# 寄稿 「データ･ファースト」な設計スタイルを身につけよう

ディービーコンセプト代表　渡辺　幸三

＜著者紹介＞
渡辺幸三（わたなべこうぞう）。業務システム開発者。システム設計手法である「三要素分析法」の提唱者。設計ツール「X-TEA Modeler」、実装ツール「X-TEA Driver」の開発者。システム設計に関する多くの著書がある。近著は「システム開発・刷新のための データモデル大全」（日本実業出版社）。ブログ「設計者の発言」。

＜本文＞
筆者は 90 年代に DOA（データ指向アプローチ）の洗礼を受けた世代で、それ以来、データベース設計を最重要課題とみなすシステム開発を実践してきた。最終的にたどりついた設計・開発手法が、「データ・ファースト」である。事業が扱うデータ全体の「あるべきデータモデル」を最初にまとめ、その形に沿ってプロセスモデル（業務フローや UI）をゼロベースでデザインする。それが「データ・ファースト」だ。

いっぽう、プロセスモデルを最初にまとめ、それに見合う DB 構造を導く従来のやり方（プロセス・ファースト）には多くの問題がある。現状のプロセスモデルを洗練させる程度では、漸進的な変化しか起こせない。それゆえに、扱われるデータの構造が合理化されないからだ。また、重複や矛盾が起こり得る非論理的なデータ構造に依拠するゆえに、プロセスモデルも合理化されない。システムの全面刷新や事業 DX を意図するプロジェクトでは明らかな失敗を招く。

■プロセス・ファーストで設計すると

プロセス・ファーストの典型的なパターンを説明しよう。その理不尽さは発注者の立場で想像することで理解しやすくなる。あなたはシステム開発の専門家ではなく、ある事業の経営者であるとしよう。起業してから 5 年たち、売上も順調に増えてきたので管理システムを合理化しようと考えている。今までは Excel で情報管理してきたが、データ項目が多いのでひどく手間取るようになってきたからだ。パッケージもいくつか調べたが、ユニークな事業なので適合するものがない。そこでシステム開発ベンダーに設計・開発を依頼した。

ベンダーの技術者がやってきてヒアリングが始まる。あなたは事業内容を説明する。そこで彼らが「では、As-Is（現状）の業務フローと UI（ユーザ・インタフェース。画面や帳票のこと）を説明してください」と訊いてくる。仕事の手順と利用している Excel の様

式を手間暇かけて説明すると、ベンダーの担当者はその内容を整理して緻密な資料にまとめてくる。それがシステム刷新後の To-Be（あるべき姿）だという（図１）。また彼らよると、現在の Excel ベースの管理システムは DB を核とするものに刷新されるらしいが、Excel の様式に合わせて進めるのだという。

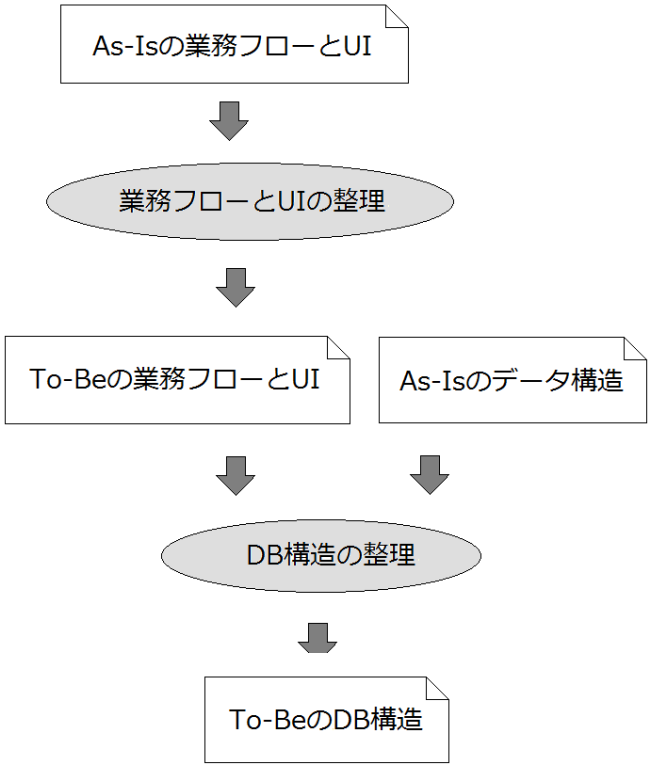


図１．プロセス・ファーストな開発過程

２カ月かけて作成・納品された膨大な設計資料を前に、事業が効率化できると確信できるだろうか。内容がわかりにくいだけでなく、あなたが説明したのは問題の多い現行仕様なのだ。そういうものを「整理」しただけのシステム仕様で本当に合理化されるのだろうか。しかもその結果はシステムが出来上がるまでわからない。この投資がバクチのようなものだと思えてこないだろうか。

