

# ユースケースの使い方に関する提案

児玉公信<sup>†</sup>

情報システムの機能要求を記述するために、ユースケースが一般的に使われるようになったが、その記述内容や粒度はまちまちであり、その意義も効果も十分に理解された上で使われているとは言えない。本報告では、行為者の仕事を「もの・こと分析」でいう「基本変換」としてとらえ、ユースケースを「手段のもの」として定義することを提案する。これによって、仕事の流れとユースケースおよびユースケース記述の位置づけが明らかになり、記述すべき内容を確定できる。

## A Proposal of How to Use Usecase

Kiminobu Kodama<sup>†</sup>

Usecase has recently become a standard method of writing functional requirements for information systems. However, different styles and granularities are being used in different usecases because of misunderstandings regarding the meanings and benefits of the usecase method. This report proposes to treat the actions of an actor as the basic transformation of a "MONO-KOTO" analysis of Nakamura and to write usecase as a "THING of MEANS," a function to support the action. This treatment enables usecase writers to understand the usage of usecase and flow of actions while ensuring the provision of a good description of usecase.

### 1. はじめに

近年、情報システムの機能要求の記述方法として、ユースケースが標準的に使われるようになった。しかし、その記述内容や粒度はまちまちであり、その意義も効果も十分に納得された上で使われているとは言えない。また、ユースケースを導く方法論についても、さまざまなアプローチが提案されている<sup>[1][2][3][4][5]</sup>が、それらは必ずしも使いやすしいものとは言えない。

本報告では、ユースケースを行為者(actor)の仕事を支援する機能要求の記述であると定義したうえで、ユースケースがどのように記述されるべきか、それは原要求<sup>[6]</sup>からどのように導かれるべきかを提案する。さらに、ここから派生して、機能とオブジェクトの関係を記述するDFD(Data Flow Diagram)の有効性を見直す。

### 2. ユースケースとは何か

Jacobson<sup>[1]</sup>は、1980年代の中頃、それまでのシステム開発の経験に基に、ユースケースの原型を考えた。これは元来、仕事の場で情報システムの機能を

「使用する状況(situation of usage)」を述べることで情報システムの要求を取り出す手法であった。彼は、これを、あえて非形式的に書こうとした。しかし、このことの利点と欠点が、今に至るまで、さまざまな議論と解釈の幅をもたらした。

#### 2.1. ユースケースの定義

UML 2.1.1 の仕様書(Superstructure)<sup>[7]</sup>によると、ユースケースとは、“a means for specifying required usages of a system. Typically, they are used to capture the requirements of a system, that is, what a system is supposed to do.”とされている。つまり、ユースケースとは、システムが提供すべき要求(仕様)の記述であり、それを使用する行為者(actor)が暗黙に存在する。ここでいう「システム」は、前もって定まっているものではなく、ユースケースの記述を通して構想される情報システムである。

ユースケースには、その要求仕様を表す名前が付けられるが、それは、それが果たすべきことがらを表すように「OをVする」と表現される。この表現の主語は一般に省略されるが、これまで、行為者であると理解されてきた<sup>[2][3][4]</sup>。つまり、行為者が行うこととされていた。しかし、上のUMLの定義からすると、ユースケース名の主語は「システム」でなければならない

<sup>†</sup> 株式会社エクサ, EXA Corporation

い。これは、ユースケースが、行為者の仕事(action)を支援する手段であることからすれば当然のことであったが、それが十分に理解されてこなかった。

以下では、行為者の仕事と、仕事の達成を支援する手段としてのユースケースを分離することについて述べ、あるべきユースケースの設計手順を提案する。

## 2.2. さまざまな使われ方

ユースケースはさまざまな目的、水準、形式で使われている。その理由は、“use case”という言葉の意味が曖昧だったことと、(情報システムの)要求記述と(ソフトウェアの)要件記述が分離されて認識されない中では、ユースケースの意義が理解されなかったことによると思われる。

Cockburn<sup>[10]</sup>は、さまざまなユースケースの使われ方を、目的レベルと設計対象の軸で整理した。目的レベルは、概略(雲)、要約(凧)、ユーザ目的(海面)、サブ機能(魚)、詳細(貝)という 5 水準に区分され、目的レベルごとに扱うべき設計対象が、組織、システム、構成要素と、異なる。

