効率の良いWeb開発のための
既存資源再利用技術に関する研究

慶應義塾大学大学院理工学研究科
開放環境科学専攻
安部 麻里

2005年度

はじめに

Web はその役割を，発祥当時のあらかじめ作成された情報を配信するための静的情報配信から，大量のデータや日々更新される情報を基にユーザの要求に応じて動的にコンテンツを生成し配信する動的情報配信，さらには商取引や各種手続き等のアプリケーションを提供するためのインタフェースとして Web を用いる Web アプリケーションへと広げてきた．同時に，Web を提供するためのサーバ側の技術も様々な発展を遂げてきた．また，Web コンテンツの記述様式の進化，コンテンツ中で利用される JavaScript や Flash 等の技術の登場，携帯電話端末などに代表される Web 閲覧手段の多様化など，Web を取り巻く環境の変化は著しい．

これらの多様化に伴い，システム・コンテンツ共に複雑化した Web を効率良く開発するためには，フォワードエンジニアリング技術であるコンテンツ管理システムや開発支援ツール，モデル駆動型開発手法などを利用することは勿論のこと，既存資源を有効活用するリエンジニアリング技術を欠かすことが出来ない．

情報発信の分野において，例えば他国語での閲覧への対応など，当初の目的とは異なった形で既存の Web コンテンツを再利用するためには，再利用目的に合わせた情報の意味づけを行うことが効果的である．しかしながら，Web コンテンツに追加情報を付加するための共通の枠組みが不十分であるため，その再利用は十分に進んでいない．また，Web アプリケーションの分野においては，特にソフトウェア基盤の変更時などにおいて，現在稼動中のシステムの仕様書やプログラムなど，既存資源の有効利用が十分行われているとは言い難い．

本論文では，まず情報発信における Web コンテンツを対象として，アノテーションと呼ばれる付加情報を付与する枠組みを提案する．提案手法を用いることにより，既存 Web コンテンツを変更することなく，情報携帯端末など様々なクライアントデバイスに対する Web コンテンツを作成するための情報の付与や，ユーザの嗜好に合わせて選択的に情報を集積するための情報の付与など，Web コンテンツの多目的への適応を支援することが可能となる．本論文では，提案手法の適用例として，多様な端末からの Web 閲覧に対応するための Web トランスコーディングと，ポータルサイト構築のための Web クリッピングについて述べ，その有用性を示す．

次に，Web をアプリケーションのユーザインタフェースとして利用する Web アプリケーションを対象に，MVC に基づくモデル駆動型開発の利点を紹介し，統合開発環境上に実装した開発支援ツールについて述べる．さらに，既存アプリケーションの振舞いを，HTTP 上で授受される情報に着目した動的解析により明らかにし，モデルの抽出を支援する手法を提案する．提案手法を用いることにより，Web アプリケーションの振舞いを表すモデルの抽出が可能となり，例えば CGI を基盤として開発された既存アプリケーションを J2EE プラットフォームへ移行するなど，実行

環境の移行に際し，モデル駆動型開発への円滑な移行を支援することが可能となる．実際の Web アプリケーションを対象とした実験により，提案手法を用いることでより多くの要件定義を満たすモデルの抽出が可能となり，既存資源をより効果的に利用可能であることを示す．

以上の様に，本論文で提案する二つの提案手法により，Web による情報発信および Web アプリケーションの双方において既存資源の有効活用を支援し，効率の良い Web 開発の実現に寄与することが可能となる．

謝辞

本論文をまとめるにあたり，学部以来御指導，御教授頂きました大野義夫教授に深く感謝致します．また，本論文の審査委員を御快諾して頂き，懇切丁寧な御指導を頂いた櫻井彰人教授，山本喜一助教授，遠山元道専任講師に深く感謝致します．

日本アイ・ビー・エム株式会社東京基礎研究所では，所長の久世和資氏をはじめ所員の皆様に本研究を遂行する上で様々な御支援を頂きました．ここに記して感謝の意を表します．特に，惜しみない議論と懇切丁寧な御指導を頂いた福田健太郎氏，田井秀樹氏をはじめとする共同研究者の皆様に心よりお礼申し上げます．また，入社以来直接御指導頂き，新たな研究分野に挑戦する機会と貴重な御助言を頂いた関西大学総合情報学部の堀雅洋教授に深く感謝致します．

最後に，研究活動を応援してくれた常に明るく前向きな家族に心から感謝します．

目 次

図 目 次

表 目 次

第1章　序章

1.1　World Wide Webの発展と課題

近年，World Wide Web[1] (以下，Web) は情報の発信・収集のための手段にとどまらず，電子商取引，遠隔教育，コミュニティの形成など，様々な分野において活用され，新たな社会基盤となりつつある．1990 年代初頭に出現した Web は，その普及に伴い，予め用意された Web コンテンツをリクエストに応じて返信する静的コンテンツ配信 (図 1.1) から，大量のデータや日々更新される情報を効率よく配信するための動的コンテンツ配信 (図 1.2) へと発展してきた．さらに，電子商取引や事務手続き等を電子的に実行するためのインタフェースとして Web を用いる Web アプリケーション (図 1.3) の出現が Web の発展に大いに寄与してきた．Web アプリケーションでは，フォームへの入力や項目の選択などアプリケーションの実行に必要なデータは，Web ブラウザ上でユーザの対話的操作により入力され，送信されたデータに基づいてサーバ側での処理が実行される．この様に，Web アプリケーションを用いることで，クライアント側に専用のソフトウェアをインストールすることなく様々な処理を実行することが可能になるという利点がある．

Web に関する技術の発展は，その提供形態にとどまらず，HTML[2]，XHTML[3]，CSS[4] など記述様式の発展や，JavaScript[5], Java Applet[6], Flash[7] など Web 中で利用される技術の充実，閲覧手段の多様化などが挙げられる．例えば，Web の閲覧手段は，デスクトップ上で用いる Web ブラウザから，音声合成技術を利用し Web コンテンツを音声で読み上げる音声ブラウザ [8-10], 携帯端末等の小画面デバイスでの閲覧を可能とする専用ブラウザ [11-13] 等多様化してきた．

Web を取り巻く環境の多様化に伴い，Web 開発のために要求される技術は HTML や Web サーバに関する知識のみにとどまらず，大規模データの管理やサーバ側でのロジックの開発等，多岐に渡る様になった．その結果，例えば複数のデータベースから構成されるシステムやサーバ間の連携を伴うシステムなど，構築されるシステムも複雑なものとなる場合も多い．一方，多様化する Web クライアントや個々のユーザの嗜好に対応するためには，Web コンテンツをそれぞれの環境にあわせて開発・提供する必要がある場合も多く，それぞれを個別に作成した場合，開発コストの増大を招くだけでなく，管理も大変なものとなる [14].

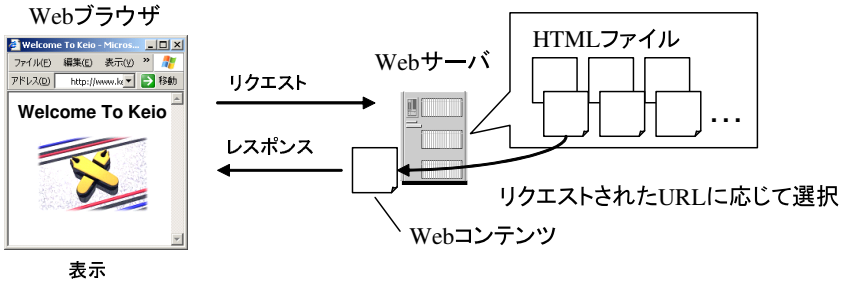


図 1.1: 静的コンテンツ配信の例

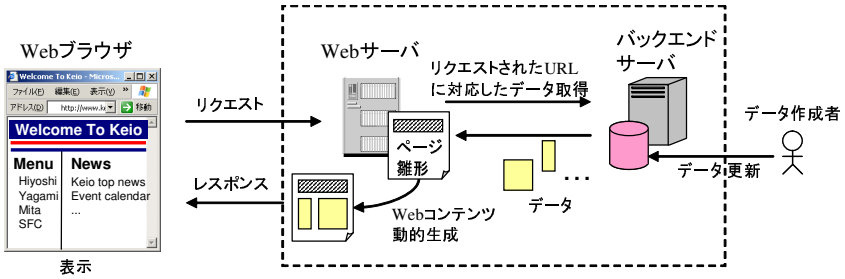


図 1.2: 動的コンテンツ配信の例

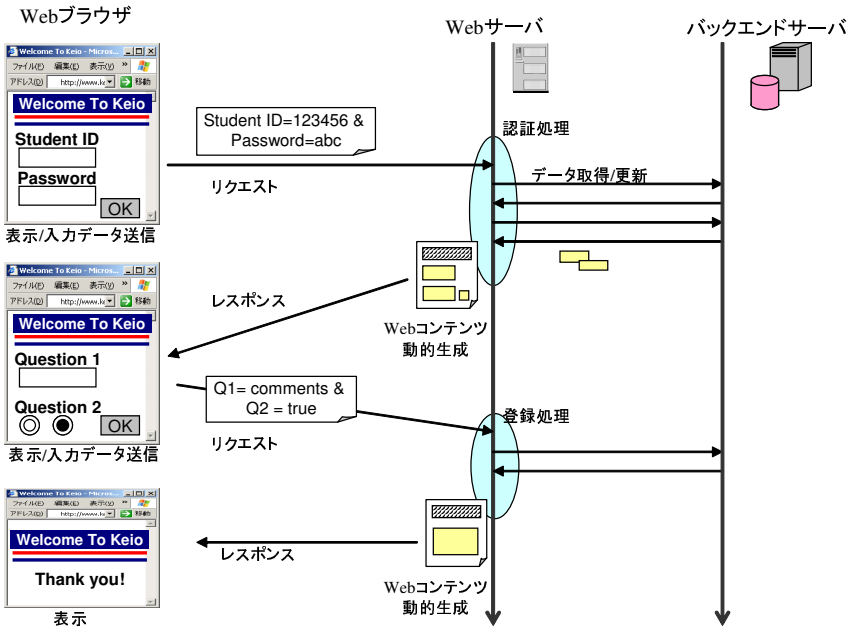


図 1.3: Web アプリケーションの例

この様に，Web 提供形態の多様化や，それに伴うシステムの複雑化に対し，効率の良い Web 開発を実現するための手法が必要とされてきた．図 1.4 に Web における一般的なライフサイクルを示す．効率の良い開発を実現するためには，一般的にフォワードエンジニアリングとリバースエンジニアリングの 2 種類のアプローチがある [15, 16]．フォワードエンジニアリングは，要件定義から設計，開発という一連の流れにおける効率を向上させるためのアプローチである．例えば，GUI を用いた編集機能や妥当性確認機能，プログラミング，テスト環境などを提供する統合開発環境 [17-20] を用いて開発を行うことで，質の良い成果物を効率良く開発することが可能となる．

一方，リバースエンジニアリングは，現状のシステムにおけるソースコード等の分析や，システム内部構造を抽象化した情報等を用いて，開発者がシステムの問題を発見する作業や，実際のシステム構成を理解することを支援するためのアプローチである．例えば，Web サーバにおける構成要素を，静的コンテンツ (HTML 等) や動的コンテンツ (JSP[21] 等)，データベースなどに分類し，それらを解析して依存関係を視覚化することによりシステム全体の理解を助けることが可能となる [22]．

また，これら 2 種類のアプローチを組み合わせて利用することにより，新たな要件に応じたシステムの変更を効率よく実施することをリエンジニアリングという [15]．リエンジニアリングを円滑に進めることで，開発サイクル全体における効率を上げることが期待される．しかしながら，Web におけるリエンジニアリングでは，Web を取り巻く環境の変化が著しいことなどから，過去に開発された Web コンテンツやロジック等の既存資源が再利用出来ないという課題も多く指摘されている [22, 23]．

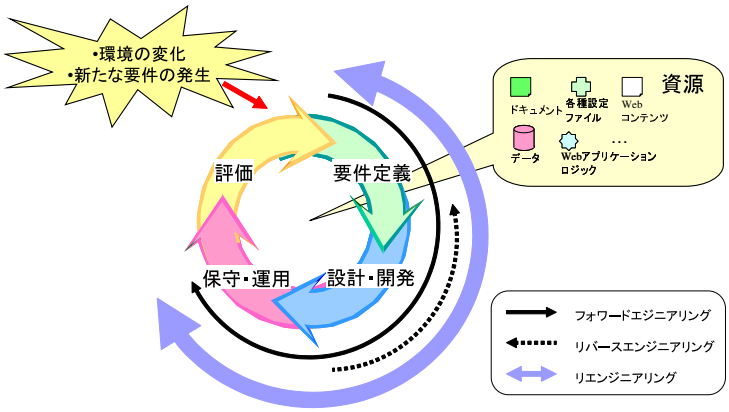


図 1.4: Web におけるライフサイクル

例えば，情報発信の分野 (図 1.1, 1.2) におけるフォワードエンジニアリング手法として，ページデザインと内容を分離して開発・管理するためのコンテンツ管理システム等が実用化されている [24-26]．これらのシステムにおいては，ページの背景やレイアウト，ヘッダ，フッタ等を規定するデザインテンプレートを利用することで，文書作成者はページのデザインを意識せずに文書の内容の作成に集中することが可能となる．一方，ページデザイナはデザイン作業に集中するこ

とが可能となり作業の効率化が期待される．さらに，Web サイト全体でデザインの変更が必要となる場合は，デザインテンプレートを変更することにより，内容を修正することなく大量の Web コンテンツを一度に変更することも可能となる．この様に，開発の初期段階において開発者及び開発成果物双方の役割分担を明確にし，開発成果物を分割・管理することは，リエンジニアリングの観点からも非常に重要である．

一方で，開発時には想定し得なかった形で Web コンテンツの再利用が必要になることもある．例えば，Personal Digital Assistance (PDA) や携帯電話など近年の携帯端末の急速な普及・発達により，携帯端末からの Web へのアクセスが日常的に行われるようになり，デスクトップを用いた Web 閲覧と同等のサービスを携帯端末向けに提供する必要が出てきた [27-29]．しかしながら，携帯端末では画像サイズや色数，表示可能なマークアップ言語等の制約があることも多く，また端末の性能の向上に伴いその制約が変更されることなどから，Web コンテンツをそのまま利用することは困難である [30]．また，Web が社会基盤へと変化を遂げるにつれ，Web コンテンツのアクセシビリティ [31] に対する配慮も必要となった．例えば，視覚障害者の多くは音声ブラウザ [8-10] やスクリーンリーダー [32, 33] 等を用いて音声により Web を閲覧しており，画像への代替テキスト付与や，見出しタグを用いた論理構造化など音声閲覧への配慮がなされていないページを理解することは難しい [8, 34]．さらに，Web は世界中のどこからでもアクセスが可能であることや，近年の国際化の潮流から，従来利用されてきた Web コンテンツを多言語に翻訳する必要が生じつつある [35](図 1.5).

図 1.5: Web コンテンツに対する新たな要件

この様な開発時に想定し得なかった要件に対しては，ミドルウェアによるコンテンツ変換技術が有効である [36]．例えば，コンテンツ変換処理エンジンをプロキシとして指定することにより，既存 Web コンテンツを編集することなくリクエストに応じてコンテンツを自動変換することが可能となる．しかしながら，コンテンツのみの情報に基づく自動変換には限界があることも指摘されている [29, 35]．

例えば，HTML を携帯端末向けの各種ブラウザ用に変換する際，マークアップ言語のスキーマ [37, 38] 間の対応付けのみで変換を行うと，画面内に収まらない縦に長いページが生成されてしまう等，非常に見づらいものとなることが多い [39]．この様な問題を解決するには，コンテンツをその重要度や意味に応じて並べ替えたり取捨選択する必要があり，その実現にはコンテンツ内のどの部分が重要なのか，どの様な役割なのかといった，コンテンツ内の各部に対する意味付けが必要となる [29]．

Web コンテンツを人手による操作無しで自動的に翻訳を行う機械翻訳の試みは，時事ニュース等，文法や意味が比較的明確な場合において有効であることが示されている [40]．しかしながら，多義語などに代表される意味が曖昧な表現や，機械翻訳のための辞書に含まれない表現が利用されていた場合，文法的に正しくない記述が含まれる場合等，機械翻訳が困難なコンテンツに対しては人手による翻訳作業が必要となる．その結果，Web コンテンツの翻訳は大変手間のかかる作業となり，コンテンツが迅速に公開されなかったり，そもそも翻訳されずに放置されることも多く，既存コンテンツを有効に利用しているとは言い難い．この様な課題に対し，Web コンテンツに文法情報や専門用語・略語の意味などを付加することで翻訳の精度を向上させ再利用を推進する手法が提案されている [35, 41, 42]．

この様に，情報発信の分野における Web コンテンツの再利用においては，再利用目的に合わせて Web コンテンツに意味付けを行うことが有効である．但し，Web コンテンツの再利用の目的は様々であったり，必ずしもコンテンツ管理者自身が意味づけを行うとは限らないことから，コンテンツに直接意味を埋め込むのではなく，分離して管理することが望ましい [29]．しかしながら，現状では意味づけを行う際の共通の枠組みが欠如しているため，再利用目的毎に編集ツールを開発するなどの作業が必要となる．その結果，再利用に必要なコストが増大し，既存資源の再利用の妨げとなっている．

一方，様々な処理を実施するためのインタフェースとして Web を用いる Web アプリケーション (図 1.3) の開発においては，Model 2 アーキテクチャ[43] と呼ばれる MVC (Model-View-Controller) プログラミングパラダイム [44, 45] に基づく開発手法やフレームワークが用いられつつある．MVC では，GUI アプリケーションの構成要素を，アプリケーションの状態と振舞いを現す Model，画面の表示を行う View，ユーザ入力に対するインタフェースの働きをする Controller に分類し，各要素の役割を明確に区分している．開発者は，アプリケーションの処理を Model として設計した後，既に開発されている View と Controller をプラグインすることが出来るなど，アプリケーションの処理と GUI 部分を切り離して設計・開発を行うことが可能となり，開発効率の向上が期待されると同時に各要素を構成するソフトウェア部品の再利用も促進される．

また，Struts[46] や Turbine[47]，Barracuda[48] など，MVC に基づく Web アプリケーションフレームワークでは，Web ブラウザからのリクエスト処理，次ページへの遷移，表示用のデータ送信という Web アプリケーションに共通して利用される基本的な機能が予めフレームワークにより提供されている．この様なフレームワークを用いて開発を行うことにより，開発者はアプリケーションに固有な処理の開発に専念することが可能となり，開発期間の短縮や成果物の品質の向上が期待出来る [49].

さらに，MVC に基づいて Web アプリケーションのモデル化を行い設計情報として用いる手法も提案されている [50-52]．これらの手法においては，MVC で規定される Web アプリケーションの振舞いを実行環境に依存しない表現で定義し，設計段階での不整合検出やフレームワークの利用，並行開発の実現による作業の効率化を目指している [53]．ここで利用されるモデルは実行環境に依存しない表現で記述されているため，実行環境の移行にも円滑に対応可能となることが期待され，リエンジニアリングの観点からも望ましい [54].

一方で，この様な新しい技術を用いて開発されていない既存 Web アプリケーションも多数存在する．これらの既存アプリケーションにおいて，アクセス数の急激な増減など新たな要件の発生により，ソフトウェア基盤の変更を余儀なくされる場合がある．例えば，CGI (Common Gateway Interface)[55] を基盤として開発された Web アプリケーションを，J2EE[43] プラットフォームへ移行し，より大規模な業務アプリケーションとして再開発する場合などである．ソフトウェア基盤の移行により，充実したライブラリ群やフレームワークによる開発効率の向上，パフォーマンスや信頼性の向上等，ソフトウェア基盤が持つ恩恵をそのまま享受することが出来る．しかしながら，アーキテクチャの変更により既存資源の再利用が困難となり，システム全体やその大部分を開発せざるを得なくなることも多い．また，システムの再開発に際しては，アプリケーションの仕様やその振舞い，プラットフォームへの依存性など，アプリケーションの現状を把握する必要がある．しかしながら，仕様書の欠如や記述の不備，プラットフォームに依存する機能と依存しない機能がプログラム中に混在すること等から，アプリケーションの現状把握には困難が伴うことも多い．例えば，既存資源の分析やシステムを抽象化した情報に基づいて現状のシステムを視覚化する等の従来のリバースエンジニアリング手法を用いたとしても，仕様書の代替とはなり得ず，またプラットフォーム依存の処理を切り分けることは難しい等，ソフトウェア基盤の変更は非常に手間のかかる作業となる．

この様に，現状では情報発信，Web アプリケーションの双方において，既存資源の再利用により，Web におけるリエンジニアリングが効率よく行われているとは言いがたい状況である．

1.2 本論文の目的とアプローチ

前節で述べた様に，効率良い Web 開発を実現するためには，フォワードエンジニアリング技術であるコンテンツ管理システムや開発支援ツール，モデル駆動型開発手法などを有効利用するのは勿論のこと，既存資源を有効活用するリエンジニアリングを欠かすことは出来ない．しかしながら，情報発信の分野において再利用を効率よく行うためには，再利用目的に合わせて Web コンテンツに意味づけを行う共通の枠組みが不十分である．さらに，Web アプリケーションにおいては，ソフトウェア基盤の変更や新たな要件に対応するためのシステム変更に際して，資源の有効利用が十分なされているとは言い難い．すなわち，現状では開発当初には想定し得なかった要件の出現に対し，既存資源を有効活用することが出来ないという問題点がある．このような課題を解決するため，本論文では利用状況や環境に合わせて，既存資源を有効活用するための再利用技術を提案する．

まず，主に情報発信における Web コンテンツを対象に，アノテーションと呼ばれる付加情報を付与する枠組みを提案する [56-59]．提案手法を用いることにより，共通の枠組みを用いて様々なアノテーション語彙の編集を支援することが可能となり，Web コンテンツの再利用のためのコストを削減することが出来る．さらに，提案手法を用いた情報携帯端末など様々なクライアントデバイスに対する Web コンテンツを作成するための情報の付与や，ユーザの嗜好に合わせて選択的に情報を集積するための情報の付与の実現により，既存 Web コンテンツを変更することなく多目的への適応を実現することが可能となる [60-64]．

次に，Web アプリケーションを対象に，既存アプリケーションの振舞いを，HTTP 上で授受される情報に着目した動的解析により明らかにし，モデルの抽出を支援する手法を提案する [53, 65-70]．提案手法を用いることにより，Web アプリケーションの振舞いを表すモデルの抽出が可能となり，例えば CGI を基盤として開発された既存アプリケーションを J2EE プラットフォームへ移行するなど，実行環境の移行などを円滑に行うことが可能となる．

以下，1.2.1 節では Web コンテンツ適応のためのアノテーションフレームワークについて，1.2.2 節では Web アプリケーションモデル抽出支援手法について，それぞれ課題と本論文におけるアプローチを述べる．

1.2.1 Web コンテンツ適応のためのアノテーションフレームワーク

Web コンテンツに対する付加情報であるアノテーションは，Web コンテンツに関する情報を人間だけでなく機械にとっても理解可能な情報として提供するもので，情報の検索や利用環境への適応など様々な分野への応用が期待されている [71]．電子化されたコンテンツに限らず文書内の特定の箇所に付与されるアノテーションには形式的なものから非形式的なもの，暗黙的なものから明示的なものまでさまざまな形態がある [72]．構造的な仕様に基づくメタデータは電子化されたコンテンツに対する最も形式的で明示的なアノテーションであり，本論文ではアノテーションをそのようなメタデータと同義として扱う．

アノテーションはその形態からさらに 2 種類に分類することが出来る．1 つはアノテーションを Web コンテンツ内の対象箇所にコメント等を用いて直接埋め込むインラインアノテーションであり，もう 1 つはアノテーションと Web コンテンツを間接的に関連付ける外部アノテーションである．インラインアノテーションは，アノテーションの内容 (annotation contents) とアノテーション対象箇所の対応関係を維持することが容易であるため，コメントや META タグなどを始め HTML ページにアノテーションを付与する場合に広く用いられてきた [73, 74]．しかしながら，インラインアノテーションでは，対象文書を直接変更する必要があるため，アノテーション作成者にそのような変更権限がなければ適用できないという問題がある．

それに対して，アノテーションを外部的なメタデータとして提供することにより，編集権限のないコンテンツに対してもアノテーションを付与することが可能となる．このような外部アノテーションでは，アノテーション内容に加え対象文書におけるアノテーションの対象となる箇所を指し示す XPath[75] のような指示表現 (addressing expression) が必要となる．つまり，外部アノテーションを生成・編集するためには，通常のコンテンツ編集環境に加え，指示表現を生成するための仕組みが必要となる．

アノテーションを生成・編集するための編集環境は，利用するシナリオにより様々な構成が存在する．例えば，Web ブラウザを用いたアノテーション編集環境 [35, 74, 76, 77] は，アノテーション作成者が対象文書に対する編集権限を持たない場合において有効である．一方，WYSIWYG エディタに基づくアノテーションツール [78] は，アノテーション作成者がアノテーションの生成のみならず対象コンテンツの編集も可能である場合において有効である．この様に，様々なアノテーション編集環境が研究・開発されているが，いずれにおいてもある特定のアノテーション語彙や利用シナリオに特化されており拡張性に欠ける [56-58].

さらに，アノテーションの対象となる Web コンテンツは時間とともに更新されることが一般的であるが，そのような変化をアノテーション作成者が予測することは困難である．例えば，Web コンテンツに対するコメントをアノテーションとして付与する場合，元のコンテンツが変更されることによってアノテーション指示対象が存在しなくなることが主要な問題点として指摘されている [79]．このような対象文書の変化に関わらず，アノテーション箇所が継続的かつ適切に指示されるようにすることは，外部アノテーションに基づく Web コンテンツの再利用にとって重要な課題である．この課題に対し，いくつかの実証的研究 [35] も行われているが，実際の Web サイトに対する指示表現の頑健性についてはこれまであまり検討されていない．

本論文では，まず，Web コンテンツに対し外部アノテーションを付与するためのフレームワークを提案する．次に，フレームワークの実装としてアノテーションエディタを提案する．アノテーションエディタは，必要となるアノテーション語彙に対応するための拡張が可能な編集環境であり，様々な指示表現を簡単に生成することも可能である．また，本論文では実際の Web サイトに対しアノテーションを生成し，Web コンテンツの変更に伴う指示表現の頑健性について評価実験を行うと共に，頑健なアノテーションを作成するための指針について検討を行う．最後に，アノテーションエディタの適用例として，多様な携帯情報端末による Web 閲覧を支援するための Web

トランスコーディングと，ポータルサイト構築のための Web クリッピングについて述べる．

1.2.2 Web アプリケーションモデル抽出支援手法

近年，Web アプリケーション開発には，Model 2 アーキテクチャ[43] と呼ばれる MVC (Model-View-Controller) パラダイムに基づく開発手法が用いられつつある．例えば，MVC に基づく Web アプリケーションフレームワーク [46, 47] を用いることで，アプリケーションに固有な処理の開発に専念することが可能となり，開発期間の短縮が期待出来る [49]．さらに，MVC に基づいて Web アプリケーションのモデル化を行い，設計情報として用いるモデル駆動型開発と呼ばれる手法も提案されている [50-52]．これらの手法においては，Web アプリケーションの振舞い，すなわち Web ブラウザからのリクエスト処理，アプリケーションの状態やデータの変更，次ページへの遷移，表示用のデータ送信等の一連の処理の流れを，実行環境に依存しない表現で定義し，設計段階での不整合検出や，フレームワークの利用，並行開発の実現による作業の効率化を目指している [53]．

また，実行環境の移行や新たな機能の追加など，アプリケーションの再構築が必要となった場合においても，MVC に基づいたアプリケーションでは，構成要素が機能毎に分類され，その依存関係や振舞いが明らかにされているため再利用が容易である [49]．しかし，既存の Web アプリケーションの中には MVC に基づいて開発されていない物も多数存在する．これらのアプリケーションの再構築に際し，MVC に基づくモデルを利用した開発へ移行することで，今後の開発効率および再利用性の向上が期待される．

これまで，Web アプリケーションの再構築支援手法としては，業務ロジックやページデザイン等の資源の再利用を支援するための研究がなされてきた [16, 22, 23]．例えば，文献 [22] では，Web アプリケーションの構成の把握を支援するために，リバースエンジニアリング手法を用いた視覚化手法が提案されている．この手法では，Web サーバにおけるアプリケーションの構成要素を，静的ページ (HTML 等)，動的ページ (ASP, JSP 等)，Web オブジェクト (CORBA, EJB 等)，データベースなどに分類し，各要素に対して解析プログラムを用意し解析を行う．次に，それぞれの解析結果を集約することにより構成要素間の依存関係を明らかにしている．

この様に，リバースエンジニアリング手法を用いることで，Web アプリケーション構成要素間の依存関係の把握や，構成要素の再利用を支援することが可能となる [16, 22, 23]．しかしながら，Web アプリケーションにおいては，ユーザインタフェースとなる Web ブラウザはクライアント側に位置し，アプリケーションは Web ブラウザから送信されたリクエストに基づいてサーバ側の構成要素を呼び出すことにより実行される．この様なリクエストを介した参照関係は，構成要素のみの解析からは把握することが難しく，従来のリバースエンジニアリング手法では Web アプリケーション全体の振舞いが明らかにされないという問題点が指摘されている [80]．

この主な原因として，Web アプリケーションは一種のサーバ/クライアント型アプリケーションであること [81] が挙げられる．サーバ/クライアント型アプリケーションでは，サーバ側リソースだけでなくサーバ，クライアント間の通信内容やクライアント側のリソースを含めた動的解析を

行わなければ，アプリケーション全体の振舞いが明らかとならない [82].

そこで本論文では，既存 Web アプリケーションの実行時の振舞いを，HTTP 上で授受される情報に着目した動的解析により明らかにし，MVC に基づくモデルの抽出を支援する手法を提案する．本手法の対象とする Web アプリケーションは，業務ロジックがサーバ側で実行され，クライアントに相当する Web ブラウザ側ではフォームおよびハイパーリンクを用いたリクエスト送信が実行される様な Thin Web Client[83] 型の Web アプリケーションである．

提案手法では，まず，HTTP セッション上で授受される情報の中から実体ページ，すなわち HTML 文書をそれぞれの類似度等に基づいてグループに分類する [66]．次に，各グループ間の呼出し関係および HTTP リクエストとして送信される各種パラメータを解析し，Web アプリケーション全体の処理の流れを表すページフローを導出する．ここで得られたページフローにより，ページ間の遷移関係，サーバ側ロジックの呼出し関係，パラメータを介したインタフェース等，Web アプリケーション全体の振舞いが明らかとなる．提案手法では，ページフローと Web アプリケーションモデルのスキーマを照らし合わせることにより，MVC に基づくモデルを抽出している．

本論文では，提案手法を実装したモデル抽出支援ツールを用いた実験により，実際の Web アプリケーションから，その振舞いを表す Web アプリケーションモデルの抽出が可能であるか否かを検証した．実験では，仕様書などの情報は一切用いず，実際の Web アプリケーションの実行結果のみに基づいてモデル抽出を行った．抽出されたモデルの検証にあたっては，Web アプリケーションの要件定義のうち，本手法で抽出可能でありモデル上での記述が可能である項目を対象に，要件定義の充足可否に基づいて評価を行った．実験の結果，提案手法を用いることにより，より多くの要件定義を満たすモデルの抽出が可能となることが明らかとなった．

1.3 本論文の構成

本論文の構成は以下の通りである．まず，第 2 章で Web コンテンツ適応のためのアノテーションフレームワークについて述べる．次に，第 3 章で Web アプリケーションモデル抽出支援手法について述べる．最後に，第 4 章で本論文をまとめる．

第2章　Webコンテンツ適応のためのアノテーションフレームワーク

本章では，Web コンテンツに対し，アノテーションと呼ばれる付加情報を付与する枠組みを提案する．提案手法を用いることにより，既存 Web コンテンツそのものを変更することなく，情報携帯端末など様々なクライアントデバイスに対する Web コンテンツを作成するための情報の付与や，ユーザの嗜好に合わせて選択的に情報を集積するための情報の付与など，Web コンテンツの多目的への変換を支援することが可能となる．尚，本章では，利用目的に応じて Web コンテンツを変換することを，Web コンテンツ適応と呼ぶ．

まず，2.1 節で Web コンテンツ適応のための外部アノテーションの枠組みについて述べ，次に，2.2 節で外部アノテーションの枠組みに基づき設計されたアノテーション文書の編集環境であるアノテーションエディタについて述べる [56, 57]．2.3 節では，外部アノテーションを用いた Web コンテンツ適応において重要な課題となっている，Web コンテンツの変更に対する指示表現の頑健性評価について実際の Web コンテンツを対象に実験を行い，指示表現の生成・編集と利用における留意点について分析を行う [58, 59, 63]．さらに，2.4 節で実際にアノテーションフレームワークを利用した適用例について述べ，その有効性を示す [60-62, 64]．最後に 2.5 節で本章をまとめる．

2.1　外部アノテーションの枠組み

本節では，Web コンテンツに対するメタデータを外部的に付与する外部アノテーションの枠組み [29, 84] について説明する．外部アノテーションの主な利点は，指示対象となる文書の文書型定義 (スキーマ) に新たな要素や属性を追加することなく，対象文書内の任意のノードを指示した上でメタデータの付与を可能とすることにある．

図 2.1 に外部アノテーションの枠組みを示す．本論文では，アノテーションの対象となる文書として HTML や XML を，アノテーション文書として XML を仮定している．アノテーション文書は複数のアノテーション記述要素からなり，各々のアノテーション記述要素に対してアノテーション対象箇所を指し示す XPath 表現 [75] が与えられる．HTML や XML の指示表現は，行数や文字数などテキスト情報による選択や Cascading Style Sheets(CSS) セレクタ [4]，XPath 表現に類似した独自の言語 [85] などがあるが，指示表現が特定のアプリケーションに依存しないこと，仕様が公開され標準化されていること，XSL Transformation(XSLT)[86] や XML Schema[38] 等他の言語でも広く利用されていること等から，本論文では XPath を用いることとした．XPath[75] は

XML 文書のオブジェクトモデルである Document Object Model(DOM) [87] のノード集合を選択するための言語で，様々なパターンによる豊富な表現力を提供する．但し，XPath 表現は文書内のノードを指示するものであるため，アノテーション文書と対象文書の関連についても指定する必要がある．そのような文書間の関連づけに関しては XLink[88] や HTML における link 要素を用いて記述することが出来る．

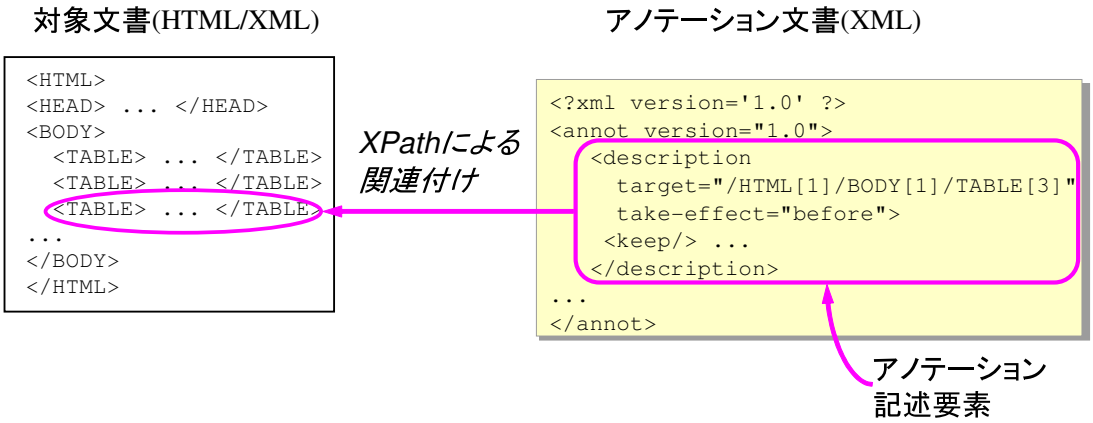


図 2.1: 外部アノテーションの枠組み

アノテーション文書は XML 形式で記述されるが，アノテーションの記述内容は利用目的によって様々であるため共通のスキーマを定義することは現実的ではない．また，同じ内容を異なる文法で記述することが可能である様に，アノテーションにおいては文法よりもその語彙が重要な意味を持つ．従って，文法的側面に依存せずにアノテーション文書を特徴付けることが，外部アノテーションの枠組みを定義する上で重要となる [56, 57].

