

## XML設計技術部会 活動報告



2010年3月16日

XMLコンソーシアム XML設計技術部会  
リーダー 遠城 秀和 (NTTデータ)

Copyright ©2010 XML Consortium

1

## XML設計技術部会活動報告



- XML設計技術講座開催結果報告
  - 昨年の勉強会を引き継ぎ、今年はXML設計技術講座を開催しました。
  - 本講座の概要およびXML設計技術の学習方法をご紹介します。
  
- 類似XMLメッセージ間の変換方法を検討して
  - 気象庁防災情報XML (JMAXML) メッセージをCommon Alert Protocol (CAP) メッセージに変換する方法を検討
  - 検討をとおして得られたXML設計時の考慮点をご紹介します。

記載されている会社名、商品名、又はサービス名は、各社の登録商標又は商標です。

Copyright ©2010 XML Consortium

2

## XML設計技術講座 開催結果報告



2010年3月16日

Copyright ©2010 XML Consortium

3

### XML設計技術講座



- 開催回数： 4回
  - 9月30日
  - 10月26日
  - 11月30日
  - 12月17日
  
- 延べ参加者：

■ 総数	66社	109名
■ 会員	39社	82名
■ 非会員	27社	27名

## 第1回XML設計技術講座



- オープニング: 13時30分～
- 講義(1): 13時35分～
  - XMLスキーマ設計全体の流れ
- 講義(2): 14時50分～
  - XMLスキーマ入門
- 演習(HandsOn): 15時30分～
  - XMLデータのインスタンスを作ってみる
    - グループ分けを行い、各グループでXMLデータを作成します。
    - 相互評価を行います。
    - 経験者が各グループに入り、リードします。
- その他: 17時55分～ 18時00分
  - アンケート
    - 以降の活動に役立てさせていただきます。
  - 連絡事項等

