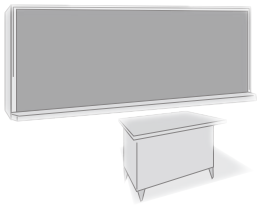
基般

解説 対話で教える コンピュータアーキテクチャ



都倉信樹　大阪電気通信大学

アーキテクチャ教育の困難を感じませんか？

コンピュータサイエンス（CS）の中心的科目である，「**アーキテクチャ**☆1」や「**OS**」の講義の現状はどうだろうか．教えにくくなっているのではないか．その理由はいくつかある．コンピュータの出現からすでに60年余が経過し，アイディアが出尽くし，標準的アーキテクチャが固まってしまった．本来この種の分野では種々な方法を考えて，そのときの技術で最も合理的なものを比較検討して選択することが重要である．しかし，講義学習の時間は十分取れない制約がある．また，欧米の大学教科書は一般に非常に分厚く内容豊富で説明が丁寧で分かりやすいものが多い．しかし，日本では，薄い教科書が求められ，その記述は標準的な1つの方式を紹介して終わることが多く，説明も少なくあまり分かりやすくはない．なぜその方式が考えられ，主流になったのかなど何も分からない．この教材の「空間的制約」のもと，一通りの標準的な方法だけが講義され，学生はわくわく感など感じることはない．いろいろな代案の中からそのときどきの目的や技術水準に応じて最適な選択をしてきたことを知り，今後新しい課題に自由に発想し，それを冷静に評価するという方法，発想を身に付けてほしいし，学生が興味を持っていろいろ考え始めるにはどうしたらいいかという問題意識で本稿を書いた．

☆1 アーキテクチャ．最近はソフトウェアやシステムのアーキテクチャという言葉遣いもされるようになったが，ここでは，コンピュータあるいはコンピュータシステムのアーキテクチャの意味である．

❑ 新しいアーキテクチャ教育教材

すでに飽和した感のあるコンピュータシステムの教育に，学生が興味を持っていろいろ考え始めるような材料を提供したいと考えた．そこで，メモリに機能を埋め込む機能メモリという発想でなにが可能かを考えていき，それを次々展開していろいろな課題を提出する．だんだん考えは違う対象にも及んで，新しいアーキテクチャへ考えが及ぶという構成を考えた．その提示方式として対話形式を採用する．教材をどういうスタイルで提示するかに悩むところであるが，積極的な卒研生，あるいは修士の学生（●）と筆者（◆）の対話形式として記述することを選択している．

教材の作成というと，教科書（本文，演習問題，補足材料など），演習問題・回答集，プレゼン素材，試験問題，採点基準等々多くの文書を作成することと考える方もおられよう．ここではそれらを全部揃えてみせることはできない．むしろ，教材の作成の際にどういうストーリーを立てるか，いわば設計段階をお見せしたい．

❑ ストーリーについて

筆者はこれまでいくつかのテーマで，教材としてストーリーを立てて使ってきた．たとえば，自動改札機の秘密，バーコードの秘密（郵便局の使うバーコードも含む）．これらは，普通何気なく見ているものから出発し，なんのために，どのように，というような疑問を発することから始める．1つの謎が解けると，次の謎が浮かんでくる．できあがったも

のの仕組みを単に説明するのではなく，どうしてそういう発想が生まれ，そのためにどういう技術を使ったかなどを結果的には伝えるのだが，できるだけ謎ときの連続になるように構成する．こういう謎は筆者だけでなく，多くの人がやはり興味を持ってくれる．こうやって，ある対象について，つぎつぎと疑問を持って調べて，仮説を立てる．どうしてもその仮説が自分の集めた証拠では確認できないこともある．実は，これらのストーリーは放送大学の情報工学という TV 講義で使って，その現場ロケを挿入することもあった．そのとき，ロケ先で担当者に聞いて謎を解決することもできた．こうしてだんだん補強し膨らませていく．聞き手に疑問を次々投げかける素材は，学生に考えさせる講義になる．聞き手も退屈せずに聞いてくれるし，インタラクティブな講義にもなる．

ある数学者と話をしたとき，ストーリーを感じさせる数学の名著が少なくないと言われ，いくつか例もご教示いただいた．少し専門的なので，ここでは最近見つけた本，『清水健一：大学入試問題で語る数論の世界』[1] を例として挙げる．入試問題から出発し，数論の中でのどういう問題かを述べ，少し条件を変えると未解決な問題となるという例を多数挙げてあり，実に興味深く最後まで読ませる力のある数学の本である．

筆者のストーリーは，放送大学の TV 講義，面接授業で放送大学学生に，出前講義で高校生に，鳥取環境大学の学部講義，あるいは，大阪大学の大学院の講義でというようにいろいろな機会に使った．もちろん，細部は講義の目的や聴き手のマチュリティ，時間に応じて，表現を変えたり数式の扱い，あるいは取り上げる疑問を変えたりはするが，基本的な謎解きはまったく同じものが使える．そして，ストーリーがしっかりできていれば，最後まで興味を持って付き合ってくれることも確信できた．ただ，これらのストーリーは私の頭の中にあり，文章で書くと長くなるし，あるレベルに固定した記述になる．これを外部に取り出して見せる方法はないかと模索してきて，今回試すのが対話形式の記述である．二者の間でのやりとりの中で，相手が考える部分は細かく記述する必要がなくなり，ストーリー部分を取り出しやすい．

昔語りから対話は始まった

◆ 70 年代，ミニコン pdp11（ピーディーピー イレブン）を知ってしばらくして思いついたアイディアが今回の話の発端なんだけど...

● 先生，待ってください．**ミニコン**，**pdp11** って聞いたことありません．

◆ え？（昭和も遠くなりにけり！［表紙がすれ切れかかかった pdp11 のハンドブック[2] を取り出し丁寧に説明を始める］）pdp11 はユニバスという共通バスを持ち，メモリも周辺機器も一様にアクセスできる仕組みになっていた．なお，メモリは 16 ビット 1 ワード（2 バイト 1 ワード）で，バイトアドレスがついている...

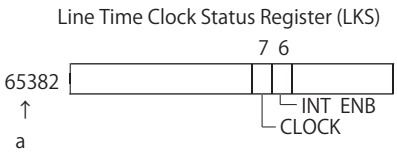
● それ，ハンドブックに書いてあるのですね．

◆ そうです．この小さいハンドブックはインターネットでも読めます[2]．アーキテクチャの勉強を始めるのにかっこうの参考書となります．英語で非常に分かりやすいのでぜひ読んでみてください．プログラマの立場から新しいアイディアを持ち込んだ非常に斬新なミニコンピュータで，現在の多くのプロセッサにその影響が見られます．また，プログラミング言語 **C**，オペレーティングシステム **UNIX** 等の発想に大きく作用したと言えます．

❑ 電源周波数クロック

◆ このミニコンには **Line Frequency Clock**（電源周波数クロック）というのがあり，それは電源の 1 サイクルを単位として時間間隔を知りました．これは商用電源を信号源とする簡単な波形変換回路（正弦波から方形波に変換する）と制御回路からなります．そして，割り込みを許可すると，1/60sec ごとに割り込みを発生します（西日本の話）．

図 -1 を見てください．動作開始時はリセットされるが，（OS が）a=65382 番地（番地は 10 進表現）



bit6	INT ENB（Interrupt Enable） 1にセットしておくと，CLOCK（bit7）が1になるたびに割り込みをかける． プログラムまたはリセットあるいは起動シーケンスでクリアされる．
bit7	CLOCK 電源に同期して毎1/60秒（あるいは1/50秒）ごとに1にセットされる．LKSを読む，リセットする，あるいは，STARTスイッチを押すと0にクリアされる．

図-1　電源周波数クロック状態レジスタ（右表にビットの説明）

の6ビット目（INT ENB 割り込み許可）ビットに1をセットすると以後7ビット目が1になるたびに割り込みがかかる．7ビット目は動作開始時OSがreset命令を発することで0にクリアされるが，電源から作った信号の変化したところで，ハードウェアが7ビット目を1とセットする．OSの時刻処理ルーチンが，a=65382番地を読む（read accessする）ことで7ビット目がクリアされる．そして，OSのシステムクロックを1/60秒分進めて割り込み処理から戻る．60回この操作を行えば，1秒経過したことになる．こうしてOSは実時間を保持したのです．

