情報処理学会論文誌 Vol. 49 No. 7 2319–2327 (July 2008)

UML 設計を対象とした品質評価モデルの検討

佐藤美穂[†1] 田村真吾[†1,*1] 上田賀一[†1]

ソフトウェア品質の低下を抑えるために，ソフトウェアを測定・分析・評価することは有効である．現在，それらを行う様々な方法が提案されているが，ソースコードを測定対象としているものが多く，検出された問題点が設計段階で作り込まれていた場合，多くの手戻り作業が発生することになる．そこで本研究では，複数のメトリクスを用いて UML で記述された設計図の品質特性を評価するモデルの検討を行う．UML 設計図を測定対象とすることで，早い段階での測定・修正が可能になり，全体の手戻り作業を削減することができる．本モデルの妥当性をみるため，複数の UML 設計図を測定・評価し，その結果を比較・分析した．また，本モデルで品質が悪いと評価された UML 設計図を測定・評価結果から目的に合わせて品質上の問題点を修正し，本ツールがソフトウェア品質の向上に役立つことを示した．

A Study of Quality Evaluation Model for UML Design

Miho Sato,[†1] Shingo Tamura[†1,*1]
and Yoshikazu Ueda[†1]

It is effective to measure, to analyze, and to evaluate the product for suppressing the decrease in the quality of software. Recently, various methods to perform them are proposed. However, most of these measure source codes. Therefore a lot of works is generated when a remediation of the design is needed. Defect should be detected and remediated early in a design phase. This study examines a method to evaluate some quality characteristics of a design diagram described in UML by using a number of metrics. Some suitable quality characteristic in the UML design is selected, and a method of evaluating the characteristic is systematized. By making the UML design diagram the measurement target, a measurement and a remediation is enabled at an early stage. To evaluate the validity of this evaluation model, a number of UML designs were measured and evaluated, and the result was compared and analyzed. Moreover, a UML design that is evaluated like as “the quality is bad” by this evaluation model is identified and remediated the part of the problem according to the purpose from the measurement result, and this evaluation method was useful for the improvement of the software quality.

1. はじめに

近年，ソフトウェアが大規模化・複雑化し，それに対応する技術者が不足している．そのため，教育が不十分で開発技術レベルの低い人材がソフトウェア開発に多く投入されている．これによってますますソフトウェア品質が低下し，変更・拡張しにくい成果物ができあがるため，機能の追加や保守をするたびに不特定多数のバグができてしまうなどの問題が引き起こされている．

ソフトウェア品質の低下を抑えるために，メトリクスを用いてソフトウェアを測定・分析・評価することは有効である．現在，様々なメトリクスが提案され，それらを用いたアプローチやツールが開発されている．これらの多くはソースコードを測定対象としている．しかしこれによって検出された問題点が設計で作り込まれたものであった場合，多くの手戻り作業が発生してしまう．そのため設計段階で作り込まれた問題点はその段階で検出し，修正することが望ましいとされる[2]．

本研究では，複数のメトリクスを用いて UML 図からオブジェクト指向設計の目的での品質特性モデルである QMOOD を基に品質特性を評価するモデルの検討を行う．抽象度の違う様々な品質特性を測定することで，品質の全体的なバランスをみることや，特定の品質特性を向上させることを目標に本評価モデルをさかのぼり品質上の問題点を特定することができる．品質は一次元的にとらえることが難しく多面的に評価した方がより妥当な評価ができるが，ソフトウェアの品質の分かりやすさや見やすさを重視するならば品質特性を一次元的にとらえることも重要である．

設計を測定対象とした場合，詳細な記述ルールを定義せずに測定することは困難であるが，これは測定対象を UML 設計図とすることで解決できる．UML を用いれば独自の記述ルールを覚える手間もなくなる．

提案した評価モデルの妥当性をみるため，調査結果・経験則から評価式を作成し，品質が悪いと思われる初心者が記述した設計図と，良い設計であるとされるデザインパターンを記述した設計図，その中間の品質であると思われる公開されている設計図の 3 グループを測

†1 茨城大学大学院理工学研究科
Graduate School of Science and Engineering, Ibaraki University
*1 現在，株式会社リコー
Presently with Ricoh Co., Ltd.

定し，その結果を比較・分析することでそれぞれの品質をとらえているかを調べる．また，本モデルで最も品質特性評価値が低かった設計図の品質上の問題点を修正し，その前後の評価値を比較する．

本論文では，以降で評価する品質特性・設計特性，測定するメトリクスを選択し，その後，選択した品質特性の評価モデルについて，評価モデルの妥当性と有効性について述べる．最後に本論文のまとめと今後の課題を述べる．

2. 各特性・メトリクスの選択

本評価モデルでは，品質特性を設計特性から，設計特性をメトリクス値からと，階層的に品質特性を求める．そのためには評価する品質特性とそれを求めるための設計特性・メトリクスを選択し，それらの関連付け（評価方法）を明確にしなければならない．ここでは，本評価モデルで選択した各品質特性・設計特性・メトリクスについて記述する．

本評価モデルは QMOOD[3] を参考に設計図のみで測定できる品質特性・設計特性を選択

表 1　品質特性
Table 1　Quality attributes.

