修 士 論 文

CASE ツールを用いたオブジェクト指向方法論 事例研究

指導教官　片山 卓也 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

岩槻 伸洋

1997 年 2 月 14 日

目 次

図 目 次

表 目 次

第 1 章

はじめに

1.1 背景と目的

ソフトウェアの開発工程に対してさまざまな方法論が提案され論じられてきた。特に分析・設計といった上流工程に対しては、如何に問題を把握し表現するかが一つの大きな研究対象となってきた。オブジェクト指向はこの問題にアプローチするための一つの概念であり、その有用性によってソフトウェア開発に広く適用されるようになった。オブジェクト指向の概念はSA/SD 法のようなこれまでのシステム開発とは違い、上流から下流の各工程に一貫性を与え、見通しの良い開発を可能にする。ここに方法論についての活発な議論が展開され、多くの事例研究が報告されている。また、方法論による開発を支援する CASE(Computer Aided Software Engineering) ツールも存在し、ソフトウェア開発に対する新しい開発環境が整いつつある。しかし、実務へのこれらの導入は依然進まず、日常的な開発形態として定着するには豊富な事例研究とツール評価が待たれるところである。

本研究はソフトウェア開発へのオブジェクト指向方法論の導入を援助するため、CASE ツール上で事例研究を行なう。これにより、ツール評価と方法論に対する知見をまとめ、計算機環境を含めた方法論適用の利点/問題点を明らかにし、またツール機能への要請を明らかにすることを目的とする。

本稿ではオブジェクト指向方法論として Shlaer/Mellor 法を採用し、CASE ツールには SES 社の SES/ObjectBench を使用した。

1.2 本稿の構成

2 章では分析作業への準備として、オブジェクト指向方法論、CASE ツール、分析対象システムについての説明をする。3 章では実際に分析を行い、問題の分析がどのように進められたかについて説明する。4 章では分析の過程/結果から得られた情報をもとに、分析に方法論を適用する利点/問題点、方法論の評価、CASE ツールの評価を試みる。5 章では総括と今後の課題を述べる。

第 2 章

分析作業への準備

本章では実際の分析作業を行う上での準備として、以下の事柄について述べる。

- オブジェクト指向分析の概論
- 本稿で採用したオブジェクト指向方法論について
- CASE の概論
- 実際に使用した CASE ツールについて
- 分析対象システムについて

2.1 オブジェクト指向分析

従来、システムを把握し理解するためには、構築しようとするシステムを機能分割するという方法が取られてきた。機能分割によるアプローチでは、システムに要求される動作は多数の機能 (関数) の” 集まり” として表れる。しかしこの方法ではシステムが巨大になるとシステム全体の把握が困難となる。また、機能中心でシステムを分割するとシステムの各構成要素が必要とするデータが散在し、管理がしにくくなるという弊害を生じる。
これに対してオブジェクト指向分析では、システムをオブジェクトの集合として捉え、オブジェクトの抽出による分析を試みる。オブジェクトはデータとそれにアクセスする操作とのパッケージで定義され、システムに要求される機能は複数のオブジェクトがメッセージ通信により協調しあうことで達成される。

2.2 オブジェクト指向分析方法論

オブジェクト指向方法論は、オブジェクト指向の概念に基づいてシステムを構築するうえでの方法論である。数多くの方法論が提案されているが、その根幹は、システムはオブジェクトで構成され、互いに協調しながら目的の性能を達するというものである。現在の主要な方法論としてShlaer/Mellor 法 ['88]、OMT 法 ['91]、Code/Yourdon 法 ['90] などが挙げられる。

2.3 Shlaer/Mellor 法による分析

Shlaer/Mellor 法は、オブジェクト指向の方法論として確立され認知された最初の方法論である。記述力の高さから現在も実用レベルで適用され、また GE 社の OMT 等の方法論にも影響を及ぼしている。Slaer/Mellor 法では次に述べる各モデルによって要求されるシステムを捉える。

- 情報モデル
- 状態モデル
- 通信モデル
- プロセスモデル

以下、各モデルについて詳しく述べる。

2.3.1 情報モデル

情報モデルは、システムを構成する**オブジェクト**と、それらの間の**関係**[1]を記述するものであり、システムの静的な側面を記述したものである。オブジェクト内で定義される**属性**やオブジェクト間の**関係**の定義には規則があり、これに矛盾した定義は記述できない。[1]
ここでオブジェクトとは実世界の実体の集合を抽象化したものであり、この集合中のすべての実体 (**インスタンス**) はオブジェクトの属性に対して値を持つことで特徴づけされる。またすべてのインスタンスはオブジェクトのルール (操作) に従わなければならないとされる。
情報をモデル化することで留意する点は、オブジェクトとその関係を記述することでシステムの構造を理解することと、オブジェクトの属性を定義することでシステムが必要とするデータを明確にすることである。また、仕様段階で不明確になっている用語の定義としても重要な働きをする。

[1] インスタンス間の” つながり” を関連、オブジェクトレベルに抽象化した” つながり” を関係と呼ぶ。

2.3.2 状態モデル

状態モデルはオブジェクトの時間的な変化を状態遷移図を拡張して記述したものであり、オブジェクトの動的な側面を記述するものである。各オブジェクトは外部との通信によって刻々と内部の状態が変化する。内部変化を起こさせるものは**イベント**と呼ばれ、オブジェクト内部で発生するものと外部から受け取るものとが存在する。また、各状態の内部にはその状態への遷移が起こった時の**動作 (アクション)** が記述される。イベントとアクションには規則があり、これに矛盾した定義は記述できない。[2]

2.3.3 通信モデル

状態モデルがシステムの構成要素の動的側面を捉えるのに対し、**通信モデル**は、状態モデルと外部の実体との間のイベント通信を記述したもので、システム全体の振舞を表すものと考えられる。実際分析段階では、イベントの発生源と到達点を特定するうえで重要な役割を持ち、モデル構築後のシミュレーションではシステムのオブジェクトレベルでの動作を確認するために使われる。

2.3.4 プロセスモデル

プロセスモデルは、状態モデルで記述したアクションをより具体的に**データフローダイアグラム**によって記述したものである。状態モデルでのアクションはシステム全体の処理の流れに注力しており、具体的なデータの流れやアルゴリズムには言及していない。プロセスモデルは、この機能的な側面にたいして詳細に記述するのが目的であり、各アクションに対してプロセスモデルが記述される。

2.4 CASE

2.4.1 CASE とは

CASE は Computer-Aided-Software-Engineering の略であり、その名の通り計算機によってソフトウェアの構築を支援するものである。支援の対象はプログラミングをサポートするような下流 CASE から分析作業といった人の知的作業をサポートする上流 CASE までさまざまである。近年では上流〜下流の各工程における支援に留まらず、システム開発に対して開発管理支援と開発支援の側面を連動させた研究もされている。[10]

2.4.2 SES/ObjectBench について

SES/ObjectBench(SES 社) は、Shlaer/Mellor 法によるシステムの分析を支援する上流 CASE ツールである。
ツールとしての支援機能は以下のようなものが挙げられる。

- 情報/状態/通信モデルを記述する図形エディタとしての機能
- Shlaer/Mellor 法のセマンティクスにそったモデルの構築
- モデルの実行、デバック
- 作成した図や、図が持つ情報をテキストとして出力する、作業プロダクトの生成機能

これらの機能により分析者は、調査、テストが可能な一貫したモデルの記述をすることができ、その成果物から異なった形態のドキュメントを作成することができる。ObjectBench はいくつかの**作業フィールド**で構成されている。各フィールドはいくつかの**ページ**で構成され、他のフィールドと相互に関連している (図 2.1)。
OIM フィールドでは情報モデルを記述する。オブジェクトの数が多くなると 1 枚の情報モデル

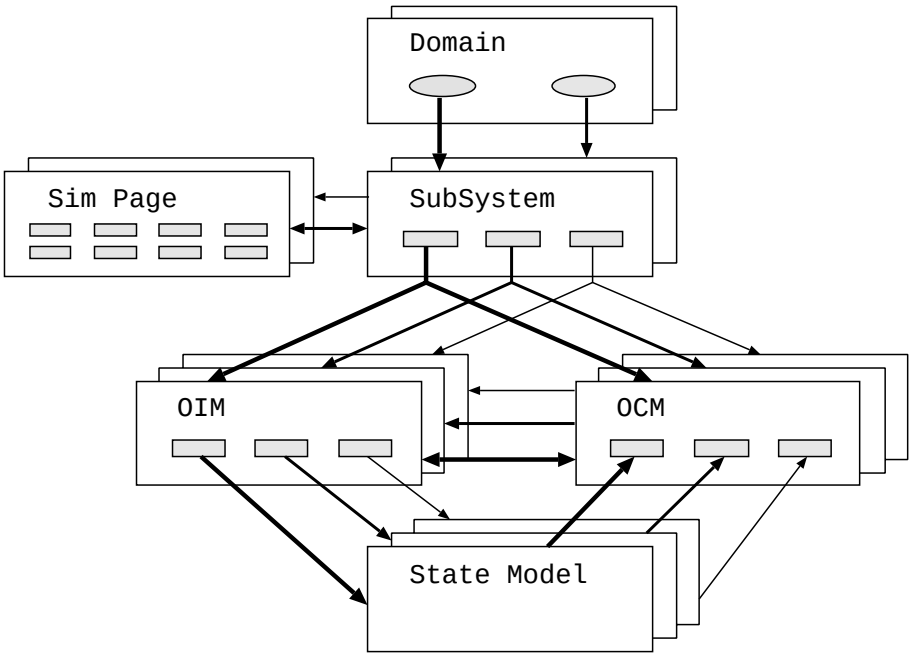

図 2.1: ObjectBench のページ構成とページ間の関連

では分析しにくくなるのでオブジェクトをクラスタリングする。このクラスタリングしたオブジェクト群の 1 つのまとまりが **SubSystem フィールド** の 1 つの要素 (**サブシステム**) に対応する。すなわち、サブシステムはオブジェクトの集合である。
同様にサブシステムもクラスタリングされ、1 階層上の **Domain フィールド** にまとめられる。すなわち Domain フィールドの要素 (**ドメイン**) はサブシステムの集合である。
State Model フィールド では状態モデルを記述する。
OCM フィールド では通信モデルを記述する。
Sim Page フィールド はモデルを実行するために、モデルの初期状態 (生成するインスタンスの数や属性の値など)、シュミレーション・パラメータ (イベントの発火時間など)、実行時に発火するイベントを記述する。

2.5 対象システム

本稿で問題として取り上げた対象システムは自動倉庫システムである。主な機器構成は図 2.2 に示される。自動倉庫システムとは移動ロボットによって搬入口 (ステーション) と棚との間で、収