[課題案1]この電源周波数クロックの回路を設計する課題も面白い．電源の正弦波を入力として1/60秒ごとに，CLOCKビットを1にセットする働きをする回路である

● 時間も電力会社に依存していたのですね．いまならもっと精度の高いクォーツのクロックを使いますよね．

◆ そうです．そもそもコンピュータが実時間をどうやって知るのか，どの程度の精度が得られているかは本質的な問題だよね．もう1つ例を挙げます．

❑ ディスクインタフェース

◆ 電源周波数クロックより複雑な装置の例として**ディスク**があります．複雑な機器の場合は指示すべき事項が増えるので，レジスタの数が増えます．今から思えばごくごく小さい当時の磁気ディスクでも次のようなものは必要でした．サーフェイス番号，トラック番号，セクタ番号，メモリブロックスタート番地，メモリブロックサイズ等です．これらのデータを入れる複数のレジスタがあります．これらを設定しておいて，制御レジスタのビットr/wを1にし，start/doneビットを1にすると，ディスクからの読み出しが行われ，**DMA**（Direct Memory Access）方式[☆2]で，指定したセクタのデータがメモリに転送される．転送が終わったところで，割り込みをさせるなら，制御レジスタの割り込み許可ビットを1に設定しておく．制御／状況レジスタは1つのレジスタに集約することが多いのです．

❑ メモリ空間に置かれたIO

◆ これらの制御／状態レジスタ，データレジスタが，機器ごとに数ワードずつメモリの特定番地に設定されます．制御レジスタに書き込むことで動作の起動などを指示し制御し，結果やデータは状態レジスタ，データレジスタで読み取れます．この方式では，主記憶アクセスのバスと別に入出力バスを設けず，1つのバスで，プログラムで周辺機器を直接操作できました．**チャネル装置**を使う大型機に比べ，格段にコストが安く，かつ，標準品ではない機器でも，公開されているバスの規約に合わせ，未使用の番地にそれらの制御／状態レジスタ，データレジスタ類を配置し，プログラムを作成して，容易に接続することができたのです．このように，ある番地にアクセスして，1語のデータの読み書きをするメモリの1部に，特定の機能を持つレジスタ群

☆2 DMA：コンピュータに接続した種々の装置と主記憶間のデータ転送を中央処理装置CPUが取り仕切る方式をPIO（Programmed Input/Output）という．これに対し，CPUのお世話にならずに装置が自律的に動いてデータ転送を行うのがDMA方式である．PIOだとハードウェアは簡単というメリットはあるものの，CPUがIOの世話をするのでCPUの効率はいいとは言えない．DMAでは，装置ごとのDMA制御器に指示を与えるのはCPUだが，たとえば，ワンブロックのデータ転送を指示されれば，DMA制御器がすべての転送を終えて，割り込みをかけるという方式がとれるので，CPUの負担は減る．なお，別の見方では，分散制御方式とも言え，共通バスにアクセスする権利の調整のためバスアービタが必要になる．

を配置でき，メモリと同様にそれらのレジスタを読み書きできる仕組みで入出力機器を制御する方法を，**memory-mapped IO**（**図 -2**）と呼んでいました．主記憶と同じメモリ空間の一部，最後の 4k 語分をそれらのレジスタを置くために使いました．なお，割り込みベクトル等はメモリ空間の 0 番地に近い方の下位空間を使いました．

● はい．

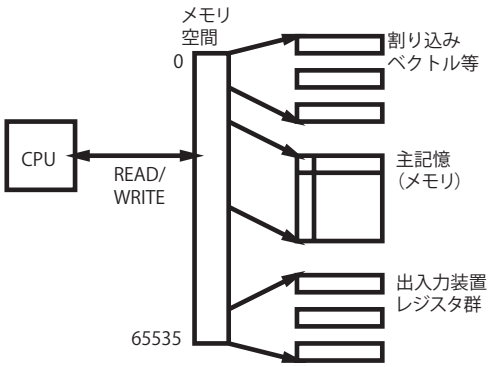


図 -2　メモリ空間に配置された IO

展開

❑ 2 進データの重み

◆ この仕組みで，当時いくつかのアイディアを持っていました．1 つ紹介します．番地 a に 2 進データ d を書き込むと，次の語（番地 a ＋ 2）のレジスタにそのデータの「重み」w(d) が得られるという演算機能を持たせたものです（**図 -3**）．

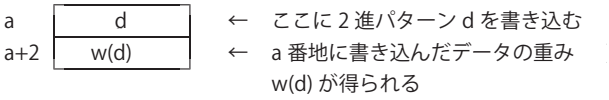


図 -3　重み算出器

● 重みってなんでしたっけ？

◆ **重み**（weight）は 2 進データ中の 1 の数です．2 進パターン d から，その重み（の 2 進表現）を求める組合せ回路は「論理設計」の演習でやらなかった？

●（きっぱり）一切記憶にありませんけど．

◆ ？... じゃ，復習にいいから最低 2 通りの組合せ回路を提案してください．なお，結構複雑になるので，小さい桁数から考えて桁数の増加にどう対応するかを考えるといいでしょう．

[課題案 2] このように論理設計の問題が作れる．これはちょっと難しい部類になります．回答は何通りもできますので，いろいろ自由に考えて利害得失を考えるといい

● はあ．それはそうとして... ほかにどういうことができるでしょうか．

◆ それは工夫次第．このように，メモリにデータを転送したら，計算結果がメモリのどこかに書かれてくるという機能を持ったメモリです．単にデータの入れ物というだけでなく，データの加工もするメモリという意味で，**機能メモリ** functional memory とか function in memory と呼んでもいいかもしれない．当時はサブルーチンコールのオーバヘッドも気になる時代ですからこの発想は魅力的でした．

● **サブルーチンコール**でどんなオーバヘッドが必要なのですか？

◆ サブルーチンコールでは，ただサブルーチンのところへ飛ぶだけでは済みません．パラメータを引き渡すとか，結果をもらうとか，サブルーチン内でレジスタを使うなら，レジスタのデータを退避しておくとか，いろいろ準備と後始末が必要です．これらがこの場合の**オーバヘッド**（Overhead）となります．まだそう高速でない時代のコンピュータです．少しでも早く動くことが求められました．いろいろ工夫を凝らして，こういう問題も 1 つ 1 つ改善されてきました．

● そうなんですか．

[課題案 3] このようなデータに対する演算機能を提供するものについて，そのインタフェースと，内部での処理の仕方を問う演習問題は多数作れる

◆ なにか 1 つのデータに対して，計算して結果を出してくれるような例を考えてみてください．たとえば，最初ちょっと設定して，あと次々データをほしいというような...

● ひらめきました．乱数生成器！　どうです？

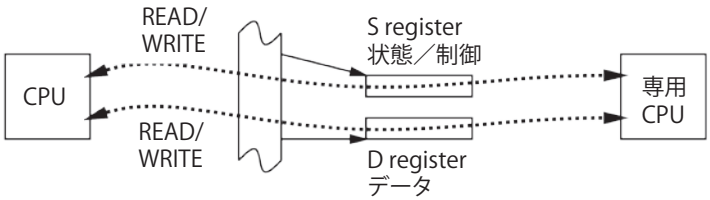


図 -4　窓口レジスタ

◆ なるほど．いい例を考えましたね．よく検討してください...（注：最大長系列，乱数，レジスタ転送表現，ハードウェアによる乱数発生などが話題になったが，紙面の都合で省略します）

結構です．じゃ，他の例はどうです？

● はい，三角関数 sin x の計算などどうでしょう．（注：三角関数は浮動小数点表示，たとえば 3 語の入力に対し，3 語の出力が得られるということで議論する．sin x を作るなら，当然ほかの FORTRAN の種々の数値計算の関数なども用意したくなるという話になる．それらは同じインタフェースレジスタの未使用の上位 8 ビットで区別するとか，データ型の変換なども扱えることを議論．省略）

❑ 実現法

◆ ここまでの話を少し整理しましょう．アプリケーションプログラムや OS が動く通常の世界から，このレジスタという窓を通して，計算してほしいと求められるわけです．通常の世界の方が当然主で，依頼されて計算する方が従です．主従の関係を master-slave という言葉を使うこともあるが，すこし言葉がよくない．なにかいい言葉はない？

