特集論説

ツール化する情報技術と実践知

Information Technology as a Tool and Practical Knowledge

出　原　至　道*
Norimichi IDEHARA

キーワード：実践知
Keywords：Practical Knowledge

1. 実践知とは

特集論説の原稿依頼が来てから、「実践知」とは何かを考え、すこし調べた。知らない概念を理解しようとするとき、対になる概念や具体例を列挙して対比すると、理解が深まることがよくある。今回も、事前調査してみた副産物として、いろいろと挙げられていた「知」の分類について自分なりに理解することができた。たとえば、「暗黙知―形式知」（知の表現可能性による分類）、「専門知―総合知―全体知」（知の統合範囲と昇華度合による分類）などがあることが分かった。

さて、それでは「実践知」の対比概念や下位概念はなんだろうか。ある人は、アリストテレスの「テクネー・エピステーメー」を挙げるだろう。また別の人は「暗黙知・形式知」と並列に並べる。Practical Knowledge で Google 検索して、何も考えずに最初のリンクをたどると [1]、「事実」と対比されて「判断を下そうとする相手に対しての実際の現場での経験や知識が必要である」とされている。多摩大学大学院では「最新の実践知を学ぶ（イノベーション、ビジネスモデル、デザイン思考、企業カルチャー、データドリブン経営、ルール形成、データサイエンス・・・）」としている [2] ので、これらが実践知の具体的なジャンルや要素だとと考えていることがわかる。

これらの「実践知」をわかりやすく分類するのも一つの成果だとは思うが、それは私の手には余るので、ざっと眺めた結果として「実践的な問題解決のために有用な行動指針を与える知」という独自の（そして当たり障りのない）定義を仮置きして議論を進める。

2. 毒キノコの思考実験

いま、ある人が無人島に漂着し、その島では多種類のキノコだけが食料になりうるとしよう。このとき、この人が「毒キノコの見分け方」の知識を持っていたとしたら、これは「実践知」と呼ぶことができるだろうか。また、その知識が「以前から実際に色々なキノコを食べてみて、

* 多摩大学経営情報学部　School of Management and Information Sciences, Tama University

ときにはお腹をこわしてのたうち回った」ことによって得られた場合と、「写真入りで詳しくポイントを解説した図鑑を読みふけった」ことによって得られた場合とで、判断は変わるだろうか。

「毒キノコの見分け方」は、ごく限られた範囲の知であって、他の知識領域とほとんど相互作用しない。「こんなものは実践知と呼ぶに値しない」という考えは、ありうる。また、実際の経験を経ることなく書籍から得られた知は実践知ではないという立場もあるだろう。読者の皆さんは、どのような理由で、どのような立場をとるだろうか？

本稿では、この場面で「実践的な問題解決＝体調を崩さず食料を手に入れたい」に対して「有用な行動指針が得られる」ことから、これは実践知であると考える。知の獲得手段は、問わない。一方、同じ人が現代の東京でこの知識を持っていたとしても、通常は、そもそも「毒キノコを見分けること」の問題解決を要求されていないため、これは実践知とはみなさない。もっとも、たとえば「雑学を披露して注目を浴びたい」「特定の相手に毒キノコを食べさせたい」という場合には有用であり、この場合には実践知である。いずれも「一見食用に見えて実は猛毒」というキノコの知識があれば、最適だろう。

3.「247247」は 13 で割り切れるか？

別の話題で、さらに実践知に関する意識を探る。

あるとき私は電車に揺られながら、「7 で割り切れる数の見分け方」を考えていた。倍数の見分け方でよく知られているのは、「2 の倍数」（一番下の桁が 0,2,4,6,8）、「5 の倍数」（一番下の桁が 0,5）、「3 の倍数」（全ての桁の数を足して 3 の倍数なら、もとの数も 3 の倍数）、「9 の倍数」（全ての桁の数を足して 9 の倍数なら、もとの数も 9 の倍数）である。6 の倍数は、「2 の倍数かつ 3 の倍数」で判断できるので、これで解けている。つぎは、7 である。

しばらく考えた結果、7 の倍数の判定方法を思いつき、その結果から直ちに「XYZXYZ の型の整数は 7 の倍数である」という結果を得て満足した。ところが、さらに思考を進めて「11 の倍数判定」「13 の倍数判定」を考えたところ、なんと XYZXYZ 型整数はすべて、7, 11, 13 の倍数であることが分かった。なんと美しい！　当然、247247 も 13 の倍数である。（この事実が与えられたあとでそれを証明することは比較的容易なので、各自試してみて欲しい）　この知識は、いっさい紙も鉛筆も電卓もコンピュータも使用していないで得られており、完全な形で共有可能な形式知である。

ロシアの数学者が、ロシア革命直後に、マクローリン級数展開の剰余項を書くことで命拾いしたという例がある [3] ので、この知識も絶対に役に立たないとは言い切れないが、これが実践知とみなされる場面はほとんどないだろう。

