情報システムと社会環境 83－12
（２００３．３．１４）

情報システムデザインにおける動作シミュレーション環境 DVLM に関する研究

李雪梅† 内木哲也‡
埼玉大学大学院文化科学研究科

情報システムをデザインする際には、分析者、開発者、利用者等デザインに関わる人全員の共通理解が不可欠である。今日、システムの開発では共通理解を得るためのコミュニケーション手段としてシステムの機能や構造を抽象的、静的な図式モデルで表現したドキュメントがよく用いられている。分析者や開発者などのシステムデザインの専門家はこれらのドキュメントを分析することにより、システムの動作を容易に理解できる。しかし、訓練を受けていない一般の利用者にとっては、これらのドキュメントだけからシステムの動きやその全貌を理解することは困難である。そのため、これらのドキュメントだけではシステムのデザインに不可欠な利用者に共通理解を求めるのが困難なばかりか、利用者との相互理解さえ得ることができない。このような問題を解決するため、本論文では、人間と組織の活動を動的な図式モデルとして示すことができる動作シミュレーション環境 DVLM(Dynamic & Visual Logic Modeling)を提案する。DVLM ではシステムが扱うデータの流れを可視化することにより、利用者が情報システムの挙動を含めた全体像を把握しやすくすると共に、望まれるシステムデザインを開発者に分かりやすいモデルとして容易に表現できるように支援する。DVLM により、利用者を含めたデザインに関わる人全員のシステムに対する共通認識とコンセプトを統一し、情報システムのデザインプロセスを支援することができる。

DVLM : Dynamic & Visual Logic Modeling Environment for Information System Design

Xuemei Li† Tetsuya Uchiki‡
Graduate School of Cultural Science, Saitama University

Common understanding by all the members concerned with the design such as analysts, developers, and users is indispensable with information system design. Nowadays, the documents are often represented by abstract and static diagrammatic models in order to obtain common understanding, which means the work as communication tool. By analyzing the documents, it is possible to understand action of system among analysts and developers. However it is difficult to understand for the ordinary users who have not trained. Therefore, common understanding by all members concerned with information system designing cannot obtain easily only by these static documents. To solve the problem, we proposed a simulation environment DVLM (Dynamic & Visual Logic Modeling) which can show data flowing in information systems as a dynamic diagrammatic model. By visualizing action of a system, users can recognize the whole information system image intuitively while the system desired can be easily represented as DFD diagrams used by SSADM for developers. DVLM can bring developers and users the common recognition and the concept to the system, and support process to information system design.

E-mail: † sgl1055@post.saitama-u.ac.jp
‡ uchiki@post.saitama-u.ac.jp

1. はじめに

情報システムをデザインする際には、分析者、開発者、そして利用者の全員が情報システムに対する共通理解を持つことが必要である。システム開発に関わる人全ての共通認識なくしては、その共通の解である情報システム像を得ることはできない。しかし、利用者と開発者との共通認識の形成が困難なため、双方に満足がいくシステムが開発できないばかりか、多くの問題が発生している[1]。

今日、システムの開発では共通理解を得るためのコミュニケーション手段としてシステムの機能や構造を抽象的、静的な図式モデルで表現したドキュメントがよく用いられている。分析者や開発者などのシステムデザインの専門家はこれらのドキュメントを分析することにより、システムの動作を容易に理解できる。しかし、訓練を受けていない一般の利用者にとっては、これらのドキュメントだけからシステムの動きやその全貌を理解することは困難である。そのため、これらのドキュメントだけではシステムのデザインに不可欠な利用者に共通理解を求めるのが困難なばかりか、利用者との相互理解さえ得ることができない。

2. DVLM の提案

情報システムの動作をシステムデザインの専門家でない一般の利用者にも理解できるようにするためには、これまでの静的なモデルに内包されている形のない、目に見えない情報処理プロセスを、視覚化し、動的に見せることが重要である。つまり、システムがやり取りする情報や情報に対する処理を分かるようにし、その変化を動的に見せることができれば、デザインされたシステムの動作をより直観的に把握できると考えられる。しかも、ブロックを組み立てるような多くの人に分かりやすい簡単な操作によってシステムの動作を定義できれば、一般の利用者がシステム動作モデルを自分で定義したり、直接修正したりしながら、システムデザインに直接関与することもできるようになると考えられる。

