ソフトウェア メトリクス

2011年7月27日

海谷 治彦

目次

- 概要
- 代表的なソフトウェアメトリクス
- ツール

Metrics

- メトリクス
- 辞書的意味: 測定基準
- ソフトウェアの測定基準
  - ソフトウェアおよびその開発にかかわる物や事を測るための測定単位(身長，体重，肥満度，年齢のようなもの).
- ある一つのソフトウェアやその開発について，多数のメトリクスで測れる.
  - ある人間を，身長，体重，年齢，肥満度等，多数の基準で測れるのと同じ.

ソフトウェアの何が測れる？

- 機能量
  - 計算機で実行される機能の量.
  - ソースコードの長さ.
- 開発期間
  - 作るのにどれだけの期間かかったか？
- 工数
  - 作るのに費やした時間は延べどれくらいか？
  - 通常は「人月」という単位を利用.
- 生産性
  - 機能量/開発期間 もしくは 機能量/工数
- 信頼性
  - 欠陥の数/機能量 もしくは 欠陥の数/期間

測ってどうする？

人の身長，体重，年齢，肥満度を測ってどうする？と基本的には同じ理由で測る．

- 「普通」と比べて逸脱してるかを知る．
  - 普通，〇〇ヶ月でできるでしょ・・・でも・・・
- 「以前」と比べて変化しているかを知る．
  - 丸一日，見直ししたらバグが減った・・・
- 将来予測(もしくは決定)をする．
  - あと，〇〇日でできるだろう・・・
  - あるメトリクスと別のメトリクスに相関があることを大抵前提とする．
    - 実際には，開発期間等，オトナの都合で変更できないメトリクスに基づき，機能の量や，信頼性の度合いが決まる・・・

尺度と測定結果の比較

- あるメトリクスで測定した値同士を比較することは前述の通り重要.
- あるメトリクスがどんな尺度かによって，どんな比較ができるか決まる.
- 名義尺度 Nominal Scale 区別
  - 値間に区別があるだけで大小等は無い.
  - 例: 性別=男 or 女
- 順序尺度 Ordinal Scale 区別+順序
  - 順序のみに意味がある.
  - 例: 競技の順位．1位と2位の差が，2位と3位の差と同じとは言えない.
- 間隔尺度 Interval Scale 区別+順序+間隔
  - 間隔の大きさには意味があるが，何倍とかは言えない.
  - 例: 摂氏温度(℃) 20℃は10℃より二倍熱いわけではない.
- 比例尺度 Ratio Scale 区別+順序+間隔+比率
  - 間隔の大きさにも，比率にも意味がある.
  - 例: 絶対温度 (K) 20Kは10Kの二倍の熱量と言える.

値の評価

- メトリクス毎，状況毎に値の評価は異なる.
- 順序尺度以上の場合，例えば，大きいと良いとか，悪いとかの評価を与えられる.
  - 体重は軽いほうが良い. (主観です)
  - 肥満度は小さいほうが良い.
- 比率尺度の場合，「何倍」良いとか悪いとかも言える.
  - 速度が三倍速いので良い.
  - 開発時間が5倍かかったのでサイアク.

どうやって測る？

- 直接測れるメトリクス
  - 値を直接測れるもの.
  - 身長や体重等.
  - ソフトウェアならコードの行数，作業時間．機能(要求)の数，クラスやメソッドの数.
  - 数を主観的に決めなければいけない場合もある.
- 間接的なメトリクス
  - 直接測れるものから計算で求める.
  - 肥満度は・・・多分，計算式があるでしょう.
  - 通常，生産性=コード行数/作業時間

相関

- 間接的なメトリクスと直接的なものの間には，論理的，構造的な相関がある．
  - 肥満度は身長と体重を構成要素とする(はず)なので，肥満度と身長，肥満度と体重には相関がある．
  - 生産性，コード量，作業時間等も同様．
- 経験的な相関もある．
  - B型には立派な人が多い．
  - 同じ規模のシステムを作るには，同じくらいの時間がかかる．
  - 長い関数(メソッド)にはバグが多い．
  - 赤い(色)と三倍速い(速度)．

