特集号のガイダンス

古 村 哲 也

1. はじめに

出典は不明だが，「ソフトウェアは産業革命以来人間が創り出した最も複雑な構築物である」，という文があった．土木や建築のような伝統ある工学に比べ，わずか半世紀で史上最も複雑な構築物を作るのだから，我々は非常に難しい仕事をやっているわけだ．伝統は短いがソフトウェア工学の立場から，構造化プログラミング，ウォーターフォール・モデル，抽象データ型，構造化分析・設計法等々，品質の良いプログラム作りとソフトウェア生産性向上のため，様々な提案がなされてきた．しかし，バグ無しソフトウェアの実現とか開発効率の面では，依然として問題を解消していない．WS や PC の普及に伴い，かなり優秀なインフラ・ソフトウェアやミドル・ソフトウェアが，低価格で提供されている．これらを利用してユーザ自身が，自前のソフトウェア開発をすることが一層容易になってきている．

ソフトウェア業の仕事は，①流通ソフトウェアの提供，②開発作業の援助（導入支援），③顧客のアプリケーション開発の三つが考えられる．かつて弊社は①から③を万遍なくやっていたが，売上の中心はメインフレームを中核としたハードウェアビジネスであった．オープン化の進展に伴い，ソフトウェアビジネスの売上に占める割合は急増している．特に③の生産効率は満足できる水準には達せず，同業他社でも同じ傾向のようだが，その改善が急務である．弊社では③の割合を上げると共にその採算性を向上させ，新しいビジネスの柱とすることを主たる目的として，変革に取り組んでいる．

かつての計算センターでの経験から判断すると，顧客から開発を依頼される仕事の中味は，圧倒的に（多分 80% 以上）業務アプリケーション，いわゆる事務計算である．事務計算は誰でもできるという気分があるが，実は複雑なデータ構造，数多い例外処理，面倒な分岐条件設定，多種多様な出力表要求がごく普通で，業務アプリケーションの開発は非常に難しい．技術計算はモデルの分かり難さと，数式の威圧感のために難しいと思われている．技術計算独特の難しさ，例えば解の安定性，収束条件，計算誤差の処理等はある．しかし，システム開発の立場では事務計算が難しく，技術計算はシステムの複雑性に限れば簡明である．したがって，数値計算に属するシステムは，ここでは一応議論の対象外とする．だが，技術計算ソフトウェアもユーザ・フレンドリーになっていかざるを得ない．ユーザの使いやすいマン・マシン・インタフェースを望まれる傾向が強まれば，やがて事務計算と同じ問題に直面するだろう．

ERP を初めとして各種の既製業務パッケージ環境が整備されており，情報処理技術（IT）の方向はそちらに向いているようだが，大型のビジネス・アプリケーション開発の芽は枯れていない．アメリカの現状を見てきた人によれば，「米国の先進的金融システム開発の現場では，金利の設定，顧客分析，デリバティブ戦略などに IT を適用することによって，意思決定の精度とタイミングの大幅な改善が達成されつつ

ある．システム開発と同時にビジネスの方法まで改善している．利用されたITは，DB，情報マイニング，数理計画法，シミュレーション，微分方程式，統計解析等広い範囲に及び，金融実務経験者のみならず，学者，軍事，原子力，宇宙開発，OR，計算機科学等の専門家に負うところが大であった」，と言う．このような知識の集約をネットワークを介したコラボレーションを通じて実現する過程で，開発会社は巧まずしてネットワークでの共同作業運用管理のノウハウも手に入れたらしい．金融システム開発は，金融の専門家だけに依存しているのではない．この種の洗練されたソフトウェアの創出は，既成部品の組み合わせだけでは間に合わない．高度な情報処理技術が求められ，衆知を集めた開発が必須である．開発業務が成功しない理由に，“業務知識の不足“が挙げられるが，業界のベテランでさえ業務全般にわたる知識を持つのが難しくなっている．専門業務知識は重要であるが，情報処理技術者に全ての習得を期待する必要はない．しだし，知識獲得能力言い換えれば理論形成能力は万全でなければならない．むしろ，システムの実装について詳細な情報を知っている我々が，システムをデザインし，必要な知識と能力を調達する役割を果たすべきである．「ビジネス・アプリケーション開発こそは，我々が主として手掛けるべきものである」，との信念に基づいてこの号を発刊する．