アノテーション文書の本質的な構造は，アノテーション記述要素の親ノードが文書のルートであり，XPath 表現を値として持つ XPath 属性がアノテーション記述要素に付随するものとして規定出来る．このような情報構造は XML Information Set (Infoset)[89] に基づいて構文的な構造に依らず定義出来る．Infoset は XML 文書を構成する各部分を情報アイテム (information item) の集合 (information set) としてとらえるもので，XML に基づく各仕様に共通のメタモデルを与える．アノテーション文書における指示表現の持ち方を Infoset として定義したものを図 2.2 に示す．図ではアノテーション記述要素 (DescriptionElement) と XPath 属性 (XPathAttribute) の関係が RDF Schema[90] に基づく RDF グラフ [91] として表されている．

本論文では，アノテーション文書のメタモデルは次の特徴を持つことを前提とする．

- アノテーション記述要素 (*DescriptionElement*) は要素 **(element item)** である．
- XPath 属性 (*XPathAttribute*) は属性 (**attribute item**) である．
- XPath 属性は値に XPath 表現を持つ．
- XPath 属性はアノテーション記述要素に格納される (*ownerDescription*).

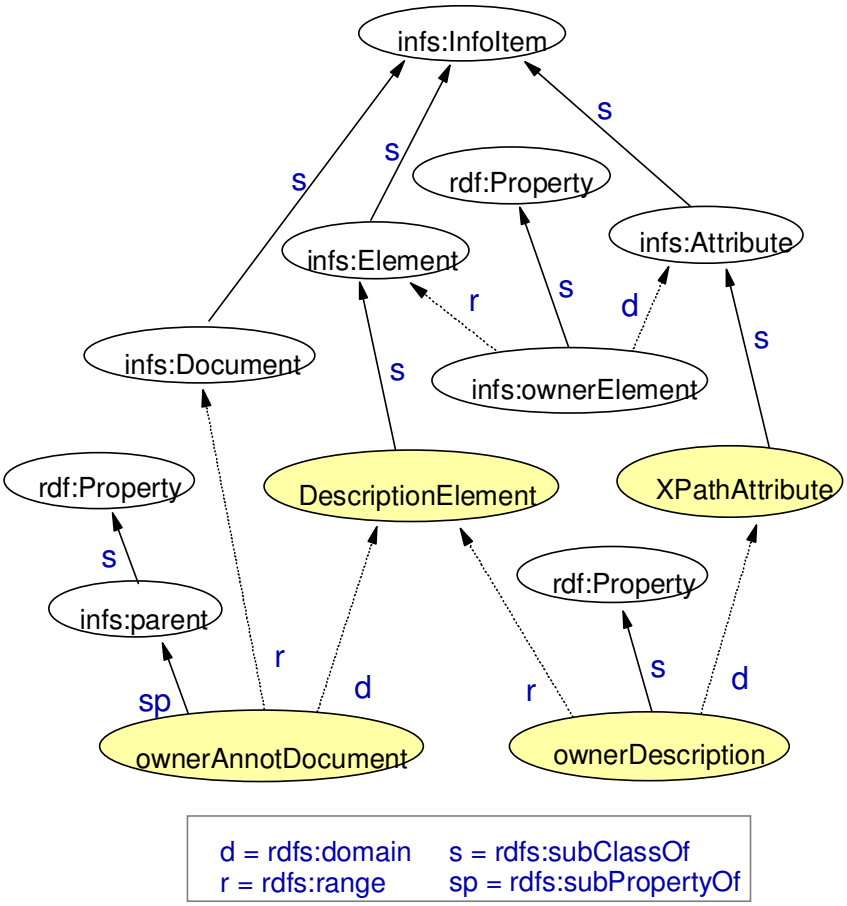


図 2.2: アノテーション文書のメタモデル

- アノテーション記述要素の親 (**parent**) はドキュメント (**document item**) である (*ownerAnnotDocument*).

ここで，ボールド体は Infoset で定義されている情報アイテムであり，イタリック体はアノテーション文書のメタモデル独自の情報アイテムである．下線で示される項目はプロパティ名である．

この Infoset は，既存の外部アノテーション語彙 [29, 91, 92] をはじめ XSLT[86] のスキーマに対しても適合可能なものとなっている．これらのアノテーション語彙は，Infoset とアノテーション文書のスキーマとの対応関係を与えることにより共通の枠組みで扱うことが出来る．すなわち，外部アノテーションの目的が異なる場合においても，対象ノードとアノテーション記述要素の関連付けや，アノテーション記述要素の生成・編集など，各アノテーション語彙に共通する処理をこのメタモデルに基づいたフレームワークにより提供することが可能となる．

次節では，以上の様なアノテーションフレームワークに基づき，統一された環境上で様々な目的に適した語彙を持つアノテーションの作成・編集を可能とするアノテーションエディタについて述べる．

2.2 アノテーションフレームワークに基づくアノテーションエディタ

外部アノテーションの枠組みは，既存 Web コンテンツを変更することなくアノテーションを付与するための構造を規定しており，各アノテーション語彙に共通するアノテーション文書の編集処理などの機能を，アノテーション作成者に提供するための基本概念となっている．ここで，アノテーションを用いた Web コンテンツの再利用を促進させるためには，携帯端末など新しい環境に合わせてコンテンツを作成するよりも，アノテーション文書を作成しコンテンツを変換させる労力のほうが少ないことが望ましい．そのためには，アノテーション文書の作成を効率良く実施するための編集環境が不可欠である [93, 94].

本節では，外部アノテーションの枠組みに基づいたアノテーション文書の編集環境であるアノテーションエディタを提案する．アノテーションエディタは，アノテーション文書の編集に必要なメニューの提示など，アノテーション語彙の選択状態に応じて編集機能が切り替わる様に設計されている．また，GUI の拡張等を行うためのプラグイン機能も用意されている．さらに，外部アノテーションに共通して利用される XPath 表現を簡単に生成可能とする機能も提供されている．このため，アノテーション語彙毎に編集環境を一新する必要がなくなり，Web コンテンツの再利用に必要な開発効率の向上が期待出来る．

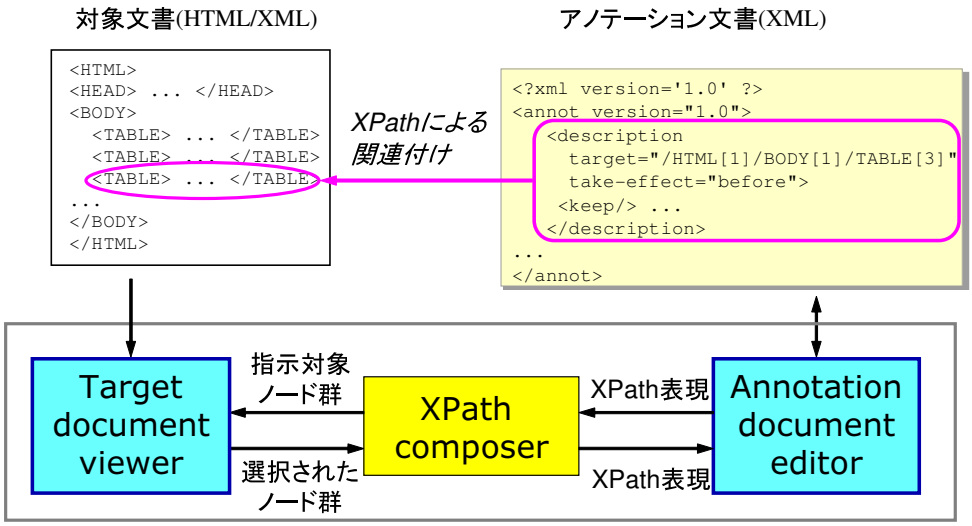


図 2.3: アノテーションエディタの構成概要

図 2.3 にアノテーションエディタの構成概要を示す．アノテーションエディタの構成概要は，前節で述べた外部アノテーションの枠組み (図 2.1) を反映したものとなっている．外部アノテーション作成時には，アノテーション文書作成者はアノテーション記述要素の作成に加え，対象文書の一部を指定するための指示表現である XPath 表現を作成する必要がある．このような前提に基づき，アノテーションエディタは主に Target document viewer，Annotation document editor 及び XPath composer から構成される．

まず，アノテーション文書作成者がアノテーションエディタ上で対象文書を指定すると，Target

document viewer 上に対象文書が表示される．次に，作成者は Annotation document editor を用いてアノテーション文書の作成・編集を行う．各文書は Document Object Model (DOM)[87] としてアノテーションエディタ内部に保持される．ここで，文書間の関連付けは XLink[88] を利用したり，URI[95] による設定ファイル [96] を利用したりすることで定義される．

アノテーション文書作成者は，アノテーションの作成・編集にあたり，まず Target document viewer 上で DOM ノードを選択する．XPath composer は選択されたノード群を指示する XPath 表現を生成する．次に，アノテーション文書作成者は，Annotation document editor を用いて付与するアノテーションの記述を行う．Annotation document editor は，ここで与えられたアノテーションの内容と，XPath composer で生成された XPath 表現を基に，アノテーション文書の記述要素を生成する．ここで，あるノードを指示する XPath 表現は複数パターン存在するが，XPath composer は複数の XPath 表現の中から，作成者が好みの XPath 表現を生成することを支援する GUI を持つ．XPath composer による XPath 生成・編集に関しては 2.2.2 節で述べる．

一方，アノテーション文書作成者は，Annot document editor 上で各アノテーション記述要素を選択することで，アノテーション記述要素が関連付けられた対象文書中の位置を確認することも可能である．この際，アノテーションエディタは，選択された記述要素の XPath 表現を基に XPath composer により指示対象ノードを算出し，Target document viewer 上でハイライトを行う．最後に，アノテーション文書作成者はアノテーション文書を外部ファイルに保存する．

アノテーションエディタの設計は，図 2.2 で示されたアノテーション文書のメタモデルを前提としており，メタモデルに適合するアノテーション語彙を扱うことが出来る．従って，必要に応じてアノテーション語彙を切り替え，編集機能のカスタマイズを行うことも可能である．

2.2.1 アノテーションエディタの実装

本節では，アノテーションエディタの実装と利用方法について，アノテーション文書の生成シナリオに沿って説明する．図 2.4 はアノテーションエディタの構成概要 (図 2.3) を実装に基づき詳細化したものである．

Target document viewer 中の TargetDocument と，Annotation document editor 中の AnnotDocument はそれぞれの文書の DOM[87] である．Annotation document editor 中の AnnotProfile は，アノテーション文書のメタモデル (図 2.2) とスキーマの関連付けを行うプロファイルである．XPath composer は対象文書のノード群とアノテーション文書のアノテーション記述要素の対応づけを行う XPath を生成・編集するための支援環境で，その詳細は 2.2.2 節で説明する．

さらに，図 2.4 の下部にある破線で囲まれた Custom extension は，ビューのカスタマイズや外部アノテーションの埋め込み処理を提供する部分である．これらの機能を用いることにより，対象とするアプリケーションに応じてビューやアノテーション語彙を提供することが可能となる．

アノテーションエディタの GUI の例を図 2.5 に示す．左上のメインウィンドウは 2 つのパネルからなり，それぞれ対象文書を表示する TargetTreeView とアノテーション文書を編集する Annot-

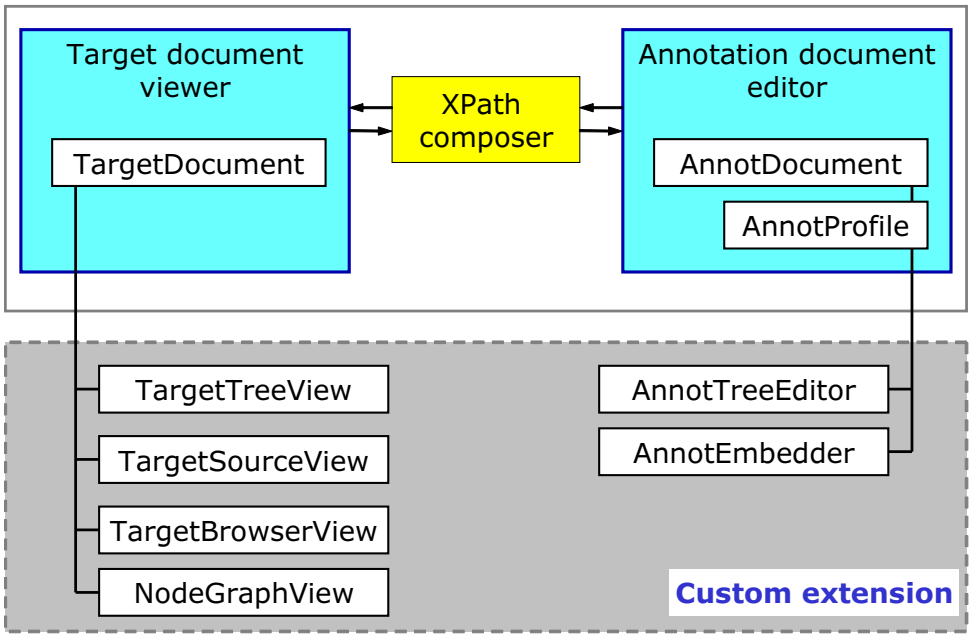


図 2.4: アノテーションエディタの構成

TreeEditor に対応している．AnnotTreeEditor はアノテーション記述を編集するためのエディタとなっており，アノテーション文書のスキーマに基づく編集支援が提供される．AnnotTreeEditor の下部は，アノテーション記述要素の属性を編集するためのプロパティエディタとなっている．

また，2.1 節で述べたメタモデル (図 2.2) とアノテーション文書のスキーマを AnnotProfile を用いて対応づけることにより，様々なアノテーション語彙に対してエディタをカスタマイズすることが可能である．このアノテーションエディタでは，XML ならびに整形式 HTML のいずれに対してもアノテーションを付与することが出来る．特に，対象文書が HTML 文書の場合は TargetBrowserView 上でアノテーション箇所を指定し，表示されるメニューから XPath 表現のパターンを選択することにより，XPath 表現を生成することも可能である．

さらに，左下のウィンドウにある NodeGraphView は，対象文書においてアノテーションが付与されているノードとルートノードに至るまでのすべての祖先ノードをグラフで表示するものである．これにより，文書に多数含まれるノードの中から，特にアノテーション記述要素が付与されているノードに着目して文書構造を見ることなどが可能となる．

また，アノテーションエディタ上におけるノードの選択状態は，各ビューやエディタ間で同期される．図 2.5 では，TargetBrowserView 上で HTML ノードを選択した時，TargetTreeView 及び NodeGraphView においても対応するノードが選択され，各ビュー上でハイライトされている様子が分かる．

以下では，AnnotProfile を用いたアノテーションエディタのカスタマイズ，アノテーション文書の編集，及び AnnotEmbedder を用いたアノテーションドキュメントのシリアライズについて説明する．

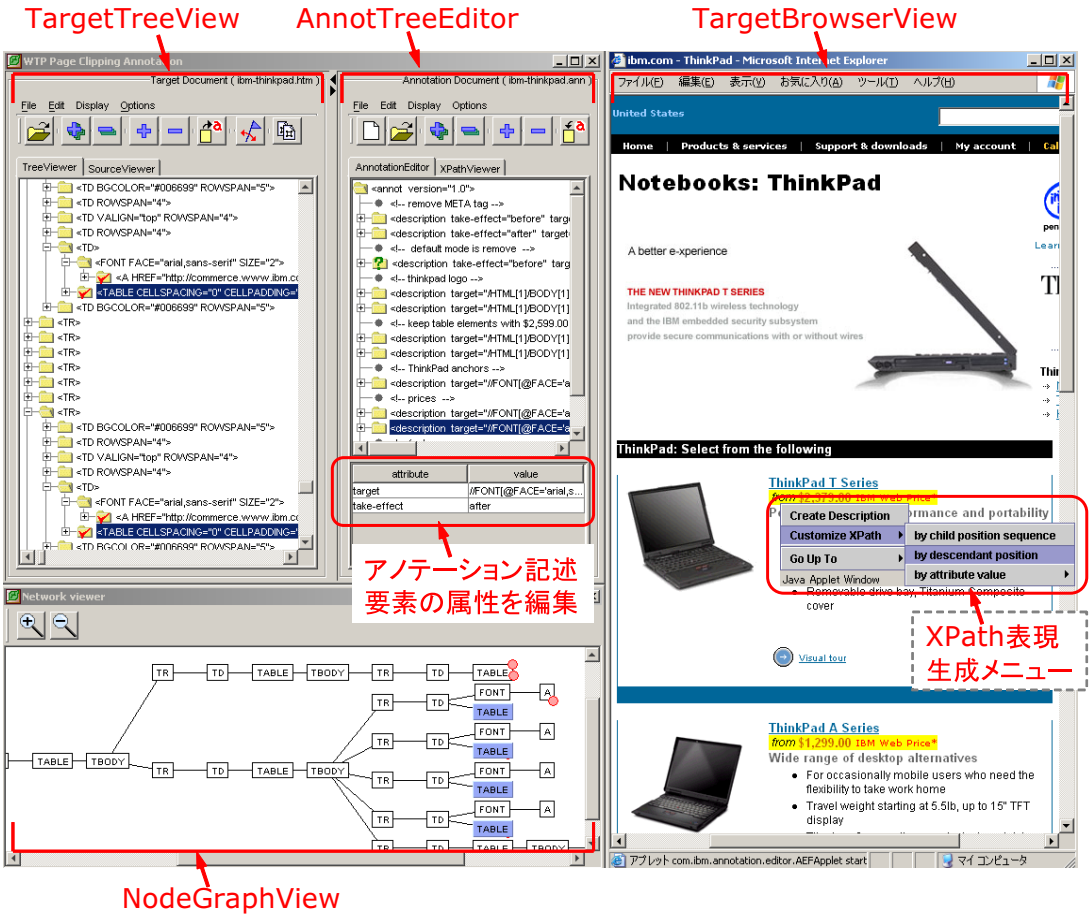


図 2.5: アノテーションエディタの GUI

(a) アノテーションエディタのカスタマイズ

アノテーション文書を作成する前に，AnnotProfile によりアノテーション語彙を特定し，アノテーションエディタをカスタマイズする必要がある．AnnotProfile の設定ダイアログを図 2.6 に示す．一番上のコンボボックスには，予め登録されているプロファイルのリストが表示される．いずれかのプロファイルを選択すると，プロファイル中に指定されているパラメータが表示される．プロファイルには，アノテーション文書のスキーマである Document Type Definition (DTD[37]), ファイル拡張子，作業ディレクトリ，及びアノテーション文書のメタモデル (図 2.2) 中で示される情報アイテム (Document, DescriptionElement, XPathAttribute) が含まれる．図 2.6 の例では，プロファイルとして “EML Sample Annotation” が指定されている．このプロファイルでは，ドキュメントルートに “annot”，アノテーション記述要素に “description”，XPath 属性に “target” がそれぞれ指定されている[1]．この DTD は，アノテーション文書のメタモデルに適合するものとなっている．AnnotProfile を指定することにより，作成されるアノテーションの DTD が指定され，エディタの GUI 上に表示されるアノテーション記述要素作成のためのメニュー項目なども合

[1]“EML Sample Annotation” の DTD は付録に記載

わせて変更される．

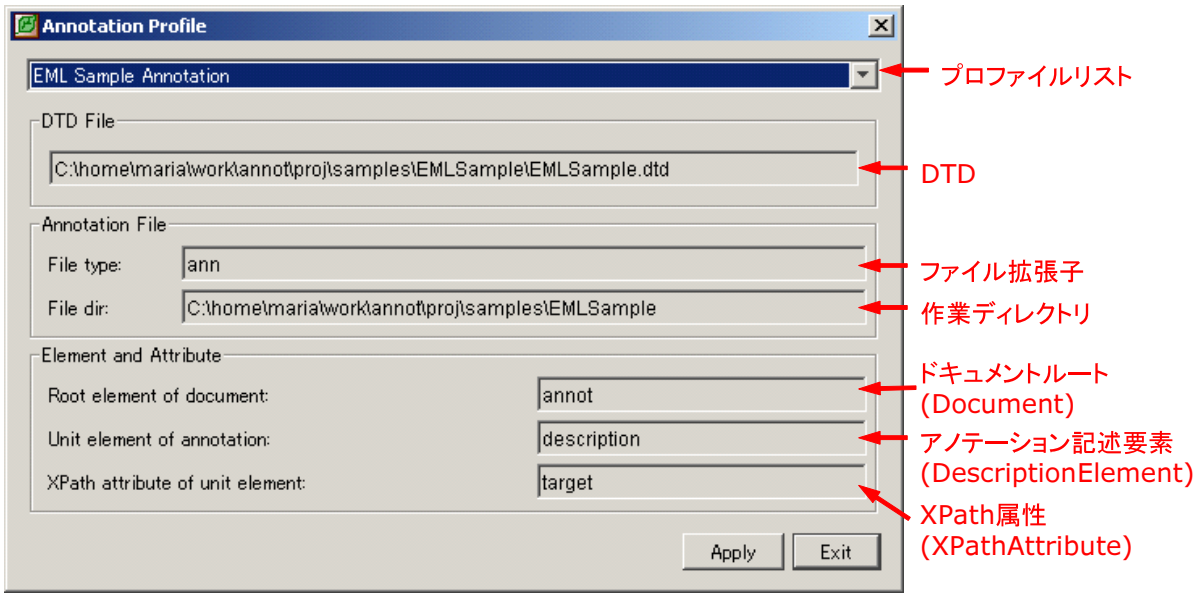


図 2.6: アノテーションプロファイル設定ダイアログ

(b) アノテーション文書の編集

アノテーションエディタではアノテーション文書を新規作成したり，既存アノテーション文書を編集することが出来る．対象ドキュメントのツリービューでは，1 つ以上のアノテーション記述要素により指示されているノードに対し，アイコンにチェックマークが表示される (図 2.7(a))．アノテーション文書のツリービューでは，指示対象ノードが存在しないアノテーション記述要素に対しクエスチョンマークが付く (図 2.7(b))．

さらに，アノテーションエディタ上では，アノテーション記述要素に格納される XPath 表現に

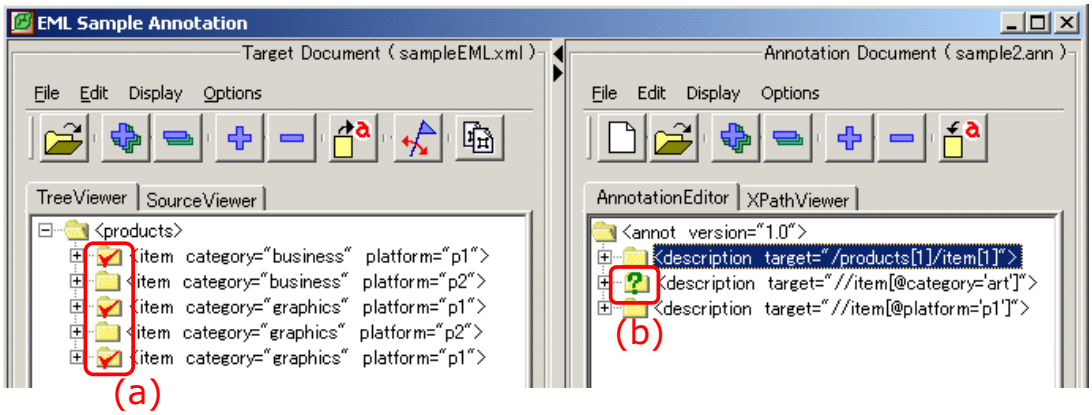


図 2.7: アノテーション文書の編集 (1)

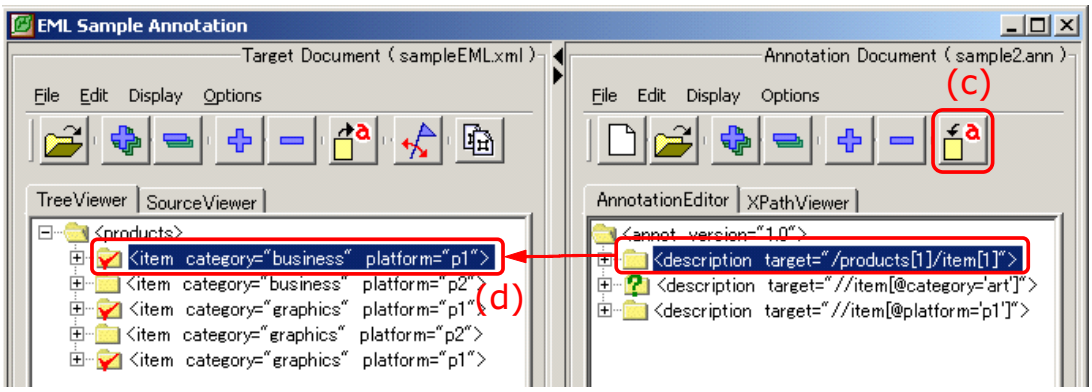


図 2.8: アノテーション文書の編集 (2)

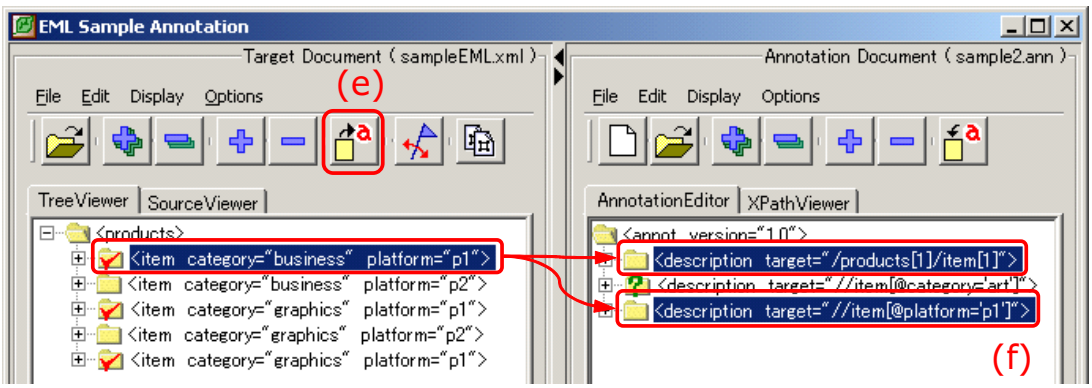


図 2.9: アノテーション文書の編集 (3)

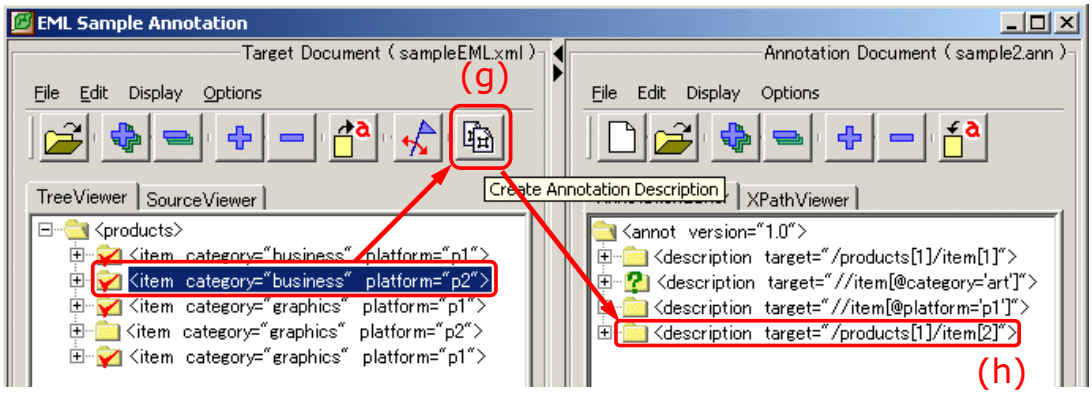


図 2.10: アノテーション文書の編集 (4)

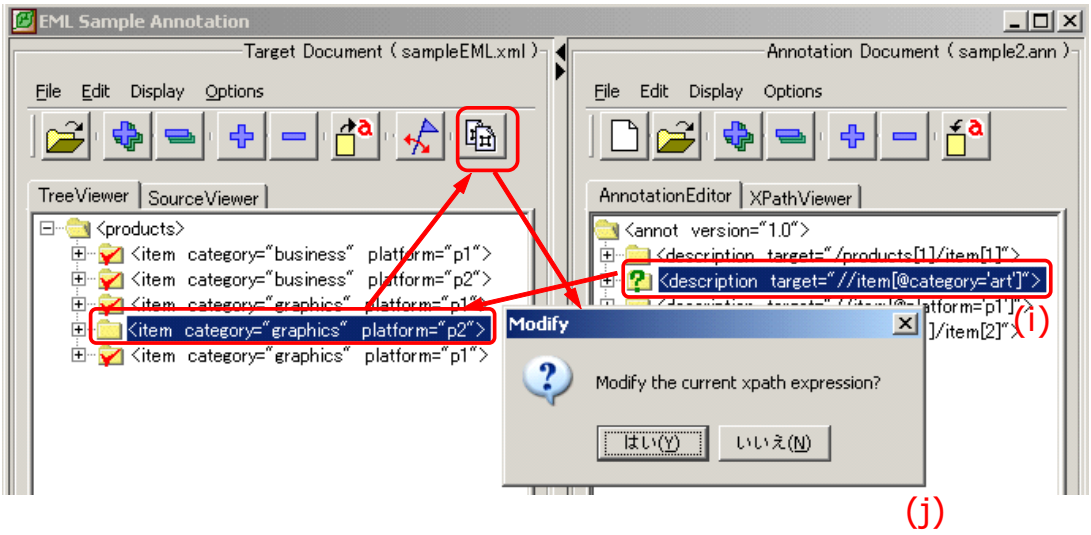


図 2.11: アノテーション文書の編集 (5)

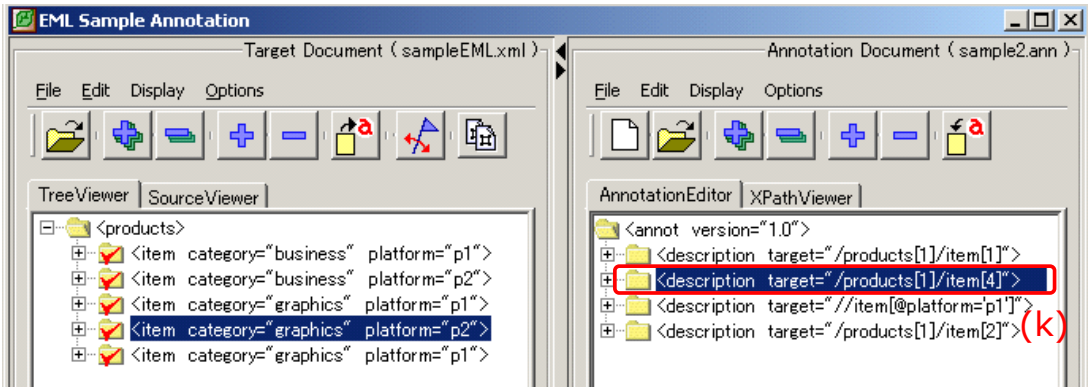


図 2.12: アノテーション文書の編集 (6)

よる参照関係を確認ことも出来る．アノテーション文書を表示するツリービューでアノテーション記述要素を選択しツールバー中のボタン (図 2.8(c)) をクリックすると，対象文書中のノードがハイライトされる (図 2.8(d))．同様に，対象文書を表示するツリービューでチェックマークが付いているノードを選択しツリービュー内のボタン (図 2.9(e)) をクリックすると，アノテーション記述要素がハイライトされる (図 2.9(f))．

アノテーション記述要素を作成するには，まず，アノテーションの対象となるノードを指示するための XPath 表現を生成する必要がある．XPath 表現の生成には XPath composer を用いているが，その詳細については 2.2.2 節で述べる．次に，生成された XPath 表現で指示される対象ノードがハイライトされた状態で，ツールバー中のアノテーション記述要素生成用ボタンをクリックすると (図 2.10(g))，アノテーション記述要素が生成され，XPath 表現が関連付けられると共に新規アノテーション記述要素を示すフォルダ型のアイコンがアノテーション文書のツリービューに表示される (図 2.10(h))．

既に生成されたアノテーション記述要素に格納された XPath 表現を修正するには，アノテーション文書のツリービュー上でアノテーション記述要素を選択し，アノテーション記述要素を生成した手順と同様の操作を用いて編集を行う．例えば，アノテーション文書上のクエスチョンマークで示される 2 番目のアノテーション記述要素を選択した後 (図 2.11(i))，対象文書の中で希望するノードを選択し，アノテーション記述要素を生成するためのボタンをクリックすると，アノテーション記述要素の XPath 表現を修正するか問うダイアログが表示される (図 2.11(j))．OK を選択すると指示対象が存在するためクエスチョンマークが消える (図 2.12(k))．

アノテーションエディタは，2.1 節で述べたアノテーション文書のメタモデルに基づき設計されているため，アノテーションプロファイル (図 2.6) で指定されたアノテーション文書のスキーマがメタモデルに適合している限り，同様のアノテーション文書編集機能を利用することが出来る．

