公共分野のお客様に対する日本IBMの取り組み

分離調達への対応と、大幅なコスト低減を可能にする、中央省庁／地方自治体向け新文書処理アーキテクチャーを開発

日本アイ・ビー・エム株式会社（以下、日本IBM）では、総務省が中央省庁におけるシステム調達のために示したガイドラインである「情報システムに係る政府調達の基本指針」の2007年7月からの適用に先立ち、分離調達に対応できる文書処理システムのアーキテクチャーづくりに取り組み、デモ・システムを完成させました。今回構築した文書処理システム・アーキテクチャーでは、SOA（Service Oriented Architecture：サービス指向アーキテクチャー）とWeb 2.0テクノロジーを取り入れることで、分離調達への対応と飛躍的なコスト低減の可能性を提案していることが特徴です。

本稿では、日本IBM公共事業の取り組みとして、官公庁担当CTO（Chief Technical Officer：最高技術責任者）オフィスが中心となって開発した新文書処理・管理アーキテクチャーDOLCE（Document Oriented business process on Loosely Coupled Environment using service oriented web 2.0 architecture）の概要をご紹介します。

日本アイ・ビー・エム株式会社
官公庁担当CTO
ディスティングイッシュト・エンジニア（技術理事）
ITアーキテクト
長島 哲也 Tetsuya Nagashima

［プロフィール］
1978年、日本IBM入社。製造システム事業部にて、システムズ・エンジニアとして、お客様のシステム構築や安定運用をリード。
1993年、ITアーキテクトとして、メインフレームからPCに至るまでのあらゆる技術、言語を担当する。
1999年、テクニカル・サポートに移籍。その後、e-ビジネス分野の先端プロジェクトの立ち上げに従事。
2003年、ソフトウェア事業部に移籍、e-ビジネス・システム構築経験に基づくお客様サポートや各種コンソーシアム活動を通じてIT業界に貢献。
2004年、EA & SOAのエバンジェリストに任命され、EA+SOAの普及活動を実施。
2007年から現職。

■ 分離調達に対応したアーキテクチャーづくり

2007年7月からわが国の中央省庁は、システム調達の新たなガイドライン「情報システムに係る政府調達の基本指針」を適用し、分離調達を実施することになりました。分離調達とは、5億円以上の開発費が必要なシステムについて、設計・開発、運用、保守を分離して調達し、特に設計・開発においては「基盤システム」とその上で稼働する「個別システム」群に分けるというものです。システム1件当たりの開発規模を小さくすることで、より多くのIT（情報技術）ベンダーの参入を可能にし、競争を促して調達コストを下げるという狙いがあります。日本IBMではこの取り組みに対応するために、2007年1月から、分離調達に対応可能でかつコスト低減に寄与できるシステム・アーキテクチャーづくりをスタートさせました。

■ 書類の書式ではなく、プロセスの観点【状態管理の観点】から共通化

わたしたちは、まず「分離調達」や「コスト低減」を可能とするITシステム・アーキテクチャーが存在するとすれば、それを、中央省庁の基幹業務を支えるITシステムへ適用できなければ無意味であると判断し、「中央省庁の基幹業務とは何なのか」から検討を始めました。各府省の役割は、「政策を立案する」「国民からのさまざまな申請や質問に対処する」などです。しかも、これらの処理をスムーズに間違いなく遂行するために「公的文書」を介して業務を遂行しています。言い換えれば、中央省庁の基幹ITシステムは「公的文書処理・管理システム」であると言っても過言ではないでしょう。

また、中央省庁や地方自治体で稼働中の公的文書

を扱うITシステムを棚卸しすると、実に興味深いことが分かります。多くのシステムは、利用者情報管理や認証、承認といった基本的な「共通処理」と、それぞれの文書を処理する個別処理の2層構造になっています。しかも、個別処理部分は、文書の種類ごとに構築されています。なぜなら、交通費精算の書類と出張命令書の書類を比べると、書式がまったく異なるため、システムも個別に構築した方が効率良さそうに見えるからです。その結果、同じような構造を持つITシステム（**図1**）を幾つも構築・運用してきたことがコスト高の一要因になっています。

