ソフトウェア工学 100－3
（1994．9．8）

オーダメイド設計・プレハブ施工型システム構築ツール
－SCAWデザインシリーズTM－

板倉征男・鈴木隆司・鈴木努

NTTデータ通信(株)

従来のシステム構築は主にオーダーメイド型とパッケージソフト型で行われていた。しかし、ユーザの抱えている経営課題をシステム企画に反映し、そこで抽出した解決策をシステムに取り込むという点で両者とも非常に課題が多い。また、オープンシステム化がさけばれるなか、オープンシステムベースでのシステム構築に関しては、十分なノウハウもソフトウェア資産もないのが現状である。

これらの問題点を解決する新しいシステム構築技法としてSCAWデザインシリーズを開発した。SCAWデザインシリーズは、トップダウン型のアプローチを取ることにより、経営課題の解決方法としてのベクトルに沿ったミドルマネージメント、ロワーマネージメントの要件分析を行う。

Customer Oriented Design And Object Oriented Development Method For Business Systems
-SCAW Design Series TM-

Yukio Itakura　Takashi Suzuki　Tsutomu Suzuki

NTT DATA Communications Systems Corp.

Toyosu Center Bldg.,3-3,Toyosu 3-Chome, Koto-ku, Tokyo 135 Japan

Information systems today are not merely a tool for more efficient business, but a tool for solving management problems to attain superiority. What becomes extremely important in constructing systems in such a situation is how to reflect management problems of users in the upper processes of planning information system and how to take in this solution that has been extracted from the upper processes.

We have developed SCAW Design Series for a new system construction method to meet these requirements.

1.SCAWデザインシリーズの概要

SCAWデザインシリーズは、トップマネージメントを巻き込んだトップダウン型のアプローチをとることにより従来の業務の効率化といった方法論ではなく、経営課題の解決方法としてのベクトルに沿ったミドルマネジメント、ロワーマネジメントの要件分析を行う。トップマネジメントのシステムの導入目的が明確に打ち出されているため手戻りが少なくなると同時に、各階層のユーザの要件に沿ったシステムを提案することができる。また、提案内容は業務機能の核をソフトウェアモジュールとして用意した部品をもとにプレハブ施工型で構築する。

SCAWデザインシリーズではユーザの要件分析を行う上で、業種等によって分類されたツールが用意されており、品質の高いシステムを構築できる。また、ユーザの要件に合った機能別のカーネル群を組み合わせることによって短期間にシステムを構築することが可能である。

本稿では、SCAWデザインシリーズのユーザの業務プロセス把握ツールであるSEP(Strategic system Evaluation and Planning)、ユーザ要件分析システム設計支援ツールであるSFB(Structured Frame method for Business system design)とシステムのモジュール群であるSIK(System Integration Kernel)の概要について説明する。

1.1 SCAW-SEP

ユーザ要件に合った基幹情報システムが構築できない最大の原因は、ユーザの経営課題を十分に把握できないこと、ユーザ側もそれを明瞭に提示することができないことである。SCAWデザインシリーズにおいて、トップダウンアプローチによりユーザの経営課題の明確化、システム化対象業務プロセスの明確化をするのがSCAW-SEPである。

1.2 SCAW-SFB

システム化対象の業務プロセスをシステム機能に展開するのがSCAW-SFBである。SCAW-SFBでは、基幹システムの核となる部品・SCAW-SIKのスペックとユーザニーズとを比較しながらSCAW-SIKの適用範囲・非適用範囲と、カスタマイズ内容の青写真を作成する。

1.3 SCAW-SIK

SCAWデザインシリーズでは基幹業務システムのモジュール群SCAW-SIKを用意している。SCAW-SIKは、基幹業務の中でも業種特性にあまり影響されないコア部分を抽象化している。これらをプレハブの部品として利用し、SCAW-SFBによって明確化されたカスタマイズ内容を施工し組み立てることで、ユーザニーズに合った情報システムを構築する。