■データ・ファーストで設計すると

では、ベンダーが次のように設計を進めたとしたらどうだろう。あなたは事業内容を説明する。すると彼らが「わかりました。では、その事業が扱うデータの本来の形をまとめましょう」と言ってくる。「『データの本来の形』といいますと？」「DB にデータを適切に格納するために、データ項目の論理的な関係を確認する必要があります。それがデータの本来の形で、データモデルともいいます」「データモデルですか。でも、事業が扱うデー

タの論理関係なんて訊かれても私は答えられません」「大丈夫です。さきほどの説明から、たたき台になりそうなデータモデルを想像できたので、ちょっと描き出しますね」

そんな調子でやりとりしながら、ホワイトボード上に「データモデル」が完成し、翌日に PC 上で整理した正式版を確認した。細かい図法についてはよくわからなかったが、事業に必要なさまざまな概念が組み込まれていることはわかった。有り難かったのは、豊富な開発経験にもとづいて、この事業に適合しそうなアイデアをいろいろと盛り込んでもらえたことだ。考えてもいなかった原価集計や粗利算定に必要なデータ項目さえ含まれていた。

しかし、多くの事柄が配慮されたとはいえ、この専門的な図面だけで確信を持てるわけではない。なによりも業務フローや UI が決まっていない。その不安を伝えると、彼らはこのように説明した。「もちろん、設計はこれで終わりではありません。来週までにこのモデルにもとづいてシステムのプロトタイプを作ってくるので、実際に使ってもらいます。それまで、商品データや取引先データの整備に協力していただけますか」「わかりました。業務フローはどうなりますかね」「プロトタイプを実際に使いながら、データモデルをブラッシュアップすることを優先させましょう。業務フローや業務マニュアルはその後で考えたらいいので（図２）」

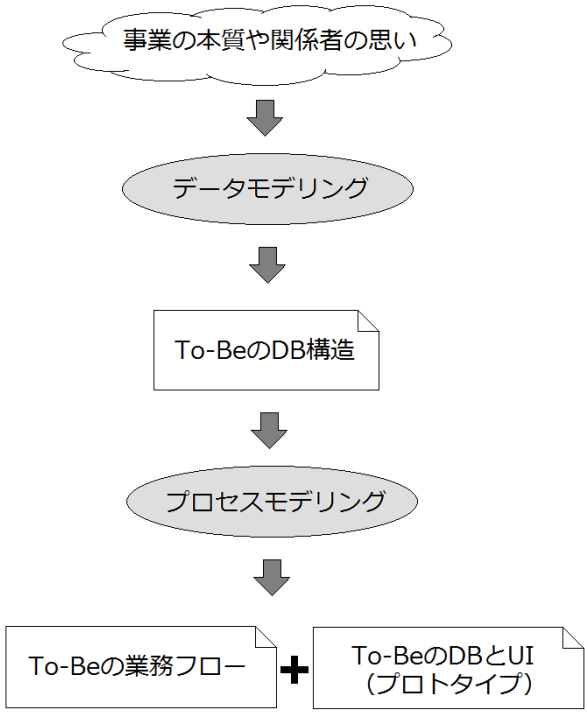


図２．データ・ファーストな開発過程

プロトタイプの効果は絶大であった。それを操作しながら実際にデータを登録することで、データモデルの不備もわかった。リアルなデータも登録されているので、利用時のイメージも明確になった。より効果的な UI も思いつけたし、業務フローや業務マニュアルも構想できた。この段階に至れば、あなたは新仕様の妥当性を理解し、確信を持って検収印を押せるだろう。あとはデータ移行と並行して、仕様にもとづいて本番システムを実装し、システムテストを重ねたらいい。

いかがだろう。どんなやり方でも発注側の不安がゼロになるわけではないが、やれるとしたら前者の「プロセス・ファースト」ではなく、後者の「データ・ファースト」のスタイルで進めたいとあなたは思わないだろうか。大きな利点はまず、データ構造が合理化されるゆえに業務フローや UI が抜本的に見直される点である。そして、プロトタイプによって仕様の正しさが検証されているので、実装の手戻りが少ないしデータ移行を早めに始められる点だ。

