連載　企業および社会における情報システムの意味を考える
第 6 回　ビジネスの変化に迅速に対応できる情報システムにするための課題 その 5

大島　正善（MBC:Method Based Consulting）

今回も、前回に引き続いて、「ビジネスの変化に迅速に対応できない情報システム」ということについて、その問題の本質と解決策を探っていきます。これまで、ビジネスの変化に関係するビジネス活動の要素にはどのようなものがあって、その構造はどうなっているのかということを中心に書いてきました。今回はそういった複雑なビジネス・モデルの構造と情報システムにかかわる設計要素との対応関係をどのように理解したらよいのかということについて記述してみたいと思います。ポイントは、以下の点です。

① ビジネスの機能（業務機能）は、タスクの作業の抽象概念である

② ソフトウェアの機能はビジネス機能をさらに抽象した概念である

③ ソフトウェアの構成要素もn次元の構造である

④ ビジネスの構造と情報システムの構造の関連と明確にして開発をコントロールすることが求められる

1. ビジネスの機能（業務機能）は、タスクの作業の抽象概念である

メールマガジン2013年1月1日号（No.07-10）の記事で、ビジネス・プロセスの構造について触れました。ビジネス・プロセスを、プロセス→アクティビティ→タスクという3階層で捉える考え方も示しました。PDCAを意識してプロセス（あるいはアクティビティ）間の関係を明確にし、プロセスをまたがったタスクレベルの関係が業務フローで表現されるという点についても触れました。

1月1日号の記事の中では深く触れなかったのですが、業務フローで示されるタスクの仕事から役割としての業務機能を識別します。当たり前のように実施している作業です。しかし、この当たり前の作業が複数の役割から共通的な内容を抽出するという“ 抽象化の作業 ”であるということを自覚している人はどのくらいいるでしょうか?この抽象化は目立たないかもしれませんが、自覚するかどうかで、業務機能というものの捉え方に違いがでてくることを知っておいて損はないでしょう。

そのことを考えるために、ひとつの例を示します。皆さんが旅行会社でホテルの申し込みをすることを考えてください。旅行の申し込みはいろいろな方法でできます。直営店に行って申し込む、代理店で申し込む、Webサイトから申し込む、あるいはヘルプデスクに

電話して申し込むこともできます。旅行業者のそれぞれの組織で行っている業務は、すべて「申し込み受付」という機能と思われます。しかしながら、組織ごとに行っている業務機能は、少しずつ違いがあります。ヘルプデスクで行っている「申し込み受付」機能の中には、お客さまから希望日を聞きながらホテルの空き状況を確認し、空きがなければ、別のホテルを紹介するという行為も含んでいます。直営店での申し込みも同じように別のホテルの紹介などをしますが、そこでは、ホテルだけでなく電車や航空機のチケットの手配と販売も行うかもしれません。代理店では、結果を旅行代理店に報告するということも必要です。Webサイトでは、別のホテルを検索したりエアチケットを手配したりするのは、あくまで顧客が主体的に行う必要があります。

業務の仕事として行われている作業には、こういった少しずつ違いがありますが、我々は普通そういった違いは無視して直感的にひとつの“ 申込受付 ”という業務機能を識別しています。つまり抽象化が行われています。それは、どの企業でも行われている販売業務の受注、発注、出荷、あるいは在庫照会でも同じです。

ここで注意しなければならないのは、抽象化された業務機能があたかも企業のある特定の組織やプロセスでのみ実行されているととらえている例が見られることです。別の表現をするのであれば、ビジネス・プロセス(をブレークダウンした最下位のタスク)と業務機能との関係を1:1ととらえて整理していることが多いのではないでしょうか?正確には、その関係はM:1(多対1)あるいはN:M(多対多)の関係がありますが、そういった管理を行っている企業はどのくらいあるでしょうか?業務機能をビジネス・プロセスとの関係で識別し、その関係をマトリックスで管理していないと、ビジネス・プロセスの変化がどういった業務機能に影響を及ぼすのか、直観にたよって分析せざるを得なくなります。それが多くの企業の現実といえるでしょう。

2. ソフトウェアの機能はビジネス機能をさらに抽象した概念である

何らかの理由でITシステム化の計画が立案されシステム開発が始まると、必ずある段階で機能一覧表というものを作ります。機能一覧表では、機能が階層的に表現され一番上位に「業務機能」が示された形式で整理されることがよくあります。たとえば図1に示すような形式です。

