ソフトウェア品質管理に向けた静的メトリックスの検討

東京工芸大学工学部コンピュータ応用学科
宇田川　佳久・新部　忠

研究の最終目的(チャレンジ):

Javaオープンソースを使った高信頼システムの構築技法・環境の確立

発表の流れ:

1. 研究の背景
2. ソフトウェアメトリックスの概要
3. JavaフレームワークStruts
4. Strutsのソフトウェアメトリックス
5. ソースコード追跡とメトリックス
6. 今後の研究方針

1. 研究の背景 (1/2)

- Javaフレームワークや Androidなど、Javaをベースとしたシステム開発が盛んになってきた
- 信頼性の観点から、オープンソースプログラムにまで立ち入った理解が要求されることがある
- 開発者によるプログラムの理解を支援する観点から、下記について検証した。
  - **ソースを追跡する機能**
  - **メトリックスによるソースの特質の数値化**
  - ◆ **対象は Struts 1.3フレームワーク**

発表の流れ:

1. 研究の背景
2. ソフトウェアメトリックスの概要
3. JavaフレームワークStruts
4. Strutsのソフトウェアメトリックス
5. ソースコード追跡とメトリックス
6. 今後の研究方針

ソフトウェアメトリックスの分類

1. ソースコードの大きさに関するメトリックス
2. ソースコードの複雑さに関するメトリックス
3. オブジェクト指向に関するメトリックス

ソースコードの大きさに関するメトリックス

No	メトリックス略称	内容
1	CountLine	ソースコードの物理行数.
2	CountLineBlank	空白行の行数.
3	CountLineCode	ソースコードの論理行数. 1つの行に実行文とコメントを含む場合は，実行文およびコメント文としてカウントする.
4	CountLineCodeDecl	宣言文を含むソースコードの行数.
5	CountLineCodeExe	実行文を含むソースコードの行数.
6	CountLineComment	コメント文を含むソースコードの行数.
7	CountSemicolon	セミコロンの数.
8	CountStmt	宣言文と実行文の合計数.
9	CountStmtDecl	宣言文の数.
10	CountStmtExe	実行文の数.
11	RatioCommentToCode	コメント行数÷ソースコード数.

ソースコードの 複雑さに関するメトリックス

No	メトリックス略称	内容
1	CountInput	入力引数の数．入力引数にはパラメータとしての引数と広域変数を含む．
2	CountOutput	出力引数の数．出力引数にはパラメータとしての引数と広域変数を含む．
3	CountPath	ソース本体に含まれるパスの数．
4	Cyclomatic[*]	循環的複雑度[*]．
5	CyclomaticModified	修正循環的複雑度．循環的複雑度において，Switch文を1とカウントする．
6	CyclomaticStrict	厳密循環的複雑度．循環的複雑度において，各分岐条件を1とカウントする．例えば，if(A＆B＆C)は，厳密循環的複雑度では3と数える．
7	Essential	本質循環的複雑度．循環的複雑度において，入力点と出力点が1個である構造を1とカウントする．
8	MaxNesting	制御構造のネストレベルの最大数．

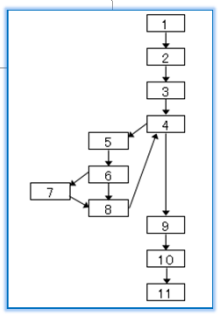
[*] CC = E − N + P．ここで，E= グラフのエッジ数，N= グラフのノード数，P= 連結されたコンポーネントの数．

循環的複雑度（Cyclomatic）

---制御構造の複雑度の一つ

- T. McCabe により提案された（1976年）
- **CC= E − N + P**
  - E= グラフのエッジ数
  - N= グラフのノード数
  - P= 連結されたコンポーネントの数

```
1   public static String RmvSpace( String InST ) {
2         String OutST= "";
3         String ST= "";
4         for( int j=0; j<InST.length(); j++){
5                 ST= InST.substring(j, j+1);
6                 if ( ! ST.equals(" ") ) {
7                       OutST= OutST+ ST;
8                 }
9         }
10        return(OutST);
11  }
```



E=12, N=11, P=1 であるから，**CC=2**

オブジェクト指向に関するメトリックス

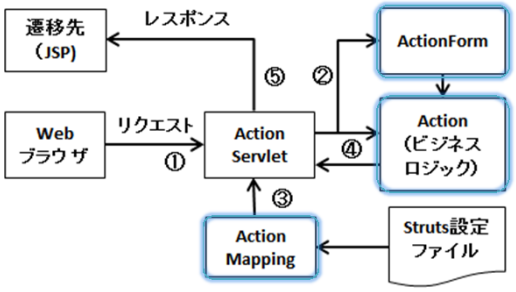
No	メトリックス略称	内容
1	LackOfCohesion	結合性の欠如の割合(100% から「クラスデータ メンバーの平均結合性」を引いた数)
2	MaxInheritanceTree	最大の継承ツリーの高さ
3	CountClassBase	基底クラスの数
4	CountClassCoupled	結合されたクラスの数
5	CountClassDerived	派生クラスの数(当該クラスを直上位クラスとするクラスの数)
6	CountDeclMethodAll	利用可能なメソッドの数(継承されたメソッドも含む)
7	CountDeclInstance Method	インスタンスメソッドの数
8	CountDeclInstance Variable	インスタンス変数の数
9	CountDeclMethod	ローカルメソッドの数

発表の流れ:

1. 研究の背景
2. ソフトウェアメトリックスの概要
3. JavaフレームワークStruts
4. Strutsのソフトウェアメトリックス
5. ソースコード追跡とメトリックス
6. 今後の研究方針

