筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

ＸＭＬデータ管理インタフェースの開発
―ＸＭＬエディタとグラフィカルビュー間の連携―

佐藤　俊輔

(コンピュータサイエンス専攻)

指導教員　田中　二郎

２００９　年　３　月

概要

企業や各機関が保持する膨大なデータを管理する新しい方式として，XML データベースが期待されている．一方で，XML データを管理するツールや視覚的に操作する XML エディタも開発されている．しかし， XML 文書全体を俯瞰したい場合，一般的な XML エディタの機能では XML 要素の偏りなど，全体をスムーズに把握することは難しい．

これに対して本研究開発プロジェクトでは，学会・研究会で XML 文書の説明を行うための資料を作成する場面などにおいて，利用者を支援することを目的と定め，XML データの管理インタフェース「XGraphi」の開発を行なった．XGraphi は一般的な XML エディタが備える編集機能を備えており，さらにズーミング機能などを実装したグラフィカルビューを持つ．

開発は筆者を含む 3 人のプロジェクトチームメンバで行った．独立性の高い機能部分を XML エディタ・グラフィカルビュー・DB 接続の三領域と設定し，各チームメンバに担当領域を割り当てて，開発を進めた．開発後期には委託者である天笠俊之講師の協力のもと，XGraphi のユーザビリティ評価を行い，高い評価を得た．

開発において筆者は，XML エディタの開発，及びグラフィカルビューとの連携を担当した．XML エディタは，テキストビューとアウトラインビューで構成されている編集機能を備えた機能領域である．テキストビューの開発では，XML 文書をタブや空白などに区切る 3 種類のスキャナを利用することで，XML 要素を分類し，テキスト表示を行う際の配色を可能にした．アウトラインビューの開発では，アウトラインビュー上での操作結果をテキストビューとグラフィカルビューに反映させるため，ツリーモデルの拡張を行った．結果として，XML エディタとグラフィカルビューの連携が実現し，XGraphi の編集に関する機能要件を満たすことができた．

目次

図目次

表目次

第1章　序論

IT 技術が普及するに従い，企業や各機関が保持するデータはより大規模化してきた．それらの膨大なデータを長期間維持・管理・活用していくためには，より使いやすく，かつ柔軟なデータベース管理方式が必要になる．現在は関係データベースが一般的に用いられているが，関係データベースはシンプルで使いやすい反面，文書データなどの非構造データの扱いや，利用するアプリケーションやデータ構造の変化に対し，柔軟に対応できなといった欠点もあげられた．それらを補う新しい方式として，XML データベースが期待されている．

XML データベースは，厳密なスキーマ定義が必須ではないため，関係データベースに比べてデータ構造の拡張性が高く，DB 設計の途中の変更にも柔軟に対応できる．また，アプリケーションデータの XML データ化などの影響から，XML データが増加しているため，XML データベースの普及・研究開発はますます加速するものと思われる．一方で，XML データベースを管理するツールや XML データを視覚的に操作するツールも必要となってきている．

本研究開発プロジェクトでは，XML データベースの研究を行っている，筑波大学大学院システム情報工学研究科の天笠俊之講師の委託である，XML データの管理インタフェース「XGraphi」の開発を行なった．本プロジェクトでは，XGraphi を開発することで，研究活動における利用や，学会や研究会で XML 文書の説明を行うための資料作成の一助になるアプリケーションを提供することを目的としている．

開発において筆者は，XML エディタの開発，及び XML エディタとグラフィカルビュー間の連携部分を担当した．XML エディタは，XGraphi において XML データの編集機能を提供する機能領域であり，アプリケーションの基盤となるものである．そのため，この部分に対する機能に漏れがあった場合，XML 管理インタフェースとしての利用価値を大きく損なう可能性がある．このことを念頭に置き，機能要件の抽出は，天笠講師に開発中のアプリケーションを使用してもらいながら，繰り返し行った．また，XML エディタとグラフィカルビュー間の連携においては，二つの機能領域間でプログラムの衝突が起こる可能性がある．これに対して，アプリケーションの設計工程から両機能領域の内部仕様の衝突に配慮し，スムーズに開発できるように努めた．

本報告書の構成は以下の通りである． 2 章では，XML と XML データベースに関する説明を行い，これらを扱う既存の XML エディタの例とその問題点を述べる．3 章では，本プロジェクトの開発体制・開発環境などについて説明し，4 章で開発した XGraphi の機能を XML エディタ・グラフィカルビュー・DB 接続の三領域に分けて説明する．5 章では本プロジェクトの推移を述べ, 各開発工程での成果物及び XGraphi の客観的な評価結果について説明する．6 章では，筆者が開発を担当した部分についての説明と，実装上の問題点・改善点について考察し，7 章で本プロジェクトに関する結論を述べる．

第2章　XML の概要

本章では，本報告書で対象とする XML とその保存に特化したデータベースシステムである XML データベースについて説明し，これらを扱うための一般的なアプリケーションである，XML エディタについて述べる．更に，一般的な XML エディタの問題点を指摘する．

2.1　XMLとXMLデータベース

XML は，1998 年に W3C (World Wide Web Consortium)によって勧告された，データの記述方法である．マークアップ言語を記述するメタ言語であるため，文書内に付加情報を与える目印（タグ）を付けることができる．マークアップ言語である HTML (HyperText Markup Language)では，決められたタグしか用いることができないが，XML では，ユーザが自由にタグをつけることができるため，データに様々な意味を持たせることができる．更に，タグ内には属性を記述することができ，データに付加的な情報を加えることができる．また，データをテキスト形式で表現しているため，マルチプラットフォーム環境でのデータ交換・処理に適している．

XML はデータを記述する言語であるため，データ構造を記述することができる．このデータ構造はスキーマと呼ばれ，同じ XML 文書内にデータとスキーマを一緒に記述することも，別な文書に分離して記述することもできる．また，スキーマを定義することによって，データ自身がスキーマの定義に合致しているが検証することができる．

1998 年の W3C 勧告以来，XML は急激な勢いで IT システムに広まった．しかし当時は，XML 文書を保存するファイルシステムおよびデータベースに，性能面での問題や，XML の構造を定義するスキーマ言語への対応の問題があった．その後，スキーマレスの XML が広まったものの，一般的に普及しているデータベースシステムである関係データベースは，XML への対応が未だに遅れているのが状況である．

そこで，XML 文書を保存するのに特化した，XML データベースが開発された．XML データベースは，XML の特長を生かすため，関係データベースでは困難とされた，データ構造の変更を可能としている．これにより，データベースの詳細な論理設計が不要であり，設計の途中で変更できるため，非常に柔軟性・拡張性に富んだデータベースだと言える．更に，昨今のシステム開発の現場では，アジャイル開発手法を採用する場合も増えてきており，XML データベースの柔軟性や拡張性はこのような反復的なシステム開発に適している．以上の事から，今後 XML データベースの開発と利用が広がっていくと予想される．

2.2　XMLエディタ

XML エディタは，XML 文書を表示・編集するためのアプリケーションの総称である．その機能・外観は，各アプリケーションによって様々である．プロジェクトで開発を進めるにあたり行った事前調査において，XMLNotepad 2007 [1]，XMLEDITOR.NET [2]，Open eXeed [3]，Eclipse XML エディタ [4]，eXist Client Shell [5]といった複数のエディタの機能及び外観を調査した．その内，eXist Client Shell 以外の XML エディタは，一般の XML 文書の表示・編集を目的として開発されており，その中でも XML Notepad2007 は広く利用されている．一方，eXist Client Shell は，XML データベースにある XML ファイルの表示・編集と，XQuery の実行を目的として開発されている．以下では，XML Notepad2007 と eXist

Client Shell について，その概要と問題点について述べる．

2.2.1 XML Notepad 2007 とその問題点

XML Notepad 2007 は以下の三種類のビューを備えている

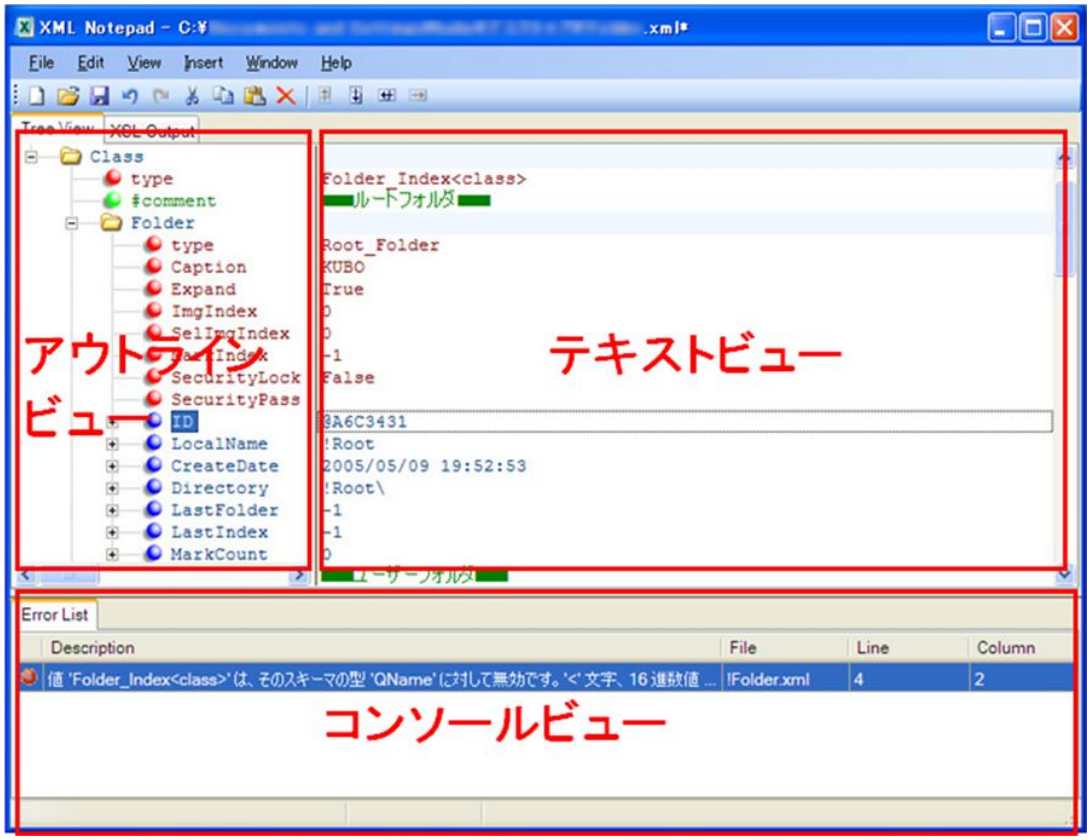

図 2-1 XMLNotepad 2007

- アウトラインビュー

  XML 文書の階層構造をツリー表示するビューである．テキスト・コメント要素などを挿入する機能やドラッグ&ドロップすることで各要素の位置を変更する機能を備えている．

- テキストビュー

  XML 文書を直接編集するためのビューである．高機能なエディタであれば入力補完機能などを備えている．

- コンソールビュー

  主にアプリケーション画面下部に位置し，提供する機能は様々である．例として XML スキーマのチェック機能や XML 要素のプロパティ情報を表示する機能などがある．

XML はデータの表現方法としてはその性質上，冗長である．これは一つの XML 文書内に同名である要素や属性が何度も出現することからもわかる．この冗長性故に，XML 文書全体を俯瞰したい場合，アウトラインビューであっても要素の偏りなど，全体をスムーズに把握することは難しい．これはアウトラインビューがスクロールを前提とした表示方式である

ため大容量データになると全体を一度に表示できないためである．このように，一般的なXMLエディタが備えているビュー・機能はXMLデータ全体の把握に時間がかかってしまうという問題点がある．また，DBへの接続機能も一般的なXMLエディタは備えていない．

2.2.2　eXist Client Shellとその問題点

2.1節で述べたXMLデータベースとしてeXist XMLデータベース（eXist）[5]が広く用いられている．eXistは，データベースの機能に加えて，図2.2に示すGUIクライアント（eXist Client Shell）を持っている．これを用いて，データベース内にあるXML文書を開き，図2.3に示すエディタで同文書の編集などの操作を行うことができる．

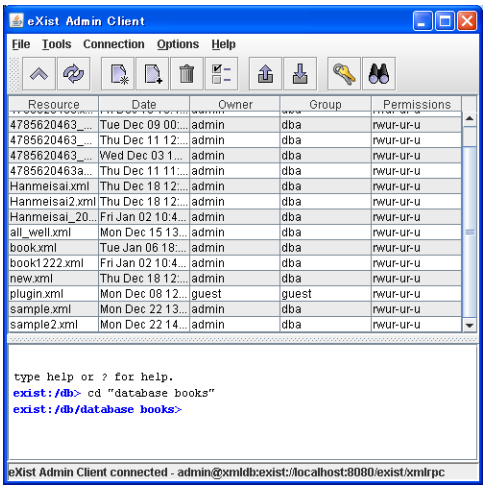

図 2-2　eXist Client Shell

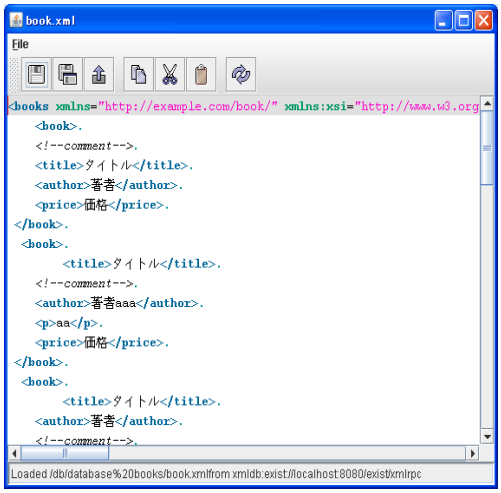

図 2-3　eXist Client Shellのエディタ

eXistを用いれば，データベースへの接続が可能となり，データベースに格納されているXML文書を容易に編集ができる．しかし，テキスト表示機能が貧弱であり，一般のXMLエディタが持つようなアウトラインビューを持たないため，XMLデータ構造を俯瞰することが難しい．

以上の事から，XMLエディタはXMLデータ構造を俯瞰でき，かつデータベースにも接続できる機能が必要である事が分かった．

第3章　開発

3.1　開発体制

本プロジェクトでは，表 3.1 に示すメンバで構成されるチームによって開発を行った．委託者である天笠講師とは，具体的な要件確定と進捗状況の確認，および成果物に対するレビュのため，定期的なミーティングを行い，共同で開発を進めた．開発は 9 月 3 日に始動し，12 月 19 日の納入確認を以って終了した．