品質特性	定義概要
再利用性	設計の一部が新しい設計に再適用しやすいか
拡張性	新しい機能（クラス）を追加しやすいか
有効性	OO 概念と技術を適切にとりいれているか
柔軟性	クラス内部を変更しやすいか
機能性	適切な責務が割当てられ，それを行っているか
理解性	簡単に構造を理解することができるか

表 2　設計特性
Table 2　Design properties.

設計特性	定義概要
サイズ	クラスの大きさが適度であるか
抽象性	適切に抽象化を用いているか
カプセル性	内部構成を外部から隠蔽しているか
結合性	クラス間の依存が疎であるか
凝集性	クラス内の要素の結び付きが強いか
コンポジション	コンポジションを適切に用いているか
継承性	適切に継承を用いているか
メッセージング	クラスの提供する機能が適度であるか
複雑性	分かりにくい設計になっていないか

表 3　メトリクス[5]
Table 3　Metrics.

メトリクス	定義概要	閾値		
		少し悪い	悪い	非常に悪い
NOA	クラスの属性の数	7〜9	10 以上	
NOM	クラスのメソッドの数	20〜30	31〜50	51 以上
NPM	クラスの private メソッドの数	14〜22	23〜34	35 以上
NMSM	メソッドのメッセージ送信数	10〜11	12〜14	15 以上
NOP	メソッドの引数の数	5〜6	7〜8	9 以上
NPubM	クラスの public メソッドの数	6〜8	9〜10	11 以上
NPV	クラスの public 属性の数		2〜3	4 以上
NAC	(抽象クラス数)/(クラス総数)	0.1〜0.12	0.08〜0.1	0.08 未満
NOS	スーパクラスが持つサブクラスの数		1	
NOAC	クラスの祖先の数	3〜4	5〜6	7 以上
NPAM	public アクセサメソッドの数		4〜6	7 以上
DAC	クラスの属性の参照型の数	3〜4	5〜6	7 以上
CBO	結合しているクラスの数	6〜7	8〜9	10 以上
NMR	他のクラスから受け取るメッセージの数		0	
MPC	クラスが送信するメッセージ数	36〜63	64〜96	97 以上
NIM	(継承したメソッド数)/(メソッド総数)	0.6〜0.8	0.4〜0.6	0.4 未満
NAM	(追加されたメソッド数)/(メソッド総数)		0.2〜0.4	0.4 以上
DIT	継承木の深さ		4〜6	7 以上
NID	継承した属性の数		0	
NOIDC	サブクラスのオブジェクトの数		0	
NACUM	public メソッドを利用する他のクラスの数		0	
NMMI	メソッドが呼び出される回数		0	
NMNP	(引数を持たないメソッド数)/(メソッド総数)	0.3〜0.5	0.5〜0.7	0.7〜1
NPVUA	他のクラスから利用される外部属性の数	2〜3	4〜5	6 以上
NTWD	異なるクラスのメソッド間の相互依存関係の数		1	2 以上
NMSSO	(同一メソッドから同一オブジェクトへの送信数)/(メソッド送信数)	0.6〜0.7	0.7〜0.8	0.8〜1
NSP	メソッドが同じ引数を送信する回数		1 以上	
NMHC	(高い複雑性を持つメソッド数)/(メソッド総数)	0.5〜0.6	0.6〜0.7	0.7〜1
NPABC	スーパクラスの public/protected 属性の数		1〜3	4 以上
NPMBC	スーパクラスの public/protected メソッドの数	2	1	0
NSAIS	同一祖先クラスを持つサブクラスすべてで定義された同一属性の数		1 以上	
NMIS	サブクラスで定義されたメソッド数		1	0

※空欄：閾値なし

した．QMOOD とは，階層品質モデル Dromey フレームワーク[4] を拡張した OO 設計品質特性を評価するアプローチであり，Bansiya らによって提案された．Bansiya らは ISO9126 をベースに設計書のみで測定が困難である「信頼性」や「使用性」をはずし，OO 設計の重要な特性だと思われる「再利用性」「柔軟性」を追加，資源を有効に使用しているかを示す「効率性」から OO 概念を有効に使用しているかを示す「有効性」に変更するなど，設計段階で測定できる品質特性を調査・定義している．このモデルの中で定義されている品質特性，設計特性を UML 設計を対象にした場合に適した定義に変更した．**表 1** と**表 2** に選択した品質特性・設計特性の概要をまとめる．

また，メトリクスは Choinzon らの設計欠陥検出アプローチ[5),6)] で定義されたメトリクスのうち，UML のクラス図とシーケンス図で測定可能なものを選出し，用いた．このアプローチでは，様々な文献 7)–10) からメトリクスとそれらの閾値を調査し，そこから得られた設計段階で測定できるメトリクス 22 個と，リファクタリングなどを基に新たに提案したメトリクス 18 個を用いて，設計の問題点を検出する．**表 3** に選択したメトリクスとその概要，調査された各閾値を示す．測定対象はクラスとメソッドとする．

