要求仕様の形式的記述
――Software 技術はここに始まり，ここに窮まる――

柳 生 孝 昭

1. 初めに，言葉の混乱が有った．

曾て‘software 危機’という言葉が世間を騒がせたが，近頃，余り聞かない．これは危機を言い続けることに，嫌気が差したためであろうか．しかし危機が解消された訳ではなく，software 開発の実態は少しも変っていないように感じられる．危機を齎した主要な原因の一つは，しばしば要求仕様が明確に示されないことであると，筆者は考えている．そもそも「仕様」という言葉からして様々な解釈が有り，その曖昧さが混乱，延いては危機の源となっている．そこで先ず語義を確認するために辞書に当たって見ると，「広辞苑」には大したことは載っていない．同じ岩波の「国語辞典」第 4 版によれば

［1］ 仕方，方法

［2］ 作るものに関し要求する，特定の形状・構造・寸法・成分・精度・性能・製造法・試験方法などの規定，

とある．的を射た説明と思うが，幾つか異種の概念を含んでいる．内的・静的な属性・構造（形状～精度），外的・動的な特性（性能），及び物理的な実現の手順（製造法・試験方法）が同居しているのである．一方，「仕様」は英語の‘specification’の訳でもあるだろう．Webster の New Collegiate 第 9 版には

Specification : a detailed precise presentation of something or of a plan or proposal for something

と書かれている．「何かあるもの」の精確な提示・計画，提案する行為や結果を抽象的に指すが，そのものが具体的に何であるかには，一切触れない．内容が捨象されているので，色々な適用が可能である．例えば‘Functional Specification’という言葉をよく聞く．しかし‘Function（機能）’とは何か，物の構造や属性に対して，働きやそれが使われる目的のことか？ 具体的には挙動あるいは入出力の関係を言うのか？ そうならば属性や構造とはどう違うのか？ 目的や意図はどう表現できるのか？ 等々の疑問が忽ち生じて来る． また‘Design Specification（設計仕様）’という言い方もされるが，それは設計に於いて解決すべき課題の如きものを指すのか，それとも文字通り「設計の記述」を意味するのか？ 後者には，筆者のように仕様と設計が全く別の概念という考えの持ち主は，釈然としない．他方，外部仕様と内部仕様の別，要求仕様，それと同義の要求定義，仕様と関連するであろう概念設計・概要設計・基本設計，更には要件定義というように，言葉の氾濫は留まる所を知らない．本節標題の所以である．

2. 初めに，言葉が有らねばならない：「要求仕様」の定義．

この先は「要求仕様」という言葉に統一して，話を進める．先ず定義を述べると，「要求仕様」とは「開発者と使用者が合意する，当該成果物が満たすべき条件の，必要・十分，且つ正確な記述」である．使用者の，使い易い製品を安く提供して欲しいと言った，要望の類ではない．当事者の一方から他方に対するのではなく，両者共通の，開発結果の在り方に対する要求なのである．また，かかる要求は開発に先立って明らかにされるべきだ，との暗黙の了解が含まれていることに，注意して頂きたい．しかしこの定義には反論，ないし疑問が有り得るだろう．その主なものを検討して置かねばならない．

第 1 は，人工物の開発・使用の実際に照らして適切・有意義か？　例えば住宅を建てる場合には，施主は希望を述べ，設計側はそれを反映した図面を提示する．施主は設計の詳細まで理解できなくても，意見を差し挟み，設計側との共通の認識，即ち家が満たすべき条件の必要且つ十分な記述に達する．Program 言語の文法・意味が処理系の開発に先立って公表され，批判を反映して確定され，programmer がそれを理解して使う場合も，開発者と使用者（の少なくとも一部）の間には合意が有ると考えてよい．一方，世の中にはこれとは違う生まれ方の人工物が，沢山有る．例えば自動車や家電製品のような量産品は，設計に当たって，一々消費者の同意を得ている訳ではない．製造者が勝手に造り，使用者は与えられた製品群の中から好みのものを選ぶ外ない．Word　processor も同断．このような製品の設計では，上の意味での要求仕様は，存在しないのではないか？　だが実状は寧ろ，開発者自身が使用者の代表に擬せられた上で，要求が把捉されていると言ってよいだろう．

