

ソフトウェア仕様記述過程分析のための基礎的実験

池 克俊†

海谷治彦†

佐伯元司†

本間 学‡

†東京工業大学 工学部 電気電子工学科 ‡産能短期大学 能率科

本論文では、人間がソフトウェアの仕様を作成する過程の履歴をとり、その分析を行なった結果について述べる。仕様の作成過程は、まず作ろうとするソフトウェアを理解し、そのメンタルモデルを作り上げる過程と、それをもとに仕様を記述する過程の2つに分けられる。我々は、ビデオカメラを用いて、被験者が各生成物を作成していく過程を記録した。具体的には、電車の切符の自動販売機を例にとり、被験者がこのシステムを理解するために書いたメモ、自然言語仕様、状態遷移図、データフロー図を、どのように作成していったかの履歴をとった。これらの記録を分析することにより、メンタルモデル、状態遷移図、データフロー図作成の流れを抽出する。これら3つの流れを種々の観点から比較、検討し、類似点と相違点を分析する。この実験の目的は、人間にとって自然な思考過程を反映していると思われるメンタルモデル作成過程と、状態遷移図やデータフロー図といった形式的仕様作成過程の相違点を調べることににより、形式的仕様作成過程に含まれる、人間にとって不自然な作業を洗いだすことである。これにより、人間にとって自然な仕様作成過程、およびその支援ツール開発への足掛かりとすることができる。

Empirical Studies of Software Specification Process

Katsutoshi Ike†

Haruhiko Kaiya†

Motoshi Saeki†

Manabu Honma‡

†Tokyo Institute of Technology

‡Sanno Junior College

This paper presents empirical studies of human process for constructing software specification. We consider this process can be divided into two sub-processes; one is the process for forming a mental model of the software system, and another is the process for constructing the specification. Using videotapes, we recorded these processes in which subjects constructed their products, e.g. memos, natural-language specifications, state transition diagrams (STD), and data flow diagrams (DFD) of a ticket vending machine system. Our experimental results contain four general flows for the mental model, the natural-language specification, the state transition diagram, and the data flow diagram processes. We compare and discuss these processes from multiple viewpoints to clarify the analogy and the difference among them. Our aim is to study the method of improving the process for constructing formal specification such as STD and DFD, and of letting it come close to the human understanding process of the system. Because the process similar to the human understanding process is suitable for constructing comprehensive formal specification of high quality.

1 はじめに

ソフトウェアの仕様化や設計の方法論や技法について種々の研究がなされ[1],[2],[3],[4]、実用化への努力もなされている。しかし、仕様化や設計過程は人的要因にかなりの部分を依存しているため、これらの技法を用いたとしても、人間の行なわなければならない高度で複雑な作業は未解決のまま残されている。また、これらの技法が人間にとってより複雑で煩雑な作業を強いている危険性もある。つまり、その技法が必ずしも人間の特性に適した作業のみを提供しているとは限らない。そこで、この問題を解決するためには、人間にとって自然な作業を明らかにし、従来の技法での作業との比較を行い、人間にとって不自然な作業が含まれているかどうかを検討しなければならない。これによって、人間にとって作業を行い易い方法論、およびその支援ツールの開発が行えると思われる。

本稿では、人間の仕様作成過程に関する実験とその分析結果について述べる。Kant[5]、Soloway[6]、Curtis[7]は、プログラム設計過程における人的要因を研究しているが、本研究では、上記の目的達成のため、彼らとは違う実験方法を用いた。仕様の作成過程は、まず作ろうとするソフトウェアを理解し、そのメンタルモデルを作りあげる過程と、それを基にある技法に基づいて仕様を記述する過程の2つに分けられる。我々は、Newell[8]の実験手法を応用し、ビデオカメラを用いて被験者が各プロダクトを作成していく作業の様子を記録した。具体的には、電車の切符の自動販売機を例にとり、被験者がこのシステムを理解するために書いたメモ、自然言語仕様、状態遷移図、データフロー図をどのように作成して行ったかの履歴をとった。これらの記録を分析することにより、メンタルモデル、状態遷移図、データフロー図作成時の思考のフローを抽出する。これら3つのフローを種々の観点から比較、検討し、類似点と相違点を分析する。メンタルモデル作成過程は、人間にとって自然な思考過程を反映していると思われる。

