{ 第1部 } 高い生産性を持つ高信頼ソフトウェア作成技術の開発＊ 7

高信頼WebWare生成技術：WebWareのテスト・解析・作成支援

桑原　寛明 ＊1　金子　伸幸 ＊2
渥美　紀寿 ＊3　山本晋一郎 ＊4
阿草　清滋 ＊5

＊1 立命館大学情報理工学部
＊2 （株）ネットレックス
＊3 南山大学数理情報学部
＊4 愛知県立大学情報科学部
＊5 名古屋大学大学院情報科学研究科

WebWareプロジェクト

今日，さまざまなサービスがWeb技術を利用してインターネット上で提供されている．商品の購入，切符の予約，銀行業務など多岐にわたるサービスが我々の生活のさまざまな場面で利用されており，一種の社会的インフラストラクチャとしての位置を占めつつある．そのため，その信頼性を確保することが社会的にも重要な課題となっている．

これらのサービスは大規模でヘテロジニアスな分散ソフトウェアとして実現されており，HTMLやXMLなどのコンテンツ記述，レンダリングのためのCSSやXSLT，JSPやServletなどのサーバ側プログラム，JavaScriptに代表されるクライアント側プログラムなどの種々の技術から構成される．我々はこのようなWeb技術を利用したヘテロジニアスなソフトウェアをWebWareと定義し，高信頼WebWare生成技術プロジェクト（以下，WebWareプロジェクトと呼ぶ）においてWebWareの信頼性と安全性の向上を目標として研究開発を行ってきた．

WebWareが実現する機能の高度化に伴い，WebWareを構成する各要素技術の機能も向上し要素間の関係は複雑化している．しかし，構造や意味まで含めてWebWareの全体を記述する枠組みが欠けており，個々の要素がどのような制約を満たすべきかが表現されない．また，システムのバックエンドを開発するエンジニアとフロントエンドを開発するデザイナが協調して作業を行うことはなかなか容易ではなく，WebWareを構成するプログラム群とプログラムが埋め込まれる文脈としてのコンテンツおよびプログラムが出力するコンテンツ間の整合性の確保が難しい．これらのことがWebWareのデバッグ，性能維持，保守を難しくしており，信頼性の低下へとつながっている．

WebWareプロジェクトではこれらの問題に対処するために，WebWareの信頼性と安全性を保証しつつ，デザイナのレンダリング・エディトリアル作業とエンジニアのシステム構築作業を高度に統合するWebWare開発環境の構築を行った．WebWare開発環境は

- WebWareテスト支援システム
- WebWare解析系と細粒度リポジトリ
- WebWare作成支援システム

の3つのサブシステムからなり，各サブシステムは複数のツール群によって構成される．本稿ではいくつかのツールを取り上げてそれぞれのサブシステムについて紹介する．

産学連携の効果として以下のことが挙げられる．WebWareプロジェクトの成果の一部はオブジェクトワークス☆1やInterstage☆2などのプロジェクト参加企業の製品に組み込まれている．大学と企業が共に議論を行い問題設定と研究開発の方向性を決めることで，企業の実際のニーズに大学の持つシーズをうまくマッチングさせることが可能となった．また，企業の技術者が開発をマネジメントして品質管理や文書作成を行うようにしたため，開発されたツールについて製品レベルあるいは製品に近いレベルの品質を達成することができた．

WebWareプロジェクトの成果はプロジェクトのWebサイト☆3において公開されている．また，一部の成果についてはSourceForge.JP☆4において管理されている．

WebWareのテスト支援

多くのWebWare開発ではStrutsなどのMVC（Model-View-Controller）アーキテクチャに基づくフレームワークが利用される．フレームワークがModel，View，Controllerを連携させる．このようなソフトウェアのテストでは，連携させる前に個々のコンポーネ