3. 品質評価モデル

2 章でも述べたように，品質特性を階層的に求めるためには階層間の関連付け（評価方法）を明確にしなければならない．この節では，各メトリクスの正規化モデルと，メトリクスと設計特性，設計特性と品質特性の評価モデルを検討する．

3.1 メトリクス値の集計

クラス単位やメソッド単位などで測定されるメトリクスの値を集計する．品質を最良値・中間値・最悪値で評価した場合をそれぞれみるために，メトリクス値を最大値，中央値，最小値の 3 種類で集計し，今後の品質評価で用いる．以後，各集計値を基にそれぞれ品質の評価を行う．

3.2 メトリクス値の正規化

各種メトリクスに対して閾値を設定し，そこからメトリクスを 0〜100 の得点に正規化する得点式を導く．

本論文では，Choinzon らのアプローチで記述されている閾値を用いる（表 3）．そのアプローチでは 3 つの閾値を用いて各メトリクス値を標準，少し悪い，悪い，非常に悪いの 4 つの範囲に分けている．この範囲を用いて，標準が 100 点，少し悪いが 50 点〜100 点，悪いが 0 点〜50 点，非常に悪いが 0 点となるように評価式を作成する．例を**図 1** に示す．このようにして得られた得点式にメトリクス値を代入してメトリクスの得点を求める．

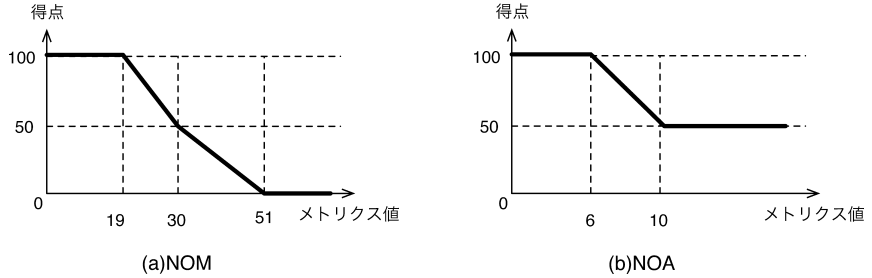

図 1　得点式の関数
Fig. 1　Score function.

3.3 設計特性評価モデル

各設計特性におけるメトリクスの比重を設定し，加重平均で各設計特性の得点を求める．設計特性得点はその観点からみて良い設計であれば得点が高い．たとえば複雑性ならば “複雑でない” 設計であれば得点が高くなる．同様に，結合性ならば “各クラス間の関係が疎” であれば得点が高くなる．以下に本研究で用いる設計特性の評価モデルを示す．

N 種のメトリクス値 $m_1, m_2, \cdots, m_N$ を用いて，設計特性の評価値 P を求める．ある設計特性の各メトリクスの比重を $a_1, a_2, \cdots, a_N$，係数の合計を $S_a = \sum_{i=1}^{N} a_i$ とするとき，設計特性の評価値は，

$$P = \frac{a_1}{S_a} \cdot m_1 + \frac{a_2}{S_a} \cdot m_2 + \cdots + \frac{a_N}{S_a} \cdot m_N \tag{1}$$

で求める．

3.4 品質特性評価モデル

設計特性の得点化と同様に，各品質特性における設計特性の比重を設定し，加重平均で各品質特性の得点を求める．設計特性得点はその観点からみて良い設計であれば得点が高いため，すべて品質特性得点と正の相関を持つ．以下に本研究で用いる品質特性の評価モデルを示す．

表 2 に示す 9 種の設計特性の評価値 $P_1, P_2, \cdots, P_9$ を用いて，品質特性の評価値 A を求める．ある品質特性の各設計特性の比重を $b_1, b_2, \cdots, b_9$，係数の合計を $S_b = \sum_{i=1}^{9} b_i$ とするとき，品質特性の評価値は，

$$A = \frac{b_1}{S_b} \cdot P_1 + \frac{b_2}{S_b} \cdot P_2 + \cdots + \frac{b_9}{S_b} \cdot P_9 \qquad (2)$$

で求める．

4. 評価実験

開発レベルが低いと思われる初心者が記述した演習程度の設計図 6 種と，良い設計であるとされる一般的なデザインパターンを記述した設計図 7 種[14]，公開されている教育用 UML などを対象とした設計図 6 種[15)–19)] の計 19 種のサンプル（**表 4**）を本モデルで評価し，各種サンプルや各品質特性の傾向をとらえているかをみることで，妥当性を調べる．初心者が記述した設計図はいずれも題材が異なり，物品管理システムや横スクロール型アクションゲーム，ロボット制御プログラミング環境など，多岐にわたる．

