ソフトウェア工学 135－10
（２００１．１１．２２）

プログラマの視点からのソフトウェアコンポーネント評価手法

佐藤 誠　岡安 二郎　水野 浩之　馬場 茂雄　平山 雅之

株式会社東芝 研究開発センター
システム技術ラボラトリー

概要

近年、コンポーネントを利用したソフトウェア開発が増加している。コンポーネントを利用したソフト開発では、ソフト開発者にとって、システムの開発がしやすい（利用性が高い）コンポーネントを選択することがポイントとなる。本稿では、ソフト開発を行うコンポーネント利用者の視点から、コンポーネント利用性を評価することを目的とするコンポーネント利用性評価手法を提案する。提案する手法は、

①プログラマのコンポーネント利用方法の分析をもとに「コンポーネント選定時の機能把握容易性」「コンポーネント利用時の実装作業容易性」などの副特性から構成される評価モデルを定義し、

②ソースコードを持たないブラックボックス・コンポーネントに対して、ソースコード以外から取得できる限られた情報を用いてこれらの副特性を計測するメトリクスを設定する

といった特徴を持つ。また、この手法の有効性確認を目的とした適用評価実験を行った結果、コンポーネントの利用性について、提案する手法（定量評価手法）と実際の作業者のコンポーネント利用時の印象（定性評価）はよく一致することが確認できた。これより本稿で提案するコンポーネント利用性評価手法が有効であることが確認できた。

The Method of Software Component Usability Evaluation Based on Programmers' Viewpoints

Makoto SATO, Jiro OKAYASU, Hiroyuki MIZUNO, Shigeo BABA and Masayuki HIRAYAMA

TOSHIBA Corporation, Corporate Research & Development Center,
System Engineering Laboratory

Abstract

Recently, software components have been used widely used in software development. In the component based software engineering, it is especially important to evaluate usability of software components. In this paper, we discuss the usability evaluation method for software components, which is composed of usability evaluation viewpoints and their evaluation metrics. The proposed method has following features;

1) Software component usability evaluation model based on the way of using software components by programmers.
2) The technique that make it possible to evaluate black-box software component by using information except for its source code.

From our application experiment, we confirmed the effectiveness of proposed method. That is, with using proposed method, we can evaluate the usability of software components effectively.

1. はじめに

近年のソフトウェアは様々な機能を実現するために複雑になってきている。こうした複雑なソフトウェアを少しでも効率的に開発するための方法として、ソフトウェア・コンポーネント（以下、コンポーネントと略記）の活用が広まりつつある。特に最近では、JavaBeans、ActiveX DLL などで様々な種類のコンポーネントが提供され、これらの流通インフラも整備されている。この結果、ソフトウェア開発者はこれらのコンポーネントを容易に入手し、ソフト開発に利用できるようになった。しかしながら、こうしたコンポーネントの普及と活用は、コンポーネントをベースにしたソフトウェア開発に、①どのようなコンポーネントを利用するか、②利用したコンポーネントの品質や信頼性が保証できるか、といった新たな問題を引き起こしている。利用したコンポーネントの使い勝手や実装のしやすさに問題があると、こうしたコンポーネントを利用することで、かえって開発効率を下げることにもなりかねない。また、利用したコンポーネントの品質や信頼性が低いと、結果としてこれらを組み込んだソフトウェア全体の品質を下げることにもなりかねない。こうした問題を未然に防ぐためには、コンポーネントの使い勝手や品質、信頼性を予め評価する仕組みが必要である。本論では、JavaBeans などに代表されるソフトウェア・コンポーネントについて、その利用性を評価するための手法を提案する。また、この評価手法を実際の JavaBeans コンポーネントに対して適用し、その利用性を評価した実験についても合わせて報告する。

2. コンポーネント利用性評価の課題点と関連研究

2.1. コンポーネント利用性評価の課題点

本論で扱うコンポーネントの利用性とは、「コンポーネントの使い勝手」と言い換えることもできる。この「コンポーネントの使い勝手の評価」については、評価方式の問題、評価情報獲得の問題、評価視点（モデル）といった点が現状の課題と考えられる。

