オブジェクト指向によるモデリング――例題 3

羽 田 昭 裕

1. はじめに

例題を通じて，開発のプロセスを示し技法の適用可能性を検討することが，与えられた課題である．技法を客観的に見る方法として，問題と方法に対する M. Jackson[4] の視点を採用した．この文献[4]では，エレベータ問題を，単純制御フレームの代表として詳しく言及している．これを手掛かりとして，問題と方法の適合度が明確になるように開発プロセスを紹介する．

2. アプローチ

2.1 課題の設定

開発工程をシームレスにするには，設計の空間を絞り込むことが必要である．M. Jackson[3]は，問題フレーム（problem frames）がこの絞り込みに有効なアプローチである，と主張する．問題フレームは問題自身のパターンであり，問題の構造（主要部分および，その特徴と関連）と解の構造から構成される．このようなシステマティックな問題の理解の後に，厳密な記述が可能になる，と述べている．また，文献[3]では問題の構造に適合した方法を採用することで歪みの無い開発を促進するとしている．線形計画法など OR（operations research）の分野では，問題の構造・解の構造・解法を明確に関連付けているため，モデルの作成と実行との距離は短い．これは，このアプローチが妥当であることを示唆している事例である．

そこで，例題の問題フレームを特定し，その解の構造をオブジェクト指向的な分析パターンと設計の指針で示すというアプローチをとる．そして，この例題に対するオブジェクト指向という方法の妥当性を検証する．

2.2 問題フレーム

エレベータの問題は，単純制御フレーム（simple control frame）に相当する．文献[4]によれば，このフレームは次のような構造を持つ．単純制御フレームの主要部分は，「制御系」と「制御対象領域」と「望ましい振る舞い」である．各主要部分の特徴と関連は次の通りである．制御系の目的は，「望ましい振る舞い」を維持させることである．制御対象領域の振る舞いは，制御系の振る舞いに関わらず持つ与えられた性質と，制御系で実現されるべき要求された性質とに分けて明示的に記述される．望ましい振る舞いは，与えられた性質を前提にして実現される．単純制御フレームでは，制御系と制御対象領域の接続は信頼性が高く，雑音も，問題となる遅れも無いもの，と仮定している．これが仮定できない場合は，接続フレームを組み合わせる．以上が，この問題フレームの枠組みである．

2.3 分析パターン

問題とその解法という枠組みは，オブジェクト指向の世界ではパターンと呼ばれ，クラス構造として表現される．制御システムに関する成果のうち，以下のものを参考

にして分析パターンを定めた．文献[5]では制御システムに関する設計パターンが示されている．これは単純制御フレームと接続フレームが混在した問題に対応する解法のパターンである．文献[8]は単純制御フレームに対応する要求分析法を示しており，先に示した制御系が持つ二つの性質を自然な性質（NAT）と要求された振る舞い（REQ）に分類している．自然な性質の制約の中で，要求された振る舞いを記述する．文献[10]では単純制御フレームの主要部分である制御系・接続・制御対象領域が system・interface・sensor/actuators として層別されている．

分析パターンを図1に示す．以下の表記は，UML（Unified Modeling Language）1.1 に準じている．この構造は文献[8]と同様であるが，資源へのアクセスが矛盾/競合する場合を想定して，「割当て」を導入した．これは文献[6][7]を参考にしている．

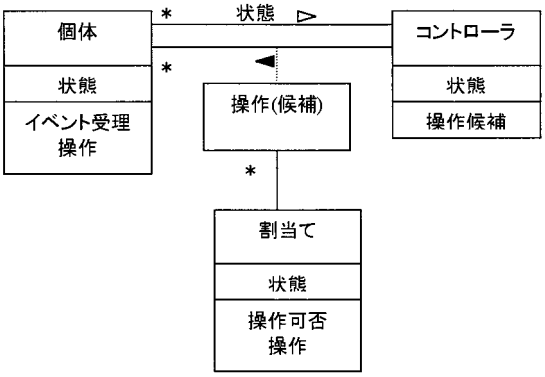

図 1　単純制御フレーム用のオブジェクト分析パターン

問題フレームと分析パターンは次のように対応する．実世界の一部である適用領域（application domain）を対象とする問題フレームと異なり，オブジェクト指向分析はシステム領域を対象とする．そこで単純制御フレームでは，制御系をオブジェクト・モデル化する．この問題フレームでは制御が本質的なシステム化対象なので，分析段階から制御を明示的に表現する．