ソフトウェアメトリクスの例

ソフトウェアメトリクスの実例

- コードメトリクス
  - LOC
  - サイクロマティック数
  - 凝集度 Cohesion
  - 結合度 Coupling
  - バグ
- 要求や仕様のメトリクス
  - ファンクションポイント
  - ユースケースポイント
- プロセスのメトリクス
  - 作業時間，期間
  - コードや設計等の変更数
- 上記を組み合わせたものもある.

Lines of Codes (LOC)

- プログラムの行数.
- 数え方は多様
  - そのまま数える
  - コメントは除く
  - 空行は除く
  - 命令文の数を数える

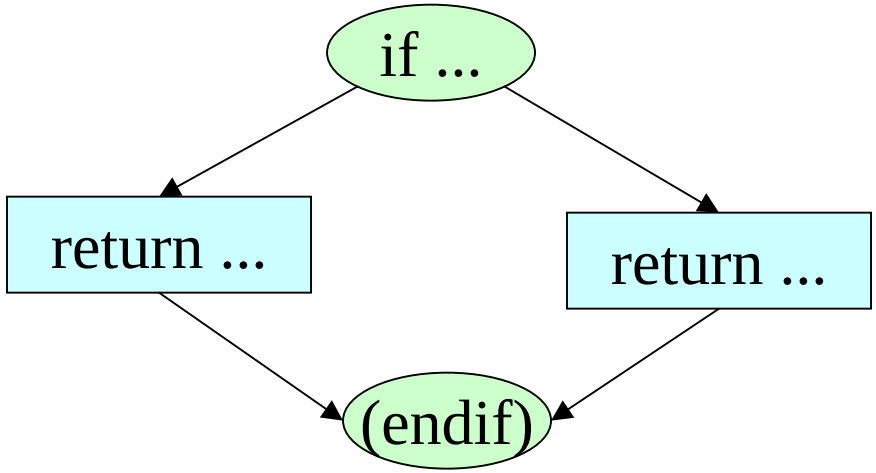
  等々
- 単純なメトリクスではあるが，強力.
  - 通常，自動的に測定可能
  - 長い関数やメソッドは通常，欠陥が多い傾向にある.

サイクロマティック数

- McCabe Cyclomatic Complexity (MCC)
  - McCabeは人の名前
- 関数等の複雑さの度合いなので，大きいほど悪い.
- MCC= cond + 1
  - cond: if while for等の二分決定数の数.
    - while for はループを続けるか否かの二択
    - 単純ifやif elseも二択.
    - if-elseif-else は三択(二択が二つ).

例

```
public Product[] getProducts() {
  if(rproducts==null)
    return new Product[0];
  else
    return rproducts.toArray(new Product[0]);
}
```



