ヒューマンインターフェース 100－14
（２００２．９．２１）

拡張ステートチャートを用いたGUIユーザビリティ自動評価システム

橋本篤史　倉本到　渋谷 雄　辻野 嘉宏
京都工芸繊維大学

良好な GUI を１回の開発サイクルで開発することは困難である．そのため，GUI においては設計，プロトタイプの作成，使いやすさのテストというサイクルを順に繰り返すという手法が活用されることが多い．ところが，近年では GUI の大規模化により開発の手間を削減することが重要となっている．本研究では，GUI の仕様からユーザビリティを自動評価することにより，テストに必要な手間を削減することを目的とする．まず，GUI の動作を主とした GUI 全体の仕様記述法としてステートチャートを拡張した拡張ステートチャートを考案した．そして，それを元にした GUI 仕様の作成支援ツールと，仕様によりユーザビリティを自動評価するツールを実装し，これらを結合した GUI ユーザビリティ自動評価システムを試作した．

An Automatic Evaluation System for GUI Usability Using Extended Statechart

Atsushi Hashimoto,　Itaru Kuramoto,　Yu Shibuya and Yoshihiro Tsujino

Recently, Graphical User Interface (GUI) has been used widely. GUI usability is unclear at early design stage so that it's difficult to design good GUI without trial and error. Therefore, we often develop GUI using the spiral method, that is, repeating following three stages: designing, prototyping, and testing. In this paper, we try to reduce the cost of evaluating GUI usability by automatic testing based on a formal GUI specification in the extended Statechart. We present an automatic GUI evaluating system, which consists of the extended Statechart editor and some simple evaluators. Each evaluator checks the specific GUI usability factor automatically or semi-automatically based on the GUI specification.

1．はじめに

近年，計算機の能力が向上してきたことにより，高度な処理をすばやく行うことが可能になった．そのため，多くのアプリケーションにおいて直感的でわかり易い GUI が使われるようになった．

一般的なソフトウェアの開発手法は，要求定義，設計，プログラミング，テストの順に行われる[1]が，GUI は“使いやすさ”など設計の段階では予測しにくい性質を持っているため一度の設計で満足のいく GUI を得るのは難しい．そこで，GUI の開発には 設計，プロトタイプの作成，テストを繰り返し行う手法が用いられる．

しかし，近年ではアプリケーションの規模が大きくなってきているため, GUI を開発する手間を減らすことが重要な課題となっている．従来の研究では， ステートチャート[2]と呼ばれる状態遷移図を拡張した記述法を用いて GUI 仕様を記述し， その記述内容から“ 使いやすさ ”の問題点を発見できることが示されている[3].

本研究では, GUI アプリケーションのユーザビリティ評価のいくつかを自動的に行うことで，評価における手間を削減することを試みる．以下ではまずデザイン原則とその評価方法について述べ, 次に GUI ユーザビリティ評価を行うために用いるステートチャートの記述方法の拡張について述べ，その記述法に従った仕様からユーザビリティ評価を自動的に行うシステムの実装について述べる．

2. GUI ユーザビリティ評価の自動化

本研究ではユーザインタフェース評価手法としてヒューリスティクス評価 [4] を用い, GUI 仕様の自動化を目指す．本章ではヒューリスティック評価のためのデザイン原則として 10 個の項目を挙げ[5]， それらを自動評価する方法を検討する．

評価をするために，言葉の意味を理解することが必要な場合があるが，それはセマンティクスに関わることであり，現段階では自動化が困難である. 本研究で自動化を試みたのは 10 個あるデザイン原則のうち ①から⑤の５つの項目である．

2 デザイン原則

①シンプルで自然な対話

ユーザがしたいと思っているタスクの流れに逆らうようなインタフェースは自然ではない. また, 不要な情報を要求することや，構造が複雑なインタフェースは避けるべきである．

本研究では,

- ユーザが何らかの目的を達成する際に， 操作回数が少ないほど対話がシンプルである．
- ある１つの状態において， そこから操作可能な方法が少ないほど対話がシンプルである．

と考える．従って，この評価を自動化するためにはユーザのいる現状態からどのような操作が可能であるのか， また, 目的の状態に到達するまでの可能な操作列と操作回数がわかればよい.

