システム改善を目的とした クローン分析ツールの適用

株式会社CIJ

永井 昌子

目的

- 顧客の多様な要求に応えつづけているあるシステムは、長年の改修を重ね巨大化・複雑化している。そのため、機能を容易に追加することができず、開発コストが増大している。このシステムをリファクタリングで改善するか、新たに作り直すかを判断する材料の１つとして、現状のシステム品質を評価し、リファクタリングによる改善の可能性を検討する

ソースコード評価

- 不吉な匂い
  - 重複したコード
  - 長すぎるメソッド
  - 巨大なクラス
- 複雑さ
  - コードの構造
  - クラス
  - クラスの依存関係

ソースコードメトリクス

- CK尺度（ChidamberとKemererの提案）
  - クラスあたりの重み付けしたメソッド（WMC)
    「循環的複雑度」によって測った、メソッドの複雑度の合計値
  - クラスの応答数（RFC)
    当該クラスのオブジェクトが受け取るメッセージに応答して実行されるメソッドの数
  - オブジェクトクラス間の結合度（CBO）
    当該クラスに結合されるクラスの数

ソースコードメトリクス

- Lorenzの提案するOO尺度
  - メソッド中のソースコードの行数（LOC)
- McCabeの循環的複雑度(VG)
- コードクローン分析
  - コードクローン含有率(CVR)
    当該ファイルのテキストが、何らかのコードクローンによって占められている割合

ソースコード品質測定

- 不吉な匂い
  - 重複したコード
    ⇒ CVR(コードクローン含有率)
  - 長すぎるメソッド
    ⇒ LOC（メソッド中のソースコードの行数）
  - 巨大なクラス
    ⇒ WMC（クラスあたりの重み付けしたメソッド)

ソースコード品質測定

- 複雑さ
  - コードの構造
    ⇒VG(McCabeの循環的複雑度)
  - クラス
    ⇒WMC(クラスあたりの重み付けしたメソッド)
    RFC(クラスの応答数)
  - クラスの依存関係
    ⇒CBO(オブジェクトクラス間の結合度)

品質測定ツール

- 測定対象言語
  - C++
- コードクローン含有率
  - CCFinderX
- コードクローン以外のメトリクス

ツール	LOC	WMC	VG	RFC	CBO
Krakatau metrics Professional	○	○	○	×	×
Resource Standard Metrics	○	○	○	×	×
Understand for C++	○	○	○	○	○

指標値

- 「ソフトウェア品質工学の尺度とモデル」
  Stephen H. Kan著
  - Lorenzの提案するOO尺度と経験則

尺度	経験則およびコメント
平均メソッド数(LOC)	C++で24LOC未満

  - クラスの応答数(RFC)＞100
  - オブジェクトクラス間の結合度(CBO)＞5
  - クラスあたりの重み付けしたメソッド数(WMC)＞100

指標値

- 「ソフトウェア開発の定量化手法 第2版」
  Capers Jones著
  - McCabeの循環的複雑度(VG)
    5以下のプログラムは単純であり、理解しやすいものとされる。
    10以下でも、それほど理解困難とは思われない。20以上のとき、
    複雑さは高いとみなすことができる。50を超えた場合、ソフト
    ウェアはテスト不可能になり、実用に併せなくなる。

指標値

- 「EASE PROJECT News Letter vol.3」
  ソフトウェアプロダクト評価とコードクローン評価
  門田暁人著
  - コードクローン含有率 (CVR)
    125個のオープンソースソフトウェアのクローンを計測した結果、50トークン（約21SLOC）の含有率の中央値は8.7%。

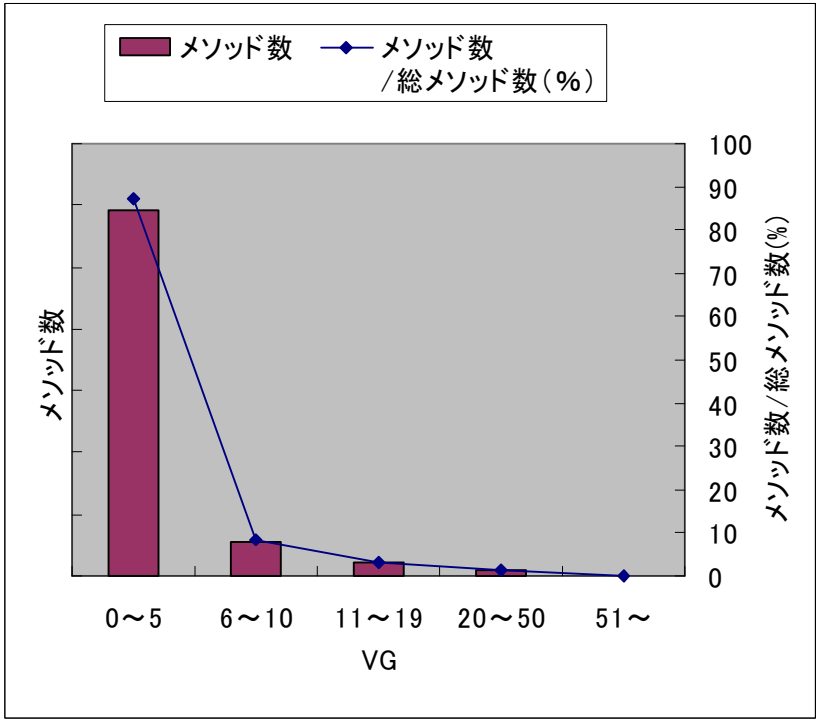
指標値の検証

- サンプリングシステムの測定結果

指標＼測定値	最大値	最小値	平均値	中央値	指標値を上回る割合(%)
CVR	99	0	0.30	0	32.88
LOC	161	1	10.90	4	10.75
VG	41	1	2.49	1	9.95
WMC	587	1	46.91	24	8.51
RFC	457	0	41.53	21	8.57
CBO	64	1	9.82	7	61.70

