研究論文

カイゼン活動の文脈でのEnd User Development
～文化的アプローチによる実証研究～

道　用　大　介

アブストラクト：

本論文は基幹システムでは対象とできない作業のシステム化をシステムを使う者（エンドユーザー）が行うEnd User Development（EUD）を扱ったものである。個人的なEUDによって開発されたシステムが、組織の業務プロセスの中に取り込まれ（個人プロセスの標準化）、それらのシステムのメンテナンスが開発者に依存し（システムの属人化）、開発者がその職場に留まり続けることが前提（前提条件の永続性）であるという問題点を先送りしながら運用されていることを指摘し、カイゼン活動の文脈の中での組織的なEUDを提案した。組織的なEUDを行うためには多くのエンドユーザーがプログラミングを行う必要があるため、プログラミングの理解を阻害する原因の調査を行い、「現実作業とプログラム間の単語の置き換え現象」「現実アルゴリズムと記述アルゴリズムの乖離」「自然言語と人工言語の乖離」「現実アルゴリズムと記述アルゴリズムの乖離、自然言語と人工言語の乖離のトレードオフ現象」「データ構造の複雑性」という問題を抽出した。抽出された問題を解決するような設計指針の設定と補助ツールを開発し、カイゼン活動という文脈の中で5年にわたる実証検証を行った結果、プログラミング経験のない参加者により組織的かつ継続的なシステム開発が行われ、827h/月のカイゼン効果が得られた。

キーワード：End User Development、カイゼン活動

1. はじめに

1.1　製造現場のIT化とEUD

情報技術の発達は企業の生産性を向上させ、社会の人々の生活スタイルを変え続けている。しかし、日本においては情報技術者の不足が指摘され[1]、グローバル化する市場での日本の相対的競争力の低下が危惧されている。特にAIやセキュリティなどの先端的なIT分野の人材確保は深刻な問題であり、従来型IT人材を先端IT人材へと移行していくことや外国人技術者の採用等が提案されている一方で、製造現場でのIT化はまだまだ進んでおらず、基幹システムは導入しているものの、生産計画に関わる情報の作成や入力、品質情報の照合、出荷情報の管理など現場の様々な作業は製造庶務職や事務職とよばれる従業員が「コンピュータを使った手作業」で行っているのが実態である。

1990年代には、EUC（End User Computing）やEUD（End User Development）といった情報システムの利用者が運用、開発に関わっていくという取り組みが行われることもあった[2]が、情報システム利用者が開発者になる

ための教育は定着しなかった。その後、企業に勤める多くの者の情報リテラシーが向上し、与えられた情報システムの利用のみならず、基幹システムから取り出したデータを表計算ソフトを用いて利用しやすい情報に加工することも一般的になり、マクロ機能を使ったり、VBA（Visual Basic for Applications）でコードを書くことによって個人的にEUDを行う者もでてきた。しかし、このような個人的なEUD（以後、個人的EUDとよぶ）は、企業内で統制のとれたものでなくシャドーIT問題の一部となり、2章で述べるようにシステムを作った者の異動や転職によってメンテナンスができなくなり、業務自体に支障をきたすこともある。ただし、業務に支障をきたす原因はEUD自体にあるのではなく、組織的にEUDに取り組んでいない状況の中で個人の能力に頼って効率化の恩恵を受けてしまったことにある。

EUDでカバーしようとした問題は現在でも解決されないままとなっているが、RPA（Robotic Process Automation）のようにほぼプログラミングを必要としないノンプラグラミングツールの開発[3]が進むなど技術的アプローチが主である。しかし、未だブレイクスルーとなるようなノンプラグラミングツールが登場していない状況にもかかわらず、EUDに関する議論は過去のものとなってしまい、地道にプログラミグを学び運用するという議論が少なくなっている。その理由は、プログラミング自体が難しく学習コストがかかりすぎるためであると考えられる。つまり、問題はあるが問題解決のためには多大なコストがかかるため、画期的な新しい技術の登場を待っている状態と言える。しかし、内木[4]がBurrellとMorgan[5]の枠組みを使って、

> *情報システムはⅠ社会・文化、Ⅱ機能・技術、Ⅲ機構・制度、Ⅳ思想・意識の4つの視点に関するダイナミズムである*

と述べたように技術的な問題だけでなく、会社の持つ文化や仕組みの中でどのように扱っていくかという議論もなされるべきである。

1.2　日本の製造業がもつ文化

日本の製造業がもつ文化に目を向けてみると、1986年に今井[6]によって紹介されたように現場で起きている問題を現場の小集団で解決していくカイゼン活動やQCサークルといったボトムアップ型の問題解決文化が根付いている。川瀬[7]はこのようなボトムアップ型（川瀬はライン中心型とよんでいる）の問題解決はトップダウン型の問題解決に比べて「すべての問題についてつねに誰かが考えている」ことがメリットであると指摘しており、本研究の主題であるEUDも基幹システムというトップダウンの仕組みでは対応しきれない部分の問題を現場のエンドユーザーが解決するという点では小集団でのボトムアップ型の問題解決プロセス（小集団活動）と類似したアプローチであると考えられる。