本報告で提案する機能要求記述の水準は「海面レベル」とする。これは、Cockburn<sup>[10]</sup>によると、ユーザインタフェースの設計をするものでなく、施主が読んで理解できるように、複雑過ぎないこと、行為者とシステムの対話の記述をできるだけ少ないステップにまとめることが求められる。

表 1 ユースケース記述のテンプレート

|          |                                                               |
|----------|---------------------------------------------------------------|
| ユースケース名  | 行為者の行為を支援する機能名。<br>「(システムは)OをVする」という文にする。                     |
| 行為者(アクタ) | このユースケースを使用する、顧客または施主の代理人のロール。                                |
| 目的       | この機能の目的を書く。「～するため」と記述し、主語は「顧客」または「施主」。                        |
| 事前条件     | このユースケースを実行する前の「はじめの状態」を書く。                                   |
| 事後条件     | このユースケースが実行された後の「終わりの状態」を書く。                                  |
| 基本系列     | 行為者とシステムとの典型的な対話を順序づけて書く。各ステップに番号をつけて、主語を「アクタ」または「システム」と明示する。 |
| 代替系列     | 例外的な対話を書く。                                                    |
| 備考       | 何を書いてもよい。特に、非機能要求を書く。                                         |
| シナリオ     | 記述内容のデバッグのため、このユースケースを使用する仕事のインスタンスを挙げる。                      |



図 1 もの・こと図<sup>[8]</sup>

このような記述のテンプレートとして、表 1 を用いることにする。

## 3. 仕事の設計

仕事、機能、および手段の関係を考察するために、「もの・こと」分析<sup>[8]</sup>を取り上げる。

### 3.1. 「もの・こと」分析

「もの・こと」分析によれば、仕事とは、“行わなければならないことを、体や頭を使って行うこと”である。つまり、仕事は人の行為である。その行為は、対象に働きかけることで、「はじめの状態」を「終わりの状態」に変化させる。ものづくりの文脈であれば、“もの”が対象であり、「はじめのもの」を「おわりのもの」に変換する。サービスの文脈であれば、“人”が対象であり、たとえば食堂というシステムは「おなかを空かせた人」を「満腹した人」に変換する。また、情報システムの文脈であれば、対象は“情報”であり、たとえば、「情報がない」状態から「情報がある」状態に変換する。このような状況を「もの・こと」図を書いて表す。図 1 は、ジュースをコップに注ぐ仕事を表す「もの・こと」図の例<sup>[8]</sup>である。

「もの・こと」分析は、システム思考に基づく、ワークデザイン<sup>[9]</sup>手法の一つであるが、目標(機能)展開を行わずに、理想的な「終わりの状態」を考えて、それを実現するために必要な「はじめの状態」、基本変換を考える点が特徴である。

「もの・こと」図を書く手順は次のとおりである。

- ① 終わりの状態を決める
- ② はじめの状態を決める
- ③ 残りものを決める
- ④ 基本変換を明確化する
- ⑤ 手段のものを決める

このとき、最終状態として、真に得たいもの、それに適うものを想定し、目的ではない余計な負荷を除去する。これを「要のもの」といい、そのような基本変

換を「要の変化」という。変化には制約条件がつき、終わりの状態には品質要件がある。品質要件を満足するために、はじめの状態にもさまざまな条件が必要であり、これを良品要件と呼ぶ。

### 3.2. 手段の定義

たとえば、図 1 で示した「栓つきのビンに入ったジュースをコップに注ぐ」仕事では、栓を抜いてジュースを注ぐことが「要の変化」であり、必須の機能である。この機能を実現する手段を「手段のもの」と呼ぶ。これは「ワークヘッド」とも呼ばれ、対象に接触して働く「もの」である。

機能を実現するための手段には多くの候補があり、制約条件を満足し、かつ最小のシステムロードで仕事を達成できるものを選択する。図 1 の例では、自分の手を使う、他人の手を使う、ロボットを使うなどの手段が考えられ、通常は、自分の手を使う方法が選択される。

このような仕事と手段の関係をモデル化するために、基本変換を仕事の手順と見て、活動図(Activity diagram)で表記し、「手段のもの」を、仕事を達成するためにシステムが提供すべき機能の実現と見て、ユースケースで表す。

