協調可能なソフトウェアマップ “CodeRidge” の開発

上野 乃毅 <ueno@unixuser.org>
春山 征吾 <haruyama@unixuser.org>
山崎 義弘 <yoya@unixuser.org>

発表の概要

- 背景と動機
- CodeRidge の特徴
- CodeRidge で利用している技術
- デモ
- まとめ

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
- 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

開発（登録）者側 ソフトウェアを広く宣伝することができる
利用（検索）者側 目的のソフトウェアを効率的に検索可能

既存の実装

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

既存のソフトウェアマップは中心化されている

既存のソフトウェアマップは，サーバに情報をためておきそれを配付する．

- Version などを変更したり
- 開発・配付サイトを移転したり

するたびに，
HTML の Form などを使って変更する必要がある

また

複数のソフトウェアマップ間での検索をするのは面倒
(結局 Google で検索してしまう)

Google での検索

- 公開されている (た) 情報が収集され，Google の基準 (リンク数など) でランク付けされている
- 変更した情報がいつ反映されるかはわからない
- 古い情報が残っていることもある (便利な場合もあるが)
- 検索に技術が必要なことがある
  (一般名詞がソフトウェア名となっている場合 (例: lookup) など，キーワードを適当に足さないと目的の情報を絞りこめない，など)

フリーソフトウェア開発の現状

ソフトウェア開発とその成果の公開において

- SourceForge (など) を利用する
- 自分サーバで公開しソフトウェアマップに登録する
- 自分サーバで公開し検索エンジンにひろってもらう

というフリーソフトウェア開発の主な手法である 3 例を考える

SourceForge

- 開発 (登録) 者から見て
  - 整理されている
  - 自由度は少ない
  - 登録・更新が手間
- 利用 (検索) 者から見て
  - カテゴリごとなどに整理されており，目的のソフトウェアを効率的に検索可能
  - 更新された情報は (ほぼ) 即時に反映されている
  - 目当てのソフトウェアが登録されていないこともある
  - 開発者が更新をさぼっているかもしれない

自分サーバ ＋ ソフトウェアマップ

- 開発 (登録) 者から見て
  - **自由度は多い**
  - **すべて自分で配置・管理を見るのは面倒**
  - 登録・更新が手間
- 利用 (検索) 者から見て
  - カテゴリごとなどに整理されており，目的のソフトウェアを効率的に検索可能
  - 更新された情報は (ほぼ) 即時に反映されている
  - 目当てのソフトウェアが登録されていないこともある
  - 開発者が更新をさぼっているかもしれない

自分サーバで公開 ＋ 検索エンジン

- 開発 (登録) 者から見て
  - 自由度は多い
  - すべて自分で配置・管理を見るのは面倒
  - **公開しておけば勝手に登録・反映される**
  - **意図通りに検索エンジンを制御することは難しい**
- 利用 (検索) 者から見て
  - **ソフトウェア情報という点からは整理されてないので検索により技術が必要なこともある**
  - **現時点での最新情報が検索できるとは限らない．**

自分サーバでの開発に好ましいソフトウェアマップ

情報の分散化 中央サーバは使わず，
開発者が自由にいじれる場所に情報を置く
分散化した情報の共有 P2P を利用する
自由度を保ちつつ手軽さを実現 共有する情報においては，枠組と基本的な要素は定義し 自由な拡張を許す
登録や更新を簡単に
- 複数の登録・更新手段を持つ
  - 更新は即時に反映される

SourceForge と自分サーバを両方利用する場合などでも，情報を統一的に扱える

ソフトウェアマップの分散化

前ページのようなアプローチで

開発（登録）者
- 開発環境に高い自由度
  - ソフトウェア情報更新の半自動化

利用（検索）者 既存のソフトウェアマップの様に整理された情報を取得

が達成できるのではないか

このような **情報生成の自由度とその情報の整理の問題** は，ソフトウェアだけではなく一般的な問題となると思われるので，これがよりよい情報共有の一歩となるといい

