責務の割り当てに着目した設計クラス図の機能性・保守性評価

Evaluation of Maintainability and Functionality of Design Class Diagram Focused on Allocation of Responsibilities

津田 直彦 * 鷲崎 弘宜 † 深澤 良彰 ‡

あらまし 定性評価は主にクラス図単位で与えられる。そのため定性評価と照らし合わせたクラス図同士の比較では、クラス図単位で集約されたメトリック測定値を用いる。しかし一般的な集約手法でクラスの関連数を集約すると、個々のクラスの関連数が多いか少ないかに関する情報は集約時に切り捨てられてしまう。これは他のクラスメトリックについても同様である。

責務の大きさを表すメトリックの測定値がある値 (閾値) より大きければ、そのクラスは責務が大きいと考えられる。そこで我々は責務が大きいクラスを数えることで、従来は見落とされやすかったクラス図内の責務割り当ての特徴を表せると考えた。

我々は評価実験により提案手法と従来のメトリック集約手法と比較し、提案手法の有用性を確認した。そして提案手法を用いた分析により、責務が大きいクラスが多く存在する設計クラス図では保守性・機能性に関する定性評価が高いことがわかった。ここでいう保守性・機能性とは、システムのわかりやすさや機能実現の際の責務割り当ての適切さのことである。これらから、クラスに割り当てる責務が十分に検討されているクラス図は、システムの中心となる責務が大きいクラスとそれに関わる責務の小さいクラスを含むものになると考えられる。

1 はじめに

熟練者による設計モデルのレビューによりソフトウェアの保守性などを早期に評価できる。しかし、設計モデルを読んで問題箇所を指摘するまでに時間がかかるため人手レビューのコストは高い。また熟練者であっても評価基準はレビュアーごとに異なるので、人手レビューによる評価結果は属人性が高い。

設計モデルとしては UML クラス図が利用されることが多い [1]。クラス図からはクラスが持つ関連数や属性数などを定量的な特徴として測定することができる。これらの測定値とクラス図に与えられた定性評価の関係を分析することが、将来的なレビューの自動化に繋がると考えられる。

定性評価は主にクラス図単位で与えられる。そのため定性評価と照らし合わせたクラス図同士の比較では、クラス図単位で集約されたメトリック測定値を用いる。しかし従来の一般的な集約手法では、関連数が多いクラスはいくつある方が良いかといった観点の考察ができない。個々のクラスの関連数が多いか少ないかに関する情報が集約時に削がれているからである。そのため従来のメトリック集約手法を利用した分析では、クラス図内の責務の割り当てに関する知見を見落としやすいと考えられる。

熟練者がクラス図の評価をする際、GRASP パターン [2] などの設計原則に基づいてクラス図内での責務の割り当てに着目する。ソースコードの評価では過度に大きな責務が割り当てられたクラスが存在した場合に、それを不適切と判断する [3]。そこで我々は、責務が大きいクラスを数えることが、クラス図内での責務の割り当て

*Naohiko Tsuda, 早稲田大学

†Hironori Washizaki, 早稲田大学

‡Yoshiaki Fukazawa, 早稲田大学

の特徴を表すメトリック集約手法となると考えた。

責務の大きさを表すメトリックの測定値がある値 (閾値) より大きければ、そのクラスは責務が大きいと考えられる。メトリック測定値の閾値を導出する方法は二つ考えられる。

- I. 同一ドメインの複数のモデルを用意して、統計的に閾値を導出する。
- II. 単一モデル内でメトリック測定値を相対比較して閾値を導出する。

我々は大きすぎる値を外れ値として、「外れ値を持つクラスの数」と「クラス図内で外れ値を持つクラスの割合」を新しいメトリック集約手法として提案する。I. の方法による NOD(The Number of Outliers in Domain) と ROD(The Ratio of Outliers in Domain)、II. の方法による NOM(The Number of Outliers in Model) と ROM(The Ratio of Outliers in Model) である。我々の知る限り、クラス図内の責務割り当てに着目したメトリック集約手法を提案し、それを熟練者による定性評価と照らし合わせて評価した研究は無い。

また、我々は設計クラス図の責務割り当てに関して「小さな責務のクラスしかない設計クラス図を書くべきでない」という仮説を立てた。実際に稼動するシステムにおいて小さな責務のクラスしか存在しない場合、そのシステムは責務割り当てが適切であると考えられる。一方、設計クラス図において小さな責務のクラスしか存在しない場合、そのクラス図は責務割り当てが十分検討されずに書かれたと考えられるからだ。