また，実際に初心者サンプル中に存在する品質上の問題点を修正し，その前後の品質特性評価値を比較することで，品質向上に本評価モデルが有効であるかを調査する．

UML 描画ツールである Rational XDE や Jude などを用いて作成されたクラス図・シーケンス図を XMI 形式で出力したファイルから，メトリクス値を収集するツールを開発した．このツールを用いて各サンプルのメトリクス値と設計特性・品質特性の評価を求める．

4.1 評価モデルの係数とメトリクスの選択

今回の実験で用いる評価モデルの係数とメトリクスの選択について説明する（**図 2**）．

QMOOD での品質特性と設計特性間の関係や経験則から導いた品質特性と設計特性の関係の強さを**表 5** に示す．今回は設計特性から品質特性を求める際の比重を 3 段階に分け，関係が強いと思われる設計特性の比重を “1”，関係が弱いかないと思われる設計特性の比重を “0”，関係があるが上記の “関係が強い” とされる設計特性ほど影響がないと思われる設計特性の比重を中央値の “0.5” とした．これらの比重を評価モデル (2) にあてはめ，品質特性の得点を求めた．

前述の文献 5) で記述されていた設計特性とメトリクスの関係を基に，設計特性を評価する際に用いるメトリクスを選択した．これを**表 6** に示す．今回の実験では設計特性の得点を用いられたメトリクスの得点の平均としたが，より妥当な設計特性の評価を得るためには各メトリクスの比重に対してさらなる調査・考察をし，適した比重を与えることが必要である．また選択されたメトリクスがサブクラスを測定対象とするものである場合，継承を用いていない設計図はサブクラス自体がないため測定できない．そのような場合，評価したい設計図に合わせてユーザがそのメトリクスを外すか，そのメトリクスの得点を 0 点などの

表 4　実験で用いたサンプル
Table 4　Samples used by experiment.

サンプル名	クラス数	継承	抽象
初心者サンプル 1	7	×	×
初心者サンプル 2	11	○	×
初心者サンプル 3	9	○	×
初心者サンプル 4	14	×	×
初心者サンプル 5	9	×	×
初心者サンプル 6	12	×	×
Builder	5	○	○
Command	4	×	○
State	4	×	○
Chain of Responsibility	7	○	○
Mediator	5	×	○
Memento	3	×	×
Visitor	7	○	○
公開サンプル 1	16	○	○
公開サンプル 2	19	×	×
公開サンプル 3	45	○	○
公開サンプル 4	12	○	×
公開サンプル 5	8	×	×
公開サンプル 6	6	×	×

○：有　×：無

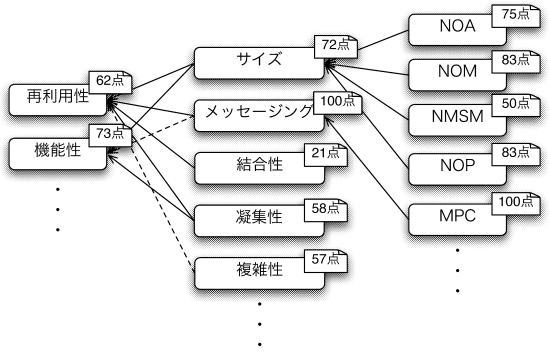

図 2　評価モデル例
Fig. 2　Example of evaluation model.

表 5　品質特性と設計特性の関係
Table 5　Relation of QualityProperty - DesignProperty.

設計特性	再利用	柔軟性	理解性	機能性	拡張性	有効性
サイズ	○		○	○		
抽象性		○	△		○	○
カプセル性		△	△			○
結合性	○	○	○		△	
凝集性	○	△	○	○	○	
コンポジション		○	△			○
継承		○	○		△	○
メッセージング	○		△	△		○
複雑性	△	○	○			

（○：関係が強い，△：関係がある，空欄：関係が弱い/ない）

表 6　選択したメトリクス
Table 6　Selected metrics.

設計特性	メトリクス
サイズ	NOA，NOM，NMSM，NOP
抽象性	NAC，NOAC
カプセル性	NPubM，NPV，NACUM
結合性	NPVUA，CBO，NTWD，NMSSO
凝集性	NOA，NOM，NMNP，NMMI，NMSSO
コンポジション	DAC
継承	NIM，NAM，NPMBC，DIT，NSAIS
メッセージング	MPC
複雑性	NOP，NMHC，CBO，DIT，NTWD

固定点とするかを選択すべきである．本実験では異なる設計図に同じ評価式を用いるため，継承を用いていないことが一概に悪いともいえない．よって今回は測定できない場合の得点を中央値の 50 点とする．

4.2　本評価モデルの妥当性

各サンプルを今回検討する評価式で測定，その結果を比較し，各種サンプルの傾向がとらえられているかを調査する．サンプルの傾向がとらえられていれば，本評価モデルとそれらの比重がある程度妥当であるといえる．

4.2.1　グループごとの平均値の比較

グループごと（初心者/デザインパターン/公開サンプル）に品質特性・設計特性の得点を集計し，各品質特性・設計特性ごとに得点平均を求めた．良い設計であるとされる一般的なデザインパターンの設計図の得点が最も高く，続いて公開されている設計図，初心者が記述した設計図となると予想された．また t 検定を用いてサンプル間に有意な差があるかを検証した．

表 7　グループごとの平均（品質特性）
Table 7　Ave. of each group (quality properties).