JavaフレームワークStruts

- Strutsは，JavaベースのWebフレームワーク
- 業務システムで多くの採用実績がある



ActionFormの定義

- ブラウザからの入力データを保持する

```
package Struts.Sample;
import org.apache.struts.action.ActionForm;
public class LogonForm extends ActionForm {
   private String UserID;          // UserID
   private String PassWD;          // Password
   private String PrjID;           // ProjectID
   public String getPassWD() {
      return PassWD; }             // Get Password
   public void setPassWD(String PassWD) {
      this.PassWD = PassWD; } // Set Password
}
```

Action.executeメソッドの定義

```
package Struts.Sample;
import org.apache.struts.action.Action;
public class LogonAction extends Action {
  public ActionForward execute(
    ActionMapping map,   ActionForm form,
    HttpServletRequest req,
    HttpServletResponse resp)  {
      <アプリケーション固有の処理>
    return mapping.findForward("SUCCESS1"); // 処理成功
    return mapping.findForward("ERROR1");   // 処理成功
  }
}
```

struts-config.xml の定義例

```
<form-beans>
  <form-bean name="LogonForm"
        type="Struts.Sample.LogonForm" />
</form-beans>
<action-mappings>  <action
     name="LogonForm"
     path="/logon"
     scope="request"
     type="Struts.Sample.LogonAction">
    <forward name="SUCCESS1" path="/Success1.jsp" />
    <forward name="ERROR1" path="/Error1.jsp" />
  </action> </action-mappings>
```

発表の流れ:

1. 研究の背景
2. ソフトウェアメトリックスの概要
3. JavaフレームワークStruts
4. Strutsのソフトウェアメトリックス
5. ソースコード追跡とメトリックス
6. 今後の研究方針

Strutsのソフトウェアメトリックス

ソースコードの大きさ

Blank Lines	4,512
Classes	**149**
Code Lines	**13,974**
Comment Lines	14,114
Comment to Code Ratio	1.01
Declarative Statements	4,237
Executable Statements	5,347
Files	**133**
Functions	**1,485**
Lines	32,538

モジュールごとのメトリックス（ソースコードの大きさ）

	Action	Chain	Config	Mock	Upload	Util	Validator
Java File	21	53	17	13	4	14	11
CountDeclClass	23	53	27	15	5	15	11
CountDeclFile	21	53	17	13	4	14	11
CountDeclFunction	207	281	359	283	53	162	140
CountLine	6847	6449	6824	2628	1014	4287	4489
CountLineBlank	875	848	1049	444	133	558	605
CountLineCode	2737	2302	2856	1532	374	1689	2484
CountLineComment	3243	3301	2922	687	507	2051	1403
CountStmtDecl	730	963	791	466	149	470	668
CountStmtExe	1119	670	1245	573	127	682	931
RatioCommentToCode	1.185	1.434	1.023	0.448	1.356	1.214	0.565
空白率	0.128	0.131	0.154	0.169	0.131	0.130	0.135

ソースの複雑さ

CountLine	557
CountLineBlank	59
CountLineCode	156
CountLineCodeDecl	58
CountLineCodeExe	61
CountLineComment	342
CountSemicolon	67
CountStmt	104
CountStmtDecl	46
CountStmtExe	58
MaxCyclomatic	4
MaxCyclomaticModified	4
MaxCyclomaticStrict	4
RatioCommentToCode	2.19
SumCyclomatic	36
SumCyclomaticModified	36
SumCyclomaticStrict	40

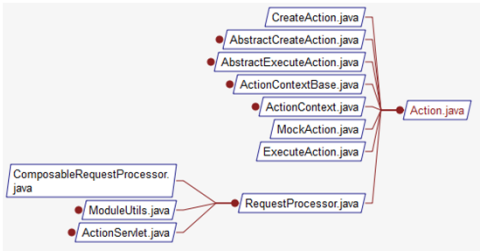
循環的複雑度を計測した例

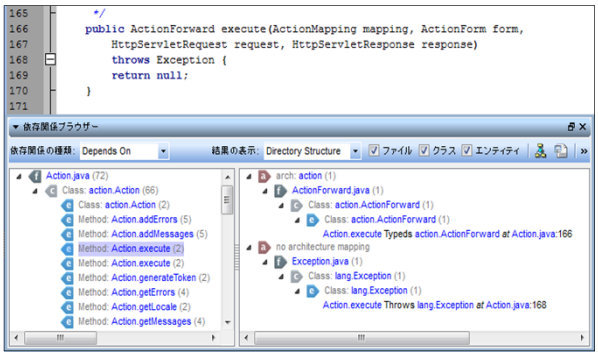
発表の流れ:

1. 研究の背景
2. ソフトウェアメトリックスの概要
3. JavaフレームワークStruts
4. Strutsのソフトウェアメトリックス
5. ソースコード追跡とメトリックス
6. 今後の研究方針

ソースコード追跡とメトリックス

- ソースコードを追跡する様々な機能がサポートされている
- メトリックスを参考データとして表示することも可能





メトリックスの類似とコードの検索

- 障害の横展開では、類似コードの検索が必須
- 下記のメトリックスでは、類似コードは検索できなかった
  - CountLin （コードの総数）
  - CountStmtExe （実行行の数）
  - Cyclomatic （循環的複雑度）
  - MaxNesting （ネストの深さ）

発表の流れ:

1. 研究の背景
2. ソフトウェアメトリックスの概要
3. JavaフレームワークStruts
4. Strutsのソフトウェアメトリックス
5. ソースコード追跡とメトリックス
6. 今後の研究方針

6. 今後の研究方針

- 結論
  既存のメトリックスや追跡機能では、コードの理解を十分に支援できるとは言えない
- 今後の研究方針
  - ◆制御文の種類を考慮したメトリックスの提案
  - ◆他のオープンソースへの適用実験

ご清聴ありがとうございました