素性論理に基づくXML文書ルール記述言語DRDLとインターネット文書交換システムへの応用

今村 誠[†] 渡邉圭輔[†]
増塩智宏[†] 渡部明洋[††]

本稿では，インターネット上の文書交換システムにおけるXML文書の検証，変換，およびデータベース連携などのXML処理の開発基盤となるXML文書ルール記述言語DRDL（Document Rules Description Language）を提案する．DRDLの技術特長は，自然言語処理分野で用いられる素性論理に基づいて，W3C推奨のスキーマ記述言語XML Schemaでは記述できないXML文書の内容や構造間の制約を簡潔に表現できる文書ルール記述言語を提案した点にある．DRDLの論理式を手続き的に解釈実行するDRDLプロセッサにより，XML文書の内容検証，内容代入，およびデータベース連携などが実現できる．また，DRDLプロセッサ，DRDLエディタ，およびDRDLルールトランスレータからなるXMLアプリケーションフレームワークを開発した．そして，電子申請，設計仕様交換，受発注文書交換，および設備管理などの実用システムへの適用を通じて，DRDLのルール記述能力とDRDLフレームワークの開発生産性が実用レベルにあることを確認した．また，XML文書検証ルールの開発生産性に関しては，従来のJava言語による検証ロジック作りこみと比較して，開発時間で約2.7倍，コード行数で約6.5倍の改善が実現できた．

An XML Document Rules Description Language DRDL Based on a Feature Logic and its Application to Document Exchange Systems on the Internet

Makoto Imamura,[†] Keisuke Watanabe,[†] Tomohiro Masushio[†] and Akehiro Watabe[††]

We propose a document rules description language (DRDL) to develop XML processing functions such as validation, transformation and relational mapping in internet-based document exchange systems. Semantic constraints over the structures and the contents of XML elements are represented in DRDL as formulae in a feature logic traditionally used for representing linguistic constraints in unification grammars. A set-valued feature is newly introduced, and existential and universal quantifiers are used to represent cardinality and uniform constraints on multiple elements. We have developed an XML application development framework based on DRDL, and evaluated it in the industrial systems for government to business document exchange, elevator design support, facility management and Web EDI. The result shows that DRDL has sufficient expressive power and software development productivity for the system development of industrial applications. The proposed framework can improve hours of developing document rules 2.7 times and code line sizes 6.5 times in comparison with coding validation logic with Java language.

1. はじめに

インターネットを用いて業務プロセスを自動化するEC（Electronic Commerce）やCALS（Continuous Acquisition and Lifecycle Support）の進展にともなって，企業間での電子商取引，行政・企業間での電子申請，および企業間での設計文書交換などを実現するXML（Extensible Markup Language）文書交換システムの開発がさかんになってきた．これらのシステムでは，XML文書の表示や変換だけでなく，業務での文書データの自動処理を促進するために，文書内容が満たすべき規約（文書ルール）を検証する機能が，文書作成支援や文書受付を実現するうえで重要になる．しかし，W3C（World Wide Web Consortium）のXMLのデータ構造定義規格であるDTD（Document

† 三菱電機株式会社情報技術総合研究所
MITSUBISHI ELECTRIC CORPORATION, Information Technology R&D Center
†† 株式会社栗菱コンピューターズ東京支社
RITSURYO Computers CO., LTD., Tokyo Branch

Type Definition)[1] や XML Schema[2),3)] では，「要素内容間の制約（内容間制約）」や「要素内容と，要素の存在有無や反復回数などの文書構造との関係制約（内容・構造間制約）」などを表現できない[4]．そのため，これらの制約の充足性を検証する機能開発では，DOM（Document Object Model）[5] などの XML 文書構造木を直接操作するプログラミングが必要になり，「アプリケーションプログラム中への検証ロジック埋め込みによる開発効率低下」，および「文書規約変更時のコードの修正・コンパイル作業による保守性低下」が生じるという問題点があった．

本問題点を解決するために，内容間制約，内容・構造間制約などを簡潔に記述できる XML 文書ルール記述言語 DRDL（Document Rules Description Language）と，DRDL に基づく XML アプリケーション開発フレームワーク（DRDL フレームワーク）を開発した．

本稿の構成は，以下のとおりである．2 章では，XML 文書を交換する CALS/EC システムにおける XML 文書内容処理への要求機能と現状技術の課題について述べる．3 章では，2 章で述べた現状技術の課題を解決するために XML 文書ルール記述言語 DRDL を提案する．4 章では，DRDL を用いた XML アプリケーション開発を支援する DRDL フレームワークについて述べる．5 章では，電子申請，設計仕様交換，受発注文書交換，および設備情報管理などの実用システムへの DRDL フレームワークの適用結果を報告することにより，DRDL が 2 章で述べた要求を満たしていることを示す．6 章では，XML 文書ルール記述の理論と実用との関係について考察し，最後の 7 章ではまとめを述べる．

2. XML 文書内容処理への要求機能と現状技術の課題

EC や CALS 向けのインターネット文書交換システムでは，交換される XML 文書データは，バックエンドの業務システムと連携できるように，文書交換に関与する組織間で合意したデータ項目や形式に関する規約を満たしているかどうかをチェックする機能が必要になる．たとえば，民間から行政に許認可を申請する電子申請では，申請書に対して，法令や行政手続を反映した書式や記載要領を満足するかをチェックする必要がある．また，客先の要求仕様を設計部門に伝達する設計仕様交換では，設計仕様書に対して，製品仕様項目の設定値が設計部門の想定範囲であることをチェックしたり，客先に提示する製品価格や製造期間を見積もったりする必要がある．また，購入会社と販売会社間で受発注データを交換する EDI（Electronic Data Interchange）では，受発注伝票に対して，購入品の価格合計値や添付文書の有無をチェックしたり，既存の資材システムが受理できる形式への変換処理を行ったりすることが必要になる．

そのため，インターネット文書交換システムは，XML 文書の作成，XML 文書の内容チェック，XML 文書の送信，XML 文書の受信，XML 文書の内容チェック，そして XML 文書の変換／業務システム登録などのモジュールから構成されている．本稿では，これらのモジュールにおける典型的な XML 処理として，XML 文書の送信前と受信後の内容チェック，XML 文書作成時の入力支援や業務システムからのデータ取り込み，および，業務システム登録などを扱う．また，業務システムのデータベースとしては，現状よく用いられている関係データベース（RDB: relational database）を想定した．

本章では，インターネット文書交換システムにおける XML 内容処理の技術課題を整理するために，2.1 節で，業務システム開発からの XML 内容処理への要求機能について述べ，2.2 節で，2.1 節の要求を実現する際に生じる現状要素技術の課題について述べる．

2.1 業務システム実現に必要な XML 内容処理機能

本節では，2.1.1 項で，XML 文書内容処理の仕様を記述する言語に要求される表現能力について述べ，2.1.2 項で，XML 文書内容処理機能の開発効率化への要求について述べる．

2.1.1 XML 内容処理の表現能力

電子申請，設計仕様交換，EDI などのインターネット文書交換システムにおける XML 文書内容処理を，XML 文書の要素内容に対する制約記述という観点から分類すると，以下に示すような「内容間制約の処理」，「内容・構造間制約の処理」，「依存型制約の処理」，および「XML-RDB 双方向写像制約の処理」に分類できる．アプリケーションにおける具体的な処理機能例については，5.2 節「アプリケーションにおける評価」の (1) で詳述することとし，本項では 2.2 節「現状技術の課題」に必要な一般的な事項について説明する．

(1) 内容間制約の処理

- XML の要素内容間制約をチェックする検証処理．たとえば，各々の要素 <A> の値が，対応する要素 <B> と要素 <C> の和とそれぞれ等しいことをチェックするなどの処理．
- 反復出現するある要素の合計値を計算して別の要

素に挿入するといった代入処理.

