イネーブラ：DRB
――エンタープライズアプリケーション統合に向けたアプローチ

Enabler : DRB
――Approach for Enterprise Application Integration

丸 岡 茂 敏, 小 枝 康 二

要　約　DRBは，システム連携要素技術であるエージェント技術やXMLを使用し，エンタープライズ環境にある様々なデータやアプリケーションを統合するイネーブラである．情報システムに求められている最新のシステムは，インターネット時代の多様化するビジネスへ柔軟に対応することであり，蓄積された情報資産を有効活用することである．本稿では，エンタープライズ・アプリケーション統合を目指すDRBの設計思想および，DRBのビジネス・プロセスの自動化にむけた取り組みを紹介する．

Abstract　DRB（Distribution Request Broker）is an enabler software which offers the cooperation and integration of various customer application systems and related data in an enterprise environment, including software products from independent software vendors, using the agent technology and XML（extensible markup language）. User requirements for the current information system are to cope in a flexible manner with more diverse business in the Internet age, and to maintain and use cumulated information assets effectively.

This paper discusses the design philosophy of DRB focusing attentions on the integration of application systems running on the enterprise systems, and the approach to be taken in automating business processes in the enterprise environment.

1. はじめに

通信技術の進化やインターネット技術の普及などにより，昨今はビジネス社会における取引のルールやコミュニケーション環境が大きく変化を続けている．例えば，商品流通における間接（卸）取引から直接取引への流れは，取引相手の数を膨張させるとともに，取引を実行する際の通信手段をよりパーソナルな手段（PC・携帯電話など）へと切り替えさせている．また，企業間の競争は企業グループによる囲い込みへと発展し，この競争に参加するためには一定の条件を満たすコミュニケーション手段を実装する必要がでてきている．

情報システムにとって，この流れは，システムの柔軟性に対する要求につながっていくことになる．つまり，最近のシステムは，多様化するメディアに対する情報の受け渡しへの対応，蓄積された情報資産の有効活用のための新サービスの提供，利用者がどこに存在しようともコミュニケーション可能な高い情報伝達能力，等々の要件を迅速に満たしつづけることが求められている．分散された企業システム全体を一つの情報資産と捉え，これらの要件に対応させていくためには，システム間の機能統合が必要となる．また，このシステム統合は継続性を要求される．常に新しいものを受け入れ可能とするために許容性に優れ，ルールの変化に対して即応できるような柔軟性

を保持する有機的なものでなければならない．

Distribution Request Broker（以降 DRB）は，複数のアプリケーションや市販パッケージ製品における連携，統合作業を支援するイネーブラ・ソフトウェアである．現在のバージョンでは，メインフレーム上の基幹業務システムと Windows システムに搭載される新デバイスや新規アプリケーション・システムとの統合を容易にするための機能を実装している．本稿では，DRB 開発にあたり着目した移動エージェントおよび XML という要素技術を中心にとらえ，連携や統合を支援するインフラストラクチャとしての実装のありかたについて論じる．

2. DRB の概要

この章では，DRB の構造，製品上の特長について説明する．

DRB は，Windows サーバを中核として稼働するアプリケーションおよび，この利用者に対する情報の流通とプロセスの連携を支援するイネーブラ・ソフトウェアである．情報の管理，発信メカニズムの一元化，データ・パッシングを基本とし，Windows サーバ環境を中心とした異種サーバ間でのプロセス連携機能を可能している．

具体的には，DRB はプロセスを統合的に管理する実行基盤と EDI，FAX/OCR，音声応答などの市販パッケージ連携のための連携部品（アダプタ）を提供している（図 1）．基幹業務アプリケーションや EDI などの各連携アプリケーション間は，メッセージファイルを媒介としてプロセス連携され，他 OS で稼働している基幹業務アプリケーションのデータを Windows システムより EDI 送信や電子メール送信，FAX 発信することが可能である．また，EDI システムより受信したデータを基幹業務アプリケーションに投入したり，他連携アプリケーションに投入したりすることも可能である．