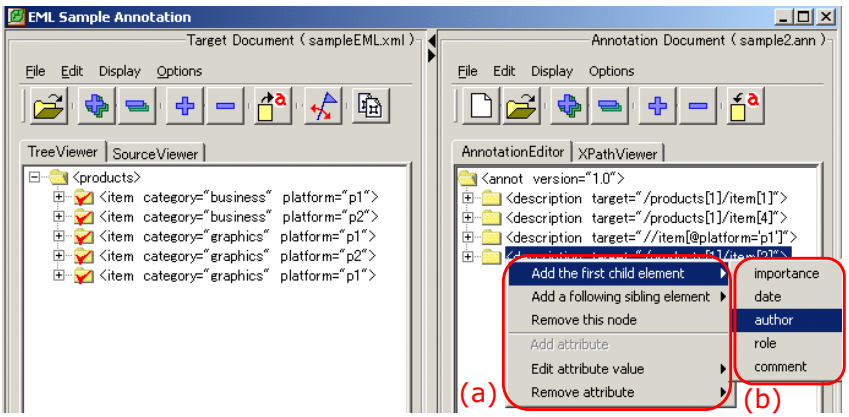


図 2.13: アノテーション記述要素の編集メニュー

例えば, アノテーション文書のアノテーション記述要素を選択し右クリックするとポップアップメニューが表示される (図 2.13)．最初のメニュー (図 2.13(a)) はすべてのアノテーション語彙に共通する内容であり，要素の追加や削除，属性の追加や削除，属性の編集機能等を選択すること

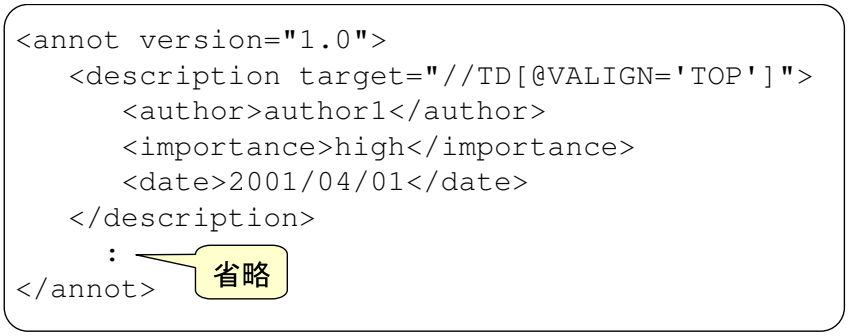


図 2.14: 保存された外部アノテーションの例

が出来る．2 番目のメニュー (図 2.13(b)) は現在設定されているプロファイルに依存する部分である．図 2.13 では，“Add the first child element” を選択した後に表示されるメニューが示されており，“importance” や “date” 等，“description” 要素の子要素となる候補が，DTD(図 A.1) に従い表示されている．“author” を選択した場合は，さらに文字列を入力するためのダイアログが表示される．

この様に，プロファイルを切り替えることで，アノテーションエディタのソースコードを修正することなく，エディタをカスタマイズすることが可能となる．さらに，スキーマに従った GUI を用いることで，DTD の文法等に関する専門知識を必要とせずに，妥当 (valid) なアノテーション文書を生成・編集することが可能となる．

(c) アノテーション文書の出力

アノテーションエディタで作成したアノテーション文書を出力する方法として，外部アノテーションとしての保存とインラインアノテーションとして対象文書へ埋め込むという 2 つの方法が実装されている．例えば，“File” メニューから “Save” を選択するとアノテーション文書は外部アノテーションとして，図 2.14 の様に XML 形式で保存される．“description” はアノテーション記述要素であり，XPath 属性として “target” を持つ．XPath 表現は “target” の値として記述されており，図の例では対象文書となる HTML 文書中で ALIGN 属性の値が “TOP” である TD 要素を指示する．この様にして，対象文書内のノード群とアノテーション記述要素の対応づけがなされている．さらに “description” は子要素として “author”, “importance”, “date” を持つ．これらはアノテーション記述の内容に相当し，それぞれ文字列により内容が記述されている．

一方，アノテーション文書をインラインアノテーションとして対象文書へ埋め込むには “File” メニューから “Embed” を選択する．“Embed” メニューは，AnnotEmbedder インタフェース (図 2.4) の実装クラスが存在している場合にのみ選択可能となる．埋め込み処理が実装されている AnnotEmbedder の実装クラスは，アノテーションプロファイルで指定されている．この実装クラスはアプリケーションに固有であり必要に応じて設定することが可能である．

図 2.15: 対象文書中 (HTML) に埋め込まれたインラインアノテーションの例

例えば，複数のアノテーション作成者が作成したアノテーションを共有・管理するための環境である Annotea[71] では，アノテーションは対象文書に埋め込まれる．また，対象文書中でアノテーションの位置を決めるために XPointer[97] が利用されている．Annotea を対象アプリケーションとして AnnotEmbedder の実装クラスを実装し，埋め込み処理を実行した例を図 2.15 に示す．図 2.15(a) は埋め込み処理前の対象文書を示しており，図 2.15(b) は埋め込み処理後の対象文書を示している．さらに，図 2.15(b) では，埋め込まれたアノテーション文書を下線で示している．対象文書である HTML 文書に id 属性や SPAN タグを用いてアノテーション文書の埋め込み処理を行っている様子がわかる．ここで示された id 属性や子要素を持たない SPAN タグは，ブラウザ上の表示に影響を及ぼさない．従って図 2.15(a), (b) はブラウザ上では同一の表示画面となる．Annotea では XPointer 表現 (`id("author1")/SPAN[2]`) を用いてアノテーション内容が参照される．この表現は，ID 属性値である “author1” で特定される子要素で 2 番目の SPAN タグを指示している．

この様に，本論文で提案したアノテーションエディタは，アノテーションドキュメントを外部アノテーションとしてもインラインアノテーションとしても出力可能である．しかしながら，様々な目的に合わせて Web コンテンツを有効利用するためには，外部アノテーションフレームワークに基づき実施されるほうがデザイン上望ましい．

次節では，外部アノテーションを実現するために必要となる，XPath 表現の生成を支援するための XPath composer について説明を行う．

2.2.2 XPath composer による XPath 生成・編集

XPath[75] は World Wide Web Consortium (W3C) により標準化された指示表現言語であり，XML 文書あるいは整形式 (well-formed)HTML 文書における任意のノード群を指示することが出来る．XPath は XSLT [86] などのトランスコーディング技術や，XML Schema [38] などの標準化

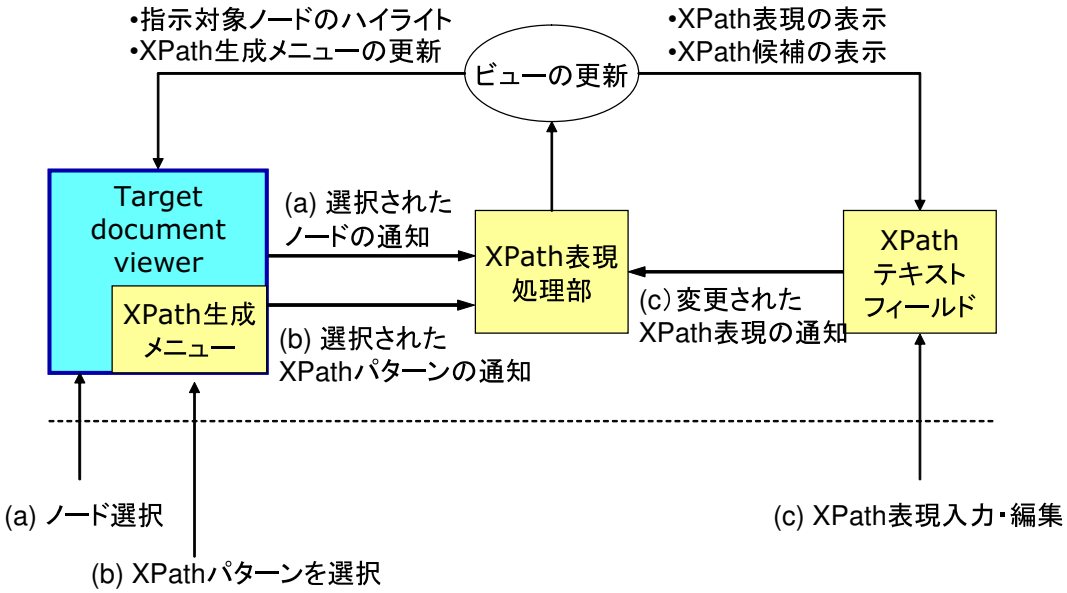


図 2.16: XPath composer の処理の流れ

された言語にも広く用いられている．しかしながら，同一ノードを指示する表現が複数存在するなど，XPath 表現は表現力が豊富なため，様々な表現の中から適切な表現を効率良く生成するには XPath の生成・編集ツールが必要となる．

XPath composer は対象文書に対し XPath を生成・編集するための支援技術であり，アノテーションエディタの一部として組み込まれている (図 2.3)．本節では，ツリービューで表示されている文書を対象に XPath composer について説明するが，ノードを対話的に選択可能であれば対象文書の表示方法は問わない．

XPath 生成・編集のための XPath composer の処理の流れを図 2.16 に示す．XPath composer は XPath 生成・編集のために自動生成機能，及び手動での XPath 編集を補助する機能を提供する．また，図 2.17 に XPath composer の GUI 例を示す．GUI を利用して XPath 表現を生成する場合は，まずツリービュー上でノードを選択する (図 2.16(a), 図 2.17(a))．ノードが選択されると XPath composer は初期設定された XPath パターンに基づき XPath 表現を自動生成し，XPath テキストフィールドに表示する．初期設定と異なる XPath 表現を生成したい場合には，ポップアップメニューなどを用いて XPath の生成パターン候補を提示し選択することで，ある程度自由度のある表現を生成することが出来る (図 2.16(b)，図 2.17(b))．

この様に予め用意された XPath 生成パターンを利用することで，ノードの選択やメニューの選択などの簡単な GUI 操作により XPath 表現の生成が可能であるが，任意の XPath 表現を生成することは難しい．そこで，XPath composer は文字列を入力して XPath 表現を手動生成するための入力フィールドを持つ XPath viewer も提供する (図 2.16(c), 図 2.17(c))．XPath viewer は GUI 操作により生成される XPath 表現を表示するビューアであると同時に，ユーザが手動で XPath を入力する際の補完機能などを備えた入力フィールドとしての機能を併せ持つ．

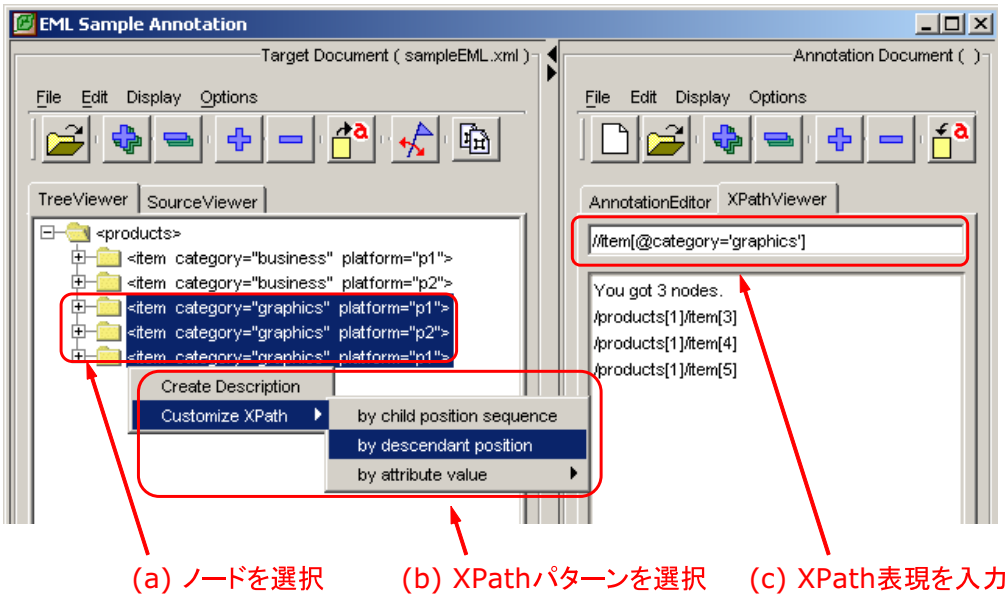


図 2.17: XPath composer の GUI

以下，XPath composer を利用した XPath 表現の生成手法について，対象文書の例を用いて説明する．

(a) 初期設定された XPath パターンに基づく自動生成

XPath composer は，GUI 上でユーザがノードを選択すると，初期設定された XPath パターンに基づいて XPath 表現を生成する．ここで，生成される XPath 表現は，ツリービュー上で選択されたノードと同じノードを指示することが前提となる．

XPath composer は 2 種類の初期パターンを提供している．1 つは chid axis に沿った出現順位による連接で表現されるパターン (*ChildPosSeq*) であり，もう 1 つは，descendant axis に沿った出現順位により表現されるパターン (*DescendantPos*) である．1 種類の axis を用い任意のノードを指示出来る XPath 表現は，基本的に child 及び descendant を用いた場合に限られ，XPath composer はそのいずれかを初期パターンとして設定することが出来る．

図 2.18 で示された XML 文書を対象ドキュメントとして用いた XPath 表現の例を以下に示す．例えば，図 2.18 中，3 番目の `item` ノードの子要素である `name` が選択された結果生成される XPath 表現は以下の通りである．

- `/products[1]/item[3]/name[1]`
  (*ChildPosSeq* (単一ノード指示))
- `/descendant::name[3]`
  (*DescendantPos* (単一ノード指示))

```
<products>
 <item category="business" platform="p1">
   <name>pocket calculator</name><price>349.99</price>
   <!--on sale--></item>
 <item category="business" platform="p2">
   <name>abacus</name><price>99.99</price></item>
 <item category="graphics" platform="p1">
   <name>photo</name><price>100</price><note>New!</note>
   <!--not available--></item>
 <item category="graphics" platform="p2">
   <name>oil painting</name><price>999</price><note>New!</note>
   <!--available--></item>
 <item category="graphics" platform="p1">
   <name>watercolor</name><price>109.99</price></item>
</products>
```

図 2.18: 対象文書の例

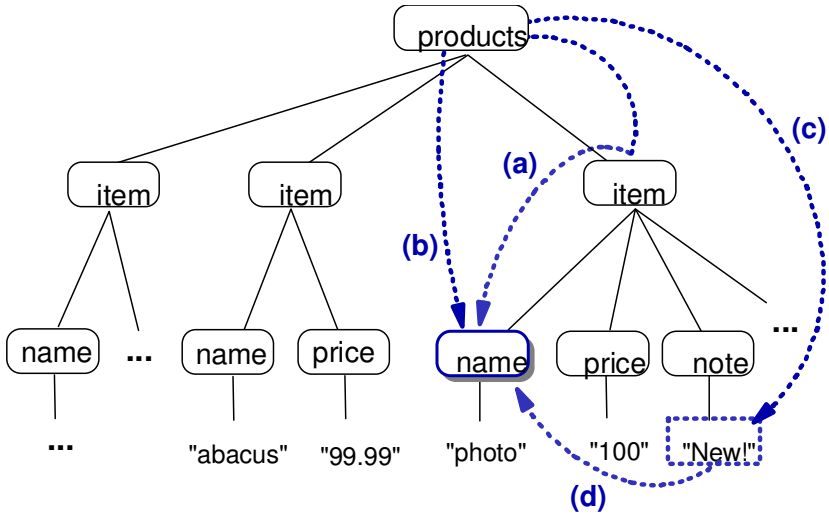
これらの指示表現は高々1 つのノードを指示するもので，該当するノードが存在しない場合はどのノードも指示しない．ChildPosSeq では，あるノードに対する子ノードは子ノードの要素名と兄弟ノード内での順位からなるロケーションステップによって特定され，指示対象ノードはルートノードから子ノードを順次たどっていくロケーションステップの連接として指定される．図 2.19(a) に，ChildPosSeq によるノードの指示の様子を示す．DescendantPos では，ルートノードに対して子孫ノードを前順に走査したときの順位によって対象ノードが指定される．図 2.19(b) の例ではノードを前順に走査した 3 番目の `name` 要素が指定されるが，これは ChildPosSeq によって指定したノードと同一ノードを指示している．

最初から 2 番目までの `item` ノードを選択するなど，利用者が複数ノードを選択した場合は，単一ノード選択の指示表現を演算子 “|” を用いて結合させた以下の表現が生成される．

- `/products[1]/item[1] | /products[1]/item[2]`
  (*ChildPosSeq* (複数ノード指示))
- `/descendant::item[1] | /descendant::item[2]`
  (*DescendantPos* (複数ノード指示))

(b) XPath パターンの選択による XPath 表現の生成

XPath composer では，より柔軟な XPath 表現の生成を支援するため，予め用意された XPath パターンの中からノードの選択状態などエディタの状態に対し有効な XPath 表現をユーザに提示



(a) *ChildPosSeq*	`/products[1]/item[3]/name[1]`
(b) *DescendantPos*	`/descendant::name[3]`
(c) *TextStringMatch*	`//text()[string(.)='New!']`
(d) *RelativeAddressing*	`//text()[string(.)='New!']/preceding::name[1]`

図 2.19: DOM ツリーの模式図及び XPath 表現による指示例

する機能を持つ．これらの表現はポップアップメニューにより選択することが出来る (図 2.16(b), 図 2.17(b))．

例えば，対象文書の例 (図 2.18) において，1 番目の `item` を選択し右ボタンをクリックすると，ポップアップメニューが表示される (図 2.20)．ポップアップメニューでは，“by child position sequence”, “by descendant position”, “by attribute value”, “by relative addressing” というメニューが表示される．最初の 2 つのメニューは，それぞれ (a) で述べた ChildPosSeq と DescendantPos に相当する．また，“by attribute value” は述語による表現，“by relative addressing” は相対指示表現の生成を支援する．以下では述語による表現と相対指示表現について説明を行う．

述語による表現　述語による表現は，`/descendant-or-self::node()/child::`*QName*`[`*Pred*`]` というパターンである．ここで，*QName* はノードテスト (node-test) として与えられた有修飾名 (qualified name) で，*Pred* は，述語表現 (predicate) である．この表現は`//`*QName*`[`*Pred*`]` と略記することも可能であり，XPath composer はこの略記された表現を生成する．XPath composer は主に 2 種類の述語による表現を生成することが出来る．ある属性の値に注目した表現 (*AttrValueMatch*) と，文字列照合による表現 (*TextStringMatch*) である．

例えば，対象文書の例 (図 2.18) において，1 番目の `item` を選択し右ボタンをクリックすると，ポップアップメニューが表示される (図 2.20)．ここで，1 番目の `item` は 2 つの属性を持っていることから，“by attribute value” も表示される．こののメニューを選択すると，さらにサブメニュー

が表示される．次に，サブメニューから “category=business” を選択すると，以下に示す XPath 表現が生成され XPath 表現の候補として XPath テキストフィールドに表示されると共に，その指示するノード群が Target document viewer 上でハイライトされる．

- `//item[@category='business']`

  (*AttrValueMatch*)

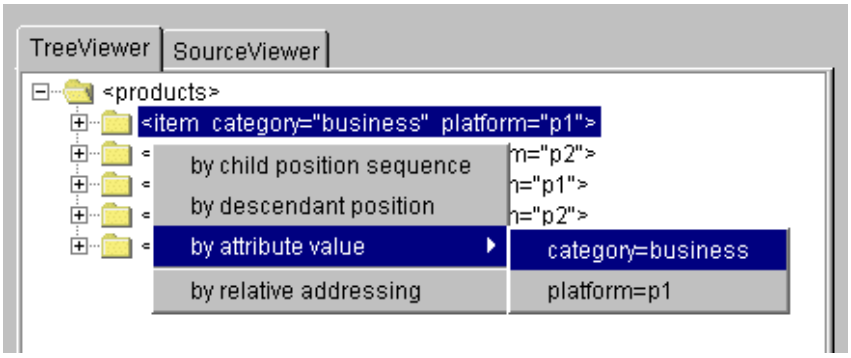


図 2.20: AttrValueMatch の生成メニュー

この XPath 表現は，属性に `category` を持ち，値が `business` である `item` ノードすべてを指示する．従って，対象文書の例 (図 2.18) では，表現を生成する際に選択された 1 番目の `item` に加えて 2 番目の `item` も指示することになる．この様に，XPath パターンの選択による XPath 表現の生成では，生成の基点となったユーザの指定したノード以外のノードも指示される可能性が存在する．従って，ユーザは Target document view 内でハイライトされた指示対象となるノード群を視覚的に確認しながら XPath 表現の選択を行う必要がある．

図 2.21 は，XPath パターン選択直後の XPath composer の様子を示している．図 2.21(a) は，XPath テキストフィールドに表示されている XPath 表現により指示されるノードの個数と，その各ノードを初期設定された XPath パターンを用いて指示した場合の XPath 表現を示している．図 2.21(b) は，XPath テキストフィールドに表示されている XPath 表現により指示されるノードが，ツリービュー上でハイライトされている様子を示している．

XPath composer は，属性値による表現である AttrValueMatch に加え，文字列の一致による表現である TextStringMatch も生成することが出来る．例えば，対象文書の例 (図 2.18) において，3 番目の `item` の子要素の `name` の子要素であるテキストノード “New!” を選択し XPath 表現を生成することも可能である．Target document viewer 上でテキストノードを選択すると，ポップアップメニューに表示されるメニューに “by text string” が含まれる．一方，テキストノードには属性がないため “by attribute value” は表示されない．利用者が “by text string” を選択すると次の XPath 表現が生成される (図 2.19(c))．

- `//text()[string(.)='New!']`

  (*TextStringMatch* (テキストノード))

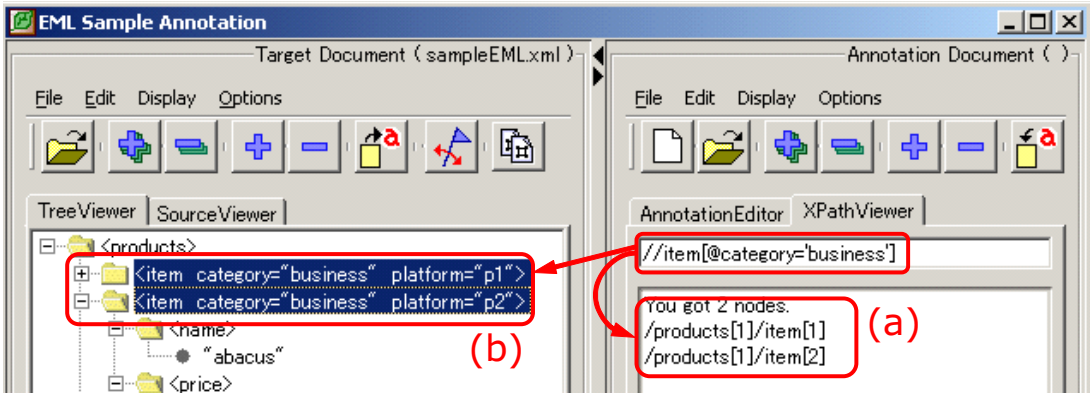


図 2.21: 述語による表現を生成した時のノードの表示

同様に，1 番目の item に含まれるコメントノード<!-- on sale -->を選択した場合，以下の XPath 表現が生成される．

- `//comment()[string(.)='on sale ']`
  (*TextStringMatch* (コメントノード))

また，文字列照合による表現は，対象文字列と等価なテキストノードを指示する XPath 表現に加えて，対象文字列を含むテキストノード `//text()[contains(.,'New!')]`，あるいは対象文字列で始まるテキストノード`//text()[starts-with(., 'New!')]` を指示する XPath 表現の中から選択することが可能である．

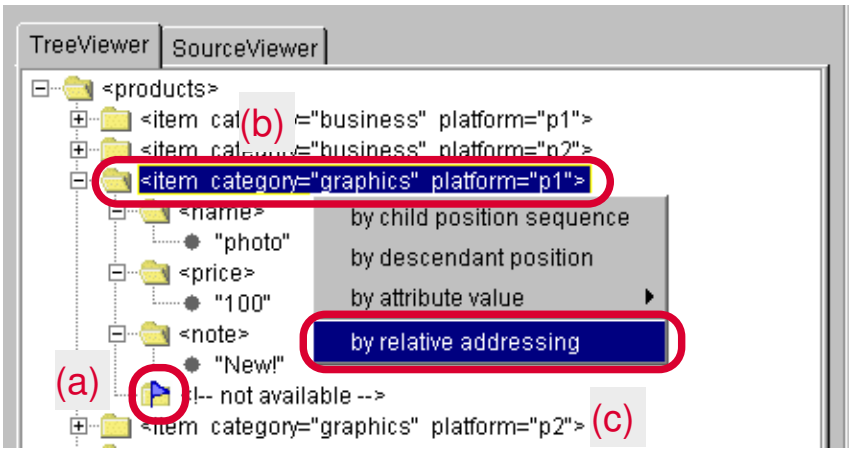


図 2.22: RelativeAddressing の生成メニュー

相対指示表現　相対指示表現 *Relative Addressing* は，あるノードを基点とし，それに対する相対位置を指定する表現である．ChildPosSeq のようにロケーションステップが何段にも連接される場合，その指示表現の一部は対象文書の変更による影響を受け易く，その結果指示対象を失う状

況が考えられる．そのような状況を回避するために，指示対象の近傍に位置し対象文書の変更による影響を受けにくいノードを基点として相対的な指示表現を生成することは有効と考えられる．相対指示表現は基点ノードを示す表現と，相対位置を示すロケーションステップとの連接によって構成され，*AnchorExp*/*AxisName*::*QName*[*PosNum*] というパターンで表すことが出来る．ここで *AnchorExp* は基点ノードを指示する任意の XPath 表現である．また *AxisName*::*QName*[*PosNum*] は相対位置を示すロケーションステップで，*AxisName* は基点ノードから指示対象ノードまでの方向を示す axis 名，*QName* は有修飾名，*PosNum* は *AxisName* に沿って文書をたどった場合の順位である．図 2.19(d) に示した例`//text()[string(.)='New']/preceding::name[1]` では，文字列 `New!`を伴うテキストノード (図 2.19(c)) を基点として，そこから preceding 方向にたどった場合に 1 番目に現れる name ノードを指示する表現である．なお，相対位置を示すための axis として child, descendant, parent, ancestor, preceding-sibling, preceding, following-sibling, following の 8 種類を用いることが出来る．

相対指示表現を生成するためには，まず，通常と同様の操作を用いて *AnchorExp* となる XPath 表現を生成する．相対指示表現も *AnchorExp* になり得る．ユーザが XPath 表現を *AnchorExp* に指定すると，旗の形をしたアイコンが *AnchorExp* が指示するノード上に表示される．例えば，コメントノード`<!-- not available -->`を選択して，TextStringMatch パターン`//comment()[string(.)=' not available ']` を生成し，*AnchorExp* として指定した様子を (図 2.22(a)) に示す．続いて，3 番目の `item` を指示対象ノードとして選択し (図 2.22(b))，ポップアップメニュー “by relative addresssing”(図 2.22(c)) を選択すると，次の XPath 表現が生成される．

- `//comment()[contains(., 'not available')]/parent::item[1]`
  (*RelativeAddressing*)

この表現は，文字列 “not available” を保持するコメントノードの，1 番目の親の `item` を指示する．以上の様に，通常は作成することが難しい相対表現を含む XPath 表現を，GUI を用いて視覚的に確認しながら，簡単な操作で生成することが可能となる．

(c) XPath 表現の入力・編集

XPath composer では，ノード選択やメニュー選択など GUI を用いた簡単な操作により XPath 表現を生成することが可能であるが，一方で生成可能な XPath 表現には制限がある．そこで，手入力支援のための補完機能を用いることにより，より柔軟な XPath 表現を生成することが可能となる．また，GUI 上の操作により生成された XPath 表現を基に入力フィールド (図 2.17(c)) 上でカスタマイズすることも可能である．

例えば，ノード選択や XPath パターンの選択により XPath 表現`/products[1]/item[2]` を生成すると，入力フィールドに生成された XPath 表現が表示される．この表現は 2 番目の `item` のみを指示するが，例えば述語を`/products[1]/item[position() mod 2 = 1]` の様に編集することにより，奇数番目に位置する `item` を指示する表現へカスタマイズすることが可能となる．

さらに，XPath composer には入力支援のための補完機能が用意されている．図 2.23 に補完機能を伴う入力フィールドを示す．フィールド上でタブキーを押下すると，カーソル以降に出現可能である XPath 表現の候補が表示される．XPath 表現の候補は対象文書と入力フィールドにある XPath 表現から計算されるため，指示対象ノードが必ず存在する XPath 表現となっている．これらの候補を選択することで手入力の手間を軽減し，文法的に誤りのない XPath 表現を生成することが可能となる．図 2.23 中，上部より 3 個の表現は，句切り子 “>” の左側に数値が示されている．これは，対象文書中 XPath 表現に適合するノードの数を示しており，単一ノードのみを指示する場合は表示されない．これらの情報から，ユーザは編集中の XPath 表現が指示するノードの数を予め知ることが可能となる．

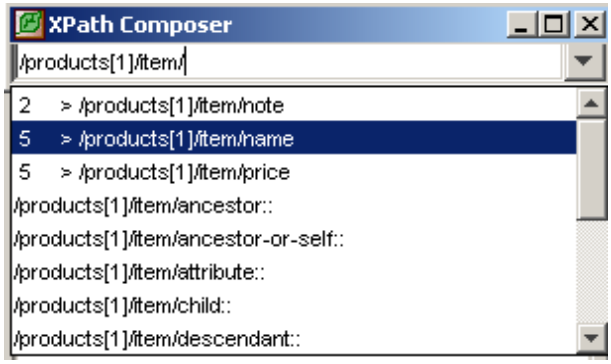


図 2.23: 手入力支援のための補完機能

この様に，XPath composer を用いることで，GUI 上の操作により典型的な表現を簡単に生成することが可能となる．さらに，より柔軟な XPath 表現の生成を支援するために，XPath composer は XPath 編集作業時の補完機能も提供する．以上の様な自動生成と手動生成が継ぎ目なく統合された XPath composer により，柔軟な XPath 生成・編集環境が実現される．

但し，対象文書の変更に対し XPath 表現が同じ対象を適切に指示し続けることは非常に困難であり，その都度調整する必要がある．従来の研究では，対象文書に変更が発生しても同じ対象ノードを指示する表現，すなわち頑健な指示表現に関するいくつかの検証も行われている [79, 85]．しかしながら，実際の Web コンテンツに対する指示表現の頑健性に関してはほとんど検討がなされていない．そこで 2.3 節では，これらの課題に対して実際の Web コンテンツを対象に実験を行い，XPath 表現の生成・編集と利用における留意点について分析を行った．

2.3 Web コンテンツ変更に対する指示表現の頑健性評価

本論文で提案したアノテーションエディタは，アノテーション文書を外部アノテーションとしてもインラインアノテーションとしても出力可能であるが，アノテーション文書の柔軟な編集を実現するためには，外部アノテーションフレームワークに基づき実施されるほうが直接 Web コンテンツを編集しないという点において望ましい [29]．しかしながら，アノテーションの対象となる Web コンテンツは時間とともに更新されることが一般的であり，そのような変化をアノテーション作成者が予測することは困難である．例えば，Web コンテンツに対するコメントや意見などをアノテーションとして付与する場合，元のコンテンツが変更されることによってアノテーションの指示対象が存在しなくなることが主要な問題点として指摘されている [14, 79]．このような対象文書の変化に関わらず，アノテーション箇所が継続的かつ適切に指示されるようにすることは，外部アノテーションに基づく Web コンテンツの有効利用にとって重要な課題である．

従来の研究では，Web コンテンツに変更が発生した場合にも同じ対象を指示するように工夫された表現に関するいくつかの検証も行われている [79, 85]．しかしながら，実際の Web コンテンツに対する指示表現の頑健性に関してはほとんど検討がなされていない．そこで本節では，これらの課題に対して実際の Web コンテンツを対象に実験を行い，XPath 表現の生成・編集と利用における留意点について分析を行った．尚，本論文では XPath の頑健性を，対象文書の変化に関わらず XPath 表現がアノテーション箇所をどの程度継続的かつ適切に指示出来るかを示すものと定義する．

以下，2.3.1 節では XPath 表現の頑健性を評価するための実験方法を，2.3.2 節では実験結果及び考察を述べる．

2.3.1 実験方法

本実験では，XPath による指示表現の中で，自動生成可能であり，かつ一意にノードを指示することが可能な表現を対象とし，頑健性の評価を実施した (表 2.1)．これらの XPath 表現はアノテーションエディタにより生成することが出来る．

表 2.1: 評価実験に用いられた XPath 表現

類型	名称	概説
単一ノード指示表現	ChildPosSeq	child axis に沿った出現順位による表現の連接 (例：`/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[2]`)
	DescendantPos	descendant axis に沿った出現順位を用いた指示 (例：`/descendant::table[8]`)

これらの XPath 表現それぞれを，ある基準日の HTML ページ内の各ノードを指示する表現として生成する．そして，それらの指示表現が基準日以降の同一ページにおいて基準日と同じノー

ドを指示しているかどうかを 540 日間にわたって検証した．指示表現の生成にあたっては表 2.2 に示した 4 種類の HTML ページを用いた．ここで，A, B は同一企業のトップページならびに製品リストのページ，C はニュースページ，D はソフトウェア会社のトップページである．

表 2.2: 評価実験に用いた HTML ページ

ページ	URL	ページ内のノード数 [ave. (max, min)]	文書木の深さ [ave. (max, min)]
A	http://www.ibm.com/	392.0 (441, 348)	21.9 (21, 20)
B	http://www.ibm.com/products/	708.7 (750, 623)	27.3 (30, 20)
C	http://public.wsj.com/	952.4 (1333, 433)	22.0 (24, 21)
D	http://java.sun.com/	908.0 (1311, 325)	21.6 (29, 13)

2001 年 6 月 12 日から 2003 年 1 月 4 日まで実施．各ページのサンプル数は 540 である．

本実験では，基準日に対して生成された XPath 表現が，後日のページにおいて同一ノードを指示しているかどうかを確認する．そのため基準日の HTML ドキュメントにおける全ノードと，後日ページ群の各文書のノードとの対応関係が必要となる．この作業は基準日ページの各ノードにはユニークな ID を属性として付与し，文書間の差分を取ることによって行った．

図 2.24 に基準日ページのノードと後日ページのノードとの対応関係を導出するための ID 付与の処理を示す．図 2.24(a) は基準日ページに対する処理，図 2.24(b) は後日ページ (N 日分) に対する処理を表している．

まず，基準日ページに ID を付与する (図 2.24(a))．ID 付与は各基準日ページに対し 1 度行う．ID として用いる属性名は HTML 文書中の他の属性と重複しないよう，例えば “uid” などで与える．ID 付与の結果を ID 付き基準日ページとして出力する．

次に，基準日ページと後日ページの 2 ページ間で差分をとり，結果を差分ファイルとして出力する (図 2.24(b))．差分計算には 3DM merging and differencing tool for XML (3DM)[98, 99] を用いた．さらに，ID 付き基準日ページに対し，差分ファイルを適用しパッチ処理を行う．これにより，後日ページにおいて基準日ページのノードに対応するノードが存在すれば ID が付与される．そして，その結果を ID 付き後日ページとして出力する．後日ページが基準日ページと全く同一の場合，後日ページのすべてのノードに対し ID が付与される．この ID によりノードの対応関係が導出される．

