協調可能なソフトウェアマップの開発

上野乃毅* 春山征吾† 山崎義弘‡

概要

オープンソースプロジェクトではソースコードを開発者以外にも公開することにより部外者の開発への参加を期待できる. プロジェクトを広く世に知らしめるためにソフトウェアマップが利用されているが, 既存のソフトウェアマップには制限がありソフトウェアの開発者・検索者にとって十分使いやすいものとなっていない.

この問題を解決するために, 協調可能な分散ソフトウェアマップ CodeRidge を開発した. CodeRidge は, ソフトウェアの情報を開発者の手元に置き, これを P2P を使って共有する. これにより, 全体として単一の大規模なソフトウェアマップを実現することが可能となる.

1 背景

1.1 フリーソフトウェアの開発

近年, Free Software Foundation[1] や GNU[2] の活動, Linux や自由な BSD の実装の普及, 安価で高速な機械やネットワーク環境の普及などのおかげで, 個人がフリーソフトウェアを開発したり, 開発したソフトウェアをフリーソフトウェアとして公開することが非常に容易に行なえるようになった. 現在, 世界中で無数のフリーソフトウェアが開発され, 公開され, 使用されている. またフリーソフトウェアの開発をさまざまな形で支援するサイトも出現している. 例えば, VA Software[3] が提供する SourceForge[4] では, 開発プロジェクトを登録することで CVS レポジトリやバグトラッキングシステム, メーリングリスト, Web ページなどをを利用することができる. この他にも同様のシステムが存在する.

フリーソフトウェアの開発を支援するサイトの一つにソフトウェアマップがある.

1.2 ソフトウェアマップ

ソフトウェアマップとは, 分散開発を支援するコミュニティサイトの入口でありソフトウェアプロジェクトを適切に分類・整理する仕組みを提供する. ソフトウェアマップは, ソフトウェアプロジェクトの特性を記述するスキーマに沿った情報からソフトウェアを適切に分類する. ここで言う特性には, 動作環境・開発に使用したプログラミング言語・開発者・ライセンス・更新状況などがある.[1].

既存のソフトウェアマップには,

- Freshmeat のプロジェクトツリー [5]
- GNU フリーソフトウェアディレクトリ [6]
- SoureForge の Trove ソフトウェアマップ [7]

などがある.

ソフトウェアの開発者は, ソフトウェアマップに自身のソフトウェアプロジェクトを登録することでソフトウェアを広く宣伝することができる. ソフトウェアの利用者は, ソフトウェアマップを利用して目的のソフトウェアを効率良く検索することができる.

*ueno@unixser.org
†haruyama@unixuser.org
‡yoya@unixuser.org

[1] ソフトウェアマップは, ソフトウェアに特化した登録型検索エンジンという面を持つ

1.3 既存のソフトウェアマップの問題点

ソフトウェアマップはソフトウェアの開発者・利用者双方に有用なものだが, いくつか不便な点がある. 以下で, 既存のソフトウェアマップの問題点を述べる.

自由度に乏しい

既存のソフトウェアマップの実装では, ソフトウェアプロジェクトの分類項目があらかじめ想定されておりソフトウェアの本質を十分に表現できない場合が多い. また, このような大雑把な分類法は, 数千以上のプロジェクト群から目的のプロジェクトを検索する場面ではほとんど役に立たない.

情報が集中している

既存のソフトウェアマップは中心化されている. すなわち, 情報が一箇所にまとめられている. このためいくつかの不便な問題がある.

第一に, ソフトウェア開発者がプロジェクト情報を変更するたびに, すなわち,

- バージョンなどを変更
- 開発・配付サイトを移転

などを行うたびに, ソフトウェアマップのサイトにアクセスし HTML のフォームを使うなど指定された方法で情報を変更する必要がある. 通常, README などの別のソースからソフトウェア情報を登録・更新したり, 情報の書かれたファイルを直接書きかえてそれを転送することはできない.

