「マルチメディア，分散，協調とモバイル
(DICOMO2014) シンポジウム」 平成26年7月

集約情報を用いた通知ダイアログへの対応支援手法

上島 愛史[1]　梶 克彦[2]　河口 信夫[2]　竹中 光[3]　武内 重樹[3]　岡本 学[3]

概要：
計算機を操作する過程では，様々な種類の通知ダイアログが表示される．それらはわかりにくい表現や専門用語で情報を提示していたり，ユーザに複数の操作選択を強いたりする場合が多い．そこで本研究ではダイアログに対応する操作案内情報を，ユーザに対して自動提示する手法を提案する．あらかじめ，サンプルデータ提供者の計算機で表示されたダイアログの内容や，そのダイアログに対する操作情報を自動的にサーバに自動集約する．支援対象者の計算機においてダイアログが発生した時には，サーバ側でダイアログの同定を行い，該当するダイアログの情報を受け取り，案内情報を自動生成して提示する．案内情報の内容は，ダイアログの遷移，ボタン押下等の統計，人手により付与された説明文である．ダイアログ情報の収集システム及び，案内情報生成システムを実装し，案内情報提示の有効性を実験により検証した．実験の結果，案内情報の提示により，ダイアログ操作の試行回数の減少，操作時の安心感・信頼感・作業の容易さの向上が確認された．

A Response Supporting Method for Notification Dialog Based on Crowdsourcing

Aiji Uejima[1]　Katsuhiko Kaji[2]　Nobuo Kawaguchi[2]　Hikaru Takenaka[3]　Shigeki Takeuchi[3]
Manabu Okamoto[3]

1. はじめに

計算機を操作する過程では，様々な種類の通知ダイアログ（以下ダイアログ）が表示される．それらは難解な表現や専門用語で情報を提示していたり，ユーザに複数の操作選択を強いる場合が多い．また，それらのダイアログが表示されている間は，他の操作を受け付けないことが多いために，ユーザはその対応を煩雑だと感じることが多くある．特に普段から計算機に接していない，いわゆるパソコン初心者にとっては，ダイアログの操作は難易度の高いものである．

これまでにも，ダイアログ操作に関する研究として，ユーザに対して親和性が高く，ストレスレスなダイアログ表示方法等の研究 が行われている [1], [2]．これらの研究では，ダイアログ自体をよりユーザビリティの高いものに設計し，操作性の問題を解決している．しかし，ユーザビリティの高いダイアログの開発は，開発者の意図やシステム環境に大きく依存しているために，アプリケーションごとの表示形態の統一は困難である．

難解なダイアログや，多フェーズにわたるダイアログを通じた複雑な作業を行う場合，IT リテラシの高いユーザであれば，ブラウザ等で手順書などを表示させて作業を行うのが一般的である．手順書は，アプリケーションベンダーのウェブサイトに公式で用意されていたり，有志のユーザが各自のウェブサイトで解説している事が多い．図 1 は，iTunes のインストール方法を示した手順書である [3]．一般ユーザは，ダイアログのタイトル等のテキストをキーワードにし，ウェブサイトを検索して調査する．しかし，初心者の中には，ダイアログの情報をインターネットで検索するという行為に，馴染みのないユーザも多く，また，有益な情報が得られない場合もある．

[1] 名古屋大学工学部
School of Engineering, Nagoya University
[2] 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University
[3] 日本電信電話株式会社 NTT サービスエボリューション研究所
Nippon Telegraph and Telephone Corporation, NTT Service Evolution Laboratories

図 1 手順書の一例

本研究では，ユーザビリティの高いダイアログを設計開発するのではなく，既存のダイアログに対して操作案内情報を生成して提示する手法を提案する．あらかじめ，サンプルデータ提供者の計算機において表示されるダイアログの内容や操作情報をサーバに自動集約しておく．支援対象者の計算機においてダイアログが発生した際には，サーバのデータに基づき，HTML 形式の案内情報を自動生成して提示する．この手法により，ユーザのストレスの軽減や作業効率の向上，及びこれまでの研究の問題を解決する．

2. 関連研究

これまで，ダイアログの表示形態に関して，Maglio ら [4] は，ユーザが行っている作業に対する妨害を最小限にし，認知しやすい表示方法を模索する研究を行い，下記のガイドラインを定めている．

