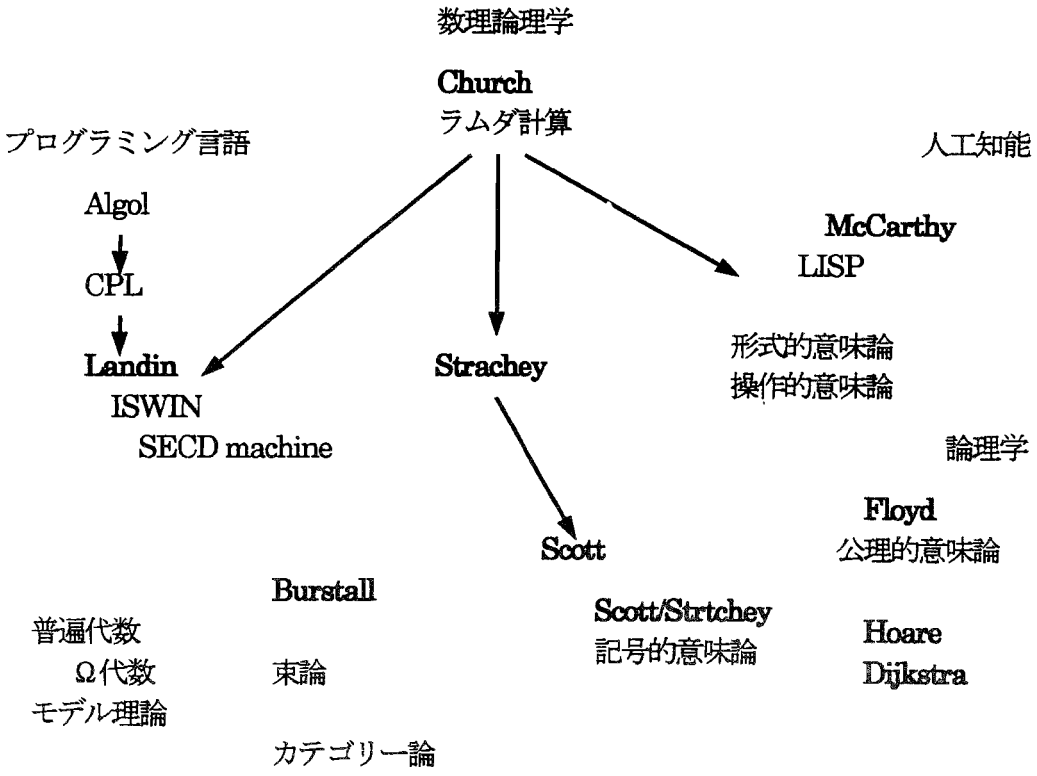


図2．意味論のアジェンダ

《例 2. 意味論のアジェンダの形成》（図２） もう 1 つの重要な例が，（普遍）代数，数学的論理学，プログラム言語，人工知能の相互作用による意味論のアジェンダの形成だ．λ計算は数理論理学の概念として 1930 年代にチャーチがつくった．1950 年代後半になって，人工知能を研究していたマッカーシーが，定理の機械証明とコンピュータによる常識的推論に関する研究の手段として復活させた．その後，統語論よりは意味論を重視した計算の数学的理論の基礎としても活用した．マッカーシーの形式的意味論のアジェンダ（λ計算は道具）はヨーロッパに伝わり，ケンブリッジのランディンやストラッチーの所に定着，さらにアムステルダムやウィーンのスコットなどにも伝わり，プログラム言語の意味論やプログラムの証明に関する研究で普遍代数とも結びついて，実り豊かな研究成果を生み出した．だが，マッカーシーの地元アメリカでは実践的な関心が支配的で，マッカーシのアジェンダには冷淡だった．

以上２つの事例の検討から以下の有用な点が明らかになるとマホーニイは言う．

(1) コンピュータ科学とそれへの数学の応用との関係は双方向的だ．数学はコンピュータ科学に数学的基礎を提供する一方，コンピュータ科学は数学の抽象的概念に「物理的」意味を与えたという具合に相互作用した．例えば，コンピュータ科学が，半群，束，圏といった最も抽象的な数学的構造に実際的な意味を与えた．コンピュータのことを考えてつくられたわけでもなく，数学においてさえ抽象的で役に立たないと言われたのに，コンピュータ科学に応用された結果，予想されなかった性質や関係が明らかになり数学研究を刺激した．λ計算により，意味論と普遍代数や圏論の間にも同様の相互作用が生じた．

(2) アジェンダ形成の初期には関心や方法に地域性が強く反映した．研究グループが異なれば，教育も好みも構想の混ざり具合も違い，違いには文化的および制度的背景の違いが反映した．そこで，地域的グループがどのようにして共通のプロジェクトに合流し，新しい学問分野としてのアジェンダを形成していくのか，実践がいかに移動していくのかが研究課題になる．

アジェンダの形成は，研究助成の獲得[6]や教育過程の編成といった，学問的実践のもっと異なる側面からも解明される．中でも注目されるのは，教育の側面から見ると，マホーニイも認めるように，アジェンダは，トマス・クーンの「パラダイム」概念[7]とほぼ重なることだ．その学問分野の専門家になるというのは，アジェンダを知ること，つまり何がなされるべきかを知ること，その分野の問題が何であり，どう取組まれ解かれるべきか，解は何かを学ぶことだ．これは，模範例とその解，つまり問題解決のモデルが次世代へと伝えられていくという，クーンのパラダイムの考え方とぴったり重なる[8]．だが，両者は全く同じというわけではない．パラダイムが同一学問分野における，例えば地球中心説から太陽中心説へといった転換（学問分野の発展とはどういうことか，それを促す要因は何か）を説明するのに適して