(2) 内容・構造間制約の処理

- ある要素の内容と，別の要素の内容の出現有無や反復数との関係制約を検証する処理．たとえば，要素 <A> の反復数は，要素 <B> の値と等しいことをチェックするなどの処理.
- 内容・構造間制約を満たすように内容を補完／代入する処理．たとえば，要素 <A> の反復数が，要素 <B> の値と等しくなるまで，要素 <A> を追加するなどの処理.

(3) 依存型制約の処理

- ある要素の値と別の要素のデータ型との関係制約を検証する処理．たとえば，要素 <A> の内容に依存して，要素 <B> のデータ型が可変となるようなデータ型を検証する処理.

(4) XML-RDB 双方向写像制約の処理

- XML 文書中の要素内容が RDB 中に格納される値と整合性がとれていることをチェックする処理．また，入力 XML 文書中の選択肢を RDB 中から抽出／代入する処理.
- 1 つの XML 文書中の要素内容を RDB の複数テーブルに切り分けて登録する処理．また，RDB 中の複数テーブルに登録されているデータを 1 つの XML 文書に復元する処理.

2.1.2 XML 内容処理機能の開発効率化への要求

XML 内容処理の開発を効率化するための要件として，「(1) 開発・保守の容易さ」と「(2) 他機能との連携の容易さ」について述べる.

(1) 開発・保守の容易さ

昨今のスピード経営に対応するには，業務変更に応じて XML 文書処理機能を効率的に開発・保守できなければならない．たとえば，電子申請では法令変更に対して，また，設計仕様交換では新機種導入や機種マイナー変更に対して，迅速に対応する必要がある．以下では，開発・保守の容易さを実現するための要件として，「文書ルール作成の容易さ」，「文書ルール再利用の容易さ」，および，「文書ルール差し替えの容易さ」について述べる.

(i) 文書ルール作成の容易さ

文書ルール作成の容易さを実現する手段には，作成時間の短縮（文書ルール作成に要する時間の短縮）と作成者の拡大（情報システム部門の担当者だけでなく，業務部門の担当者でも文書ルールを作成できるようにする）の 2 つがある．ここで，業務部門の担当者とは，たとえば，電子申請では，申請書の様式作成者であり，また，設計仕様交換では，機種の仕様設計者である.

(ii) 文書ルール再利用の容易さ

文書ルールは，文書を交換する組織ごとの業務システム用に再利用しやすいものであることが望ましい．たとえば，EDI では，購入側企業と販売側企業の両者の企業にとって再利用しやすいものであるとよい.

また，文書ルールの寿命は，情報システムの実行環境の寿命より長い場合も多い．そこで，HTML から Java への実行環境移行などにともなうシステム再構築の際に，文書ルールは，移行システムの実行環境用の言語への自動変換などにより，再利用できることが望ましい.

(iii) 文書ルール差し替えの容易さ

業務変更に応じた文書処理機能の変更を容易にするために，文書ルールは簡単に差し替え可能である必要がある．たとえば，電子申請での法令の変更や，設計仕様交換での機種のマイナー変更の際に，ソースコードの変更やコンパイルをせずに，文書ルールファイルの差し替えで対応できることが望ましい.

(2) 他機能との連携の容易さ

対象業務が複雑になると，文書処理機能は他のプログラムと密に呼び合いながら動作させる必要が生じる．たとえば，設計仕様交換では，クライアント側の文書入力支援やサーバ側の文書変換において，CAD（Computer Aided Design）や設計支援システム用の特殊なデータを処理する業務特化プログラムと密に連携させる必要が生じる場合がある.

2.2 現状技術の課題

本節では，2.1 節で述べた XML 内容処理を記述するための標準的な言語の課題について，XML スキーマ言語と XML 変換言語に分けて整理する.

2.2.1 スキーマ言語

XML 文書の内容に対するデータ制約を記述するために，DTD の制約記述能力を向上させた様々なスキーマ言語が提案されている．たとえば，オブジェクト指向である W3C 推奨の XML Schema，文法指向の RelaxNG [6]，およびパターン指向の Schematron [7] などがある．しかし，XML Schema や RelaxNG は，構造定義を主な目的としており，2.1 節で述べた内容間や内容構造間の制約のチェックや代入処理は対象範囲外である.

一方，Schematron は，XPath [8] を用いたパターン記述により，内容間制約や内容構造間制約を表現できる．XPath は，「要素 A の内容が要素 B の内容に等しい」や「要素 A の内容が，反復出現する要素 B の内

容の合計値に等しい」といった内容間検証を記述することができる．さらに，Schematron は，if 文などの導入により，XPath の記述能力の不備を補っている．しかし，2.1.1 項で述べたような「反復出現する複数の要素間で共通の等式を満たすなどの制約」や依存型制約を表現できないという問題がある．また，XPath の構文と Schematron が新たに導入した構文とに機能の重複があるので，内容間制約を JavaScript に自動変換するといった文書ルール再利用を目標とする言語としては，意味論の簡潔性に欠けている．

2.2.2 変換言語

XSLT（XSL Transformations）[9] は，XPath を用いて XML 文書の変換規則を記述する W3C 推奨の変換言語である．XSLT は，XPath では記述できない if 文や反復要素ごとの制約を表現する for-each 文を持ち，さらに，XSLT プロセッサの拡張機能を用いることにより複数 XML 文書間の処理を記述できるので，2.1.1 項にあげた XML 文書内容処理をほぼ実現できる表現能力を持っている．

しかし，XSLT は，XML 文書内容検証や代入処理用に，システム開発者が手軽に使う記述言語としては以下のような問題点がある．

(i) XPath が指し示すカレントノードごとに変換規則を適用するので，XML 文書中の 2 つ以上の要素内容間を比較する処理では，比較対象要素を対称的に記述できず，要素内容のコンテキスト指定の記述が煩雑になる．その結果，2.1.2 項で述べた「文書ルール作成の容易さ」や「文書ルール再利用の容易さ」の実現が困難になる傾向にある．

(ii) 表現能力は大きいが，低レベルの雑多な構文を含む大きな言語仕様であり，習熟に手間がかかる．

3. XML 文書ルール記述言語 DRDL

本章では，2.1 節の要求を満足する記述力を持つとともに，簡潔な構文と意味を持つ XML 文書ルール記述言語 DRDL を定義する．以下順に，DRDL の設計方針，DRDL のコア部分である XML Constraint Language（XCL）の構文，XCL の操作的意味，アプリケーション開発用に XCL を拡張した言語 DRDL，そして，DRDL の記述例について述べる．

3.1 言語の設計方針

Schematron や XSLT の記述が複雑になっている主な原因は，XML の要素内容の指定に用いている XPath 中の制約記述の構文と，新たに導入している制御構文とに機能の重複があるので，言語の意味論が複雑になっていることによる．本稿では，自然言語処理分野のユニフィケーション文法[10] で用いられる素性論理をベースにして，XPath の自然な拡張として，if 文，for-each 文，およびデータ型制約などを含む一階述語論理の部分言語を定義する．以下，提案言語の構文と意味の設計方針を述べる．