☆1 http://works.nri.co.jp/
☆2 http://interstage.fujitsu.com/jp/
☆3 http://www.agusa.i.is.nagoya-u.ac.jp/research/webware/
☆4 http://sourceforge.jp/projects/ridual/

ントについてテストを行いたい．ビジネスロジックを担当するModelについては従来の単体テストの手法を適用することができる．しかし，ユーザインタフェースとなるViewのテストには，ユーザ入力に対するバックエンド処理の結果が必要である，Webブラウザを利用して大量のテストデータを入力しなければならない，複数のページ群に対して一連の操作を行う必要がある，といった問題がある．

WebWareのテスト工数を削減するためには，テスト設計やテスト作業を自動化したり，静的検査を行ってテスト以前にバグを検出することが必要である．さらに，闇雲にテストを行うのではなく，テストの十分性を評価するための基準を設けて行うべきテストの集合を明らかにして不必要なテストを削減することが重要である．以下ではWebWareのテスト支援として，Viewの実装に広く利用されているJSPを対象とする単体テスト支援フレームワークとテストケース生成技術を紹介する．

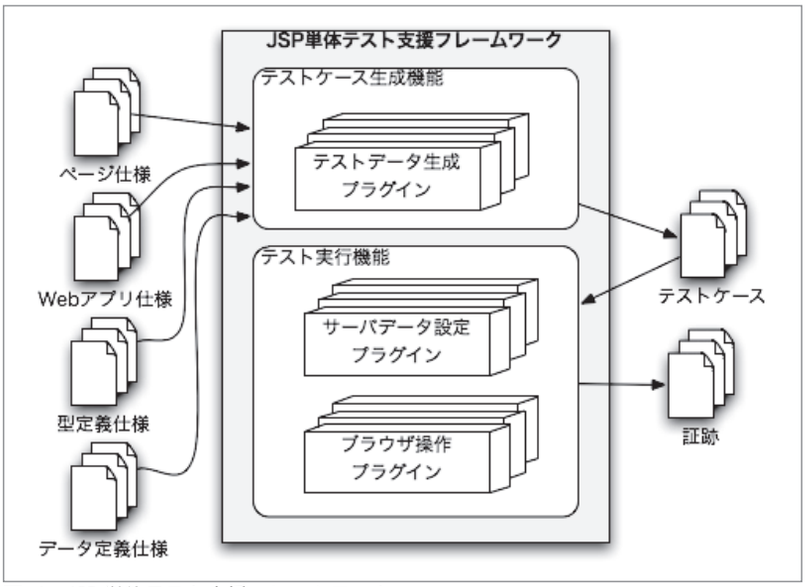

図-1　JSP単体テスト支援フレームワーク

【JSPの単体テスト支援フレームワーク】

WebWareプロジェクトでは，多くのフレームワークでViewを実装するために利用されているJSPに対し，その単体テスト工程の一部を自動化するJSP単体テスト支援フレームワークを開発した（**図-1**）．JSP単体テスト支援フレームワークはテスト支援として(1)テストケース生成，(2)テスト自動実行，(3)証跡保存の自動化を行う．JSP単体テスト支援フレームワークは，JSP単体テスト支援において特定の実装や戦略に依存する個所をホットスポットとして差し替え可能となるよう設計されている．

JSP単体テスト支援フレームワークは，ページ仕様，Webアプリ仕様，型定義仕様，データ定義仕様として記述された情報からテストケースを生成する．各仕様では以下を定義する．

- ページ仕様
  テスト対象アプリケーションのページを定義する．ページ生成の際に参照されるデータ，フォーム項目などを定義する．
- Webアプリ仕様
  テスト対象アプリケーションが利用するフレームワークやテスト対象となるページを表示するためのプロセスを記述する．
- 型定義仕様
  テスト対象アプリケーションで利用されるデータの型を定義する．型定義では，型の名前と型に付与可能な制約を定義する．
- データ定義仕様
  テスト対象アプリケーションで利用されるデータを定義する．データ定義では，データの名前とデータが属する型，およびデータに付与される制約を定義する．