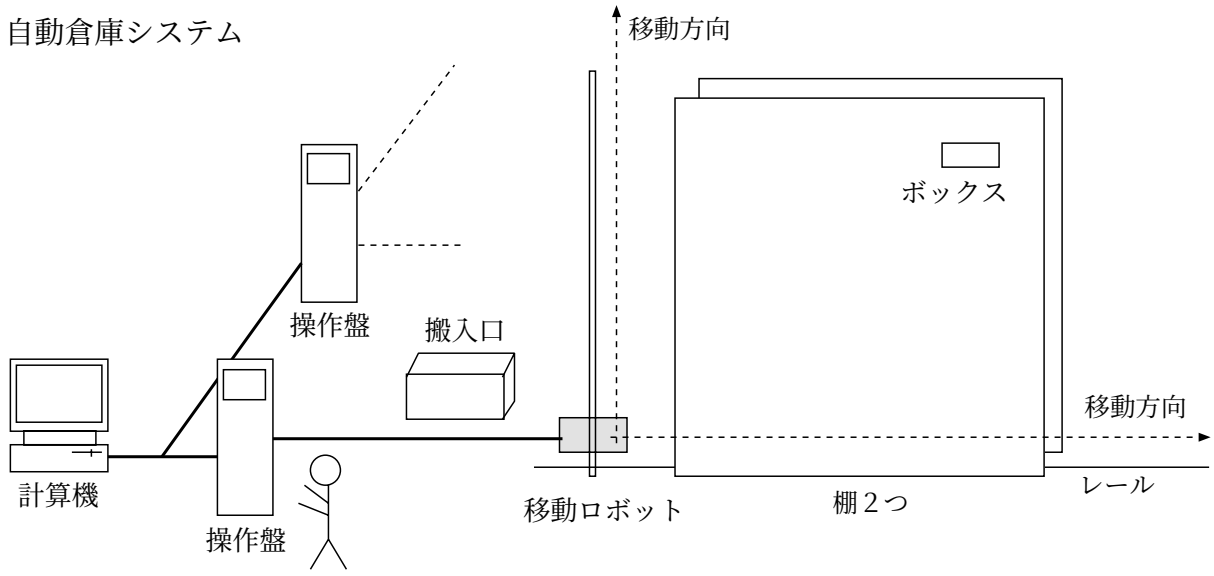

図 2.2: システム構成図

納されている荷物 (バケット) の移動を無人で処理するシステムである。システムはロボット制御部 (移動ロボット本体) と指令操作部 (操作盤) に大きく別れ、シリアル通信により連動し、作業を行なう。移動ロボットが走るレールを 1 レーンとして倉庫内のシステムは複数のレーンで構成され、1 レーンを 1 つの操作盤が管理する。計算機は 1 システム内の操作盤を統括したデータの管

理を行なう。移動ロボットは表 2.1 にあるような指令により決められた動作を行う。

表 2.1: 動作指令とロボット動作

指令	ロボット動作
入庫	搬入口にある荷物を棚へ収納する。
出庫	棚にある荷物を搬入口へ取り出す
ピッキング	棚にある荷物を搬入口へ取り出し、その後作業者が荷物の一部を取り出し、残った荷物は元の棚へ収納する。
積み増し	棚にある荷物を搬入口へ取り出し、その後作業者が荷物の積み込みを行い、積み増しされた荷物は元の棚へ収納する。
移動	棚にある荷物を別の空き棚へ移動する。

第 3 章

分析

本章は、手作業による分析作業と CASE ツールを使用した分析作業で構成される。前者と後者を比較することで、次章でのツール機能への要請が明らかになる。複数のモデルによってシステムを多角的に分析する方法では、あるモデルの変更内容が他のモデルに波及するため、情報モデル→状態モデル→通信モデル→プロセスモデルのように一方向に分析作業は進行しない。分析は、各モデルを相互に行き来 (**ラウンドトリップ**) し、モデルを詳細化しながら行われる。各モデルの構築をラウンドトリップしながら分析することで、全体として徐々に詳細になっていく。

3.1 情報モデルの作成

3.1.1 オブジェクトの認識

最初に、対象システムの機器構成からオブジェクトを抽出する。自動倉庫システムの開発では、要求仕様の中に実際のシステムの最低限の機器構成が示されている。しかし、設計段階でオブジェクトやその役割を現実世界にマッピングする際、要求仕様に明記されていない機器の導入が行われるかもしれない。このため分析の初期段階ではシステム構成図から実世界の” もの” として認識されるオブジェクトを発見しシステム構造の骨格を得ることを目的とする。
なお、実世界の” もの” をオブジェクトとして認識する際、要求仕様での機器名 (インスタンス) による構成を抽象的な名前 (オブジェクト) による構成に置き換える作業を行う。この作業でオブジェクトとして認識されたものはを図 3.1 に示す。

移動ロボット	レール	部品	計算機	制御盤
作業者	ステーション	棚	バケット	

図 3.1: 認識されたオブジェクト

3.1.2 関係の記述と定式化

次に、認識したオブジェクト間の関係を考察する。まず、オブジェクト間に以下のような関係を定義する。

- R1:「計算機は操作盤に指令を出す」
- R2:「操作盤は移動ロボットに指令を出す」
- R3:「移動ロボットはステーションにバケットを置く」
- R4:「部品はバケットに収納される」
- R5:「移動ロボットはレールを移動する」
- R6:「バケットは棚に置かれる」
- R7:「作業者は操作盤に指令を入力する」

次に実世界での関連の多重度を考えることで関係の種類を決定する。なお関係は、あるオブジェクトのインスタンスと、それに関係するオブジェクトのインスタンスとの対応によって 10 種類に分類される。インスタンスの対応イメージとその関係の種類の表記法を図 3.2 にまとめる。ここで先に述べた関係の定義を多重度を加味して以下のように修正する。

- R1:「計算機は” 複数” の操作盤に指令を出す」
- R2:「操作盤は” １つの” 移動ロボットに指令を出す」
- R3:「移動ロボットは” ２つの” ステーション (搬入出口) を持つ」
- R4:「バケットは” 複数の” 部品を収納する」
- R5:「移動ロボットは” １レール上” を移動する」
- R6:「棚は” 複数の” バケットを格納する」
- R7:「操作盤は” 複数の” 作業者に指令を入力される」

関係の記述の次に関係の定式化を行う。定式化の目的は、ある関係において、どのインスタンスとどのインスタンスが関連するのかを特定できるようにすることである。これは一方のオブジェクトが、関係する別のオブジェクトの識別子を参照属性として持つことで実現される。識別子は

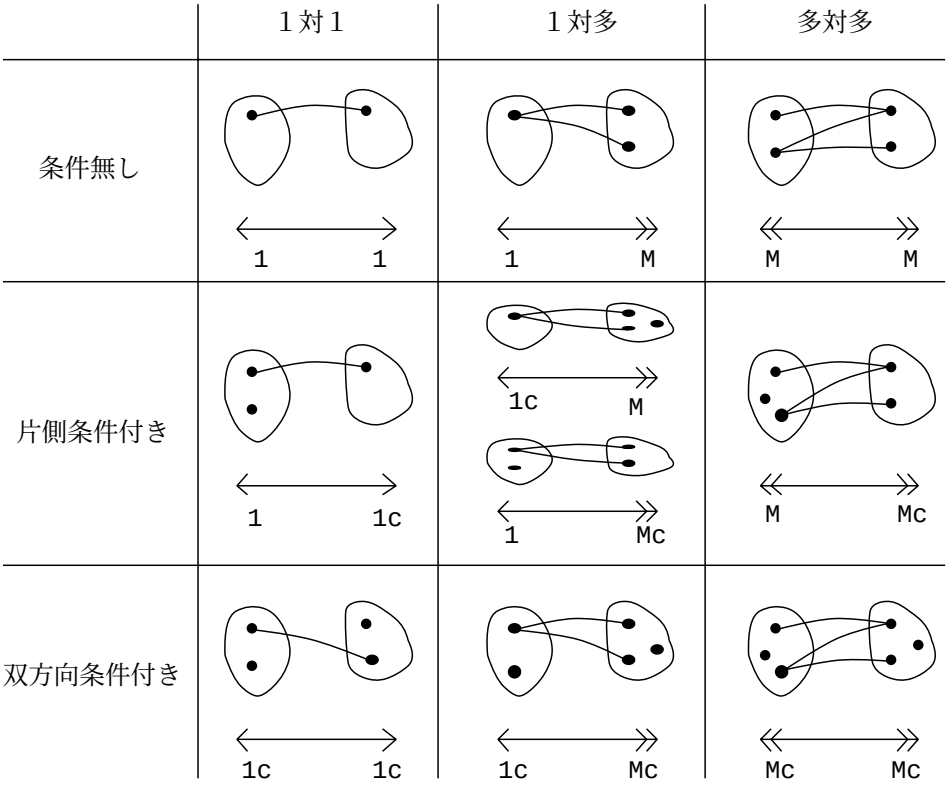

図 3.2: 関係の分類とアークによる表記

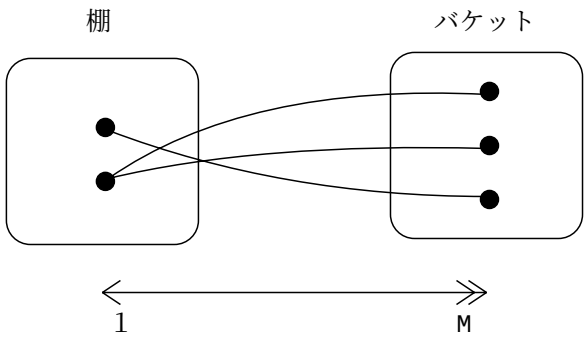

図 3.3: 例：棚とバケットの関係

インスタンスの生成時に機械的に割り振られる数値 (ID) として各オブジェクトに追加した。参照属性の追加は、

- １対１関係の場合、一方のオブジェクトに他方のオブジェクトの識別子を追加する。
- １対多関係の場合、多側のオブジェクトに他方のオブジェクトの識別子を追加する。
- 多対多関係の場合、両オブジェクトの識別子を参照属性に持つ関連付けオブジェクトを作成する。

に従い、機械的に行う。例えば棚とバケットの関係は１対多であるのでバケットに参照属性として” 棚 ID” を追加する (図 3.4 参照)。ここまでの段階で情報モデルは図 3.5 のようになる。

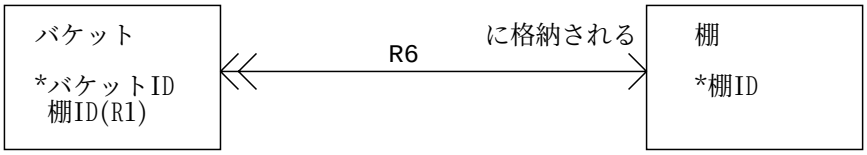

図 3.4: 棚とバケットの関係の定式化

3.1.3 情報モデルの再考

分析過程において情報モデルは、新たなオブジェクトの発見、オブジェクトの分割、複数オブジェクトの抽象化が頻繁に行われる。具体的には、搬送指令は計算機－操作盤間と操作盤－移動ロボット間では別のオブジェクトとした方が自然と考え、「搬送指令」「制御指令」の２つのオブジェクトに分割する。また後述の通信モデルの作成過程からセンサーオブジェクトを追加し、完了通知オブジェクトを追加する。このようにして最終的に図 3.6 のようになる。

