



SIDを使用したOSS/BSS統合

Telemangement Forum SID (Shared Information/Data) 実装の
課題および解決策

目次

> 1.0 はじめに	1
> 2.0 TM ForumのSIDを使用する理由	1
> 3.0 トリプルプレイの実装例	3
> 4.0 SIDと他のシステムとの間の複雑なデータマッピング	4
> 5.0 ビジネスルールに基づくモデルベースのデータ一貫性	7
> 6.0 モデルベースのセマンティック・データルーティング	8
> 7.0 モデルベースのエラー管理による誤入力処理	10
> 8.0 影響分析による変更管理	12
> 9.0 結論	14

図表

> 図1: 抽象化による情報共有	2
> 図2: SIDとの対話型GUI	3
> 図3: トリプルプレイの実装処理	4
> 図4: SIDのCustomerクラスのプロパティ	5
> 図5: 計算属性の値の取得例	6
> 図6: SIDからデータソースへのマッピング	7
> 図7: ビジネス検証ルールの定義	8
> 図8: モデルベースのセマンティック・データルーティング	9
> 図9: モデルベースのエラー管理	10
> 図10: RulesResultsに追加されたエラーメッセージ	11
> 図11: customerNameの変更による影響のサマリー	12
> 図12: 影響分析の詳細	13

1.0 はじめに

通信業界団体の Telemanagement Forum (TM Forum) は、共通情報 (SID : Shared Information/Data) モデルを用いた通信業界における企業運営の共通言語を開発しました。オリジナルの SID モデル用の UML (統一モデリング言語) 定義に加え、XSD (XML スキーマ定義) 表現を追加しました。この SID XSD は重要な進展をもたらし、統合ビジネスアプリケーション開発にあたり、再利用可能なデータモデル基盤を提供します。プログレスソフトウェア社では、Progress® DataXtend® SI (Semantic Integrator) を利用することで、共通データモデルとしての SID の価値を生かし、堅牢な統合環境の構築を支援し、BSS/OSS (ビジネス/オペレーション サポート・システム) 統合プロジェクトのスピード、敏捷性、再利用性、およびデータの品質を高めます。

SID を使用した OSS 統合の実装に利用されるツールは、SID と他システムとのやり取りを容易にし、次の機能を提供する必要があります。

- > 複雑なデータマッピングおよびデータ変換
- > データの一貫性および検証
- > コンテンツベースのデータルーティング (コンテンツの内容によるルーティング)
- > 入力エラー管理
- > 変更の影響分析

このホワイトペーパーでは、SID モデルを OSS/BSS 統合プロセスに使用するための上記の要件について詳述し、DataXtend SI がそれら要件を満たす上で果たす役割について説明します。

2.0 TM ForumのSIDを使用する理由

SID モデルは、TM Forum の NGOSS (Next Generation Operations Systems and Software) イニシアチブの主要コンポーネントである抽象化された共通モデルです。NGOSS が目指すのは、市販のテクノロジーを使用したオープンな分散システムによる OSS/BSS の推進です。NGOSS は、大規模なシステム統合プロジェクトにおいて、コンポーネントアプリケーションのインターフェース間の連携を図るための、技術的に中立なアーキテクチャーのフレームワークを提供します。

疎結合したシステムが必要とするデータ抽象化を、SID は業界標準モデルとして提供します。再利用という観点では、異なるビジネスや業務においてデータを共有できるように、標準モデルも抽象的でなければなりません。SID は、システム統合に使用できる唯一の通信業界標準の抽象化された共通モデルと考えられています。抽象化された共通モデルの利点を図 1 に示します。