まず，二つの性質に対応させて，制御系を個体とコントローラに分割した．個体は制御対象領域と共有現象を持ち，コントローラは狭義の制御を行なう．与えられた性質は個体の状態遷移として表現され，実現されるべき性質はコントローラの状態遷移として表現される．

次に，制御対象領域と個体との共有現象は2種類に分ける．制御対象領域に主導権がある場合，現象を個体はイベントとして受理する．制御系に主導権のある現象は，個体が操作信号として送り出す．例えば，センサーから送られた信号はイベントとしてある個体に送られ，いずれかの個体からアクチュエイターに操作信号が送られる．そして，コントローラの振る舞いは，個体の状態と操作に基づいて表現する．すなわち，個体の状態を監視して，自分の状態遷移を決定し，その状態に応じた操作を決定する．ところで制御系の望ましい振る舞いは，優先度・時間的制約を判断する必要がある場合が多い．この部分を担当する「割当て」を独立させる．「割当て」は，コン

トローラが決定した操作の優先度を判断して，その場で可能な操作を個体に依頼する．

以上のような対応により，オブジェクト間のインタフェースを明確に定めることができる．また，「個体」「コントローラ」「割り当て」のようなクラスのカテゴリは，UML のステレオタイプに対応する．

2.4 設計の指針

アーキテクチャ設計は視野外である．するとメッセージングと状態遷移の方式を設計することが中心となる．ここでは詳細を示さないが，例題と関連する部分を述べる．

2.4.1 メッセージング

接続の問題を捨象したため，オブジェクト間の関連と起動の方法に焦点を絞る．オブジェクト間の関連は，相手のオブジェクトの状態を参照する関係とするか，操作を働きかける関係とするかであり，起動の方法はイベント駆動型（eventual）にするか，周期的（periodic）にするかである．選択の基準は文献[5][9]等に具体的に示されている．

2.4.2 状 態 遷 移

状態遷移の設計に関する指針としては，第一に statechart の特徴づけの評価，第二に文献[11]の state パターンの評価を示す．

statechart は，UML の状態図（state diagram）の原型であり，元来リアルタイム系制御システムを記述することを目的としている．具体的には，有限状態機械（finite state machine；以下，FSM）に階層化・直交化・ガード（guard）を導入することで状態遷移の見とおしをよくしている．まず，階層化は幾つかの状態をグループ化した上位状態（super state）の入れ子として表現する．上位状態を継承する形で記述することにより，共通する遷移の記述を節約できる．次に，直交化により，並列する動作を表現する．逐次的な状態遷移ごとにグループ化することで，並列処理を記述する複雑さを解消している．最後に，ガードは遷移するか否かの条件付けであり，一部の状態を変数で代替する．これはメモリ付きの FSM を表現している．以上から，制御フレームに伴う状態記述の複雑さ・困難さを解決する要素を備えていると評価し，statechart を図法として採用する．

state パターンは，状態をオブジェクトとして表現することにより，状態遷移を管理するものである．文献[11]では，単純な FSM に対する実現方法を示しており，状態の階層化・直交化やメモリ付き FSM については言及していない．この意味での state パターンの応用形は文献[8]に示されている．ところで，「割り当て」により状態遷移に伴う操作が無効となる場合は，状態遷移を巻き戻す必要がある．これは，取り消し可能なオペレーションを実現する command パターン[11]などを採用して，状態遷移を複雑にしないことが必要である．statechart の FSM は，このように改造できる．state パターンとその応用をどのように選択するかは，個々の問題に応じて判断する必要がある．

3. エレベータ問題の分析

例題だけでは，問題の記述として自由度が大きすぎる．この問題の一般的な記述[1]，および M. Jackson[2]の記述も参考とした．

3.1 分析の道筋

制御対象領域を”エレベータ”と”階”と”扉”に分割した.”扉”はエレベータに付属した内側の扉のみに注目し，各階にエレベータ毎にある外側の扉は内側の扉の開閉に応じて開閉するものとして無視する.各領域の制御対象は次のように抽出した.階の領域に属する制御対象は，上（下）へ行きたいことを要求するための「上昇（下降）ボタン」である．エレベータの領域に属する制御対象は，ドアを開く（閉じる）よう指示するための「開（閉）ボタン」，停止したい階を要求する「停止階選択ボタン」，エレベータのある階への停止を検知する「エレベータ到着センサー」およびエレベータの上昇・下降を操作する「モータ」である．扉の領域に属する制御対象は，扉の開閉を検知する「ドアセンサー」，扉の開閉を操作する「ドア」である．これらの対象を基に，与えられた性質（表1）と要求された性質（表2）に分けて洗い出した結果を示す．なお，表中の□は採用した性質，■は採用しなかった性質である．