評価基準を設定し,

(現在の状態から目的の状態に到達するまでの操作回数)＜(操作回数の評価基準)

(現在の状態から操作可能なものの数)＜(操作可能なものの数の評価基準)

を満たしていないものを検出することで評価を実現することが可能である．ただし， ユーザ層によりその評価基準が変化すると考えられるため，初心者ユーザと熟練ユーザで評価基準を変更する必要があると考えられる．

②ユーザの言葉で話す

ユーザはコンピュータの専門家ではないが，本来するべきタスクの分野では専門家である．そして，専門用語は複雑な概念を一言で同僚の専門家に伝えるために必要なものである．

本研究では，システムに用いられている言葉がユーザの言葉としてふさわしいかどうか３段階評価（“ 言葉の意味と実際の操作が理解できる ”,“ 言葉の意味はわかるが実際の操作と一致しない ”, “ 言葉の意味が理解できない ”の３段階）でユーザ自身に判定してもらうことを考え，このアンケートの作成，集計を自動で行う．これを実現するためには，各ウィンドウ内で使われている言葉の情報が必要であると考えられる．

③一貫性

一貫性とは「どんな場面でも同じ事をするには同じようにすればいい」ということである．それは，操作手順やシステムの機能に関してだけでなく，同じ情報はどの画面やダイアログボックスでも同じ場所に表示するという位置の一貫性や，使用しているオブジェクトの大きさ，色などの一貫性も挙げられる．そうすることによって，初めて

使うソフトウェアでも簡単に使用することができる．

本研究では次のような一貫性を検証する．

・各ウィンドウに存在するボタンなどの位置や大きさの一貫性
・同じラベルを付加してあるボタンは同じ操作結果をもたらすべきであるという操作の一貫性

これらの一貫性を検証するためには各ウィンドウに存在するオブジェクトの位置や大きさの情報や，オブジェクトの種類の情報，ラベルの情報が必要となる．

評価方法として，１つのウィンドウ内に存在する同じ種類あるいは同じ目的を持ったオブジェクトの大きさや色，位置が一貫しているのかどうかを検証し，違っている場合は通知する．また，あるウィンドウに存在するオブジェクトと同じ操作を行うことができる別のウィンドウに存在するオブジェクトの位置や大きさ，ラベルの情報などが一貫しているのかどうかを検証し，違っている場合は通知する．

④出口を明らかにする

ユーザが選んだボタンやメニュー項目の機能はそのときには使いたくないものであるかもしれず，何もしないままで抜け出す手段が必要になる．一般的に使用されているものとして，最初に戻る機能(WWW ブラウザにおけるホームボタンのようなもの)，直前の操作を取り消す機能(undo)，キャンセル機能などがある．

本研究では，undo の検出を行う．そのためには，ある操作を行った際に，システム側の動作が undo なのかどうかという情報が必要である．つまり，今いる状態を a，状態 a からある操作で遷移した状態を b，a の前の状態を c とすると，状態 b と状態 c が同じであるかどうかを比較し，同じであればそれを undo と判断する．

⑤ショートカット

ユーザはいつまでも初心者であるわけではなく，毎回同じメニュー階層をたどって実行したいメニュー項目を発見するのは時間がかかる．よって，ショートカットを作成することによって経験をつんだユーザが，頻繁に実行する操作を，素早く，効率的に行うことが可能となる．

本研究における評価方法として，今いる状態から目的の状態までの操作手順を検証し，より少ない操作（理想は１回）で行えるものをショートカットと判断し，ショートカット操作が存在することを通知する．

⑥ユーザの記憶負荷を最小限にする

何の補助もなしに何かを再現するよりは，提示された情報を再認する方が容易である．また，システムの機能に新しい操作を加える場合には，以前から用いられている操作方法で実現することにより，操作方法を新たに学習する必要がなくなるので，ユーザの記憶負荷を軽減することができる．

ユーザの行える操作が画面上に全て現れているのかを検証することによって評価することが可能だと考えられる．

⑦フィードバック

システムはユーザに，現在システムが何をしているのか，ユーザの入力がどう解釈されているのか，などのフィードバックが必要である．フィードバックがないとユーザは不安に陥る可能性があるからである．よって，フィードバックは抽象的に行うのではなく，ユーザの行った動作を的確に表現するようにしなければならない．また，画面を大きく書き換えてしまうようなフィードバックはユーザが直前の操作を忘れてしまう可能性があるので避けるべきである．