3.1.1 構　　文

2.1 節の XML 文書内容制約を扱うために，Smolka [11] や Jhonson [12] らの素性制約式に，集合値を持つ素性やデータ型に関する制約を記述できるように拡張した XCL 式を導入する．言語を定義するにあたっては，一階述語論理との類推がきくように，項（term）と式（formula）という概念を用いる（一階述語論理の論理式との類似性に注目した素性論理の素性制約式の解説は文献 13) に譲る）．

(1) XCL 項

素性制約式の構成要素である項に相当する XCL 項として，W3C 勧告 XPath のロケーションパスを用いる．ロケーションパスとは，要素名を “/” で連結したもので，XML の階層構造をたどるパスである．

(2) XCL 式

素性制約式に相当する XCL 式として，Smolka のパスを XCL 項で置換することにより得られる素性制約式を用いる．XCL 式は，真偽値が定義されるものであり，XPath の式（expression）に相当するが，一階述語論理との対応がより明確なので，XPath の式と比較して，構文や意味がより簡明になっている．Smolka らが扱った言語的な制約と XML の制約との本質的な差異は，同じ素性名（XML では，要素名）が反復出現する点にある．そのため，素性の値として集合値を扱う必要が生じるので，XCL では，反復要素の内容がある等式を共通に満足するなどの制約を表現するために全称限量子 $\forall$ を導入し，また，反復要素の出現数に関する制約を表現するために存在限量子 $\exists$ を導入する．さらに，素性論理のソートに相当する概念として XML Schema のデータ型を用いる．データ型に関する制約を，要素内容に関する制約と同格に扱えるようにすることにより，ある要素の値に依存したデータ型（依存型）が定義できるようになる．

3.1.2 意　　味

(1) XCL 項

XCL 項の意味は，XPath の場合と同様に，ロケーションパスが指し示す XML 文書中の一部分（ノードの集合）で，“ノードリスト”，“ブール値”，“数値”，“文字列” に属する値とする．

(2) XCL 式

ユニフィケーション文法の文解析における素性構造

のユニフィケーションでは，素性論理における素性制約式の充足可能性の判定手続きを用いていた．しかし，素性制約式の充足可能性判定を 2.1 節で述べた課題解決に用いるには，「アルゴリズムが実装上遅いこと」と「双方向性を持つ計算が文書ルールの記述者にとって分かりにくい」という問題がある．そこで，実用上要求されているのは，2.1 節で述べた内容検証と内容代入の処理であるという特殊性を生かして，素性制約式の操作的な意味としては，与えられた XML 文書の内容をチェックする「チェック解釈」と，与えられた XML 文書の内容を書き換える「代入解釈（破壊的代入解釈）」の 2 つを導入する．

チェック解釈では，XCL 式の意味を，XML 文書を入力として，真偽値を出力として返す関数として定義する．入力 XML 文書が指定されると，限量子の束縛変数がとりうる値は，あるロケーションパスが指し示す値からなる有限集合になるので，XCL 式の充足可能性判定は，命題論理の充足可能性に帰着される．そのため，チェック解釈では，XCL 式の操作的意味を，通常の手続き型言語の論理演算と同様の手続きとして定義できる．

代入解釈では，XCL 式の意味を，XML 文書を入力として XML 文書を出力として返す関数として定義する．具体的には，通常の手続き型言語の代入文と同様に，等式を左辺の変数に対する右辺の値の代入と解釈することにより，与えられた XCL 式を満たすように XML のノードの値を書き換える処理となる．

3.2 DRDL のコア言語 XCL の構文

本節では，DRDL のコアとなる言語である XCL の構文について述べる．通常の一階言語の場合にならって，XCL の項（term）と式（formula）を，下記のように定義する．

(1) XCL 項の構文

XCL 項は，次のように定義される，

(a) 定数は，XCL 項である．
(b) 変数は，XCL 項である．
(c) XPath におけるロケーションパスは，XCL 項である．

(2) XCL 式の構文

XCL 式は，次のように帰納的に定義される．

(a) s と t が XCL 項ならば，s=t，s!=t，s<=t，s>=t，s>t，および s<t は，XCL 式である（XCL パス比較式と呼ぶ）．s=t を XCL パス等式と呼ぶ．
(b) s が XCL 項であり，t が XML Schema で定義されたデータ型ならば，s: t は，XCL 式である．s: t を XCL 型式と呼ぶ．
(c) Φ と Ψ が XCL 式ならば，Φ ∧ Ψ，Φ ∨ Ψ，¬Φ，および Φ ⇒ Ψ は，XCL 式である（総称して，XCL 論理式と呼ぶ）．
(d) x が変数，s が XCL 項，そして Φ が XCL 式ならば，∀x ∈ s. Φ は，XCL 式である（全称 XCL 式と呼ぶ）．
(e) x が変数，s が XCL 項，n は自然数，そして Φ が XCL 式ならば，∃x ∈ s s.t. (count(.)=n). Φ は，XCL 式である（存在 XCL 式と呼ぶ）．ただし，n は評価結果が自然数となる式でもよい．たとえば，n は，3+5 や，XPath 式 t が指し示すノードの数である count(t) などでもよい．また，= は，!=，<=，>=，または，> などの比較記号でもよい．

ここで，XCL 式「∀x ∈ s. Φ」では，XCL 項 s（通常は，ロケーションパス）が指し示す要素がすべて与えられた制約を満たすことを表現したいので，一階述語論理との類推がきくように，全称限量子記号 ∀ を用いた．また，同様に，XCL 式「∃x ∈ s s.t. (count(.)=n). Φ」では，XCL 項 s が指し示す要素のうち少なくとも 1 つが与えられた制約を満たすことを表現したいので，存在限量子の記号 ∃ を用いた．さらに，XCL 項 s が指し示す反復要素の出現数に関する制約を表現できるように，少なくとも n 個以上（あるいは，多くとも n 個以下）の要素がある制約を満たすことを表現する構文「count(.)=n」を追加した．

3.3 XCL 式の操作的意味

本節では，3.2 節で述べた XCL 式の操作的意味を，チェック解釈と代入解釈ごとに述べる．

(1) XCL 比較式

【チェック解釈】左辺と右辺の XCL 項が指し示す値の対が比較演算子（=，< など）を満たす場合は真を返し，そうでない場合は，偽を返す．ここで「値が等しい」とは，ノードリストの要素の XML 文書中の出現順序も含めて等しい場合とする（XPath の等式とは意味が異なる）．

【代入解釈】パス等式で左辺が定数でない XCL 項の場合にだけ定義され，左辺の XCL 項が指し示す XML 文書中のノードに対して，右辺が指し示す値

を代入する．DOM-API の値の代入（Node クラスの setNodeValue）に相当する．

(2) XCL 型式

【チェック解釈】左辺の XCL 項が指し示す値が，右辺のデータ型に属する場合には真を返し，そうでない場合は，偽を返す．XCL 型式は，XCL 論理式や限量子付 XCL 式中にも出現できるので，XML Schema による型チェックよりも表現能力が大きい．

【代入解釈】定義されない．

(3) XCL 論理式

【チェック解釈】XCL 論理式は，命題論理式と同様の解釈を用いて，真偽値を返す．

【代入解釈】「$\Phi \wedge \Psi$」は，XCL 式 Φ を実行した後に，XCL 式 Ψ を実行する．XSLT の逐次文に対応する．「$\Phi \vee \Psi$」と「$\neg\Phi$」は，代入解釈の対象外とする．「$\Phi \Rightarrow \Psi$」は，XCL 式 Φ をチェック解釈したときに真であるときのみ，XCL 式 Ψ を実行することと定義する．XSLT の if 文に対応する．