### 3.3. 活動図とユースケース

仕事の手順を表す活動図は、まず基本変換の手順を行為としてそのまま書き、それにオブジェクトフローを追記する。選択された手段によって手順が制約されることがあるので、必要に応じて手順を細分化または統合を行う。次に、行為ごとに行為者を決め、行為を支援する手段をノートシンボルに書く。

このような手段をソフトウェアで実現しようとするのがユースケースである。一方、それを他の人に割り当てようとするれば、ジョブ・ディスクリプションとなる。

図 2 は、図 1 の基本変換を活動図で表したものである。図 2 左は基本変換をそのまま活動図にしたも

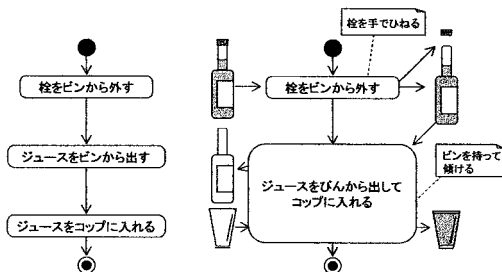


図 2 活動図の例

表 2 栓をビンから外すためのユースケース記述

|          |                                                                                                     |
|----------|-----------------------------------------------------------------------------------------------------|
| ユースケース名  | (システムは)栓を手でひねる                                                                                      |
| 行為者(アクタ) | ジュースを飲みたい人                                                                                          |
| 目的       | 栓をビンから外したい。                                                                                         |
| 事前条件     | 栓が外れていない。                                                                                           |
| 事後条件     | 栓が外れている。                                                                                            |
| 基本系列     | ①行為者がこれを起動する。<br>②「手(システム)」はビンの口がどこにあるか行為者に聞く。<br>③行為者はその位置を提示する。<br>④「手(システム)」は栓を握って、外れるまで左回りにひねる。 |
| 代替系列     | A.基本系列④で、栓が固くてひねることができない場合、..<br>B.基本系列④で、ひねってもひねっても栓が抜けない場合、..                                     |
| 備考       | ①栓が王冠またはコルクの場合は、別のユースケースを使う。                                                                        |
| シナリオ     | 省略                                                                                                  |

の、図 2 右はオブジェクトフローを追記した結果、手を使って注ぐという手段を考えた例である。このとき、「ジュースをビンから出す」行為と、出された「ジュースをコップに入れる」行為を一緒に行った方が良いと考えて、行為を統合した。また、これらの行為をそれぞれ別の行為者に割り当てる意味はないと判断した。そして、それぞれの行為に「栓を手でひねる」と「ビンを持って傾ける」という手段をノートで示した。

表 2 は、「栓を手でひねる」をユースケース記述として書いたものである。これは本来、ソフトウェアで実現する手段ではないので、奇異な感じがする。

こう考えてくると、仕事を達成するための手段の設計は、仕事で定義されてはじめて可能であることがわかる。これまでのユースケース設計の手法は、仕事のあり方をあまり意識していなかった。むしろ、ユースケースそのものを仕事の設計とみなす傾向すらあった。それは、多くのユースケース名の主語が行為者であったことからわかる。

### 3.4. 行為とユースケースの関係

UML には、行為とユースケースの関係を直接記述する記法はないので、該当する行為にノートシンボルを付け、そこにキーワード「usecase」を付してユースケース名を書くことにする。

一般に、1 つの行為を 0 個以上のユースケースが支援し、1 つのユースケースは 1 つ以上の行為を支援する。さらに、1 つのユースケースが記述されると、その手段が、もともと想定されていなかった行為で利用できることに気がついて、仕事の手順を変化させ

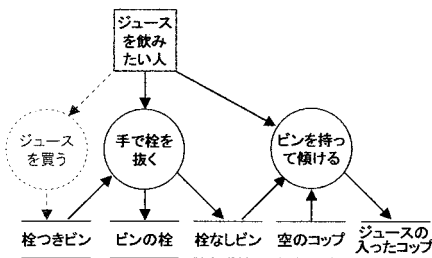


図3 DFDの例

ることもある。ユースケースは仕事の設計から導き出されるが、逆に、仕事の設計がユースケース記述から影響を受けることで、仕事の設計と手段の設計は相互作用する。運用段階においても、これと同様のことが言える。

### 3.5. DFDとBCE図

このように、行為と手段とが分離されると、オブジェクト(および型)と手段との対応関係を示す適当なUMLの図法がないことに気がつく。活動図は、行為とオブジェクトの関係を記述するものであり、ユースケース図は行為者と手段の使用の関係を記述し、システムが備える手段全体を一覧するものである。