実際、設計クラス図は要素を省略して書かれる場合が多く [1]、要素の省略が多いクラス図に基づいた開発では実装時に責務割り当てをしなければならなくなる。このような設計と実装の間のギャップは設計モデルの利便性の低下に繋がるとされる [4]。

本研究で我々は 65 個の組み込みシステムの設計クラス図を対象に責務の大きさを表す 4 種の基本的なクラスメトリクス (属性数、メソッド数、関連数、サブクラス数) を測定し、提案する集約手法をメトリックごとに適用した。そして、設計クラス図に対してソフトウェア開発の専門家が与えた保守性・機能性に関する定性評価との関係を分析した。ここでいう保守性・機能性は、システムのわかりやすさや機能実現の際の責務割り当ての適切さを表す。

分析の際、我々は提案する 4 つの集約手法の有用性を確認することを目的とし、Research Question として RQ1 と RQ2 を設定した。また設計クラス図の責務割り当てに関する仮説を検証するために RQ3 を設定した。

- RQ1: 提案する 4 つの手法による集約結果は、設計クラス図の定性評価と関係があるか。
- RQ2: NOD、ROD、NOM、ROM、従来の集約手法とで、得られる知見の性質に違いはあるか。
- RQ3: 小さな責務のクラスしかない設計クラス図に対する定性評価は低いか

RQ1 では、NOD と NOM は定性評価との相関が認められやすく、ROD と ROM は認められにくいことがわかった。RQ2 では、それぞれの集約手法から得られる知見の性質に違いがあることがわかった。RQ3 では、設計クラス図では責務が大きいクラスが多く存在する方が保守性・機能性に関する定性評価が高いことがわかった。このことから、クラスに割り当てる責務が十分に検討されているクラス図は、システムの中心となる責務が大きいクラスとそれに関わる責務の小さいクラスを含むものになると考えられる。

2 背景

この章では、まず従来のメトリック集約手法で削がれる情報を説明する。また、

我々が提案する手法の要素技術として Robust-Z スコアを説明する。

2.1 従来のメトリック集約手法で削がれる情報

例として二つの UML 設計クラス図である図 1 と図 2 を見る。これらの図は ET ロボコン 2010 という大会に提出された図で、組み込みシステムをドメインとする。

総関連数は図 1 では 46、図 2 では 42 である。総関連数で見た場合、二つの図はほぼ同一視される。しかし二つの図を見比べて見ると、両者は似た構造をしていない。そして、ソフトウェア開発の専門家が与えたシステム構造の定性評価では図 1 は評価が高く、図 2 は評価が低い。

熟練者によるレビューでは、GRASP パターン [2] などのソフトウェア設計原則に従って設計不良を特定している。クラスに対する過度な責務割り当ての見極めなどがそれにあたる。図 2 の関連のつけ方を見ると、一部のクラスにのみ関連が集中していることがわかる。このことが不適切な責務割り当てと捉えられたために、図 2 の構造が低く評価されたと考えられる。

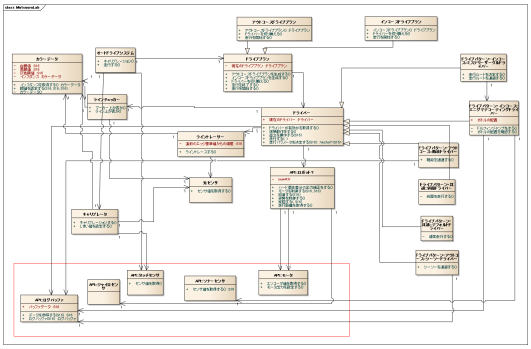

図 1 ET ロボコンの設計クラス図 1 総関連数=46, 構造の評価=A

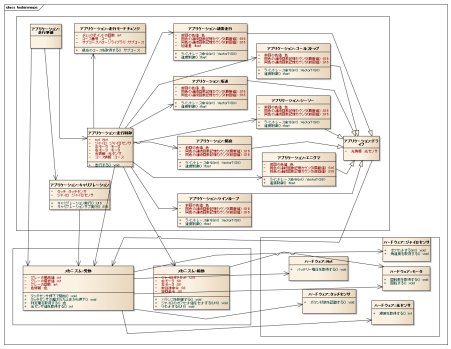

図 2 ET ロボコンの設計クラス図 2 総関連数=42, 構造の評価=C

定性評価とクラスメトリック測定値の関係を分析すれば定量的な知見を得られる。分析の際、クラス図単位で与えられた定性評価に合わせてメトリック測定値はクラス図単位に集約される。上記の例の「関連が集中しているクラス」を表す約手法としては最大値をとることが考えられるが、最大値では関連が集中しているクラスの数がわからない。総計・平均値・最大値といった従来の一般的な集約手法では、個々のクラスの関連数がいくつであったかという情報が集約時に削がれてしまう。そのため従来の集約結果と定性評価との分析では、関連数が多いクラスの量やその割合はどの程度が適切かという知見を見落としやすいと考えられる。