(4) 全称 XCL 式

【チェック解釈】「$\forall$`x` $\in$ `s`. Φ」は，`x` の出現を `s` が指し示すノード `n` に置換した式 XCL 式 Φ が，すべてのノード `n` に対して真を返す場合に真を返し，それ以外の場合は偽を返す．

【代入解釈】「$\forall$`x` $\in$ `s`. Φ」は，`s` が指し示すノード `n` ごとに，`x` の出現を `n` に置換して得られる XCL 式 Φ を実行する操作と定義する．XSLT の for-each 文に対応する．

(5) 存在 XCL 式

【チェック解釈】「$\exists$`x` $\in$ `s s.t. (count(.)=n)`. Φ」は，`s` が指し示すノード集合の要素のうち，パス制約式 Φ が真である要素の数が `n` に等しいときのみ真を返し，それ以外の場合は偽を返す．出現数 XCL 式「`count(.)=n`」は，XML Schema における要素出現数をチェックする属性 minOccurs や maxOccurs に相当するが，右辺の `n` が変数や計算式などを記述できるので，XCL 式の方が記述能力が大きい．

【代入解釈】「$\exists$`x` $\in$ `s s.t. (count(.)= n)`. Φ」は，「`count(s) = n`」を満たすように XML 文書を操作した後で，XCL 式 Φ を実行する．「`count(s) = n`」の部分は，XCL 式 Φ を満たすように，XML 文書中のノードを追加／削除する操作であり，DOM-API のノード操作メソッド（Node クラスの appendChild や removeChild）に相当する．

3.4 XML 文書ルール記述言語 DRDL

XML 文書ルール記述言語 DRDL では，3.3 節で述べた XCL 式をベースとして，応用上有用な「ベクトル演算」，「RDB とのマッピング定義」，「XML 文書中のコンテキスト指定」，および「XPath の計算関数の併用」などの機能を拡張した．以下，順に説明する．

(1) ベクトル演算（ベクトル和・差，スカラ積，内積）

XML 文書中の内容間制約検証でよく用いられる反復要素全体に対する演算として，ベクトル和・差，スカラ積，内積を追加した．

`s+t` は，XCL 項 `s` と `t` が指し示すノードリストの要素を，XML の出現順に並べたベクトルと見なして，ベクトルの和と差を計算した結果を返す．ベクトル和や差は，反復要素どうしで同じ順序の値ごとに和や差をとる場合に用いる．同様に，XCL 項を評価した結果をベクトルと見なして，スカラ積と内積をとる演算を追加した．

(2) RDB とのマッピング定義

2.1.1 項 (4) で述べた XML-RDB 双方向写像制約を処理するために，「XML 文書の RDB 登録」や「RDB の検索結果の XML 文書への取り込み」を記述するための構文を追加した．

(3) XML 文書中のコンテキスト指定

XPath 式のカレントノードを XPointer で指定する構文の追加により，XCL 式を簡潔に記述できるようにした．XPointer を用いるので，複数 XML 文書間の内容制約を記述できる．

(4) XPath や XSLT の組み込み関数の利用

反復要素の合計値を計算する sum や，文字列を連結する concat などの XPath や XSLT の組み込み関数を利用できるようにした．

3.5 DRDL の記述例

本節では，2.1.1 項で述べた「内容間制約」，「内容・構造間制約」，「依存型制約」，および「XML-RDB 双方向写像制約」の DRDL による記述例を示す．

(1) 内容間制約の例

図 1 に示す XCL 式は，**図 2** のような入力 XML 文書を想定して，『「小計」は，「購入品」の「価格」の合計値と等しい』という制約を表現している．チェック解釈を用いれば，内容チェック機能を実現できる．また，代入解釈を用いれば，表計算ソフトウェアのセル値間計算に相当する編集支援機能を実現できる．

(2) 内容・構造間制約の例

図 3 に示す XCL 式は，**図 4** のような XML 文書を想定して，『要素「装置」の出現数は，要素「装置数」の内容に等しい』という制約を表現している．

チェック解釈を用いれば内容チェック機能を実現できる．また，代入解釈を用いれば，“appendChild” や “removeChildDOM” などの DOM-API を用いて実

```
∀ x ∈ //購入品リスト. x/小計 = sum(x/購入品/価格)
```

図 1 内容間制約の記述例
Fig. 1 Cross-field constraint with DRDL.

```
<購入品リスト 購入日＝"2002-10-01">
 <購入品><名称>PC</名称><価格>95000</価格></購入品>
 <購入品><名称>Disk</名称><価格>50000</価格></購入品>
 <小計> </小計>
</購入品リスト>
<購入品リスト 購入日＝"2002-11-01">
 <購入品><名称>note</名称><価格>200</価格></購入品>
 <購入品><名称>pen </名称><価格>100</価格></購入品>
 <小計> </小計>
</購入品リスト>
```

図 2 処理対象 XML 文書の例
Fig. 2 An input XML document.

```
∃ x ∈ //装置 s.t.( count(.)= //装置数 )
```

図 3 構造・内容間制約の記述例
Fig. 3 Cross-field constraint with DRDL.

```
<装置数> 5 </装置数>
<装置リスト>
   <装置> ... </装置>
   <装置> ... </装置>
</装置リスト>
```

図 4 処理対象 XML 文書の例
Fig. 4 An input XML document.

```
∀x ∈ //商品. ( x/カテゴリ = “通信”) ⇒ (x/番号：通信機器コード)
```

図 5 型制約式の例
Fig. 5 Cross-filed-type constraint with DRDL.

現するような要素の追加や削除を実現できる．図 4 の XML 文書を入力として，図 3 に示す XCL 式を代入解釈で実行すれば，『要素「装置」を 3 つ追加して，要素「装置」の出現数を要素「装置数」の内容である 5 と等しくなるようにする』という処理が実行される．この処理を XML 文書入力支援ソフトウェアから呼び出すことにより，XML 文書中のある記載項目の内容に応じて，他の記載項目の入力枠数を可変にするような動的な入力フォームを実現できる．

(3) 依存型制約

XPath 型制約式を用いることにより，XML Schema では表現できない「ある要素内容に依存したデータ型」を定義できる．**図 5** は，『要素「商品」の子要素「カテゴリ」が「通信」の場合には，「番号」の型は「通信機器コード」でなければならない』という制約を表現している．ただし，ここで「通信機器コード」は，XML Schema のデータ型として定義されているものとする．

```
01: ∀ x ∈outputfile.xml!設備情報/機器ID .
02: (( y = select DATE, DENATSU from kikitenken where KIKINO = '$x'")
03:   (∃z ∈x/点検情報リスト/点検情報  s.t.(count(.)=count(y/Row))  .
04:      ( z/点検日 = y/ROW[position(z)]/DATE        ∧
05:        z/電圧 = y/ROW[position(z)]/DENATSU ) ) )
```

図 6 XML-RDB 対応制約の例
Fig. 6 RDB mapping constraint with DRDL.

```
<設備情報>
    <機器ID>0001</機器ID>
    <点検情報リスト>
        <点検情報>    </点検情報>
    </点検情報リスト>
</設備情報>
<設備情報>
    <機器ID>0002</機器ID>
  ...
</設備情報>
```

図 7 入力 XML 文書の例
Fig. 7 An input XML document.

```
<設備情報>
   <機器ID>0001</機器ID>
   <点検情報リスト>
      <点検情報>
         <点検日>04-02-01</点検日>
         <電圧>2000 </電圧>
      </点検情報>
      <点検情報>
         <点検日>04-02-02</点検日>
         <電圧> 3000</電圧>
      </点検情報>
   </点検情報リスト>
</設備情報>
<設備情報> ... </設備情報>
```

図 8 出力例
Fig. 8 An output XML document.

```
(a) //a = //b + //c
(b)∀x ∈ //a. ( n=position(x) ∧ ( //a[n] = //b[n] + //c[n] ) )
```

図 9 ベクトル演算を用いた制約式の例
Fig. 9 Vector expression with DRDL.