実は、DFD(Data Flow Diagram)<sup>[11]</sup>は、手段(プロセス)とオブジェクト(データストア)との関係を記述する図である。これも誤解されて、行為とデータの関係、あるいは行為の手順を記述するように誤用された例をよく見る。DFDは、恣意的な機能分解に使用しない限り、有効な図法である。特に、オブジェクトの生成漏れを推定できる点は優れている。この点について次に述べる。

図3は、「手で栓を抜く」と「ピンを持って傾ける」の手段をDFDで書き表した例である。これによると、データストア「栓つきビン」と「空のコップ」は取り出される(オブジェクトの参照)だけで、補給(つまり、オブジェクトの生成)がないことに気がつく。ここから、ジュースを飲むためには、「ジュースを入手する」、「コップを入手する」という機能が必要であることがわかる。また、データストア「ピンの栓」や「ジュースの入ったコップ」は書き込まれる(生成される)だけで、使われない(参照されない)。これを使う機能は何かを検討する。

これと同等の記法として、BCE(Boundary Controller Entity)図がある。これはUMLのコミュニケーション図をアイコン表記したものである。Controllerオブジェクトをプロセス、Entityオブジェクトをデータストアと見て、上のようなチェックに利用できる。

## 4. 仮想事例

具体的な活動図とユースケースの記述について、仮想事例を用いて、従来との違いを説明する。

システムインテグレーションサービスを生業とするA社では、技術系社員のスキルレベルを毎年評価して、技術者の戦略的な育成、およびプロジェクトチームの構築の最適化に役立てている。しかし、そのスキル体系が最近の市場動向にフィットしないという現場からの声が挙がり、スキル管理システムを見直すことにした。

以下に、その設計結果の一部を、従来型の業務フローとユースケース記述で示す。これを本報告で提案する記述形式に書き直し、対比することで本質的な違いを議論する。

### 4.1. 業務フロー

従来タイプの業務フローを図4に、本報告で提案する業務フローを図5に示す。

まず、行為の表現について見る。従来タイプでは、「獲得スキルおよびレベルを入力する」というように、

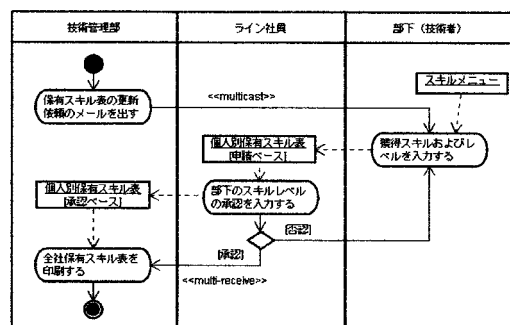


図4 従来型の業務フロー(活動図)

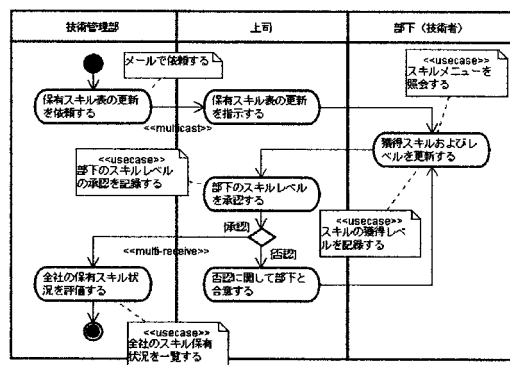


図5 提案する業務フロー(活動図)

表 3 従来型のユースケース記述の例

|                                                            |
|------------------------------------------------------------|
| ユースケース名: 部下のスキルレベルを承認する                                    |
| アクタ: ライン社員                                                 |
| アクタの目的: ライン社員が部下の未承認スキルレベルに対して承認を行う。                       |
| 事前条件: 次の条件を満たしていること。<br>・アクタはアクセス認証・認可済みである。               |
| 事後条件(成功時): 次の条件を満たしていること。<br>・未承認スキルレベルがすべて承認できている。        |
| 基本系列                                                       |
| 1. アクタが本ユースケースを起動する。                                       |
| 2. アクタが、自分のすべての部下の参照を要求する。                                 |
| 3. システムが、アクタのすべての部下の氏名を提供する。                               |
| 4. アクタが、部下を 1 名指定して、対象の部下のすべてのスキル要素とそのスキルレベル、承認状態の参照を要求する。 |
| 5. システムが、指定された部下に対するすべてのスキル要素のスキルレベル、承認状態を提供する。            |
| 6. アクタが、指定した未承認スキルレベル(複数件選択可能)の承認を要求する。                    |
| 7. システムが、指定された未承認スキルレベルの承認を確定してよいか、アクタへ確認する。               |
| 8. アクタが、指定された未承認スキルレベルの承認確定をシステムへ要求する。                     |
| 9. システムが、指定された未承認スキルレベルを承認し、承認したことをアクタへ伝達する。               |
| 代替系列: 省略                                                   |