JSP単体テスト支援フレームワークによって生成されるテストケースには，ページを表示するための操作とページに入力するテストデータが記述されている．ページを表示するための操作(以下，本操作を単にサーバデータ設定という）は，(1)ページが参照するデータのアプリケーションサーバへのセットと(2) Controllerの操作であり，これらはWebアプリ仕様の情報から生成する．テストデータはデータ定義と型定義を基にテストデータ生成プラグインにより生成する．これらの生成されたテストデータは，ページ仕様に定義されているページ生成の際に参照されるデータやフォーム項目のデータとして利用する．

アプリケーションで利用されるデータ型の概念はアプリケーションが利用されるドメインによって異なる．また，各々の型に対してどのような値を妥当な値，不当な値とするかは戦略により異なる．このため，JSP単体テスト支援フレームワークではテストデータ生成機能をプラグインとして差し替え可能とした．テストデータ生成プラグインの標準実装としては，文字列と日付，数値，URI，メールアドレスなどに対するプラグインを実装している．

テスト実行では，(1)サーバデータ設定と(2)テストデータの入力と証跡の保存を行う．このうち，(1)はテスト対象アプリケーションが利用するフレームワーク

	設計・生成	実施・評価
A	7.9	22.7
B	0.5	1.4
C	0.1	3.8

表 -1　テスト実施工数（人日）

	表示系	入力系
A	207	4785
B	204	111
C	133	779

表 -2　生成されたテストの数（件）

	ツール有	ツール無
A	30.6	76.2
B	1.9	5.8
C	3.9	11.6

表 -3　テスト工数の比較（人日）

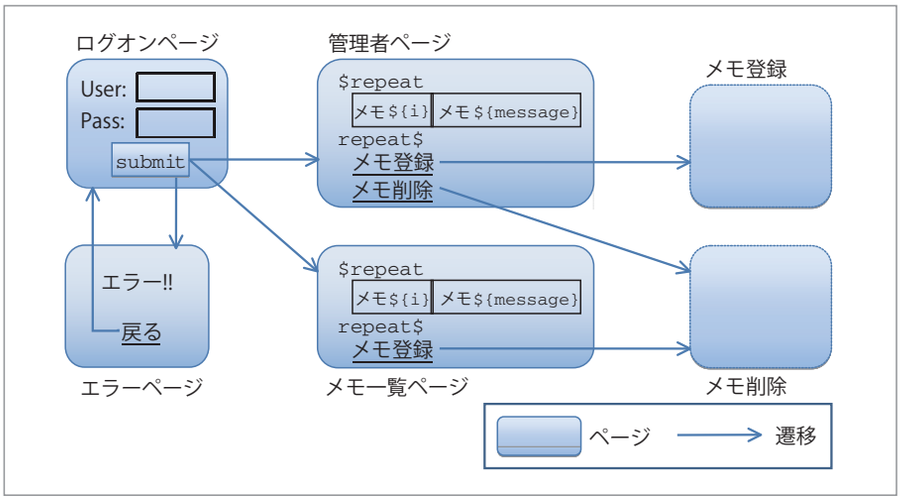

図 -2　ページモデルの例

に，(2) はテストで利用する Web ブラウザに応じてその実現が異なる．そのため，(1)，(2) のそれぞれに対する基本操作を API として定義し，サーバデータ設定と Web ブラウザ操作をプラグインとして差し替え可能とした．

JSP 単体テスト支援フレームワークでは，サーバデータ設定プラグインの標準実装として Struts とオブジェクトワークスに対するプラグインを，Web ブラウザ操作の標準実装として Internet Explorer に対するプラグインを実装している．