表 1　与えられた性質

□	エレベータは，n+1 階を経由すること無く，n 階から n +2 階に移動することはない.
□	モータの回転方向が上昇になっており，そのときにモータのスイッチがオフからオンになったとすると，エレベータは 一定時間以内に上昇を開始する.
□	エレベータがある階に到着すると，その階のエレベータ到着センサーがオンになる.
□	エレベータには，ドアの開閉ボタンや停止階を示すボタンがある.
□	各階には，上または下へ行きたいことをリクエストするボタンがある.
□	最上階には，そこより上に上がるボタンが無い．最下階には，そこより下に下がるボタンが無い.

表 2　要求される性質

□	エレベータ内部の停止階選択ボタンが点灯している階に止まらずに，その階を通過してしまうことはない.
■	エレベータがある階に止まっていて，その階の上（下）方向ライトが点灯していないときには，上（下）方向には出発しない.
□	ある階の上昇（下降）ボタン（未点灯）が押されると，ボタンに対応したライトが点灯する．この点灯はエレベータがやってきてこの階に止まり，上（下）方向に出発していくまで消えることが無い.
□	エレベータがある階で停止しない限り，エレベータのドアを開くことはない.
□	エレベータ内の上昇ボタンと下降ボタンは，リフトが到着すると点灯しているボタンが消灯される.
■	エレベータ内の緊急停止ボタンが押されると，エレベータは止まる.
□	エレベータ内の開ボタンが押されると，解除されるまで，ドアは閉まらない.
□	エレベータ停止時に，エレベータ内の開ボタンを押すと，ドアが開く.
□	エレベータ停止時に，エレベータ内の閉ボタンを押すと，ドアが閉まる.
□	エレベータ内の開ボタンと閉ボタンは同時押せない.
□	エレベータ内部の停止階選択ボタンが点灯している階に止まった時には，そのボタンは消灯する.
□	エレベータ停止時に，フロアの上昇（下降）ボタンを押すと，ドアが開く.

3.2 モ デ ル

分析モデルをクラス図（図2）と状態遷移図（図3）で示す．まず，クラス図では，制御対象をクラスとした．そして，制御領域をその構成体（composite）とし，コントローラを制御領域毎に設けた．コントローラと個体の関係は，便宜上コントローラ

と構成体の関係で表現した．多重度は，m 階の建物に，n 台のエレベータがあるとして記述している．また，各種のランプはボタンの状態に従って，点灯/消灯するものとして省略した．

あるフロアで上りの要求が出た時，どのエレベータがそこへ迎えにいくのか，という問題を解決する機能は「割り当て」で実現される．これは文献[2]での「認可」プロセスの役割に相当し，制御系を人間の意思決定をモデル化したものと見ると，全体のゴールを目指す決定メカニズムと捉えることができる．「割当て」は，記述を省略した．

以下の点から，このモデルは要求を満足していると評価した．①与えられた性質が個体の振る舞いとして，要求された性質がコントローラの振る舞いとして記述できている．②コントローラの振る舞いが，個体の状態・操作の組み合わせで記述されている．また，以上のことは，このモデルが問題フレームに適合していることを示している．