■データモデルとプロセスモデル

ここであらためてデータモデルとプロセスモデルの関係を見よう（図３）。事前に確立されたデータモデルがあるとして、異なる担当者がそれにもとづいてプロセスモデルを設計するとする。出来上がったものを比べると、細かい違いがあっても基本的には似たり寄ったりになる。それは関数従属性や定義域制約といった数学的枠組みを用いてデータ構造が形式化されているおかげなのだが、ようするにデータモデルから導かれるプロセスモデルは「収束」する。

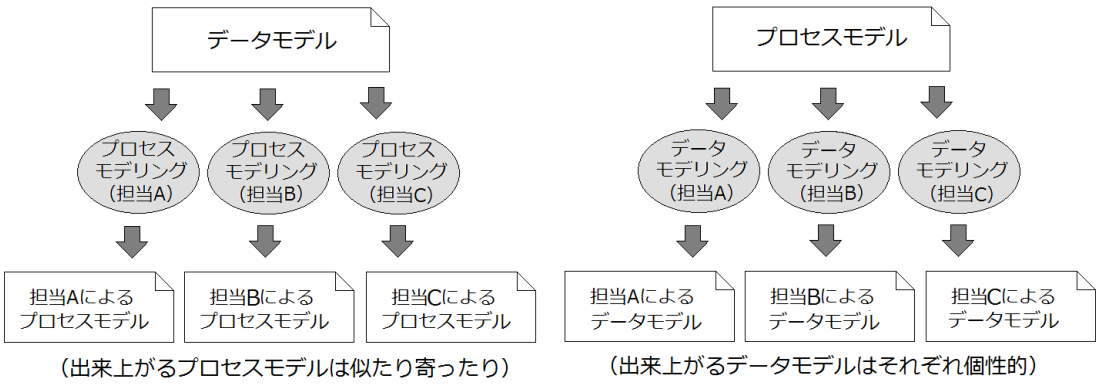


図３．データモデルとプロセスモデルの関係

反対に、事前に確立されたプロセスモデルだけにもとづいて複数の技術者がデータモデリングするとしよう。この場合、導かれるデータモデルは「発散」する。細かい部分から大枠まで、さまざまなバリエーションを含むデータモデルが出来上がる。その中に適切な

モデルが含まれる可能性はあるが、確率は低い。適切なデータモデルを導ける技術者はプロセス・ファーストではなく、データ・ファーストで設計を進めているはずだからだ。

２つのモデルのこういった関係は、システム開発を合理化するための鍵となる。データモデルから導かれるプロセスモデルに収束傾向があるということは、「プロセスモデルの導出過程をルール化できる」ということだ。この事実は重大である。

まず、事前に確立したデータモデルを与件として、UI をともなうアプリ群をルールにしたがって（半）自動生成できるようになる（こういった仕掛けはある種のローコード基盤として既に実現されている）。そして、これらをプロトタイプとして動作させながら、業務フローを考えたらいい。その過程で一部のアプリは作り替えられるし、場合によってはデータモデルも修正できる。そういった試行錯誤を経て、効果的なシステム仕様が短時間で確立される。筆者はこのやり方を 10 年以上実践しているので、その桁違いの生産性や設計品質の高さを日々実感している。

■データ・ファーストは「高度専門職」の証

このように説明すると、「業務フローもアプリもわからないままでデータモデルを書けるわけがない」と判で押したように反論される。それは経験からの思い込みでしかない。システム刷新において現行仕様は全面的に破棄される。新規事業向けであれば、参考になる現行仕様さえ存在していない。存在しているとしても、全面的に破棄されるべき As-Is を分析して To-Be を導き出せるとあなたは本当に思えるだろうか。

もちろん、細かいルールや制約など旧システムを参考にせざるを得ないケースはある。しかし、旧仕様を全面的に参照しないと新仕様を設計できないとしたら、システム設計のプロとして頼りなく思えないだろうか。顧客にとって困るのは、旧システムと似たり寄ったりの仕様が生み出されやすい点だ。家を改築する際に、もともと建っていたのが「竪穴式住居」だったので「建材だけが更新されたピカピカの竪穴式住居」が出来上がるようなものだ。顧客がそれを望むのであれば別だが、全面改築したいのであればシャレにならない。本来であれば、施主の思いと建築士の最新の知見が統合された「より合理的な形」になるように設計されるべきである。

