社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DD－57 (1)
2006／9／29

コンポーネントベースの視覚的なXQuery構築支援環境

芦川 将之
(株)東芝 研究開発センター
知識メディアラボラトリー

田中 譲
北海道大学
知識メディアラボラトリー

概要

大量なXMLデータを効率的に使用するために、XMLデータベースに対する需要は急速に高まっており、その代表的な問い合わせ言語としてXQueryを挙げることができる。しかし、XQueryは多機能で記述能力に優れ、複雑な内容をデータベースのスキーマなどに制限されず記述することが可能だが、その多機能性、多様性からユーザが検索条件を記述するに当たって非常に難易度の高いものとなっている。本研究ではXQueryを構築するに当たって規範的なXQueryを定め、組み合わせることによって複雑なXQueryを簡易に構築する方法について検討する。またその方式を元にしたコンポーネントベースの開発環境を提供する。

Component-based Interactive Visual Query Environment for XML Databases

Masayuki Ashikawa
Toshiba Research and Development Center
Knowledge Media laboratory

Yuzuru Tanaka
Hokkaido University
Meme Media Laboratory

Abstract

XML is becoming a standard for representing and exchanging information contents. These query languages such as XQuery can describe complex queries without the need to specifying DB schemata of stored XML contents. This advantage, however, also works as a disadvantage for the ease of use. In order to give an integrated solution to these problems, this paper first proposes a canonical representation of queries in XQuery, then divides this representation form into component descriptions, and associates this decomposition and recomposition of a query to the query modification in XQuery to define XML views. This paper then proposes a visual component system in which each view and view component is represented as a visual software component.

1.はじめに

TX1[1][2]は(株)東芝によって開発されたネイティブXMLデータベースである。ネイティブXMLデータベースの特徴として、従来のrelational databaseとは異なりXMLデータをその構造のまま格納し、扱うことが出来るという利点がある。従来のrelational databaseにおいてXMLデータを格納する場合、XMLが持つ構造データをrelational databaseのスキーマへ対応するように変換する必要があり、この変換コストは非常に大きい。

TX1では検索言語であるXQuery[3]の持つ多機能性に着目し、XMLデータを取得する際の検索条件記述言語として採用している。XQueryはその記述能力において非常に多機能であり柔軟性を持つが、その反面文法が複雑であり習得が困難であるという欠点も併せ持つ。

本研究ではXQueryに関する知識が少ないユーザがどのようにすればXQueryを容易に記述できるかを検討し、その結果として得られた記述を容易にするためのGUIについて述べる。

本研究では、XQueryが持つ記述難易度の高さに対する解決法として、XQueryの記述の規範となるXQueryの記述条件(canonical XQuery)を定める方

法を提案している。この canonical XQuery はユーザに対してXQueryを用いた検索条件の記述を容易にするだけではなく、複雑な XQuery の記述を canonical XQuery の表現の集合として扱うことで、ユーザが XQuery の持つ検索の意味を容易に理解することが出来るようにする。

また、GUI にて行う XQuery 開発環境として、canonical XQuery の組み合わせ、またはパラメタ変更による XQuery 作製作業を視覚的操作にて行うことが可能な開発環境を提供する。

上記 GUI は IntelligentPad 技術[4]をベースとしている。ユーザは XQuery 用のコンポーネント、及び既存の IntelligentPad 技術が持つコンポーネントライブラリを組み合わせることによって、他のユーザが作製した既存のアプリケーションへの XQuery を用いた検索機能の組み込みや、新規アプリケーション構築などを容易に行うことが可能となる。

第 2 章では、XQuery 記述を容易にするための既存研究において調査、分類を行う。第 3 章では既存研究において問題点として指摘されている部分に対する本研究の解決方法を述べる。第 4 章では本研究において IntelligentPad 技術を用いることによって得られる利点に関して述べる。第 5 章では IntelligentPad 技術を用いた本研究の実装に関して述べる。第 6 章では本研究における XQuery 開発環境を用いて作製したアプリケーションの例を示す。第 7 章ではまとめと研究における今後の検討事項に関して述べる。