基準ページにおける ID を持つノードは，差分を計算する時の基準ノードとなる．まず，ID 付き基準日ページの各ノードについて ChildPosSeq と DescendantPos の 2 種類の XPath 表現を生成する (図 2.25(a))．この XPath 表現を ID 付き後日ページへ適用する (図 2.25(b))．ID 付き後日ページにおいて XPath 表現により指示されたノードの ID が，ID 付き基準ページの ID と等しくかつ他のノードを指示しない場合に限り正答とする．XPath 表現は文書中複数のノードを指示する可能性があるが，本実験では単一ノードを指示する XPath 表現を対象とした．従って正答となる XPath 表現は，基準ページと同一 ID である単一のノードを指示する表現となっている．

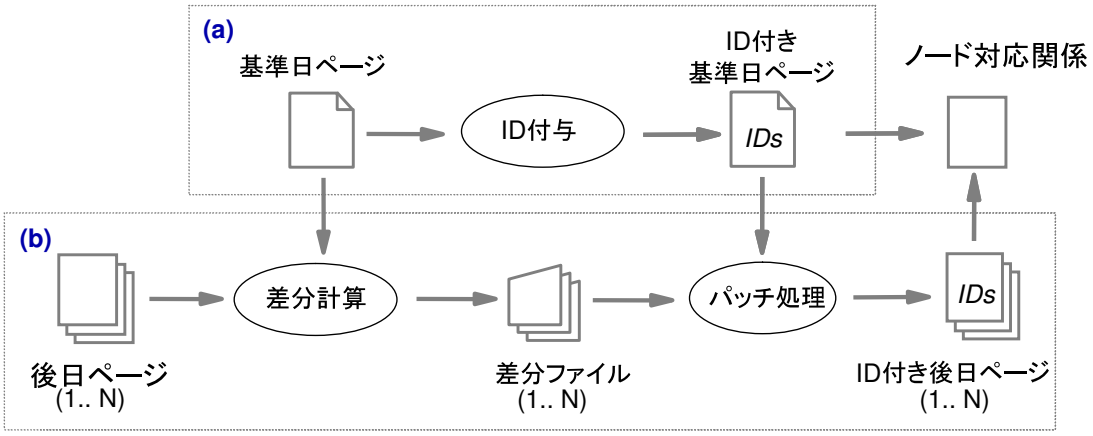


図 2.24: 基準日ページと後日ページのノード対応関係導出のための ID 付与の処理

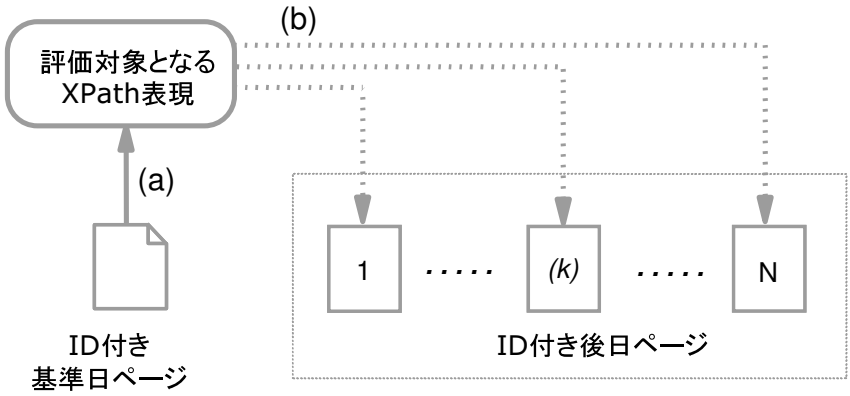


図 2.25: 評価対象となる XPath 表現の生成及び ID 付き後日ページへの適用

540 日に及ぶ実験期間における ID を持つノード数の推移を図 2.26 に示す．実験期間中，Web コンテンツの変化に伴いノードが削除・挿入・更新された．新規挿入されるが，ノードは基準日において対応するノードが存在しないため ID を持たない．本実験で用いた差分ツール [98, 99] は多くのツリー差分エンジンと同様，属性値の変更をノードの新規挿入及び対象ノードの削除とみなす．これは，ある属性値が変更されノードが更新された結果，ID が無くなることを意味する．

図 2.26 において，ページ A の 59 日目，ページ B の 62 日目，ページ C の 229 日目，ページ D の 365 日目それぞれにおいて ID を持つノード数が急激に減少していることが分かる．これはページのスタイルの変更によるものである．一方，ページ D の 271 日目から 10 日間一時的に ID を持つノード数が減少しているのは，この時期にページ所有者による一時的なイベントが開催され，特別なスタイルを用いたページに変更されたからである．ID を持つノード数は，ノードの削除・挿入・更新により減少する．差分アルゴリズムは常に基準ページと後日ページに対し適用されるため，ID を持つノード数は必ずしも単調減少しない．もし基準ページと類似しないページから類似するページに変更が行われた場合は ID を持つノード数は増加する．

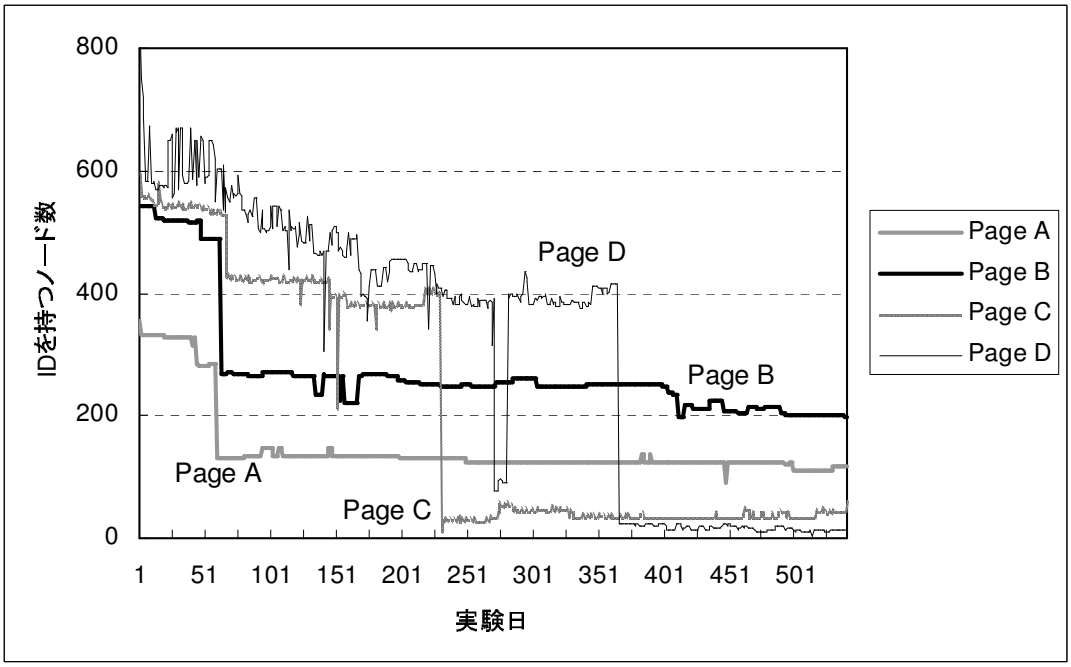


図 2.26: 実験期間における ID を持つノード数の推移

2.3.2 実験結果

図 2.27 に実験期間中の XPath 表現毎の正答率を示す．図中，ChildPosSeq は DescendantPos より高い正答率を示している．特にページ C 及びページ D において，ChildPosSeq が 70%以上を示しているのに対し，DescendantPos が 50%未満である．DescendantPos はタグ名と出現順位のみで構成されているため，指示対象ノードと同じタグ名の挿入や削除に対して影響を受けやすい．従って，DescendantPos は文書に含まれるノード数が多い程，基準日と同じノードを指示することが難しい．実際，ページ C 及び D は 900 以上のノードが含まれており，ページ A 及びページ B より多い (表 2.2)．このため，ページ C 及びページ D において DescendantPos は ChildPosSeq より正答率が低かったと考えられる．

さらに，図 2.27 のページ A，B において ChildPosSeq 及び DescendantPos の正答率は 50%を下回っている．これは，ページ A の 59 日目，ページ B の 62 日目それぞれにおいて ID を持つノードの数が急激に減少する (図 2.26) など，実験初期段階にページスタイルが変更されたことに起因すると思われる．従って，スタイル変更の影響を出来る限り排除するため，ID を持つノード数の変動幅に閾値を与え，ID を持つノード数が多い通常の変更時と，ID を持つノード数が少ないスタイルの変更時とを分離して正答率を算出した (図 2.28)．

ID を持つノード数が 70%以上の実験期間における正答率を図 2.28 に示す．この様に，ID を持つノード数に閾値を与えた結果，ページ A では初日から 58 日目，ページ B では 61 日目，ページ C では 121 日目，ページ D では 3 日目迄が対象期間となった．図 2.28 から，ChildPosSeq はページ A，B，C において 99.9%, 93.9%, 96.4%といずれも高い正答率を示している．一方で，ページ

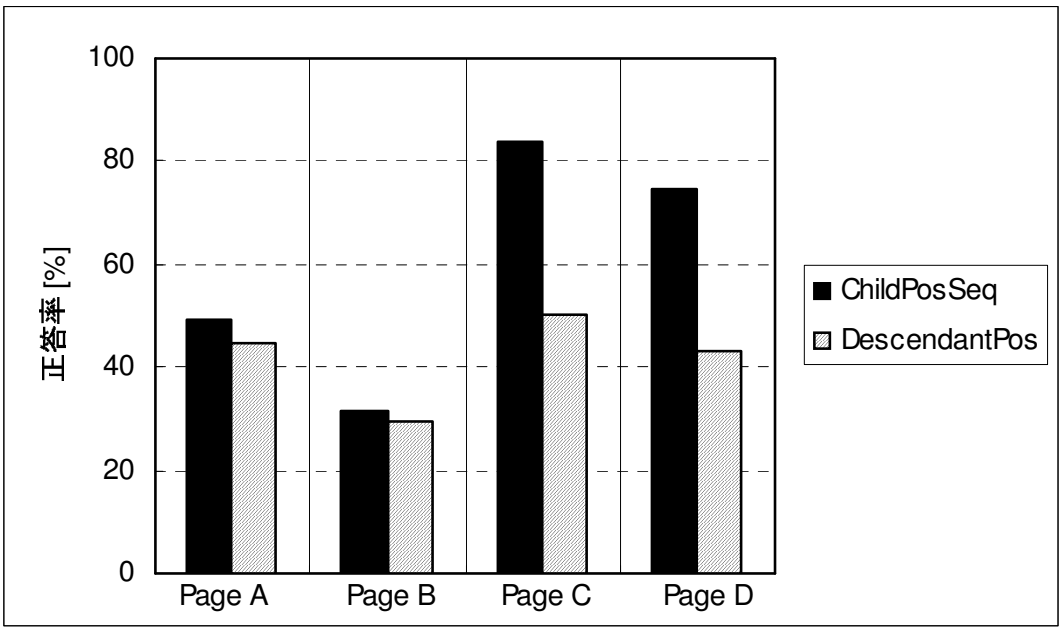


図 2.27: 実験期間中ページ毎の XPath 表現の正答率

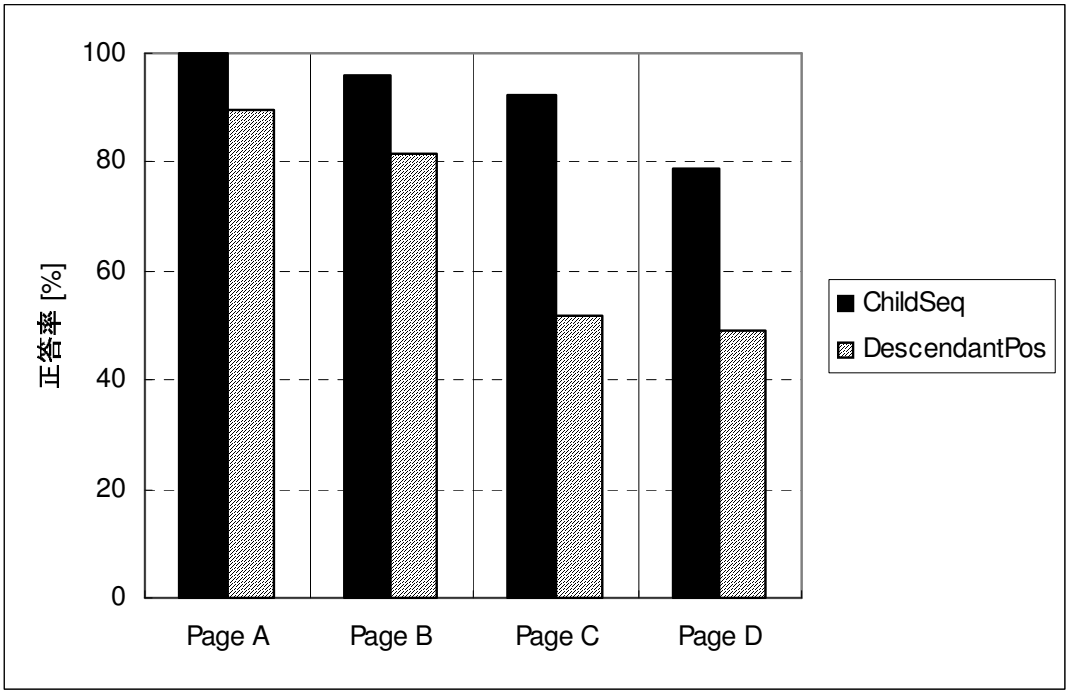


図 2.28: ID を持つノード数の割合が 70%以上の実験期間における正答率

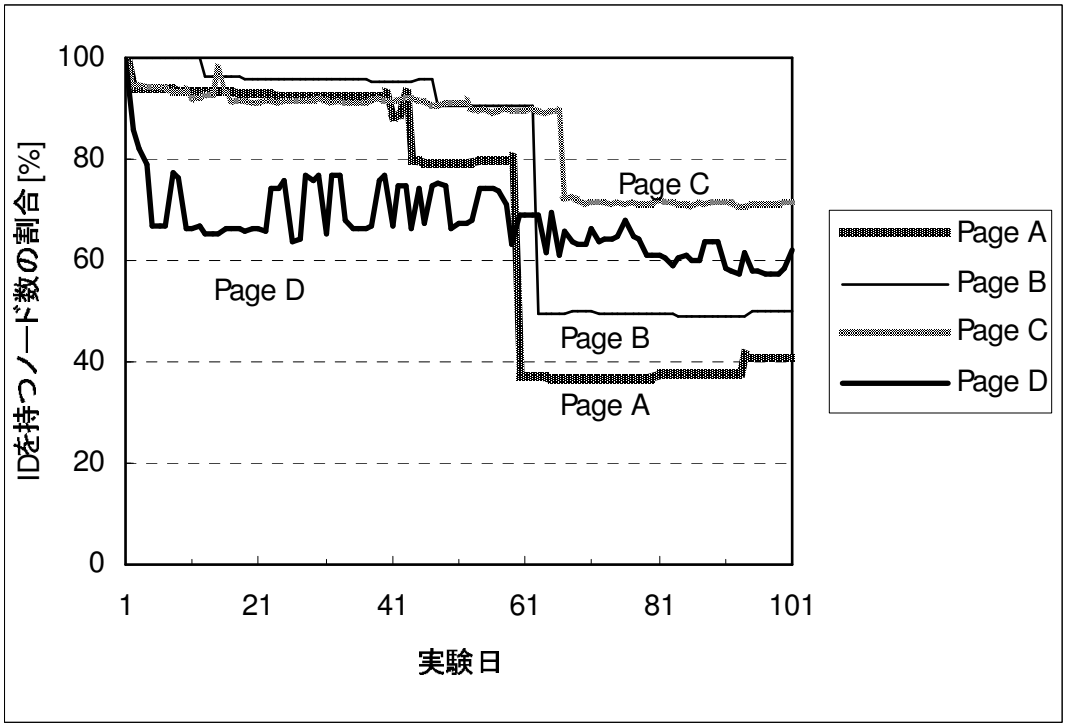


図 2.29: 実験開始 100 日間における ID を持つノード数の割合の推移

D においては，3 日間のみで 79%と比較的低い正答率を示している．この原因に関しては，実験開始時点で付与された ID の変化の様子を検討すると明らかになる．

実験開始後 100 日間における ID を持つノード数の割合の変化を図 2.29 に示す．ページ D では，実験開始直後から ID を持つノード数に変化が見られ，開始直後に約 70%前後で絶えず変動している．従って，もし基準ページを後日ページのいずれかと置き換えれば，基準日ページと後日ページの差は減少し，ID を持つノード数の割合は増加すると考えられる．さらに，ID を持つノード数の増加に従い，ページ D における ChildPosSeq の正答率は改善されると考えられる．

ID を持つノードの割合は基準日ページと後日ページの類似度としても扱うことが出来る．何故なら，ID は基準ページから後日ページへ変換するための操作から構成される差分ファイルに基づき挿入されるからである (図 2.24)．ページ間の差分から類似度を算出し，ある一定の閾値を超えたときに XPath を作り直すことが出来れば，ChildPosSeq の頑健性を改善することが可能になると考えられる．近年，対象文書の変更に伴い動的に指示表現を変更するための，適応アルゴリズムに関する研究がなされている [85]．しかしながら，適応アルゴリズムを含むこれらの研究は独自の指示表現に関してであり，標準である XPath 表現に関する研究はなされていない．本実験による結果は，XPath 表現の動的適用を実現するための，実際のデータを元にした有益な知見を与えるものである．

本実験では，正しいノードを指示する XPath 表現を，基準ページと同じ ID を持つノードを指示する表現として扱った．本論文ではこの状態を exact とする．XPath 表現が正しいノードを指示し

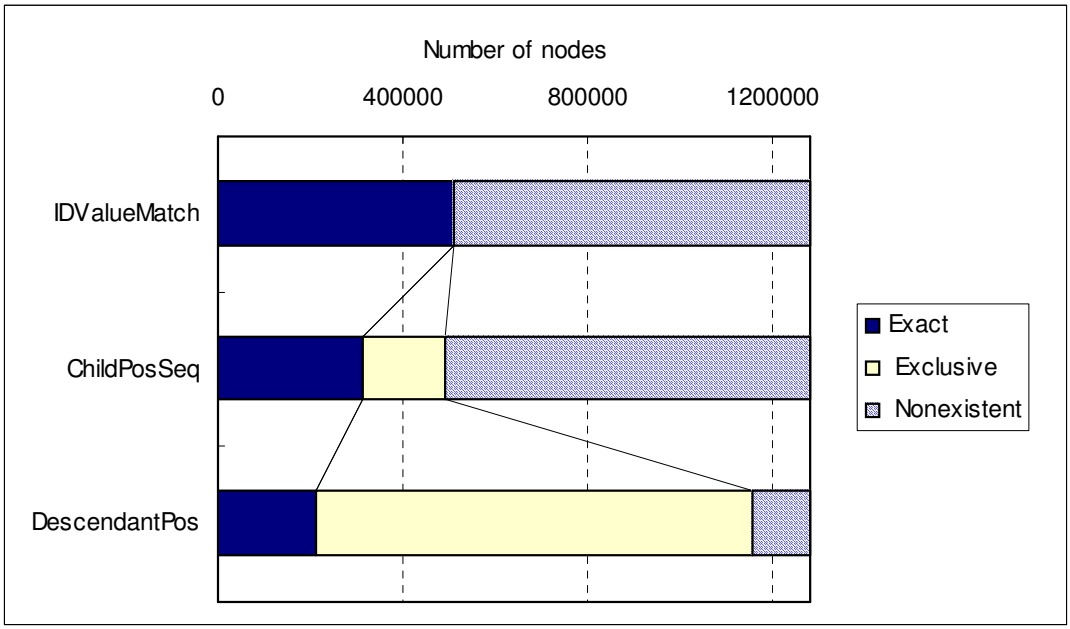


図 2.30: 詳細な正誤区分に基づく実験対象ノードの分類結果

ない場合，その性質から 2 種類の状態に分類することが出来る．それらは，nonexistent, exclusive である．nonexistent とは，XPath 表現による指示対象ノードが存在しない場合である．exclusive は XPath 表現による指示対象ノードは存在するが，正しいノードを指示していない場合である．

図 2.30 に，この詳細な正誤区分に基づく実験対象ノードの分類結果を示す．540 日間に及ぶ実験期間中，4 種類のページにおける実験対象ノードすべてを積算すると，およそ 100 万ノード (1,280,880) である．図 2.30 では 2 種類の XPath 表現 (ChildPosSeq,DescendantPos) に加え，ID を用いた XPath 表現 (IDValueMatch) による実験対象ノードの分類も示されている．IDValueMatch は，正答率を算出するために用いた ID 属性付与処理 (図 2.24) で与えられる ID による指示表現である．例えば，`//*[@uid='N35']` という表現では，“uid” が ID 属性付与処理 (図 2.24) における ID 属性名に相当する[2]．

図 2.30 より，DescendantPos において exclusive による誤答が多いことが明らかである．exclusive は，XPath 表現による指示対象はあるものの，正しいノードを指示していないことを意味する．すなわち，もし DescendantPos が利用された場合，指示状態のみでは exact か exclusive か判断出来ない．従って，DescendantPos を用いたアノテーションは，アノテーション生成時に指示していたノードと異なるノードに適用される可能性がある．例えば XPath を指示表現として採用している XSLT[86] を用いて変換を行う場合，変換対象文書が更新された結果，もともと指示対象でなかったノードに対して変換処理が行われる可能性がある．DescendantPos は XPath の表現としては簡潔であるため表現内容は理解し易いが，対象文書の変更に伴い予想外のノードを指示する恐れがあるという意味で慎重に利用すべきであると言える．

[2] “uid” という属性名は，実験対象ページにおいて重複しないことを確認している

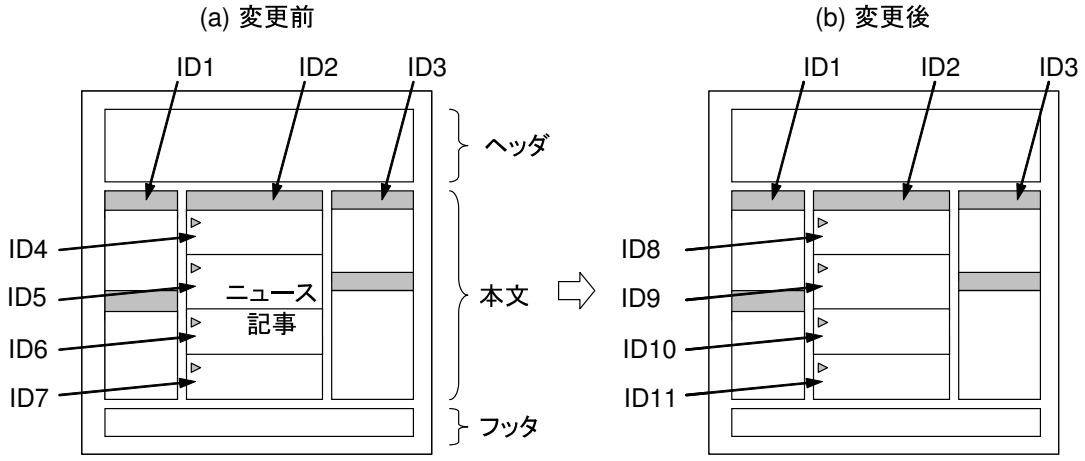


図 2.31: ID を利用している実際の Web コンテンツの例

一方で，IDValueMatch はユニークな識別子を用いた表現であり，静的 Web ドキュメントに対して確実にノードを指示することが出来る．しかしながら，図 2.30 から明らかなように，IDValueMatch は nonexistent の割合が 50%以上を占めている．これは，実際に Web コンテンツに対する動的な変更が多く行われていることを示す．本実験では，文書に対し後からノード ID を付与したが，文書生成と同時にノード ID を付与することが可能であれば．nonexistent の数を減少させることが出来る．しかしながら，ID の付与を実施することのみでは XPath の頑健性が改善されるとは限らない．次節では，XPath の頑健性をさらに改善するための今後の課題について考察する．

2.3.3 考察

近年 Web コンテンツの開発では，ある要素に ID 属性や CLASS 属性を与え，開発効率を向上させる手法が用いられつつある．例えば，これら属性を CSS[4] 等と組み合わせて適切に利用することにより，Web コンテンツの内容とページのレイアウトとを分離して開発・管理することが可能となる．従って，ID 属性や CLASS 属性の多くは，Web コンテンツにおける要素の識別や分類のために付与されていると考えられ，また，レイアウトの変更などにも影響されにくいことから，これらを適切に利用することで XPath の頑健性の向上が期待出来る．

ID を利用している実際の Web コンテンツ[3]の例を図 2.31 に示す．この Web コンテンツは主にヘッダ，本文，フッタから構成されており，本文はさらに 3 列から構成される．各列のタイトルに相当するアンカータグ (A) には ID が付与されている (ID1-ID3)．さらに，本文の中央列のブロック要素にも ID が付与されている (ID4-ID7)．これら 4 つの ID は，DIV タグ内で記載されているニュース記事に付与されており，ニュース記事が更新された場合は異なる ID が付与される (図 2.31(b))．

一方，Web コンテンツを携帯電話等小画面デバイスで閲覧する場合，オリジナルの Web コンテ

[3]2004 年の 10 月 30 日と 11 月 10 日の World Wide Web Consortium トップページ (http://www.w3c.org)

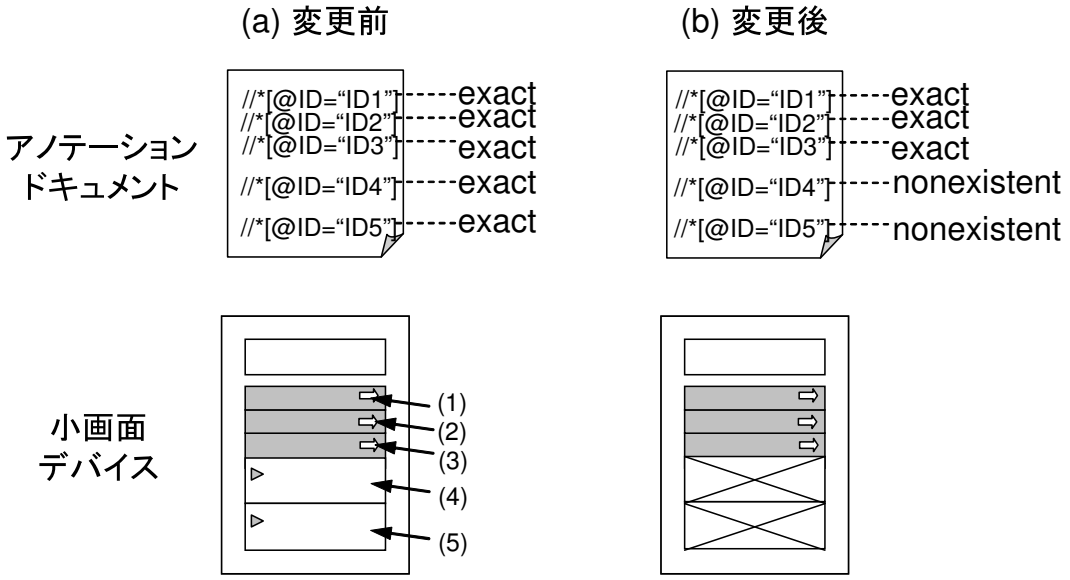


図 2.32: ID による XPath 表現及び小画面デバイスの表示例

ンツの一部を加工して表示させる (クリッピング) 手法 [96, 100] が用いられる場合がある．図 2.32 に，ID を利用した XPath 表現によるアノテーション文書に基づき，クリッピングを実施した場合の表示例を示す．尚，クリッピングによる Web コンテンツ適応に関しては 2.4 節で詳しく述べる．

例えば，限られた表示領域を有効利用するために，コンテンツのタイトルのみを表示させ詳細な情報はリンク先で表示を行うことなどにより表示領域を削減したり (図 2.32(1-3))，最新のニュース記事の上位数個を抜粋したり (図 2.32(4, 5)) することが必要とされる．この際，オリジナルの Web コンテンツの一部を指示するために XPath 表現の生成が必要となる．ここで，ID による XPath 表現を生成した場合，この Web コンテンツ中では，ニュース記事の更新に伴い ID が変更されるため，変更後に適切に画面に表示されないことも多い (図 2.32(b))．これに対し，ニュース記事に対する XPath 表現生成時に，ニュース記事の ID (ID4, ID5) を用いずにタイトルの ID (ID2) からの相対位置で指定した場合，変更後でも指示対象が存在しより頑健な XPath となる (図 2.33)．

この様に，ID による XPath 表現を用いる場合，Web コンテンツに対しどの様に ID が付与されているのか，どの要素が変更頻度が高いのか，アノテーションをどのような目的に利用するのか等を検討した上で XPath 表現を適切に選択することが，XPath の頑健性を改善する上で重要となる．現在，World Wide Web Consortium (W3C) では，XML 文書内の要素を識別するための統一的な方法が提案されている [101]．今後，この様な仕組みを利用した Web コンテンツ開発手法等，頑健性を改善するための手法についてさらなる検討を行っていきたい．

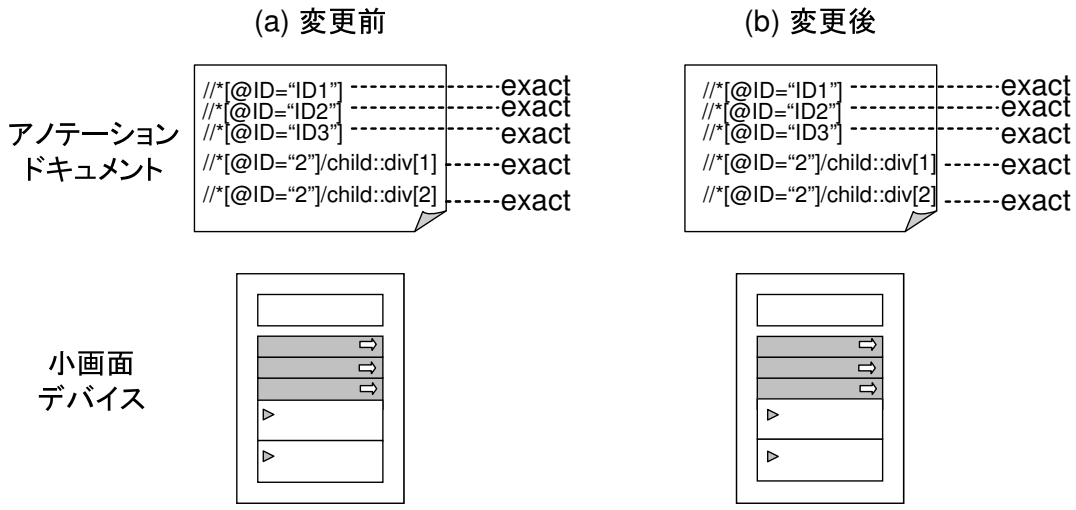


図 2.33: ID と相対指示による XPath 表現及び小画面デバイスの表示例

2.4 適用例

近年，Web の閲覧手段は，デスクトップ上で用いる Web ブラウザから，音声合成技術を利用し Web コンテンツを音声で読み上げる音声ブラウザ [8-10], 携帯端末等の小画面デバイスでの閲覧を可能とする専用ブラウザ [11-13] 等多様化してきた．一方，多様化する Web クライアントや個々のユーザの嗜好に対応するためには，Web コンテンツをそれぞれの環境にあわせて開発・提供する必要がある場合も多く，それぞれを個別に作成した場合，開発コストの増大を招くだけでなく，管理も大変なものとなる．

例えば，電子商取引のための Web コンテンツは製品情報や製品画像，他のサイトへのリンクなど様々な情報が含まれている．この様な情報集積度の高い Web コンテンツはデスクトップ上での閲覧を想定し開発されていることが多い．これらの Web コンテンツの一部を，携帯電話や PDA などに代表される小画面の端末向けに再利用し情報発信することは，開発コスト削減や保守性の点から有用であると考えられる．さらに，ユーザの嗜好に合わせてポータルサイトの Web コンテンツを構築する際，既存の Web コンテンツの一部をポータルサイトにはめ込むこと等も実際に行われている．本節では，まず，アノテーションに基づく Web コンテンツ適応の基盤となる Web トランスコーディング [36, 96] について簡単に説明を行う．次に，アノテーションフレームワークの適用例として，携帯端末向け Web トランスコーディング [59] とポータルサイト構築のための Web クリッピング [63, 64] について述べる．

2.4.1 Web トランスコーディング

Web 閲覧の多様な利用形態に応じて Web コンテンツを変換し，コンテンツの再利用を促進させるためには，変換に必要なモジュールをリクエストに応じて呼び出す共通の基盤が必要となる．Web トランスコーディング [36, 96] は，各種変換エンジンを呼び出すランタイムの仕組みであり，その実装は製品として出荷されている．

Web トランスコーディングの概要を図 2.34 に示す．Web 閲覧では，通常クライアントから送信されたリクエストや Cookie 等に保存されたユーザのプロファイルなどに従い，HTTP サーバ上の Web コンテンツが取得される．トランスコーディングプロキシはこの中間に位置し，コンテンツの変換を行う．クリッピングエンジンや音声コンテンツ生成エンジン，翻訳エンジン等トランスコーディングに必要なモジュールは，トランスコーディングプロキシにプラグインすることが可能である．トランスコーディングプロキシは，HTTP リクエストヘッダに記述されたデバイスの機種等のクライアント情報やリクエストされた URI に基づき，適宜必要なプラグインを駆動する．プラグインにアノテーション文書が必要な場合は，同時にアノテーション文書も取得される．アノテーション文書は HTTP サーバ上に存在する場合もあれば，他のサーバに存在する場合もある．

例えば，クライアント情報から，クライアントがデスクトップで通常使用される Web ブラウザと同等の性能と判断された場合は，どのプラグインも駆動せずにオリジナルの Web コンテンツを送信する (図 2.34 (a))．クライアントが携帯端末の場合は，HTML 文書から Compact HTML[102]