2.2 クラスの責務の大きさとメトリック測定値

設計クラス図から測定できるメトリクスである [サブクラス数・属性数・メソッド数・関連数] は、クラスに割り当てられた責務の大きさを表す。これらの測定値がある値 (閾値) より大きいクラスは責務が大きいと考えられる。メトリック測定値の閾値を導出する方法は二つ考えられる。

- I. 同一ドメインの複数のモデルを用意して、統計的に閾値を導出する。
- II. 単一モデル内でメトリック測定値を相対比較して閾値を導出する。

前者はベンチマーク [5] を、後者は RobustZ-スコア [6] を利用して定量的に導出することができる。

従来の集約手法では、クラス図内に含まれる責務が大きいクラスの量に関する情報は切り捨てていた。そこで責務の大きさで条件を設けてクラスを数えれば、責務

割り当ての特徴を表せると我々は考えた。

2.3 Robust Z スコア

平均値 μ と標準偏差 σ を利用して外れ値を検出する手法として Z スコアがある。実数値集合 X 内の要素 x_i を相対的に比較して、 μ より $+\sigma$ 大きければやや大きな値、$+2\ \sigma$ 大きければ大きな値と判断できる。

$$X = \{x_1, ..., x_n\} \tag{1}$$

$$Z(x_i, X) = \frac{xi - \mu(X)}{\sigma(X)} \tag{2}$$

この判断方法では、偏差が $+\sigma$ や $+2\ \sigma$ より大きいかどうかを見ている。上記の例では+1、+2 といった値が Z スコアとなる。

しかし要素数が少ないと平均値が外れ値 (極端に大きい値など) に近づいてしまうため、Z スコアでは外れ値でないものも外れ値として検出することがある。このような場合、四分位を利用した Robust-Z スコアの方が相応しい [6] 。Robust-Z スコアでは平均値 μ の代わりに中央値 Median を、標準偏差 σ の代わりに標準化四分位範囲 NIQR を用いる。Z スコアと同様に Robust-Z スコアが+1、+2 より大きいかどうかを見て外れ値検出ができる。

$$NIQR(X) = IQR(X) * 0.7143 \tag{3}$$

$$RobustZ(x_i, X) = \frac{xi - Median(X)}{NIQR(X)} \tag{4}$$

3 メトリック集約手法の提案

3.1 ドメインで見た場合の外れ値

複数のクラス図があれば、Alves らが提案する手法 [5] でメトリックの閾値を導出できる。例として図 1 と図 2 から関連数の閾値を導出する。図 1 の各クラスの関連数を集合 A_1 とする。図 2 でも同様に A_2 を求める。次に、A_1 の分位数 Quantile を実数 $q(0 \leq q \leq 1)$ に対して求めて、これを t_1 とする。同様に A_2 からも t_2 を求める。q=0.7 とすると、t_1 は 3.0、t_2 は 1.6 となった。最後に t_1 と t_2 の中央値 2.3 を閾値として得る。

$$A_i = \{a_{i,1}, ..., a_{i,ni}\} \tag{5}$$

$$T(q) = \{t_1, ..., t_k | t_i = Quantile(A_i, q)\} \tag{6}$$

$$th(q) = Median(T(q)) \tag{7}$$

Alves の方法では、$q = 0.70$ での閾値を $th70$ として、$th70$ 以内を普通の値、$th70$ 以降 $th80$ 以内をやや大きい値、$th80$ 以降 $th90$ 以内を大きい値、$th90$ 以降をとても大きい値としている。Alves の方法で閾値を求めれば、関連数の外れ値 (閾値より大きい) を検出できる。我々はあらかじめ特定のドメインについて閾値を求めて、メトリック測定値が外れ値となるクラスを数える手法を NOD(The Number of Outliers in Domain) として提案する。そして、クラス図内での外れ値を持つクラスの割合を ROD(The Ratio of Outliers in Domain) として提案する。また本研究では、$q = 0.70$ で求めた NOD を便宜的に $NOD70$ と定義する。$ROD70$ も同様である。