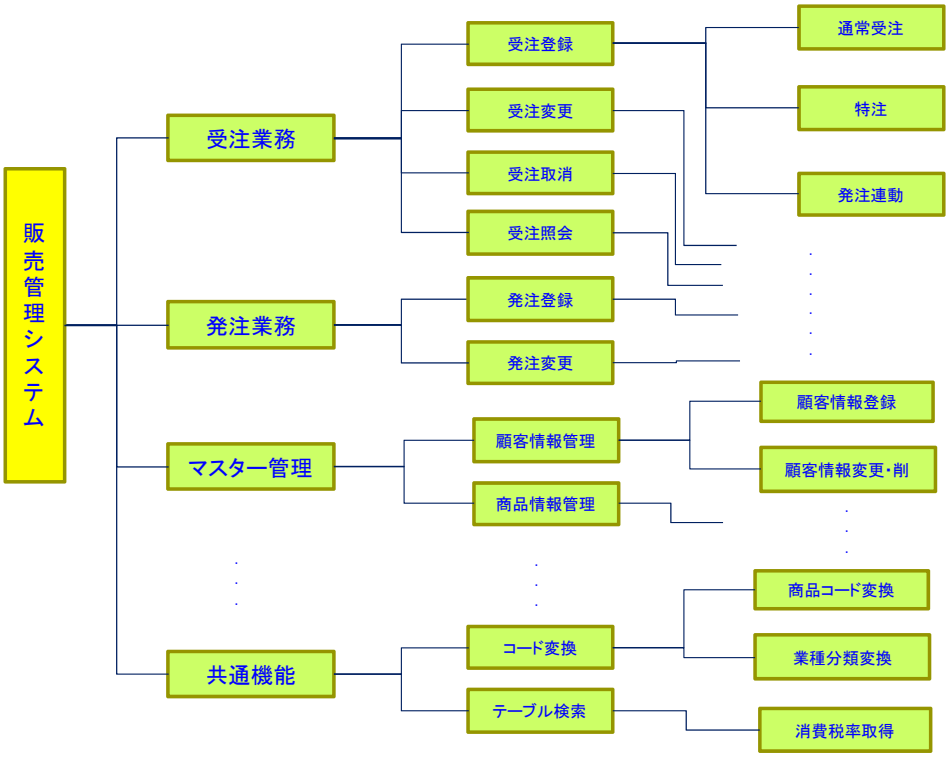

図1　機能一覧表の例

よく見られるこういった機能一覧表では、ソフトウェアで実現する機能が業務機能を最上位として構造的に展開された形式で表現されます。また、業務共通の機能として識別された「マスターデータを管理する業務機能」が定義され、その下位にマスター情報ごとの管理機能を展開しています。さらに、“ 業務共通の機能 ” が識別され、その種類が下位の階層に示されています。さて、何か問題があるでしょうか?ソフトウェアの機能はどのようにして識別されているのでしょう?

ソフトウェアの機能をどのように識別しているのか?という問いの意味は分かりにくいかもしれません。あまりに当然すぎて、考えさえしていないのが普通だと思います。

最近は、多くの業務システムが存在しており、一からソフトウェアを開発することが少なくなったこともあり、ビジネス・プロセス図や業務フロー図を作成するのでなく、既存のプログラムから識別したり、入出力の画面や帳票名称に「機能」を付与しているだけのこともあるようです。あるいは、業務の名称に「機能」を付与している例もあります。そのような場合は、業務機能のうちソフトウェアで実現できる部分を機能と考えているといってよいでしょう。本当にそれだけでしょうか?

ソフトウェアの機能としての顧客情報照会機能を考えてみましょう。顧客情報照会機能といっても一種類ではなく多くの照会パターンがあります。住所や電話番号といった基本情報の照会から家族情報や、商品の購買状況などの照会機能もあるでしょう。それらの複数の顧客情報照会をソフトウェアの機能として設計する方法はいろいろ考えられますが、照会の条件を汎用的に扱えるようにすれば、ソフトウェアでは一つの機能として設計・実装することはよくあることです。この場合、複数の業務機能を一つのソフトウェアの機能として設計しているのであり、そこには抽象化が行われています。