実際のプロジェクト A，B，C に対して JSP 単体テスト支援フレームワークを利用し，そのテスト支援効果を評価した．各プロジェクトにおける JSP のファイル数はそれぞれ (A) 67，(B) 95，(C) 133 である．(A) では，本フレームワークを初めて利用する開発者がテストを実施し，(B) と (C) では，本フレームワークの利用法を習得している開発者がテストを実施した．

JSP 単体テスト支援フレームワークを利用したテスト実施の工数を**表 -1** に，生成されたテスト数を**表 -2** に，フレームワーク利用の有無によるテスト工数の変化を**表 -3** に示す．ただし，表 -3 において，ツールを利用しなかった場合のテスト工数は，テスト数とシステムの特徴（フォームの項目数やエラー画面の数など）を基に他のプロジェクトの実施経験から算出した値である．JSP 単体テスト支援フレームワークを利用することにより，テスト工数が約 1/3 に削減された．

【 View プログラムのテストシーケンス生成 】

WebWare におけるユーザインタフェースとなる View をテストするためには，View が生成するページの構成を明確にする必要がある．そこで，ページ中のテキストと画像を示すコンテンツ，およびフォームとリンクを示す遷移要素を基本要素とし，基本要素と基本要素を配置する構造要素を用いてページをモデル化する．表やリストによって値を変えながら同じ要素が繰り返し出力される場合は，その繰り返しを反復要素として表現する．WebWare プロジェクトでは，WebWare を構成するすべてのページのモデルに基づいてページ，遷移，パスに関するテスト指標を定め，そのカバレッジを 100% にできるだけのテストケースを生成する手法を提案した [1]．

例として，複数ユーザ間でメモを共有する WebWare に対するページモデルを**図 -2** に示す．ユーザは，“ログオンページ” でユーザ名とパスワードを入力して `submit` ボタンをクリックする．ユーザが管理者であれば “管理者ページ” が，一般ユーザであれば “メモ一覧ページ” が，登録されていないユーザであれば “エラーページ” が表示される．この例では，簡便化のため “メモ登録” と “メモ削除” に当たるページに対する記述は省略している．図 -2 中の各ページ内の矢印の出ている要素が遷移要素，表や `form` 入力部が構造要素，`$repeat` と `repeat$` で囲まれた部分が反復要素，表内およびその他の画像やテキストがコンテンツを表す．

十分なテストを行ったかどうかを判定するために，ページ網羅，遷移網羅，パス網羅をテストの指標として採用する．ページモデルによって表現されたページ，遷移，パスの数を基準として，テストセットによりどれだけのページ，遷移，パスについてテストが実施されたか算出する．ページに対するカバレッジを 100% にするテストシーケンスを用いてテストを行うことにより，View のすべての制御パスに対して仕様通りのページが生成されることが確認できる．遷移に対するカバレッジを 100% にするテストシーケンスを用いてテストを行

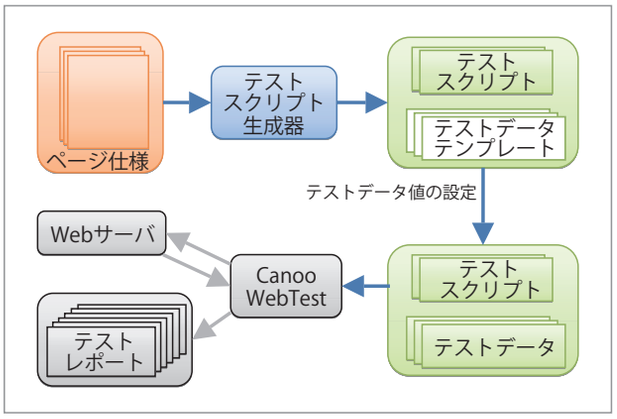

図 -3　テストシーケンス生成ツールの構成

うことにより，View によって生成されたページ中の遷移が仕様通りに生成されていることが確認できる．パスに対するカバレッジを 100% にするテストシーケンスを用いてテストを行うことにより，遷移経路に依存せず仕様通りのページが生成されることが確認できる．