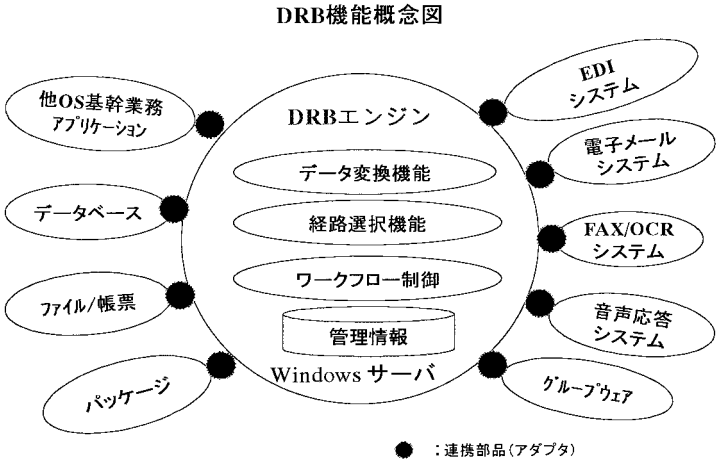


図 1　DRB 機能概念図

2.1 構　　造

DRB は以下の要素で構成されている．

・DRB エンジン（スクリプト言語とその実行管理基盤）
複数サーバ間におけるプロセスの連続実行機能を実装したジョブ実行管理機能.

・データ変換（トランスレータ）
COBOL 書式と可変長テキスト間の変換処理など，汎用的なデータ変換機能.
スクリプトよりオブジェクトとして取り扱い可能な実行機能を提供する. また，データ連携を行う相互のシステムが定めるデータ・マッピングおよび，セマンティックスの変換形式を定義するための GUI プログラムも提供する.

・データ交換
各サーバ間のデータ転送手段として，FTP・メール送受信などの機能を提供する.
次のアダプタとして提供する機能のうち，EDI システムとの連携機能なども一種のデータ交換手段であると捉えることができる.

・アダプタ
EDI 関連処理や FAX 通信機能に代表されるような業種・業務に関わりなく広範囲での使用が見込まれる機能について，連携対象のパッケージ製品を特定して DRB エンジンのもとで基幹業務システムなどのシステム環境と連携可能とするモジュールである．また，このアダプタに相当する部分を利用者が独自に開発できるように，デザインツール，変換部品およびガイドライン等を開発キットとして準備中である.

・サービス
DRB 内部に独自に提供するアプリケーション相当のモジュール.
現時点では「登録と発行（Subscribe & Publishing)」モデルに基づくクライアント向け情報提供を支援する DRB デリバリサービスを実装している.

DRB の論理構成を図 2 に示す.

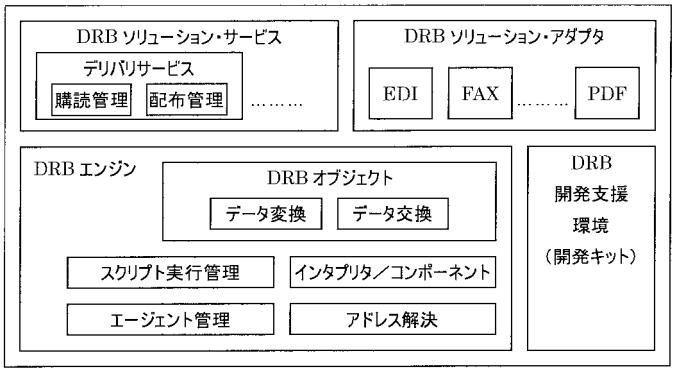


図 2　DRB 論理構成

2.2 動 作 概 要

DRB の基本となるエンジンは，ネットワーク・ノード上に配置された Windows

サーバ上で稼働する．アプリケーション間のデータ連携は，このDRBエンジンを介して行う．複数のアプリケーション・システムが単一のDRBエンジンを介して互いに通信しあう方式は，ハブ・アンド・スポーク・トポロジーに基づいている．多数のアプリケーションが相互に連携する場合，ハブ・アンド・スポーク方式を取ることで，システム間の接続の数を減らすことができる．また，この方式では，経路選択処理や変換処理を一元管理することが可能となる．そのため，追加変更の多い連携構成に対して柔軟に対応できるようになる．　複数のノードに配置されたDRBエンジンは互いに通信レベルで結合し，相互間でのプロセスとそれに付随するデータの移動を可能にしている．この機能は移動エージェントの考え方をもとに実装しており，実行すべき処理を個別のエージェント自体が管理することで，処理の追加変更や環境変化に対する自由度を高めている．複数ノードをつなぐDRBエンジンとその周りに配置されるアプリケーションあるいはパッケージ・ソフトウェアはスノーフレーク型に展開され，より広い範囲のネットワーク上に展開されるシステムに対する適用が可能となる（図3）．