初心者中心

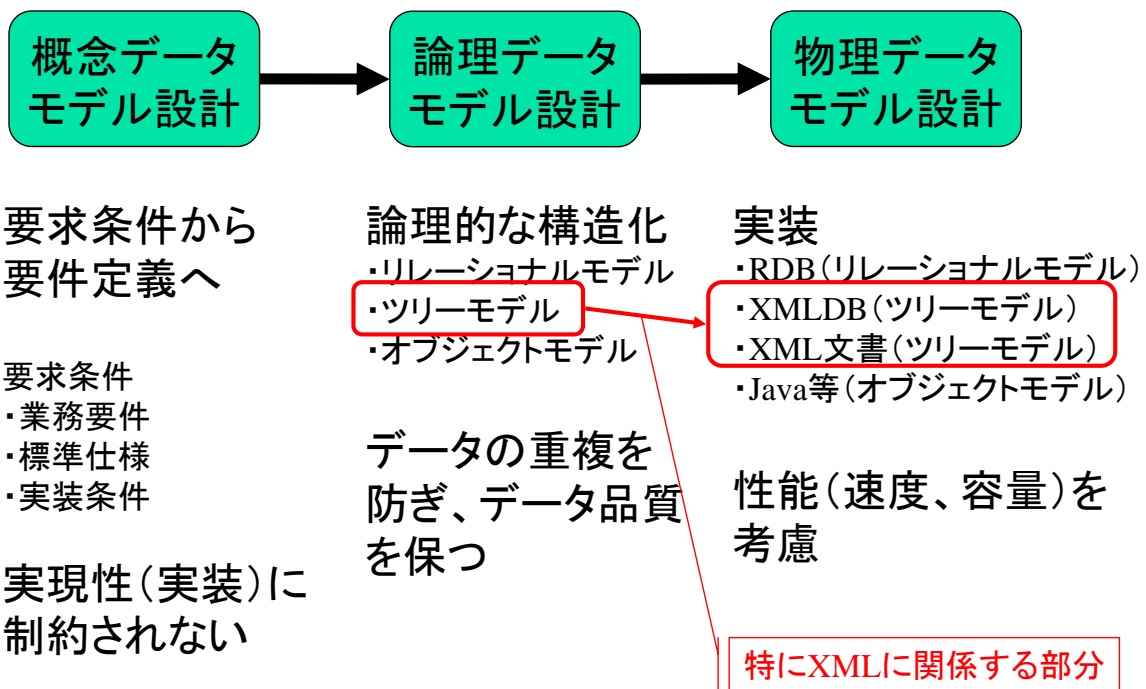
## 第2回XML設計技術講座



- オープニング: 13時30分～
  - テーマ:
    - 先人の知恵(ContactXMLのXMLスキーマ)を読み解く(1)
    - ～ XMLスキーマを作ってみよう ～
- 講義: 13時35分～
  - 構成パターンで分類した既存スキーマの紹介(1)
    - ContactXMLのXMLスキーマ
- 演習(HandsOn): 14時30分～
  - XMLデータのスキーマを作ってみよう
    - 各自でXMLスキーマ化を検討します。
    - グループ分けを行い、各グループでXMLスキーマを作成します。
    - 相互評価を行います。
    - 経験者が各グループに入り、リードします。

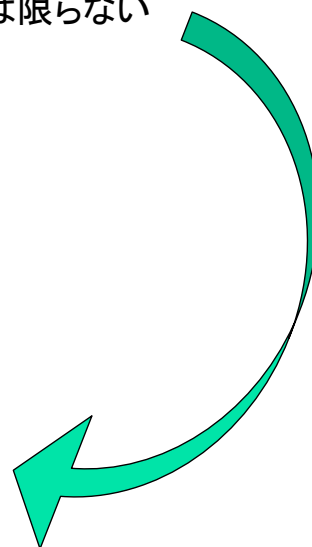
- オープニング: 13時30分～
  - テーマ:
    - 先人の知恵(NewsMLのXMLスキーマ)を読み解く(2)
    - ～ 共有性を高める辞書を作ってみよう ～
  
- 講義: 13時35分～
  - 構成パターンで分類した既存スキーマの紹介(2)
    - NewsMLのXMLスキーマ
  
- 演習(HandsOn): 14時30分～
  - 辞書を作り、共有性を高める
    - 各自でXMLスキーマ化を検討します。
    - グループ分けを行い、各グループでXMLスキーマを作成します。
    - 相互評価を行います。
    - 経験者が各グループに入り、リードします。

- オープニング: 13時30分～
  - テーマ:
    - 設計作業を効率化する便利な道具
    - ～ パターンを使ってみよう ～
  
- 講義: 13時35分～
  - 設計作業を効率化する便利な道具
    - XMLスキーマ作成および検証のツール紹介
  
- 演習(HandsOn): 14時30分～
  - パターンを使いXMLインスタンス、XMLスキーマを作る
    - 各自でXMLスキーマ化を検討します。
    - グループ分けを行い、各グループでXMLスキーマを作成します。
    - 相互評価を行います。
    - 経験者が各グループに入り、リードします。



## 概念データモデル設計

- 設計と言っても、分析が中心
- 情報表現の要求条件を整理し、要件定義にまとめる
- 分析結果はデータモデルで表現するとは限らない
- データモデルの分析
  - 情報の特性の抽出
  - 情報の使い方の抽出
- どの範囲で共有するデータか
  - どの範囲で共有するデータか
  - システム的にどこで使われるデータか
- 演習 (HandsOn):
  - XMLデータのインスタンスを作ってみる



- 設計と言っても、分析が中心
- 情報表現の要求条件を整理し、要件定義にまとめる
  - 要求条件
    - 実現性がないものもある
      - 要件定義からは落ちるが、要求はある
  - 要件定義
    - 実現できる要求条件
      - 実現方法を一つに限定するとは限らない
    - 実現方法の限定化は、論理データモデル設計で、
- 分析結果はデータモデルで表現するとは限らない
  - データモデル表現
    - ER図やClass図など
  - データ例示
    - XMLデータ (≠ XMLスキーマ)