しかし、ここで用いた数学は、素数とその周辺にひろがる豊穣な世界である「整数論」に属している。この分野から、インターネット上で現在広く用いられている公開鍵暗号方式 RSA が生み出された。公開鍵暗号とは「暗号通信を始めたいが、そもそも、暗号を生成するための鍵を暗号通信の前にどうやって相手に届けるのか？」という問題を数学的に解決してくれるものである。公開鍵暗号では、「暗号にするための鍵」と「暗号を戻す鍵」を別々に用意でき、「暗号にするための鍵」を誰かに見られたとしても、その鍵から「暗号を戻す鍵」が簡単には計算できないように設計されているところがミソである。

暗号通信のもつ「鍵のやり取りをどうするか」という問題を解決した点で、整数論は実践知であるということができる。本稿では、純粋な形式知であったり、専門領域に限定されたりしていても実践知であり得るという立場を取っている。

4. プログラミングにおける実践知（例１）

ここから、プログラミングや情報教育の場面で、実践知の意義や課題について考える。

まずはじめに、プログラミングの課題として図１の問題を考える。形式知に見えるかもしれないが、これまでの定義から、実践知につながる問題設定である。

N 個の正の数が与えられます。このうち、大きい方から順に２つの数字を探して出力しなさい。(ただし N≧３)

図１：プログラム課題例１

この問題を、プログラミングを始めて３ヶ月程度の要領の良い学生は図２のように解くだろう。（実際に、類似の問題で同じような解法を見てきた）

N個の数を大きい順番に並べ替えて、 最初から２個の数を出力する	sort(A); writeln(A[1], A[2]);

図２：解答例１

これに対して、図３の解法でも同じ結果が得られる。

大きい方から２つの数を覚えておいて、
 	
数をずっと見ていって、いま覚えている数より大きい数が見つかったら書き換える
	m1:=-1;
	m2:=-1;
	for i:=1 to n do begin
	  if A[i] > m2 then begin
	  if A[i] > m1 then begin
	  m2 := m1;
	  m1 := A[i];
	  end else begin
	  m2 := A[i];
	  end;
	  end;
	end;
	writeln(m1, m2);

図３：解答例２

読者は、どちらが優れたプログラムだと思われるだろうか？　どちらのプログラムも、与えられた課題に対して正しい答えを返すという点では正解である。

しかし、データ数が 10,000,000 を超えるあたりで図２のプログラムは数時間待っても返答が

返ってこなくなるのに対し、図3のプログラムはデータ数が 10,000,000,000 でも数秒で演算が終了する。図2で使われている「並び替え」の命令 sort の内部の演算回数（計算量）が、データ数の増加に対して比例関係よりも大きく遅くなるためである。

もし、データ数が十分少ないことが保証されていて、演算時間がクリティカルな要素でないならば、図2のほうが「よい」プログラムであると判断されるかもしれない。プログラムの見通しの良さ、保守性の高さ、プログラミングに必要な時間などで、図3よりも優れている。逆に、多くのデータが予測され演算処理時間に強い制約がある場合は、図3がはるかに「よい」プログラムである。

このように、同じ課題に対して同じ正解を返すプログラムであっても、実装によって評価に差がでる。これを意識し、課題の要求に対して適切な（＝よい）手法を選択できる能力は、「実践的問題解決のための有用な行動指針」であるため、プログラミングにおいて実践知であるということができる。

5. プログラミングにおける実践知（2）

世界最大級の VR の祭典 Laval Virtual 2021 において、私の研究室の学生による展示企画「Slugger」が審査を通過し、展示を獲得した。このシステムでは、野球の打者の体験ができる。開発過程での課題の一つが、システムの標準物理モデルの演算誤差であった。ボールが実際よりも飛びすぎてしまうのである（図4）。

システムの作成に使用したゲームエンジン Unity では、ゲーム内の物体（GameObject）に対し「物理演算に従うこと」「重力の影響を受けること」の指定を行うだけで、放物線にそって落下する・ぶつかる・跳ね返る・力を受けて運動を変えるなどの「動き」を与えることができる。システム内部では、微分方程式の陽解法（おそらく Runge-Kutta 法）が実装されているだろう。20 年前なら3年生後期科目で一ヶ月以上かけて学習する内容だったが、いまでは、完全な初心者でもプログラミング開始後 10 分で動かすことができる。

陽解法による演算は、「これまでの状態」だけから「次の時点の状態」を演算するため、どうしても計算誤差が出てしまう。特に物体の速度が大きくなると、この誤差が無視できなくなる。Slugger でも、ストライクゾーンに入るよう計算した時速 80 km のボールが、大きく上に外れることが観測された（図4）。これを、正確に物理法則に従った動きにするために、もとから用意されている物理演算を使わずにボールを動かしたことが、本システムの一つの成果である。