表 3.1　開発メンバ

委託者/アドバイザ	天笠俊之講師
開発チームメンバ	柿沼基樹
	佐藤俊輔
	ラトナマラララシト

3.2　開発環境

本プロジェクトは，表 3.2 に示す環境で開発を行った．Eclipse は IBM によって開発された統合開発環境の一つであり，オープンソースとして全世界で開発・利用されている．今回開発したアプリケーションは，Eclipse のフレームワークおよび関連ライブラリ上で開発が容易であり，かつ保守性と拡張性が非常に高く保てるという利点を考慮し，今回は Eclipse のプラグインとしてアプリケーションを開発した．

FuzzyXML は Amateras Project で開発がおこなわれている，GUI ツール等でのインタラクティブなテキスト処理に使用することを前提とした XML パーサであり，Common Public License 1.0 に準じている．本アプリケーションを開発する上で非常に有効であったため，ソースコードおよびライブラリを利用した．

eXist は Java で記述されたオープンソースのネイティブ XML データベースシステムであり，本アプリケーションが対象としているデータベースである．

表 3.2　開発環境

OS	Windows XP, Vista
統合開発環境	Eclipse SDK 3.4
開発言語	Java (jdk1.6.0_10)
利用ライブラリ	FuzzyXML (XML パーサ) eXist (XML データベース)

3.3　初期開発日程

本プロジェクトの初期の開発スケジュールを，図 3-1 に示す．初期のスケジュールでは，完成日時を 12 月 15 日と予定していた．10 月の 2 週目までに要件定義を行い，その後，反復型開発プロセスを利用し，イテレーション（反復）を 2 回まわすことで開発を行う計画を立てた．計画における各イテレーションで開発する機能を表 3.3 に示す．

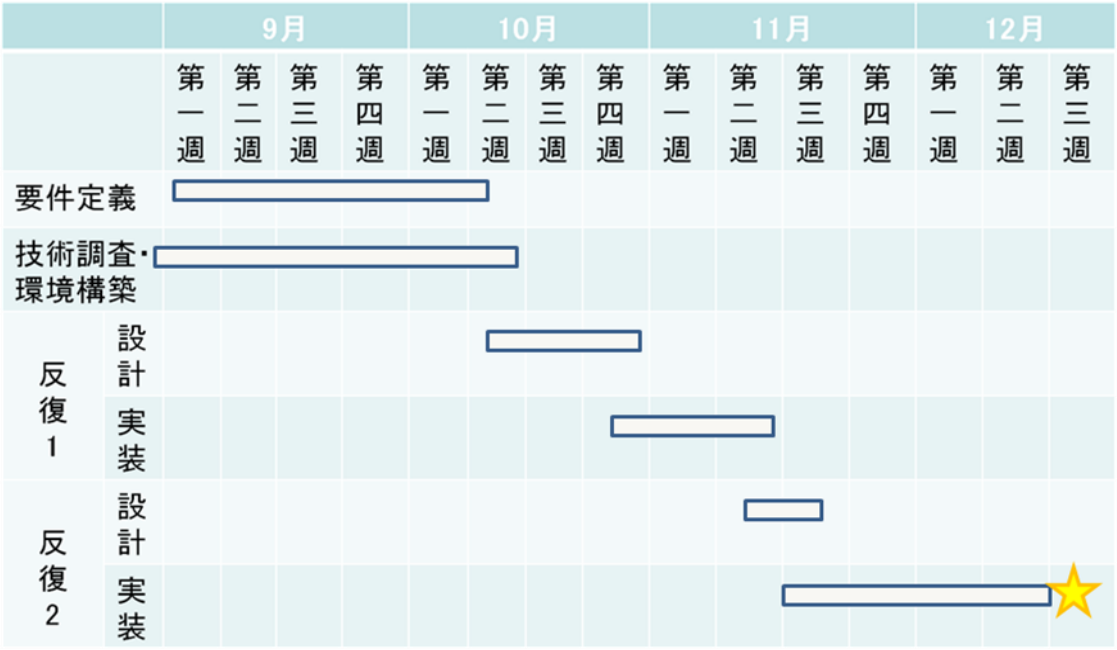

図 3-1　初期開発スケジュール

表 3.3　各イテレーションの開発機能

イテレーション（反復）	開発領域	開発機能の計画
1	XML エディタ	テキストビュー
		アウトラインビュー
	グラフィカル表示	グラフィカルビュー
	XML データベース	XML データベースから XML 文書のインポート
2	XML エディタ	アウトラインビュー上で，要素の追加と削除
	グラフィカル表示	ズーミング
	XML データベース	XML データベースから XML 文書のエクスポート

必須機能の内，技術的難易度の低い機能から高い機能へ段階的に実装していくことを前提として各イテレーションで開発する機能を決定した．

3.4　著作権等について

開発したアプリケーションの利用者に制限は設けず，かつ無償で提供する．また，開発した XGraphi のソースコードの著作権や改編権はすべて天笠講師, および筑波大学システム情報工学研究科に譲渡する．

第4章　XGraphi の概要

XGraphi の開発に先立ち，我々は開発機能をそれぞれ XML エディタ，グラフィカルビュー，DB 接続の 3 領域に分類した．分類の目的は，要件を系統立てて整理する事及び開発効率向上のためにチーム内での開発を分担する事である．我々は機能要件を分類した後，次の通り開発領域を分担した．

・XML エディタ：佐藤俊輔
・グラフィカルビュー：柿沼基樹
・DB 接続：ラトナマララシト

次に各領域の機能概要を説明する．

4.1　XMLエディタ

XML 文書を編集する機能を備えたエディタである. 図 4-1 の赤枠で囲まれた部分は, XML エディタが提供するインタフェースである．XML エディタは次の二つのビューで構成される．

- テキストビュー
  XML 文書のデータをテキスト表示する領域である．
- アウトラインビュー
  データの階層構造をツリー表示する領域である．

両インタフェースを通して，XML データの直接編集や，階層構造の変更を行うことができる．

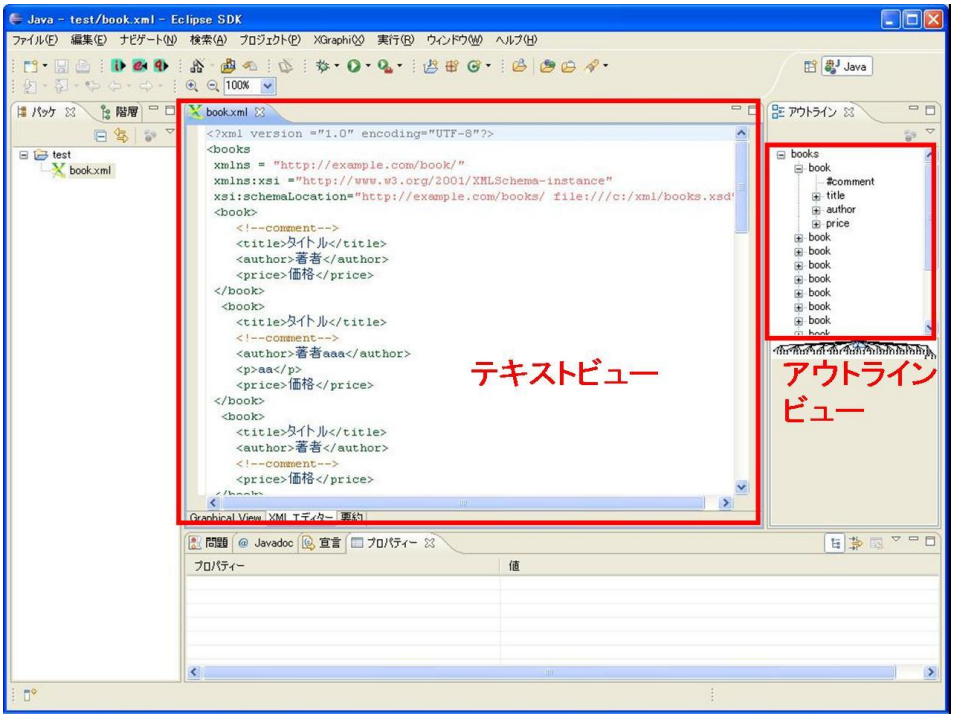

図 4-1　XML エディタの表示画面

4.1.1　テキストビュー

XML 文書を XGraphi で開くと，テキストビューでは図 4-1 のような画面が表示される．XML データの内，処理命令・要素・CDATAsection・DTD(Document Type Definnition)・コメント・属性・テキストはそれぞれ色を分けて表示される．XML データを色分けすることで直接編集する際に，どの部分を編集しているのかが分かり易くなっている．XML データの種類と配色の関係を表 4.1 に示す．

表 4.1　XML データの種類と配色の関係

種類	配色
処理命令：<?>	グレー
要素：<>	グリーン
CDATAsection：<![CDATA[]]>	ライトブルー
DTD：<!DOCTYPE>	イエロー
コメント：<!---->	オレンジ
属性	レッド
テキスト	ブルー

テキストビューでは XML 文書の直接編集が可能であり，編集結果は「名前を付けて保存」や「CTR+S」で保存できる．編集が保存されると，XML エディタ及びグラフィカルビューに反映される．

4.1.2　アウトラインビュー上での選択と各ビューのフォーカス

アウトラインビューでは，ツリー要素を選択するとテキストビューやグラフィカルビューにおいて対応した要素の周辺にジャンプして表示させる(フォーカス)機能を持つ．この様子を図 4-2 に示す．また，グラフィカルビュー内の要素を選択した場合は，アウトラインビューにおいて対応した要素が選択される．この時選択された要素がアウトラインビューのツリー内に折りたたまれていた場合は，選択した要素が現れる部分までツリーを展開表示する．この様子を図 4-3 に示す．

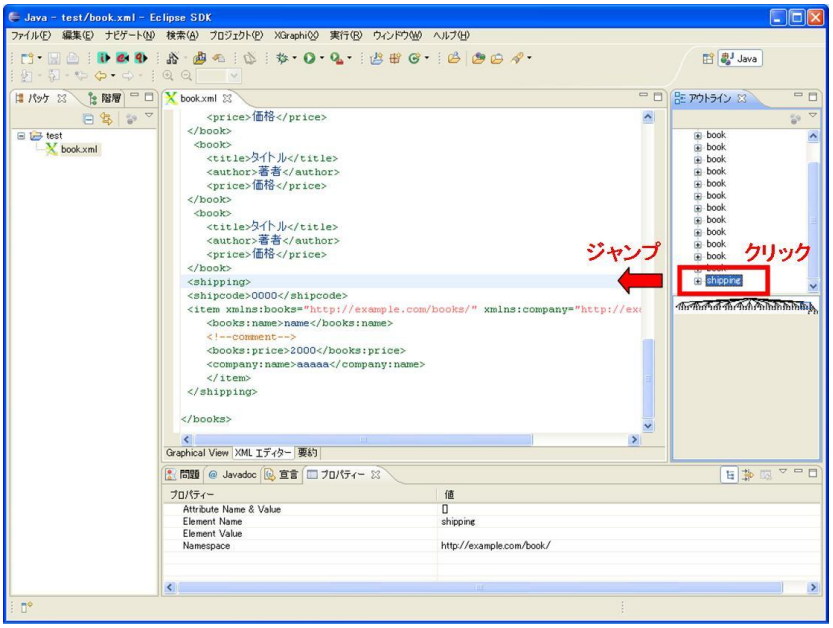

図 4-2　選択とジャンプ

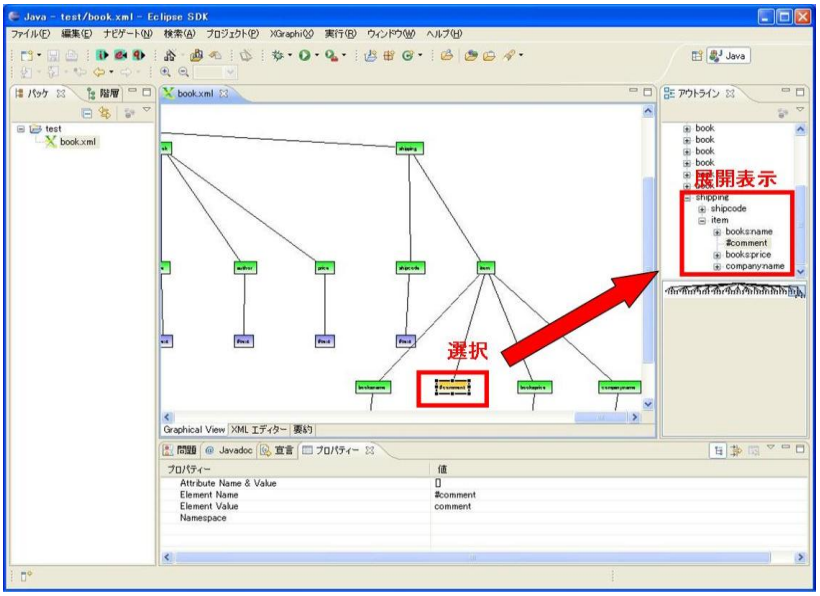

図 4-3　グラフィカルビューでの選択とアウトラインビューの展開表示

4.1.3　アウトラインビューの操作

アウトラインビューでは「要素の追加・削除」「名前・値の変更」「要素の移動」といった操作が可能である．これらの操作を行った時，XGraphi は編集結果を自動的に保存し，全てのビューに反映する．

(a)　要素の追加・削除

この操作は図 4-4 に示すポップアップメニューから行う．ポップアップメニューはアウトラインビュー内で右クリックすると表示される．要素の削除を行うと選択中の要素とその子にあたる全ての要素が削除される．

要素の追加ではコメント・要素・CDATAsection・テキストの追加を行うことができる．追加する位置として，選択した要素の前後または子として指定することができる．ただし，子要素を持つ事ができないコメント要素等に子要素を追加するなど，不正な操作が行われた場合はエラーを表示して操作を反映しない．