第 2 に，要求仕様は可能か，また必要か？　書いても未完のまま終わるのではないか？　こういう疑問を持ったまま開発するのは「見切り発車」であり，居直りである．見切り発車の齎した害は，枚挙に暇が無い．安易な「居直り」にこそ，顧客と我々を悩ましている問題の源が在る．後に航空機の自動操縦装置の例に触れるが，人命に危険が及ぶ場合も有るのだから，見切り発車は厳に慎まなければならない．

第 3 に，要求仕様と実現に本質的な差は有るのか？　粗い記述に始まり，段階的に精密化・具体化して行き，最後は実現に至る訳で，仕様と実現は連続しているという考えである．この考えについては，更に検討を要する．

第 4 は，開発者と使用者の何れが要求仕様を書くのか？　筆者の考えでは，開発者である．使用者が要求を自由に語るのは当然としても，正確な技術文書としての仕様の記述は，技術的専門性を必要とする．仕様には開発の技術的・経済的可能性の裏付けが無ければならないが，具体的な方法，困難や解決の見通しを持っているのも，開発者であるからだ．

第 5 に，如何なる言語を用いて，要求仕様を書くか？　この workshop が「形式的仕様」を謳っているのは，自然言語による記述は曖昧になりがちだ，という意識が有るからだろう．逆に形式的言語は難しい，特に使用者に取って理解が困難だ，という批判も聞く．しかし形式的言語の代表例である Z や RAISE について言うと，決して易しいとは言えないまでも，論理や集合の初歩的な知識が有れば，理解不可能ということはない．ただ，日頃使い慣れてはいない語句が含まれているので，それらに馴染

む必要が有る．大事なことは，形式的言語が自然言語に翻訳可能であること，しかも論理ないし集合の初歩的な概念は日常言語に翻訳可能であること，である．そう言うと更に異論を呼ぶだろうが，筆者はそう堅く信じている．

最後に，所謂 prototyping による開発手法の問題が有る．素朴な waterfall model に対する批判としては尤もな点も有るし，要求仕様を定めるための一手段としては，少なくとも部分的には非常に有効だと思う．しかし prototype が発展して実用に耐えるものに至る，という考えは危険だ．それは青天井の開発に陥り，多くの場合，赤字開発になる．

3. 言葉と「もの」：要求仕様は何故必要か？

Software の開発は工業技術としては新参である．新参は先輩である hardware の開発に学ぶべきなのに，それが足りないのが software 開発の未成熟・前近代性の原因だ，という主張が有る．しかし「近代的」な hardware にも随分，驚くべき欠陥・不良が見られる．我々は寧ろ，反面教師としてのそれらの例に，学ぶべきかも知れない．Software が純粋に言葉のみから構成されているのに対し，hardware 特に機械製品は本質的に物理的・感覚的な存在，「もの」である．言葉は「もの」が何であるかを尽くし得ず，人は言葉に頼らずに，更には言葉を超える所を，直観によって理解する．「もの」の理解のこのような直接性は設計，製造，施工，使用等の様々な場面に現れる．それは言葉による記述を省く利益と，そうしてはいけない場合でも省いてしまう，即ち言葉を軽視する弊害を併せ持つという意味で，諸刃の剣であると，筆者は思う．

例えば高速増殖炉「もんじゅ」の温度計の破断という事故が有った[1]．原因とされる鞘の鋭い段差については，設計の常識では考えられないこと，と評する人が居る．他方，図面に描かれていなくても，加工の際に自然に丸みが付くのを期待するのも常識だ，という意見も聞く．実際「もんじゅ」に先立つ実験炉「常陽」では，温度計の鞘は taper 状であった（図 1）．また，段差は加工に用いた工具の形による制約の結果である，と報告されている．実験炉の設計は継承されず，設計意図は（仮に正しかったとしても）図面を介しては伝達されなかった．「常識」とは，言わずとも分かっている筈のこと，の謂であろうが，言葉の不在は，二重の意味で災いを齎したのである．更に根本的な問題は，環境との干渉によって発現する惧れの有る，且つ避けなければいけない温度計の振動特性についての明示的な記述が，全く欠けていることである．それが ASME（米国機械学会）の data の安易な参照に導いた．正に仕様無き設計と言う外ない．