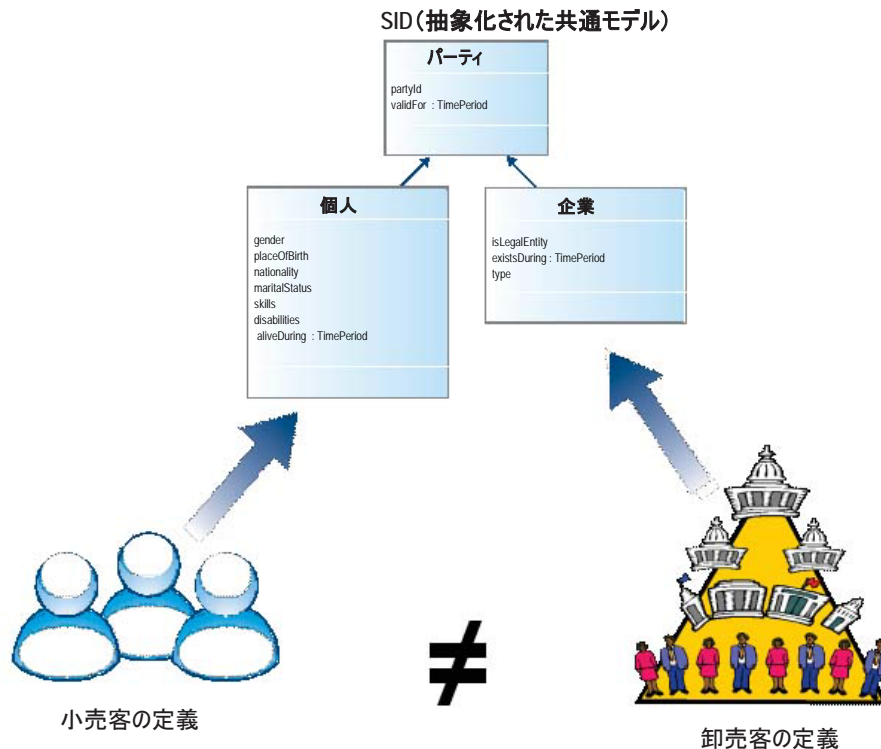


図 1: 抽象化による情報共有

同じ通信会社の中でも、個人向けの小売部門と企業向けの卸売部門とでは顧客の定義が異なります。小売部門の顧客定義が単純であるのに対して、卸売部門の顧客の定義には、VAT（付加価値税）識別番号など、小売客の定義にはみられない多くの属性が含まれる場合があります。SID が提供するパーティという抽象化されたデータモデルを使用すると、個人顧客と企業顧客の両方を表現できるため、小売部門と卸売部門が同じ情報および情報モデルを共有できます。

約 1,000 のクラスからなる SID モデルは、Point、Curve、Surface という OpenGIS ジオメトリクラスのような非常に全般的なコンセプトから、WAN（広域ネットワーク）プロトコルである PPP および X.25 のような非常に限定的なコンセプトまで、多岐にわたります。SID は難しく思えるかもしれませんが、ナビゲーション、表示、および SID とのやり取りを簡単に行える実装ツールを利用すれば、時間をかけずに SID によるシステム統合を進められるはずです。

DataXtend SI のツールは、パッケージ、クラス、リレーションシップからクラスの属性およびルールに至るまで、SID のあらゆる要素に対して動的なナビゲーションを提供し、GUI を使ったの統合作業を実施できます（図 2）。

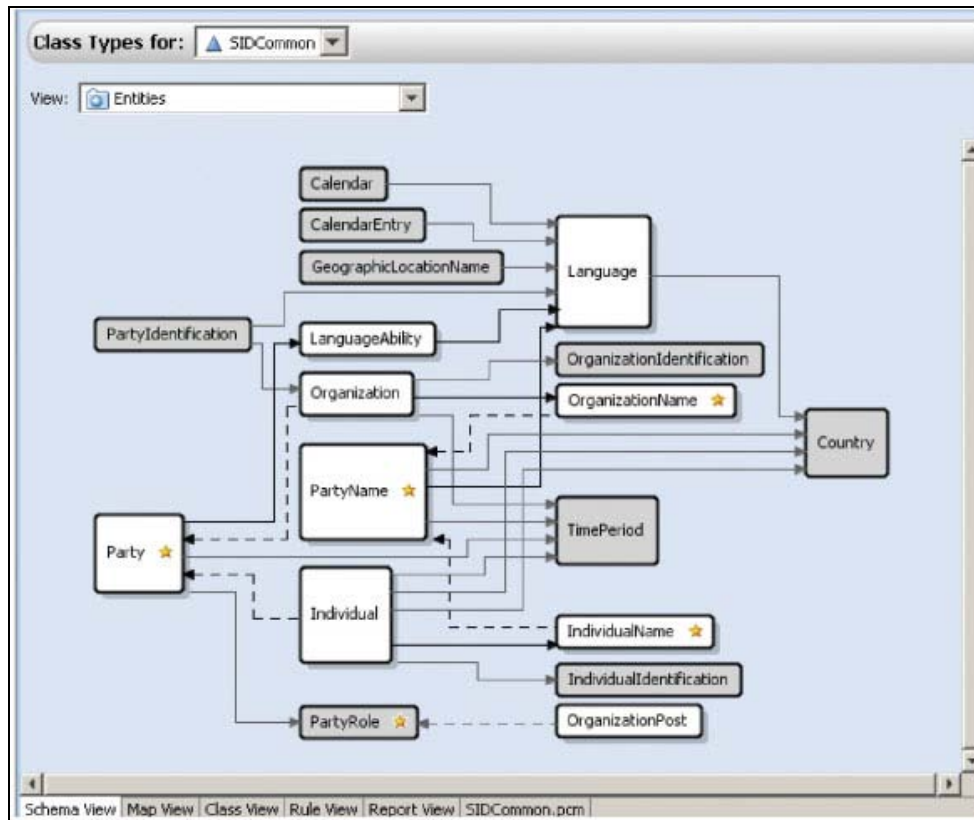


図2: SID との対話型 GUI

3.0 トリプルプレイの実装例

このホワイトペーパーでは、IP（Internet Protocol）の技術を通じて提供される、音声通話、映像配信、データ通信の3種のサービスを1本の回線で顧客に提供するトリプルプレイの実装例をご紹介します。