ソフトウェア開発技法の間口は広い．CASE/4 GLを含む言語，DB，テスト技法，ネットワーク・コンピューティング，エージェント技法等々．しかし，一冊の技報に全てを総花的に採り上げるのは，量的・質的に得策ではない．ソフトウェアの開発は，対象を具体化するのが出発点であるから，本号はモデル化と仕様記述とに的を絞った．「オブジェクト指向（OO）で考え，モデルと仕様を形式的（OOの記述記法を含む）に記述する」ことを開発の第一歩と考え，本号では“OOと形式仕様記述”を採り上げた．他のテーマに関しては，今後の企画を待ちたい．

全ての技法には二つの重要な側面，“厳密さと分かり易さ”がある．OOは分かり易さに優れ，形式手法は厳密さを極端に重視する．両者には明らかに視座の違いは存在するが，決して背反するものではない．性格の異なる技法を組み合わせることで，ソフトウェアの設計・作成に有効な道を切り開くことを期待する．

2. オブジェクト指向

SEは初めに開発対象を観察し，モデル化し，記述する．我々は多くの場合に実装まで引き受ける．従来のモデルは，事務処理現場の手続きを先ず大雑把に把握し，順次それを精緻化するプロセスの記述が主であった．処理プロセスの記述をシステム分析と言っている場面をしばしば目にするが，それはモデル化とは意味が違う．処理は手続きとして理解できるが，システムの仕様はもともと非手続き的に書かれるものである．モデルを考えるとき最も留意すべきは，初めに実装を配慮するのではなく，実世界の概念を使ってシステムの基本原理を，出来る限り抽象化することである．現存手順の率直なコピーは，小規模であれば実現可能であるが，ある程度の規模以上では見通しが立たない．仮に成功しても後の変更への追従が難しい．現行の手続きを，単にコンピュータに載せかえるだけでは，コンピュータ化の意義が乏しい．人手による書類交換が作り上げた過去の事務処理方法は，その体制下での合理性を持っている．

その合理性がコンピュータには非合理性に働く可能性がある．電算化は別の体制下での作業であるから，そこで完備するシステムにしなければ意味が無い．モデルを作成し新システムの振舞いを検討すべきである．システム分析の段階では主として情報モデルが，広い範囲をカバーし実際的である．

システム分析から実装をシームレスに行える手法として，オブジェクト指向が定着してきた．OO はデータ構造と振る舞いが一体となったオブジェクトの集合として，ソフトウェアを構成する方法である．従来のプログラム言語では，データ構造と操作の結合関係が非常に弱かった．OO の方法論は適用領域のモデル作成，システム設計に分かりやすい図が用意され，問題の本質を掴むばかりでなく，ソフトウェア修正にも役立つ．またソフトウェアの部品化と再利用を行い易くし，ソフトウェア開発の生産性を向上させ，保守性をも改善する技術として，OO への期待は大きい．

OO の原点は 1960 年台の半ばに登場した，離散型シミュレーション言語の Simscript[1]や Simula[2]といわれる．クラス，継承，部品化・再利用，ポリモルフィズムという発想は無いが，共通な性質を抽象してカプセル化したり，時間と操作の順序の記述, 値の変換を行うシステムの側面の記述等, 基本的な発想が OO に近い. Simscript でのシステム開発は，モデルを完全に作成し，それをベースにプログラムが設計され，最後に実装が実現するプロセスを踏む必要があった．思い付く順にコーディングし，最終的に辻褄を合わせる開発ができない点に意味がある．モデル化は現実世界の問題を分析し，システムの仕様として抽象世界を記述する手続きで，情報処理技術者にとって本質的な仕事である．

