第一回 SPARTS プロジェクトミーティング 資料

# ソフトウェア検索の現状分析

亀井 俊之　　粕淵 健郎　　佐藤 弘紹　　門田 暁人　　松本 健一 *

2002 年 3 月 18 日

## 1　はじめに

WWW 上のソフトウェア資源を収集・解析・検索するシステムを構築するにあたっては，一般の WWW 検索エンジンと同様，ソフトレア資源の情報を格納するデータベースサーバが不可欠である．その中でも，ソフトウェア資源の収集という点に着目して考えると，ただ闇雲に，全てのサーバ上のファイルを逐一解析していくのでは非常に無駄が多い．よって，詳細な解析を行う前段階で，ソフトウェア資源を含む（と推定される）サイトだけを効率良く自動収集（**クロウリング**，crawling）することのできる仕組みが必要になってくる．

そこで，本発表では，従来のソフトウェア検索の手法や一般的な WWW 検索エンジンのクロウリング方法を紹介するとともに，WWW 上のソフトウェア資源を効率よくクロウリングするためにはどのような工夫が必要であるか考察し，今後のソフトウェア検索システム構築において我々が取るアプローチ手法を説明する..

## 2　従来のソフトウェア検索手法

ここでは，これまで WWW 上のソフトウェアの検索はどのように行われてきたかについて述べる．また，ソフトウェア検索の検索対象にはどのようなものがあるのか列挙し，そういった既存ののソフトウェア検索手法を，どういった方向に拡張していくべきかを考え，今後の指針として提案する．

### 2.1　ソフトウェア検索手法

#### 2.1.1　WWW 検索エンジン＋ブラウジング

特に専用のサービスは用いることなく，通常の WWW 検索エンジンを使うことでソフトウェアを検索する方法も，よく用いられている．これは，ソフトウェアを検索するというより，その配布元や関連サイトなどを検索エンジンで発見することにより，さらにそこからブラウジングすることによってソフトウェアへたどり着く方法である．

この方法では，登録型の専用サービスに比べると多種多様なソフトウェアを発見できる可能性を持つ．しかし，専用サービスでないがために，ソフトウェア以外のものが検索結果に出力されることも多く，検索の精度が著しく低いというのが欠点である．また，利用者にはブラウジングの手間がかかるのも問題である．

#### 2.1.2　登録型ソフトウェア検索サービス

- 登録型ソフトウェアライブラリ

  検索対象とするソフトウェア（と，その配布元サイト等のデータ）を人手で登録し，種類ごとに幾らかのカテゴリに分けてデータベース化し，それを検索させるような Web サービス．通常の WWW 検索エンジンで言うところの「ディレクトリ型検索」にあたる手法を用いており，現在広く利用されている．例として，”Vector” [1]，” 窓の杜” [2]，”Softwareexchange.com” [3] などがある．検索の手段としては，分野に応じた**カテゴリ検索**や，サイト内の**全文キーワード検索**などの手段がある．人間によってデータベース化を行うために，検索の精度（precision）はある程度高くなる．しかし，登録されたソフトウェアしか検索対象に含めることができず，WWW の広域を検索対象にすることはできない．

- 登録型ソースコードライブラリ

  手動で登録済みのソースコードを検索することができる Web サービス．キーワード検索とカテゴリ検索があり，登録型ソフトウェアライブラリのソースコード版のようなシステムになっている．例とし

*奈良先端科学技術大学院大学

[1] http://www.vector.co.jp
[2] http://www.forest.impress.co.jp
[3] http://www.softwareexchange.com.sg

て，”Layer-8 SourceCode Exchange”[4] ，”Fresh-Meat”[5] ，”SourceForge.net”[6]

その特徴は登録型ソフトウェアライブラリと同じで，検索の精度は高いが，広範囲を検索対象にすることができないことである．しかも，登録型ソフトウェアライブラリに比べて知名度は著しく低く，サイトの種類や登録件数は非常に少ない．