2.既存研究に関する調査

クエリ記述における簡易化に関しては既存研究が幾つか存在する。ここでは既存研究が提唱する手法を基に、それぞれをカテゴリに分類する。

1)XQuery の記述をサポートするためのツール

このカテゴリに分類される手法では、XQuery の文法に関するサポートを行うことで XQuery の記述を容易にしている。例として文法のチェック、文法記述時における関数名記述の補佐、デバッグ機能などが挙げられる。しかし、この方法ではユーザが XQuery の文法をある程度習得済みであることが前提であり、XQuery の知識を全く持たないユーザにおける XQuery 記述を容易にするという目的には適していない。このカテゴリに属するものとして QURSED[5]や Stylus[6]などが存在する。

2)サンプルの記述による検索条件指定

このカテゴリに分類される手法はユーザが取得したい XML の例を示すことで、取得条件を記述している。例をデータベースに与えることで、例と同じ構造、また条件として記述した制限事項を満たした結果を得ることが出来る。しかしこの方法ではユーザが例となる XML データを記述するに当たって、データベースに含まれる XML データ構造を前もって把握しておかなければならない。よってデータベースを構築したユーザと検索を行うユーザが同一ではない場合は、データベースを構築したユーザは検索を行うユーザに対してデータベースの構造情報を提示する作業コストが発生する。このコストは非常に大きい。このカテゴリに属するものとして Xing[9]、XQBE[10]、QBE[11]などが存在する。

3)オブジェクトによる XQuery 記述

このカテゴリに分類される手法は、XQuery の文法をユーザに意識させることなく、定めたオブジェクトを組み合わせることによって XQuery の構築を行うものである。本研究はこのカテゴリに分類される。既存の研究として FoxQ[14]が挙げられるが、FoxQ においては XQuery の文法においてサポートしているものが少なく、XQuery の持つ記述能力を制限していると言う問題がある。

本研究ではこれらの問題への解決法を検討し、ユーザの XQuery 記述を容易にする手法を提供することを目的とする。

3. 複雑な XQuery 記述のための対応策

本研究では XQuery の記述を容易にするために、規範となる XQuery(canonical XQuery)を定める。canonical XQuery は XQuery の記述能力に制約を加えた形式であり、他の XQuery を記述するための基礎的な部品として扱われる。ユーザはこの canonical XQuery を組み合わせることで複雑な XQuery を容易に記述することが可能となる。

また、canonical XQuery を組み合わせることで更に複雑な XQuery を表現することが可能である。この canonical XQuery の組み合わせによって表現された XML データを view と呼ぶ。view に関しては後述する。

3.1 query component

前述のように canonical XQuery は XQuery の一形式であり、その他のクエリの生成部品となる。しかし、canonical XQuery を直接記述するのは困難であるため、本研究では更に細分化した部品として、XQuery の持つ基礎的な機能をその意味に応じて 6 つのカ

テゴリへと分類を行った。本研究ではその分類を適用したオブジェクトを query component と呼ぶ。ユーザはこれらの query component を組み合わせることによって canonical XQuery を記述することが可能である。また、これらの query component はそれぞれ複数のパラメタを持つ。

以下に各 query component の詳細を述べる

1) Variable Binding

この query component は XML データベース上におけるユーザが取得したい XML データの位置情報を定義する。これは、XML データベース上における取得したい XPath を用いて XML データのルートノードの位置を示すことが必要である。しかし、直接 XPath をユーザが記述することは困難であるため、本研究の XQuery 開発環境においては GUI を用いて入力のサポートを行っている。

2) Skeletal Restriction

この query component はユーザが取得したい XML データの構造における構造条件を定義する。ユーザは通常の XML ファイル、XML データからこの query component を作製することが可能である。この query component はユーザが取得したい XML データの中に、作製時に用いられた XML データに存在するノードが存在することを保証する。