(1) 評価方法の問題

一般にコンポーネントの評価を考える場合、静的評価、動的評価の２つに分けて考えることができる。

静的評価：評価対象であるコンポーネントを動かすことなく評価する方法。通常は、カタログやマニュアルといったコンポーネントに付随する情報を利用した評価となる。

動的評価：評価対象であるコンポーネントを実際にソフトウェアに組み込んで利用し、動かしてその特性を評価する。

しかしながら、これら２つの方法はそれぞれ長所短所がある。ドキュメントの情報などを利用した静的評価では表面的な評価までしかできず、実際にそのコンポーネントの使い勝手までこれらの情報で評価・判断することは難しいといった問題がある。

一方、動的評価では、利用候補のコンポーネントを実際に実装して利用するため、コンポーネントの使い勝手について、より詳細で具体的な評価ができる反面、評価に要するコストは極めて大きくなるといった問題がある。

(2) 評価情報獲得の問題

ソフトウェアの利用性を効率的に評価することを考えた場合、静的評価の精度を上げるアプローチを考えることができる。即ち、ソフトウェアに付随したドキュメントの情報だけではなく、ソースコード解析などの手法で、ソフトウェアの構造上の特性から、その使い勝手を類推する方式が有効である。しかしながら、対象がコンポーネントである場合には、ソースコードなどの内部情報が必ずしも提供されるとは限らない。このため、コンポーネントの静的評価では、どのような情報を、どこから獲得するかが大きな問題となる。

(3) 評価視点（モデル）の問題

従来より、ソフトウェアの品質評価については、Boehm や ISO/IEC 9126 などによって、その評価視点(モデル)が提案され利用されている[1]。また、オブジェクト指向ソフトウェアについても、Chidamber、Kemerer らの CK メトリクス[2]などが提案されている。しかしながら、これらをコンポーネントの評価に利用した場合には、ソースコードが提供されない場合、ソースコード内部情報を必要とするようなメトリクスについて計測できないものが少なくない。

また、これらのソフトウェア品質評価モデルは、開発したシステムの全体に関する特性を評価することを主な目的としており、対象とするソフトウェアの粒度は比較的大きなものを想定している。このため、コンポーネント利用者がコンポーネント利用する際の評価という観点からは、使い勝手の悪い評価モデルになっている。

2.2. 関連研究

我々の研究では、これらの課題を解決して、ソフトウェア・コンポーネントを利用する際の使い勝手を、効率的に評価するための手法の開発を目的としている。このために我々が提案する方式は、

① コンポーネントの静的評価の精度を上げて、ブラックボックス・コンポーネントの利用性に関する情報を取得し、
② コンポーネントの利用性評価に適した評価モデルに照らし合わせて、その特性を定性評価する。

といった特徴を持っている。これにより、コンポーネントの静的評価でも、動的評価と同等の評価結果を得ることができる。

コンポーネント評価に関しては、以下のような関連

研究が報告されているが、「コンポーネントの使い勝手」を評価するという目的に照らし合わせると、必ずしも十分とはいえない。

① コンポーネントに関する静的情報獲得

ソースコードが公開されないようなブラックボックス・コンポーネントから、メトリクス計測に用いる情報を抽出する方法が提案されている[3][4]。これらの方法は、ブラックボックス・コンポーネントであっても、例えばC/C++のDLLではヘッダファイルを解析することでクラス情報やメソッド情報を抽出したり、JavaBeansに対してはJavaのリフレクション機構を用いてメモリ上のオブジェクトの内部情報を抽出するなどの方法を用いている。

② 評価視点（モデル）

さらに関連研究[4]では、このようにして抽出した内部情報を用いて、既存のメトリクスや独自に追加したメトリクスの計測を行っている。しかし、コンポーネント利用者の視点からの「コンポーネントの利用性」――特に「コンポーネントの組み込み・実装のし易さ」といった観点までは議論されていない。

以上より、本報告では、評価視点（モデル）に焦点を合わせ、コンポーネント利用者の視点からの「コンポーネントの利用性」――特に「コンポーネントの組込み・実装のし易さ」といった観点を中心とした新しい評価モデルを提案し、実際の適用事例をもとに考察を加える。

3. コンポーネント利用性評価モデル

3.1. 評価フレームワーク

我々が目指すコンポーネント利用性評価のフレームワークは、実際のソフト開発者のコンポーネント利用方法に基づいて定義したコンポーネント利用性評価視点（モデル）と、これを定量的に計測するための評価メトリクスから構成される。これによって「コンポーネントの使い勝手」を静的に評価するものである。

・コンポーネントを利用してソフト開発を行うプログラマの視点から、コンポーネント利用時にどのような特性があれば利用性が高いと言えるかを導出し、コンポーネント利用性評価モデルとして定義する。