MCC=1+1=2

凝集度 Cohesion

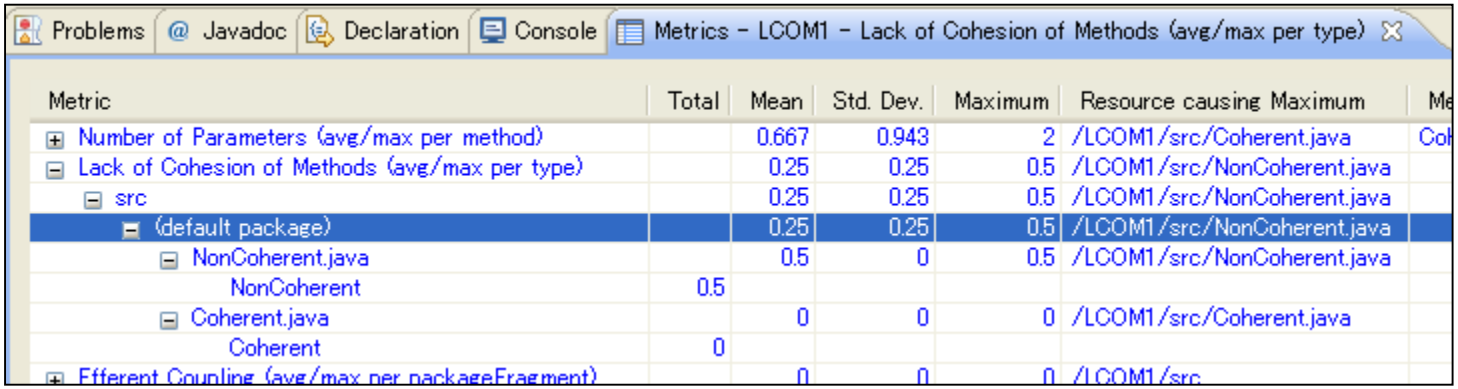
- あるクラスが，どの程度まとまっているかのメトリクス.
- 相互に関係無い概念をまとめてクラスにしたりすると，Cohesionが下がる.
- 基本，まとまっていたほうがいい.

- 通常は凝集度の欠如(Lack of Cohesion of Methods: LCOM)のほうをメトリクスとする.

わざとらしい例

```
public class Coherent {
public int a;
public int b;
public Coherent(int a, int b){
        this.a=a;
        this.b=b;
}

public int add(){return a+b;}

public int sub(){return a-b;}
}
```

```
public class NonCoherent {
public int a;
public int b;
public NonCoherent(int a, int b){
        this.a=a;
        this.b=b;
}
public int getA(){return a;}

public int getB(){return b;}

}
```



Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Me
Number of Parameters (avg/max per method)		0.667	0.943	2	/LCOM1/src/Coherent.java	Col
Lack of Cohesion of Methods (avg/max per type)		0.25	0.25	0.5	/LCOM1/src/NonCoherent.java	
src		0.25	0.25	0.5	/LCOM1/src/NonCoherent.java	
(default package)		0.25	0.25	0.5	/LCOM1/src/NonCoherent.java	
NonCoherent.java		0.5	0	0.5	/LCOM1/src/NonCoherent.java	
NonCoherent	0.5					
Coherent.java		0	0	0	/LCOM1/src/Coherent.java	
Coherent	0					
Efferent Coupling (avg/max per packageFragment)		0	0	0	/LCOM1/src	

実例

Problems | Javadoc | Declaration | Console | Metrics - ex2a - Lack of Cohesion of Methods (avg/max per type)

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		0.694	0.775	3	/ex2a/src/ShoppingSystem.java	buyProduct
Lack of Cohesion of Methods (avg/max per type)		0.181	0.256	0.583	/ex2a/src/Customer.java	
src		0.181	0.256	0.583	/ex2a/src/Customer.java	
(default package)		0.181	0.256	0.583	/ex2a/src/Customer.java	
Customer.java		0.583	0	0.583	/ex2a/src/Customer.java	
Customer	0.583					
ShoppingSystem.java		0.5	0	0.5	/ex2a/src/ShoppingSystem.java	
ShoppingSystem	0.5					
Ex2GUI.java		0	0	0	/ex2a/src/Ex2GUI.java	
Ex2.java		0	0	0	/ex2a/src/Ex2.java	

Problems | Javadoc | Declaration | Console | Metrics - ex2b - Lack of Cohesion of Methods (avg/max per type)