3) Value Restriction

この query component はユーザが取得したい XML データのデータ値条件を定義する。

4) Output Template

この query component はユーザが定めた Variable Binding や view を新しい構造へと変化させることが可能である。ユーザは Skeletal Restriction と同様に XML ファイル、XML データを基に、この query component を作製することが可能であり、この query component へ Variable Binding や view の名前を組み込むことで新しい構造の XML データを生成することが可能となる。

5) Relation Restriction

この query component は Output Template に組み込まれている view や Variable Binding 間の関係における制限を定義する。

6) Sort

この query component は結果として返される XML データの順番を定義する。順番決定条件は検索条件によって存在を保証されたノードの状態に対して条件判定を定めることで定義する。

3.2 view

view は以下のように定義される。

・query component を組み合わせて作製された canonical XQuery

・view を組み合わせることによって作製された canonical XQuery

3.2.1 view の materialize

view は canonical XQuery の集合として保存され、必要に応じて view から XQuery 文字列を生成し、XML データをデータベースから取得する機能を持つ。この機能による一連の作業を materialize と呼称する。materialize 作業は不可逆であり、materialize された結果からは元の view を生成することはできない。その為、view の利点を保持し、再利用率を高めるために view は materialize されていない状態で保存される。

3.2.2 view への参照

view は XQuery へと materialize することが可能であり、その XQuery を処理することで特定の XML データを得ることが出来る。よって view を XML データとして見なすことが可能であるため、view を XML データベースと同様に検索対象とすることが可能である。

図 1 は view 同士の参照関係を示した図である。view B は view A に対する検索条件を query component を組み合わせることによって記述した結果得られた view である。この場合 view B は view A に対して参照関係を持つ。

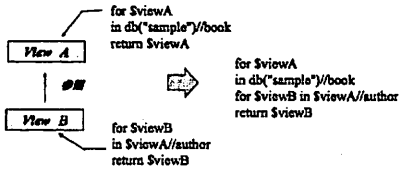

図 1 : view の参照

3.2.3 制約の伝達

前述のように view はその他の view から参照することが可能である。図 2 では view N から view T が生成された例である。ここで view T を materialize すると参照元の view N も連鎖的に materialize され、両方の view データから得られた情報を元に XQuery 文字列を生成する。ここで view N に対して Value Restriction

を追加した場合、view N は Value Restriction で追加された条件を満たす XML データの集合へと絞り込みが行う view へと意味を変える。view T は view N の XML データに対する view であるため、連鎖的に view T も Value Restriction の影響を受ける。即ち、ある view への制約追加を行うことによって、その view から派生した view すべてに一括して制約追加を行うのと同等の効果を得ることが可能である。

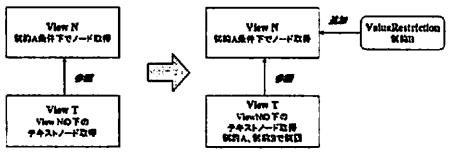

図 2：制約の伝達

4. IntelligentPad

IntelligentPad は Pad と呼ばれる数多くのオブジェクトから成り立つアプリケーション開発環境である。この Pad はそれぞれ異なる機能、データ、表現能力を持ち、それらを組み合わせることでアプリケーション開発者の要件を実現し、アプリケーションを構築することが可能である。IntelligentPad 技術を用いて本研究における XQuery 開発環境を実現することにより、以下の利点を得ることが出来る。

第一に、IntelligentPad は既に GUI のオブジェクト操作の概念を持っており、その概念を利用することが可能な点がある。IntelligentPad では Pad と Pad を組み合わせ、Pad の持つ slot と呼ばれるパラメタを変更することでアプリケーションを構築しており、それぞれを本研究における query component と query component を組み合わせ、query component が持つパラメタを変更することによる canonical query の構築へ適用することで、IntelligentPad の GUI 操作における概念にて視覚的に query component を用いて canonical query を構築することが可能となる。