● やはりクライアントとサーバという見立てができます．ただ，クライアント―サーバというとほかと混同する心配もありますけど，この文脈だと了解していただけば，別にネットワークの世界などとは混同されないでしょう．

◆ 整理すると，通常の計算が展開される方がクライアント側で，そこからリクエストを受け取り計算して結果を返すサーバがあるということですね．サーバをどう実現するか？

● 専用回路を用意するハードウェア方式も考えられますが，柔軟なシステムを作りたいとすれば，当然プロセッサを置き，ソフトウェアを使ってという考えになりますね．

◆ そう．そういうサーバ側のアーキテクチャをどうすればいいかを考えるのが，また，非常に面白い．高速な応答を求められる機能ならハードウェアで作ってもいいが，サーバ側にもプロセッサとメモリがあって，クライアントからの要求を引き受けると考えてもいいですね．むしろ，この考えの方が柔軟性があって，いろいろな発想が湧いてくると思います．両者の間に存在するレジスタ群を**窓口レジスタ**（**図 -4**）と呼ぶことにしましょう．そのうち，データを置くレジスタを **D レジスタ**，制御／状態レジスタを **S レジスタ**と略称します．

● そうみると，結構問題がたくさん作れますね．クライアントは従前の **CISC** で作り，このサーバ側は **RISC** で作るというようなこともあっていいわけですね．いままで習ったことをいろいろ思い出しながら，どうすればいいかを考えるといいですね．

◆ そう，課題・問題群を提出するのがこの対話の目的なので，そう思ってくれるといい．

データ構造・オブジェクトの扱い

◆ ここまで 1 変数関数を扱ったが，多変数関数ならどうだろう．

● 多分そうなるだろうと，ちょっと考えましたが，1 変数のときと変わり映えしないのです．つまり D レジスタを必要数設置し，そこにデータを用意して，S レジスタを通して「計算始め」といえば，結果が帰ってくるというだけで，D レジスタが増えるだけです．

push	D ← data（data を書き込むと自動的に push される）
top	D のデータを読むだけ（変化なし）
pop	S.pop ← 1（S の pop ビットが 1 の間は pop 作業継続中．終わると自動的に 0 に戻る）
isEmpty	S.empty で表示（1 ならスタック空，0 なら空でない）
isFull	S.full で表示（スタックが一杯のとき 1 と表示）
InitStack	S.Init ← 1（スタックの初期化をする．このビットが 1 の間は作業継続中で，終了すると 0 に戻る）

表 -1　スタックのインタフェース

empty	スタックが空のとき 1，そうでないとき 0
full	スタックが一杯のとき 1，そうでないとき 0
pop	1 とすると，1 語分 POP する
Init	1 とすると，初期化する
busy	1 のとき新しい指令は受け付けない．0 のとき S，D レジスタに書き込める
INT ENB	1 で割り込み許可．誤操作のときに割り込み

表 -2　スタックの状態語のビット

◆ そうだね．じゃ，今日は違う問題群を考えていきましょう．「データ構造とアルゴリズム」という講義を聞きましたね．

● はい．これはコンピュータサイエンスの最重要のコア科目ですから，ちゃんと勉強しましたよ．任しといてください．

◆ おっ！ 心強い言葉だね．いろいろなデータ構造が出てきましたね．

スタックから考えましょうか．スタックもここで考えている機能メモリに埋め込めます．

❑ スタック

◆ たとえば，1 語のデータを push したり，pop したりでき，top のデータはいつも直接見られるという**スタック**（stack）は簡単にできますね．

● はい．データレジスタ D と，状態レジスタ S とを用意します．そして，**表 -1** のようにすればいいと思います．

◆ 有限サイズのスタックという考え方でインタフェースを考えましたね．

● はい．抽象データ型のところでそういうことを聞きました．

◆ これはスタックの機能について整理したものですが，具体的な S レジスタのビット割り当ての案を作ってみてください．

● 図は省いて，ビット割り当てと役割を**表 -2** にまとめてみます．

◆ 状態レジスタのビットの位置は，システム内で一定の規約を設けて，同じ意味のビットは同じ位置にくるようにすると，プログラムミスを減らせるし，ハードウェアで制作する場合，回路を標準化できます．

● このシンプルなスタックでいいのかがちょっと心配です（以下，式の計算をスタックで実現する方法，アルゴル流言語のスコープルールを採用するような場合なども議論のテーマとなる．略）．

● 少し思ったのですが，こういう話をすると，なんでもサーバ側に任せないといけないと思う人がいますが，クライアント側でもちろん，これらのデータ構造は扱えるので，どっちがいいかのメリットデメリットを考えてやればいいことですね．

◆ そう，そういう考え方が大事です．いろいろな案を比較検討して，適切なのを選ぶのです．

[課題案 4]キュー，プライオリティキュー，表などの課題も面白い．まずインタフェースを考えるが，余裕があれば，内部での実現も検討してみるといい

◆ 抽象データ型の考えもこの方法となじむことも分かったと思います．ここまで来れば次はオブジェクト指向ですね．さあ，どうなるでしょう．

❑ オブジェクト

● オブジェクト指向プログラミングで扱うオブジェクトをサーバ側に預けるというのは適当かどうかはちょっと分からないのです．たとえば，基本データ型に相当する整数などのオブジェクトをわざわざこんなインタフェースを通して渡す必要はない気がします．ただ，システムの複雑な機能やセキュリティが問題になるようなクラスに属するものをサーバ側に任せて，クライアントからはメッセージを窓口レジスタを通してオブジェクトに送るという形が考えられますね．大体制御レジスタは空きビッ

トがあるので，そこで，あるオブジェクトに対するメソッドの区別をすれば，1つのオブジェクトに多数のレジスタを用意する必要はなくなりますね．

◆ そうだね．そういう切り分けが有用となる可能性があります．それから，new(p) でインスタンスを次々作るようなときにどうしますか．

● これはサーバ側にメソッドの処理を任せるが，データはクライアント側に置いて，そのアドレスを窓口レジスタに書いて，サーバからそのアドレスでアクセスしてもらうという方法で，クライアント側にデータを持つことはできます．

◆ そう，そういう考え方もできるし，データもサーバ側に押しつけたければどうする？

● うーん．そうですね．窓口は1つだけど，生成されるインスタンスを区別するために**タグ**（一連番号）を用意して，それを使ってサーバ内にあるオブジェクトの実体データを区別すればよいと思います．

◆ そういう考え方もできるね．こうすれば，データはサーバ側に隔離することもできる．結局，オブジェクトだからといって特にこれまでと変わるところはない．その実現法はいろいろ考えられるということで，課題がいくつも作れるでしょう．特に，クラスライブラリで提供されているようなオブジェクトをこういう仕組みで利用するのも面白いと思われます．その際，継承などのクラスにかかわる概念をどうするかもありますし，そのためにどういう仕掛けが必要で効果があるかの検討も必要ですね．

● なるほど．まだまだ深く考える問題はあるということですね．

OS とセキュリティ

◆ 実は今までの話を振り返ると，サーバ側はクライアントのいろいろな要求を聞いて，データ処理をしたり，入出力を代行したりします．ということは，広義での OS になっているとも見えるわけです．そこで，逆に，「サーバ側にいるものを OS である」と視点を変えてみると，どういう景色が見えるかということです．

● 私も薄々そういう予感がありました．プログラムから OS に要求を発する**スーパーバイザコール**（SVC，supervisor call）も，この窓口レジスタのやり方で自然に実現できます．スーパーバイザコールは，プログラム割り出し，OS へのリクエスト等種々の用語があるが，実行中のプログラムから OS にいくつかのパラメータを伴って，依頼の種類を番号で伝えるものです．したがってパラメータを D レジスタに設定し，依頼の種類を S レジスタに設定し，サーバ側に伝えればいままでと同じ形で実現できます．クライアント側のメモリ空間にサーバ側が DMA のようにアクセスすることでクライアントのメモリの読み書きをする．こうして，通常の OS の機能はここでいうサーバ側にすべて持っていけるのですね．クライアント側はユーザタスクの実行のみをやり，サーバ側に OS がいて，サービスするという役割分担ができます．

◆ そうですね．さて，OS だと思えばどうなるか．宿題を出しておいたでしょう．

● 次のようなことを考えました．いわゆる**マルチタスク**が簡単にできるようになります．各タスクはタスク番号で管理します．たとえば，仮に0から31とすれば，32タスクを扱えます．