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Parameters (avg/max per method)		0.689	0.812	3	/ex2b/src/ShoppingSystem.java	buyProduct
Lack of Cohesion of Methods (avg/max per type)		0.248	0.29	0.667	/ex2b/src/Sale.java	
src		0.248	0.29	0.667	/ex2b/src/Sale.java	
(default package)		0.248	0.29	0.667	/ex2b/src/Sale.java	
Sale.java		0.667	0	0.667	/ex2b/src/Sale.java	
Customer.java		0.571	0	0.571	/ex2b/src/Customer.java	
Customer	0.571					
ShoppingSystem.java		0.5	0	0.5	/ex2b/src/ShoppingSystem.java	
ShoppingSystem	0.5					
Product.java		0	0	0	/ex2b/src/Product.java	
Ex2.java		0	0	0	/ex2b/src/Ex2.java	
LoadSave.java		0	0	0	/ex2b/src/LoadSave.java	
Ex2GUI.java		0	0	0	/ex2b/src/Ex2GUI.java	
Efferent Coupling (avg/max per packageFragment)		0	0	0	/ex2b/src	

演習2の解答例の場合，b型(Saleクラスを使ってる)のほうが，ちょっとだけ，凝集度の欠如(LCOM)が下がってる．

結合度 Coupling

- パッケージ間(もしくはクラス間)の結びつきの強さを示す.
- 結びつきが弱ければ，個々に再利用しやすい.
- 強ければ，セットで使わないといけないので，ちょっと面倒.

例: 簡単なUMLエディタ

Problems | Javadoc | Declaration | Console | Metrics - violet1 - Afferent Coupling (avg/max per packageFragment)

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
Number of Overridden Methods (avg/max per type)	67	0.985	1.398	6	/violet1/src/com/horstmann/v...	
Number of Attributes (avg/max per type)	122	1.794	2.731	14	/violet1/src/com/horstmann/v...	
Number of Children (avg/max per type)	36	0.529	1.728	12	/violet1/src/com/horstmann/v...	
Number of Classes (avg/max per packageFragment)	68	34	2	36	/violet1/src/com/horstmann/v...	
Method Lines of Code (avg/max per method)	4052	9.859	25.971	423	/violet1/src/com/horstmann/v...	EditorFra...
Number of Methods (avg/max per type)	389	5.721	4.902	19	/violet1/src/com/horstmann/v...	
Nested Block Depth (avg/max per method)		1.348	0.821	7	/violet1/src/com/horstmann/v...	GraphPanel
Depth of Inheritance Tree (avg/max per type)		2.544	1.288	6	/violet1/src/com/horstmann/v...	
Number of Packages	2					
Afferent Coupling (avg/max per packageFragment)		17	14	31	/violet1/src/com/horstmann/v...	
src		17	14	31	/violet1/src/com/horstmann/v...	
com.horstmann.violet.framework	31					
com.horstmann.violet	3					
Number of Interfaces (avg/max per packageFragment)	4	2	2	4	/violet1/src/com/horstmann/v...	
McCabe Cyclomatic Complexity (avg/max per method)		2.002	2.252	24	/violet1/src/com/horstmann/v...	getPath
Total Lines of Code	6626					
Instability (avg/max per packageFragment)		0.5	0.412	0.912	/violet1/src/com/horstmann/v...	
Number of Parameters (avg/max per method)		0.912	1.077	6	/violet1/src/com/horstmann/v...	drawString
Lack of Cohesion of Methods (avg/max per type)		0.2	0.291	0.846	/violet1/src/com/horstmann/v...	
Efferent Coupling (avg/max per packageFragment)		17	14	31	/violet1/src/com/horstmann/v...	
src		17	14	31	/violet1/src/com/horstmann/v...	
com.horstmann.violet	31					
com.horstmann.violet.framework	3					
Number of Static Methods (avg/max per type)	22	0.324	0.756	3	/violet1/src/com/horstmann/v...	

violetはviolet.frameworkに大きく依存してます. (31>3)
よって, violet単体で使うのは困難です.
(afferent 導入 efferent 導出)

バグ

- 所謂，欠陥，fault, defectに相当.
- 自動的に計測できない場合が多い.
- 「理解」に関するバグ
  - 仕様通りにプログラムされていない.
  - 最も重大かつ発見が困難なバグ
- 「整理」に関するバグ
  - コンピュータにやらせたいことを，コンピュータに都合よい形で整理されてない.
  - 設計に関するバグといってよい.
  - 識別がグレーなバグ.
- 「翻訳」に関するバグ
  - プログラム言語の文法エラー.
  - 昨今ではIDE(Eclipse等)が勝手に直してくれる場合もある.
  - 文法的には合っていても，異なる意味となってしまう場合もあるので，完全な自動検知は困難.