第二に、ユーザが XQuery を構築するだけでなく、XQuery を用いたアプリケーションの構築が容易になる点がある。ユーザが求める XML データを取得するには XQuery を記述しただけでは足らず、その記述した XQuery をデータベースへと送り、結果を取得する必要がある。IntelligentPad はその為の通信機能や取得した結果データを管理するためのアプリケーション構築を容易に行うことが可能である。

第三に、IntelligentPad で作成された既存のアプリケーションへの親和性が高い点がある。データベースとの通信能力を持たない既存のアプリケーションに対しても、本研究の XQuery アプリケーションを組み合わせることで、容易に条件入力による検索能力を付加することが可能となる。

5. インターフェイスの実装

前記の IntelligentPad を用いた XQuery 開発環境の概念を元に、実際に開発環境の実装を行った。

5.1 IntelligentPad と query component

本研究では XQuery 構築の為の基本となる canonical query を構築するために 6 種類の query component を定義した。そしてそれらの query component を実際に IntelligentPad 環境で使用するために、query component が持つそれぞれの機能を有した Pad の実装を行った。さらに query component がオプションとして持つパラメタを pad の Slot を用いて実装をし、それらの Slot を組み合わせることによってそれぞれの Pad から、その他の Pad が持つ情報を利用することを可能とした。

更に、IntelligentPad を用いて作成した view を実際に処理するための materialize Pad、データベースへの検索 Pad、結果を処理、表示するための XML Pad などユーティリティ系の Pad の実装を行った。

5.2 GUI インターフェイスの実装

IntelligentPad 環境を用いた XQuery の構築を簡易に行う環境を提示したが、まだ query component の Slot におけるパラメタの入力や Pad 同士の組み合わせなど、IntelligentPad を用いてアプリケーション構築の経験がないユーザにとっては困難な作業が残っている。ここでは、それらのサポートを行うための GUI の提供に関して述べる。

しかし、GUI の提供においては、利便性を求めた機能の自動化に対し、既述できる XQuery に制限が生じる可能性がある。その利便性とアプリケーション構築のための多様性の間にはトレードオフの関係にある。それを前提に、それぞれのレベルに応じて以下及び図 3 にて、三種類の提供パターンを検討する。

1) query component pad のみ

ユーザは query component のそれぞれの機能を持つ Pad とそれらに関するユーティリティ系の Pad のみ使用が可能とする。最も基本的な Pad しか与えられないため、ユーザには Pad アプリケーション構築のための知識が必要とされるが細部にわたる複雑なアプリケーションを構築することが可能である。

2) Middleware Pad が使用可能

Middleware Pad はユーザの行動を予測し、利用可能性が高い一連の作業を行う機能を持ったアプリケーション構築用の Pad を提供するものである。パラメタの入力補佐など、ユーザにおける作業を軽減することが出来、また Middleware Pad 自体が IntelligentPad で構築されているため、その他のアプリケーションへの流用が可能であるなどの利点を持つ。Pad を用いたアプリケーション構築経験が少ないユーザは Middleware Pad のサポートを得ることで作業が容易になるが、必要な場所がすべて自動化されているわけではないので、全く知識がないユーザにとってはまだ難易度が高い。

3) フルサポート GUI の提供

IntelligentPad で構築された XQuery 構築のための GUI アプリケーションを提供する。ユーザはアプリケーションにおけるガイドラインに従い条件を既述することで容易に XQuery を構築することが可能だが、細部の変更が困難、結果を新しいアプリケーションへ転用することが困難などの問題がある。

本研究ではユーザに対して query component Pad などの基本となる Pad と、GUI アプリケーションの両方を提供することで、ユーザの持つスキルに応じた選択が可能なように環境を構築した。ここでの GUI アプリケーションはウィザード形式の XQuery 構築 GUI アプリケーションであり、提示されるガイドラインに沿って必要事項を選択、入力していけば結果として view を構築できる仕組みである。しかし、自由度が低いのでユーザの行動に制限が課せられる。この GUI アプリケーションの外観を図 4 にて示す。