2 実験、分析の手順

2.1 仕様作成過程の概要

人間の仕様化過程は、大まかに以下の2つの過程に分けることができる。

1. 問題を理解する：対象システムのメンタルモデルの形成

最初に仕様記述者は、ソフトウェアシステムの内部動作モデルを形成する[6]。このモデルは、様々なメンタルシミュレーションに用いられる。メンタルシミュレーションとは、具体的なデータを与え、頭の中でシステムのメンタルモデルを実行することである。メンタルシミュレーションを通じて、記述者は、ある入力システムに与えられたら、システムがどのような応答をするかを知ることができる。この過程では、システムのラフなメンタルモデルと、いくつかのメンタルシミュレーションの結果を得る。

2. 実際の形式的仕様の記述

次に仕様記述者は、メンタルモデルとシミュレーション結果から、必要なシステムの情報を抽出し、システムの抽象モデルを形成する。記述者は、形式的仕様記述言語を用い、抽出された情報を記述する。記述者が抽出しなければならない情報は、形式的言語の概念モデルに依存していることになる。例えば、もし、形式言語として状態遷移図を用いるのであれば、メンタルモデルとシミュレーション結果から、状態と遷移を表すコンポーネントを抽出する作業過程となる。

2.2 仕様作成過程分析の着眼点

上記の仕様作成過程を分析するために、人間の思考の流れに着目する。本研究で行った実験では、被験者のふるまい——被験者が実験で要求されている最終生成物を完成させるために、何かを考えている様

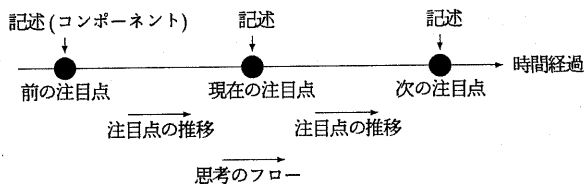


図1: 思考のフロー

子など——をビデオで録画した。何かを書いたり、言ったりする被験者のふるまいは、被験者自身の思考過程を表しているものとする。ビデオを用いた理由は、実験によって被験者の思考が妨げられたり、誘導されたりすることのないようにするためである。

実験の後、ビデオから被験者の考えの記述の順序、それに要した時間を抽出した。記述をするのに要した時間は、彼の思考に要した時間と一致すると仮定する。もし、ビデオで不明な点があれば、被験者にそれを見せ、その時何を考えていたかを質問した。

被験者の記述順序を思考の順序とみなすことは問題があるかもしれない。被験者がある考えを記述している間に、他のことを考えているかもしれないからである。例えば、被験者が同時に2つのことを考えていたとしても、それらを逐次的にしか記述することができないため、順序づけして記述してしまう。もし、記述の順序を局所的に詳細に解析するのであれば、逐次性と同時性の違いが現れるかもしれない。しかし、全体的に解析すれば、僅かな違いしかないと思われる。つまり、構文的かつ概念的に接近している記述を1つのまとまりを持ったコンポーネントとして捉え区分し、これを分析対象とする。記述物からコンポーネントを抽出するにあたって、記述時間の隔たりも重要な手がかりとなる。

図1に、本実験での思考過程の捉え方および分析法を示す。被験者は、あるコンポーネントを記述し終えるまで、そこに注目しているとす。被験者が次の新しいコンポーネントを記述するまでの間、彼が何を思考していたのかを分析する。このとき、なぜ被験者が、前のコンポーネントから現在のコンポーネントへ思考を推移させたかという、因果関係に注目する。因果関係が自明でない場合は、被験者へのインタビューを行う。この因果関係をもとに、被験者のプリミティブな思考活動を抽出し、分類整理する。作業は、このようなプリミティブな思考活動の列として表現される。全ての被験者についてこれを行ない、共通に見られるプリミティブな思考活動とその出現列を抽出し、フロー図として表現する。これらのフロー図は、仕様作成過程での人間の共通なふるまいを表しており、それにより、メンタルモデルの形成や形式仕様の記述過程を抽象化して表現することができ、相互の比較分析を行うことができる。

もし、記述されたコンポーネント間に何の関係も見い出せなかった場合は、それらのコンポーネントの間に思考のギャップがあると考ええる。

3 実験方法

前節の方針に従って、2つの過程、各々について実験を行なった。

実験1 システムを理解する：メンタルモデルの形成

被験者に作成するシステムの名称のみを与え、そのシステムについて考えたことを書き出してもらった。記述様式は言葉、文、図等全て、被験者にまかせた。

実験2 形式的仕様の記述

実験2-1 最初に、実験1で得られた生成物を同じ被験者に与え、自然言語でシステムの仕様を記述してもらった。このシステムの仕様には、「他人に読めるもの」という条件を課した。