タスクごとに使う高速レジスタ群を，**タスクレジスタ**と仮に呼びます．これには，そのタスク用のプログラムカウンタ PC，プログラム状態語 PSW（むしろ，タスク状態語と呼ぶ方がいい），汎用レジスタなどが含まれます．タスク番号が32個あれば，タスクレジスタを32個用意し，タスク番号で切り替えて使います．普通のメモリはキャッシュを使うとして，仮想ブロックと実ブロックの対応は OS 側が管理しています．対応表はタスク番号をアドレスの上位ビットにつけて，変換表（連想記憶）を検索し，実ブロックアドレスを見つけてアクセスするわけです．OS は，クライアント側の実行タスク番号レジスタを制御して，そこに実行すべきタスクの番号を設定し，CPU に実行を命じれば，自動的にそのタスクに必要な高速レジスタ群が選択され，通常のメモリアクセスもタスク番号が論理アドレスの上位に

使われます．このような方式で，タスクスイッチは，単にタスク番号の書き換えだけで済み，初期の OS で問題になったタスクスイッチのオーバヘッドは非常に小さくなります．

◆ そう，**タスクスイッチ**のオーバヘッドを小さくできる．もちろん，スケジューリングは OS の方でやって，次に起動するタスクを決めておくという作業は必要です．

● でも，2 つのプロセッサとメモリを使うというのは少しもったいないという気がしますが．

◆ 最近は quad core のプロセッサもあります．集積度が向上し，むしろシリコンの面積は余っていて，そこに何を埋めるか，いろいろ工夫しているわけです．

● そういえば，ワンチップマイクロプロセッサは多くの機能をてんこ盛りにしてますね．

◆ どういうアーキテクチャが合理的かは，そのときの技術で決まります．pdp11 を知って，この機能メモリを考えたけれど，とても当時の技術では合理的ではなかった．だから，スタックくらいしかそのときは検討しませんでした．

● 今なら使えるかもしれないということですか．

◆ そうですね．今やシリコンチップには相当の機能が埋め込めます．並列処理の機会が多ければ同一のプロセッサを多く持つアーキテクチャが有利だけれど，この機能メモリの考えは，むしろ機能分割的な発想です．CPU は同じでなくてもいい．

こうなると，サーバ側は OS 専用の CPU にしてもよい．従来は 1 つの CPU で OS もユーザプログラムも実行するので，OS 専用という発想はなかったでしょう．

● はあ．これで OS 専用のハードウェア・ソフトウェア協調設計の契機になりますね．

◆ そう，OS に特化したアーキテクチャを検討することは面白いと思いますよ．それとメモリ空間とプロセッサを分割することで，セキュリティを高める方向のアーキテクチャが考えられます．このように，どんどん自由に考えていいのですよ．

● じゃ... **TCP/IP** 方式の通信処理は必須のものとなっていますけど，この処理の部分をさらにサーバの奥のサーバに分離し，パケット処理を行うのはどうでしょう．

◆ 奥か，傍かはまあいいとして，処理を分離して安全性を高めようということですね．それも面白いし，重要ですよ．

● この調子だと，特別の入出力処理部を切り離すとか，データベース処理を別に切り出すとか，いろいろ窓口レジスタというインタフェースを介して連絡を取りつつ自律的に動く複合システムの形に全体を再構成するという案もありですね．

◆ そう，そのとおり！ わくわくしてきたでしょう．

● はい．アーキテクチャを自由に考え直せばいいのですね．（対話はここまで）

対話型教材とその利用

通常，講義案を大体立てて，それに応じたテキストや演示教材を作成し，自分の講義の準備をするであろう．そのテキストを公開して他の教員の参考にするという方法があるが，目的や対象の異なる場合，テキストを読み取った上で，また一からテキストを構成すると膨大な手間がかかる．その際，著作権を侵害しないことや自分独自の説明をしたいと考えると大きな負担がかかる．ここで教材案の提示法として，対話型を選んだのは，これを自分の講義の目的にあった教材作成の一種の設計書として利用できるのではないかという考えからである．ここで試みたのは，対話形式にすることで，不要な細部の記述をさけ，大きな流れを示したり，一種の抽象化を図っており，他の講義にも適用する幅を広げて，より広く使っていける方法としたいという意図がある．実際の対話型講義を一段抽象化したもの，あるいは，粗いシナリオと言ってもいい．謎を相手に投げかけ，やりとりを進めていくことでストーリーを記述している．ここで単に一方的に知識を伝える講義でなく，考えさせる講義をという意図も見えてくるであろう．

この例ではあまり感じられないだろうが，シナリオ，あるいは，対話型の場合，その教育の目的や歴

史的説明，その他のメタ情報を◆の言葉として埋め込むことができる．教科書は，著者の意図は前書き等に書かれることはあるが，できてしまった教科書の記述から，その背後の意図を推察することは難しいことが多い．しかし，ここで採用した対話型はそういう記述を埋め込む自由度があることも指摘しておきたい．つまり，教材の見せ方として対話型の可能性をお見せしたかったのである．

情報処理教育をさらに良くするために，もっと教材研究を進め，それを公開し検討する機会を学会としても強化するとよいと考えている．この本稿はそのために，対話型という提示方法の可能性をアーキテクチャ教育の事例で示したものである．なお，この事例はオープンエンドにしてあるが，これに刺激されてその先を考える人が現におられることは筆者のよろこびとするところである．そして，この例に限らず，多くの科目で新しいストーリーで，興味深い，学生を巻き込む講義の可能性が残っていると信じて，教材開発をしていただきたいと考えている．

また，学生に考えさせる講義の具体的な提案を，本号教育コーナー「ぺた語義」に収録していただいているので参照していただきたい．図 -2，図 -4 は久野靖先生によるものであり，ここに感謝申し上げる．

参考文献

1) 清水健一 : 大学入試問題で語る数論の世界，素数，完全数からゼータ関数まで，ブルーバックス，講談社 (2011)，ISBN978-4-06-257743-4.
2) pdp11 handbook, Digital Equipment Corporation (1969), http://research.microsoft.com/en-us/um/people/gbell/Digital/pdp%2011%20Handbook%201969.pdf

（2011 年 7 月 31 日受付）

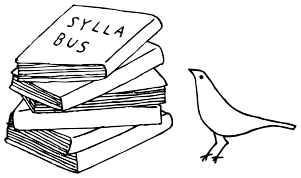
都倉信樹（正会員） ntokura@nike.eonet.ne.jp
CS'90，CS'97，IS'97，J07-IS 等の策定に参画した．大阪大学，鳥取環境大学名誉教授，大阪電気通信大学学長．本会フェロー．

ぺたこん

【第 10 回】

contents

応般 Column



各大学のシラバスを比較してみると

各大学のシラバスは最近，記述が充実してきている．授業は大学が学生に提供するサービスの大きな部分で，授業のシラバスはいわば学生と大学との間の契約書と言える．にもかかわらず（少なくとも情報専門学科について）シラバスから学科の教育内容を客観的に調査しようとした試みはほとんどなかったようである．筆者らは本会コンピュータ科学教育委員会の活動の一環として，シラバスの調査を行った．詳細は本誌 2011 年 8 月号を見ていただきたいが，本稿では調査に際して思ったことを述べたい．

今回，シラバスによって，49 の理工系情報学科のカリキュラムの内容を調査した．カリキュラム標準 J07-CS[1] の知識体系のコアトピック（必修）をどの程度カバーしているかを，15 のエリアごとに，時間数で定量的に調べた．調査の報告からいくつかの結果を述べる[2]．各学科のシラバスと CS 標準を比べてみたものである．CS（Computer Science コンピュータ科学）とは，情報およびコンピュータの学習分野をどちらかというと理論的にとらえたもので，以下は結果のまとめの一部である．

（1）アーキテクチャやオペレーティングシステムなどの，CS で「伝統的な」エリアでは，カバー率は高い．
（2）ヒューマンコンピュータインタラクションなど比較的「新しい」エリアでは，学科間のバラツキが大きい．カバーすべき内容量が少なめのせいもある．
（3）ネットワークエリアは，コアが少なめのこともあるが，カバー率は高い．
（4）ソフトウェア工学は，コアの要求内容は多いが，カバー率は低い．