第二に, 複数のソフトウェアマップ間での検索をすることは—それぞれが中心化されているため—行なうことが難しい. (Google[8] のような外部の検索エンジンを利用して検索することはできるが, この方法ではソフトウェア情報に特化した検索は行なえない)

2 この論文の構成

3 章では 1 章で述べたソフトウェアマップの現状を踏まえて, ソフトウェアマップの新しい形を提案する. 4 章では, 開発したソフトウェアマップ “CodeRidge” について, 特長 (4.1 節) や 構成 (4.2 節), 機能の詳細 (4.3, 4.4 節), 利用している技術 (4.5 節), 各コンポーネントの詳細 (4.6 節) を解説する.5 章では, 開発の分担について述べる. 6 章では現在の CodeRidge が抱える問題点について述べ, 今後考えられる研究開発について述べる. また, 7 章では今後の展望について述べる. 最後に 8 章でまとめを行なう.

3 ソフトウェアマップの分散化

1 章で説明した背景を考慮して, 我々はソフトウェアマップを分散化させることを提案する.

ソフトウェアマップを分散化することで, 登録するソフトウェア情報を開発者の手元に置き直接ファイルを書きかえたり 別のソースから自動的に生成することが可能になる.

情報は Peer-to-Peer (P2P) を利用して共有し, ソフトウェア情報を持つピアを起動することでピアの持つ情報がソフトウェアマップ全体から検索の対象となる.

また, 既存のソフトウェアマップの情報をキャッシュするピアを作ることで, 複数のソフトウェアマップの情報が検索できる大きなソフトウェアマップを構築することができる.

4 分散ソフトウェアマップ CodeRidge

我々は, 開発した分散ソフトウェアマップを CodeRidge[9] と名付けた. これは ソフトウェアのプロジェクトが分散している様を山脈を俯瞰した際に山々に尾根 (ridge) が連なっているのに例えたものである.

4.1 特長

CodeRidge は以下のような特長を持っている.

- ソフトウェアの情報は RDF (Resource Description Framework) [10] で定義されており, 開発者が自由に構成・拡張できる. 開発者は, 以下のような複数の方法で情報を操作できる.
  - HTML のフォームから入力

  - ファイルをエディタで編集
  - 適当なソースから自動的に生成
- ソフトウェア情報は P2P を利用して共有され検索の対象となる (ソフトウェアを登録したサーブレットとピア を起動すれば, 自動的に全体からの検索の対象となる)
- RDF のみを利用しているため, 大規模なデータベースシステムなどを別途必要としない.
- 標準的化された技術を利用しており, 特定の実装に依存しない.

4.2 構成

CodeRidge は以下のコンポーネントから構成されている.

coderidge-rdf: 軽量な RDF データベース, RDF パーザ, 検索式の評価器などを含む

coderidge-servlet: Java サーブレットによるユーザインターフェース

coderidge-data: 他のソフトウェアマップの情報を取得するツール とデータ変換のための XSLT ファイル

FANAL: JXTA 上の P2P フレームワーク. あるピアから行なわれた問い合わせを他のピアに送り, その返答を受け取る枠組を定義

coderidge-soap: JXTA (FANAL) ピアからの問い合わせを受けつける SOAP (Simple Object Access Protocol)[11] サービス

T4: TEX 風の構文をもつマクロプロセッサ. RDF やその他の形式をより扱いやすい構文から生成するために用いる.

各コンポーネントの詳細は 4.6 節で述べる.

4.3 主な機能

4.3.1 ソフトウェア情報の検索

CodeRidge では, 検索者はサーブレットを介して検索を行う. 検索サーブレットでは, ローカルに保持している情報と CodeRidge ネットワークに接続した各ピアが保持している情報に検索範囲を限定することができる.

ローカルに保持している情報の検索は, RDF で記述されたデータファイルを直接参照することで行われる. CodeRidge ネットワーク全体に対する検索は, P2P を利用して行われる. ネットワーク全体に対する検索の流れを以下に示す (**図 1**).

