

# 金融メッセージの XML 化の流れ

## SWIFT コミュニティにもたらす XML のメリット

### 1. エグゼクティブサマリー

本ペーパーは拡張可能マークアップランゲージ:XML の背景を解説し、XML の理解となぜ SWIFT が XML を選んだかということに対する理解の必要性にこたえるものである。

技術の進歩に伴い、業務要件や対象業務が拡大し複雑となってきたため、SWIFT は 1998 年に標準化に対するアプローチを拡張する必要があることを認識し、新しいビジネスエリアには、従来の MT に利用されているシンタックスに改革の必要があることを認識した。新しい分野への適用について、既存 MT シンタックスの大幅な変更という方法はコミュニティ全体に大きな影響を与えるため、とり上げられた。むしろ既存の MT を補って、異なるシンタックスを必要に応じて採用することが良いと考えられた。その方法とは XML であり、人間の目でも、システムへの取り込みも可能な特徴を生かした汎用言語として取り組まれている。

更に 1999 年時点で既に多くの業界で XML がメジャーとなり、多くのソフトウェア産業でも商業的にサポートされつつあった。

一方で、XML の導入は多くのユーザーの IT 部門にとって大幅な変更を伴うものであり、特にコストカット要請の大きい金融機関では今まさに XML の採用がビジネス上判断されつつあるところである。主要な懸念材料は、XML メッセージの長さ、処理パフォーマンス、ネットワーク上の処理速度、勘定系との連動などが上げられている。

これらの問題点を考えるにあたって、SWIFT は XML の導入経験をもついくつかの金融機関やソフトウェア会社とも相談をしたところによると、その全てが XML は内部システムの再構築の際に XML を技術基盤の一つとして採用したということである。確かに、組織内の複数の勘定系 (Legacy) システムをより効率的に繋ぎ、自動化を進めるための共通 Format としては有効であり、従ってそれら金融機関同士を繋ぐメッセージのスタンダードとしては有効であることは、自然な流れである。しかしながら、グローバルに金融機関の環境を見ると、その動きには温度差があり、XML の採用に問題なしとしない。

一方で、XML 対応をしているアプリケーションはスタンダードの保守にかかわり、所謂 “Injection” (---差し替えるという意味) という方法で新しいスタンダードの適用に関する開発・導入コストを大幅に押さえることができる。

### 2. XML の歴史

コンピュータシステムの胎動期である 60 年代ではコンピュータプログラムは特定のデータ形式を必要としており、アセンブラやコボルによる開発が主流であった。

その後 PC の普及もあり、書類の印刷という分野から、ハードウェアに依存しない方法が模索された。より複雑な印刷データを処理し且つハードウェアに依存しないですむために、汎用言語の開発が望まれた。そのために IBM や LinoType が中心となって設定したのが SGML (Standardised General Mark-up Language-汎用標準マークアップ言語) ISO8879:1985 であり、米軍における全ての文書は国防総省によりこの標準が要求された。

一方で Internet の浸透に従い、SGML から派生した Web 用の言語が急速に広まった。(HTML-Hyper Text Mark-up Language) HTML は画面上の表現に優れた、ユーザーにとってわかりやすい言語であるが、構造的な点やより正確な表現には甘いところがあり、近來の情報量の爆発的な拡大に対しては、より厳密に情報を管理する方法が望まれている。(HTML では正確な検索が出来ない例として “mouse” と代表的な検索エンジンである Google で検索をすると 29 百万の結果がでてくる。)

ソフトウェア業界ではこの種の問題に対応すべく、SGML から派生した、XML(eXtensible Mark-up language)を提唱し、画面上の表現とデータのハンドリングを同時に可能とすることをめざした。



その初期には XML 文書という形で浸透したが、データを管理する上位言語として、強力であることが認識され、多くの業界でコンピュータ間の連携に XML の利用が急速に広がっており、現在ではソフトウェア産業全体がコアの技術として採用しているところとなっている。最新の XML ベースのアプリケーションでは、XML と Web 技術との融合により、データと再生ソフトウェアの配信等に利用されている。