#### 2.1.3 FTP 検索サービス

FTP サーバが持っているファイルの全リストを，各地の FTP サーバから取得してリストを作成し，ファイル検索をさせるような Web サービス．例として，”Archie”[7] ，”AllTheWeb.com”[8] などがある．FTP 検索サービスでは，世界中の FTP サーバの内容を検索対象にすることができるものの，ファイル名による検索しかできず，検索しにくいことが問題となる．このことを解決するための研究も行われている．広瀬らの研究 [1] では，ファイル名だけではなく，カテゴリ，OS，用途などパッケージの持っている属性を検索キーとして指定できるコンテンツベースの検索システムを提案している．

#### 2.1.4 WWW 検索エンジン＋自動インデキシング

ここでは，自動インデキシングを行うソフトウェアコンポーネント検索システムとして，Agora [2] を紹介する．

Agora は，Java Beans 及び CORBA オブジェクトのためのコンポーネント検索エンジンで，**クロウラ**により全世界から自動的に Index を作成するものである．その構造を図 1 に示す．AltaVista のＳＤＫを使って実装している．Agora では，WWW に対してクロウリングを行った後，収集された HTML 文書中に applet タグがあるかどうかを調べ，applet を自動ダウンロードし，applet 中で参照されている Java Beans を Index 化する．また，CORBA ネーミングサービスを利用して CORBA オブジェクトを探す．

#### 2.1.5 登録型＋クロウラ型（ハイブリッド型）

ここでは，登録型ではあるもののクロウリングによって自動的にインデクスを生成する CodeHound [9] を紹介する．

[4] http://layer-8.com/sce/
[5] http://freshmeat.net
[6] http://sourceforge.net
[7] http://archie.emnet.co.uk
[8] http://www.alltheweb.com
[9] http://codehound.com

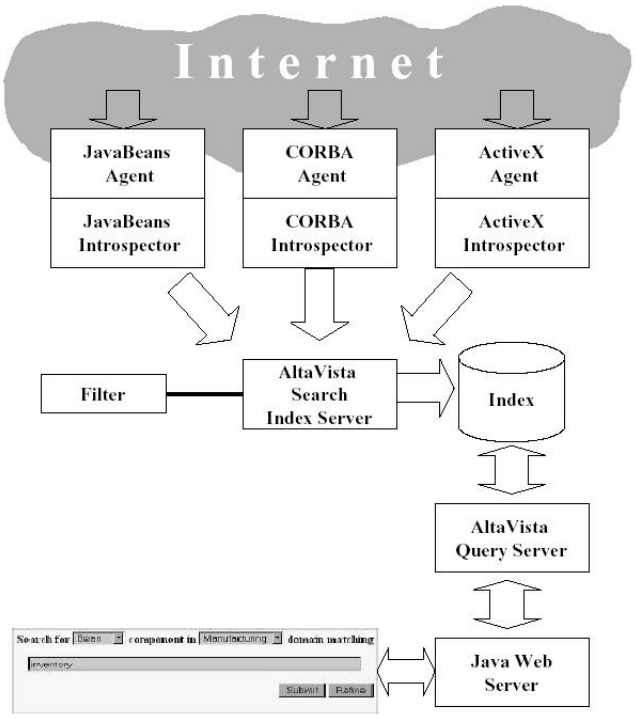


図 1: Agora の構造（[2] より抜粋）

CodeHound では，一般的なディレクトリ型検索エンジンのように，手動で Web サイトを登録する．この登録は，サイト管理者，及び利用者によって行われる．登録されたサイトだけを定期的に探索し，各サイト内のテキスト情報を Index 化する．そして，登録サイト内のアーカイブを解析し，圧縮ファイル（Zip/Tar/GZip など）の中身を調べる．

### 2.2 WWW 上のソフトウェア関連資源

WWW 上には，膨大な数の様々な資源が点在している．その中でも，今回必要となるのは，ソフトウェアプロダクトに関連するものだけである．そうした資源を，ここでは**ソフトウェア関連資源**と呼ぶことにする．ソフトウェア関連資源には以下のようなものがあると考えられる．

1. **バイナリデータ**
   ソースコードおよびドキュメントがないものは，非常に解析が困難になる．独立したソフトウェアプロダクトであれば，この形式で WWW 上に存在することは少なく，ドキュメントとともに圧縮ファイルとして配布されていることが多い．