1. 利用 (検索) 者が Java サーブレットに検索を要求する
2. Java RMI[12] を利用して, サーブレットから JXTA ピアにリクエスト発行要求が送られる
3. FANAL を利用して, JXTA ピアが JXTA ネットワークで CodeRidge サービスを提供しているピアすべてに対してリクエストを投げる (なお, ソフトウェアの詳細な情報を取得する場合は, 一つのピアにのみリクエストを 投げる)
4. 他の JXTA ピアが そのリクエストに対応する
5. リクエストに対応する JXTA ピアはそれぞれ, SOAP を利用して Java サーブレットにリクエストを投げる
6. RDF データベースに対して検索が行なわれ, 結果が SOAP を利用して JXTA ピアに返される
7. JXTA ネットワーク を介して検索元の JXTA ピアに結果が返る
8. 他の JXTA ピアでも同様の処理が行なわれ, 結果が返る
9. Java サーブレットが, RMI を利用して結果を取得しそれを表示する.

4.3.2 ソフトウェア情報の取り扱い

CodeRidge では, ソフトウェア情報の登録で複数の手段が選択可能である. サーブレットを介して登録を行う場合には, 登録用の HTML フォームを利用してソフトウェア情報を入力する. 入力した情報から検索用の RDF ファイルが生成されて保存され, 検索の対象となる.

また, マクロプロセッサ T4(4.6.6 節) や, 既存のソフトウェアマップから抽出した情報 (XML で保持す

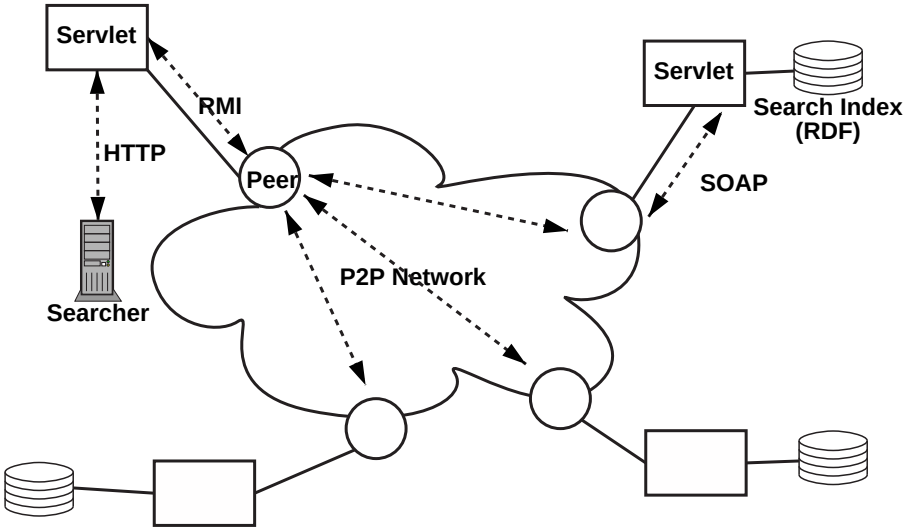

図 1: CodeRidge: P2P を利用する検索

る) を XSLT (XSL Transformations)[13](4.5.6 節) で変換することで, 検索対象として登録を行うこともできる. もちろん RDF ファイルを直接編集することも可能である.

また, 検索式を満たすソフトウェアの一覧やプロジェクト毎の詳細を表示することができる. RDF や XML などの形式で保持されるプロジェクトの詳細のソースから XHTML への変換を行い詳細を表示する. 変換に XSLT を用いる場合もある.

CodeRidge におけるソフトウェア情報の取り扱いのイメージを **図 2** に示す.

4.4 その他の機能

その他に, CodeRidge には以下のような機能がある.