- > 音声通話： VoIP（Voice Over Internet Protocol）を使用
- > ビデオ放送： IPTV（Internet Protocol Television）プロトコルを使用
- > データ通信： 高速インターネット（ブロードバンド）を使用

このトリプルプレイの実装では、様々なベンダー製のアプリケーションインターフェース間を連携する共通モデルとしてSIDを使用しています。合計100近いオペレーションを持つ十数種類のアプリケーションインターフェースを利用し、在庫、サービス保証、CRM（顧客関係管理）、製品の価格設定、注文管理などの機能をスムーズに連携し活用しています。

SIDには、実装時にマッピングしなければならない必要な属性がすべてそろっているわけではありません。特定のデータサービスまたはデータソースに必要な属性がSIDに存在しない場合は、カスタム属性を作成できます。カスタム属性を作成し、計算式を関連付けることにより、その属性のデータ変換を実行時に自動化することもできます。

共通モデルを標準のまま維持することにより、共通モデルの新しいバージョンがリリースされても簡単にアップグレードができるので、実装ツールは、共通モデル自体を変更せずにカスタマイズできることが重要です。DataXtend SI ツールは、カスタマイズしたすべての情報をSIDモデルとは別にメタデータとして保管するため、SID自体に変更が加えられることはありません。

SIDの次の要素は、SID自体を直接変更せずにカスタマイズできます。

- > サブクラス
- > 単一属性
- > 計算属性
- > ルール

たとえば、SIDでは提供されていない形式での「顧客名」が必要であるとします。次の図は、DataXtend SIでSIDのCustomerクラスのプロパティを表示した画面です。

Properties of class "Customer"	
General Properties	
Name:	Customer
Abstract:	<input type="checkbox"/>
Last Modified By:	User
Java Package:	.sidCommon.CustomerDomain.CustomerABE
Primary Key:	[none]
Superclass:	PartyRole
Contained By	
Schema:	SIDCommon
Model:	SIDCommon
Exchange Model:	SDP
Subclasses	
Simple Attributes	
customerRank:	Long
customerStatus:	String
ID:	String
partyRoleId:PartyRole:	Object
status:PartyRole:	Object
Computed Attributes	
customerName:	String

図 4: SID の Customer クラスのプロパティ

この画面を見てみると、Customer クラスには顧客名のプロパティがありません。そこで、customerName という計算属性（属性名は任意）を追加し、顧客名を実行時に割り当てるようにします。計算属性の値には、たとえば SID の別のクラスから他の属性の値を割り当てることができます。

顧客名は、組織名の場合もあれば個人名の場合もあります。DataXtend の Expression Builder のようなツールを使用すると、属性の値を実行時に導出する計算式を作成できます。この計算式は、次のように記述することができます。

```
if 「サービスを注文する顧客が企業としてのパーティ・ロールを持っている」
then (その場合は)
```

```
「customerName は、SID の OrganizationName クラスの単一属性である tradingName と判定されま
```

```
す。」
else (サービスを注文する顧客が企業としてのパーティ・ロールを持っていない場合)
```

```
「customerName は、SID の IndividualName クラスの属性である fullName と判定されます（この場合
fullName は、個人名の部分を表現した単一属性を連結して導出される計算属性です）。」
```