2.SCAW-SEP：経営課題と業務プロセスの把握

2.1 経営課題の把握

2.1.1 業務プロセス・フレーム

SCAW-SEPでは、情報システムがその企業にとって理想的なシステムになるよう方向付けを行うために、まずトップヒアリング・ミドルヒアリングにより企業の重要経営課題領域の明確化、問題の発見と原因の把握、課題構成の設定と報告を行う。そして、この一連の作業の結果を、クロス管理のフレームワークとして用意している「業務プロセス・フレーム」にポジショニングし、整理・分類する。（図2.1）

Pi	Pi	マネジメントプロセス PLAN	DO	CHECK	ACTION
指揮命令プロセス	方針	P11	P12	P13	P14
	計画	P21	P22	P23	P24
	管理	P31	P32	P33	P34
	運用	P41	P42	P43	P44

図2.1 業務プロセス・フレーム

「業務プロセス・フレーム」は各部門の業務機能の概念をマネジメント・プロセス（横軸）、指揮命令プロセス（縦軸）により16分割した表で、個々のボックスP(ij)を「業務」と呼んでいる。縦の指揮命令プロセスは上から方針－計画－管理－運用の階層からなり、マネジメントプロセスはPlan－Do－Check－Actionのプロセスからなる。

「業務プロセス・フレーム」では、運用（Operation）中心の従来型の要件把握アプローチでは死角となり十分に捉えられなかった管理（Control）・計画（Plan）・方針（Strategy）レベルの課題・問題を明示する。

2.1.2 部門内における業務プロセスの関係

個々の業務を命令系・報告系・業務の流れという3つの視点から捉えて整理すると以下のようになる。（文書中の（P23）等の表示は一例として図2.2とあわせている）

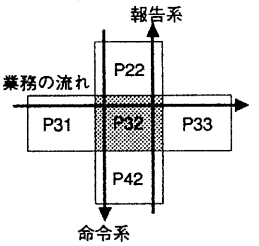


図2.2 部門内業務プロセス

1.命令系の流れ（縦：上→下の業務プロセス）

業務（P22）はある命令をイベントとして発生し、その1レベル下の業務（P32）を起動するための命令を発生させる。命令自体は人間により発せられ、個々の業務は必ずその前にある業務の結果をうけて発生した命令に基づき実施される。つまり、命令系において業務は上→下の業務プロセスを形成する。

2.報告系の流れ（縦：下→上の業務プロセス）

報告系のでは、命令のもとに発生した業務（P42）の結果（情報）は、命令発行元の業務（P32）の管理者たる人間に返される。つまり、業務は情報系においては下→上の業務プロセスを形成する。

3.業務の流れ（横：左→右の業務プロセス）

業務はそれ自体が完了次第、次の業務に流れ（工程上タイムブランクを発生することも多々あるが）、それらが一連となってPDCAサイクル、つまり、左→右の横の業務プロセスを形成する。

2.1.3 部門間における業務プロセスの関係

企業において、各部門は独立していながらも他の部門との連携の元に機能し、部門としての目的を達成する。つまり、各部門のみを意識しただけでは、情報システムも部門最適化システムとなってしまい、組織の壁を取り払って見ない限り、全社的最適化を実現することはできない。

「業務プロセス・フレーム」では、営業・経営・生産等の部門間にまたがる業務プロセスは、ユーザと第一に接点を持つ営業部門をスタートとして、各部門同一レベルの業務を串刺しする形で捉えられる。（図2.3）

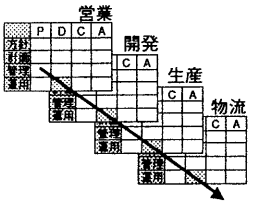


図2.3部門間業務プロセス

2.2 作業プロセスの把握

2.2.1 作業プロセス・フレーム

SCAW-SEPでは「業務プロセス・フレーム」で機能分割された業務を起-承-転-結からなる実行プロセス(縦軸)と作業の管理サイクルであるCAP-Do(Check-Action-Plan＋Do)からなる作業プロセス(横軸)によりさらに16分割する。これを「作業プロセス・フレーム」と呼ぶ。(図2.4)