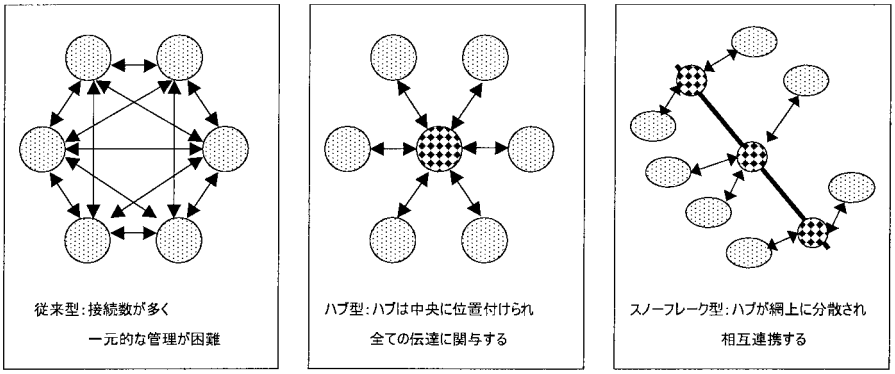


図 3　従来型連携とハブ型，スノーフレーク型連携

実装の一例として，取引業務など多数の利用者をもつシステムを複数の入出力機能に対応させる場合などがあげられる．基幹システムと入出力側の接続（FAX，WWW，メール，プリンタなど）を個々の仕様に応じてpoint-to-pointで接続を繰り返すことは，接続数分のアプリケーションが必要となる．これに対して，DRBはハブ型ネットワークの中心で経路選択処理を行い，また変換機能によってそれぞれのノードが指定するデータ仕様の相違を吸収させる仕組みとなっている．この方式は無駄が少なく，2番目以降の接続作業に対する効率を向上させ，また実装済み機能に影響を与える可能性を削減することが可能である．

2.3 データ変換

異なる仕様を持つシステム間を仲介するにあたり，データ構造を変換するための機能は重要である．データ変換の方式は，あるシステムが持つ固有形式のデータを別のシステムの要求する固有形式に直接変換する方式と，XML（eXtensible Markup Language）のような標準的な言語を中間書式とし，これを介して多対多の変換を可能と

する方式に大別される．

DRB は前者の方式を独自のシステム中に提供するとともに，後者のあり方に焦点をあて，XML トランスレータとの連携および，レガシー環境での XML トランスレータの実装をすすめている．

3. システム連携技術

この章では，DRB の提供にあたり採用した要素技術の概要と，DRB 上における具現化アプローチについて解説する．

3.1 移動エージェント

移動エージェントとはデータとプログラムが一体となり，オブジェクトとしてネットワーク上を移動し，様々なサーバ上で処理を実施し最終的に，処理結果を持って送り出し元のサーバに戻り，結果データをその処理の要求者に返すことをいう．

あるアプリケーション・システムを別のサーバ上に構築されたアプリケーション・システムと結び付ける仕組みとして，DRB ではこの移動エージェント方式を採用している．エージェントは，実行に必要なプログラムコード自体に加えて変数値などの内部状態と処理中のファイル化された中間データ，およびプログラムの実行状態（プログラムコードの第何行目を実行中であるか)を保持した状態でサーバ間を移動する．エージェントは，移動前の実行状態を移動後のシステム上に再現化する．この構造は Telescript に類似した実装方式である[1]．