そこで、日本における製造業の小集団活動の歴史を概観すると、現在は一般的な活動ではあるが、文化として根付くまでにはある程度の時間を要している。1949年に日本科学技術連盟にQCリサーチグループが設立され、日本における統計的品質管理（SQC）がはじまり、1950年には品質管理の大家であるデミング（Deming, W. E.）が来日し講義を行なった。その後、品質が保証された製品を継続的に生産していくためには作業者の役割が重要であり、直接生産している作業者や職長がしっかりしなければQCはうまく進まないという考えのもと日本短波放送で職組長のQC教育講座が開始されたのが1957年である。1962年には雑誌「現場とQC」（後に「FQC」「QCサークル」に改称）が創刊され、QCサークルという小集団グループによるQC活動を呼びかけられたのがQCサークルのはじまりであり[8]、現在の製造業における小集団活動の原型となっている。普及にあたって強調さ

れたことは「命令ではなく、自主的にやりたい人から始めてほしい」「勉強してほしい」「視野を広くするために他のサークルと相互啓発してほしい」「職場全員参加を目指さなければならない」という点であった[9]。また、その際に高度な統計的知識がなくても統計的品質管理を実践できるようにQC7つ道具（パレート図、特性要因図、グラフおよび管理図、チェックシート、ヒストグラム、散布図、層別）が考案された。

1.3 小集団活動から得られるヒント

QCサークルの思想と実現プロセスで特徴的なことは「職場全員参加を目指した」ことと「QC7つ道具」という実現のための道具を作ったことである。これらの特徴は集団の中で生じるモチベーションの維持、全体の知識レベルの底上げ、個人能力差をカバーする、組織としての継続性などの観点で小集団活動の文化的定着に寄与したと考えられる。例えば、「職場全員参加を目指した」ことによって多くの従業員が問題解決活動に参加し、自らの職場の問題を自主的に解決する雰囲気が醸成される。特に定型業務が多い従業員にとっては問題解決活動で工夫を凝らすことで高次の欲求が満たされる[10]。また、「QC7つ道具」のような道具は個人的な能力差を吸収し、難解な統計理論を理解できなくてもオペレーションレベルでの問題発見を容易にして、より多くの従業員の活動参加を促した。

このような先例は、EUDの問題点である個人的で組織的でない、プログラミング自体が難しくて実施が難しいという問題に対して示唆を与えてくれる。つまり、QCサークルやカイゼン活動のような小集団活動文化を踏襲し、その文化的文脈の中で対象となる人々が使えるレベルの適切な道具を作り、社内の小集団活動として組織的EUDを行うことが可能なのではないかという示唆である。

1.4 目的

本論文の目的は、EUDにおける問題点とエンドユーザーがプログラミングを扱う上で障壁となる要素を分析した上で、カイゼン活動の一環としてEUDを行い、文化的文脈を付与することで、継続的かつ組織的に行うことが可能かを検証することである。

なお、本論文では招集集団活動やQCサークルなどを含むボトムアップの問題活動を総じてカイゼン活動とよび、それらが社内に根付いた状態の価値基準の体系をカイゼン文化とよぶこととする。

2. EUDの現状分析

2.1 調査対象企業の現状調査

本章ではEUDの現状の問題点を明確にするためにインタビューによる現状分析を行なった。調査対象は従業員数100名以上の5つの事業所（製造業）とした。

調査対象とした企業では既に活発にカイゼン活動が行われていたが、事務作業に関しては製造に関わる作業であってもカイゼン活動の対象とはなっておらず、他の多くの企業と同じように基幹システムはあってもエンドユーザーはMicrosoft社の表計算ソフトExcelのファイルで様々な計画、管理、書類作成を行なっていた。これらのエンドユーザーの事務作業は手順や照合基準の変更が多く、各工場や部署の独自の作業も多いため、大規模なパッケージソフトの一部に取り込むことは現実的ではないと判断されていた。トップマネジメントへのインタビューでも事務作業の効率化は必要であるとの意見は一致していたが、これら個別作業をシステム化することは作業内容があまりにも多岐にわたり、外部業者や社内システム部門を使っての開発は開発コストと導入効果が見合わないと判断されていた。

業務を行なっている現場に目を向けてみると、Excelのプログラミング機能であるVBA

（Visual Basic for Applications）を利用して個人的に開発したシステムが複数存在した。そこで、それらのシステムに関わった従業員が経験したことを抽出するため、開発者や利用者にインタビューを行なった。その結果、次のような7つの事例が確認された。