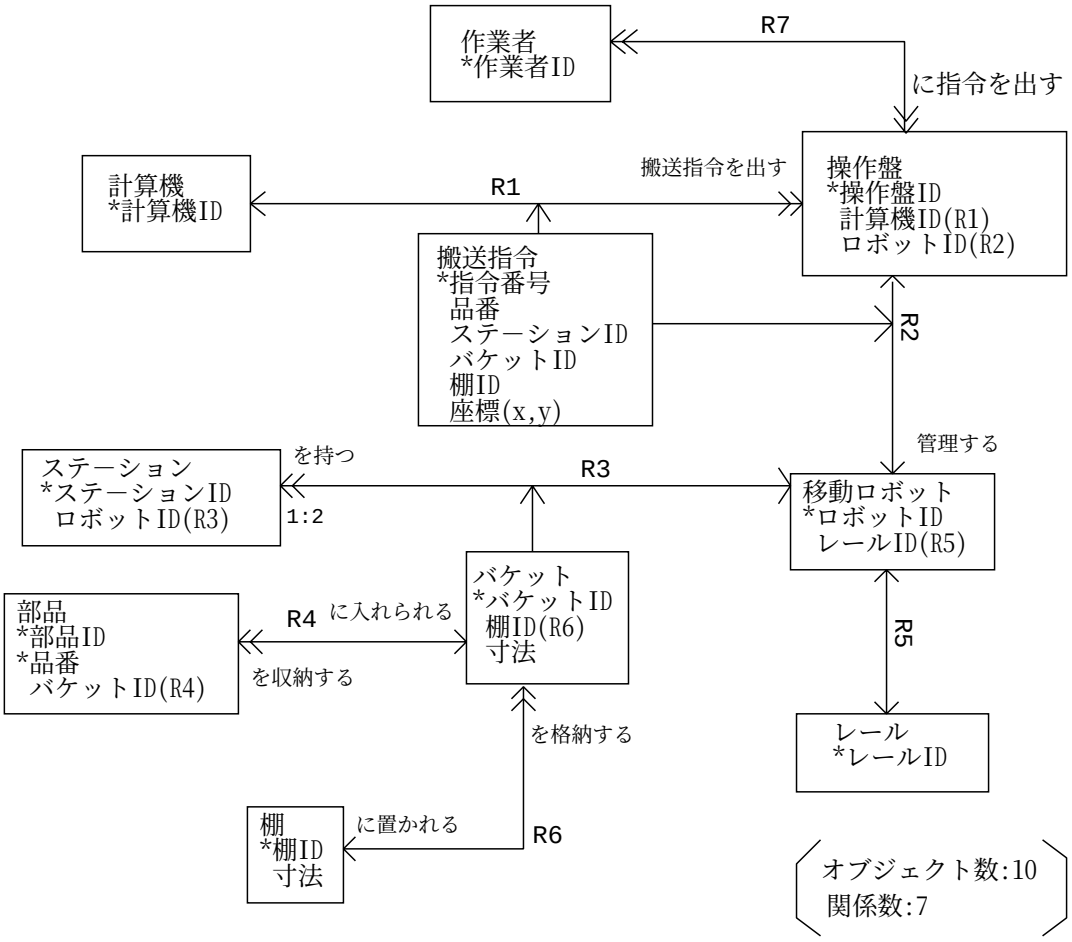
作業者
*作業者ID
R7
に指令を出す
計算機
*計算機ID
搬送指令を出す
R1
操作盤
*操作盤ID
計算機ID(R1)
ロボットID(R2)
搬送指令
*指令番号
品番
ステーションID
バケットID
棚ID
座標(x,y)
R2
管理する
ステーション
*ステーションID
ロボットID(R3)
を持つ
1:2
R3
移動ロボット
*ロボットID
レールID(R5)
バケット
*バケットID
棚ID(R6)
寸法
部品
*部品ID
*品番
バケットID(R4)
R4
に入れられる
を収納する
R5
を格納する
レール
*レールID
棚
*棚ID
寸法
に置かれる
R6
オブジェクト数:10
関係数:7

図 3.5: 情報モデル 1

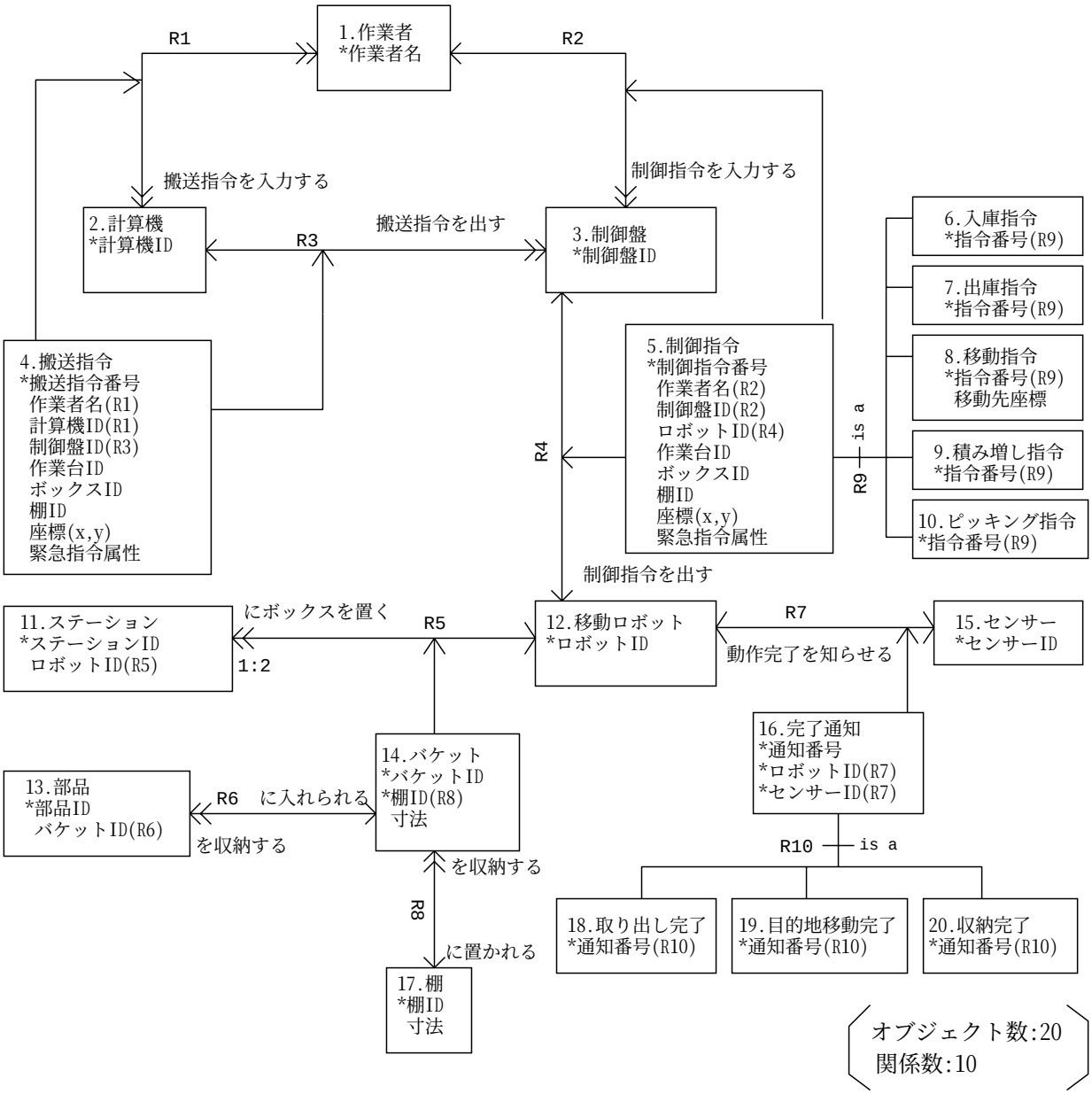
1.作業者
*作業者名
R1
R2
搬送指令を入力する
制御指令を入力する
2.計算機
*計算機ID
R3
搬送指令を出す
3.制御盤
*制御盤ID
4.搬送指令
*搬送指令番号
作業者名(R1)
計算機ID(R1)
制御盤ID(R3)
作業台ID
ボックスID
棚ID
座標(x,y)
緊急指令属性
5.制御指令
*制御指令番号
作業者名(R2)
制御盤ID(R2)
ロボットID(R4)
作業台ID
ボックスID
棚ID
座標(x,y)
緊急指令属性
R4
R9
is a
6.入庫指令
*指令番号(R9)
7.出庫指令
*指令番号(R9)
8.移動指令
*指令番号(R9)
移動先座標
9.積み増し指令
*指令番号(R9)
10.ピッキング指令
*指令番号(R9)
制御指令を出す
11.ステーション
*ステーションID
ロボットID(R5)
にボックスを置く
R5
1:2
12.移動ロボット
*ロボットID
R7
動作完了を知らせる
15.センサー
*センサーID
16.完了通知
*通知番号
*ロボットID(R7)
*センサーID(R7)
14.バケット
*バケットID
*棚ID(R8)
寸法
13.部品
*部品ID
バケットID(R6)
R6
に入れられる
を収納する
R10
is a
を収納する
R8
に置かれる
18.取り出し完了
*通知番号(R10)
19.目的地移動完了
*通知番号(R10)
20.収納完了
*通知番号(R10)
17.棚
*棚ID
寸法
オブジェクト数:20
関係数:10

図 3.6: 情報モデル2

3.2 状態モデルの作成

3.2.1 主要動作のモデル化

移動ロボットの状態モデルを作成することにより、移動ロボットの主要動作をモデルによって仕様化する。モデルは、「入庫」「出庫」「移動」「ピッキング」「積み増し」の５つの動作をシミュレート可能でなくてはならない。まず状態を認識するため、個々の動作について、ロボットの状態列を考える。主要動作についての動作モデルを図 3.7〜図 3.10 に示す。