また，日本の理工系情報学科が，CS 型，IS 型，メディア型，経営工学型などに分かれるのが，エリアごとのカバー率によって分類できそうだと分かった．しかしおそらく米国ほどには明確に分かれないと思われる．

一般に，ある分野の授業を，異なる学科の間でシラバスによって比較することは，教育の内容の改善のために効果が大きい[3]．しかし今回の調査では，そもそも各学科のシラバスの表現の形式が大学ごとにまちまちなのに驚いた．米国の大学科目では，たとえば授業の番号付けなど，形式がもう少しは統一されているようである．

学科間のシラバスを比較しやすくするには，記載の基本的な形式を統一してしまうのが効果的である．シラバス文書に限らないことだが，標準的な様式をたとえばエクセルや XML で作成し共通で使用することを提案したい．学科の教育内容が，シラバスを通して容易に比較できるようになることは，教育内容の切磋琢磨を促し，さらに学士力強化方法の具体化にもつながるであろう．

参考文献

1）情報処理学会カリキュラム標準コンピュータ科学 J07-CS 報告書，151p. (Jan. 20th. 2009), http://www.ipsj.or.jp/12kyoiku/J07/20090407/J07_Report-200902/4/J07-CS_report-20090120.pdf
2）疋田輝雄，石畑　清：シラバスに基づく理工系情報学科のカリキュラム調査，情報処理，Vol.52，No.8，pp.1020-1025 (Aug. 2011).
3）関谷貴之，山口和紀：カリキュラム標準 J07-CS と CC2001CS の比較に関する報告，情報教育シンポジウム，SSS2011.

疋田輝雄　（明治大学）

ロゴデザイン ● 中田 恵　　ページデザイン・イラスト ● 久野 未結

解説

考える講義を目指して

基般

都倉信樹
大阪電気通信大学

コンピュータアーキテクチャに関する教育を活性化したいという問題意識で，別稿「対話で教えるコンピュータアーキテクチャ」[1] では，講義内容のあるストーリーの提案をさせていただいた．ここでは，できるだけ学生にいろいろ考えてもらうことを目指した講義のある考え方について紹介する．

インタラクティブ講義と時間外にもしっかり学習していただく講義

まず，講義はできる限り双方向的にする．そのために，少し説明しては小さい**クイズ**（5 択を数題など）を出す．配布したコピー用紙に自分の回答を書かせる．ごく簡単なクイズなので，一通り机間巡回しつつ回答状況を見て，教壇に戻って回答する．時々引っかけ問題も混ざっており，学生はやられたと笑いつつ，引っかかる原因となった思い込みや定義の不注意な読みなどをしっかり理解する．特に理論系の科目のときは，定義をいい加減に読むとあとあと困るので，そこをたたき込むのに，説明してすぐにクイズで確認するのが効果的である［例は付録に］．こうすると寝たり私語をするのは難しく，大体講義の流れに付いてきてくれる．しゃべりが下手な筆者でもこの方法で学生と双方向のコミュニケーションが成り立っているなと思われる時間を持てる．

クイズは講義の中で特にしっかり注意して理解してほしいことを定着させるためであるが，自習を求める**テキスト**の方の**小問**は，そのパラグラフで読み取ったことを基に問題を解いたり，関連したことを考えさせるものである．学生は講義範囲のテキストの小問を回答してレポートすることを求められる．かなり多数の小問があるし，次週までにレポートする必要があり，かなりのロードになる．講義が終わると学生たちの幾人かは，さっそく勉強会をやったりして，励まし合いながらレポートを早く出そうとがんばっている姿がよく見られた．全部解けとは求めていないが，トップクラスの学生はすべて解答してくれたし，それに続くレベルの学生も奇数番目などできる限り解答してくれた．提出されたすべてのレポートをみて，回答をしっかりしている学生は褒め，それほどでない学生はもうちょっとがんばるように激励したりする．ただ，筆者が回答の採点をするわけではない．それは次の仕掛けによる．

補助教材と称して，すべての小問の解答・参考情報を記したものを学内 Web にアップする．学生には補助教材の対応する解答を読んで**自己採点**することを求める．自己採点が適正かどうかを講義中 2 回ほど筆者はチェックしているが，おおむね適正に自己採点している．その採点が重要なのでなく，とにかく取り組んで自分で考えることが大事であることを学生も理解しており，答えをコピーしてごまかすことはない．理由は解答を見に行くと，通常の問題集の解答のように結果だけが示されているのではない．問題の背景の説明や，どう取り組めばよいか，解くための考え方を解説したり，可能な範囲で，別の考えを示したり，関連問題，発展問題などを含めている．こうして，1 つの小問の解答を見に行って

も，それに関連したかなりの文章を読むことになる．コピペなどまったく意味をなさないのである．「考えさせる」，それを徹底したいという思いである（日本語力の低下も話題に上るが，日常的に文章を多読させることも有用と考えた上である）．

それら大量の解説を読んでもらいたいので，その文章はいろいろ工夫する．題材の選択が最も重要だが，記述形式も工夫する．文献 1）で見ていただいた対話形式を使うこともある．学生が間違いやすいところを両者で検討したり，うっかり陥る発想などを対話しつつ吟味するという形にする．楽しく読んでもらいたいので，いわゆる「ボケと突っ込み」的に書いたり，時には土屋賢二先生（「ツチヤの貧格」文春文庫他，独特の文体で笑いとともに，多少哲学的思考を誘う多数の著書を発表）のスタイルをまねてみたこともあるが，文才の欠如を思い知っただけであった．

筆者は担当していた 4 ～ 6 講義で，この考えでテキストを作成し印刷配布した．ほとんどの科目で，A4（40 × 40）で 200 ページ程度のものになるが，補助教材は pdf で全体ではテキストと同じくらいのボリュームになることもある．このほかに用語解説や過去問，未来問，デモプログラムなど，学生の学習を誘導する種々の情報を学内 Web の担当者のサイトに掲載した（なお，講義は黒板での手書きで，プレゼンソフト類は使わない．講義のビデオ記録は学内 Web で公開しており，就活で欠席の学生などが利用している）．

私の講義では大量のレポートを書かされることを学生はよく知っている．しかし，多くの学生はおどろくほど熱心に自分で解答を示し，自己採点し，そして，毎週「**話題提供**」と称する数百字の身辺雑記的なレポートを書いてくれた．この話題提供にはすべて返信をした．レポートは電子的に提出し，教員はそれにコメントでき，それらを全員が共有する形のシステムを使っている [2)]．かなりしんどい科目である．講義に参加しレポートに取り組むといろいろ深く理解でき，分かったという感じが持てるようにしている．欠席の多い学生を別として，ほとんどの学生は合格圏に入る．多くの学生を相手にする座学で，学生に必要な基礎知識を伝えるだけでなく，いろいろ多面的に考えさせるという訓練をするのに上の方法が使える．特に，代案がいくつもあることを講義中では必ずしも十分には扱えない場合でも，補助教材ではそのことが説明できる．学生は，その具体的な検討をいろいろな事例で行うことになる．

情報処理教育の活性化に向けて

アーキテクチャ，OS，コンパイラ等の分野ではすでに膨大な先人の成果が蓄積されている．それを単に伝える方法では学生は萎縮する．また，現行方式の背後には実に多くの代案があり，現在あるいは少し過去の時点の技術レベルで適切なものが使われているに過ぎない．技術が変われば当然方式も変わる可能性があり，いくつもの代案の中から最適なものを選ぶという考えかたを学生に持ってもらいたい．そして，自分で自由にいろいろ代案を提案するようになってほしい．そういう分野で，どう講義をするか．

そのために，ほぼアイディアが出尽くしたかに見える分野で新しく問題を作り出し，再検討する材料とすることを目指した対話型教材の一案を本誌の文献 1）で提示した．対話型は教材の提示方法の一例として試みたものであって，それ以外の方法でも構わない．また，講義だけでは十分なことを伝えられないのが現状であるが，しっかり学習してもらう講義の仕方の一例をここで示した．学生に確実に分かるようにクイズを活用し，きちんと聞けば分かるのだと思わせ，興味をつなぐストーリーで，たっぷりの時間外学習の小問（ここではいろいろな代案に触れるようにする）を提供し，それをやればやるだけ理解が深まると実感させる講義の提案である．

筆者がこの種の科目を担当した時期はいろいろアーキテクチャについてのアイディアが登場した．新着の雑誌を見ては，「これは面白い，講義で扱えそうだ」と思ったアイディアを手書きのプリントで配って講義する時代で，講義が実に楽しかった．それは過去のことではあるが，今でもなにか楽しめる題材

を見つけて，講義を活性化させること，これが情報教育にかかわる我々の責務の 1 つであろうと考える．

すでに成熟し，なにも新しいことはないなどと諦めず，既成のテキストに囚われることなく自由に発想して，新しいストーリーをどんどん開発し，それを互いに公開して情報処理教育の活性化，レベル向上に使うようになってもらいたいというのが筆者の願いである．

参考文献

1) 都倉信樹 : 対話で教えるコンピュータアーキテクチャ，情報処理，Vol.53, No.2, pp.162-170 (Feb. 2012).
2) 永井孝幸，松前　進，都倉信樹：教員の作業効率向上を目指した授業支援システムの構築と運用，工学教育 , Vol.53, No.2, pp.64-69 (Mar. 2005).