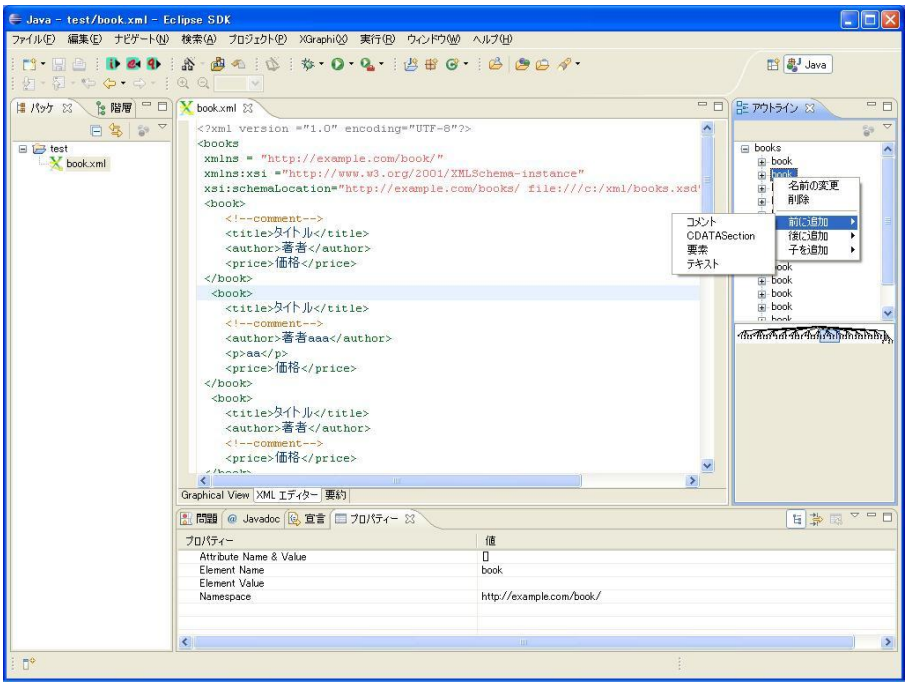

図 4-4　ポップアップメニュー

(b)　名前・値の変更

この操作はポップアップメニューから行う．ポップアップメニュー内の「名前の変更」を選択すると，図 4-5 に示す変更ダイアログが表示される．このダイアログのテキストボックスを編集すると，選択中の要素の名前が変更される．また，要素の名前以外にテキスト値・コメント値の変更もこの操作から行える仕様にしている．

図 4-5　名前変更ダイアログ

(c)　要素の移動

この操作はアウトラインビュー内で要素をドラッグ＆ドロップすることで行う．この様子を図 4-6 に示す．この操作で移動する要素が子要素を持っていた場合，その子要素も同様に移動する．また，ツリーの先頭であるルート要素と同じ階層に要素をドラッグ＆ドロップしても，不正な操作としてエラーを表示し，操作を反映しない．

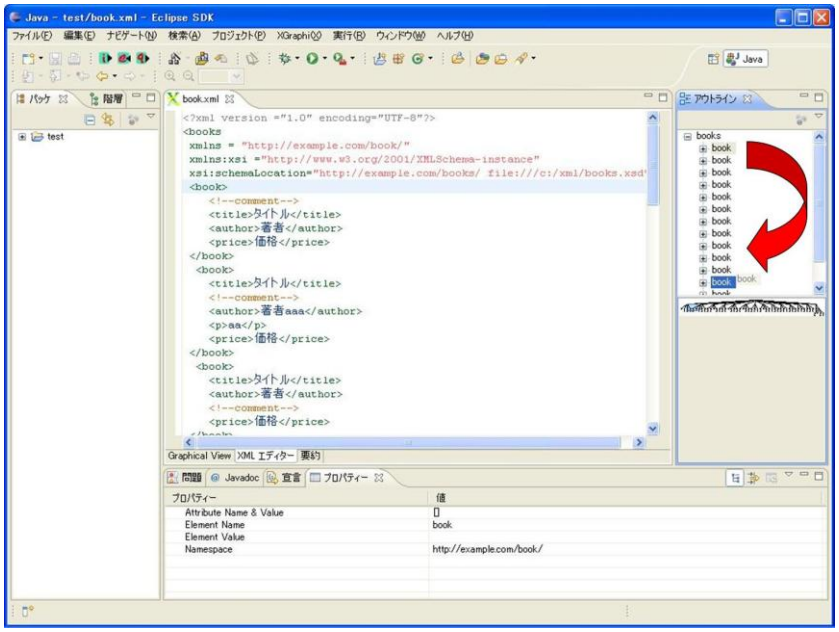

図 4-6　要素の移動

4.2 グラフィカルビュー

グラフィカルビューは，XML データを視覚的に表示することで，XML 文書が持つデータ構造をグラフィカルに表示する機能である．グラフィカルビューが持つ機能を以下に示す

・グラフィカルビューでの XML データ表示
・プロパティビューでのデータの詳細表示
・Summary ビューでの XML 文書概要表示
・要素の格納・展開
・ズーミング
・サムネイル表示

これらの機能について，以降で説明する．

4.2.1 グラフィカルビューでの XML データ表示

XML 文書を XGraphi で開くと，図 4-7 のような画面が表示される．XML 文書内の XML 要素名（グリーン），要素値（ブルー），コメント（オレンジ），および DTD 宣言（イエロー）を四角ノードとしてグラフィカル領域に表示する．XML 文書の構造が木構造と似ている事から，ノードを木構造に表示することで，データ構造を直感的にわかりやすくしている．色つきのノードが表示されている領域が，グラフィカルビューである．

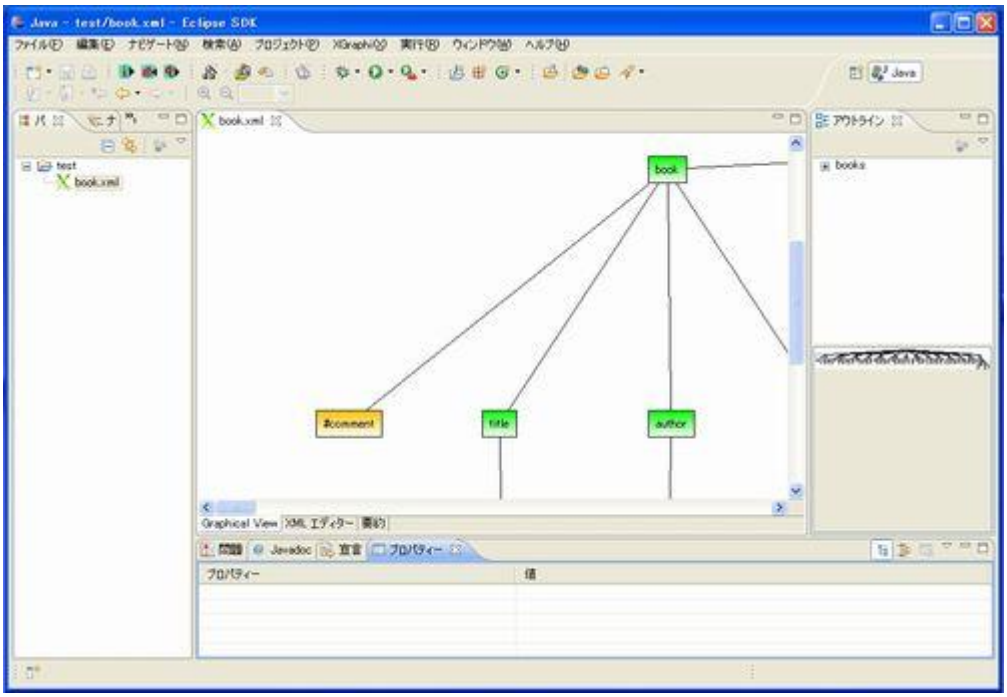

図 4-7 グラフィカルビューの表示画面

各ノードはマウスで移動を行うことができる．ノードが移動してもノード間につながれているコネクションは維持される．また，ノードの拡張，縮小も行うことができる．

画面下方に向かうノードほど，データ構造上でのデータの示す深さが大きくなり，同じ高さに表示されているノードはデータの深さが同じことを示している．また，親ノードと子ノード間は線でつながれている．要素名のノードは子ノードを持つが，要素値ノード，コメントノードは子ノードを持たない．

このビューを構成する XML 文書自身が編集，保存されると，グラフィカルビューは自動的に変更が反映（リフレッシュ）される．しかし，位置情報については保存を行っていないため，リフレッシュの段階で位置情報は再計算される．この仕様は，XML 文書が変更されることによって，新しいデータ（ノード）が追加または削除され，データ構造が変更する可能性があり，ノードの位置情報を再計算するほうがより合理的なためである．

4.2.2　プロパティビューでのデータの詳細表示

グラフィカルビューは，eclipse のプロパティビューと連携を行っており，XGraphi 上でプロパティビューを開くと，グラフィカルノード上で選択したノードの詳細情報を閲覧することができる．このビューを図 4-8 に示す．図 4-8 において，赤線で囲った範囲がプロパティビューである．

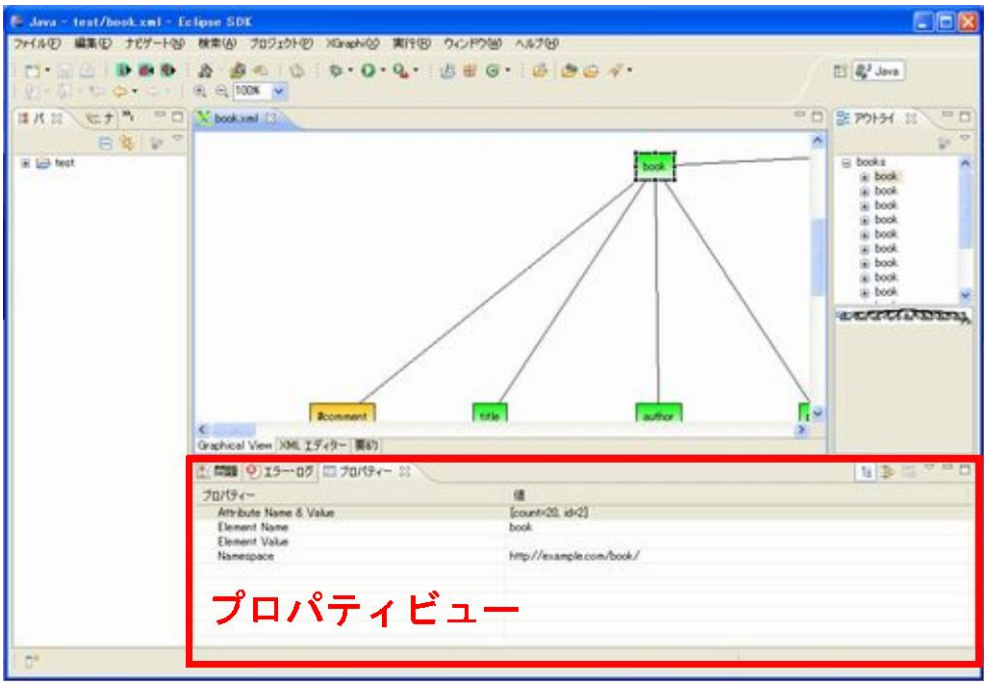

図 4-8　プロパティビューの表示画面

グラフィカルビュー内で要素名のノードを選択した場合，要素値以外の値が表示される（属性名，名前空間は任意）．　その他のノードが選択された場合は，要素の名前と値が表示される．このビュー上では XML データの編集は行えないが，ノードの属性値や名前空間など，グラフィカルビュー内およびアウトライン上で表示できなかった情報を表示することができる．

4.2.3　Summary ビューでの XML 文書概要表示

Summary ビューとは, XML 文書の情報を簡単な表にしたものであり, XML バージョン, XML エンコーディングや，要素の個数，コメントの個数を表示してくれる．このビューを図 4-9 に示す.

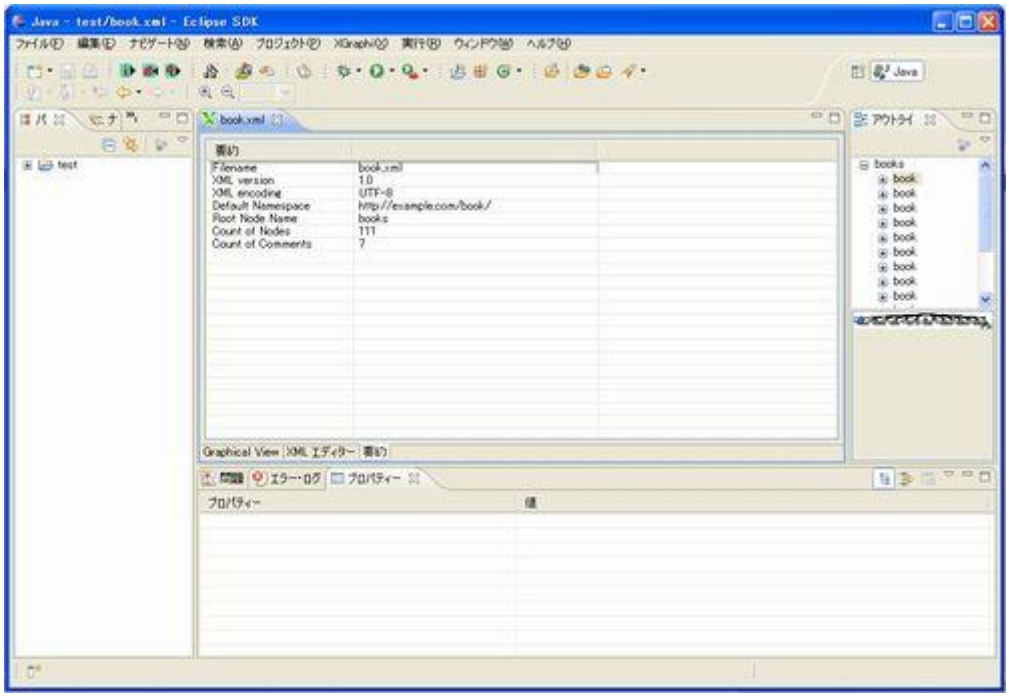

図 4-9　summary ビューの表示画面

このビューは XML 文書が編集され，保存が行われると，その変更が反映される．XML バージョンやエンコーディングが記述されていない場合は，表示されない.

4.2.4　要素の格納・展開

グラフィカルビュー内での要素は，木構造でレイアウトされているが，本アプリケーションでは，子ノードを持つ親ノードをマウスでダブルクリックすることで，子ノードを格納・展開する機能を設けた．これにより，データ構造上見づらい箇所や，一時的に子ノードを不可視にしたい場合などに役に立つ．ノードの格納・展開の様子を図 4-10・図 4-11 に示す.