そこで、本論文では、システムの中でやり取りされる情報を動的に表示することで、一般の利用者にもシステムの挙動が直観的に分かりやすいシステムのデザイン環境 DVLM を提案する。DVLM 環境では、図式で描かれた動作（動的なブロックの組み合わせ）を実際にシミュレートして人間と組織の活動を示すことができる。これにより、情報システムのデザインに関わる利用者がそのシステムを直観的に理解できるものと考えられる。このような環境により、開発者と利用者とのシステムに対する共通認識とコンセプトを統一でき、情報システムのデザインプロセスが円滑に進むよう支援することができると考えられる。

3. DVLM の表記方法

3.1 動作記述方法の検討

情報システムのモデル化にあたっては、システムの挙動を分かりやすく表わせ、分析できるような表現が求められる。この点に着目すると、作業の流れではなく、システムがやり取りする情報およびその情報に対する処理を明示化するデータ主導型の表現方法が適していると考えられる。そこで、DVLM では DFD 表現を用いて構造的にシステムを分析し、設計する SSADM の表記方法[2]を用いることとした。

3.2 TileDesigner

DFD モデルは、比較的簡単な図式モデルであるが、多くの矢印が行き互うため、書面および画面上で記述したり、修正したりするのが容易ではない。しかし、このようなモデルをブロックを組み立てるような感覚で記述することができれば、一般の利用者にも受け入れられやすく、かつ記述や修正も容易になると考えられる。

このような環境として視覚的プログラミング環境が多く提案および実現されている。中でもプログラミングの初心者である小中学生でもサッカーロボットのような複雑な挙動を記述でき、シミュレートできる環境である TileDesigner[3]が注目される。

TileDesigner は機能タイルを並べることでロボットの挙動を視覚的にプログラミングできる環境であり、埼玉大学教育学部野村泰朗助教授により開発された。TileDesigner では機能タイルの操作やロボットの動作を直接的に表現することができる。そのため、TileDesigner を用いることで、コンピュータ

プログラミングやロボットに詳しくない小中学生のような利用者でも簡単にロボットの動作モデルを表現可能である。その上、教育者と受講者との間のコミュニケーションツールとしても非常に優れており、TileDesigner 上のモデルを通して双方の意思疎通が教育現場で円滑に行われている。また、ロボットの挙動をシミュレートする機能があるため、モデリングした図形表現を動的かつ視覚的に確認できる。

以上のような特徴は、DVLM に望まれる環境であるため、TileDesigner を基礎として DVLM の表記方法をデザインすることとした。

3.3 具体的な表記方法

図1のようにTileDesignerはタイルを並べてプロセスの流れを示す。これに対して、DVLM ではデータの流れを見せるため、このままの表記方法を用いることができない。そこで、DFD モデルの記述に適した形を以下のようにデザインした。

図 1　TileDesigner の表記方法

DFD モデルのデータ源・行き先、機能長方形、データ庫などをオブジェクトタイルとして扱う。楕円、機能長方形、データ庫の記述がそれぞれ三種類のオブジェクトタイル ACTOR、ACTION、DATABASE に対応する。基本構造は図 2 のように定義する。

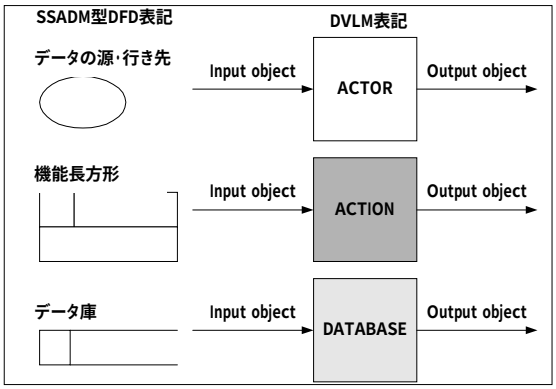

図 2　SSADM 型 DFD 表記と DVLM の表記の対応

オブジェクトタイルは Input object あるいは Output object のような入出力オブジェクトを少なくとも一つは持ち、それらを二つずつ以上持つことも可能である。例えば図 3 のように書ける。