4.4.1 ソフトウェア情報に期限を設定

ソフトウェア情報に有効期限を付けることができる. これにより発展的解消が予定されているソフトウェアプロジェクト (一時的に本家と並行した作業を行なうサブプロジェクトなど) の情報を管理することが容易となる.

4.4.2 ソフトウェア情報のキャッシュ

他のピアのソフトウェア情報を取得した際, その情報をキャッシュしておくことができる. これによりあるピアが落ちている場合でもそのピアが持つ情報のうち他のピアでキャッシュされているものは, CodeRidge 全体で利用することが可能になる.

4.4.3 他のソフトウェアマップとの連携

(既存の中心化された) ソフトウェアマップが構造化されたソフトウェアの情報を提供している場合, CodeRidge でその情報を利用して連携することが可能である.

現在, 以下に挙げるソフトウェアマップの情報が CodeRidge で利用可能である.

Freshmeat: プロジェクト情報 (XML) を HTTP で配布している

Ruby Application Archive (RAA)[14]: SOAP で情報を取得できる

これらのソフトウェアマップの情報を, (XML で配付されていないものはまず XML 化し) XSLT で変換して RDF 化し, CodeRidge の検索の対象とする.

CodeRidge で複数のソフトウェアマップの情報をキャッシュすることで, 複数のソフトウェアマップの

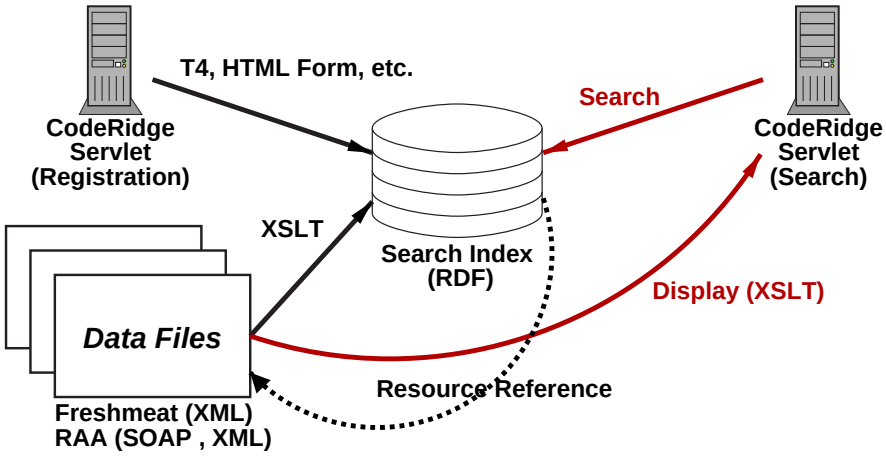

図 2: ソフトウェア情報の取り扱い

情報が検索できる大きなソフトウェアマップを作成することもできる.

4.4.4 ソフトウェア情報の記述の手間の軽減

マクロプロセッサ T4(4.6.6 節) を利用してソフトウェア情報の記述の手間を軽減する. 一つの T4 ソースから複数の形式 (CodeRidge 用 RDF,README,INSTALL など) を生成することも可能である.

4.5 CodeRidge で利用している技術

CodeRidge はその機能を実現するために次のような技術を利用している.

4.5.1 RDF

ソフトウェアの情報を検索の対象とするために, RDF を利用した.

RDF は, WWW などのネットワークリソースのメタ情報を処理するための基盤技術であり, 仕様は W3C 勧告として公開されている. RDF によるソフトウェア情報の記述の例を**図 3** に示す.

RDF のモデルでは, 主語 (サブジェクト) と述語 (プロパティ), その目的語 (オブジェクト) の三者関係によりリソースのメタ情報を表現する. 関係の連鎖は概念的に有向グラフとして捉えることができる. **図 3** の有向グラフによる表現を **図 4** に示す.

RDF は計算機による扱いが容易であることに加え, スケーラビリティを確保する上で有利な点が多い. 特に重要な特長のひとつは, プロパティに対しても一意に定まる URI (qualified URI) が割当てられることである. これにより, 異なる用途を想定して同名のプロパティを定義した場合にも, プロパティに与えられた URI により識別が可能である.