XMLインスタンス  
が出来れば、  
達成と考えます

目標です

- データモデルの分析
  - 情報の特性の抽出
    - 情報(帳票など)はどの様な形をしているか？
    - 情報のまとまりを考える
      - 繰り返しの単位
      - 全体と部分
    - 情報の塊が具体的な実体を表しているか？
      - 一つの実体は一つの塊で
        - 情報の重複を防ぐ
    - 情報の塊の間関係
  - 情報の使い方の抽出
    - 機能の設計ではないので、想定ですが

塊を考えてみて  
ください

塊の入れ子関係を  
考えてみてください

考えてみる程度で  
結構です



- XML — XMLスキーマ設計
  - 命名規則と名前空間
    - 名前空間:モジュール化、タグ名、属性名の作り方
  - 型と構造
    - ルート要素、関連の表現、多重度と空要素、属性と要素
  - XMLスキーマ記述方針
    - 記述パターン選択、XMLスキーマ分割方針、など
- XMLDB — XMLDB設計
  - ...
- (将来の)柔軟性のある仕様
  - 将来の柔軟性とは、「要件が変わっても修正が少ない」
  - 共通性の高い仕様
- 演習(HandsOn):
  - XMLデータのスキーマを作ってみよう

0. 方針作成
  - 方針案の確認/レビュー
1. 辞書(ドラフト)の作成
  - 辞書(ドラフト)のレビュー
  - 辞書のXMLスキーマ(ドラフト)の検証
2. メッセージのデータモデル作成
3. ハーモナイゼーションの実施
  - メッセージ作成時に見つかった新たな情報項目、コードを辞書中の項目と刷り合わせる
4. 辞書の完成
  - ハーモナイゼーション結果を統合する。
  - 辞書のレビュー
  - 辞書のXMLスキーマの検証
5. メッセージのXMLスキーマを作成
  - メッセージのXMLスキーマの検証
6. 辞書、XMLスキーマの公開



辞書と  
ハーモナイゼーションが  
ポイント



# XMLスキーマを作るまで

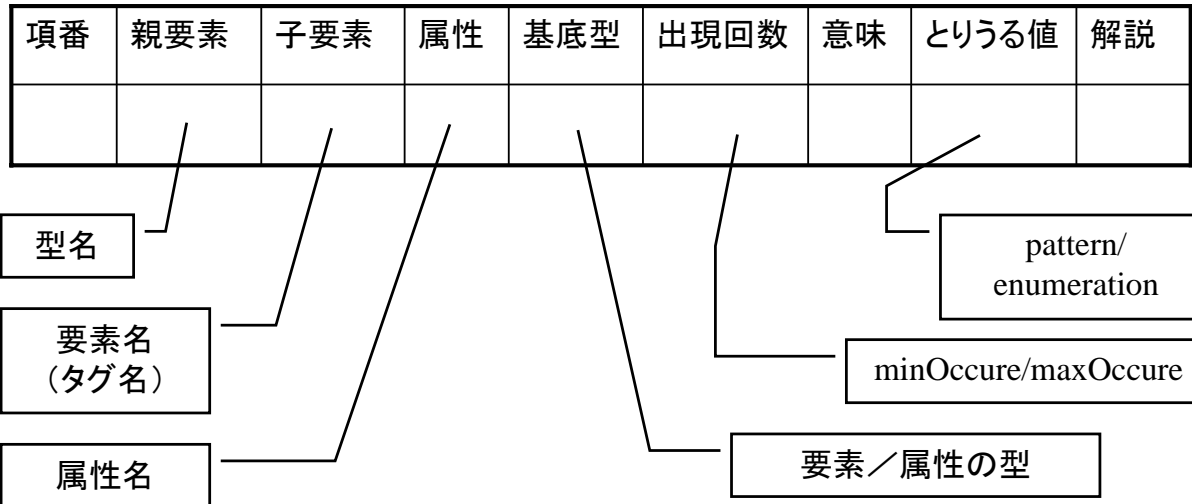
## 辞書の作り方

### 「辞書」

- 「辞書」とは
  - デイリーコンサイス 国語辞典第3版
    - 言葉を一定の順序に並べ、発音・意味・用法を記した書物、辞典
    - コンピューターで、仮名漢字変換用のデータファイル
- しかし、ここでは
  - データ項目の名称や意味を登録したもの。
    - 情報項目の一覧
    - データ辞書、データ・ディクショナリとも呼ばれる。
- 例えば、