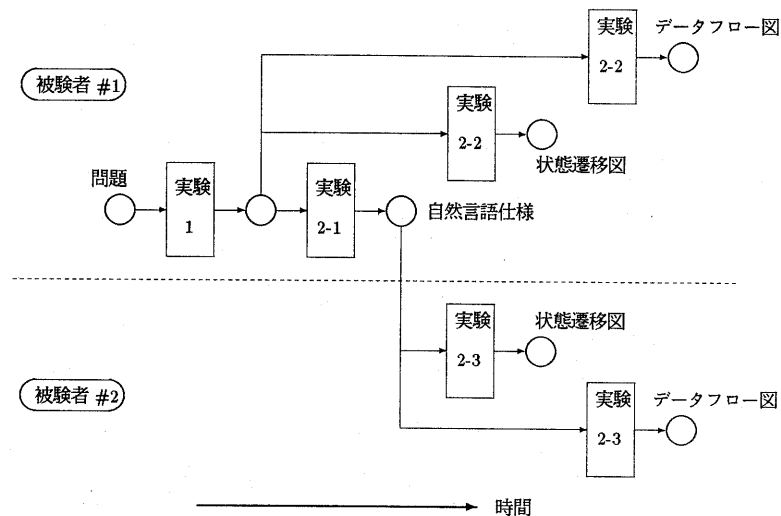


図 2: 実験の流れ

実験 2-2 実験 2-1 の後、実験 1 の生成物を同じ被験者に与え、状態遷移図とデータフロー図を記述してもらった。被験者は、データフロー図より状態遷移図の方が経験を積んでいた。最初に、状態遷移図、次にデータフロー図を記述してもらった。

実験 2-3 実験 1 から 2-2 の被験者とは違う 3 人に、実験 2-1 で得られた自然言語仕様を与え、状態遷移図とデータフロー図を記述してもらった。実験 2-2 と同じように、被験者は状態遷移図の方が、データフロー図より経験を積んでいた。

図 2 は、どのように実験が行なわれたかを示している。各実験から得られたメモ、自然言語仕様、状態遷移図、データフロー図、それらを記述するのに要した時間、記述順序などを解析した。実験で用いた問題は、切符の自動販売機である。被験者はいずれも、ユーザという立場から、このシステムの使用経験をかなり積んでいる。

4 実験結果と分析

4.1 メンタルモデル形成過程：[実験 1]

図 3 に、自動販売機のメンタルモデル形成過程で観られた、思考のフローを示す。

1. ユーザへの必要な応答 (すなわち出力) の抽出
被験者は、例えば「切符」などのユーザへ返す応答を列挙した。これは、自動販売機システムと、そのユーザの境界を特定しているものと思われる。
2. 応答を得るための入力シーケンスを見つける
被験者は、ユーザが最終応答である「切符」を得るための入力シーケンス (操作シーケンス)「お金の投入 → ボタンを押す → 切符の発行、おつりの返却」を書き出した。このシーケンスは、自動販売機のユーザモデルである。
3. 入力シーケンスの詳細化
被験者は順番に、入力シーケンスから 1 つシーケンスを取り出し、詳細化している。例えば、シーケンス「お金の投入 → ボタンを押す → 切符の発行、おつりの返却」に注目していたとすると、「お金」、「ボタン」、「切符」、「おつり」というオブジェクトを次々と詳細化していった。注目しているオブジェクトでシー

ケンスに分岐を見つけた時は、そのことを覚えておき、後でそれらの分岐を詳細化した。この詳細化の過程で被験者は、注目しているオブジェクトから、それと関連のあるシステムのふるまいに注目点を移し、そのふるまいが含んでいる部分シーケンスを取り出して詳細化していった。そして、その詳細化されたシーケンスが、全体のシーケンスに対して一貫性があるかどうか、インテグレーションテストを行なった。

4. メンタルモデルのシミュレーション、レビュー

被験者は、詳細化した全てのシーケンスを、メンタルモデルにインテグレートした。ここで、いくつかのデータを用い、メンタルモデルのシミュレーションを行なった。

大まかにいえば、自動販売機のようなリアクティブシステム [4] の場合、仕様記述者は、通常の入力シーケンスに従い、そのシステムのふるまいからメンタルモデルを形成するように思われる。