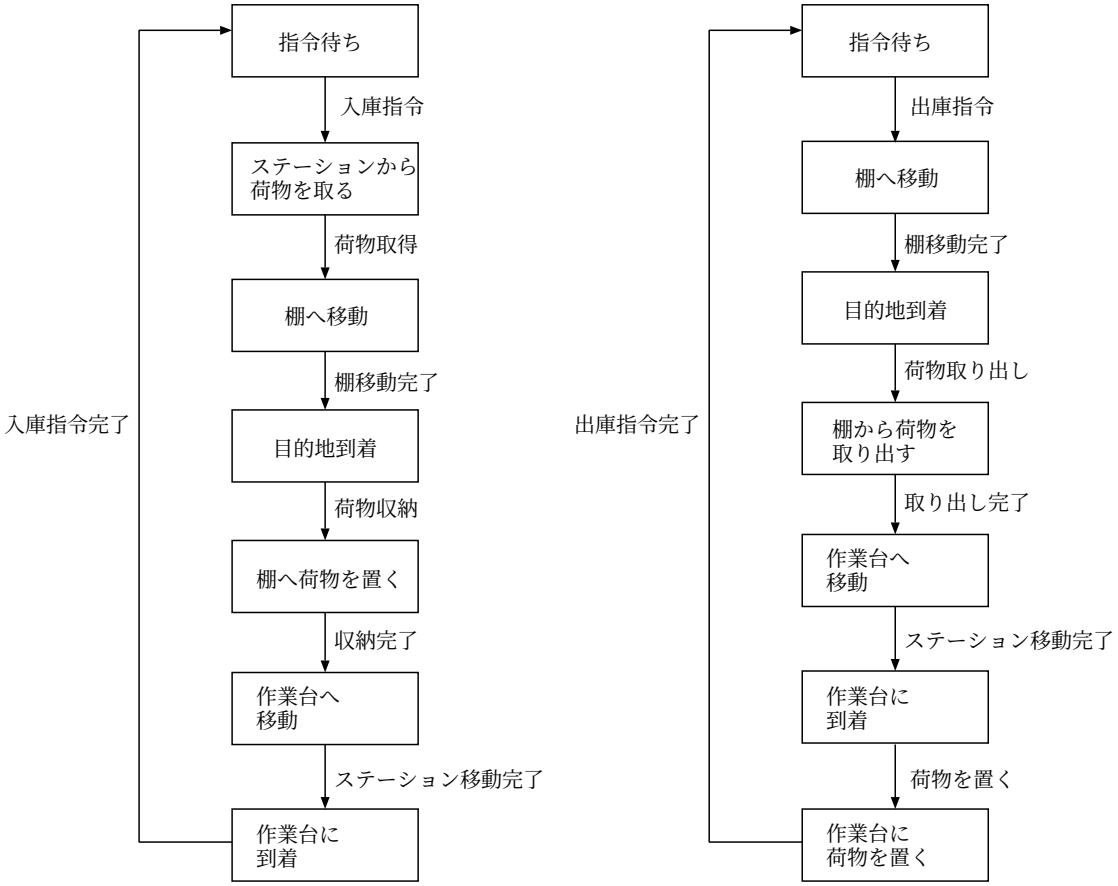

図 3.7: 入庫/出庫の動作モデル

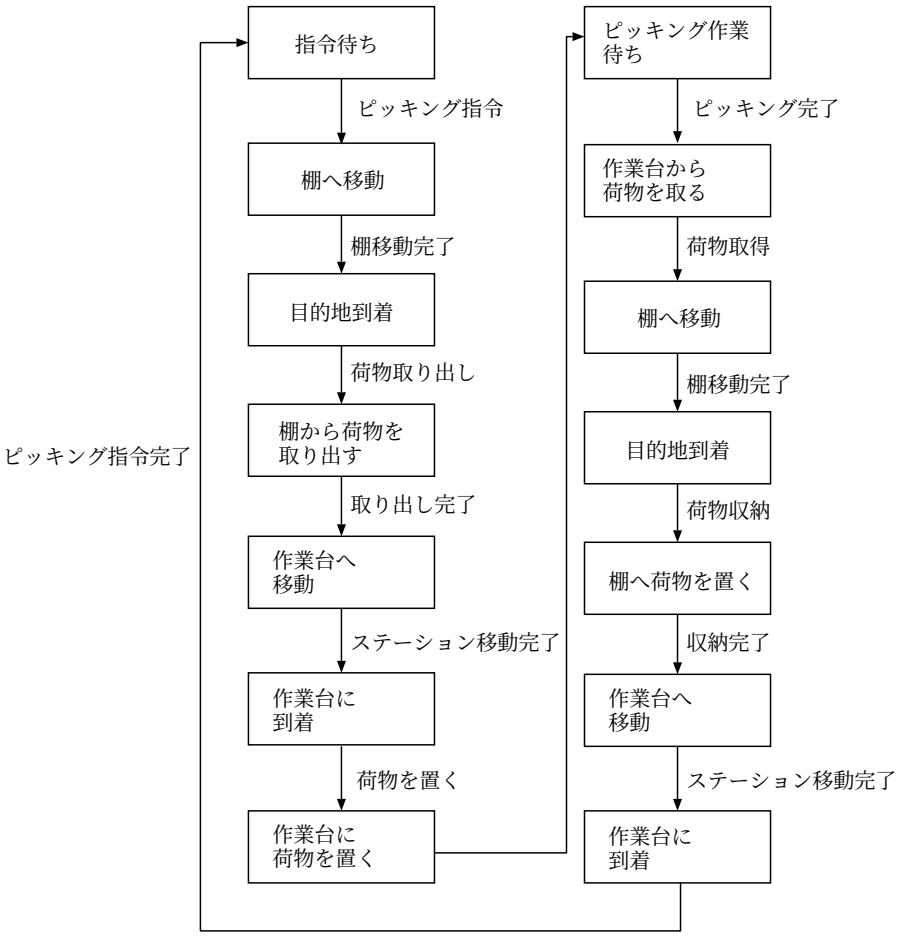

図 3.8: ピッキングの動作モデル

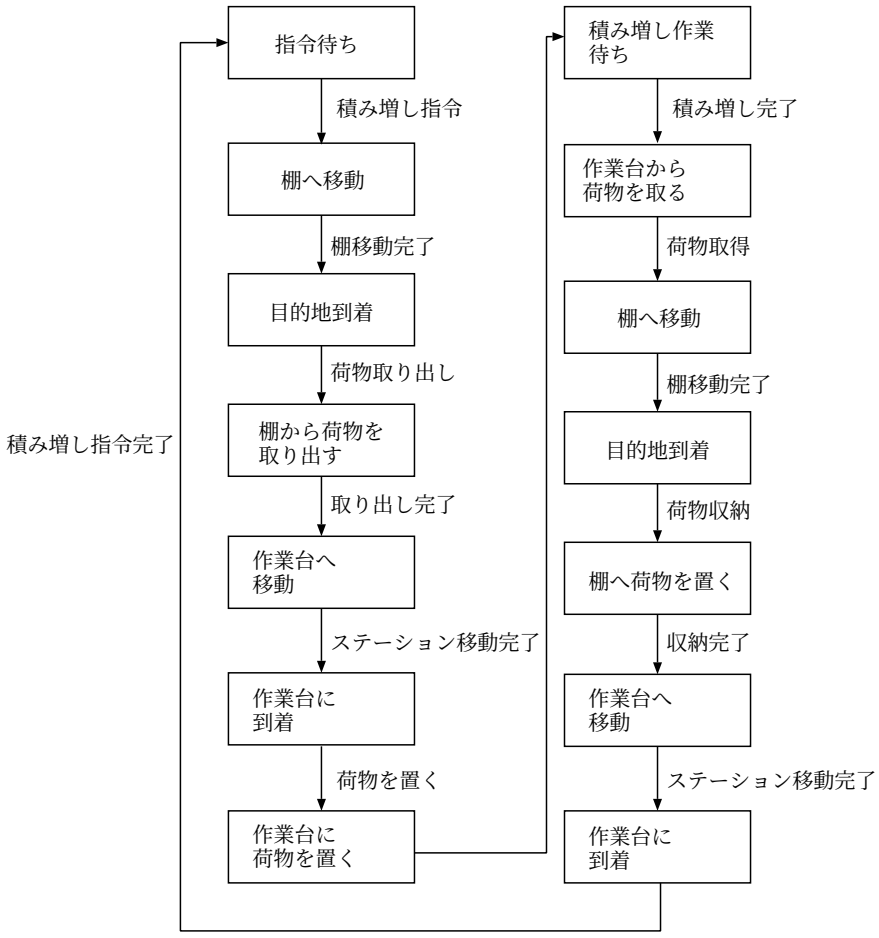

図 3.9: 積み増しの動作モデル

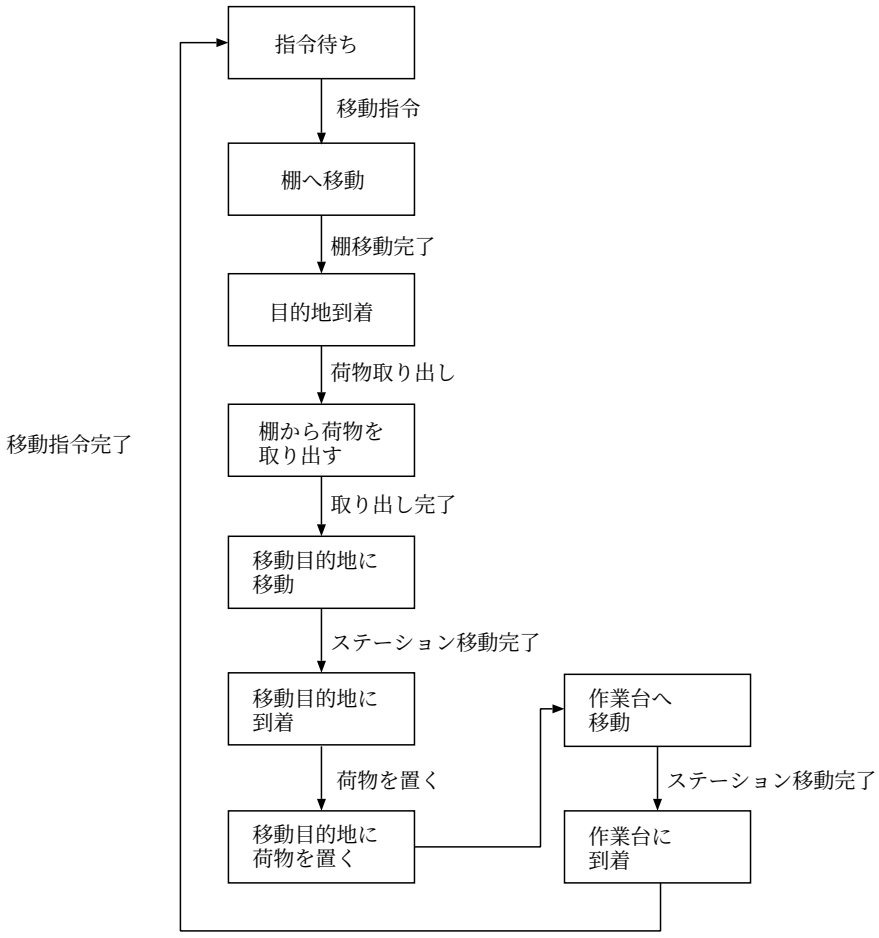

図 3.10: 移動の動作モデル

3.2.2 主要動作モデルの統合

ロボットの状態モデルは５つの主要動作モデルを統合することで得た。統合はモデル間で共有できる状態とイベントを識別し、状態名とイベント名を同一にして一枚の図で表現するようにした。図 3.11 参照。