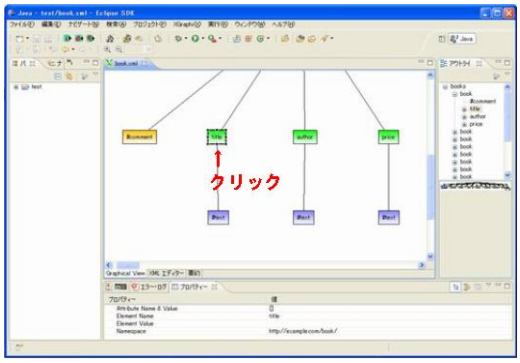

図 4-10　要素の格納_1

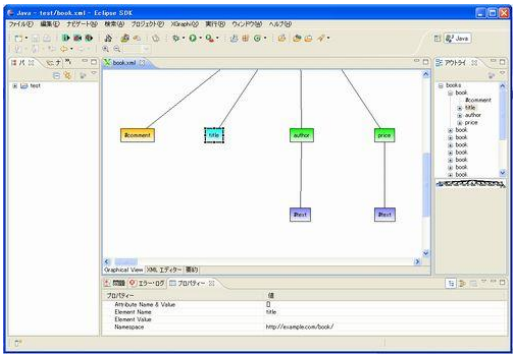

図 4-11　要素の格納_2

格納・展開の動作情報は，内部的にスタックとして積まれているため，格納されていたノードがさらに格納された場合でも，以前に格納されていた情報を失わない．よって，図 4-12～図 4-15 に示すように，格納の動作が連続しても，以前の格納が保持されたまま格納されることになる．

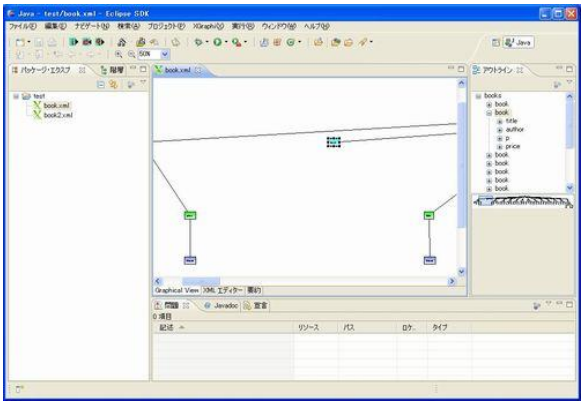

図 4-12　要素の展開_1

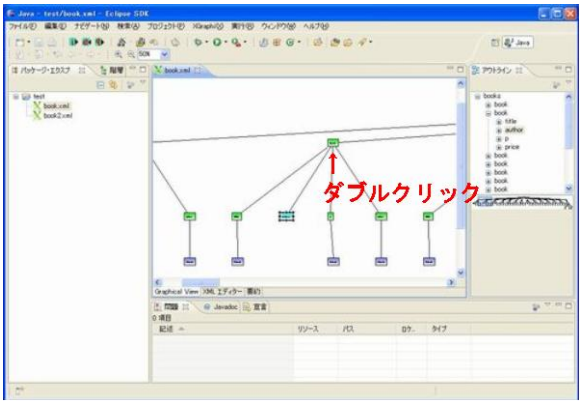

図 4-13　要素の展開_2

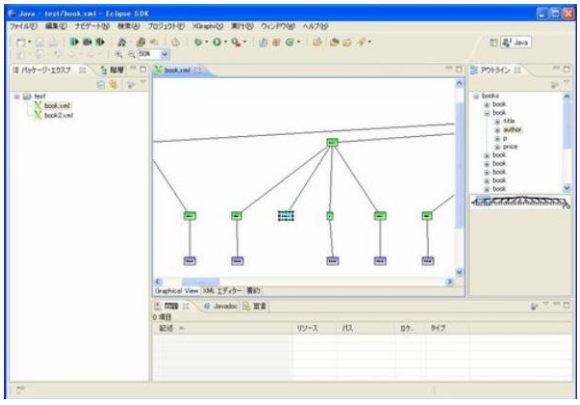

図 4-14　要素の格納_3

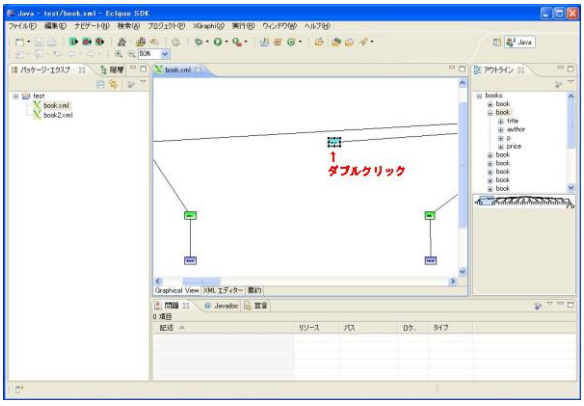

図 4-15　要素の格納_4

また，格納・展開の回数に制限はなく，子ノードを格納する場合は，子ノード以下のノードすべての子ノードを再帰的に格納している．例えば，ルートノード（XML 文書の先頭の要素）をダブルクリックすると，ルートノード以外のすべての要素が格納される．

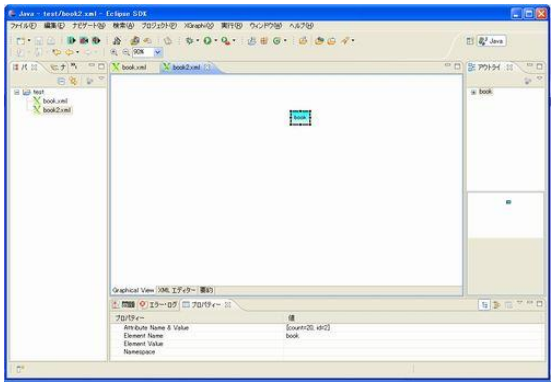

図 4-16　ルートノード以下の格納_1

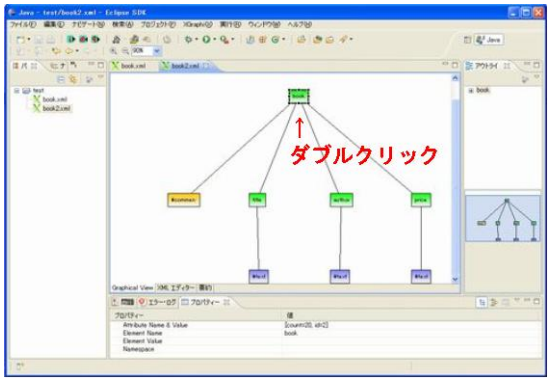

図 4-17　ルートノード以下の格納_2

4.2.5　ズーミング

XML 文書のデータ構造をグラフィカルに表示する際，XML 文書の要素が多いと，グラフィカル領域に収まらず，結果としてデータ構造の可視化につながらない可能性がある．これを回避するため，グラフィカル領域を拡大・縮小して表示をすることで，要素数が多い XML 文書に対しても，データ構造の可視化を可能にする機能を設けた．グラフィカルビュー内の拡大・縮小の様子を図 4-18・図 4-19 に示す．また，ズーミングの可能な倍率を表 4.2 に示す．

表 4.2　ズーミング可能な倍率

縮小倍率	1%　5%　10%　15%　20%　30%　40%　50%　60%　70%　80%　90%
拡大倍率	100%　150%　200%　250%　300%
その他	ページ　高さ　幅　　（それぞれの値に合わせて表示する）

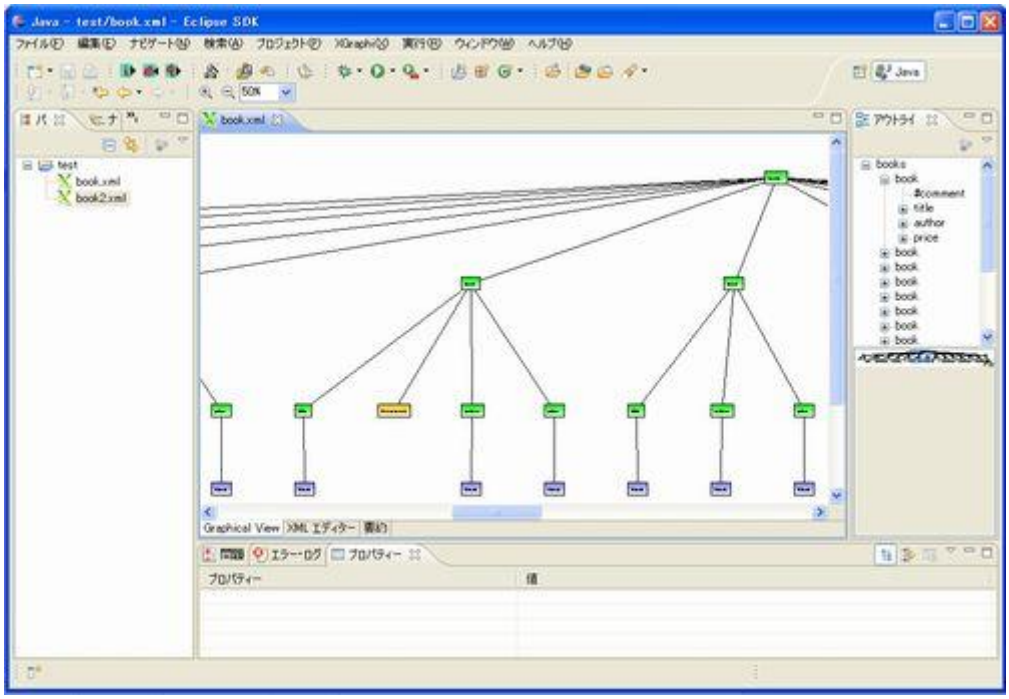

図 4-18　縮尺 50%のズーミング

図 4-18 では，倍率 50%に縮小した図を表示している．データ構造が若干見易くなっているものの，全体像は見えない．そこで，倍率を幅に合わせると，図 4-19 に示す通りの表示になる．

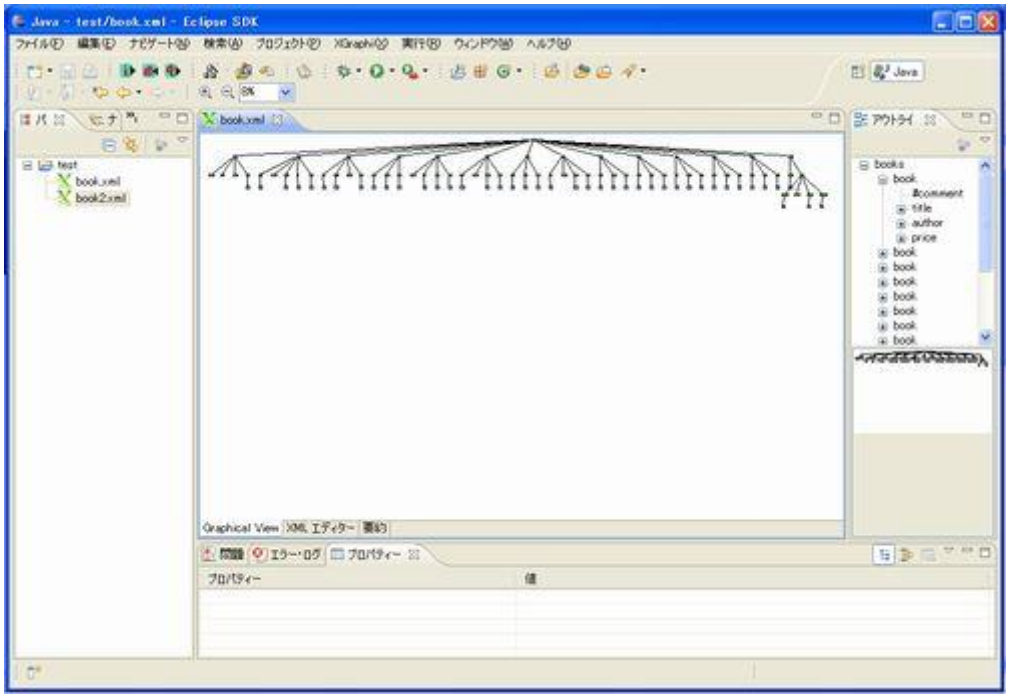

図 4-19　幅に合わせたズーミング

グラフィカルビューの縮小・拡大機能を設けることによって，XML 文書のデータ構造を視覚的に理解する支援ができる．

4.2.6　サムネイル表示

4.2.5 項で説明したズーミング機能とは別の機能として，グラフィカル表示を常に縮小した，サムネイル表示を行う機能がある．サムネイル表示上では，現在利用者がグラフィカルビュー上に置いているスコープを表示するため，グラフィカルビューのどの位置にスコープを当てているか一目でわかるようになっている．また，そのスコープをマウスでドラッグすることによって，現在のスコープを変更することができる．さらに，グラフィカル領域がズーミングによって変化すると同時に，サムネイルの表示の大きさとスコープも相応に変化するようになっている．

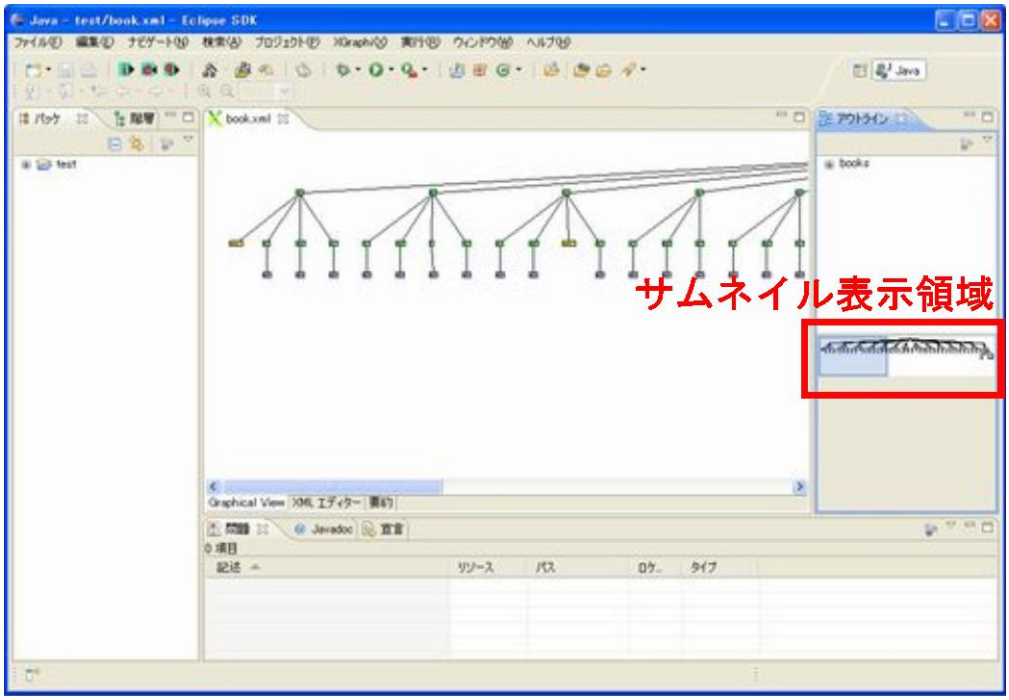

図 4-20　サムネイルの表示画面

4.3 DB接続

一般の XML エディタでは，XML データベースに接続し，DB 内にある XML 文書を扱うことができる機能が設けられていない．それに対して本プロジェクトでは，XML データベース内にある XML 文書においても，通常の XML 文書（XML データベースではなく，ローカルコンピュータにある XML 文書）と同様にエディタで操作（4.1 節と 4.2 節で述べた操作）ができるようにしたいという要件があった．