システムに組み込まれたモデル自体が、「さまざまな力に対応できる」「演算の時間間隔が均等でなくても演算できる」「単純なオイラー法に比べて高精度」など、これまでに微分方程式の差分化に関して積み重ねられてきた実践知が詰め込まれたものである。しかし、それが万能ではない。

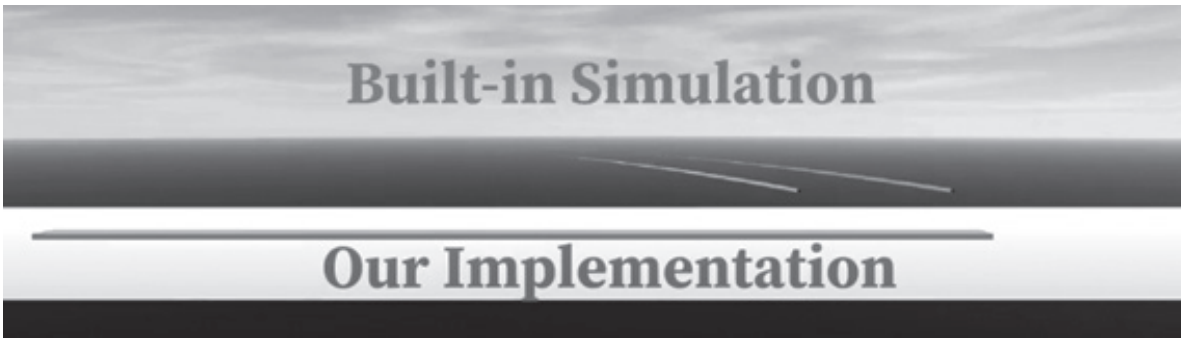

図 4：Slugger における物理シミュレーションの改良

6. ツール化する実践知の限界と課題

プログラミングの分野では、4 節の sort 関数や 5 節の物理演算モデルに限らず、特定の処理を内部でまとめて実行する仕組み・モノを用意して、そのモノの「使い方」だけを知っておけばよいという設計スタイルが主流である（たとえば、オブジェクト指向プログラミング）。これは、このあとも（もちろん 2030 年に向けても）一層加速するだろう。このスタイルには、分業が簡単、再利用がしやすい、効率的に本体のプログラミングが開始できる、管理が容易という利点がある。モノは、その内部に、対象となる課題の解決のためにそれまでに培われた知が詰め込まれた「形式化された実践知のかたまり」である。実際、「すでに動いている優秀な仕組みのプログラミングコードを読み解くこと」がプログラミング学習の有効な手法とされている（コードリーディング）。

しかし、このような便利なモノをツール化・ブラックボックス化して、その理論的な背景や限界を意識しないまま使おうとすると、落とし穴に落ちる。

4 節で、そもそも sort 関数の存在を知らない学生は、いろいろと試行錯誤をしたあとで自力で図 3 のような解法にたどり着くことがある。私の研究室で半年ほどプログラミングを学んでいれば、「sort だと遅いですよね」と言いながら、最初からこれを使わない解法を実装するだろう。「なんだかよくわからないけど並び替えてもらえる便利なツール」の存在を知っているばかりに、非効率なプログラミングをすることがあるという例である。

要求された課題に対して「よい」解決を与えるためには、「よさ」を自分で定義した上で、モノをどのように組み合わせるのか、あるいは、そもそも既存のモノを使わず自分で解法を探求するべきなのかを判断しなければならない。このとき、それぞれのモノ（実践知）についての理解と評価が不可欠である。

情報技術教育とは、決して、ツールの使い方を学ぶ場ではなく、モノに込められた実践知を理解し、活用しながら、課題に対する形式化された実践知を創造する場であり、そのために必要な専門知を身につける場であると考える。

実践知に関して、情報技術教育での（しかしこの領域に限定されないかもしれない）私の問題意識を挙げて、本稿のまとめとする。

1. 暗黙知のままの実践知は、無意味なだけにとどまらず、有害である
2. 創造される実践知も、形式知でなければならない
3. 実践知を創造するためには、専門知・形式知が不可欠である

4. 既存の実践知を採用してよいかどうかを判断する高次の実践知が必要である。さらにそれを判断する実践知が……と階層構造が作られる。完成はしない

参考文献

[1] Yael Schonbrun, Barry Schwartz, "How Practical Wisdom Helps Us Cope with Radical Uncertainty" https://behavioralscientist.org/how-practical-wisdom-helps-us-cope-with-radical-uncertainty/

[2] 多摩大学大学院 ,「知の再武装」http://tgs.tama.ac.jp/about/rearmament/

[3] 数学セミナー「100 人の数学者」(技術評論社) [1989]

[4] 相澤大虎, 尾崎真由子, 柳萬真伸 , ELISABETH Marine, JATOB Nicolas, THAMMAVONG Stella, "Slugger" (2021), Laval Virtual Revolution Student Competition