CodeRidge の特徴

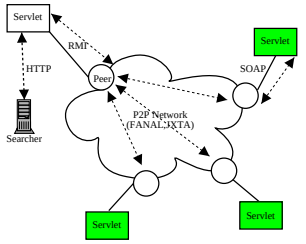
分散ソフトウェアマップ CodeRidge の特徴

- ソフトウェア情報は P2P を利用して分散化されている
- ソフトウェアを登録した Peer を起動すれば，自動的に全体からの検索の対象となる
- ソフトウェアの情報は RDF で定義されており，開発者が自由に構成・拡張できる
  - HTML の Form からも入力できるし
  - 手でいじることもできるし
  - 適当なソースから自動的に生成することもできる
- RDF のみを利用しており DBMS などは必要ない
- 標準的な技術を利用しており，Java でなくても実装可能

CodeRidge の解説

- 概観
- 検索 (RDF)
- 分散化 (P2P,FANAL)
- Servlet-Peer 間通信 (SOAP)
- 既存のソフトウェアマップとの連携 (Freshmeat,RAA)
- データの取扱い (T4,XSLT)

CodeRidge 概観



CodeRidge 概観

検索がどのように行なわれるか (リクエスト)

- ユーザがサーブレットに検索を要求する
- サーブレットから Java RMI を利用して JXTA Peer にリクエスト発行要求が送られる
- JXTA Peer から JXTA Network に対してリクエストが投げられる
- 他の JXTA Peer が そのリクエストに対応する
- SOAP を利用して サーブレットにリクエストを投げる

CodeRidge 概観

検索がどのように行なわれるか (レスポンス)

- RDF によるデータベースに対して検索が行なわれ，結果が SOAP を利用して返される
- JXTA Network を介して検索元の JXTA Peer に結果が返る
- RMI を利用して 結果を取得する
- 他の JXTA Peer でも同様の処理が行なわれ，結果が返る

ソフトウェアについて詳細な情報を得る場合に，特定の Peer に対して問い合わせが行く．

検索

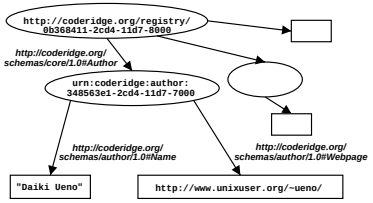
RDF モデルによる索引の表現:

- RDF (Resource Description Framework)
- 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) の三者関係によりメタ情報を表現する手法
- 半構造のデータ構造 (スキーマの存在を仮定しなくてよい)
- プロパティに対しても一意な URI (qualified URI) が割当てられるため，同名のプロパティを識別可能

RDF によるソフトウェア情報の記述

```
<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>
```


RDF モデル



検索式

複雑な検索式のサポート:

- 文字列の完全一致と正規表現によるマッチング
- 一階の述語論理式による検索式の組み合わせ

Java API による検索の手順

```
// 検索式のコンパイル
Query query = new Query ("authorName = /Ueno/", keywords);

// 一致したデータの抽出
Resource[] subjects = query.match (triples);
triples = triples.filter (subjects[0]);
TripleClassifier classifier =
    new TripleClassifier (triples, new TreapMap ());

// プロパティ値の取得
Object[] properties = classifier.getObjectResolveReference
    ("http://coderidge.org/schemas/core/1.0#Webpage");
```


分散化 (P2P)

P2P の実装として JXTA を利用する

- JXTA License のもとで利用できる
- Firewall 越しの通信が考慮されている
- 一般の Peer は固定した IP アドレスを持つ必要がない
- Java の実装がある

CodeRidge では,
Peer を起動すれば CodeRidge ネットワーク全体からその Peer が持つ情報を検索できるようになる。

JXTA の上に フレームワーク FANAL を構築した.

FANAL

P2P での リクエスト–レスポンス システムを構築するためのフレームワーク

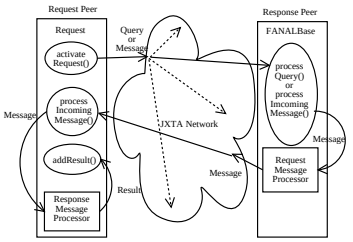
JXTA CMS (ファイル共有システム) を参考にして,
より抽象化したもの.