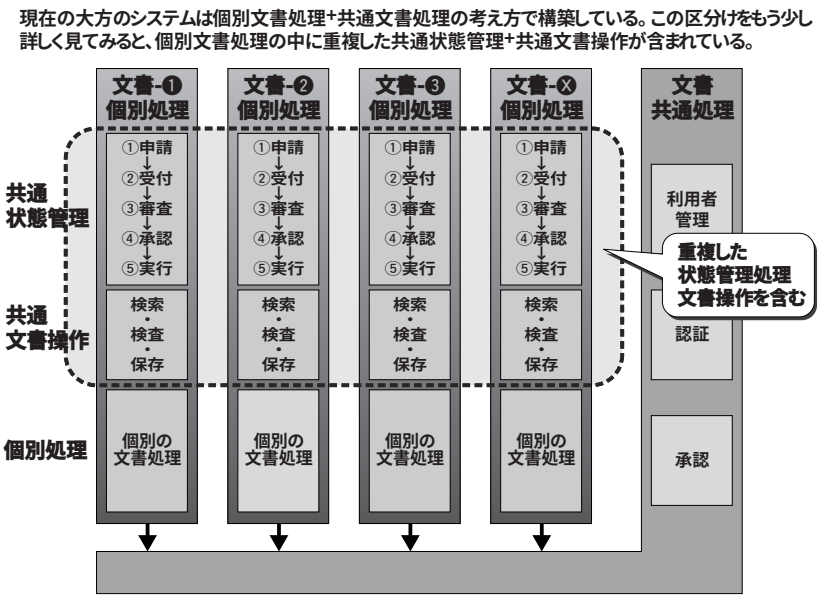

図1. 中央省庁や地方自治体の公的文書処理・管理システム現概念アーキテクチャー

しかし、書式が異なるそれぞれ個別のように見える文書処理を、書式の観点ではなく、そこで発生する書類処理の観点から分析してみると、申請者が「申請する」、申請された書類を「受け付ける」、上長に「承認をもらう」、それを別部門で「審査する」、審査結果に基づいて「実行する」、という処理は同一であることが分かります（図1：共通状態管理部）。表題には「プロセスの観点から共通化」と表現しましたが、実際には、プロセス管理のように、順番が問題ではなく、公的文書が「申請状態」「受付状態」「審査状態」「承認状態」「実行状態」にあることを管理できていることが重要です。

つまり、申請－受付－審査－承認－実行という状態管理の観点で、共通する処理をくくることができれば、書式の異なる書類も共通のIT基盤で処理することができます。極端に言えば、中央省庁に一つ設置した共通システムですべての省庁が扱う文書処理／管理が可能となり大幅なコスト低減が可能となります。また、ここで導出された「申請－受付－審査－承認－実行」という状態管理パターンもすべての府省で一種類ではないにしろ、数～数十パターン程度にまとめられるように思えました。

さらに、書類そのものをXML（Extensible Markup Language）で表現できれば、書類検索や書類検査、書類保存などの「文書そのもの」の操作部分も共通処理としてくくり出すことが可能であると考えました（図1：共通文書操作部）。

■ 低コストで共通文書処理・管理システムを構築するための問題点

図1で示した「公的文書の処理・管理」を実現しているITシステムでは、利用者管理・認証・承認の「共通処理」と、文書の種類ごとに構築されている「個別処理」の2層構造を取っていること、さらに、個別処理部分は、「共通状態管理」「共通文書操作」「個別処理」の三つに分けられ、「共通状態管理」「共通文書操作」部分は、追加共通部分としてくくり出せることを説明しました。しかし、追加の共通処理部分や個別処理部分を低コストで実現できなければ意味がありません。

まずは、「共通状態管理」部分ですが、「状態を持たせ、管理する」ためにはステートフルなデータベース・アプリケーションを開発しなければなりません。また、状態管理のパターンも前述したように1パターンにまとめることはできず、数十パターンになると考えるのが現実的です。実際、状態管理を行いつつ、エラー時に処理を元に戻したり、同期を取ったりする処理を数十パターン分もプログラミングするのは大変なばかりでなく、コストも掛かりますから、低コストで構築するための工夫が必要です。

