ソフトウェア開発におけるユーザインタフェース設計評価方法の位置付け方法の提案

来住伸子[†] 甲洋介[††] 山本理浩[††]

ユーザインタフェース設計の様々な評価方法が提案されているが，実際のソフトウェア開発で，いつ，どれを使用すべきかについての手法はほとんど提案されていない．そのため，仕様レビューや製品検査などの各段階でユーザインタフェース評価が行われても，その結果が，ユーザインタフェース設計に効果的に反映されないことがあるのが現状である．そこで，ソフトウェア開発において使用すべき評価方法の選択の目安として，評価方法の有用性の評価基準を提案し，さらに，評価方法の位置付けの枠組を提案する．有用性の評価基準は，(1)評価の時期，(2)発見できる問題点の数と重大度，(3)評価にかかる労力，(4)評価者による個人差の4点からなる．また，評価方法の位置付けの枠組では，(1)評価の手法—発見的か形式的か，(2)評価の対象—システムか仕様か，(3)評価に参加するユーザ—実際のユーザか仮想されたユーザか，の3要素で，各種評価方法を位置付ける．これらの提案が有効であるかどうかを検証するために，4種の評価方法を同一のソフトウェアに対して使用した経験を検討した．その結果，有用性の評価基準と評価方法の位置付けの枠組の関係がある程度確認でき，ソフトウェア開発において複数のユーザインタフェース評価方法を採り入れる必要性を示すことができた．

Categorization of Usability Evaluation Methods in Software Development Processes

Nobuko Kishi,[†] Yosuke Kinoe[††] and Michihiro Yamamoto[††]

While various usability evaluation methods have been proposed, there are few criteria and categorization of usability evaluation methods which are useful in choosing a method suitable for a particular product development process. Inappropriate usability evaluation methods are often applied in design reviews and product quality tests, whose results are not reflected in the product design. We propose criteria to choose a usability evaluation method in a software development process, and categorization methods to characterize various evaluation methods. Four criteria were proposed to evaluate the methods : the number of usability problems, the timing of an evaluation, the variations in usability caused by testers, and the workload required for an evaluation. Categorization of evaluation methods were also proposed : formal vs. heuristic, specification vs. system, and hypothetical users vs. real users. Using these criteria and categorization, we examined four usability evaluation methods used in a software development process. We found out that a single usability evaluation method could not have discovered all the usability problems. Furthermore we argue that a development process needs to include several usability evaluation methods, which have different characteristics, based on the proposed categorization methods.

1. はじめに

ユーザインタフェース設計の評価方法は多数提案され，実際に使用されている方法が多いが，ソフトウェア開発における標準的なユーザインタフェース評価方法と呼べる評価方法はほとんどない．評価者もしくは製品によって異なる評価方法が使用されることが多い．不適当な評価方法が選択されたため，信頼性の低い評価結果が出されたり，評価を実施する時期が不適切であるため，評価結果がユーザインタフェース設計に生かされないということが起きやすい．

そこで，この研究では，ソフトウェア開発において使用すべき評価方法の選択の目安として，評価方法の有用性の評価基準を提案し，また，評価方法の位置付けの枠組を提案する．これらの提案が有効であるかどうかを評価するために，4種の評価方法を同一のソフトウェア開発に使用した例を分析した．その結果，有

† 津田塾大学学芸学部数学科
Department of Mathematics, Faculty of Liberal Arts, Tsuda College
†† 日本アイ・ビー・エム株式会社 アジア・パシフィック製品開発統括本部
IBM Japan, Ltd.

用性の評価基準と評価方法の位置付けの枠組の関係がある程度確認でき，ソフトウェア開発においてユーザインタフェース評価方法を複数とり入れる必要性が確かめられた．さらに，この評価方法の位置付けの枠組から，複数の評価方法を効果的に運用するための今後の研究方向を提案する．

様々な評価方法が存在する主な理由は，二つある．一つは，使いやすさには多くの側面があり，評価方法によって注目している使いやすさの側面が異なるという点と，もう一つはユーザインタフェース設計の評価を行う目的自体が，次のように分化してきているという点があげられる．

1. 人間（ユーザ）の認知機構について新しい知見を得る参考資料とする．
2. 製品選定の参考資料とする．
3. ソフトウェア開発において，ユーザインタフェース設計を行うための参考資料とする．

過去のユーザインタフェース評価研究を見てみると，初期の頃は，目的の1と密接に結び付いていた．たとえば，Card らの打鍵レベルモデルを代表例としてあげられる[1]．