行為者の仕事というよりは、情報システムの操作が書かれている。そして、これがユースケース名となっている。つまり、この業務フローはユースケースの実行順を書いているのである。

このような立場では、情報システムを使用しない行為が表記されない。逆に、「○○システム」なるレーンが作られて、それと他のレーンとの間で頻繁に遷移の矢印が行き交うことで、情報システムとの関わりが強調される。あるいは、オブジェクトフローが書き込まれることによって、情報システムと関わりが強調される。オブジェクトフローが書かれること自体は問題ではないが、施主が業務フローを理解するためには、伝票やカンバンといったビュのオブジェクトが受け渡されるように書いた方がよい。

提案する業務フロー(図 5)では、行為の目的や行為者の責任が強調される。たとえば、「部下のスキルレベルを承認する」のが目的であり、従来タイプの「承認を入力する」ことが目的とは考えない。ソフトウェアを設計する以前の業務フローは、このような仕事の流れの記述としてとどめておき、その後、行為を支援するユースケースを考えて、あとからノートを付け

表 4 提案するユースケース記述の例

|                                                                 |
|-----------------------------------------------------------------|
| ユースケース名: 部下のスキルレベルの承認を記録する                                      |
| アクタ: 上司                                                         |
| ユースケースの目的: 本人が申請したスキルレベルを、全社基準に整合させたい。                          |
| 事前条件: スキル表に未承認・否認のエントリがある。                                      |
| 事後条件: スキル表に未承認・否認のエントリがない。                                      |
| 基本系列                                                            |
| 1. アクタが本ユースケースを起動する。                                            |
| 2. システムは、承認しようとする部下の識別の提示を促す。                                   |
| 3. アクタは、それを提示する。                                                |
| 4. システムは、該当する部下が申請したスキル表の全エントリを表示する。表示内容は{スキル名、レベル、承認状態}のくりかえし。 |
| 5. アクタは、エントリごとに承認状態を設定する。設定できる承認状態は承認か否認のどちらか。                  |
| 6. システムは設定された内容を記録する。                                           |
| 代替系列: 省略                                                        |
| 備考:                                                             |
| 1. 本ユースケースを起動する前に、上司であることの認証が必要。                                |
| 2. 部下の識別の提示に当たっては、一覧表示から選択させても良い。                               |
| 3. 承認状態は、未承認、承認、否認の 3 状態である。                                    |

てそれを示すことが適当と思われる。

#### 4.2. ユースケース記述

従来タイプのユースケース記述を表 3 に、本報告で提案するユースケース記述を表 4 に示す。

従来型では、ユースケース名が手段名ではなく、「(ライン社員は) 部下のスキルレベルを承認する」のように行為者の行為名となっている。また、目的欄にアクタの目的を書こうとしているが、顧客以外のアクタが主体的な目的を持つことはない。そのため、アクタの仕事の内容が目的として書かれている。事前条件には、このユースケースを起動する前提条件だけが書かれている。これもよく見られる誤りである。基本系列では、2~4 で部下の一覧表を示して、そこから対象を選ばせるようなユーザインタフェースを設定している。

提案するユースケース記述では、上司が「部下のスキルレベルを承認する」ために、ユースケース「(システムは) 部下のスキルレベルの承認を記録する」を使う。目的欄には、行為の目的を書くべきであり、このユースケースが達成されたときに、何が満足されるかを書く。事前条件と事後条件は、このユースケースが行う変換の、はじめの状態と終わりの状態を記述する。基本系列は、Cockburn のガイドに沿って、複



雑すぎないように、少ないステップでまとめている。

このユースケースを起動する前提条件、および対象とする部下の識別をどのように与えるかについては要求仕様とはしないで、備考に書くことでその判断を施工設計の段階に意識的に遅らせている。