DB 接続機能は，eXist XML データベース内にある XML 文書において，エディタで操作可能なインタフェースを提供する．DB 接続機能は，主に以下の機能から構成される．

- XML 文書のインポート
- XML 文書のエクスポート
- XQuery の実行と実行結果のインポート

以下，それぞれの機能について説明する．

4.3.1 XML 文書のインポート

eXist XML データベースにある XML 文書を Eclipse プロジェクトにインポートし，エディタで表示する機能である．

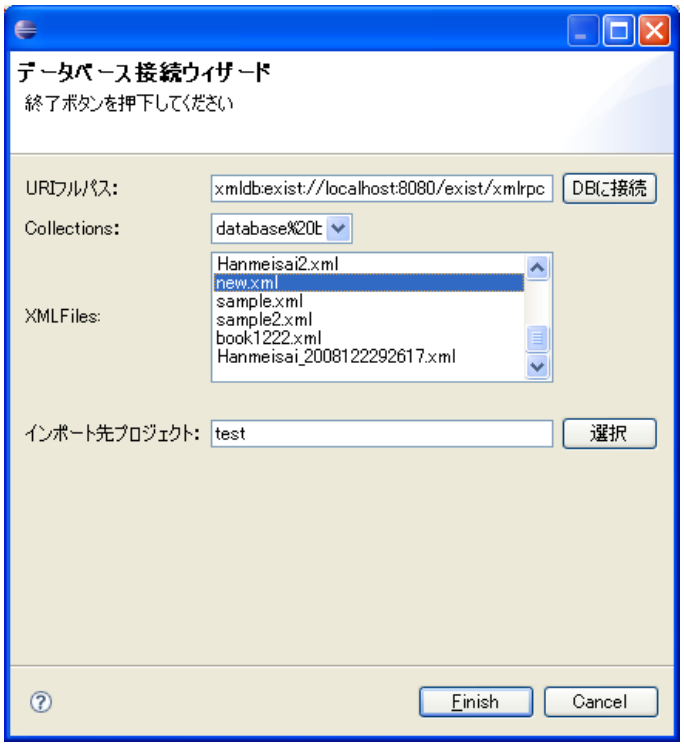

図 4-21 XM 文書のインポート実行ウィザード

図 4-21 に表示するウィザードにおいて，DB の URI（Uniform Resource Identifier）を入力し，DB に接続する．接続が正常に終了したら，DB 内にあるコレクションが読み込まれ，コレクションコンボボックスに表示する．コレクションとは，複数の XML 文書の入れ物でありコンピュータのディレクトリに相当する．コンボボックスからコレクションを選択する

と，コレクション内にある XML 文書が読み込まれ，リストに表示する．リストから XML 文書を一つ以上選択し，インポート先プロジェクトを入力（または，選択）することにより，インポートを行うことができる．また，DB に接続する際，ユーザ名とパスワードによるユーザ認証を行っており，セキュリティも考慮した仕様となっている．インポートした XML 文書に対し，通常の XML 文書と同様の操作を行うことが可能となる．

インポート機能では，コレクションと XML 文書の自動的な読み込み，インポート先プロジェクトの自由な選択などにより利便性を向上させ，使いやすいインタフェースを実現した．また，DB 接続時にユーザ認証を行うことでセキュリティを向上させる工夫をした．

4.3.2　XML 文書のエクスポート

Eclipse プロジェクト内にある XML 文書を eXist XML データベースにエクスポートする機能である．この場合，エクスポートする XML 文書は，3.1.3.1 でインポートしたものだけではなく，通常の XML 文書も可能となっている．

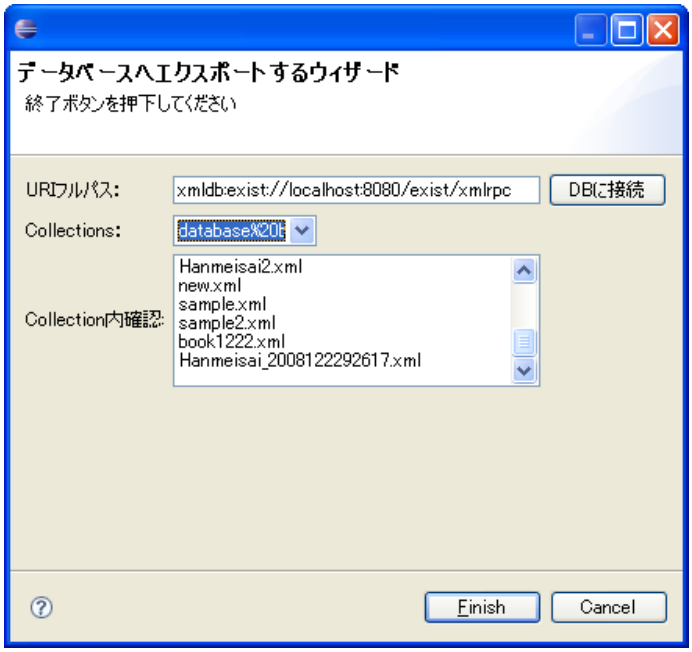

図 4-22　XM 文書のエクスポート実行ウィザード

エクスポートする XML 文書を選択し，図 4-22 に表示するウィザードにおいてエクスポート先の DB の URI とコレクションを指定することでエクスポート（コレクション内に同じ名前の XML 文書があった場合，上書きされる）を行うことができる．この際もインポート機能と同様に，DB に接続する際，ユーザ認証を行う．

エクスポート機能では，エクスポートする XML 文書の情報を入力する必要なく，選択されている XML 文書がエクスポートされるようにすることで，利便性を向上させる工夫をした．

4.3.3　XQuery の実行と実行結果のインポート

eXist XML データベース内にある XML 文書において XQuery を実行し，実行結果を新たな XML 文書として Eclipse プロジェクトにインポートし，エディタで表示する機能である．

XQuery とは，XML 問い合わせ言語のひとつであり，関係データベースにとっての SQL と同様に XML 文書に対してさまざまな問合せを行う言語である．XQuery による問い合わせ結果（XQuery の実行結果）が XML 文書として出力される．また，XQuery は，XML 文書の特定の部分を指し示す XPath から構成される．本機能では，図 4-23 に表示するウィザードにおいて，XPath の入力を行い，内部的に XQuery の作成と実行を行っている．

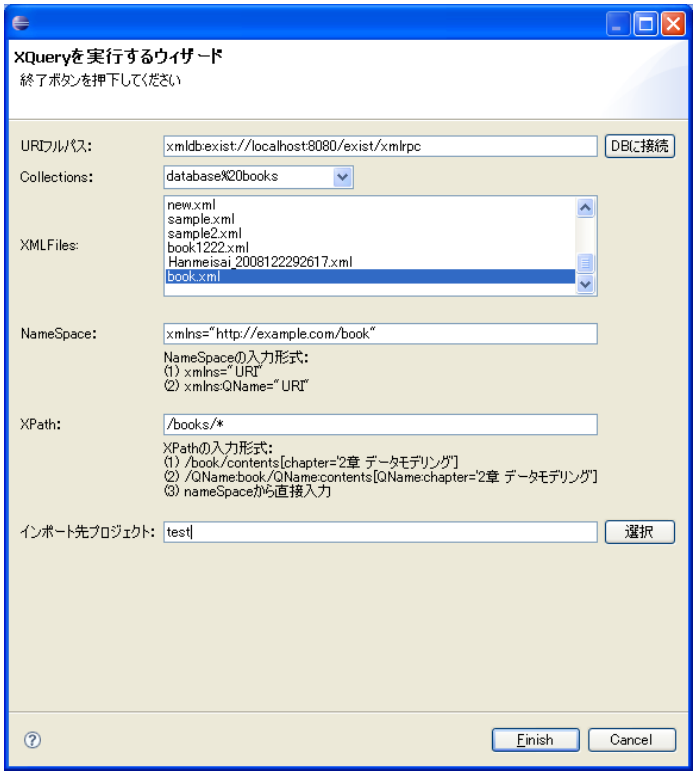

図 4-23　XQuery 実行ウィザード

また，XML 文書は，XML ネームスペース（NameSpace）を持つ場合とそうでない場合がある．XML ネームスペースは，XML 文書で使用する要素や属性の名前を一意に区別するためのメカニズムを提供する．以下に示すように，xmlns が XML ネームスペースのことを表し，URI 参照により特定されるネームスペースに結びつけることができる．

```
<books xmlns="http://example.com/book/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://example.com/books/ file:///c:/xml/books.xsd">
```

XML ネームスペースを持っている XML 文書に対して XQuery を実行する場合，ネームスペースの情報も XQuery に含める必要があるため，図 4-23 のウィザードにおいて，XPath

以外にネームスペースの情報を入力するテキストボックスを設けている．ネームスペースの入力があった場合，内部的にネームスペースと XPath を結合して XQuery を作成し，実行を行う．

図 4-23 において，DB の URI の入力とコレクションの選択までは，4.3.1 項のインポート機能と同じであるが，この場合，XML 文書のリストから一つのみ選択できるようになっており，選択した XML 文書に対して XQuery を実行する仕様となっている．また，XQuery の実行結果のインポート先プロジェクトの入力（または，選択）も，4.3.1 項のインポート機能と同様であるが，XML 文書は，「XML ファイル名__年月日時分秒.xml」（例：sample_20081222142111.xml）の形式で作成される．その理由は，1 つの XML 文書に対する XQuery の実行結果を区別させることである．

第5章　開発の推移と結果

5.1　開発の推移

本プロジェクトにおける開発スケジュールの計画と実績を図 5-1 に示す．プロジェクト終了日程の実績が 12 月 19 日であった．初期スケジュールで計画した 12 月 15 日より，4 日程度の遅れが出たが，ほぼ計画通りに終了することができた．

また，図 5-1 より，イテレーション 1 における実装が大幅に遅れたことが分かる．この理由として，XML エディタとグラフィカルビューの同期に関する処理を見積もっていなかったことが原因として考えられる．この対応策として，集中的に作業を行うなど，チームとして作業時間を増やすことや，作業場所を統一することで意見交換を容易にできるようにしたことが挙げられる．その結果，遅れを取り戻すことができ，納期に間に合わせることができた．

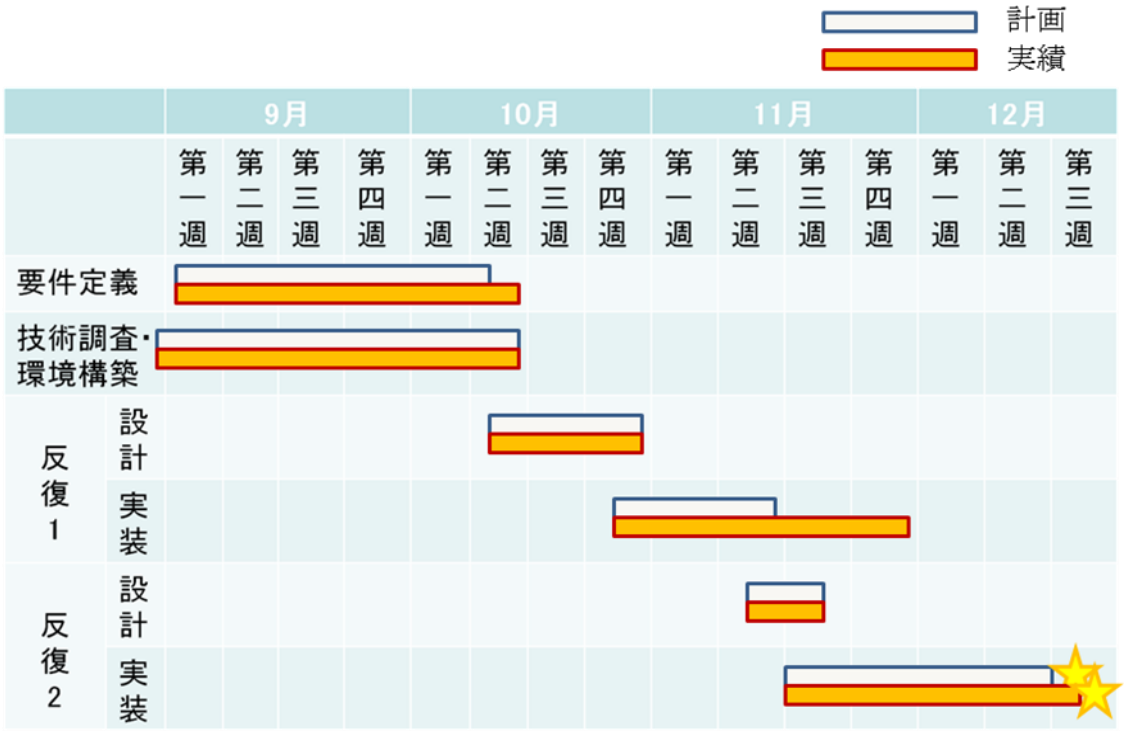

図 5-1　開発の計画と実績

各イテレーションにおける開発機能の実績を表 5.1 に示す．3.3 節で計画した機能の定義の粒度が大きかったため，実績では，機能をさらに詳細化したものを示す．

表 5.1　開発機能の実績

イテレーション（反復）	開発領域	開発機能の実績
1	XML エディタ	テキストビュー
		アウトラインビュー
	グラフィカルビュー	グラフィカルビュー
		プロパティビュー
		ズーミング
		サムネイル
	XML データベース	XML データベースから XML 文書のインポート
		XML データベースから XML 文書のエクスポート
2	XML エディタ	アウトラインビュー上で，要素の追加と削除
		アウトラインビュー上で，要素名の変更
		アウトラインビュー上で，要素の移動
	グラフィカルビュー	Summary ビュー
		要素の格納・展開
	XML データベース	XQuery の実行と実行結果のインポート*