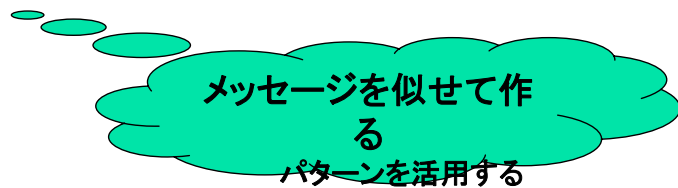
項番	親要素	子要素	属性	基底型	出現回数	意味	とりうる値	解説

- 辞書からの自動生成
  - 「辞書」からXMLスキーマを機械的に作成出来ると、XMLスキーマのメンテナンスが楽に



注)本検討は途中状況で確定仕様とは異なる場合があります。

- 機能パターン
  - 要素を構築するパターン
- 統合パターン
  - 複数要素を統合するパターン



- 意味的構造パターン
  - 意味的な詳細
  - ContactXML
    - 住所
    - 氏名
    - 連絡先
  - 既存辞書を参考に

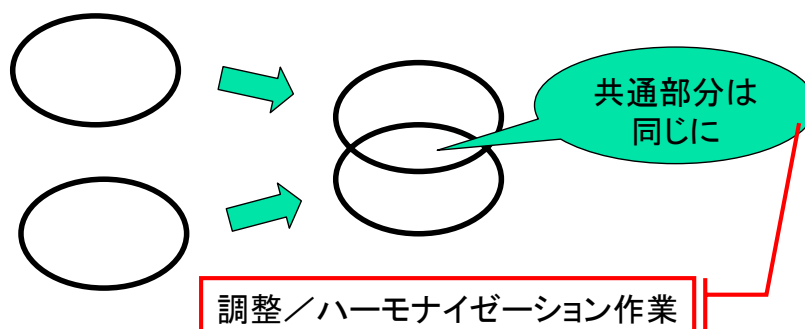
親要素	子要素	属性	基底型		
ContactXMLItemTYPE					
	PersonName				
	PersonID				
	Address				
	Occupation				
	...				

- 演習 (HandsOn):
  - パターンを使いXMLインスタンス、XMLスキーマを作る

## 辞書(ドラフト)が出来ました

### 共通性を高めるために ハーモナイゼーションをしましょう

- データ設計
  - トップダウン設計
    - 関連する構造を使って設計
      - 約款、業務規定、組織構造、標準仕様、など
  - ボトムアップ設計
    - 抽出した詳細項目をまとめ構造化して設計
      - 伝票、既存DB
- 調整／ハーモナイゼーション
  - ボトムアップ設計とトップダウン設計
  - 複数のボトムアップ設計の間



## XMLスキーマを作るまで

### XMLスキーマを 書くコツがあります

Copyright ©2010 XML Consortium

23

## メッセージのXMLスキーマ作成

- 前もってXML Schemaの使い方のルールを作成
  - ネームスペースの使い方
    - メンテナンスを考えると小さい単位がいい
    - XMLの形での処理を考えると大きい単位がいい
    - メッセージのグループ化が出来るのであれば、その単位で
  - 「要素」と「属性」の使い分け
    - 原則「要素」を使う
    - 「属性」は補足的にしか使わない
  - グローバル定義、ローカル定義の使い分け
    - 使い分けには4種類ある
    - 定義を再利用するならグローバル
    - 名前を再利用するならローカル
    - 長く使うならグローバルの方がいいと思う
- データモデルに従ってXMLスキーマを作成
- なるべく作成は自動化
  - スキーマの記述生成は、ルールが十分に整備できるとツール化できそう。