人工衛星「みどり」の太陽電池板の案内機構の故障も，実際の環境（宇宙空間）とは異なる条件の下での物性 data の採用という，真に初歩的な過ちに起因すると聞く．こういう時，報道機関は決まって単純な「設計ミス」と言う．しかし「設計ミス」とは意味不明の言葉である．正しくは要求仕様の欠落または誤りと言うべきであり，仕様が正確・明晰に記述されており，にも拘らず設計がそれを満たしていない場合に初めて，設計の誤りと言える．両者を混同してはならない．

遥かに深刻な例は 4 年余り前の，名古屋での中華航空機の事故である[2]．着陸体勢

図 1 「常陽」（左）と「もんじゅ」（右）の温度計

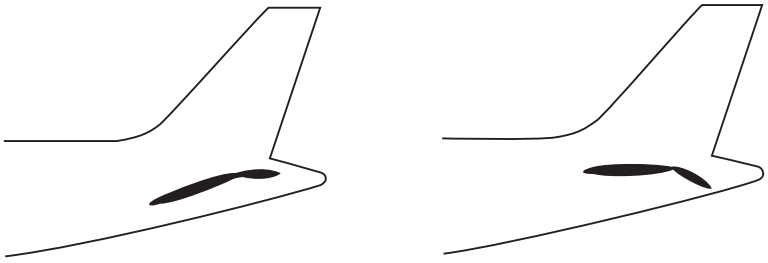

図 2 自動操縦装置（左）と操縦士（右）の操作

に入った後，操縦士は誤って GA（Go Around）mode に入れたが，着陸の操作を続け，自動操縦装置は反対に機体を上昇させる向きに作動したので，水平尾翼が「へ」の字型になり，制御不能に陥った（図 2）．更に失速防止装置が働いたが，速度の増加は迎え角を増大させ，却って失速を招く結果となった．この事故について，幾つかの点を指摘したい．第 1 に，仮に人が過ちを犯しても，機体を制御不能の状態に陥れたり失速に導いたりするのは，組込み software の仕様の，致命的な欠陥である．第 2 に，製造元（Airbus Industrie）は以前に生じた類似の異常事態の後 software の改訂を行ったが，それは，操縦桿に一定の力を加えれば自動操縦装置が解除されるという，全く対症療法的な処方に過ぎない．第 3 は，事故調査報告書も報道機関も，software については版管理の適切さへと問題を矮小化し，要求仕様の次元に立ち入って論ずる所が無い．第 4 に，航空機の専門家や操縦士の立場からの論評には，「人間に反抗する機械」，「コンピュータという神の気紛れ」等の無意味な情緒的修辞や，組込み software の動きは，航空機を知らない software 技術者の恣意的な産物だというような，無責任な発言すら見られる[3]．当事者（Airbus 及び中華航空公司）の主張に至っては強弁・屁理屈，殆ど泥仕合と言ってよい．多くの人命を失うという結果の重大さもさりながら，要求仕様の欠陥が俎に上ることの無いのは，災いの種を残したままという点で，問題を一層深刻なものとしている．

Software 化された自動操縦装置の働きを理解するには，言語を通じる外に手段は無い．それが手動操縦の手応えに直観的・感覚的に馴染んだ操縦士に取って容易でな

いことは，しばしば指摘されている．ましてその働きが一方的に開発者側の手に成るとすれば，尚更のことである．使用者である操縦士と開発者の共通理解の正確な言語化，即ち要求仕様は必須である．

4. 禁欲の精神：自然言語と形式的言語．

先に，形式的言語は自然言語に翻訳可能であると言った．そのことを敷衍したい．また，そうならば形式的言語は要らない，要求仕様は自然言語を使えばよいではないか，という疑問が生じるだろう．それについても触れたい．

先ず形式的言語とは何か，定義とまでは行かないが，特徴として次の 3 点が挙げられる：第 1 に定まった語彙と文法を持つ，第 2 に記述が（例えば自然言語による補足説明を要せずに）当の語彙と文法の枠内で完結している，第 3 は文を解釈する規則も厳密に定まっている．少なくともこれらのことが，必要条件として求められるだろう．1 階の論理記号に一組の非論理記号（常数，関数，及び述語記号）を加えたものが，一つの範例である．これを用いて記述される形式理論（論理式の集合）の各々が，一つ一つ形式的言説，即ち形式的要求仕様を与える．