・上記評価モデルの特性を評価するメトリクスとして、ブラックボックス・コンポーネントから取得できる情報により計測可能なメトリクスを定義する。

このようなアプローチにより、従来の動的評価を代替し、動的評価と同等の評価結果を得ることができるコンポーネント静的評価手法を実現する。

3.2. 評価視点（モデル）の定義

コンポーネントの利用性を評価する場合、コンポーネントがどのような流れで実際に利用されるかを考慮することが必要である。一般に、ソフトウェア開発サイクルは通常、分析、設計・実装、テスト・デバッグ、保守などの工程に分類される。コンポーネントを利用したソフト開発において、コンポーネントの使い勝手が特に重要となるのは設計・実装工程であるため、ここでは設計・実装の観点に絞って、コンポーネント利用性評価視点（モデル）を定義する。

コンポーネントを利用する場合、設計・実装では、

① コンポーネントを選定し、そのおおよその機能などを理解するフェーズ

② 実際に組み込み対象のソフトウェアにコンポーネントを組み込むフェーズ

の２つに分けて考えることができる。これらは、それぞれ、**「コンポーネント選定時の機能把握容易性」「コンポーネント利用時の実装作業容易性」**の２つの副特性によって評価できる。なお、これらの副特性を考慮したコンポーネント利用性評価モデルを図１に示す。

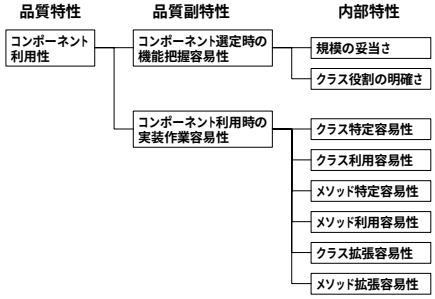

図 1：コンポーネント利用性評価モデル

副特性―1：コンポーネント選定時の機能把握容易性

コンポーネントを選定し、導入しようとする際には対象とするコンポーネントがどのような機能を持ち、これを利用するとき、どのクラスを使えば実現できるかなどが理解しやすいことが重要である。このため、コンポーネント利用性の副特性として、**コンポーネント選定時の機能把握容易性**を考えることができる。

コンポーネント選定時の機能把握が容易であるためには、コンポーネントの規模が大きすぎないこと、クラスの役割が分かりやすいことなどが言えることが重要であるから、コンポーネント選定時の機能把握容易性を詳細化し、**規模の妥当さ**、**クラス役割の明確さ**という内部特性を定義する。

副特性―2：コンポーネント利用時の実装作業容易性

コンポーネントを導入後、実際にコンポーネントをシステムに組込む際の実装作業の容易さが重要であるため、コンポーネント利用性の副特性として、**コンポーネント利用時の実装作業容易性**が考えられる。

コンポーネントの実装作業は、コンポーネントをそのまま利用する場合と拡張して利用する場合の

２つの場合を考える必要がある。

コンポーネントの機能のうち、利用したい機能の効果を得るためには、その機能を実装するクラスと、クラスに実装されているメソッドを特定して、これを利用する。これらの実装作業が簡単であるほど利用性が高いと考えられる。これらより、コンポーネント利用時の実装作業容易性は、**クラス特定容易性**、**クラス利用容易性**、**メソッド特定容易性**、**メソッド利用容易性**などの内部特性によって評価できる。

また、コンポーネント本来の機能に対して、ソフト開発者が拡張して利用する場合には特定したクラスやメソッドが拡張しやすい方がよく、**クラス拡張容易性**、**メソッド拡張容易性**などの内部特性を利用して評価できる。

3.3. 評価メトリクス

表 1、表 2 に図１で示した評価視点（モデル）に対応する評価メトリクスを示す（各内部特性の列で、○印が付いているメトリクスが対応する）。なお、提案しているメトリクスはいずれも、C++ライブラリなどのヘッダファイルを解析することで計測可能である。

例えば、「副特性-1：コンポーネント選定時の機能把握容易性」に関連する内部特性「規模の妥当さ」を評価する場合には、表 1 に示すように(1)クラス数、(2)ルートクラス数、(3)クラス毎のメンバ数、の３つのメトリクスを計測することで評価できる。ここでそれぞれのメトリクスは、以下のような意味合いを持つ。