5.2　各工程での成果物

5.2.1　要件定義工程と成果物

図 5-1 に示すように，本プロジェクトでは，要件定義と技術調査を並行的に行った．技術調査により，開発に必要な要素技術を把握した後，要件の決定を行った．その理由は，本プロジェクトは，少人数，短期間で行うため，リソースに対して適切なスコープを決定することであった．

技術調査と並行して要件定義を行う際，チームメンバの役割に応じて分担を行い，要件定義書を作成した．図 5-1 に示すように，要件定義行程をほぼスケジュール通りに終了させることができた．

5.2.2　設計・実装工程と成果物

本プロジェクトは，3 カ月という短期間かつ 3 人という少人数での開発のため，ドキュメントに関しては，設計開発に必要と考えたものを適宜作成し，同時に開発を進めることで，できるだけ早い段階で動くアプリケーションを目指した．

今回のアプリケーション設計では，まず技術調査を行い，それに基づいて開発するアプリケーションのモデリングを行い，それをパッケージ図（付録参照）およびクラス図（付録参照）に表現した．また，各クラスにおいては，クラス仕様書を作成した（付録参照）．

アプリケーション開発においては，パッケージ・クラス図を参考に進めたが，変更があった場合の周知を徹底したことと，各々の開発部分の独立性が非常に高かったため，大きな問題なく開発が進んだ．開発したアプリケーションのプログラムの概要を表 5.2 に示す．

表 5.2　プログラムの概要

ソースコード規模	全体　（作成部分+ 既存 XML パーサ）	13104 step
	作成部分	7676 step
	作成部分　（有効行）	6045 step
クラス数	全体　（作成部分+ 既存 XML パーサ）	92 クラス
	作成部分	52 クラス

5.2.3　テスト工程と成果物

XGraphi の納入に際し，外部から見たアプリケーションの機能を検証するブラックボックステストを行った．まず，テストするアプリケーションの対象領域を，「共通」「テキストビュー」「アウトライン」「Graphical Viewer」「DB 接続」「Summary ビュー」「プロパティビュー」の 7 つの大領域に分け，さらに大領域を表示・アクション等の小領域に分けて，テスト項目の抽出を行った．

チームメンバ各自が担当した機能部分に対するテスト項目を設けたチェックシートを作成し，動作の確認を行った（付録参照）．動作の確認はチームメンバそれぞれが担当機能とは異なる部分のチェックシートのチェックを行った．確認には正常に動作する(○)，時折バグが発生する(△)，正常に動作しない(×)の三段階評価を用い，発見したバグを全て対処するまで繰り返し行った．図 5-2 にチェックシートの一部を示す．

Xgraphi：機能確認シート		項目番号	操作	結果	動作確認	コメント
Graphical Viewer	表示	4-1-1	Graphical Viewerを表示する	XML文書の要素・DTD・テキスト・コメントノードが色分けされている	○	直接記述によるDTDは表示されないようです。
		4-1-2		XML文書のノードと親子関係のコネクションが表示される	○	
		4-1-3		0～500KBかつ要素数が1万個以下までのXMLファイルを開くことができる	○	
		4-1-4		500KB～1MBかつ要素数が1万個以下のXMLファイルを開くことができる	×	512MBでは無理
		4-1-5		1MB以上（大きな）のファイルを開くことができない	○	4-1-4が無理なので
		4-1-6		XMLの文法が間違っているファイルを開くことができない	○	
		4-1-7	XMLファイルを編集し、保存アクションを行う	Graphical Viewがリフレッシュされる	○	
	アクション（要素の移動）	4-2-1	ノードを一つ選択し、ドラッグする	選択したノードが移動し、ノードに接続しているコネクションは接続を維持する	○	
		4-2-2	ノードを複数選択し、ドラッグする	選択した各ノードが移動し、各ノードに接続している各コネクションは接続を維持する	○	選択時の9点の囲みの表示がおかしい？
		4-2-3	要素を拡大、縮小する	選択した各ノードを拡大、縮小することができる	○	
		4-2-4	XMLファイルを編集し、保存アクションを行う	要素の位置が初期化される	○	
	アクション（要素の格納・展開）	4-3-1	子ノードを持っていて、子ノードを展開表示しているノードをダブルクリックする	ダブルクリックしたノードの色が淡青色に変化し、子ノードが格納される	○	
		4-3-2	子ノードを持っていて、子ノードを格納しているノードをダブルクリックする	ダブルクリックしたノードの色が本来の表示色に変化し、子ノードが展開表示される	○	
		4-3-3	子ノードが格納されているノードの親ノードをダブルクリックする	ダブルクリックしたノードの色が本来の表示色に変化し、淡青色のノードが格納され	○	ダブルクリックした親ノードの色は淡青色になった
		4-3-4	子ノードが格納されているノードの親ノードをダブルクリックする	ダブルクリックしたノードの色が本来の表示色に変化し、淡青色のノードが展開される	○	
		4-3-4	子ノードを持っていないノード（テキスト、コメント）をダブルクリックする	何も起きない	○	
		4-3-5	XMLファイルを編集し、保存アクションを行う	要素の位置が初期化され、すべて展開表示される	○	
	アクション（ズーミング）	4-4-1	Graphical Viewer内の要素をクリックし、フォーカスをあてる	ツールバーの「虫眼鏡」と「倍率コンボボックス」が使用可能になる	○	
		4-4-2	4-4-1が正常に動作している状態でツールバーの「+-虫眼鏡」をクリックする	Graphical Viewerの表示領域がズームイン・ズームアウトする	○	
		4-4-3	4-4-1が正常に動作している状態でツールバーの「倍率コンボボックス」から適当な倍率を選択する	Graphical Viewerの表示領域が指定した倍率に変化する（1%～300%、ページ、高さ、幅）。それに伴い、サムネイル表示も変化する。	○	
要約	表示	6-1-1	要約を表示する	ファイル名、XMLのバージョン、エンコード、デフォルト名前空間、ルート要素、総ノード数、コメント数が表示される	△	XMLバージョン情報が表示されない
		6-1-2	XMLファイルを編集し、保存アクションを行う	要約が更新される	○	
プロパティビュー	表示	7-1-1	eclipseのツールバーの「ウィンドウ」-「ビューの表示」-「その他」-「一般」-「プロパティー」を選択する	プロパティビューが表示される	○	
		7-1-2	Graphical Viewerにおいて要素ノードを選択する	プロパティビューにタグの要素名・要素値・属性名・属性値が表示される	○	

図 5-2　テスト用のチェックシート

5.2.4　マニュアルおよび保守用ドキュメント

本アプリケーションに関するマニュアルは，Google Site を利用した．Google Site とは，Google のアカウントを持つ者同士が編集・閲覧が可能な Wiki に似た WEB サイトの事であり，インターネットに接続できる環境であれば，どこからでも閲覧・編集ができる．Google Site を採用した理由は，天笠講師からマニュアルを WEB ベースで作成するよう依頼があったためである．作成したマニュアルの一部を図 5-3 に示す．

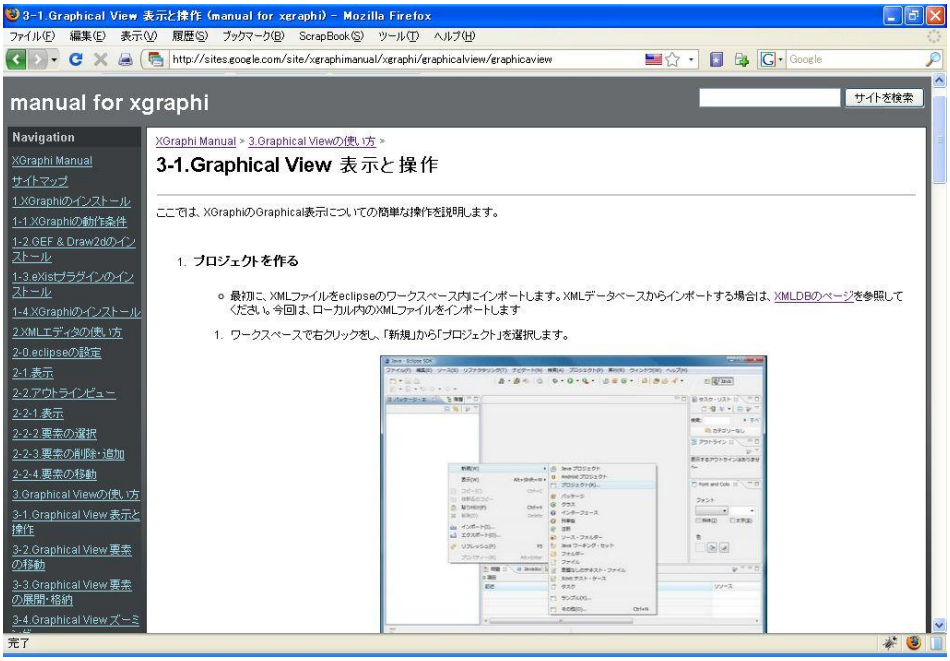

図 5-3　作成した操作マニュアル

また，保守用のドキュメントについては，今後の拡張を考慮して，開発したソースコードとともに Javadoc を添付した．Javadoc とは， Java のソースコードから，Java クラスの仕様書における標準書式で，HTML 形式の仕様書を生成するものであり，開発したアプリケーションの API の理解を支援するドキュメントである．これを作成するために，コーディングでは Javadoc の出力形式に対応するよう，コメントの書き方を統一した．作成した Javadoc の一部を図 5-4 に示す．

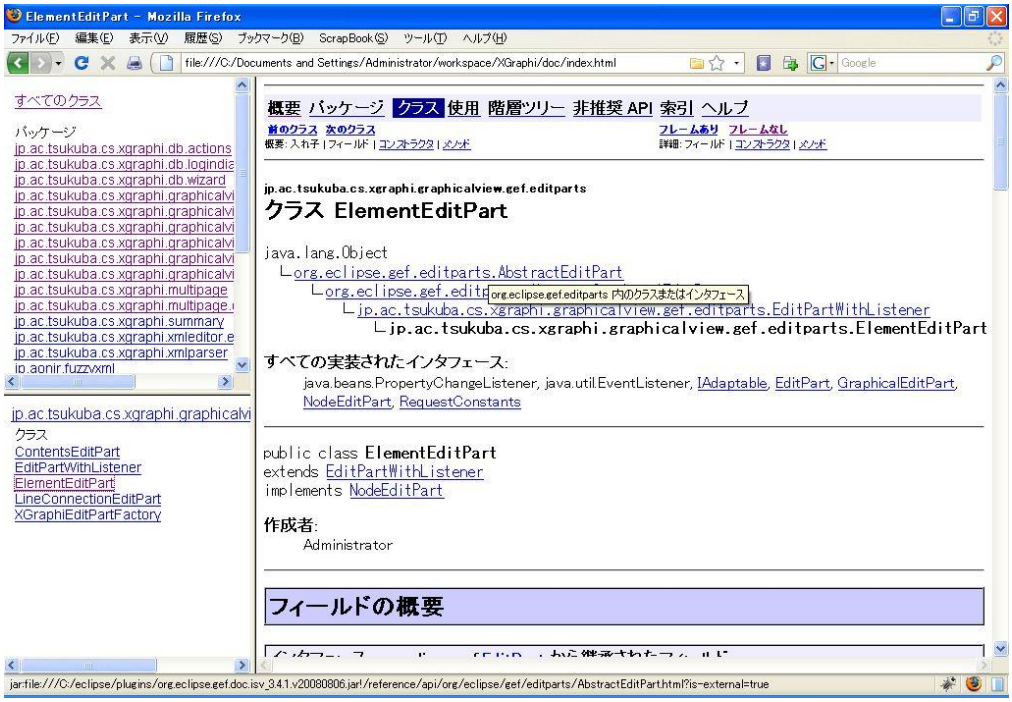

図 5-4　作成した javadoc

5.3　XGraphiの評価

5.3.1　既存の XML エディタとの比較

XGraphi と既存のアプリケーションを比較することで, XGraphi の機能的特徴を説明する. 比較対象には既存アプリケーションの一例として，2 章で紹介した XML Notepad 2007 と eXist を用いる.

XGraphi は一般的な XML エディタにおける XML データの表示・編集に関係する機能を実装している．具体的にはアウトラインビューやテキストビューがこれにあたる．さらに XGraphi は 4.3 節で述べた DB 接続機能を持っている．DB 接続機能は一般的な XML エディタには実装されていないが，CVS を始めとするネットワーク接続に対応している Eclipse としては基本機能にあたるといえる．表 5.3 は XGraphi と XMLNotepad 2007，及び eXist の基本機能を比較したものである.