目的の2で使われることが多い手法は，基準となる作業を決め，それらに対し，

- 作業に必要な機能があるかどうか
- 作業の遂行にかかる時間
- 実際に作業をしている時にユーザがした誤りの数

などを評価・測定するという方法である．この方法を研究としてとり上げた例としては，Roberts によるテキストエディタの評価がある[14]．以上の目的1と2のためのユーザインタフェース評価方法は，評価の対象となるシステムが実際にあり，実際に使用するユーザがいるという条件の下で使用され，評価結果と使いやすさとが相関関係をもつかという正確さが重視されていた．

近年注目を集めているのは，目的の3のための評価方法，つまりソフトウェア開発におけるユーザインタフェース評価方法の研究である．これは，グラフィカルユーザインタフェースなどの普及に伴い，ユーザインタフェース設計がソフトウェアの機能の一部，もしくは機能以上に重要な要素として扱われるようになったためである．システムが開発中であるため，目的1や2の評価方法と異なり，設計書などの実際のシステムでない記述を使用し，実際のユーザがいなくてもできる評価方法が提案され，使用されるようになってきた．この場合，ユーザインタフェース設計は，設計開発の費用と使いやすさの両面から評価され，最良の組み合わせを発見することが，ユーザインタフェース設計の重要な技法となりつつある．このような方向の研究としては，ユーザインタフェース評価の対費用効果を重視した，Nielsen らの一連の研究があげられる[12),13)]．Nielsen らは，具体的な数字で対費用効果を計算し，多くの例についてデータを集め，結論として，発見的な評価で効率的な評価ができることを主張している．しかし，評価者の熟練度を一定と仮定するなど，モデル化のための単純化や，見逃しのリスクは高くないと仮定をしていることなどから，実際のソフトウェア開発の様々な条件の下で，彼らの予測通りの費用で評価できるとは限らない．

本研究でとりあげる，複数の評価方法を位置付ける研究は，目的3の評価方法の研究とともに始まった．たとえば，Karat の研究では，実際のシステムを使い，評価者のチームによる系統的評価（Walkthrough）と実際のユーザによる使用テスト（Usability Test）を比較した[3]．この結果は，対象となる2種の評価方法の中から1種だけ選ぶという目的に利用できる．しかし，ソフトウェア開発の各段階の制約に合わせて，複数の評価方法を組み合わせて使用する場合については，まだ検討していない．

2. ソフトウェア開発におけるユーザインタフェース評価方法の評価基準

ソフトウェア開発におけるユーザインタフェース評価方法では，対費用効果が重視されると述べたが，具体的な対費用効果としては，種々の量の比が使われる．たとえば，次のような2種の対費用効果がある．

A 評価にかかる費用と，評価のもたらす情報の質と量

B 評価のもたらす情報によって必要となった設計の変更のための費用と，変更によって使いやすさの増える量

これらの対費用効果が測定できれば，ソフトウェア開発における適切なユーザインタフェース評価方法が分かる．しかし，現実にはソフトウェア開発費用のデータを集めることは難しい，または，公開されないことが多い．また，ユーザインタフェースの場合，原則として最終的には一つの設計しか残らないので，他の設計にした場合に比べてどれくらい良くなったかという評価ができない．つまり，Bの意味での対費用効果の

データを集めることは非常に難しい．

そこで，対費用効果の代わりに，ある評価方法が開発中に使用できるかどうかという評価基準として，次のような評価基準の使用を試みる．

1. 評価の時期
2. 評価で発見できる問題の数と重大度
3. 評価にかかる労力
4. 評価者による個人差

これらを評価基準に選んだ理由は二点あり，一つは，ソフトウェア開発において有用と思われる情報を得られるかどうかである．もう一つは，実際に計量することが可能かどうかである．

2.1 評価の時期

ソフトウェア開発において，開発期間の中の早い時期の設計変更は，遅い時期の変更より，一般に労力が少なくてすむ．そのため，早い時期に使用できる評価方法は，発見された問題点をなくすための労力がより少なくてすむと推定できる．実際にどれくらいの労力が節約できたかを計量することは難しいことが多いが，評価方法が使用できる時期は，評価方法の種類が決まれば一定であることが多い．

2.2 評価で発見できる問題点の数と重大度

ソフトウェア開発における設計評価では，使いやすさの程度を測定するのに加えて，使いやすさを妨げている問題点をどれだけ多く発見できるかということが重要になってくる．また，問題点の数だけでなく，問題点の重大度も重要になってくる．問題点の重大度の分類法は数種あるが，比較的簡単にできる分類法としては，たとえば，次のようなものがる．