図 1・図 2 で関連数の $NOD70$ と $ROD70$ を求めて比較してみる。まず、図 1・図 2 と同じドメインの設計クラス図 65 個から関連数の $NOD70$ 用の閾値として 2.0 を得る。次に、図 1 の各クラスの関連数を集合 X として、X の中で 2 より大きい値を数える。図 1 ではクラスは 22 個、関連数が 2 より大きいクラスは 10 個であった。そのため $NOD70$ は 10、$ROD70$ は 10/22(=0.45) となった。また図 2 では $NOD70$ は 3、$ROD70$ は 3/19(=0.16) となった。

$NOD70$ と $ROD70$ が小さいことから、図 2 では一部のクラスに関連が集中していることがわかる。このように NOD と ROD はクラス図が含む責務が大きいクラスの量を表す。統計的に求めた閾値を利用して求められる NOD・ROD とクラス

図の定性評価との関係を分析すれば、責務の割り当てに関する知見が得られると考えられる。

$$X = \{x_1, ..., x_n\} \tag{8}$$
$$NOD(X, th) = |\{x \in X | x > th\}| \tag{9}$$
$$ROD(X, th) = \frac{NOD(X, th)}{|X|} \tag{10}$$

3.2 単一モデル内で見た場合の外れ値

NOD と ROD はあらかじめ用意した閾値を利用して算出する。しかし、設計クラス図は利用目的が異なると属性数などのメトリックがとる値の傾向が異なってくる[7]。そこで、我々はあらかじめ閾値を用意しない手法として NOM(The Number of Outliers in Model) と ROM(The Ratio of Outliers in Model) を提案する。NOM は外れ値を持つクラスの数、ROM はクラス図内での外れ値を持つクラスの割合である。

単一モデル内でのメトリック測定値のみでも Robust − Z スコアを用いて外れ値検出ができる。Robust-Z スコアが+1 以降ならやや大きく、+2 以降なら大きく、+3 以降ならとても大きいと判断できる。この方法では Robust-Z スコアが+k 以降であるかを見ている。本研究では、$k = 1$ として求めた NOM を便宜的に $NOM1$ と定義する。$ROM1$ も同様である。

図 1・図 2 で関連数の $NOM1$ と $ROM1$ を求めて比較してみる。図 1 の各クラスの関連数を集合 X とする。このとき X=[0, 0, 3, 3, 4, 5, 2, 3, 3, 3, 1, 1, 2, 2, 4, 4, 3, 1, 1, 1, 0, 0]、X の中央値は 2.0、X の NIQR は 1.43 となる。X の中で Robust-Z スコアが+1 より大きくなるのは、3.43(中央値+1*NIQR) より大きい [4, 5, 4, 4] の 4 個であり、それぞれ Robust-Z スコアは [1.4, 2.1, 1.4, 1.4] となる。結局、図 1 では $NOM1$ は 4、$ROM1$ は 4/22(=0.18) となった。また図 2 では 1.71 より大きい関連数が外れ値となり、$NOM1$ は 6、$ROM1$ は 6/19(=0.32) となった。

図 1 よりも図 2 の方が $NOM1$ と $ROM1$ が大きい。図 2 では関連数が 2 と 3 でも外れ値とされているのに対し、図 1 では関連数が 4 以上のものが外れ値とされている。単一クラス図内で見て周りより関連数が大きいクラスを数えているために、上記の差が生じる。

均一な関連数のクラスが多いと NIQR が小さくなり、関連数が中央値より少し大きいだけで外れ値扱いされる。そのため、関連数が大きいクラスと小さいクラスの二極端なものしかないクラス図では関連数の NOM と ROM が大きくなると考えられる。このことから NOM と ROM が表す責務割り当ての特徴は、NOD と ROD が表すものと異なると考えられる。

$$NOM(X, k) = |\{x \in X | RobustZ(x, X) > k\}| \tag{11}$$
$$ROM(X, k) = \frac{NOM(X, k)}{|X|} \tag{12}$$

4 評価実験

4.1 実験について

表 1 に示す 4 つの基本的なクラスメトリクスはクラスが担う責務の大きさを表す。我々は 65 個の設計クラス図それぞれで、この 4 つのクラスメトリクスを測定した。そして、3 章で提案したメトリック集約手法を適用した。