必ずしも上の特徴を備えない全ての言語を，広く自然言語と呼ぼう．形式的言語による記述を自然言語に書き改めることはできるが，その逆はできない．ある形式的体系を特徴付ける語彙・文法・意味規則が（究極的には）自然言語によって説明されない限り，我々はその体系を「言語」として受け入れそうにない，というのが（今の主張の前半の）理由である．従って異論が有るとすれば，この定義では論理学，数学，科学等の専門的な文章も自然言語に含まれてしまう，つまり自然言語の範囲が広過ぎる，ということだろう．確かに専門的な書物や論文には，自然言語とは言い難い特殊な単語，言い回し，記号が多く出て来る．しかし（少なくとも）そのあるものは，自然言語の表現に，正確に置き換え得る．例えば論理記号∧，¬，∀や推論則がそうだと言えば，論理学の初歩を学んだ人はきっと賛成するだろう．実際，論理学の教科書の多くは，これらに関する文法と意味を，自然言語で説明しているではないか．専門用語や記号，またそれらの組合わせは，冗長さを厭わなければ，自然言語に還元可能である，逆に専門語は自然言語の記法的な経済版に過ぎない，というのは言い過ぎだろうか？

上の議論は，我々の言語行為から一切の専門性を奪うものではない．筆者はしかし，専門（領域固有，非日常的）と非専門（領域共通，日常的）の別を言語ではなく，概念の軸に求めたい．例えば「集合」という言葉は日常生活でも使われるが，数学ないし数学基礎論的な概念としては（再帰的な操作によって限りなく大きな集合を作り出せるというのと，しかし「大き過ぎる」ものは集合として認めない，という二点で）日常性を超えている．固有 class という概念は，例えば $\{x : x \notin x\}$ という形式化された表現を持つが，これを日本語で言い直すのも容易である．しかし Russell の背理が問題にしていることは，日常的概念にはとても還元できまい．曖昧な言い方になるが，多くの非専門家に取って，背理は文章としては理解できるが意味は全く分からない，ということになるだろう．形式的と非形式的（自然）に分かつ言語の軸と，専門的と非専門的(日常的)から成る概念の軸の交錯する様を，図 3 に示す．一つ補言すると，

専門性領域と日常性領域は重なるが，形式的言語と自然言語は分離していると考えられる．

概念の軸		言語の軸：自然言語	言語の軸：形式的言語
日常的・非専門的		売れ残った食品の中で、賞味期限の切れたもの	{x: (x∈S)∧(days(x)≦0)}
日常的・非専門的		毒を飲めば死ぬが、解毒剤を併用すれば、助かる。	CE(0, neutral, alive; 1, poisoned, alive; 2, poisoned, dead)∧ CE(1, poisoned, alive; 1^{+}, antidoted, alive; 2, antidoted, alive)
日常的・非専門的		同時に走り出せば、アキレスはやがて、亀を追い越す。	v(A)＞v(T) → ∃t((∞＞t＞0)∧(d(A, t)＞d(T, t)))
非日常的・専門的		自分自身が要素として属さないような集合の全体は、自分自身に属するとも、属さないとも言えない（B. Russell)。	¬∃S((S∈Σ)∧(S＝{x: x∉x}))
非日常的・専門的		毒と解毒剤を併用すると、体内で化学反応αが生じ、…… 死に至らない。	
非日常的・専門的		アキレスは先ず、亀が今居る場所に、到着しなければならない。 その時に亀は既に、少し進んでいる。これが無限に繰り返されるから、アキレスは遂に亀に追い付けない（Zeno)。	

図 3　言語の軸と概念の軸の交錯

特殊な記号や術語は何も含まない，極く日常的な文章や会話の言葉が，自然言語の原型と言える．だがそのような言葉も忽ち，形と内容の両面で，発展・深化を始める．プラトンやウパニシャドの昔から今に至るまで，深遠な哲学的問題が普通の言葉で論じられて来た．「アキレスと亀」の話は日本語で語り得るが，その中身は勿論，そもそもこういう問題を考えること自体が，真に日常的でない．一方，自然言語は必要と

あれば記号，数式，外来語を殆ど無節操に取り込み，同化してしまう．例えば所得税の確定申告書には‘ (9)＝ト＋{ (チ＋リ)×1/2} ’の類の説明が有る．また金融 system の要求仕様書に，「額面（A），割引価格（B），n 年満期の割引債の年利（r）は，A＝f(r，n，B)＝B(1＋r)^{n} の逆関数 g(A，n，B)＝(A/B)$^{1/n}$−1 として計算する」という記述が含まれていても少しも不自然ではないし，この文章は全体として日本語の文と言えるのではないか．