4.5.2 分散化 (P2P)

分散化したソフトウェア情報を共有するために Peer-to-Peer (P2P) ネットワークを利用する. CodeRidge では, 実装として JXTA[15] を利用した. これは以下の理由に依る.

- BSD License like な JXTA License の下で利用できる
- 通信の基盤が高度に抽象化されており, ファイアウォール越しの通信が考慮されている. また, 一般のピアは固定した IP アドレスを持つ必要がない
- プロトコルの標準化が積極的に進められており, Java による参照実装がすでに存在する

JXTA は現在も活発に開発が行なわれている. 我々は Stable Builds (STABLE_20020924T1446PDT) を

```
<rdf:RDF xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
         xmlns:core='http://coderidge.org/schemas/core/1.0#'>
  <rdf:Description about="http://coderidge.org/registry/502e13bf4">
    <core:Name>LSDB (The Lovely Sister Database)</core:Name>
    <!-- reference to GPL -->
    <core:License rdf:resource="urn:coderidge:license:7ee47d392"/>
    <!-- reference to Daiki Ueno -->
    <core:Author rdf:resource="urn:coderidge:author:cc3f30109"/>
    <core:Category>
      <rdf:Bag>
        <rdf:li parseType="Literal">Application</rdf:li>
        <rdf:li parseType="Literal">Database</rdf:li>
      </rdf:Bag>
    </core:Category>
  </rdf:Description>
```

図 3: RDF による記述の例

利用した.

4.5.3 Java サーブレット, JSP

ソフトウェア登録者・検索者のためのインターフェースを提供するために Java サーブレット [16] と JSP (Java Server Pages) [17] を利用した. また, SOAP インターフェースの配置にも Java サーブレットを使用している. 実装として, Apache Tomcat (開発時のバージョンは 4.1.18) [18] を利用した.

4.5.4 SOAP

JXTA ピアと Java サーブレット 間の通信に SOAP を利用した. SOAP を利用することでプログラミング言語や JXTA に依存せずに通信を行なうことができる. 実装として Axis (開発時のバージョンは 1.0)[19] を利用した.

4.5.5 WSDL

JXTA ピアと Java サーブレット 間の通信のインターフェースの定義には WSDL (Web Services Description Language)[20] を利用した. WSDL でインターフェースを定義し Axis の WSDL2Java ツールを利用して Java のコードを生成した. なお, WSDL から他の言語のコードを生成することも可能である.

4.5.6 XSLT

XSLT (XSL Transformations) とは, XML 文書を別の XML 文書や XHTML などに変換するための言語である. Java^{TM} 2 Platform, Standard Edition 1.4[21] の実装を利用している.

4.6 CodeRidge の各コンポーネントの詳細

以下で, 4.2 節で紹介した CodeRidge の各コンポーネントについて詳細を述べる

4.6.1 coderidge-rdf

このコンポーネントは, RDF(4.5.1 節) を読みこみ N-Triples[22] 形式で保持する. そして, 保持した情報からリソースや検索式に対応する情報を抽出することができる.

フィルタリング

リソースに対応する情報を抽出することができる. たとえば ソフトウェアのカテゴリ一覧やライセンス一覧, また (URI で指定される) 個別のソフトウェアなどの情報を取得できる.