リクエストの種類

- Multicast (検索に用いる)
- Unicast (検索結果の取得に用いる)

情報共有などの P2P アプリケーションの構築が容易にできる

FANAL 概要



SOAP

SOAP (Simple Object Access Protocol)

JXTA (FANAL) Peer と ソフトウェア情報 DB (Servlet) との通信に利用.
⇝ JXTA プロトコルやプログラム言語に非依存になる.

実装には Axis を利用.

WSDL の利用

WSDL (Web Services Description Language, Web サービスのインタフェイスを記述する言語) にてインタフェイスを定義し, Axis の WSDL2Java ツールを利用して Java のコードを生成した. (WSDL から他の言語のコードを生成することも可能)

WSDL での記述

```
...
 <message name="getContentRequest">
    <part name="uri" type="xsd:string"/>
  </message>
  <message name="getContentResponse">
    <part name="result" type="xsd:base64Binary"/>
  </message>
...
 <operation name="getContent"
              parameterOrder="uri">
      <input message="tns:getContentRequest"/>
      <output message="tns:getContentResponse"/>
 </operation>
...
```


生成される Java コード (クライアント側)

```
...
    public byte[] getContent (java.lang.String uri)
        throws java.rmi.RemoteException {
      if (super.cachedEndpoint == null) {
          throw new org.apache.axis.NoEndPointException ();
      }
      org.apache.axis.client.Call _call = createCall ();
      _call.setOperation (_operations[3]);
      _call.setUseSOAPAction (true);
      _call.setSOAPActionURI ("getContent");
      _call.setOperationName (new javax.xml.namespace.QName
          ("http://coderidge.org/soap/1.0", "getContent"));
      setRequestHeaders (_call);
      ...
    }
```

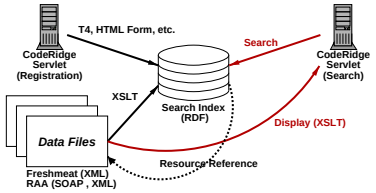

既存のソフトウェアマップとの連携

CodeRidge は, 構造化されたソフトウェアの情報を提供しているソフトウェアマップと連携が可能

Freshmeat プロジェクト情報 (XML) を HTTP で配布
Ruby Application Archive (RAA) SOAP で情報を取得可能

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作成することもできる。

データの取扱い



T4

TEX 風の構文をもつマクロプロセッサ

なぜマクロプロセッサ?

- 独自形式の構造化文書が氾濫している
  POD や RD, 各種 Wiki
- マクロプロセッサとして実装すれば, 出力形式や実装言語を意識せずに, 利用者が自由に構文を拡張できる

なぜ TEX?

- XML よりは書きやすい
- 段落を引数に取るマクロを書ける

T4 マクロによるソフトウェア情報の記述

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

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


XSLT

Contents (FreshMeat の XML,RAA の情報) から

- 検索用 RDF
- XHTML

への変換を XSLT (XSL Transformations) を利用して行う.

CodeRidge 独自の情報は RDF から XHTML への変換を XSLT を利用して行う.

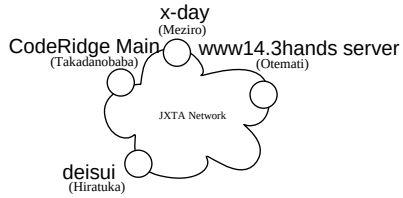
デモ

- ソフトウェアマップとしての機能
- P2P による検索

のデモを行なう

デモ

Peer の配置:



TODO

- 情報のキャッシュ
- プロジェクトのモニタ
- 別ブランチでの開発 (プロジェクトの fork) 支援
- UI の改善
- 簡単に配置できるようにする

まとめ

分散ソフトウェアマップ “CodeRidge” を開発中

自分サーバでのソフトウェア開発を支援

- Peer を起動すれば検索の対象となるのでソフトウェアマップへの登録や更新の手間を削減する
- 既存のソフトウェアマップのように整理された情報を提供できる

今後, 高速な回線とマシンの普及により開発環境の分散化がさらに進むと思われるので, それとも協調したい

また 既存のソフトウェアマップとの連携を深めたい

各種情報

http://coderidge.org/
http://sourceforge.net/projects/coderidge/
http://fanal.org/
http://fanal.jxta.org/
テストサーバ
未踏 Wiki CodeRidge