1. ユーザが使用を続けられない．
2. ユーザが使用に困難を感じる．
3. ユーザが使用できるが，ユーザが期待した時間より時間がかかる．または，その他の客観的に観察できる尺度の測定値が期待より低い．
4. ユーザが使用できるが，ユーザの主観的評価が低い．

2.3 評価にかかる労力

ソフトウェア開発において設計変更の労力を計量することは，難しいと述べたが，評価自体にかかる労力を評価に必要な作業に関わった人数と時間で計ることにした．たとえば，

1. 評価者または設計者が作業実験をする環境を作る．
2. ユーザ（被験者）が，定められた作業をする．
3. 評価者がユーザの作業記録をとる．
4. 評価者がユーザの作業記録を解析する．

などの作業の人数と時間でみる．

2.4 評価者による個人差

評価方法のいくつかは，評価者の知識や，評価者の主観を必要とする．知識や主観を利用することによって，評価の労力を減らせることもあるが，評価者によって評価結果が異なることになりやすい．評価結果が異なると，評価のやり直しや，評価結果をあまり利用しないことになり，評価結果の利用度が低くなることになる．そのため，評価者による個人差を，評価の有用性の中に含めることにした．評価者の個人差の測定は，厳密には難しいが，ある評価方法を学習したばかりの人と，最も経験のある人を想定し，その2者の差がどれくらいかで判断する．

なお，ここで問題にしているのは，あくまでも評価者の個人差であり，ユーザの個人差のことではない．ユーザの個人差を評価分析するのは重要な課題である．しかし，同じユーザを対象としているのに，評価者によって解釈が異なる場合，ソフトウェア設計をどのように変更すべきかという判断が難しくなる．したがって，評価者によって個人差が大きい評価方法は，ソフトウェア開発には取り入れにくいと言える．

3. ソフトウェア開発におけるユーザインタフェース評価方法の位置付けの枠組

以上4点の有用性の評価基準を要約すると，ソフトウェア開発におけるユーザインタフェース評価方法は，

1. 早い時期に使用できる．
2. 多くの問題点を発見でき，重大な問題点を見落とさない．
3. 労力がかからない．
4. 評価者による個人差が少ない．

であれば，理想的な評価方法であるということができる．したがって，このような性質をもつ評価方法を考え出すことが，評価方法の研究目的の一つであると考えることができる．

しかし，実際には，これらの4点の評価基準を同時に十分に満たすユーザインタフェース評価方法は，考案されていない．そこで，我々はユーザインタフェースの各評価方法を次のような枠組に位置付けることによって，一つの評価方法で，4点の評価基準を同時に満たすことができないことを説明することを試みた．

この枠組では，各評価方法は，次の3個の構成要素

によって位置付ける．

1. 手法
2. 対象
3. ユーザ

3.1 手　　法

まず，評価者の使用する手法の形式性の程度で位置付ける．形式的な手法とは，評価者による個人差が無くなるような形式性を持った方法や規則によって評価をするものである．たとえば，打鍵レベルモデルや，TAG などがある[1),14)]．一方，形式的でない手法とは，発見的な手法を意味し，主観評価法やチェックリスト方式など，評価者の経験や知識を利用して，評価を行う手法を指す．

3.2 対　　象

対象とは，評価者が，評価の対象として使う記述をさす．実際に完成するシステムにどれくらい近いかで位置付ける．実際のシステムを評価対象にするものとしては，作業終了時間や誤り数の測定などがあげられる．実際のシステムを使用しない手法は，システムの仕様記述やマニュアルを使用する．この位置付けの中間的な存在として，プロトタイプがあるが，使い捨て型プロトタイプだと仕様書の側に近く，完成直前のシステムがプロトタイプであれば，実際のシステム側に近いことになる．

3.3 ユ　ー　ザ

3では，実際に使用するユーザにどのくらい近いユーザが評価に参加することが必要かによって位置付ける．実際のユーザが参加するとは，完成したソフトウェアを使用する予定のユーザがシステムを実際に使用して行う評価，たとえば，主観評価や作業終了時間を使用することをさす．実際のユーザが参加しない場合，評価者が仮想的なユーザを仮定して，評価を行うことになる．

よく知られている評価方法が，この評価方法の位置付けの枠組では，どの位置を占めるかを示したものが**表1**と**表2**である．ただし，簡単にするために，どちら側に近いかで位置を2分することにした．通常は同じ名称の評価方法が，この評価方法の位置付けの枠組では，別々の位置におくことになる．たとえば，チェックリスト方式の非常に詳細な方式は形式的な評価方法とみなし，単純化したチェックリスト方式は発見的な方法とした．また，前述したプロトタイプを使用する場合でも，出来上がったプロトタイプを実際のユーザが使うのか，評価者がデザインレビューを行うのか，で位置が異なってくる．