表 1：コンポーネント選定時の機能把握容易性

メトリクス ＼ 内部特性		規模の妥当さ	クラス役割の明確さ
(1)	クラス数	○	
(2)	ルートクラス数	○	
(3)	クラス毎のメンバ数	○	
(4)	クラスの結合数		○
(5)	クラスのカプセル化の強さ		○
(6)	クラスのメンバ構成		○

表 2：コンポーネント利用時の実装作業容易性

メトリクス ＼ 内部特性		クラス特定容易性	クラス利用容易性	メソッド特定容易性	メソッド利用容易性	クラス拡張容易性	メソッド拡張容易性
(4)	クラス結合数	○	○			○	
(5)	クラスのカプセル化の強さ	○	○			○	
(6)	クラスのメンバ構成	○	○			○	
(7)	クラスのネーミング規則性	○	○			○	
(8)	メソッド機能度の規則性			○	○		○
(9)	メソッドのネーミング規則性			○	○		○
(10)	I/Fの仮引数の明示度			○	○		○
(11)	メソッド名の関連度				○		
(12)	メソッドのオーバーロード数				○		
(13)	メソッドのデフォルト実装度				○		
(14)	クラス毎のメソッド数					○	
(15)	継承が必要なクラスの割合		○				
(16)	継承が可能なクラスの割合					○	
(17)	継承不可能なクラスの割合					○	
(18)	継承が必要なメソッドの割合				○		○
(19)	継承が可能なメソッドの割合						○
(20)	継承不可能なメソッドの割合						○

(1) クラス数：コンポーネントが持つクラスの総数。クラスをユーザに機能を提供する塊と見なし、クラス数を数え、コンポーネントの規模を推測する。

(2) ルートクラス数：コンポーネントが持つクラスの中で、継承関係の頂点にあたるクラスの数。クラスが機能を提供する塊であるとすると、継承されたクラスはそのサブ機能と言える。したがって、ルートクラスはそれらサブ機能群の大元の機能に相当し、ルートクラス数からコンポーネントの大分類的な規模を推測できる。

(3) クラス毎のメンバ数：クラスが持つメソッドやメンバ変数の数。これらはクラスを構成する要素であり、クラスがどれだけの要素から構成されているかを数えることで、クラスの規模を類推できる。

4. 評価実験

4.1. 実験目的

提案したコンポーネント利用性評価手法によって、実際のコンポーネントの使い勝手を評価することが可能であることを確認するために実験を行った。この実験の主たる目的は、評価モデルと評価メトリクスの妥当性の確認である。特に本実験では**クラスの規模および継承関係がコンポーネントの使い勝手に与える影響**を中心に調べることを目的とした。このために、本手法を用いたコンポーネント評価結果と、実際にソフト開発者がコンポーネントを使用して評価した結果を比較し、本手法の評価結果が妥当であることを確認する実験を行った。

4.2. 実験対象

実験に用いるコンポーネントとして、コンポーネント内部情報取得の容易さや、入手できるサンプルの豊富さなどから、JavaBeans を実験対象とした。

JavaBeans は、jars.com [1] や jfind.com [2] などのコンポーネント公開サイトで多くのコンポーネントを入手しやすく、またそれらを実際に利用したユーザによる評価結果も併せて公開されていることから、前述した関連研究[4]においても利用されている。

[1] http://www.jars.com
[2] http://www.jfind.com

jars.com では、9 種類のカテゴリに分類されて多数のコンポーネントが登録されている。この中から、ソフトウェア部品としての性質が高い Programming カテゴリを選び、規模や機能に関わらず、無作為に 34 個のコンポーネントを抽出した。これらのコンポーネントには、実際に使用したユーザの主観に基づき、0.5 単位で最高 4 点の評価点が付けられている[3]。これら 34 個のコンポーネントの評価点の内訳は、表 3 の通りであった。この評価結果を参考に、評価の優劣をより明確にするため、34 個のコンポーネントの中から評価点 3.5 のものを除き、評価点 4 のコンポーネント 10 個と、評価点 3 以下のコンポーネント 14 個を実験対象として採用した。

表 3：jars.com のコンポーネント評価点の分布

評価点	4	3.5	3 以下
コンポーネント数	10 個	10 個	14 個

4.3. 実験手順

実験は、予備実験と本実験から構成される。

予備実験：採用するメトリクスの基準値を求めることを目的とする。前節で挙げた jars.com のコンポーネントをサンプルとしてメトリクスを計測し、評価が高いコンポーネントのメトリクス計測値が取り得る適正範囲（基準値）を決定する。