次に業務の受注機能と発注機能について考えてみましょう。業務機能は全く別のものです。普通はソフトウェアの機能としても受注データ登録と発注データ登録は別の機能として設計することが多いでしょう。しかし、受注も発注もデータ構造としては受注（あるいは発注）単位の情報と明細単位の情報を持つという意味で似ており、クラスの定義の仕方によっては、同一クラスの別インスタンスとして設計することが可能です。この場合も、業務機能とシステム機能は1：1に対応しているわけではありません。

このように、ソフトウェアの機能は業務機能にそのまま対応したものとして設計するわけではなく、“ 抽象化された共通機能 ”として定義します。業務機能とソフトウェアの機能は抽象度が違っており、結局のところ業務機能とソフトウェアの機能も、今まで見てきたほかの例と同じくマトリックスの関係にある、ということになります。

なお、この図に示されている“ マスター管理 ”や“ 共通機能 ”というのは業務機能でしょうか?実際にはシステム機能といったほうが適切だと思います。こういう一覧表は業務機能からシステム機能が整然と構造的に展開されて一見問題ないように見えますが、業務機能とソフトウェアの機能を同一レベルに並べており正確な構造を示しているわけではありません。

3. ソフトウェアの構成要素もn次元の構造である

あらためて詳しく述べる必要もないでしょう。ソフトウェアで実現される機能を分割して下位に展開した要素と上位の要素はマトリックスの関係になる可能性があります。下位の要素となるソフトウェアは通常複数の親のソフトウェアから呼び出されます（親が一つだけであることが明確であれば、分割する必要がないのですから）。共通機能というのは本来的に複数の機能から呼ばれるもので、ひとつの機能だけの下位の要素に位置づけられるものではありません。UIの部品もそうですしデータアクセス部品も同じです。開発者はそのようなことは当然と思って（と言うか、そういうことは考えもせずに）開発をしています。

オブジェクト指向言語のクラスは、そもそも汎用機能なのでクラス群の集合であるソフトウェア全体の機能が二次元の上下階層で表現できるわけではありません。

しかし、機能一覧表にすると我々はどういうわけか単純に上下関係の機能として表現してしまいます。そして、その2次元の構造を前提にサブシステムを定義し開発分担を決めます。

そこにシステム開発の大きな問題が潜んでいるように感じるのは私だけでしょうか。ソフトウェアの機能間の関係がマトリックスの構造にあるということは、単純に上下関係の構造である場合とでは、機能のくくり方（サブシステム、コンポーネントやサービスの単位の決め方）、開発体制、作業の順序・組み立て、工程定義、進捗管理など、あらゆる面で考慮すべきことが違ってくるはずです。

ひとつのモジュールの変更が及ぶ範囲をきちんと管理（見える化）せずに設計や開発を行っているのが多くのプロジェクトの実態であり、このことがプロジェクトのコントロールを困難にしている大きな理由ではないかと思います。ソフトウェアの機能（モジュール、クラス）全体の構造もn次元になっていることが実態であり、そのn次元の関係を把握しながら開発を進めることが本来必要ではないでしょうか。このことを考えると、サブシステムやコンポーネント、あるいはサービスという塊の持つ意味も変わってくることがわかるでしょう。その点については、議論すべき点が多々あり今後執筆していきたいと考えています。機能をどう括るのがよいのか?あるいは、どのような視点から分割すべきなのか?ソフトウェアの設計では古くからのテーマですが、ソフトウェアの構成要素が複雑なn次元構造をしているということを認識したときに、解決すべきことが多く横たわっていることをあらためて理解できるでしょう。

4. ビジネスの構造と情報システムの構造の関連と明確にして開発をコントロールすることが求められる

10月からの一連の記事で述べてきたようなやり方、つまり「ビジネス・モデルの構造をメタモデルで管理し、情報システム・モデルの構造とのトレーサビリティを管理するという方法」を採用している企業はおそらく皆無ではないでしょうか?経営の変化に迅速に対応できるためには、「ビジネス・モデルと情報システムのモデルのトレーサビリティを管理すること」は必要条件だと考えます。これができていないので、「経営環境の変化に迅速に対応できない情報システム」となるのは当然です。ですから、この課題は10年以上たっても解決できないのではないかと思っています。

