高信頼Webシステム開発技術への取り組み

野村総合研究所
技術調査室長

中元　秀明（なかもとひであき）

専門はエンタープライズ系情報システムの分析・設計・開発に関する技術。
名古屋大学情報科学研究科客員助教授

野村総合研究所
生産技術部　主任テクニカルエンジニア

森川　和明（もりかわかずあき）

情報技術本部にてオブジェクト指向技術を活用したシステム開発に取り組むITエンジニア。
最近はWebシステムの生産性向上技術研究・開発に従事している。

野村総合研究所
企画・業務管理室　主任ITデザイナ

三井　英樹（みついひでき）

ITデザイン動向の調査研究を行うITデザイナ。専門は、UIデザインおよびWebサイト開発ディレクション関連技術。RIA（Rich Internet Application）コンソーシアム、RIAエバンジェリスト。ライフワークは、デザイナとエンジニアの協労環境作り。

要旨

NRI情報技術本部では、2003年度より文部科学省リーディングプロジェクト「e-Society基盤ソフトウェアの総合開発」の1テーマである「高信頼WebWareの生成技術」（代表者：名古屋大学／阿草 清滋教授、和歌山大学／鯵坂 恒夫教授）に参画して、信頼性の高いWebシステム開発を可能にする技術開発に取り組んできた。

2005年4月には、この研究成果の製品化第1弾として、NRI製品オブジェクトワークスに静的整合性チェックツールを組み込んでリリースした。現在は、Webシステムのテストケースやテストデータの自動生成に取り組んでいる。また、Webサイト視覚化ツールであるNRI製品RidualにJavaScript解析機能を組み込む研究も行っている。

キーワード：Webシステム、リポジトリ、静的解析技術、オブジェクトワークス、Ridual、産学連携、ソフトウェア工学

NRI Information Technology head office has been participating in "Generation technology of high reliability WebWare" which is a theme of "Comprehensive Development of e-Society Foundation Software", the leading project of Ministry of Education, Culture, Sports, Science and Technology since the beginning of the fiscal year of 2003. The project is conducted by Professor Seiji Agusa of Nagoya University and Professor Tsuneo Ajisaka of Wakayama University. We have been engaging in developing technologies to enable highly reliable Web system through this project.
In April 2005, we have released NRI brand ObjectWorks with embedded statistical coordination verification tool as the first product resulted from this project. Currently, we are engaged in tasks to enable automatic generation of test cases and test data of Web systems. We are also researching on embedding JavaScript analyzing function into Ridual, NRI's web site visualization tool.

Keywords：Web system, Repository, Static analysis technology, Object Works, Ridual, Academic-industrial alliance, Software engineering

1．Webシステムの課題

Web技術は、低コストなシステム基盤として一般消費者向けのインターネット販売や企業間取引、企業内の情報システムなど様々な用途で使用されている。あたりまえのように使われているWeb技術であるが、例えばインターネット販売では一般消費者が直接利用することから、従来の企業内に閉じていた情報システム以上に信頼性の高さが求められるといえる。しかし、Webシステムにおいて、高信頼性を確保するのは必ずしも容易ではない。

Webシステムは複数の構成要素からなる。例えば、一般的に適用されるMVCモデル（Model-View-Controller）に基づくWebシステムは、画面表示に関するプレゼンテーションロジックや、画面遷移制御等を行うコントローラ、業務処理を実行するビジネスロジックから構成され、さらにそれらが異なるプログラム言語やスクリプト言語によって記述されるという特徴をもつ。例えば、画面表示内容はHTMLで記述され、画面上でのデータ処理にはJavaScriptが使用される。サーバ側の処理はJSP、PHP、Java等が使われる。

これらの異なる開発言語で構成される情報システム全体が正しく動作するには、各構成要素間の論理的な整合性が取れている必要がある。このような論理的な正当性の確認を、ソースコードを読みながら人間が行うには大きな労力を要する。作業の自動化が期待されるが、これまで複数の異なる言語間での整合性の確認を自動的に行う技術は存在しなかった。そこで現実には、実際にアプリケーションを実行して問題が起こるかどうかテストするという方法がとられることが多い。この場合、テストで発見しにくい不良を見逃してしまう危険性が大きい。これはWebシステムの信頼性を低下させる原因の1つとなっている。

また、テストにおいても、膨大なテストケースの設定とテストデータの準備、テストの実行と結果の検証に多くの作業量と時間を要している。作業の効率化が大きな課題となっている。

もう1つ、Webシステムには、Webサイトデザインが重要な役割をもつという特徴がある。特に、インターネット販売のように対顧客ビジネスプロセスが直接Webサイトで実施される場合、見通しのよいサイト構成、特別な訓練なしに容易に扱うことのできる操作性のよさ、魅力的なデザインの実現等は、商品の販売実績そのものに影響を及ぼすと考えてよいであろう。

一般に企業が外部向けに提供するWebサイトは、相互に関連した膨大なページから構成される。また、プログラムによるページの動的な生成や関連付けも行われる。さらにWebサイトの魅力を高めるには、コンテンツやサイト構成を頻繁に変えることも求められる。このとき、現状のサイト構成を正確に把握することが必要となるが、そのための労力は大きい。

情報システム開発の短工期化、高品質化の

要求がますます高まる中で、Webシステムの抱える前述のような問題点を解決することは重要な課題となっている。

2. 高信頼Webシステム開発技術への取り組み

文部科学省リーディングプロジェクト「e-Society基盤ソフトウェアの総合開発」は、世界最高水準の高度情報通信システム形成のための鍵となるソフトウェア開発を実現させ、いつでもどこでも誰でも安心して参加できるIT社会を構築することを目的に、2003年度より実施されている。

当プロジェクトにおいて名古屋大学／阿草清滋教授、和歌山大学／鯵坂 恒夫教授を中心とするグループは、高信頼なWebシステムを構築するための研究開発「高信頼WebWare生成技術」に取り組んでおり、NRIもこの研究に参加している。

WebWareとは、インターネット取り引きや企業内の情報共有システムなど、現在のビジネス情報システムで広く一般的に利用されている「Web技術を使って作られた情報システム」を表す概念である。

名古屋大学では長年にわたりプログラム言語の静的解析技術の研究を行っている。この技術を適用すると、プログラムに記述された処理内容やデータ項目定義などの情報を知ることができ、それらに対して各種の論理的なチェックを行うことが可能となる。前述のようなWebシステムの課題に対して適用すれば、プログラムの異言語間の整合性チェックを自動的に行うことなどが可能になる。NRIでは、大学の開発言語解析に関する研究成果を製品に組み込むことで、高品質なWebシステムの開発を可能とすることを目的に技術開発を行っている。

2005年4月には、この研究成果の製品化の第1弾として、Webアプリケーションの静的整合性チェックツールをNRIの開発フレームワーク製品オブジェクトワークスR6.5に搭載しリリースした。