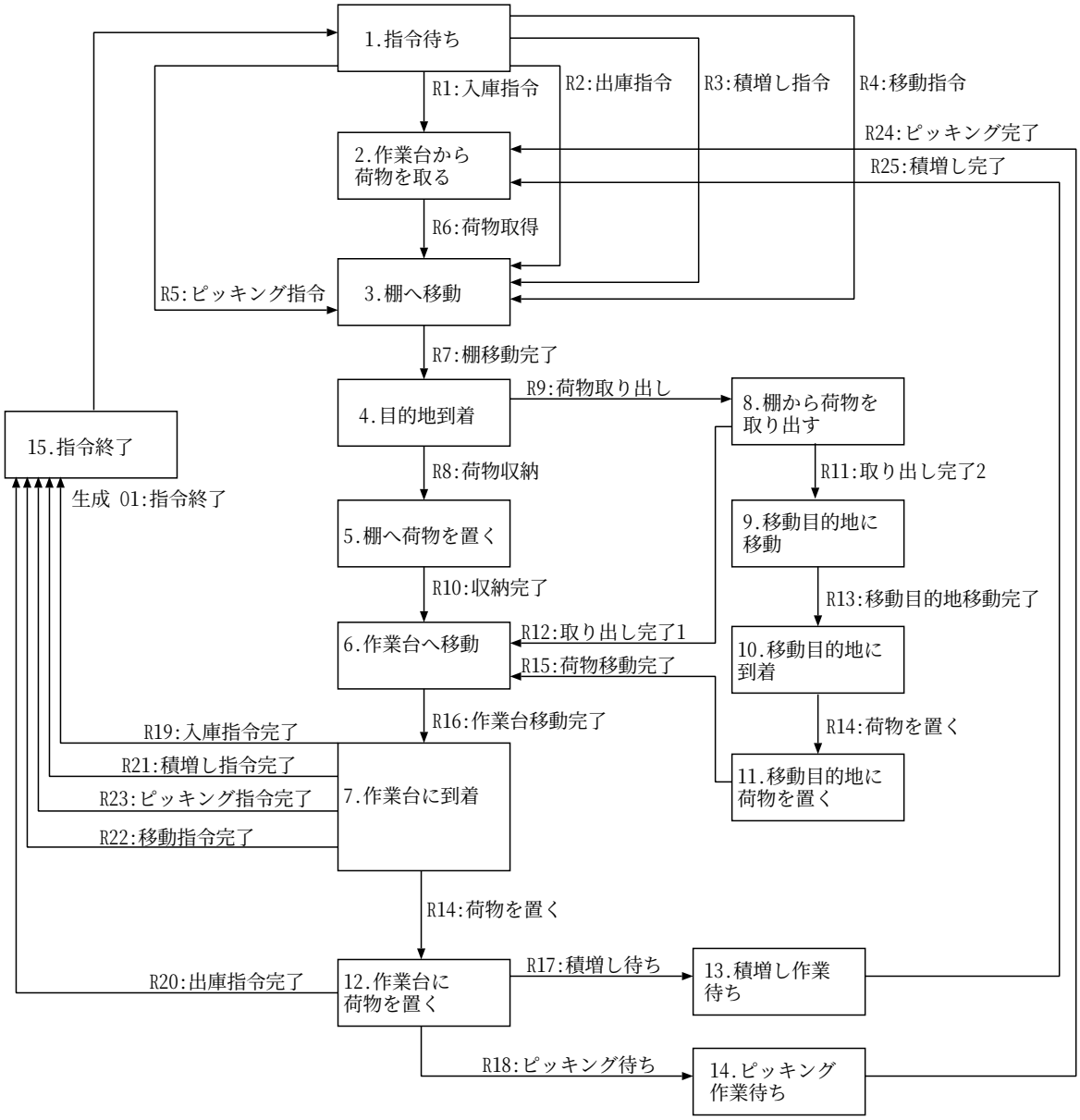

図 3.11: 移動ロボットの状態モデル

3.3 通信モデルの作成

3.3.1 イベント発生源と到達点

通信モデルの作成は、まずオブジェクトの状態モデルで認識/定義したイベントを、そのオブジェクト内部で生成されるイベント (内部イベント) と外部のオブジェクトから受け取るイベント (外部イベント) に分ける作業から行う。次に外部イベントがどのオブジェクトで発生するのか定義する。外部イベントをオブジェクト間のアークで表現した通信モデルを図 3.12 に示す。

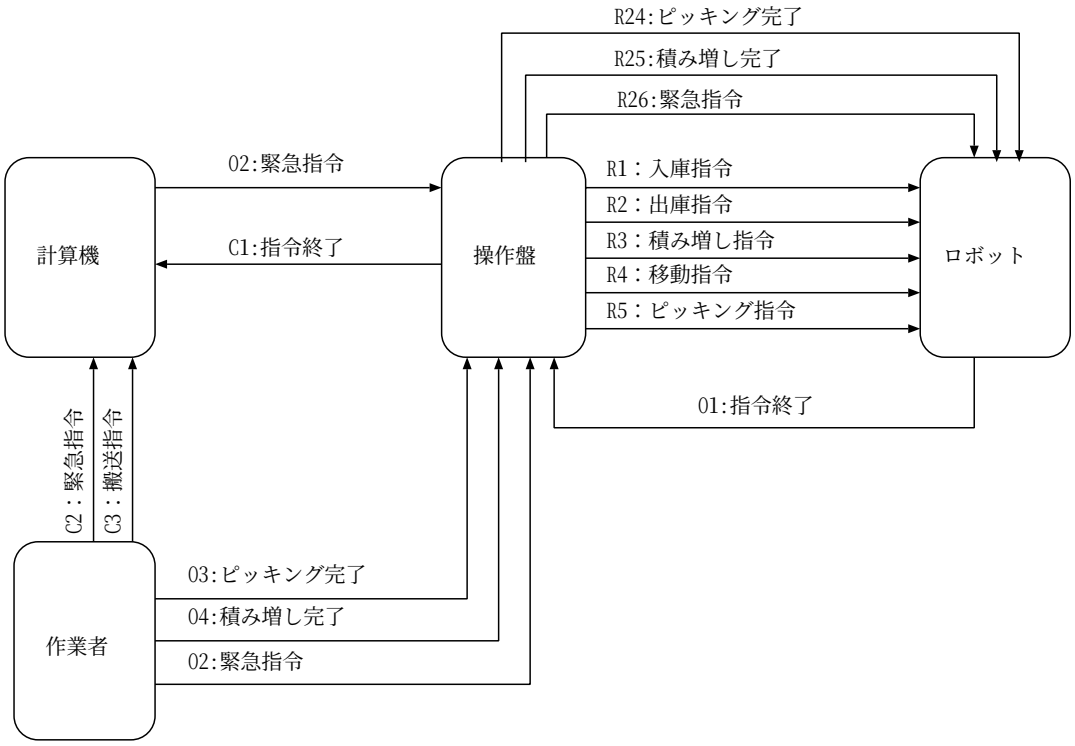

図 3.12: 通信モデル

3.3.2 新たなオブジェクトの追加

ここまでの分析結果で認識されているオブジェクトだけでは、生成されないイベントが発見された。例えば「移動ロボット」オブジェクトが受信するイベントの中の「収納完了」や「取り出し完了」は「移動ロボット」内で無条件に生成されるイベントではなく、実世界のロボットと荷物の状態によって生成するかどうか決定されるイベントである。ここで２つの解決方法が考えられる。１つは「移動ロボット」がアクションの中で荷物の状態を監視し、荷物が棚に格納されたことを

確認したら「収納完了」を生成する方法と、もう１つはイベントが発生するための条件を監視し、満たされた時にロボットに対して「収納完了」や「取り出し完了」イベントを送信するようなオブジェクトを追加することである。ここでは後者を選択し、センサーオブジェクトを新たに追加した。
なお、Shlaer/Mellor 法ではオブジェクトが他のオブジェクトの状態 (属性) を参照可能としているが、これは情報隠蔽の概念に反しているわけではない。分析では、オブジェクト間にどのようなデータアクセスが必要なのか決定することも目的の１つだからである。情報隠蔽の概念は設計段階で反映される。

3.3.3 異常イベントの追加

現実世界で、ロボットが荷物の格納や取り出しに失敗したとき、3.3.2 節で追加した「センサー」オブジェクトは「収納完了」や「取り出し完了」のイベントを発行せず、「R30:異常」イベントをロボットに対して送信するようにする。また、ロボットは指令が異常終了したことを操作盤に知らせるため、ロボットから操作盤へ送信される「O5:異常」イベントを定義する (図 3.13)。

3.4 CASE ツールを使用した分析作業

手作業で作成したモデルを CASE ツール上に乗せて、ツールを使ったモデルの洗練を行った。モデルは、
情報/状態/通信モデルの構築→モデルの検証 (シンタクスとセマンティクスのチェック)
→モデルの実行
を繰り返し行うことで洗練されていく。モデルの実行とは、構築したシステムに対してイベントを投じた時にできる制御スレッド (イベントとアクションの列) と、ある時点でのシステムの状態 (あるオブジェクトのインスタンスがいくつあるか、等) を観察/調査できる機能である。

モデルのクリティーク

まず始めに、手作業で作成したモデルをクリティーク機能にかけた。クリティークは、コンパイルの前段階として各モデルのノードとアークに行える。これによってモデルのセマンティクスに反した記述が排除された。

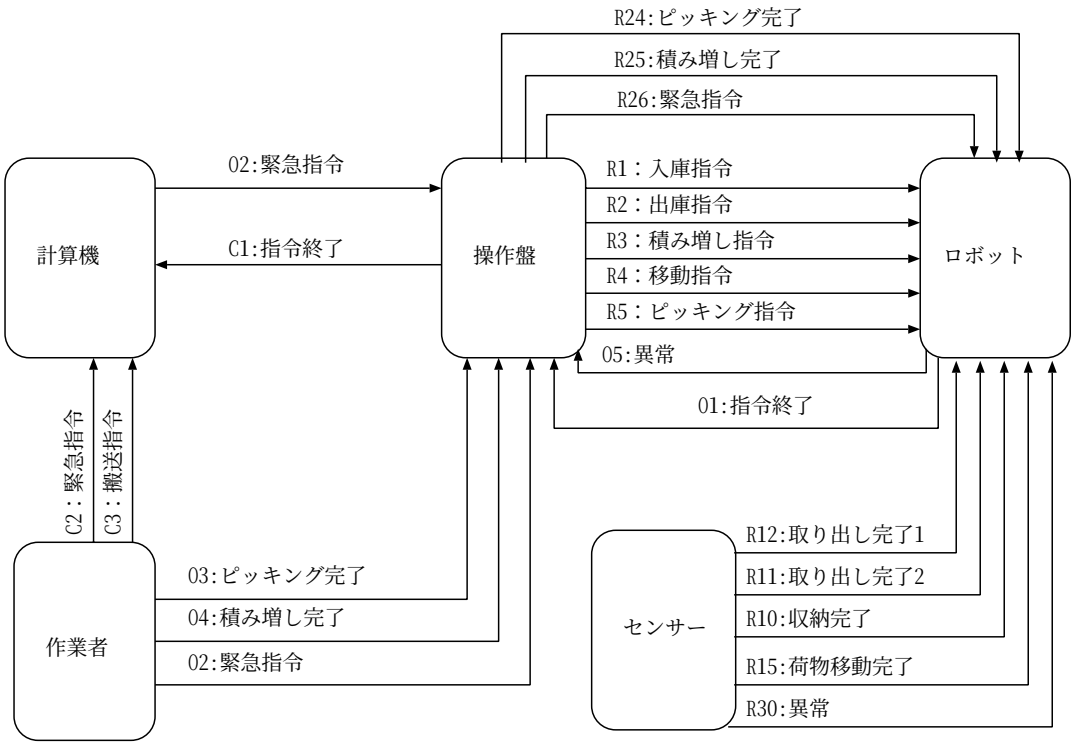

図 3.13: 通信モデル (オブジェクト追加後)