理解に関するバグの例

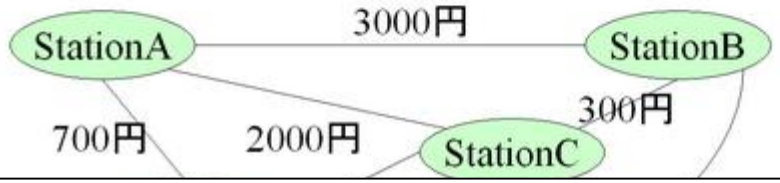
2011 ソフトウェア工学 演習1

問題

オイスターカード(London)やSuica(関東)等の，先払い式の電子切符の役割を模倣する
る．

- 入金することで先払い運賃をカードに追加する，所謂，Top-upする．
- 乗車する駅を記録する．
- 下車する駅において料金を支払う．支払えない場合，改札を出れない・・・

駅もクラスとして実現し，本演習では以下のStationA, B, C, Dの4つの駅が，以下のよ
由して目的駅に到着したとしても，運賃は途中駅を経由せず移動した場合の金額(最



「追加」してないじゃん！(バグ)

```
private int amount;
public void topUp(int a){
  amount = a;
}
```

とある手法におけるバグ分類

- 文書: コメント文やメッセージ記述のバグ
- 構文: スペリング，区切り，誤字，命令形式
- ビルド・パッケージ: 変更管理，ライブラリ，版管理
- アサインメント: 代入や変数宣言
- インタフェース: 関数，メソッド等の呼び出し．入出力関係．
- チェッキング: エラーチェックの不備
- データ: 構造や内容
- 機能: 論理，ポインタ，ループ，再帰，計算，機能欠陥
- システム: 構成，タイミング，メモリ
- 環境: 設計，コンパイル，テスト，支援システムの問題

W. Hamphrey. A Discipline for Software Engineering. 1995.

バグを数えたり分類してどうする？

- プログラムの単位行あたりのバグ数が，ソフトウェアの品質とされることが多い．
  - 品質を測るために，バグを数える．
  - 無論，品質は高いほうがいい．
- デバッグが進捗していることを確認する．
  - 品質が上がっていることを確認する．
- 同じ人や組織は同じ傾向のバグを出しやすい．
  - デバッグで，どこに重点を入れるか決められる．

ファンクションポイント (FP)

- Function Point
- ソフトウェアがどのようにできているか(行数等)では無く，何をしてくれるかに基づき，ソフトウェアの規模を測るメトリクスの代表.
- 要求や仕様を入力として，当該ソフトウェアがどれくらい大きいかが分かる.
- 大きいソフトなら，作るのに時間や費用がかかるという説明も無理なく行える
  - ことを信じたい・・・

FPの基本

- 仕様書を見て，システムの入力，出力，インタフェース，データベース等を見つける.
  - これらを構成要素と呼ぶ.
- 構成要素毎に，容易，普通，複雑の点数をつける.
  - 点数は表で与えられる．表は組織(会社)毎に作らねばならない.
  - ある要素がどのレベルかは人間が主観で判断しなければいけない.
- つけた点数の合計(Unadjusted FP: UFP)が基本的に仕様に基づくシステムの規模となる.
- UFPにシステム全体の複雑さ等を考慮した係数をかける場合が普通.

