# PC 操作支援技術の提案

恵木正史、直野健、櫻井隆雄†

近年、PC 業務の効率改善のニーズが高まっているが、１つの業務プロセス中に、異なるアプリケーション（以下アプリ）へ同じ入力を要する等、非効率的な実態となっている。そこで報告者は、PC に導入するエージェントにより、ユーザが少ない手順で、迅速かつ確実に業務アプリを操作できるよう支援する技術を開発した。本エージェントは、アプリに対するイベント割り込み制御により、アプリの変更なく、自動転記や入力ミス抑止を実現する。本手法の有効性を検証するため、社内のアプリに本技術を適用し、１案件の登録に要する時間を計測した。その結果、支援後は、支援前の約３割の時間で迅速に操作できている事を確認した。

# Proposal of PC Operation Support Technology

Masashi Egi, Ken Naono, Takao Sakurai†

Recently, the demand on improving workers' PC operations on business processes has been increasing. However, today's business applications are not so efficient to operate, that is, the workers are often forced to entry same data into two different applications even in one business process. Then we developed a PC operation support technology which enables the workers to quickly and surely operate existing applications by the agent program installed in each PC. Our agent realizes, for example, auto data transfer between applications and prevention of workers' typing error. To examine the effectiveness of our technology, we applied it to existing in-house applications and measured the lead time to entry one transaction data. Then we found that our technology could reduce the lead time to about one third compared with traditional one.

## 1. はじめに

近年、コンタクトセンタ、受発注センタ等を中心に、多数の派遣社員やパートからなるオペレータを一箇所に集め、大量定型の PC 業務を行う業態が進展している。一方、PC 業務の年間 TCO に占める人件費の割合は 40%にも達し[1]、上記の業態では業務効率の改善が急務の課題となっている。そのため、実際に PC 業務を行う現場のユーザ部門では、既存の業務アプリケーション（以下アプリ）の使い易さを向上するニーズが高まっている。

しかし、一般にこのような現場では、１つの業務プロセス中に、Web アプリ等のクライアント・サーバ型のアプリから、表計算アプリなどのクライアント・ローカル型のアプリまで、多数の業務アプリを組み合わせて利用している[2]。これらのアプリは相互に連携していないため、同じデータを入力する必要がある等、業務効率を劣化させる原因となっている。また、各業務アプリの開発形態も、他社製、自社製の両方があり、さらに自社製であっても、情報システム部門製、ユーザ部門製など多様である。そのため、使い易さを向上するために業務アプリを改善したくても、ユーザ部門主導によるアプリ改変は困難である。その結果、業務アプリは改善されること無く使われ続け、オペレータが長い時間を掛けて順応して行かざるを得ないのが実情である。

この様に現在の IT システムでは、ユーザが IT システムの都合に合わせなくてはならない傾向が強い。この問題を解決する １ つの方法として、報告者は、ユーザと IT システムとの界面である PC に知的なエージェントを常駐させ、そのエージェントにより、IT システムの使い勝手の悪さを吸収し、ユーザが「気が利く」と感じる支援を実現する、と言う構想を建てた。

本構想実現の第一歩として、本研究では、多様なアプリに対応し、ユーザがより少ない手順で簡潔かつ迅速にアプリを操作できるように支援する基盤技術を確立し、その有効性を検証することを目標とした。

## 2. PC 業務の現場と課題

### (1) PC 業務の現場で利用されるアプリ

PC 業務の現場における業務プロセスの典型例を図１に示す[2]。業務プロセスは幾つかの工程からなり、各工程では対応する別々の業務アプリを利用している。図１の例では、工程１ではメーラーを利用して案件処理を受付け、次に工程２では表計算アプリを利用して案件の帳票データを作成し、さらに工程３では社内共通の業務 Web ア