事例1：開発者

高校でプログラミングを学んだことがあるので、自分なりに勉強をしてマクロの記録を使って、自分の仕事を自動化した。今は、そのシステムを使って自分の仕事を行なっているが、作業時間が大幅に減った。

事例2：利用者

コンピュータに詳しい人がExcelで複雑な仕事をシステム化してくれて、ボタンを押すだけになったので非常に助かっている。作業時間も減ったが、何よりもミスが減った。

事例3：利用者

前任者が自分で作ったシステムを使った仕事を引き継いだが、引き継いだ後暫くして取り扱う品種が変わった際にどのように変更すればいいかわからず、前任者から引き継いだシステムは放棄して、エクセルを使った手作業に変更した。引き継ぐ際にシステム前提の引き継ぎ方をしたので、システム内で処理されているプロセスがわからず、プロセスを自分で調べることで大変な苦労をした。そのため､手作業のほうが仕事のプロセスがわかり、引き継ぎの際もプロセス自体を教えられるので安心できる。

事例4：利用者（開発されたものを変更）

前任者から引き継いだシステムを変更しようとプログラミングを勉強したが、マクロの記録を使って記述されており、不要なコードが多すぎて解読するために大変な時間がかかった。

事例5：利用者

システムに詳しい従業員に作ってもらったシステムを変更しようとしたが、英語ばかりで見ただけで諦めた。エクセルの関数などは理解していたが、VBAは全く別物でわからなかった。

事例6：利用者

システムに詳しい従業員に作ってもらったシステムを変更したいと思ったが、パスワードがかけてあってプログラムを見ることができなかった。また、その従業員は退職してしまい、もうどうにもできない。

事例7：開発者

自分のために作ったのに、担当が変わってもメンテナンスしてほしいと連絡がきてしまう。

2.2　個人的EUDの問題点

事例1、2はEUDによるメリットである。それに対して事例3～6は一時的なメリットはあったものの、状況の変化に対応できなかったEUDのデメリットの体験である。また、事例3～6はシステムを開発した本人の体験ではなく、前任者からシステムを引き継いだ者が引き継いだ当初はうまく行っていたが、状況が変わり修正が必要になった場合に対応ができないという共通点があり、それらの状況から事例7のような事例が生まれていることが推察される。個人的な道具として使い始めたシステムは、最初はあくまで個人的に工夫したプロセスであったはずだが、効率的であるがために徐々にその業務の標準プロセスへと変化（**個人プロセスの標準化**）し、担当者が変更になった後もシステム自体は使い続けられた。しかし、システムを変更する必要性に迫られ、変更しようとしてもできないことがわかって“**システムの属人化**”という問題にフォーカスがあたる。属人化したシステムは、開発した本人しか変更することが

できないため、その効果を継続させるには対象作業の前提条件を全く変更せずに使用するか、開発した担当者が永続的にその仕事に関わるしかない。つまり、ある程度の数の従業員にシステム開発能力がない状況では個人的EUDによる効果の持続は“**前提条件の永続性**”もしくは“**システムの属人化**”を前提としないと成り立たないといえる。しかし、経営環境の変化や配置転換、そもそもEUDの対象となる作業は変更が多いという特性などを考慮すると、これらの前提条件をおくことは現実的ではない。

以上のことから、基幹システムでカバーできない現場レベルの生産性向上を目指したシステム開発を行うためには、ある程度の人数のエンドユーザーがシステム開発・変更ができる能力を持ち組織的EUDに取り組むことが必要であると考えられる。すなわち、それは人材育成の一環として非IT人材をIT人材に育てる取り組みであるといえる。しかし、既に中等・高等教育課程を終えて企業でシステム開発とは関係のない業務を行なっている人材を簡易なシステム開発をできるIT人材に育てる研修は、教育を受ける側、教育を行う側の双方に大きな負担が生じる。そこで、次節では本節で得られた事例において使用されたシステムのコードを分析し、組織的EUDを実施する考慮するべき問題を抽出する。

2.3　理解と解読を阻害する要因の抽出

前節のインタビューで得られた7事例で使われていた15個のプログラムを分析し、プログラムの理解と解読を阻害している要因として下記の5つを抽出した。

1）**現実作業とプログラム間の単語の置き換え現象**
2）**現実アルゴリズムと記述アルゴリズムの乖離**
3）**自然言語と人工言語の乖離**
4）**2、3のトレードオフ現象**
5）**データ構造の複雑性**