### 3. SWIFT が XML に注力する理由

SWIFT のスタンダードに関しても、XML が出てきた経緯に比して、ある種同じような進化がなされてきている。

SWIFT が当初登場した時期には Telex の代替としての機能が要求されており、MT のフォーマットも Telex のそれに似たものであった。

その後金融産業はより複雑な構造を持ったメッセージを必要としたが、所謂 Legacy システムの範囲の中で、MT として進化した。

Field とメッセージ TAG で定義される MT は明解に取り交わされる情報を定義できたが、一方でより複雑な情報をハンドルするには限界が近づいてきた。特に証券系のように複雑な取引を対象とする分野に置いては、MT の利用が障害ともなったが、ISO15022 に対応した一連の MT において Syntax ベースでの対応を行った。

また、国連が定めた企業間のデータのやり取りの自動化を進めるための EDIFACT は SWIFT にも影響を与え、MT105,106,121 といった MT が定義された。

その後、Internet の影響もあり、企業間・企業・消費者間でのデータのやり取りは急速な進歩を遂げ、PKI によるセキュリティ・IP ネットワーク・XML といった技術をベースに多くの企業がシステムのデザインを確立している。

これに並行して、ミドルウェアの利用の台頭があり、このミドルウェアも XML の利用をベースに多くの企業で利用されている。

このようなニーズや背景に従いながら、SWIFT では、既存 MT を保持しながらも、新しい分野では XML ベースの開発を行っている。

## The syntax evolution in and around SWIFT

- The 1970s
  - Automating simple telex message types
- The 1980s
  - MT syntax for more message types
  - EDIFACT in other industries (limited adoption)
- The 1990s
  - Enhanced MT syntax for more complex needs (ISO 15022)
- 2000 and beyond
  - Widespread automation and middleware
  - Open standards like IP, PKI and XML
  - MT syntax complemented by XML



STC062004\_v.1\_ppt.

Slide 51



## XML の簡単な例

Syntax(MT)がデータ構造そのものを表象する一方、XML では、“タグ”により、データの中身を挟んで定義している。例: 下記ご参照

```
<Book>
  <Title>Illusions</Title>
  <Author>Richard Bach</Author>
</Book>
```

一方で XML は構造の仕様を表す言語でもあり、メッセージにおける Field の順番やその Field が必須・Option などの表現も可能になる。これにより、コンピュータシステムから見て極めて明確なデータを作ることができる。以下は簡単な XML スキーマ<sup>1</sup>の例である。

```
<xsd:element name="Book">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="Title" type=xsd:string"/>
      <xsd:element name="Author" type=xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

金融機関間でのデータのやり取りに関して、MT と XML の違いは、XML が追加的な機能を持つということである。MT ではユーザーハンドブックに記述されたルールに基づいてその利用が理解されたが、それには人間の理解によりプログラムをするという手順が要求される。

XML はこの種負荷の軽減の可能性を持っている。先ず、IP や PKI と同じくオープンな業界標準であり、W3C(World Wide Web コンソーシアム)が定義している。XML はそれ自身は XML ファミリーと呼ばれるものの一部をなすものであり、画面上の表現をする XSL(スタイルシート)や複数の書類をリンクするための XLL(リンキング・ランゲージ)、ドキュメントの構造の変更をサポートする XSLT(XSLトランスフォーメーション)などが含まれている。これらの構成要素は多くの場合商業的なパッケージツールに含まれており、XML によって表現された書類の処理を簡単にすることができる。こにより、既存のプログラムとの連動が容易になる。

更に、XML は XMI により UML(モデリング・ランゲージ)との連携により Web 上のアプリケーションに有効な言語となっている。

<sup>1</sup> SGML の仕様を定義する DTD(Document Type Definition)に代わり XML では仕様を定義する際に XML Schema という言語体系を利用する。Microsoft の BizTalk も XML の Schema に準拠する。



# What is XML?

## syntax & specification language

```
<Book>
  <Title>Illusions</Title>
  <Author>Richard Bach</Author>
</Book>
```