† (株)日立製作所 中央研究所 プラットフォームシステム研究部

プリを利用して帳票データを処理し、最後に工程４では現場の部署で独自に開発した業務 Web アプリを利用して、処理結果を上長に報告している。

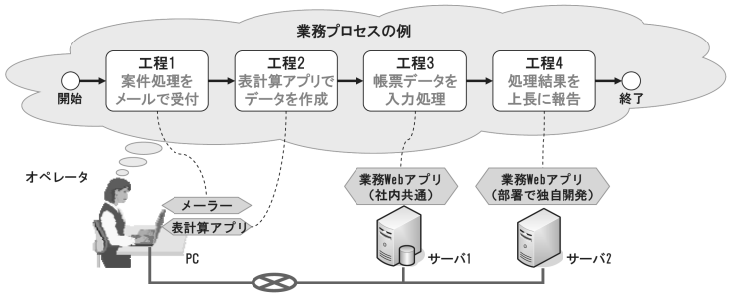


図１　PC 業務プロセスの模式図

このように１つの業務プロセスであっても、複数のアプリを組み合わせて利用するのが一般的である。表１に示す通り、PC 業務の現場では一般に、メーラーや表計算アプリなどのクライアント・ローカル型アプリから、クライアント・サーバ型アプリまで多様なアプリが利用される。また、クライアント・サーバ型アプリは、情報システム部門が提供する社員ポータルのような社内共通のアプリや、部署内で情報共有するための案件管理システムのような現場で独自開発したアプリ、さらに一般の Web サイトや、取引先の EDI システムなど社外のアプリなど多岐に渡る。また、クライアント-サーバ型アプリも、Web アプリだけでなく、非 Web アプリも多数利用されている。

表１　PC 業務の現場で利用されるアプリの分類

| # | アプリの分類 | | | アプリの例 |
|---|---|---|---|---|
| 1 | クライアント・ローカル型 | | | メーラー、表計算アプリ |
| 2 | クライアント サーバ型 | 社内 | 全社共通 | 受発注システム、社員ポータル |
| 3 | | | 現場独自 | 案件管理システム、ナレッジ管理システム |
| 4 | | 社外 | | 一般の Web サイト、取引先の EDI システム |

### (2) 現場の問題

上記のように PC 業務では多様なアプリを利用するが、オペレータが高い業務効率を発揮するには、各アプリは表２のような要件を満たしている事が理想である。業務効率を向上するには、実際にアプリを操作するオペレータの意見を吸い上げ、継続的にアプリを改善して、そのような理想状態へブラッシュアップして行く必要がある。

表２ 業務効率の高いアプリの要件

| # | 要件 | 補足 |
|---|---|---|
| 1 | レスポンスが速い | 高頻度に使う機能ほど迅速である。 |
| | | 低頻度に使う機能は遅くても良い。 |
| 2 | GUI が使い易い | 入力順序が上から下、左から右で直感的である。 |
| | | 画面のスクロールがない。 |
| | | 画面数が少ない。 |
| | | 画面間の行き来が少ない。 |
| 3 | ミスを予防する | 入力された値の異常を検知する。 |
| | | オートフィル等によりミスの余地を削減している。 |
| 4 | 自動化が十分 | 手入力は、自動化が困難な項目に限定されている。 |
| | | アプリ同士が連携し、入力の手間を軽減する。 |

しかし、表３に示す通り、PC 業務の現場で利用されるアプリのうち、現場主導で改善できるものは極一部である。そのため、使い易さを向上するために業務アプリを改善したくても、現場主導によるアプリ改善は困難であった。その結果、業務アプリは改善されること無く使われ続け、オペレータが長い時間を掛けて順応して行かざるを得ないのが実情である。

表３　PC 業務の現場で利用されるアプリの改善の容易性

| # | アプリの分類 | | | 改善の容易性 |
|---|---|---|---|---|
| 1 | クライアント・ローカル型 | | | ×（パッケージなので改変不可） |
| 2 | クライアント サーバ型 | 社内 | 全社共通 | △（情報システム部門の判断で可能） |
| 3 | | | 現場独自 | ○（現場で改変可能） |
| 4 | | 社外 | | ×（社外なので改変不可） |

### (3) 研究の課題

表２の要件のうち、#1 の「レスポンスが速い」と#2 の「GUI が使い易い」は、演算性能やアプリの画面設計に関わる要件なので、アプリのソースに手を加えなければ改善できない*。一方#3「ミスを予防する」と、#4「自動化が十分」は、以下のようにアプリのソースに手を加えなくても、改善できる可能性が残されている。

* Web アプリに限定すれば、ブラウザの機能拡張機能、或いはローカルプロキシサーバを通じて、Web アプリの HTML ソースに手を加える技術が多数提案されている[7-12]。