以下に、これらの内容について説明する。

1）現実作業とプログラム間の単語の置き換え現象

多くのプログラミング言語はアルファベットを使って記述されるため、日本語に対応していない。そのため、変数を宣言する際もアルファベットを使い適切な英語の意味を持った変数名にすることが一般的である。しかし、英語に慣れていない初学者にとってはこのことが大きな障壁になる。英語を理解できなかったり苦手な者がアルファベットで書かれた文字の羅列に対して、積極的に関わろうとしないことは容易に想像できる。今回の調査で使われていたVBAは変数名やプロシージャ名に日本語が使用できるため、変数をアルファベットで記述する必要がないにも関わらず、対象企業で調査した事例ではアルファベットで記述されていた。つまり、現実の作業をプログラムに落とし込んだ際に“**使用単語の置き換え**”が起きていた。普段使用している単語とは別の言葉を使うことで、プログラムでどのような処理がされているかの概略を理解することさえ難しくなっていた。それらのシステムを開発し、インタビューが可能であった3名に聞き取り調査を行った結果、その理由として「自分が勉強した書籍はアルファベットで書かれていたので、日本語を使用できることを知らなかった」「基本文法は全てアルファベットであることからアルファベットで記述したほうが良いのかと思った」「基本的なことはアルファベットで記述しなければいけないので、日本語変換するのが面倒だった」という回答が得られた。

2～4）乖離現象とトレードオフ

Excelのワークシート関数による集計とアルゴリズムによる集計の方法は異なる。例えばB1セルにA列の1行目から10行目までの合計を求める場合、ワークシート関数ではセ

ルに

```
=SUM (“A1:A10”)
```

と記述するだけで合計を求めることができるが、アルゴリズムで求める場合は

```
Dim 合計
合計 = 0
For i = 1 To 10
   合計 = 合計 +Cells (i, 1)
Next i
Cells (1, 2) = 合計
```

と記述する。ノンプログラマーにとっては、このようなアルゴリズムで合計を求めることは非生産的であるように感じるようである。日常業務で行なっている関数などを使った作業手順は広義の意味でアルゴリズムであることから以後、現実アルゴリズムとよび、プログラムとして記述されているアルゴリズムを記述アルゴリズムとよぶこととすると、上記の例のように現実アルゴリズムと記述アルゴリズムが異なる現象は“**現実アルゴリズムと記述アルゴリズムの乖離**”といえるであろう。一方で、合計を求めるSUM関数をVBAのプログラムの中で使用し

```
Cells (1, 2) = Application.WorksheetFunction.
Sum (Range (Cells (1, 1), Cells (10, 1))
```

というように1行で記述することもできる。このように記述すると現実アルゴリズムと記述アルゴリズムの乖離は小さいが、自然言語とはかけ離れた命令文が必要となり、英単語に慣れていないノンプログラマーからすると理解が難しく、何をしているかわからないという意見が聞かれた。つまり、現実アルゴリズムと記述アルゴリズムの乖離を避けようとすると“**自然言語と人工言語**（プログラミング言語）**の乖離**”が発生してしまうのである。しかし、自然言語と人工言語の乖離を避けるために複雑な人工言語を使わず平易な人工言語表現だけで記述しようとすると、記述が長くなって現実アルゴリズムとの乖離が発生してしまう。つまり、これらの**二つの乖離現象はトレードオフ**のような関係にあるのである。このような現象はマクロの記録機能を使ってプログラムを作成した場合に顕著に現れる。マクロの記録とは記録中に行った作業をすべてプログラムに変換する機能でプログラミングができなくてもプログラムを作成できる仕組みである。マクロの記録機能を使って作成したプログラムは現実アルゴリズムをそのまま記述しているため、現実アルゴリズムと記述アルゴリズムの乖離は全くない。しかし、記述されたプログラムは自然言語とはかけ離れた文字の羅列になるため、事例4でも述べたように多少プログラムが理解できる者でもマクロの記録機能によって記述されたプログラムを解読することは困難になる。

5）データ構造の複雑性

もう一つ観察された事象は、必要以上に複雑な処理を試みようとしていることである。調査対象とした企業においてエクセルの用途を調査したところ、計算という役割だけでなく、入力画面としての役割、印刷する書類として役割を持って活用されていた。例えば、ワークシートを利用した生産計画表では、その表の一部には数式が入れられており、担当者が在庫や必要量を確認しながら生産量を決定し、生産量が入力されると数式によって在庫量が自動計算され、入力が終わるとそのワークシートの印刷や、ファイル共有と形で関係部署に配布される。生産計画表に限らず、計画・進捗管理に関わる表は生産・物流現場では多く見られ、これらの表には業務に必要な多くの情報が記載されていたり、見やすくするために空白行や空白列を挿入するなどの工夫がされている。このような工夫は現場の知恵によって作られたものであるが、あくまで「人間」が作業しやすい・見やすいものである。人間が見やすい表を元にデータ処理をしようとすると、例外処理の連続となり非常

に複雑なプログラムになる。つまり、コンピュータで処理するにはデータの並び方が複雑すぎるのである（**データ構造の複雑性**）。一方でデータベースのテーブルのような形のデータは作業担当者にとっては見づらいが、コンピュータでのデータ処理としては決まった列に決まった属性のデータが入力されているため例外処理の少ないプログラムとなる。データ構造の複雑性が原因で、業務としては非常に簡単なものであってもプログラムにすると複雑なものとなり、解読の容易性を下げる一因となっていた。