対象	再利用性	拡張性	有効性	柔軟性	機能性	理解性
初心者	67	54	66	47	78	63
DP	86	80	88	68	94	83
公開	78	61	77	56	89	73

DP：デザインパターン

表 8　グループごとの平均（設計特性）
Table 8　Ave. of each group (design properties).

グループ	初心者	DP	公開
サイズ	80	100	95
抽象性	50	93	56
カプセル性	57	82	71
結合性	33	60	41
凝集性	64	86	76
コンポジション	72	100	100
継承性	63	63	59
メッセージング	90	100	100
複雑性	69	83	74

DP：デザインパターン

各サンプルの最大値・中央値の品質特性と設計特性得点には差がほとんどみられなかった．これは今回用いたメトリクスの閾値が設計欠陥検出のために定義されたものであったため閾値自体が低めに設定されており，全体として閾値を下回ることが少なかったことが原因と考えられる．また，最大値でも得点が低いメトリクスも存在したが，継承を用いていないためメトリクス値が 50 点であるなど，最大値，中央値，最小値がすべて同じ値になるものが多くみられた．そのため，最大値，中央値に差がみられなかった．最大値，中央値で評価したい場合には，閾値を今回より高めに設定する必要がある．

最小値は継承性以外予想どおり（得点がデザインパターン ≧ 公開サンプル ≧ 初心者サンプル）の結果となった（**表 7**，**表 8**）．継承性の得点が予想に反した原因は，今回検討した評価式では継承を用いていないサンプルが必然的に 60 点になること，継承を用いていないサンプルが多かったこと，NIM・NAM メトリクスでオーバライドを考慮しておらず特にデ

表 9　t 検定の片側検定の結果（p 値）
Table 9　Result of one-sided t-test (p-value).

対象グループ	再利用性	拡張性	有効性	柔軟性	機能性	理解性
DP – 公開	0.0242	0.0008	0.0005	0.0005	0.0654	0.0051
公開 – 初心者	0.1077	0.0322	0.0064	0.0098	0.0729	0.0414

ザインパターンではオーバライドされているメソッドが多かったため得点が低くなったことがあげられる．

各品質特性の得点に関してデザインパターン・公開サンプル間，公開・初心者サンプル間に有意な差があるかをみるため，t 検定の片側検定を行った．その結果を**表 9** に示す．有意水準 5%のとき，公開・初心者サンプル間の再利用性，機能性以外の品質特性はグループ間に有意な差があるといえる．再利用性と機能性は 4.2.2 項でも後述しているが，ほとんどのサンプルで得点が高くなったため有意な差がみられなかった．

4.2.2　品質特性得点の傾向

サンプルの品質特性得点の傾向を調査する．4.2.1 項でも述べたように，最大値，中央値には差が少なく，個々の特徴がとらえにくいので，最小値に的を絞って以下に記述する．メトリクスの集計を最小値で行った場合の各サンプルの品質特性の得点を**表 10** に示す．

- 再利用性
  ほとんどのサンプルで得点が高かった．最も得点の低いサンプルは，責務がうまく分割されておらず，大きな責務を担うクラスが存在していた．そのため，サイズが大きいクラスとなり，結合性やメッセージングなどの得点が共に低くなったため，再利用性の得点が低くなったと考えられる．
- 拡張性
  62 点以下の得点が低いサンプルは，すべて抽象化を用いていなかった．得点が高いサンプルは，抽象クラスを用いており，また他クラスとの関係が疎であるという特徴が見られた．
- 有効性
  抽象化を用いていないサンプルや，参照型の属性が多いサンプル，必要以上にインタフェースを公開しているクラスが存在するサンプルなど，オブジェクト指向の主要な概念に反するサンプルの得点が低くなった．
- 柔軟性
  得点が低い傾向にある．抽象化を用いていないサンプルが多く存在したという原因の他に，継承性の得点も低い傾向にあるためであることが分かった．継承性は NIM，NAM メトリクスでオーバライドを考慮しなかったため高得点を取ることが困難になっていた．この 2 つのメトリクスの測定方法を変更する必要がある．
- 機能性
  ほとんどのサンプルで得点が高かった．簡単で小規模なサンプルが多かったため，サイズ，メッセージングの得点が高くなりやすかったためであると考えられる．得点が低いサンプルは，責務の分割が適切に行われず，属性やメソッド数，また他クラスへのメッセージ送信数が大きいという特徴がみられた．
- 理解性
  理解性は，設計特性すべてから影響を受けるとしたため，総合的な得点に近い．他サンプルに比べ，設計特性の得点が高い Chain of Responsibility サンプルの得点が最も高く，逆に，設計特性の得点が他よりも低い初心者サンプル 4 の得点が最も低かった．初心者サンプル 4 は，抽象化，継承を用いておらず，また属性・メソッド数，他クラスへのメッセージ送信数が多いクラスが存在するなど責務の分割が適切に行われなかったと

表 10　品質特性の得点
Table 10　Score of quality properties.