4. 4種の評価方法のケーススタディ

評価方法の位置付けの枠組と有用性の評価基準の関係をさらに詳しく調べるために，4種の評価方法を使用した経験を調べ直すことにした．文献5)，6)では，同一のソフトウェア開発期間中に，次のようなユーザインタフェース評価方法を使用した経験を比較した．

方法a：使い捨て型プロトタイプの作成
方法b：チェックリスト方式
方法c：記憶負荷の机上評価
方法d：形式的な言語プロトコル分析

この比較では，評価の対象となったシステムは，ワークステーション上のウィンドウシステムで動作する文書作成ソフトウェアである．評価方法aおよびbは，初期の外部仕様に対して用いられた．方法cは，数回の仕様の変更の後，実際に開発されたソフトウェアの仕様に対して用いられた．方法dは，方法cで使用した仕様にもとづいたソフトウェアに対して用いられた．実際に完成したソフトウェアは，さらに変更が

表1 ユーザインタフェース評価方法の位置付け—その1
Table 1 Categorization of usability evaluation methods—Part 1.

	対象：仕様	
手法 / ユーザ	形式的	発見的
仮想	打鍵レベルモデル 記憶負荷の机上評価(c') プロトタイプの作成(a) Cognitive Walkthrough	簡易化チェックリスト方式(b) 評価者によるプロトタイプの使用
実際	—	ユーザによるプロトタイプの使用 Cooperative Evaluation

表2 ユーザインタフェース評価方法の位置付け—その2
Table 2 Categorization of usability evaluation methods—Part 2.

	対象：システム	
手法 / ユーザ	形式的	発見的
仮想	詳細なチェックリスト方式	評価者による主観評価 System Walkthrough
実際	作業終了時間 誤り数 形式的なプロトコル分析(d)	ユーザによる主観評価 発見的なプロトコル分析 Questionnaire

加えられている．

4.1 方法a：使い捨て型プロトタイプの作成

方法aでは，仕様書を読むだけではなく，低い捨て型のプロトタイプを作成することによって，仕様書レビューをより正確に行うことを試みた．画面設計の仕様をもとに，プレゼンテーションソフトを利用して画面遷移に関連する入出力を実現した．実現の作業中および作成されたプロトタイプを使用することによって発見された問題点が付録1の1から26である．

4.2 方法b：チェックリスト方式

評価方法bは，仕様書を読む際に，チェックリストを使用して問題点を発見しようと試みた．チェックリストとは，使いやすいユーザインタフェース設計が守るべき規則や，守った方がいい設計規則を箇条書きにしたものを指す．チェックリストによる評価方法では，チェックリストの各項目が守られているかどうかを判定することによって，設計の問題点を見つける．ただし，

- 仕様書をチェックするのか，実際のシステムをチェックするのか．
- 必ず守らなければいけない規則かどうか．

というような点で違いがあり，各種のチェックリスト方式が存在する．大規模なチェックリストとしては，MITRE のものが知られている[9]．また，最近では，各種のグラフィカルユーザインタフェースのスタイルガイドに，推奨する設計スタイルを設計チェックリストとして示すことも多い．

評価方法bでは，Molich らによる単純化されたチェックリスト方式 (Heuristic Evaluation)[10] を採用した．この方式は誰にでも手軽に使用できるように，つぎの9項目だけからなるチェックリストを使用する．

1. 単純で自然な表示内容 (Simple and Natural Dialog)
2. ユーザにわかる言語の使用 (Speak the User's Language)
3. ユーザの記憶負荷の最小化 (Minimize the User's Memory Load)
4. 一貫性 (Be Consistent)
5. フィードバック (Provide Feedback)
6. 分かりやすい取消操作(Provide Clearly Marked Exits)
7. コマンドの省略形を用意する (Provide Shortcuts)
8. 分かりやすいエラーメッセージ (Provide Good Error Messages)
9. エラーを防ぐ (Error Prevention)

チェックリスト方式を形式的手法にしようとしている方式に比べると，Molich の方式の項目数は非常に少ない．しかし，各社から公表されている設計指針[2),7)]に載せられている項目数も十数個止まりなので，種々のアプリケーションに対して使用できるチェックリストの代表例として妥当なものであると考える．