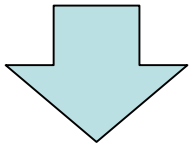
測定結果

- 局所的に複雑度の高い部分を検出
- クローン含有率が高い



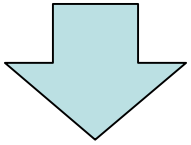
評価

- 局所的に複雑度の高い部分を検出



リファクタリングで改善する可能性がある

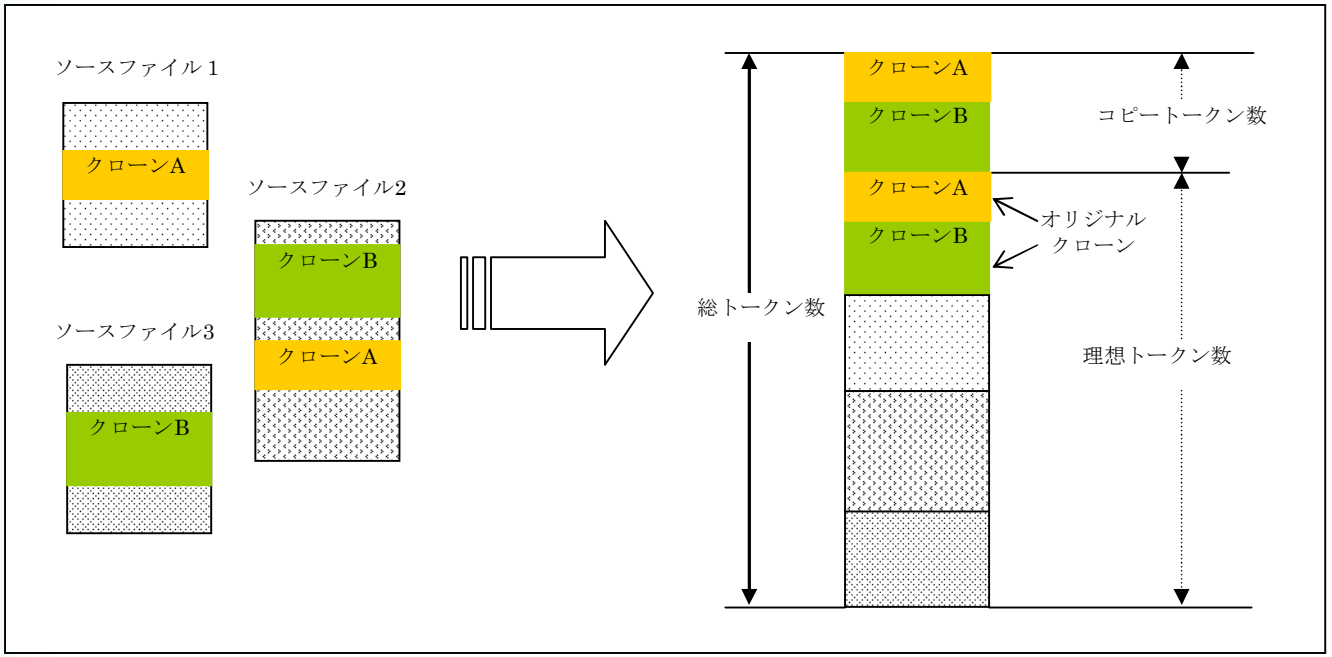
- クローン含有率が高い



ソースコードをどの程度圧縮できるか？

コードクローン測定結果分析

- ソースコードをどの程度圧縮できるか？
  - コピーしたコードの量を求める



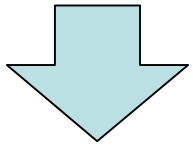
コードクローン測定結果分析

- コピーしたコードの量を求める
  コピーしたコードの正しい量は求められないが、クローンセットの片方をオリジナル、片方をコピーと仮定し、重複を排除しておよその量を求める。

  – GemXを使用してコードクローンデータを取得
  – クローン片の位置情報からオリジナルとコピーを特定し、コピーのコード量を求める

コードクローン分析結果

- コードコピー率
  コピーしたコード量の全体に対する割合をコードコピー率とする
- コードコピー率が高い
  - １つのコード片から多くのコードが派生している



ソースコードを大きく圧縮できる可能性がある

まとめ

- リファクタリングによる改善が可能
  複雑度が高い部分が局所的であるため、リファクタリングは困難ではない
- リファクタリングの効果は高い
  派生したコード量が多く、リファクタリングにより、全体のコード量を大きく圧縮できる
- 改善策の判断支援（今後の課題）
  リファクタリング前後のシステムの開発工数の経年変化をシミュレートすることにより、リファクタリングの費用対効果を予測し、改善策の判断材料とする

参考

- 「リファクタリング」マーチン・ファウラー著
- 「ソフトウェア開発の定量化手法第2版」
  Capers Jones著
- 「ソフトウェア品質工学の尺度とモデル」
  Stephen H.Kan著