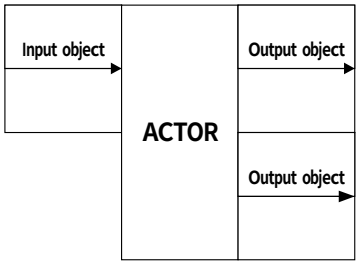

図 3　DVLM の表記方法

DFD モデルにおいて流れるデータを連結タイルとして表現した。連結タイルはオブジェクトタイル同士を連結するため使用する。連結タイル上にデータが流れる方向を矢印で示し、オブジェクトタイル間で授受される入出力オブジェクトを記述することができる。連結タイルから一定の依存関係を持つ入出力オブジェクトを受け取ったオブジェクトタイルは、その状況に応じて新たな入出力オブジェクトを出力するのである。

4. DVLM 環境の実装

4.1 DVLM 環境の画面

図 4　DVLM 環境の画面

DVLM のプラットフォームとしては、TileDesigner の実行環境を採用した。DVLM

環境画面は TileDesigner の画面構成を基礎としているが、図 4 に示したように、画面を大きく以下の 4 つのフィールドで構成した。

① タイルフィールド
② キャンバス
③ 依存関係設定フィールド
④ シミュレーションフィールド

タイルフィールドには三種類のオブジェクトタイル（ACTOR、ACTION、DATABASE）と連結タイル[注1]が用意されている。キャンバスはタイルを並べて動作を記述するワークスペースがあり、タイルを追加したり、移動したり、削除したり、連結したりすることができる。現在、４つのワークスペースを提供しており、記述の相違によるシステムの動作の違いを検討したり、並列に動作するシステムを４つまで独立に記述することができる[注2]。依存関係フィールドには連結タイルの依存関係（実行条件）を設定する。あるタイルにおける依存関係は状況に対応させて複数設定可能である。作成されたシステムの動作は、シミュレーションの進行に従ってシミュレーションフィールドに動的に表示される。

4.2 基本機能

格子状に区切られたキャンバス上にタイルを貼り付けることでシステムのモデルを記述することができる。具体的には、SSADM 型 DFD モデルを DVLM の表記方法に従って、個々のオブジェクトタイルを連結タイルで連結しながら次々と並べて記述することとなる。また、DVLM では単に DFD を記すだけでなく、その動作をシミュレートしながら記述することができるため、インタラクティブに動作をチェックしながらモデルを詳細化していくことが可能である。DVLM 環境では、以下のような基本機能が用意されている。

タイルの追加　用意されているタイルをマウスでキャンバス上にドラッグすることで、キャンバス上にタイルを置くことができる。タイルはキャンバス上で必要に応じて横長、縦長に伸ばすことができる。なお、タイルの枚数には制限がなく、キャンバス上にスペースがある限り、任意の種類のタイルを任意の枚数だけ配置することができる。

タイルの移動・削除　キャンバス上のタイルは、上述の条件でスペースが確保されている限り、他の位置に移動できる。また、ツールバーの切り取りアイコンにより削除することができる。

タイルの依存関係設定　キャンバス上の連結タイルの依存関係を設定することができる。複数の依存関係設定することも可能である。これらの設定はオブジェクトタイルの実行条件としてデータベースに組み入れる。例えば、「利用者の貸出情報」と「本の貸出情報」があるときに実行される連結タイルの依存関係は図 5 のようにして指定する。