現在は、テストケースの自動設定、テストデータの自動作成の技術開発に取り組んでいる。

NRIには、他にWebサイトの構造を視覚的に表現することでWebサイトデザインの支援を担う製品としてRidualがあるが、これに対しても名古屋大学と和歌山大学で開発されたJavaScript解析機能を取り込み、視覚的表現の可能な解析対象を拡大する研究に取り組んでいる。

3. Sapid

はじめにSapidについて概要を説明する。

Sapid（Sophisticated APIs for CASE tool Development）は、名古屋大学阿草研究室で開発されたソフトウェアで、CASEツール作成の共通基盤を目指した細粒度ソフトウェアリポジトリである。[1]

CASEツールにおいて、ソースコードからプログラムに関する情報を入手し、それを利用して様々な処理を行うことが必要となる。このとき、プログラムの情報を詳細に捕捉す

るには、変数名や分岐文といった構文要素の単位でプログラムを解析する必要がある。しかし、プログラムをこのような構文要素の単位で扱う際の基準や方法は、CASE ツールの設計者に委ねられるため、設計者によってまちまちとなる。また、対象とするプログラム言語は様々であり、新規の言語も頻繁に登場するので、それへの対応が迅速に行える必要がある。

そこで、字句解析と構文解析の部分をツールから切り離すことで、ツール作成の効率を改善させ、さらに様々なプログラムを構文要素の単位で扱うことのできる基準となるプラットフォームとして提案されたのが Sapid である。当初 C 言語を対象としていたが、現在では様々な言語のソースコードを解析・モデル化を行い、リポジトリに格納する。Java 言語、JSP ページの解析も可能となっている。

Java 言語を対象とする解析器を「Japid」と呼ぶ。Japid は、Java ソースを jx-model という Java 言語の特徴を踏まえたモデルに変換する。jx-model は、Java ソースコードを構成するクラスの定義やリテラル等を要素として含んでおり、利用者は、それらの情報を要素の種類を示す識別子付で読み出すことができる。

また、JSP を対象とする解析器を「Wapid」と呼ぶ。Wapid は JSP を wx-model という JSP の特徴を踏まえたモデルに変換する。wx-model には、スタイルシートの情報やタグで囲まれた JSP の要素を含んでいる。jx-model と同様に、これらを識別子付で読み出すことができる。

JavaScript についても、Japid や Wapid と同等の解析機能を、名古屋大学と和歌山大学にて開発した。

以降では、これらを応用した成果を述べる。まず、Web アプリケーションの整合性チェックツール及びテストケース自動生成ツールについての詳細を「4. 高信頼 Web システム開発の効率化技術」で述べる。次に Ridual への JavaScript 解析機能の組み込みについての詳細を「5. Web サイト全体像俯瞰の高度化技術」で述べる。

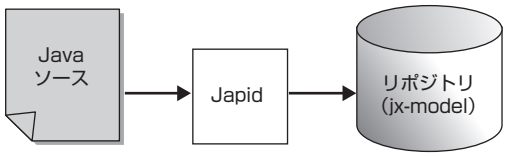


図 1　Japid

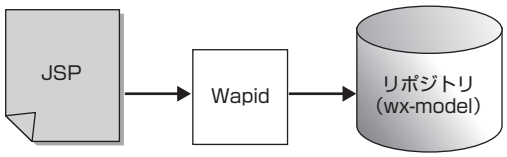


図 2　Wapid

4. 高信頼 Web システム開発の効率化技術

本節では、高信頼 Web システム開発の効率化技術研究におけるこれまでの成果を述べる。最初に Web アプリケーションの特徴について説明する。次に研究成果である整合性

チェック技術、単体テスト支援技術について紹介し、最後に今後の活動予定を述べる。

(1) Webアプリケーションの特徴

現在、高度な操作性等の特別な要件がない場合、企業システム構築にはブラウザを用いて操作を行うWebアプリケーション形式が採用されることが多い。その場合、Webアプリケーションを作成する技術の選択肢としてはCGI、ASP、ASP.NET、J2EE等が挙げられるが、J2EE技術が選択されることが比較的多く見受けられる。J2EE基盤ではPL（プレゼンテーションレイヤー）、BL（ビジネスロジックレイヤー）、およびPLとBLを結ぶコントローラを分けて、それぞれに適した技術、言語で作成することが推奨されている。一般にPLはJSP（JavaServerPages）、コントローラはJava Servlet、BLはJavaまたはEJBを用いて作成される。さらに、開発生産性を高めるためにNRIのオブジェクトワークスやオープンソースソフトウェアのStrutsのようなJ2EE基盤向け開発フレームワークを採用することがほぼ前提となりつつある。

システムをWebアプリケーションで作成した場合、複数言語、技術を組み合わせているため個別には動作しない。開発フレームワークを用いたとしても同様である。そのため、開発現場では、Webアプリケーションとして組み立ててから動作を確認しつつデバッグ、テストを行うことが一般的である。

ここに静的解析技術を応用することを考えた。静的解析技術を用いると、実行前に開発、テストに必要な情報を取得することが可能となった。この情報を活かし、ソース間の整合性をチェックする手法と、テストを支援する手法を考案した。

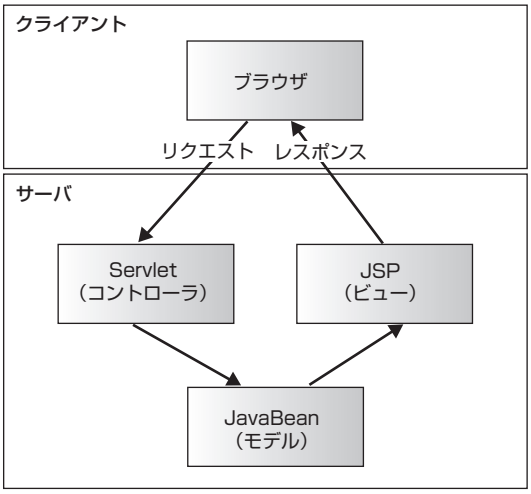


図3　Webアプリケーション構成例

(2) 整合性チェック

2003年度には、オブジェクトワークスおよびStrutsを使用したWebアプリケーションについて、構成するソース間の整合性をチェックする仕組みの研究を行い、ソース間整合性チェックツールの試作を行った。

ソースコード間に不整合があった場合、アプリケーションは正常に動作しない。複数言語を用いて開発したWebアプリケーションの場合、これまでは動的解析、つまり、アプリケーションを実行してその振舞いを観察することで不整合を検出していた。しかし、この手段では、不整合が複数あったとしても

1箇所ずつしか検出できず、不整合の数だけ実行を繰り返す必要があった。そこで静的解析技術を応用した整合性チェックツールを開発し、これを用いることによって実行を必要とすることなく、まとめて不整合を検出できるようになった。