次に「共通文書操作」についてですが、文書そのものの表現形式や文書のITとの接点である画面を、従

来のようにミドルウェア製品や特定のベンダー技術を用いて構築すると、共通に文書を操作することに制約が生じやすくなります。例えばHTML（Hypertext Markup Language）で文書を操作する画面を構築した場合を想定しましょう。通常この画面上には文書の起票者情報が配置されています。HTMLのタグ自身は画面上のレイアウトを規定するものですから、<space>
などの画面レイアウトに必要なタグは存在します。しかし、<name>のような情報の属性を表現できるタグは存在しません。従って入力されたHTML画面から姓や名を取得し起票者情報として利活用するには自分自身で固定入力エリアからデータを取得し<name>タグを用いてXMLに変換したり、データベースに格納したりする独自の処理を実装する必要があります。「共通文書操作」では、このような独自の処理をなるべく実装しないでも済むように、文書の表現や文書の共通処理に工夫が必要です。

最後に個別処理部分についてです。前述の「情報システムに係る政府調達の基本指針」に沿った共通部分と個別部分の分離調達を実現させるためには、アーキテクチャー上、この個別部分と「共通文書操作」や「状態管理」との関係を疎結合に保つことが最も重要であり、工夫が必要です。さらに、個別部分は文書数分の画面作成が必須ですから、低コストで構築できることも重要です。

■ DOLCEにおける問題の解決方法

そこでわたしたちは、「共通状態管理」を行うために、SOAの技術を積極的に取り入れることとしました。XMLベースのワークフロー記述言語であるBPEL（Business Process Execution Language）を、フロー処理に用いるのではなく、ステート・マシンとして状態管理に使うことで、目指すアプリケーションの開発が可能ではないかと考えたのです。また、BPELのツールを利用すれば、状態管理パターンが20パターンあっても30パターンあってもプログラムレスで容易に対応できると考えました。ただ、BPELを用いて「状態管理」を行う場合、実際に処理する人間との接点が必要となります。つまり、昔、管理職の机の上にあったような「処理待ち箱」「処理済み箱」に相当する機能を用意しなければなりません。「処理すべき書類がたまっているか？」「どの書類は処理が終わっているのか？」が一目で分かるような仕組みが必要なのです。BPELにはそのような機能は搭載されていないため、わたしたちは、2007年6月にリリースされ、OASIS（Organization for the Advancement of Structured Information Standards）により標準化が進められつつあるBPEL4People（WS-BPEL Extension for People）に注目しました。複数のサービスを連携させることが得意なBPELは、人間の行動がからむ処理には向いていません。そこで、BPEL4Peopleを用いて、申請－受付－審査－承認－実行という各状態に「処理待ち箱」と「処理済み箱」を置くことで、人間の処理との接点を提供することにしました。

また、「共通文書操作」を実現するために、特に注目したのはXMLです。現在、情報のやり取りにXMLはもはや標準になっています。本格的に情報をやり取りしようと思ったときにXMLは必須であると言っても過言ではないでしょう。「共通文書操作」では、XMLを用いることで、書類検索や書類検査、書類保存などの「文書そのもの」の操作部分も共通としてくくり出すことができると考えました。特に昨今話題となっているXMLデータベースを利用することによって、「検索」についてはXQueryによりプログラムレスで定型／非定型の共通操作が可能となります。「検査」についてはルール・エンジンを用いることによって大方の処理をプログラムレスで実装可能となると考えています。例えば、「<a>3 4 <c>10</c>」と入力された文書に対しての検査とは、「a + b < 50」のようなチェックをすることです。これはまさにルールを記述することにほかならないと考えています。最後に、「保存」については、データの永続的利用（移行性）と可視性を機能的に保障することができればよく、最終的には管理のしやすい媒体（ファイル・システムやXMLデータベース）を選択することになると考えています。いずれにしても、「共通文書操作」では、XMLで公的文書を表現することで「検索」「検査」「保存」のすべてにおいて、共通に処理可能であると考えています。

DOLCEでは、こうした最新のテクノロジーを用いることで、図1に示した「共通状態管理」「共通文書操作」「個別処理」のうち、「状態管理」「文書操作」を共通に

処理可能とする環境をつくりました。さらに、「個別処理」からこれらの共通機能を疎結合で利用するためのAPI（Application Programming Interface）を準備しました。このような対応を取ることで「個別機能」自体は共通部分から独立させて開発できるようにしました。