(1) 表示されるテキストの動きは最低限にする．
(2) テキストに動きを用いる場合は，それは常に動き続けるべきではない．
(3) 動きがあり，ダイアログ内でテキストが一旦停止する表示方法が理想的である．
(4) 視覚的な情報表示に関しては，ユーザの聴覚よりも視覚に対して注意を喚起する方が良い．
(5) テキストがスクロールする方向は問題ではない．

McCrickard ら [5] は，スクロールやフェード等のアニメーションを用いた，テキスト表示方法についての研究を行い，Somervell ら [6] はそれに加えて，目盛りのようなグラフィカルな表示方法との比較を行った．その結果，テキストによる表示と，グラフィカルな表示方法と間には，ユーザのタスク処理効率に及ぼす影響に，差は無いということを明らかにした．

Bartram ら [7] はアイコンを用いて，それらの色，形の変化，動きなどによる表示方法についての研究を行った．その結果，ユーザの認知率，認識時間は，動きによる提示が最も良いということを示した．

Czerwinski ら [8] は，ダイアログの表示のタイミングの違いによる，ユーザの認知についての研究を行い，メインタスクの開始時に情報を提示するのが良いということを示した．

三好ら [1] はこれらの研究を踏まえ，実験により以下のガイドラインを定めた．

(1) 気づきやすさと，作業の妨害の小ささのバランスを考慮する場合，静的ウィンドウでのテキストのアニメーションを用いる場合が効果的である．
(2) テキストのスクロールは水平方向よりも垂直方向に動く場合が，気づかせやすさと妨害の小ささのバランスが良い．
(3) ハイライトやビープ音のようなユーザの注意をひく要素は，テキストが可読になるタイミングで発生させても，ユーザの作業効率への影響は少ない．

このように，ユーザビリティの高いダイアログの表示方法の研究が進められており，実験を通して，具体的な手法が示されている．しかしながら，ダイアログの表示形態は開発者の意図，及び動作環境に大きく依存している．そのために実際問題，各アプリケーション毎でのダイアログの表示形態の統一及び移行は，開発コストが高く困難である．

3. ダイアログ操作案内情報提示システム

2 章において述べたように，ユーザビリティの高いダイアログの開発は，完成されている既存のアプリケーションのダイアログへの適用が困難であり，各アプリケーション毎での統一が課題となる．これらの問題を解決し，ダイアログの操作に対する抵抗感や煩雑感を緩和させるためには，デザインや UI の改良ではなく，図 1 のような操作案内情報を，ダイアログ本体とは別のウィンドウで提示する手法が最適であると考えられる．また，必要となる操作案内情報の作成コストを緩和させるために，多数ユーザの実際のダイアログの選択操作に基づいて，自動的に操作案内情報を生成する必要がある．さらに，収集された情報をサーバで最適な形式に編集して管理し，案内情報に変換する必要がある．これらの手法を総合し，ダイアログに対する案内情報提供システムとする．図 2 に本システムの構成を示す．本システムは大きく 3 つの要素で構成されている．

- サンプルデータ提供クライアント
- データ管理サーバ
- 支援対象クライアント

3.1 サンプルデータ提供クライアント

本手法ではまず，複数のサンプルデータ提供者のクライアントからダイアログと操作の情報を収集する．複数ユーザからダイアログの情報を収集し，同一のダイアログに対する操作方法のユーザ間の操作差異を，最終的に統計情報として支援対象ユーザに提示することを目的としており，また，異なる操作によるダイアログの遷移の違いの情報を

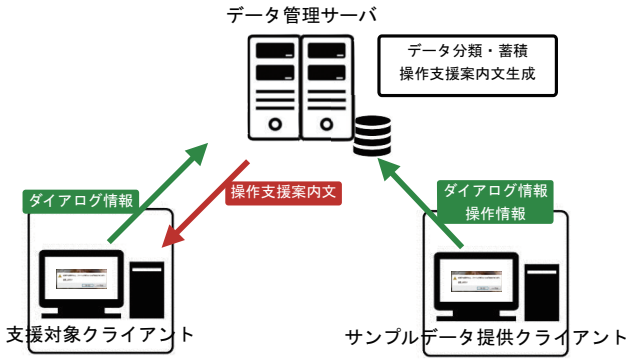


図 2 本研究システムの構成図

カバーする．

サプルデータ提供クライアントは，計算機の操作に慣れており，ダイアログ操作に関する情報提供をしてもよいと同意したユーザの計算機にインストールすることを想定している．サンプルデータ提供クライアントは計算機上で常駐し，ダイアログの発生を検知し，ダイアログ内のテキストやボタン等のオブジェクトを解析して，その情報を JSON データに収める．そして，ユーザのダイアログに対する操作を監視し，選択されたボタン等の情報を同一の JSON データ内に収める．また，ダイアログ本体のビットマップデータを圧縮し，JSON データとともにサーバに送信する．これらの処理はすべて自動的に行われる．