一般に、実行中のアプリのプロセスでは、「ファイルを開いた」「ボタンが押された」など多種多様なイベントが発生している。セキュリティ関連のソフトウェア製品では、ファイル開閉や印刷に関するイベントに割り込むことで、アプリのソースに手を加えなくても、ユーザの操作に一定の制限を加えることを実現している[4-7]。このメカニズムを、業務効率化という観点で活用すれば、#3 と#4 の要件を満たすように、オペレータの操作を支援できると考えられる。

そこで、報告者は研究の課題を、アプリに対するイベント割り込み制御により、オペレータがより少ない手順で、迅速かつ確実にアプリを操作できるようにする PC 操作支援技術を確立し、その有効性を検証する事と設定した。

## 3. PC 操作支援技術

### 3.1 支援範囲の検討

報告者が提案する PC 操作支援技術のアーキテクチャを図 2 に示す。

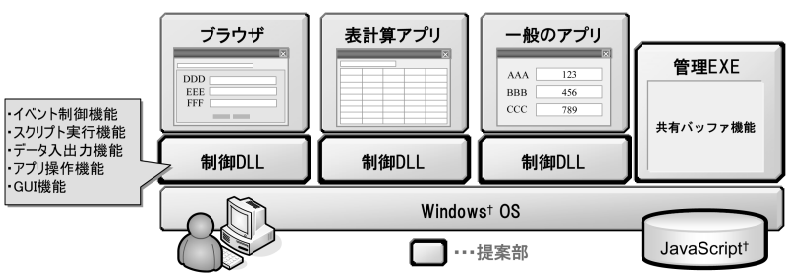


図 2　PC 操作支援技術のアーキテクチャ概要図

本技術は各 PC にインストールするエージェントソフトウェアにより実現する。本ソフトウェアは３つのモジュール、制御スクリプト、制御 DLL(Dynamic Link Library)、管理 EXE で構成される。まず、制御スクリプトは、予め現場に常駐する SE 等が本技術の専用の JavaScript†拡張言語により「どのようなイベントが発生したら、どのように支援するか」を記述したものである。また、制御 DLL は表４に示した各種機能を有し、制御スクリプトに従い、各アプリのプロセスで目的のイベント発生を検知すると、それに割り込んで操作支援を実行する。

表 4 制御 DLL の機能

| # | 機能名 | 概要 |
|---|---|---|
| 1 | イベント制御機能 | アプリでイベントが処理される前に割り込み、スクリプト実行機能を動作させた後に、処理を再開、または破棄する。 |
| 2 | スクリプト実行機能 | 制御スクリプト(専用の JavaScript 拡張言語)を実行する。内容に応じて、データ入出力機能、アプリ操作機能、GUI 機能を呼び出す。 |
| 3 | データ入出力機能 | 既存アプリ画面の指定された部位のデータを読み書きする。データの格納場所は、スクリプト内のローカル変数や、プロセス横断の共有バッファなど多様。 |
| 4 | アプリ操作機能 | 既存アプリの起動・終了させたり、既存アプリ画面のボタン、プルダウンなどの GUI 部位を操作する。 |
| 5 | GUI 機能 | メニュー、メッセージボックス、インプットボックスなどの GUI を表示し、ユーザと対話する。 |

さらに、管理 EXE は、制御 DLL 間で共有して読み書きする共有バッファ機能を提供すると共に、制御 DLL をアプリのプロセスに接続・解除する役割を担う。

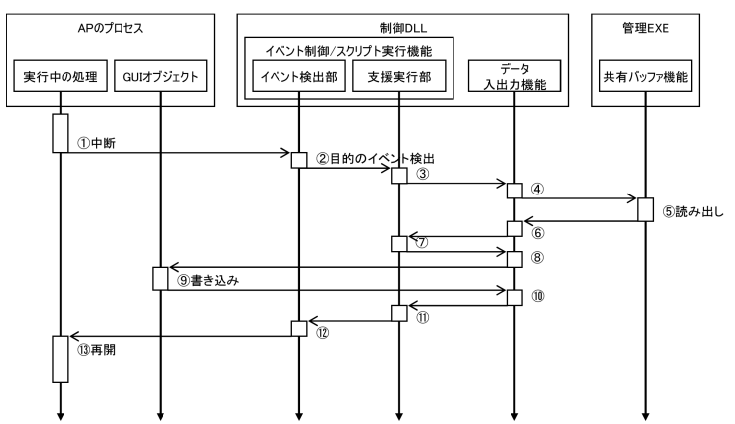


図 3　処理の流れの一例（シーケンス図）

† Windows は米国 Microsoft Corporation の米国およびその他の国における登録商標、JavaScript は Sun Microsystems, Inc の米国およびその他の国における登録商標である。