常に増減して止まない語彙，緩やかな文法，しばしば曖昧な意味という非形式性は，自然言語の強力な記述能力と分かち難い．我々が日常的な概念を語る際に，何の不思議も無く用いている語句を形式化しようとすると，並々ならぬ困難に出会うという経験が，そのことを示している．「因果関係」がその一例だ．この言葉は解明が難しいことで悪名高いのだが，その困難は，概念軸上で日常的から科学的・哲学的へ深める方向と，言語軸上での形式化の方向とに，二重に現れる．先の自動操縦装置の場合でも，因果的な働きが重要な役割を果たしていたように，組込み software の要求仕様に於いて，因果関係の定式化は必須である．その際，目的は日常的な意味の把捉であって，何ら専門領域的なそれではない．とは言え，例えば「毒薬を飲む（原因）と死ぬ（結果），しかし解毒剤を併せ飲めば，死なない」のような非単調性を始め，標準的な論理には無い幾つかの特性を反映する必要が有る．それも一筋縄では行かないことなのだ．言葉が全てである software の要求仕様を記述する上で，自然言語の曖昧さは致命的な欠陥となる．語彙の制約，及び文法と意味規則の厳密化，即ち形式化は，正にこの欠陥の解消と記述の正確さの獲得を目指す．しかしその目的は，強力な記述能力と引換に達成される．ここでは「禁欲の精神」が支配する．

形式的言語にはまた「Ockham の剃刀」が適用され，最小限の語彙，冗長を排した文法的概念，再帰的・効率的な構文規則を備えるのが普通である．それが堅苦しく，取っ付き難い印象を与え，敬遠されがちになるのは，否めない．形式的言語は正に「不自然」なのだ．もし常に達意の自然語文によって，正確な要求仕様を書くことができたら，理想だろう．しかしそのような能力を多くの人に求めるのは，形式的言語に馴染むことを期待するよりも難しい，と言うより不可能に近い．自然言語で書けるものを何故厄介な形式言語を使うのか，という苦情の本音は，正確な仕様化は面倒だ，曖昧な書き方で済むのならば，その方が楽だ，という嘆きであろう．しかし計算機の，益々高度に複雑な問題への応用を諦めるのでなければ，この種の嘆きに耳を貸す訳には行かない．幸いに計算可能な問題は，計算可能性の定義によって厳密に，形式的に記述可能である．だから真実は，形式言語で十分賄える場合まで，自然言語に依存する必要は無い，ということなのだ．

抽象 data 型の代数的仕様は，形式的記述の代表例の一つである．特に stack の例が昔から好まれているのは，それが言語処理のような基幹技術の重要な素材であると共に，極めて簡潔な記述の内に，先に挙げた形式的言語の特徴が余す所無く現れているからであろう．しかし現場の software 技術者に違和感を与えるのも，事実である．画を描けば一目で分かるのに何故，難しい代数式を持ち出さなければいけないのか，実数型の処理に代数的方法は通用するのか，data 型と複雑・巨大な実用 system は同日の談ではない，等々．実験的な規模の問題についての議論を知って，現実の問題の

考察に活かすのは，凡庸の業ではない．現実に近い例示が求められる所以である．その意味で Bjørner 氏の論文[4]は，少し古いが，良い．ここで取り上げられている例は鉄道の運行計画で，実務 system の一部であると聞く．単純ではないが学習に不向きな程複雑に過ぎず，理解に専門知識も要しない．しかも形式的記述と英語によるものとが並置されているのが，興味深い．しかしよく読むと両者に喰い違いが有って，その点も欠陥というよりは寧ろ，教育的である．

5. 語り得ることは，明瞭に語らねばならない：論理の勧め．

筆者は言葉の問題を何故，長々と書いて来たのか？　第 1 の理由は，語り得ることは考え得ることの全てである，ということ．それが言い過ぎであるとしても一歩譲って，明晰に考える，またその考えを他人が正確に理解し得るためには，明晰に語らねばならない．第 2 は形式的言語の枠組によって人は言わば，明晰に語ることを強制されるのである．第 3 に形式的に記述された要求仕様を，必ずしも形式的言語に慣れていない使用者が理解し得るには，使用者の言葉，即ち当該の専門領域の概念を含む（拡大された）自然言語に翻訳しなければならない．形式的言語の自然言語への翻訳可能性は，理論的な主張であるのみならず，要求仕様の形式的記述の実務的な妥当性の根拠でもある．このような理由により，言葉の問題は，正確な技術文書としての要求仕様の問題の核心に位するのである．