これらの対応をしたDOLCE共通処理環境、すなわち「共通状態管理」や「共通文書操作」部分の開発には、最新のテクノロジーをふんだんに用いることになるため、開発するエンジニアにとっては確かに敷居の高いものになるかもしれません。しかしながら、この部分は文字どおり共通化されますから高度なスキルを持ったITエンジニアが1回だけ開発します。一方、「個別処理」の部分は、「共通状態管理」や「共通文書操作」のAPIの使い方のみを理解すれば、容易に開発・保守できます。仮に書類が1万種類あったとしても、開発そのものは単純・簡単ですので、それこそ低コストな通常のITエンジニアすなわち「高度なスキルを必要としないITエンジニア」を多数動員することで解決できるでしょう。

また、DOLCEでは、独立した「個別機能」はどのようなテクノロジー／ミドルウェア／開発スタイルを取ってもよいようにしています。イントラネット・アプリケーションで高度なGUI（Graphical User Interface）が必要であればクライアント／サーバー型のファット・クライアントで作成してもよいでしょう。また、インターネット・アプリケーションであれば、ブラウザー・ベースのシン・クライアントで作成も可能です。開発者にとって大切なことは、文書をITに写像する画面を速く簡単に作成できることです。また、利用者にとって大切なことは、マニュアルを参照しなくてもほとんどの操作が標準的な方法で利用可能となっていることです。これらの相反する要件を現実の世界で達成させる動きが、昨今ちまたをにぎわせているWeb 2.0ととらえることができます。特に、画面作成・利用の標準であるAjax（Asynchronous JavaScript + XML）は前述の作成者／利用者どちらの要件も満たしていますので、DOLCEでは推奨開発スタイルと位置付けています。例として交通費精算のアプリケーションを想定してみます。経路情報の入力は「個別処理」開発ですが、その情報を基にインターネット上に存在する乗り換え案内とマッシュアップして運賃を求め、交通費精算アプリケーションに入力させるような処理を、現在ではAjax対応のオープン・ソース開発環境を用いて簡単に作成できます。

■ DOLCEにおける文書処理／管理の流れ

図2では、DOLCEを用いた文書処理・管理の流れを順を追って説明しています。ここでは、国民が国に対し開示請求の申請処理を実施した後の、中央省庁の職員の方々が受付をする処理を例にとって説明します。

(1)職員が「受付」を実施するには、「受付」処理と対応する「受付処理待ち箱」から、当該職員が処理に必要な書類一覧を一覧取得APIを通じて取り出し、画面に一覧表を表示します。

(2)一覧表から処理する書類を選び、「受付画面処理」により書類を表示させます。つまり「受付処理待ち箱」から指示された書類を取り出して、作業のために机の上に置いた状態です。

(3)書類の入力項目を埋めていきます。その際に、後述する「個別入力アシスト」機能を利用して、ヘルパー・ファンクションを呼び出すことも可能です。例

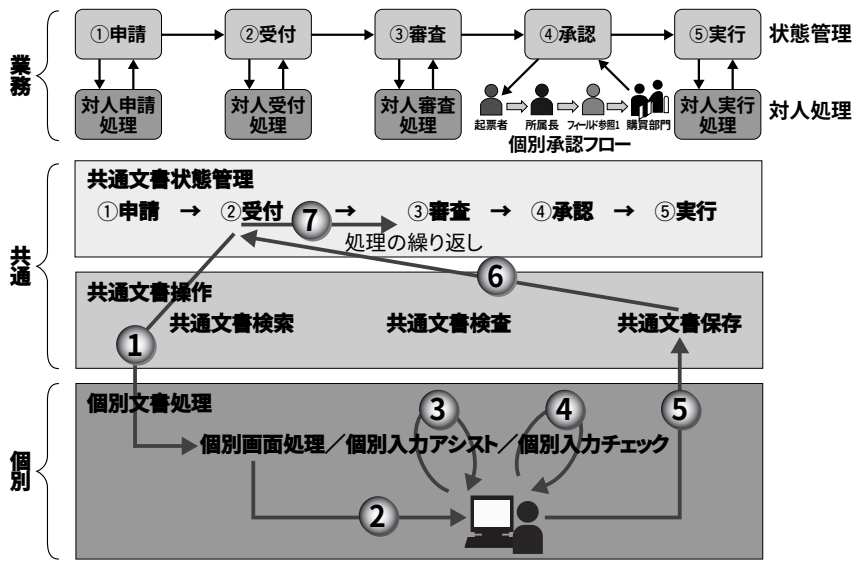

図2. DOLCEにおける文書処理／管理の流れ

えば、日付の入力欄でカレンダーを表示させたり（図4参照）、交通費の申請であれば、路線検索の画面が表示され、経路を入力すると自動的に金額が計算されたりするような処理です。