Copyright ©2010 XML Consortium

24

- XMLスキーマ作成パターン選択
  - メリット
    - 冗長性の低減とスキーマ全体の統一性が高まる
  - バリエーション
    - グローバルな定義は、再利用性が高い
    - ローカルな定義は、コンパクトに作れる
  - スキーマ作成パターン
    - Russian Doll
      - Element: ローカル、Type: ローカル
    - Salami Slice
      - Element: グローバル、Type: ローカル
    - Venetian Blind
      - Element: ローカル、Type: グローバル
    - Garden of Eden
      - Element: グローバル、Type: グローバル

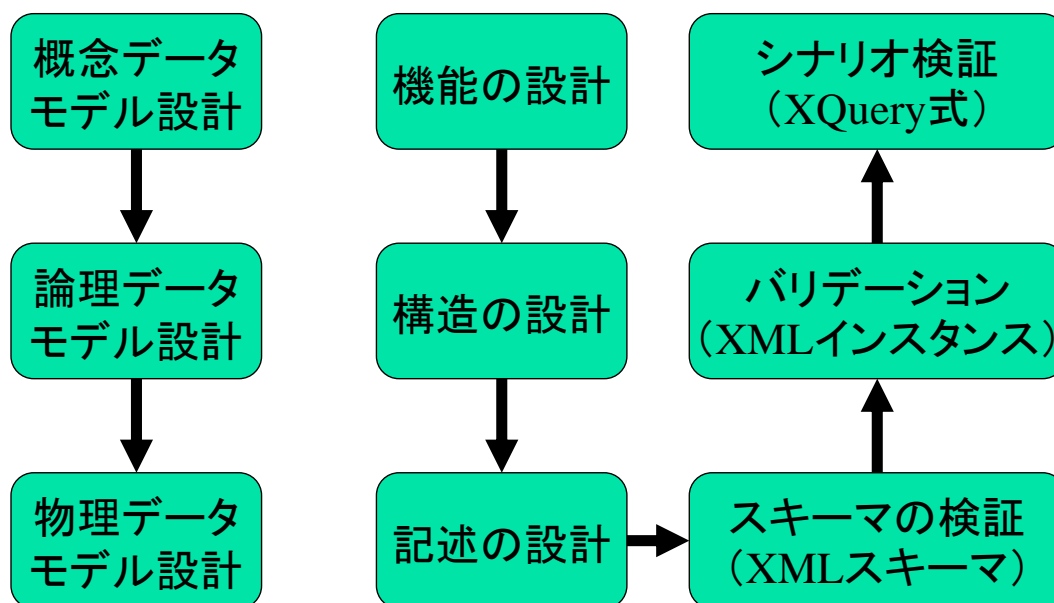
同じ名前を  
使い回せる

名前空間を越えて  
使われる項目 (Element)

- 気象庁支援プロジェクトで色々出た課題
  - スキーマ言語は何を使えばよいか？
    - XML Schema
    - RELAX NG
  - バージョニングはどうすればよいか？
    - 追加変更とバージョンをどう関係付けるか？
  - XML Schemaでどう書けばいいのですか？
    - 項目と属性のどちらを使うのですか
    - 全部文字列にしてよいのでしょうか
      - 基本要素を作りましょう
    - 表はどう表現すればよいのでしょうか。
      - 基本構造を考えましょう
    - 電文に急な追加があるのですが
    - 名前空間の接頭辞指定がうまくいかないのですが

## XMLスキーマが書けました

### 検証(試験)をしましょう



注)XMLスキーマ検証のツールを使えば容易です。

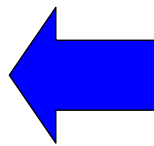
ぜひ  
柔軟かつ共通性の高い  
XMLスキーマを  
設計しましょう

参考

- メタ構造機能パターン
  - 構造化による拡張
  
- コンテナ
  - 様々コンテンツを格納
  - 例: NewsML
  - Anyを使う

```
<aaa>
  <xxx> </xxx>
  <yyy> </yyy>
</aaa>
```

```
<aaa>
  <zzz> </zzz>
</aaa>
```



```
<element name="aaa" type="aaaType"/>
<complexType name="aaaType">
  <sequence>
    <any namespace="##any"
      processContents="lax"
      minOccurs="0"
      maxOccurs="unbounded"/>
  </sequence>
</complexType>
```