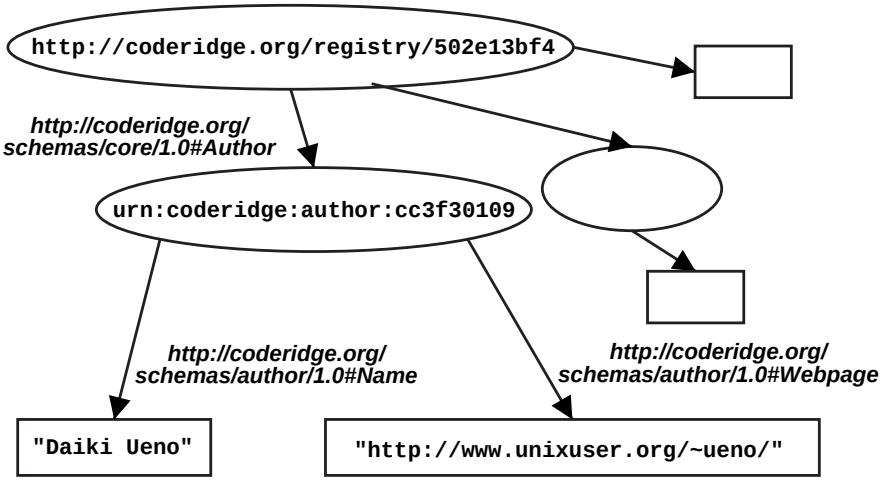

図 4: RDF モデル. 楕円はリソース, 長方形は文字列 (リテラル) を表す.

検索式

ソフトウェア情報を検索するために以下のような性質の検索式を利用できる.

- 文字列の完全一致ないし正規表現によるマッチングが可能
  (例: 完全一致: authorName = “Daiki Ueno”
  正規表現: authorName= /Ueno/)
- 任意の一階の述語論理式による検索式の組み合わせが可能
  (例: authorName = “Daiki Ueno”, description = /emacs/ ignorecase
  ‘,’ は AND を, ignorecase は大文字小文字を無視することを表す)

ABNF [23] による構文規則を **図 5** に示す.

```
expression  = match
            / *WSP expression *WSP
            / expression "," expression ;and
            / expression ";" expression ;or
            / "-" expression            ;not
            / "(" expression ")"        ;grouping
match       = identifier *WSP "=" *WSP pattern
              [1*WSP match-types]
match-types = match-type 1*(":" match-type)
match-type  = [no] ("regexp" / "ignorecase" /
                    "resource" / "word")
pattern     = %x22 string %x22
            / "/" string "/" ;shortcut to "regexp"
            / identifier     ;shortcut to "resource"
identifier  = (ALPHA / "_") *(ALPHA / DIGIT / "_")
string      = *(%x20-21 / %x23-7E)
```

図 5: 構文規則

4.6.2 coderidge-servlet

このコンポーネントは, Java サーブレット・JSP を利用して以下のユーザインターフェースを提供する.

- ソフトウェアの検索
- 検索されたソフトウェアの情報の詳細を表示
- ソフトウェアの登録
- 登録したソフトウェア情報の変更
- ユーザ情報の登録・変更

クロスサイト・スクリプティング対策

クロスサイト・スクリプティング (cross-site scripting, XSS) とは, 動的に生成される Web ページにおいて意図せず悪意のあるコードを訪問者のブラウザに送ってしまう脆弱性のことをいう [24].

XSS を防止するため JSP のタグライブラリ [25] を利用する手法を応用している [26]. この手法では Web ページの動的に生成される部分を数字参照でエンコードして悪意のあるコードの実行を防ぐ.

4.6.3 coderidge-data

このコンポーネントは,Freshmeat や RAA の情報を取得し XML ファイルとして保存するためのツールも提供する.

また, 以下の XSLT ファイルを提供する.

- 他のソフトウェア情報の XML ファイルから検索用 RDF ‘を生成する XSLT ファイル [2]
- 他のソフトウェア情報の XML ファイルから表示用 XHTML を生成する XSLT ファイル[3]

4.6.4 FANAL

CodeRidge のために, JXTA 上のリクエスト–レスポンスのためのフレームワーク FANAL[27] を作成した.

FANAL はファイル共有システム JXTA Content Management System (CMS)[28] を元にし, 高度に抽象化した. P2P でのリクエスト–レスポンス システムの実装を容易に行なうことができる.