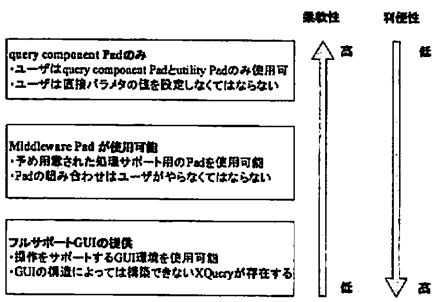

図 3:サポートレベル

6. XQuery アプリケーションの例

ここではXQueryアプリケーションの構築例を示す。

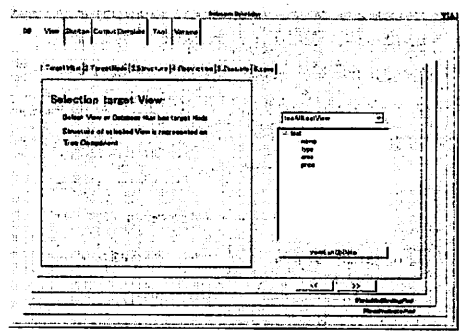

図 4:Wizard Application

また、ユーザの負担を軽減することを前提としているため、アプリケーション開発経験が少ないユーザを対象とする。

XQuery アプリケーションの例として PDB(Protein Data Bank : http://pdb.protein.osaka-u.ac.jp)のデータを XML 形式で XML データベースに登録し、ユーザの要望に応じた PDB データを XML データベースから取得するようなアプリケーションの構築について検討する。この PDB 検索アプリケーションはユーザから与えられた検索用のキーワードを基に XQuery を構築し検索要件を満たす XML データを取得する。

PDB 検索アプリケーション構築に当たって以下のことを前提とする。

・アプリケーション構築ユーザに関して

アプリケーション構築ユーザは XQuery に関する知識がなく、データベースプログラミングの経験も無い。XML、IntelligentPad に関する最低限の知識を持つ。

・アプリケーション使用ユーザに関して

アプリケーション使用ユーザは最低限のコンピュータを扱うための技術力(Microsoft word, Microsoft Excel を使用可能な程度)を持ち、検索を行う概念を持つ。

・データベース構築ユーザに関して

PDB データは XML 形式で既述されていないため、データを XML 形式に変換してから XML データベースへと格納する必要がある。図 5 はその例である。データベース構築ユーザはその為に必要な XML に関する知識を最低限持つものとする。

6.1 PDB 検索アプリケーションの機能

XQuery を構築するための検索条件をユーザが入

力するに当たって、そのサポートとなるインターフェイスを提供する。例として、検索キーワードとして使用可能であるキーワードを列挙し、選択することでそのキーワードを検索条件としたXQueryの構築を行うなどの方法がある。

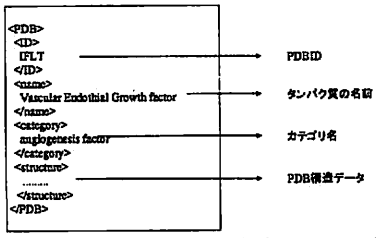

図5:PDBデータから作成したXML構造

6.2 アプリケーション構築

アプリケーション構築ユーザがアプリケーションを構築するに当たって、先ず必要なviewを定義する必要がある。

・Phase 1

データ取得のための前準備として二つの基本viewを作製する。一つ目は<PDB>タグをルートノードとして持つXMLデータを取得するためのview。このviewは各PDBデータを取得するための基本となるviewである。二つ目は<name>タグをルートノードとして持つXMLデータを取得するためのview。このviewは一つ目のviewから作製され、一つ目のviewへの参照関係を持つ。

そして更に詳細な結果を絞り込むviewとして、二つ目に作製した<name>取得用のviewに対して各<category>で絞り込みを行ったviewを作製する。これらの関係は図6に示される通りである。