(4) XML-RDB 対応制約

図 7 の XML 文書を入力として，**図 6** の XCL 式を代入解釈で実行することにより，RDB の検索結果が，XML 文書中の該当する機器 ID の点検日と電圧に書き込まれる．具体的には，機器 ID ごとに 02 行目の SQL 文を実行し中間結果を変数 `y` に格納した後，03 行目から 05 行目で該当する箇所に点検日と電圧の値を挿入し，**図 8** の結果を得る．

(5) ベクトル演算

ベクトル演算を用いれば，反復出現する複数要素間の計算を簡潔に表現できる．**図 9** の式 (a) と式 (b) は，いずれも，要素 `a`，`b`，`c` が反復出現する XML 文書において，『「`n` 番目の要素 `a` の値」が，「`n` 番目の要素 `b` の値」と「`n` 番目の要素 `c` の値」の和に等しいこと』を示している．ここで，`position(x)` は，全称記号で束縛されるノード `x` を与えると，XCL 項（本例では，`//a`）の評価結果として得られるノードリスト中での出現順序数を返す関数である．

4. DRDL に基づく XML アプリケーション開発フレームワーク

本章では，2.1.2 項で述べた「XML 文書処理機能の開発・保守効率化への要求」に応えるために開発した DRDL フレームワークについて述べる．

4.1 DRDL フレームワークの全体構成

DRDL フレームワークは，XML 文書ルールを記述した XML 文書である DRDL ファイル，DRDL ファイルを作成するための DRDL エディタ，DRDL ファイルを解釈実行することにより XML 文書内容処理を実行する DRDL プロセッサ，および XML 文書ルールを既存の XML ツール用の記述形式に変換する DRDL ルールトランスレータからなる（**図 10**）．

以下，順に説明する．

4.2 DRDL ファイル

2.1.2 項 (1)(iii)「文書ルール差し替えの容易さ」を実現するために，アプリケーションプログラムから独立したファイル（DRDL ファイルと呼ぶ）として文書ルールを記述できるようにした．この DRDL ファイルを伝送・差し替えするだけで，クライアント側の XML 文書入力ツールにおける内容処理機能を更新したり，また，サーバ側の XML 文書変換機能を更新したりできる．

DRDL ファイルは，3 章で述べた DRDL の個々の XCL 式を XML の構文で表現したものである．**図 11** に，図 1 の XCL 式の XML 構文での表現例を示す．

4.3 DRDL エディタ

2.1.2 項 (1)(i)「文書ルール作成の容易さ」を実現するために，XML，Java，Web などの特別な専門知識を持たない業務部門担当者でも XCL 式を作成できる DRDL エディタを開発した．DRDL エディタは，**図 12** に示すようなテーブル形式の GUI を持ち，左部分の文書スキーマ記述部を常時表示させながら，右部分の文書内容制約記述部や XML-RDB 関係部を切り替えながら，XML 文書ルールを編集できるようにした．

以下，文書スキーマ記述部，文書内容制約記述部，XML-RDB 関係部について説明する．

(1) 文書スキーマ記述部

DTD や XML Schema で記述できる XML 文書の構造やデータ型を指定する部分である．市販の DTD や XML Schema 設計支援ツールとの併用を可能にするために，文書スキーマ記述部は，XML Schema をインポートできるようにした．

(2) 文書内容制約記述部

2.1.1 項で述べた「内容間制約」と「内容・構造間制約」を記述する．これらの制約を簡単に記述できるように，以下の特徴を持つ簡易 DRDL 言語を開発した．

(i) if 文，代入文，× や + などの算術演算などの既存のプログラミング言語で使い慣れた構文で XML 文書ルールを入力できるようにした．

(ii) 反復出現する要素は，自動的に全称記号やベクトル演算などを補って解釈するマクロを導入した．

(iii) 対象 XML 文書の構造を自由に定義できる場合は，末端の要素名を一意とすることにより，内容間制約を記述する際にパスの文脈を指定しなくてよいようにした．

(iv) 「要素の反復数に関する制約」や「要素への値の代入規則」の指定用に列を設けることにより，XCL 式を意識させずに，行の左端に示す XML 要素に対する設定と自然に解釈できるようにした．

(v) ある要素の内容に依存して，別の要素がとりうる値の選択肢が可変となる制約式を編集する際は，

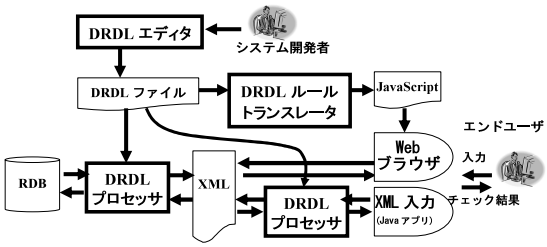

図 10 DRDL フレームワークの全体構成図
Fig. 10 Architecture of DRDL framework.

```
<all-path bind-val=" x"  path="//購入品リスト">
   <path l-path="{$x}/小計" op="eq" r-path="sum(./購入品/価格)">
</all-path>
```

図 11 XML 構文による XCL 式の記述
Fig. 11 A DRDL formula with XML syntax.

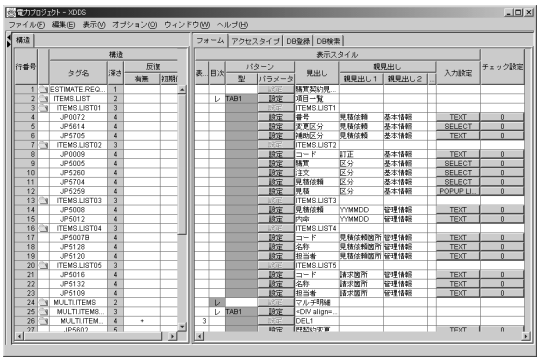

図 12 DRDL エディタ
Fig. 12 A DRDL Editor.

アプリケーションごとのパターンに応じた GUI を用意した．

(3) XML-RDB 関係制約記述部

XML 文書のタグごとに，複数の RDB のフィールド名を指定できるようにした．1 つの対応指定ごとに，対応するテーブル名，フィールド名，データ型，サイズ，主キー，外部キー，検索条件，および一覧表示時のラベル名などを指定する．

4.4 DRDL プロセッサ

2.1.2 項 (2)「他機能との連携の容易さ」を実現するために，他のプログラムと DOM 木を共有メモリとして共有しながら協調的に XML 文書を処理できるような DRDL 処理系（DRDL プロセッサ）を開発した．DRDL プロセッサは，アプリケーションから，API を介して呼び出され，処理対象 XML 文書に対して，DRDL ファイルを解釈実行した結果を返す．また，呼び出し API の引数，または，DRDL ファイル中の XCL 式ごとの設定により，「チェック解釈」と「代入解釈」のいずれかを選択できる．

また，数値計算処理，テキスト処理，バイナリデータ処理など DRDL では記述できない処理，あるいは無理に DRDL を用いて記述すると生産性が落ちる処理を実現する場合には，Java などで記述した文書内容処理プログラムを呼び出しながら，アプリケーション特化の文書内容処理を実現できるようにした．