ページモデルを XHTML を用いて記述したページ仕様を入力として，提案した 3 種類のカバレッジを 100% とするテストシーケンスを生成するツールを試作した．本ツールの概要を**図 -3** に示す．本ツールは，生成したテストシーケンスを実行するための Canoo WebTest に対する入力となるテストスクリプトと，テストスクリプト内のデータ入力で利用するテストデータテンプレートを出力する．ページ数 30，ページ生成プログラム数 14，フォーム数 12，リンク数 21 で構成される WebWare を対象にパスに対するカバレッジを 100% にするテストシーケンスを生成した結果，20 のテストシーケンスが約 2 分で生成され，現実的な時間でテストシーケンスの生成が可能であることを確認した．生成されるテストシーケンスの数は，ページ数，遷移数，遷移のループ数に依存するが，提案手法ではループを 2 度以上辿らないため，これらの数が多くなっても現実的な数で抑えられる．また，テストに要する時間は生成されたページ内の検査項目および操作項目の数に依存するが，100 個の検査項目を 3 秒程度で検査可能であり，現実的な時間でテスト可能である．

WebWare の解析技術

高信頼 WebWare 開発環境の実現には，WebWare を構成する個々の要素のソフトウェアモデルと，それらを統合した WebWare 全体の構造を表すモデルが重要な役割を果たす．モデルを中心として，WebWare のソースコードを解析してモデルを構築するリバース・エンジニアリングと，モデルをベースとして WebWare を開発するフォワード・エンジニアリングを統合したラウンドトリップ・エンジニアリングが実現できる．WebWare プロジェクトでは，大学で以前から研究開発が行われているリバース・エンジニアリング技術と，企業の研究成果である自動生成技術を融合することでラウンドトリップ・エンジニアリングの実現を目指した．

そのためのモデルとして WebWare プロジェクトでは WebWare 意味モデル[5] を提案している．WebWare 意味モデルは，WebWare を構成するページとページ遷移を実装技術に依存することなく記述する．ページとページ遷移は WebWare の基本構造であると同時に WebWare の意味を表現している．WebWare 意味モデルの構成要素は，入出力ポートを持つ Page と Action の 2 種類のコンポーネント，および入出力ポートを接続するコネクタである．Page は論理的なページ，Action は論理的なサーバ側ロジックを表し，コネクタは遷移を表す．入力ポートは URI などのトリガとなる指定情報を表し，出力ポートは次の遷移先のリクエストである．リバース・エンジニアリングの場合は，ソースコードやフレームワークの設定ファイルから Page や Action, 各ポートの情報, 接続関係を抽出する．フォワード・エンジニアリングでは，各ポートの情報と接続関係を実装に用いるフレームワークにおける記述にマッピングする．

たとえば，代表的なフレームワークである Struts を用いた WebWare（以下，Struts アプリケーションと呼ぶ）の場合，以下のようにして WebWare 意味モデルの構築に必要な情報を抽出する．Struts では HTML や JSP によって Page が実装される．これらはパスによって一意に特定できるため，パスを Page の入力ポート情報とする．Page の出力ポート情報は遷移先の URI であり，遷移を表す HTML の `a` 要素や `form` 要素，JSP の `html:link` 要素などにおいて遷移先を指定する属性の値から抽出する．Action は JavaServlet によって実装されるが，リクエストと Action の対応や Action から遷移する Page の指定は Struts 設定ファイル（struts-config.xml）に記述される．そこで, Action の入力ポート情報として Struts 設定ファイルから `action` 要素の `path` 属性の値を，出力ポート情報として `action` 要素の子要素である `forward` 要素の `path` 属性の値を抽出する．抽出した情報に基づいて Struts アプリケーションの WebWare 意味モデルが構築できる．