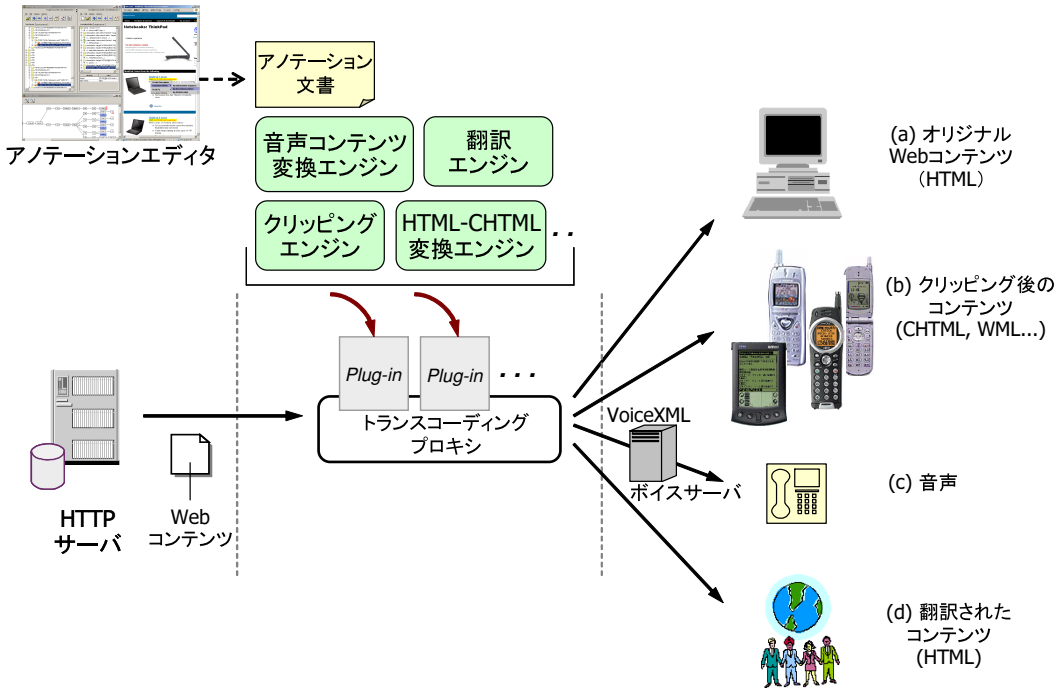


図 2.34: Web トランスコーディングの概要

に変換するための HTML-CHTML 変換エンジンなど，携帯端末向けのマークアップ言語に変換するためのプラグインが駆動され，Compact HTML や Wireless Markup Language[103] などにトランスコーディングされたコンテンツが送信される (図 2.34(b))．ここで，Web コンテンツをその重要度や意味に応じて並べ替えたり取捨選択する必要がある場合は，変換の前処理としてクリッピングエンジンが駆動され，あらかじめコンテンツの変換が行われる．クリッピングエンジンとクリッピングのためのアノテーション語彙の詳細については，次節で説明する．

また，Web ブラウザを搭載していない電話機などのデバイスからアクセスされた場合には，VoiceXML[104] 等音声用コンテンツが生成され，さらに VoiceXML から音声へ変換するボイスサーバ [105] などを経由して音声が送信される (図 2.34(c))．さらに，翻訳エンジン [106] を利用することにより機械翻訳されたコンテンツを送信することも可能である．また，全盲のユーザなど，音声ブラウザ [8-10] やスクリーンリーダー [32, 33] 等の読み上げ用ソフトウェアを用いて Web を閲覧しているユーザに対して，ページ内のナビゲーションや ALT 属性の付加などを行うトランスコーディングなども研究されている [8, 9]．

この様に，Web トランスコーディングを用いることで，Web 閲覧の多様な利用形態に応じて Web コンテンツを変換しコンテンツの再利用を促進することが可能となる．以下，2.4.2 節及び 2.4.3 節では，本研究で提案したアノテーションフレームワークの適用例として，携帯端末向けトランスコーディング及びポータルサイト構築のための Web クリッピングについて説明する．

2.4.2 携帯端末向け Web トランスコーディング

携帯端末向け Web トランスコーディングは，クリッピングエンジン [36, 96] と 2.2 節で述べたアノテーションエディタにより実現される．クリッピングエンジンは，クリッピングのためのアノテーション文書と HTTP で送信されるリクエストを基に，オリジナルの Web コンテンツを変換する．まず，オリジナルの Web コンテンツをアノテーションを基にクリッピングし，HTML ファイルとして出力する．次に，その HTML ファイルを基に，Compact HTML 等携帯端末用のマークアップ言語に変換する．

図 2.35: Web コンテンツのクリッピング例

ページクリッピングのためのアノテーション語彙は，クリッピングの状態を記述するための語彙から構成される[4]．このアノテーション語彙は次節で説明する Web クリッピングにも用いられている [107]．例えばアノテーション文書作成者は，`keep` や `remove` 等アノテーション記述要素を用いてコンテンツの取捨選択を指定することが出来る．Web コンテンツのクリッピング例を図 2.35 に示す．また，オリジナルの Web コンテンツの HTML ソースの概要を図 2.36 に，クリッピング後の HTML ソースの概要を図 2.37 に示す．

クリッピングされたページ (図 2.35(b)) は，オリジナルのページ (図 2.35(a)) と比較すると，ロゴ，ニュース記事のヘッドライン及びヘッダとフッタに含まれるリンクが保持され，他の部分は破棄されている．さらに，オリジナルのページでは，ピクセルの絶対値 (760 ピクセル) によりヘッダやフッタの幅が指定されていたのに対し (図 2.36)，クリッピングされたページでは小画面にコンテンツが収まるよう画面幅に対する相対値 (100%) により幅が指定されている (図 2.37)．

次に，図 2.35 の Web クリッピングに利用されているアノテーション文書の一部を図 2.38 に示す．`description` はアノテーション記述要素であり，`target` や `take-effect` を属性として保持する．

[4] ページクリッピングのためのアノテーション語彙 [96] を付録 (表 A.2) に示す

```
<html>
  <head>
    <title>IBM United States</title>
  </head>
  <body bgcolor="#ffffff" leftmargin="2" topmargin="2" marginwidth="2"
        marginheight="2" alink="#006699">
    <table width="760" border="0" cellspacing="0" cellpadding="0">
        :
    </table>
    <table width="760" border="0" cellspacing="0" cellpadding="0">
        :
    </table>
    <table width="760" border="0" cellspacing="0" cellpadding="0">
        :
    </table>
  </body>
</html>
```

IBMロゴやサーチフォーム，リンクなどを含むヘッダ

サイドメニューやニュース記事を含むメインコンテンツ

フッタ

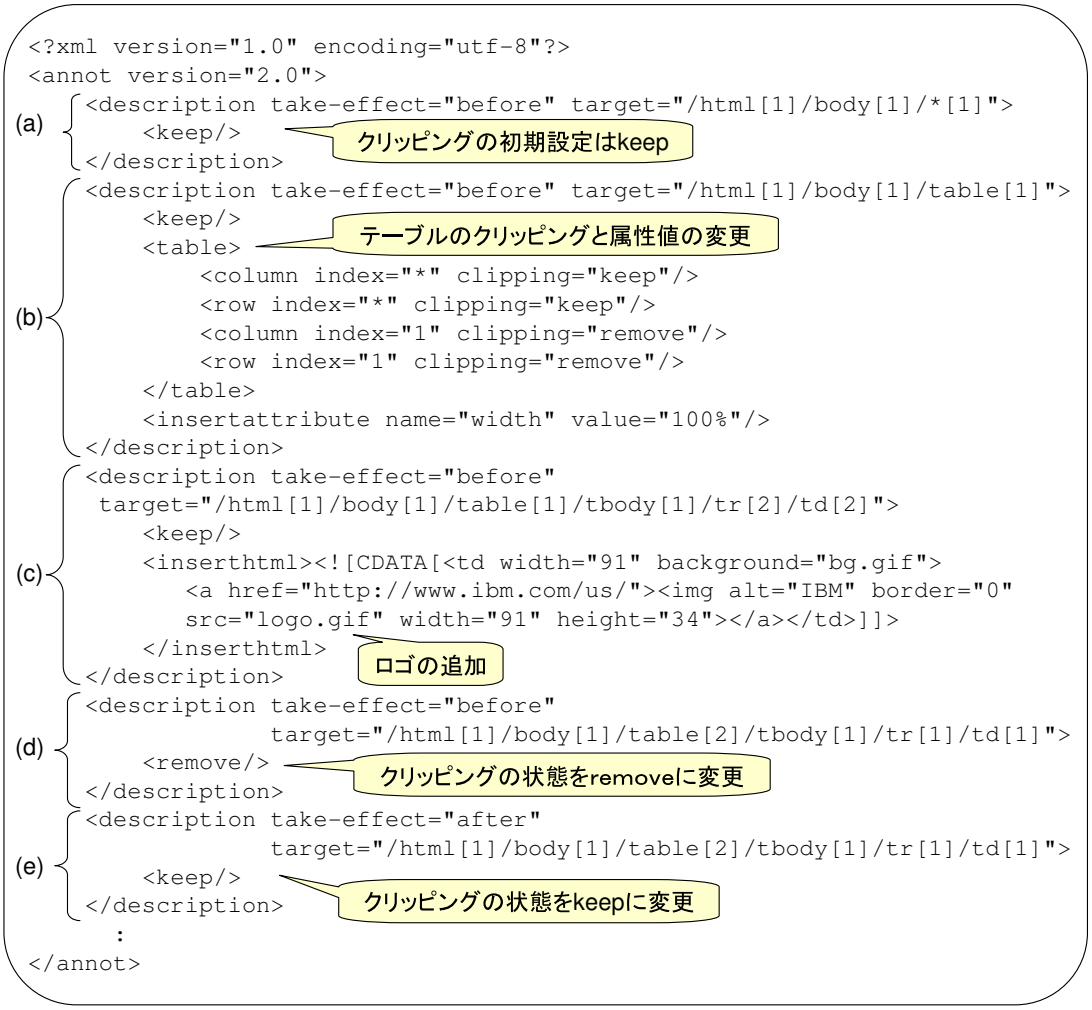
図 2.36: オリジナル Web コンテンツの HTML ソースの概要

```
<html>
  <head>
    <title>IBM United States</title>
  </head>
  <body bgcolor="#ffffff" leftmargin="2" topmargin="2“ marginwidth="2"
        marginheight="2" alink="#006699">
    <table width="100%" border="0" cellspacing="0" cellpadding="0">
      :
    </table>
    <table width="100%“ border="0“ cellspacing="0“ cellpadding="0">
      :
    </table>
  </body>
</html>
```

IBMロゴ，リンク及びニュース記事

フッタ

図 2.37: クリッピング後の Web コンテンツの HTML ソースの概要




```
<?xml version="1.0" encoding="utf-8"?>
<annot version="2.0">
(a) <description take-effect="before" target="/html[1]/body[1]/*[1]">
        <keep/>
    </description>
(b) <description take-effect="before" target="/html[1]/body[1]/table[1]">
        <keep/>
        <table>
            <column index="*" clipping="keep"/>
            <row index="*" clipping="keep"/>
            <column index="1" clipping="remove"/>
            <row index="1" clipping="remove"/>
        </table>
        <insertattribute name="width" value="100%"/>
    </description>
(c) <description take-effect="before"
     target="/html[1]/body[1]/table[1]/tbody[1]/tr[2]/td[2]">
        <keep/>
        <inserthtml><![CDATA[<td width="91" background="bg.gif">
           <a href="http://www.ibm.com/us/"><img alt="IBM" border="0"
           src="logo.gif" width="91" height="34"></a></td>]]>
        </inserthtml>
    </description>
(d) <description take-effect="before"
                target="/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]">
        <remove/>
    </description>
(e) <description take-effect="after"
                target="/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]">
        <keep/>
    </description>
      :
</annot>
```

図 2.38: Web トランスコーディングのためのアノテーション例

target 属性は値として XPath 表現が指定されており，この XPath 表現は，アノテーションの適用対象となる対象文書中のノードを指示する．take-effect 属性は，指示対象ノードの前後どちらを基準にしアノテーション記述要素が有効化されるかを示している．例えば，図 2.38(a) で示されるアノテーション記述要素は，target 属性値として/html[1]/body[1]/*[1]，take-effect 属性値として before が指定されている．これは，body 要素の 1 番目の子要素の前からアノテーション記述要素が有効化されることを示している．さらに子要素として keep を指定することで，新たにアノテーション記述要素が出現しない限り対象文書の内容は保持される．

テーブルに対するクリッピング処理は複雑になるため，row や column 等テーブル専用の語彙が用意されている．テーブル専用の語彙を用いることで，複雑なアノテーション記述要素や XPath 表現を記述することなくテーブルをクリッピングすることが可能となる．図 2.38(b) では，1 番目の table 要素の前から対象文書の内容を保持することや，insertattribute によりテーブル幅を 100%に変更することが示されている．insertattribute は対象文書に属性名と属性値を指定すための語彙である (表 A.2).

さらに図 2.38(b) では，XPath で指示されるテーブルの最初の縦列と最初の横列を破棄し，その他の部分は保持することが示されている．XPath 表現`/html[1]/body[1]/table[1]` は図 2.35 中のロゴや検索フォーム，“Home”, “Products & Services” 等のメニュー項目を含むヘッダ部分を指示する．ここで，`index=‘‘*’’` は，他の `row` や `column` で指定されない限り，すべての縦列や横列にクリッピング操作を適用させるためのワイルドカードである．この様に，テーブル処理専用の語彙は縦列及び横列を処理単位とし，セルを処理単位としない．しかしながら，図 2.38(b) で示される注釈要素を対象文書 (図 2.35) に適用すると，ロゴが削除されてしまう．このため，ロゴを追加するためのアノテーション記述要素 (図 2.38(c)) が必要となる．図 2.38(c) 中，`inserthtml` 以下の CDATA はロゴを追加するための記述である．

図 2.38(d) で示されるアノテーション記述要素は，2 番目の table の 1 番目の tr 要素の 1 番目の td 要素から対象文書の内容を破棄することを示しており，図 2.38(e) はその td 要素の後に出現する対象ドキュメントの内容を保持することを示している．この 2 つのアノテーション記述要素により，`/html[1]/body[1]/table[2]/tbody[1]/tr[1]/td[1]` で示される td 要素とその子孫のみが破棄される．

以上の様なアノテーションを用いることで，既存の Web コンテンツから，必要な部分のみを切り出す等の加工を行い，携帯端末に適した形式で情報配信を行うことが可能となる．本研究で提案したアノテーションエディタは，クリッピングのためのアノテーション語彙を編集するツールとして製品化されており [96, 100]，実際の Web コンテンツ変換に利用されている．さらに，音声コンテンツへの変換や翻訳等の Web コンテンツ適応に対しても，アノテーションの語彙を切り替えることで，アノテーションエディタを用いて付加情報を付与し，変換の精度を高めることが出来ると考えられる．また，図 2.38(b), (c) の様な複雑なアノテーション記述要素の作成には，HTML の WYSIWYG エディタ上での編集操作からアノテーションの作成を可能とする「例示によるアノテーション作成ツール」[60, 61] 等も有効であると考えられる．今後これらのツールをアノテーションエディタ上に統合することで，アノテーションの種類やアノテーション作成者の好みに応じてエディタを適宜切り替えながらアノテーションを作成することを可能とし，より効率の良いアノテーション編集環境の提供を目指したい．

2.4.3 ポータルサイト構築のための Web クリッピング

近年，高度情報化社会の進展に伴い，大量の Web コンテンツから必要な情報を効率よく収集するための手段や，コミュニティ間で情報を共有するためのコミュニケーション手段が必要とされてきた．例えば，ニュース記事や株価などの必要な情報及びその情報へのリンクを単一のコンテンツに集約させ，Web の入り口 (portal) として情報提供するポータルサイトは，この様な要求に対し広く利用されつつある．

ポータルサイトの構築に際しては，ポータルサイト構築のための支援環境であるポータルサーバ [107, 108] を利用することで，複数の Web コンテンツ及び Web アプリケーションを集約し単一

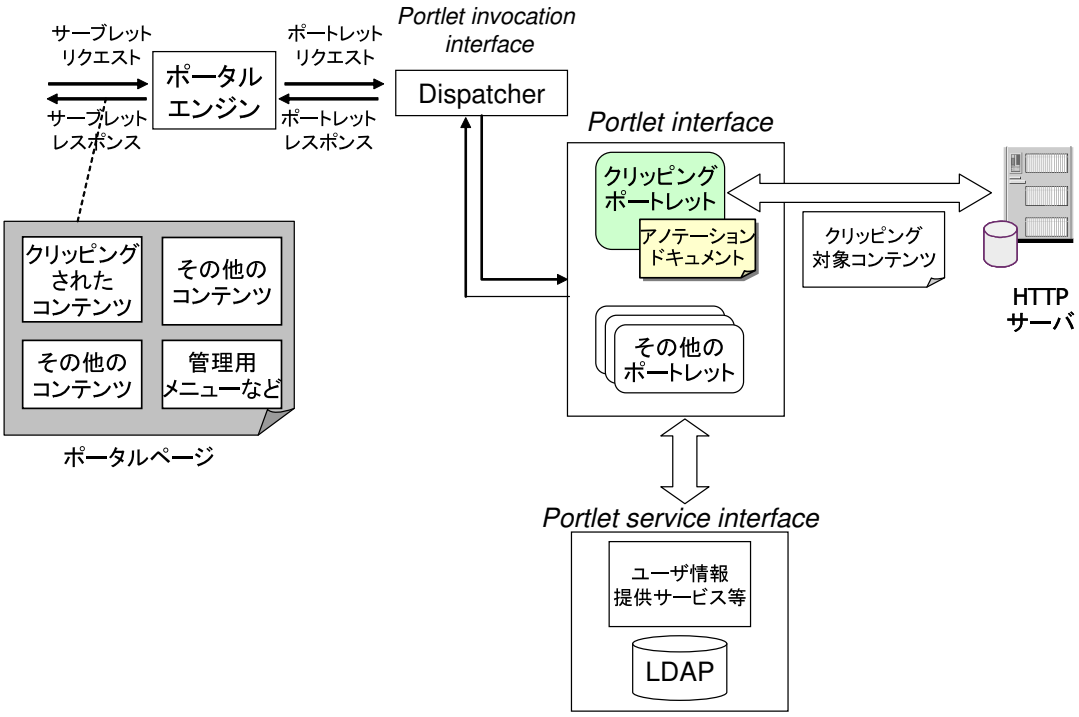


図 2.39: クリッピングポートレットを含むポータルサーバの構成

Web コンテンツ (ポータルページ) を簡単に構成することが出来る．各 Web コンテンツ及び Web アプリケーションはポートレット [109] と呼ばれる部品として扱われ，ポータルページ生成のために自由に組み合わせることが出来る．

この様なポータルページ生成に際し，既存の Web コンテンツの一部を切り出してポータルページへはめ込むためにクリッピングエンジンを活用することは，ポータルページ生成の効率向上にも有用であると考えられる．図 2.39 にクリッピングポートレットを含むポータルサーバの構成を示す．ポータルサーバはサーブレットアプリケーションとして動作する．まず，ポータルエンジンはサーブレットリクエストをサーブレットコンテナから受け取り，複数のポートレットリクエストに変換する．その後，そのリクエストは Dispatcher により適宜ポートレットにディスパッチされる．ポートレットはユーザ情報などを提供するポートレットサービスを利用しコンテンツを受け取る．そして，ポータルエンジンは複数のポートレットレスポンスを集約しポータルページを生成する．クリッピングポートレットはクリッピングエンジンと連携しポートレット [109] として動作することで，既存 Web コンテンツの一部をポートレットとして利用することが可能となる．

図 2.40 にクリッピングポートレットの設定を行うためのツールとして，アノテーションエディタを応用した様子を示す．アノテーションエディタの機能は，ウィザード形式の Web アプリケーションとして提供され，アノテーション対象箇所をブラウザ上で選択しアノテーション文書を保存するなど，編集ステップに沿ってクリッピングポートレットを設定することになる．例えば，ポートレット管理メニュー (図 2.40(a)) から “Web Clipping” を選択し，クリッピング対象となる Web コンテ

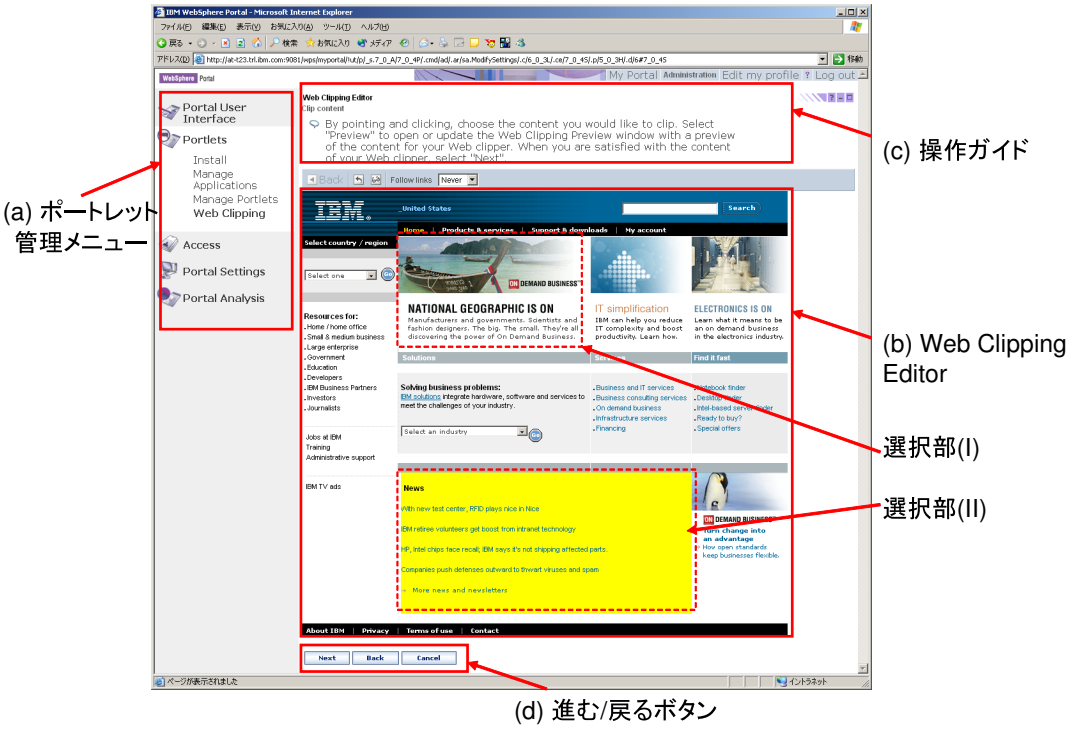


図 2.40: クリッピングポートレットのためのアノテーションエディタ

ンツの URL を指定すると，画面中央に Web Clipping Editor(図 2.40(b)) が操作ガイド (図 2.40(c)) と共に表示される．クリッピングしたい箇所をブラウザ上で選択し，進むボタン (図 2.40(c)) を押下すると，Web Clipping Editor の選択状態に応じて HTML 中の要素を指示する XPath 表現やその取捨選択などの操作内容がアノテーション文書として生成され，ポータルサーバに配備 (deploy) される．クリッピングポートレットは，これらのアノテーション文書に基づいた処理を実施することで，既存 Web コンテンツからユーザの望む内容をクリッピングしポータルページ中に表示することが可能となる．

クリッピングポートレットによる出力を含むポータルページの例を図 2.41 に示す．図中左上には，クリッピングされた結果コンテンツに含まれる画像 (図 2.40 選択部 (I)) とニュースヘッドライン (図 2.40 選択部 (II)) がポータルページの一部として表示されている．その周りには世界各国の標準時刻やニュースなど他のポートレットの出力結果が表示されており，クリッピングポートレットと合わせて利用することが出来る．

次に，図 2.40 の例から生成されたアノテーション文書を図 2.42 に示す．図 2.42(a) は BODY 以下のノードに対するアノテーション記述要素であり，デフォルトのクリッピング操作は `remove` となっている．図 2.42(b), (c) はそれぞれ画像とニュースヘッドラインを保持するためのアノテーション記述要素であり，図 2.40 中の選択部 (I), (II) に対応する．

クリッピングポートレットのためのアノテーションエディタはポータルサーバとともに製品として出荷され [110]，実際のポータルサイト構築に利用されている．例えば，ある企業のポータル

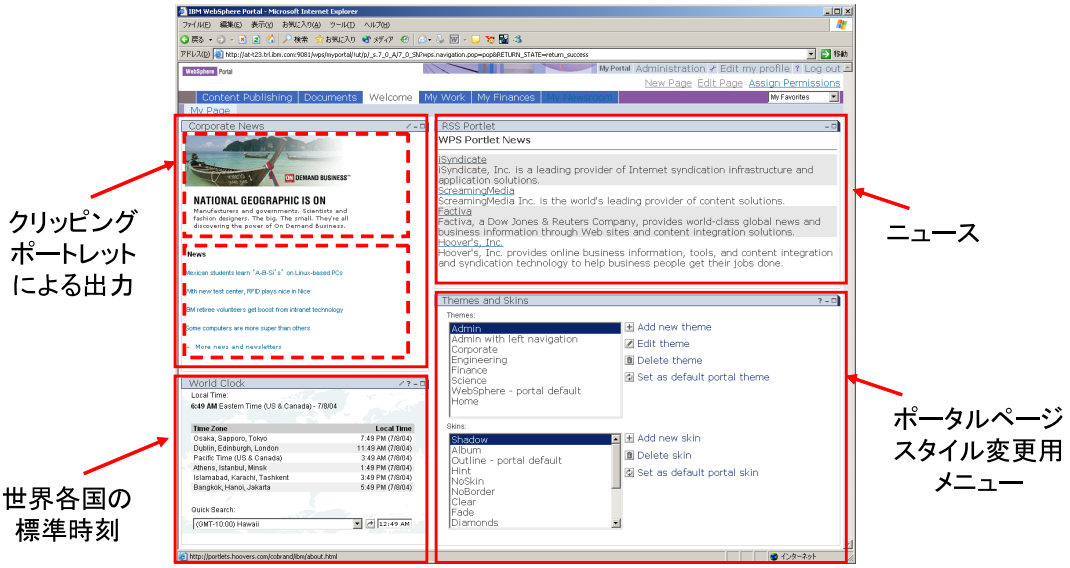


図 2.41: クリッピングポートレットによる出力を含むポータルページ

サイト開発に際し，数千に及ぶ関連企業のページを集約する必要があった．しかし，すべての関連企業に対しポータルページ用の Web コンテンツを別途開発させると莫大な開発コストがかかり，短期間でのポータルページ構築を実現することは出来ないという課題があった．そこで，本論文で提案したアノテーションフレームワークを用いてクリッピングを行い，関連企業が所持している既存の Web コンテンツを有効利用することで，開発コストを抑えつつ短期間でポータルサイトを開発することが可能となった．この様に，本論文で提案した Web コンテンツ適応のためのアノテーションフレームワークは，既存 Web ページの効率良い再利用を可能とし，開発コストの削減や開発効率の向上に大きく貢献するものであったと言える．

```
<?xml version="1.0" encoding="UTF-8"?>
<annot version="2.0">
(a) <description take-effect="before" target="/html[1]/body[1]/*[1]">
      <remove/>
    </description>
(b) <description take-effect="before“ target="/html[1]/body[1]/table[2]
                          /tbody[1]/tr[1]/td[3]/table[1]/tbody[1]/td[1]">
      <keep/>
    </description>
    <description take-effect="after"  target="/html[1]/body[1]/table[2]
                          /tbody[1]/tr[1]/td[3]/table[1]/tbody[1]/td[1]">
      <remove/>
    </description>
(c) <description take-effect="before"
     target="/html[1]/body[1]/table[2]/tbody[1]
      /tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/p[2]/strong[1]/div[1]">
      <keep/>
    </description>
    <description take-effect="after"
     target="/html[1]/body[1]/table[2]/tbody[1]
      /tr[1]/td[3]/table[2]/tbody[1]/tr[1]/td[1]/p[2]/strong[1]/div[1]">
      <remove/>
    </description>
      :
</annot>
```

図 2.42: クリッピングポートレットに利用されるアノテーション文書の例

2.5 結論

本章では，Web コンテンツ適応のためのアノテーションフレームワークを提案した．まず，対象文書に対し付加情報であるアノテーションを外部的に与えるための，外部アノテーションの枠組みを提案した．外部アノテーションの枠組みは，対象文書のスキーマや文書自体を変更することなく，アノテーションを付与するための構造を規定している．さらに，アノテーション語彙のメタモデルの提案も行った．メタモデルとアノテーション語彙のスキーマとの対応付けを行うことで，特定のアノテーション語彙に依存しないエディタの設計が可能となった．次に，外部アノテーションの枠組みに基づき設計されたアノテーションエディタについて述べた．アノテーションエディタは，アノテーション文書の編集環境であり，アノテーション語彙に対して拡張することが可能である．また，アノテーションエディタは対象文書中でアノテーションの対象となるノードを指示するための XPath 表現の生成補助機能を有する．

次に，本章では Web コンテンツの変更に対する XPath 表現の頑健性評価について実際の Web コンテンツを対象に実験を行い，指示表現の生成・編集と利用における留意点についても分析を行った．この結果，レイアウトの変更など大規模な Web コンテンツの変更が加わらない通常の変更時において ChildPosSeq は頑健である一方，DescendantPos は対象ドキュメントの変更に伴い予想外のノードを指示する恐れがあるという意味で慎重に利用すべきであるということが明らかとなった．さらに，今後頑健性を向上させる上での課題について検討を行い，指示表現に ID や CLASS 属性を用いることによる頑健性向上の可能性と，運用上の注意点を明らかにした．

最後に，アノテーションエディタの適用例として，携帯情報端末向け Web トランスコーディングとポータルサイト構築のための Web クリッピングについて述べた．これらは共に製品化され実際に企業のホームページ作成などに利用されている．これらの実際の適用例などからも，本章で提案したアノテーションフレームワークは，開発コストをかけずに短期間で Web コンテンツを再利用する必要のある状況に対し有効であるといえる．

第3章　Webアプリケーションモデル抽出支援手法

本章では，Web アプリケーションを対象に，既存アプリケーションの振舞いを，HTTP セッション上で授受される情報に着目した動的解析により明らかにし，モデルの抽出を支援する手法を提案する．提案手法を用いることにより，Web アプリケーションの振舞いを表すモデルの抽出が可能となり，例えば CGI を基盤として開発された既存アプリケーションを J2EE プラットフォームへ移行するなど，実行環境の移行への対応を円滑に行うことが可能となる．

まず，3.1 節では Web アプリケーション開発におけるフォワードエンジニアリングを効率良く行うための，モデルに基づく Web アプリケーション開発 [53, 67-70] について述べる．次に，3.2 節では Web アプリケーション再構築のための，動的解析に基づくモデル抽出支援手法について述べ，さらに，3.3 節では実際の Web アプリケーションを対象としたモデル抽出のための実験を行い，提案手法を用いることでより多くの要件定義を満たすモデルの抽出が可能となり，既存資源をより効果的に利用可能であることを示す [65, 66]．最後に，3.4 節で本章をまとめる．

3.1　モデルに基づく Web アプリケーション開発

3.1.1　Web アプリケーションの設計

近年，Web アプリケーション開発には，Model 2 アーキテクチャ[43] と呼ばれる MVC (Model-View-Controller) パラダイムに基づく開発手法が用いられつつある．Model 2 アーキテクチャ(図 3.1) では，すべての HTTP リクエストが Controller によって処理され (図 3.1(1))，Controller がサーバ側プログラムであるアクションを起動することによりメソッド呼出しやサーバオブジェクトの生成が行われる (図 3.1(2))．Controller のアクション起動により呼び出されるアプリケーションロジックや，Enterprise JavaBeans (EJB) [111] 等を用いたアプリケーションの状態を保持するバックエンドデータ及びその処理などを Model と呼ぶ．アプリケーションロジックなどの実行により Model の状態が更新されると，Controller はその結果に基づきページ遷移先を決定した上で，View へと処理を移行する (図 3.1(3))．

次に，Web ブラウザ上に表示される内容をレスポンスとして生成する View においては，Model 上のサーバオブジェクトを参照することにより，動的に生成される部分の表示内容を決定する (図 3.1(4))．このように，Model 2 アーキテクチャでは Model と View の役割が明確に区分され，ページフローの制御に関わる処理は Controller に集約される．

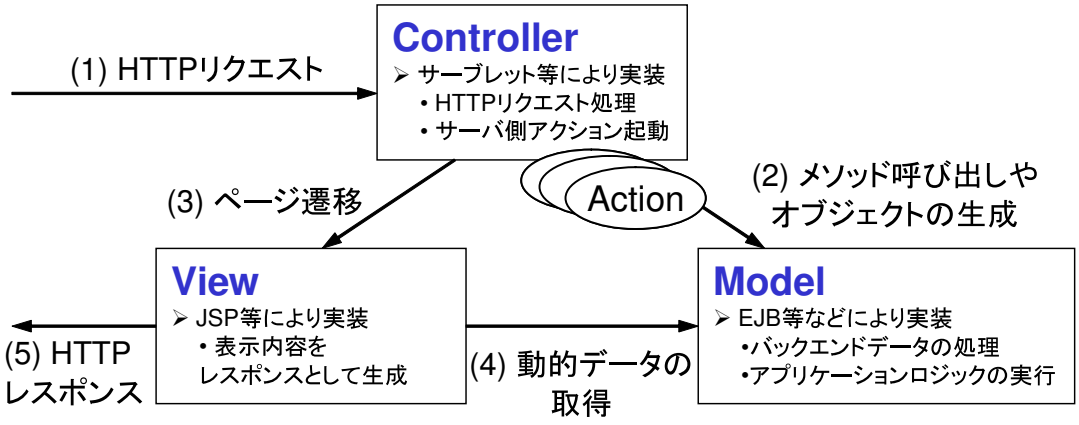


図 3.1: Web アプリケーションにおける MVC パターン

MVC は Web アプリケーションのアーキテクチャデザインを規定するものであり，MVC に沿った設計を行うことで，各コンポーネントの再利用性の向上や，並行開発が容易になる等の利点がある．しかしながら，設計から実装段階に移行する際には，実行環境で必要となる様々なファイル (JavaServer Pages (JSP) [21]，Java クラス，構成情報など) 間で，そのそれぞれが参照する情報について整合性を保つ必要があり，その作業は各成果物に責任のある開発者間で行う必要がある．例えば，図 3.2(a) に示したようにフォームの入力によりアクションが起動される場合，ページ遷移先はアクションに該当するサーブレットプログラム (図 3.2(b), “/simpleShop/SearchProduct.do”)，あるいはそのアクションに引き続いて起動される他のアクションによって決定される．したがって，ページ遷移先の URI がどのアクションで記述されているかは自明ではないため，単純に遷移先ページの URI を変更するだけでも容易ではない．

さらに，図 3.2(b) で name=“keyword” として定義されたパラメータは，サーブレットプログラム内で図 3.2(c) のようなステートメントで参照される．そのため，HTTP リクエストを介してパラメータを利用するサーブレットのプログラマは，HTML 中における入力フォームの対応するパラメータ名との整合性に注意を払わなければならない．例えば，ページデザイナが入力フォームのパラメータ名を複数形 (“keywords”) に変更しただけで，サーブレットプログラムでは実行時にパラメータの適切な値が参照できなくなる．この様な不整合は実行時エラーとして検出されないため，各開発物の統合時に不具合が生じてもその根本原因を突き止めることは容易ではない．さらに，どのようなパラメータを与えるかは，Web アプリケーションの実行環境が決まるまで分からないことが一般的である．このため，すべてのパラメータ名が Web アプリケーションの設計時に作成する仕様書に記述されていないことも多く，開発効率向上の妨げとなっている．

3.1.2 モデルに基づく Web アプリケーション開発

本論文では，アプリケーション構築に関わる様々な開発者 (設計者，ページデザイナ，プログラマ) が共通の前提として合意出来る実行環境に非依存のアプリケーションの仕様 (モデル) を提案