次の 2 つのリクエスト方法が定義されている.

- Multicast リクエスト
- Unicast リクエスト

Multicast リクエストは, 特定のサービス名を登録したすべてのピアに対してリクエストを送る. CodeRidge ではソフトウェアの検索に用いている.

Unicast リクエストは PeerID を指定して一つの ピアに対してリクエストを送る. CodeRidge では結果の詳細の取得に用いている.

FANAL の詳細については “P2P フレームワーク FANAL”[29] で解説している.

4.6.5 coderidge-soap

このコンポーネントは, SOAP を利用して JXTA (FANAL) ピアからの問い合わせを受けつける. 問い合わせには

- ソフトウェアの検索
- ソフトウェアの詳細情報の取得

などがある.

ソフトウェアの検索では, 検索式を coderide-rdf で解釈しソフトウェア情報を結果として返す.

ソフトウェアの詳細情報の取得では, 対応する情報を返す.

[2]CodeRidge 独自の情報は 最初から RDF で保存される

[3]CodeRidge 独自の情報については, coderidge-servlet 内部で N-triples 表現から XHTML に変換している

```
\name{LSDB}
\liecense{GPL}
\homepage{http://lsdb.sourceforge.jp}
...
\author{Daiki Ueno}

\category{
\li Application
\li Database
}
```

図 6: T4 マクロを利用したソフトウェア情報の記述例

4.6.6 T4

ソフトウェア情報の登録を容易にするために, 新たな構造化文書形式を定義した.

一般に人間にとって, RDF だけではなく XML による記述は扱いにくい. このため, POD や RD, 各種 Wiki などの自然言語に近い形で記述できる構造化文書形式が広く利用されている. しかしながら, これらの構造化文書形式には互換性がなく, 習得までの手間が大きい. また, CodeRidge で利用するためには複数の出力形式に対応している必要があり, 開発者が独自の情報をソフトウェアに付加できるよう構文を拡張できることが望ましい.

このような要求を満足する構造化文書形式を定義するために, TEX 風の構文をもつマクロプロセッサ T4 を開発した.

図 3 の RDF を**図 6** のような T4 の記述から生成することができる.

5 開発の分担

上野は coderidge-rdf(4.6.1 節) や coderidge-soap(4.6.5 節), T4 (4.6.6 節) を主に担当した.

春山は coderidge-servlet(4.6.2 節) や coderidge-data (4.6.3 節), FANAL (4.6.4 節) を主に担当した.

山崎は CodeRidge の運用試験や調査を主に行なった.

6 今後考えられる研究開発

検索結果の表示順

現在, 検索結果は検索された順番に表示しているだけで検索者の意図に合わせるなどの並べかえを行なっていない. このため, 大量の結果が得られた際には目的のソフトウェアがあるかどうか探すのが面倒なことがある.

適当な並べかえをすることが好ましい. 与えられた検索式から検索者の意図を推測したり, 検索者が並びかえの方法を指定できるようにする方法が考えられる. Freshmeat は 投票による重み付けを行なっているが, 分散化ソフトウェアマップでそのようなことを行なうのは難しい.

大量のデータの取り扱い

現状の CodeRidge では, 非常に大量のデータの取り扱いに問題がある.

大量のデータを保持している場合に大量の検索結果が得られる検索がなされると, その過程で java.lang.OutOfMemoryError[4]がスローされることがある.

また,P2P 経由でデータを送信するときに単純に検索に該当するすべての検索結果を転送すると転送量が莫大になる可能性があるため, なんらかの制限が必要である. 現在はある程度以上の量の結果が得られた場合はエラーを返すようにしている.

Freshmeat の完全な代替となるような, 非常に大規模なソフトウェアマップとして十分機能させるためには, 大量のデータの取り扱いをより効率的に行なう必要がある.