3.1.1 ダイアログ出現の検出

ダイアログ情報の収集を自動的に行うために，まず，ダイアログの出現を検出を行う必要がある．Windows 上ではダイアログはウィンドウの一種として管理されており，全てのダイアログに共通して *#32770* というクラス名が振られている．よって，ダイアログの出現を検出するには，全ウィンドウの生成を監視し，各ウィンドウのクラス名からダイアログのみを判定する手法が最適であると考えられる．

この手法を実装するために WindowsAPI[*1] を用いる．デスクトップ上のウィンドウを列挙する関数, EnumWindows 関数[*2]を用いて，更に，ウィンドウのクラス名取得する関数，GetClassName 関数[*3]でクラス:*#32770* のみをフィルタリングする．

3.1.2 ダイアログの含有情報の取得

ここでは，ダイアログを生成したアプリケーションの情報，及びダイアログ内に表示されているテキストやボタン等のオブジェクトの情報と，ダイアログ本体のビットマップデータをダイアログの含有情報と呼ぶ．後にダイアログの同一判定を行う際に，アプリケーションの情報が必要であり，ダイアログ内のオブジェクトの情報，及びビットマップデータが案内情報生成の元データとなる．また，オブジェクトの情報として用いるのは，

- オブジェクト内に表示されているテキスト
- オブジェクトの配置座標
- オブジェクトのクラス名

の 3 つである．

Windows 上では，ダイアログ内の各オブジェクトも，一種のウィンドウとして管理されている．更に，ウィンドウはウィンドウハンドル (以下ハンドル) と呼ばれる番号で管理されている．つまり，ダイアログ内の各オブジェクトは，ハンドルと特定のクラス名を保有しており，そのハンドルはダイアログが有するハンドルの子ハンドルという位置づけにある．ダイアログ含有情報を取得するには，これらのハンドルを利用するのが最適である．

ダイアログを生成したアプリケーションは，親ウィンドウのハンドルを取得する関数，GetAncestor 関数[*4]を用いて，ダイアログのハンドルの親子関係を辿ることより特定し，ウィンドウのタイトルバーのテキストを取得する関数，GetWindowText 関数[*5]でプロセス名を取得する．

オブジェクトの情報も同様にハンドルを利用し，WindowsAPI を用いて取得する．オブジェクト内に表示されているテキストは GetWindowText 関数, オブジェクトの配置座標はウィンドウの座標を取得する関数，GetWindowRect 関数[*6]，オブジェクトのクラス名は GetClassName 関数により取得する．ただし，GetWindowRect 関数で取得されるのは，デスクトップ全体を基準とする座標であるため，ダイアログ本体のウィンドウ領域を基準とした座標に計算し直す必要がある．

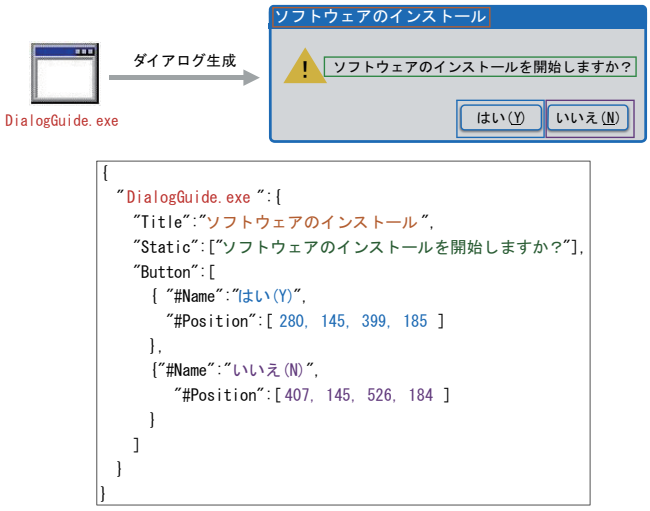
これらの情報をダイアログ毎でひとつのデータに纏める必要がある．各オブジェクトの親子関係をデータ内で表現し，案内情報生成時にパースしやすいデータ形式を採用するために，本手法では JSON データ形式で情報を纏める．図 3 はその一例である．