いるのに対し，アジェンダは，コンピュータ科学のような新生の学問分野形成を研究するのに適している．アジェンダと道具の組合せにより他分野との相互作用の中で新分野形成をダイナミックに捉えることができるし，またアジェンダ形成に焦点をあてることで，学問分野形成の社会的，政治的，経済的，文化的要因をも考慮して多面的な理解が可能になる．

1-2. 数学とソフトウェア工学

理論コンピュータ科学が，ソフトウェアの開発や利用といった日常の実践にどこまで役立っているのかということは，しばしば問題になる．ホーアやダイクストラのようなコンピュータ科学者は，これまでの科学技術における理論と実践の伝統的関係に従って，理論コンピュータ科学がソフトウェアの実践に対して基礎科学の役割を果たすことを期待した．だが，現状は理想から程遠く，両者の間には埋め難いギャップが存在する[9]．だが，歴史家から見れば，この一貫したギャップがあることこそ研究課題だ．マホーニイはこの問題にどう取組むか，いくつか研究の方向を示唆した[10]が，そのうちの１つを紹介したい．ソフトウェア開発のウォーターフォールモデルを取り上げ分析して，こうした理論と実践のギャップは，ホーアのようなコンピュータ科学者の理想を追求しさえすれば解決するものではないという，ソフトウェアの別の側面を明らかにしたのだ．

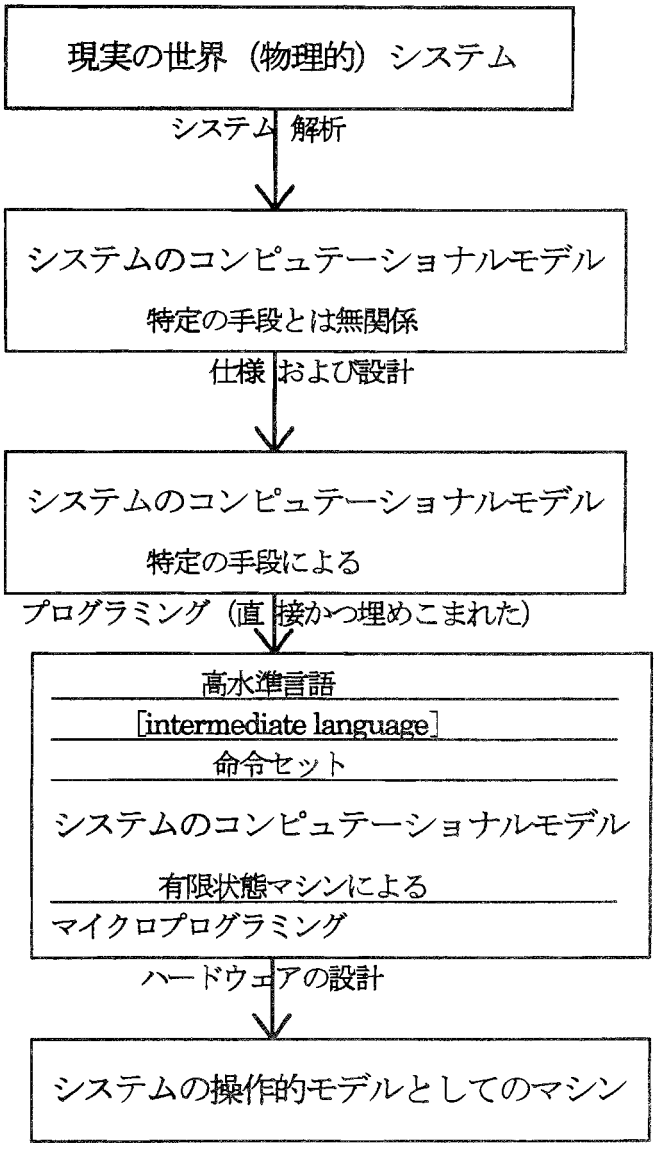


図.３．ソフトウェア開発：抽象の各段階

図３のモデルで図式の下半分は，数学的に最も良く理解できる部分，つまり理論コンピュータ科学の対象となる部分であり，ここで重大な誤りが生じることはほとんどない．モデルの図式上半分はソフトウェア工学が注目する，大量の

決定的誤りが生じる場所だ．この分析は，ソフトウェア開発の立場から，フレデリック・ブルックスがソフトウェア構築の「偶有的」作業と「本質的」作業と呼んだものに対応する[11]．アリストテレスの用法に従って，「本質的」とはソフトウェアの性質に固有なもの，「偶有的」とは実現するとき派生する付随的，副次的なものを意味する．ブルックスは，ソフトウェア生産性向上の可能性を探るためソフトウェア構築上の主要な困難を分析した．その結果，難しいのは何を表現するのか決定すること（本質的作業）であって，表現することそれ自体（偶有的作業）ではないとわかった．

マホーニイはこのブルックスの分析結果を，ソフトウェアとしての科学，つまり科学研究におけるコンピュータ利用の問題として読み替えた．するとこの場合にもやはり，ソフトウェア開発と同様，現実の世界の一部をソフトウェアつまりコンピュータを利用したモデルによってどう表現するのか，そのモデルが適正か否かといった問題が生じる．だが，それは，マホーニイが論じている科学としてのソフトウェアつまり理論コンピュータ科学の問題（つまり，プログラムが行なうべきこととして我々が書いたことをプログラムが実行していると我々はどのようにしてどの程度まで確かめられるのか）ではない．ソフトウェア開発者もしくはコンピュータを研究手段として利用する科学者にとって，対象とする現実世界の問題なのだ．この問題は，コンピュータの利用が大きく進み，科学のあり方にまで影響を及ぼすかのような現代科学にあっては，科学史研究の重要な課題だ．そうマホーニイは言う．