それでは以下に、オブジェクトワークスの概要、オブジェクトワークスにおけるソース間の関連、オブジェクトワークス整合性チェックツールの仕組みを説明する。

①オブジェクトワークスの概要

整合性チェックの具体的な説明の前に、まずオブジェクトワークスの概要について述べる。

図4はオブジェクトワークスの実行時の動きおよび開発時に作成する定義の概要を示している。

a 開発の流れ

オブジェクトワークスを用いたアプリケーションの開発の流れを簡単に紹介する。開発者は最初に画面遷移および画面遷移時の処理を設計し、画面遷移定義シートを記述する（ア）。

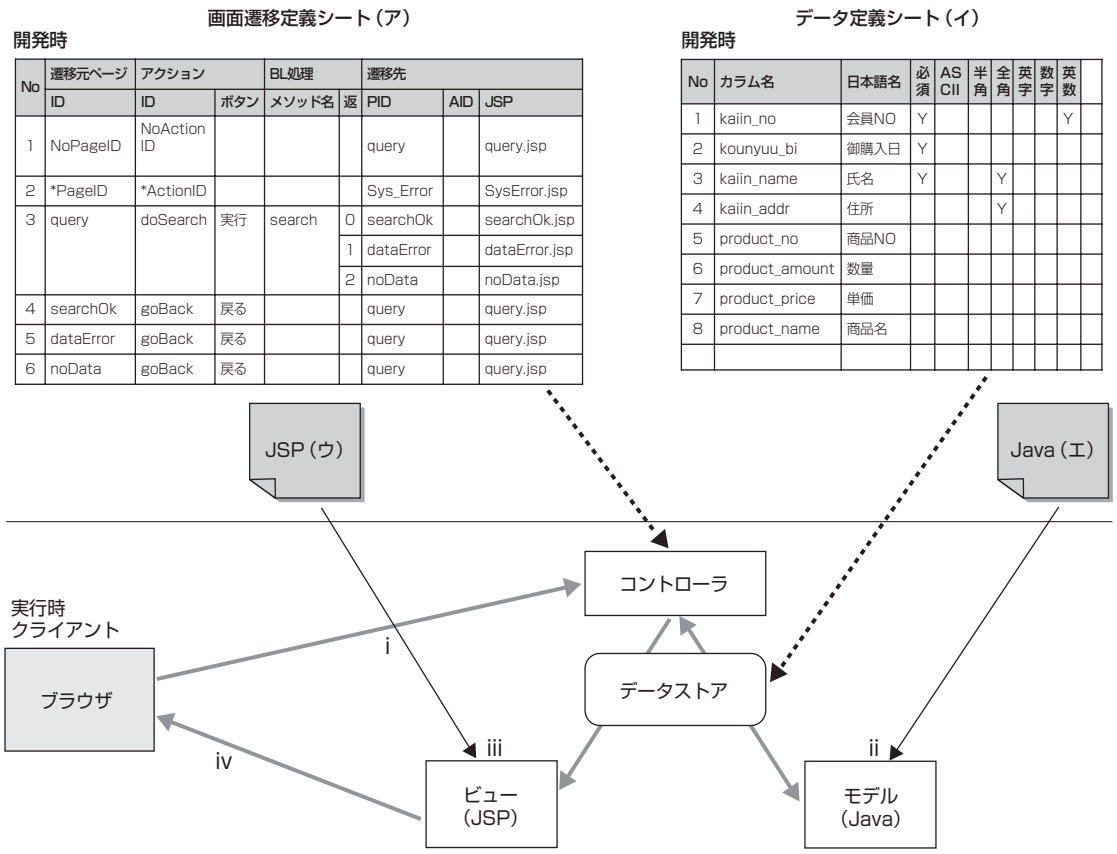


図4　オブジェクトワークスアプリケーション

次に、各画面で使用されるデータ項目にユニークなカラム名を付け、データ項目の属性と共にデータ定義シートに記述する（イ）。

最後に、画面（ウ）をJSPで、画面と画面の間で行われる処理（エ）を、今回の例ではJava言語で作成する。

両定義シートからオブジェクトワークスのツールを用いて定義情報XMLファイルを生成し、BLをコンパイルしたclassファイル、JSPを合わせてWebアプリケーションサーバに配置すると実行が可能となる。

b 実行の流れ

オブジェクトワークスはMVCモデルに基づいたフレームワークであり、アプリケーションの実行の流れは図4のようになる。

ブラウザからコントローラにリクエストが送られる。リクエスト内容を画面遷移定義シートの記述と照合し、適切なBLを起動する。この時リクエストに含まれるデータは、データストアに格納してBLに渡される。BLの処理結果に基づいて、画面遷移定義シートの記述に従い、適切なJSPを選択し、画面を作成する。この時もデータはデータストアを用いて渡される。作成された画面をブラウザに返す。

コントローラに当たる部分はオブジェクトワークスの部品として提供されていて、画面遷移定義に基づいて動作する。データストアと呼ぶ連携用オブジェクトについてもあらかじめ部品として提供されており、開発者が作成したデータ定義シートに基づいて動作する。オブジェクトワークスの特長は、コントローラの動きとシステム内で使用するデータをコーディングではなく設計情報を用いて定義する点である。これにより、画面遷移や各所で使うデータ項目を柔軟に変更することができるようになっている。

②オブジェクトワークスにおけるソース間の関連

①で述べたようにオブジェクトワークスを利用する際には画面遷移定義シート、データ定義シート、JSP（画面）、Java（BL）の4つを作成する。

ソース、定義間の整合性が守られていれば、開発者が定義シートを作成するだけでモデル、ビュー、コントローラ間連携のためのコーディングが不必要であり、アプリケーションがシンプルな構造になるという利点がある。

オブジェクトワークスのソース、定義間で満たされるべき整合性を以下に示す。

a 画面遷移定義シートとJSP

オブジェクトワークスは、図5のように記述して画面遷移を行う。"ACT_doSearch"と記述した場合、このボタンの押下イベントが発生すると、アクションIDとしてdoSearchが設定されてサーバ処理が行われる。表示されているページIDと、JSPに記述されたアクションIDの組み合わせが画面遷移定義シートに存在するとき、その行の記述に基づいてBLの起動および次画面への遷移が行われる。

b 画面遷移定義シートとJava

ユーザのアクションが起こったとき、画面遷移定義シートに記述されている名前で、返り値、引数がオブジェクトワークス形式であるJavaのメソッドが起動される。

c データ定義シートとJSP

JSP内でデータ項目を使用するときは、使いたいデータ項目のデータ定義シートにおけるカラム名を用いる。カラム名を指定してデータを取得するメソッドが用意されている。

d データ定義シートとJava

Java内でデータ項目を使用するときは、使いたいデータ項目のデータ定義シートにおけるカラム名を用いる。カラム名を指定してデータを取得するメソッドが用意されている。