WebWare 意味モデルはフォワード・エンジニアリングだけでなく，WebWare のデータフロー解析[3] や整合性検査[2] などにも活用できる．WebWare のデータフローは，個々の Page や Action 内のデータフローを WebWare 意味モデルにおける接続関係に基づいて連

結して得ることができる．また，WebWare は Page や Action の実装とフレームワークの設定ファイルが一体となって構成されており，実装と設定ファイルの不整合や動的に結合される要素間の不整合は実行時エラーとなる．WebWare 意味モデルはソースコードや設定ファイルから接続関係を抽出して得られるため，一部の不整合は WebWare 意味モデルを調べることで検出できる．

WebWare プロジェクトでは代表的なフレームワークである Struts を対象として，WebWare 意味モデルの構築と視覚化，データフロー解析，整合性検査，コーディング規約検査などを行う各種ツールを実現した．JSP や JavaServlet といった個々のソースコードの解析には既存の CASE ツール・プラットフォームである Sapid☆5 を利用して WebWare 意味モデル生成系を構築し，その上に各種ツールが実装されている．視覚化の例を**図 -4** に示す．視覚化に加えて，デッドリンクと未使用フォワードの検出，デッドページの検出，全到達ルートの検出，入出力データの表示，依存データの追跡などの機能が実現されている．

WebWareの作成支援

WebWare を構成する技術は大きくサーバサイドとクライアントサイドに分けられる．サーバサイド技術は，Java などのプログラミング言語やデータベースであり，従来の開発手法に基づいた作成支援が可能である．WebWare プロジェクトでは，クライアントサイドや双方を連携する技術に対して以下に挙げる作成支援技術を開発した．

- JavaScript アプリケーション作成支援
- CSS 記述の作成支援
- スタイル変換技術によるバックエンドとユーザインタフェースの連携フレームワーク
- ページ構成決定支援

以下では，ページ構成決定支援について紹介する．

WebWare におけるページ構成の決定には，情報を分かりやすくまとめ，伝えるための表現技術である情報アーキテクチャが重要である．WebWare プロジェクトでは，WebWare と都市のアナロジーに着目した情報構造の設計ルールを定義し，ページ構成の決定を支援するツールを開発した[6]．

Kevin Lynch の「都市のイメージ[4]」によれば，都市は以下の要素により構成される．

- パス：観察者が通る可能性のある道筋．

☆5 http://www.sapid.org/

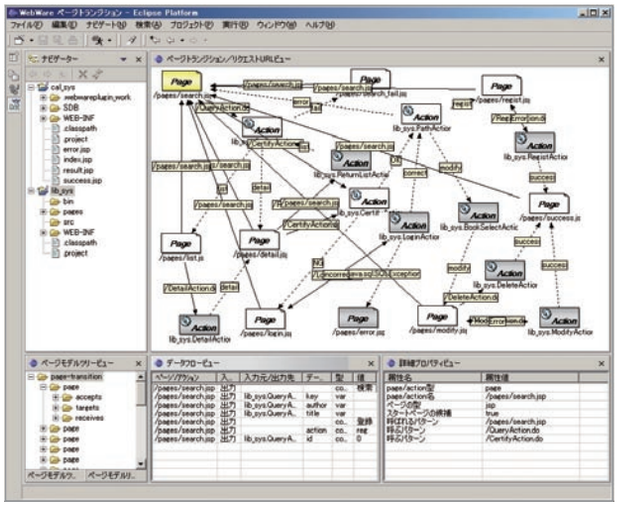

図 -4　WebWare 意味モデルの視覚化の例

- ノード：観察者がその中に入ることのできる焦点．その代表的なものは，パスの接合または何らかの特徴の集中によってできたもの．
- エッジ：パスでない線状の要素で 2 つの局面のある境界．
- ディストリクト：観察者が心の中でその内側に入ることができ，しかもその内部に何らかの同じ特徴が見られる都市の部分．
- ランドマーク：点を示すものであり，観察者からは離れて存在し，いろいろな大きさの単純な物理的要素から成り立っているもの．