1-3. コンピュータ科学と数学

ソフトウェアとしての科学と科学としてのソフトウェア（理論コンピュータ科学）との関係が今述べたようなものであれば，理論コンピュータ科学とはどのような学問であるべきかという本質的問題に立ち戻ることになる．科学研究においてコンピュータを用いてモデルをつくることの意味が問われるからだ．

伝統的なモデル観では，モデルとは，より良く理解できるか直接操作可能なメカニズムもしくは数学的関係であり，モデルの目的は現象の働きを理解するのに役立つこと，モデルの優秀さの基準はモデルと観測された現象との経験的な適合度にあった．17 世紀以来の近代科学では，自然は物理モデルに還元され，物理モデルは数学的関係に還元された．その際，数学的関係は物理モデルの構造を反映し，物理モ数学は，定理，無限過程，静的関係を扱うものだが，コンピュータ科学の対象はアルゴリズム有限構成，動的関係デルは自然の構造を反映するものとされた．それに対して，フォン・ノイマンは，自然と数学とを物理モデルが仲介するという近代科学のモデル観を否定し，数学的構造自体が物理的世界でも社会でもモデルになるという新しいモデル観を主張した．科学者は現象にぴったりのモデルをつくりさえすれば良く，モデルが現実世界の本当の構造の反映であるか否かに関心を持つ必

要はないのだ.

> まず初めに強調しなければならないのは，もう前に聞いたことがきっとあると思うのだが，やはり繰返し述べておかなければならない言明だ．科学は説明しようとしなくて良い．解釈しようとさえしなくて良い．科学とはモデルをつくることだ．モデルが意味するのは，一定の言葉による解釈と一緒に，観測された現象を記述する数学的構造だ．そのような数学的構造の正当化は，機能するように期待されるということ，つまり正当な広さの範囲から現象を正しく記述するということだけだ．それに加えて，ある種の審美的基準を満たさなければならない．つまり，どのくらい多くを説明するかに関して，むしろ単純でなければならない．例えば，言葉を使うより，こうした漠然とした事柄を主張する方が価値があると思う．「単純」がいかに単純かを正確に説明することはできない．我々が採用する理論のあるもの，我々がそのおかげでとても幸福でそれを誇りに思っているモデルのいくつかは，初めてそれらに接した人々にとくに単純なものとして感銘を与えるようなことはなかっただろう．[12]

要するにフォン・ノイマンにとって，科学とはモデルをつくることであり，モデルとは観測された現象を正しく記述する数学的構造であり，審美的観点からモデルは単純であることが望ましいのだ．重要なことは，フォン・ノイマンは，伝統的モデル観を完全に否定したのではないということだ．モデルの数学的構造が解析学的に取扱い可能なものであり，研究者はモデルがどう働くか理解することは前提だった．

だが，この新しいモデル観がフォン・ノイマンを困らせることになった．当時の主流の数学つまり解析学では解けない非線形（偏微分方程式に関連した）問題を解くためにコンピュータを利用することになったからだ．問題はコンピュータの与える数値解がモデルの条件を満たすかということだ．数値解は，解析的モデルのように研究者に理解可能な構造を提供しない．解が予想に反したときどこに問題があるか確定するのが難しいし，打ち切りや丸めによる誤差などコンピュータを利用することで生じる独自の問題もあった．コンピュータに適した数値解法も研究された[13]が，それだけでなく新しい計算理論が必要になった．そこで，フォン・ノイマンはコンピュータの数学的基礎としてオートマトンの理論を考えたのだ．オートマトンに関する Hixon Lecture (1948) でこう述べた[14].

> 今日では，形式論理学，とりわけ数学に応用される論理学のとても入念に仕上げられた体系が存在する．このことは，学問分野として優れた側面でもあるが，同時に深刻な弱点でもある．これは，優れた側面を拡大する機会ではない．それを軽視するつもりもないが．だが，不充分さについてはこう言える．形式論理学を研究してきた者は誰でも，数学の技術的に最も手に負えない部分の一つであることを認めるだろう．その理由は，形式論理学が厳密な全か無かの概念を扱うからであり，実数や複素数の連続の概念，つまり数学的解析とほとんどつながりを持たないからだ．

だが，解析学は数学の中で技術的に最も成功した最も入念につくられた部分である．それゆえ，形式論理学は，そのアプローチの性格からして，数学の最も良く耕された部分から切離され，数学の最も困難な部分である組合せ数学に組込まざるを得ない．デジタル式の全か無かタイプのオートマトンの理論は，今まで考察したことから，形式論理学の一章であることは間違いない．その理論は数学的観点からすると解析学ではなく，組合せ論に属さなければならない．