- メタ構造パターン
  - 拡張メカニズム
    - 抽象的なタグを属性で具体化して拡張
    - 例: NewsML
    - Microformat的な使い方

```
<aaa type="xxx"> zzzzzzz </aaa>
```

```
<address class="vcard">
  <a class="fn url" href="http://tantek.com/">
    Tantek Çelik
  </a>
</address>
```

- ボキャブラリーコントロール
  - 固有のボキャブラリー定義により要素や値を拡張する
  - 例: NewsML
  - XML Schemaと似た考え方
  - 拡張メカニズムとの組み合わせ

```
<typeDef name="t1" values="x1 x2"/>
<typeDef name="t2" pattern="x[1-9]"/>
```

```
<aaa type="t1"> x1 </aaa>
<aaa type="t2"> x8 </aaa>
```



■ メタ構造パターン

■ ヘッダ・ボディ

- 例: JMAXML
- 制御用情報と処理用情報を分ける

```
<doc>
  <header>      </header>
  <body>        </body>
</doc>
```

■ サマリ・本文

- 例: JMAXML
- よく使われるサマリを本文の前に出す

```
<doc>
  <summary>      </summary>
  <body>        </body>
</doc>

<doc>
  <header>      </header>
  <summary>
</summary>
  <body>        </body>
</doc>
```

■ 機能要素パターン

- 属性の使い方

■ 言語設定

```
<Name xml:lang="ja"> 遠城 </Name>
```

■ ふりがな

```
<Name pronunciation="えんじょう"> 遠城 </Name>
```

■ 単位

```
<Length unit="cm"> 100 </Length>
```

■ 和暦日時

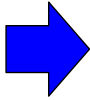
```
<DateJP date="2009-12-17"> 平成21年12月17日 </DateJP>
```

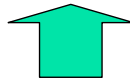
- . . .

■ 統合パターン

■ マージ統合

■ 子要素を混ぜる

<pre>&lt;aaa&gt;   &lt;x1&gt; &lt;/x1&gt; &lt;/aaa&gt;  &lt;aaa&gt;   &lt;x2&gt; &lt;/x2&gt; &lt;/aaa&gt;</pre>		<pre>&lt;aaa&gt;   &lt;x1&gt; &lt;/x1&gt;   &lt;x2&gt; &lt;/x2&gt; &lt;/aaa&gt;</pre>
---	---	---



```
<complexType name="aaa">
  <sequence>
    <element name="x1" minOccurs="0" maxOccurs="1"/>
    <element name="x2" minOccurs="0" maxOccurs="1"/>
  </sequence>
</complexType>
```

■ 統合パターン



■ 型統合

■ Union/Choice


- UnionやChoiceを使ってXML Schema上で型を統合する

■ 多重度統合

■ オプション化

<pre>minOccure="1" maxOccure="1"  &lt;attribute name="type"   type="string" /&gt;</pre>	 	<pre>minOccure="0" maxOccure="1"  &lt;attribute name="type"   type="string" use="optional"/&gt;</pre>
---	--	---

■ 条件を緩める

<pre>minOccure="1" maxOccure="2"</pre>		<pre>minOccure="0" maxOccure="unbounded"</pre>
--	---	--

### ■ 統合パターン

#### ■ 複数化

- 一個だけでなく複数書ける様に

#### ■ 子要素化 — 子要素で書く

```
<aaa>
  <aaaltem> xxx </aaaltem>
  ...
  <aaaltem> yyy </aaaltem>
</aaa>
```

#### ■ 親要素付加 — 親タグで括る

```
<aaaList>
  <aaa> xxx </aaa>
  ...
  <aaa> yyy </aaa>
</aaaList>
```

## XMLスキーマ記述のコツ: 項目と属性の使い分け

### ■ 項目でも属性でも構造化できるが、使い分けは？

- DTDでは項目の詳細を指定できないが、属性は詳細を指定できる。
- XML Schemaでは項目も属性も詳細を指定できる。
- 属性の値に構造を入れられないが、項目なら入れられる。