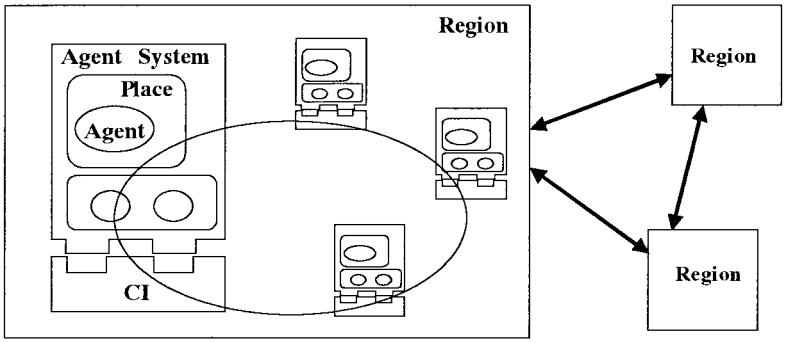
エージェント技術に関する標準化としては，OMG（Object Management Group）の MASIF（Mobile Agent System Interoperability Facilities）および FIPA（Foundation for Intelligent Physical Agents）の活動があげられる．MASIF は，異種エージェント/エージェント・システムの相互運用を確保するための標準化を目指している．また，エージェント/エージェント・システムの論理構成を規定するとともに，エージェント移動に必要なエージェント間相互運用性について定義している．FIPA は，エージェントが相互作用する環境におけるコンポーネント間のインタフェースを規定する．この団体はエージェント・システムの論理構成とエージェントの内外のコンポーネントに対する振る舞いを仕様化し，FIPA 仕様に基づくサブシステム間における相互運用性を保証することを目的としている[2]．

MASIF における移動エージェントに対する定義は，

> *A mobile agent is not bound to the system where it begins execution. It has the unique ability to transport itself from one system in a network to another*[3].

となっている．移動エージェントは「ネットワークを越えて，自身が別コンピュータに移動する能力を持ち，起動元システムに所属しないもの」であり，この実行体とそれを支援する論理的な環境を MASIF は図 4 のように定義している．

これに対し，DRB のエージェント・システムは，図 5 のような構成を持つ．特長としては，運用管理効率の向上と分散環境への容易な展開を目的に，Region（リージョン：同じ権限を持つ複数のエージェントの集合体）内における Place（エージェント実行の場）の階層化を行っている．DRB では一つのエージェント・システムは暗黙的な一つの Place と同等と捉えている．リージョンを構成するサーバ群のうち，



- Agent：個人／組織の代理人として振る舞うプログラム
- Agent System:Agentを生成，翻訳，実行，転送，終了するプラットフォーム
- Place:Agent実行の場，Agent Systemは暗黙的にひとつ，もしくは明示的にひとつ以上のPlaceを内包する
- Region:同じ権限を持つ複数の（タイプの異なる）Agent Systemの集合体
- Communication Infrastructure(CI):Region内／外のAgent System間通信のためのトランスポート及び名前解決，セキュリティ・サービスを提供する

図 4　MASIF によるエージェント実行環境定義

連携の中心となる役割を持つサーバが DRB ハブである．リージョン内の他のサーバ上では DRB サテライトが稼働する．エージェント移動はハブ/サテライトの区別なく実行されるが，エージェントを発生させ，その実行状況をリージョン内全体にわたって管理するための場が DRB ハブである．移動してきたエージェントを受け入れ，再度移動するまでの一時的な状況のみを管理する場がサテライトの機能である．

リージョン内全体のエージェントに対する管理と運用の機能は DRB ハブ上に一元化されている．

モジュール構成における DRB エンジンは，図 5 に示すサーバ上に配置され，各々のサーバ上における Place 管理とコミュニケーションの役割を持っている．エージェントを実行するための構成情報と実行結果であるログ情報は，DRB ハブのみがこれを管理し，管理者に提供する．

次に，二つのエージェント基盤間での移動に対する実装について解説する．

FIPA は「エージェントの移動性」の仕様上で，エージェント移動に関する二つの参照モデル（‘ Simple Mobility Protocol ’と‘ Full Mobility Protocol ’）を定義している．‘ Simple Mobility Protocol ’はエージェント基盤，すなわちエージェントの実行管理を行うエンジン側で移動機能を提供するモデルである．移動処理に関するアプリケーション側の負荷は少なくて済むが，移動の範囲や厳密性はエージェント基盤側の実装に依存する．一方‘ Full Mobility Protocol ’はエージェント自体が移動機能を実装する．これはアプリケーション開発面での複雑性は増すが，エージェント自体が通信機能を使用して移動処理の完了を確認することで移動機能における確実性を高めるものである[4]．

DRB エージェント環境におけるエージェント移動は前者の‘ Simple Mobility Protocol ’を採用している．具体的なエージェント移動の流れは次の通りである．