2. **ソースコード**
   テキスト形式で，特徴メトリクス等が解析可能である．これも圧縮されている場合が多い．

3. **ドキュメント**
   readmeやヘルプファイルなど，人間にとってわかりやすく，また解析も容易で，ソフトウェアの分類などに効果的であると考えられる．通常バイナリとともに圧縮されているが，テキストやHTML形式でWWW上に公開されている場合も多い．しかし，必ずしもバイナリと同じサーバ上に存在するとは限らない．

4. **圧縮ファイル**
   多くのソフトウェアプロダクトは圧縮されて配布されていることが多い．しかし，ソフトウェアプロダクト以外にも，圧縮されたWWW資源はたくさんあり，そのままでは判別できない（ファイル名等から多少は判別できるかもしれない）．解析作業を行うには解凍が必要．

5. **関連サイト**
   ソフトウェアの配布元のサイトや，紹介サイトなども，ソフトウェア関連資源であるといえる．上に挙げた圧縮ファイルは，単独では収集時にはソフトウェアであるかどうか判断が不可能だが，そのリンク元がこうした関連サイトであれば，ソフトウェアである可能性がある．逆に，リンク元や，サイトの上位階層に関連サイトがなければ，ソフトウェアである可能性はかなり低いと思われる．

6. **文書中に組み込まれたソースコード**
   HTMLなどの文書中にも，プログラムのソースコードが記述されていることがある．例えば，プログラム入門サイトに紹介されるサンプルコードなどがこれに当たる．今までの例とは少々異なり，単体で独立したソフトウェアプロダクトを表すものではないが，一応ソフトウェア部品であるといえる．

### 2.3 ソフトウェア検索の今後の指針

以上，従来のソフトウェア検索手法について説明した．列挙した手法は，それぞれに長所と短所を備えているが短所を最小限にとどめるソフトウェア検索が今後必要であると考えられる．ここでは，自動的なクロウリング及びインデキシングを行うAgoraのアプローチを，JavaBeansとCORBAオブジェクト以外の言語へも広げることが，1つの有力な方向性であると思われる..

一方，Webページ中の全てのアーカイブを自動ダウンロードして中身を調べるのは，実行効率の点から無理がある．フィルタリングを行う．(明らかにソフトウェアと関係ないと思われるページを探索しない）アーカイブの一部だけをダウンロードする．ff（たとえば，LZHファイルは，ファイルの先頭に圧縮ファイルのリストが入っている）

## 3 クロウリング

ここでは，一般的なロボット型WWW検索エンジンにおけるクロウリングシステムの紹介を行う．

### 3.1 WWW検索エンジンの構成

全文検索を行うロボット型の検索エンジンは，

1. クロウラ (スパイダー)
2. フィルタリングエンジン
3. インデクサ
4. 検索クライアント

から構成される [3] [4]．この構成の全体図を図2に示す．

まず，クロウラが検索エンジン固有の巡回アルゴリズムに基づいて，新規に更新されたWebページを収集し，その結果をWeb情報データベースへ保存する．

次に，フィルタリングエンジンが，Web情報データベースに収集されたWebページを検索プロファイル機構に照合しながら分類し，分類データベースに登録する．登録する情報は，Webページのカテゴリ分類のほか，リンク先の情報や階層構造など，各検索エンジンごとに必要とされるデータも収められる．

フィルタリングエンジンと並行して，インデクサが全文検索エンジンによりWeb情報データベースの内容をインデキシングし，単語単位の索引を作成する．検索クライアントでは，検索語やカテゴリ検索などのクエリを受け取ると，各ページの索引をもとに該当するページのURLを導出する．