本実験：実際のコンポーネントをこの手法で提案する提案するメトリクスによって定量評価する。また、実際のコンポーネント利用者に使い勝手に関するインタビューに基づき定性評価を行う。

4.3.1. 予備実験

実験の流れを図 2 に示す。評価の高いコンポーネント 10 個と、評価の低いコンポーネント 14 個について、それぞれメトリクスを計測し、各メトリクス計測値の分布を比較することで、メトリクス値の適正範囲を決定する。今回の実験では、クラスの規模、継承関係に関する以下の 5 つのメトリクスを対象とした。

・規模の妥当さに影響するメトリクス
　①クラス数、②ルートクラス数
・クラス利用容易性に影響するメトリクス
　③継承が必要なクラスの割合
・クラス拡張容易性に影響するメトリクス
　④可能なクラスの割合、⑤継承不可能なクラスの割合

4.3.2. 本実験

本実験はコンポーネント利用性に関しての定量評価、定性評価から構成される。実験の流れを 3 に示す。

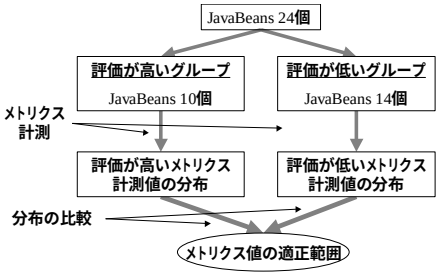

図 2：予備実験の流れ

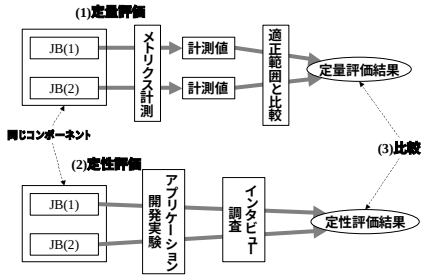

図 3：本実験の流れ

(1)定量評価

予備実験で用いたものとは別のコンポーネント、JB1（仮名）と JB2（仮名）の 2 種類を用意し、予備実験と同じく 5 つのメトリクスについて計測する。これらの計測値と、予備実験で得た適正範囲を比較することにより、コンポーネントの内部特性を評価する。

(2)定性評価

表 4に示す 4 人のプログラマを被験者とし、定量評価で用いたコンポーネントを利用して表 5 に示す 4 つのアプリケーション開発実験を行う。その際、コンポーネントの利用場面や利用方法についてインタビュー調査を行い、使いやすかった特徴などを主観的に評価してもらう。

表 4：被験者

被験者-A	**ソフトウェア開発に従事する上級プログラマ**
被験者-B	**ソフトウェア開発経験が3年程度の中級プログラマ**
被験者-C	**ソフトウェア開発経験が浅い初級プログラマ**
被験者-D	**同上**

表 5：実験で開発したアプリケーション

アプリ-1	**JB1の機能をそのまま利用して実現できる小規模アプリケーション**
アプリ-1'	**JB1の機能を自分で拡張しなければ実現できない小規模アプリケーション**
アプリ-2	**JB2の機能をそのまま利用して実現できる小規模アプリケーション**
アプリ-2'	**JB2の機能を自分で拡張しなければ実現できない小規模アプリケーション**

[3] 実際に使用したユーザによる、表現性・機能性・新規性の総合評価。厳密にはコンポーネント利用性と同等ではないが、傾向は類似すると思われ、ここではコンポーネント利用性の評価結果を代替するものとした。

5. 実験結果

5.1. 予備実験

前述の評価が高いコンポーネント 10 個を Score High のグループ、評価が低いコンポーネント 14 個を Score Low のグループとする。これらに対し、メトリクス計測を行った結果を示す。

5.1.1. 実験データ

(1) クラス数

図 4に、コンポーネントが持つクラス数の分布を示す。2 つのグループ間に有為差があるか仮説検定※1 を行ったところ、Score High のコンポーネントと、Score Low のコンポーネントの間に有為差は認められなかったが、評価の高いコンポーネントはクラス数 120 個以下に収まっている傾向が見られた。これより、クラス数があまりに多いコンポーネントは理解性の低下に繋がると考えられる。