1）　サーバ間をまたがって実行すべきデータ処理は，一貫性を持つ作業の流れとし

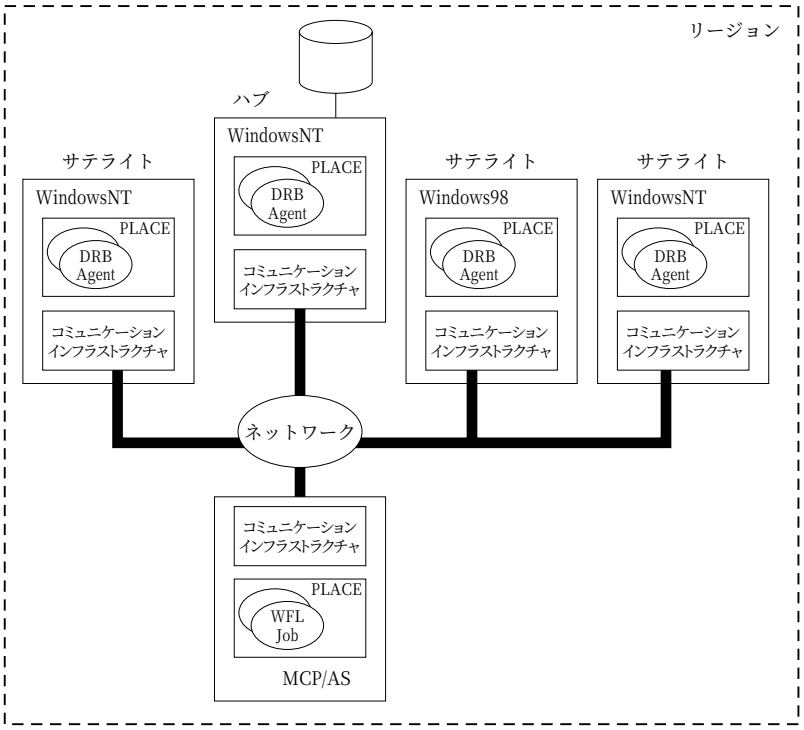


図 5　DRB エージェント環境

てスクリプト言語によって記述される．

2)　記述されたスクリプトは，DRB 基盤に対して投入されることにより，DRB 基盤が管理するエージェントとして実行される．

3)　エージェントの実行中において，他サーバへの移動処理が指示された場合，エージェント基盤はその実行を中断し，エージェントの実行環境全体をシリアライズし，移動先の DRB 基盤に対して移動を依頼する．

4)　依頼を受けた DRB 基盤は移動先の DRB 基盤とネットワークを介して会話を行い，この凍結されたエージェント・コードを宛先のサーバへ転送する．

5)　転送を受けた側の DRB 基盤ではこれをデシリアライズし，作業中の内部状態を復元した後に，移動処理の次のフローより実行を再開する．

6)　エージェントの挙動やイベント等に遭遇した際の報告は全てエージェントを最初に起動した DRB ハブに対して行われる．DRB ハブにアクセスすることで，複数のサーバ間を渡り歩くエージェントに対する単一の視点よりの監視が可能である．

図 6 は DRB の提供するスクリプト上における言語仕様とその動作状況を表す．

3.2　XML

XML は W 3 C（World Wide Web Consortium）の XML ワーキング・グループによって制定されたマーク付け言語である．自由なタグ付けによるデータ表現が広範囲のアプリケーションによる活用の可能性を高めている．WWW アプリケーションや情報系システムなどでの活用がすすめられており，近い将来には EDI など企業間の

```
Dim agent as BAgent
Dim rc as Integer

＜移動前の処理記述＞

‘抽象的または具体的 PC 名(必須)
agent.Target = “Satellite1”
‘移動先名を名前解決するかどうか
agent.ResolveRequired = “YES”
‘「ME」で「このスクリプト自身の移動」を示す
‘他のスクリプトを「派遣」する機能も同じ動作に含まれる
agent.Script = “ME”
‘自身が移動する場合, Param と Loc は使用しない
agent.Param = “”   ‘新規エージェントに渡す引数
agent.Loc = “”     ‘新規エージェントのスクリプトの所在
‘移動開始
rc = agent.Go

＜移動後の処理記述＞
```

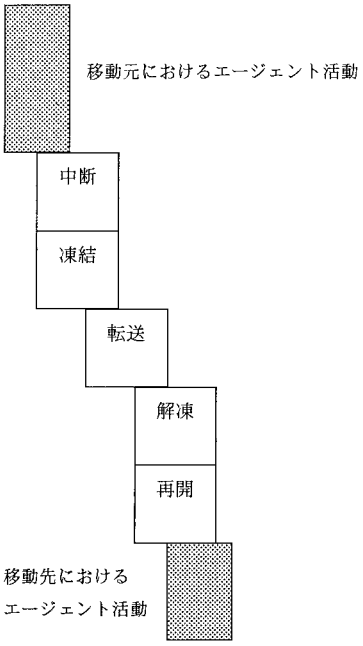


図 6　DRB エージェントの移動

情報交換においても業務データの表現として XML が取り扱われる可能性が高い[5].