```
<element name='Book'>
  <complexType>
    <sequence>
      <element name='Title' type='string'>
      <element name='Author' type='string'>
    </sequence>
  </complexType>
</element>
```

- Derived from SGML/HTML technology
- Open standard (owned by W3C)
- Part of a larger family of languages
- Supported by commercial tools
- Basis for emerging technologies (web services, ...)



STC062004\_v.1\_ppt.

Slide 35

## 4. XML のメリット・デメリット

### メリット

XML はその定義により、メッセージ構造全体をを表す言語であるため、MT のように仕様書 (UHB) で仕様を定義するのとは根本的に異なる。XML には以下の機能がある。

- ツリー構造と“入れ子”の構造により、階層的にメッセージの構造が表示される。MT ではシーケンス・サブシーケンス・フィールド・サブフィールドといった形で表記されるものが、階層的に表示される。
- シーケンス(順番): MT のシーケンスに類似して XML でもシーケンスが存在する。
- 選択: 幾つかあるフィールドから特定のフィールドを選ぶときに使われる。MT で類似の例としては、論理的なルール (semantic rule) として、“Field A が存在する場合、Field B 又は Field C は存在してはならない。Field A が無いときは B 又は C のどちらかがないといけない”のように表記される。
- 反復・複数指定: その Field が Optional か必須かを示し、またその Field が反復しているか否かを示す。MT ではユーザーハンドブックで定義される。
- 明示的タグ: 特定のフィールドについてどこがスタートでどこがエンドであることを示す。MT の場合は“:32A:”のようにスタートのみタグがあり、サブフィールドのタグを含んでいないので、追加データの判定に困難を生じる。
- データタイプ: 所与のフィールドにどのような種類のデータが許容されるかを示す。XML ではどのようなデータが有効であるかを示す。例としては日付、特定の文字列、小数点など。MT の場合フィールドのフォーマット定義でこれらを表す。
- メタデータ(上位データ): 情報の詳細を表す。例としては、特定のコードリストをどの組織が所有しているかなど。MT ではユーザーハンドブックのレベルでしか記述できない。

### ソフトウェア業界のサポート

XML は急速にソフトウェア業界全体でサポートされており、XML ドキュメントを処理するための広い範囲の商用のツールが提供されている。これにより、画面上の表示や XML ドキュメントの Validation や XML ドキュメントからのデータの抽出(データ・パーシングと呼ばれる)が簡単に可能になっている。

以下は、IBM, Microsoft, Sun 他多数の企業が提供している、一例のツールである。

- DTD やスタイルシート作成ツール
- XML との変換ツール
- ドキュメント管理システムと、検索エンジン
- XML ドキュメント作成ツール



- e) XMLドキュメントのブラウザー
- f) XMLドキュメントからのデータ抽出ツール

既存アプリケーションへの自動的な“Injection”ツールも充実しており、開発負荷を押さえることが可能である。

更に、XLL や XSLT などの機能により、RDB やその他のデータ変換が容易に出来る。

開発言語や OS・ハードウェアに依存しない。

人間の目でわかるように表示ができると同時に機械による処理に有利な構造をもっている。

## Advantages of XML

- XML has many features to specify structured messages
- XML is supported by many (types of) commercial tools offered by many software companies
- XML has injection capabilities
- XML can be complemented by system integration tools
- XML is platform- & programming language neutral (portable technology)
- XML has interpretation possibilities for people and machines



STC062004\_v.1\_ppt.

Slide 52

### XML のデメリット

幾つかのデメリットと考えられる点が指摘されている。

#### XML の賞味期限という疑問

今後も技術が進化するために、XML が何時まで有効であるかという疑問が提示されている。