ダイアログのビットマップデータは，GetWindowRect 関数を用いて座標を特定してキャプチャし，JPEG 形式に圧縮する．この JSON データと画像データには，ダイアログの生成時刻をタイトルに付けて管理する．また，これら 2 つのデータはダイアログが閉じられたタイミングでサーバに送信する．

3.1.3 ユーザ操作とダイアログの遷移情報の取得

支援対象クライアントに対して最適な操作案内情報を提供するには，サンプルデータ提供者がダイアログに対して

[*1] WindowsAPI: http://msdn.microsoft.com/en-us/library/cc433218.aspx/
[*2] EnumWindows 関数: http://msdn.microsoft.com/ja-jp/library/cc410851.aspx
[*3] GetClassName 関数: http://msdn.microsoft.com/ja-jp/library/cc364600.aspx
[*4] GetAncestor 関数: http://msdn.microsoft.com/ja-jp/library/cc364581.aspx
[*5] GetWindowText 関数: http://msdn.microsoft.com/ja-jp/library/cc364815.aspx
[*6] GetWindowRect 関数: http://msdn.microsoft.com/ja-jp/library/cc364769.aspx




```
{
  "DialogGuide.exe":{
    "Title":"ソフトウェアのインストール",
    "Static":["ソフトウェアのインストールを開始しますか?"],
    "Button":[
      { "#Name":"はい(Y)",
        "#Position":[ 280, 145, 399, 185 ]
      },
      {"#Name":"いいえ(N)",
        "#Position":[407, 145, 526, 184 ]
      }
    ]
  }
}
```

図 3　ダイアログ情報の JSON データ化の例

行う操作の情報と，それによるダイアログの表示内容の遷移を収集する必要がある．ダイアログの出現時からダイアログが閉じられるまで，ユーザの操作を監視し，選択操作が行われた際に，選択されたオブジェクトと，選択後に遷移したダイアログの情報を取得して，3.1.2 節で作成された JSON データに追記する．ユーザの操作を監視にはグローバルフックを用いてマウスポインタのクリック座標を取得するのが最適である．

まず，Windows フック用関数，SetWindowsHookEx 関数[*7]を用いて，フックプロシージャ用の DLL を作製し，ユーザのクリック操作とその座標を監視する．座標よりウィンドウハンドルを取得する関数，WindowFromPoint 関数[*8]より，選択されたボタンを特定する．また，ユーザの操作によって遷移したダイアログの情報も，3.1.2 節の方法を用いて収集し，JSON データに追記する．例えば，図 4 のように，ボタンの選択によりダイアログが遷移した場合，JSON データ内の対象オブジェクトの要素に対して，遷移後のダイアログ情報を収める JSON データ名を追記する．

このデータ形式でダイアログの遷移情報を表現する．ダイアログの各オブジェクトの情報に，遷移情報として遷移後のデータ名を挿入するのは，1 つのビットマップデータと，それに対応する JSON データとをセットにして管理し，かつ，JSON データ内の情報を冗長にせず，簡潔に表現できるという利点がある．

3.2　データ管理サーバ

データ管理サーバでは，ダイアログの情報及びユーザの操作情報のデータを，データベースに格納する．また必要に応じて，管理者は手動で操作案内情報を編集する．更に，支援対象ユーザからのリクエストに応じて，データベース

*7　SetWindowsHookEx 関数: http://msdn.microsoft.com/en-us/library/cc433218.aspx/
*8　WindowFromPoint 関数: http://msdn.microsoft.com/ja-jp/library/cc364874.aspx