測定対象	再利用性	拡張性	有効性	柔軟性	機能性	理解性
初心者サンプル 1	79	59	72	54	91	72
初心者サンプル 2	80	57	70	54	89	70
初心者サンプル 3	64	57	72	48	67	64
初心者サンプル 4	33	39	60	41	42	44
初心者サンプル 5	75	57	72	50	83	69
初心者サンプル 6	73	56	52	35	83	61
Builder	92	90	89	73	100	87
Command	82	78	89	67	92	80
State	82	78	89	67	92	80
Chain of Responsibility	96	96	94	78	100	94
Mediator	86	80	89	70	92	83
Memento	82	62	78	58	92	76
Visitor	81	76	85	65	92	78
公開サンプル 1	82	63	80	59	92	78
公開サンプル 2	82	62	78	58	92	76
公開サンプル 3	62	63	81	54	73	63
公開サンプル 4	76	55	74	52	92	69
公開サンプル 5	82	62	77	58	92	76
公開サンプル 6	80	60	74	55	92	74

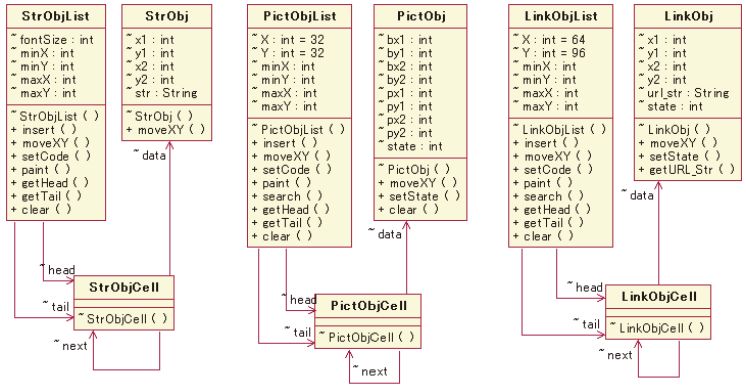

図 3　初心者サンプル 4（クラス図の一部）修正前
Fig. 3　Before remendiating beginner sample 4.

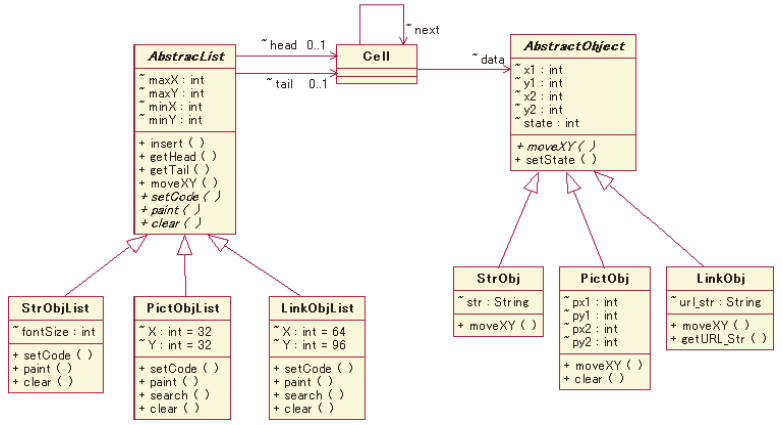

図 4　初心者サンプル 4（クラス図の一部）修正後
Fig. 4　After remendiating beginner sample 4.

思われる．そのため，設計特性の得点が他に比べて低くなったと考えられる．

参考とした QMOOD では再利用性に重点をおいている．この再利用性評価の妥当性は，実際にその UML 設計あるいはその一部を再利用する局面において確認できるものである．本研究では，最初の UML 設計において本質的に有している再利用性をとらえることに限ったが，この再利用性と再利用の局面において生じる再利用性を関係的にとらえることができれば，さらに設計品質の改善に役立つものと考える．

4.3　本評価モデルの適用例

初心者サンプルはすべて情報工学科学生がオブジェクト指向を勉強するために初めて記述した UML 設計図であった．そこでオブジェクト指向概念と技術を設計に適切に取り入れているかを示す品質特性「有効性」の評価に着目してサンプルの修正を行う．オブジェクト指向を適切に取り入れることで，再利用性，拡張性など，全体の品質も上がると予想される．例として，他のサンプルと比較して全体的に得点が低かった初心者サンプル 4 の品質上の問題点を実際に修正し，本ツールが目的に合わせた品質の向上に役立つことを示す．

4.3.1　サンプルの品質上の問題点とその修正

今回，有効性に影響する設計特性は抽象性，カプセル性，コンポジション，継承，メッセージングの 5 つとしている．初心者サンプル 4 には似たような構造を持つクラス群が存在していたが継承，抽象化を用いていなかった．**図 3** にこのサンプルのクラス図の一部を示す．これらのクラス群はそれぞれ文字列，画像，リンクの情報を保持している．初心者サンプル 4 を作成した学生にオブジェクト指向の基本概念である継承・抽象クラスの使用方法を学習させた後で再設計を試み，これらのクラス群を継承，抽象化を用いてまとめた．修正後のサンプルを**図 4** に示す．

