XMLによるスケジューラとMES連携の実現

The Actualization of Collaboration Between Scheduler and MES utilizing XML

阿部　秀幹　Abe Hideki
山森　慎也　Yamamori Shinya
真嶋　　剣　Majima Ken

製造業では，顧客ニーズの多様化や製品ライフサイクルの短期化により，多様な製品を短期間で製造することが課題となっており，製造現場の状況変化に即応できるシステムへの期待が高まってきている。このニーズに対応するため，㈱日立東日本ソリューションズの生産計画スケジューラである「SynPLA」と米国USDATA社のMESである「Xfactory」を連携したソリューションビジネスを展開中である。その一環として，中間データにXMLを利用してマスタデータとトランザクションデータを共有することで，SynPLAとXfactoryとの連携を実現した。両システム間のデータ変換およびデータ配信には，システム間のデータマッピング，データ配信管理等の機能を持つMicrosoft BizTalk Serverを適用し，従来よりも少ない工数でのシステム連携を可能とした。

1　はじめに

生産計画スケジューラ「SynPLA」は2001年のリリース以来，基幹系システムであるERP（Enterprise Resource Planning）や計画系システムであるSCP（Supply Chain Planning）との連携を中心にビジネスを展開してきた[1)]。

また，製造業での最近の動向として，スケジューラによって立案する生産計画と，MES（Manufacturing Execution System：製造実施システム）[2)]によって収集した製造実績を連携させ，刻々と変化する製造現場の状況に即座に対応できるシステムのニーズが高まってきている。

このような背景から，SynPLAのソリューション領域の拡大を図るため，実施系システムであるMESとの連携によるソリューションビジネスの展開を推進している。その一環として，米国USDATA社のMESである「Xfactory（国内総代理店㈱ダイセック社）」とSynPLAとの連携を実現した。

SynPLA－Xfactory連携では，中間データにXML（eXtensible Markup Language）[3)]を利用することでマスタデータとトランザクションデータを共有した。両システム間のデータ変換およびデータ配信には，Microsoft BizTalk Server[4)]を適用した。本報告では，この実現方式と有効性について述べる。

2　システム構成

スケジューラ・MES連携の目的は，MESがリアルタイムに収集した製造実績や設備状況を，スケジューラが立案する生産計画に反映し，製造現場への的確な指示を行うことにより生産効率を向上させること[2)]である。