（2011 年 7 月 31 日受付）

都倉信樹（正会員） ntokura@nike.eonet.ne.jp
CS'90，CS'97，IS'97，J07-IS 等の策定に参画した．大阪大学，鳥取環境大学名誉教授，大阪電気通信大学学長．本会フェロー．

付録　ミニクイズの例

講義で出すミニクイズの具体例を示そう．実はいろいろなパターンがある．仮説実験授業と似て，与えられた情報の中で，どれが正しいと思うかという 5 択問題を出して，学生に推理してもらうというクイズも効果的だし，単に学生の意識を知るために，「○○について」という自由記述問題を出すこともある．

理論的な講義では，定義をしっかり理解してもらうことがまず大事で，そこを猛スピードで飛ばすようなことをすれば，学生をそこで振り落とす結果になる．次のようなクイズをやって定義をしっかり分からせると学生は「この講義は分かるように教えてくれる」と理解して，あとの話も聞いてくれる．具体的な例のために，定義を説明する．

「Σ は記号の有限集合で，Σ の上の長さ n の記号列とは，s =（s_1，s_2，…，s_n）のように，n 個の Σ の記号の並んだものである．すなわち，$s_i \in \Sigma$（i=1，…，n）．また，記号列 s の長さ｜s｜は，s に含まれる記号の数と定義する．長さ 0 の記号列を特に ε で表す．すなわち，ε =（），｜ε｜= 0．なお，記号として文字記号を想定するとき，文字列 s は上の記号列の表現から両端の（と）とカンマを取り除いて文字のみを書き連ねた簡略表現形で表す．なお，Σ の上の長さ 0 以上の Σ の上の記号列のすべての集合を Σ * と表す．
例．Σ ={0，1} のとき，これは 0 と 1 という 2 つの文字からなる記号列集合である．記号列（0，1，0）は，010 と簡略表現できる．010 ∈ Σ * である．」

こういう説明をして学生にクイズを出す（この例は正誤判定問題）．

1. **Σ ={a，b，c} のとき，abcadca ∈ Σ *.**
2. **Σ =（0，1）のとき，001 ∈ Σ * であり，｜001｜=3.**
3. **Σ = {a，b，c} のとき，abcdcba ∈ Σ *.**
4. **Σ = {0，1} のとき，｜000111｜=7.**
5. **Σ がなんであっても，ε ∈ Σ ***

以上のような問題を板書し，「**5 つの文章について，それぞれ正しい文章かどうかを答えなさい．しっかり考えてください**」と言って，机間巡回に移る．

解答 5 のみ正みたいだが，句点がないので文章としてはだめ．すべて誤である．学生から「そんなあ」というブーイングがきたら，もっけの幸い．「諸君はプログラムを書くのだから，一字一句にも目を光らせる習慣をつけましょう」と切り返す．実はこちらは問題を黒板の前で即興的に考えるので，頭が回らず，学生に指摘されて気がつくこともままある．引っかけ問題と称しているが，実は教員の過誤に起因するもの多数である．「教員たるもの，教材や問題には一文字も誤りがあってはならない」と厳しく教員に注文する教職の指導者もおられるが，誤りを一切認めない態度は，実は例のナントカ神話のような恐ろしい方向に向くのである．「人間だから」と言い訳するのではないが，間違いにも大きな役割・教育効果がある．なお，こういう定義の確認以外でもミニクイズをしばしば出して，漠然と講義を聞くのではなく，論理ギャップがないようしっかり考えていくことを求めていく．計算問題も上手く出せば可である．また，上の説明で，ε という記号は Σ に含まれていないと暗黙に仮定してその説明はしていない．そういうこともクイズに出してもいい．要は鵜呑みにしない訓練を学生にするつもりならクイズのネタはいくらでもある．

解説

JABEEを通じた大学教育の質的保証（後編）：
ソウル協定と情報分野の分野別要件

応専

筧　捷彦
早稲田大学

JABEEでの情報分野アクレディテーション

前編[1]で解説があった通り，本会での情報専門教育の改革・改善を推進する活動は，JABEEの情報および情報関連分野での認定という形をとって展開されることになった．

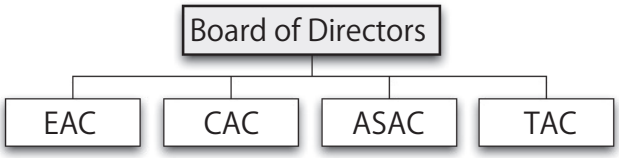
JABEE情報および情報関連分野の分野別要件は，本会と電子情報通信学会とが中心になって策定した．2001年以後のACM・IEEE-CSによるカリキュラム標準の動き（後に本会ではJ07[2]として結実する）に合わせると，そこで取り上げられた多種類の学科類型に対応できるものとしたいところである．しかしながら，あくまでJABEE認定基準の下で用意される16技術分野の1つとしての分野別要件を記述することになるから，学科類型に対応する細別は，例示にとどめてそれぞれの学会でWebページに参考情報[3]として掲げることとした．学科類型としては諸外国との対応を考えて，CS（Computer Science），IS（Information Systems），SE（Software Engineering），CE（Computer Engineering）の4つの領域として示し，あえて日本語名称は与えなかった．2003年度から本格的に情報および情報関連分野の認定が始まり，受審して認定を受ける教育プログラムが増えていった．領域的にはCSやCEに近いものが中心だが，ISやどの領域にも属さないものも少数ではあるが含まれている☆1．

☆1 JABEEの認定は領域の区分を行わないから，この分類は筆者の判断によっている．JABEEに対応すると教育の多様性が失われるとの意見も聞かれるが，それが誤解であることが分かるだろう．

JABEEは2005年に念願のワシントン協定加盟を果たす．それと前後して，情報および情報関連分野で認定された教育プログラムは，ワシントン協定がengineering教育に対する協定であったことをいやが上でも認識させられる．認定がとれるとJABEEからその教育プログラムの英語でのprogram titleを知らせるよう要請がくる．学科名称の英語表記，たとえば，“Computer Science”をそのまま届けようとすると，“engineering”という単語を含んだものにしてくれと再要請がきてしまう．ワシントン協定に加盟しているJABEEが認定した教育プログラムは，“engineering education”を行っているものであり，その“title”には“engineering”という単語が含まれていなければならないというのである．かくて“Engineering in Computer Science”などといった苦肉の策の“title”を届け出るはめになった教育プログラムがいくつもある．

ソウル協定の誕生

ワシントン協定での代表的な認定機関としては，米国のABETがある．そのABETは，engineering教育の認定を行うEAC（Engineering Accreditation Commission）と，computing教育の認定を行うCAC（Computing Accreditation Commission）を別単位として設けている（**図-1**参照）．ワシントン協定に加盟しているのはEACだが，EACではプログラム名（program title）に“engineering”が入っているのを



EAC : Engineering Accreditation Commission
CAC : Computing Accreditation Commission
ASAC : Applied Science Accreditation Commission
TAC : Technology Accreditation Commission

図 -1　ABET の組織

原則として基準が作られている．これに対して CAC は，プログラム名が“Computer Science”，“Information Systems”，“Information Technology”などの教育プログラムを扱うという構図である．ワシントン協定加盟の他の国の例でみても，一般に情報分野の教育プログラムは，Computer Engineering や Software Engineering などと“engineering”という語を含んだものだけが扱われている．