2.4 阻害要因と対応策

前節で挙げたプログラムの理解と解読を阻害する要因に関して、本節では対応策を検討する。「現実作業とプログラム間の単語の置き換え現象」に関しては変数名の付け方という非常に簡単な問題のため、研修で設計ルールを定めるなどの対応が可能であると考えられる。また、「データ構造の複雑性」は、プロセスは増えるがアウトプットのデータ構造から単純なデータ構造に変換するプロセス、生産計画などの目的のデータ処理を行うプロセス、アウトプットのスタイルに転記するというプロセスに分けるという設計ルールを定めることで主要なアルゴリズムの複雑性はある程度回避できると考えられる。しかし、「現実アルゴリズムと記述アルゴリズムの乖離」「自然言語と人工言語の乖離」に関してはプログラミングをする上でどうしても生じる問題であり運用ルールなどで解決できる問題ではない。研修に時間をかけて理解できるように促すか、何かしらの道具を使って問題を小さくする必要があると考えられる。研修での理解を深めるという方策は難解なことを従業員の能力に依存することであり、組織的に多くの従業員の参加を促すためには個人の能力に依存しないためのQC7つ道具のような実践を容易にするための道具づくりが必要であろう。また、これら「現実アルゴリズムと記述アルゴリズムの乖離」「自然言語と人工言語の乖離」はトレードオフの関係にあるため、別々に扱うのではなく同時に扱うべき問題であるが、2つの乖離を同時に回避するためには日本語ベースの新しいプログラミング言語の開発が必要になり、既にExcelという表計算ソフトをベースとした仕事が定着している現状を考慮すると、新しい言語を開発して仕事のやり方自体を変更することは非現実的な方策である。そこで、現実アルゴリズムと記述アルゴリズムの乖離を小さくしつつ、自然言語と人工言語の乖離も小さくするように記述できるラッパー（wrapper）のような道具が適していると考えられる。ラッパーとは特定の機能を使いやすくしたり、他のプログラミング言語でも使えるようにしたりするものであるが、本研究では特定の機能を自然言語のように使える関数という意味で扱う。例えば、先に挙げた「B1セルにA列の1行目から10行目までの合計を求める」という例の場合、現実アルゴリズムでは合計範囲を設定し、合計値をセルへの代入、という手順になる。その処理を

```
Set 範囲 = 範囲設定(Cells(1, 1), Cells(10, 1))
Cells(1, 2) = 合計を求める(範囲)
```

というように記述できれば、現実アルゴリズムに近い形の記述アルゴリズムとなり「現実アルゴリズムと記述アルゴリズムの乖離」を回避できると考えられる。また、人工言語の文法に従った文であっても「範囲」や「合計」のように普段使われている単語を命令文として用いることによって、「自然言語と人工言語の乖離」もある程度回避できる。そこで、次章では実質的な重要度が高い機能を抽出したうえでラッパーの開発を行う。なお、本研究で開発するラッパーはあくまで対象企業において重要度が高い機能であり、必ずしも汎用性が高いものではない。一般的に抽象度を高くしてしまうと複雑性が増し、結局は「現実アルゴリズムと記述アルゴリズムの乖離」

が起きてしまうため、使用する組織で許容できる一般化を行うことが好ましいと考えられる。

3. EUD補助道具の開発

3.1 分析方法

本章ではラッパーの開発を行うが、企業や業務内容によって使用する命令文には偏りがある可能性がある。そこで、4章で実験を行う対象企業での実用的なラッパーを開発するために、4章の実証実験を行う企業で実際の作業をプログラムに置換する際に使用する文法や命令文の使用頻度に関する分析を行った。分析手順は、プログラミング経験者が対象企業の14名の従業員が担当している29種類の作業を実際にプログラムに落とし込み、実際に業務で使用できることを確認した上で、作成されたプログラムを単語ごとに分割し、その出現回数により、それらの命令文の使用頻度を集計した。使用頻度の多さは、必ずしも重要度と一致するものではないが、開発補助を行う上での目安になると考えられる。なお、算術演算子、代入演算子、等価比較演算子、論理演算子、繰り返しのためのFor-Next、Each、Do-While、Untilは基礎的な文法のため、集計からは除外した。

3.2 集計結果

集計結果を**表1**に示す。使用回数が最も多かった範囲設定は、合計の集計やグラフの作成など意思決定や判断基準となるデータの作成する際のプロセスで利用されたり、繰り返し処理の最終行を探すためなどの処理が含まれる。ファイル操作に関する項目はエクセルファイルの作成や他のファイルからデータを取得する際に利用されるもので、基幹システム外の業務プロセスでデータベースもない業務の場合はエクセルファイルのやり取りが行われることが多いため、よく使用される。集計は合計や条件を指定しての合計や件数のカウントを行うなどの内容である。書式はセルの罫線の設定やセル背景色の設定である。更新表示とは処理速度を高速化するために画面の再描画を一時的に停止するテクニックである。シートとはワークシートの追加や削除、移動などである。検索とは値の検索で様々な処理の最初に記入行を探したり、マスタテーブルから値を探す際に用いられる。抽出は条件に合致するデータの抽出（フィルタ機能）である。