4.5 DRDL ルールトランスレータ

2.1.2 項 (1)(ii)「文書ルール再利用の容易さ」を実現するために，XML 文書ルールを既存の XML ツール用の記述形式に変換する DRDL ルールトランスレータを開発した．本トランスレータにより，Web ブラウザ上での XML 文書入力画面の内容チェック用 JavaScript を XCL 式から自動生成することができる．自動生成された JavaScript は，文献 14) で述べた方式で生成された XML 入力用 XSLT スタイルシートと併用することで，Web ブラウザでの XML 文書入力用チェックに利用できる．

5. 評　　価

本章では，4 章で述べた DRDL フレームワークを実用アプリケーションシステムに適用した結果を報告することにより，DRDL の実用性を示す．

5.1 DRDL を利用したアプリケーションシステム

本節では，DRDL を適用したシステム例として，電子申請システム，設計仕様交換システム，設備情報管理システム，および WebEDI システムを紹介する．

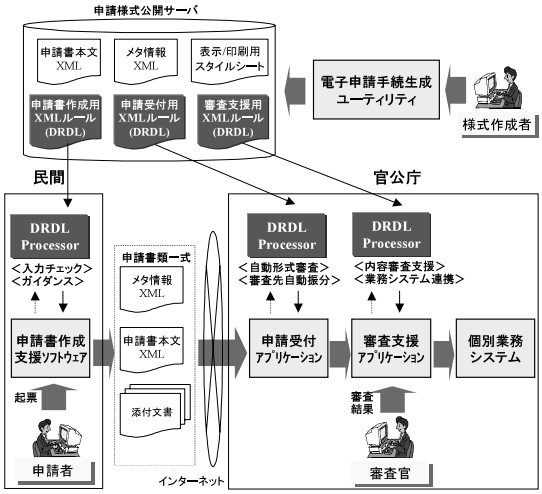

図 13 電子申請システムへの適用例
Fig. 13 Electronic application system with DRDL framework.

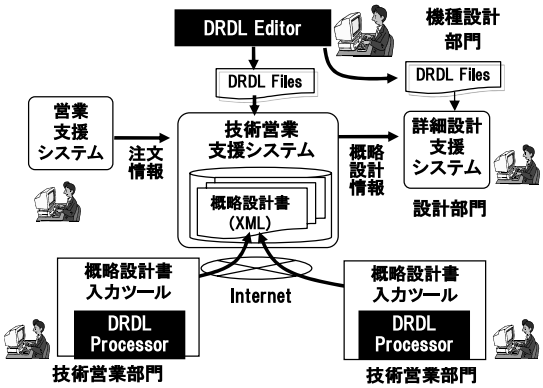

図 14 設計仕様交換システムへの適用例
Fig. 14 Design specification exchanging system with DRDL framework.

(1) 電子申請システム

図 13 に，提案方式の電子申請システム[15),16)] への適用例を示す．本システムでは，民間会社からインターネットにより，XML 形式の申請書が官公庁に送付される．ここで，DRDL プロセッサは，申請書の官公庁規定の文書規約への適合性を検証するために用いられている．提案方式により，メタ情報（申請書類一式自体に関する管理情報），申請書，および添付文書などからなる申請書類一式に対する検証や代入処理を実現できる[17)]．

(2) 設計仕様交換システム

図 14 に，グローバルな企業連携による共同設計／開発を容易にするためのエレベータ設計仕様交換システムへの適用例を示す[18)]．本システムでは，最初に，機種設計部門が，概略設計書用の設計項目やチェック

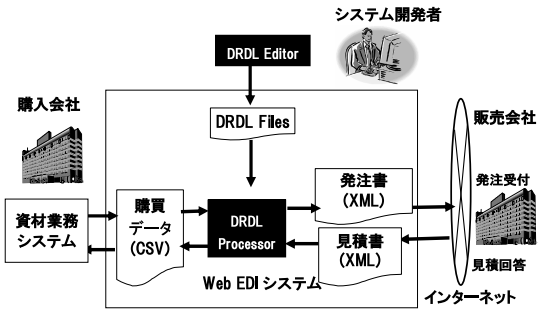

図 15 WebEDI システムへの適用例
Fig. 15 WebEDI system with DRDL framework.

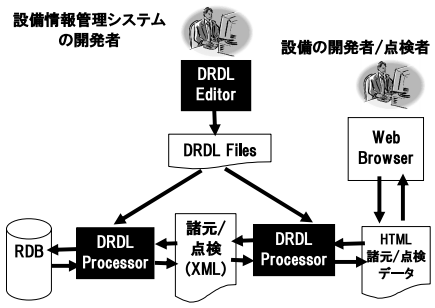

図 16 設備管理システムへの適用例
Fig. 16 Equipment management system with DRDL framework.

ルールなどの機種ごとのマスタ情報を記述した DRDL ファイルを作成し，技術営業支援システムに登録する．次いで，技術営業部門は，概略設計書入力ツールにより概略設計書（XML 形式）を作成し，インターネットにより技術営業支援システムに伝送する．最後に，概略設計書はデータ変換され，機械，電気，意匠などの詳細な設計作業を支援する詳細設計支援システムへ伝送される．

DRDL プロセッサは，概略設計書入力ツール上で，DRDL ファイルを解釈実行することにより，機種ごとの入力画面生成，仕様チェック，および工期自動判定の諸機能を提供する．また，技術営業支援システム上では，XML 形式の概略設計書をレガシーな設計支援システム用の入力形式へのデータ変換機能を提供する．

(3) WebEDI システム

図 15 に，Web ブラウザを用いて受発注を電子化するシステムへの適用例を示す．本システムでは，インターネット上で XML 形式の見積書，発注書，および検収書を物品の購入会社と販売会社側との間で交換する．

DRDL プロセッサは，「資材業務システムとのデータ連携のための XML/CSV 双方向変換」に利用されている[19]．また，DRDL トランスレータは，「Web ブラウザ上での XML 文書入力のための JavaScript による入力チェックプログラム生成」に利用されている．

(4) 設備管理システム

図 16 に，設備を構成する機器の仕様を記載した設備台帳や，機器の保守点検記録などの設備情報を XML 形式で管理するシステムへの適用例を示す[20]．大規模設備では，その設備を構成する機器類の種別が多くなるだけでなく，種別ごとに管理すべき仕様項目が異なる．そのため，設備台帳データベースのデータ設計の手間が大きくなるという問題があったが，DRDL フレームワークを用いて設備台帳を XML 化することにより，データ設計の手間を小さくできるだけでなく，インターネット環境で設備台帳を閲覧することが容易になる．DRDL プロセッサは，HTML 入力フォーム中の諸元／点検データを XML 形式に変換し，設備管理 DB に登録するために用いられている．

5.2 アプリケーションシステムにおける評価

本節では，2.1 節で述べた課題ごとの評価結果について述べる．

(1) XML 文書の内容処理の表現能力（2.1.1 項）

2.1.1 項の「(1) 内容間制約」，「(2) 内容・構造間制約」，「(3) 依存型制約」，および，「(4) XML-RDB 双方向写像制約」の記述能力については，実システムで必要となる内容チェックや内容代入処理をカバーしていることを，5.1 節で述べたアプリケーションシステムの開発を通じて確認した．ただし，設計仕様中のバイナリデータに対する検証や，資材帳票における複雑な業務依存処理では，4.4 節で述べた DRDL の外部手続呼び出し機能を用いた．

2.2.1 項で述べた従来のスキーマ言語では記述しにくかった内容処理の代表的な記述例としては，以下のものがある．その他の例については，文献 15)〜20) に譲る．

- 輸出入の許認可を申請する電子申請システムでは，本文中に記載される「輸出対象の商品種別」と「輸出先の地域分類」によって異なる許認可に必要な添付文書の有無を確認するような複数文書間の内容間制約のチェックを実現している．
- 昇降機の設計書を交換する設計仕様交換システムでは，昇降機の「かごの数」や「ビルの階数」に応じて，「かごのサイズ」，「ドア」，および「停止階」の記載項目の数が変化するという内容・構造間制約のチェックを実現している．
- 上下水道の設備機器を管理する設備管理システムでは，設備機器の種別（受電設備，電気設備，建築物，土木建造物，計装ループ，ポンプ設備，場

表 1 1 ルールあたりの開発時間の比較
Table 1 Comparison in working hours of developing document rules.