評価方法としては全ての状態に対して，音を鳴らす，ユーザの操作に対する現在のシステムの状態を表示するなどのフィードバックがシステムから起こっているのかを検証し，フィードバックがないものを検出し，それを通知すればよいと考えられる．

⑧適切なエラーメッセージ

エラーはユーザビリティにとって２つの理由で重要である．まず，システムが何らかの問題に直面しており，ユーザの目的を達成できない状態であることを知らせるという点である．次に，エラーメッセージは，その問題の予想される原因や対策などの関連知識を提供するので，ユーザにシステムに対する新たな知識を得る機会を与えるという点である．

エラーメッセージの内容が適切でないと，ユーザはエラーの原因がわからず，エラーから復旧することができないため，また同じエラーを引き起こす可能性がある．

エラーメッセージの内容が適切であるかどうかを検証するためには，そのメッセージの意味が適切にユーザに伝わるのか，またそのエラーに対する正しいメッセージなのかという言葉の意味を検証する必要がある．

⑨エラーを防ぐ

頻繁に使用するボタンの近くに危険なボタンを配置したり，緊急時のボタンが緑であったり，数字限定など制限された入力を要求するダイアログボックスには入力例や説明がないとユーザがエラーを起こす可能性が高くなる．このようなエラーを起こしやすい条件を見つけ，ユーザがそのような状態に陥ることを避けるようにシステムをデザインすることが重要である．

評価方法としては入力エラーを防ぐために入力例が記入されているかを検出するという方法が挙げられる．

⑩ヘルプとドキュメンテーション

ユーザインタフェースの使用にヘルプやドキュメンテーションが必要ないシステムが理想ではあるが，実際にはユーザが，いつでもヘルプやドキュメンテーションを参照できることが必要である．

評価方法として，ヘルプがいつでも閲覧可能であるかといった検証に加え，そのヘルプがユーザに適切なものであるのか，また，実際に行いたい操作のヘルプと一致しているのかを検証する．

3. ステートチャートの拡張

3. 1 ステートチャート

従来の状態遷移図では以下のような問題点があることが知られている[6]．

・状態遷移図は平板であり，階層やモジュールの概念がないので，段階的な記述作成には向かない．
・複数の状態における同じイベントを効率良く書くことができない．
・独立にとらえることのできるイベントの発生順を考慮する必要があるので，状態数が爆発的に多くなってしまう．
・状態遷移図は本来逐次的記述しかできないため同時性を扱うことができない．

これらの問題点を解決するためにステートチャート[3]では以下のような拡張が行われた．

・複数の状態を括った拡大状態の導入：関連する機能を持つ状態をくくることでモジュール化でき，ボトムアップに記述できる．あるいは，ある機能を持つ状態をその部分機能を持つ部分状態に詳細化することでトップダウンに記述できる．また，複数の状態から同じイベントによって同じ状態に遷移する場合，それらの状態を括った拡大状態からの１つの遷移として書くことによって遷移の数を減らすことができる．
・状態をそれぞれ独立に動作する直交状態に分解する機能：一つの状態において，独立に機能していると考えられるものを部分状態として独立に書くことにより，状態数が指数的に増加するのを防ぐことができ，直感的にも理解しやすくなる．
・状態の独立成分間の通信機構：これにより独立成分が場合によっては同期を取ることも可能になる．

3. 2拡張ステートチャート

3.1 に述べたことから,GUI の動作の仕様を記述するのにはステートチャートを用いるのが適していると思われるが，本研究において，自動評価を行うためにはオブジェクトの位置，種類，色，ラベルの情報など，より多くの情報を必要とする．したがって，前章で述べたような評価を行うためにステートチャートを拡張する必要がある.また，GUI とステートチャートとの対応を決める必要がある.

3. 2. 1状態記述の統一と拡張

GUI をステートチャートで記述する際には，どの部分を１つの状態として考えるかを決める必要がある．本研究では，アクティブなウィンドウやダイアログボックスを１つの状態と考える.また，ウィンドウやダイアログボックスを拡大状態と捉え，ウィンドウやダイアログボックス内のオブジェクト情報に変化が生じた場合は部分状態の遷移が起こったと考える．従ってウィンドウやダイアログボックス内に表示されている操作可能なオブジェクトも１つの部分状態であると考えることができる.

また,“②ユーザの言葉で話す”や“③一貫性”を評価する際に必要と考えられるウィンドウやダイアログボックス内に存在するオブジェクトの種類や位置，大きさ，ラベルの情報などはその拡大状態であるウィンドウやダイアログボックスが持つこととする.