にもかかわらず、現行仕様にこだわる進め方が一般的である理由は何か。プロセス・ファーストにおけるベンダーの役割が、現行システム仕様や（システム設計に関して素人であるはずの）ユーザが語る要望を様式に沿って清書する「代書屋」でしかないからだ。一定の手順や様式にしたがえば誰でもやれるゆえに、一定の事務能力さえあれば特別な適性は問われない。リクルートしやすいし育成も容易で賃金も抑えられる。ようするに、プロ

セス・ファーストな手法を旨とするシステム開発事業は「手軽」である。

それは事業の利点といってもいいが、経済社会に貢献しているとは言い難い。関係者に迷惑をかけることで得られる利点だからだ。膨大な工数を見積もって、スキルレベルの低い要員を大量に集めて長時間残業させ、高額のマージンで稼ぐ。大枚をはたいてデスマーチのあげくに手に入る成果物が「ピカピカの竪穴式住居」だとしたら、顧客や開発現場の犠牲によって成り立つ詐欺まがいの事業と批判されても仕方がない。

いっぽう、データ・ファーストの過程は高度かつ複雑で、担当者のスキルレベルによって成果物の品質が違ってくる。しかもその技量は一朝一夕に身につくものではない。知識だけでなく経験やセンスが求められる。しかしそれは欠点ではなく、建築士、あるいは映画音楽の作曲家といった創造性や感性が求められる高度専門職であるゆえの一般的な特性でしかない。職業適性によって選抜され、かつ必要な「訓練」を経てようやく現場に立てる。立つだけでは不十分で、より良い実績を生み出し、継続的な学習を続けなければ食べていけない。高度専門職であるとはそういうことだ。

これに関して、「誰がやっても同じ結果にならないとしたら、方法論として不十分ではないのか」と批判されて驚いたことがある。プロセス・ファーストでは誰がやっても似たような（低品質な）結果になるゆえの指摘らしい。そうであるならば、発注者が外部の専門家に頼る必要はない。緻密な手順書に従って発注者自身でやればいいし、暇がないなら学生バイトにでもやってもらえばいい。そのうち AI あたりを使った自動化さえ可能だろう。しかしシステム設計の仕事がそういうものではないことをわれわれは知っている。担当者の経験や習熟度によって結果が変わるのは高度専門職としては当然の話だ。だからこそ専門家同士の健全な競争も起こるし、業界のレベルも向上してゆく。

また、データ・ファーストについて「顧客が DB 構造の改善を要求していないのに、勝手に改変すべきではない」と批判されたこともある。これも無責任な言い草だ。DB 構造が不合理であるゆえにシステムにさまざまな問題が生じている。その事実はシステム開発のエキスパートしか気づけない。顧客はひたすら業務フローや UI の使いにくさを訴えるものだが、それは彼らがシステム設計に関して素人であるゆえに他ならない。「DB 構造の改善が要求されないのだからやらない」という姿勢は、老朽化したマンションの建替プロジェクトにおいて、内装だけをリフォームして済まそうとするようなものだ。その理由を問われて「管理組合が鉄筋や基礎の刷新を求めなかったから」と言い訳して通用するだろうか。

考えてみればそもそも、データ・ファーストは当たり前の設計方針である。プロセスモデル（業務フローや UI）は何のためにあるか。それは事業で扱うデータを適切に登録・維

持するために存在する。つまり、事業で扱うデータがどんなものであるかによって、プロセスモデルは決まる。プロセスモデルがデータモデルの前に決まっているのではない。しかも、業務システムで扱われるデータ構造は、他のソフトウエア分野に比較して複雑である。これらを考慮すれば、データ・ファーストが業務システムの標準的な設計方針であることは納得しやすい。見方を変えれば、システム設計という営為を高度専門職として成熟させるための鍵がデータ・ファーストである。

にもかかわらず、ほとんどの技術者がプロセス・ファーストでしかシステム設計できないのはなぜか。他でもない。プロセス・ファーストが常識化している技術組織に所属しているゆえに、データモデリングのリテラシーや訓練が十分に与えられていないからだ。そしてそれゆえに、データ・ファーストが実践可能であることを想像できないからだ。

馬には乗ってみよ、人には添うてみよ。IT 技術者はぜひ、データモデリングとこれに付随する多様なモデリングパターンを学び、データ・ファーストを試してみてほしい。それでも気に入らなければプロセス・ファーストに戻ればいい。しかし私は断言するが、いったんこれを経験するとシステム開発が俄然面白くなって、以前のやり方には戻れなくなる。システム開発にこんな有意義かつ創造的な側面があったとは――そのことに気づいたらもう後戻りできない。それがデータ・ファーストだ。