上記の目的を実現するため，SynPLA－Xfactory連携のシステム構成は，図１に示す通りとした。

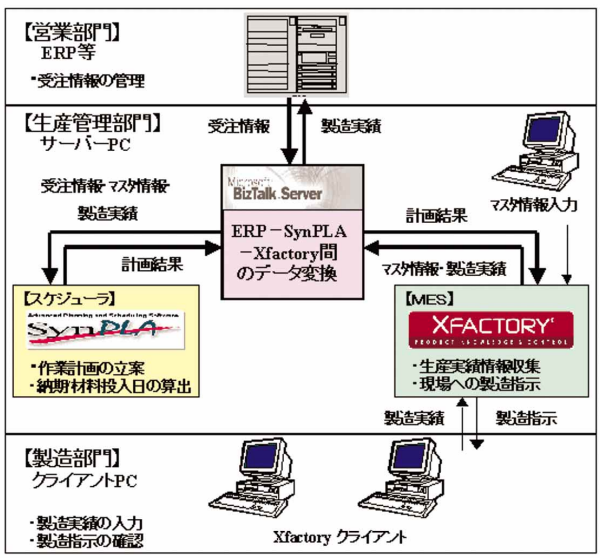


図１　SynPLA-Xfactory 連携のシステム構成

図１で，業務処理の流れは以下①～⑥の通りである。

① サーバPC上のXfactoryから工程情報，設備情報等のマスタ情報を登録する。

② Xfactoryから入力したマスタ情報と，上位システム（ERP等）から送信される受注情報を元に，サーバPC上のSynPLAで作業計画を立案する。

③ ＳｙｎＰＬＡの計画結果はＸｆａｃｔｏｒｙに送られ，Xfactoryから現場への製造指示を行う。

④ 製造部門のクライアントPCでXfactoryからの製造指示を確認し，製造作業を行う。

⑤ 製造作業の実績をクライアントPCから入力し，Xfactoryに製造実績を収集する。

⑥ SynPLAではXfactoryからの製造実績を反映し，現場の実態を反映した作業計画を立案する。

以下，③～⑥を繰り返し，日々の運用を行う。

3　実現方式

3.1　技術課題

図１で示すような業務処理を実現するためには，SynPLAとXfactoryが，共通のマスタデータと同期のとれたトランザクションデータを使用することが前提である。しかし，SynPLAとXfactoryはデータ構造が異なるため，両システム間でデータ交換をする場合，データ構造の変換が必要となる。この場合，従来の方法では連携するシステム毎に個別のデータ変換用プログラムを作成しなくてはいけない。また，各システム間のデータ配信を考慮すると，連携するシステムが増加した場合，人手によるデータ配信では管理が煩雑となりデータの不整合が発生する可能性がある。このためデータ配信の自動化も必要である。

この技術課題を解決するために，SynPLA－Xfactory間で送受信する中間データをXMLとし，データ構造の変換処理およびデータ配信にMicrosoft社のBizTalk Serverを適用した。

BizTalk Serverの適用により，データ構造の変換処理およびデータ配信処理を統合できる[4)]ため，システム間で個別のデータ変換処理プログラムを開発する必要はない。また，BizTalk Serverはデータフローをビジュアルに定義できるツール群を備えている[4)]ため，XMLに関する専門知識がなくても容易にデータフローを構築できる。

3.2　データフロー

図２に，SynPLA－Xfactory連携におけるデータフローを示す。図２をもとに，XfactoryからSynPLAへ製造実績を送信する場合のデータフローを説明する。なお，SynPLA－Xfactory連携では，Xfactory側がデータベース管理システムをMicrosoft社のSQL Server 2000に限定しているため，SQL Server 2000を標準のデータベース管理システムとする。

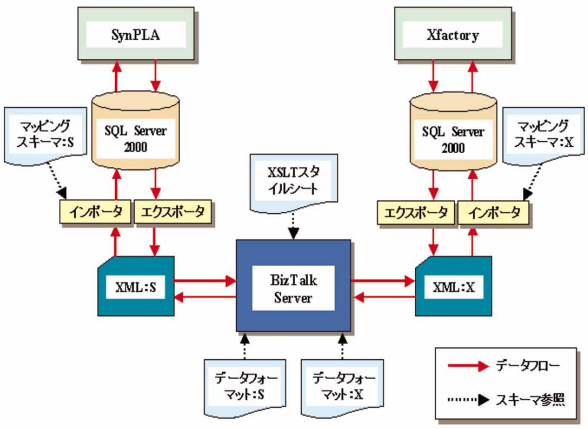


図２　データフロー

XfactoryからSynPLAへ製造実績を送信する場合，Xfactoryの「エクスポータ」を起動する。「エクスポータ」は，データベース上の“実績情報テーブル”のレコードをXML形式に変換し，テキストファイルである「XML：X」に出力する。「XML：X」が出力されると，BizTalk Serverが，「データフォーマット：X」を参照して「XML：X」を読み込む。BizTalk Server は，「XML：X」を読み込むと，まず「XSLT（eXtensible Stylesheet Language Transformations）スタイルシート[2)]」の定義を参照して「XML：X」を「XML：S」に変換する。次に，「XML：S」を所定の場所に配信し，「データフォーマット：S」を参照して出力する。「XML：S」が出力されると「インポータ」が起動する。「インポータ」は「XML：S」を読みこみ，「マッピングスキーマ：S」を参照してXML形式のデータをレコード形式に変換し，SynPLAの“実績テーブル”に挿入する。

3.3　インポータ／エクスポータの構築方法

3.3.1　インポータ

SQL Server 2000はXMLに対応するための拡張機能を備えている[5)]。これを利用することで，データベースにXML形式のデータをインポートする処理を容易に構築

できる。SQL Server 2000の拡張機能を利用し，データベースへXML形式のデータをインポートする手法として，次の二つについて，SynPLA－Xfactory連携での業務処理の特性を考慮して適用を検討した。

（１）BulkLoadオブジェクトによる一括挿入[6)]

（２）OPENXMLを使用したストアドプロシージャによる逐次処理[5)7)]