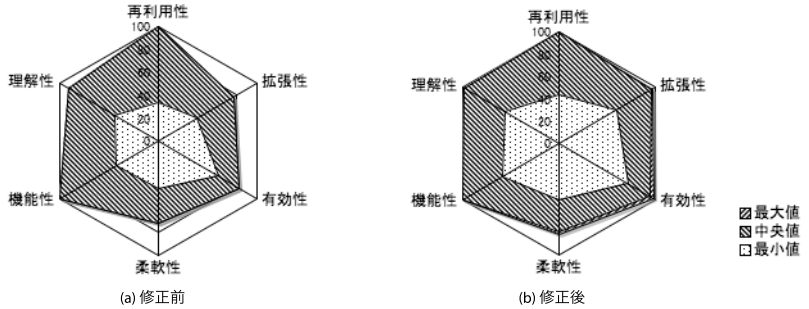

図 5　品質特性の評価
Fig. 5　Evaluation of quality properties.

4.3.2　修正前後の評価値の分析

修正前後の品質特性の得点のバランスチャートを**図 5** に，有効性に関係する設計特性の得点を**表 11** に，抽象性，継承性に関するメトリクスの得点を**表 12** と**表 13** に示す．継承性

表 11　有効性に関わる設計特性の得点（修正前 → 修正後）
Table 11　Score of design properties towards effectiveness.

	カプセル性	コンポジション	メッセージング	抽象性	継承性
最大値	100→100	100→100	100→100	50→89	60→80
中央値	100→ 83	100→100	100→100	50→89	60→71
最小値	50→ 50	100→100	42→ 42	50→89	60→60

表 12　継承性に関わるメトリクスの得点（修正前 → 修正後）
Table 12　Score of metrics towards inheritance.

	NIM	NAM	NPMBC	DIT	NSAIS
最大値	– →100	– →100	– →100	100→100	– →100
中央値	– →100	– →100	– → 88	100→100	– →100
最小値	– → 68	– → 18	– → 75	100→100	– →100

–：測定不能

表 13　抽象に関わるメトリクスの得点（修正前 → 修正後）
Table 13　Score of metrics towards abstraction.

	NAC	NOAC
最大値	0→78	100→100
中央値	0→78	100→100
最小値	0→78	100→100

の評価で用いる NIM，NAM メトリクスについて，上記では抽象メソッドのオーバライドを考慮していなかったがここでは考慮に入れ，オーバライドしているメソッド，つまりスーパクラスとサブクラスの同名メソッドを合わせて 1 つのメソッドとして数えることにする．

抽象クラス，継承を用いたことでオブジェクト指向らしい設計となり，目的どおりに有効性が向上している．またオブジェクト指向を適切に取り入れることによって，再利用性や拡張性など他の品質特性も向上している．

5. 関 連 研 究

XMI を入力とする UML ソフトウェア設計メトリクスツール SDMetrics が Wüst によって開発されている[9]．SDMetrics の提供するメトリクスだけで設計品質を測定することは困難であるが，XML で独自にメトリクスを定義することが可能であり，また設計ルールを測定することも可能であるため，設計品質を測定するためのメトリクス計測ツールとして用いることを検討している．

小坂らは UML モデルの品質測定方法の構築を行っている[12] が，この測定方法は UML の他にも様々な入力が必要であり，定性的にしか評価できない内容も含めている．しかし本研究では UML 図のみで測定できる設計品質を対象とする．したがって測定する品質特性の内容が異なる．

Tian は保守性の早期指標としてクラス図の理解性・変更性を得ることを目標に，サイズ・構造複雑性に関するメトリクスの提案を行い，提案したメトリクスと保守測定を関連付けた予測モデルを検討している[13]．しかしここで提案されているサイズ・構造複雑性メトリクスを用いるだけでは保守性の測定は困難であると思われる．

6. お わ り に

本研究ではソフトウェア品質の低下を抑えるため，UML 設計での測定に適した品質特性を選択し，設計構造の問題を検出するメトリクスとその閾値を用いて，それらの特性を階層的に評価する方法を体系化した．3 つのグループに分けられる複数のサンプルを測定し，得られた測定値がグループ・品質特性ごとの傾向をつかめているかを考察し，検討した品質評価モデルがある程度，品質をとらえられていることを実証した．実際にソフトウェアの品質の向上を支援できることを，測定したサンプルの中で最も品質評価値が低かったものとその結果を考察し，品質上の問題点を修正することで示した．

本研究では他の論文から得られたメトリクスの閾値を用いたが，メトリクスの閾値は一意的に決定できるものではなく，ユーザ自身が持つデータを分析し，目的とあわせて閾値を設定するべきである[11]．しかし複数のメトリクスを用いる場合，閾値を設定すること自体に手間がかかる．そのため，ユーザが持つ多量なサンプルから閾値を求める方法の提案が必要となる．評価モデルの係数についても同様のことがいえる．