オブジェクト関係の再考

ツールが定義している４つの言葉を選択することで自動的に関係の多重度が記述される。言葉とアークの対応関係を図 3.14 に示す。ここで全てのオブジェクト関係について再考した結果、

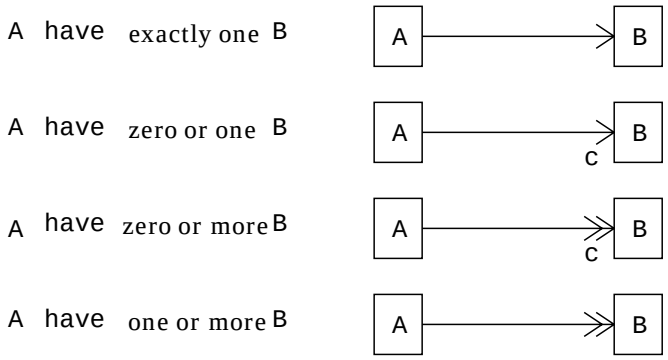

図 3.14: 「A は B を持っている」の例

R34,R23,R27,R6 が変更された (図 3.15 参照)。「指令」オブジェクトは関係から切り離し、作業者との関係を考えた。また作業履歴と異常履歴を記録する新たなオブジェクトを追加した。センサーオブジェクトは異常イベントの識別作業によって２つのセンサーが必要と分かったため２つのサブタイプに分割した (図 3.15 参照)。

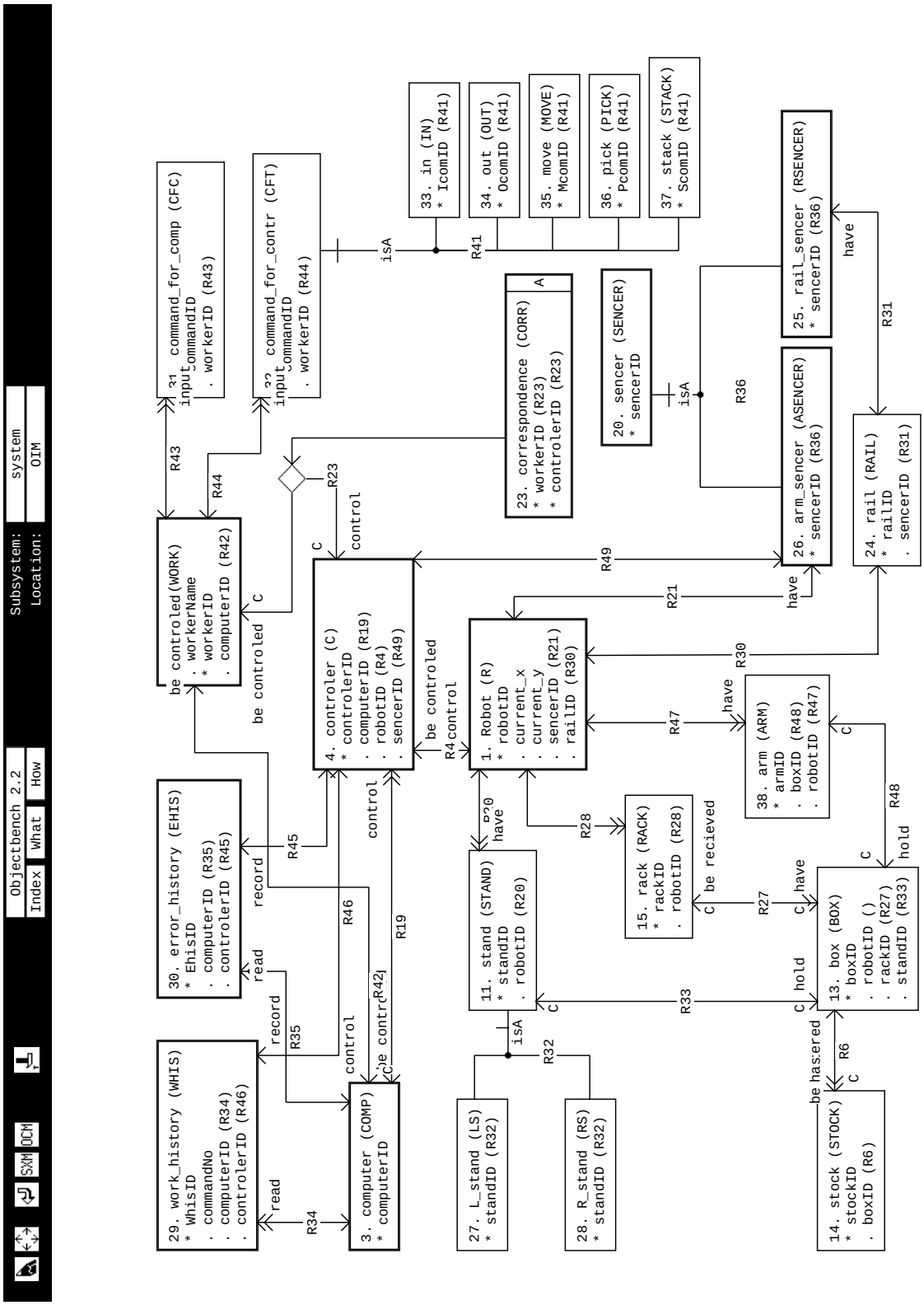

図 3.15: ObjectBench 上の情報モデル

3.4.1 モデルのシミュレーション

モデルのシミュレーションは以下２つの側面から行う。

(1) システムの状態変化の順序、すなわち動作シーケンスに着目した場合のシミュレーション

(2) システムの経時的変化に着目し、モデルに時間情報を与えた場合のシミュレーション

(1) に対してのシミュレーション作業は以下の手順で行う。

1 シミュレーション対象となるオブジェクトを決定する。
2 システムの初期設定をする。
3 モデルを実行する。
4 表 2.1 の５つの主要動作の発火点となるイベントを外部イベントとして実行中のモデルに与える。
5 動作シーケンスをチェックする。

1. について、対象オブジェクトを「操作盤」「ロボット」「センサー」とする。初期設定は各オブジェクトについてインスタンスを１つ生成する。主要動作の発火イベントは、モデル実行時に「操作盤」のアクションで

生成〈発火イベントラベル〉:〈発火イベント名〉(〈引数〉)

を実行すると生成される。具体的には、通信モデルで'controler' から'Robot' に流れる R1〜R5 を生成させる (図 3.19 参照)。

(2) に対しては時間情報が必要である。ObjectBench はイベント転送時間、アクション時間、居住時間の設定が可能である。

(1) で動作シーケンスが正しいことを確認した後に、モデルの時間情報を設定した。イベント転送時間は、オブジェクトからオブジェクトへのイベントの送信時間である。時間を 0 時間以上にした場合、イベントは生成されるが転送時間が経過するまで受け取り側のオブジェクトに到達しない。アクション時間は、アクションを実行するために必要な時間である。居住時間は、アクションの実行のあと、インスタンスがその状態にとどまる時間である。

ここでは、「操作盤」が「ロボット」に送る指令イベント、「ロボット」が「操作盤」に送る指令完了イベント、「センサー」が「ロボット」に送る異常イベントに通信の遅延を設定した。「操作盤」は「ロボット」から指令完了のイベントを受けるたびに毎回同じ指令イベントを即時送信するようにした。センサーからの異常イベントはランダム時間で発生させた。ロボットはセンサーから異常イベントを受け取ると操作盤に異常を受けたことを知らせるイベントを送信する。正常動作

としてはセンサーが異常イベントを送信した場合、操作盤は異常が送信された時点以降の指令イベント送信を停止しなければならない。しかしこの設定は異常が送信されたにもかかわらず、指令イベントの送信が確認されてしまった。これはこのモデルがセンサーから送信されるイベントの遅延を考慮していなかったからである。この問題回避のため、「操作盤」と「センサー」に関係 (R49) づけを行い、「センサー」から直通で異常の発生を「操作盤」に知らせるようにした。なおモデル改良後も異常イベントの遅延の数値によっては動作不良が確認された。原因は「ロボット」から送信されるイベントの遅延時間より異常イベントの遅延時間を大きく設定したことであった。この設定ではロボットの指令完了イベントが異常イベントよりも早く操作盤に到着する場合が存在するためである (図 3.16, 図 3.17 参照)。

3.4.2 作業プロダクト

分析作業の成果物として以下のドキュメントを得た。

- 情報モデル (図 3.15)
- 状態モデル (図 3.18)
- 通信モデル (図 3.19)
- オブジェクト属性に関するテキスト出力
- 状態に関するテキスト出力
- イベントに関するテキスト出力
- 関係に関するテキスト出力