定義シートに記述されたカラム名、メソッド名等は、プログラム中では文字列として記述される。スペルミス等で不整合が生じたとしてもコンパイル段階では検出できない。

③オブジェクトワークス整合性チェックツールの仕組み

カラム名等の両定義シート内の各種IDを用いることによりMVC連携については単純化されるが、そのための記述はプログラム中で文字列として処理される。そのために上述

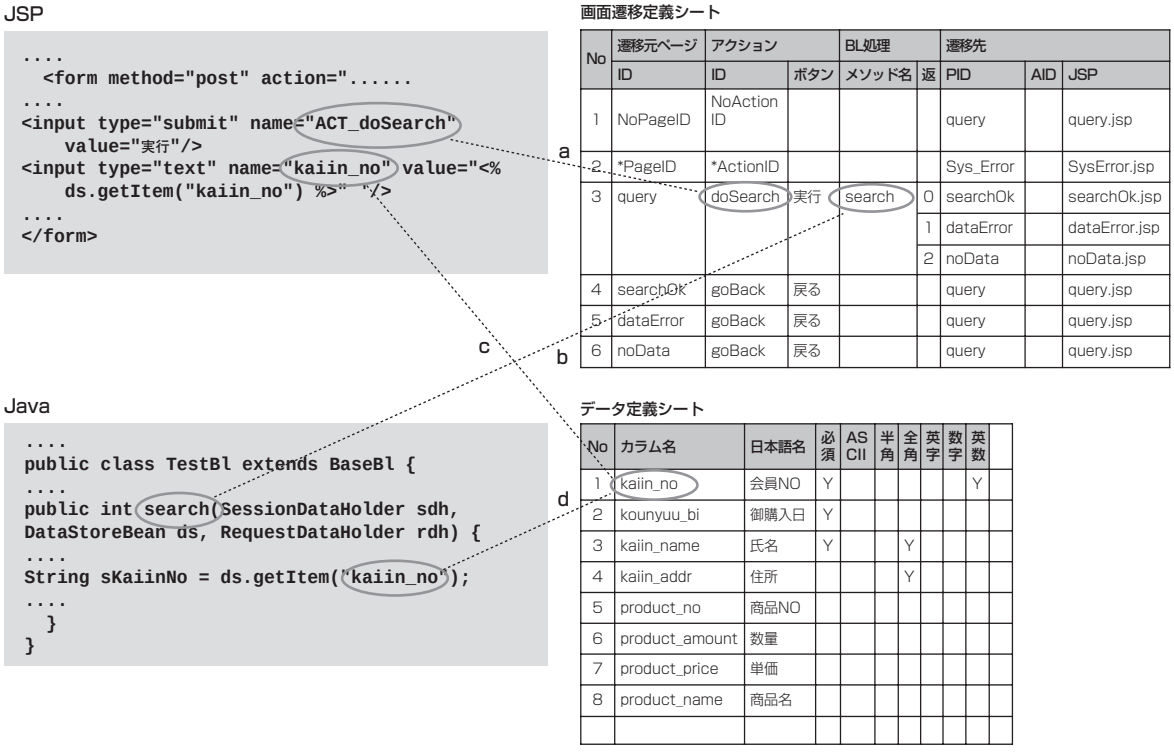


図5 オブジェクトワークス ソース間関連図

のような関連にミススペル等で不整合が生じたとしてもコンパイラ等では検出されない。そこでカラム名等のIDを静的解析技術により抽出し、チェックを行う仕組みを開発した。オブジェクトワークスのカラム名およびアクションIDを例にとって見てみる。カラム名およびアクションIDは以下のように抽出を行う。

a カラム名の整合性

データ定義シート、JSP、Javaのソースコードに記述されたカラム名間の整合性を調べるために、ソースから内部で使用されているカラム名を抽出する必要がある。

データ定義シートからの抽出は、データ定義シートから生成したXMLファイルから行う。

JSPからの抽出は、JSPをWapidで処理して行う。オブジェクトワークスでは入出力項目について、INPUTタグのname属性をカラム名と結びつけることによってバインディングしている。Wapidで処理されたリポジトリから、INPUTタグのname属性一覧を作成する。

また、JSP内での表示項目については、DataStoreオブジェクトから出力用メソッドを用いてデータ取得を行っている。JSP内から、DataStoreオブジェクトの出力メソッドを取り出し、メソッドの定義に従って、カラム名が受け渡される引数の位置にある記述を取得し、一覧を作成する。

BL（Javaソース）からの抽出は、BLをJapidを用いた解析結果から行うことで、正確に効率的に実施できる。

BL内での入出力項目については、DataStoreオブジェクトから入出力用メソッドを用いてデータ取得を行っている。BL内から、DataStoreオブジェクトの入出力メソッドを取り出し、メソッドの定義に従って、カラム名が受け渡される引数の位置にある記述を取得し、一覧を作成する。

b アクションIDの整合性

画面遷移定義シート、JSPに記述されたアクションIDの整合性を調べるために、ソースから内部で使用されているアクションIDを抽出する必要がある。

画面遷移定義シートからの抽出は、画面遷移定義シートから生成したXMLファイルから行う。

JSPからの抽出は、JSPをWapidを用いた解析結果から行うことで、正確に効率的に実施できる。オブジェクトワークスではアクションIDについて、記述法を複数用意しているが最も一般的な記述を例に説明する。

一般的には、typeがボタンまたはサブミットボタンのINPUTタグのname属性にアクションIDを記述する。その時、アクションIDに"ACT_"を前置するのが決まりである。

Wapidの解析結果から、typeがボタンまたはサブミットボタンのINPUTタグのname属性を取得し、先頭の"ACT_"を取り除いたものの一覧を作成する。

上記作業で作成された一覧同士でマッチング処理を行い、整合性をチェックすることが

可能となった。

解析処理を行うことによって、単に文字列比較によりカラム名やアクションIDを抽出するのに比べて、精度が格段に向上した。

④Strutsの整合性チェック

Strutsを用いたWebアプリケーションの場合、開発者はStruts-config.xml、ActionFormBean(Java)、JSP、ActionClass(Java)をソースとして作成する必要がある。Struts-config.xmlは画面遷移を含むWebアプリケーション全体の動作を定義するソース、ActionFormBeanはMVC間でデータを受け渡しするためのオブジェクトの定義、JSPは画面、ActionClassはBLまたはBLを起動するメソッド群である。4つのソースについてもオブジェクトワークスと同様の関連性があり、同様の仕組みで整合性をチェックすることができる。

(3)単体テスト支援ツール

静的解析技術のもう1つの応用例として、オブジェクトワークスを使用したWebアプリケーションの開発過程におけるPL部の単体テストを支援する仕組みについて研究を行っている。