### 3.2 クロウリング

ロボット型のWWW検索エンジンは，Web上のハイパーテキスト構造を自動的にたどるようなプログラムである**クロウラ**（crawler）を持っており，

1. ある一つのWebページを取得する．
2. そのWebページから参照されているすべてのWebページを見に行く．

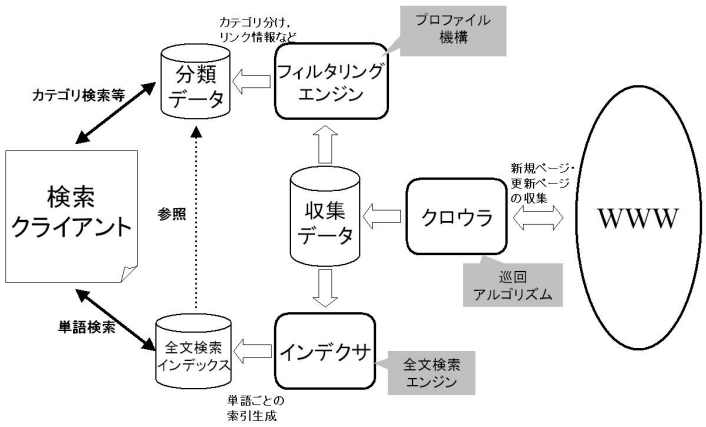


図 2: ロボット型 WWW 検索エンジンの構成

といったことを再帰的に繰り返していくことで，検索対象となる Web ページを増やしている．このような処理を**クロウリング**（crawling）という．

クロウラは新規 Web ページを収集する他に，収集済みの Web ページの更新を確認し，索引を更新するという処理も行っている．以下に，これらの処理について説明する．

#### 3.2.1 新規 Web ページの収集

新規 Web ページを収集していくための巡回アルゴリズムはクロウラによって異なるが，一般には，過去に収集した Web ページから得られる URL リストを起点にする [5]．たいていのクロウラでは，手動で URL を入力しロボットに訪問させることもでき，投稿記事や，メーリングリストの公開アーカイブなどから集めた URL が使われることもある．

起点となる URL のリストが与えられると，クロウラはその中から訪問する URL を選び，次々に文書を解析しながらどんどん新しい URL を収集していく．クロウラは，訪問した URL にあるページを認識できれば，その内容を Web 情報データベースに加える．

このように，起点となる URL にはなるべく多くのリンク情報が含まれていることが望ましい．

#### 3.2.2 索引の更新

WWW 検索エンジンにおける索引の更新としては，**一括更新型**と**逐次更新型**に大きく分けられる :.

1. **一定更新型**：Google [10] ，Goo [11] ，Excite [12] ...
   一括更新型では，一定期間ごとに新しい索引を最初から作り直し，完成した時点で旧索引と入れ替える．索引の更新周期は検索エンジンによって様々だが，短いもので 1ヶ月ごと（Google など），長いものでは数ヵ月ごと（Excite など）になる．

2. **逐次更新型**：infoseek [13] ，Lycos [14] ...
   逐次更新型では，現在の索引の一部分を更新することにより少しづつ索引を変えていく．実際には稼働中の索引の他に，内容更新するためのコピーを持っており，この 2 つを頻繁に入れ替えて稼動している．逐次更新型では，索引が逐次変化するため，検索件数やランク付けは日々変動する．

逐次更新型の中でも，独自の巡回アルゴリズムを用いている検索エンジンもある．例としては，フレッシュアイ [15] の独自のクロウリング/データベース化技術である**スマート巡回アルゴリズム** [6] [7] が挙げられる．この方法では，頻繁に更新され，かつ情報の信頼度の高い Web サイト（企業，政府，マスコミなどが提供する 5000 以上のサイト）を対象に優先的に巡回し，適宜索引に反映させる．このような方法により，フレッシュアイでは 1 日に 140 回（10 分に 1 度）の索引更新を実現し，新鮮さが重要な意味を持つ Web ページの最新の状態を検索できることを特色としている．

これを参考に考えると，ソフトウェアの検索システムに適合した巡回アルゴリズムを考案することで有効にソフトウェアを収集することが重要になってくるといえる．

### 3.3 クロウリングの指針

ソフトウェア資源のクロウリングにおいては，以下のような部分に重点を置かなければならない．

- 新規 Web サイトをクロウリングする**起点 URL** に，ソフトウェアが含まれている可能性が高いものを選択すること

- 頻繁にソフトウェアの更新が行われるサイトを重視する**巡回アルゴリズム**を用いること

[10] http://www.google.com
[11] http://www.goo.ne.jp
[12] http://www.excite.co.jp
[13] http://www.infoseek.co.jp
[14] http://www.lycos.co.jp
[15] http://www.fresheye.com

### 3.3.1 起点 URL

起点 URL の選び方によってシステムの性格付けが行われることは先に述べたが，本システムでは,「既存の WWW 検索エンジンを用いた起点 URL の選定」という方法を採用する予定である．