ここでフォン・ノイマンが言いたかったことは，コンピュータを利用して現実世界のモデルをつくる前提として，新しい計算理論（コンピュータの抽象的な働きを数学的に理解する理論）が必要であり，それがオートマトンの理論だということだ．解析学で解けない問題をコンピュータを利用して解くというのであれば，コンピュータを利用する計算過程に関する厳密な数学的理論としてオートマトンの理論が必要なのだ（マホーニイはこれを解析学の失敗を穴埋めすると説明した）．オートマトンの理論は，チューリングやマカロック＝ピッツのような先行研究を考えると，形式論理学に属するように思えるが，解析学に頼らず，形式論理学（数学的に言えば組合せ数学）だけではオートマトンを十分理解することはできない．そこでフォン・ノイマンは，オートマトンの理論を「解析学の領域に引き戻したい」と考えた[15]．「組合せ論的ではなく，もっと（数学的な意味で）解析学的な数学的理論を得たいと思った」[16]のだ．コンピューティングの基礎となる新しい数学理論として，解析的モデルが研究者にとって理解可能なのと同じくらい理解可能（で厳密な数学的基礎に基づいた）な数学的モデルを提供できるオートマトンの理論が必要だ，そういうフォン・ノイマンのコンピュータ科学のアジェンダが継承され発展した流れを示すのが図４だ．この図で注目されるのは，研究のもとになる関心が一貫して，コンピュータによる数値解法，つまりコンピュータを利用して現実世界のモデルをつくるということにあり，プログラミングへの関心から生じたものでないことだ．図の中でアジェンダの発展の終点近くに位置するＪ・ホランドの次の言葉もそのことを示す．

> 数学は旅のこの部分にぜひとも必要だ．… 数学が決定的な役割を果たすのは，厳密な一般化かもしくは原理を定式化できるようにするからだ．物理学の実験もコンピュータに基づいた実験もそのような一般化を提供できない．物理学の実験は一般に厳密なモデルのための入力と制約を供給するだけだ．実験自体が演繹的な説明を許容する言語で記述されることはほとんどないからだ．コンピュータに基づいた実験は厳密な記述を持つが，特定の場合を扱うに過ぎない．よく設計された数学的モデルは，何といっても，物理学の実験，コンピュータに基づいたモデル，それに学問分野を超えた比較によって明らかにされる詳細を一般化する．… 数学だけが我々を十分遠くまで連れて行ってくれる．[17]

ここに示された数学的モデルの役割は，デカルト以来の数学像を反映している．「モ

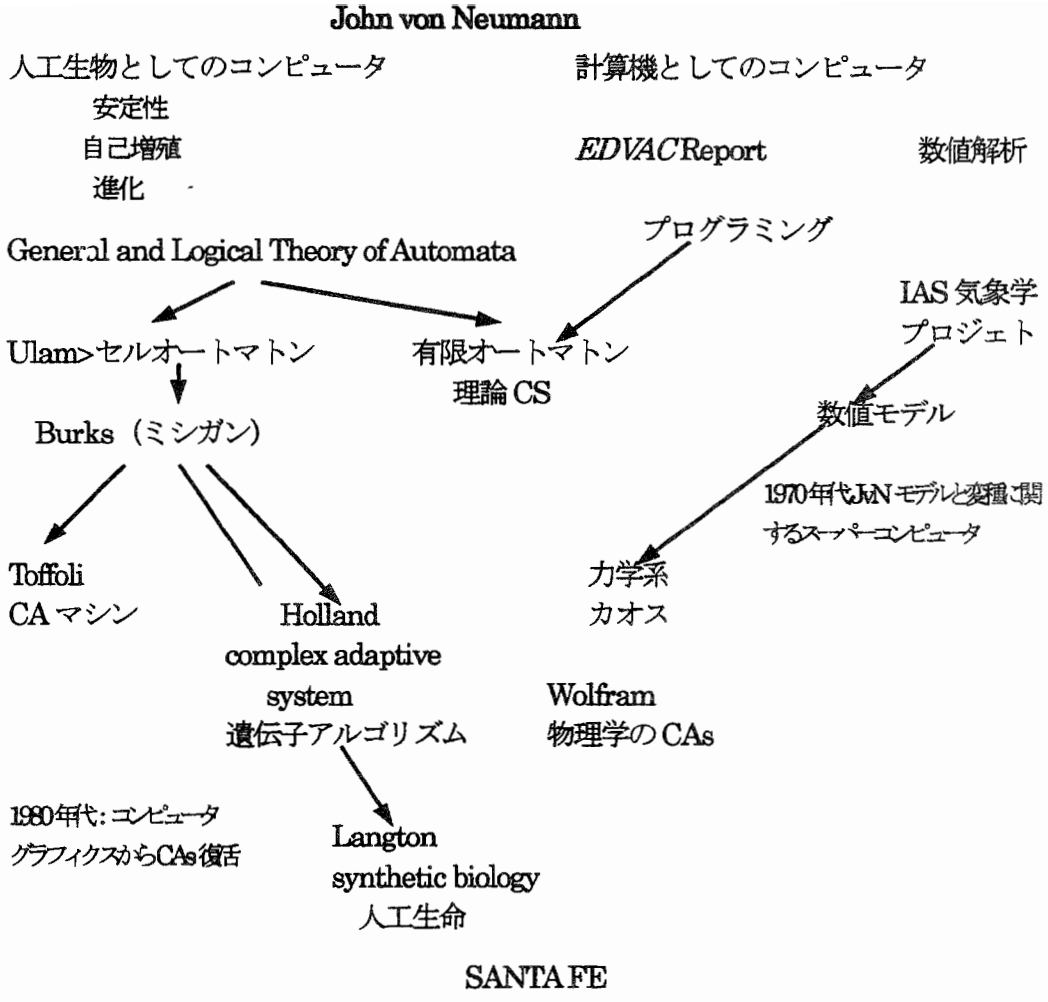


図 4. Von Neumann のコンピュータ科学のアジェンダ

デルの数学的構造が解析学で取扱い可能で，研究者はモデルの働きを理解できる」ことを前提しているのだ．解析の手段が解析学からコンピュータに変化したことで，「関心を持つ世界の部分を信頼性をもってシミュレートし，プロセスの解析と理解とを可能にする仕方でシミュレートする」モデルの問題となったのだ．世界を理解することがコンピュータの計算過程の理解に左右され，ソフトウェアが科学の象徴になろうとしている今，20 世紀後半にそれがどのようにして起こったのかが，間もなく科学史の大問題になる．そのとき「歴史感覚に優れた」ソフトウェア史は問題を解く助けになる．これがマホーニイの結論だ．マホーニイは，理論コンピュータ科学（科学としてのソフトウェア）の形成のみならず，コンピュータを利用する科学研究（ソフトウェアとしての科学）とそこから生じるモデルの問題を提起することで，科学史研究におけるソフトウェア史の意義を明らかにした．