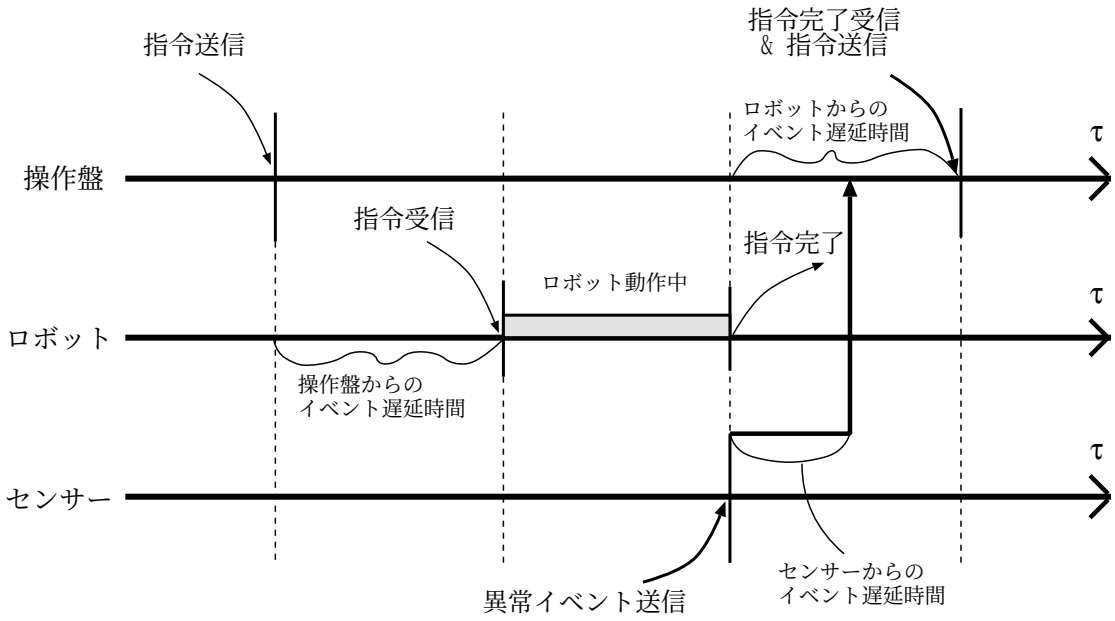

図 3.16: 正常に動作する時のイベント遅延時間設定

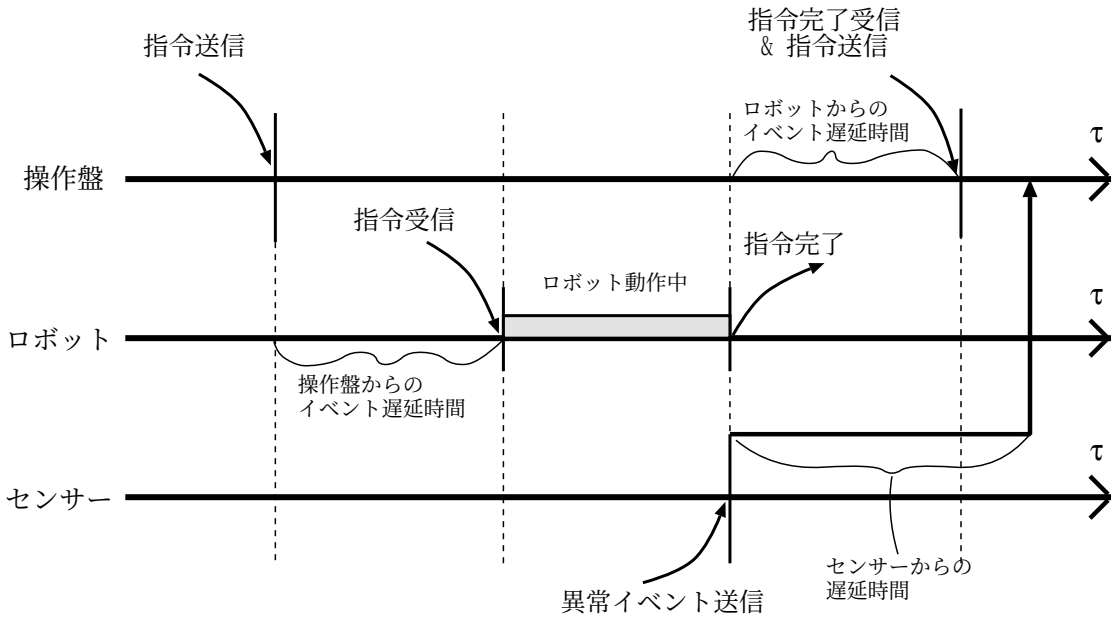

図 3.17: 正常に動作しない時のイベント遅延時間設定

図 3.18: ObjectBench 上の状態モデル

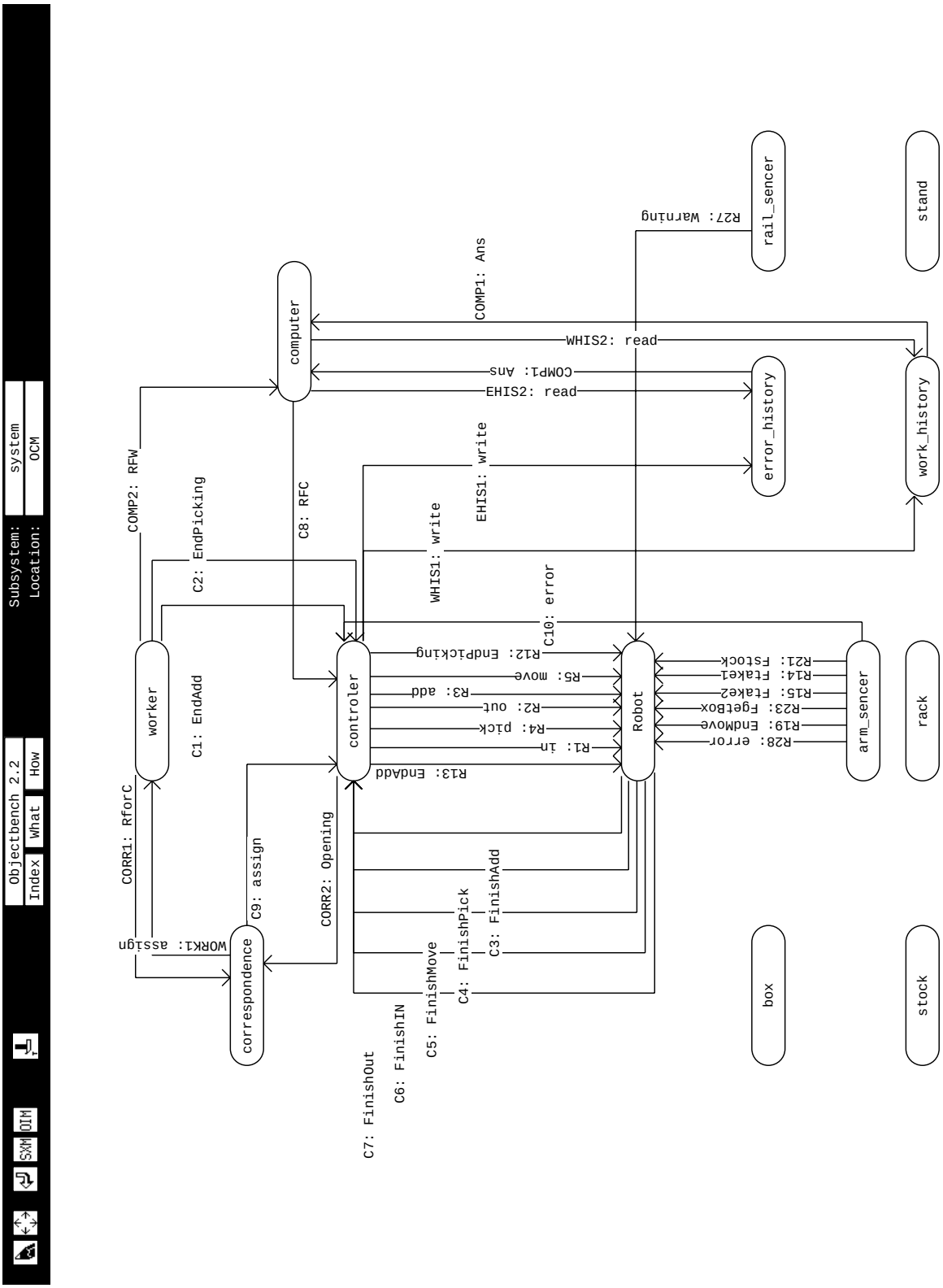

図 3.19: ObjectBench 上の通信モデル

第 4 章

議論

4.1 方法論を問題に適用した時の利点/問題点

オブジェクト指向方法論にもとづいて分析を行った際に役だった点、困難だった点についてまとめる。

利点

方法論を問題に適用した時に作成されるドキュメント群が利点の１つである。方法論を導入するとドキュメント体系が作られ、異なった問題ごとに同じ形式 (フォーマット) のドキュメントが蓄積されるようになる。これらの成果物 (特にモデル図) は、分析作業そのものを表現しているので別の類似した問題の分析に流用できる可能性を秘めている。
分析に使われる資料 (分析途中のモデル図他) はそのままでドキュメントとしての使用価値がある。特にモデル図はシステム開発者と発注者の接触点になると思われる。従来は要求使用を詰める段階で、開発者と発注者の間に共通の言語が無かった。このため開発者の思い込みや、発注者の頭の中に隠れている仕様を洗い出せず、開発途中で致命的な変更をうける可能性を持っていた。また致命的ではないにしても、あきらかに分析段階で洗い出せなかった仕様変更がだらだら続く状態は、ソフトウェア開発において多大な負荷となる。モデルを接点とした要求仕様の洗い出しは、すくなくともこのような状態を回避する手段になると思われる。本稿の事例ではエンジニアとのミーティングにモデル図を使用したが、意志の疎通は十分であったと思われる。隠れた仕様が洗い出された例として、移動ロボットの運搬能力があげられる。まず分析の過程でロボットの運搬能力が疑問になった。というのは、持てる荷物の個数が１つから２つに増えた時点でロボットの状態変化が飛躍的に複雑になるからである。ここで現段階の仕様では１つしか持てないが、近い

将来２つ持てる仕様になることが確認された。初期の分析ではロボットオブジェクトに、運搬されるオブジェクトを関係づけていたが、ロボットを本体と腕のオブジェクトに分割し、腕オブジェクトと運搬されるオブジェクトを関係づけたのは将来の変更に耐えるためである。

問題点

方法論を導入する際、従来の設計法からのパラダイムシフトに予想以上の労力が払われた。項目として

- オブジェクト指向の概念の理解
- 方法論の理解
- 開発関係者への分析パラダイムの説明

があげられる。方法論を理解するには２つの段階を理解する必要がある。１つは記述体系におけるシンタクスとセマンティクスであり、１つはそれを利用した作業プロセスである。とくに作業プロセスの部分は重要で、道具はそろったが使い方が分からないという状況に陥りやすい。本事例で採用した方法論ではこの作業プロセスについての提案が明確でなく、試行錯誤で分析を行うこととなった。

4.2 Shlaer/Mellor 法の利点/問題点

実務問題に適用した知見から、Shlaer/Mellor 法に対して、方法論としての評価を与える。

4.2.1 情報モデルによる分析

情報モデルは全てのオブジェクトと関係を定義するものであり、状態モデルや通信モデルに影響を与える。状態モデルは情報モデルで定義されたオブジェクトごとに作成され、通信モデルに記述される外部イベントは情報モデルで関係づけられているオブジェクト間で定義される。状態モデル/通信モデルの洗練による新たな変更点は情報モデルに反映される。プロセスモデルから状態モデルへの変更要請は見られなかった。方法論としてモデル間の整合性は良く、一貫したモデルの把握が容易に行えることが利点である。
また、情報モデルでの関係が明確に定義されているところが大きな特徴である。関係を考えることはインスタンスレベルでのシステムの構造を考えることである。分析/設計の境界は明確にされ

ていないが、実行可能なモデルが作成でき、構築したモデルに対して動作シミュレーションが行えることは、その後の設計段階へのモデルの変更が容易に行えることを示唆している。