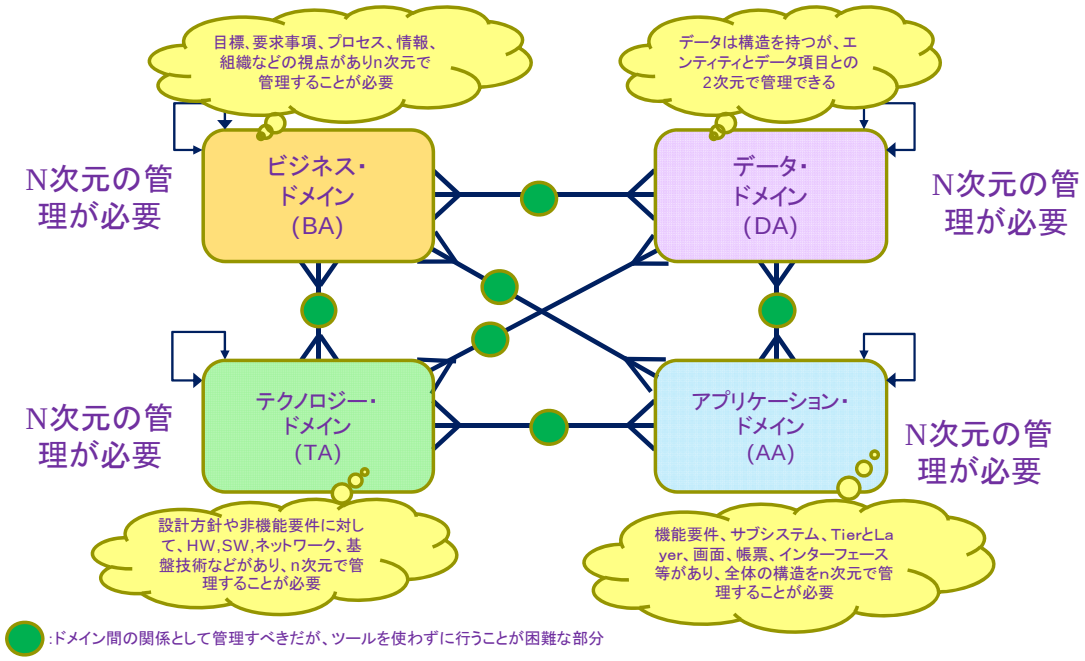

図2　EA体系に基づく成果物間の関連

図2は、EA体系の成果物間の関係の全体像を示しています。詳しい説明は省略しますが、アーキテクチャ・ドメインごとに多くの成果物があり、ひとつの成果物だけでもn次元構造（たとえば、ソフトウェアの機能構造）です。そして、関連を持つほとんどすべての成果物間に多対多の関係があります。今は、そういった複雑な構造をもった設計成果物をOfficeソフトで文書化し、それをもとに人がプログラミングをしているのが多くのプロジェクトの実態です。

設計・開発ツールの利用はまだ広まっているとは言えません。システム開発プロジェクトの多くは、“ 管理できている ” とは言えない状況です。図2に示すように、システム開発にかかわる要素は非常に多く、それらの要素間の関係を最初からすべて構築しようとするのはリスクが高すぎます。

開発された情報システムが、経営の変化に迅速に対応できるようにするためには、EA(Enterprise Architecture)体系の4つのドメインの関係のうち、少なくともBAドメインの業務機能とAAドメインのアプリケーション機能との対応関係は正しく管理することが必要です。それにより、現在よりもビジネスの変化の影響を迅速に把握できるようになるでしょう。情報システムの開発にかかわる方々は、ビジネスの構造と情報システムの構造が複雑なn次元のマトリックス構造であるとの認識を持つことが非常に重要であると感じています。

ビジネスの変化に迅速に対応できるようにするためには、トレーサビリティの管理は必要条件ですが十分条件ではありません。トレーサビリティが管理できるので影響を受けるシステム要素の特定は容易になりますが、要件の分析、設計、開発、テスト、文書化、マニュアル改訂などの作業は必要です。これらの工数と期間を短縮する現時点で可能な方法は、設計情報をもとにコードを自動生成する方法でしょう。テストもある程度は自動化できます。

企業の情報システムは、ますます業務そのものに深く入り込んでいきます。将来は、BPMS，BRMSが業務に組み込まれ、それらのツールに登録されているビジネス・プロセス、ビジネス・ルール、組織、戦略、目標、方針などを変更することで、プログラムが自動生成される世の中になると確信しています。今まで述べてきたようなビジネスとシステムのすべての要素を関連付けたメタモデルを管理する企業は、競争優位に立てることは間違いありません。ですから、グローバルな環境でビジネスを展開する企業は、このような管理の仕組みの構築に早速取り掛かるべきと考えます。

以上