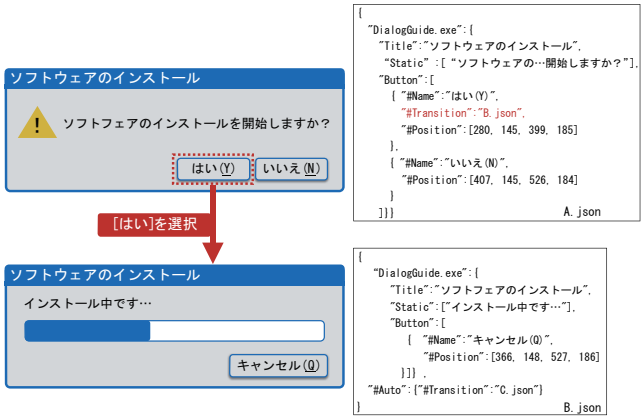


図 4　ダイアログ遷移の例

のダイアログ情報に基づいて案内情報を自動生成して返答する．

3.1.1 節のダイアログ検出方法では，本研究で取り扱う通知ダイアログの他に，ファイル選択やオプションウィンドウといった，モーダルウィンドウも検出することになる．これらのモーダルウィンドウと通知ダイアログの識別する必要がある．そこで，本研究ではダイアログ内のアイコンを利用する．確認や警告などの通知ダイアログのほとんどは，特定のアイコンを保有している (図 5)．そこで，サーバ上でダイアログの画像データに対して，OpenCV[*9]によるテンプレートマッチングを行い，通知ダイアログ以外を除外する．

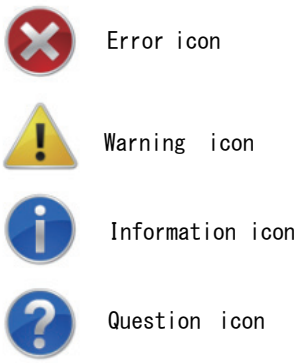


図 5　ダイアログ内のアイコンの例

サーバでのデータ管理は，後のデータベースに対しての検索を考慮し，ダイアログを生成したアプリケーション毎にディレクトリを分割して管理する．更に，ダイアログのタイトルでもディレクトリを分割する．すでに同一タイトルのダイアログのデータが，ディレクトリに保存されていれば，JSON データ内の配列構造とオブジェクトを比較し，それらが完全一致していれば，同一のダイアログであるとみなす．

データが同一のダイアログの情報であると判定できれば，

*9　OpenCV: http://opencv.org/

JSON データのマージを行い，ひとつのダイアログに対して異なる選択を行った場合の，ダイアログ情報の遷移を補間していく．その際，マージされた回数を JSON データ内に追記しておく (図 6)．この処理により，各ボタン（チェックボックス）などが押下（選択）された割合が記録される．

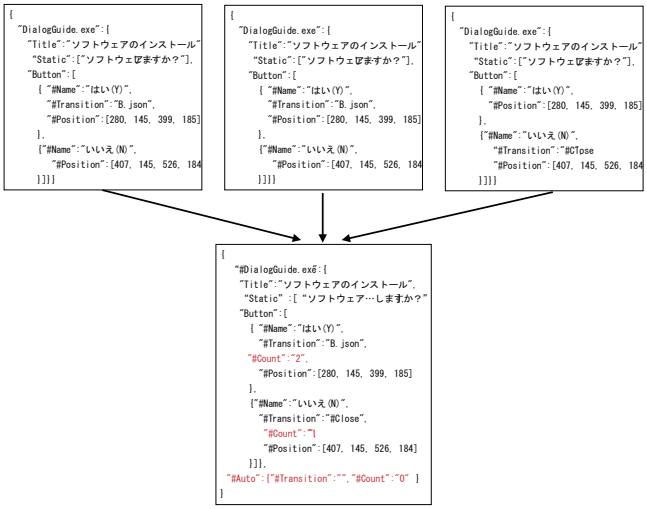


図 6　JSON データのマージの例

ダイアログがボタン選択以外の操作を要求するような場合，案内情報として説明や注釈があるのが望ましい．そこで，サンプルデータ提供クライアントより取得した情報に，データ管理者がアノテーションを可能にする．本システムでは，JSON データ内に *#note* という要素名と任意の文章を追記すれば，案内情報の中に説明文として表示される．

3.3　支援対象クライアント

支援対象ユーザ側のクライアントでは，サンプルデータ提供クライアント同様，ダイアログの発生を検知し，ダイアログの情報，ユーザの選択操作情報をサーバに送信する．サーバでは，支援対象クライアントより送信された情報をデータベースと比較し，同一のダイアログのデータに基づいて HTML 形式の案内情報を生成して，支援対象ユーザーに返答し，ブラウザに表示させる．