表 5.3　既存の XML エディタとの機能比較

機能	XMLNotepad2007	eXist	XGraphi
XML 文書のテキスト表示と編集	○	○	○
XML 文書のアウトライン表示とアウトライン上での操作 (要素の追加・削除・名前変更)	○	×	○
XML 文書全体を把握する	△	×	○
XML データベース内の XML 文書の取り扱い	×	○	○

XGraphi は一般的な XML エディタには無い，独自の機能を備えている．4.2 節で述べたグラフィカルビューがこれにあたる．XML データをグラフィカルに表示するというコンセプトで制作されたアプリケーションとしては，東芝プロセスソフトウェア株式会社の「Altova XMLSpyⓔ」がある．しかし， これは XML の編集を目的としており，XGraphi の様に学会などでの発表資料作成を目的としていない．そのため図 5-5 にある「Altova XMLSpyⓔ」の実行画面からも分かるように，第三者がビューを見て XML のデータ構造を一目で把握することは難しい. このようなことから XGraphi のグラフィカルビューは製品レベルでは他に例を見ない独自機能であるといえる．

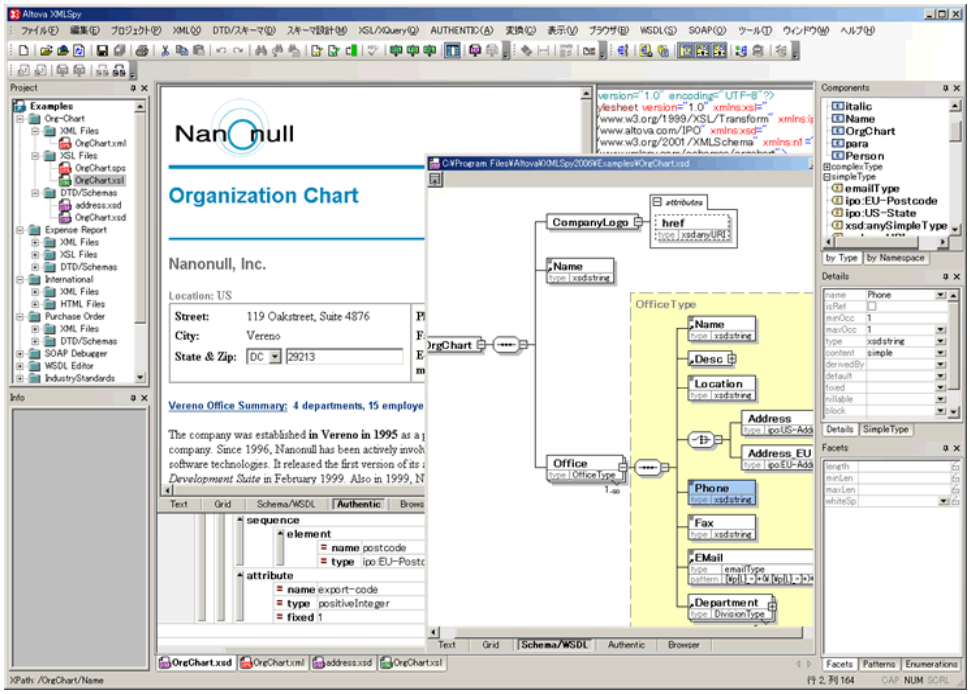

図 5-5　Altova XMLSpyⓔ

5.3.2　委託元からの評価

XGraphi の最終確認として，12 月に天笠講師の協力を得て評価を行った．評価項目にはテスト工程で用いたチェックシートにおいて，ユーザ視点であるものを抽出して利用した(付録参照)．評価方法は実際にアプリケーションを天笠講師に使用していただき，5.2.3 項で述べた 3 段階の動作確認及び小領域毎のユーザビリティ(使い易さ・見易さ)を 5 段階で点数付けするものである．ユーザビリティを評価する主な目的は，文面に表れ難い顧客の満足度を把握し，向上させる事である．ユーザビリティ評価の結果を図 5-6 に示す．

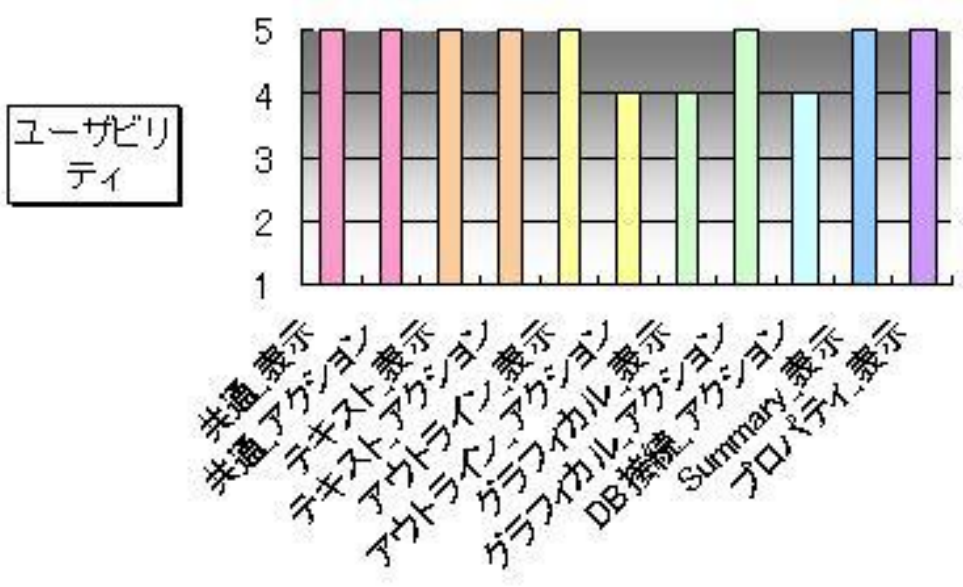

図 5-6　ユーザビリティの評価結果

図 5-6 の横軸は小領域が並んでおり，縦軸は小領域の評価点数の平均である．評価点は 5 に近い方が良いと定めている．概ね高評価であるが，アウトラインビューのアクションとグラフィカルビューの表示，及び DB 接続のアクションに関する部分で評価が低下している．この原因は主に二つある．一つは確認シートの説明文が誤解を与える内容になっていたことである．これには確認シートの修正を行うことで対処した．二つ目の原因は再現性のないバグが発生したことである．内部評価の段階では抽出できなかったバグがこの段階で顕在化した．これはバグ原因の特定を優先的に行い，対処した．

第6章　XML エディタの開発

本章では筆者が開発を担当した XML エディタの実装について説明し，その実装上の問題点及び改善案について述べる．

6.1　テキストビューの実装

4.1 節で述べたように, テキストビューは XML 要素の種類毎に異なった色を割り当てることで, XML 文書を直接編集する際に，どの部分を編集しているのかを分かり易くしている．XML 文書内では要素・属性・処理命令・CDATAsection・DTD・コメントなどが記述されるが，その他にも空白・タブ・改行などの XML が表現するデータとは関係がない構成要素も存在する．これらの構成要素を区切り，配色するために以下の役割をもつ 3 種類のスキャナを実装した．スキャナは，テキストの中から条件に合致する部分文字列を検出するものである．

テキストビューは，編集対象となる XML 文書をテキストファイルとして読み込み，以下の 3 種類のスキャナを用いて構成要素を区切り，テキストを表示する際に配色を行う．

- 空白・タブ・改行を検出するスキャナ
  XML 文書内にある「　」「¥n」「¥t」を検出するものである．図 6-1 に検出の例を示す．

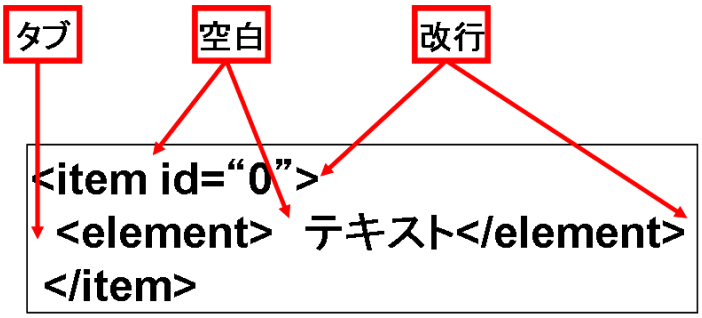

図 6-1　空白・タブ・改行の検出

タブ・空白・改行は「＜＞」で囲まれた XML タグの内外に存在する可能性がある．タグやテキスト（図 6-1 の「テキスト」という文字列）の外側に存在した場合は，XML データの構造上意味を為さないものとして無視する．また，タグやテキストの内側に存在した場合，XML データに関わる部分として取り扱う．

- タグを検出し，要素・処理命令・CDATAsection・DTD・コメントを検出するスキャナ

XML 文書内にある「＜＞」で囲まれたタグを検出し，開始タグ「＜」以降の文字で各構成要素に振り分けるものである．表 6.1 に開始タグ以降に続く文字と振り分けの対応関係を示す．

表 6.1　各構成要素の振り分け

開始タグ以降に続く文字	構成要素の種類
?	処理命令
![CDATA[	CDATAsection
!DOCTYPE	DTD
!--	コメント
その他	要素

- 要素内の属性を検出するスキャナ

先に述べたタグを検出するスキャナにより，要素として認識されたタグ内の属性を検出するものである．属性を XML で表現する場合，図 6-1 の「id=”0”」のようにダブルクオート又はシングルクオートで属性値を囲うという制約がある．この制約を利用して要素内のダブルクオート又はシングルクオートを検出し，囲まれた値を属性値として取り扱う．

6.2 グラフィカルビューとの連携

XML エディタはグラフィカルビューとは独立して開発を行った．XML エディタとグラフィカルビューを連携させ，一つのアプリケーションとして完成させる過程で，アウトラインビューの実装を行った．

6.2.1　アウトラインビューの実装

アウトラインビューは 4.1 節で説明したような操作機能を，テキストビューとグラフィカルビュー上で実現する必要があった．

アウトラインビューを実装するために満たすべき条件は以下の通りである．

[条件 1]アウトラインビューにツリー状のデータモデルを適用すること

アウトラインビューは XML データのツリー構造を表すものである．このため，XML データをツリー構造として表現するデータモデルを適用する必要がある．

[条件 2]アウトラインビューとグラフィカルビューに適用しているモデル同士が同期すること

アウトラインビュー上での操作結果は，グラフィカルビューにも反映しなければならない．このため，アウトラインビューに適用しているモデルの変更があった場合，グラフィカルビューに適用しているモデルの変更を行うように，モデル同士を同期させる必要がある．

[条件 3]アウトラインビューが利用しているモデルとテキストビューが参照しているテキストデータが同期すること

アウトラインビュー上での操作結果はテキストビューにも反映しなければならない．また，テキストビュー上の変更も同様にアウトラインビューに反映する必要がある．どちらかが利用しているデータに変更があった場合，もう一方のデータにも変更を適用するように両データを同期させなければならない

これらの要件を満たすために,図 6-2 に示すような XML 文書とツリーモデルを利用した．図 6-2 において XML 文書をツリーモデルに変換する際は，自作した XML パーサを利用して変換している．このツリーモデルは，チームメンバがグラフィカルビューを開発する際に設計を行っている．

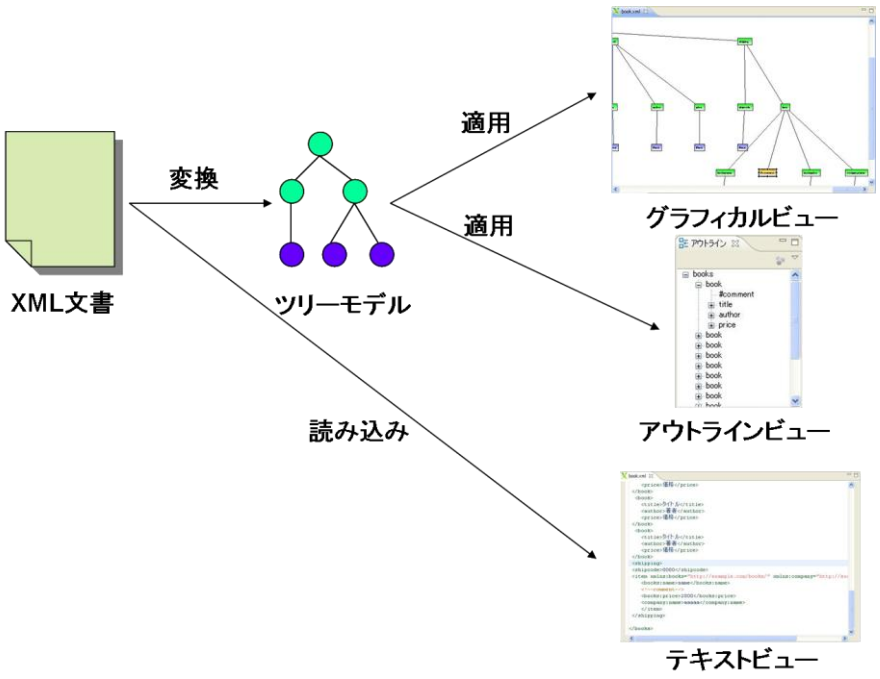

図 6-2　各ビューが利用するデータ

まず，[条件 1]及び[条件 2]に対しては，グラフィカルビューに適用されているモデルと同一のモデルをアウトラインビューに適用することで対応した．このモデルはルート要素を最上位ノードとしたツリーで構成されており，[条件 1]と[条件 2]を満たす上でアウトラインビューに適したモデルであるといえる．これにより不必要な新モデルを作成することなく，ア

ウトラインビュー上での操作結果をグラフィカルビューに反映することを可能にした．図 6-3 にその様子を示す．

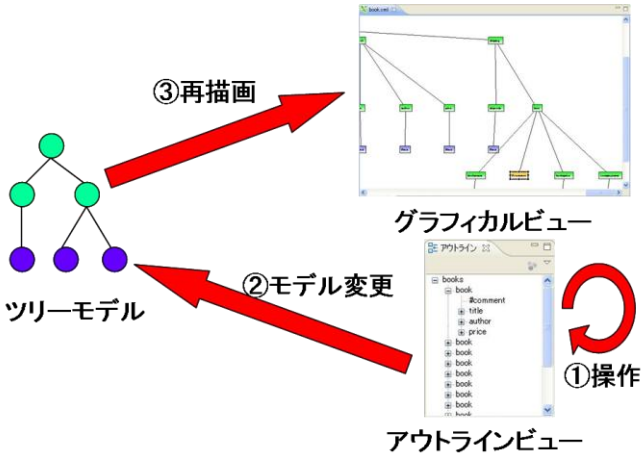

図 6-3　アウトラインビューからグラフィカルビューへの反映

アウトラインビュー上で要素の追加など，ツリーモデルに変更を及ぼす操作が行われる（①）．モデルの変更が通知（②）されツリーモデルが変更される．ツリーモデルが変更されたことを受け取り，グラフィカルビューは新たなツリーモデルを元に再描画を行う．

次に[条件 3]を満たすために行った実装を説明する．6.1 節で説明した通り，テキストビューは XML 文書をテキストファイルとして直接読み込む仕様にしている．このため，アウトラインビューに適用しているツリーモデルの変更を，元の XML 文書に反映し，通知するアルゴリズムを開発した．図 6-4 にその様子を示す．

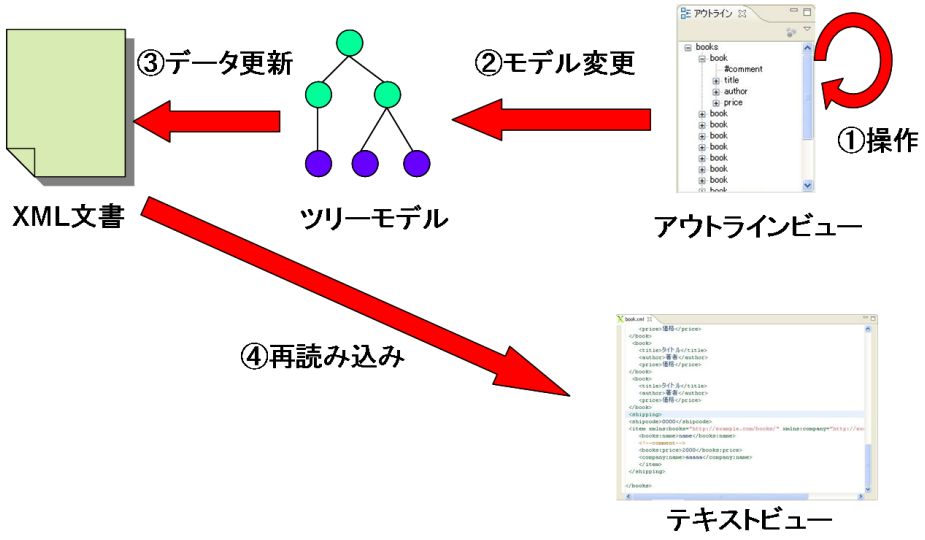

図 6-4　アウトラインビューからテキストビューへの反映

アウトラインビュー上で要素の追加など，ツリーモデルに変更を及ぼす操作が行われる（①）．ツリーモデルの変更後（②），変更部分を伝え，元の XML 文書を更新して保存する（③）．テキストビューは更新された XML 文書を再読み込みする（④）．

また, テキストビューによって XML 文書の変更が行われた場合は, 図 6-4 とは逆に, XML 文書の変更をツリーモデルが受け取り，アウトラインビュー及びグラフィカルビューを再描画する.