次のスキーマ図は、計算属性 customerName の値が取得される箇所を示しています。

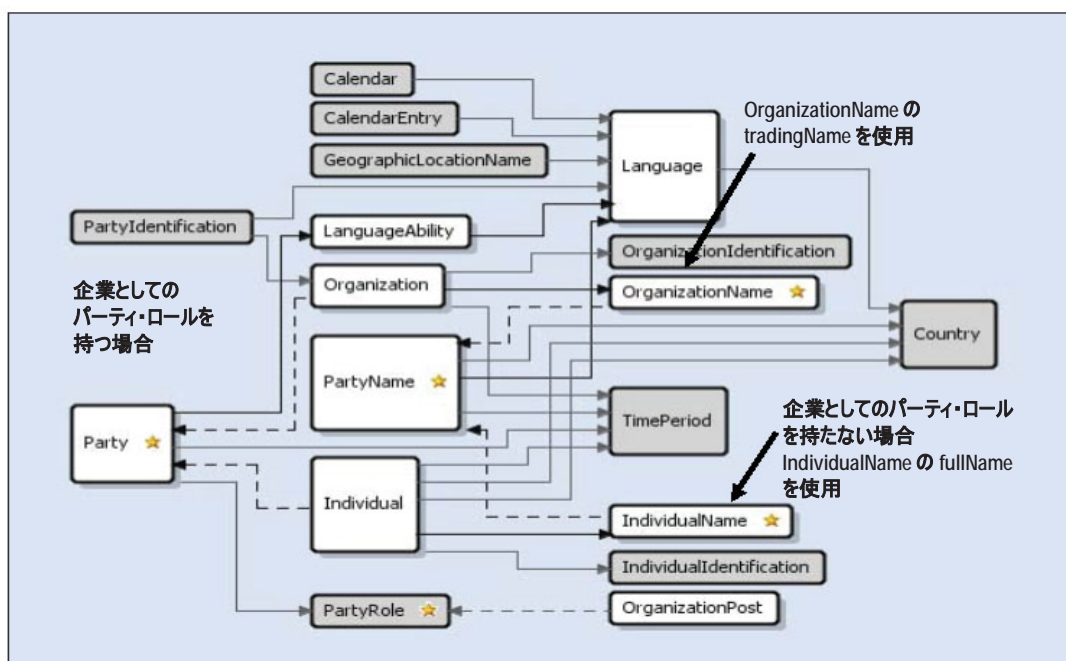


図 5: 計算属性の値の取得例

実装ツールでは、複雑な SID 共通モデルと他のアプリケーションインターフェースのデータモデルに対して、簡単なナビゲーションを提供する必要があります。GUI で容易にマッピングを作成、変更できなければなりません。次の図は、DataXtend SI で customerName 計算属性を、SID の Customer クラスからデータソースの SubscriberInfo クラスの Name 属性にマッピングする場合の例です。

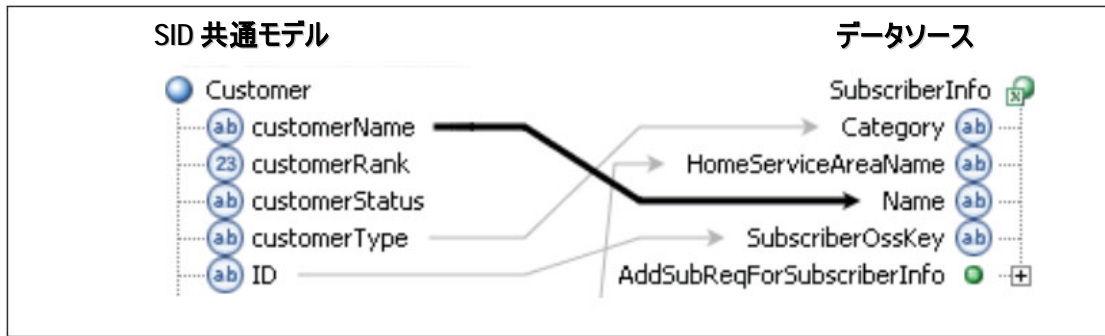


図 6: SID からデータソースへのマッピング

図 6 内のグレーの矢印は、実装に必要となるすべての単純、または複雑なデータのリンクを実装ツールで作成する様子を示しています。

5.0 ビジネスルールに基づくモデルベースのデータ一貫性

便利な実装ツールは、アプリケーション間でやり取りするデータの一貫性を保つためにビジネス検証ルールをデータに適用するメカニズムを備えている必要があります。たとえば、トリプルプレイ実装のビジネス検証ルールは次のようなものであると想定できます。

VoIP (voice over IP) の加入者はデータ通信サービスにも加入する必要がある

この種のビジネスロジックは次の特性を持っています。

1. XML だけでは設定できないルールがあります。XML は、メッセージフォーマット（フィールド長、数字か英数字かなど）の整合性を検証することについては威力を発揮しますが、この VoIP の例にあるようなフィールド間の依存関係や条件の定義を検証することはできません。
2. ビジネス上は明白なことであっても、複数の統合プロジェクト間では定義や実装に一貫性がないことが多く、データの品質やルールの運用に重大な問題が生じるおそれがあります。このようなルールは一般的に、アーキテクチャ全体（アダプタ内またはバス上）にプロシージャコードで記述されています。
3. 非常に複雑なルールを処理できる半面、処理速度向上やトレーニングにコストがかかるビジネスルール・エンジンを使用するほど複雑ではありません。

このようなルールを「ビジネスルール」に対して、検証ルールまたはセマンティックルールと呼んでいます。（一般的に、ビジネスルール・エンジンで管理すべきルールは、税金や価格の算出といった計算の実行時に適用されるビジネスルールです。）

