分散トランザクション処理に適したダイアログ制御の記述法

7H−3

山下 高生, 小野 諭
NTT ソフトウェア研究所

1 はじめに

OLTP(Online Transaction Processing) アプリケーションをフォーム部分とタスク部分という2つの部分に分けて考えた場合、前者は直接操作感の実現、後者はデータベースの最適化という全く異なった性格をもっている。よって両者を分割し依存関係を減少させることで、それぞれの目的に専念することができ、さらに大規模ソフトウェア開発における開発作業の並行性を上げることが可能となる。

このように2つに分割した場合、ダイアログ制御にはタスク側から制御する内部制御型、フォーム側から制御を行なう外部制御型、トランザクショナルな部分をタスク側から制御し非トランザクショナルな部分をフォーム側から制御する協調制御型がある。ここではフォーム部分とタスク部分の両方で、それぞれの動作を容易に記述できる協調制御型を用い、両部分が協調して動作するために必要なダイアログ制御の記述法について述べる。

2 ダイアログ制御の記述法

協調動作のための記述法を決めるとき

1. GUI 技術の要求からくる会話の並列性向上
2. ネステッド・トランザクション [1] 技術の要求からくるオペレーションの並列性向上
3. 分散 DB の最適化という要求からくる静的なトランザクション境界の実現 [2]

という前提条件も満足しなければならない。

以上のようなことを踏まえた上で記述しなければならないことは、オペレーションおよびデータの対応付けである。オペレーションの対応づけとは、タスク部分で可能なこととフォーム部分でオペレータとって可能なことを一致させるということである。例えば、タスク側で並列に処理できるオペレーションは、フォーム側でもオペレータが並列に操作できるようするといったことである。データの対応付けとは、データの内容はフォーム側とタスク側で重複して保持しているが、この内容は一般的には異なっている。よってタスクとフォームの両部分でデータの状態や属性について認識していなければならないということである。例えばタスク部分で変更不可能なデータについては、やはりフォーム部分でもそのようにしなければならないといったことが挙げられる。

記述事項を以下に列挙する。

- オペレーションが実行できる条件（データの状態や属性など）
- オペレーションが実行されたときの状態
- オペレーションの制御構造（オペレーションの並行性、逐次性、繰り返しなど）
- 静的に決まるデータの状態

次にこのような事項の記述方法について説明する。まず、各オペレーションをつぎのように表す。

〈オペレーション〉＝
[前条件]《オペレーション本体》[後条件]

ここで [前条件] はオペレーション本体が実行可能な条件（データの状態など）を表すものであり、[後条件] は実行の結果どのような状態になるかを表すものである。この [前条件] および [後条件] を用いてフォーム部分とタスク部分で動的に決まるデータの状態や属性などについての約束を決める。制御構造を表すオペレーションの関係としては

1. 独立直積関係
2. 従属直積関係
3. 直和関係
4. 繰り返し

を定義する [3][4]。1は各オペレーションが並行に実行可能な関係であり、$O_0||O_1||O_2||O_3$ と表す。2は各オペレーションを逐次に実行しなければならない関係であり、$O_0 \cdot O_1 \cdot O_2 \cdot O_3$ と表す。3は1つでもオペレーションが成功したとき、その他のオペレーションは実行しないという関係であり、$O_0 + O_1 + O_2 + O_3$ と表す。4は0または1回以上の繰り返しであり O^* で表す。

以上のようにして、オペレーションとデータをフォーム側とタスク側で対応付けるわけであるが、オペレーションには様々なレベルの汎用性や機能性をもったものが存在する。そこで、この階層をオブジェクト指向の手法を用いて実現するものとする。フォーム側とタスク側で共通のデータを持つ別々のオブジェクトを考え、それぞれインタラクションオブジェクトとセマンティックオブジェクトと名付ける。そしてルート側には機能的には弱いが汎用性があるものを配置し、リーフ側には強い機能を持つ特定業務専用のものを配置する。このようにすることで、コードを集約することができ保守性と再利用性を向上させることが可能となり、さらにフォーム側から見たときの柔軟性を向上させることができるため、直接操作感の向上に有効であると考えられる。

Dialog Description for Distributed Transaction Systems
Takao YAMASHITA and Satoshi ONO
NTT Software Laboratories

3 例

ここでは具体例としてレンタカーの予約業務を考える。図1はこれを表すE/R図である[5]。この業務処理を〈基本予約〉、〈顧客選択〉、〈レンタカー選択〉という3つのオペレーションに分割する。それぞれのオペレーションに関連するデータを共通に持つセマンティックオブジェクトとインタラクションオブジェクトの組が存在することになる。これらの各オペレーションには主要なエンティティーがあり、例えば〈基本予約〉ついては RESERVATION である。この主要なエンティティーはオペレーションの進行状況に応じた様々な状態をとり、これらの状態が各オペレーションの前条件あるいは後条件となる。まず、これらのオペレーション間の関係を上述の方法で記述すると次のようになる。