表1　使用回数

単位：回

範囲設定	6777
ファイル操作	3738
集計	2305
書式	1222
更新表示	781
シート	671
検索	73
抽出	24

これらの集計結果を参考にして、重要な機能を単純な日本語で実行できる49の機能を持つラッパーを開発した。これらのラッパーは関数であり、一つのモジュールにまとめておき、そのモジュールをエクセルファイルにコピーすることで使用することができるようになっている。本研究の目的はラッパー自体の開発ではなく、あくまでEUDの実現の可能性を検討するものであるため、この開発したラッパーは常に使いやすいように改良を繰り返していき、開発時点での有用性の検討は行わないこととする。

4. 実証実験

4.1 実験期間と実験方法

本論文は企業のカイゼン文化の中で組織的EUDを実施し、それらが効果的かつ継続的に行われるかを検証するものである。これらの効果は1度の実験の学習効果で測定できる

ものでなければ、短期間の観察により継続性を考察できるものでもないことから、実証実験は5年という長期の実験期間を設定した。また、教育方法などの要因を固定して複数の要素を比較する実験は各要素の効果を明らかにできるが、貴重な人的資源を割き時間のかかる実験を企業の中で行うことは難しい。そこで、要因は固定せずに実施を成功させるために企業内で行われる様々な改善努力などのダイナミズムは参与観察の対象とすることとした。また、実験の対象企業は既にカイゼン活動を行なっている製造業で従業員数が約500名の企業A社とした。実験では対象企業で行われていたカイゼン活動のサイクルと実施方法に従って、1年度を1期として下記のようなステップでEUD活動を行った。

ステップ1：事前研修

約2時間の基礎文法等の研修行い、2章の分析結果に基づき下記のような設計方針を伝える。

- 変数は日本語を用いる
- データ処理は単純なデータ構造にした上で行う
- 開発したラッパーを使用し、できるだけ現実のエクセルでの操作と同じようなプログラムを記述する

ステップ2：カイゼンテーマの設定

事前研修で参加者が日々行っている事務作業の中でシステム化する作業をカイゼンテーマとして決定する。決定の際は上長も含めた話し合いを行い、教育面と職場での有用性を含めた検討を行う。

ステップ3：月に一度の個別フォロー

カイゼンテーマが決まってからは、自分でプログラムを作れるようになるまでは、月に一度のペースで外部講師が1時間程度の個別フォローを行う。個別研修以外では他のカイゼン活動と同様に個々で勤務時間中に時間を見つけて、学習やシステム開発に取り組む。尚、対象企業では「カイゼンは業務の内」という方針をとっていた。

ステップ4：カイゼン成果報告会

年度末には社長も参加するカイゼン成果報告会で1年間の改善成果を報告する。

参加者の募集は職場ごとに行い、上記のような1年単位のカイゼンプロセスを実施し、各年度初めに上司と話し合いの上で参加者を募った。募集の際は、できる限り単年度で終わらせず、継続的な参加をするようによびかけた。

評価指標はカイゼンの時間的効果、継続性、プログラミング能力の自己評価、相談環境の有無、補助ツールの主観的評価とした。時間的効果の測定と継続性は各年度終了後に調査を行い、その他の調査は5年間の実験終了後に追跡調査が可能な者にのみアンケート調査を行なった。5年間の実証実験に参加した人数は38名で実験終了後に追跡調査が可能であったのは27名であった。追跡調査が可能であった参加者27名の最終学歴の内訳とプログラミング学習経験は**表2、3**の通りである。