その後、各集約結果とクラス図に与えられた定性評価との間のスピアマンの順位相関の大きさを求めた。用いた集約手法は、総計 Total、平均値 Mean、中央値 Median、最大値 Max、最小値 Min、$NOD70$、$NOD80$、$NOD90$、$ROD70$、$ROD80$、

表 1　責務の大きさを表す基本的なクラスメトリクス

クラスメトリック	略称	説明
サブクラス数	NChld	直下のサブクラスの数。
属性数	NAttr	継承によるものを含まない属性の数。
メソッド数	NOp	継承によるものを含まないメソッドの数。
関連数	NAssoc	継承によるものを含まない関連の数。

$ROD90$、$NOM1$、$NOM2$、$NOM3$、$ROM1$、$ROM2$、$ROM3$ である。

$NOD70$ と $NOM1$ はやや大きめ以上の値を、$NOD80$ と $NOM2$ は大きめ以上の値を、$NOD90$ と $NOM3$ はとても大きい値を外れ値とする。ROD と ROM についても同様である。

4.2　実験対象

今回の実験に用いたデータは、組み込みシステムの技術教育を目的とした ET ロボコン 2010 [8] という大会に提出された設計クラス図 65 個である。これらは自律走行するロボットの設計クラス図であり、ソフトウェア開発の経験を持つ学生や社会人によって描かれている。

この大会ではソフトウェア開発の専門家がチェックリストに基づいたレビューによって、設計モデルを A〜D の順位尺度で定性評価している。この定性評価は複数人の合議によって決められており、一定の妥当性があると考えられる。

定性評価には複数の評価項目があり、クラス図やシーケンス図などの設計モデルを評価している。そこで、我々は設計クラス図と強く関連する 5 つの評価項目を実験に利用した。5 つの評価項目とその評価基準を表 2 に記した。また表 2 の「観点」に、評価項目が保守性・機能性のどちら (または両方) を表すかを記した

これらの評価項目ではシステムのわかりやすさや機能の妥当性を基準として、ソフトウェアの保守性・機能性を評価している。「見やすさ」はクラス図を見た際のわかりやすさを表す。「補足」は説明が審査員にうまく伝わるかどうかを評価するので、間接的にシステムのわかりやすさを表す。「構造」はシステム構造の明快さと機能実現する際の責務割り当ての適切さを表す。「トレーサビリティ」はクラス図とそれ以外の図の間の整合性を評価するので、システムの保守性と関係する。「妥当性」はシステムが実現する機能の合目的性と関係する

表 2　ET ロボコンの審査員評価項目のうち設計クラス図に関連するもの

観点	評価項目	説明
保	見やすさ	図のレイアウトや要素数が適切か。
保	補足	モデルの内容を十分に文章などで説明しているか。
保, 機	構造	責務の割り振りや、階層化、要素間の関係が妥当か。
保	トレーサビリティ	モデル図同士での整合性があるか。
機	妥当性	工学的に見て、汎用性と妥当性か。

4.3　実験結果

実験対象について 4 種のメトリクスの各種集約結果を求め、定性評価項目とのスピアマンの順位相関の大きさを表 3-6 にまとめた。”p”は p 値が 0.05 より大きかったために順位相関が有意でなかったことを表し、NA は順位相関が求められなかったことを表す。「見やすさ」の評価と集約結果の間には相関が認められなかった。

表 3　「補足」と各集約結果の順位相関

	NChld	NAttr	NOp	NAssoc
Total	0.33	0.31	0.46	0.26
Mean	p	p	p	p
Median	NA	p	p	p
Min	NA	-0.28	p	-0.25
Max	0.27	p	0.35	p
NOD70	0.37	0.30	0.31	p
NOD80	0.37	p	0.31	p
NOD90	0.37	p	0.31	p
ROD70	p	p	p	p
ROD80	p	p	p	p
ROD90	p	p	p	p
NOM1	0.37	0.31	0.29	p
NOM2	0.38	p	p	p
NOM3	0.38	p	p	p
ROM1	p	p	p	p
ROM2	p	p	p	p
ROM3	p	p	p	p

表 4　「構造」と各集約結果の順位相関

	NChld	NAttr	NOp	NAssoc
Total	p	0.33	0.31	0.41
Mean	p	p	p	p
Median	NA	p	p	p
Min	NA	p	p	p
Max	p	p	0.30	p
NOD70	p	0.33	p	0.53
NOD80	p	p	p	0.31
NOD90	p	p	p	0.31
ROD70	p	p	p	0.42
ROD80	p	p	p	p
ROD90	p	p	p	p
NOM1	p	0.38	0.28	p
NOM2	p	0.29	0.27	p
NOM3	p	p	p	p
ROM1	p	p	p	p
ROM2	p	p	p	p
ROM3	p	p	p	p