オブジェクトタイルの連結　Input object を受け入れるオブジェクトタイルと Output object を送出するオブジェクトタイルに接続された連結タイルは、自動的に連結状態となり、各オブジェクトの連結タイルと related object の情報が更新される。しかし、オブジェクトタイル同士を隣り合わせに配置しても連結状態とはならない。例えば図 6 の「貸出受付カウンター」と「書庫から取出す」は連結状態ではない。

論理チェック　依存関係に従って、キャンバス上の記述の論理的な整合性をチェックすることができる。論理的に整合が取れていると、SSADM 型 DFD 画面がシミュレーションフィールドに表示される。論理的に整合が取れていない場合は、キャンバス上の記述が正しくないと判断され、SSADM 型 DFD 画面は表示されず、非整合箇所が示されることとなる。

シミュレータ機能　作成した DFD モデルの挙動を、設定した初期条件に応じ、動的に見せることができる。つまり、シミュレーションの進行に応じてやり取り情報や情報に対する処理および変化を動的に見せることができるのである。

モデルの保存と参照　キャンバス上で作成したシステムモデルは、それぞれディスクに保存することが、必要に応じてキャンバス上

注1 以下本論文では、オブジェクトタイルと連結タイルを「タイル」と呼ぶことにする。
注2 現段階ではワークスペース間でデータを授受するための明示的な機能についてはデザインされていない。

で参照したり、修正したりすることができる。

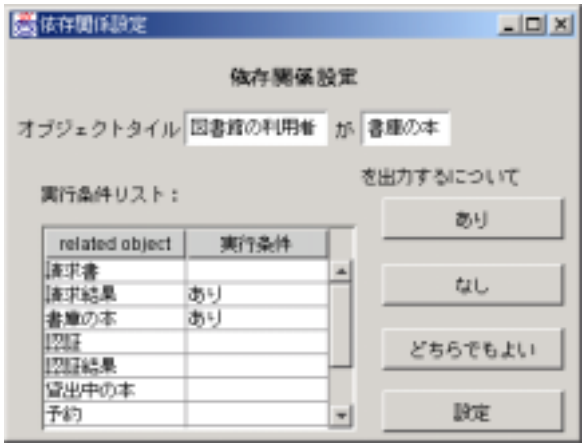

図 5　依存関係設定画面

4.3　シミュレーション動作の仕組み

キャンバス上の記述は論理的に整合が取れていると、SSADM 型 DFD 画面がシミュレーションフィールドに表示される。シミュレータを起動するためには、あらかじめ“ 初期条件設定 ” を行っておく必要がある。その後、シミュレーションは以下のような手順に従って動作する。

①全てのオブジェクトが実行状態となる。

②実行状態となると、現在の状態で実行可能な依存関係を持つ Output object が出力される。依存関係を満たすプロセスが複数個あれば、それが並列に実行される。しかし、どの依存関係も成立しないときは、④の待ち状態となる。