仕様書を検討する際に，このチェックリストの各項目と照合することによって発見できると，我々が考えた問題点は，対象1の問題点の中の( )で囲まれた番号がふられているものである．番号は先述の Molich のチェックリストの項目の番号に対応している．

4.3 方法c：記憶負荷の机上評価

これは，ある作業を行うのに必要な操作ステップを，マニュアルに基づいて定義し，各操作ステップを記憶負荷カテゴリーによって分類することによって，記憶負荷の面から，ユーザが誤りやすい操作を抽出しようという方法である．付録2の項目で，*印がついたものがこの机上評価で発見された問題点に対応している[15]．

4.4 方法d：形式的な言語プロトコル分析

言語プロトコル分析とは，実際のシステムを使ってユーザに典型的な作業を発話 (Think-aloud) しながら行ってもらい，その際のユーザの発話内容（言語プロトコルデータ）を分析して，問題点を抽出する方法である[8]．言語プロトコル分析は，発見的な手法で行うことが多いが，ここで用いた方法は，分析者による分析視点のずれや分析手続きの曖昧さを無くすために，分析視点を予め明確化し，形式的な手順により分析を行う．従って，形式的なプロトコル分析としてみなすことにした[4]．被験者は2名で，計算機の専門家ではないが，ワープロ，キーボードやマウスなどの操作にある程度慣れている人に参加してもらった．付録2の各項目が，この形式的なプロトコル分析で発見された問題点を示す．

表3 使用した4種の評価方法の位置付け
Table 3 Categorization of four usability evaluation methods.

手法 / ユーザ	対象：仕様		対象：システム	
	形式的	発見的	形式的	発見的
仮想	a c′	b	c	
実際	—		d	d′

5. 評価方法の比較

以上の4種の評価方法の使用経験の調査結果をまとめたものが，**表4**である．問題点の数については，具体的な数をあげたが，他の評価については，＋と－を4種の評価方法の中で相対的に比較して判定した結果のみを示した．問題点の重大度については，付録にあげた問題点の一覧から分かるように，4種の中で差があるとは必ずしもいえない結果になったので含めないことにした．

ここで，4種の評価方法を，先ほどの評価方法の位置付けの枠組に当てはめると，aは仕様に対する形式的な評価，bは仕様に対する発見的と見なせる．また，cはマニュアルに対する形式的な評価，dは実際のシステムに対する形式的な評価と見なせる．また，dは，実際のユーザによる評価，他の3つは，仮想的なユーザによる評価とみなせる．したがって，この使用経験の比較では，先述の位置付けの位置を2分して，考えた可能な8通りの組合せのうちの，表3のような場合について調べたことになる．方法cは本来，仕様書に対して用いることができる方法だが，a，bで使用した初期の仕様書ではなく，実際に完成したシステムのマニュアルを利用して行ったので，c′の位置ではなく，cの位置においた．

5.1 手法による差

評価方法aとbは，手法が発見的か形式的かであるところが異なる．方法aとbを比較してみると，aの方が問題点を数多くあげることが可能であると観察できる．これは，機能のもれおよび仕様の曖昧さというような，検出のアルゴリズムが知られている問題点は見落とさないというところに，形式的な手法の強みがあるからだといえる．しかし，発見的手法である方法bによって見つかる可能性の問題点，

- アプリケーション特有のユーザ作業の知識が必要な問題点（Molichのリストの7）
- システムを使ってみないと発見しにくい問題点（5）
- ユーザの主観評価が必要な問題（1，2）

を，方法aは，あまり見つけていない．これは，方法aが，画面設計と画面遷移という点に重点を置いた手法であったためである．そこで，形式的手法は，その手法が想定している範囲内ならば，問題点を数多く見つけられ，範囲外の問題点を見つけにくいと考えられる．作業量としては，方法aを行うには数日かかったが，方法bは数時間もあれば十分であった．仕様書からプロトタイプを作る作業量と，仕様書を読んでチェックする作業量の差，が主になるので，形式的手法の方が作業量が大きいといえる．

5.2 ユーザによる差

方法cとdは，ユーザが仮想的なユーザか実際のユーザであるかという点と，対象がマニュアルであるか実際のシステムであるかという点が異なる．そこで，方法dでは見つかったが，方法cでは見つからなかった問題点13個（付録2）を調べてみると，以下のように分類できる．

- 記憶負荷に関係しないため，方法cでは見つけられなかった問題点
  問題点 7，10
- 実際のユーザであったために発見できた問題点
  問題点 1，8，15，16，17，18，19
- 実際のシステムであったために発見できた問題点
  問題点 3，5，6，20

したがって，実際のユーザを使用したほうが，問題点を数多くあげることが可能であると観察できる．これは，仕様記述や仮想的なユーザだと，評価者や設計者が想定していない問題点を見逃しやすいためと考えられる．また，実際のユーザに参加してもらう場合，実験の準備・実施や教育や記録の整理などに時間がかか

表4 種の評価方法の評価
Table 4 Grading of four usability evaluation methods.