3. 2. 2遷移記述の統一

GUI で行われるユーザの操作によるイベントはおおまかに分けると，マウスによる操作とキーボードによる操作の２つである.

マウス操作にはシングルクリック(右クリック，左クリック)，ダブルクリック，ドラッグ&ドロップの操作がある．そこでマウス操作によるイベントの記述を次のようにする.

- 左クリック(対象となるオブジェクト名)
- 右クリック(対象となるオブジェクト名)
- 左ダブルクリック(対象となるオブジェクト名)
- 右ダブルクリック(対象となるオブジェクト名)
- ドラッグ&ドロップ(対象となるオブジェクト名，ドロップする場所)

また，キーボード操作の場合は押すキーの名前をイベントとして記述する．ただし複数のキーを同時に押す場合は‘+’でつなぐことで表現する.

3. 2. 3特殊な記述

テキストボックスに入力を行う場合のように，ユーザがどのような入力を行うかが自由な場合には，入力ごとに状態を変化させると状態数が爆発的に増加する．そのような場合に拡張ステートチャートでは初期の状態と入力途中(入力完了も含む)の状態の２状態だけを考える記述をすることで状態爆発を回避する．この場合の操作の取り消し(undo)記述は，入力途中の状態から自分自身に戻ってくる自己閉ループの矢印を用いて記述する(図１)．またその際，遷移のラベルのアクションには undo と記述することとする.

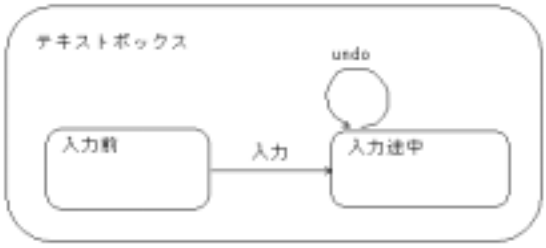

図１　特殊な記述の例

3. 2. 4 記述例

上記のことを踏まえて図2のプリンタ追加ウィザードのダイアログウィンドウを拡張ステートチャートで記述すると，図 3 のようになる．3.2.1 の記述法に従い,「ローカルプリンタ」と「ネットワークプリンタ」の選択は,「ラジオボタン」という状態における部分状態の遷移である．また，ラジオボタン,「戻る」,「次へ」,「キャンセル」ボタ

ンはそれぞれ「プリンタの追加ウィザード」という状態の部分状態である.

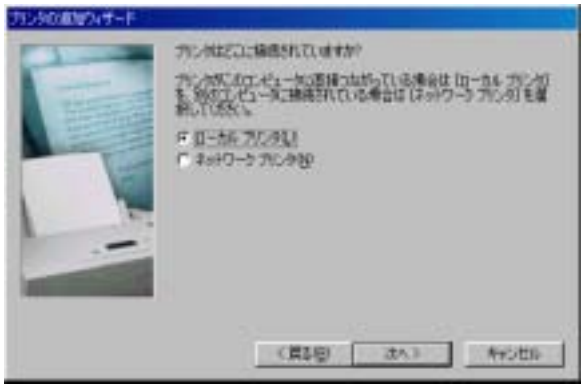

図2　プリンタ追加ウィザード

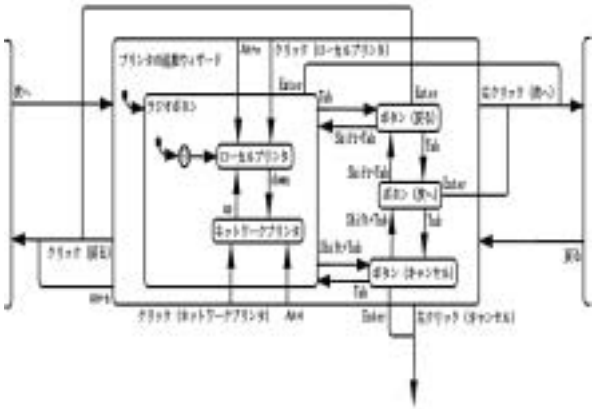

図3　拡張ステートチャートの記述例

4. GUI ユーザビリティ自動評価システム

4. 1 システム概要

本研究で作成した GUI ユーザビリティ自動評価システムは大まかに 2 つの部分に分けられる．1つは拡張ステートチャートを用いて GUI 仕様の作成を支援する部分(拡張ステートチャート作成支援ツール，オブジェクトレイアウト作成ツール)であり，もう 1 つは作成された GUI 仕様から先に述べたガイドラインに従ってユーザビリティを自動評価する部分である．まず拡張ステートチャート作成支援ツールで評価対象となるシステムの状態遷移を記述し，オブジェクトレイアウト作成ツールでレイアウトを作成し，評価に必要な情報を入力して評価を行う.