次に、イベント発生から、支援実行までの一連の処理の流れの一例を図 3 に示す。アプリのプロセスでイベントが発生すると、イベント制御機能が処理に割り込み、イベント検出部により発生したイベントが制御スクリプトの条件部を満たすかどうかを判定し、満たさなければ直ちに元の処理を再開し、満たせば制御スクリプトの命令部を実行する。図 3 は、命令部を実行する場合を表している。

支援実行部は制御スクリプトの命令部に従い、データ入出力機能、共有バッファ機能、GUI 機能、アプリ操作機能を動作させる。図 3 は、共有バッファからデータを読み出し(図 3 中③から⑦)、その内容をアプリの GUI オブジェクト(テキストボックス、プルダウン等）に書き込む（図 3 中⑦から⑫）例を表している。

この様に、アプリで発生するイベントをトリガーに、制御スクリプトを実行する事により、多様な操作支援が可能となる。次節で、本技術の応用例を述べる。

### 3.2 応用例

本節では、実際に社内で利用されている 2 つのアプリを例に、提案技術の応用を検討する。

**(1) 応用例１　業務アプリ A から業務アプリ B への自動転記する例**

ある現場では図 4 に示す様に、従来、オペレータは業務アプリ A に手入力した内容とほぼ同一の内容を、別の業務アプリ B に再び手入力する効率の悪い作業を行っていた。

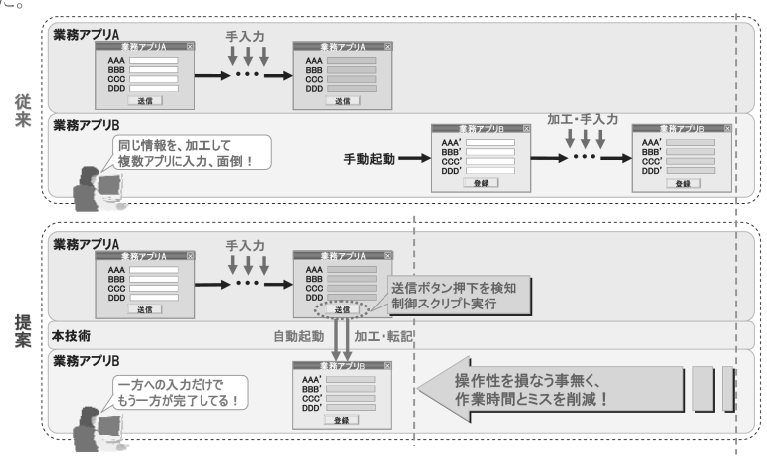


図 4 応用例 1 の説明図

このような場合、本技術により業務アプリ A の送信ボタン押下のイベントに割り込んで制御スクリプトを実行すれば、オペレータは業務アプリ A に手入力するだけで、業務アプリ B への入力が自動的に完了するため作業時間を大幅に削減することができる。