2. 信頼できる人工物としてのソフトウェア

マホーニイが論じた第 2 の問題，すなわち理論コンピュータ科学がソフトウェアの開発や利用といった日常の実践にどこまで役立っているのかという問題を，別の観点から考察したのが，分科会「信頼できる人工物としてのソフトウェア」分科会の報告者で社会学者のドナルド・マッケンジー（エディンバラ大学社会学部）だ．マッケンジーは，技術の社会学，とくに技術の社会構成と呼ばれるグループの中でも今最も活発で注目を浴びている研究者の 1 人だ．マッケンジーはこの問題を「Sonnenbichl からの眺め：ソフトウェアとシステムのデペンダビィティに関する歴史社会学について」と題して，コンピュータソフトウェアの信頼性（デペンダビィティ）[18]の観点から考察した．コンピュータへの依存をますます深める現代社会にとって，差し迫った問題であるという点で，社会学者らしい優れた問題設定といえる．マッケンジーの考察の出発点は，「ソフトウェア危機」が診断され，解決策として「ソフトウェア工学」が提案された，有名な 1968 年のガルミッシュ会議だ．会議では，ソフトウェアの生産性向上だけでなく，人間の安全がコンピュータシステムに頼ることになったときに予想される深刻な事態も話し合われた．その意味でマッケンジーは，この会議を，「ソフトウェア史における自己反省の明確な契機として，信頼性の歴史に有用な出発点」と位置付けた．問題は，その後ソフトウェア信頼性の歴史はどう展開したかだ．マッケンジーの考察の基礎データは，1976 年から 1992 年に至るコンピュータが関係した可能性のある事故死の記録だ．総数約 1100 の事例を分析した結果．主要な原因（死の 90%以上）は人間とコンピュータの相互作用における欠陥であり，ハードウェアの欠陥（フォールト）が約 4%，ソフトウェアの「バグ（ソフトウェア設計の欠陥）」が 3%強つまり 30 件だ．ソフトウェアが原因とされもののうちに，放射線治療機や湾岸戦争中のパトリオット迎撃ミサイルシステムの事故も含まれる．要するに，ソフトウェア設計の欠陥による事故死はガルミッシュ会議で心配されたほどに多くない．

この分析結果から，ソフトウェア信頼性の歴史にとって重要な研究課題が浮かび上がる．なぜ，大衆注視の中で，ソフトウェア設計の欠陥が明白に関与したような破滅的な事故はまだ起こっていないのか，という問題だ．マッケンジーは問題を「ホーアのパラドックス」と名づけた．問題を深く考察したのが，歴史家ではなくコンピュータ科学者のＣ・Ａ・Ｒ・ホーアだからだ．ホーアは，ソフトウェア信頼性を達成する手段はソフトウェアを形式的推論の支配下に置くことだという，ガルミッシュ会議の精神の強力な推進者であり，コンピュータ科学者の中で厳密で明確な思考を重んじる形式主義者の 1 人だ．論文「コンピュータプログラミングの公理論的基礎」(1969) [19]で，自分の考えをこう説明した．「コンピュータプログラミングが精密科学であるのは，プログラムの性質全体と，所与の環境でプログラムを

実行した結果全体とが，原則としてプログラムのテキスト自体から純粋に演繹的推論によって求まる点にある.」1つのプログラムが達成しようとすることについて，形式的仕様をつくることができ，関係するプログラム言語をうまく設計できれば，プログラムの「正しさ」を定理として表すことができ，定理を証明することができるはずだ.「プログラムの証明」(つまり「プログラムの正しさの証明」) は，「理論的探求」であると同時に，プログラム設計の欠陥を防ぎ，欠陥が存在しないという信頼を達成するためにとても重要な実地の技術であり，従って信頼性への重要な貢献にもなる．これが，ホーアの信念だった.

ところが四半世紀後，ソフトウェア信頼性の問題を再検討したホーアが認めたのは[20]，電話交換システムや航空機の飛行を制御する現代のソフトウェアがかつてのソフトウェアに比べてはるかに巨大でありながらはるかに信頼度を増していることだ．だが，このソフトウェア信頼性の成功は，ホーアが主張してきた「プログラムの証明」のおかげではない．1990 年代半ばになっても，そのような証明が産業界で実行されることはほとんどないからだ[21].「ソフトウェアは証明なしにどうやってそれほどの信頼性を獲得したのか？」とホーアを悩ませた問題，それがマッケンジーの言う「ホーアのパラドックス」だ．証明なしでも現代のソフトウェア工学の実践が*機能*している理由についてホーアは考えた．コンピュータハードウェアの能力の増大と相対コストの減少のおかげで，プログラミングでかつて禁止されていたことができるようになり，誤りを予想した「防御的プログラミング」が実行され，設計とプログラムをそれらに直接責任を負わないチームメンバーに見直させることで誤りの検出に目覚しい成果を挙げるようになったことなどだ.