〈予約〉＝(〈基本予約〉{・〈レンタカー選択〉}) || 〈顧客選択〉

〈基本予約〉 ＝ [ ]〈基本予約〉[基本予約済]
〈レンタカー選択〉 ＝ [基本予約済]〈レンタカー選択〉[レンタカー選択済]
〈顧客選択〉 ＝ [顧客登録済]〈顧客選択〉[顧客選択済]

ここで{}は省略可能であることを表す。

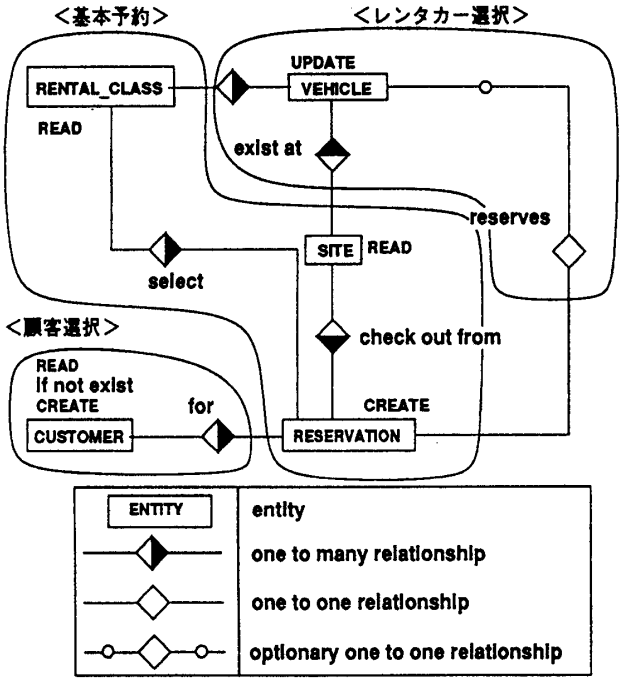

図 1: 予約タスクのE/R図

次にアプリケーション全体の動作について説明する。まずオペレータによって予約業務が選択されるとタスク側は基本予約および顧客選択の2つのオペレーションを開始する。これらは並列実行できるように、スレッドのような技術を用いて実装される。そして、オペレーション開始をフォーム側に通知すると、フォーム側はオペレータとの会話を行なう。このとき会話側でも2つのオペレーションを並列に実行できることを、ウィンドウを別にするなどの手法を用いて視覚化し、直接操作感を向上させる（図2）。

最後にオブジェクト指向による階層化の例について説明する。例としては、レンタカーにどのような車があるのかということの検索オペレーションが考えられる。このオペレーションは〈基本予約〉だけではなく〈レンタカー選択〉でも必要となる。このように汎用性の高いオペレーションは、〈基本予約〉に関連するセマンティックオブジェクトとインタラクションオブジェクトのような下位のオブジェクトではなく、〈基本予約〉と〈レンタカー選択〉の両方の上位のオブジェクトとして管理することによって（図2参照）上述のようにコード集約による保守性や再利用性を向上させることができる。また、フォーム部分の設計という面でも、より柔軟性を持たせることができる。

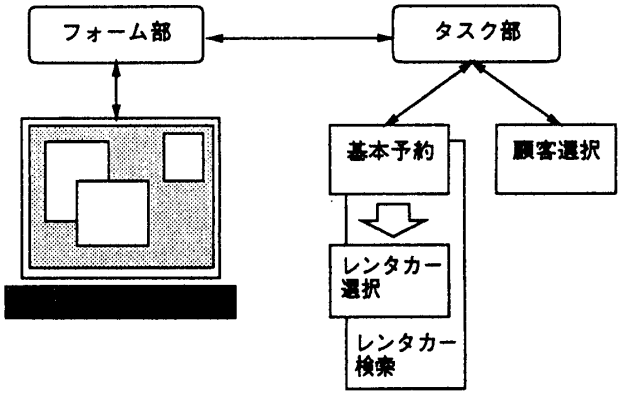

図 2: 予約タスクの動作

4 むすび

以上、オペレーションの並列性の向上、静的なトランザクション境界の実現、直接操作感の向上、開発期間の短縮、保守性の向上などを目的とし、協調制御型ダイアログ制御を用いた場合に必要なダイアログ制御の記述方法、および汎用性のレベルに応じたオペレーションの階層化をオブジェクト指向的手法によって実現する方法について述べた。

今後、OSF/DCE上で動作するの分散TPモニターであるEncinaなどオープンな環境を用いた実現法について検討していく予定である。

参考文献

[1] W.E.Weihl,"Theory of Nested Transactions" in Distributed Systems,Addison Wesley(1989).

[2] 構造化トランザクション定義言語(STDL),NTT MIA テクニカクリクワイヤメント VERSION 1.0, 2-5,TR550001(1991).

[3] 何 克清他,"ソフトウェアプロセスの基本制御構造", 情報処理学会論文誌, Vol.33, No.11, pp.1414-1422.

[4] L.G.Williams,"Software Process Modeling: A Behavioral Approach", ICSE '88, pp.174-186.

[5] T.J.Teorey,"DATABASE MODELING AND DESIGN", Morgan Kaufman, 1990.