4. 2GUI 仕様作作成支援ツール

ツール上にあるキャンパスに，実際に拡張ステートチャートを用いて GUI 仕様の記述をし，ユーザビリティ自動評価を行うためのデータ構造を作成する． このツールは 2 つの部分からなる．1つは拡張ステートチャート自身を描く，拡張ステートチャート作成支援ツール (図 4)であり， もう 1 つはウィンドウやダイアログボックスの中に存在するオブジェクトがどのように配置されているのか，またそのオブジェクトの色や大きさを描く，オブジェクトレイアウト作成ツール(図 5)である.

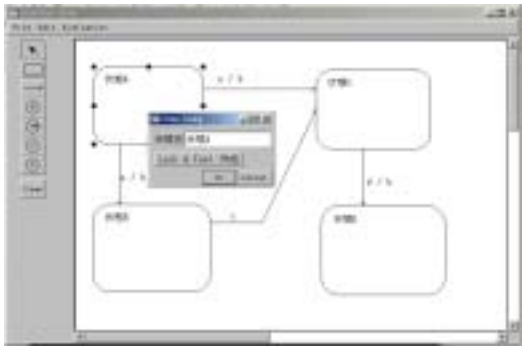

図4　拡張ステートチャート作成支援ツール

図5 オブジェクトレイアウト作成ツール

4. 3GUI 自動評価ツールとシステム使用例

4. 3. 1 シンプルで自然な対話

まず評価を行う前にダイアログボックスで評価基準の設定を行う. ここでは操作回数を 3 とする. 評価を実行すると結果は図 6 のようになる．これは初期状態から状態Ｃに遷移するのに操作回数が 3 以上のものを太線で示している.

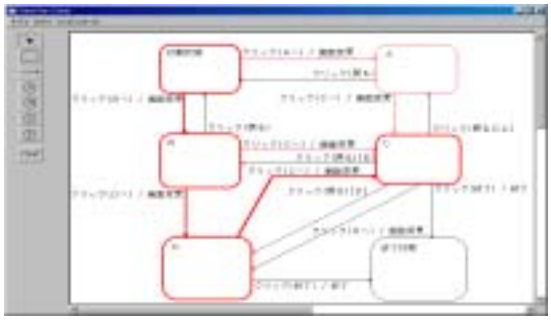

図 6 “ シンプルで自然な対話 ” の評価結果

4. 3. 2 ユーザの言葉で話す

オブジェクトレイアウト作成ツールでオブジェクトのレイアウトを作成することによって，オブジェクト内の文字情報を得ることができ，評価を行うことが可能である．拡張ステートチャート作成ツール内の,「Evaluation」ボタンをクリックすることで，評価用ダイアログボックスが自動作成され，評価者はユーザの理解度を調査することができる．図 7 は例として「初期状態」,「状態A」,「状態B」,「状態C」の「戻る」という文字の意味がわかり易いかどうかを評価している．ここで評価対象のうち１つでも言葉の意味があいまいであるとユーザが評価すると，コメントが出力される(図 8)とともに図 9 のように修正しなければならないものを含む状態が強調表示される．

図 7　アンケートダイアログボックス

図 8　コメント出力の例

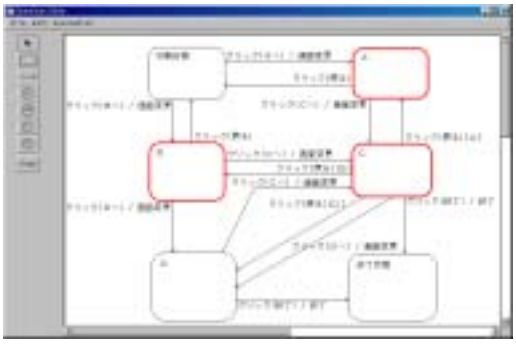

図 9 “ ユーザの言葉で話す ” の評価結果

4. 3. 3 一貫性

オブジェクトレイアウト作成ツールで入力されたデータをもとに，一貫性に欠けているものが存在すれば保持されていない状態をコメントとして出力し, また問題のある状態を強調表示する(図 10).