数年前に我々は，電力中央研究所の原子力発電所の運転員行動モデル・システム[3]を開発した．当初，同研究所の原子炉技術者は，ソフトウェアを専門とする我々の理解を助けるために，フローチャートで運転の実状を説明された．しかし，それがかえって理解を妨げるケースがあった．受けた説明を基に OMT（Object Modeling Technique）によりモデル化した．彼らは初めこそ見慣れない記法に戸惑ったが，直ぐに理解し以後は OMT でモデルを作成するようになった．柔軟で強力な抽象化能力がある OO は，情報処理技術者ばかりでなく他分野の技術者にも，容易に馴染ませる力がある．優れた発想が優れたモデルを創る良い例を経験した．

オブジェクト指向型の思考過程は，個別モデル作成ばかりでなく，様々なプロジェクトを通じてビジネス・プロセスを分析し，ビジネスの基本パターンの作成に有効である．ビジネスの変化に応じてシステムも変化するから，変更箇所を極少にしたり基本構造の頑健性は当然として，構成部品の取り替えを容易にするためには，顧客のビジネスの正確な理解と，ホットスポットの局所化が要求される．ビジネス・プロセスを正しく分析することにより，デザイン・パターン利用の展開も開ける．OO の定着はそのような可能性も含んでいる．

1997 年当社の OO ワークショップで，著しい成果が報告された．某社の資材購買システムを OO でモデルを作り，SYSTEM ν［nju：］（弊社開発の分散トランザクション機能を持った ORB）で実装する開発をしたときのことである．従来はシステムの分析・設計と実装は，概ね全体の半々の工数を要していたが，このシステム開発では分析・設計に 7 割，実装が 3 割で，しかも全体工期が短かった．このようなコスト

配分の場合は，プログラムの大きさで開発費を割り出す現行方式では，少々ステップ数の見積りにリスクを見込んだくらいでは，費用回収に追い付かない．分析とかモデリングが商品にならなければ，SE として収益が確保できない．仕事の進め方，見積り方法にパラダイム・シフトが起こっているのだ．

3. 形式仕様記述

モデルはシステムが何をやるか（what）を決めることで，我々の言葉で外部仕様と言う（顧客の要望をまとめた要求仕様も同じものと考える）．モデル作成の次の作業は外部仕様の表現である．分析結果やモデルは，明快に正確に記述されなければならない．外部仕様の延長に内部仕様がある．内部仕様は，外部仕様をどのように実現するか（how）を扱うものであり，記述は実装者サイドに委ねられる．ここでは外部仕様を対象にし，内部仕様には立ち入らない．外部仕様の記述には自然言語を用いるのが一般的である．自然言語の柔軟性と記述力の強さは，極めて有力な武器であるが，反面それ故に曖昧さを内蔵し，厳格な情報の授受には向いていない．厳密さを優先する言語に，形式仕様言語とプログラム言語がある．形式仕様言語は自然言語に強い制限を付けて，記述は勿論，解釈の仕方まで規定するものである．一方，コンピュータの言語理解は極めて単純で，CPU の制御実行の順序で与えられる．これを行うのがプログラム言語で，一切の紛れの無い正確さを要求される．したがって，プログラム言語では，参照が不透明になりシステムの正当性が見え難く，現実事象の完全な表現は困難である．「形式手法で仕様を書く時間でプログラムを書けばよい」という主張は，この事実から棄却される．形式仕様言語はプログラム言語と独立の存在である．