評価方法	a使い捨てプロトタイプ	b単純化されたチェックリスト	c記憶負荷推定	d言語プロトコル分析
評価時期（＋：早い，－：遅い）	＋	＋	＋	－
問題点の数	26	14	7	20
作業量（＋：小，－：大）	－	＋	＋	－
個人差（＋：小，－：大）	＋	－	＋	＋

るので，問題点あたりの労力という点では方法cの方が少なくてすむ．

5.3 対象による差

ソフトウェア開発初期段階では，実際のシステムは存在しないので，したがって，実際のシステムを必要としないユーザインタフェース評価方法の方が早い時期から使用できる．したがって，方法dよりは方法a，b，cの方がより早く使用できる．さらに，仕様書にも様々あり，外部仕様の一部としてユーザインタフェースが記述されている段階では，形式的な手法をとるには不十分な記述しかないことが多い．そのため，より早く使用できるという点から，方法bの方が方法a，cより有利である．

6. 考　　察

以上の比較から，有用性の評価基準と評価方法の位置付けの枠組は，次のような相関関係を持つと確認できた．

1. 評価の時期は，評価の対象を何にするかで定まる．評価方法の位置付けの枠組では，仕様かシステムかで2分しただけにとどまったが，実際には仕様を対象にしても，最初の仕様であるか最終の仕様であるかによって，時期の差が大きくなることがある．
2. 評価で発見できる問題点の数は，手法が形式的であるか，実際のユーザが参加した場合に，多くすることができる．
3. 評価にかかる労力は，手法が発見的であるか，ユーザを仮想した場合に，少なくすることができる．
4. 評価者による個人差は，今回の比較ではあてはまる評価方法がなかったので調べていない．しかし，手法が形式的であれば，個人差は少ないというのは，自明である．たとえば，プロトコル分析を形式的にする試みの例としては，文献4)を参照してほしい．

表5は，以下から推定できる有用性の評価基準と評価方法の位置付けの枠組の関係を示す．この結果から，1種の評価方法では，有用性の評価基準すべてを満たすことはかなり難しいことが推定できる．まず，評価の時期を早めるために，仕様を対象とする評価方法にしたとする．ところが，仕様を対象とした場合，実際のユーザが参加して評価することは難しいので，ユーザを仮想して評価することになり，また初期の仕様の不備を補うために発見的な手法をとることになる．そのため，発見されない問題点のリスクがかなり高くなる．設計初期の段階から実際のユーザに参加してもらう方法として，Participatory Design や Cooperative Evaluation などがあるが，評価および開発の労力のある程度の増加が必要となる[11]．

また，表5から，ソフトウェア開発において，単独の評価方法によるユーザインタフェース評価の使用は非常に難しいことが分かる．たとえば，発見的手法で，仮想ユーザで，対象が仕様という評価方法の場合，評価時期も早く，作業量も少ないが，個人差が大きいため，評価者が熟練していないと発見できる問題点の数が少ないことがある．

したがって，ソフトウェア開発において，複数の評価方法をとりいれることが必要になってくる．上述の考察から，問題点の数と評価の労力は，相反する関係にあることが分かるので，少ない労力でできる評価方法と，問題点をできるだけ数多く発見できる評価方法を組み合わせることが必要なことが推定できる．たとえば，上述のaからdの4種の評価方法から2種だけを選ばなくてはいけない場合，方法bとdというような組合せが最良の組合せだと考えられる．また，最近，グラフィカルユーザインタフェース作成ツールの普及により，より少ない時間と労力で，ユーザインタフェース評価のためのプロトタイプの作成が可能になってきた．このようなプロトタイプを用いて，実際のユーザが開発時期半ばに参加して評価する方法が可能ならば，そのような方法とbの方法の組合せがより良い組合せだと考えられる．

表5 評価方法の位置付けの枠組と有用性の評価基準の推定される関係
Table 5 Estimated grading of each category.