制約の種別	DRDL	Java	改善比率
内容間制約	70 sec	182 sec	2.6 倍
内容・構造間制約	46 sec	142 sec	3.1 倍
依存型制約	68 sec	180 sec	2.6 倍
平均	61 sec	168 sec	2.7 倍

表 2 コード行数の比較
Table 2 Comparison in the quantity of rule descriptions.

制約の種別	DRDL	Java	改善比率
内容間制約	2.5 行	16.5 行	6.6 倍
内容・構造間制約	2.0 行	15.0 行	7.5 倍
依存型制約	3.0 行	17.3 行	5.8 倍
平均	2.5 行	16.3 行	6.5 倍

内配管，弁類など）によって管理項目が異なるといった依存型制約の内容チェックを実現している．また，様々な種別の設備機器が混在する設備台帳データを，設備機器の共通項目と非共通項目を分けて管理する RDB に自動登録する XML-RDB 双方向写像制約の処理を実現している．

(2) 文書ルール作成の容易さ（2.1.2 項 (1)(i)）

2.1.2 項 (1)(i) の「文書ルール作成の容易さ」における「作成時間の短縮」の効果を評価するために，昇降機の設計仕様交換システムでの仕様チェックルールを対象として，記述言語として簡易 DRDL 言語と Java を用いた各々の場合の開発時間とコード行数を比較した．評価対象のチェックルールは，1 機種あたり約 300 ある仕様チェックルールから，内容間制約，内容・構造間制約，および依存型制約ごとに典型的なものを 5 つ選択した．また，被験者は，Java と XML の両者のプログラム開発経験のある 4 名とした．

被験者 4 名の 1 ルールあたりの開発時間の平均値を**表 1** に示し，コード行数の平均値を**表 2** に示す．制約の種類ごとにばらつきはあるが，開発時間は平均 2.7 倍，コード行数では平均 6.5 倍の向上がみられた．3.5 節 (2) で例示した内容・構造間制約ルールに対して被験者の作成した DRDL 簡易記述と Java プログラムの例を，**図 17** と**図 18** とに各々示す．

文書作成の容易さにおける「作成者の拡大」については，従来，情報システム部門が実施していた文書ルール作成を，機種仕様を作成する技術部門が実施できたことにより確認した．文書ルール作成業務を，情報システム部門から技術部門に移管することにより，両部門間で文書ルールを伝達／確認する作業が不要に

```
count(//装置) = //装置数
```

図 17 DRDL 簡易記述例
Fig. 17 Sample code of simple DRDL.

```
public Result validate (Document doc) {
   try { nodelist list = getElementList(doc, "//装置");
         int len1 = list.size();
         String txt2 = getText(doc, "//装置数");
         int len2 = Integer.parse(txt2);
         if (len1 != len2) { return new Result(false);}
           else { return new Result(true);}
   } catch (Exception ex) { ex.printstacktrace();} }
```

図 18 Java プログラム例
Fig. 18 Sample code of Java.

なり，「新機種投入や機種マイナー変更時に，システム機能を迅速に変更可」と「伝達ミスの削減」という効果が得られた．

(3) 文書ルール再利用の容易さ（2.1.2 項 (1)(ii)）

2.1.2 項 (1)(ii) の「組織ごとの業務システムに応じて，文書ルールを容易に再利用できること」については，設計仕様交換システムで効果を確認した．具体的には，DRDL ルールトランスレータを用いて，Java アプリ用に開発した入力チェック用の DRDL 記述から，Web 画面用の入力チェック JavaScript プログラムを自動生成できた．この自動生成機能により，Web 版から Java アプリ版（リッチクライアント）へのシステム移行開発の効率を向上させることができた．

(4) 文書ルール差し替えの容易さ（2.1.2 項 (1)(iii)）

2.1.2 項 (1)(iii) の「文書ルール差し替えの容易さ」については，電子申請システムと設計仕様交換システムで確認した．文書ルールがプログラム中に埋め込まれている場合には，プログラムの修正や他プログラムとの整合性検証などのソフトウェア試験のために，文書ルール変更にともなう改修コストが大きくなる．一方，DRDL フレームワークでは，文書ルール変更時には，DRDL ファイルを差し替えるだけでよいので，文書ルールの差し替えが容易になる．

(5) 他機能との連携の容易さ（2.1.2 項 (2)）

2.1.2 項 (2) の「他機能との連携の容易さ」については，他のアプリケーションと DOM 木を共有することで実現している．電子申請システムと設計仕様交換システムにて，Java アプリケーションとして実装された XML 入力画面ソフトウェアと DOM 木を共有しながら，文書編集や検証が可能であることを確認した．DOM 木を用いた密なデータ連携は，ファイル入出力をともなう疎な連携では速度性能が出せない場合に有効である．さらに，設計仕様交換システムにおいては，業務に依存したバイナリデータを処理する Java プロ

グラムとの連携により，後工程の工事設計支援システム用のデータ変換が実現可能なことも確認した．

6. 考　　察

本章では，XML 文書ルール記述の理論と実用との関係について，「ルールの表現能力と作成容易さのトレードオフ」，「ハードコーディングしたプログラムとの実行性能の比較」，および，「素性構造ユニフィケーションの XML 文書処理への応用」について考察する．

(1) ルールの表現能力と作成容易さのトレードオフ

テキスト処理や数値処理を含む複雑なロジックの実装では，Java や C などの通常のプログラミング言語を用いることが多い．そこで，DRDL の設計にあたっては，「大規模ないし複雑な業務ロジックは別途実装したモジュールとの連携処理を想定し，エンドユーザが作成／保守する XML 特有の内容チェックや内容代入操作を簡潔に記述可能」と「ルール自動変換が容易な単純な構文と意味を持つ制約記述言語」という特徴を持つ XML スクリプト言語の実現を目標にした．そのため，XCL 式から自然に導き出せない While 文は，我々の応用上必要がなかったこともあり，あえて追加しなかった．

(2) ハードコーディングしたプログラムとの実行性能の比較

本稿で提案した XML 文書処理方式は，文書ルールを記述した DRDL ファイルを解釈実行するインタプリタ方式なので，同等の処理をハードコーディングした Java プログラムと比較すると，処理速度やメモリ消費などの実行性能は劣ったものになる．実際，初期の DRDL プロセッサの場合，電子申請や設計仕様交換の文書例では，処理速度性能は約 1/2 倍程度，また，メモリ消費量は約 2 倍程度であった．この問題を改善するには，本質的には，ルール記述から Java などのプログラムを生成するコンパイラを開発する必要がある．しかし，今回適用したアプリケーションシステムの開発では，処理時間の大半を占める XPath 処理の高速化により速度性能を 2 倍以上改善することで，実用上の要求性能を達成できた．