実装ツールは、どのようなルールでもコーディングの必要なく、簡単に定義できなければなりません。次の図を参照してください。

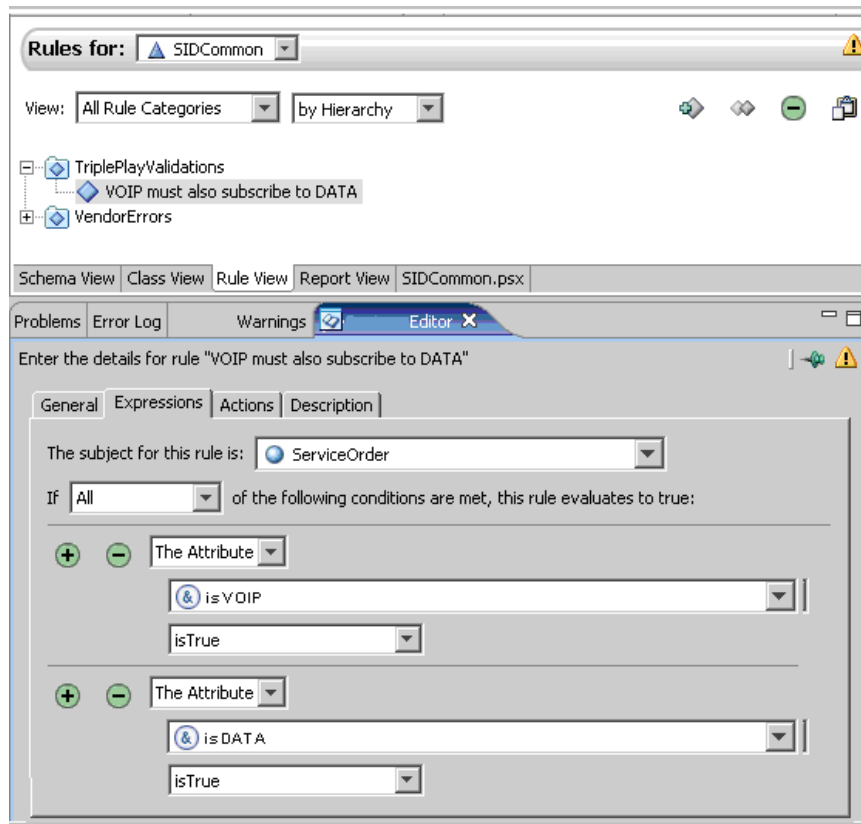


図 7: ビジネス検証ルールの定義

この例では、DataXtend SI の Expression Builder を使用して、ServiceOrder クラスの次のルールを作成しています。

VOIP must also subscribe to DATA (VoIP の加入者はデータ通信サービスにも加入する必要がある)

このルールは、ブーリアン属性である isVIOP および isDATA がどちらも真 (true) であるかどうかをチェックし、どちらも真であれば True と評価します。

6.0 モデルベースのセマンティック・データルーティング

優れた実装ツールは、どのようなデータ項目でもカスタム・コーディングなしに SID データ項目へマッピングできることが重要です。また、計算属性を前章のように使用することで、自動データ変換を実現できます。

しかし実装ツールは、データを実行時に分析できるルールを定義することによってセマンティック・ルーティングを行い、正しいメッセージインターフェースを自動的に割り出し、構築することによって、ESB がそれを物理的システムにルーティングできるようにする必要があります。セマンティック・ルー

ティングは、(メッセージの作成後に発生し、ESBがそのメッセージを実際のシステムのエンドポイントにルーティングする)メッセージバス・ルーティングと比較して、異なりますがが必要です。

DataXtend SIのExpression Builderは、データ変換とデータルーティングをメッセージの内容に基づいて管理し、カスタマイズプログラムのコーディングなしでSIDに定義されている前提条件に従いマッピングできます。DataXtend SIを使用すると、データ変換およびルーティングの条件ロジックがコードとしてではなく、モデルとして定義され、メタデータとして取得されます。そのため、再利用が可能で、どのような変更も迅速に実装できます。この機能がなければ、複雑なif-then-elseコーディングを記述しなければならず、カスタムコードとしてESBで再利用することも、BPM(ビジネスプロセスマネジメント)アプリケーションで再利用することもできません。そして、たとえSIDに密接に関連したメタデータであったとしても、そのロジックはSIDの他のメタデータとともに取得されることはありません。

たとえば、トリプルプレイのオーダーが3つのサービスすべてに対するものであったとしても、実装では、オーダーがサービスに応じて異なるデータソースにルーティングされる必要が発生する場合があります。次の図に示すように、バックエンドでは、VoIPへのオーダー、ブロードバンド(データ通信)へのオーダー、IPTVへのオーダーがそれぞれ別のデータソースに行く場合が想定されます。