SynPLA－Xfactory連携では，受注情報，計画結果，製造実績等のトランザクションデータをインポートする場合は（１），工程情報，設備情報等のマスタデータをインポートする場合は（２）を用いることにした。

トランザクションデータは，計画立案の都度，数千～数万件をインポートする。したがって，データ量が増加してもメモリ消費量が増加せず，性能面でも優れた（１）の手法を用いた。

マスタデータは，システムのメンテナンス時に部分的に追加，更新または削除を行う。したがって，XML形式のデータに対して条件検索が可能で，検索結果によって追加，更新または削除が行える（２）の手法を用いた。

以下にそれぞれの手法の特長と，SynPLA－Xfactory連携への適用方式について述べる。

（１） BulkLoadオブジェクトによる一括挿入

XML文書内のデータを全件挿入する場合に使用する。インポート処理はレコード単位で行われるため，大量データを挿入する場合も，メモリ消費量の増大による性能劣化は発生しない。データベースとXML文書ではデータの表現形式が異なるため，XML文書の各要素または属性と，データベースのテーブルおよびフィールドとの対応関係をマッピングスキーマで定義する必要がある。

ｉ）マッピングスキーマの作成例

図３に，マッピングスキーマの作成例として，SynPLAの“実績テーブル”の各フィールドにXML文書の属性を対応付ける場合のマッピングスキーマを示す。

図３の作成例について説明する。まず，SynPLAの“実績テーブル”の名称である「OprRslt」を「name」属性の値とする親要素を定義した。親要素の「sql:relation」属性には「OprRslt」を設定し，親要素をSynPLAの“実績テーブル”と対応付けた。次に，親要素に対して，“実績テーブル”の各フィールドに対応する七つの子要素を定義した。各子要素の「sql:field」属性には対応するフィールド名を設定した。

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
        xmlns:sql = "urn:schemas-microsoft-com:mapping-schema">
<xsd:element name = "OprRslt"  sql:relation = "OprRslt"
        type = "OprRsltType"/>
<xsd:complexType name="OprRsltType">
  <xsd:sequence>
    <xsd:element name = "lotCode"  sql:field = "lotCode"
        type = "xsd:string"/>
    <xsd:element name = "oprCode"  sql:field = "oprCode"
        type = "xsd:string"/>
    <xsd:element name = "divNo"  sql:field = "divNo"
        type = "xsd:string"/>
    <xsd:element name = "rscCode"  sql:field = "rscCode"
        type = "xsd:string"/>
    <xsd:element name = "rsltDate"  sql:field = "rsltDate"
        type="xsd:dateTime"/>
    <xsd:element name = "qty"  sql:field = "qty"
        type = "xsd:integer"/>
    <xsd:element name = "status"  sql:field = "status"
        type = "xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

図３　マッピングスキーマの作成例

ii）BulkLoadオブジェクトの利用方法

BulkLoadオブジェクトを利用してXML文書の一括挿入を行う場合は，次の三つの入力パラメータを指定し，Executeメソッドを呼び出す[6)]。

・XML文書へのパス

・XMLスキーマへのパス

・OLE DB 接続文字列

図４に，BulkLoadオブジェクトを利用する上で，最低限必要な処理をMicrosoft Visual Basic（以下VB）で記述した例を示す。

```
'XML 一括ロードオブジェクトのインスタンスの作成
Dim BulkLoad As New SQLXMLBulkLoad3

'OLE DB 接続文字列の設定
BulkLoadObj.ConnectionString  =  "OLE DB 接続文字列"

' 一括ロードの実行
BulkLoadObj.Execute  "XML スキーマへのパス",  "XML 文書へのパス"
```

図４　BulkLoad オブジェクトの使用例（VB）

図４に示した通り，BulkLoadオブジェクトを利用することにより，マッピングスキーマの作成と少量のコーディングでXML文書内のデータをデータベースへ一括挿入することを実現した。図３のマッピングスキーマによりSynPLAの実績テーブルへXML形式の製造実績を挿入した例を図５に示す。

●挿入するXML文書

```
<?xml version="1.0" encoding="Shift_JIS" ?>
<SynPLA>
  <OprRslt><lotCode>A</lotCode><oprCode>a</oprCode>
  <divNo>0</divNo><rscCode>R1</rscCode>
  <rsltDate>2003/1/1</rsltDate><qty>100</qty><status></status>
</OprRslt>
</SynPLA >
```