4.2 自然言語による仕様記述過程：[実験 2-1]

以下に、実験 2-1 で得られた切符の自動販売機の自然言語仕様を示す。

切符の自動販売機

販売する切符は 120 ～ 800 円とする。
お金が投入されたら、まず、その金種を区別する。
扱う金種は、10, 50, 100, 500 円各硬貨と 1000 円札とする。
投入された金額を記憶し、その投入金は保持しておく。
そして、投入金額を表示する。
また、投入された金額以下の切符ボタンを点灯する。
ここで、もし、投入金額以上のボタン、つまり、点灯していないボタンが押されても無視する。
点灯しているボタンが押された場合、
まず、投入口を閉じ、新たに金が投入されないようにする。
そして、押されたボタンに対応する切符を発行し、さらにおつりを計算して、それぞれを出す。
ここで、投入された金は、別のボックスに入れる。
また、お金が投入される前に、毎回、残りのつり銭をチェックする。
残りが、

- 合計が 880 円以下となる。

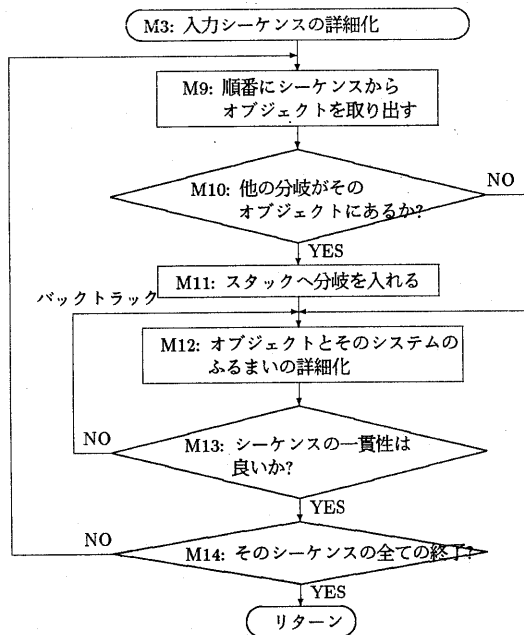
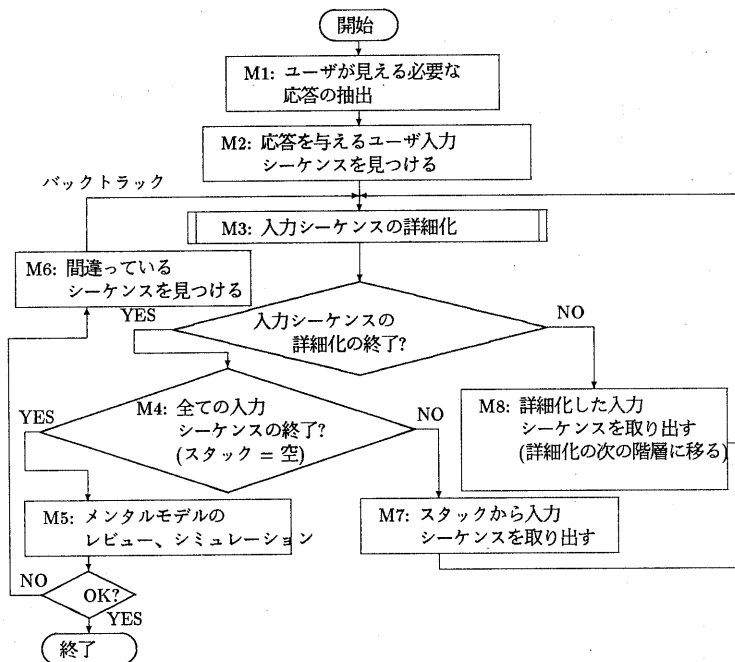