このようにアウトラインビューにおける操作機能を開発するに先立ち，グラフィカルビュー・テキストビュー・アウトラインビューと各ビューが利用するデータの関係を整理することで，見通しの良い開発を行うことができた.

しかし，先に述べたツリーモデルはグラフィカルビューで利用することを想定して設計していたため, 4.1.2 項で説明したフォーカス機能の実装に必要な情報をツリーモデルが保持していないという問題が発生した．この問題は，フォーカス機能が開発中期に抽出した機能要件であり，ツリーモデルの設計時には考慮しきれなかったことに起因する．フォーカス機能を実装するために，次で説明するツリーモデルの拡張を行い，この問題に対処した

6.2.2　アウトライン上での選択と各ビューのフォーカス機能の実装

アウトラインビューは，ツリー要素を選択するとテキストビューやグラフィカルビューにおいて対応した要素の周辺をフォーカスする機能を持つ．この機能の実現手法を，グラフィカルビューとテキストビューに分けて説明する.

- グラフィカルビューのフォーカス

アウトラインビューとグラフィカルビューは 6.2.1 項で述べた通り，同一のツリーモデルを利用している．このため，容易にフォーカスする部分を指定することが可能である．図 6-5 にアウトラインビュー上での選択をトリガとして，グラフィカルビュー上でフォーカスが行われる様子を示す．グラフィカルビュー上で選択が行われた際は，図 6-5 と逆の過程を辿り，アウトラインビューの対象要素がフォーカスされる.

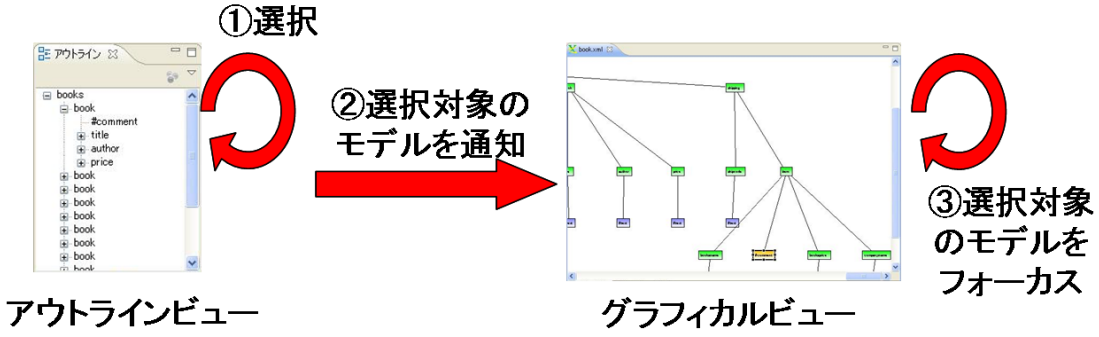

図 6-5　アウトラインビュー上での選択とグラフィカルビューのフォーカス

- テキストビューのフォーカス

テキストビュー上でフォーカスを当てるためには，テキストファイルの先頭位置（0）から何文字分ずれた位置なのかを表すオフセット値を指定する必要がある. 6.2.1 項で述べた通り, XML 文書からツリーモデルへ変換する際に自作の XML パーサを利用しているが, この XML パーサはオフセットを計算していないため，ツリーモデル中の各ノードは自身

のオフセット情報を保持していないのである．そのため，アウトラインビュー上で選択操作が行われた場合でも，テキストビューに対して，オフセットを指定することができないという問題が発生した．この問題に対処するために，FuzzyXML パーサを導入し，各ノードがオフセット値を保持できるようにツリーモデルを拡張した．FuzzyXML パーサの特徴は，パースして生成したツリーモデルにおける各ノードが，オフセット値を保持しているという点である．この特徴を利用して，自作の XML パーサが生成したツリーモデルの各ノードに対し，FuzzyXML パーサが生成したツリーモデルの各オフセット値を付加するという手法を取ることで，独自にオフセット計算アルゴリズムを開発するよりも，短期かつ確実に開発を進めることができた．

オフセット値を保持したツリーモデルを利用したテキストビューがフォーカスを行う様子を図 6-6 に示す．

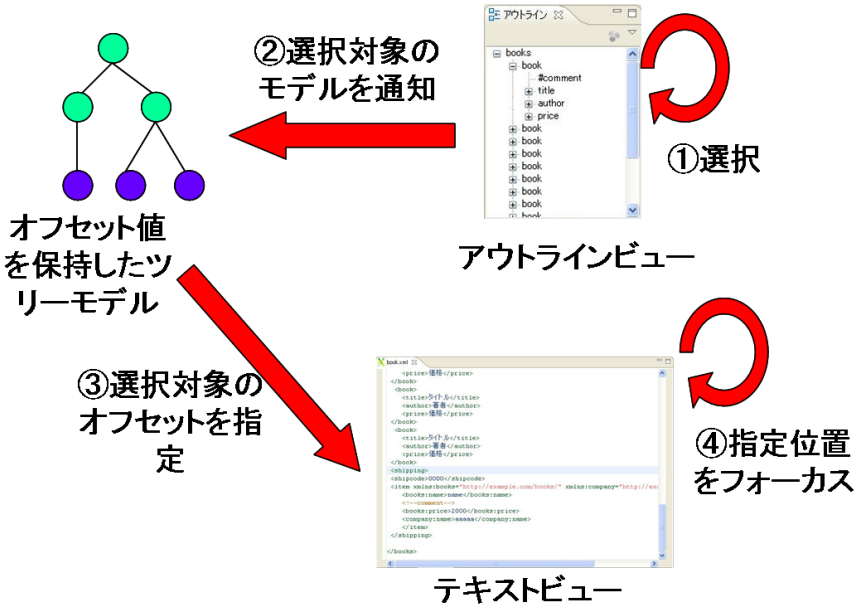

図 6-6　選択とテキストビューのフォーカス

アウトラインビュー上で要素の選択が行われる（①）．ツリーモデル内で対応するノードが通知され（②），それを元にテキストビューに対してオフセット値を指定する（③）．テキストビューは指定されたオフセット値を利用し，フォーカスを行う（④）．

6.3 問題点と改善案

6.3.1 問題点

開発を担当した XML エディタ領域における問題点として，利用者が明示的にファイルの保存（「名前を付けて保存」など）をしなくても，XML 文書が自動的に保存されてしまうという点が挙げられる．一例としてアウトラインビュー上で要素の追加操作が行われた場合を挙げる．この場合，図 6-4 中の③に示した通り，一度 XML 文書が自動的に保存される．

このように，操作を行う度に XML 文書が更新されるため，編集内容を破棄したとしても編集内容が XML 文書に反映されたままになってしまう．編集内容を破棄した場合，XML 文書は編集前の状態であることが望ましい．

6.3.2 改善案

利用者が明示的に保存を行わない限り，元の XML 文書が編集前の状態を維持するための改善案を図 6-7 に示す．

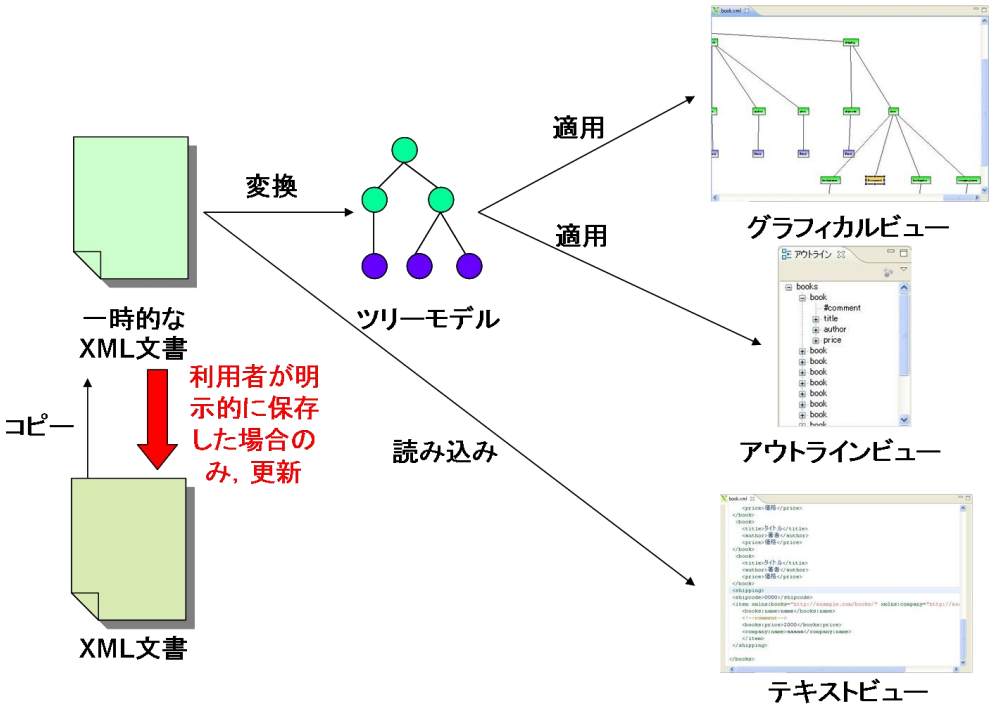

図 6-7　改善案におけるデータの取り扱い

改善案では各ビュー・モデルが直接，元の XML 文書を取り扱わず，元の XML 文書をコピーした一時的な XML 文書を取り扱う．このため各ビューで編集が行われ，自動的に保存が行われた場合でも，一時的な XML 文書が変更されるのみであり，元の XML 文書は変更されることはない．また，一時的な XML 文書から元の XML 文書への更新処理を，利用者が明示的に保存を行った場合のみ行うようにすることで，問題点を解決することが可能である．本改善案は，現行のプログラムを廃棄することなく，拡張を行うことで実現可能である．このことから本改善案は開発効率の観点においても有効なものであるといえる．

第7章　結論

本プロジェクトでは，XML データ管理インタフェース「XGraphi」を開発した．XGraphi は基本的な XML 文書の編集操作や DB への接続など多くの機能を実装しながら, XML 文書をグラフィカルに表現するという独自の機能を実装している. これにより XGraphi は通常の XML エディタとしても利用可能であり, さらに XML 文書全体を俯瞰するような場面においても有用である.

本プロジェクトは，概ね当初計画した通りに高品質なアプリケーションを納入することができたことから，成功したプロジェクトであるといえる．XGraphi が多くの利用者を獲得することを期待している.

また，筆者が開発を担当した XML エディタについては，全体として見通し良く行うことができた．開発中はアウトラインビュー上での選択による各ビューのフォーカス機能を実装する場面で，アウトラインビューが利用しているツリーモデルの情報不足という問題が発生した．この問題は，この機能要件の抽出が中盤以降であったため，ツリーモデルの設計時に考慮しきれなかったことに起因する．既存の技術である fuzzyxml パーサを活用することでこの問題に対処したが，反復型開発を行う上での設計の難しさ・リスクを改めて認識することになった.

XML エディタは納入後に行った評価により，XML データ管理インタフェースとして十分な XML 文書の編集機能を備えていることが分かった. しかし現状の実装では, 編集中に度々 XML 文書が自動的に保存されるため，編集内容の破棄が不可能であるという問題が残っている．今後，システムやアプリケーション開発において，このような機能要件として抽出し難い要件に対する配慮を，より慎重に行っていきたい.

謝辞

本研究開発プロジェクトを進めるにあたって，委託者としての立場に留まらず積極的にご指導いただいた天笠俊之講師に深く感謝致します．私たちからの急なご連絡にも常に明快かつ丁寧なご指導をいただきました．

指導教員である田中二郎教授には，私が研究室に入ってから 3 年に渡ってアドバイスをいただきました．至らない点が多くあったと思いますが，ご指導ありがとうございました．

「高度 IT 人材育成のための実践的ソフトウェア開発先週プログラム」専任教授である菊池純男教授，駒谷昇一教授には 2 年もの間，熱心にご指導いただきました．また，講義時間だけでなく学校生活においても親身に相談に乗っていただきました．この場だけでは表わしきれない程の感謝の念を抱いております．本当にありがとうございました．

また，本研究開発プロジェクトで同じプロジェクトチームだった柿沼基樹君，ラトナマララシト君にもお世話になりました．二人のプロジェクトに臨む姿勢は，大変勉強になりました．

最後に私を公私共にサポートしていただいた本研究科の同期生ならびに関係者の方々，さらに温かく見守っていていただいた私の家族にも心から感謝致します．