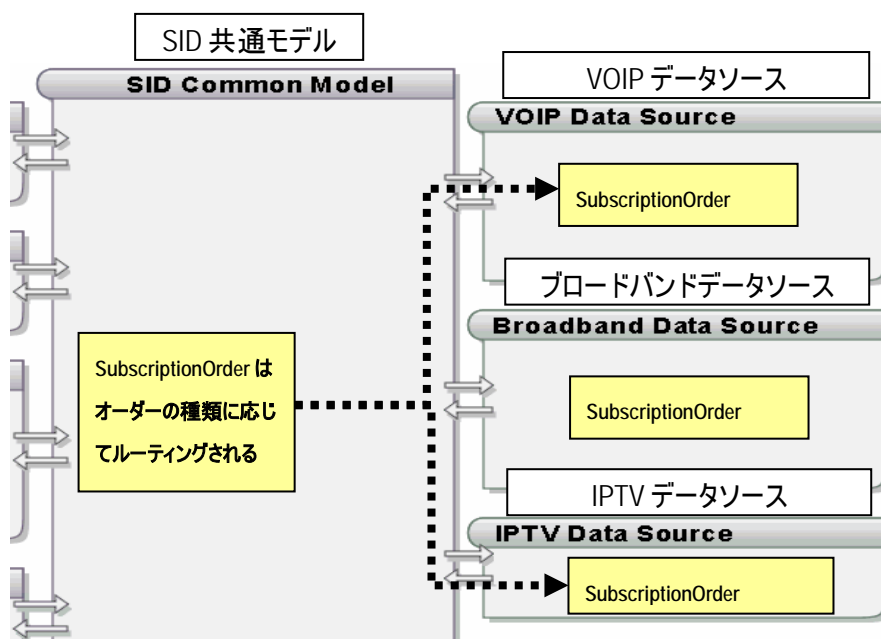


図 8: モデルベースのセマンティック・データルーティング

このルーティングはモデルのセマンティックに基づいていますが、この例では、あらかじめSIDで定義されている「サービス」のタイプに従いルーティングされています。

7.0 モデルベースのエラー管理による誤入力処理

実装ツールは、不正確なデータがシステムに送信されないように、データ検証機能を持っている必要があります。誤ったデータが送信されると、データソースやデータベースが破損しかねません。スキーマは既にあるレベルでデータ入力エラーの検証機能を持っており、たとえば数値データであるべきところに誤って文字列データが入力されることを防止してくれます。

大半の企業は、エラー処理に要する手間を減らそうとして、独自のエラー処理システムを開発しています。しかし多くの場合、それらは個々のプロジェクトレベルのエラー処理システムであって、企業全体で共通に使用することはできません。また、企業規模のエラー処理システムの開発に膨大な工数をかけても、メタデータとして取得できず、プロジェクト間で簡単に再利用はできません。

より高度なモデルベースのエラー管理であれば、エラー内容のメッセージを返し、誤入力を防止し、おそらく何らかの入力エラーの訂正も可能となります。モデルベースのエラー管理は、最初のエラーでは終了せず、エラーに関係なくメッセージ全体を検証する必要があります。XSLT だけではこのような高度な検証を提供できないため、エラー処理のカスタムコードで XSLT を「上書き」することで、より完成度の高い分析を提供する必要があります。

また、モデルベースのエラー管理でエラーが発生した場合、BPM アプリケーションまたはその他の応答受信側が正確に対応できるよう、宣言的で詳細に富んだ応答が返される必要があります。しかし、XSLT などの現在のテクノロジーは、詳細情報をほとんど示さない、あいまいなエラー・コードのみを返すため、アナリストがかなりの手間をかけてエラー・コードを解釈し、エラーの原因を特定しています。