図 3: メンタルモデル形成の一般的な流れ

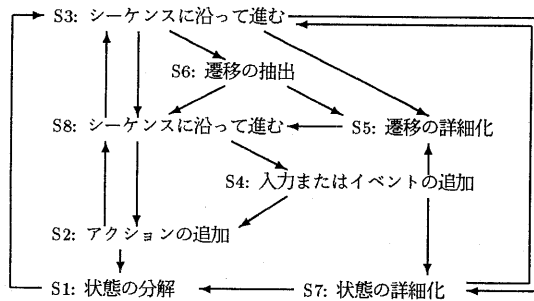


図 4: 状態遷移図におけるフロー

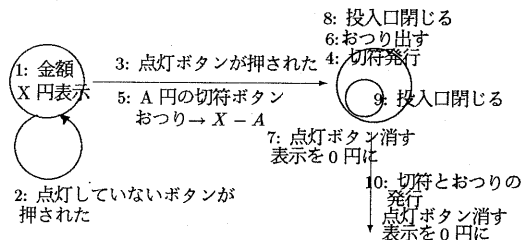


図 5: 状態遷移図の作成過程

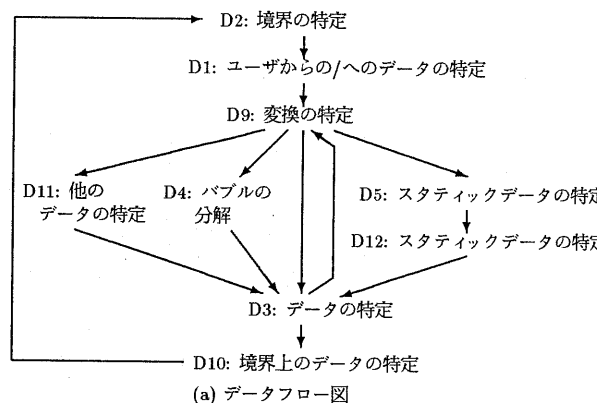
- 50 円硬貨がなくなり、10 円硬貨が 8 枚以下となる。
- 10 円硬貨が 3 枚以下となる。

のいずれかの条件を満たすとき、「販売中止」と表示して、投入口を閉じる。

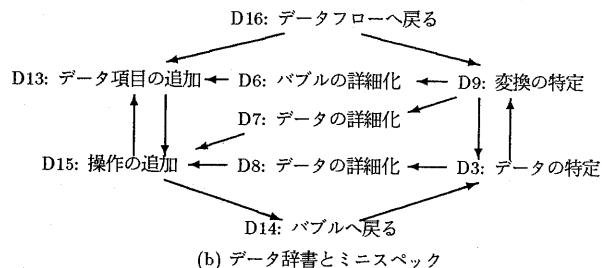
また、投入してから、ボタンが押されるまでに、「取り消しボタン」が押されたときは、保持してある投入金を下から出す。

4.3 状態遷移図記述過程：[実験 2-2, 2-3]

状態遷移図は、構文的に状態と遷移から成っている。状態遷移図を記述するためには、状態と遷移を見つけ出さなくてはならない。つまり、状態遷移図を作成するための注目点は、状態と遷移になる。本論文では、状態遷移図に関して、4 人の被験者に対して実験を行なった。それらの結果について、共通の注目点の推移を特定する。表 1 に、状態遷移図過程における共通の推移のパターンを示す。もし、推移に因果関係を見い出せない時は、その推移は不規則なものとして仮定する。S1 は、1 つの状態をいくつかの状態に分解して考えたことを示す。現在の注目点は、分解される状態であり、次の注目点は分解の結果、新しく作られる状態になる。最もよく観られた思考活動は、元の状態から状態遷移のシーケンスを追う S3 と S8 であった。もし被験者が、現在の状態がいくつかの出力を出すことに気付けば、S6 を行なうであろう。図 4 に、状態遷移図の記述に関する思考活動のフローを示す。S1 と S7 は、STATEMATE[4] の OR 分解のように、状態を階層的に分解するアクティビティである。図 5 は、状態遷移図の作成過程における思考活動の流れの一部を表したもので、これは客からの投入金額を表示し、客が切符ボタンを押すと、切符とおつりを出す部分である。



(a) データフロー図



(b) データ辞書とミニスペック

図 6: データフロー図過程におけるフロー

被験者の思考は、金額が表示されている状態 1 から次に起こるイベント (入力) へと移っている。この思考活動は、表 1 の S3 である。次に、被験者は点灯されていないボタンが押された場合の遷移から、点灯されているボタンが押された場合の遷移に思考を移した (S6)。次状態は 4: 「切符の発行」(を行っている) 状態として捉えたが、5 の入力データの追加 (押されたボタンの金額など) や 7 の遷移時のアクションにより、6: 「おつりを出」している、8: 「投入口閉じ」ているという 2 つの概念を追加し、この状態を詳細化していった (S7)。その後、この状態を分解し、「投入口閉じる」のみの 1 つの状態にし、他を次状態への遷移時のアクションとした。