①活動概要

2004年度についてはテストケースの自動

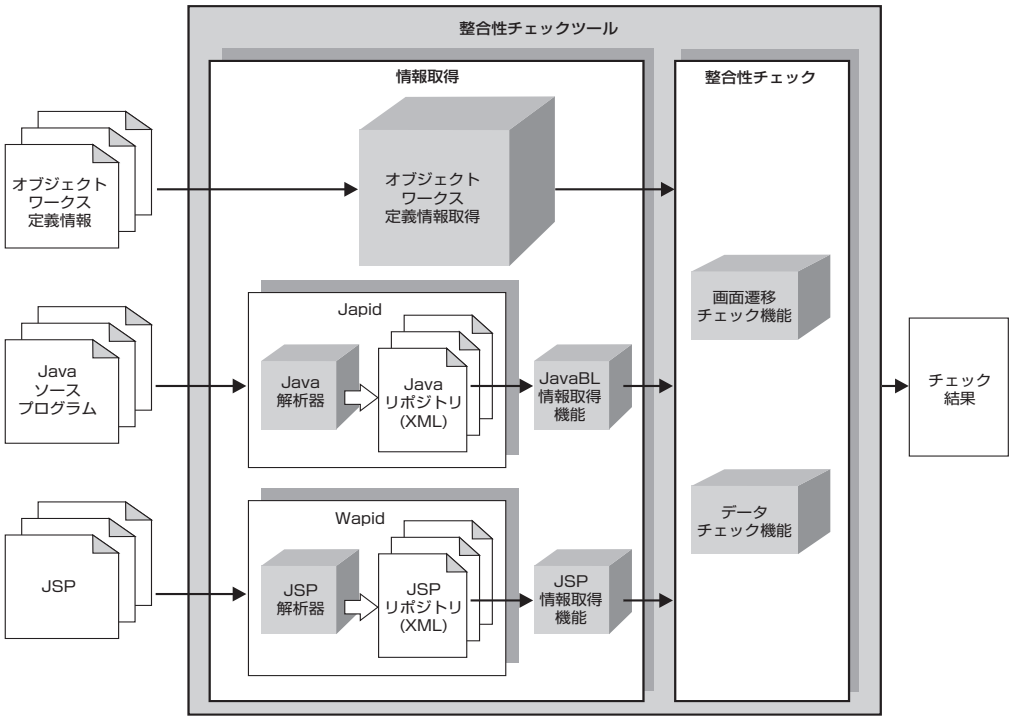


図6　整合性チェックツール

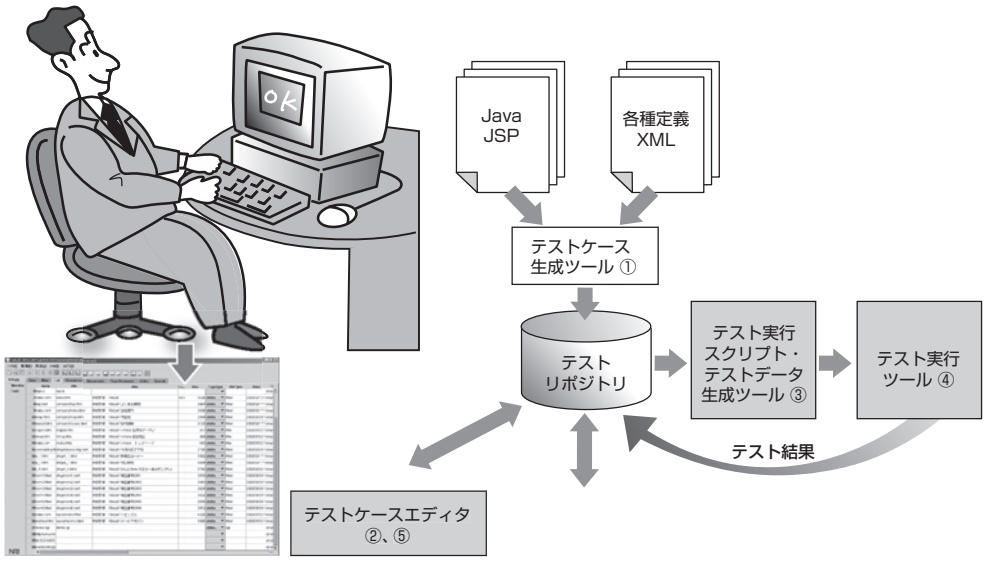
生成ツール、テストケースおよびテスト結果を一括管理するエディタ、ケースからテスト実行スクリプト及びテストデータを生成するツールを試作し、実際のアプリケーションに適用して効果を測定した。本研究で開発したテストツールの概要を図7に示す。

本ツールにより、従来手作業で行っていた作業が自動化され、テストに費やしていた工数を大幅に削減できる。この中で、テストケース生成ツールに静的解析技術を用いている。

以下では静的解析技術を使ったテストケース生成について述べる。

a PLの単体テストと問題点

PLのテストでは、例えば、「数値項目に対して英字を入力できない」とか、「8桁限定の入力項目に対して正しく全桁表示されなければならない」とか、「日付入力項目で実際にはない日付（2月30日等）が入力されたとき、エラーダイアログを表示しなければならない」といったことを確認する必要がある。通常は、このようなテストケースを全画面全項



ツール機能

①テストケース生成
Java、JSP、各種定義ファイルを基にテストケースを自動生成、テストリポジトリに保存

②テストケース編集
自動生成され、リポジトリに保存されたテストケースの確認、編集

③テストスクリプト・テストデータ生成
テストケースから、テストを実行するためのスクリプト及びテスト対象のデータを自動生成

④テスト自動実行
生成されたスクリプトとデータを連携してテストの自動実行、結果をテストリポジトリに保存

⑤テスト結果確認
開発者が実行結果を確認し、合否判定を入力

⑥テスト結果集計
テストの合否を集計

図7　テストツールの概要

目について用意する必要があり、非常に手間がかかっている。

b 静的解析とテストケース生成

以下の処理を行うことによってテストケースを自動生成することができる。

画面遷移定義から対象となるJSPファイルを取得する。

JSPファイルをWapidで処理し、カラム名一覧を作成する（表1）。

カラム名
kaiin_no
kounyuu_bi
.....

表1　カラム名一覧

データ定義シートからカラムに関する属性を抽出し、カラム名一覧にマージしてカラム名・属性一覧表を作成する（表2）。

カラム名	型	チェック	
kaiin_no	string	半角英数	桁数:8
kounyuu_bi	string	日付	
.....			