Pk＼Pt	作業プロセス			
	CAP	DO1	DO2	DO3
実行プロセス 起	P4211	P4212	P4213	P4214
実行プロセス 承	P4221	P4222	P4223	P4224
実行プロセス 転	P4231	P4232	P4233	P4234
実行プロセス 結	P4241	P4242	P4243	P4244

図2.4 P42の作業プロセスフレーム

実行プロセスの起は市場又は商品にかかわる作業、承は起に対応する設備・体制にかかわる作業、転は必要な資金にかかわる作業、結は利益又はコストにかかわる作業をあらわす。

作業プロセスは、PDCAサイクルよりCAP-Doサイクルの方が妥当である。なぜなら日常管理において、PDCAサイクルを端からまわすことは難しく、反省なくして計画を立案すると、計画が思い付き・願望・スローガンになってしまうことが多く、当初の計画を達成することなど無理になるからである。つまり、作業自体は計画から行うのではなく、反省からはじめることが望ましい。

また作業プロセスに関しては、通常、実務作業自体が既にルーチンワーク化されており、Chack-Action-Planにかかる作業は個々に機能することよりもCAPを一体化して機能することの方が多い。そのため、CAPを一体化した作業機能とし、Doを3つの作業機能として表示している。基本的にDo1は人に係る作業機能、Do2は物に係る作業機能、Do3は金に係る作業機能とする。

2.2.2 作業プロセスとシステム機能の関係

各部門の業務機能は、この作業プロセス・フレームに展開された時点で$4^2 \times 4^2 = 256$に分割されている。SCAWデザインシリーズでは、この作業プロセスをシステム化対象業務機能の最少単位として位置付けている。個々の作業プロセスは複数のシステム機能から成り立ち、以降のSCAW-SFBの工程においてその作業プロセス内で必要とおもわれる具体的なシステム機能と結び付られる。

3 SCAW-SFB：業務プロセスと情報システムの位置付け

3.1 システム機能の抽出～設計

3.1.1 DFDによるシステム機能抽出

SCAW-SFBは通常の基本・詳細設計工程に該当し、作業プロセスまで展開された業務機能をもとに、効率良く、業務機能にマッチしたシステム機能を抽出できるよう構成されている。

SCAW-SFBでは、まずSCAW-SEPにより展開された作業プロセス毎にDFD（Data Flow Diagram）を用意しており、このDFDによって各作業プロセスの機能（Function）および、情報の流れのひな型をユーザに明示し、SCAW-SIKのシステム機能の選択を行う。SEはDFDの中からユーザ業務にあった機能とそれに伴うフローを提示しながら選択することで、ユーザとの同意の元にシステム機能を抽出することになる。

また、DFDでは2.1.3で述べた部門間の関係、つまり外部システムとのインタフェースをターミネータというかたちで表現しており、個々のシステム機能に必要な外部情報が把握できる。そして、個々の作業プロセスは、そのプロセスで必要となる複数のシステム機能を含んでいるため、DFDは作業プロセス内の各業務機能がシステム機能と1:1の関係になるよう作成されている。

3.1.2 モジュール標準機能表

次にDFDによって抽出されたシステム機能は、モジュール標準機能表に引き継がれる。モジュール標準機能表は、システム機能の標準的な仕様と、その機能が必要とする情報アイテム（要素・資源をあらわす）や各情報アイテムが属すエンティティなどで構成され、DFDの1機能に対し同じ機能でありながら扱う項目やエンティティの違うN種類（例えば業種・業態別）のモジュール標準機能表が用意されている。（図3.1）

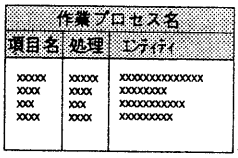


図3.1 モジュール標準機能表

モジュール標準機能表は各システム機能の標準的な仕様を表現しているため、営業SEは効率的にシステム機能の仕様確定が行える。モジュール標準機能表は、必要とおもわれる情報アイテムや、処理が表現されていない場合、そこに機能追加し、ユーザごとの追加事例を蓄積し整理することで、業種・業態別や、管理方式別のデータベースが作成されるようになっている。