以下に、総括的な状態遷移図過程の特徴を述べる。

- 初期段階で、被験者は、システムのふるまいとユーザのふるまい (または入力) の区別をなかなかつけられず、混乱しているようであった。被験者は、ほとんどの場合、自動販売機システムの記述において出てきたキーワードを最初に状態として捉えていた。
- 被験者は、状態を自動販売機が行なうアクションと捉えていた。状態に付加されたラベルは、「点灯」、「合計の表示」、「切符の発行」というような動詞であった。さらに、各ユーザ入力について、対応する遷移が存在した。
- 被験者は、遷移シーケンスに沿って進むよりは、1 つに状態において全ての遷移分岐を調べる傾向があった。図 4 において、パス「S3 → S6 → S8」は「S3 → S8」に優先して行なわれた。

4.4 データフロー図記述過程：[実験 2-2, 2-3]

状態遷移図過程と同じように、データフロー図の構文要素を、データフロー図記述の注目点とする。表 2 に、4 つの実験結果から得られた注目点の推移の共通なパターンを示す。図 6 (a) にバブル、外界、ファイル、矢印のみから成るデータフロー図の記述のフローを示す。ミニスペックとデータ辞書は、図 6 (b) に示されているような流れに沿って記述された。システムとユーザの境界を特定する活動は、図 6

表 1: 状態遷移図過程における注目点の推移

現在の注目点	次の注目点	思考活動	ラベル
状態	状態	状態の分解	S1
	状態	状態へのアクションの追加	S2
	出力遷移	入力またはイベントシーケンスに沿って進む	S3
	入力遷移	必要な入力またはイベントを、状態または遷移へ追加	S4
遷移	遷移	遷移の詳細化	S5
	遷移	同じグループでの遷移の抽出 (場合分け)	S6
	状態	状態の詳細化	S7
	状態	入力またはイベントシーケンスに沿って進む	S8

表 2: データフロー図における注目点の推移

現在の注目点	次の注目点	思考活動	ラベル
外界	データフロー	ユーザから与えられる/ユーザが受け取るデータの特 定	D1
	外界	ユーザとシステムの境界の特 定	D2
バブル	データフロー	バブルにより与えられた/バブルが必要なデータの特 定	D3
	バブル	バブルの分解	D4
	ファイル	バブルに必要なスタティックデータの特 定	D5
	ミニスベック	バブルの詳細化	D6
	データ辞書	バブルにより与えられた/バブルに必要なデータの詳 細化	D7
データフロー	データ辞書	データフロー上のデータの詳 細化	D8
	バブル	データの変換の特 定	D9
	外界	ユーザとシステムの境界に あるデータの特 定	D10
	データフロー	同じグループの他のデータ の特 定	D11
ファイル	データフロー	スタティックデータの特 定	D12
ミニスベック	データ辞書	ミニスベックで必要な新 しいデータ項目の追 加	D13
	バブル	一致するバブルへ 戻 る	D14
データ辞書	ミニスベック	データを与える/データ を使う新しい操作の追 加	D15
	データフロー	一致するデータフローへ 戻 る	D16

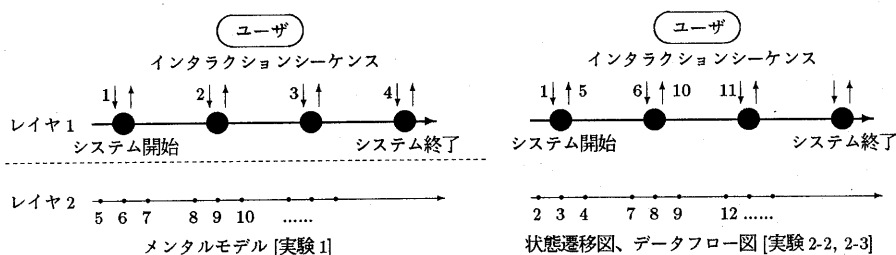


図 7: メンタルモデル過程と状態遷移図、データフロー図過程の比較

表 3: システムの分析を自分で行った場合の特徴

	有利	不利
1 階層的記述	✓	
2 どんな仕様化方法にも柔軟に対応	✓	
3 記述に系統性がない		✓
4 バックトラック、デッドロックが少ない	✓	
5 詳細な記述にならない		✓