## 5. おわりに

ユースケースの定義から始めて、システムが提供する機能とは何かを考え、「もの・こと」分析を通じて、仕事と手段を分離して設計することを考えた。

仮想事例に対して、仕事と手段を分離しない従来タイプと、本報告で提案した新しいタイプによる業務フローとユースケース記述の例を示して比較した。これらは一見大きく違わない。しかし、ユースケース名の主語を誰にするか、また目的をどうとらえるかは、単なる表記上の趣味の問題ではなく、仕事の設計をどう考えるべきかの問題なのである。

初期のユースケース記述を見る限り、Jacobson が意識していたとは思えないが、実は、ユースケース記述とソフトシステムズ方法論(SSM)<sup>[12]</sup>のCATWOEとはよく似ている。実際、ユースケース記述の中には、明示的に行為者(Actor)、事前条件と事後条件による変換(Transformation)、目的(Weltanschauung)があり、暗黙に顧客(Customer)と施主(Owner)が含まれている。環境(Environment)としては、そのユースケースを構成要素とする上位システムを想定すればよい。つまり、ユースケースは一つの小さなシステムなのだ。このような観点からも、ユースケース記述のあるべき形を追求したい。

ただし、このような表記上のルールの再整備が、情報システムのよさに対してどの程度の効果を持つかはわからない。これ以外にも、情報システムのよさを左右する要因がたくさんあるからである。たとえば、ソフトウェアの品質はもとより、できあがった情報システムを利用者がうまく使ってくれるかといった、最終的には組織の人間関係にまで波及するような要因もある。

そこまで立ち入ることは避けて、少なくとも、業務フローやユースケースは、施主が読んで分かる設計文書とすることを目指すべきである。施主自らが問題を指摘できる、そして心からの納得が得られるような設計文書の記述レベルを、まずは確保したい。

## 参考文献

- [1] Christerson, M., Jacobson, I., and Övergaard, G., “Object-Oriented Software Engineering: A Use Case Driven Approach,” ACM Press, 1992, 西岡ほか訳, 「オブジェクト指向ソフトウェア工学 OOSE—use-case によるアプローチ」, トップラン, 1995.
- [2] Jaconson, I., Ericsson, M. and Jacobson, A.: “The Object Advantage,” Addison-Wesley, 1995, 本位田監訳: 「ビジネスオブジェクト—ユースケースによる企業変革」, トップラン, 1996.
- [3] Schneider, G. and Winters, J. P.: “Applying Use Cases: A Practical Guide,” Addison-Wesley, 1998, 羽生田訳: 「ユースケースの適用—実践ガイド」, ピアソンエデュケーション, 2000.
- [4] Wiegers, K. E.: “More about Software Requirements: Thorny Issues and Practical Advice,” Microsoft Press, 2006. 宗監訳: 「要求開発と要求管理」, 日経 BP ソフトプレス, 2006.
- [5] Jacobson, I., Booch, G., and Rumbaugh, J.: “The Unified Software Development Process,” Addison-Wesley, 2000. 日本ラショナルソフトウェア訳: 「UMLによる統一ソフトウェア開発プロセス」, 翔泳社, 2000.
- [6] 児玉公信: 「情報システムサイクルと原要求の記述」, 日本情報経営学会誌, 28(2), 77-87, 2007.
- [7] OMG: “Unified Modeling Language: Superstructure.” UML2.1.1\_SS\_07-02-03.pdf, 2007
- [8] 中村善太郎: 「シンプルな仕事の構想法」, 日刊工業新聞, 1992.
- [9] 五百井清右衛門, 黒須誠治, 平野雅章: 「システム思考とシステム技術」, 白桃書房, 1997.
- [10] Cockburn, A.: “Writing Effective Use Cases,” Addison-Wesley, 2001. ウルシステムズ株式会社監訳, 「ユースケース実践ガイド—効果的なユースケースの書き方」, 翔泳社, 2001.
- [11] DeMarco, T.: “Controlling Software Projects: Management Measurement and Estimation,” Yourdon Press, 1982.
- [12] Checkland, P. B.: “Systems Thinking, Systems Practice,” John Wiley & Sons, 1981. 高原, 中野監訳, 『新しいシステムアプローチ』, オーム社, 1985.