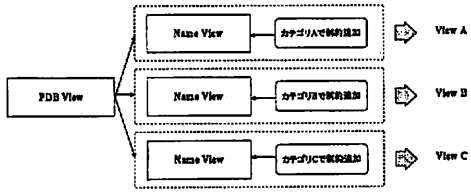

図6:Phase 1

・Phase2

このアプリケーションの目的は<structure>タグ下に格納されているPDBの構造情報を取得することである。その為、<structure>タグをルートノードとして取得するviewの定義が必要である。phase1にて<PDB>タグをルートノードとするviewの定義は行っているため、phase2ではそこから<structure>タグをルートノードとするviewの定義を行う。しかし、このviewをmaterializeした場合、結果として得られるのはすべてのPDBデータにおける構造情報になってしまうため、ユーザの要望に応じた制限事項を付加する必要がある。これらのviewの関係は図7で示される通りである。

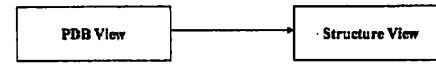

図7:Phase2

・Phase3

phase2で作製した<structure>タグをルートノードとするviewを使用するに当たって絞り込みを行うための制限事項の作製を行う。ユーザは目的となるPDB構造情報を取得するために二段階の絞り込みを行う。最初にphase1で作製した各<category>で絞り込みを行ったviewを選択し、materializeすることで<category>で絞り込みを行った<name>の一覧を取得することが出来る。次に得られたnameから求める条件に合致したものを一つ選択し、その<name>情報を基にした制約条件を<PDB>タグをルートノードとするviewへ付加することで、<category>、名称の二つの条件で絞り込んだ<PDB>を取得するviewとして扱うことが出来る。<structure>タグをルートノードとするviewは<PDB>タグをルートノードとするviewから派生し、参照関係を保持しているため絞り込み条件は伝播する。よって<PDB>を取得するviewに対する制約条件を付加したまま<structure>タグをmaterializeすることで、ユーザの検索条件を満たす結果を得ることが出来る。これらのviewの関係は図8で示される通りである。

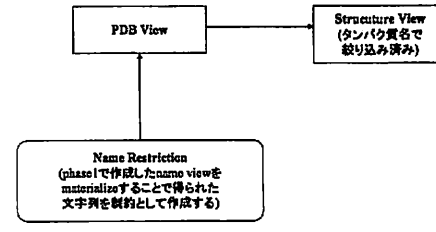

図8:Phase3

このアプリケーションを実際に実装した場合の例を図9に示す。図9におけるGUIの部品の意味は以

下の通りである。

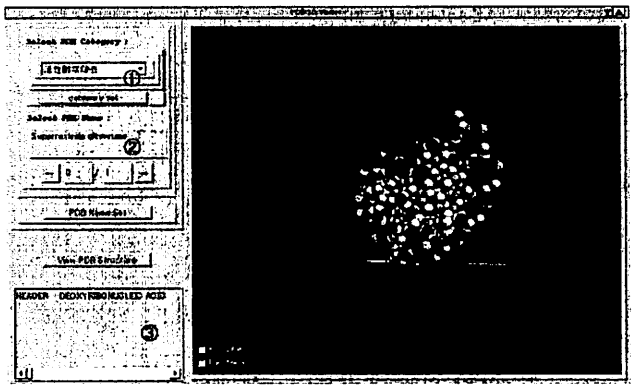

図 9:PDB 検索アプリケーション

(1) Phase1 において生成された<category>毎に制約された<name>を取得するための view 名。<category>の名前がそのまま view の名前となっており、ここでカテゴリ名を選択することで(2)における<name>の選択範囲が変わる。
(2) (1)で選択された view を materialize した結果得られる<name>の一覧。ここで選択した<name>を制約条件として付加することで得られる PDB 構造情報が変化する
(3) (2)で選択した情報を基に制約条件を作製し、付加したことで得られた PDB 構造情報。
(4) (3)の構造情報を基にタンパク質の構造を立体的に 3D で表現する部分