●挿入先のテーブル

テーブル名：OprRslt

#	lotCode	oprCode	divNo	rscCode	rsltDate	qty	status
1	A	a	0	R1	2003/1/1	100	

図５　BulkLoad オブジェクトによる一括挿入の実行例

（２）OPENXMLを使用したストアドプロシージャによる逐次処理

XML文書内のデータに条件検索を行い，結果に応じてデータベースに対して挿入，更新または削除を行う場合に使用する。

OPENXMLを使用することで，XML文書をメモリに展開し，テーブルまたはビューに見立ててアクセスすることができる[5)7)]。したがって，通常のSQL文の作成と同じ要領で，XML文書よりデータを抽出し，既存テーブルに挿入，更新または削除を行う処理を記述することができる。図６に，OPENXMLを使用したストアドプロシージャの作成例を示す。

```
CREATE PROCEDURE UpSertOpr @xmlDoc TEXT AS
DECLARE @iTree INTEGER
EXEC sp_xml_preparedocument @iTree OUTPUT, @xmlDoc
DELETE FROM Prt
    WHERE Prt.prtCode IN (
        SELECT prtCode FROM
            OPENXML(@iTree, '/SynPLA/Prt', 2)
                WITH (prtCode nCHAR(50))
    )
EXEC sp_xml_removedocument @iTree
GO
```

図６　OPENXML を使用したストアドプロシージャの例

図６に示したストアドプロシージャの詳細は以下の通りである。

XML文書は，「sp_xml_preparedocument」によりエッジテーブルと呼ばれるメモリ上の架空の行セットに展開される。エッジテーブルには，エッジテーブルハンドル「@iTree」を通してアクセスする。OPENXML文は，SLECT文のFROM句に続いて記述する。OPENXML文には，次の３つのパラメータを指定する。

ｉ）エッジテーブルハンドルを格納している変数

ii）抽出するノードを示すXpath式[3)]

iii）属性または要素の優先指定（属性：１，要素：２）

WITH句には，行セットから抜き出す列名とデータ型を指定する。

3.3.2　エクスポータ

データベース上のレコードをXML形式でエクスポートするには，ADO（ActiveX Data Objects）のXMLサポート機能[8)]を利用する。ADOはバージョン2.5より，通常のSQLクエリを使用してデータベースからデータを取得し，そのデータを，ADO レコードセットの adPersistXML オプションを使用して XML 形式で保存することが可能となった[3)]。図7に，ADOを利用してデータベース上のデータをXML文書として取得する処理をVBで記述した例を示す。

```
'ADOレコードセットオブジェクトのインスタンスの作成
Dim rs As New ADODB.Recordset

' データベースに接続し SQL クエリによりデータを取得
rs.Open  "SELECT * FROM Sample",  "OLE DB 接続文字列"

'adPersistXML オプションを指定し XML 文書として保存
rs.Save  "保存するファイルのパス",  adPersistXML
```

図７　ADO を利用した XML 文書の取得処理の実装例

図７で示した処理によって，Xfactoryの“実績情報テーブル”のレコードをXML形式で取得した実行例を図８に示す。

●取得対象のテーブル

テーブル名：viewXWipStep

#	WIPID	StepID	DivNo	Equipment ID	EndTime	TxNext Cnt	OprRslt -Status
1	A	a	0	R1	2003/1/1	100	1

●取得されたXML文書

```
<xml xmlns:s='uuid:BDC6E3F0-6DA3-11d1-A2A3-00AA00C14882'
        xmlns:dt='uuid:C2F41010-65B3-11d1-A29F-00AA00C14882'
        xmlns:rs='urn:schemas-microsoft-com:rowset'
        xmlns:z='#RowsetSchema'>
<rs:data>
  <z:row WIPID='A' StepID='a' DivNo='0' EquipmentID='1'
         EndTime='2003/1/1' TxNextCnt='100' OprRslt-Status='1' />
</rs:data>
</xml>
```