3.3.1　案内情報の自動生成

案内情報の生成は，あらかじめ用意された JSON データをスクリプトによって解析し生成する．本システムでは案内情報のデータ形式は HTML を採用する．HTML 文であれば，ブラウザ等で表示が可能であり，専用のビュアを実装する必要がない．

図 7 に案内情報の一例を示す．案内情報では，ダイアログの遷移を，ダイアログのビットマップデータを順に並べて表示し，矢印で接続して，遷移を表現する．また，3.2 節で JSON データに追記された，マージの回数情報を用いることで，他のユーザが一般的にどのような選択を行っているかを，統計として提示する．さらにその統計情報を元に，最も選択確率が高い選択肢と遷移を 1 カラムになる様に表示し，そうでない，遷移は別のカラムになるようにする．この手法により，他のユーザが行っている一般的な操作手順を明確に表現することが可能となる．

本システムでは，遷移を表現する矢印を HTML5 の Canvas により自動生成する．3.1.2 節の手法により収集したオブジェクトの座標情報に基づいて，ダイアログの画像上のボタンのある位置に div エレメントを埋め込む．この div エレメントと遷移後のダイアログの画像を，jQuery-ArrowMark プラグイン[*10]を用いて，動的な矢印を生成して接続する．これにより，後にダイアログの遷移情報が新たに追加されても，案内情報のレイアウトに大きな変更を加える必要がない．

さらに，3.2 節で追加入力された，*#note* のオブジェクトより，管理者がアノテーションとして入力した説明文が提示される．

3.3.2　案内情報の表示とユーザ操作の追跡

案内情報の生成は，ダイアログの発生時だけでなく，ダイアログが遷移する度に行われ，ダイアログの状態に対応した案内情報はリアルタイムに変更される．このため，ユーザは案内情報に対してスクロール等の操作を行う必要がない．従来，何フェーズにもわたるような複雑なソフトフェアのインストール時では，ブラウザで手順情報等を表示しながら行うのが一般的である．しかし，この手法を用いれば，案内情報がダイアログの状態に追従するため，シームレスな操作方法の確認が可能である．

4.　評価実験

通知ダイアログの操作時における案内情報提示の有用性を検証する．本実験の検証項目は，案内情報の有無によるダイアログ発生時から操作完了までに要する作業時間，及び操作選択時の安心感・信頼感，作業の容易さである．

4.1　実験方法

本実験では，ブラウザ上に架空のソフトウェアインストールのためのダイアログ画像を表示し，被験者はその画像内のボタンをクリックして操作選択を行う．実験画面のサンプルを図 8 に示す．被験者は，同じダイアログに対して，案内情報を提示しない状態，提示する状態の，2 つのパターンにおいてインストール作業を行う，被験者は，10 代から 20 代の男女 20 人である．各パターンの試行順序についてカウンターバランスをとり，実験を実施した．ダイアログ操作の選択肢に，作業が中止される選択肢を織り交ぜ，作業が完了するまでにかかった時間と試行回数を計測する．最後に，5 段階評価で安心感・信頼感，作業の容易さをアンケートで集計する．