ワインバーグ[4]は，プロジェクトが成功する 3 条件の第 1 に，「要求仕様を理解する」を挙げている．開発業務につまずく最大の原因は，明確な仕様を欠いたまま作業を開始すること，別の言い方をすれば形式仕様の重要性の欠如にあること，に基づくものであろう．要求仕様の不完全さによる不都合，作業のやり直し，工程遅延，それに伴う生産コスト高等が，情報処理業界を悩ましている．IPA ではこれに対処するために，創造的ソフトウェア技術の研究で，開発現場での要求仕様書作成のためのシステムを開発している[5]．不明確な仕様のために難航したプロジェクトは管理を強化しても，立て直すことはできない．問題は管理の精度にあるのではない．技術の問題を管理の仕掛けでは解消できない．技術的に認められた手順を，正確に実行するべきである．紛い物の仕様から仕事を始めないのが原則だが，もし仕様が紛い物なら正すしかない．技術者に本来の仕事のみを求める集団になりたい．

仕様を正しく理解するために，形式仕様言語を採用する必要がある．開発規模が大きくなればなるほど仕様記述は大変になるが，それだけ曖昧性が減少し開発業務が健全に進み，工期の短縮に役立つはずである．ただし，形式仕様言語に，モデルの正しさの証明まで期待できない．矛盾を見つけやすくなってはいるが，それを発見するのは個人の洞察力である．

VDM や Z の名前は昔から知られている割合に，我々の近くの現場で使われていない．形式手法はヨーロッパのもので，日米では冷ややかであると思われていた．最近なってアメリカの事情が分かってきたところ，米国では軍事方面で使われており，そ

のために情報が伝わらなかったらしい．日本だけが形式手法の埒外に居る．理由はユーザ側と推進者側双方にある．

大部分のソフトウェア従事者にとって，形式仕様は難しいとか，取っ付き難いという先入観がある．しかも形式手法は仕様を書くだけで，プログラムは別に作成する二重作業を強いるもの，との誤解がまかり通っている．開発作業が危機に陥ると，マンパワーの増員と，次にその方面の専門家を派遣することが多い．この方法は大量のマンパワーと時間を費やし危機を乗り越えても，しばしば課題含みのプログラムを多く残して終わる．この日本特有の全社一丸・戦闘精神が，我が国での形式手法定着の悪さの所以であろうか．

他方，形式手法の推進者側の罪も深い．形式仕様の理解に，集合論と述語論理は避けて通れない．しかし，多くのソフトウェア従事者は，それらの数学的素養が満足とはいえないし，本質的にプラグマティストである．合理性をいくら説いても，それだけで大衆の理解を得るのは困難である．逆にペダンティックな態度への，いわれ無き反発もある．しかも論文の例題は，スタックとかソートの単純なモデルが多い．大衆は「形式仕様は分かりきった小課題を，数学的に記述する学術的なもの」と思い込んでいる．役に立つことを実例で教えるのが一番の早道と思う．必要な数学的知識は後で勉強すればよい．

仕様言語は何種類か存在している．OO の世界でも人により様々な記述法を提案している．分かりやすく，厳密さを保つためには，一通りの記述法では耐えられないものだ．提案者の“つもり”を反映するために，複数の表記法が世に出る．OO では記述の形式性を重んじて，UML（Unified Modeling Language）が提案されている．OO と形式手法の親和性を示す一端である．モデルをどの記法でどう表現するかは，モデル作成者に付きまとう課題で，各人の判断が優先する．

4. ガイダンス

従来の技報は，情報処理の先端技術者を対象に，分野別に先端的な話題を提供してきた．今回は趣を変えて，OO も形式仕様も殆ど知らない技術者をメインの読者に想定して，チュートリアル・エディションとして作った．したがって，両世界の専門家の鑑賞に耐えるものではない．

本号は，以下に概要を紹介する解説編および実装編から構成される．

[解 説 編]

「オブジェクト・モデリングへの期待」は，オブジェクト・モデルの概要の理解，どんな場面でどのように有効か，開発のプロセスはどうなるか，将来の期待とトレンドはどうか等を解説している．OO のモデル作成技術を修得したい人に向けたものである．