図８　ADO を利用した XML 文書の取得処理の実行例

3.4　BizTalk Serverによるデータ変換・配信機能の構築

BizTalk Serverによるデータ変換・配信機能の構築

は，次の手順で行う。

（１）BizTalkエディタ[4)]によるデータフォーマットの登録

（２）BizTalkマッパー[4)]によるデータ構造変換定義の作成

（３）BizTalkメッセージマネージャ[4)]によるデータ配信の経路設定

以下にそれぞれの詳細を述べる。

3.4.1　BizTalkエディタによるデータフォーマット登録

BizTalk Serverで交換されるデータは，すべて構造化されたテキストメッセージであり，そのフォーマットは，XMLスキーマ言語の一つであるXDR（XML-Data Reduced）で記述する必要がある。

BizTalkエディタは，BizTalk Serverで取り扱うデータのフォーマットをXDRにより作成し，登録するためのツールである。BizTalkエディタのGUIを利用することで，XDRに関する専門知識がなくても容易にデータフォーマットを作成することができる（図９）。

特に，他のＸＭＬスキーマ言語であるＤＴＤ（Document Type Definition）[3)]またはXML Schema[3)]で記述されたXMLスキーマは，BizTalkエディタにインポートするだけで自動的にXDRによるデータフォーマットに変換される。したがって，SynPLAおよびXfactoryのデータフォーマットは，XML Schemaで記述したマッピングスキーマ（図３）をBizTalkエディタにインポートするだけで登録できた。

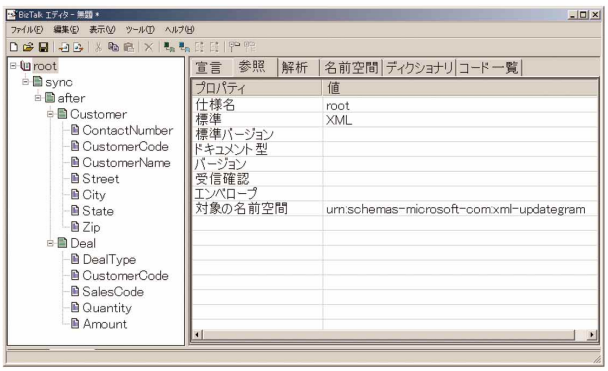


図９　BizTalk エディタの操作画面

3.4.2　BizTalkマッパーによるデータ構造変換定義の作成

データ構造の変換定義は，BizTalkマッパーを用いて作成した。データ構造の変換処理は，変換定義であるXSLTスタイルシートにもとづいて行われる。XSLTスタイルシートを手作業で作成するにはXSLTに関する専門知識が必要であり，変換が複雑な場合は工数の超過や不良の作り込みを招く恐れがあった。しかし，BizTalkマッパーを利用することで，正確なXSLTスタイルシートを容易に作成することができた。

BizTalkマッパーでは，登録されているデータフォーマットから送信元データフォーマットと送信先データフォーマットを表示させ，グラフィカルにレコードやフィールドを結び付けることによってXSLTスタイルシートを作成する。対応関係は１対１だけでなく，１対多や多対１も扱うことができる。また，対応関係にデータ処理を加えることもできる。

図10に，Xfactoryの“実績情報テーブル”のデータフォーマットとSynPLAの“実績テーブル”のデータフォーマットを対応付けた操作画面を示す。画面左側に表示したXfactoryの“実績情報テーブル”の各フィールドと，画面右側に表示したSynPLAの“実績テーブル”の各フィールドを線で結ぶイメージで対応付ける。