[*10] jQuery-ArrowMark:　https://github.com/lislis36/jQuery-ArrowMark

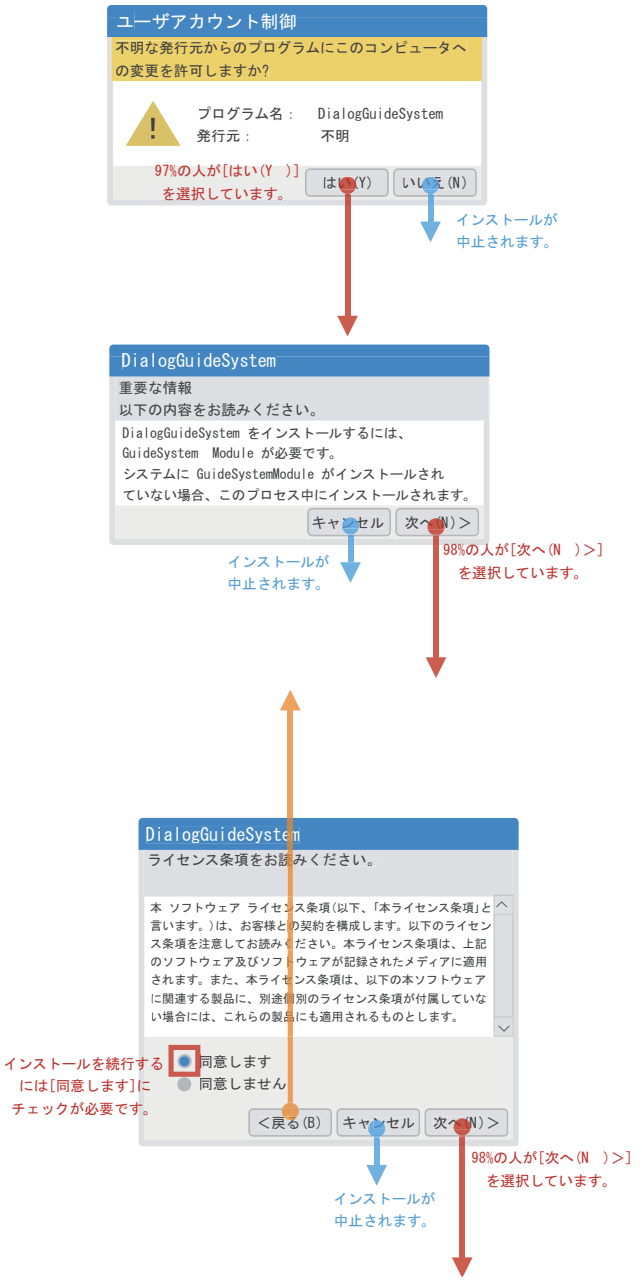


図 7　案内情報提示の一例

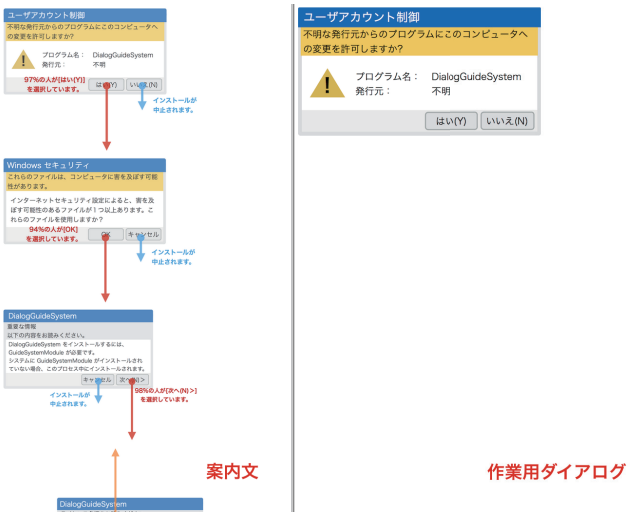


図 8　実験用ブラウザ画面 (案内情報あり)

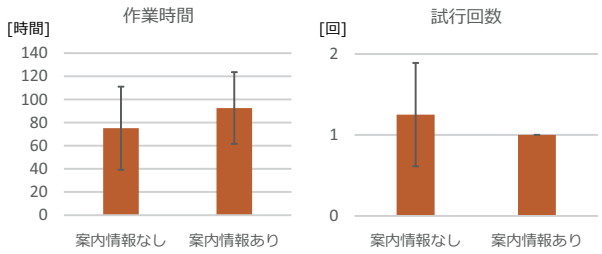


図 9　案内情報提示の有無による作業時間と試行回数の違い

4.2　実験結果

実験結果を表 9 に示す．作業に要する時間は，案内情報なしが約 75 秒，案内情報ありが約 93 秒となり，案内情報の提示によって作業時間が長くなってしまうことが示された．一方で，作業が完了するまでの試行回数は，案内情報なしが 1.25 回，案内情報ありが 1 回であり，案内情報の提示によって全被験者が試行回数 1 回で作業を完了させることができた．

アンケートの結果を表 10 に示す．アンケートの結果，案内情報の提示によって作業の安心感・信頼感，及び作業の容易さの向上が確認できた．案内が表示されない場合の安心感・信頼感の 5 段階評価の平均は，2.05 ポイントであり，案内が表示される場合は 4.05 ポイントと，2.00 ポイントの増加が示された．作業の容易さも同様に，案内情報が表示されない場合は 2.50 ポイント，案内情報が表示される場合に 4.15 ポイントで，1.65 ポイントの増加が示された．