モデルベースのエラー管理手法を用いると、次の図の DataXtend SI にあるような分析画面でルールに対するエラーメッセージを記述できます。

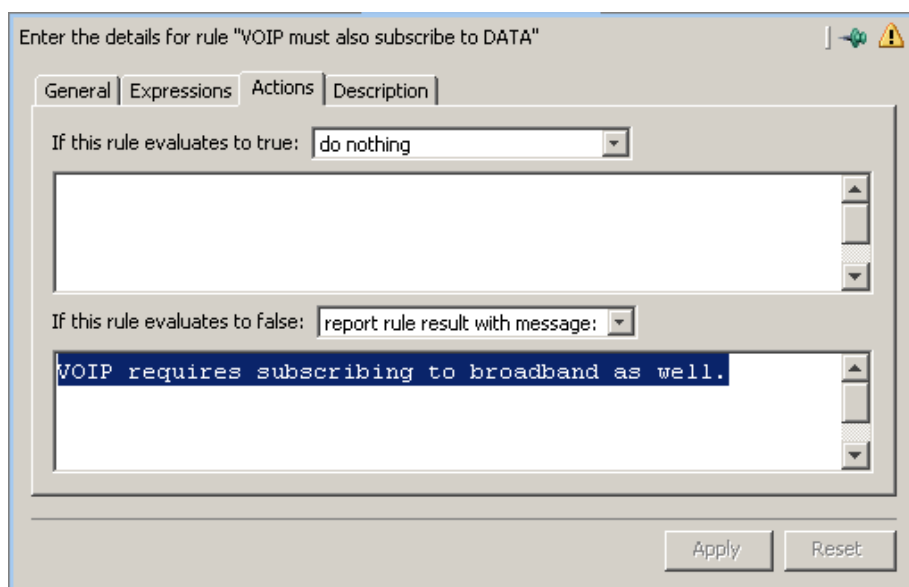


図 9: モデルベースのエラー管理

このルール（7ページに記述）では、加入者が VoIP サービスに加入する場合、データ通信（ブロードバンド）サービスとあわせての加入が必要でした。エラーメッセージは必要に応じて詳しく記述することができ、実行時に解釈される変数を含めることが可能なため、エラー発生時のエラー内容の理解が大幅に改善されます。DataXtend SI では、このルールが実行時に False と判定した場合、このメッセージは RulesResults リスト（次の図を参照）に追加され、API のオペレーションによって解釈されます。

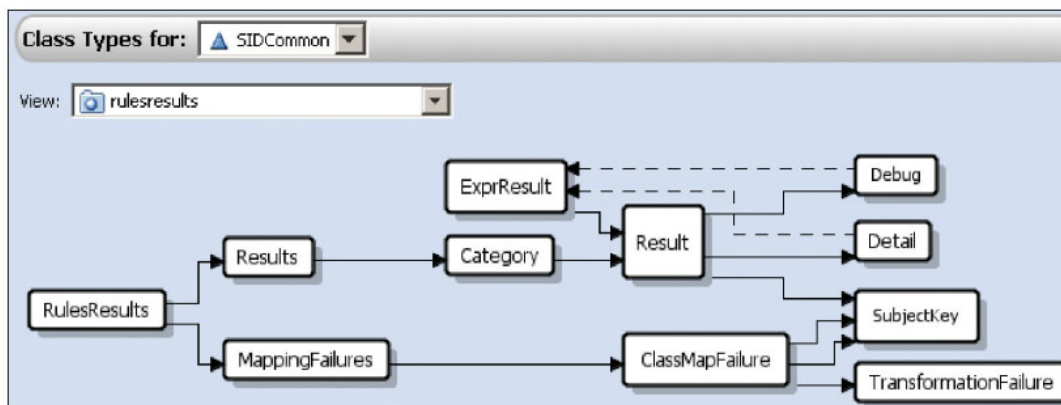


図 10: RulesResults に追加されたエラーメッセージ

8.0 影響分析による変更管理

システムの大型化および複雑化に伴い、影響分析は、ソフトウェアの変更による影響を予測、管理するためのテクニックおよびツールを提供するようになっています。実装のどの部分が互いにどのような影響を及ぼし合うのかを知る必要があるため、実装ツールは、カスタム機能を含めて、任意のSIDとベンダーのエンティティに関する影響分析を提供できなければなりません。

実装ツールは、特定のエンティティが他のスキーマのどこで使用されるかなどを示すだけでなく、そのエンティティを使用するオペレーションについての明確な理解を提供し、どのように使用されるのかについても説明する必要があります。これには、すべてのオペレーションのシステム内の位置に関係なく、ある変更を行ったときに、他のどのオペレーションが再テストを必要とすることになりそうかを特定することも含まれます。

たとえば次の DataXtend SI レポートを見ると、customerName 計算属性に変更を加えた場合に、変更やテストが必要となる箇所が分かります。











Type Affected	
	1 Class
	3 Computed Attributes
	1 Domain Schema
	6 Rules
	4 Simple Attributes
	4 Transformation Rules
	6 Web Service Operations
	13 XML Complex Types
	1 XML Data Source Operation
	2 XML Schemas

図 11: customerName の変更による影響のサマリー

この表を見ると、customerName に加える変更が、1つのクラス、3つの他の計算属性、1つのドメインスキーマなどに影響を及ぼすことが分かります。実際にどのクラス、計算属性、およびその他のエンティティが影響を受けるのかについては、次の図に示すような詳細レポートで確認できます。