これに対して，「ホーアのパラドックス」を歴史家及び社会学者の目で見て，マッケンジーは，3つの研究課題を定式化した．第1に，デジタル電話交換やフライバイワイヤのような現実世界の信頼できるシステムの詳細な歴史を徹底的に調べることだ．それらのシステムでどんな欠陥が生じ，どれが誤りにつながり，誤りから障害が生じるのをどのようにして防いだか？システムの長い進化において信頼性はいかにして保たれ改善されたか？異なるシステムで信頼性に優劣があるのはなぜか？ を調べることだ．第 2 に，信頼性を達成する手段についてもっと視野を広げる必要がある．ソフトウェアの証明とは別に，もっとありふれた防御的プログラミング，ソフトウェア (開発の) 監査，リカバリーブロックの歴史，それにソフトウェアの見直しとテストの役割に注目すべきだ．それらは現実世界のシステムの信頼性に大きく役立った可能性が大きいからだ．第 3 に，信頼性への形式主義の影響を，単なる技術を越えたもっと広範なものと見るべきだ．ソフトウェアとはどのようなものでありあるべきかという考え方や文化の問題も含めてだ．マッケンジーが提起したこれらの問題は，コンピュータ・ソフトウェア史はもちろん，コンピュータ史でも今までほとんどとり上げられてこなかった.

それでは，マッケンジー自身はこの問題にどう応えようとしているのか？ それを知るよい手掛かりが，会議後の昨年出版された著書 *Mechanizing Proof---Computing, Risk, and Trust*[22]だ．著書の主題はコンピュータを利用したソフトウェアの証明であり，演繹的方法という人類の歴史に深く根付いた文化的伝統がコンピュータの利用によってどう変化しようとしているのかを問題にしている．だが，表題からもわかるように，それだけに止まらず，問題を，コンピュータへの依存を深める現代社会にとって決定的な問題であるコンピュータ信頼性の問題と結びつけた所に，マッケンジーの優れた問題設定能力が示されている．著書では，この報告でも取上げた 1960 年代のソフトウェア危機とソフトウェア工学を始め，人工知能と自動化された定理証明，コンピュータを利用した数学の定理の証明（4 色問題），米国国防総省の支援で発展したコンピュータセキュリティ研究，自動化されたソフトウェアの証明に対する批判，航空管制システムのプログラム証明のような実践における証明と現実世界の関係，機械化された証明の鍵を握る手段としての証明システムの発展やそういうシステムを証明に利用することの意味と問題点など，多彩な話題が取り上げられている．しかも文献のみならず関係者からの膨大なインタビューに裏付けられた，まさに現代コンピュータソフトウェア史の圧巻といえる．

ここで注目されるのは，最後の章でやはりコンピュータ関連事故の歴史とホーアのパラドックスを取上げていることだ．そしてホーアのパラドックスに対して社会学の立場から 2 つの説明を与えた．1 つは，ホーアのような「道徳事業家」の活躍が功を奏して，ソフトウェアの危険を警告することで結果的にリスクを減らしてきたというものだ．これは，逸脱の社会学，とくに 1960 年代以来のハワード・ベッカー等のラベリング理論に基づいた説明だ[23]．ラベリング理論では，逸脱，つまり社会的規則違反のような行為を，その前提となる社会集団が規則をつくり執行することから考察する．「逸脱とは，人間の行為の性質ではなく，他者によってこの規則と制裁とが『違反者』に適用された結果なのである．逸脱者とは首尾よくこのレッテルを貼られた人間のことであり，また逸脱行為とはこのレッテルが貼られた行動のことである．」この理論で，行動に対する反応および行動自体にとって重要なのは，それが（「(社会の) 聴衆」によって）逸脱とみなされるかどうかだ．マッケンジーによれば，これはコンピュータ信頼性の領域にもそのまま当てはまる．つまり，コンピュータ信頼性は絶対的な問題ではなく，どの程度の信頼度で十分かとか，どんな出来事が障害とされるのかを判断するのは，コンピュータシステムの「聴衆」なのだ．マッケンジーが格好の事例としてとり上げたのは，1994 年に起きたインテルのマイクロプロセッサーの「割り算バグ」事件だ．インテルは出荷前にバグを検出していたが，深刻な障害を生み出す可能性は小さいと判断した．ところが，ある数学者がこの欠陥を発見し，欠陥とそれに対するインテルの態度をスキャンダルと考え，インターネットのニューズグループで社会に広く知らせた．すると，別

の技術者がそれに応えて，普通のユーザーが自分のプロセッサーに欠陥があるかどうか簡単にチェックできる深刻な事例を考え出し，やはりインターネットを通じて警告した．おかげでうわさはインターネットを越えて，一般のマスコミに取り上げられるまでになり，インテルは製品の出荷停止と回収に加えて，株価低落と大きなダメージを受けることになった．ラベリング理論では，欠陥を告発した数学者や技術者を「道徳事業家」と呼ぶ．なぜなら「規則は自動的につくられるものでない．たとえ，ある種の活動が客観的な意味から集団にとって有害であるとしても，その害悪は発見され摘発される必要がある．人々は，何らかの手段を講じるべきことを思い知らされる．」コンピュータシステムの信頼性というのは一般原理であって，例えばこの場合インテルとユーザーとで異なるように様々に解釈可能であり，具体的な規則が自動的に生み出されるわけではない．何者かが問題の事態に大衆の注意を喚起し，規則（この場合，どこまで欠陥を許容できるか）の創設に導いていかなければならない．彼等を道徳事業家という．なぜ「道徳」というのかは，社会の道徳体系，つまり社会の善悪の掟に関わる問題だからだ．こうした「道徳事業家精神」は，ガルミッシュ会議やホーアのようなコンピュータ科学者の活動，最近では2000年問題をめぐる動向を始め，ソフトウェア信頼性の領域でもあちこちで見られ，重要な役割を果たしている．これがマッケンジー説明だ．