XML は，アプリケーション連携におけるデータの正規化を行うケースにおいて，最有力の選択肢と考えられている．また，XML は System to System インタフェースを提供可能としている．XML の基本仕様のみを使用してフォーマットされたデータは，理解し易さという点でシンタックスの相互運用性をすぐにも享受できる．アプリケーション連携をする際に，XML ベースの技術を使用することにより，カスタム・ロジックを削減でき，構築コストを減少できる．さらに，XML は Unicode 標準対応であり，データ構造とスタイルシートの分離が可能なことから，WWW 環境や 3 階層モデルのシステム構築において，これまでにないダイナミックなアプリケーション連携が期待できる．

現在 Netscape Navigator 6.0 や Internet Explore 5.0 など主要なブラウザが XML ネイティブサポートとなっている．また，XML 対応のアプリケーション製品も増えている．データベース関連では Microsoft，IBM，Oracle，Sybase などが XML を入出力データとして取り扱える．ERP 関連では Oracle，SAP などが XML でデータ交換できるように XML インタフェースを実装している．今後，XML 対応のアプリケーションが業務データを直接アクセスできるようになり，シンタックスだけでなくセマンティックスも含めてデータ相互運用性を実現できるようになる．ただし，XML だけではアプリケーション統合を成立させることはできない．EAI（Enterprise Application Integration：全社的アプリケーション統合）など他のテクノロジーやビジネス規約と組み合わせて使用すべきであり，長期的な視野で XML の持つ有用性を考えるべきである．

DRB では XML とアプリケーション固有仕様のデータ間における変換のルール付けを行うことで単一のデータ仕様を複数の連携要求に容易に変換することが可能であることに着目し，XML を中間言語とした変換機能をコンポーネントとして提供する．この XML を基盤としたデータを移動エージェントのもとで運用することで，柔軟性に富むシステム間のデータ交換を可能としている．

特殊な例では，帳票を汎用データ化することを目的に，座標情報などのレイアウト属性を持っているデータ集合に対する XML を DRB 上で定義した．この目的は，印刷イメージを電子ドキュメントや FAX イメージなど複数のメディアに対して受け手の要求に応じて変換することにある．例えば，請求書など，複数の相手に対する情報伝達を行うケースにおいて，相手の要求または状況に応じて，FAX で送信したり，電子ファイルをメールで送付したりといった柔軟な対応を取ることが可能となる．

尚，現在 W 3 C ではグラフィック表現の XML 規約として SVG（Scalable　Vector Graphics）が検討されている．SVG 仕様中には，やはり座標点や形状，フォントなどのレイアウトや表現に関する属性が定義されており，前記の DRB 独自の XML 定義は将来的にはこのような標準化された XML 表現に置き換えられていくべきと考える．以下は現在提示されている SVG 仕様における文字列を表現するタグ表記と現行の DRB における文字列タグの表記を比較したものである（図 7，8）．

```
<text x="250" y="150"
style="font-family:Verdana; font-size:42.333; fill:blue">
Text in user space
</text>
```

図 7　SVG 仕様における TEXT タグ[6]

```
<TEXT　STR_X="480"　STR_Y="1"　FONT="STANDARD_MINCHO"　SIZE="96"
CHAR_PITCH="0"　LINE_PITCH="120"　H_DBL="0"　V_DBL="0"　BOLD="0"
ITALIC="0" UNDERLINE="0" COLOR="0">
TEST DATA OF BDXML_CONVERT
</TEXT>
```

図 8　DRB が定める TEXT タグ

DRB 側では固定ピッチフォントに対する補助属性がいくつか含まれている．これはメインフレーム帳票との互換性を維持するためのものである．

データに対する XML の変換機能では，XSLT（XSL　Transformations）と呼ばれる変換エンジンを使用する．XSLT は，XSL（eXtensible　Stylesheet　Language）と呼ばれるスタイルシートを変換指示書として定義することによって，XML ファイルを別の XML，HTML，テキストファイルなど様々な形式に変換することができる．