ところで「論理」の「理」は物事の筋道，「論」は順を追って述べること，合わせて「論理」という言葉は，正確に，筋道を立てて述べることを意味する．また‘ logic ’の先祖の‘ *λογos* ’は「言葉」でもあった．論理は正に，我々の言語行為の抽象的・形式的に純化された姿に他ならない．従って要求仕様の形式的記述のための言語を，（広い意味での）論理の枠組の中で考案するのは，自然なことである．言語の考案だけではない．与えられた言語による個々の課題の記述，つまり発話行為もまた，論理的に為されるべきである．それは仕様化という情報 system 開発の重要な過程の可視化，従って技術の教育・伝承に資するであろう．

6. 矢は，毒が廻らぬ内に抜かねばならない：実践の勧め．

Max Planck 研究所所長を務めた物理学者 C.F.v. Weizsäcker は言う[5]：「実践的な仕事を進めることから回り道をして，純粋な反省の塔の中に戻る者の持つ反省は，哲学的に無内容になるだろう」．

形式的仕様記述は新しい技術ではなく，既に多くのことが論じられて来た．にも拘らず普及に程遠いのは，争えない事実である．その原因は，次のような疑問に在るのではないか．第 1 は要求仕様らしきもの，またはそれに準ずるものは今でも（自然言語で）書かれており，それなりに機能している．必要十分とか形式性とか，何故そこまでの厳格さを求めるのか？　第 2 に形式的言語の習得は，開発者に対して新たな負担を課す．第 3 に要求仕様記述の工程を重くすることは，開発期間を長くし，費用を増す．これらの疑問と答について論ずるのは意味が有るし，本稿でも触れて来た．しかし今や重要なのは，実務への適用と評価を通して，有効性を示すことである．しかし実践に踏み切るには，それが答える筈の疑問を，予め乗り越えておかねばならない

という，些かの矛盾も有る．だから適用の対象としては，時間の余裕が有り，多額の費用を要しない，成功すれば確実な効果が期待できるが，失敗した場合の危険は小さい，というものを（選択の幅は狭いが）選ぶのが望ましい．筆者の推奨は，例えば既存の，よく利用されるが明確な仕様書を欠く，いずれ改良ないし再開発が必要になるであろう，手頃な規模の sub－system の仕様記述である．言わば reverse engineering としての仕様化である．一つの実践が成功すれば，更に規模の大きい，危険も伴う次の対象に向かう決断が，ずっと容易になるであろう．一番良くないのは，情報 system の開発の上手く行かないのを嘆きながら，抜本的な試みに挑むことはせず，徒に手を拱くのみ，という態度である．矢は，毒が廻らぬ内に抜かねばならない．

7. 経験から科学へ：Software 技術者の職業的矜持．

全ての製品が使用者の要求を満たすためのものであることは，自明である．しかしこのことは決して，供給者の事情を軽視しても良いことを意味しない．供給者が製品とその製作の過程に，自信と誇りを持てなければ，使用者の満足を得ることも期し難い．特に製品が知的・技術的な営みの産物である場合には，自信や誇りは単なる主観性ではなく，社会的・金銭的な評価のみによって正当化される訳でもなく，当の営みが堅固な工学的・科学的基礎の上に展開されているという事実により初めて，真正なものとなる．炉心の設計者や航空機の設計者の職業的矜持は，例え本人が意識せずとも，原子力工学・炉物理学や航空工学・流体力学等の高度の専門性によって支えられている筈である．同様に，software 技術者は知的専門家を以て自任すべきであり，そのためには，彼/彼女の技術は software 工学と計算機科学に根差すものでなければならない．しかし実態はどうだろう？　現場の仕事は理論と無縁であり，研究者は開発の実務に関心を示さない，ということではないか．残念ながら社会は，software の開発に携わることを，尊敬に値する仕事とは見なさない．‘ Software 危機 ’ の余燼が未だ燻っている頃，かの「第 5 世代計画」発足の式場で挨拶に登壇した関係官庁の役人は，同計画が，世期末には 90 万人を越えると予想される（と当時は喧伝されていた）情報処理技術者の不足の事態を救うであろう，と言って退けた．あれは冗談の心算だったのか，本気の話か，筆者には遂に分からない．質の問題を量の不足に転嫁する，理論に現実を救わせるという二重の意味での怪弁（それとも卓見？）を，研究側はどう聞いたのだろう．何れにせよ，少数精鋭の Prolog programmer が百万の Cobol programmer に取って替わる事態は出現しなかったし，今後も出現しないだろう．しかし真面目な話，論理的な枠組が色々な工程で用いられれば，software 開発の合理化と生産性の向上に役立つであろうと，筆者は信じている．