上記の各要素はそれぞれ以下のように Web における要素として置き換えることができる．

- パス：リンク
- ノード：情報のかたまり（情報チャンク）
- エッジ：「外部リンク」という表示や明らかに広告と理解できるバナーなど．パスの先が他の領域であると明示されたとき，そこにエッジが存在すると考える．
- ディストリクト：同じ特徴を持つ情報の集まり，すなわちカテゴリ
- ランドマーク：トップページやグローバルナビゲーションなど，サイト内をナビゲーションする上で手掛かりとなるもの

置き換えたそれぞれの要素に対して，「特徴的なノードを中心として，その周囲にディストリクトを構成する」や「ランドマークはサイトの規模に合わせて，適切な数だけ配置されるべきである」などの設計ルールを定義する．また，要素間の相互関係に対するルールを決定する．

これらのルールに基づいて，ボトムアップのアプローチで情報の構造を構築するプロセスが定義できる．このプロセスを以下に示す．

(1) 情報をできる限り漏れのないように列挙する．この

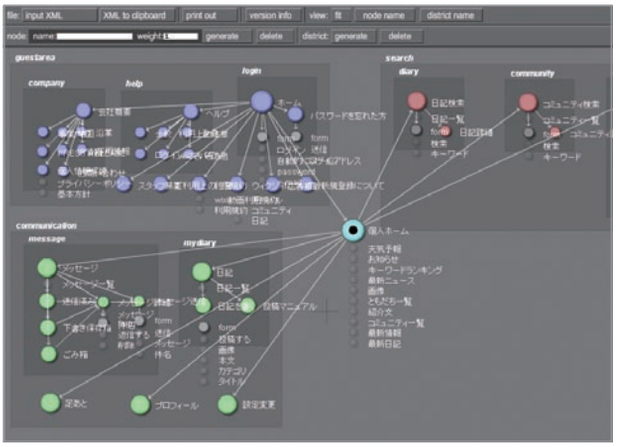

図 -5　ページ構成設計支援ツール

情報のかたまりを情報チャンクと呼ぶ.

(2) 相対的に重要度の高い情報と低い情報を区別し，情報チャンクの大きさによって重みをつける．内容が近いと思われるもの同士を近くに配置し，全体をいくつかのまとまりに分ける.

(3) 得られた情報のまとまりにエッジとディストリクトを設定する．このとき，ノード・エッジ・ディストリクトの設計ルールを適用し，情報の構造化を行う.

(4) 各ディストリクトに代表ノードを設ける.

(5) 代表ノードを設けるプロセスを繰り返すことでディストリクトを統合し，トップページを設定する.

(6) トップページから末端のノードまでパスをつなぐ．これによりすべてのノードへの構造化されたアクセス経路が確保される.

最大 4 階層，36 種類のページで構成されるサイトを対象に上記のプロセスに従ってリデザインを行った結果，最大 5 階層，38 種類のページになった．第 1 階層のカテゴリ数が 10 項目から 8 項目に整理される一方で，1 つのページに含まれていた複数の情報が分割され，サイト利用者の目的ごとに情報へのパスを分けることができた.

提案したプロセスを支援する機能をツールとして実装した（**図 -5**）．このツールは，以下の作業を支援する機能を含む.

- 情報チャンクの配置・重み付け（構築プロセスの (1), (2) に対応）
- グルーピングと代表ノードの選別 ((3), (4), (5) に対応）
- リンク構造の決定 ((6) に対応）

また，本ツールで作成したサイト構造は，XML 形式で Web サイトの雛型として出力可能である．たとえば，出力された XML ファイルから struts-config.xml などの Web アプリケーション設定ファイルを生成することができる.