ホーアのパラドックスに対するマッケンジーのもう 1 つの説明は，知識社会学[24]のアプローチによるものだ．マッケンジーによれば，2000 年問題からも明らかになように，我々がコンピュータに望むことは，安全で確実であるだけでなく知ることなので，ソフトウェア信頼性は知識社会学の問題となる．知識社会学では，コンピュータのような人工物の性質に関する知識を 3 つに分類する[25]．第１に権威，つまり我々の信頼する人々が，それらの性質は何であるかを我々に話す．第２に 帰納，つまり人工物やシステムをテストし利用することで，我々がそれらの性質を学ぶ．第３に演繹，つまり科学理論などから我々がそれらの性質を推測する．知識社会学の重要な鍵は，事物の性質に関する知識は人々の性質に関する知識と完全には分離できないということ，つまり知識はそれを生産したり所有する人間に属していると考えることにある．プログラムの証明の場合で言えば，演繹は人間に簡単に理解できる短い推論の連鎖ではなく，自動化された検証システムに頼ることが多い．従って，知識の生産と所有はシステム（とそれを設計した人間）に移るので，その場合に知識の性格が変わるのかというのがマッケンジーの著書の主題だ．だが，ソフトウェアの開発・生産のような社会 - 技術的実践においては，コンピュータシステムの性質に関する知識のうち，帰納と権威に基づいた形式の方がより効果的であるように思われる，それがパラドックスに対するマッケンジーの第２の説明だ．

こうしたマッケンジーの説明には，不十分な所もあるかもしれない．だが，重要なことは，コンピュータソフトウェアの信頼性のように，一見技術的に見える問

題においても，様々な社会的ファクターや人間的ファクターが技術と複雑に相互作用していることだ．それを象徴的に示す事例が，マッケンジーの提示したホーアのパラドックスだ．これを，最初に話したマホーニイの報告と併せて考えて，こう結論したい．すなわち，理論コンピュータ科学やソフトウェア工学についてこれまでの科学や技術の閉じた概念によって，ソフトウェアの歴史を構築することはできない．2 人の報告は，もっと広範な社会的，政治的，経済的，文化的背景の中で問題をより総合的かつ多面的にとらえなければならないという，歴史研究の新しい方向を指し示すものだ．

註と参考文献

[1] 会議の報告集は今年 7 月に出版された．Ulf Hashagen, Reinhard Keil-Slawik, Arthur Norberg (Eds.), *History of Computing: Software Issues*, Springer, 2002.
[2] Michael S. Mahoney, *The Mathematical Career of Pierre de Fermat 1601 – 1665*, 2nd ed., Princeton University press, 1973.
[3] 主な論文を挙げると以下の通り．Michael S. Mahoney, "The History of Computing in the History of Technology," *Annals of the History of Computing*, 10 (2), 1988, pp. 113-125 (吉田晴代訳「技術史におけるコンピューティングの歴史」，『産研論集』，札幌大学経営学部附属経営研究所，23, 2000 年，pp. 67-83)；"Computers and Mathematics: The Search for a Discipline of Computer Science," J. Echeverria, A. Ibarra, T. Mormann (eds.), *The Space of Mathematics --- Philosophical, Epistemological, and Historical Explorations*, De Gruyter, 1992, pp. 349 – 363; "Computer Science: The Search for a mathematical Theory," J. Krige and D. Pestre (eds.), *Science in the 20th Century*, Harwood Academic Publishers, 1997, Chap. 31; "The Structures of Computation," R. Rojas and U. Hashagen (eds.), *The First Computers --- History and Architectures*, MIT Press, 2000, 17-32. マイケル・S・マホーニィ著，佐々木力解説，林知宏訳，「コンピュータ科学の形成」(上)・(下)，『UP』，2001 年 343 及び 344 号，東京大学出版会.
[4] 元々は「なされるべきもの (things to be done)」というラテン語の複数動名詞だが，英語では「なすべきもののリスト (list of things to do)」という意味の単数名詞だ．マホーニイは「なすべきもの多数の集まり」という意味で複数名詞を使う．注 2 の文献 (Mahoney, 2000)，p.20 より．
[5] これらの事例の詳しい内容については，注 2 の文献 (Mahoney, 1992)，(Mahoney, 1997)，(Mahoney, 2000)，及び (マホーニィ, 2001) を参照.
[6] マホーニイによれば，研究助成の問題は新生の学問分野のアジェンダ形成につな

がる重要なファクターだ．研究助成は単なる金の問題ではなく，資金援助者である政府機関（コンピュータ科学の場合，国防総省高等研究計画局や全米学術財団）と科学者のコミュニティの相互作用はアジェンダの形成に重要な影響を及ぼす．自分達がどのような学問的実践を追及しようとしているのかを資金援助者の要求や期待に応える形で明らかにすること，その際，他の関連学問分野（コンピュータ科学の場合，数学）からの評価も，新生の学問分野にとって重要だ．

[7] トーマス・クーン著，安孫子誠也・佐野正博訳『本質的緊張』第2巻，第12章「パラダイム再考」．

[8] そこで，マホーニイによれば，教科書や教育課程は新生の学問分野の発展を研究するための重要な資源となる．教育課程をつくるには，その学問分野がどんなものか，何が中心で何が周辺か，何が誰でも知るべき問題で何が選択科目かといった問題について学問分野内部で合意形成が必要なので，そこにアジェンダが反映する．歴史家から見れば，完成した教育課程のみならず，それをつくる課程もまた重要な研究対象なのだ．