(a) の D1, D2, D10 において行なっている。D9 と D3 のループにおいて、被験者は、データフローに沿って、データの出所となっているバブルから、そのデータの行き先となっているバブルへと進んでいった。パス「D13 ⇄ D15」は、ミニスベックとデータ辞書の間のインタラクションを示しており、これは何回も実験で観られた。

以下に、データフロー図過程の特徴を示す。

- 被験者は時々、意識せずに、バブルをそれらが実行される順序に従ってつないでいった。
- データフロー図の方法論では、状態遷移図とは異なり、システムとユーザの境界を明らかにしなければいけない。しかし、完全な特定は、初期段階においては難しく、時にはミニスベックを記述している時(つまり D6)に行なわれた。
- 被験者は、データフローを全体的に注目するというのではなく、バブルにより与えられる、または、必要とされるデータの特定に注目することが多かった。

5 議論

5.1 メンタルモデル過程と状態遷移図、データフロー図過程の比較: [実験 1 と実験 2-2, 2-3]

メンタルモデルを形成する過程と、仕様を作成する過程を比較することは重要である。ここでは、その比較について述べる。

状態遷移図過程とデータフロー図過程は、部分的にはメンタルモデル過程と似ている。しかし、全体の構造はメンタルモデル過程とは異なる。主な違いは、詳細化の階層に関してである。メンタルモデル過程では、被験者は最初の M1 と M2 の段階でユーザモデルを形成し、次に、ユーザモデルを詳細化し発展させ、自分のメンタルモデルを形成していった。つまり、メンタルモデルの形成には、2つの階層があると思われる。ユーザモデルの形成により、システムの開始から終了までのふるまい全体を掴むことが可能になる。一方、状態遷移図、データフロー図過程では、そのような階層は観られなかった。これらの過程では、自然言語仕様に従い、被験者は、最初からシステムの詳細な要素を記述していった。つまり、システム全体のふるまいを掴む前に、彼らは、部分的なシステムのふるまいを詳細化していったことになる。図 7 に、これらの流れの違いを示す。状態遷移図過程における、この詳細化のアクティビティの流れは、S1, S5, S7 から S3 ⇄ S8 へ戻るパスに観られた。データフロー図過程では、被験者は、全ての境界を特定し、システムの全体像を掴む前に、D9 から D4 へ行き、バブルを分解していった。

ユーザモデルを先に形成することにより、解り易い形式的仕様を作成することができると考えられる。このように、ユーザモデルを形成する方法論を、積極的に、状態遷移図、データフロー図過程に導入するべきであろう。

5.2 システムを自分で解析する場合とそうでない場合の比較: [実験 2-2 と実験 2-3]

最後に、実験 2-2 と 2-3 の比較を行なう。表 3 に、目立った違いを示す。これらの違いは、状態遷移図とデータフロー図の記述の前に、被

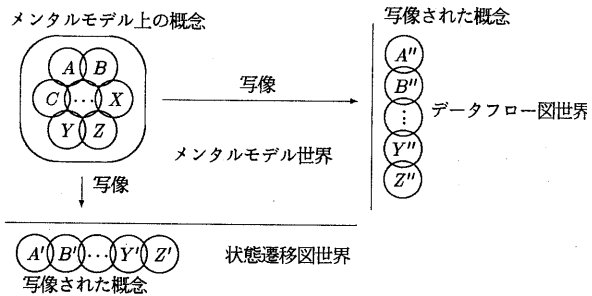


図 8: メンタルモデルからの写像される概念

験者が、自分でシステムの分析を行なったかどうかである。自分で思考錯誤を繰り返し、システムの分析を行なった被験者は、そうでない被験者よりも、より安定したメンタルモデルを形成していると考えられる。それに対し、実験 2-3 の被験者は、自然言語仕様による情報を優先し、そのみからモデル形成を行なうため、欠落した部分のあるモデルができると考えられる。以下に自分でシステムの分析を行なった場合の利点、欠点を、表 3 に従い示す。

最初に、分析を行なった被験者は、状態遷移図、データフロー図共に、システムを階層的に記述している。これは、メンタルモデルにより、彼がシステムのふるまいを 1 つのまとまったものとして捉えていたためである。一方、自然言語仕様のみを与えられた被験者は、システムを階層的に記述しなかった、つまり 1 つの層にのみしか記述しなかった。彼らは、最初にシステムの 1 つの要素にしか注目していなかったため、多くの要素を持つシステム全体を見渡すことができなかった [9]。