(a) 入力フォーム

(b) 入力フォームのソース (KeywordSearchPage.jsp)

```
<form action="/simpleShop/SearchProduct.do" method="GET">
   keyword: <input name="keyword" type="text">
   <input type="submit" value="listProduct">
</form>
```

(c) サーブレットプログラム (KeywordSearch.java)

```
public void doGet(HttpServletRequest request,
                  HttpServletResponse response){
...
   searchKeyword = request.getParameter(“keyword”);
}
```

図 3.2: 入力フォームとサーブレットプログラムの例

する．さらに，モデルに基づいてサーバページやアクションのためのプログラムを自動生成し，開発支援を行うための環境を提案する．モデルに関しては 3.1.3 節，開発支援環境に関しては 3.1.4 節でそれぞれその詳細を述べる．

図 3.3 にモデルに基づく Web アプリケーション開発の概要を示す．まず，モデル設計者が作成したモデルを基に，開発成果物の雛形であるスケルトンコードが自動生成される．各開発者はスケルトンコードを基に並行開発を行う．開発成果物が仕様に適合しているか否かは，モデルとの整合性確認をすることにより判断出来る．これにより，開発プロセスの依存関係に起因する作業の遅延をある程度回避することが可能となり，並行開発を円滑に進めることが可能となる．以下に各開発者毎の役割を説明する．

モデル設計者

モデル設計者は，Web アプリケーションの仕様であるモデルの作成および管理を担当する．モデルは仕様の曖昧性を排除するよう，計算機で読み取り可能な形式言語で記述される．大規模な Web アプリケーションの開発においては，モデルの設計・管理作業は，複数のモデル設計者の間で分担して実施出来ることが望ましい．この為，モデルの設計・管理を支援する開発支援環境においては，モデルの一部に変更が発生した場合に，モデル全体の妥当性を確認する必要がある．また，従来からアプリケーションの仕様として利用されてきた自然言語による文書が存在する場合は，モデル設計者がそれらの文書に基づいてモデルを作成する必要がある．

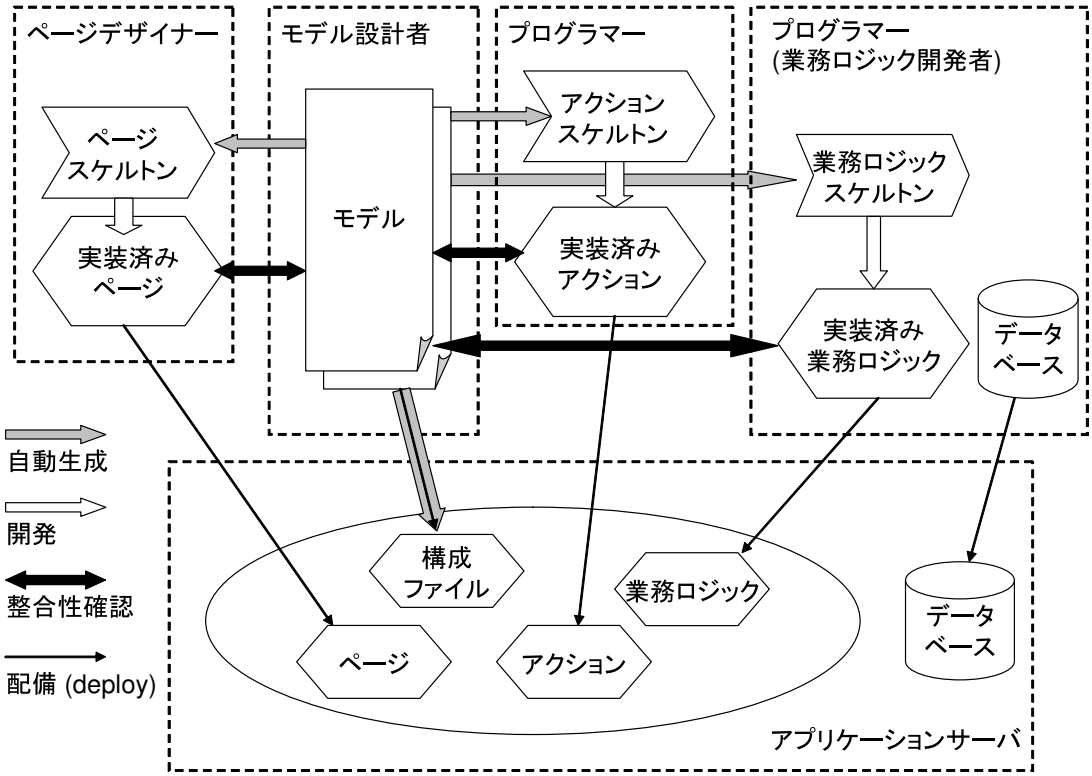


図 3.3: モデルに基づく Web アプリケーション開発

ページデザイナ

ページデザイナは JSP や HTML 等の View に相当する部分の作成を担当する．開発支援環境を用いることで，モデル設計者が記述したモデルから，各ページに必要となる要素を含んだ View の雛形であるページスケルトンを作成することが可能となる．ページデザイナは，このページスケルトンに背景や画像，レイアウト情報などを追加し実際の JSP や HTML を作成する．Web サイト内で用いられる View が増加するに従って，スケルトンコードを利用することによる作業量軽減の効果は大きくなる．ページデザイナは JSP や HTML の作成が終了すると，モデルと実際のページの整合性確認を実施することにより，仕様に則したページであるか否かを判断することが出来る．

プログラマ

プログラマは HTTP リクエストにより呼び出されるプログラムであるアクションの実装を担当する．ページスケルトンと同様に，モデルから受け渡される変数などを含んだプログラムの雛形であるアクションスケルトンが自動生成され，プログラマはアクションスケルトン内に処理の実装を行う．プログラマは開発終了後のプログラムコードとモデルとを比較し，整合性を確認する必要がある．

プログラマはバックエンドにある業務ロジックやデータベースの設計を行う場合もあるが，バッ

クエンドのコンポーネントはWebブラウザ以外のクライアントにも共通して利用出来るように設計すべきであるとの指摘もあり，Webに固有の仕組みとは分離して設計することが望ましい．

各開発者の実装が終了した後，これらの成果物をアプリケーションサーバへ配備(deploy)し，統合テストを行う．提案手法では，deploy前にモデルと開発成果物の整合性確認を行うため，各成果物間でのパラメータ名の齟齬等を防止することが可能となるなど，統合テスト時の負荷が軽減される．

3.1.3 Web Application Descriptor

Web Application Descriptor[50](以下，WAD)はMVCに基づく実行環境により実行されるページ遷移と，各コンポーネントのインタフェースをモデル化するための記述言語である．WADを用いることで，Webアプリケーションの振舞いであるページフローやパラメータを介したデータの授受の流れを定義することが可能となる．一方，ページのレイアウトやデザインの詳細，アクションの実装等，ソフトウェア基盤やクライアント環境に依存する様な情報はWADでは規定せず，ページデザイナやプログラマーの裁量に委ねられる．

Unified Modeling Language (UML) [112, 113] のクラス図による WAD の概要を図 3.4 に示す．WADはページ遷移に関する情報とページ内の表示項目を定義するNavigationPart，アクションのインタフェースを定義するActionPart，さらにサーバオブジェクトに関する情報を定義するBeanPartから構成される．NavigationPart(図3.4(a))はWebアプリケーションにおけるページ遷移を3種類のWadNode，すなわち論理的なページを表すLogicalPage, アクション呼び出しを表すActionInvocation, ページ遷移をグループ化するRegionによって表す．特にRegionは任意個のWadNodeを内部に含むことができ，ページ遷移を階層化された有向グラフとして定義することができる．ActionPart(図3.4(b))は，Controllerにより呼び出されるアクション(Action)のインタフェースとして，入力パラメータ(ActionParam), アクションの実行結果(成功か失敗等)を表す結果コード(ActionResultCode)を定義する．

BeanPart (図3.4(c))に含まれるBean要素は名前とスコープによって識別され，サーバオブジェクトのクラス名を保持する．スコープの値としてはサーブレットAPI[114]で定義されるスコープ名(page, request, session, application)のいずれかが与えられる．BeanAccess要素はアクションや論理ページがサーバオブジェクトに及ぼす実行時の作用を定義するもので，属性値としてCreate, Read, Update, Deleteのいずれかが指定され，図3.1(2), (4)に対応する振舞いを規定する．

NavigationPartに含まれるActionInvocationは1つのEntryPortと，呼び出されたActionResultCodeに応じた複数のExitPortを持つ．ただし，実際にどのアクションが呼び出されるかは，ActionMappingを介して関連付けられたAction要素によって決定される．EntryPortはそのアクションを呼び出すためのURIを表し，ExitPortは他のポートに接続されることによって遷移構造を規定する．このようにActionInvocationのExitPortとその接続先は，図3.1(3)に相当する

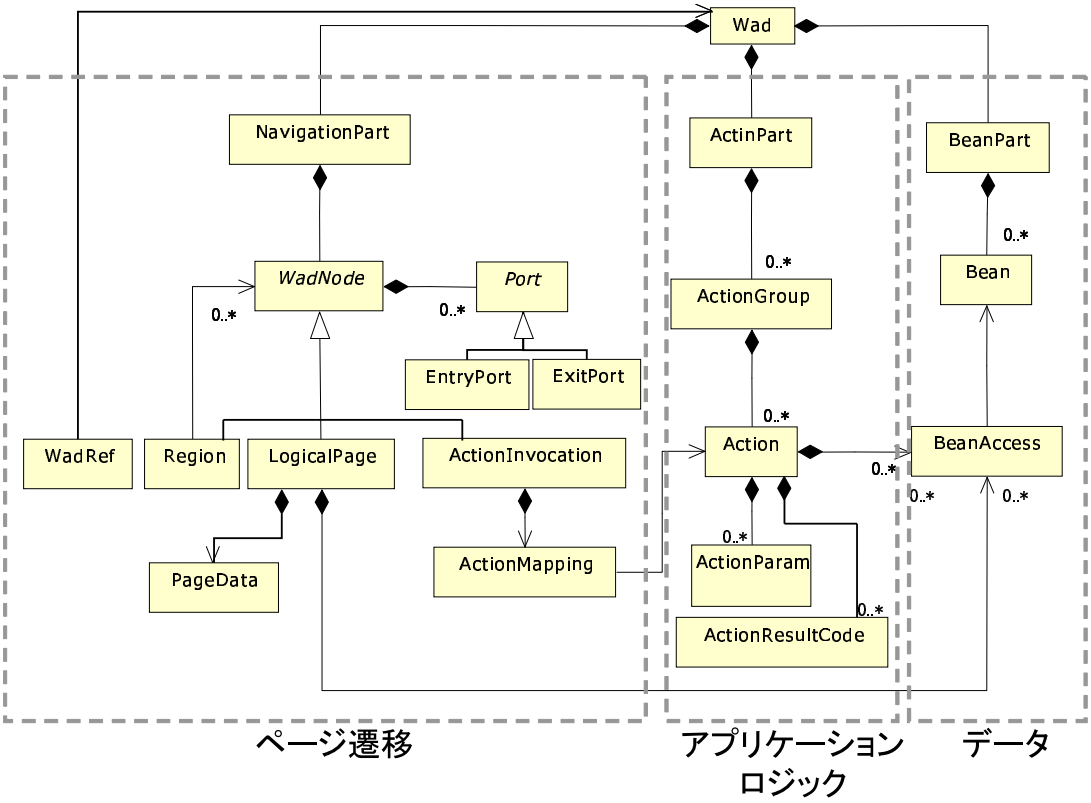


図 3.4: Web Application Descriptor

振舞いをモデル化するものとなっている．また，LogicalPage は通常 1 つの EntryPort と複数の ExitPort を持つ．EntryPort は LogicalPage によって表されるビュー (JSP 等) 自身の URI を表し，ExitPort はそのビューに含まれるハイパーリンクを表す．ExitPort は，HTML のアンカー (<a> タグ)，HTML フォーム，あるいは JavaScript を介したフォーム送信等によって実装される．一方，LogicalPage にはサーバオブジェクトから取得 (図 3.1(4)) されたデータが表示されるが，各ページに表示されるべきデータ項目は PageData 要素によってモデル化される．

3.1.4 開発支援環境 WAST

WAST ワークベンチ [53, 67-69] は，WAD により記述されたモデル (WAD ファイル) に基づいてサーバページ，アクションのためのスケルトンや構成ファイルを自動生成する開発支援環境である．WAST ワークベンチは，サーバページやアクション等の成果物間で相互に参照される情報 (URI，フォームのパラメータ名など) の整合性をチェックする機能を提供する．WAST はオープンソースのツール統合プラットフォームである Eclipse[115] 上で実現され，MVC に基づく様々なランタイムエンジンに対してカスタマイズ可能なフレームワークとして実装されている．

図 3.5 に WAST ワークベンチの画面例を示す．ページフローエディタ (図 3.5(c)) ではアイコン表示された LogicalPage, ActionInvocation, Region を配置したり，マウス操作によりノード間の

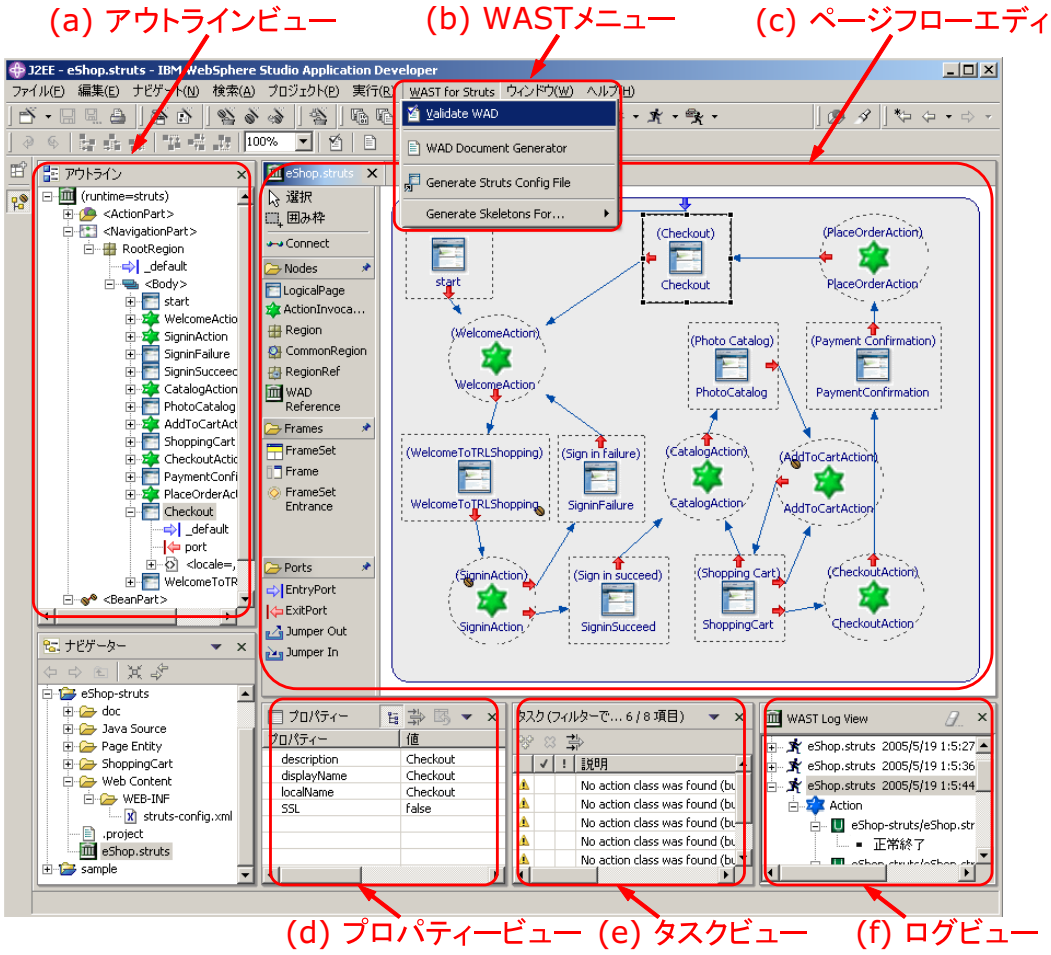


図 3.5: Struts 用 WAST ワークベンチ

接続関係を追加・削除することが出来る．図 3.5(a) のアウトラインビューでは，アプリケーションのモデルで用いられているすべての WadNode が列挙され，他のビュー，エディタとの連携により各要素の確認・変更が可能である．例えば，ページフローエディタで LogicalPage のノードを選択すると，該当するノードがアウトラインビューでも強調表示される．さらに，アウトラインビュー内のノードをマウスで選択することにより，そのノードに付与された属性がプロパティービュー (図 3.5(d)) に表示され，属性値を確認・修正することが出来る．

以上の様に，WAST 上で GUI を用いて WAD ファイルを編集した後に WAST メニュー (図 3.5(b)) からコマンドを実行することにより，WAD ファイルの妥当性確認，アプリケーションの実行に必要な各種スケルトンや構成ファイル等を自動生成することが出来る．ここで，実際に生成される成果物は WAST ワークベンチがどのランタイムエンジン向けにカスタマイズされているかによって異なり，例えば Apache Struts[46] をはじめとした様々なランタイムエンジンに対するコード生成が可能である．

妥当性確認メニューを選択すると，タスクビュー (図 3.5(e)) には，LogicalPage の論理名等必須

項目の入力漏れや，遷移のつなぎ忘れなどエラーや警告がタスク項目として表示される．モデル設計者は，このタスク項目を修正することで正しいモデルを設計することが可能となる．

ログビュー (図 3.5(f)) には，各種スケルトンや構成ファイルを自動生成する際のログが表示され，もし正常に生成されなかった場合には，ログを参考に原因を特定し修正を行う．

この様に，WAST 上で WAD ファイルの編集，妥当性確認を行い，その WAD ファイルを基にアプリケーションの実行に必要なページスケルトンやアクションスケルトンなどを自動生成することで，開発効率を向上すると同時に，ページ遷移の誤りやパラメータ名の齟齬等を防止し，開発成果物の質を向上することが可能となる．

3.2 動的解析に基づくモデル抽出支援手法

3.2.1 Web アプリケーションの再構築

3.1 節で述べた様に，モデルに基づく Web アプリケーションでは，Web アプリケーションの振舞い，すなわち Web ブラウザからのリクエスト処理，メソッド呼出しやオブジェクトの生成によるアプリケーションの状態やデータの変更，次ページへの遷移，表示用のデータ送信，という一連の処理の流れを実行環境に依存しない表現で定義し，設計段階での不整合の検出や，フレームワークの利用，並行開発の実現による作業の効率化を実現している．

しかしながら，既存の Web アプリケーションの中にはモデルに基づいて開発されていない物も多数存在する．これらのアプリケーションの再構築に際し，モデルを利用した開発へ移行することで，今後の開発効率および再利用性の向上が期待される．

これまで，Web アプリケーションの再構築支援手法として，業務ロジックやページデザイン等のリソースの再利用を支援するための研究がなされてきた [16, 22, 23]．例えば，文献 [22] では，Web アプリケーションの構成の把握を支援するために，リバースエンジニアリング手法を用いた視覚化手法が提案されている．この手法では，Web サーバにおけるアプリケーションの構成要素を，静的ページ (HTML 等)，動的ページ (ASP, JSP 等)，Web オブジェクト (CORBA, EJB 等)，データベースなどに分類し，各要素に対して解析プログラムを用意し解析を行う．次に，それぞれの解析結果を集約することにより構成要素間の依存関係を明らかにしている．

この様に，リバースエンジニアリング手法を用いることで，Web アプリケーション構成要素の依存関係の把握や，構成要素の再利用を支援することが可能となる [16, 22, 23]．しかしながら，Web アプリケーションは Web ブラウザから送信されたリクエストに基づいて構成要素を呼び出すことにより実行される．この様なリクエストを介した参照関係は，構成要素のみの解析からは把握することが難しく，従来のリバースエンジニアリング手法では Web アプリケーション全体の振舞いが明らかにされないという問題点が指摘されている [80]．

例えば，図 3.2 で示された例において，入力フォーム (図 3.2(a), (b)) (KeywordSearchPage.jsp) とサーブレットプログラム (図 3.2(c)) (KeywordSearch.java) を従来のリバースエンジニアリングを用いて解析した結果の一例を図 3.6 に示す．図 3.6 は UML のクラス図 [112, 113] である．図より，KeywordSearch.java が HTTPServlet のサブクラスでありいくつかのメソッドを持つこと，KeywordSearchPage.jsp が JSP であることは明らかになるが，KeywordSearchPage.jsp と KeywordSearch.java の間には，入力パラメータ (keyword) を介した参照関係があるにも関わらず，解析結果にその関係は含まれていない．

さらに，MVC に沿った設計を行うことで，各コンポーネントの役割が明確に区分され，再利用性の向上や並行開発が容易になる等の利点があるが，解析結果では Model と Controller が混在しておりコンポーネントに分割されているとは言い難い．CGI を用いたアプリケーションでは View, Model, 及び Controller がすべて混在している単一のプログラムコードが利用されることもあり，従来のリバースエンジニアリング手法のみではそのようなプログラムコードを MVC の各コンポー

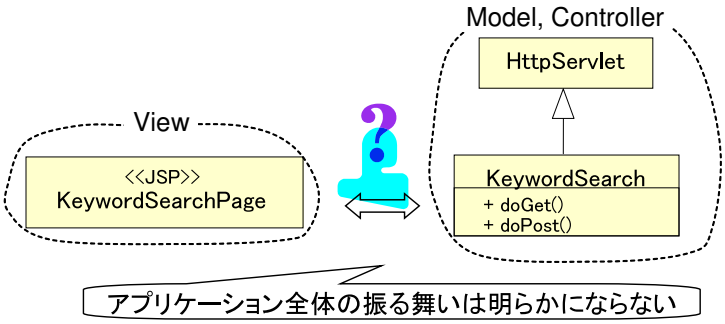


図 3.6: 従来のリバースエンジニアリング解析結果の例

ネントに分割することが出来ず，アプリケーション全体の振舞いが明らかにならないという問題も存在する．

この様に，従来のリバースエンジニアリング手法のみでは Web アプリケーション全体の振舞いが明らかにならないことの主な原因として，Web アプリケーションは一種のサーバ/クライアント型アプリケーションであること [81] が挙げられる．サーバ/クライアント型アプリケーションでは，サーバ側リソースだけでなくサーバ，クライアント間の通信内容やクライアント側のリソースを含めた動的解析を行わなければ，アプリケーション全体の振舞いが明らかとならないという問題点がある [82]．

3.2.2 モデル抽出支援システム

本論文では，既存 Web アプリケーションの実行時の振舞いを，HTTP 上で授受される情報に着目した動的解析により明らかにし，MVC に基づくモデルの抽出を支援する手法を提案する．本手法の対象とする Web アプリケーションは，業務ロジックがサーバ側で実行され，クライアントに相当する Web ブラウザ側ではフォームおよびハイパーリンクを用いたリクエスト送信が実行される様な Thin Web Client[83] 型の Web アプリケーションである．

提案手法では，まず，HTTP 上で授受される情報の中から実体ページ，すなわち HTML 文書をそれぞれの類似度等に基づいてグループに分類する [66]．次に，各グループ間の呼出し関係および HTTP リクエストとして送信される各種パラメータを解析し，Web アプリケーション全体の処理の流れを表すページフローを導出する．ここで得られたページフローにより，ページ間の遷移関係，サーバ側ロジックの呼出し関係，パラメータを介したインタフェース等，Web アプリケーション全体の振舞いが明らかとなる．提案手法では，ページフローと Web アプリケーションモデルのスキーマを照らし合わせることにより，MVC に基づくモデルを抽出している．

モデル抽出支援システムの構成を図 3.7 に示す．提案システムは Transaction Recorder, Analyzer, Model Generator，及び開発者がこれらのモジュールを操作し解析結果を確認/調整するための GUI から構成される．また，提案システムと Web アプリケーションの MVC との関連を表 3.1 に示す．

まず，Transaction Recorder は，Web アプリケーションを実行した際の HTTP リクエスト/レ

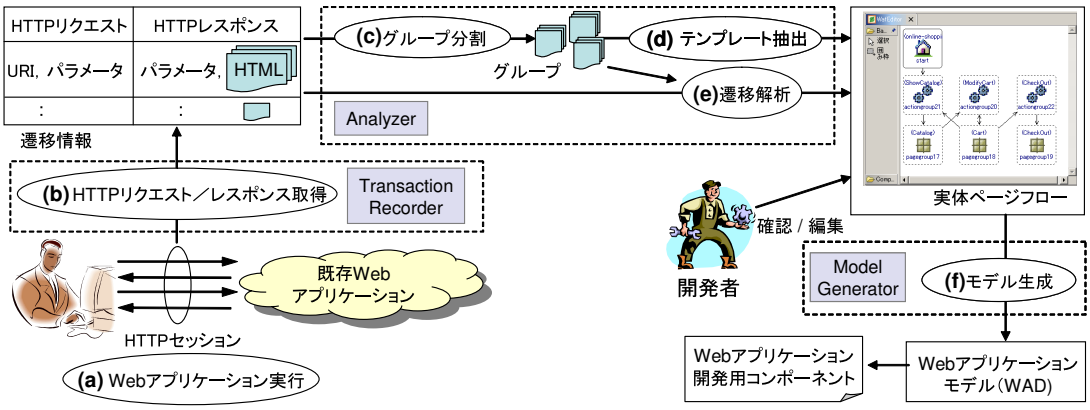


図 3.7: Web アプリケーションモデル抽出支援システムの構成

表 3.1: MVC と提案システムとの関連

MVC	提案システム (Analyzer)		抽出 Web アプリケーション	開発用コンポーネント
	解析モジュール	解析対象	モデル (WAD)	(Struts の場合)
Model	遷移解析	URI, パラメータ	サーバオブジェクト (BeanPart)	内部ロジック雛形 (EJB 等), FormBean
View	グループ分割, テンプレート抽出	HTML 文書, パラメータ	サーバページ (LogicalPage, PageData)	JSP 雛形, TagLib 宣言等
Cont- roller	遷移解析	URI, グループ 分割結果	ページフロー (NavigationPart), サーバロジック呼出し/分岐 (ActionPart)	構成ファイル (struts-config 等), Action クラス雛形

スポンスを，URI やクエリー引数，POST されたデータ等の各種パラメータと HTML 文書に区分し，遷移情報として保存する (図 3.7(a), (b))．ここで，アプリケーションを実行するのは，システム開発者でなく一般のアプリケーション利用者でも構わない．

Analyzer は，Transaction Recorder で取得した遷移情報を解析することにより，MVC の各要素を抽出する．具体的には，HTML 文書群を構造/役割の類似度に基づいてグループに分割し (図 3.7(c))，同一グループに分離されたページ群から表示形式の雛形や入出力データを抽出する (図 3.7(d))．この結果，MVC における View に相当するサーバページが抽出される．次に，遷移情報中の URI とグループ分割の結果に基づいて解析を行い，グループ間の呼出し関係を明らかにする (図 3.7(e))．遷移解析の結果，MVC における Model に相当するサーバオブジェクト，及び Controller に相当するページフローやサーバロジック呼出し/分岐が抽出される．

以上の解析結果に基づいて，Analyzer は実体ページフローを生成する．ここで実体ページフローとは提案システムで用いる中間表現で，HTML 文書とその呼出し関係を記述し MVC との対応関係を表すものであり，HTML 文書やサーバロジック呼出しをグループ分割した結果を保持する．開発者は実体ページフローエディタ上で，実体ページやサーバロジック呼出しの属性を確認した

り，ページフローの調整を行うことも可能である．ただし，開発者が初期グループ分割結果に変更を加えた場合は，該当する箇所に対しテンプレート抽出 (図 3.7(d)) 及び遷移解析 (図 3.7(e)) が再度実行される．

開発者による確認・調整が終了した後，本システムは実体ページフローと Web アプリケーションモデルのスキーマを照らし合わせ，モデルを抽出する (図 3.7(f))．サーバオブジェクトは 3.1.3 節で述べた WAD の BeanPart に，サーバページは LogicalPage や PageData に，ページフローとサーバロジック呼出しはそれぞれ NavigationPart と ActionPart に相当する．抽出した Web アプリケーションモデルを用いることで，様々な開発用コンポーネントを生成することが出来る．例えば，Struts[46] を対象フレームワークとした場合の開発用コンポーネントの例を表 3.1 の最右列に示す．ここで，MVC における Model に相当する内部ロジック (EJB 等) に関しては，従来のリバースエンジニアリング手法 [22] を用いた解析結果を再利用することも可能である．

3.2.3 Analyzer における解析

本節では，HTTP リクエスト/レスポンスを記録した遷移情報から，HTML 文書のグループ分割，グループからのテンプレート抽出，及び遷移解析を行う Analyzer の詳細について述べる．

グループ分割

グループ分割モジュール (図 3.7(c)) では，HTML 文書間のページ構造/役割に関する類似度に基づいて HTML 文書群をグループに分割し，MVC における View に相当するサーバページの抽出を実現する．

本論文では，HTML 文書の類似度を導出するため，TABLE，TD，DIV などのレイアウトに影響を与えるタグ (以降，レイアウトタグ) の構造を比較し，コンテンツ間の距離を数値化する手法 [116] を利用した．ただし，文献 [116] の手法はレイアウト，すなわちページ構造が同一か否かを判定することに着目していたため，本論文ではページ構造及び役割に関する類似度を導出可能とするための拡張を行った．以下では，HTML 文書の類似度導出手法の概要，及び本論文で行った拡張について説明を行う．

HTML 文書の構造に関する類似度の導出では，まず，HTML 文書からレイアウトタグのみを抽出した XML 文書を作成する．この際，各タグの属性値，当該タグのサブツリー内のエレメントに関する情報等を特徴値として関連付ける (図 3.8)．次に，本論文ではコンテンツの類似度をより詳細に導出するため，差分計算 [98, 99] によりレイアウトタグの対応関係を明らかにした (図 3.9)．最後に，これらの対応関係を用いて文書 (A), (B) 間の距離 D を次の式により導出する．

$$D = \sum_{i=0}^{O} \Big\{ W_c(X_i) * \left(m(R_i) + \sum_{j=0}^{K} f(A_{ij}, B_{ij})\right) \Big\} \tag{3.1}$$

式 (3.1) 中，O は 2 文書間のタグの対応関係の総数である．また，i 番目の対応関係におけるタ

グを一意に指示する XPath[75] 表現の組を X_i，対応関係の種類を R_i としている．さらに，タグ i に関連付けられる特徴値の総数を K，文書 (A), (B) 中でのタグ i における j 番目の特徴値をそれぞれ A_{ij}, B_{ij} とする．

また，式中の W_c はタグの種類及び文書内での位置 X_i に基づく重み付けで，タグツリーの上位に位置するほど大きな値をとる．次に，$m(R_i)$ は文書間でのタグの対応関係に関する距離を返す関数で，次の条件を満たすものとする．

$$\begin{array}{rcl} & m(R_i) = 1 & (\mathrm{insert}, \mathrm{remove}) \\ 0 < & m(R_i) < 1 & (\mathrm{move}) \\ & m(R_i) = 0 & (\mathrm{keep}) \end{array} \tag{3.2}$$

即ち，$m(R_i)$ は対象となるタグが一方の文書にしか存在しない場合には 1，双方の文書で同一の位置に存在する場合には 0，移動している場合にはその移動量に応じて 0 から 1 の間の値を返す関数である．

最後に，式 (3.1) 中の $f(A_{ij}, B_{ij})$ は文書 (A), (B) 間の特徴値の差異に応じた距離を与える関数で，特徴値の差が大きい場合ほど大きな値を返す．また，$f(A_{ij}, B_{ij})$ は次の式に示すように，その総和が 1 を超えないよう設定する．

$$0 \leq \sum_{j=0}^{K} f(A_{ij}, B_{ij}) \leq 1 \tag{3.3}$$

ただし，対象となるタグが一方の文書にしか存在しない場合には，タグに関連付けられた特徴値のうち，当該タグの属性値に関する距離はすべて最大値を取るものとし，当該タグのサブツリー内のエレメントに関する特徴値に関しては，比較対照が空のサブツリーであるものと仮定して距離を導出している．

以上の式を用いることにより，HTML 文書の構造に関する類似度を数値として導出することが可能となる．

次に，本論文ではフォームの送信方法や送信先 URI 等のパラメータに着目して HTML 文書の役割に関する類似度を求める手法を導入した．本手法では，HTML 文書から FORM タグ及びその構成要素である INPUT, SELECT 等を抽出し，それぞれの属性やタグ内の文字列などを特徴値として関連付け，文書間で比較することにより類似度を求める．ただし，役割に関する類似度においてはタグの出現順序など，HTML 記述上での構造の相違に関しては無視するものとした．また，HTML 文書内に含まれるリンク先 URI を役割に加味することも可能である．

以上の手法により導出された HTML 文書の構造及び役割に関する類似度を用いることで，HTML 文書のグループ分割が可能になる．例えば，類似度に閾値を設けることなどにより自動分割を行うことも可能であるが，対象となる Web アプリケーションの特徴により，適切な閾値の値や，構造・役割の距離に対する重み付けは変動する．このため，最適な分割結果を自動で得ることは難しい．そこで，分割結果を GUI を用いて開発者に提示し，閾値や重み付けの調整および結果の確認を行うことで，グループ分割の精度を高めることとした．GUI を利用した確認・調整ツールの詳細に関しては 3.2.4 節で述べる．