例えば，WWW から入力されたデータを XML ファイルとして保存し，その XML ファイルの表示方式を以下のように変更することができる．

1）　注文受付としてブラウザ表示

HTML 用の XSL と XSLT により HTML に変換され，追加情報を付加した形

でブラウザ上に表示する．

2） 注文伝票作成

PDF（電子ファイル）用のフォーム情報 XSL と XSLT により，PDF ファイルを作成する．

3） 注文伝票を FAX 送信

FAX 用のフォーム情報 XSL と XML ファイルを使用して FAX サーバから送信する．

4） 受注処理

固定長ファイル用 XSL と XSLT を使用してファイルを作成し，ホストシステムに送信，データベースを更新する．

このようにして，WWW からオーダーから即座に注文伝票を作成としたり，注文先に応じた形式で伝票出力したり，ホストシステムの受注処理と連携することが容易に可能となる．

上記のアプリケーション連携は DRB エンジンを実行基盤とし，DRB/PDF 作成アダプタや DRB/FAX 連携アダプタなどの部品を組み合わせることによって実装可能である．また，これに EDI 連携アダプタなどを用いれば多種多様なシステム間連携が可能である．

XML は W 3 C が開発したという経緯もあり，一般的には WWW 指向と考えられている．WWW ベースのアプリケーション統合を容易にするものであるが，アプリケーション統合におけるデータの正規化に有効であり，データ相互運用性を高める価値を持っている．

4. アプリケーション統合

4.1 EAI

EAI とは，企業全体における情報の流れをスムーズにし，より活性的なビジネス環境を構築することを目的としている．EAI はアプリケーション相互を連携させるためのミドルウェア製品を活用して，これまで拡散していた多様な業務システムやパッケージ製品を有機的に統合することを指しており，これまで蓄積されてきたレガシー・アプリケーション，パッケージソフトおよび新規開発アプリケーションを統合する有効な方法と考えられている．

複数のアプリケーションを統合する手法として，次の 3 タイプがあげられる．

1） 複数ステップによるプロセス型

2） データ一貫性型

3） 複合アプリケーション型[7]

EAI はこの複合アプリケーション型統合を実現するための手段であり，以下のことがらに主眼をおいた方法論である．

1） 統合対象に手を加えない，もしくは，1）変更を最小限に抑える

2） リスクの最小化と迅速化

4.2 複数ステップによるプロセス型統合

複数ステップによるプロセス型統合とは，明確に定義し順序付けられたステップを

実行することによってビジネス・プロセスを自動化し，各アプリケーションを統合することをいう．また，送受信側入出力データやメッセージのシンタックスおよびセマンティックスを一致させなければならない．例えば，システム 1 の出力データは，シンタックスやセマンティックスを変換し，受信側システム 2 の入力データとして使用できるようにしなければならない．

プロセス型統合では，以下の 3 項目が重要である．

1) 関係するシステム間の関係が密接である
2) アプリケーションレベルで統合され，トランザクションの完全性を保証する
3) 関連するシステムがプロセス中心で設計されている

4.3 データー貫性型統合

もう一つの「統合」パターンとしてデータ一貫性型統合がある．複数のアプリケーションから発生するデータは，完結しており，一貫性があり，タイムリーであり，適切であることをデータ型統合では要求される．つまり，データ一貫性型統合はセマンティックスの観点から単一の集合体にまとめられ，データ中心にアプリケーション統合を実現する．

以下の 2 項目が特徴である．

1) 疎結合型のシステム
2) データベースレベルで統合される

データ一貫性型統合を達成するためには，複数の独立したシステムに格納されている冗長なデータを一貫性があるように見方を統一することが重要である．しかし，アプリケーションを再構築し，冗長性を取り除くことは現実的には困難な作業である．複数システムのデータを統合したのちの冗長で重複したデータを一貫性のある集合体にまとめる必要がある．

4.4 複合アプリケーション型統合

企業システムの E ビジネス対応においても，この新しいビジネス・モデルを提供するための仕組みを既存の基幹システムに統合する必要が生じる．また，複数の製品の組み合わせが必須であり，従来の統合対象であるデータ，アプリケーション，WWW システムやグループウェアなどとの幅広い統合とリアルタイム連携が必要となってきている．

