統合型システム構築における複数視点からのOOAについて

2J−7

入江 豊　斉藤 悦生　本位田 真一
株式会社 東芝　研究開発センター

1 はじめに

複雑な問題領域を分析し，モデル化するための手法としてOMT [1]に代表されるオブジェクト指向分析(OOA)が注目されており，仕様変更に対するソフトウェアの保守性，再利用性の向上に寄与することが期待されている．しかし，多数のシステム参加者（オペレータ）が参加するシステムでは，システム運用の変更が発生した場合，問題領域のオブジェクトに対するアクセスが手順が変化するためにダイナミックモデルの大幅な変更が発生しやすい [4].

本論文は，分析段階での視点数の不足が動的モデルのフレキシビリティを失わせているという観点に立ち，運用変更への対応を考慮したダイナミックモデルの構築を，イベントのとり扱いを中心に検討する．

2 OMTの概要

OMTでは，問題領域をオブジェクトモデル（情報モデル）・ダイナミックモデル（動的モデル）・ファンクショナルモデル（データフローによる機能モデル）3つのモデルにより表現する．

このうち，ダイナミックモデルは，オブジェクトを状態遷移機械とみなし，オブジェクト毎の動的特性を，状態遷移グラフによって表現する．モデル中の状態または遷移パスに，オブジェクトが行なうべき処理であるアクションを記述する．状態遷移機械を直接的にマッピング可能な言語を仮定すると，このダイナミックモデルの概念は，プログラムの上流設計に直接的に反映可能なものになる．この状態図を作成する手がかりを得る手段として，典型的なシナリオを想定し，イベントトレース図を作成する．

OMTでは，この状態遷移図の表記法としてHarelのstatechart [2] を用いている．また，statechartをオブジェクトの状態機械として利用する考え方について [3]にて扱われている．本論文は主としてこの表記に準じている．

3 イベントトレース図

OMTの手順に従うと，まず最初にオブジェクトモデルの構築によって，オブジェクトの集合が得られる．

次に，オブジェクトのダイナミックモデルを構築するために，まずイベントトレース図を作成する．これは，オブジェクトのふるまいの分析において，まずオブジェクト間でやりとりするイベントが問題領域より得やすい（分析者に明示される）ことを意味する．イベントトレース図では，通常は（クラスでなく）インスタンスを仮定ため，以下では，オブジェクトとはインスタンスを意味するものとする．また，簡単のために，対象問題のすべてのイベントトレースを1つの図上に実施したものと仮定する．一般には，トレース図内に同一種のイベントが複数個現れるが，その場合のダイナミックモデルへの反映は本論文の主題ではないため，すべてのイベントのユニーク性を仮定する．

ここまでの段階において，問題領域よりオブジェクトとイベントの集合が獲得される．

$$S=\{o_1,o_2,\ldots,o_l\},\ E=\{e_1,e_2,\ldots,e_m\}$$

また，オブジェクトの集合Sがシステム全体を表現するものとする．実際のオブジェクトモデルはクラスを扱うものであるが，ここでは簡単のため，クラス集合は，有限のインスタンス集合に対応可能なことを仮定する．

また，イベントトレース図から，上記のイベントのやりとりについて，どのオブジェクトからどのオブジェクトにイベントが渡されるか，その対応関係の集合Rが与えられる．個々の対応関係は，サービスのリクエスト，イベント，サーブ関係の三つ組 $<r,e,s>$ として表現できる．

$$R=\{<r_1,e_1,s_1>,\ldots,\}$$

$$r_i\in S,\ s_i\in S,\ e_j\in E,\ (1\le i\le l,\ 1\le j\le m)$$

さらに，イベントトレース図には，これらの時間的順序関係が表現される．これらをイベント順序列と呼ぶことにする．1つのイベントトレース図からは，次の1つの順序列Tが得られる．

$$T=[t_1,t_2,\ldots,t_k]$$

ここでは，三つ組 $<r_i,e_i,s_i>$ を t_i と表している．以上が，イベントトレースの作成によって，分析者に改めて明示される情報である．分析者（または設計者）に以後必要な作業は，このインスタンスの振舞いから「クラス」のダイナミックモデルを構築することである．

4 複数視点の導入

4.1 想定する視点の意味

多くのオペレータが運用に参加する事務処理を主目的とした統合型システムには，分析において次のような困難が見受けられる．

1. 問題領域のオブジェクトに関して十分な情報モデルは，例えば従来から用いられてきた帳票など，文書等から得られるのに対し，動的なモデル獲得に必要な材料は，十分には文書化されていない場合が多い．従って，仕様獲得のためには，問題領域専門家へのインタビューが不可欠である．
2. システムのすべてを包括的に知っている人物が少数である．または存在しない．

The OOA Method from Multiple Viewpoint for Constructing Integrated Systems
Yutaka IRIE, Etsuo SAITO, Shinichi HONIDEN
Toshiba, Research & Development Center

システムが大規模になればなるほど，この２つの条件が重なる場合が多いと思われる．この場合，問題領域に関する知識は個別に得られ，それらを融合する作業が必要になる．本報告にて，視点と呼ぶのは，問題分析において，インタビューされる立場の人々を想定している．