「オブジェクト指向モデリング技法 BOAD 法」は，弊社で開発した BOAD 法の設計思想と使い方を解説したものである．ビジネス・アプリケーションに特化したオブジェクト指向の方法論として，独自の存在感を主張する方法である．ビジネス分野での OO の適用に関心がある人には，参考になると思う．

「オブジェクト・モデリングと実装」は，OO への考察と意見，モデリングから実

装で言われているシームレスな展開とは何であるかを，OO 関連メンバーによる E メールのディスカッションの記録である．OO をやりながらも確信を持てない人，DOA に拘る人，OO に疑問を持つ人，そんな方々には是非読んで頂きたい．

「要求仕様の形式的記述」は，要求仕様とは何か，何故必要か，誰がどう記述すべきか，等仕様に関する幅広い話題に触れている．情報処理の新人達，実装作業の担当者，要求仕様書を作成してきた人には，仕様の意味と記述の重要さが分かるはずである．

「形式仕様言語の現状」は，世上に名が知れている仕様言語を採り上げ，言語の特徴とそれが適用されるアプリケーションのドメインを解説している．ここで書かれた内容は各言語の概要であって，直ぐに使える知識を身に付ける目的ではない．更に進んだ知識を知るためには，文献案内が役に立つ．

「形式仕様言語 Z による開発事例」は，約 2 年前に当社の情報システム部のアプリケーションを，Z で開発したときの成果報告である．ある程度の規模の開発例として，最初の成功ケースであった．

「文献案内」は，OO と形式仕様に関する文献の紹介と，読み方のガイダンスが紹介されている．OO 関連は日本語の本と翻訳本が大量に流布していて選択方法が難しい．一方，形式仕様関連は未だ原書本が中心であり何を入手するか悩ましい．事情通の目から評価された文献を読むことにより，読書のパフォーマンスが向上することは請け合である．

[実 装 編]

実装編は，例題のモデル化と実装に絡む報告である．モデルや仕様書を渡すだけで，実装結果がどうであったかを評価し，今後のモデル作成とプログラムの外注に参考となればありがたい．

実装は，UML でモデル化したものと，仕様言語で書かれたものを基にして，北京の科優公司に依頼した．これは，モデル化と仕様作成段階には全く参加していない科優が，文書だけの情報で如何に実装が可能かを実験する試みである．当社と該社間の情報手段は電子メールのみとし，開発途上の情報交換の記録を保存した．それぞれでどんな質問のやり取りがあったかは，実装編を参照されたい．

参考文献
［1］ P.J. Kiviat, SIMSCRIPT II Programming Language, Prentice-Hall, 1969.
［2］ O. Dahl, K. Nygaard, SIMULA A Language for Programming and Description of Discrete Event Systems, Introduction and User's Manual, Norwegian Computing Center, Oslo, Norway.
［3］ 羽田昭裕，OMT を使った「発電プラント運転員行動シミュレーションシステム」の開発，ユニシス技報 41 号，1994.
［4］ G.M. ワインバーグ，D.C. ゴーズ，要求仕様の探検学，共立出版，1993.
［5］ 古宮誠一，インタビューによる要求抽出作業を誘導するシステムの実現方法，情報処理振興事業協会論文集，1998.

執筆者紹介　古　村　哲　也（Tetsuya Komura）

1962年慶応義塾大学卒業．同年三菱原子力工業(株)入社．三菱計算センターで主としてORのプログラム開発とモデルングを担当．1969年に日本ユニバック総合研究所入社．データーベース関連の研究・開発，社会システムの開発に従事．日本ユニバック(株)に移籍後は，各種応用ソフトウェアのデザイン，開発に従事．日本統計学会，日本オペレーションリサーチ学会，日本シミレーション＆ゲーム学会，Informs，American Statistical Association会員．