JABEE がワシントン協定加盟を果たしたのだから，情報および情報関連分野で認定を受けた教育プログラムは，ABET-CAC で認定を受けた教育プログラムとの相互承認が行われることになってほしいと思うが，残念ながら，ワシントン協定はあくまで engineering 教育認定の相互承認協定であって，たとえば ABET-CAC での認定は対象外なのである．

同じ問題意識を他の国々の情報処理関係の学会の人たちも持っていたとみえて，2006 年 7 月に IEEE-CS の呼びかけで IFIP 傘下の大規模学会の代表が集まってもたれた相互連携に関する会合での意見交換や，韓国 KIISE 会長来日の際の非公式な意見交換があった後で，韓国が動いた．韓国 ABEEK が，米国 ABET，オーストラリア ACS，英国 BCS，カナダ CIPS，日本 JABEE に声を掛け，情報専門教育認定の相互承認協定の設立を目指すべく 2007 年 11 月にソウルで会議が開かれたのである．その会議で，その協定をソウル協定と名付けて，早期に設立を目指すとした Seoul Declaration が採択されたのである．併せて，協定設立に必要となる，定款案等の策定，対象教育の標準設定，経営的枠組み案の策定の 3 つの WG を設けて準備を進めることとなった．

図 -2　ソウル協定

JABEE では，検討 WG を置き，JABEE 会員の中で情報専門教育認定に関係の深い本会，電子情報通信学会，電気学会，経営工学関連学会協議会から委員が出てこの動きに対応した．2008 年 6 月には WG の準備状況を確認する会議がソウルで開かれ，同年 12 月にソウルで開かれた会議で，各国の認定機関の代表が協定に調印してソウル協定が始動した（**図 -2** 参照）．最終的には，それぞれの認定機関が批准を済ませて正式に発足した．

ソウル協定は，協定書本文，定款・規則，ガイドラインの 3 文書[4]からなる．加盟機関を会員として 2 年に 1 度開催される総会ですべてを決定する．2 年任期で選出された議長が必要な業務を事務局とともに実行する．現在，ABET-CAC の Joe Turner 氏を議長とし，ABEEK が事務局を務めている．その後に加盟した台湾 IEET と香港 HKIE を加えて 8 カ国の認定機関が会員である．

ソウル協定の卒業生属性

ソウル協定準備の過程で，対象教育の標準設定 WG の議長を務めたのは現協定議長の Joe Turner 氏であった．Turner 氏は ACM Computing Curriculum

ソウル協定	ワシントン協定
知識（基盤・専門）	知識（基盤・専門）
問題分析力	問題分析力
問題解決・デザイン力	問題解決・デザイン力
	調査力
ツール活用力	ツール活用力
社会的責任	社会的責任
	環境・持続可能性
倫理	倫理
チームワーク力	チームワーク力
コミュニケーション力	コミュニケーション力
	財務・管理運営
生涯学習	生涯学習

表 -1　卒業生属性定

国際的相互承認の枠組み	JABEE の枠組み	情報分野の対応
ワシントン協定	エンジニアリング系 • 学士課程プログラム • 修士課程プログラム	情報通信，コンピュータおよびその関連の工学分野 •CE（Computer Engineering） •SE（Software Engineering）
ソウル協定	情報専門系 • 学士課程プログラム • 修士課程プログラム	•CS（Computer Science） •IS（Information Systems） •IT（Information Technology） • 情報一般
国際建築士連合	建築系 • 学士修士課程プログラム	

表 -2　JABEE の国際対応

1991 策定の議長を務めた人である．

ソウル協定で定めるのは，CS や IS といった個々の学科での教育内容ではなく，協定対象の教育認定対象プログラムが修了生にどんな力を身に付けさせているものであるかである．こうした標準としては，エンジニアリング教育を対象とするワシントン協定・シドニー協定・ダブリン協定が用意する“卒業生属性”(graduate attributes)がある．これにならってソウル協定でもその卒業生属性[4]をガイドラインの附属書として定め，これを相互承認における対象教育の標準とすることになった．ワシントン協定のものと比べると，財務に関する項目などを省いたものになっている(**表 -1** 参照)．

JABEE のソウル協定対応

JABEE の検討 WG は，まず，ソウル協定に対応して情報専門系教育プログラムの認定を，従来からの認定とは別建てにして行うべきであるとした．ソウル協定に加盟することの第一の意義は，これまで国際的な相互承認の枠外に置かれていた情報専門系教育プログラムの国際的相互承認が得られるようにすることにある．具体的にいうと，ABET の例を見ても，CS，IS，IT が情報専門系の認定対象なので，これらを主に扱うこととし，国際的にはエンジニアリング教育としてワシントン協定で扱われている CE と SE は従来からの認定に残すこととした．情報系として CS，CE，SE，IS，IT を 1 つの分野にまとめておきたいという希望よりも，教育認定の国際的対応を重視したのである．

こうして，新しく情報専門系教育プログラム用の認定基準を定め，その中に CS，IS，IT の分野を置くことになった．また，これから出てくると思われる新しい内容の教育プログラム（たとえばメディアを主とするものなど）にも対応できるように，“情報一般”という分野も設けることとした．なお，ワシントン協定対応の従来の認定基準では，これまでの“情報および情報関連分野”は廃止し，これまでの“電気・電子・情報通信およびその関連分野”を 2 つに分けて“電気・電子および関連の工学分野”と“電子情報通信・コンピュータおよび関連の工学分野”として後者の中で CE と SE も扱うこととなった（**表 -2** 参照）．なお，この新分野で認定を受けたプログラムも情報専門系教育プログラム認定基準によって認定を受けたプログラムも，その修了生は技術士情報工学部門の第一次試験が免除される．

情報専門系教育プログラム用の認定基準[5]は，JABEE 認定の枠内にあって，なおソウル協定の定める卒業生属性に対応できるものにする必要がある．それには，基準 1（1）に掲げる項目 (a) ～ (h) で対応を取ることになるが，従来の (a) ～ (h) では，チームワーク力とツール利用力が抜けている．ツー

従来からの基準	情報専門系教育プログラム用基準
(a) 地球的視点から多面的に物事を考える能力とその素養	(a) 地球的視点から多面的に物事を考える能力とその素養
(b) 技術が社会や自然に及ぼす影響や効果，および技術者が社会に対して負っている責任に関する理解（技術者倫理）	(b) 技術が個人・組織・社会に及ぼす局所的・全体的な影響を分析する能力，技術者に要求される職業倫理，法的・社会的な責任，および情報セキュリティに対する責任に関する理解
(c) 数学，自然科学および情報技術に関する知識とそれらを応用できる能力	(c) 数学（離散数学および確率・統計を含む）および自然科学に関する知識とそれらを応用できる能力
(d) 該当する分野の専門技術に関する知識とそれらを問題解決に応用できる能力	(d) 該当する分野の専門技術に関する知識とそれらを問題解決に応用できる能力
(e) 種々の科学，技術および情報を利用して社会の要求を解決するためのデザイン能力	(e) 問題を分析し，モデル化を行い，その解決に必要な情報処理上の要件を抽出し定義する能力，および，与えられた要求に対して，各種制約の下でコンピュータを用いたシステム，プロセス，コンポーネントまたはプログラムをデザインし，実装し，評価できる能力
(f) 日本語による論理的な記述力，口頭発表力，討議等のコミュニケーション能力および国際的に通用するコミュニケーション基礎能力	(f) 論理的な記述力，口頭発表力，討議等のコミュニケーション能力および国際的なコミュニケーション基礎能力
(g) 自主的，継続的に学習できる能力	(g) 自主的，継続的に学習できる能力
(h) 与えられた制約の下で計画的に仕事を進め，まとめる能力	(h) チームとして計画的に目標を達成していく能力

表 -3　JABEE 基準 1（1）学習・教育目標に具体化して設定すべき項目

ル利用力の方は，(e) のデザイン力の項目の中で扱うことにすることとし，チームワーク力だけは項目として追加することとした．ただし，なるべく従来からの認定基準と項目数などの枠を合わせておくことを原則として，従来の (h) にあった計画遂行力を (e) に含めることにして，(h) の内容をチームワーク力に置き換えることとした．(e) デザイン力は，以上のことを踏まえ，また情報専門系に特化して，その内容を一新した．また，以上に述べた大きな変更のほかにも，情報セキュリティについての責任の認識を持たせること，数学として離散数学と確率・統計も身に付けさすことなど，こまかな修正を施した（**表 -3** 参照）．