インターネットによる電子商取引など，標準化のスピードが速く，新技術が矢継ぎ早に投入される環境においては，取引を支えるビジネス・ルールが変化するスピードも速い．システム構築も短期間・低コストであることが求められる．この局面において EAI が提供するゴールは，「対象となるビジネス・プロセス間のシームレスな統合」および「インタフェースの統一を目的とするシステム間のデータ共有もしくは交換」を実現することであり，以下の技術がこれを実装する[8]．

- エンタープライズレベルでの共通セマンティックスの確立によるプロセス間のシームレス（透過的）なデータフローの提供．ビジネス・プロセスの統合およびプロセス間の相互影響に対する即応性の確保．
- XAI（XML Application Integration）として規定されるような XML で定義されるメタモデルの活用による多対多統合のコスト削減．

- イントラネットにおける MOM（Message Oriented Middleware），企業間においては WWW ベースのプロトコルを基本とする，他システムに対する明確なデータ伝達経路の確立.
- 柔軟でダイナミックに構成可能なアプリケーション間のデータ交換インタフェースの提供.

EAI は，このような技術の組み合わせによって，システム間のメッセージ交換やメッセージルーティングの機能を統合管理するミドルウェアであり，アプリケーションがロジックを大幅に変更することなく他のアプリケーションと連携する事を可能にする.

DRB は，今後この EAI に要求されるニーズに対応すべくメッセージレベルでの交換機能を強化する方向にある．近日中に MQ（Message Queueing）との連携機能および WWW プロトコルによる DRB 基盤間の通信を実装予定である．また，データ交換，変換技術の核として XML への対応強化と，XML を資産として有効活用するためのメッセージウェアハウスの提供を予定している.

5. お わ り に

今後のビジネスモデルもまた，現在のこのスピードを緩めることなく変化しつづけるであろうと予測される．アプリケーションシステムが新旧のビジネスプロセスに対応し続けるためには，その多面性をますます広げていかねばならない．複雑な機能の取り合わせをシステム化するためにはコンポーネント技術が欠かせない．DRB はバックグラウンド処理におけるアプリケーションやパッケージの機能全体をコンポーネントとして捉え，これらを組み合わせてビジネスプロセスを形成するための実行環境である.

DRB の目的とするところは，企業がビジネスを遂行する上で必要なプロセスや情報・データを分散したシステム間において連携させることにある．すなわち，一枚岩のシステムを最初から構築するのではなく，DRB の持つシステム間のメッセージ交換やメッセージルーティング機能を活用して，アプリケーション統合を実現させることで，業務プロセス開発の迅速化と効率化を実現することである.

参考文献
［1］ 最新エージェントテクノロジ, 長尾確編 bit Vol. 31 No. 2〜No. 8
［2］ FIPA Specification, http : //www/fipa/org/spec/index.htm
［3］ MASIF-RTF Results, http : //www.omg.org
［4］ FIPA Specification Part 11‘ Agent Mobility ’http : //www/fipa/org/spec/fipa 8 a 27. doc
［5］ 日本ユニシス情報技術研究会， WWW における XML の活用， 東京電機大学出版局 1999 年 4 月
［6］ Scalable Vector Graphics（SVG）1.0 Specification, http : //www.w 3.org
［7］「ビジネス問題に即した統合テクノロジ」, ガートナーレポート, JITS : AIMS−99−96, 1999 年 8 月 31 日付
［8］ Enterprise Application Integration Tutorial, JP Morgenthal XMLSolutions, Inc. OMG EAI Workshop http : //www.omg.org

執筆者紹介　丸　岡　茂　敏（Shigetoshi Maruoka）
1961 年生．1984 年学習院大学文学部哲学科卒業．同年日本ユニシス入社．金融関連システム開発業務に従事した後，汎用パッケージ製品の開発を担当．現在，W 2 K テクノロジーセンター ES イネーブラ開発室に所属．

小　枝　康　二（Yasuji Koeda）
1963 年生．1988 年日本大学法学部政治経済学科卒業．同年日本ユニシス(株)入社．B 1000・V シリーズから A シリーズへの移行業務に従事．現在，W 2 K テクノロジーセンター ES イネーブラ開発室に所属．