表2　参加者の最終学歴の内訳

単位：人

高　校　卒　業	18
高　専　卒　業	2
大学卒業（理系）	3
大学卒業（文系）	4

表3　参加者のプログラミング学習経験の内訳

単位：人

全くない	24
勉強した経験有 簡単なものであれば作ることができる	2
勉強した経験有 挫折した	1

4.2　実験結果

4.2.1　時間的効果

図1は1～5期目の1月当たりの作業時間短縮効果を集計し、各期の短縮効果を積み上げたグラフである。5期目には実証実験開始前と比べて累積で827時間/月の作業時間が削減されていることがわかる。これは約5人月分の就業時間であり、有効なシステム開発を行えたといえる。推移に着目してみると、1期目は27時間/月、2期目には100時間/月、3期目には392時間/月と単期で削減できる作業時間は実験を進めるにしたがって大きくなった。このことから、組織として着実にスキルアップができていると考えられる。4期目は3期目のような大きな時間的効果は得られていないが、これは外部講師から社内の自立運営に切り替えたタイミングであることが影響している。実証実験開始当初は社内で教える人材がいなかったため外部講師に頼る形となっていたが、本来は社内で自立運営をすることが望ましい。そのため、実験開始当初から自立運営ができる状況になれば外部講師に頼った運営から自立運営に切り替える予定であった。実験開始から3年が経過し、1期目から継続的に活動している者の中にはチューターとして学習者をサポートできる者が複数名いたため、4期目からは外部講師に頼らない自立的な運営に切り替え、学習者に社内チューターをつけて定期的に学習サポートをするという体制をとった。その際、次節でも述べるが、4期目からは初めて学習補助を行うチューターの負担を軽減するために、参加人数を意図的に減らしたことが時間的効果の低下につながっている。

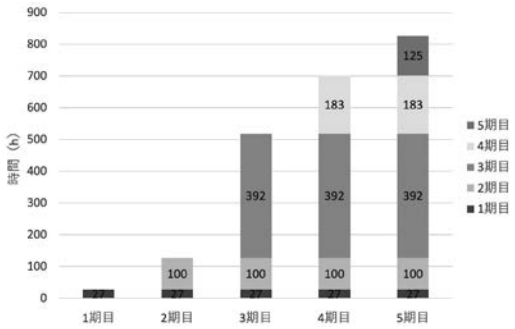

図1　月あたりの作業時間短縮効果

4.2.2　継続性

表4はEUD活動への期毎の新規参加人数と次年度も継続した人数を示したものである。5年間に及ぶ実証実験で38名が参加し、61%にあたる23名が2年以上の活動を続けることができた。しかし、継続的な参加をよびかけたにも関わらず、1期目の14名の参加者のうち、2期目も続けていた者は7名だけであった。半数の参加者が時間の都合や学習における過負荷を理由に続けることはできなかった。後の調査でわかったことだが継続しなかった参加者の中には本人との合意がないまま、上司の意思で参加した者も多かったことから、本人のモチベーションが影響していると考えられる。そのため、2年目以降は希望した者のみの参加とした。その結果、2期目以降の継続率は4期目を除き1期目より向上している。4期目から新規参加人数が減っていることに関しては前節で述べたように、運営を外部講師から社内の自立運営に切り替えたタイミングであり、意図的に参加者を減らしたためである。また、4期目の継続率が低下したことも自立運営による試行錯誤により、参加者が思ったようにスキルアップできず、モチベーションの低下につながったためであると考えられる。

表4　継続率

開始期	新規参加人数	次年度継続人数	継続率
1期	14	7	50%
2期	8	6	75%
3期	8	5	63%
4期	3	1	30%
5期	5	4	80%

4.2.3　プログラミング能力

5年間の実証実験終了後に追跡が可能であった27名に対して行なった調査で「どれくらいシステム開発ができるようになりましたか？」という質問には7名（26%）が「誰の助けも借りず開発できるようになった」と回答し、17名（63%）が「誰かの助けを借りればシステム開発ができるようになった」と回答した。一方で「全くシステムを開発できない」と回答したのは3名（11%）であった。この3名は全員活動歴が1年の参加者であり、ある程度の開発ができるようになるには1年以上の期間が必要であることが推察される。試験による確認を行なっていないため、各参加者のレベルは自己申告でしか確認はできないが、職場の中で助け合いながらカイゼン活動を行う文化的環境があれば89%がEUDを実践できる状態までスキルアップしたといえる。また、プログラミング能力が向上した者はチューターとして積極的に他の者の開発にも関わることで2章で述べたようなシステムの属人化を防ぐことができていると考えられる。

4.2.4　相談環境

5年間の実証実験終了後に追跡が可能であった27名に対して行なった調査で「日常的に悩みなどを相談ができる人は近くにいましたか？」という質問には22名（81%）が「いた」と回答し、5名（19%）が「いなかった」と回答した。敷地が広い工場では部署によって建屋が異なることもあり、職場が分散されることが原因の一つであると考えられる。また、「いなかった」と回答した5名のうち活動年数1年が3名、2年が2名であったことから先に活動をはじめていた従業員のコミュニティに参加しづらかったという可能性も考えられる。

相談環境が開発に与える影響を考察するために開発したシステムの数が5つ以上の人数を比較してみると、相談できる環境があった場合は22名中11名（50%）、相談できる環境がなかった場合は5名中2名（40%）であった。「いなかった」と回答した人数が5名と少なかったため単純な比較はできないが、悩みを相談できる人が物理的に近くにいる環境を作ることも組織的EUDを促進する要素の1つであると考えられる。実際にインタビューをしてみると、「近くに相談できる同僚がいて、やりやすかった」「テーマは違うが同じようにプログラミングの習得に取り組んでいる仲間がモチベーションの維持につながった」との意見も得られた。その他にもインタビューでは「自分が続けられたのはチューターの存在のおかげである」「チューターもプログラミングを全く知らないところから始めた人だったので、わからない人の気持ちになって教えてくれたので続けられた」など、チューターとの関係性がモチベーションになったとの意見も多くあった。以上のことから、わからないことを教えてくれるチューターのような存在や日常的に悩みなどを相談できる環境がないと組織的なEUDの実践は難しいと考えられる。実験を行なった企業ではカイゼン活動という文化が根付いていたため実践できたが、このような文化的背景がない企業の中での組織的なEUDを行うことは難しいであろう。