[4]「メモリ不足のために Java 仮想マシンがオブジェクトを割り当てることができず, ガベージコレクタによっても使用可能なメモリをこれ以上確保できない場合にスローされ」るエラー [http://java.sun.com/j2se/1.4/ja/docs/ja/api/java/lang/OutOfMemoryError.html]

7 展望

7.1 分散開発の支援

CodeRidge は分散開発の支援をすべく作られたソフトウェアマップなので, なんらかの形で CodeRidge ないし CodeRidge のコンポーネントが分散開発の支援に生かされることを我々は望んでいる.

たとえば, P2P を利用した新しい形のソフトウェア分散開発支援システムが出現する際に, CodeRidge の枠組を利用することが考えられる. ソフトウェアに関する雑多な情報のみならず, パッチやバグ報告など, 開発の過程で必要となる様々なリソースを P2P を利用して統一的に扱うことも可能となる.

7.2 ソフトウェアマップ以外への応用

CodeRidge は, ソフトウェアマップ以外にも P2P を利用した情報共有・検索システムの枠組として利用できる.

たとえば, 現在特定のサーバのディレクトリを (NFS,SMB などで) マウントしそこに情報を集めて共有しているものを, CodeRidge を利用して中央サーバを利用せず個人のマシンの特定のディレクトリのファイルを共有することで置き換えることが可能である. このようにすることで以下の利点がある.

- いちいちファイルを中央サーバに置く必要がない
- 単純な全文検索だけでなく,coderidge-rdf の検索式を利用した検索も行なうことができる.

8 まとめ

開発者に優しくスケーラビリティのあるソフトウェアマップの実装として, CodeRidge を開発した. CodeRidge は, 開発者の手元で動作し P2P ネットワークを介してソフトウェア情報の更新を即時に反映する.

参考文献

[1] http://www.fsf.org/.

[2] http://www.gnu.org/.

[3] http://www.vasoftware.com/.

[4] http://sourceforge.net/.

[5] http://freshmeat.net/browse/.

[6] http://www.gnu.org/directory/.

[7] http://sourceforge.net/softwaremap/.

[8] http://www.googole.com/.

[9] http://coderidge.org/.

[10] Resource Description Framework(RDF) Model and Syntax Specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[11] Simple Object Access Protocol (SOAP) 1.1. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/.

[12] Sun Microsystems. Java™ Remote Method Invocation (RMI). http://java.sun.com/products/jdk/rmi/.

[13] XSL Transformations (XSLT) Version 1.0. http://www.w3.org/TR/1999/REC-xslt-19991116.

[14] http://raa.ruby-lang.org/.

[15] Project JXTA. http://www.jxta.org/.

[16] Sun Microsystems. Java™ Servlet Technology. http://java.sun.com/products/servlet/index.html.

[17] Sun Microsystems. Java Server Pages™ Technology. http://java.sun.com/products/jsp/.

[18] http://jakarta.apache.org/tomcat/.

[19] http://ws.apache.org/axis/.

[20] Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[21] Sun Microsystems. Java™ 2 Platform, Standard Edition (J2SE™). http://java.sun.com/j2se/1.4/.

[22] N-Triples (W3C RDF Core WG Internal Working Draft). http://www.w3.org/2001/sw/RDFCore/ntriples/.

[23] D. Crocker and P. Overell. RFC 2234:Augmented BNF for Syntax Specifications: ABNF, Nov. 1997.

[24] http://e-words.jp/w/XSS.html.

[25] Sun Microsystems. Java Server Pages™ Tag Libraries. http://java.sun.com/products/jsp/taglibraries.html.

[26] Paul Lee. クロスサイト・スクリプティング: カスタム・タグ・ライブラリーを使って、動的コンテンツをエンコードする. http://www-6.ibm.com/jp/developerworks/security/021115/j_s-csscript.html.

[27] http://fanal.org/.

[28] http://cms.jxta.org/.

[29] 春山征吾. P2P フレームワーク FANAL. http://www.unixuser.org/%7Eharuyama/P2P/FANAL_ja.pdf.