XML は確かに日常的に目に触れるわけではないが、Web の技術に深く浸透している。例としては、Yahoo のホームページにもリンクが貼られている。また、全てとは言わないまでも、数多くの商業ソフトに採用されている、又はそのツールが存在する。(例 MS Office, Adobe 等)次に業界全般で XML を利用する動きがある。(一例としては、自動車産業、旅行業界、厚生サービス、保険、通信など) その上、いくつかの金融機関が XML をグループ内の標準として利用する動きがある。

RosettaNet のようにハイテク業界の企業間データ通信に XML を採用したり、ERP との連携に利用するなど、深く浸透していることから判断すると、XML の将来が疑問になるよりは、今後も更に発展するものと思われる。

#### データのオーバーヘッド(追加的に必要なデータ)

XML は MT よりもデータの量が多いが、これがパフォーマンス上又は料金上不利にならないか？

確かに、より詳細までをカバーできるので、結果としてメッセージは比較的長くなる。その結果 SWIFT 上のインフラでコストやパフォーマンスに不利に働く可能性はある。しかしながら、処理能力全体や回線処理スピードの上昇により、対応が可能と考えられており、SWIFT のインフラに関してはこの部分の影響をみこんだ上で、以下の対応を用意している。

- a) メッセージ長の縮小: SWIFT の XML である SWIFT MXs ではフィールドの表記に略号を使って、且つ詳細を表現できるように配慮してある。
- b) 価格の変更: SWIFT のプライシングポリシー上 MXs が高くつかないようにしていく。



- c) パフォーマンス上の保証: ネットワークとその能力が見合うように、常に SLA に従った能力の向上を行っている。  
一方ユーザー側の影響としては、以下ことが考えられる。
- a) 通信上の処理能力(Throughput): XML は幾つかの金融機関では、グループ内のデータ通信上、基本の技術として採用されていることから、必ずしも処理能力(Throughput)の観点から大きな障害になっているとは考えにくい。更に XML メッセージを圧縮することにより、効率化が図れている。
  - b) システム上の処理能力(Performance): 長いメッセージを処理することは当然のことながら、その時間が長くなるが、ハードウェアの処理能力の向上は目覚しく、実務上の問題にはなりにくい。
  - c) データ保管: 多くのケースでは全量データの保管ではなく、(タグを除いた)実データを保管することで対応が可能と考えられる。更に圧縮技術を使うことにより効率的に保管が可能。
  - d) 変化への対応: XML が更に利用されることにより、周辺技術が進み全体の処理能力が高まるであろう。具体的には XML-Chip のような XML データの処理の最適化を行うようなハードウェアの開発も進んでいる。(ハードウェアコンパイラやアクセラレーターなど。)

#### 勘定系を始めとした既存システムへの影響

では、XML は金融界の既存のシステムにどのように影響をあたえるのであろうか？  
SWIFT の Syntax を金融機関内部でどのようにデータとして表現(画面・印刷イメージなど)するかについては、何ら必須のやり方があるわけではない。むしろ、どのようにして SWIFT の XML である MX と旧来の MT のデータの相互運用性を確保するかが重要である。つまり、MT と MX が共存することが重要である。

#### SWIFT としてのビジネス・ケース

XML により以下のように SWIFT にとって明確に有利である:—

- 1) XML はオブジェクト指向とモデリングに基づいた SWIFT の標準化の開発に適しており、新たな SWIFT のメッセージタイプに有効である。更に、モデリング作業から SML スキーマを自動的に生成することが可能である。
- 2) XML スキーマにより、SWIFT 上への XML メッセージの導入には特段のソフトウェア開発を伴う必要がなく、コスト面で有利である。
- 3) XML により、新しいサービスを提供することやコミュニティへの標準化の実現が容易になる。
- 4) “Injection”という方法により、SWIFT のインターフェースへの新しい XML ベースのスタンダードの導入が時間的・コスト的に容易になる。

XML なかりせば:—

標準化の収斂に関する取り組みや、ISO20022 など ISO の取り組みに困難が生じるほか、他の標準化団体である、UN/CEFACT、EAN/UCC、FIX、FpML、MDDL、RosettaNet などの共同作業が困難になる。標準化の収斂は SWIFT の鍵となる戦略であり、XML 抜きには語れない。



## Business case for SWIFT

- Easy fit with overall approach to standardise data objects
- Possibility to automate generation of XML schemas
- Injection of validation rules in messaging application
- Opportunities for new products and tools
- Injection possibilities in interface

Imagine the opposite: life without XML ...  