3 節で，航空機の組込み software の開発が開発者側の恣意で進められているという見解に触れたが，software 技術者の心情はあるいは，納得は行かぬが使用者の求めるままに作らされた，ということかも知れない（よく有る話だ）．開発結果の振る舞いについて予め理解を共有しなければならない当事者が，被害者意識と相互不信に陥っているとすれば，不幸なことである．文字通り命懸けの乗客は，たまったものではない．仕様を開発者に任せ放しにする無関心が一方の極端ならば，開発の専門家は要らない，software は使用者が作ればよいと言わんばかりの software 技術，及び技

術者に対する蔑視が，他方の極端に在る．曰く：「コンピュータ利用技術のプロという人種の存在がなぜおかしいか…そのことを端的に象徴しているのが名古屋空港での中華航空機の墜落事故である．…飛行機のことも操縦のことよく知らないおじさんが，飛行機の自動操縦のプログラムを書いている．…コンピュータ化の動機と実施主体が真の利用者にあれば，…危険はある程度和らぐ．」（[3]に引用されている文章）．ここには使用者の関与が programming に手を出すことではなく，要求仕様の水準にこそ求められる，という認識が，全く欠けている．

要求仕様の確定は software 開発の最初の，且つ最も重要な工程である．それを経験にのみ委ねるのではなく，理論化・科学化するのは従って，software 工学の原点である．しかも理想的に記述された要求仕様は，実行可能な実現の prototype を与え，更に code の自動的/半自動的生成の出発点となる，と期待される．その意味で要求仕様の形式的記述は，software 技術の一つの極点を成す．

Bjørner の論文の標題が‘生き残る Software 産業’を謳っているのは，示唆的であった．Software 技術者が工学的・科学的に裏付けられた職業的矜持を持たなければ，産業としての software 開発の未来は無い．それは益々 software への依存度を高めざるを得ない，現代社会全体の未来をも，暗くするだろう．Weizsacker の著書には，前節に引いた文章に先立って，次のような警句が有った：「我々のこの複雑な世界を，できるだけ理論的に理解するよう促進するのを怠る者は，正に実践の場に於いては，結局は利益よりも害を齎すことが多い」．

参考文献
［1］ 動力炉・核燃料開発事業団，「高速増殖原型炉もんじゅ，ナトリウム漏えい事故の原因究明結果について」，科学技術庁，平成 9 年 2 月 20 日．
［2］「航空事故調査報告書」―中華航空公司所属，エアバス・インダストリー式 A 300 B 4-622 R 型 B 1816　名古屋空港，平成 6 年 4 月 26 日―，運輸省航空事故調査委員会，平成 8 年 7 月 19 日．
［3］ 遠藤浩：「ハイテク機はなぜ落ちるか」，講談社（ブルーバックス），1998. 5.
［4］ D. Bjørner :‘ Prospect for a Viable Software Industry ’, IISF Symposium of the ACM Japan Chapter, 1994. 3.
［5］ C.F.v. ヴァイツゼカー/野田保之・金子晴男訳：「科学の射程」，法政大学出版局，1969. 6.

執筆者紹介　柳　生　孝　昭（Takaaki Yagiu）
1957 年東京大学理学部数学科卒業．58 年，日本レミントン・ユニバック株式会社（現日本ユニシス株式会社）入社．米国駐在員，応用ソフトウェア部長，システム本部長，常務取締役を経て，現在，同社顧問．東京大学人工物工学研究センター客員研究員．著書：‘ Modeling Design Objects and Processes ’, Springer-Verlag, 1991.，訳書：「ある数学者の生涯と弁明」（G.H. Hardy），シェプリンガー・フェアラーク東京，1994.，共著：「新工学知」（吉川弘之監修），東京大学出版会，1997.