### ■ 原則

- 項目で表現し、属性は使わない。
- 例外的に単位やコード表の指定は属性を使う。

## ■ 物理量

- 単位(m/sなど)が付く
- 時間当たり(1時間当たりの雨量、3時間当たりの雨量)
- 時系列データ(9時、10時、、、)

<xxx type= unit= refID= condition= altString= QC= >

- type 分類
- unit 単位 “ミリ”、“メートル”など
- refID 時系列参照番号
- condition 状態 “約”、“以上”など
- altString 代替文字列 “不明”、“約100ミリ”
- QC 品質管理情報 “正常”、“非常に疑わしい”など

## ■ 風向

- 「北東の風のち南の風」
  - 最初の状態
  - 断続現象
  - 変化後の状態
- 地域毎の風向
  - 複数の風向で表現
  - 風向と地域を組みに

注)本検討は途中状況で確定仕様とは異なる場合があります。

## ■ 時刻・時間

- 日時時刻の有効桁数
  - YYYY-MM-DDThh:mm:00
  - YYYY ? YYYY-MM-DD ? YYYY-MM-DDThh ?
    - 属性formatで有効桁を指定
- さまざまな時刻
  - 発現時刻、発表時刻、報告時刻、発効時刻、失効時刻
    - 要素名で識別?
- あいまいな時刻
  - 日 時刻が特定できない
  - 日頃 日付が特定できない
  - 時 当該時刻
  - 時 分が特定できない
  - 時頃 時が特定できない

<xxxDateTime significant= precision= dubious= >  
YYYY-MM-DDThh:mm:00 </xxxDateTime>

注)本検討は途中状況で確定仕様とは異なる場合があります。

## ■2次元的構造をどう表現するか

## ■木構造に押し込む

```

<A>
  <B>
    <C> </C>
    <C> </C>
    ...
  </B>
  <B>
    <C> </C>
    <C> </C>
    ...
  </B>
</A>

```

コンパクト  
縦横が固定される

## ■IDで参照

```

<A>
  <E id="1"> ... </E>
  <E id="2"> ... </E>
  <E id="3"> ... </E>
  ...
  <B1>
    <C1>
      <D1 ref="1" />
      <D1 ref="2" />
      ...
    </C1>
    <C1>
      <D1 ref="3" />
      ...
    </C1>
  </B1>
  <B2>
    <D2>
      <C2 ref="2" />
      <C2 ref="3" />
      ...
    </D2>
  </B2>
</A>

```

やや冗長  
縦横両方を同時に  
指定可能

## ■電文に急な追加

- 例えば、急に地震観測装置を設置したので、、、
- そのため
  - 新しい仕様やスキーマの配布が間に合わない
  - 新しい仕様やスキーマへの対応が間に合わない
- 追加を許容するXMLスキーマを考えておかないと、、、

## ■追加項目も許容するXMLスキーマ

- デメリット
  - 検証をしないことになる。
  - 記述する意味がなくなる。
- メリット
  - 柔軟性が高くなる。
  - 追加が起きる場所が限定され、事前に想定したシステムが作れる。

## ■追加される対象

- コード値
- 項目

- コード値の追加
  - Enumerationを使ってコード値を指定
  - Unionを使って任意のコード値を受け付ける単純型を組み合わせる
  
- 項目の追加
  - Anyを使えば、任意の項目も許容できるはず、、、
  - Unique Particle Attribute 規則に違反!!!
  - 直前の項目を必須にして回避
    - 解決策とは言いがたい。
  
- もう一工夫できないか!!!