表2　カラム名・属性一覧

カラム名・属性一覧の情報を基にして必要なテストを決定し、テストケースの生成を行う。

Sapidの静的解析技術を用いることにより、ソースコードからの情報取得を高精度で行うことができるようになった。

c 実験結果

2004年度に各機能の設計/試作を行い、実アプリケーションへの適用実験を行った。結果、オブジェクトワークスに関連するテストケース100%の自動生成に成功した。

開発時に実施されたテストケース数	493件
上記のうち、オブジェクトワークス関連	340件
今回ツールで生成されたテストケース数	340件
充足率	100%

表3　テストケース生成結果

（4）今後の予定

今年度は、各機能の連携の強化、複数開発者の同時使用に耐えうる管理機能の開発、実プロジェクトへの適用実験を行い、実用に耐えうる統合テスト環境の構築を行う予定である。

そして、次年度以降は、静的解析技術とリポジトリシステムを応用した、Webアプリケーション開発の簡易化を行う予定である。

5. Webサイト全体像俯瞰の高度化技術

（1）Ridual

次にJavaScript解析機能のWebサイトデザインへの適用について述べる。

まず、JavaScript解析機能を組み込む対象であるRidualについて説明する。

Ridualとは、「Rapid Information Development + Visual」の頭文字を組み合わせた造語である。今後のWebサイト開発を高速化することを目的に開発されたツールであり、基本的

に既存ツールがカバーしていない領域に特化した機能を提供している。

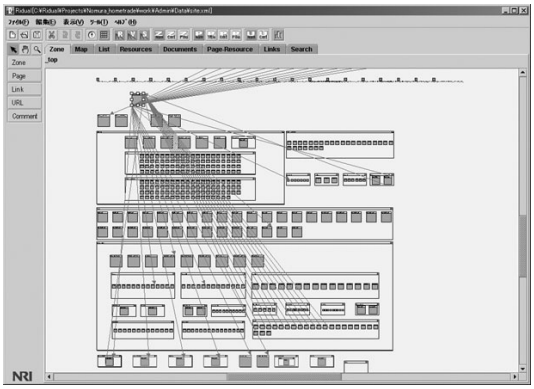


図8　Ridualのユーザインターフェイス（ZonePanel）

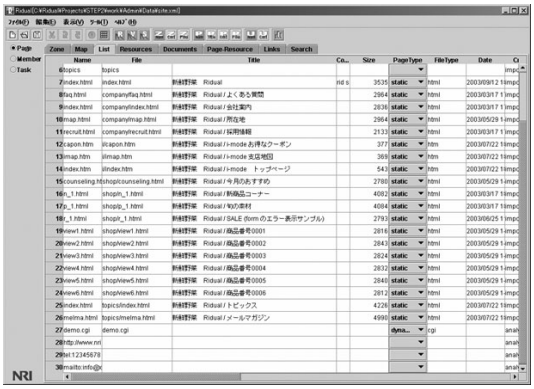


図9　Ridualのユーザインターフェイス（ListPanel）

①XML化、あるいは正規表現

Ridualプロジェクトでは、現状のWebサイト（Webアプリケーション）開発における最大の課題を、「全体像の俯瞰性の低さ」と仮定している。

個々のページや、個々の機能についての、デバッグや開発ツールは現状では様々なベンダーから提供されている。しかし、サイト全体や提供機能全体という立場から、利用者の利便性を確認するツールはほとんど存在しない。その結果、個別の機能やコンテンツの構築に関しては、それなりに標準化されたノウハウが蓄積されているが、サイトの全体像を把握する段になると、途端に属人的な要素が強まる。多くの機能やコンテンツの重なり具合の調整（これがWebサイトの全体像になる訳だが）は、ディレクタと呼ばれる人の頭の中で行われているのである。

そこで、Ridualプロジェクトでは、その頭の中で構築されているWebサイトの全体像を、他人とも共有できる「データ」の形にすることにした。他人に手渡しできたり、比較できる「データ」であれば、標準的な育成プログラムも実現できると考えたからである。幸いなことに、Webサイトには「トップページ」という概念があり、複数の入り口を想定しているサイトであっても、その中に優劣が付けられる場合が多く、基本的には「ツリー構造」の形態といえる。これは、情報の遷移やナビゲーションを考慮する際に、一番自然な形であるためと考えられる。そして、それをデータ化するには、「XML」の形が必要十分だと考えた。

親子関係にあるページ群があり、ページにぶら下がっているようなリソース群（イメージファイルやスタイルシート等）がある。こうしたファイルの関係性をXML化した。その際に、新規サイト構築時と既存サイト解析時で、その関係性を同じ扱いにすることに注

意を払っている。HTMLの習得において、既存サイトのソースを見ることが最良の方法であったように、サイト構造も既存のものを見ながら、習得し、構築できるようにするためである。Ridualプロジェクトの最大の成果は、後述する解析能力や情報の視覚化ではなく、実はこうしたサイトをデータ化するという、一種の正規化することに成功したことにある。現在の表現（XMLのDTD［Document Type Definition］）が十分なものかどうかは、今後の検討課題ではあるが、まったく別のWebサイトの構造を比較することが可能となった。Webサイトを語る上で、主観的ではない「モノサシ」ができたことの意義は小さくないだろう。

②HTML解析

次に、Ridualの特性として、解析能力がある。HTMLやSWFファイル（Macromedia社Flashファイル）を解析して、ファイルの親子関係やformタグのパラメータやmetaタグ情報等をデータ化する。この意味は2つあり、1つは上述の正規化のための情報収集であり、2つ目は開発作業効率の向上である。ページ間のリンク関係の解析において、Ridualはある仮定を設定している。「類似の情報は、ファイル構造上も類似の場所に保存される」という点である。つまり、同じような情報群は同じフォルダ（ディレクトリ）に整理されていることが多い。さらに、その情報を見せる場合には、何かしらの「玄関口」にあたるページが用意されていることも一般的だろう。簡単に言えば、類似情報の塊はフォルダという形でまとまっていて、その中には「index.html」のような情報トップ画面が用意されている。この仮定をベースにすると、ツリー構造が容易に記述できる。後述する情報視覚化が、それなりに人間が行ったものと似通っているのは、ここに鍵がある。

そして、現実のWebサイト開発において一番時間が取られるのは、直接的な開発作業ではなく、間接的あるいは補助的な作業である場合が多々ある。アイデアを煮詰めたり、凝ったレイアウトを作成したり、DBのチューニングをしたりするよりも、ページ一覧表を作成するためにExplorerからファイル名のコピーペーストをしたり、DB情報をExcelに書き写したりする作業の負荷が馬鹿にならないほどある。特に、ドキュメント作成のための情報収集作業。しかも、最新情報をまとめなければいけない作業である。こうした情報収集は、本来人間が単純なミスを犯す危険性を持ちつつ行うべきではない。開発中の状態であっても、作り込んだモノからプログラム的に抽出する方が正確である。正確な情報収集こそが、高品位なWebサイトやWebアプリケーションを開発する近道である。

このようにして収集した情報を、利用シーンに合わせて加工しやすいように、データ変換を行うことが、人間の手間を軽減することになる。Ridualは、表形式で表示するデータはCSV形式で、後述する視覚化情報はSVG

(Scalable Vector Graphics) 形式で外部ファイル化でき、情報の再利用性を高めている。こうした単純作業から、開発メンバを解放することが、直接的な作業に集中できる環境を生み、それがより信頼度の高いWebシステムの開発に繋がるだろう。解析可能なHTMLタグについては、以下を参照のこと。
http://www.ridual.jp/function/ability.html

なお、解析機能の実装で一番苦労したのは、古いHTMLと規格外のHTMLの存在である。ブラウザのレンダリングエンジンに許容度がありすぎて、HTMLの仕様に準拠しなくてもある程度正常に表示されてしまう。その結果、一般的なサイトの解析を行うと、そのほとんどが文法的なエラーを抱えている結果になる。それでもブラウザ上では閲覧可能な形で稼動しているので、解析エンジンにもそうしたファジーさを残さなければならなかった。