(4) 書類作成時には、「個別入力アシスト」と同時に「個別入力チェック」の機能も提供します。例えば、「A欄の値は、B欄とC欄の合計より多くなければならない」とか「A欄に入力する値は0〜100の範囲でならなければならない」といった制限に合わせて入力チェックを行います。

(5) 書類が完成すると「共通文書操作」の「共通文書保存」機能により、公的文書として保存されます。実際には、保存の前に「共通文書検査」機能を用いて保存するデータが正しいことを検査してから保存します。

(6) こうして「受付」の一連の処理が完了すると、「共通状態管理」に戻り、書類は「受付」処理の「処理済み箱」に入ります。

(7)「受付」処理の「処理済み箱」に入った書類は、自動的にさらに「審査」処理の「処理待ち箱」に移ります。こうした処理を繰り返して、申請－受付－審査－承認－実行という各処理が実行されます。

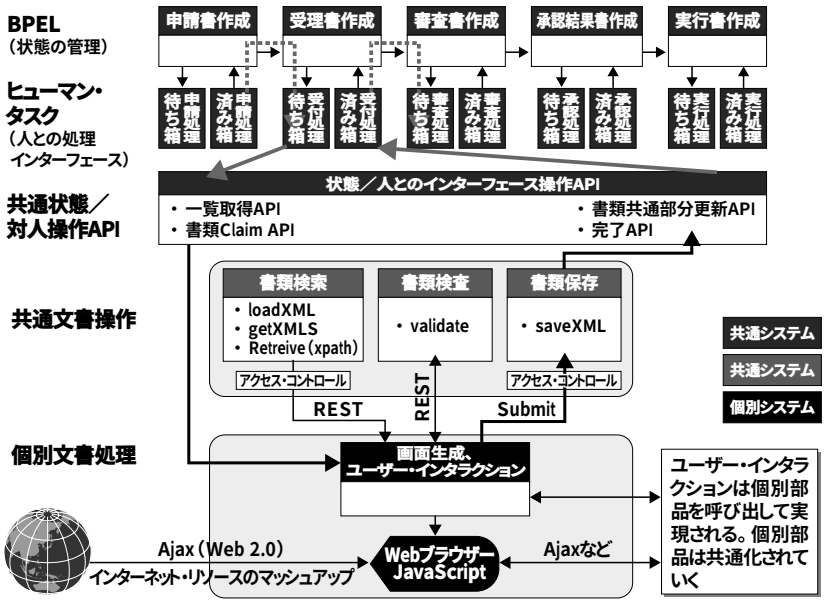

図3. DOLCE論理アーキテクチャー

■ DOLCE実装における考慮事項【共通と個別の処理を切り分け、疎結合を実現する平易なAPIセットを構築】

DOLCEを実装の観点から示したものが**図3**です。

前述のように、DOLCEアーキテクチャーは、大きく「共通状態管理」「共通文書操作」「個別文書処理」の三つに分けた構造をしています。

図中央の「共通状態/対人操作API」に、DOLCEで使われる数個のAPIを示しています。「一覧取得API」は、前述したように「処理待ち箱」の中にたまっている書類の一覧表を取ってくるAPIです。二つ目の「書類ClaimAPI」は、一覧から選んだ書類を処理する前にほかの職員から2重処理されないように排他制御の印を付ける働きをします。「Claim」とは「求める／要求する」という意味です。書類をClaimした後は、実際の書類の中身を取得する必要がありますが、こちらは、「共通文書操作」中の「書類検索」に分類された「loadXML」APIが受け持ちます。ここからの処理は、ほとんどが個別文書処理になり、共通機能は何ら関与することはありません。