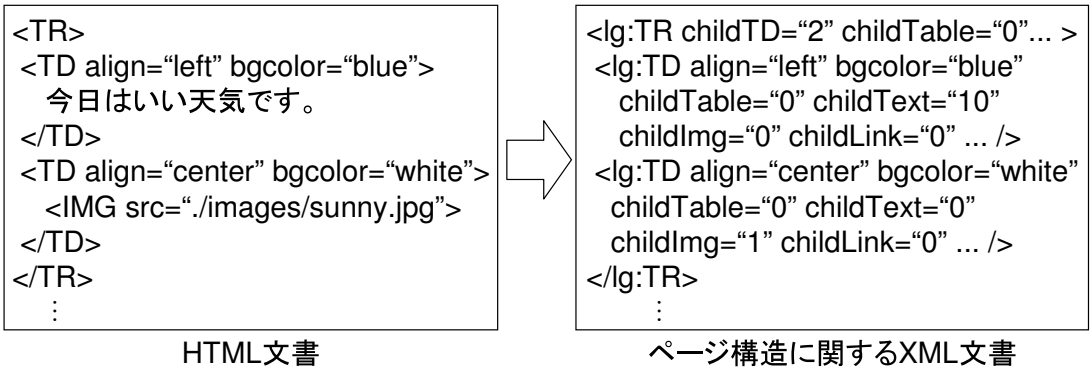


図 3.8: レイアウトタグの抽出

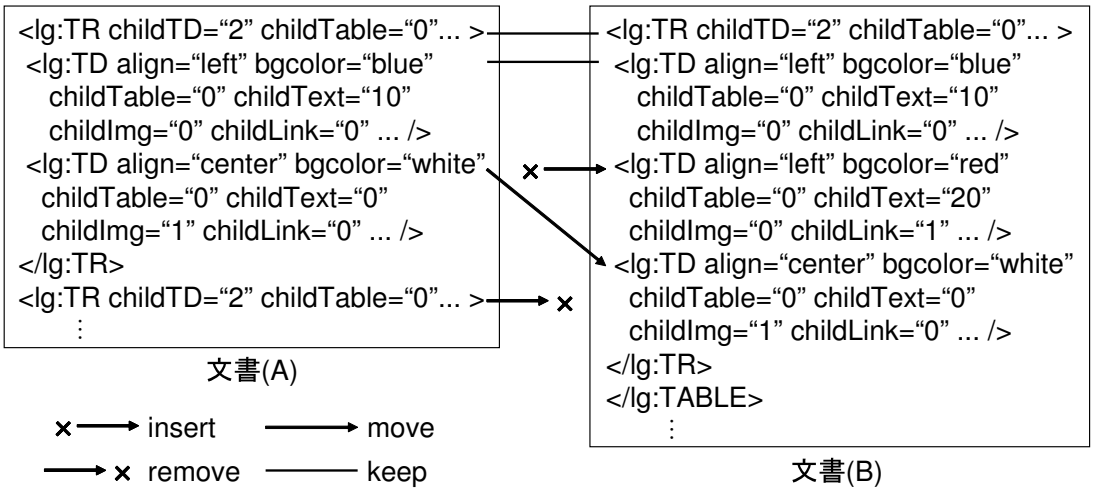


図 3.9: レイアウトタグの対応関係の導出

テンプレート抽出

テンプレート抽出モジュール (図 3.7(d)) では，3.2.3 節の手法を用いて分割された HTML 文書のグループにおいて，ページ全体のテンプレート，特定表示形式の繰り返し，及び状況に応じて変更される部分を特定することで，MVC における View に相当するサーバページの再構築や，View と Model 間の呼び出し関係の抽出を支援する．

まず，グループ化された HTML 文書群において，グループ内で共通に利用されている固定部分を差分演算 [98, 99] により抽出する．図 3.10(b) はグループ内 (図 3.10(a)) で共通に利用されていた部分を HTML のテンプレートとし，状況により変化が生じる部分を斜線を用いて表示したページテンプレートの例である．

次に，状況により変更が発生した部分 (図 3.10(c), (d)) に対し，コンテンツ抽象化手法 [34] を用いて特定書式の繰り返しを検出し，その書式を抽出する．ここで，発見された繰り返し表現の書式とその表示例を開発者に提示し，編集・確認を行うことで表示形式のテンプレートとして用

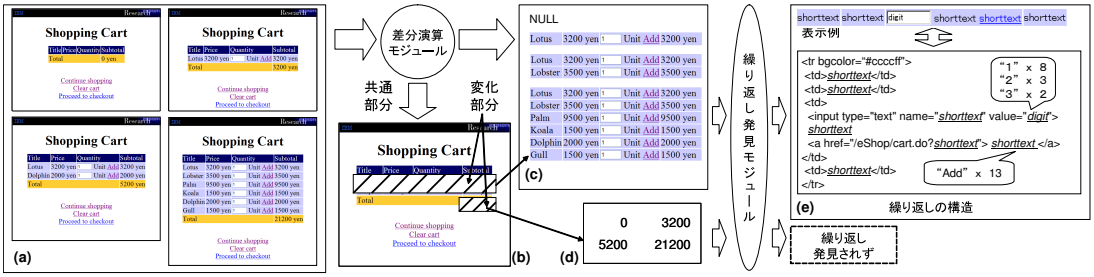


図 3.10: 差分演算及びコンテンツ抽象化を用いたページテンプレートの抽出

いることが出来る (図 3.10(e))．また，書式中で抽象化されている部分 (図 3.10(e) 下線部) に対し，実際に用いられた値 (図 3.10(e) 吹き出し部分) を開発者に提示することで，書式の修正や再構成を支援する．

最後に，状況により変更が発生するが繰り返しには属さない部分 (図 3.10(d)) は，アプリケーションの利用状況や時刻等に応じて変更される部分 (Welcome メッセージや買い物かごの合計金額など) と判断出来る．

以上の分類を行うことにより，サーバページの再構築に際し，既存 Web アプリケーションで利用されているページの書式を再利用することが可能となる．

遷移解析

遷移解析モジュール (図 3.7(e)) では，Transaction Recorder が取得した遷移情報と 3.2.3 節で得られたグループ分割の結果から，MVC における Model に相当するサーバオブジェクト，及び Controller に相当するサーバロジック呼出しやその入力変数，サーバロジック分岐を導出する．

まず，ある 1 つのグループから呼び出される URI 群から，共通の URI を抜き出し，その集合を 1 つのサーバロジック呼出しと考える．この際，遷移情報内のパラメータから得られるクエリーの引数や POST データ等は，サーバロジック呼出しの入力変数，及びサーバオブジェクトの保持する変数として保存する．

さらに，既存アプリケーション固有の知識を用いて URI を解析することにより，サーバロジック呼出しを詳細に推定する．例えば，URI のパスが “.html” で終端する場合は静的リンクとし，“.cgi” で終わる場合は CGI のスクリプト呼出しと推定し，サーバロジック呼出しを割り当てる．

また，既存の Web アプリケーションにおいては，サーバロジックがモジュール別に分かれておらず，複数のロジックが単一 URI パスで表現されている場合も存在する．その様な場合には，クエリーの引数や POST データを利用してサーバロジック呼出しを分割する．例えば，
URI “`foo.cgi?userinfo=guest&opt=email`” と “`foo.cgi?item=book1&count=2`” において，URI パス (`foo.cgi`) は共通であるが，クエリーの引数名の相違 (`userinfo`，`opt` と `item`，`count`) から異なるサーバロジック呼出しと判断する．この際，解析に利用するパラメータを，正規表現等を用いて取捨選択することも可能である．

図 3.11: Web アプリケーションモデル抽出ツール

Web アプリケーションにおいて，サーバロジックを実行した結果，遷移先のグループが複数存在することがある．例えば，あるロジックを実行した結果，処理続行を促すグループに遷移する場合と，エラーを表示するグループに遷移する場合である．この様な場合は, 遷移先のグループ数に応じてサーバロジック分岐数を割り当てる．

以上の解析によりグループ間の呼出し関係が抽出され，Web アプリケーションの振る舞いである実体ページフローが明らかとなる．

3.2.4 モデル抽出支援ツール

我々は，提案システムを統合開発環境である Eclipse[115] 上にモデル抽出支援ツールとして実装した．このツールを用いることにより，モデル抽出やモデル抽出後の編集・開発を統一された環境で実施することが可能となっている (図 3.11)．

ツールを利用する際には，まず，ブラウザビュー (図 3.11(a)) を用いて Web アプリケーションを実行する．実行内容は Transaction Recorder により遷移情報として記録され，ナビゲーションビューに表示される (図 3.11(c))．Web アプリケーションの実行が終了した時点で, 解析実行を指

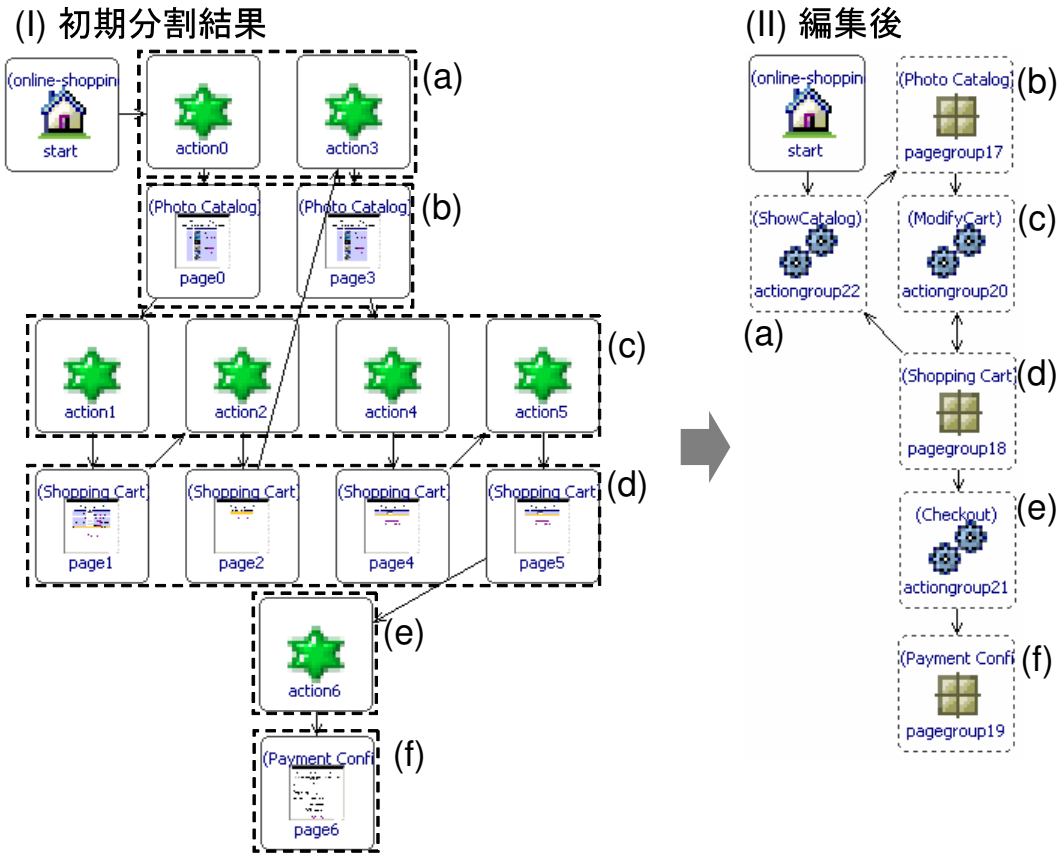


図 3.12: 実体ページフローの編集

示すると，Analyzer によるモデル抽出が実行され，生成された実体ページフローがエディタ上に表示される (図 3.11(b))．また，エディタ左側のアウトラインビュー (図 3.11(d)) には，実体ページフローの概要が表示される．アウトラインビューを用いることで，各グループに含まれる実体ページ群やロジック呼出しなどがツリー表示により確認出来る．

また，本ツールではグループコントローラ (図 3.11(f)) を用いて 3.2.3 節のグループ分割における閾値や，構造/役割間の重み付けの変更を行いながら，実体ページフローを確認・調整することが可能である．この際，グループ分割の結果に影響を及ぼす閾値や重み付けの境界を予め計算し，グループ分割のパターンを開発者に提示することが可能である．

実際の実体ページフロー編集の様子を図 3.12 に示す．図中の左上端にあるアイコンはアプリケーションの開始点を示している．図 3.12(I) は，初期設定値に基づいて実体ページフローを生成した結果で，実体ページ (ページ画像のアイコン) やロジック呼出し (星型のアイコン) がグループ毎に横一列にレイアウトされ，それぞれの呼び出し関係が矢印で表されている．図中，(b, d, f) は実体ページ，(a, c, e) はロジック呼出しのグループの初期分割結果を示している．

次に，エディタ上のアイコンを選択すると論理名や座標などの属性がエディタ左下のプロパティビュー (図 3.11(e)) に表示される．また，実体ページを表すアイコン中には，HTML の TITLE タ

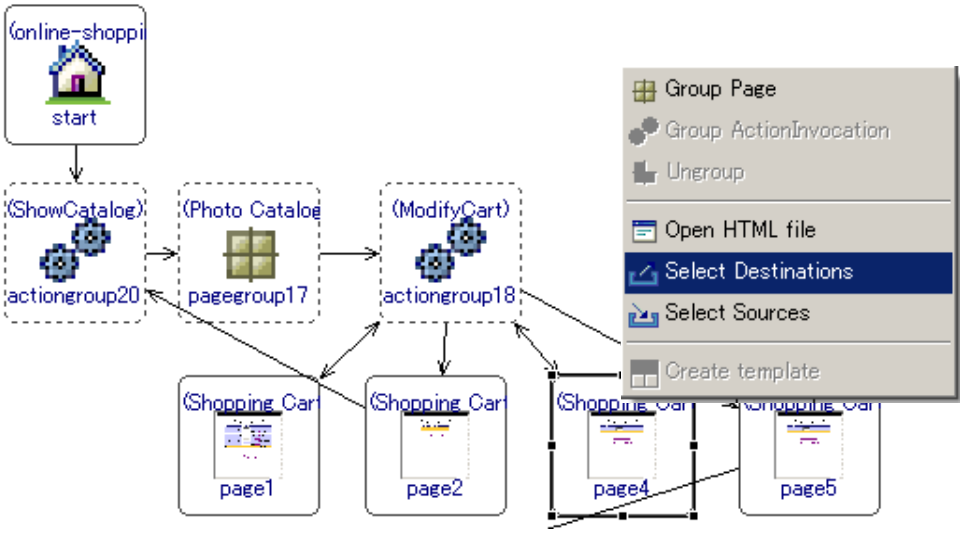


図 3.13: グループ化支援のためのメニュー

グから抽出された名前や，各ページの画面イメージが表示され，グループを調整する際の指針に用いることが出来る．さらに，アイコンに対応する HTML 文書を実際に開いて確認することも出来る．

また，本ツールでは Analyzer によるグループ分割結果を詳細に調整するため，各要素の遷移元や遷移先を選択するメニューが用意されている (図 3.13)．例えば，異なる認証処理へのロジック呼出しを行う類似したページが多数存在する場合，遷移先のロジック呼出しが一般ユーザ認証用か管理者認証用か等に応じて，さらに詳細なグループ分割を行うことが出来る．

エディタ上での確認・調整が終了したグループに対し，グループ化メニューを選択することで，グループが決定され，実体ページは四角のアイコン，ロジック呼出しは歯車のアイコンに置き換えられる．閾値や構造/役割間の重み付けの変更によるグループ分割結果の調整は，グループが決定される前の実体ページ及びロジック呼出しのみに適用される．

以上の作業を経て，グループ化された実体ページ (四角のアイコン) 及びロジック呼出し (歯車のアイコン) により実体ページフローが表現される (図 3.12(II))．図 3.12 では，初期分割結果と編集後のグループの対応関係をアルファベットで示している．

最後に，実体ページフローを入力として Model Generator(図 3.7(f)) を起動する．現在，Model Generator はスキーマとして WAD[50] もしくは UX model[83] のいずれかを選択し，Web アプリケーションモデルを抽出出来るよう実装されている．例えば，グループ化された実体ページは，WAD における LogicalPage に対応するという様に，実体ページフローと Web アプリケーションモデルのスキーマを照らし合わせることにより，Web アプリケーションモデルを抽出している．この様にして抽出されたモデルから Action クラスの雛形など開発用コンポーネントを生成することが可能となる [53].

3.3 Web アプリケーション移行支援に対する評価実験

本節では，提案手法を利用することにより Web アプリケーションの振舞いを表すモデルの抽出が可能となるか，実験を通して検証を行う．実験では，仕様書などの情報は一切用いず，Web アプリケーションの実行結果のみに基づいてモデル抽出を行った．また，比較の対象として，提案手法を用いずにモデルエディタ上 [53] で直接モデルを作成する手法に関しても実験を行った．抽出されたモデルの検証にあたっては，Web アプリケーションの要件定義のうち本手法により抽出可能であり，モデル上での記述が可能である項目を対象にその充足可否に基づいて評価を行った．

以降，3.3.1 節では実験対象となる Web アプリケーションの概要を述べ，3.3.2 節で実験方法の詳細について説明を行う．最後に，3.3.3 節で実験の結果について述べ，考察を行う．

3.3.1 対象アプリケーションと要件定義

実験では，対象アプリケーションとして，CGI を実行環境とする “オンラインショップ”，“掲示板” 及び “アンケート” の 3 種類を用いた．これらのアプリケーションには，ログオン認証，データの参照・検索・作成・更新・削除など，Web アプリケーションの基本的な機能が含まれている．また，フォームの基本的な機能をすべて利用している．

表 3.2 に，オンラインショップの要件定義の一部を示す[1]．この要件定義は，実際の開発に用いられている要件定義の一例に則して作成したものであり，画面要件，機能要件，データ要件に分類される．画面要件は MVC の View, Controller に，機能要件，データ要件は Model に相当する．また，各要件には鍵括弧で示される識別名が付加され，従属項目として要件の内容が与えられる．さらに，一部の要件には必要に応じて説明が加えられている．また，画面要件に対しては，遷移先画面の識別名が括弧内に記されている．

次に，各アプリケーションの画面要件，機能要件，データ要件の数を表 3.3 に示す．表では，各要件の従属項目数の合計を括弧内に示している．例えば，オンラインショップの画面要件は，7 個 (即ち 7 画面で構成される) であり, 各画面に対する従属項目の合計が 26 個である．さらに，これらの要件定義のうち，提案手法により抽出可能であり，モデル上での記述や充足可否の判断が可能である要件数を表 3.3 の鍵括弧内に，その合計を表の最右列に示す．このように評価の対象となる要件を絞った理由を以下に述べる．

まず，機能要件に関しては，内部ロジックの実装が完了した後でなければ要件を満たすか否か判断出来ない．そこで，本実験では，入力変数の有無等，ロジックの実装に必要な条件を満たすかどうかに基づいて評価を行うものとした．また，データ要件に関しては，システム内で生成可能なもの (日時や識別番号など) や，データベースに格納されている項目に関しては評価の対象から除き，HTTP 上で授受されるデータを対象にした．また，HTTP 内のパラメータとして明示的にリクエストが送信されない機能要件に関しても対象から除外した．これらの条件により，評価の対象外とした従属項目を $*$ で示した．

[1] オンラインショップ，掲示板及びアンケートの要件定義はすべて付録に記載しておく

また，画面要件中の可変部分，例えばログオンメッセージなどの生成に関連する項目 (表 3.2 中 ◇) は，3.2.3 節のテンプレート抽出を用いることにより，実体ページ群から推定することが可能であるが，本実験では充足可否が明確に判断出来る項目に重点を置き，これらの項目も評価の対象外とした．

表 3.2: オンラインショップの要件定義例

種別	要件	従属項目	説明
画面要件	[A01] ログオン (A02, A03)	◇ メッセージ	ログオン時のメッセージを表示
		ユーザ名入力フィールド	
		パスワード入力フィールド	
		送信ボタン	
	[A02] ログオン失敗 (A01)	◇ メッセージ	ログオン失敗時のメッセージを表示
		ログオン画面へのリンク	
	[A03]Welcome(A04)	◇ メッセージ	ログオン成功時のメッセージを表示
		◇ ユーザ名	
		製品一覧画面へのリンク	
機能要件	[B01] ログオン認証	認証処理	
	[B02] 製品情報一覧取得	∗ 製品名称取得	製品一覧情報より製品項目リストを取得
		∗ 製品画像取得	
		∗ 製品単価取得	
	[B03] カート内容照会	◇ 製品名称取得	カート内容に含まれる製品項目より取得
		◇ 製品単価取得	
		◇ 製品個数取得	
		◇ 小計計算	小計は単価と個数を製品毎に計算
		◇ 合計金額計算	小計の総和を計算
		カート ID 取得	
データ要件	[C01] 製品項目	製品 ID	
		◇ 単価	
		◇ 画像	
	[C02] 製品一覧	∗ 製品項目リスト	製品項目がリストとして登録されている
	[C03] ユーザ情報	ユーザ名	
		パスワード	
		◇ 配送先	

3.3.2 実験方法

本実験では開発者として，Struts [46] を用いたアプリケーションの実装経験がない人 (開発者 a, d), Struts を用いたアプリケーションの実装経験を持つ人 (b, c, e, f) の合計 6 人の協力を得た．実験前の準備として，各開発者に簡単な Web アプリケーションからのモデル抽出を実施させ，提案手法を実装した抽出支援ツールの基本的な使い方などを習得させた．

実験では，開発者を 2 グループに分け，提案手法を利用する場合と利用しない場合のそれぞれについてモデル抽出を実行させた (表 3.4)．この際，実験結果がアプリケーションに対する知識に依存しないよう，3.3.1 節で述べた 3 種類の対象アプリケーションをオンラインショップ，掲示板，

表 3.3: 実験対象アプリケーションの概要

	画面要件	機能要件	データ要件	対象要件合計
オンラインショップ	7(26) [7(14)]	6(14) [5(5)]	3(7) [2(3)]	14(22)
掲示板	9(45) [9(33)]	7(12) [6(8)]	3(10) [3(7)]	18(48)
アンケート	7(27) [7(21)]	3(4) [2(3)]	3(9) [2(8)]	11(32)

表 3.4: 実験内容と提案手法利用の有無

開発者	実験内容		
	1 回目 (オンラインショップ)	2 回目 (掲示板)	3 回目 (アンケート)
a, b, c	提案手法なし (S_N)	利用 (B_T)	なし (Q_N)
d, e, f	提案手法利用 (S_T)	なし (B_N)	利用 (Q_T)

アンケートの順で一度ずつ用いた．

提案手法の利用に当たっては，グループ分割の初期値として，経験的に良い分割結果の得られる構造/役割間の重み付け (1：1) と閾値 (距離の最大値の 30%) を用いて開発者に実体ページフローを提示した．開発者は，必要に応じて初期値から分割パターンを調整した後グループを決定し，モデル抽出を行った．

3.3.3 実験結果及び考察

本実験で各開発者が抽出したモデルの概要と，要件定義との関係を表 3.5 に示す．また，抽出されたモデルの例として，オンラインショップ (S_T-f)，掲示板 (B_T-c)，アンケート (Q_T-d) の WAD ファイルを WAST のページフローエディタで表示した様子を図 3.14，図 3.15，図 3.16 にそれぞれ示す．図中，四角いアイコンはグループ化された実体ページから導出された LogicalPage を，丸いアイコンはグループ化されたロジック呼出しから導出された ActionInvocation を示している．また，各サーバページの画面要件名を吹き出しとして付加してある．

表 3.5 の左列は，生成されたモデル中のページ数及びロジック呼出し数を示している．次に，各モデルが満たしている要件の数を表 3.5 の要件定義充足数に示す．ここで，括弧内は従属項目数を表しており，従属項目が 1 つでも満たされない要件は，要件を満たさないものとした．最後に，表 3.5 の右列には各開発者が要件を満たすことの出来なかった原因を挙げた．本実験でモデルが要件を満たせなかった原因は 2 種類存在し，HTML のリンクやボタン等をたどり忘れた場合 (実行忘れ) と，実行はしたもののモデルとして記述し忘れたり，呼出し関係の記述を間違えた等，モデルに間違いがあった場合 (モデル間違い) であった．

表 3.5 の結果から，提案手法を利用しない場合に，モデル間違いが多く発生しているのに対し，提案手法を用いることでモデル間違いを防止出来ていることが分かる．例えば，オンラインショッ

表 3.5: 作成されたモデルと要件定義との関係

実験	開発者	ページ数	ロジック呼出し数	要件定義充足数	誤りの原因(従属項目)	
					実行忘れ	モデル間違い
オンライン	a	7	2	12(19)	0	3
ショップ	b	7	6	14(22)	0	0
(S_N)	c	7	6	14(22)	0	0
オンライン	d	7	8	14(22)	0	0
ショップ	e	7	6	14(22)	0	0
(S_T)	f	7	6	14(22)	0	0
掲示板	d	7	9	13(42)	1	5
(B_N)	e	6	8	13(35)	10	3
	f	8	9	17(47)	0	1
掲示板	a	8	13	14(44)	4	0
(B_T)	b	9	10	18(48)	0	0
	c	9	12	18(48)	0	0
アンケート	a	7	2	7(28)	0	4
(Q_N)	b	8	5	6(27)	0	5
	c	7	4	7(28)	0	5
アンケート	d	7	5	11(32)	0	0
(Q_T)	e	7	5	11(32)	0	0
	f	7	5	10(31)	1	0

プ (S_N, S_T) では，提案手法を利用しない S_N-a にモデル間違いが 3 箇所存在した．実験中，開発者 a はアプリケーションをくまなく実行しているにも関わらず，相当するモデル構成要素を作成していなかった．一方，提案手法を利用した場合には，実行したページやロジック呼出しは必ず実体ページフロー上に出現するため，この様なモデル記述の抜け落ちを防ぐことが出来る．

また，提案手法を利用せず掲示板，アンケートのモデル抽出を行った場合，全員が何らかの形でモデル間違いを発生している (B_N, Q_N)．この様なモデル間違いの原因としては，提案手法を用いない場合には，受け渡されるパラメータの見落としが発生することや，実体ページやロジック呼出しの役割を把握することが困難であることなどが挙げられる．

特に掲示板は，Web アプリケーションの全ページで同一のタイトル “掲示板” が用いられており，HTML のタイトルから要件を推測することが困難であった．さらに，ページのデザインも見た目では区別が付きにくく，ロジック呼出しはすべて同一 URI パスを用いた上でクエリーのみが異なるものであった．このため，開発者がアプリケーションを実行しながらモデルを記述している間に，複数のサーバページやロジックを混同してしまい，結果として誤ったモデルを生成していた．

一方，提案手法を利用した場合 (B_T) においても，ページのデザインや呼出し URI が類似して

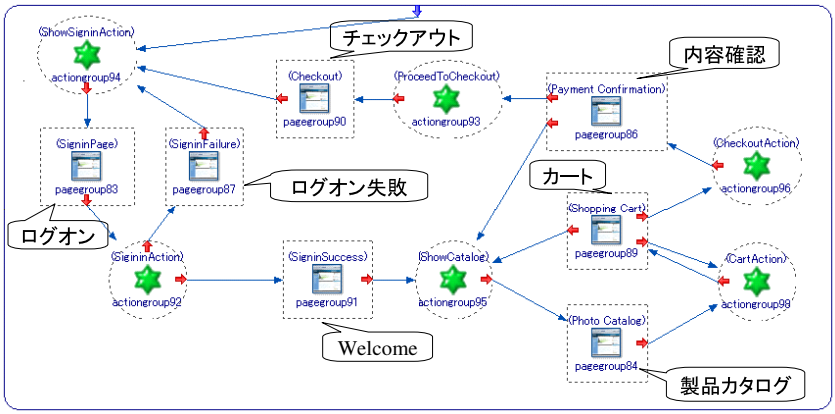


図 3.14: オンラインショップ (S_T-f)

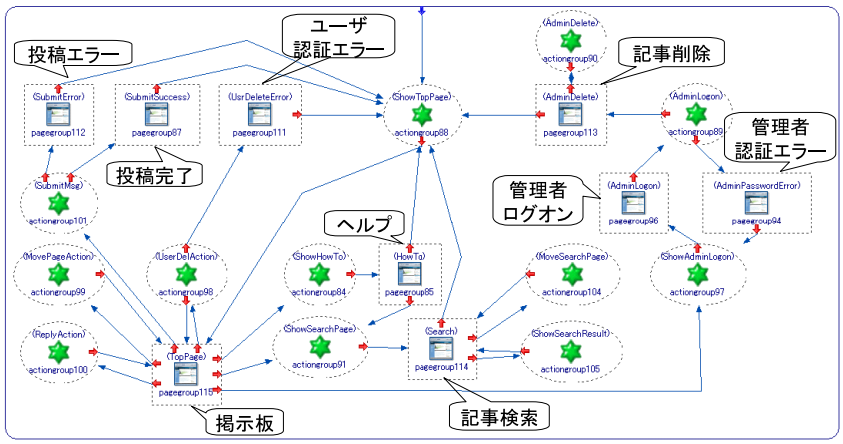


図 3.15: 掲示板 (B_T-c)

いるため，初期分割結果では複数のサーバページやロジックが混同されて表示される．しかしながら，実体ページフローエディタにおいて，グループ分割における閾値変更機能や，遷移元/先確認機能を用いることにより，各要素の詳細な役割や振舞いを把握し，実体ページフローを調整することが可能である．実際，開発者 a, b, c はこれらの機能を利用して実体ページフローの調整を行い，結果として多くの要件を満たすモデルを抽出している (表 3.5).

以上の様に，提案手法を用いることで Web アプリケーションの振舞いの把握を支援し，正確なモデルの抽出を可能としていることが分かる．この結果，提案手法を用いた場合に充足可能な要件 (従属項目) 数の平均値は，提案手法を用いない場合と比較して高い値となっている (図 3.17).

ただし，本実験では提案手法の利用の有無を問わず複数の開発者に実行忘れが見られた (表 3.5). 実行忘れが発生すると，対応するサーバページやロジックに関する情報が欠落してしまうため，要件定義の充足に大きな問題を引き起こしてしまう．実行忘れの多くは，ページ内に判別しづらいリンクが存在する，ページ内に多数のリンクが存在する，フォーム内の選択肢の組み合わせが多岐に

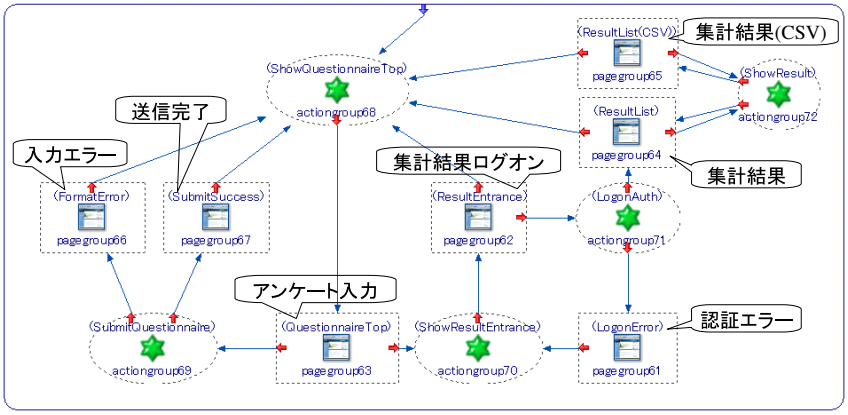


図 3.16: アンケート (Q_T-d)

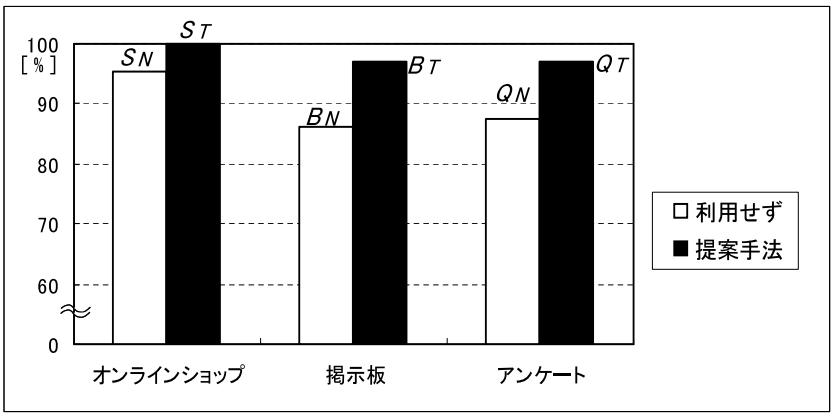


図 3.17: 要件定義充足率 (従属項目)

渡るなどの原因で発生する．例えば，掲示板において開発者 a, d, e に共通した実行忘れは，図 3.15 中のヘルプページから記事検索ページへのリンクのたどり忘れであり，その原因としては，このリンクがページ上ですぐにはリンクと判別しづらいものであったことが挙げられる．この様な実行忘れを防止するためには，Web アプリケーション実行中に HTML 文書の解析を行い，まだ実行していない URI に対し警告を表示したり，自動実行を行うなどの方法が有効であると考えられる．

最後に，実験所要時間の平均を図 3.18 に示す．提案手法を利用した場合については，作業内容をアプリケーション実行とモデルの確認/調整の 2 項目に分けて結果を示している．図から，提案手法を用いることにより，3 種類のアプリケーションいずれにおいてもモデル抽出にかかる所要時間が短縮されることが分かる．特に，遷移元/遷移先などの情報を用いた調整を必要としないオンラインショップでは，所要時間が約 40%短縮されている．また，詳細な調整作業が必要となる掲示板，アンケートにおいても，作業時間は約 20%以上短縮された．

以上の結果から，提案手法を利用することで，Web アプリケーションの振舞いの把握を支援し，より多くの要件定義を満たすモデルを，短時間で抽出することが可能となることが明らかとなった．

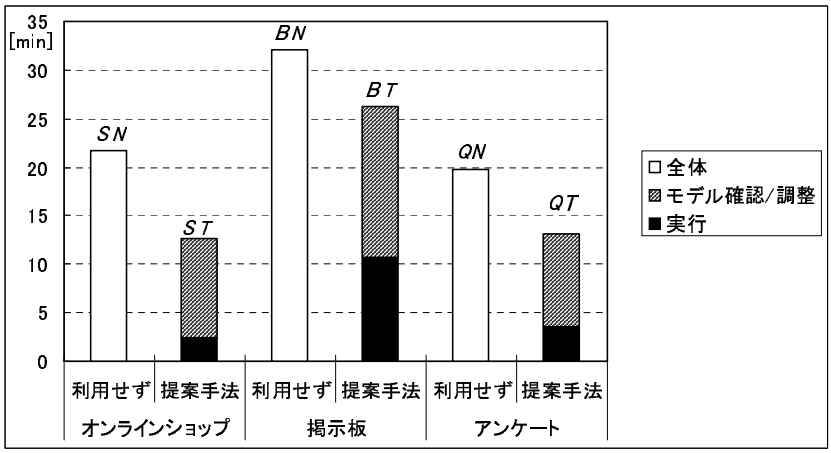


図 3.18: モデル作成の所要時間

今後の課題としては，大規模なアプリケーションでは，例えばログオン等の特定シナリオに限定してモデル抽出を行うなど，実行単位を区分して扱う仕組みを実装する必要があると考えられる．この様な区分は，実体ページフロー抽出の精度を向上させる上でも有効である．さらに，アプリケーション実行時の画面操作を再現する機能 [117] なども，ページフローの調整に有用であると考えられる．また，提案手法と従来のリバースエンジニアリング手法とをより緊密に連携させることにより，さらなる再利用性の向上が期待される．今後，検討を行っていきたい．

3.4 結論

本章では，まず Web アプリケーション開発を効率良く行うための，モデルに基づく Web アプリケーション開発について述べた．ここでは，実行環境に非依存のアプリケーションモデル記述言語 WAD 及び開発支援環境 WAST を提案した．WAST 上では GUI を用いて WAD ファイルの編集を行うことが可能であり，また，WAD の妥当性確認を行うことが可能である．さらに，WAST は作成した WAD ファイルの内容に基づき，Web アプリケーションの実装に利用されるページスケルトンやアクションスケルトン，各種設定ファイル等を自動生成することも可能である．これらの機能を利用することで，開発効率を向上すると同時に，ページ遷移の誤りやパラメータ名の齟齬等を防止し成果物の質を向上することが可能となる．

次に，Web アプリケーション再構築のための，動的解析に基づくモデル抽出支援手法を提案した．提案手法では，既存の Web アプリケーションの振舞いを，HTTP 上で授受される情報に着目した動的解析により明らかにし，モデルの抽出の支援を行う．提案手法を用いることで，ページ間の遷移関係やサーバ側ロジックの呼出関係，パラメータを介したインタフェース等，Web アプリケーション全体の振舞いが明らかとなり，MVC に基づくモデルを抽出することが可能となる．

さらに，提案手法を実装したツールを用いて，実際の Web アプリケーションを対象としたモデル抽出の実験を行った．実験では，仕様書などの情報は一切用いず，Web アプリケーションの実行結果のみに基づいてモデル抽出を行った．実験の結果，提案手法を用いることで，より多くの要件定義を満たすモデルを効率良く抽出可能となることが明らかとなった．

本章で提案した WAST 及びモデル抽出支援ツールは，同一の統合開発環境上に実装されており，互いに連携して動作することが可能である．これらのツールを用いることにより，既存の Web アプリケーションの資源を効果的に利用し，モデルに基づく Web アプリケーション開発へと円滑に移行することが可能となる．

第4章　まとめ

本論文では，効率の良い Web 開発のための既存資源の再利用技術の提案を行い，実際の Web 開発における経験，及び実験に基づいて提案手法の有用性を示した．

まず，第 2 章では Web コンテンツに対し外部アノテーションを付与するための枠組みを提案し，その実装としてアノテーションエディタを開発した．アノテーションエディタは，アノテーション語彙に対して拡張可能な編集環境であり，アノテーション対象ノードを指示するための XPath 表現生成を支援する機能を有する．アノテーションエディタを用いることで，目的毎に様々な語彙を持つアノテーションを統一された環境で効率よく生成することが可能となる

また，実際の Web コンテンツに対する XPath 表現の頑健性について評価実験を行い，XPath 表現の生成・編集とその利用における留意点について分析を行った．さらに，アノテーションエディタの適用例として，多様な携帯情報端末による Web 閲覧を支援するための Web トランスコーディングと，ポータルサイト構築のための Web クリッピングについて述べ，その有用性を示した．

第 3 章では，Web をアプリケーションのユーザインタフェースとして用いる Web アプリケーションを対象に，MVC に基づくモデル駆動型開発手法を提案した．さらに，既存 Web アプリケーションの実行時の振舞いを，HTTP 上で授受される情報に着目した動的解析により明らかにし，MVC に基づくモデルの抽出を支援する手法を提案した．本研究では，提案システムを統合開発環境上で動作するモデル抽出支援ツールとして実装した．このツールを用いることにより，モデル抽出からモデル抽出後の開発に至るまで，統一された環境で実施することが可能となった．さらに，モデル抽出支援ツールを用いた実験により，実際の Web アプリケーションから，その振舞いを表す Web アプリケーションモデルの抽出が可能であるか否かを検証した．実験の結果，提案手法を用いることにより，より多くの要件定義を満たすモデルの抽出が可能となることが明らかとなった．

以上の様に，本論文で提案した Web コンテンツ適応のためのアノテーションフレームワークと，Web アプリケーションモデル抽出支援手法を用いることで，近年の Web の主要な利用形態である情報発信と Web アプリケーションの双方において，既存資源を有効活用した効率の良い Web 開発を支援することが可能となる．

また，2 つの手法を組み合わせて利用することで，Web 開発の効率をさらに高めることが出来る可能性がある．例えば，Web アプリケーションに対して，パラメータの追加・削除やリンク先の変更，文章の修正などの新たな要件定義や変更の内容をアノテーションとして付与しておき，Web アプリケーションモデル抽出の際にそれらのアノテーションの内容に基づいて抽出したモデルを変更したり，モデルの各要素に対する修正点として関連付ける手法が考えられる．この様な手法

を用いることで，開発者が実際に動作している Web アプリケーションの GUI 上で変更点を記述することが可能となり，特にリンク先の変更や文章の修正といった具体的な変更の場合はモデル上で変更を行うよりも作業が容易になると考えられる．今後，この様な手法の実現に向けてさらなる研究を行って行きたい．

今後の課題としては，Web コンテンツの開発サイクルをより円滑に進める手法に関する，さらなる検討の必要性も挙げられる．例えば，現在 W3C では，XML 文書内の要素を識別するための統一的な方法が提案されている [101]．今後，この様な仕組みとアノテーションフレームワークとの連携を検討し XPath の頑健性を高めるなどの検討を行っていきたい．さらに，近年，ソフトウェアを機能毎に分離してインタフェースを Web 上で公開し，それらを組み合わせることで大規模な業務アプリケーションを構築する手法が提案されている．例えば，BPEL[118] や BPML[119] 等ビジネスプロセス記述言語を用いて業務プロセスを定義し，各プロセスから Web サービスを呼び出すことで，大規模システムを構築する手法が提案されている [120]．これらの手法を用いることでより柔軟なシステムの構築が期待される．今後，この様な技術と本論文で提案した手法との親和性を高めることを検討し，適用範囲をさらに拡大していきたい．

参考文献

[1] T. Berners-Lee, R. Cailliau, J.-F. Groff, and B. Pollermann, "World-wide web: The information universe," *Electronic Networking: Research, Applications and Policy*, vol. 1, no. 2, pp. 74–82, 1992.