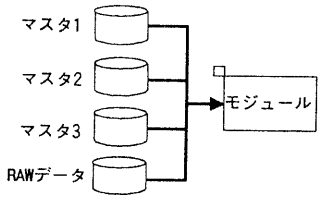
4. SCAW-SIK：処理機能のモジュール化

4.1 RDB 設計

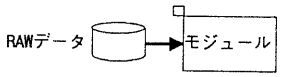
4.1.1 データの正規化・アクセスパスの最適化

データの正規化を行っていない場合、あるエンティティに関する情報が複数のテーブルにまたがって存在する可能性がある。データベース上の項目が重複して存在すると、1つの処理にたいして複数のエンティティ、フィールドに影響が及ぶので処理が複雑になり、メンテナンスビリティの低下に繋がる。（図4.1）そこでデータベースの正規化を行うことにより、項目の重複を排除し整合性を保ち、データの追加・登録・削除による影響範囲を限定する必要がある。正規化をすることにより1つの処理に対しての影響範囲が限定されるため、それに関わる処理自体も簡素化され、モジュールのメンテナンスビリティの向上に繋がる。

しかし、システム全体のパフォーマンスという観点から考えると、一概に正規化を行えばいいということではない。正規化されたデータベースは、データの更新処理には整合性を保つという意味で適しているが、参照処理を行う際はいくつものエンティティを参照するようになるため、パフォーマンスの低下を招く。



処理の最適化を行ってない場合



処理の最適化を行う場合

図4.1処理の最適化

SCAWデザインシリーズでは、発生した生のデータ(RAWデータ)をそのままタンキングしたデータベースを準備し、RAWデータに立ち戻ったデータの再利用が可能になるように図っているためRAWデータベースには頻繁にアクセスされる。その頻繁にアクセスされるRAWデータベースをきちんと正規化しておくと照会のたびに、いくつものエンティティをアクセスすることになりパフォーマンスの低下を招く。そこでRAWデータベースはデータの重複にはなるが、あえて正規化を崩し、ある事象において発生したRAWデータを名称等も含めた形で保存しておく。そうすることにより1つの処理を、RAWデータベースのみにアクセスすることで完結できるようになるので処理の最適化が図られ、パフォーマンスの向上が図られる。

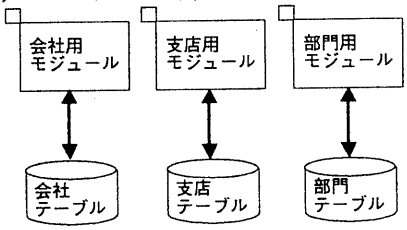
4.1.2 データの抽象化

SCAW-SIKはプレハブ施工型システム構築技法の部品として扱われるので、様々な顧客要求に耐え得るように、モジュールの汎用性を高める必要がある。そのためには、データを抽象化して捉えることが重要である。データを抽象化してモジュールの汎用性を高める例として会社の組織に関するケースを挙げる。（図4.2）

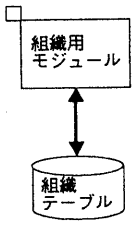
データを抽象化して捉えなかった場合、会社の組織に関するエンティティは会社テーブル、支店

テーブル、部門テーブル･･･のように組織階層毎にエンティティを定義することになり、それにかかわるモジュールもエンティティ毎となるので複数必要になる。また顧客要求で組織階層を１つ増やしたいというような場合は、エンティティを新たに１つ追加定義し、それにかかわるモジュールも新規作成しなければならないので、部品としてのモジュールの汎用性は非常に低い。しかしデータを抽象化して捉えて、会社の組織に関するエンティティを1つにまとめて定義してあるような場合は、モジュールも組織階層を判断する仕組みさえ作っておけば、組織階層毎には必要なくなる。