③情報視覚化

そして、情報の受け取り方にも着目した。膨大な解析データをそのままの形で渡されても、それを読み解くのにスキルが必要になってしまう。それでは、間接的な作業が増えるだけなので元も子もない。そこで、多くの情報を俯瞰できる方法として、視覚的な表現方法を模索した。

現段階では、3つのサイトマップを提供している。ツリー構造を上から下へ階層的に並べていく「Hiermap」、左から右へと展開される多少ページを斜めに描いた「Sitemap」、そして同心円状にトップページからのクリック数に応じて並べた「Circlemap」。

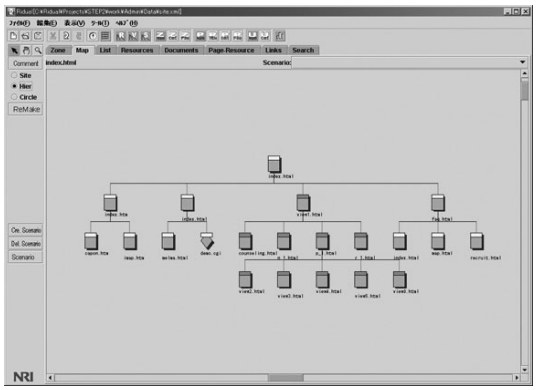


図10　Hiermap

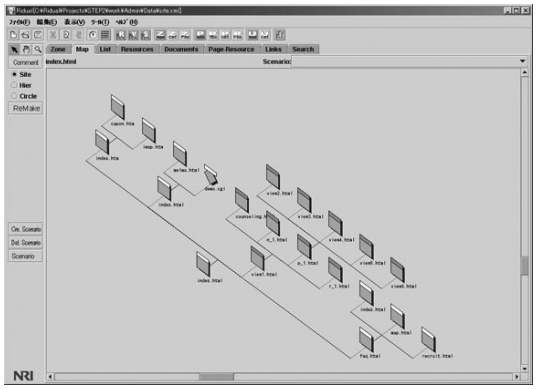


図11　Sitemap

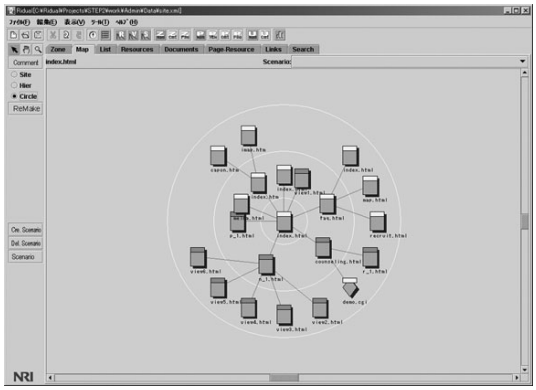


図12　Circlemap

このサイトマップ作成については、メジャー既存ツールにない工夫を施している。Webサイトは、ページ間で相互にリンクが張られているので、それを正確にサイトマップで描こうとすると、同じページが何度も出現してしまう。しかし、サイトの構造を考える上では1つのページは1度きりしか出現して欲しくない。そうでないと情報整理が複雑になってしまうからである。そこで、Ridualでは、ページ間のリンク関係を親子関係に置き換え、親をユニークに選出するアルゴリズムを採用した。親を一意に限定してしまうリスクは存在するが、リンクを実リンクと論理リンクの2種類に分けて、開発者が任意にリンクを移し変える自由度を残した。その結果、人間が最上流工程で描くようなサイトマップが自動描画されるようになっている。

実際、サイトのリニューアル提案をする際に、多くのドキュメントよりも、このRidualが生成した1枚のサイトマップが、クライアントの心を捉えた例もあり、直感的でわかりやすい視覚化情報がもたらす生産性向上率はかなり大きい場合がある。

(2) Ridualの機能と対象ユーザ

①機能

Ridualは、Webサイトの構造をGUI的に記述でき、サイトの全体像を検討しながらの設計が可能である。ページ内デザインに進む前に、サイト全体のナビゲーション等の検討を進めることができる。雛形的な仮のHTMLを生成可能で、内部の詳細なデザインは既存のHTMLエディタを呼び出すことにより開発を可能としている（Generatorの機能）。既存サイトの解析のためには、まず対象サイトのファイル群をローカルディスクに収集することから始める。ダウンローダに、既存サイトのURLを入力することで、そのサイトの構成ファイルを入手できる。こうして入手したものや、他ツール等で自作しているサイトを、解析機能（Analyzer）を用いてデータ化する。

②対象ユーザ

a アナリスト：ダウンローダを用いた競合分析やアクセスログをサイトマップ上に視覚的に重ね合わせることを用いてサイト解析結果をわかりやすくレポートしやすくなる。サイトの規模感や使用されているリソース一覧、リンク切れ一覧などの作成に活用可能。

b 初心者：Webサイト開発における要の部分である、ページ間のナビゲーションを体験的に習得できる。自作用ツールとしても、既存サイトの解析用ツールとしても活用可能。ページ内のデザインに踏み込む前に、サイト全体の構成を詰めることができる。

c クライアント：納品されたサイト一式のリンク切れ等の簡易チェックが可能。リソース一覧やページ一覧など、従来時間をかけてデザイナに依頼していたタスクを自動で処理できる。

d デザイナ：サイト構成の組み立ての試行錯誤から、構築中のデバッガ的な使用、競合サ

イトや参照すべきサイトの解析などにも活用できる。既存ツールとの連携を重視しているので、従来のワークフローにそれほど大きな変更を加えることなく導入が可能。アイデアの実装という要の部分に最大限の時間をかけることができるように、間接的業務（（単純情報収集作業など）の自動化支援も有効。

e エンジニア：デザイナとの協労をサポート。ページ内に組み込まれたFormタグ関係の情報を一覧表にして取り出し、CGIやデータベース設計との比較検討が可能。また、修正がたび重なるサイトマップを自動でアップデート（生成）。コーディング担当している部分だけではなく、サイト全体のそのタスクの位置付けを意識しながら開発を進めることを支援。プロジェクトに関わる情報をPIP（Project Information Portal）と呼ぶサイトに集約して、情報の拡散も予防。

（3）JavaScript解析技術の組み込み

①課題

Webサイトの情報視覚化に有効なRidualではあるが、次のような課題があった。

Ridualは、ファイルの親子関係を中心に解析や情報の視覚化を行ってきた。ファイル間の関係性だけが対象と言っても過言ではない構造を持っていた。しかし、多様化するWebサイトは、ファイルのそうした関係性だけでなく、プログラム的な側面を持ち始めていた。特に、ブラウザ依存性への対処や、Flashなどのプラグイン対応、保守性を重視した関数の領域などにおいて、JavaScriptが多用され、その論理的解析の必要性が高まっていた。ファイルの親子関係だけでは、Webサイトを表現することが困難な時代に入ってきたとも考えられる。しかし、JavaScriptの解析は、単なるファイルのURL情報の抽出だけではなく、変数に格納されたURLの断片の検出や、document.write構文によって可能な再帰的なHTMLとの組み合わせ解析など、正確さを期すほど技術的な難易度が高く、Ridualとしては未着手な状態であった。