今後の展開

5 年間のプロジェクトにおいて多くの研究成果が得られ，その一部は実製品に取り込まれるなど目標をおおむね達成することはできた．しかし，WebWare の高機能化と大規模化の進み方は速く，求められる信頼性のレベルも向上し続けている．本プロジェクトの成果をより広く活かしつつ，今後も継続的に研究開発を進めたい.

謝辞　WebWare プロジェクトを進めるにあたりさまざまな課題について熱心に議論し研究開発を推進していただいたプロジェクト参加企業および大学の皆様に感謝いたします.

参考文献

1) 渥美紀寿，桑原寛明，金子伸幸，山本晋一郎，阿草清滋：高信頼 Web アプリケーションのためのページ生成プログラムのテスト手法，コンピュータソフトウェア，Vol.24, No.4, pp.153-164 (2007).
2) 金子伸幸，桑原寛明，山本晋一郎，阿草清滋：StrutsLint：Web アプリケーションコーディングチェッカ，コンピュータソフトウェア．(to appear.)
3) 黒川　翔，桑原寛明，山本晋一郎，阿草清滋：Web アプリケーションにおけるデータ依存グラフ，日本ソフトウェア科学会 FOSE'05, pp.237-246 (2005).
4) Lynch, K.：*The Image of the City*, MIT Press (1960).（丹下健三，富田玲子訳：都市のイメージ，岩波書店 (1968).）
5) 松塚貴英，阿草清滋，山本晋一郎：ラウンドトリップエンジニアリングを目指した Web アプリケーションのための意味モデル，情報処理学会論文誌，Vol.46, No.5, pp.1145-1154 (May 2005).
6) 大庭ありさ，満田成紀，松延拓生，鰺坂恒夫：「都市のイメージ」を利用した Web サイトにおける情報設計方法の提案，ヒューマンインタフェース学会研究報告集，Vol.8, No.4, pp.43-48 (2006).

（平成 20 年 9 月 10 日受付）

桑原　寛明（正会員）　kuwabara@cs.ritsumei.ac.jp
2001 年名古屋大学工学部卒業．同大学院に進学．2007 年より立命館大学情報理工学部助教．博士（情報科学）．並行計算モデル，プログラム解析に関する研究に従事．日本ソフトウェア科学会会員.

金子　伸幸（正会員）　kaneko@netreqs.co.jp
2002 年名古屋大学工学部卒業．同大学院に進学後，2007 年同大学院情報科学研究科附属組込みシステム研究センター研究員．現在，（株）ネットレックスに勤務．ソフトウェア再利用，要求工学，ソフトウェア開発環境などの分野に興味を持つ．日本ソフトウェア科学会会員.

渥美　紀寿（正会員）　natsumi@it.nanzan-u.ac.jp
2000 年名古屋大学工学部電気電子情報工学科卒業．同大学院に進学．2005 年より南山大学数理情報学部講師．ソフトウェアの再利用に関する研究に従事．日本ソフトウェア科学会，電子情報通信学会各会員.

山本晋一郎（正会員）　yamamoto@ist.aichi-pu.ac.jp
1987 年名古屋大学工学部卒業後，同大学院に進学．1991 年同大助手，1996 年講師，1998 年愛知県立大学情報科学部助教授，2007 年同大准教授．プログラミング言語処理系，ソフトウェアの形式的開発手法，ソフトウェア開発環境に関する研究に従事．電子情報通信学会，日本ソフトウェア科学会各会員.

阿草　清滋（正会員）　agusa@is.nagoya-u.ac.jp
1970 年京都大学工学部卒業．同大学院に進学．1974 年同大助手．講師，助教授を経て 1989 年より名古屋大学教授．工学博士．専門分野はソフトウェア工学，ソフトウェア開発方法論，知的開発環境，ソフトウェアデータベース，仕様化技法，再利用技法，マンマシンインタフェース．電子情報通信学会，日本ソフトウェア科学会，IEEE, ACM 各会員.