評価方法	手法		ユーザ		対象	
	発見的	形式的	仮想	実際	仕様	システム
評価時期 (+：早い，−：遅い)					+	−
問題点の数 (+：大，−：小)	−	+	−	+		
作業量 (+：小，−：大)	+	−	+	−		
個人差 (+：小，−：大)	−	+				

また，評価方法を名称ではなく，評価方法の位置付けの枠組により，手法・対象・ユーザの3点によって区別することによって，有用性の評価基準適用がよりしやすくなることが4種の評価方法の調査で確認することができた．たとえば，一口にユーザビリティ評価といっても，実際のユーザが使用するのか，発見的な手法をとるのかといった点が異なることが多い．評価方法の位置付けの枠組では，それらを区別することが必要になる．

7. まとめ

ソフトウェア開発における有用性を推定するという観点から，有用性の評価基準と評価方法の位置付けの枠組を提案した．そして，4つのユーザインタフェース評価方法の使用経験から，有用性の評価基準と，評価方法の位置付けの枠組の間に関係があることを推定した．この関係から，どの評価方法を採用するにしろ，ソフトウェア開発では，1種類の評価方法だけでは，十分なユーザインタフェース評価を行うことが困難であることを説明した．また，有用性の評価基準と評価方法の位置付けの枠組の関係から，複数の評価方法を併用する必要があることを示した．

また，これからの評価方法の研究課題として，1種類の評価方法の正確さを求めるだけでなく，評価にかかる労力や，評価の個人差を減らす研究も重要であることが分かる．特に，

- 形式的手法をより手軽に使用できるようにする方法
- 実際のユーザに早くから評価作業に参加してもらう方法
- ソフトウェアツールによる評価作業の効率化

などがあげられる．すでに，プロトタイピングとCASEツールの重要性は認識されつつあるが，ユーザインタフェース評価のためのソフトウェア環境の整備と，それを利用するソフトウェア開発プロセスの研究が重要であると思われる．

参考文献

1) Card, S.K., Moran, T.P. and Newell, A.: *The Psychology of Human-Computer Interaction*, p. 469, Lawrence Erlbaum Associates, New Jersey (1983).
2) IBM Corp.: *SAA Common User Access Advanced Interface Design Guide, SC 26-4582*, IBM Corp. (1989).
3) Karat, C-M., Fiegel, T. and Campbell, R.: Comparison of Empirical Testing and Walkthrough Methods in User Interface Evaluation, *Proceedings of ACM CHI '92 Conference on Human Factors in Computing Systems*, pp. 397-404 (1992).
4) Kinoe, Y.: The VPA Method: A Method for Formal Verbal Protocol Analysis, *Designing and Using Human-Computer Interfaces and Knowledge Based System*, pp. 735-742, Elsevier (1989).
5) 来住伸子，甲 洋介，山本理浩：ソフトウェア開発におけるユーザーインターフェイス評価方法についてのケーススタディ，計測自動制御学会第6回ヒューマンインターフェイスシンポジウム，pp. 425-430 (1990).
6) Kishi, N. and Kinoe, Y.: Assessing Usability Evaluation Methods in a Software Development Process, *Human Aspects in Computing: Design and Use of Interactive Systems and Work with Terminals*, pp. 597-601, Elsevier (1991).
7) Apple Corp.: *Macintosh Human Interface Guideline*, Addison-Wesley (1986).
8) Mack, R.L., Lewis, C.H. and Carroll, J.M.: Learning to Use Word Processors: Problem and Prospects, *ACM Trans. Information Systems*, Vol. 1, No. 3, pp. 254-271 (1983).
9) Smith, S.L. and Mosier, J.N.: *Guidelines for Designing User Interface Software, ESD-TR-86-278*, p. 478, The MITRE Corporation, Massachusetts (1986).
10) Molich, R. and Nielsen, J.: Improving a Human-Computer Dialogue, *Comm. ACM*, Vol. 33, No. 3, pp. 338-348 (1990).
11) Monk, A., Wright P., Haver, J. and Davenport, L.: *Improving Your Human-Computer Interface*, p. 99, Prentice-Hall International (UK) Ltd. (1993).
12) Nielsen, J.: Usability Engineering at a Discount, *Designing and Using Human-Computer Interfaces and Knowledge Based Systems; Proceedings of the Third International Conference on Human-Computer Interaction*, Vol. 2, pp. 394-401, Elsevier (1989).
13) Nielsen, J.: Finding Usability Problems through Heuristic Evaluation, *Proceedings of ACM CHI '92 Conference on Human Factors in Computing Systems*, pp. 373-380 (1992).
14) Roberts, T.L. and Moran, T.P.: Evaluation of Text Editors: Methodology and Empirical Results, *Comm. ACM*, Vol. 26, No. 4, pp. 265-283 (1983).
15) 山本理浩：ユーザーインターフェースの机上比較評価，計測自動制御学会第5回ヒューマンインターフェイスシンポジウム，pp. 67-72 (1989).