[9] コンピュータ科学者のHoareは，「Hoareの原理」とも言うべき自分の学者としての信念と現実の乖離についてこう述べた．

我々の原理は以下の4つの見出しに要約される．

(1) *コンピュータは数学的機械である*．その働きのあらゆる側面が数学的厳密さで定義でき，どの細部も純粋な論理学の法則を用いて，この定義から数学的正確さで導かれ得る．

(2) *コンピュータプログラムは数学的表現である*．それらは，先例のない厳密さで，どんな微小な細部でも，それらが実行するコンピュータの働きを，意図されたものかどうかに関わりなく，記述する．

(3) *プログラム言語は数学的理論である*．それは，概念，記号，定義，公理および定理を含み，プログラマーが仕様に合ったプログラムを発展させ，プログラムが仕様通りに動くことを証明するのを助ける．

(4) *プログラミングは数学的活動である*．応用数学や工学の他の分野同様，数学的理解，計算，証明の伝統的方法を明確かつ細心に適用することが実践を成功させるために必要だ．

これらは一般的な哲学的かつ道徳的原理であり，私には自明なことに思える．そう言った方が良いのは，現実の証拠がそれらの原理に反するからだ．私が述べたようなことは現実には成り立たない．コンピュータについても，プログラムについても，プログラム言語についても，プログラマーについてもだ．(C. A. R. Hoare, "The Mathematics of Programming," in his Essays in Computing Science (Hemel Hempstead, 1989), 352., 講演は1985年)

[10] 1つは，工学としてのソフトウェアの側面，つまりアルゴリズムに解消されない

コンピューティングの機構，手段そして非決定論的（偶然的）に重要な側面に注目することであり，これは，社会的，政治的，経済的要素とも関係する．もう１つは，科学としてのソフトウェアの側面に注目することだ．数学が，定理，無限過程，静的関係を扱うものであるのに対し，コンピュータ科学の対象はアルゴリズム，有限構成，動的関係である点で，数学とコンピュータ科学は異なる性格の学問なのだ．

[11] フレデリック・Ｐ・ブルックス，Jr. 著，滝沢徹・牧野祐子・富澤昇訳『人月の神話 --- 狼人間を撃つ銀の弾はない』，アジソンウェスレイ，1996，第16章「銀の弾などない --- ソフトウェアエンジニアリングの本質と偶有的事項」．

[12] John von Neumann, "Method in the Physical Sciences," in *The Unity of Knowledge*, ed. L. Leary, 1955); reprint in John von Neumann, *Works*, VI, 492.

[13] ウィリアム・アスプレイ著，杉山滋郎・吉田晴代訳，『ノイマンとコンピュータの起源』，産業図書，1995，第5章．

[14] John von Neumann, "On a Logical and General Theory of Automata," in *Cerebral Mechanisms in Behavior---The Hixon Symposium*, ed. L. A. Jeffries (New York, 1951), 1-31; reprint in *Papers of John von Neumann on Computing and Computer Theory*, ed. William Aspray and Arthur Burks (Cambridge, Mass./London/Los Angels/San Francisco, 1987), 391-431 at 406.

[15] マイケル・Ｓ・マホーニィ著，佐々木力解説，林知宏訳，「コンピュータ科学の形成」（下），『UP』，2001年343及び344号，東京大学出版会，p.29.

[16]．ウィリアム・アスプレイ，前掲書，pp. 207－208.

[17] John H. Holland, *Hidden Order: How Adaptation Builds Complexity*, 1995.

[18] 用語の問題について言えば，会議の主催者がマッケンジーに依頼したテーマは，"Software as Reliable Artefact" だが，マッケンジーは "reliability" を "dependability" に取替えた．どちらも日本語では「信頼性」となって区別がつかないが，ソフトウェアの分野では伝統的に前者を使うようだ．マッケンジーは，1992年に出版されたIFIP Working Groupの "dependability" をめぐる用語法の研究成果Jean-Claude Laprie, ed., *Dependability: Basic Concepts and Terminology in English, French, German, Italian and Japanese* (Vienna: Springer, 1992)をもとに後者を採用した．

[19] C. A. R. Hoare, "An Axiomatic Basis for Computer Programming," *Communications of the ACM* 12 (1969): 576-83.

[20] C. A. R. Hoare, "How Did Software Get So Reliable Without Proof?" Presentation to Awareness Club in Computer Assisted Formal Resoning, Edinburgh, 21 March 1994; Hoare, "How Did Software Get So Reliable Without Proof?" typescript, December 1995, 3.

[21] 例えば，Donald MacKenzie and M. Tierney, "Safety-Critical and Security-Critical Computing in Britain: An Exploration," *Technology Analysis & Strategic Management* 8 (1996): 355-79 参照．

[22] Donald MacKenzie, *Mechanizing Proof--- Computing, Risk, and Trust*, MIT Press, 2001.
[23] ハワード・S・ベッカー著，村上直之訳，『新装アウトサイダーズ---ラベリング理論とはなにか』，新泉社，1993．とくに第1章「アウトサイダー」，第7章「規則とその執行」，第8章「道徳事業家」参照．
[24] マッケンジーが挙げた科学の知識社会学の入門書として，Barry Barnes, David Bloor, and John Henry, *Scientific Knowledge: A Socoological Analysis* (London and Chicago, 1996).
[25] Donald MacKenzie, "How Do We know the Properties of Artefacts? Applying the Sociology of Knowledge to Technology," in *Technological Change: Methods and Themes in the History of Technology*, ed. Robert Fox (London, 1996), 247-63.