7. まとめと今後の研究方針に関して

本研究では XML データベースの検索言語として採用されている XQuery を簡易に記述するための方法を検討した。具体的な方法として、XQuery を構築するための規範的な XQuery として canonical XQuery を提示し、さらにそれを構築するための component query として XQuery が持つ機能を 6 つのカテゴリに分類し、それらを組み合わせることで容易に XQuery を構築可能であることを示した。また、複雑な XQuery を既述するに当たって canonical XQuery の組み合わせで既述することが可能であること、それを view として定義し、view を組み合わせることで更に複雑なクエリの既述が可能であることなどを示すことが出来た。また、その概念の実装として IntelligentPad 技術を用いた XQuery 開発環境の実装を行い、その実装を用いてアプリケーションの構築を行った。また、この開発環境においてはデータベースのメンテナンス、大量に発生する view の管理など二次的なサポートを行うための機能を実装した。これらの結果を前提とし、今後の研究方針として以下の点を検討している。

7.1 view 管理

XQuery を生成する結果、及び課程で発生する view を効率的に管理する方式の検討が必要である。view はユーザが XQuery 開発環境を利用し、繰り返し検索を行うに従って急激に増加していく。これらの view の管理方法として二つの対応方法を検討している。

一つ目は view に対して追加情報を付加し、その情報を基に分類して管理を行う方法である。追加情報はユーザによって付加され、内容に応じてレベルを設けることによって、階層構造によって view の関係を表現することが可能となり求める view の探索が容易となる。

二つ目は view が持つ参照関係を線で結び、view 同士の参照関係を明瞭にする方法である。この対応により、ユーザが新たに参照先として検討すべき view の探索を容易にする。しかしこれら二点の対応方法では急激な増加を続ける view に対する効果的な対処法としては効果が弱い。その為、新たな対応方法を検討中である。

7.2 より効率的な GUI

本研究における query component の組み合わせやパラメタの変更、view 同士の組み合わせを用いてアプリケーションの構築を行うに当たって、IntelligentPad に関する知識を持たずとも容易に開

発できるためのサポート GUI として、ウィザード形式の GUI を提供した。これにより IntelligentPad に関する知識を持たずとも、ウィザードのガイドに従って必要事項を入力することでアプリケーションを構築することが可能となったが、このウィザード形式の GUI 自体はあまり直感的なものとは言い難い。その為、ユーザがより直感的な操作で XQuery を構築できるような GUI を検討する必要がある。

参考文献

[1]服部雅一: XML 技術; 東芝レビュー, Vol.56, No.11

[2]服部雅一: XML Database, 日本データベース学会論文誌, Vol.43

[3]W3C: XQuery1.0 An XML Query Language, http://www.w3.org/XML/Query/

[4]Yuzuru Tanaka: MEME MEDIA and MEME MARKET ARCHITECTURES, wiley-interscience(2003)

[5]M.Petropoulos, Y.Papakonstantinou, V.Vassalos: Graphical Query Interfaces for Semistructured Data The QURSED System, ACM Transactions on Internet Technology(2005)

[6]Stylus Studio: Stylus, http://www.stylusstudio.com/

[7]E.Harold: XQuisitor, http://www.cafeconleche.org/xquisitor/

[8]S.Berger, F.Bry, S.Schaffert, C.Wieser: Xcerpt and visXcerpt From Pattern-Based to Visual Querying of XML and Semistructured Data, VLDB2003(2003)

[9]M.Erwig: Xing A Visual XML Query Language, Journal of visual languages and computing(2003)

[10]D.Braga, A.Campi, S.Ceri, E.Augurusa: XQuery by Example, lecture notes in computer science(2004)

[11]D.Braga, A.Campi, S.Ceri, E.Augurusa: Design and Implementation of a Graphical Interface to XQuery, SAC(2003)

[12]Z.Qin, B.Yao, Y.Liu, M.McCool: A Graphical XQuery Language Using Nested Windows, lecture notes in computer science(2004)

[13]S.Ceri, S.Comai, E.Damiani, P.Fraternali, S.Paraboschi, L.Tenca: XML-GL a Graphical Language for Querying and Restructuring XML Documents, Computer Networks(1999)

[14]R.Abraham: FoXQ XQuery by Forms, Human Centric Computing Languages and Environments(2003)