③Output object は連結タイルに連結されたオブジェクトに Input object として渡される。

④各オブジェクトタイルは Input object が入力されるまで待ちの状態となるが、Input object が入力されると、②の実行状態となる。

5.　DVLM 環境の使用例

本研究で提案した DVLM を用いたデザインプロセスを、大学図書館での本の貸出システムを例として示す。

まず、シミュレートしたいシステムモデルを作成するのに必要なオブジェクトタイルと連結タイル上に流れる object を定義する。

ACTOR オブジェクトタイル

「図書館の利用者」

「貸出受付カウンター」

ACTION オブジェクトタイル

「本の請求」

「書庫から取出す」

「認証する」

「書庫へ返却する」

DATABASE オブジェクトタイル

「蔵書と貸出状態リスト」

「書庫」

「登録者リスト」

連結タイル上に流れる object

「請求」

「請求結果」など

以上のように、定義されたタイルをキャンバス上に移動させ、連結させる（図 6）。そして、各連結タイルの依存関係を設定する。

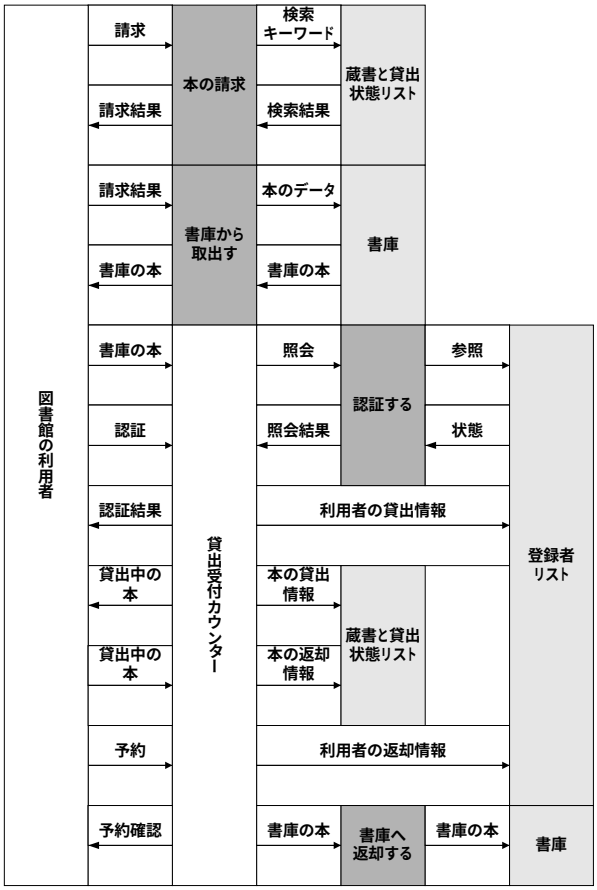

図 6　キャンバス上の記述

ここまでのステップが終了したところで、論理チェックを行うと、論理の整合性が取れていれば、図 7 のような SSADM 型 DFD 画面が表示される。

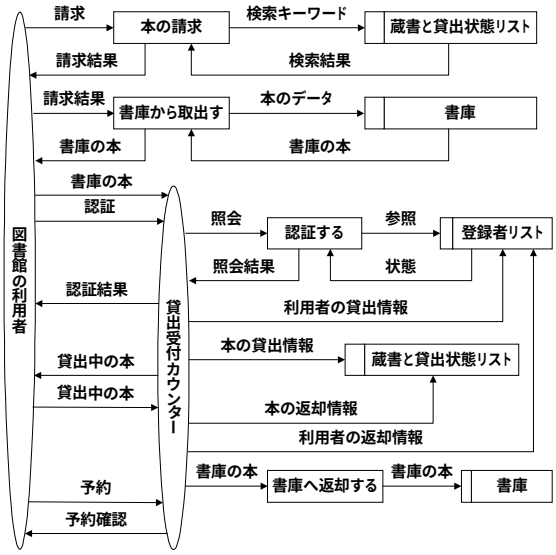

図 7　SSADM 型 DFD 画面

ここで、シミュレータを起動するための“初期条件設定”を行い、シミュレーションを開始すると、図 8、図 10 と図 12 に示したようにオブジェクトタイルの状態が変化し、図 9、図 11 と図 13 のようにデータの流れる様子が表示される。詳しく説明すると、「図書館の利用者」を図 8 のような状態に設定すると、まず Output object として「請求」が「本の請求」オブジェクトに出力される。「本の請求」オブジェクトは「請求」を受け取ると、「蔵書と貸出状態リスト」オブジェクトに対して「検索キーワード」を出力する。同様にして「検索結果」と「請求結果」がやり取りされ、「図書館の利用者」オブジェクトは「請求結果」を受け取ることとなる。このとき、「書庫から取出す」、「貸出受付カウンター」などのオブジェクトは実行できる状態にないため、データとして Input object が入力