この方法では，既存の WWW 検索エンジンに何らかの検索質問語（**クエリ**）を与え，その検索結果として得られる URL リスト群をソフトウェア収集の起点 URL に用い，それ使ってクロウリングを行い，新規ページの情報を得る．もちろん，その際使用されるクエリは，なるべくソフトウェア関連資源を多く含む URL を得るようなものでなければならない．

起点 URL を既存検索エンジンから得ることができるようになれば，次のような利点が生まれる．

- 人的資源を節約できる．
- 検索質問語（クエリ）を拡張することにより，精度の高い情報の収集を期待できる．
- 機械的に処理するので，大量に収集できる．
- 新規 Web サイト発見の効率が向上する．

また，利用する検索エンジンの数を複数にすることによって WWW のカバー率を上げ，多くのサイトを対象にすることも考えられる．しかしこの際，取得する結果が多くなり，かえってノイズ（ソフトウェア関連資源以外のもの）が含まれる可能性があるため，絞り込み作業を行わなければならない．絞り込みにはベクトル空間法や TF/IDF 法を用いた文書解析を用いる予定である．

### 3.3.2 巡回アルゴリズム

一般的に，Web ページのリンク関係には「リンクを張っている側も張られている側も似たようなジャンルを扱っている」という特徴があるということが知られている．これを利用し，Web 上のコミュニティを探す研究がいくつか進められている [8]．こうした研究を参考に，ソフトウェア関連資源を多く持っているコミュニティを探し，その内部のみを重点的にクロウリングすることによって効率よくデータを収集することを目指す．

具体的なソースコードの場所の特定は，リンク元の HTML ファイルの文書構造を解析しておこなう．今現在どのような形態でソースコードやソフトが Web 上で公開されているかを学習するようなエージェントを設ける必要がある．

## 4 まとめと今後の展望

以上，ソフトウェア検索の現状分析を行った．

検索対象が多く，精度のより高いソフトウェア検索システムをつくるには，データベースの更新頻度が高く，ソフトウェア資源だけをなるべく多く収集して回るクロウリング手法の検討が必須であると分かった．そのようなクロウリング手法を実現するためには，起点 URL と巡回アルゴリズムを工夫する必要性がある．

大きく分けて，次の 3 種類の方向性で工夫を行っていくべきである．

- サイトの発見

  新規のソフトウェア資源を発見するに必要な起点 URL は，既存の検索エンジンを複数台使って調べる．そのために，既存の検索エンジン対し，それぞれ最良のソフトウェア発見クエリを考える．

- 更新の確認等

  巡回アルゴリズムは，ソフトウェアをいっぱい持っているコミュニティサイトを重点的に（高頻度で）調べることができるようなものを考える．

- データベースへの追加の検討 HTML のタグを解析するなどして，検索対象文書が存在するパターンを学習するできるようにすることで，関連文書を識別する．

## 参考文献


[1] 広瀬　雄二，大駒　誠一，” インターネット上のソフトウェア資源検索システムの設計と評価”，情報処理 39 巻 4 号 pp.1108-1115，1998

[2] Robert C.Seacord，Scott A.Hissam，Kut C.Wallnau，”Agora: A Search Engine for Software Components”，AMU/SEI-98-TR-011 TECHNICAL REPORT

[3] 原田昌紀，” 道しるべ:WWW サーチエンジンの作り方”，情報処理 41 巻 11 号，pp.1280-1283，2000

[4] 福島俊一，”WWW 情報検索技術と評価の問題”，情報処理 41 巻第 8 号，pp.913-916，2000

[5] Martijn Koster，”The Web Robot Pages”，`http://www.robotstxt.org/wc/robots.html`

[6] ASCII24 ニュース記事，” フレッシュアイ，リアルタイム検索サービスを開始”，2000 年 10 月 19 日

http://ascii24.com/news/i/serv/article/2000/10/19/618983-000.html

[7] 東芝:プレスリリース，”基礎資料「フレッシュアイのサービスとテクノロジー」”，東芝:プレスリリース 1998.6.29，
http://www.toshiba.co.jp/about/press/1998_06/j2901/fe2.htm

[8] インプレス WATCH ニュース記事，”米 NEC の研究者たちがインターネットの新しい検索アルゴリズムを発見”，2002 年 3 月 6 日，
http://www.watch.impress.co.jp/internet/www/article/2002/0306/neci.htm