4.2.5　補助ツールの有効性

5年間の実証実験終了後に追跡が可能であった27名に対して行なった調査で「補助ツールがなくても、EUD活動を続けられたと思いますか？」という質問には16名（59%）が「続けられたと思わない」と回答しており、補助ツールはEUDの容易性と継続性に貢献したと考えられる。「わからない」という回答は8名（30%）であったが、補助ツールを使わずに開発した経験がないため、判断ができなかったと考えられる。また、3名（11%）は「続けられた」と回答したが、そのうち2名は最終学歴が大学（理系）でプロ

グラミングの学習経験のある者であった。

その他にインタビューでは「補助ツールを使うということで思考プロセスが統一化されるので、人が書いたプログラムでもパッと見ただけで大体のことが理解できる」と述べた回答者がいた。つまり、補助ツールという機能が限られたラッパーを使うことを前提にアルゴリズムを考えると、全員のプログラムが似たようなプログラムになってくるというのである。この点に関しては、今後更なる検証が必要であろう。

5. まとめ

本研究では、個人的EUDの問題点を明らかにしたうえで、カイゼン文化の文脈の中で組織的EUDの実現性について実証実験を通して検討した。

個人的EUDよって開発されたシステムは、組織の業務プロセスの中に取り込まれ（個人プロセスの標準化)、それらのシステムのメンテナンスが開発者に依存し（システムの属人化)、開発者がその職場に留まり続けることが前提（前提条件の永続性）であるという問題点を先送りしながら運用されていた。

そこで、組織的にEUDを実施するためにプログラムの理解に関する分析を行った結果、プログラムの理解と解読を阻害する要因として

1）現実作業とプログラム間の単語の置き換え現象
2）現実アルゴリズムと記述アルゴリズムの乖離
3）自然言語と人工言語の乖離
4）2、3のトレードオフ現象
5）データ構造の複雑性

が抽出された。これらの問題を解決するにあたり、ルール作りで対応できない部分、難解な部分は平易な日本語で呼び出せる補助ツール（ラッパー）を開発することによって解決を試みた。

5年間に及ぶ実証実験をカイゼン活動の一環として行った結果、1ヶ月当たり827時間の作業時間が短縮された。また、5年間で38名がこの活動に参加し、61％にあたる23名が2年以上の活動を続けることができた。実験終了後に追跡が可能であった27名のうち26％が誰の助けも借りず開発できるようになったと回答し、63％が誰かの助けを借りればシステム開発ができるようになったと回答した。

以上のことから、日本企業にあるカイゼン文化を利用し、プログラムの書き方のルールや道具の工夫、サポート体制の工夫により、組織的EUDは可能であると考えられる。実験を行なった企業では実験終了後もカイゼン活動の中で組織的EUDは続けられており、他人が開発したシステムの変更も行われていた。今後は組織的EUDによって開発されたシステムの管理の必要性や引き継ぎ文書のあり方なども議論の対象にしていくべきであろう。

参考文献

［1］ 独立行政法人情報処理推進機構：IT人材白書2020
［2］ 栗田るみ子，宮寺庸造，横山節雄：EUC（End User Computing)・EUD（End User Developing）を目指した情報基礎教育カリキュラムの開発，情報教育シンポジウム2001論文集
［3］ 佐々木康浩：RPA（Robotic Process Automation）の可能性，2017年春季全国研究発表大会　p201-204（2017）
［4］ 内木哲也：持続可能な情報システムへのデザインアプローチ，経営情報学会2010年秋季全国研究発表大会要旨集（2010）

［5］ Burrell G. and Morgan G., Sociological Paradigms and Organisational Analysis, Heinemann, 1979（鎌田伸一，他訳『組織理論のパラダイム―機能主義の分析枠組―』千倉書房，1986）
［6］ M.Imai“Kaizen: The Key To Japan's Competitive Success”, McGraw-Hill（1986）
［7］ 川瀬武志：「IE問題の解決」，日刊興業新聞社（1995）
［8］ 石川馨：「日本的品質管理―TQCとは何か」，日科技連（1984）
［9］ QCサークル本部：“QCサークル綱領”，日科技連出版社（1970）
［10］ 松井賚夫：QCサークル活動の組織効率、従業員モチベーションの改善効果，産業・組織心理学研究，Vol.1，No.1，21 ～ 27，（1987）