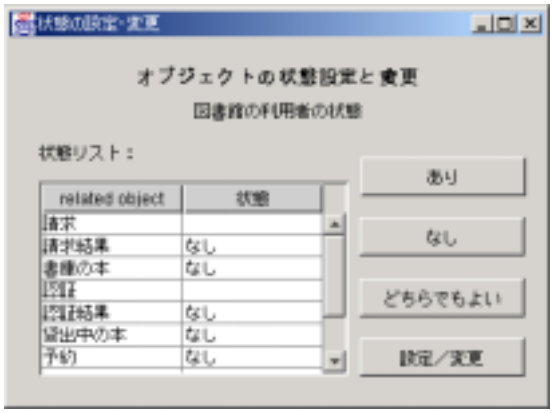

図 8　設定・変更画面 1

されるまで待ち状態となっている。

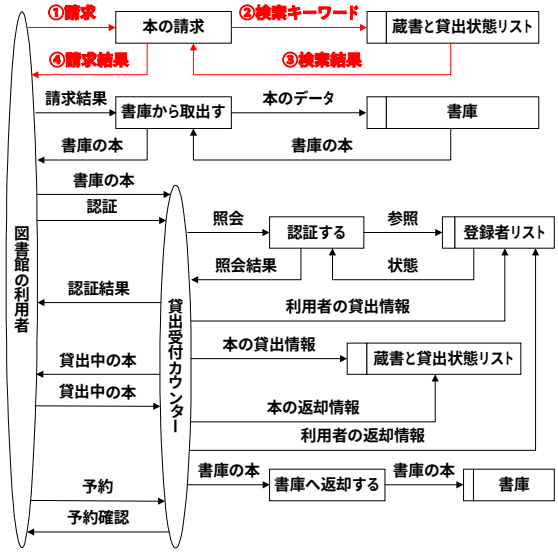

図 9　シミュレーション結果 1

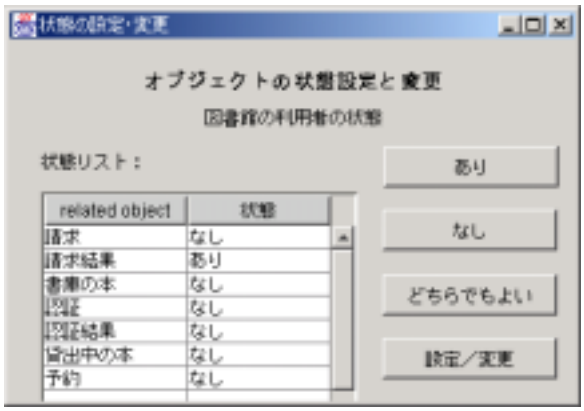

図 10　途中経過の状態

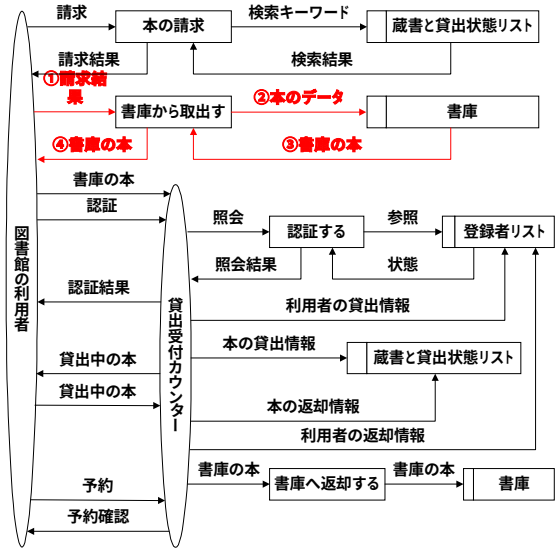

図 11　シミュレーション結果 2

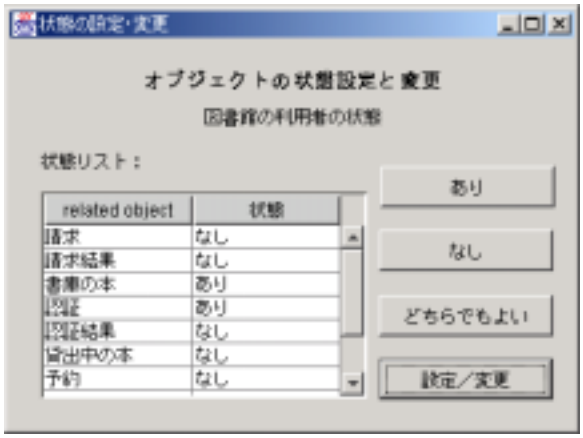

図 12　設定・変更画面 2

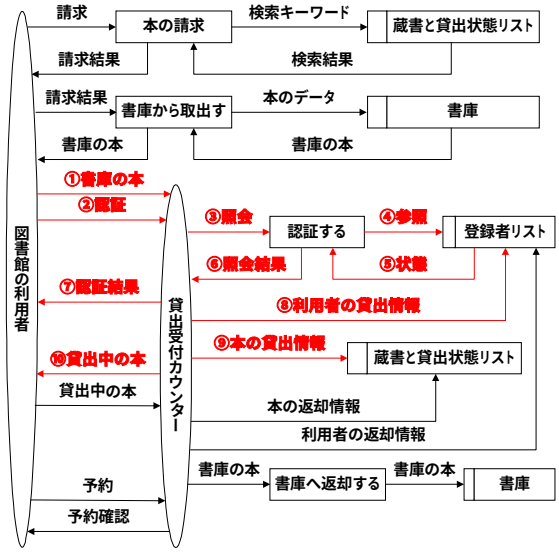

図 13　シミュレーション結果 3

次に、「図書館の利用者」オブジェクトが図 10 のように状態が変更されると、「請求結果」が Output object として「書庫から取出す」に出力される。「書庫から取出す」オブジェクトは「請求結果」を受け取ると、「書庫」オブジェクトに対して「本のデータ」を出力する。同様にして、「書庫の本」がやり取りされ、「図書館の利用者」オブジェクトは「書庫の本」を受け取ることとなる。このとき、「貸出受付カウンター」、「認証する」などのオブジェクトは実行できる状態にないため、データとして Input object が入力されるまで待ち状態となっている。

また、「図書館の利用者」オブジェクトが図 12 のような状態に設定されると、Output object として「書庫の本」と「認証」が「貸出受付カウンター」に出力される。「貸出受付カウンター」オブジェクトは「認証」を受け取ると、「認証する」オブジェクトに対して「照会」を出力する。同様にして、「参照」、「状態」など一連のデータがやり取りされ、「図書館の利用者」オブジェクトは「貸出中の本」を受け取ることとなる。このときも「書庫へ返却する」、「書庫」などのオブジェクトは実行できる状態にないため、データ待ち状態となっている。

このようにシミュレーション結果を通してデータがどのようにしてどこまで流れているかを見ることができる。

なお、図 6 のモデルでは、図書館の利用者が今貸出中の本に対して「予約」することができるように記述されている。しかし、「認証あり、予約あり」という状態を設定すると、シミュレーション結果は図 14 のようになってしまい、貸出受付カウンターから予約確認のメッセージは永久に返ってこない。この理由は、貸出受付カウンターは利用者の予約情報と本の予約情報を登録していないことにある。論理動作モデルでのデータの流れをシミュレートすることにより、このようなプロセス記述の欠陥を明らかにできるのである。