次に、図 8 に示すように、メンタルモデルが安定していれば、メンタルモデルの要素を、形式的仕様記述言語の様々な概念に写像することができる。安定したメンタルモデルを持つ仕様記述者は、どんな形式的仕様記述手法にも柔軟に対応できるということである。例えば、B をメンタルモデル世界の概念「お金の挿入」とすれば、写像の後、状態遷移図世界の B' は入力イベント「お金を挿入する」となり、データフロー図世界の B'' は「お金とスロット」となる。それに対し、実験 2-3 の被験者は、自然言語仕様の構造から得られたモデルしか形成しなかったため、柔軟性に欠けると考えられる。

3 番目に、自分で解析を行なった被験者は、システムの入力シーケンスに合わせることなく形式的仕様を記述することができた。また彼は、メンタルモデルを用いシステム全体を眺めることができたため、彼が注目することのできたシステムの任意の部分から記述を始めた。しかし、彼は記述しなければいけない、いくつかのシステムの要素を記述しないことがよくあった。5 番目の項目にあるように、彼は、実験 2-3 の被験者が行なったような詳細な記述は行なわなかった。これは、人間は知っていることよりも、知らないことに興味を示すという事実 [6] に一致する。実験 2-3 の被験者は、実験 2-1 で得られた自然言語仕様の順序に従い記述を行なった。

4 番目に、自分で分析を行なった被験者は、システム全体を眺めることができたため、異なる階層にあるシステムの要素を有機的に結び付けることができた。一方、そうでない被験者は、全体を眺めるということはず、常に 1 つの要素に注目していた。彼らが、1 つの要素に注目している時は、その要素の周辺や他の要素とのインタフェースを考えておらず、後でそれら要素を結合する際に正確さを欠いた記述になってしまった。特に、要素の結合において、安定したメンタルモデルを形成していない記述者は、バックトラックや行きづまりが多くなる。

結局、両方とも利点、欠点がある。

6 今後の課題

問題点および今後の課題として、以下の項目が考えられる。

1. 実験方法について
人間の思考に掛かる時間は1時間が限度であると考えられるので、現実の大規模システム開発における過程を分析する方法を考えなければいけない。
2. 階層化について
実験2-3の被験者は、システムを階層的に記述しなかったが、これは問題が簡単だった理由も考えられるので、この点を明らかにする。
3. 場合分けについて
被験者は、場合分けがあるところでは、同時に全ての場合を気付いたようだが、多くの場合分けがある時に、どうするかを明らかにする。
4. 記述の順序について
被験者は、与えられた自然言語仕様に沿ってシステムを記述していたが、自然言語仕様の書き方が、どれくらい思考に影響を及ぼすかを明らかにする。
5. 他の仕様記述手法について
構文的に今回と異なる要素を持つ仕様記述手法、例えば、JSD法、代数的仕様記述などについて、同様な実験を行なう。

参考文献

- [1] G. Booch. Object-oriented development. *IEEE Trans. on Soft. Eng.*, 12(2): 211-221, 1986.
- [2] T. DeMarco. *Structured Analysis and System Specification*. Yourdon Press, 1978.
- [3] M. A. Jackson. *Principles of Program Design*. Prentice-Hall, 1983.
- [4] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, and A. Shtul-Trauring. STATEMATE: A working environment for the development of complex reactive systems. In *Proc. of 10th ICSE*, pp. 396-406, 1988.
- [5] E. Kant and A. Newell. Problem solving techniques for the design of algorithms. *Information Processing and Management*, 28(1): 97-118, 1984.
- [6] B. Adelson and E. Soloway. The role of domain experience in software design. *IEEE Trans. on Soft. Eng.*, 11(11): 1351-1360, 1985.
- [7] B. Curtis. Psychological research on software development. In Skwirzynski (ed.), *Software System Design Methods*, pp. 155-184, Springer-Verlag, 1986.
- [8] A. Newell and H. Simon. *Human Problem Solving*. Prentice-Hall, 1972.
- [9] G. Miller. The magical number seven, plus or minus two—some limits on our capacity for processing information. *The Psychological Review*, 63(2): 81-97, 1956.
- [10] E. Soloway and K. Ehrlich. Empirical studies of programming knowledge. *IEEE Trans. on Soft. Eng.*, 10(10): 595-609, 1984.
- [11] 佐藤, 内田, 門田, 山下. 設計過程における人間の思考過程の分析. 第36回情報処理学会全国大会論文集2R-6, 1988.