付録1 プロトタイプ作成時に見つかった問題点

(番号) がついているものは，チェックリスト方式の対応する番号の規則によって見つかった問題点．

(—)がついているものは見つからなかった問題点．

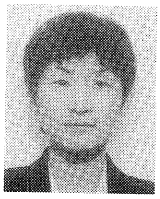
1. (6)取消の後の動作の記述がない
2. (2)ヘルプ機能の画面および操作の記述がない
3. (6)終了機能の画面および操作の記述がない
4. (—)文書スタイルの編集機能の画面および操作の記述がない
5. (—)ホストとの接続機能の画面および操作の記述がない
6. (—)administration utility 機能の画面および操作の記述がない
7. (4)設計指針準拠の画面レイアウトにするための記述が欠けている
8. (—)属性機能のための画面と書式設定変更のためのダイアログボックスの記述が明確さに欠けている
9. (1)ダイアログボックスの画面上での位置が不適当
10. (2)縦書きにおけるタブ機能の記述が不適当
11. (3) text annotation mode において置換しかできないことの表示がない
12. (3)数式入力の時の制限の表示が欠けている
13. (—)シート位置調節の動作の仕様記述に矛盾がある
14. (—)文字カーソルの形の記述が不足している
15. (—)脚注記号の指定の操作の記述がない
16. (—)シート連結の操作の記述がない
17. (—)ファイル名の付け方の説明がない
18. (4)ユーザが項目を選択する方法の記述がない
19. (—)色指定のためのダイアログボックスの記述がない
20. (—)制作プルダウンメニュウで選択できる項目が欠けている
21. (—)属性プルダウンメニュワで選択できる項目が欠けている
22. (8)プリンタが指定されていないときのエラーメッセージがない
23. (1)各機能の操作手順のフローチャートが欠けている
24. (1)数式を削除するときの操作の記述が欠けている
25. (1)重なったオブジェクトを選択するときの注意が欠けている
26. (3)ファンクションキーとシフトキーを使用するときの機能の表示がない

付録2 形式的な言語プロトコル分析で見つかった問題点

(*) が付いた項目は，記憶負荷の机上評価でも見つかった問題点．

1. カーソルが見にくい
2. (*)カーソルが出ない
3. 鉛筆ポインタが出てこない
4. (*)段落のピックがしにくい
5. 文字のピックがしにくい
6. 図形のピックがしにくい
7. 区切りの解除が煩雑である
8. 区切りでカーソルが動かないのがなぜか理解できない
9. (*)段落の区切り方，つなぎ方がわからない
10. フォントの変更がやりにくい
11. (*)段組を変える方法がわからない
12. (*)センタリングの方法がわからない
13. (*)段落の右寄せの方法がわからない
14. (*)キャビネットウィンドウで何も起こらないのでとまどう
15. カーソルを出そうとするが出ないのがなぜか理解できない
16. 文字が入力できないのがなぜか理解できない
17. テキストの搬入ができないのがなぜか理解できない
18. 搬入方法の変更の指定を忘れる
19. ディレクトリの変え方がわからない
20. 図形描画ダイアログボックスが図形を隠す

(平成6年1月28日受付)
(平成6年9月 6 日採録)

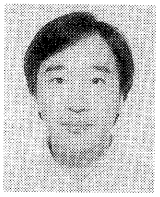


来住　伸子（正会員）

1958 年生．1984 年東京大学大学院工学系研究科情報工学専門課程修士課程修了．工学修士．1984 年より 1992 年まで日本アイ・ビー・エム(株)勤務．1992 年から津田塾大学学芸学部数学科勤務．主な研究テーマは，ソフトウェア工学，特にユーザインタフェース設計および評価．

甲　　洋介（正会員）

1986 年慶應義塾大学大学院管理工学専攻修士課程修了．同年日本アイ・ビー・エム(株)東京基礎研究所入社．認知工学研究グループを経て現職．人間の認識の情報処理過程，類推過程，ヒューマンインタフェース，協調的発想支援システムに興味を持つ．ACM (SIGCHI), IEEE, 人工知能学会，人間工学会など各会員．



山本　理浩

1957 年生．1983 年北海道大学文学部行動科学科卒業．1984 年日本アイ・ビー・エム(株)入社．現在，アジア・パシフィック製品統括本部にてソフトウェア製品の品質保証業務に従事．認知心理，ソフトウェアの人間的要因について興味をもつ．