表 5　「トレーサビリティ」と各集約結果の順位相関

	NChld	NAttr	NOp	NAssoc
Total	0.27	p	0.38	0.45
Mean	p	p	p	p
Median	NA	p	p	p
Min	NA	p	-0.29	p
Max	p	p	p	p
NOD70	0.26	p	p	0.45
NOD80	0.26	p	p	0.29
NOD90	0.26	p	p	0.29
ROD70	p	p	p	p
ROD80	p	p	p	p
ROD90	p	p	p	p
NOM1	0.26	p	0.34	p
NOM2	0.26	p	0.32	p
NOM3	0.26	p	p	p
ROM1	p	p	p	p
ROM2	p	p	p	-0.26
ROM3	p	p	p	p

表 6　「妥当性」と各集約結果の順位相関

	NChld	NAttr	NOp	NAssoc
Total	0.37	p	p	0.35
Mean	p	p	p	p
Median	NA	p	-0.26	p
Min	NA	p	-0.45	-0.28
Max	0.31	p	p	p
NOD70	0.42	p	p	0.26
NOD80	0.42	p	p	p
NOD90	0.42	p	p	p
ROD70	0.28	p	p	p
ROD80	0.28	p	p	p
ROD90	0.28	p	p	p
NOM1	0.42	p	0.26	0.39
NOM2	0.44	p	p	p
NOM3	0.44	p	p	p
ROM1	0.28	p	p	p
ROM2	0.28	p	p	p
ROM3	0.28	p	p	p

5　考察

5.1　RQ1: 提案する 4 つの手法による集約結果は、設計クラス図の定性評価と関係があるか

表 3-6 から、*NOD* と *NOM* は設計クラス図の定性評価と相関が認められやすいことがわかった。また、*ROD* や *ROM* では相関が認められにくいことがわかった。

Total では相関が認められやすく、Mean では相関が認められにくかったことも合わせると、責務の大きさを表すクラスメトリックはその規模 (総クラス数) で正規化すると、定性評価との相関が認められにくくなるという結果となった。このことから、ET ロボコン 2010 大会の審査員によるレビューではクラスメトリックの総計や大きな値の有無を重視し、クラス図全体で見た平均値は重視していなかったと考えられる。公開されている審査基準 [8] では、「構造」の評価では要素の凝集度に着目していたと述べられている。

この大会ではソフトウェア開発の専門家複数人がチェックリストと合議によって

定性評価を与えており、その評価には一定の妥当性があると考えられる。そのため、RQ1 に対する考察で得られた知見は一般化できると我々は考える。またこのことから、Mean・ROD・ROM のような規模で正規化した値は、定性評価ではなく実際の欠陥数などと照らし合わせた分析に利用した方が良い考えられる。

5.2 RQ2: NOD、ROD、NOM、ROM、従来の集約手法とで、得られる知見の性質に違いはあるか

表 3-6 から、Total による集約結果は設計クラス図の定性評価との相関が認められやすいことがわかった。そして、Mean と Median では相関が認められにくいことがわかった。

Total では総計される前の個々のメトリック測定値の区別がつかなくなるので、定性評価との相関から得られる知見はクラス図の特徴をおおまかに表すものとなる。

Min からは「メソッド数の最小値が大きいと妥当性の評価が低い」という知見が得られた。これは妥当性がないクラス図では、もっともメソッドが少ないクラスでさえそのメソッド数が多いということである。これは責務の分割ができていないクラス図は評価されにくいことと合致する。

提案手法からは以下の知見が得られた。

- $NOD$70: やや大きめ以上の関連数のクラスが多いほど構造の評価が高い
- $NOM$1: クラス図内で相対比較して、やや大きめ以上の属性数のクラスが多い方が構造の評価が高い
- $ROD$70: やや大きめ以上の関連数のクラスが占める割合が高いほど構造の評価が高い
- $ROM$1: クラス図内で相対比較して、やや大きめ以上の関連数のクラスが占める割合が低いほどトレーサビリティの評価が高い