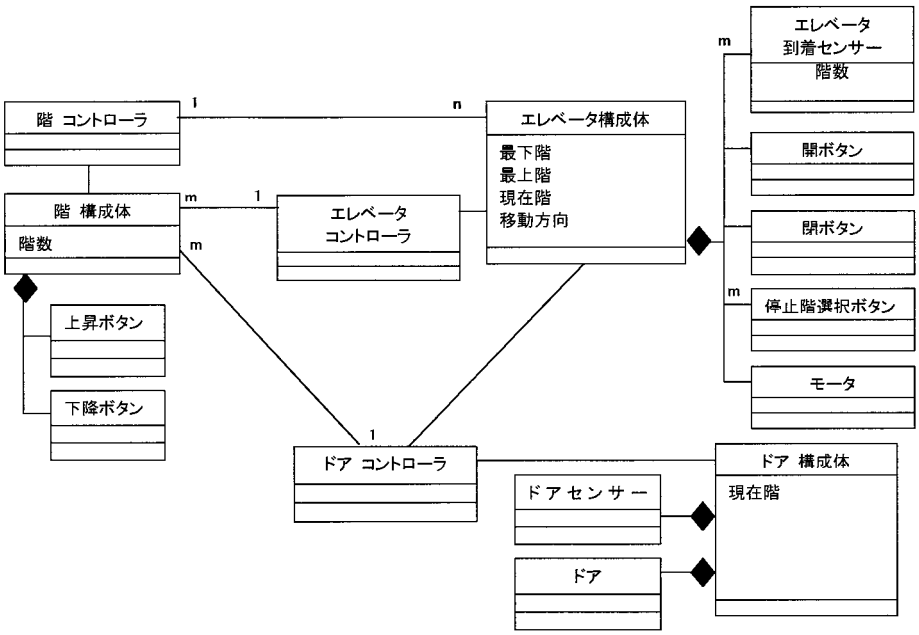

図 2　エレベータ問題のクラス図

状態遷移図は，エレベータコントローラのみ示す．個体の状態は，イベントに従って変化するという単純な状態遷移なので図示しない．例えば，ボタン類は押されると状態が変化し，その状態を isOn メソッドで参照できる，としている．

図 3 では，「現在の移動方向＝上昇（下降）」を上位状態として表現している．これは，エレベータがある階に停止している状態から扉が閉じた後に上昇するか下降するか，移動中に次の階で止まるか通過するかは，どちらもそれ以前にエレベータが上昇してきたか下降したきたかに依存するため，共通の状態として記述を簡略化したものである．なお，ガード条件は OCL（Object Constraint Language）に準拠した疑似

コードで記述した.

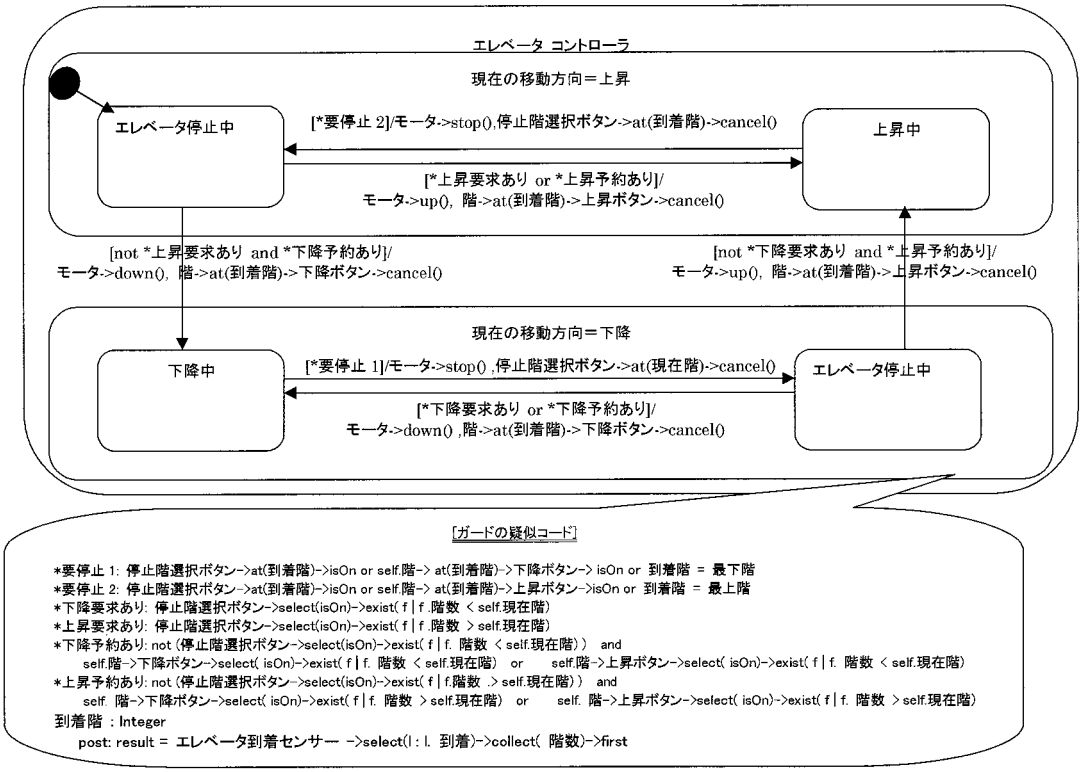

図 3　エレベータコントローラの状態遷移図

3.3 設計の方針

設計のポイントである状態遷移とメッセージングの管理について，以下の方針を定めた.