例

とあるシステムの特性

- 入力 4個 容易
- データ 1個 容易
- 出力 2個 普通

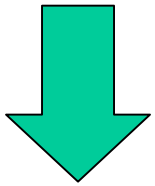
	容易	普通	複雑
入力	3	4	6
出力	4	5	7
データ	7	10	15
インタフェース	5	7	10
DB照会	3	4	6

とある組織の点数表

- このシステムの規模=
  4*3 + 1*7 + 2*5 = 29

FPの問題点

- 昨今のFPは構成要素が凄く多い.
- 組織毎に妥当な点数表を準備するのが困難.
- 点数表があっても，ある要素がどのレベルか判断するのが困難.



- 要はめんどくさいのであまり使われていない.

ユースケースポイント (UCP)

- Use Case Points
- FPの考え方をユースケース図，ユースケース記述に適用して，システムの規模を測定する方法.
- 基本的には，それぞれのアクター，ユースケースを単純，普通，複雑に分類し，点数付けして合計する.
  - 未調整UCP(UUCP)と呼ばれる.
- 技術や環境の要因を考慮して，UUCPに係数をかける.

よく使われるメトリクス

- 出荷後のバグ数
- 変更，または，変更要求の数
- 顧客満足度
- 開発中の摘出バグ数
- ドキュメントの完全性，正確性
- バグ特定/修正時間
- 種類別バグの分布
- 機能毎のバグ数
- テストにおける仕様カバレッジ
- コードのテストカバレッジ