このようにデータの抽象度を上げておくと、モジュールの汎用性を高めることも可能になり、メンテナンスビリティの向上にも繋がる。



データを抽象化していない場合



データを抽象化している場合

図4.2 抽象化

4.2 カプセル化

SCAW-SIKではメンテナンスビリティを高めるため、RDBの3層スキーマ構造の概念とそれをサポートする4GLの機能を利用し開発している。3層スキーマ構造の基本的考え方は、データベースの構造を外部スキーマ・概念スキーマ・内部スキーマの3階層で捉えたものである。外部スキーマとはアプリケーションプログラムに対応した部分だけのデータ構造である。概念スキーマはデータ定義を概念的に捉える場合、そのデータの組み立てられかたや、各データに対する制約条件を定義したものである。内部スキーマは概念スキーマを実装するのに必要な実際のDBMSコードで構成されている。（図4.3）

SCAW-SIKは、この3層スキーマ構造の概念を利用し、各外部スキーマに共通して必要になるデータの一般的なチェックロジックなどを個々の外部スキーマ上に記述するのではなく、概念スキーマに記述するようにしている。概念スキーマ上に処理を記述しておくと、データの属性が変更されそれに伴いデータのチェックロジックなどを変更しなければならない場合も、概念スキーマ上の処理を変更するだけで済むのでメンテナンスビリティの向上に繋がる。また同じ処理が2度・3度と別々の外部スキーマに記述されることがないので、全体のコーディング量も少なくなる。

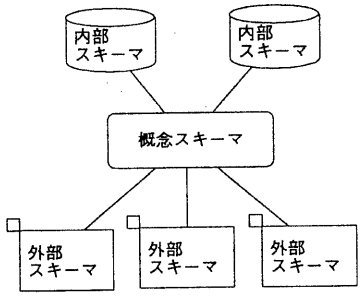


図4.3 3層スキーマ構造

4.3 エンドユーザコンピューティングへの対応

4.3.1 EUCの必要性

ユーザの要件は、業務やマネジメントレベルなどによって全く異なっており、これらを一つのシステムで満足させることは不可能である。またユーザ要件は時間軸に対して動的であり、これは「ユーザを取りまく環境が急激に変化している」ということと「システムを導入することによって新たにユーザの要件がでてくる」ということの2

点に起因している。すなわち、ある時点でシステムの機能を固定化することはできない。

あらゆるユーザの要件を満足するシステムを構築するためには、従来のようにユーザの要件を最大公約数的に満たすように設計された固定的なシステムではなく、ユーザ自身が自分の要件に合うように随時システムを改善できるようなダイナミックなシステムでなければならない。これを実現するためにEUCを考慮したシステムを構築する必要がある。

EUCが可能になってきた条件として、クライアントマシンのパフォーマンスの向上、サーバー上のRDBMSとの接続可能な表計算ソフトやクライアント上で動くデータベースの登場、Windowsの登場によるGUI環境の充実、などが挙げられる。このように一般のエンドユーザが利用可能な開発環境が整ってきたことによって、ユーザインターフェース部分などの簡単なものであればノンプログラミングで十分に実用に耐えうるものを開発できるようになってきた。

4.3.2 EUCを前提にしたデータベース設計

EUCを考える際に最も重要になってくるのはデータベースの構成である。EUCに適したデータベース構造の条件として、次の3点が挙げられる。

①構造がわかりやすいこと。

②参照するテーブル数が少ないこと。

③ユーザの必要な情報がすべて入っていること。

従来のようにデータベースの正規化のみを考えて設計した場合、リレーションが複雑になり、エンドユーザにとって必要な情報をすぐに参照することができない。4.1.1で述べたように、正規化されたデータベース構造はデータを更新する処理には向いているが、データを参照する処理には向いていない。エンドユーザが必要としている機能のほとんどがデータを参照する処理であることを考えると、エンドユーザの必要なデータ、特にRAWデータに関しては正規化は行わないほうがよい。EUCを行う場合にも、エンドユーザの必要とする情報ができるだけ少ないテーブルから得られるほうが、処理は簡単になる。ビューを用いて参照する表を少なくする方法もあるが、アクセスパスは変わらずかえってパフォーマンスの悪化を招いてしまう。