3.3.1 メッセージング

個体とコントローラの関係では，コントローラが主導権を持つこととした．このシステムは，操作対象の制御状況から実行できる操作が決まっている，コントローラの状態は複雑なガード条件により決定する，という性質がある．このような性質と実時間制約を考慮すると，エレベータの到着や接近をイベントとして発生させるよりも，コントローラが個体の状態を参照する方が簡明であるからである．この決定に従い，コントローラの起動は周期的にタイマーが起動する形を取った.

3.3.2 状態遷移

状態を変数で持つという最も単純な形で，状態遷移を実装することとした．以下の性質から，カプセル化のメリットが実装の複雑さに引き合わないためである．その性質とは，状態の状態遷移がガード条件によって制約されることからガード条件がカプセル化されていれば，状態の遷移や操作は複雑ではないこと，状態の階層が浅いことである.

4. 評　　価

単純制御フレームの解を構成する手段として，オブジェクト指向分析・設計は充分な記述力を持っている．今回の記述では，実行可能性を示すことは範囲外としたが，要求を充足できることは示せた．

但し，以下に述べるような，オブジェクト指向開発方法において制御システムの問題を記述する標準の枠組をもたない状況で，技法の適用可能性を判断することは困難である．

出発点である問題を記述するために，ユースケース（use　case）[12]を用いることが現在の主流となっている．ユースケース・モデルは，責務（responsibility）を明示する点で，分析以降のモデル図と異なっている．個々の責務は具体的な文脈（context）で決まる．従って，文脈と要求から構成される問題を記述する基本的な枠組みを，ユースケースは持っている．

しかし，単純制御フレームで記述する要求は，人が要求しシステムが反応するという型の責務ではない．制御系は人間の意思決定に対応していると想定され，能動性を持っている．そのため，ユースケースとは別の形で問題記述する方法が必要である．このことは，今回の問題フレーム・分析パターンが，意思決定をモデル化したシステム[13]と同型であることからも推測される．文献[9]では，構造化分析・設計から全体文脈図（context　diagram）を導入することにより，制御システムの特徴を明確にし，ユースケースを補完する試みを示している．

以上から，有体物（tangible）に対応したものではなく，要求をオブジェクト・モデル化した結果の要素[12]として制御系を捉える．要求された性質を維持するという制御系の性格から当然の帰結である．この例題では，エレベータシステム本体(subject)の振る舞いを，コントローラという対象（object）に対応させている，とも見える．いずれ，問題や要求を記述することが起点であり，そのための枠組を従来のユースケース・モデル以外にも揃える必要がある．

参考文献
［1］ J-R. Abrial, The B-Book, 1996, Cambridge Univ. Press.
［2］ M. Jackson, 山崎・大野訳, システム開発法 JSD, 1989, 共立出版.
［3］ M. Jackson, 田村訳,“ 問題, 方法, そして特化 ”, 情報処理 vol. 36, no. 1, 1995.
［4］ M. Jackson, 玉井・酒匂訳, ソフトウェア博物誌-世界と機械の記述, 1995, トッパン.
［5］ P. Molin & L. Ohlsson, “ The Point and Deviation Pattern――Language of Fire Alarm System ”, Pattern languages of program design 3, 1998.
［6］ S. Shlaer and S J. Mellor, 本位田・伊藤監訳, 続・オブジェクト指向分析-オブジェクト・ライフサイクル-, 1992, 啓学出版.
［7］ S. Shlaer & N. Lang, Shlaer-Mellor Method : The OOA 96 Report, http : //www. projtech. com/
［8］ S. Faulk, et all, The Core Method for Real-Time Requirements, IEEE Software, Sep. 1992.
［9］ B.P. Douglass, Real-Time UML-developing efficient objects for embedded systems, 1998, Addison Wesley.
［10］ J R. Ellis, Objectifying Real-Time Systems, 1985, Prentice Hall.
［11］ E. Gamma , et al, 本位田・吉田監訳, デザインパターン, 1995, ソフトバンク.

［12］ I. Jacobson, 西岡他訳, オブジェクト指向ソフトウェア工学 OOSE, トッパン, 1995.
［13］ 羽田昭裕, “ OMT を使った「発電プラント運転員チーム行動シミュレーションシステム」の開発 ”, 技報, Vol. 14, no. 1, 1994

執筆者紹介　羽　田　昭　裕（Akihiro Hada）
1984 年一橋大学卒業．同年日本ユニシス(株)入社．意思決定支援ソフトウェアの開発・適用に従事．その後，業務システムとその基盤の要求分析・開発に従事．現在，情報技術部技術研究開発室に所属．情報処理学会会員．