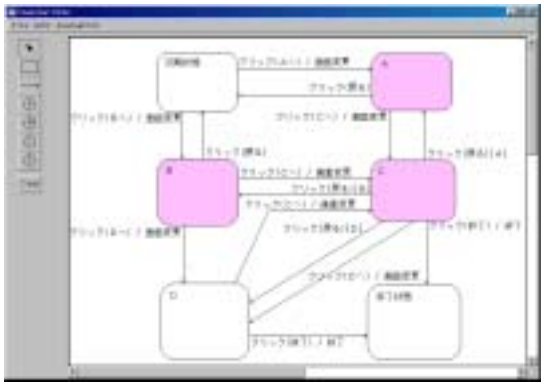

図 10 “ 一貫性 ” の評価結果

4. 3. 4 出口を明らかにする

出口のないものが存在した場合，どの状態に出口が存在しないのかを知らせ，問題のある状態を強調表示する. 図 11 では「初期状態」,「状態D」,「終了状態」が強調表示されている.

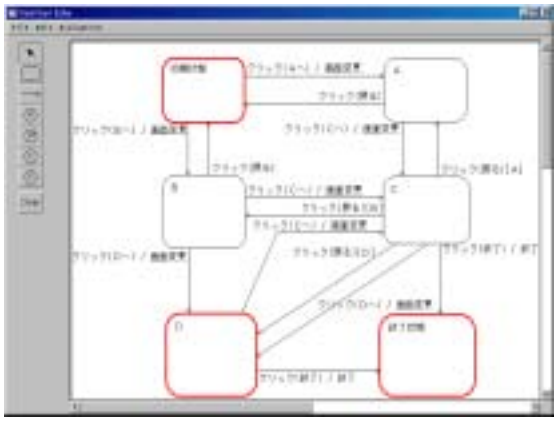

図 11“出口を明らかにする”の評価結果

4. 3. 5 ショートカット

ダイアログボックスでショートカットを探索する状態を選択すると，最も少ないステップ数のものを強調表示する(図 12).

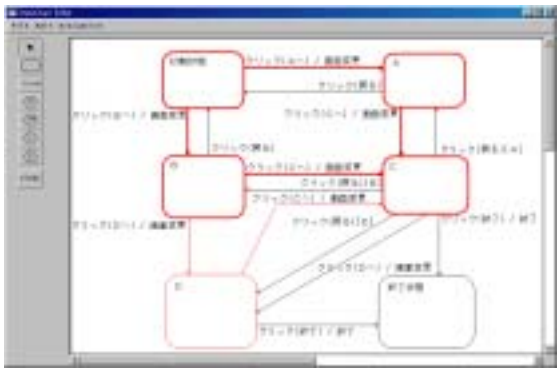

図 12“ショートカット”の評価結果

5. おわりに

本研究ではユーザビリティ評価を自動で行うための GUI 仕様を記述するためにステートチャートの記述方法を拡張し，拡張ステートチャートで書かれた GUI 仕様のユーザビリティを実際に自動的に評価するシステムの試作を行った．また実際にこのシステムを用いて GUI 仕様を拡張ステートチャートで描き，実際にその仕様のいくつかのユーザビリティ評価を自動的に行えることを確認した.

今後の課題としては，現段階で保留されている他のユーザビリティ評価項目の自動化を検討し，今回作成した自動ユーザビリティ評価システムに追加していく必要がある.

参考文献

[1] 藤野， 花田：“ソフトウェア生産技術”， 電子情報通信学会 (1985)

[2] Harel D. :“Statecharts: A visual formalism for complex systems”， Science of Computer Programming， Vol. 8, No3， pp. 231~274 (1987)

[3] 深谷美登里：“状態遷移図とログ分析による Web サイトのユーザビリティ評価”， ヒューマンインタフェースシンポジウム'99 論文集， pp. 525~530 (1999)

[4] Jakob Nielsen：“Usability Engineering”, Academic Press (1993)

[5] 田村博：“ヒューマンインタフェース”， オーム社 (1998)

[6] Harel D．：“On visual formalisms”, Communication of the ACM， Vol. 31, No5, pp. 514~530 (1988)