4.3.3 RAWデータとEUC

エンドユーザの必要とする機能は、マネジメントレベルにかかわらずデータを参照する処理がほとんどである。エンドユーザが蓄積されたデータを自由に加工し、必要なときに必要な形で手に入れることができるためには、すべてのユーザが必要としているすべて情報がRAWデータに投影されていなければならない。また、RAWデータはオンラインで登録され、即時にシステムへ反映される。従来のようにバッチ処理でデータを加工するのではなく、RAWデータをクライアント側で加工するので、エンドユーザは常に最新の情報を手に入れることができる。

SCAWデザインシリーズでは情報を人、物、金に束ね、エンティティの抽象度を高め、できるだけエンティティ数を少なくする事によって次のような効果を得た。

①テーブル構造がわかりやすくエンドユーザが利用しやすい。

②アクセスパスを少なくし参照時のパフォーマンスをに向上させる。

③汎用性を高め、機能拡張を容易にする。

しかしエンティティの数を減らすことによって、一つのテーブルへの問い合わせが多くなると極端に処理が遅くなる。このためサーバーマシンの処理速度とユーザーの利用頻度を検討し、システム設計を行わなければならない。SCAWデザインシリーズでは、テーブルに対するアクセス頻度と処理速度を測定し、パフォーマンスの限界値内に押さえるように設計されている。

4.4 EUCを可能にした要因

コンピュータのパフォーマンス向上に伴って、従来ではクライアント側ではできなかったような複雑な処理が簡単にできるようになってきている。EUCを可能にしたツール群の説明とそれをどのようにシステムに組み込むかを説明する。

1)クライアント上で稼働するRDBMSと表計算ソフト

クライアント上で利用可能なこれらのツールは、EUCを考える上でフロントエンドを強力にバックアップする。クライアントから直接サーバーのRDBMSを参照することができ、しかも開発ツールの充実によって、エンドユーザはほとんどプログラミングの必要がない。また、ローカルにあるデータベースとのリンクも可能であるので、分散データベース環境でのシステム構築も可能になる。

2)4GL

データーオリエンティッドでかつイベントトリブンな構造を持つ4GLは、従来の3GLに比べて数倍から数十倍の効率で開発することが可能で、しかもメンテナンスビリティが高い。ただし高い生産性を維持するには、データベース構造が整然とわかりやすくなければならない。

3)RDBMS

以前はパフォーマンスの面で問題があり、基幹系のシステムで使用されることが少なかったが、コンピュータの処理能力の向上に伴って使用されることが多くなってきた。特に最近はストアド・プロシジャーなどクライアントサーバーシステムを考慮にいれた機能を持っているものもある。この機能によって、クライアント側では処理しきれないような複雑な処理に対しては、ストアド・プロシジャーを利用してサーバ側に依頼し、単純な処理はクライアント側で行うといったシステム構成を取ることができる。これによってネットワーク負荷とメンテナンス負荷を軽減する。

5 終わりに

SCAWデザインシリーズは従来の開発工法に比べ費用、工期等について以下のような効果が得られた。

①コーディング量の削減

RDBの設計方法の確立と4GLの機能を最大限に活用することにより、従来の3GLベースのコーディング量に比べ80%削減した。

②開発工期の削減

コーディング量の削減によって工期を40%削減した。コーディング量の削減効果よりも効果が小さいのは設計-施工（カスタマイズ時）をスパイラルに行うためである。プロトタイピングと試験を繰り返しブラッシュアップすることによって、ユーザの要件に合ったシステムを構築することができる。

③メンテナンスビリティの向上

コーディング量やテーブル数の削減によってシステム構造自体がシンプルになり、カスタマイズ等が容易になった。カーネルを抽象化し汎用性を持たせることによって、一つのカーネルを複数の処理に利用できる。