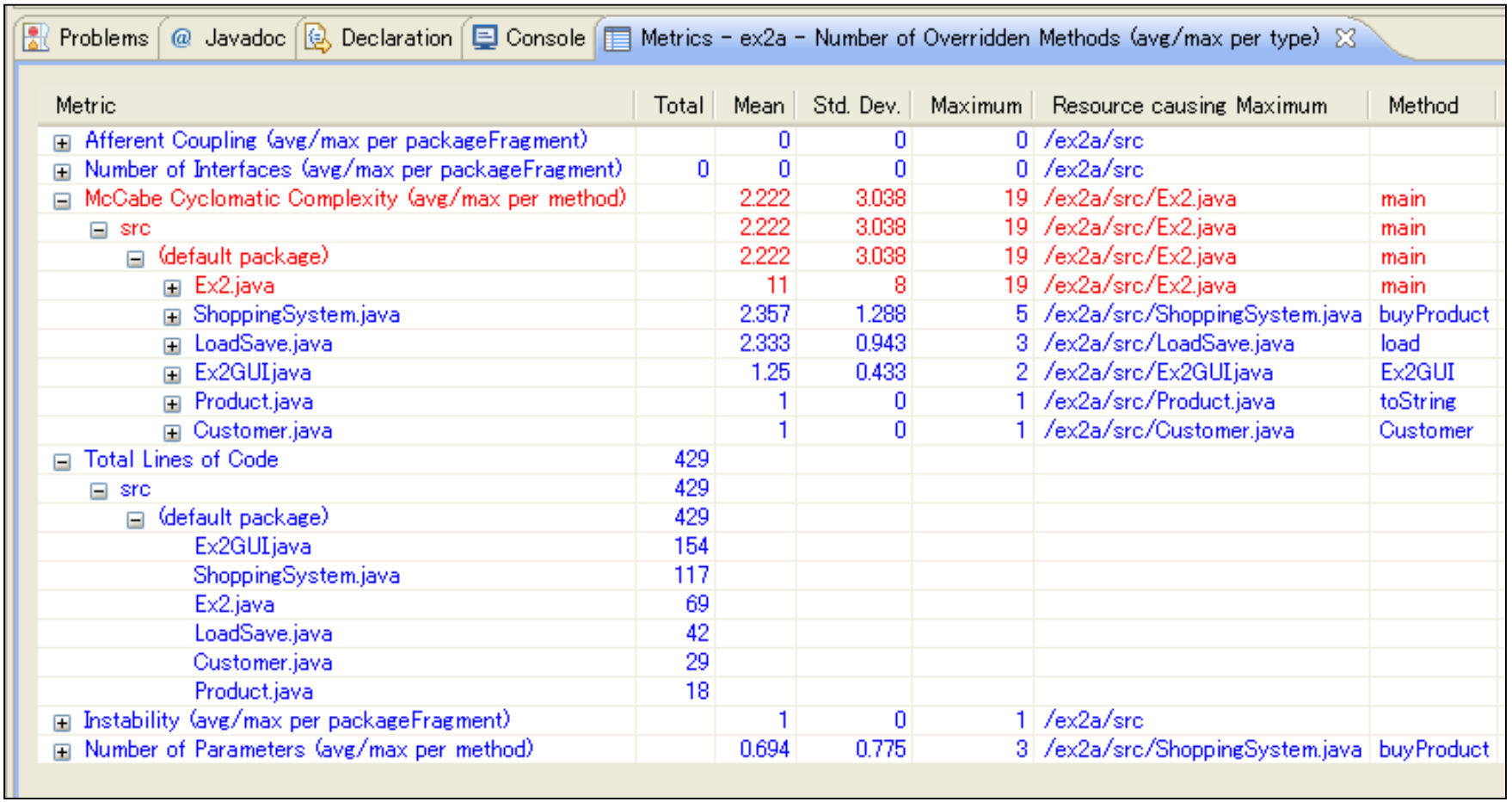
あまり使われないメトリクス

- モジュール複雑度
- 出荷したコード行数
- ドキュメントのサイズ・複雑性
- 再利用コード行数
- ファンクションポイント

ツールの紹介

- Eclipse Metrics Plugin (Frank Sauer)
- http://metrics.sourceforge.net/
- インストール
  - 基本上記ページ参照
  - Eclipseにおいて，help → Install new Software で以下のサイトを指定.
  - http://metrics.sourceforge.net/update
- 使い方
  - Window → Show View → other で，MetricsのMetrics View を開く.
  - メトリクスを見たいプロジェクトのpropertiesを開いて，Metrics のところで，enable metrics にチェックを入れる.

画面例



Problems | Javadoc | Declaration | Console | Metrics - ex2a - Number of Overridden Methods (avg/max per type)

Metric	Total	Mean	Std. Dev.	Maximum	Resource causing Maximum	Method
⊞ Afferent Coupling (avg/max per packageFragment)		0	0	0	/ex2a/src	
⊞ Number of Interfaces (avg/max per packageFragment)	0	0	0	0	/ex2a/src	
⊟ McCabe Cyclomatic Complexity (avg/max per method)		2.222	3.038	19	/ex2a/src/Ex2.java	main
⊟ src		2.222	3.038	19	/ex2a/src/Ex2.java	main
⊟ (default package)		2.222	3.038	19	/ex2a/src/Ex2.java	main
⊞ Ex2.java		11	8	19	/ex2a/src/Ex2.java	main
⊞ ShoppingSystem.java		2.357	1.288	5	/ex2a/src/ShoppingSystem.java	buyProduct
⊞ LoadSave.java		2.333	0.943	3	/ex2a/src/LoadSave.java	load
⊞ Ex2GUIjava		1.25	0.433	2	/ex2a/src/Ex2GUIjava	Ex2GUI
⊞ Product.java		1	0	1	/ex2a/src/Product.java	toString
⊞ Customer.java		1	0	1	/ex2a/src/Customer.java	Customer
⊟ Total Lines of Code	429					
⊟ src	429					
⊟ (default package)	429					
Ex2GUIjava	154					
ShoppingSystem.java	117					
Ex2.java	69					
LoadSave.java	42					
Customer.java	29					
Product.java	18					
⊞ Instability (avg/max per packageFragment)		1	0	1	/ex2a/src	
⊞ Number of Parameters (avg/max per method)		0.694	0.775	3	/ex2a/src/ShoppingSystem.java	buyProduct

文献

- 演習で学ぶソフトウエアメトリクスの基礎
  - リンダ・M・ライルド，キャロル・ブレナン (著), 野中 誠, 鷲崎 弘宜 (翻訳)
  - 2009年
  - 日経BP社