- |   |   |
|---|---|
| <ul style="list-style-type: none"> <li>■ スタイルはどれにするか。           <ul style="list-style-type: none"> <li>■ ロシアンドール               <ul style="list-style-type: none"> <li>■ 項目: ローカル、型: ローカル</li> </ul> </li> <li>■ サラミスライス               <ul style="list-style-type: none"> <li>■ 項目: グローバル、型: ローカル</li> </ul> </li> <li>■ ベネチアンブラインド               <ul style="list-style-type: none"> <li>■ 項目: ローカル、型: グローバル</li> </ul> </li> <li>■ ガーデンオブエデン               <ul style="list-style-type: none"> <li>■ 項目: グローバル、型: ローカル</li> </ul> </li> </ul> </li> <br/> <li>■ グローバル: 辞書内で唯一           <ul style="list-style-type: none"> <li>■ 沢山の名前が必要(×)</li> <li>■ 構造を同じに(○)</li> </ul> </li> <li>■ ローカル: 親の項目内で唯一           <ul style="list-style-type: none"> <li>■ 名前の使いまわしが可能(○)</li> <li>■ 構造が個別に(×)</li> </ul> </li> <br/> <li>■ ベネチアンブラインドを採用</li> </ul> | <pre> &lt;A&gt;   &lt;B&gt;  &lt;/B&gt;   &lt;C&gt;     &lt;B&gt;  &lt;/B&gt;  &lt;— 名前の                   使いまわし   &lt;/C&gt; &lt;/A&gt;   名前(項目)がローカル </pre> <pre> &lt;A&gt;   &lt;B&gt;  &lt;/B&gt;   &lt;C&gt;     &lt;CB&gt;  &lt;/CB&gt;   &lt;/C&gt; &lt;/A&gt;   名前(項目)がグローバル </pre> |
|---|---|

<a:A>

<a:B> </a:B>

<a:C>

<b:D> </b:D>

<b:E> </b:E>

</a:C>

</a:A>

- Cがローカルな項目だと
  - ロシアンドール、ベネチアンブラインド
  - Cの名前空間はa???

<a:A>

<a:B> </a:B>

<b:C>

<b:D> </b:D>

<b:E> </b:E>

</b:C>

</a:A>

- Cがグローバルな項目だと
  - サラミスライス、ガーデンオブエデン

■どちらが直感に合うか？

- 方針
  - 基本はベネチアンブラインドスタイル
  - 外部参照される項目はグローバルに宣言

### ■ 辞書・XMLスキーマの分類

- 共通辞書、分野別辞書
- (電文の種別)

### ■ バージョン管理

- Namespaceに、版情報を入れられるようにして
  - コンパチビリティがある版には同じNamespace
  - 異なるXMLスキーマには別のNamespace
- バックワードコンパチビリティ
  - 過去(前版)のインスタンスも許容するXMLスキーマか？
- フォワードコンパチビリティ
  - 将来(次版)のインスタンスも許容するXMLスキーマか？

## 類似XMLメッセージ間の 変換方法を検討して



2010年3月16日

Copyright ©2010 XML Consortium

47

### 類似XMLメッセージ間の変換方法を検討



- 目的
  - 類似XMLメッセージ間の変換を想定した際に、考慮する点を明らかにする。
- 対象
  - 気象庁防災情報XML(JMAXML)メッセージをCommon Alert Protocol(CAP)メッセージに変換する方法を検討



→変換できそう、○固定、□不明、■使っていない



注) 本対応関係は検討案であり、保証するものではありません。

## 検討結果

- XMLメッセージ(JMAXML)とXMLメッセージ(CAP)の対応付けは容易
  - 辞書レベルで共通化が図れると対応付けはより容易
- 考慮点
  - 項目の値の対応付けは、意味の対応付けが必要で、実際の運用を想定した検討が必要
    - 中身の判る利用者が入らないと判断が難しい
    - XMLの設計としては不要だが、値の意味は明確にしておくとい
  - 全ての情報が対応付けるのは難しく、元のメッセージへの参照が必要
    - 外部から参照可能とするために、識別子を用意しておくとい
    - 複数項目を組み合わせて唯一性が保証されていればよい