Details of Impact Analysis					
Steps	Type Affected	Name Affected	Reason	Affected by	Occurs in
1	Class	Customer	owns	customerName	SIDCommon
1	Transformation Rule	customerName => Name	transforms from	customerName	SIDCommon => LeapstoneDataSource
2	Computed Attribute	CustomerAffected	uses	Customer	SIDCommon
2	Computed Attribute	CustomerAffected	uses	Customer	SIDCommon
2	Borrow Schema	SIDCommon	uses	Customer	SIDCommon
5	XML Complex Type	AddSubscriber	contains	AddSubReq	LeapstoneDataSource
5	XML Complex Type	CramerAssignResource	contains	MsgAssignResource	CramerDataSource
5	XML Data Source Operation	updateSubscriber	uses	SubscriberRecord	SubscriberDTDDataSource
6	Computed Attribute	correlationId	uses	AddSubscriber	LeapstoneDataSource
6	Web Service Operation	addSubscriber	uses	AddSubscriber	LeapstoneDataSource
6	Web Service Operation	assignResource	uses	CramerAssignResource	CramerDataSource
6	XML Schema	LeapstoneDataSource	uses	AddSubscriber	LeapstoneDataSource

図 12: 影響分析の詳細

これは、影響を受けるエンティティのリストの一部です。各エンティティへの影響を詳しく説明しています。たとえば、SID 共通モデルの Customer クラスは属性 customerName を所有しているため、直接の影響を受けます（1 ステップ）。また、計算属性 CustomerAffected は自身の customerName を持つ Customer クラスを使用しているため、customerName から 2 ステップ離れています。変更の影響は、すべてのアプリケーション API で任意のレベルまで管理できます。

有益な影響分析ツールおよびレポートは、変更に必要な時間および労力を見積もるだけでなく、そのシステムの変更または修正に対して別のアプローチ取るべく判断材料も提供します。

9.0 結論

TM Forum の SID モデルは、データのマッピングおよびデータ変換のための共通のセマンティック（意味論）モデルを提供することにより、通信プロバイダによる OSS/BSS 統合の簡易化を支援します。ただし、モデルベースの実装を行うには、実装ツールが次の機能およびメリットを持っていない限りなりません。

> **SID との間の複雑なデータマッピングおよびデータ変換**

共通モデルと他のメッセージフォーマットとの間で属性を GUI でマッピングします。必要に応じて、データ変換のためのカスタムコードの追加および計算属性を定義できます。

> **XML では表現できない、データの一貫性および検証**

カスタムコードを使用せずに GUI で正当性を確保するためのルールを定義できます。

> **コンテンツベースのデータルーティング（セマンティック・ルーティング）**

実装でデータを実行時に検証するルールを定義し、適切なアプリケーションインターフェースに対して適切なメッセージを作成します。

> **入力エラー管理**

エラーと詳細なエラーメッセージをモデリングします。企業はこれを使用して適切な回復経路をモデリング、定義でき、ユーザーの満足度および信頼性を高めることができます。

> **変更の影響分析**

変更による影響分析を提供し、統合のライフサイクル全体で変更コストを削減します。

Progress Software Corporation について

Progress Software Corporation (NASDAQ: PRGS) は、ビジネスアプリケーションの開発、運用、統合、管理のためのアプリケーション・インフラストラクチャー・ソフトウェアを提供しています。IT のメリットを最大限に引き出す一方で、その複雑さと総所有コストを最小限に抑えることを目指しています。詳しくは、当社 Web サイト www.progress.com にアクセスするか +1-781-280-4000 にお電話ください。

日本プログレス株式会社

米国プログレス ソフトウェア・コーポレーションの日本法人として、ソニックソフトウェア株式会社とデータディレクトテクノロジーズ株式会社を統合し、2008 年 12 月 1 日に設立され、SOA 基盤ソリューションや金融業界向けのソリューションを提供します。

SOA 基盤である Progress Sonic ESB や、無停止配信ソリューションの Progress SonicMQ に加え、IBM メインフレームの情報資産活用を行う DataDirect Shadow、バッチ処理やオンライン処理におけるデータベースの高速化を可能にする DataDirect Connect、SOA ガバナンスを行う Progress Actional によって、お客様の企業規模や要求に合わせた SOA 基盤からガバナンスを含めたソリューションを実現します。さらに、金融業界で昨今ますますニーズの高まっているアルゴリズムトレーディング分野で実績のある Progress Apama の提供も行っています。



本社

Progress Software Corporation, 14 Oak Park, Bedford, MA 01730 USA Tel: +1 781 280-4000 Fax: +1 781 280-4095
www.progress.com

日本

〒102-0082 東京都千代田区一番町18番地 川喜多メモリアルビル Tel: 03-3556-7610
<http://www.progress-japan.co.jp>

各支社の所在地と連絡先については、www.progress.com/worldwideを参照してください。

© Copyright 2009 Progress Software Corporation. All rights reserved. ProgressとDataXtendは、Progress Software Corporation、または米国や他国におけるその子会社、系列会社の商標または登録商標です。本書で記述されているその他の商標は、各所有企業の私有財産です。