今回，品質が良い設計図として，一般的なデザインパターンを用いたが，デザインパターンの性質上クラス数が少なく，とてもシンプルである．そのため本ツールで測定した際に品質が良い設計図としての特徴以上にデザインパターンの特徴が出てしまっている可能性がある．よってサイズと他のメトリクス値の依存関係などの調査をする必要がある．

また，本評価モデルで良いとされた設計がそれを基に実装されたコードでも品質が良くなることで本モデルの有用性が示される．よって設計と実装したコードからメトリクスを測定，品質を評価し，結果を比較することで本モデルが有用であるかをさらに検討する必要がある．

参 考 文 献

1) ISO/IEC 9126-1:2001, Software engineering - Product quality - Part 1: Quality model.
2) Boehm, B. and Basili, V.: Software Defect Reduction Top10 list, *IEEE Computer*, Vol.34, No.1, pp.135–137 (2001).
3) Bansiya, J. and Davis, C.G.: A Hierarchical Model for Object-Oriented Design Quality Assessment, *IEEE Trans. Softw. Eng.*, Vol.28, No.1, pp.4–17 (2002).
4) Dromey, G.R.: Comering the Chimera, *IEEE Software*, Vol.13, No.1, pp.33–43 (1996).
5) Choinzon, M. and Ueda, Y.: Detecting Defects in Object Oriented Designs Using Design Metrics, *Proc. JCKBSE'06*, Tallinn, ESTONIA, pp.61–72 (2006).
6) Choinzon, M., Taki, Y. and Ueda, Y.: Quality Metrics for Object-Oriented Design Assessment，電子情報通信学会技術報告（KBSE），Vol.104, No.725, pp.19–24 (2005).
7) Chidamber, S.R. and Kemerer, C.F.: A Metrics Suite for Object Oriented Design, *IEEE Trans. Softw. Eng.*, Vol.20, No.6, pp.476–493 (1994).
8) Lorenz, M. and Kidd, J.: *Object-Oriented Software Metrics – A Practical Guide*, Prentice Hall (1994).
9) Wüst, J.: The Software Design Metrics tool for the UML (online). available from http://www.sdmetrics.com/ (accessed 2007-10-10).
10) Hitz, M. and Montazeri, B.: Chidamber and Kemerer's Metrics Suite: A Measurement Theory Perspective, *IEEE Trans. Softw. Eng.*, Vol.22, No.4, pp.267–270 (1996).
11) Kan, S.H.（著），古山恒夫，富野　壽（監訳）：ソフトウェア品質工学の尺度とモデル，共立出版株式会社 (2004).
12) 小坂　優，渡辺博之，井山幸次：UML モデル品質測定の指標，組込みソフトウェアシンポジウム ESS2004，pp.164–165 (2004).
13) Tian, J.: Quality-evaluation models and measurements, *IEEE Software*, Vol.21, No.3, pp.84–91 (2004).
14) 結城　浩：増補改訂版 Java 言語で学ぶデザインパターン入門，ソフトバンククリエイティブ (2004).
15) SESSAME WG 2：組込みソフトウェア開発のためのオブジェクト指向モデリング，翔泳社 (2006).
16) 福沢貴則ほか：MDD ロボットチャレンジ 2005 参加報告ムンムン考房，MDD ロボットチャレンジ 2005 産業連携によるモデルベース組込み開発の実践，情報処理学会，pp.107–116 (2006).
17) Objects by Design – Object-Oriented Calculator (online). available from http://www.objectsbydesign.com/projects/calc/overview.html (accessed 2007-10-10).
18) CIS 235 Home Page (online). available from http://www.csupomona.edu/~rvstumpf/cis235spr00/cis235.htm (accessed 2007-12-27).
19) ゼロから学ぶソフトウエア開発完全入門，日経 BP 社 (2005).

(平成 19 年 10 月 18 日受付)
(平成 20 年 4 月 8 日採録)

佐藤　美穂

昭和 59 年生．平成 19 年茨城大学工学部情報工学科卒業．同年より茨城大学大学院理工学研究科博士前期課程情報工学専攻に在籍．オブジェクト指向ソフトウェア設計の評価に関する研究に従事．

田村　真吾

昭和 58 年生．平成 20 年茨城大学大学院理工学研究科博士前期課程情報工学専攻修了．同年（株）リコー入社．在学中はソフトウェア設計やメトリクスに関する研究に従事．

上田　賀一（正会員）

昭和 36 年生．平成元年名古屋工業大学大学院工学研究科電気情報工学専攻博士後期課程修了．同年同大学工学部電気情報工学科助手．平成 2 年茨城大学工学部情報工学科講師．平成 14 年同大学助教授．平成 19 年同大学准教授．現在に至る．ソフトウェア工学，組込みソフトウェアに関する研究に従事．工学博士．電子情報通信学会，日本ソフトウェア科学会，ACM，IEEE Computer Society 各会員．