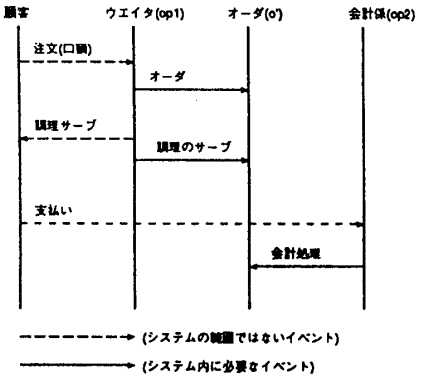

図 1: イベントトレース図の例

4.2 個々の視点から得られるイベント

個々の視点は，システム参加者，またはそれを代理するオブジェクトである．視点は必ずオブジェクトを構成するものとし，オブジェクト集合 O に含むものとする．１つの視点は，他のオブジェクト同様に，イベントトレース図中に１本の縦線として現れる．

これまでにリストアップしたオブジェクトの集合から，上記の意味の複数の視点にかかわるオブジェクトの集合 o_p を取り出す．すなわち，複数の視点オブジェクトに対して，イベントの授受関係をもつオブジェクト o' を列挙する．これは，イベントトレース図より容易に得られる．

オブジェクト o' が受信および発信するイベントを，その対象となる視点オブジェクト o_v 毎に取り出す．このイベントの集合を，$E \triangleright o' \triangleright o_v$ と表現する．また，このオブジェクト列から構成されるイベント順序列を T' とする．これもイベントトレース図より得られる．

$$E \triangleright o' \triangleright o_v = \{e | \exists o_v \exists o' < o_v, e, o' > \in R\} \cup \{e | \exists o_v \exists o' < o', e, o_v > \in R\}$$

$E \triangleright o' \triangleright o_v$ のみから構成される３つ組順序列 $T(o', o_v)$ は，イベントトレース図全体の順序列 T の部分順序列を構成する．

$$T(o', o_v) = [t_{i1}, t_{i2}, \ldots, t_{in}]$$

$$t_{ik} \in \{< o_1, e, o_2 > | e \in E \triangleright o' \triangleright o_v\}$$

4.3 最大抽象としての集約

視点オブジェクトは，それがシステムオペレータを表していることから，能動的なオブジェクトの性質を持つ．また，各々のオペレータは，端末が複数存在するシステムの場合，本質的に並列に動作するはずである．イベントトレース図は，相互に影響しあう状態遷移機械の可能なふるまい中の１つの動作であるが，新規システム分析において，可能なトレースがすべて得られるとは限らず，見落としの可能性も高い．

複数の視点オブジェクト $o_{v1}, o_{v2}, \ldots$ とやりとりを行なうオブジェクト (相対的に受動的なオブジェクト) o' が受容すべきイベントの順序列の最大集合として，o_{v1}, o_{v2} からのイベント列 $T(o', o_{vk})$ を並行マージした順序列集合が考えられる．並行マージ演算 ||| (CSP における parallel composition by interleaving と同様の意味) を用いて，このイベント列集合は，

$$tr = T(o', o_{v1}) ||| T(o', o_{v2}) ||| \ldots ||| T(o', o_{vn})$$

と表現できる．これは，意味的にはそれぞれの視点から見たオブジェクト要素を集約したものに相当する．

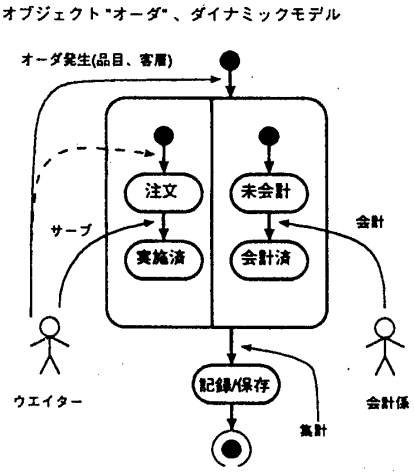

図 2: イベントとダイナミックモデル

5 おわりに

集約したイベント列集合 tr は，オブジェクトの”最大抽象”を表現している．イベント列 tr の中には，存在しえないイベント順序列も含まれる．例えば，あるオブジェクトを生成するイベントは消滅イベントよりも先行しなければ意味がない．“最大抽象”を強調する理由は，この初期の抽象設計を出発点とし，オブジェクト内部に制約をあてはめる形で詳細設計に向かうガイドラインとなること，および状態遷移図表現に直した場合，statechart のＡＮＤ分割表現によってわかりやすい表現が可能な点である．

謝辞

稚拙な質問にも丁寧にお答え下さった慶応大学の飯島氏に感謝いたします．

参考文献

[1] J.Rumbaugh, M.Blaha, et.al., *Object-Oriented Modeling and Design*, Prectice Hall, 1991.

[2] D.Harel, Statecharts: a visual formalism for complex systems, *Science of Computer Programming 8*, pp.231-274, 1987.

[3] D.Coleman, F.Hayes, S.Bear, Introducing Objectcharts or How to Use Statecharts in Object-Oriented Design, *IEEE Trans Software Eng.*, vol.18, no.1, pp.9-18, 1992.

[4] M.Lubars, G.Meredith, C.Potts, C.Richter, Object Oriented Analysis for Evolving Systems, *Proc. of the 14th ICSE '92*, pp173-185, 1992.