また、個別文書処理の中では、ヘルパー・ファンクションを用いて入力アシスト機能を実装します。例えばAjaxを用いて、マップ・サービス、検索サービス、運賃計算サービスなどのSaaS（Software as a Service）をマッシュアップすれば開発の手間は掛かりません。もちろんSaaSである必要もありません。自前のデータベースから入力候補を検索しポップアップ形式のリストボックスで入力アシスト機能を作成することも可能ですし、イントラネット上で提供されているSaaSサービスを利用するということも考えられます。例えば、書類に地図を書き込みたいのであれば、Googleマップのような地図情報エンジンを利用することもできるでしょうし、外部のサービスを利用することに問題があるならイントラネット上にマップ・サーバーを置くということも考えられます。

こうして「個別書類の作成」の処理が終わると「共通文書操作」の「書類検査」機能を用いて、入力された情報が正しいか否かを縦・横・斜めの観点から整合性をチェックします。整合性が確保できたら、「書類保

存」に分類された「saveXML」APIを用いて書類を保存します。これらの操作で一連の処理は完了ですから、完了したことを「共通状態管理」に通知するために「共通状態／対人操作API」中の「完了API」を発行します。

つまり、個別処理を担当するITエンジニアは、主に「共通状態／対人操作API」と「共通文書操作」に分類された数種類のAPIを覚えれば開発が可能になります。実際この部分の開発を容易にするために、DOLCEでは「共通状態／対人操作」「共通文書操作」API群は最も平易なインターフェースであるREST（Representational State Transfer）を採用しました。RESTであれば、それを使いこなすための技術情報や開発ツールは、無料でインターネット上のWebサイトから手軽に入手できます。その結果、例えば地方自治体が、地場の中小規模のソフトウェア開発会社に開発を依頼した場合に、共通状態管理機能内部の複雑な実装をすべて理解しなければ開発できないようなことがなくなり、個別部分を自社の抱えるITエンジニアの背丈に合った技術・責任範囲で独立して開発可能になります。

また、申請ー受付ー審査ー承認ー実行という各処理中の、承認処理については、他処理のように個別に書類処理を作成する代わりにワークフロー構築ツール、具体的には、業界標準のBPELサーバーなどを用いて構築します。また、極めて複雑な承認経路をサポートしたり、承認者を動的に変更したりするような複雑な要件が確定している場合には、ITベンダーが既に提供している専用ワークフロー構築ツール（例えば、IBM FormWave for WebSphereなど）を利用して組み込むことも可能です。このように既存ツールを組み込みやすくなっているのも、「共通状態管理」や「共通文書操作」と「承認個別機能」が疎結合で分離されているDOLCEの設計思想に負うところです。

■ SOAの新しいアプローチとして Business Model Patentも申請

現在のSOAメソドロジーは次の手法により共通として利用可能なサービスを抽出する方法を取ります。

(1)特定のビジネス・ドメインの分解により適切な粒度のサービス候補を抽出。

(2)ボトムアップ手法により、現存するITシステムが提供している機能をサービス候補として抽出。

(3)(1)と(2)で抽出されたサービス候補の中から適切な粒度でサービスを決定する。

この分析方法では、文書処理機能を細かく分解することになります。例えば、「1. 交通費精算」の書類処理を分解すると、第一レベルでは、「1ー1. システム利用のための認証処理」「1ー2. 申請者情報の処理」「1ー3. 経路情報と金額の処理」「1ー4. 手当情報の処理」…「1ーX. 承認者と承認経路設定処理」と分解できます。また、「2. 出張申請」の書類処理を分解すると、「2ー1. システム利用のための認証処理」「2ー2. 申請者情報の処理」「2ー3. 出張場所の処理」「2ー4. 出張目的の処理」…「2ーX. 承認者と承認経路設定処理」と分解できます。この二つの例から抽出される共通サービスは、結局、当解説の冒頭で紹介した「利用者情報管理」(1ー2と2ー2から)、「認証」(1ー1と2ー1から)、「承認処理」(1ーXと2ーXから)にとどまってしまいます。しかも、もともとこのレベルの共通化では個別文書処理システム部分の開発規模が依然として大きいことが問題でした。