上記の知見を見ると、それぞれの集約手法でクラス図の見方が違うことがわかる。Total ではクラス図全体での責務の量に関する知見を得られた。Min では責務分割に関する知見を得られた。また NOD と NOM では責務が大きいクラスの配置量に関する知見を得られた。ROD と ROM では責務が大きいクラスがクラス図全体で占める割合に関する知見を得られた。このことから、それぞれの集約手法から得られる知見の性質には違いがあると考えられる。

5.3 RQ3: 小さな責務のクラスしかない設計クラス図に対する定性評価は低いか

表 3-6 を見ると、NOD と NOM はいずれも正相関となっている。これは「責務が大きいクラスが多いクラス図の保守性・機能性に関する定性評価が高い」ことを意味する。この結果は設計クラス図が実際に稼動するシステムより単純な構造として省いて描かれることに起因すると考える。

設計段階でクラスに割り当てる責務が十分に検討されていれば、システムの中心となる責務が大きいクラスとそれに関わる責務の小さいクラスが混在する形でシステムが構築されると考えられる。そのため、設計の段階で責務が大きい箇所が少ないクラス図は責務分割が不適切であると考えられる。

また、責務が大きい箇所がいくつだと少ないといえるかは設計対象に依存すると考えられる。ET ロボコンの自律走行ロボットよりも大規模な設計を必要とするドメインのクラス図であれば、より多くの責務が大きいクラスが必要であると考えられる。

6 妥当性への脅威

6.1 定性評価への脅威

クラス図に与えられた審査員評価はソフトウェア開発の専門家複数人がチェックリストを用いながら合議を通して与えたものである。そのため本研究ではその評価

に一定の妥当性があると判断した。

評価項目の選定では、クラス図を全く扱わない評価項目は選ばないようにした。しかし、クラス図以外の影響が強い評価項目が選定されていた可能性は少なからずある。今後の研究では、設計クラス図のみを対象に与えられた定性評価を分析に利用したい。

本研究では設計クラス図の特徴と定性評価の関係を分析した。しかし、クラス図の定性評価はそのクラス図を元に実装したソフトウェア自体の良し悪しをそのまま表すとは限らない。今後の研究では、設計クラス図を元に実装されたソースコードの特徴 (バグ数など) とクラス図の定性評価との間に相関関係を分析したい。

6.2 一般化への脅威

今回の実験は組み込みシステムをドメインとする大規模ではないクラス図を対象に行った。そのため、本研究で得られた知見はそれと類似したドメインに適用可能であると考える。一方、異なるドメインに対して知見を一般化することには妥当性への脅威がある。

また、本研究では単一メトリックと定性評価の関係を単相関を分析した。しかし定性評価には複数の基準が関係しているため重相関分析をした場合には違った結果が得られる可能性がある。今後の研究ではレビュー過程や根拠が残っている定性評価を利用することで、各メトリックと定性評価の関係を明らかにしていきたい。

本研究では、$NOD$70 と $NOM$1 はやや大きめ以上の値を、$NOD$80 と $NOM$2 は大きめ以上の値を、$NOD$90 と $NOM$3 はとても大きい値を外れ値とするものと考えた。しかし、これらの判断は過去の研究や経験に基づくものの属人性が残る。NOD における q と NOM における k について、適切な値が何であるか、またそれがどのような場合に違ってくるかは今後の研究が必要である。

7 関連研究

Vasilescu らはクラスメトリクスなどのミクロな単位の測定値をシステム単位のマクロな形に集約することで、ソフトウェアの保守性分析の際にその進化の洞察を提供できると述べている [9]。我々が提案した手法では Vasilescu らが述べるマクロな集約が可能であり、クラスに対する責務割り当てが開発の進展につれてどのように変化していくかを洞察できる。

Serebrenik らはメトリックをシステム単位に集約する手法として、計量経済学における不平等尺度であるタイル尺度を挙げている [10]。タイル尺度の値を解釈には不平等尺度の知識が必要である。一方、我々が提案する NOD や NOM は閾値を超える測定値を持つクラスの数をそのまま表すので解釈が容易である。

我々が提案する NOD を算出する際の閾値導出には Alves が提案する手法を用いた [5]。これはベンチマークを利用して統計的にロバストな閾値を導出する方法である Alves は [0.7, 0.8, 0.9] で分位数を求めることで「やや危険・危険・とても危険」を区別するための閾値を求められると述べている。なお、この方法では大きすぎる値が望ましくないメトリックを想定している。