(2) ルートクラス数

図 5に、コンポーネントが持つルートクラス数の分布を示す。Score High と Score Low の間に有為差は認められなかったが、評価の高いコンポーネントはルートクラス数 12 個以下に収まるという傾向が見られた。コンポーネントの機能を実現するのがクラスだとすると、ルートクラス数は機能の大分類の数を表し、クラス数はその関連機能を表すと解釈できる。これより、ルートクラス数が多すぎるコンポーネントは機能規模が複雑になり、理解しにくくなると考えられる。

(3) 継承が必要なクラスの割合

図 6に、継承が必要なクラスの割合を示す。Score High と Score Low の間に有為差が認められ、分布の 95%信頼区間より、評価の高いコンポーネントは 6%以下の継承が必要なクラスを持つ傾向があった。評価の高いコンポーネントは継承しなければ使えないクラスが少なく、利用しやすいと考えられる。

(4) 継承が可能なクラスの割合

図 7に、継承が可能なクラスの割合を示す。Score High と Score Low の間に有為差が認められ、分布の 95%信頼区間より、評価の高いコンポーネントは継承が可能なクラスを 90%以上持つ傾向が見られた。継承が可能なクラスの割合が高い方が、コンポーネントを拡張して利用しやすいと考えられる。

(5) 継承不可能なクラスの割合

図 8に、継承が不可能なクラスの割合を示す。Score High と Score Low の間に有為差が認められなかったが、評価の高いコンポーネントは 3%以下の継承不可能なクラスを持つ傾向があった。継承できないクラスの割合が低い方が、コンポーネントを拡張して利用しやすいと考えられる。

※1 分布は正規分布であるとし、有為水準 0.05 で片側検定を行った。

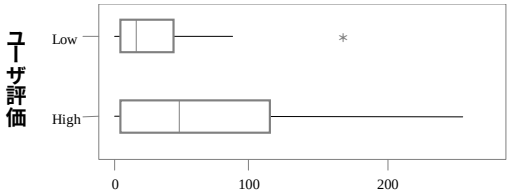

図 4：クラス数（個）

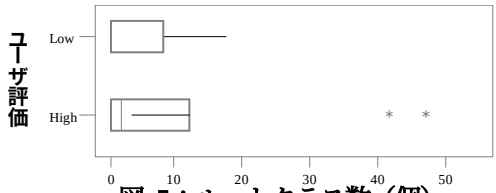

図 5：ルートクラス数（個）

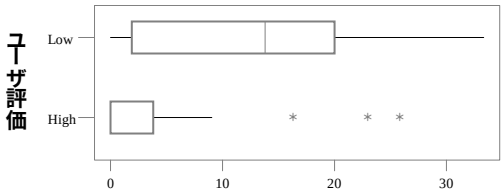

図 6：継承が必要なクラスの割合（%）

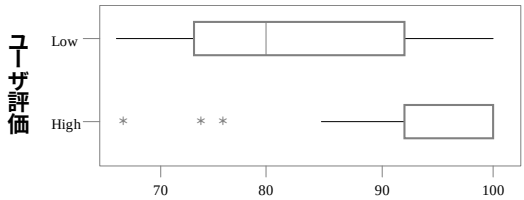

図 7：継承が可能なクラスの割合（%）

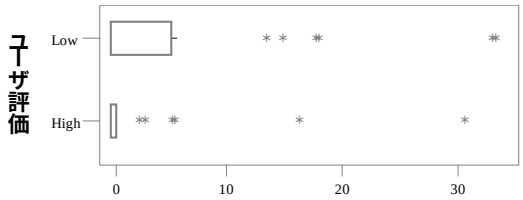

図 8：継承不可能なクラスの割合（%）

5.1.2. メトリクス計測値とユーザ評価傾向のまとめ

(1)～(5)より決定したメトリクス計測値の適正範囲を表 6に示す。これらの適正範囲をコンポーネント利用性の評価基準値として利用する。

表 6：メトリクス計測値の適正範囲

メトリクス	基準値
クラス数	120個以下
ルートクラス数	6個以下
継承が必要なクラスの割合	コンポーネントをそのまま利用するとき 6%以下
継承が可能なクラスの割合	コンポーネントを拡張して利用するとき 90%以上
継承不可能なクラスの割合	コンポーネントを拡張して利用するとき 3%以下