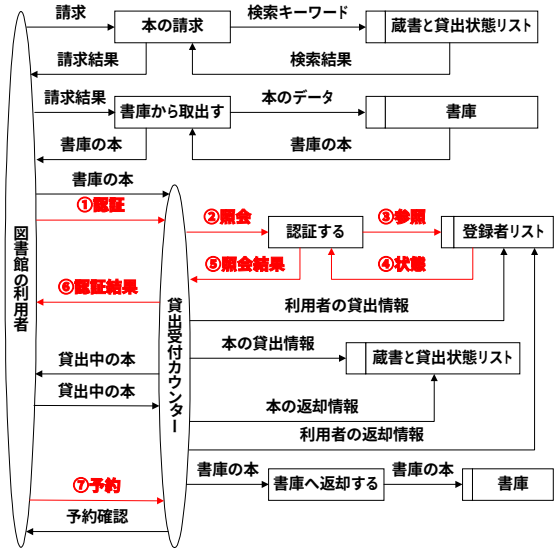

図 14　シミュレーション結果 4

6.　考察

先の使用例で示したように、DVLM 環境では DFD モデルの動作をシミュレートしながら

記述することができるため、インタラクティブにモデルを詳細化していくことが可能なのである。しかも、ブロックを組み立てるような感覚で DFD モデルを記述することができるため、これまでのエディタ等に比べてより簡単な操作でモデルの定義や修正が容易になっている。そのため、このような環境は、一般の利用者にも受け入れられやすいだけでなく、この環境を用いることで、一般の利用者がシステムデザインに直接関与することができるようになると考えられる。

DVLM 環境は、システムの動作を可視化することにより、人間と組織の情報行動に必要な機能と情報との関係を生成するものである。そのため、静的なモデルに内包されている目に見えない情報処理プロセスの問題点を発見し、提示することに役立つと考えられる。しかも、作成された DFD 図やそのシミュレーションの結果はシステムデザインに関する議論を深めることができるため、関係者のコミュニケーションツールとしても利用できる。このような作業を通して、参加者の頭の中にあるシステムの「共通理解モデル」が次第に明確となり、論理的なレベルで矛盾のない共通認識が形成されていくと考えられるからである。

一方、これとは異なり、モデル中の欠陥が先の使用例のような自分のミスではなく、プロセス自体が内包している欠陥であるならば、システム化以前にまずそのプロセス自体の見直しが必要である。そのような観点からすると、DVLM は BPR に際して、その企業や組織の改革後の情報処理プロセスに欠陥や冗長がないかどうかを確認するためのツールとしても利用可能であると考えられる。

ただし、現在の実装環境では、大量のタイルを定義し、配置すると、動作オブジェクトのデータ量が大きくなってしまい、キャンバス上に書かれているシステムモデルが見にくくなってしまうという問題点がある。しかも、動作シミュレーションの実行時間が極端に遅くなる要因ともなってしまう。

この問題に対する 1 つの解決策としては、ワークスペース間でデータを授受する機能を用い、マクロ的に表現することで部分的にも全体的にも見られるようにすることが考えられる。これにより、システムの一部の機能の中身を抽象化して、システムの全体像が見えるようにすると同時に、表示するデータも減少することからシミュレーションの実行時間も向上すると期待できる。

7. まとめ

本論文では一般の利用者に分かりやすく、かつシステムのモデル記述を支援する動作シミュレーション環境 DVLM を提案した。また、TileDesigner をプラットフォームとしてその基本機能、ユーザインターフェースを実現した。

また、この環境を用いて大学図書館での本の貸出システムを例としてシステムのモデル化を行い、そのプロセスを通して DVLM 環境の有用性を確認した。

今後の課題としては、実際の開発現場や情報システムに関するトレーニング現場で DVLM を使用し、その有効性を評価すると共に、より実用的な環境について検討を加える予定である。

謝辞

本研究は東洋大学経営学部の小林真士氏との共同研究であり、研究を進めるにあたり、小林氏の助言と協力を得た。ここに深く感謝の意を表す。

参考文献

[1]神沼靖子，内木哲也『基礎情報システム論－情報空間とデザイン－』共立出版, 1999。
[2] G.Cutts 著，浦昭二ほか訳,『情報システムの分析と設計－SSADM とその実践』培風館，1995。
[3] 野村泰朗，中島進，土肥俊郎「豊かな人間性と創造力を養うものづくり教育に関する研究（第三報）～RoboCup Junior と関連付けた新技術科カリキュラムにおける教材開発～」『埼玉大学紀要教育学部』，第 50 巻，第 2 号，2001。
[4] Liu,K., Sun,L., Barjis,J., and Dietc,J., "Capturing Organizational Behavior with Dynamic Modeling," Proceedings of 16th World Computer Congress, 2000.