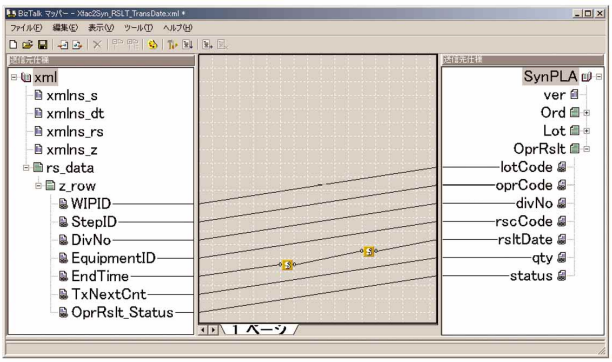


図10　BizTalk マッパーの操作画面

3.4.3　BizTalkメッセージマネージャによるデータ配信経路の設定

データの配信経路の設定は，BizTalkメッセージマネージャで行った。図１に示したシステム構成に基いて，メッセージを送受信する「組織」としてSynPLA，

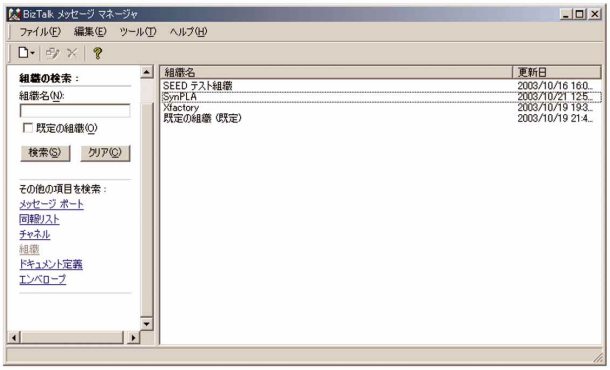


図11　BizTalk メッセージマネージャの操作画面

Xfactoryおよびホストを登録した。データの配信経路は送信先「組織」と受信先「組織」を画面上で選択し，指定することで設定した（図11）。

4　評価

上記実現方式に従いSynPLA-Xfactory連携を実現し，平成14年10月～平成15年２月にかけて，航空機部品メーカーへの導入を行った。

導入の際に検討したシステム性能，開発工数，品質，保守性，拡張性および導入効果の点から，SynPLA-Xfactory連携の有効性を評価する。

4.1　システム性能

SynPLA-Xfactory連携の導入を行う前に，性能設計の基礎データとして，SynPLA-Xfactory間のデータ転送時間を以下の条件で測定した。

OS：Windows 2000 Server SP4
CPU：Pentium4 2.4 GHz
メモリ：512MB
測定データ：1レコードのフィールド構成
（文字列型×３，倍精度浮動小数点型×３，
日付時刻型×３，整数型×１）
測定結果を表１に示す。

表１　SynPLA-Xfactory 間データ転送時間

レコード件数	データ転送時間（秒）
100	1.60
1,000	4.21
10,000	53.75

本システムを導入するメーカーでは，データ転送は１回／日，想定されるデータ件数は最大6,000件である。表１の結果から，6,000件での転送時間については運用上問題ないと判断し，本システムの開発に着手した。

最終的には，システム導入後に本番環境で実測し，その時点の最大データ件数（4,349件）で実測した結果8.16秒であった。

この結果から，本番運用においてデータ件数の増加，転送回数の増加が発生した場合でも，実用に耐えられる性能であると考えられる。

4.2　開発工数

4.2.1　開発実績

本システム連携部分の開発は，BizTalk Serverを使用したシステム構築の経験が無い担当者が行った。開発工数の実績について表２に示す。

表２　SynPLA-Xfactory 連携開発工数

作　　業	工数（人日）
インポータ／エクスポータ作成	4.0
スキーマ定義	1.0
マッピング	1.0
配信経路設定	2.0
テスト	2.0
合　計	10.0

インポータ／エクスポータの作成方法はBizTalk Server付属のオンラインドキュメントにVBを使用した作成例が掲載されており，また，スキーマ定義・マッピング・配信経路を設定する各ツールの詳細な使用方法もオンラインドキュメントや参考書籍に掲載されているため，これらを参考にすることで，BizTalk Serverの使用経験が無くても連携部分の開発が可能である。

4.2.2　専用インターフェース構築時と比較した開発工数の評価

BizTalk Serverを利用した連携システムのインターフェース（以下，IF）構築と，VBやMicrosoft Visual C++でのコーディングにより専用IFを作成した場合の工数とを比較する。

これまでの開発経験から，１つのIFの作成には開発～テスト工程で（データ関連の複雑さにもよるが）0.5人月の工数が必要である。SynPLA-Xfactory連携システムは「SynPLA」・「Xfactory」・「上位システム（ERP等）」の３システムの連携であり，３つのIFが必要である。したがって専用IFの開発工数は1.5人月になる。BizTalk Serverを利用した場合の連携システムの開発工数は4.2.1節で述べたように0.5人月（10人日）であることから，開発工数全体としては60%以上削減された。

各開発工程について評価すると，表２に示した作業は専用IF開発のプログラミング工程から組合せテスト工程までに相当する。本方式では，これらの開発工程の作業を，SQL Server 2000のXMLに対応する拡張機能及びBizTalk Serverの各種ツールにより効率化することがで