基準 1 のほかには，基準 2 (2) の学習保証時間に対する要求を情報専門系教育プログラムに適した形に書き直した．この 2 つのことがらを除くと，情報専門系教育プログラム用の認定基準は，従来の基準と同一である．

2008 年度には，この情報専門系教育プログラム用の認定基準に基づいて，(a) すでに継続審査を経験している教育プログラム，(b) 継続審査は未経験ながら次の審査としては継続審査が来る教育プログラム，(c) まだ審査を受けたことのない教育プログラムの 3 つのプログラムについて認定審査の試行が行われた．その結果，基準 1 (1) (h) として新たに設けたチームワーク力についても，教育プログラムが対応可能であることが分かった [6), 7)]．

2010 年度から情報専門系教育プログラム用の認定基準に基づく認定審査が始まった．ただし，移行措置として，2010 年度と 2011 年度は従来からの基準での情報および情報関連分野で受審することも可能となっている．そして，実際に 2010 年度には，情報専門系教育プログラム用基準に基づいて認定された教育プログラム第 1 号が誕生した [8)]．

JABEE の基準改訂と教育改革

情報専門系教育プログラムに関しては，ソウル協定への加盟という新事態があって，その卒業生属性に対応するために，認定基準の，とりわけ基準 1 (1) の変更が行われた．同じことは，ワシントン協定に加盟している JABEE のその他の技術分野認定につ

いても行う必要があった．また，建築分野が，国際建築士連合の国際相互承認が得られるように学士修士一貫の 6 年制教育プログラムの認定を設けたこともあって，これらを統括できる形の認定基準に整備し直す必要性もあった．

こうして，JABEE は，設立以来，初めて大幅な認定基準の改訂を行い，2012 年度から実施することとなった[9]．その柱は，(1) 学士，修士，学士修士一貫などの種別，エンジニアリング系・情報専門系の枠組みの違いを超えて統一的な基準提示とすること，(2) PDCA サイクルに即した基準項目配置にすること，(3) アウトカムズ評価の考え方を徹底して学習時間など教育内容に関する細目を縛る条件を除くことにある．この結果，文書としては，JABEE 認定のすべてに共通する共通基準，基準種別ごとの個別基準に分けることとなった．また，これまで，直接には文書化したものがなかった JABEE 認定の大枠を示す“技術者教育認定にかかわる基本的枠組”を用意した．従来の基準 1 (1) は，従来の (a) ～ (h) に (i) チームワーク力を加えたものに改められた．したがって，ソウル協定に対応する情報専門系学士課程認定もこの共通基準に従った上で，個別基準で各項目について勘案事項を与えて 2010 年からの情報専門系教育プログラム用認定基準と実質的に同じ内容としている．なお，2015 年度までは 2012 年度制定の新基準への移行期間とし，プログラムは従来の基準によって受審してもよいことになっている．

2012 年度からの JABEE の基準改定は，教育の質の保証と教育改善を目的とするという JABEE の原点に立ち返って，その認定審査全般を改めていくという方針の表明でもある．世界を見渡すと，エンジニアリング教育はもとより，大学教育についての改革が進んでいる．知識伝達の教育から，学生自らが学ぶ教育へ，学生自らが学ぶことを学ぶ教育へと急速に切り替わっている．JABEE による認定―アクレディテーション―もそうした教育改革を推進するのに資するものに変貌をとげようとしているのである．

ここでは情報系プログラム認定に関してもっぱら JABEE の側からの視点で解説した．これに対して，実際に認定を受けた愛媛大学からの解説が 4 月号に掲載を予定されている．また，文部科学省・産業界・認定校を含めたより広い視点からの特集「大学教育の質的保証」が 7 月号に予定されているので，併せてお読みいただけると幸いである．

参考文献

1) 牛島和夫：JABEE を通じた大学教育の質的保証（前編）：大学教育改革とアクレディテーション，情報処理，Vol.52，No.12，pp.1562-1566 (Dec. 2011).
2) 筧　捷彦，他：特集「情報専門学科カリキュラム標準 J07」，情報処理，Vol.49, No.7 (July 2008).
3) 情報処理学会アクレディテーション委員会，http://jabee.ipsj.or.jp
4) Seoul Accord Documents, http://www.abeek.or.kr/accord/contents.jsp?menu_l=144
5) JABEE，日本技術者教育認定基準（ソウル協定対応プログラム用）2011 年度適用，http://www.jabee.org/OpenHomePage/joho/criteria2011_cac_110207.pdf
6) ソウルアコード―情報専門学部教育認定の国際相互承認―，第 71 回全国大会シンポジウム (5)，http://www.ipsj.or.jp/10jigyo/taikai/71kai/71program/html/event/index.html, (Mar. 2009).
7) 佐渡一広：ソウル協定の現状報告，第 106 回コンピュータと教育研究発表会，Vol.2010-CE-106, No.6 (Sep. 2010).
8) JABEE, Computing and IT-Related Programs Leading to Bachelor's Degree Accredited by Japan Accreditation Board for Engineering Education (JABEE), http://www.jabee.org/english/OpenHomePage/Computing&IT-related_programs_bachelor's2010.pdf
9) JABEE，2012 年度基準改定について，http://www.jabee.org/OpenHomePage/accreditation_o.htm

(2011 年 11 月 16 日受付)

筧　捷彦（正会員） kakehi@waseda.jp
1970 年東京大学大学院工学系研究科修士課程修了．同大工学部助手，立教大学理学部講師・助教授を経て，1986 年早稲田大学理工学部教授，2007 年から早稲田大学理工学術院教授．日本ソフトウェア科学会，ACM 各会員．本会フェロー，情報処理教育委員会委員長．

応般 特別 Column

お大師様を訪ねて（1）　本来無一物

学会役員選挙の Web 投票システムではパスワードを入力するようになっている．このパスワードは英数字で構成された重なりのないパスワードである．パスワードを生成する乱数発生器はアルゴリズムの大家として知られる早稲田大学の筧捷彦教授に作成を依頼した．

「何でまた私が」など言いながらも先生は快く引き受け，旬日のうちに Java コードを送付してきた．このコードの出来はかなり良く流石は筧先生と感動したが，実は M2 学生の作品であった．学生が作ったのでは無料と言う訳にもいかず，バイト代として 10K 円支払うというと，当該学生は「別件のバイトで作ったコードを流用したので要らない」とのことで，それではとそのお金は筧研究室で浩然の気を養う足しにされた（と思う）．

この乱数発生器の要求仕様は曖昧さのないように（元富士通の私としては）気を入れて書いた．筧先生はこのしっかり書かれた要求仕様と傑作 Java コードをこのままにしておくのはもったいないとのワンガリ・マータイ精神で，コードの所々を隠した穴埋め式試験問題に流用した．「お陰様で試験問題を一問考えないで済みました」とは転んでも只では起きない筧先生のお言葉である（「勿体無い（もったいない）」とは，仏教用語の「物体（もったい）」を否定する語で物の本来あるべき姿がなくなるのを惜しみ嘆く気持ちを表している，とは Wikipedia の解説である．一方お大師様（弘法大師のこと）の教えには「本来無一物：万物は実体ではなく空にすぎないのだから執着すべき対象は何 1 つない」というのもあり，四国歩き遍路により俄仏教徒となった私はますます混乱の極みなのである）．

さて当該学生プログラマのその後であるが，私としては当然日立や富士通等のメーカに就職し，東証・みずほ・JR・全日空と不祥事続きの日本のソフトウェア業界に新風を吹き込んでくれることを期待したのであるが，任天堂でゲーム情報学に邁進することとなった．メーカ育ちの私としては大いに嘆いた次第だが，国立大学工学部を卒業して寿司職人になった大学院生もいるとのことなので，それに比べれば情報処理に関与しているだけまーいいかと諦めの気持ちである．

湖東俊彦（日本信頼性学会）

「ぺた語義」のコラムではさまざまな方に教育をテーマに自由に書いていただいています．在任中大変お世話になった，湖東前情報処理学会事務局長は，教育について一家言お持ちなので，ぜひにと依頼したのですが，書かれたいことが山積みとのことで，検討の結果，今号から 5 回（予定）に渡って連載コラムとして掲載させていただくこととなりました．お楽しみください．

（ぺた語義編集委員　久野　靖）

イラスト ● 久野 未結