(3) 素性構造ユニフィケーションの XML 文書処理への応用

本稿では，「できるだけ多くの人が使えるようにする」と「実用的な速度性能を得る」という応用からの要求を満たすために，DRDL の意味としては，Java や C といった手続き型言語と類推が利くように「チェック解釈」と「代入解釈」を扱った．しかし，理論的には，XCL 式の充足可能性の判定手続きを与えることにより，XCL 式間のユニフィケーションを定義することができる[4),21)]．ここで，ユニフィケーションとは，2 つの XCL 式が与えられたとき，その式に含まれる変数に対して適切な代入を行うことで，2 つの式を同一の表現にする演算である．XCL 式のユニフィケーションの応用の可能性として，以下の 2 点がある．

(i) スキーマの上位互換性の判定

XML 文書が XML Schema を満たすかどうかの妥当性判定は，XML Schema の制約を XCL 式で表現することにより，XCL 式の充足可能性判定により実現できる．また，スキーマの上位互換性の判定（前版の XML Schema を満たすすべての XML 文書は，次版の XML Schema を満たすことの判定）は，XCL 式の包摂関係の判定により実現できる．

(ii) XML 文書処理用の制約論理型言語

XML 文書を項とするような論理型言語を定義できる．つまり，Prolog における項を XCL 式で置き換えることにより，XML 文書を処理する制約論理型言語を定義できる．この言語は，XSLT とは別のパターン記述に基づく変換言語になる．

7. おわりに

W3C 勧告である XPath のロケーションパスを項として，論理演算（連言 $\wedge$，選言 $\vee$，否定 $\neg$），限量子（$\forall$，$\exists$），等式，およびデータ型制約を持つ一階述語論理の部分言語である XML 文書ルール記述言語 DRDL を提案した．CALS/EC での XML 文書ルール記述と自然言語分野での素性制約との本質的な差異は，反復出現する素性名（XML では，要素名）を扱う点にあり，DRDL では，反復要素の内容がある等式を共通に満足するなどの制約を表現するために全称限量子 $\forall$ を用い，また，反復要素の出現数に関する制約を表現するために存在限量子 $\exists$ を用いることにより対処した．また，DRDL の論理式の操作的な意味として，チェック解釈と代入解釈を定義した．チェック解釈により，XML Schema では記述しきれない内容間制約や内容・構造間制約の検証を実現でき，代入解釈により，DOM-API を用いたノード追加／削除やノードへの値代入や，XSLT の if 文や for-each 文に相当する制御構造を実現できる．そして，電子申請，設計仕様交換，および設備管理などの実用のインターネット文書交換システムへの適用を通じて，DRDL エディタ，DRDL プロセッサ，および DRDL ルールトランスレータからなる DRDL フレームワークが，文書ルールの作成・再利用・差し替え，および他機能連携の容易さなどの従来の XML 文書処理ツールの課題

を解決していることを確認した.

参 考 文 献

1) Bray, T., Paoli, J. and Sperberg-McQueen, C.M.: Extensible Markup Language (XML) 1.0 (1998). http://www.w3.org/TR/1998/REC-xml-19980210
2) Thompson, H.S., et al.: XML Schema Part 1, Structures (2000). http://www.w3.org/TR/xmlschema-1/
3) Biron, P.V. and Malhotra, A.: XML Schema Part 2, Datatypes (2000). http://www.w3.org/TR/xmlschema-2/
4) Daum, B.: Validation beyond XML Schema, *Modelling Business Objects with XML Schema*, pp.323–362, Morgan Kaufmann Publishers, Heidelberg, Germany (2003).
5) Wood, L., et al.: Document Object Model (DOM) Level 1 Specification Version 1.0 (1998). http://www.w3.org/TR/REC-DOM-Level-1/
6) Clark, J. and Murata, M.: RELAX NG Specification (2001). http://www.oasis-open.org/committees/relax-ng/spec-20010811.html
7) Jelliffe, R.: The Schematron (2004). http://xml.ascc.net/schematron/
8) Clark, J. and DeRose, S.: XML Path Language (XPath) Version 1.0 (1999). http://www.w3.org/TR/1999/REC-XPath-19991116
9) Clark, J.: XSL Transformations (XSLT) Version 1.0 (1999). http://www.w3.org/TR/xslt
10) Shieber, S.M.: An Introduction to Unification-based Approaches to Grammar, *CSLI Lecture Notes*, No.4, Stanford University (1986).
11) Smolka, G.: A Feature Logic with Subsorts, *LILOG Report33*, IBM Deutschland, Stuttgart, F.R.G. (1988).
12) Johnson, M.: Attribute-Value Logic and the Theory of Grammar, *CSLI Lecture Notes*, No.16, Stanford University (1988).
13) 今村 誠：素性構造の単一化，情報処理，Vol.32, No.10, pp.1070–1078 (1991).
14) 今村 誠，森口 修，大場雅之，鈴木克志，依田文夫：木・表構造間写像モデルに基づく XML 入力画面自動生成方式，情報処理学会論文誌，Vol.46, No.12, pp.3066–3077 (2005).
15) 今村 誠，富川直毅：電子政府における XML 利用技術の動向，情報処理，Vol.42, No.7, pp.654–662 (2001).
16) 居駒哲夫，池田健一郎，今村 誠，秋間孝道：電子政府を実現する IT ソリューション，三菱電機技報，Vol.76, No.9, pp.22–27 (2002).
17) 今村 誠，長浜隆次，鈴木克志，渡部明洋：電子申請における XML 文書利用技術―電子政府実現に向けた外為法 EDI システム（JETRAS）への適用，情報処理学会デジタルドキュメントシンポジウム 2000 (2000).
18) 今村 誠，森口 修，鈴木克志：XML 文書ワークフロー構築方式，情報処理学会デジタルドキュメント研究会 27-1, pp.1–8 (2001).
19) 今村 誠：データ交換・共有技術，IT が拓く電力ビジネス革命，6-4 節「データ交換・共有技術」，pp.171–181 (2001).
20) 外崎道夫，道行泰代，南部雅彦，今村 誠：上下水道維持管理支援 ASP 情報サービス，三菱電機技報，Vol.76, No.10, pp.19–22 (2002).
21) Mukai, K.: Constraint Logic Programming and the Unification of Information, Doctoral Dissertation, Dept. of Computer Science, Faculty of Engineering, Tokyo (1991).

(平成 17 年 6 月 15 日受付)
(平成 18 年 1 月 6 日採録)

今村 誠（正会員）

1984 年京都大学工学部数理工学科卒業．1986 年同大学大学院修士課程修了．同年三菱電機（株）入社．自然言語処理と文書処理の研究開発に従事．現在，同社情報技術総合研究所音声・言語処理技術部言語処理チームリーダ．

渡邉 圭輔

1991 年東京工業大学工学部情報工学科卒業．1993 年同大学大学院修士課程修了．同年三菱電機（株）入社．音声認識，音声対話，文書処理の研究開発に従事．日本音響学会，日本認知科学会各会員．

増塩 智宏（正会員）

2000 年東京工業大学大学院総合理工学研究科修士課程修了．同年三菱電機（株）入社．文書処理，Web システムの研究開発に従事．

渡部　明洋

1969 年三菱電機（株）入社．火力・原子力発電システム，ミニコンピュータ，文書管理システム，CALS システム，電子申請システム，ブロードバンド（FTTH）のストリーミング配信システム等の開発に従事．現在，（株）栗菱コンピューターズの東京支社長．