きた。

4.3　品質

本方式でプログラミングを必要とする部分は，インポータ・エクスポータ部分のみであり，図４及び図７に示すとおり数十Stepの開発で済む。このようにプログラミングを必要とする部分が少ないため，専用IFを作成する場合に比べ不良を作り込む可能性は低く，品質確保は容易である。

4.4　保守性

品質の評価で述べたように，プログラミングを必要とする部分が少ないため，システムの変更が発生した場合でも変更箇所を特定しやすい。

また，インポータでは実処理部分をストアドプロシージャとしているため，変更してもコンパイルしなおす必要は無い。

さらに，マッピングスキーマ，XSLTスタイルシート，データフォーマット等の定義はBizTalk Serverの各種ツールでビジュアルに設定できるため，変更が容易である。したがって保守性は高いといえる。

4.5　拡張性

本システムは，上位システムであるホストからの受注情報を受取るため，ホストとも連携している。

ホストとの連携も，SynPLA-Xfactory連携と同様にBizTalk Serverのツールを利用し次の手順で構築した。

（１）BizTalkエディタによるデータフォーマット登録
（２）BizTalkマッパーによるデータ構造の変換定義作成
（３）BizTalkメッセージマネージャによるデータ配信経路設定

表２の開発工数は，これらのホストとの連携作業も含めた値である。したがって，SynPLA-Xfactory以外のシステムと連携する場合も，短期間で連携が実現できることから，本方式の拡張性は高いと評価できる。

4.6　導入効果

本システムの導入により，顧客からは以下のような点が評価されている。

・製造部門からの実績値を基に標準時間や設備能力等のマスタ情報を設定するため，マスタ情報の精度が向上した。また，スケジューラとMESが同期のとれたマスタ情報を共用するため生産計画の精度が向上した。
・製造部門の進捗状況を生産管理部門がリアルタイムに把握できるようになり，最新の実績データを反映した生産計画の見直しおよび製造部門に対する的確な製造指示が可能となった。

また，現時点で本システムは部品加工部門に使用されているが，顧客の意向としては，今後は本システムを組立部門にも展開していきたいとの考えである。

5　おわりに

本報告ではXMLによるスケジューラ・MES連携の実現方式およびその有効性について述べた。BizTalk Serverにより両システムの連携を容易に実現することが可能であり，導入効果からもわかるように，両システムの連携が生産管理上の問題に対するソリューションとして有効であることが確認できた。また，既存のパッケージシステムを組合せることにより，一からのシステム開発に比べ，開発工数も大幅に削減できたと考えられる。

このような各業務システムの統合は，情報資産の共有化を推進し，企業活動に新たな付加価値を生み出すものとして注目されている。

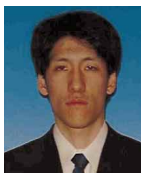
今後は，本技術をスケジューラとMESとの連携に限らず，他システム間の連携にも積極的に活用し，ソリューションの幅を広げていく所存である。

参考文献

１）捧　保浩，他：SCMを支える意思決定支援システムSYNAPSEsuite，日立TO技報第６号，pp. 16－22，2000
２）中村　実，他：MES入門，工業調査会，2000
３）竹中　祐，他：XMLマスターベーシック，翔泳社，2002
４）Stephen Mohr，他：プロフェッショナル BizTalk，インプレス，2001
５）佐藤　親一：XML on SQL Server 2000，オーム社，2001
６）Karthik Ravindran：XML一括ロードの概要，http://www.microsoft.com/japan/msdn/sqlserver/sql2000/sqlxml_bulkloadover.asp
７）Dejan Sunderic，他：SQL Server 2000 ストアドプロシージャプログラミング，翔泳社，2001
８）MSDN Online BizTalk Server：Microsoft BizTalk Server 2002 とのデータベース統合，http://www.microsoft.com/japan/msdn/biztalk/biztalk2002/bts_dbintegration.asp

阿部　秀幹　　1992年入社
産業システム部
製造業対応ソリューションの提供
hidekabe@hitachi-to.co.jp

山森　慎也　　2001年入社
研究開発部
生産計画システムの研究・開発
yamamori@hitachi-to.co.jp



真嶋　　剣　　2001年入社
産業システム部
製造業対応ソリューションの提供
majima@hitachi-to.co.jp