ここに名古屋大学と和歌山大学で開発されたJavaScript解析機能を適用した。

②結果

組み込みに関しては、まだ解析系（Analyzer）のみであり、様々な点で実験室レベルに留まっているが成果は上がっている。JavaScriptの解析がなされることで、サイトの全体像の把握が飛躍的に進むケースがあることが実証された。

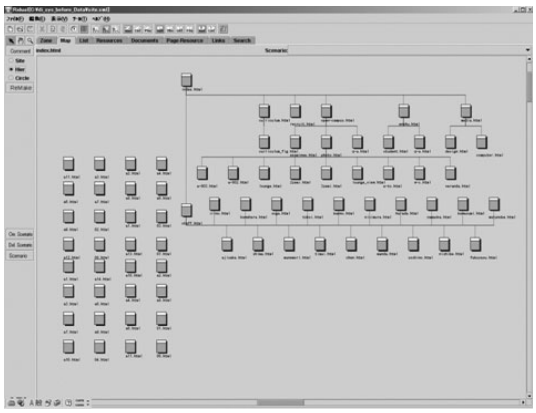


図13　JavaScript解析技術組み込み前の解析結果

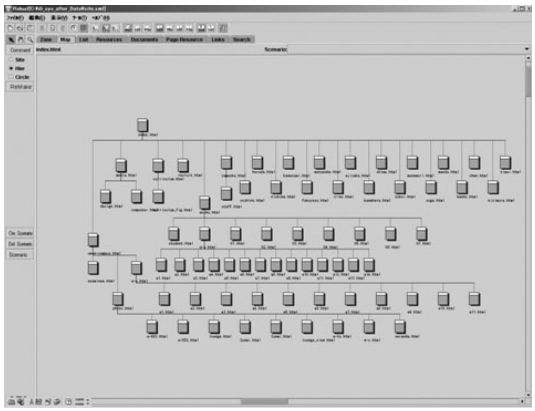


図14 JavaScript解析技術組み込み後の解析結果
http://www.sys.wakayama-u.ac.jp/di/

さらに、JavaScriptをポップアップウィンドウなどに用いる場合、Webサイト内の機能ごとに分割された「小サイト」の連結部分にそれらが活用される場合が多い。そうした場合には、全体のコード量のうちのJavaScriptが占める割合以上に、検出可能となったファイル群は多くなる傾向があり、組み込みによる解析能力の向上は数パーセントのレベルではなく数割から数倍のレベルに達することが予想できる（実際の解析率は、実際のサイト構造を知り得る場合にのみ算出できるので、既存サイトでの実測は第三者的には不可能であり、実施していない）。

特に、この解析能力の向上は、複雑に作り込まれたWebサイトほど顕著に効果が見えるものであり、そうした複雑に作り込まれたWebサイトほど欠陥が潜んでいる可能性が高く、こうした解析ツールの必要性が高いと予想される。

今後は、ダウンローダへの組み込みや、パフォーマンスの向上、サーバ化の計画やインストールの簡易化など製品化に向けての取り組みが検討されている。

（4）方向性

こうした解析系のエンジンは、通常は開発工程におけるデバッガとしての用途に活用されがちだが、Ridualは現状ではリアルタイムの修正機能を持っていない。

それは、研究開発プロジェクトとして生まれた時点で、安全性確保のため、解析対象ファイルに対するread権しか許さないポリシーにしているためである。しかし、その後もそれを堅持したのは、RidualのWebサイトの総合レポーティングツールとして活用を考慮したためである。複雑化かつ大規模化が進むWebサイト開発において、常に「本物（ソース）」を見なくても全体像を把握できるということには大きな意味があるだろう。

解析エンジンとしてのRidualの今後の方向性としては、少なくとも解析対象を増やしていくことを検討している。静的なHTMLやswfの解析、そしてロジック部分であるJavaScript。現時点で重みを置いているのは、スタイル系であるCSS（Cascading Style Sheet）と、コンテンツ自体の解析である。

元々、コンテンツとスタイルの分離こそが、健全なWebサイトの発展に寄与すると言われ続けてきた。体系的なスタイル定義とそのデバッグは今後益々重要になってくるし、適切な文言が使われているか（どちらかという

と不適切な言葉の検出）という解析も必要になってくると思われる。後者の場合は、データマイニング系の技術との融合や、ニュースサイトなどへの適応によって社会調査的な応用も可能と考えている。

今後のWebサイト開発では、メンテナンス性と開発生産性の双方を高めた技術がトレンドとなって行くと予想している。複雑化と巨大化と短工期化、低予算化の中で開発体制を維持できるのは、そうした道しかないからである。

その路線上で重要になってくるのは、実は解析技術であることが予測されるし、事実そうしたマーケットが定着しつつある。そして、解析結果をわかりやすく共有していく技術の重要度も高まっている。JavaがJavaDocによって、開発陣を仕様書ドキュメント作業から解放したことは生産性に寄与している。同じように、Webサイトそのものからドキュメントが出力できたなら、かなりの生産性向上に繋がると思われるからである。そして、このドキュメント（レポーティング）化というプロセスの中に、最初に述べた、正規化と解析と情報視覚化の三要素が混在している。Ridualは産学連携プロジェクトを通してさらにこれらの可能性を追求して行こうと考えている。

6. 終わりに

以上、我々が取り組んでいる高信頼Webシステム開発技術への取り組みを報告した。

Webアプリケーション整合性チェックツールは、既にオブジェクトワークスR6.5に組み込まれてリリースされている。単体テスト支援ツールも、実用化に向けた機能充実に取り組んでいる。Ridualについても、Webサイトの解析対象の拡大という課題にJavaScript解析機能が適用できることが確認された。

このように言語解析技術は、プログラムの検査や、ドキュメンテーション、テストの効率化などに大いに有効である。しかし、この技術は情報科学に関する高度な専門性を必要とする。アプリケーションシステム開発を主とする企業がこのような技術を継続的に研究開発するのは、難しい点がある。

今回のように大学から企業で適用しやすいソフトウェアの形で言語解析技術の提供を受けられたこと、及び適用にあたり様々な指導・助言を得られたことは、非常に有意義である。

大学で行われる研究成果をビジネスに結びつける試みは、今後ますます重要になるだろう。我々も引き続き、産学連携の特徴を活かした技術開発に取り組んでいきたい。

◆参考URL◆

［1］Sapidのホームページ
http://www.sapid.org/
［2］オブジェクトワークスのホームページ
http://works.nri.co.jp/
［3］Ridualのホームページ
http://www.ridual.jp/