前述の単純な例からも明らかなように、現在のSOAメソドロジーを用いてサービスの抽出を試みると、最初の第1レベル分解時点で、交通費精算の本来の目的である「1ー3. 経路情報と金額の処理」「1ー4. 手当情報の処理」や、出張申請の本来の目的である「2ー3. 出張場所の処理」「2ー4. 出張目的の処理」は個別処理に分類されてしまいます。つまり、文書処理などのバックオフィス業務に対して、分解により共通部分を見いだしても、「利用者情報管理」「認証」「承認処理」以外は共通部分が発見できない業務分野であると考えられます。そこで、わたしたちは公的文書の処理を現在のSOAメソドロジーとは逆に、抽象化して考えてみることとしたのです。その結果「公的文書」は文書の種類にかかわらず基本型として、「申請」「受付」「審査」「承認」「実行」という五つの状態を持っており、書類の処理は状態の遷移によって成り立っていることを発見しました。また、この状態には交通費精算のような省庁内処理であれば、基本型から「受付」状態を除いた変形パターンがあることも発見しました。この発

見は「DOLCEにおける問題の解決方法」の章で述べたように、従来個別機能として開発していた難しい部分を共通機能として一回構築し、その後は共有する部分と、本当に個別に開発する部分とに分離することによって、個別部分は単純・簡単に開発可能とすることができるようになったと考えています。また、このアプローチを採ることによって、総務省のガイドラインである「情報システムに係る政府調達の基本指針」への対応（いわゆる「分離調達」への対応）も可能になったと思います。実際、今回採ったアプローチは、過去に提案されたSOAのどんな方法論の中にもない手法であり、その意味からもBMP（Business Model Patent）を申請中です。

■ フォーム作成標準ツールを利用してスピーディーな開発を実証

DOLCEの特徴は「従来、個別部分に位置付けられていた機能を共通に持っていける」ことを実証したほかに、「低コストで個別部分を作成可能にした」ことが挙げられます。そこで、DOLCEを用いて実際に開発を行った場合、どの程度の効果が得られるのか、確定申告のフォームを作って（**図4**）実際に試してみることにしました。ここで使用したツールは、W3C（World Wide Web Consortium）の勧告であるXFormsというXMLによる文書定義の標準に対応したIBM Workplace™ Forms Designer（以下、Workplace Forms）です。結果はレイアウト部分作成に2時間、書類として完成させるために入出力変数タグとレイアウトのマッピングに3時間、合計約5時間で作ることができました。こうして作成したフォームに、実際に文書処理を行う機能やヘルパー・ファンクションを組み込む必要がありますから、それらも含めて1〜2日で完成します。単純に掛け算で計算できるわけではありませんが、仮に前述の検証結果から1人のエンジニアが月に10フォームを作れるとすると、100人月で1,000画面の作成が可能と計算できます。*

こういったフォーム作成開発ツールを用いると、生産性を大きく向上できるようになります。その意味では、文書個別処理部分を独立して作成可能なDOLCEアーキテクチャーは非常に有効であるということができます。

＊DOLCEは、Workplace Formsを前提としているわけではありません。

画面作成の時間短縮
- **既存書類を土台にして入力エリアを定義する手順で画面を作成する。**
- **左の確定申告フォーム作成にかかった時間は、5〜10時間であった。**

画面のヘルパー機能の作成
- **画面遷移せずに、普通の操作性を確保する（タブ飛ばし、カレンダー・ガジェットなどの動的吹き出し）。**
- **それを実現するプログラミング・スタイルとしてAjaxを採用する。**

画面定義のフォーマット
- **政府（総務省）勧告「調達仕様書の要求要件として、オープンな標準を優先して記載する」に従い、W3C勧告の標準であるXFormsを採用する。**

図4. 個別画面の作成

■ DOLCEを広く紹介するために常設デモ環境を設置

日本IBM公共事業としてはこのDOLCEを広め、中央省庁や地方自治体のシステム構築に携わっているすべてのITベンダーの方々に取り入れていただくことが重要と考えています。

そこで、2007年8月に東京・丸ビルに設置されたイノベーション・ラボや、東京・渋谷のソフトウェア・コンピテンシー・センター（SWCoC：Software Center of Competency）で実際にデモをご覧いただけるように準備中です。また、これらのデモ環境へはリモート接続も可能なので、ご依頼があればお客様にお伺いしてデモをすることも可能です。

わたしたちはこのアーキテクチャーに「usingサービス・オリエンテッドWeb 2.0アーキテクチャー」という名前を付けましたが、Web 2.0の技術をエンタープライズ・システムに積極的に取り入れた事例としても、興味深いのではないでしょうか。