[2] W3C Recommendation, "HTML 4.01 Specification." `http://www.w3.org/TR/html4/`, December 1999.

[3] W3C Recommendation, "XHTML 1.0 The Extensible HyperText Markup Language." `http://www.w3.org/TR/xhtml1/`, January 2000.

[4] W3C Recommendation, "Cascading Style Sheets." `http://www.w3.org/TR/CSS1`, May 2001.

[5] Netscape and Sun Microsystems, "JavaScript 1.5." `http://www.mozilla.org/js/`.

[6] Sun Microsystems, "Java Plug-in Developer Guide." `http://java.sun.com/j2se/1.5.0/docs/guide/plugin/developer_guide/contents.html`.

[7] Macromedia, "Whitepaper: The Flash Platform," June 2005.

[8] J. Maeda, K. Fukuda, H. Takagi, and C. Asakawa, "Web Accessibility Technology at the IBM Tokyo Research Laboratory," *IBM Journal of Research and Development*, vol. 48, pp. 735–749, September 2004.

[9] C. Asakawa and T. Itoh, "User Interface of a Home Page Reader," in *Proceedings of the International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 1998)*, pp. 149–156, April 1998.

[10] The Productivity Works, "pwWebSpeak." `http://www.prodworks.com/`.

[11] jig.jp, "jig browser." `http://br.jig.jp/pc/`.

[12] Opera Software, "Opera for Mobile." `http://www.opera.com`.

[13] Programmers' Factory, Inc., "Scope." `http://www.programmer.co.jp/`.

[14] J. Freire, B. Kumar, and D. Lieuwen, "WebViews: Accessing Personalized Web Content and Services," in *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pp. 576–586, May 2001.

[15] E. J. Chikofsky and J. H. Cross II, "Reverse Engineering and Design Recovery: A Taxonomy," *IEEE Software*, vol. 7, pp. 13–17, January 1990.

[16] S. Tilley and S. Huang, "Evaluating the Reverse Engineering Capabilities of Web Tools for Understanding Site Content and Structure: A Case Study," in *Proceedings of the 23rd ICSE 2001*, pp. 514–523, May 2001.

[17] S. Parkin, "Rapid Java and J2EE Development with IBM WebSphere Studio and IBM Rational Developer." IBM Rational white paper, `http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/wp-radrwd-medres.pdf`, 2004.

[18] IBM, "IBM Rational Application Developer for WebSphere Software." `http://www-306.ibm.com/software/awdtools/developer/application/index.html`.

[19] IBM Corporation, "IBM Rational Web Developer for WebSphere Software Version 6.0." `http://www-306.ibm.com/software/awdtools/developer/application/index.html`.

[20] Eclipse Foundation, "Eclipse Web Tools Platform (WTP) Project." `http://www.eclipse.org/webtools/index.html`.

[21] Sun Microsystems, "JavaServer Pages (JSP)." `http://java.sun.com/products/jsp/`.

[22] A. E. Hassan and R. C. Holt, "A Visual Architectural Approach to Maintaining Web Applications," *Software Visualization: From Theory to Practice*, vol. 734, pp. 219–242, 2003.

[23] G. Antoniol, G. Canfora, G. Casazza, and A. D. Lucia, "Web Site Reengineering Using RMM," in *Proceedings of the 2nd International Workshop on Web Site Evolution*, pp. 9–16, March 2000.

[24] Zope Corporation, "Zope." `http://www.zope.org/`.

[25] IBM Corporation, "Lotus Domino Document manager (Domino.Doc)." `http://www-6.ibm.com/jp/domino07/lotus/home.nsf/Content/65`.

[26] IBM Corporation, "IBM WebSphere Portal Family." `http://www-306.ibm.com/software/info1/websphere/index.jsp?tab=products/portal`.

[27] T. W. Bickmore and B. N. Schilit, "Digestor: Device-independent Access to the World Wide Web," *The International Journal of Computer and Telecommunications Networking*, vol. 29, pp. 1075–1082, September 1997.

[28] F. Rousseau, J. A. Garcia-Macias, J. V. de Lima, and A. Duda, "User Adaptable Multimedia Presentations for the World Wide Web," in *Proceedings of the 8th International World Wide Web Conference (WWW8)*, pp. 195–212, 1999.

[29] M. Hori, G. Kondoh, K. Ono, S. Hirose, and S. Singhal, "Annotation-Based Web Content Transcoding," in *Proceedings of the 9th International World Wide Web Conference (WWW9)*, pp. 197–211, 2000.

[30] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Seeing the Whole in Parts: Text Summarization for Web Browsing on Handheld Devices," in *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pp. 652–662, May 2001.

[31] W3C Recommendation, "Web Content Accessibility Guidelines 1.0." `http://www.w3.org/TR/WCAG10/`, May 1999.

[32] IBM Corporation, "JAWS for Windows." `http://www-6.ibm.com/jp/accessibility/soft/jaws.html`.

[33] Kochi System Development, "PC-Talker." `http://www.pctalker.net/`.

[34] 福田 健太郎, 高木 啓伸, 前田 潤治, 浅川 智恵子, "視覚障害者のための Web ページ情報提示手法," **ヒューマンインタフェース学会論文誌**, vol. 5, pp. 465–474, November 2003.

[35] K. Nagao, Y. Shirai, and K. Squire, "Semantic Annotation and Transcoding: Making Web Content More Accessible," *IEEE Multimedia*, vol. 8, no. 2, pp. 69–81, 2001.

[36] R. Barrett and P. Maglio, "Intermediaries: New Places for Producing and Manipulating Web Content," *Computer Networks and ISDN Systems*, pp. 509–518, 1998.

[37] W3C Recommendation, "Extensible Markup Language (XML) 1.0 (Third Edition)." `http://www.w3.org/TR/REC-xml/`.

[38] W3C Recommendation, "XML Schema Part 1: Structures." `http://www.w3.org/TR/xmlschema-1/`, May 2001.

[39] G. Buchanan, S. Farrant, M. Jones, H. Thimbleby, G. Marsden, and M. Pazzani, "Improving Mobile Internet Usability," in *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pp. 673–680, May 2001.

[40] Asia-Pacific Association for Machine Translation (AAMT), **機械翻訳** *21* **世紀のビジョン**. アジア太平洋機械翻訳協会, November 2000.

[41] H. Watanabe, K. Nagao, M. C. McCord, and A. Bernth, "An Annotation System for Enhancing Quality of Natural Language Processing," in *Proceedings of the 19th International Conference on Computational Linguistics (COLING2002)*, vol. 2, pp. 1303–1307, Aug 2002.

[42] H. Kanayama and H. Watanabe, "Multilingual Translation via Annotated Hub Language," in *Proceedings of Machine Translation Summit IX*, September 2003.

[43] I. Singh, B. Stearns, M. Johnson, and Enterprise Team, *Designing Enterprise Applications with the J2EE Platform, Second Edition.* Addison-Wesley, 2002.

[44] G. E. Krasner and S. T. Pope, "A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System," *Journal of Object Oriented Programming*, vol. 1, no. 3, pp. 26–49, 1988.

[45] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object Oriented Software.* Addison-Wesley Publishing Company, 1995.

[46] Apache Software Foundation, "Jakarta Struts Web Application Framework." `http://jakarta.apache.org/struts/`.

[47] Apache Software Foundation, "Jakarta Turbine Web Application Framework." `http://jakarta.apache.org/turbine/`.

[48] Enhydra.org Project, "Brracuda: MVC presentation framework for Web applications." `http://barracuda.enhydra.org/`.

[49] 津久井 浩, 中所 武司, "Web アプリケーションにおける予約業務フレームワークの実現と再利用性の評価," **オブジェクト指向最前線** *2003*, pp. 33–40, July 2003.

[50] 堀 雅洋, 田井 秀樹, "モデルに基づく Web アプリケーション開発," **情報処理学会誌**, vol. 45, no. 1, pp. 16–21, 2004.

[51] P. Fraternali and P. Paolini, "Model-driven development of Web applications: the Autoweb system," *ACM Transactions on Information Systems*, vol. 18, pp. 323–382, October 2000.

[52] 位野木 万里, 山田 広佳, "Web アプリケーション開発における設計・設計検証・テストプロセスの提案," **情報処理学会 研究報告** (SE-143-007), pp. 45–52, July 2003.

[53] 田井 秀樹, 根路銘 崇, 安部 麻里, 堀 雅洋, "モデルに基づく web アプリケーション開発支援環境," **情報処理学会論文誌**, vol. 44, pp. 1498–1508, June 2003.

[54] S. J. Mellor, A. N. Clark, and T. Futagami, "Model-Driven Development," *IEEE Software*, pp. 14–18, September-October 2003.

[55] Software Development Group of the National Center for Supercomputing Applications at the University of Illinois at Urbana, "The CGI Specification." `http://hoohoo.ncsa.uiuc.edu/cgi/interface.html`.

[56] M. Abe and M. Hori, "A Visual Approach to Authoring XPath Expressions," in *Proceedings of Extreme Markup Languages 2001*, pp. 1–15, Aug. 2001.

[57] M. Abe and M. Hori, "A Visual Approach to Authoring XPath Expressions," *Markup Languages: Theory & Practice*, no. 2, pp. 191–212, 2002.

[58] 安部 麻里, 堀 雅洋, "外部アノテーションのための XPath 表現の頑強性について," **人工知能学会第** *57* **回知識ベースシステム研究会** *(SIGKBS)*, pp. 19–26, 2002.

[59] M. Abe and M. Hori, "Robust Pointing by XPath Language: Authoring Support and Empirical Evaluation," in *Proceedings of 2003 Symposium on Applications and the Internet (SAINT2003)*, pp. 156–165, January 2003.

[60] M. Hori, K. Ono, T. Koyanagi, and M. Abe, "Annotation by Transformation for the Automatic Generation of Content Customization Metadata," in *Proceedings of the First International Conference on Pervasive Computing (PvC 2002)*, vol. 2414 of *Lecture Notes in Computer Science*, pp. 267–281, August 2002.

[61] K. Ono, T. Koyanagi, M. Abe, and M. Hori, "XSLT Stylesheet Generation by Example with WYSIWYG Editing," in *Proceedings of the 2002 Symposium on Applications and the Internet (SAINT2002)*, pp. 150–159, January 2002.

[62] M. Hori, M. Abe, and K. Ono, "Extensible Framework of Authoring Tools for Web Document Annotation," in *Proceedings of the International Workshop on Semantic Web Foundations and Application Technologies (SWFAT)*, pp. 1–16, March 2003.

[63] M. Hori, M. Abe, and K. Ono, "Robustness of External Annotation for Web-page Clipping: Empirical Evaluation with Evolving Real-life Web Documents," *Second International Conference on Knowledge Capture (K-CAP 2003), Knowledge Markup and Semantic Annotation Workshop Notes*, pp. 65–72, October 2003.

[64] M. Hori, M. Abe, K. Ono, and T. Koyanagi, "Generating Transformational Annotation for Web Document Adaptation: Tool Support and Empirical Evaluation," *Journal of Web Semantics*, vol. 2, no. 1, pp. 1–28, 2004.

[65] 安部 麻里, 福田健太郎, 堀 雅洋, 田井 秀樹, 根路銘 崇, 小野 康一, 大野義夫, “動的解析による Web アプリケーション・モデル抽出支援手法,” **情報処理学会論文誌**, vol. 46, pp. 683–693, March 2005.

[66] 安部 麻里, 福田 健太郎, 田井 秀樹, 根路銘 崇, 堀 雅洋, “動的逆解析による Web アプリケーション・モデルの抽出,” **日本ソフトウェア科学会第** *20* **回大会論文集**, pp. 546–550, September 2003.

[67] H. Tai, T. Nerome, M. Abe, and M. Hori, “Model-driven Development of Dynamic Web Applications,” in *Proceedings of Extreme Markup Languages 2002*, August 2002.

[68] 田井 秀樹, 根路銘 崇, 安部 麻里, 堀 雅洋, “モデルに基づく Web アプリケーション開発環境 WAST,” **オブジェクト指向最前線** *2002*, pp. 35–42, August 2002.

[69] H. Tai, K. Mitsui, T. Nerome, M. Abe, K. Ono, and M. Hori, “Model-Driven Development of Web Applications,” *IBM Journal of Research and Development*, vol. 48, pp. 797–809, November 2004.

[70] 根路銘 崇, 小野 康一, 田井 秀樹, 安部 麻里, “Web アプリケーションモデルに基づく JSP の動的検証,” **ソフトウェアテストシンポジウム** *(JaSST'04)*, 2004.

[71] Jose Kahan and Marja-Riitta Koivunen and Eric Prud'Hommeaux and Ralph R. Swick, “Annotea: An Open RDF Infrastructure for Shared Web Annotations,” in *Proceedings of the 10th International World Wide Web Conference (WWW10)*, pp. 623–632, 2001. Hong-Kong.

[72] O. Lassila, “Web Metadata: A Matter of Semantics,” *IEEE Internet Computing*, vol. 2, no. 4, pp. 30–37, 1998.

[73] C. C. Marshall, “Toward an Ecology of Hypertext Annotation,” in *Proceedings of the ninth ACM Conference on Hypertext and Hypermedia*, pp. 40–49, 1998. Pittsburgh, PA.

[74] M. Erdmann, A. Maedche, H.-P. Schnurr, and S. Staab, “From Manual to Semi-automatic Semantic Annotation: About Ontology-based Text Annotation tools,” in *Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content*, 2000.

[75] W3C Recommendation, “XML Path Language (XPath) Version 1.0.” `http://www.w3.org/TR/xpath`, November 1999.

[76] L. Denoue and L. Vignollet, “An Annotation Tool for Web Browsers and its Applications to Information Retrieval,” in *Proceedings of the 6th Conference on Content-Based Multimedia Information Access (RIAO 2000)*, 2000.

[77] T. Sakairi and H. Takagi, "An Annotation Editor for Nonvisual Web Access," in *Proceedings of the 9th International Conference on Human-Computer Interaction (HCI International 2001)*, pp. 982–985, 2001.

[78] M. Hori, K. Ono, G. Kondoh, and S. Singhal, "Authoring Tool for Web Content Transcoding," *Markup Languages: Theory & Practice*, vol. 2, no. 1, pp. 81–106, 2000.

[79] A. J. B. Brush, D. Bargeron, A. Gupta, and J. J. Cadiz, "Robust Annotation Positioning in Digital Documents," in *Proceedings of the 2001 ACM Conference on Human Factors in Computing Systems (CHI 2001)*, pp. 285–292, 2001.

[80] M. Han, C. Hofmeister, and R. L. Nord, "Reconstructing Software Architecture for J2EE Web Applications," in *Proceedings of the 10th Working Conference on Reverse Engineering(WCRE'03)*, pp. 67–79, November 2003.

[81] J. Conallen, "Modeling Web Application Architectures with UML," *Communications of the ACM*, vol. 42, pp. 63–70, October 1999.

[82] M. Salah and S. Mancoridis, "Toward an Environment for Comprehending Distributed Systems," in *Proceedings of the 10th Working Conference on Reverse Engineering(WCRE'03)*, pp. 238–247, November 2003.

[83] J. Conallen, *Building Web Applications with UML, Second Edition*. Addison-Wesley, 2002.

[84] M. Hori, "Semantic Annotation for Web Content Adaptation," *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, pp. 403–430, December 2002.

[85] T. A. Phelps and R. Wilensky, "Robust Intra-document Locations," in *Proceedings of the 9th International World Wide Web Conference (WWW9)*, pp. 105–118, 2000. Amsterdam, Netherlands.

[86] W3C Recommendation, "XSL Transformations (XSLT) Version 1.0." `http://www.w3.org/TR/xslt`, November 1999.

[87] W3C Recommendation, "Document Object Model (DOM) Level 3 Core Specification Version 1.0." `http://www.w3.org/TR/DOM-Level-3-Core/`, 2004.

[88] W3C Recommendation, "XML Linking Language (XLink) Version 1.0." `http://www.w3.org/TR/xlink/`, June 2001.

[89] W3C Recommendation, "XML Information Set (Second Edition)." `http://www.w3c.org/TR/xml-infoset/`, Februrary 2004.

[90] W3C Recommendation, "RDF Vocabulary Description Language 1.0: RDF Schema." http://www.w3.org/TR/rdf-schema/, Februrary 2004.

[91] W3C Recommendation, "RDF Primer." http://www.w3.org/TR/rdf-primer/, Februrary 2004.

[92] V. D. Mea and C. A. Beltrami, "HTML Generation and Semantic Markup for Telepathology," in *Proceedings of the 5th International World Wide Web Conference (WWW5)*, pp. 1085–1094, 1996.

[93] 福田 健太郎, 高木 啓伸, 前田 潤治, 浅川 智恵子, "アクセシビリティ向上のための Web コンテンツトランスコーディングシステム," **電子情報通信学会 技術研究報告** (WIT2001-15), pp. 31–36, August 2001.

[94] H. Takagi, C. Asakawa, K. Fukuda, and J. Maeda, "Site-wide Annotation: Reconstructing Existing Pages to be Accessible," in *Proceedings of the International ACM SIGCAPH Conference on Assistive Technologies (ASSETS 2002)*, pp. 81–88, June 2002.

[95] Internet Engineering Task Force (IETF), RFC 2396, "Uniform Resource Identifiers (URI): Generic Syntax." http://www.ietf.org/rfc/rfc2396.txt, 1998.

[96] IBM Corporation, "WebSphere Transcoding Publisher Version 4.0 Developer's Guide." ftp://ftp.software.ibm.com/software/webserver/transcoding/brochures/tpdgmst.pdf, 2001.

[97] W3C Working Draft, "XPointer xpointer() Scheme." http://www.w3.org/TR/xptr-xpointer/, December 2002.

[98] T. Lindholm, "The "3DM" XML 3-way Merging and Differencing Tool." http://www.cs.hut.fi/~ctl/3dm/.

[99] T. Lindholm, "A 3-way merging algorithm for synchronizing ordered trees - the 3DM merging and differencing tool for XML," Master's thesis, Dept. of Computer Science, Helsinki University of Technology (http://www.cs.hut.fi/~ctl/3dm/thesis.pdf), 2001.

[100] R. Spinks, B. Topol, C. Seekamp, and S. Ims, "Document Clipping with Annotation." http://www.ibm.com/developerworks/ibm/library/ibm-clip/, 2001.

[101] W3C Proposed Recommendation, "xml:id Version 1.0." http://www.w3.org/TR/xml-id, July 2005.

[102] W3C Note, "Compact HTML for Small Information Appliances." http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/, February 1998.

[103] Wireless Aplication Protocol Forum, Ltd., "Wireless Markup Language Version 2.0," September 2001.

[104] W3C Recommendation, "Voice Extensible Markup Language (VoiceXML) Version 2.0." http://www.w3.org/TR/voicexml20/, March 2004.

[105] IBM Corporation, "WebSphere Voice Server." http://www-306.ibm.com/software/pervasive/voice_server/, 2001.

[106] IBM Corporation, "WebSphere Translation Server." http://www-6.ibm.com/jp/software/websphere/bp/translation/, 2001.

[107] IBM Corporation, "IBM WebSphere Portal V4 Developer's Handbook." http://www.redbooks.ibm.com/redbooks/pdfs/sg246897.pdf, 2003.

[108] C. Wege, "Portal Server Technology," *IEEE Internet Computing*, vol. 6, pp. 73–77, May/June 2002.

[109] Java Specification Requests, "JSR 168: Portlet Specification." http://www.jcp.org/en/jsr/detail?id=168.

[110] S. DeWitt, "Basic Web Clipping Using WebSphere Portal Version 4.1." http://www7b.software.ibm.com/wsdd/library/techarticles/0206_dewitt/dewitt.html, 2002.

[111] Sun Microsystems, "Enterprise JavaBeans Specification, Version 2.1."

[112] Object Management Group, Inc, "OMG Unified Modeling Language Specification." http://www.omg.org/technology/documents/formal/uml.htm.

[113] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. Addison-Wesley Object Technology Series, 1998.

[114] Java Community Process, "Java Servlet Specification." http://jcp.org/aboutJava/communityprocess/final/jsr154/.

[115] Eclipse Foundation, "Eclipse Platform." http://www.eclipse.org/.

[116] K. Fukuda, H. Takagi, J. Maeda, and C. Asakawa, "Layout Group Extraction fromWeb Content for Effective Adaptation," *IBM Research Report* (RT0493), November 2002.

[117] NetResults Corporation, "WebVCR." http://www.netresultscorp.com/.

[118] S. Thatte, T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, I. Trickovic, and S. Weerawarana, "Process Execution Language

for Web Services Version 1.1." ftp://www6.software.ibm.com/software/developer/library/ws-bpel.pdf, May 2003.

[119] A. Arkin, "Business Process Modeling Language." http://www.bpmi.org/BPML.htm, March 2001.

[120] M. P. Papazoglou, "Service -Oriented Computing: Concepts, Characteristics and Directions," in *Proceedings of the Fourth International Conference on Web Information Systems Engineering*, pp. 3–12, IEEE Computer Society, December 2003.

付 録A　アノテーション文書のスキーマ例

```
<!ELEMENT annot (description|comment)*>
<!ATTLIST annot version CDATA "1.0">
<!ELEMENT description (author|role|date|importance|comment)* >
<!ATTLIST description target CDATA #REQUIRED>
<!ELEMENT author (#PCDATA)>
<!ATTLIST author id  ID>
<!ELEMENT date (#PCDATA)>
<!ELEMENT importance (#PCDATA)>
<!ELEMENT comment (#PCDATA)>
<!ELEMENT role (#PCDATA)>
```

図 A.1: EML Sample Annotation プロファイルで参照される DTD (EMLSample.dtd)

Annotation Element	Description
<description>	Prescribe a unit of annotation instructions
<remove>	Remove associated HTML element(s)
<keep>	Keep associated HTML element(s)
<table>	Effect overall table, in particular the heading
<column>	Remove a complete table column
<row>	Remove a complete table row
<field>	Modify fields within a form
<option>	Specify a selectable option
<insertattribute>	Allow insertion of attributes into an HTML element
<inserthtml>	Allow insertion of an HTML element
<replace>	Replace an associated content with another content
<replacewithhtml>	Replace an associated HTML tag with another HTML tag
<setpreference>	Set a preference parameter for transcoding
<splitpoint>	Indicate a preferred point of document splitting

図 A.2: ページクリッピングのためのアノテーション語彙

付 録B　要件定義

表 B.1: オンラインショップの要件定義 (1)

種別	要件	従属項目	説明
画面要件	[A01] ログオン (A02, A03)	◇ メッセージ	ログオン時のメッセージを表示
		ユーザ名入力フィールド	
		パスワード入力フィールド	
		送信ボタン	
	[A02] ログオン失敗 (A01)	◇ メッセージ	ログオン失敗時のメッセージを表示
		ログオン画面へのリンク	
	[A03]Welcome(A04)	◇ メッセージ	ログオン成功時のメッセージを表示
		◇ ユーザ名	
		製品カタログ画面へのリンク	
	[A04] 製品カタログ (A05)	◇ 製品名称	
		製品選択リンク	製品を選択しカートへ追加するためのリンク
		カートへのリンク	
	[A05] カート (A04, A05, A06)	◇ 製品名称	
		◇ 個数	該当する製品の個数を表示
		個数追加リンク	個数を増やしカートへ追加するためのリンク
		◇ 小計	該当する製品の小計を表示
		◇ 合計金額	カートの合計金額を表示
		製品カタログ画面へのリンク	
		カート内容削除用リンク	
		チェックアウト処理へのリンク	
	[A06] 内容確認 (A05, A07)	◇ カート内容	
		◇ 合計金額	
		チェックアウト画面へのリンク	
		製品カタログ画面へのリンク	ショッピングを続けるためのリンクを表示
	[A07] チェックアウト (A01)	◇ メッセージ	チェックアウト時のメッセージを表示
		ログオン画面へのリンク	

表 B.2: オンラインショップの要件定義 (2)

種別	要件	従属項目	説明
機能要件	[B01] ログオン認証	認証処理	
	[B02] 製品情報一覧取得	＊ 製品名称取得	製品一覧情報より製品項目リストを取得
		＊ 製品画像取得	
		＊ 製品単価取得	
	[B03] カート内容照会	◇ 製品名称取得	カート内容に含まれる製品項目より取得
		◇ 製品単価取得	
		◇ 製品個数取得	
		◇ 小計計算	小計は単価と個数を製品毎に計算
		◇ 合計金額計算	小計の総和を計算
		カート ID 取得	
	[B04] カート内容変更	個数増加	
	[B05] カート内容削除	カート内容削除	
	[B06] 清算処理	カート内容取得	
		＊ 合計金額計算	
データ要件	[C01] 製品項目	製品 ID	
		◇ 単価	
		◇ 画像	
	[C02] 製品一覧	＊ 製品項目リスト	製品項目がリストとして登録されている
	[C03] ユーザ情報	ユーザ名	
		パスワード	
		◇ 配送先	

表 B.3: 掲示板の要件定義 (1)

種別	要件	従属項目	説明
画面要件	[A01] 掲示板 (A05,A06,A07,A09)	◇ タイトル	
		ヘルプ画面へのリンク	
		記事検索画面へのリンク	
		管理者ログオンへのリンク	
		名前入力フィールド	投稿者の名前を入力
		メールアドレス入力フィールド	
		題名入力フィールド	
		記事内容入力フィールド	
		URL フィールド	
		削除キー入力フィールド	投稿記事の削除に必要な数値 4 桁を入力
		送信ボタン	入力された記事を送信
		◇ 投稿記事一覧	過去に登録された記事の一覧表示
		返信ボタン	
		記事番号入力フィールド	記事一覧から削除する記事番号を入力
		削除キー入力フィールド	記事一覧から削除する記事の削除キーを入力
		削除ボタン	
		ページ送りボタン	
		ページ戻しボタン	
	[A02] 投稿エラー (A01)	◇ メッセージ	入力値の妥当性確認結果を表示
		掲示板へのリンク	
	[A03] 投稿完了 (A01)	◇ メッセージ	
		掲示板へのリンク	
	[A04] ユーザ認証エラー (A01)	◇ メッセージ	
		掲示板へのリンク	
	[A05] ヘルプ (A06)	◇ メッセージ	
		記事検索へのリンク	
		掲示板へのリンク	
	[A06] 記事検索 (A01)	トップページへのリンク	
		◇ 使い方説明	
		キーワード入力フィールド	
		条件選択メニュー	
		表示件数選択メニュー	
		検索ボタン	
		◇ 検索結果に該当する記事一覧	
		ページ送りボタン	
	[A07] 管理者ログオン	◇ メッセージ	
		パスワード入力フィールド	
		認証ボタン	
	[A08] 管理者認証エラー (A07)	◇ メッセージ	
		管理者ログオン画面へのリンク	
	[A09] 記事削除 (A01)	◇ メッセージ	
		ログオフ用ボタン (掲示板へのリンク)	
		削除実行ボタン	
		◇ コメント一覧	
		記事削除チェックボックス	

表 B.4: 掲示板の要件定義 (2)

種別	要件	従属項目	説明
機能要件	[B01] 記事表示機能	◇ コメント取得	コメント一覧よりコメント取得
		◇ リンク作成	表示上限を超えたときページを送るリンクを作成
	[B02] 記事送信機能	記事妥当性検証	記事フォーマットの妥当性確認
		記事追加	
	[B03] 記事返信機能	引用文作成	対象記事の引用文を作成
	[B04] 記事削除機能	パスワードと番号の認証	
		記事削除	
	[B05] 記事検索機能	検索必須項目取得	
		∗ 記事検索	記事一覧より検索
		∗ リンク作成	表示上限を超えたときページを送るリンクを作成
	[B06] 管理者認証機能	パスワード認証	
	[B07] 管理者用記事削除機能	記事削除	
データ要件	[C01] 記事項目	◇ 番号	
		◇ 日付	
		タイトル	
		名前	
		メールアドレス	
		URL	
		記事	
		削除キー	
	[C02] 記事一覧	∗ 記事項目の一覧	
	[C03] 管理者情報	パスワード	

表 B.5: アンケートの要件定義

種別	要件	従属項目	説明
画面要件	[A01] アンケート入力 (A04)	集計結果ログオンへのリンク	
		◇ タイトル	
		名前入力フィールド	アンケート回答者の名前入力
		メールアドレス入力フィールド	アンケート回答者のメールアドレス入力
		設問 1(プルダウン)	プルダウンで示された選択肢から選択
		設問 2(チェックボックス)	
		設問 3(ラジオボタン)	
		設問 4(テキストフィールド)	
		設問 5(ファイル)	好みの画像ファイルを選択
		送信ボタン	
	[A02] 送信完了 (A01)	◇ メッセージ	
		アンケート入力へのリンク	
	[A03] 入力エラー (A01)	◇ メッセージ	エラーのあったアンケート項目を表示
		アンケート入力へのリンク	
	[A04] 集計結果ログオン (A01)	アンケート入力へのリンク	
		パスワード	
		送信ボタン	
	[A05] 認証エラー (A04)	◇ メッセージ	
		集計結果ログオンへのリンク	
	[A06] 集計結果 (テーブル)(A01, A07)	アンケート入力へのリンク	
		パスワード	
		◇ アンケート結果一覧	
		CSV 出力ボタン	
	[A07] 集計結果 (CSV)(A01, A06)	アンケート入力へのリンク	
		パスワード	
		◇ アンケート結果一覧	
		テーブル出力ボタン	
機能要件	[B01] アンケート集計	入力値の妥当性確認	
		集計	送信されたアンケートを結果一覧へ追加
	[B02] パスワード認証	結果表示のための認証	
	[B03] 結果表示機能	∗ 結果一覧取得	
データ要件	[C01] アンケート項目	名前	
		メールアドレス	
		設問 1 回答	
		設問 2 回答	
		設問 3 回答	
		設問 4 回答	
		設問 5 回答	
	[C02] 結果一覧	∗ アンケート項目のリスト	
	[C03] ログイン情報	パスワード	