5.2. 本実験

5.2.1. 定量評価

ここでは、実験対象とした2つのコンポーネントJB1とJB2に対してメトリクスを計測し、前述の基準値と比較してコンポーネントの内部特性を評価した結果を示す。

(1) 規模の妥当さ

規模の妥当さの観点については、クラス数とルートクラス数をもとに評価する。表 7に、JB1とJB2のクラス数とルートクラス数の計測値を示す。

クラス数：JB1のクラス数27個で基準値内に収まっているが、JB2は170個となっており基準値（120個以下）を大幅に越えていることがわかった。

ルートクラス数：JB1が5個、JB2が3個となっており、どちらも基準値内（6個以下）に収まっていることが確認できた。

これらより、規模の妥当さについては、JB1がJB2より若干勝っていると判断できる。

(2) クラス利用容易性、拡張容易性

クラス利用容易性および拡張容易性については、継承可能あるいは必要なクラス数、継承不可能なクラス数によって評価する。表 8に継承が必要なクラス、継承が可能なクラス、継承不可能なクラスの割合を示す。

継承が必要なクラスの割合：JB1は14.8%、JB2が4.1%であり、JB1は基準値(6%以下)を超えている。

継承が可能なクラスの割合：JB1は85.2%、JB2が88.2%であり、どちらも基準値(90%以上)を下回っている。

継承不可能なクラスの割合:JB1の0%に比べて、JB2は7.7%と多く、基準値（3%以下）を超えていることが確認できた。

これらを総合的に判断すると、コンポーネントをそのまま利用するときのクラス利用容易性は、JB1は評価が低く、JB2は評価が高いと考えられる。また、コンポーネントを拡張して利用するときのクラス拡張容易性は、JB1は中、JB2は低いと評価できる。

なお、この評価結果をまとめたものを表 9に示す。

表 7：規模の妥当さに関する評価

	JB1	JB2
クラス数	27	170
ルートクラス数	5	3

表 8：クラス利用容易性、クラス拡張容易性の評価

	JB1	JB2
継承が必要なクラスの割合	14.8 %	4.1 %
継承が可能なクラスの割合	85.2 %	88.2 %
継承不可能なクラスの割合	0 %	7.7 %

表 9：JB1 と JB2 の定量評価結果

内部特性	評価 JB1	評価 JB2	メトリクス	評価 JB1	評価 JB2
規模の妥当さ	高	中	クラス数	高	低
			ルートクラス数	高	高
クラス利用容易性	低	高	継承が必要なクラスの割合	低	高
クラス拡張容易性	中	低	継承が可能なクラスの割合	低	低
			継承不可能なクラスの割合	高	低

5.2.2. 定性評価

被験者がJB1とJB2を利用してアプリ仕様1、1'、2、2'を開発した際の、コンポーネントの使い勝手の評価結果を表 10に示す。また、これらの実装作業において、被験者がコンポーネントに対する印象をインタビュー調査した結果を表 11に示す。

表 10：アプリ開発実験結果

	被験者 A（上級者）	被験者 B（中級者）	被験者 C（初級者）	被験者 D（初級者）
仕様1	◎	○	◎	○
仕様1'	○	○	×	×
仕様2	◎	○	○	◎
仕様2'	○	△	×	×

◎利用しやすかった　○どちらかといえば利用しやすかった
△どちらかといえば利用しづらかった　×利用しづらかった

表 11：JB1、JB2 の利用に関するインタビュー結果

JB1に関する意見 **・機能分割や継承関係が適切** **・適度に複雑で高度な利用ができる**
JB2に関する意見 **・機能が多目的でいろいろなことができる** **・どの機能をどのクラスで実現しているかがわかりにくい** **・個々の機能は単純だが拡張してもたいしたことはできない**

以上の実験結果およびインタビュー結果から、次のことが言える。

1)JB1 は機能分割や継承関係などの設計がわかりやすく、機能把握が容易であったと言える。

2)JB2 はクラスが多すぎて分かりにくい点が指摘されており、機能把握が難しかったと言える。

3)コンポーネントをそのまま利用するとき、アプリ仕様1、2がどちらもうまく実装できたことから、JB1、JB2ともに実装作業は容易だったと言える。

4)JB1を拡張して利用するときは、初級プログラマには難しく、ある程度のスキルを持つ中級以上のプログラマはうまくできたことから、ある程度拡張して利用しやすかったと言える。