**(2) 応用例２　入力ミスを抑止する例**

ある現場では図 5 に示す様に、従来、既存業務 Web アプリに入力値のチェック機能が無かったため、承認者からの差し戻しが頻繁に発生していた。

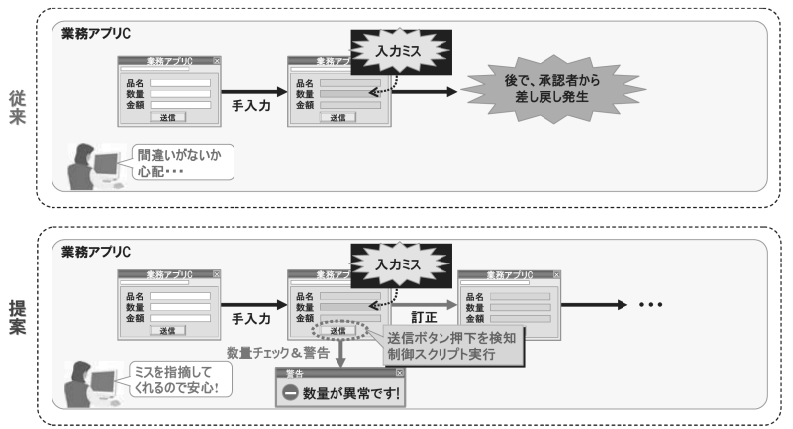


図 5 応用例 2 の説明図

このような場合、本技術により既存業務 Web アプリの送信ボタン押下のイベントに割り込んで制御スクリプトを実行すれば、オペレータの入力ミスを自動検知して、警告画面を表示すると共に、送信ボタン押下を破棄できるため、入力ミスを確実に抑止することができる。

## 4. 有効性の検証

提案手法の有効性を検証するため、社内の旅費精算システムと就業管理システムを連携する初期評価実験を行った。以下では、まず両システムを横断した業務について述べ(4.1 節)、次に提案手法適用の前後での 1 案件の登録時間の計測結果について述べる(4.2 節、4.3 節)。

### 4.1 適用対象の業務システム

弊社では、従業員は出張活動を行った際、目的の異なる二つのシステムに出張情報を入力しなくてはならない。１つ目は従業員の出張に関わる費用を精算するための旅費精算システム、２つ目は従業員の日々の勤休状況等を管理するための就業管理システムである。それぞれの画面の模式図を図６に示す。

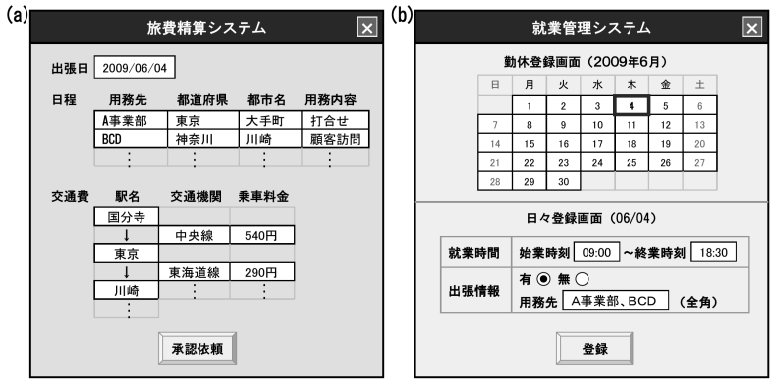


図６ 旅費精算システムの画面と、就業管理システムの画面の模式図

旅費精算システムは図 6a に示す通り、出張日、日程、交通費（駅名、乗車料金）などの入力項目を持ち、一方、就業管理システムは図 6b に示す通り、日毎に就業時間、出張情報（出張の有無、用務先のリスト）などの入力項目を持つ。

両システムにおける出張日と用務先は共通する入力項目であるが、両システムは目的毎に独立に導入された経緯があるため、互いのデータ連携は行われていない。そのため、従業員は同じ情報を両システムに重複して入力する必要があり、効率的ではない。

また、旅費精算システムへの入力は出張日の翌日に行うが、就業管理システムへの入力は何日分かまとめて入力するなど、両システムへの入力を同期させていない従業員も多数いる。そのため、従業員の入力ミスにより、双方に入力したデータが整合せず、後日監査で指摘される場合も見受けられる。

そこで、提案手法の適用により、従業員が旅費精算システムに入力した内容を、就業管理システムに自動的に反映する仕組みを実現し、適用の前後で、就業管理システムに１案件登録する時間がどの程度効率化するかを評価した。

### 4.2 提案手法適用前の案件登録時間

提案手法適用前の就業管理システムにおいて、5 件の登録時間を測定した結果を表 5 に示す。

表５ 提案手法適用前の登録時間計測結果

| # | マウスクリック数 | キーボード打数 | 登録時間（秒） |
|---|---|---|---|
| 1 件目 | 6 | 20 | 27.2 |
| 2 件目 | 7 | 23 | 23.1 |
| 3 件目 | 6 | 19 | 22.8 |
| 4 件目 | 6 | 24 | 24.6 |
| 5 件目 | 7 | 30 | 21.7 |
| 平均 | 6.4 | 23.2 | 23.9 |