以上よりダイアログを選択操作する場合の案内情報の提示は，作業時間が長くなってしまうというデメリットがあるものの，正確な操作・安心感・信頼感・容易さといった，計算機初心者が重視する評価項目の面においては，効果的な手法であることが確認できた．

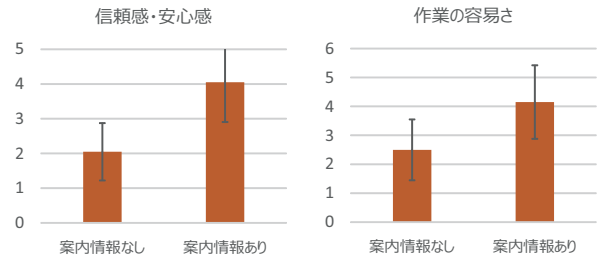


図 10　アンケートの結果

5.　おわりに

本研究では，ユーザビリティの高いダイアログの開発は，開発者の意図やシステム環境のために，表示形態の統一が困難であるという，従来研究の問題を解決し，既存のダイアログ全てに対応するために，ダイアログに対応する操作

案内情報を，ユーザに対して自動提示する手法を提案した．

本提案手法では，サンプルデータ提供クライアントでダイアログの発生を検知し，ビットマップデータをキャプチャする．更に，表示コンテンツや発生元のアプリケーション，ユーザがどのボタンを選択したか等の操作情報を抽出し JSON データ化する．これらの処理はすべて自動的に行われ，データはサーバに送信される．サーバサイドでは，同一ダイアログに対して収集された複数のデータを比較し，ダイアログの発生から閉じられるまでの操作情報を，単一のデータに纏める．また，案内情報に表示させたい説明文を，必要に応じて手動でデータ内に入力しておく．支援対象クライアントではダイアログ発生を自動検知し，サーバに対して，必要な案内情報を要求する．サーバはその要求に応じて，HTML 形式の案内情報を集約データより自動生成し，他のユーザが選択しているボタンの統計情報や，ダイアログの遷移情報，及びサーバで手動入力された情報を提示する．

案内情報提示の有効性を実験により検証し，ダイアログ選択操作の安心感・信頼感，及び作業の容易さの向上が確認された．

今後の課題として，ダイアログの同一判定方法の改善が挙げられる．本手法では現在，ダイアログを構成するオブジェクトの情報が完全一致している場合のみに，ダイアログが同一であると判定している．しかし，オブジェクトの情報がシステム環境に依存する場合があり，その場合には本手法では同一判定ができない．そのため，今後同一判定の方法について検討する必要がある．

参考文献

[1] 三好史隆, 倉本到, 渋谷雄, 辻野嘉宏. タスク集中度と認知時間を指標とした周辺表示法の評価 (ヒューマンコミュニケーション). 電子情報通信学会論文誌. A, 基礎・境界, pp. 831–839, October 2006.

[2] 三好史隆, 倉本到, 渋谷雄, 辻野嘉宏. マウス使用環境において周辺表示法がユーザの情報認知に与える影響. 電子情報通信学会技術研究報告. HIP, ヒューマン情報処理, pp. 65–70, January 2006.

[3] iTunes インストール／アップデート手順. http://www2k.biglobe.ne.jp/~t_muto/ipod/howto_installitunes.html.

[4] P.P. Maglio and C.S. Campbell. Tradeoffs in displaying peripheral informaiton. In *Proc. CHI'2000 ACM*, pp. 241–248, 2000.

[5] D.S. McCrickard, R. Catrambone, and T.J. Stasko. Evaluating animation in the periphry as a mechanism for maintaining awwareness. In *Proc. INTERACT*, pp. 148–156, 2001.

[6] J. Smarvell, R. Srinivasan, K. Woods, and O. Vasnaik. Measuring distraction and awareness caused by graphical and textual displays in the periphery. In *Proc. 39th Annual ACM Southeast Conference*, pp. 14–27, 2001.

[7] L. Bartram, C. Ware, and T. Calvert. Moving icons: Detection and distraction. In *Proc. IFIP TC.13 International Conference on Human-Computer Interaction (INTERACT 2001*, pp. 157–165, 2001.

[8] M. Czerwinski, E. Cutrell, and E. Horvitz. Instant messaging: Effects of relevance and timing. In *British Computer Society*, pp. 71–76, 2000.