5)JB2を拡張して利用するときは、拡張性の低さがイ

ンタビュー結果でも指摘されており、拡張して利用しづらかったと言える。

これらより、定性評価結果を表 12にまとめる。

表 12：JB1 と JB2 の定性評価結果

利用性の副特性	JB1	JB2
機能把握容易性	高	低
実装作業容易性（そのまま利用）	高	高
実装作業容易性（拡張利用）	中	低

6. 考察

6.1. 定量評価結果と定性評価結果の比較

ここでは表 9に示した定量評価結果と表 12の定性評価結果を比較する。

(1) コンポーネント選定時の機能把握容易性

コンポーネント選定時の機能把握容易性については、内部特性として「規模の妥当さ」を定量評価した。定量評価結果、定性評価結果ともに JB1 の評価が JB2 の評価より高くなっており、評価結果は一致している。

(2) コンポーネント利用時の実装作業容易性

コンポーネントをそのまま利用したときの実装作業容易性については、内部特性としてクラス利用容易性の評価結果が参考になる。この定量評価結果と定性評価結果については、JB2 の評価については一致したが、JB1 の評価については食い違いが見られた。

また、コンポーネントを拡張利用した際の実装作業容易性については、内部特性としてクラス拡張容易性の評価結果が参考になる。これについては、定量評価結果と定性評価結果は JB1、JB2 とも評価結果が一致した。

6.2. 実装作業容易性の評価結果の差異

コンポーネント利用性に関して、提案手法による定量評価とインタビューによる定性評価の結果は大方が一致する傾向が確認できた。しかし、コンポーネントをそのまま利用する際の実装作業容易性の評価については、定量評価結果と定性評価結果に食い違いが見られた。コンポーネントをそのまま利用する際の実装作業容易性は、継承が必要なクラスの割合をもとに定量評価を行った。しかしながら、継承が必要なクラスの割合がコンポーネント利用性に影響するのは、実際にコンポーネントを利用するときに、それらのクラスを使う必要があるときである。メトリクス計測値はそのようなクラスを使うことになる確率を表すため、今回の実験では、それらのクラスを使わないでアプリが実現できる仕様であったことから、実装作業容易性との結びつきが弱く、これが原因となって定量評価と定性評価の結果に食い違いが生じたと考えられる。コンポーネントが持つクラスを網羅的に用いる場合などでは、クラス利用容易性が実装作業容易性に与える影響が大きくなると考えられる

6.3. 評価モデルの妥当性

定量評価結果と定性評価結果の一致度合いなどをみると、コンポーネントの利用容易性、特に今回実験の範囲とした「コンポーネント選定時の機能把握容易性」「（コンポーネントを拡張して利用する際の）実装作業容易性」については、定量評価と定性評価の結果がよく一致している。これより、提案した評価視点（モデル）およびメトリクスによってコンポーネントを定量評価することで、おおよその利用性を把握することが可能であると言える。

また、「（コンポーネントをそのまま利用する際の）実装作業容易性」については、提案のメトリクスでは必ずしも適切にその特性を評価しきれず、このメトリクスに加え、評価条件や他のメトリクスとの複合判定が必要と考えられる。

7. おわりに

本稿では、ソフトウェア開発におけるプログラマの実際のコンポーネント利用シーンを考慮したコンポーネント利用性評価手法を提案した。また、提案の評価モデルの有効性を示すため、JavaBeans を対象とした評価実験を行い、提案手法によってコンポーネント利用性が適切に評価できることを検証した。さらにこれにより、本手法で採用したコンポーネント利用性評価視点（モデル）が実際のユーザの感覚に合致したものであることも確認した。

一方で、今回行った実験では、評価視点（モデル）やメトリクスを限定しており、十分に検証できていない観点、メトリクスも残っている。これらについては引き続き実験を行い検証していきたい。また、予備実験で求めた各メトリクスの評価基準値についても暫定的に求めたものであるため、今後、様々なコンポーネントの計測を通して基準値の精度向上を図っていきたい。

参考文献

[1] ソフトウェア品質評価ガイドブック，財団法人日本規格協会，1994

[2] Chidamber, S., Kemerer, C., “A Metrics Suite for Object Oriented Design”, IEEE Trans. on Software Engineering, 1994

[3]佐藤，岡安，田村，水野，平山，“ ソフトウェアコンポーネント評価手法の提案 ”，情報処理学会第 62 回全国大会講演論文集(1)，pp281-282，2001 年

[4]山本，鷲崎，深澤, “静的側面から見たソフトウェアコンポーネントの品質”，信学技報 TECHNICAL REPOPRT OF IEICE, SS2001-10, 2001