Larman はソフトウェアの設計原則として GRASP パターンを提案した [2]。この設計原則は、オブジェクト指向設計におけるクラスの責務の取り扱いの方針を提供している。Longman は、大きな責務を持つクラスは悪い匂いを持つという経験則を紹介している [3]。我々はこれらの定性的な観点を定量的に捉えるためのメトリック集約手法を提案した。

Camargo らは UML ダイアグラムとソースコードの双方で測定可能なメトリクスにおいて、その測定値の傾向が異なる場合があることを指摘した [11]。このことから、Alves の方法で用いるベンチマークを用意する際には設計モデルとソースコードを混在させるべきでないと考えられる。

Lange らは設計モデルは同じ種類の UML ダイアグラムで描かれたモデル同士であっても、モデルごとの利用目的が違うとメトリクス測定値の傾向が違ってくると述べた [7]。このことから、設計クラス図をベンチマークとしてあらかじめ閾値を用意しても、その閾値が不適切となる場合があると考えられる。そのため、NOD を求める際にはベンチマークとするモデルの利用目的や詳細度を揃えておくべきと考えられる。

8 まとめと将来の展望

定性評価は主にクラス図単位で与えられる。そのため定性評価と照らし合わせたクラス図同士の比較では、クラス図単位で集約されたメトリック測定値を用いる。我々はクラス図内の責務割り当てに着目したメトリック集約手法を提案し、その有用性を確認した。そして提案手法を用いた分析により、責務が大きいクラスが多く存在する設計クラス図では保守性・機能性に関する定性評価が高いという結果を得た。このことから、クラスに割り当てる責務が十分に検討されているクラス図は、システムの中心となる責務が大きいクラスとそれに関わる責務の小さいクラスを含むものになると考えられる。

今後の展望としては定性評価だけでなく、設計クラス図を元に実装されたソースコードの特徴 (バグ数など) との関係を分析する予定である。

また、本研究ではドメインで見た外れ値検出で Robust-Z スコアを用いなかった。NOD と NOM で外れ値検出の手法を一貫させるには、Alves の方法の代わりに Robust-Z スコアを用いて閾値を導出することの妥当性を検証する必要がある。

本研究では設計クラス図を対象としたが、この集約手法をソースコードにおいても適用することは有意義であると考える。クラス図やソースコードの特徴を比較・分析する研究において提案手法が利用されることを期待する。

参考文献

[1] Marian Petre. Uml in practice. In *Proceedings of the 2013 International Conference on Software Engineering*, ICSE '13, pp. 722–731, Piscataway, NJ, USA, 2013. IEEE Press.

[2] Craig Larman. *Applying UML and patterns: an introduction to object-oriented analysis and design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1998.

[3] *Refactoring: improving the design of existing code*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.

[4] Alexander Egyed. A scenario-driven approach to traceability. In *Proceedings of the 23rd International Conference on Software Engineering*, ICSE '01, pp. 123–132, Washington, DC, USA, 2001. IEEE Computer Society.

[5] Tiago L. Alves, Christiaan Ypma, and Joost Visser. Deriving metric thresholds from benchmark data. In *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ICSM '10, pp. 1–10, Washington, DC, USA, 2010. IEEE Computer Society.

[6] National Association of Testing Authorities. *GUIDE TO NATA PROFICIENCY TESTING*, 2004.

[7] Christian F. J. Lange and Michel R. V. Chaudron. Managing model quality in uml-based software development. In *Proceedings of the 13th IEEE International Workshop on Software Technology and Engineering Practice*, STEP '05, pp. 7–16, Washington, DC, USA, 2005. IEEE Computer Society.

[8] 一般社団法人 組込みシステム技術協会. ET ロボコン 2010 モデル審査基準, 2010.

[9] Bogdan Vasilescu, Alexander Serebrenik, and Mark van den Brand. You can't control the unfamiliar: A study on the relations between aggregation techniques for software metrics. In *Proceedings of the 2011 27th IEEE International Conference on Software Maintenance*, ICSM '11, pp. 313–322, Washington, DC, USA, 2011. IEEE Computer Society.

[10] Alexander Serebrenik and Mark van den Brand. Theil index for aggregation of software metrics values. In *Proceedings of the 2010 IEEE International Conference on Software Maintenance*, ICSM '10, pp. 1–9, Washington, DC, USA, 2010. IEEE Computer Society.

[11] Ana Erika Camargo Cruz. Exploratory study of a uml metric for fault prediction. In *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ICSE '10, pp. 361–364, New York, NY, USA, 2010. ACM.