マウスクリック数が安定しているのに対し、キーボード打数が大きく上下しているのは、入力する文字列が案件毎に異なるためや、文字の打ち間違えによるものである。

### 4.3 提案手法適用後の案件登録時間

報告者は、提案手法により両システムに新たな機能をアドオンするにあたり、極力新たな機能の操作方法を習得しなくても良いよう配慮し、表６に示すアドオン機能を制御スクリプトにより作成し、それぞれのシステムに適用した。

表６ アドオン機能の概要

| # | 対象 | アドオンした機能 | |
|---|---|---|---|
| | | 割り込んだイベント | 実行内容 |
| 1 | 旅費精算システム | ユーザが「承認依頼」ボタンを押下した瞬間 | 画面からユーザが既に入力した内容(出張日、用務先等)を読み出し、データベースに格納する。 |
| 2 | 就業管理システム | ユーザが「日々登録画面」を開いた瞬間 | 画面から日付を読み出し、当該日付に一致する出張情報をデータベースに問い合わせ、該当出張情報あった場合、画面の「有」のチェックボックスにチェックを書き込み、画面の「用務先」入力欄に該当出張情報の用務先の値を書き込む。 |

このように実装することにより、旅費精算システムについては従来と同一の使用方法を提供し、また就業管理システムについては、ユーザが出張のあった日の「日々登録画面」を開くと、出張情報が既に値が埋まっている状況を提供できる。

提案手法適用後の就業管理システムにおいて、5 件の登録時間を測定した結果を表 7 に示す。

表 7　提案手法適用後の登録時間計測結果

| # | マウスクリック数 | キーボード打数 | 登録時間（秒） |
|---|---|---|---|
| 1 件目 | 1 | 12 | 7.2 |
| 2 件目 | 1 | 12 | 7.3 |
| 3 件目 | 1 | 21 | 10.0 |
| 4 件目 | 2 | 13 | 9.3 |
| 5 件目 | 3 | 12 | 8.2 |
| 平均 | 1.6 | 14.0 | 8.4 |

提案手法適用前後を比較すると、適用後は適用前の約 3 割の時間で迅速に操作できていることが分かった。この様にして、既存システムの従来の使い勝手と整合するやり方で、操作を支援することが出来る。以上のようにして、提案手法の有効性を確認することができた。

## 5. まとめ

本報告では、PC に導入するエージェントにより、ユーザが少ない手順で、迅速に業務アプリを操作できるよう支援する技術を開発した。

本エージェントはユーザの PC 操作を常時モニタリングしており、特定のイベント発生を検知すると、当該イベントに割り込んで、制御スクリプト(専用の JavaScript 拡張言語)を実行することができる。また、制御スクリプトから実行される支援機能を多数備えており、これらの機能により、本エージェントはユーザのアプリ操作を柔軟に支援することができる。

本手法の有効性を検証するため、社内のアプリに本技術を適用し、1 案件の登録に要する時間を計測した。その結果、支援後は、支援前の約 3 割の時間で迅速に操作できている事を確認した。

## 参考文献

1) Thomas Pisello, “Return on Investment For Information Technology Providers - Using ROI as a Selling and Management Tool“, Information Economic Press (2002).
2) 直野, 恵木, 櫻井, “オフィス業務の「見える化」～PC モニタリングによるプロセス改革～”, uValue コンベンション(2007).
3) 株式会社アスキーソリューションズ, “Desktop Scout”, http://www.asciiisolutions.com/products/dts/index.html (2005).
4) Verdasys 社、”Digital Guardian”,　http://www.verdasys.com (2003).
5) 株式会社ソリトンシステムズ, “InfoTrace”, http://www.soliton.co.jp/news/pdf/25_09_infotrace14.pdf (2005).
6) 株式会社ウイング, “ALL Watcher”, http://www.all-watcher.com (2003).
7) Trixie, http://www.bhelpuri.net/Trixie/Trixie.
8) Grasemonkey, https://addons.mozilla.org/en-US/firefox/addon/748.
9) SeaHorse, http://www.fenrir.co.jp/sleipnir/plugins/seahorse.html.
10)　UserJavaScript, http://www.opera.com/browser/tutorials/userjs/.
11)　Proxomitron, http://www.proxomitron.info/.
12)　中山心太:次世代 Grasemonkey, Tsukikage System の紹介, 第 49 回プログラミングシンポジウム ( 2008).