- SWIFT would not have driven ISO 20022  
- No basis for cooperation with RosettaNet, IFX, FIX, FpML, UN/CEFACT, MDDL, ...  
- No acceptable basis to start convergence work

**SWIFT couldn't and can't ignore XML**



STC062004\_v.1\_ppt.

Slide 38

### 5. SWIFT のユーザーコミュニティに対するビジネス・ケース

1997 年に Standard Green Paper, Standard White Paper として SWIFT は標準化に関しどのような問題があるかを報告した。MT syntax における MT 番号の枯渇やフィールドタグの不足、各 MT 間でのフィールドの不整合などが上げられたが、より本質的な問題として、データの流れ全体を通じた見方の欠落や、誤った解釈の可能性、仕様書の品質、データ対象の曖昧さ、定義と実際の内容の相互関係の曖昧さなどが報告された。

特に重要な点は MT syntax そのものとは離れて、背景となる業務のより良い理解ためのモデリングの活用や、データ項目の集中保管、全体の流れを見越した標準化、仕様の明確化などが必要であるという報告であった。その時点では XML の仕様は確定していなかった。(98 年 2 月が最初) 更に付け加えると、従来の方は実装に関する負荷も大きくテストの実施や品質管理に幾多の問題があることが指摘され、その結果全体としての STP の推進に困難があった。コミュニティはこのような困難を排除し、システムが処理しやすく、容易に実装“Inject”(流し込み又は差し替え)できる方法を望んだ。

その結果、1999 年に ER811-XML syntax により、メッセージの大幅な変更や再定義を行う場合 XML に基づくという方向性が定められた。

過去 5 年間に XML の重要性は高まってきており、多くの産業でグループ内(Intra-Organisation)を中心にデータ交換の基盤として定着していることは、実装された幾つかのケースや経験をつんだユーザー・パートナーなどからのフィードバックにより確認される。その主な利点は前述の通りであるが、概要としては:

#### 複雑な情報交換をサポートする機能

- XML は情報の構造化に優れたメカニズムを持っている。社内はもとより、顧客へのより良いサービスを提供するための情報活用を容易にする。
- タグの構文的な使用および仕様言語の先進的な機能など XML の明示的な性質により、不明確性が減少する。
- XML は、テクノロジー標準が XML ベースで構築されている(例: ウェブサービス、BPEL(ビジネスプロセスを表現する XML ベース言語)、RSS(ニュースヘッドラインなど、ウェブコンテンツを集約的に配信する XML ベース言語))ことから見て、将来的にも使い続けることができる技術である。これらのアプリケーションで使用される情報が XML で表現されているた



め、上記のような新興の標準に基づいたアプリケーションを作成するほうが容易である。

#### IT リソースの生産性を向上:

- XML は単一のプラットフォームおよび単一のテクノロジーであるため、メンテナンスコストを削減し、リソースの再利用性を高める。
- XML のドキュメント構造は形式的記述であるため、多様なタイプのツールがあるほか、統合や実装が容易である。
- また XML ドキュメントは、明確な構造であるためマニュアルでの修正も容易である。
- オブジェクト指向のデータベースや Java のみならず、Cobol など旧来の開発言語との相性も良い。

#### 安価な商業製品の利用が可能

- XML を使用することにより、商業製品に使用されている独自フォーマットに依存しないで済む。XML ベースのアプリケーションは多様な環境に移植可能であるため、単一のソフトウェアやハードウェアに固定されてしまうことがない。
- 開発環境において多様なツールの使用が可能(構文解析、視覚化ツール、検証エンジンなど)であるため、自社開発の必要性が軽減される。
- XML テクノロジーは一般に低コストでの導入が可能であるため、小規模ユーザーにも適している。

実装が容易に可能であり、仕様としての誤解の余地のない XML は、標準化作業において新しい且つ有効なアプローチであり、SWIFT としては今後進めていくことが望ましいと考えられる。XML 利用の効果を計量的に測定するのは、システムデザインの構造の変化を伴うことから、非常に困難ではあるが、開発過程のテスト(定例修正)などは明確に軽減される。

## 6. 結論

過去 5 年間に XML の重要性は高まってきており、多くの産業で、グループ内(Intra-Organisation)を中心にデータ交換の基盤として定着してきた。個々の組織での取り組みには温度差があり、より広い分野での活用には尚時間がかかるかもしれないし、SWIFT が世界各国に利用されていることから、よりその傾向が強いかもかもしれない。しかしながら、本ペーパーで述べられたように、XML の良い点・悪い点を踏まえながら、今後更にコミュニティと相談をしながら、将来 XML に移行をしていくにあたり、具体的なステップを踏んでゆくことを提唱する。