4.2.2 リアルタイム性の考慮

Shlaer/Mellor 法はリアルタイムシステムの分析を考慮した議論を展開している。なかでも制御スレッドの概念におけるアクション時間と居住時間の導入は、モデルに時間情報を持たせることを可能にし、システム動作の分析をより深く進めるために有用である。これはシステムの安全性を示すうえで有効な手段となる。

4.2.3 並行状態の定義

並行状態と情報モデルの対応が明確に示されている。オブジェクトにおける並行状態は、オブジェクトのスーパータイプ/サブタイプ構造 (**is-a 関係**) に対応づけされている。共通な遷移列はスーパータイプの状態モデルで保持し、並行状態はサブタイプごとの状態モデルとして記述できる。これにより状態モデルにおけるオブジェクトの挙動を情報モデルに反映させる明確な手段を得たことになる。すなわち、並行状態を認識した時点でそのオブジェクトは情報モデルにおいてスーパータイプ/サブタイプ構造をもつことが分かり、各並行状態に対応するサブクラスを作成する。

4.2.4 内部状態の記述の問題

状態モデルにおける状態は階層構造を持たないので、記述能力にやや難点がある。オブジェクトの状態遷移を考えた時、内部状態の記述が使えると図が簡潔になり見通しがよくなる。しかし Shlaer/Mellor 法では、詳細化した状態遷移列はそのまま元の図に埋め込む方針を取っている。

4.2.5 分析プロセスの整備の問題

各モデルの記述力や意味付けの完成度は高いが、モデルの作成過程について詳細な手順は示されていない。これは分析する問題によって、知識の定式化の過程はさまざまであり、一様に言えないことが原因である。そこで、本事例研究で行われた作業履歴から具体的なモデル作成プロセスと分析過程全体を通した分析のポイントを示し、類似の問題に対する指針を与える。

情報モデルの作成プロセス

1 システム構成図を作成する。

2 構成要素をオブジェクトとして認識する。

3 関係するオブジェクトどうしを見付ける。

4 関係を付けたオブジェクト間で関係のイメージを作図する。

5 関係のイメージ図にしたがってオブジェクトをアークで接続する。

6 オブジェクトに識別子を記入する。

7 関係の定式化にしたがって参照属性を記入する。

状態モデルの作成プロセス

1 システムの主要動作を仕様化する。

2 主要動作を実現する重要なオブジェクトを特定する。

3 主要動作の目的別に動作モデルを作成する。

4 同一オブジェクトに対して作成した動作モデルが複数ある場合、共有できる状態と共有できない状態の識別を行う。

5 同様に共有できるイベントと共有できないイベントを識別する。

6 共有できる状態とイベントに共通の名前をつけ、状態遷移図を統合する。

7 アクションを記述する。

分析作業のポイント

1 履歴情報はオブジェクトとして表現する。

2 必要なイベントは状態モデルを作成する時に認識される。

3 イベントが外部イベントか内部イベントかを意識しながら、状態モデルのアークにイベントのラベルづけを行う。

4 通信モデルで外部イベントとして認識したイベントをもれなく記述する。

5 状態モデルの遷移の分岐点に着目して、分岐条件に他のオブジェクトのデータに対する参照がないか分析して見る。

4.3 CASE ツールの評価

本事例研究で使用した ObjectBench に対して評価を与える。

表 4.1: ObjectBench の作業フィールドと方法論のモデルの対応表

方法論のモデル	ObjectBench の作業フィールド	表現形式
情報モデル	情報モデルのページ	図式
状態モデル	状態モデルのページ	図式
通信モデル	通信モデルのページ	図式
プロセスモデル	アクションフィールド	コード

4.3.1 方法論とその実現法との対比

ObjectBench は Shlaer/Mellor 法による分析に特化した分析作業支援ツールである。よって分析者は方法論がどのようにツール機能として実装されているか知る必要がある。方法論の各モデルと ObjectBench の作業フィールドの対応を表 4.1 に示す。

対応関係を見て分かる通り、情報/状態/通信の３つのモデルは、方法論の表現形式をモデルのセマンティクスを含めてほぼ完全に再現している。ただしプロセスモデルに関しては図式表現として再現されていない。これは方法論において、プロセスモデルがアクションの記述をアルゴリズムのレベルで図式表現 (データフロー) したものであることに関係する。つまりツールでは、アクションを状態内の抽象的な動作記述に止めず、モデル内のデータを扱うプログラムコードとして記述してしまうのである。

4.3.2 モデル構築のサポートとその効果

ObjectBench は大きく分けると以下の２つの側面から分析を支援していると考えられる。

- Shlaer/Mellor 法におけるモデルのシンタクスとセマンティクスチェック
- シミュレーションによる分析モデルの調査/観察

ツールは分析という人の知的作業そのものはサポートしないが上記の２つの支援機能を使うことで、より深い分析の手助けを行う。特にシミュレーションの機能は、分析したモデルに妥当性を与え、以後の設計段階において有用な情報を提供する。非同期で発生するイベントに対して正常なモデルの振舞いを示し、設定したイベントの発火タイミングに対して正常なイベント列が得られるようにモデルを変更/作成していくことは、モデルの安全性を高めることであり、システムの安全性について議論できるようになる。

4.3.3 インターフェースによる作業プロセスの強制

ツールのインターフェイスが状態モデルと通信モデルの作成過程に１つの強制を与えている。ツールのイベント入力の流れは、まず通信モデルでオブジェクトが受けるイベントを全て入力する。つぎに状態モデルの各遷移アークにオブジェクトに定義されているイベントから選択してラベル付けする。このイベント入力の方法は、通信モデルと状態モデルの間のラウンドトリップを強制させる。これの良い面としては、外部イベントと内部イベントの違いを常に意識した分析を分析者に強制し、イベントの定義漏れを防止させる効果もある。しかし、本来なら必要なイベントの認識は状態モデルで行うべきであり、そこでイベントの定義 (入力) ができる方が作業として自然である。また、認識するイベントが多くなるとモデル間を行き来するためのツールのオーバーヘッドが多くなるので改善の余地はあるだろう。

4.3.4 支援機能への要請

以下、分析作業におけるツールの支援機能を要請する。

- 図の自動配置機能。エディタとしての支援機能だが、地味なようでいて実は図の理解のために重要である。特にオブジェクトは認識、分割、削除、関係の付け変え、が頻繁に行われるのでエディタとしての能力が高くないと分析作業以外に多大な労力を使う羽目になる。
- インスタンスの対応関係の表示機能。関係アークを定義する時、図 3.2 に示したような集合の関係図を利用したい。
- 情報モデルのクラスタリング機能。ツール上で簡単にオブジェクトのクラスタリングを行いたい。例えば、情報モデルでクラスタリングしたいオブジェクトの範囲指定をするとSubSystemフィールドにサブシステムを時動的に作ってそこに移動させるような機能。
- オブジェクトのアクションに関する情報表示機能。アクションはあるオブジェクトが他のどのオブジェクトのどんなデータを参照/変換しているか、といった多くの情報を含んでいる。これに対する情報を表示して欲しい。
- イベント列生成機能。イベントに同期/非同期の定義をした後、あるオブジェクトからあるオブジェクト間で発生し得るイベント列を生成して欲しい。これはシミュレーションにおいてシステムに投入するイベント列として使える。

第 5 章

おわりに

5.1 まとめ

本研究はソフトウェアプロセスへのオブジェクト指向方法論の導入を援助するため、CASE ツールを用いて事例研究を行った。問題として自動倉庫システムを取り上げた。方法論にはShlaer/Mellor法を採用した。

分析はある程度まで手作業で行い、その後 CASE ツールを使用した。分析内容は情報モデルの作成、状態モデルの作成、通信モデルの作成に分けられた。情報モデルの作成ではオブジェクトの認識と関係の形式化を行った。状態モデルの作成ではシステムの主要動作を状態モデルに反映させた。通信モデルの作成ではオブジェクトが受けとるイベントの定義と異常イベントの認識を行った。CASE ツールを使用した分析ではモデルのシミュレーションを行い、「ロボット」と「操作盤」と「センサー」間で正しく動作しないようなイベント送信時間の設定を発見し、なぜ動作しないのかをつきとめた。

最後に、分析の過程や結果から、オブジェクト指向分析に対する知見を深め、方法論とツールに関する議論を展開した。ここでは特にモデルの作成過程と分析作業のポイントを示し、また CASE ツールへの支援機能の要請を提示した。

5.2 今後の課題

分析作業について。

- モデルをさらに詳細化する。各オブジェクトのデータがまだ詳細化されていない。
- シミュレーションの範囲を拡大する。現在は３つのオブジェクトを対象にシミュレーションを行なっているが徐々にシミュレーション対象とするオブジェクトを増やす。

- 仕様変更に対する対処法を観察する。オブジェクト指向方法論を問題に適用した時、一般にシステムは仕様変更に強くなると言われている。これを実際に確かめる。

ツール機能について。

- 版管理機能の調査。Objectbench は版管理データベースと連動して過去に行なった分析作業を版管理できる。版管理は CASE ツールにとって重要な機能の１つであるので調査する必要がある。
- 共同分析支援の調査。分析作業は複数人で行なわれることもある。人の協調作業を支援する機能は CASE ツール今後ますます研究されていく分野であり、調査する必要がある。

参考文献

[1] S.Shlaer/S.j.Mellor 著, 本位田真一/山口亨 訳, オブジェクト指向 システム分析, 近代科学社.

[2] S.Shlaer/S.j.Mellor 著, 本位田真一/伊藤潔 訳, 続オブジェクト指向 システム分析, 近代科学社.

[3] j. ランボー/M. ブラハ/W. プレメラニ/F. エディ/W. ローレンセン 著, 羽生田栄一 訳, オブジェクト指向方法論 OMT, トッパン.

[4] 中谷多哉子, オブジェクト指向 CASE はなぜオブジェクト指向か, ソフトウェア工学 93-17.

[5] 中山裕子/吉田浩之, オブジェクト指向を用いた業務分析についての一考察, ソフトウェア工学 103-11.

[6] 二本木伸佳/山本修一郎/安原隆一, 業務分析ツールの適用評価, ソフトウェア工学 98-11.

[7] 山城明宏/吉田和樹/入内島裕子/斎藤悦生, 画像ファイリングシステムへの適用, 情報処理 Vol.35.No.5.

[8] 梶原清彦/浜田雅樹, ディジタル移動通信システムへの適用, 情報処理 Vol.35.No.5.

[9] 本位田真一/伊藤潔, OOA/OOD の上流 CASE, '91/8/20 日本ソフトウェア科学会サマーチュートリアル資料.

[10] Colin Potts and Glenn Bruns, Recording the Reasons for Design Decisions, 10th ICSE,pp.418-427,1988.

[11] 倉谷/東田雅宏/藤枝和宏/鈴木潤一/落水浩一郎 著, オブジェクト指向分析の事例研究 (生産管理システム), '93 北陸先端科学技術大学院大学修士論文.

[12] 池田克則/落水浩一郎 著, PCTE による CASE ツールの統合とその応用, '93 北陸先端科学技術大学院大学修士論文.