

オブジェクト指向による乗務員ダイヤシステムの開発

6J-9

梶井金徳・矢野純一

日立製作所九州システム部第3グループ

1. はじめに

西日本鉄道では電車総合情報システム構築の一環として、乗務員の勤務ダイヤの自動作成に着手した。列車ダイヤから乗務員ダイヤをスケジューリングするには、労働協約や乗換駅の特性などを考慮する必要がある。これまで、専門家がまったくの手作業で数か月を費やしていた割付業務は、定型的な処理手順がなく、問題の解法も不明瞭であった。そこで、試作品に改良を加えながら徐々に完成度を高め、実用レベルに到達する手法に、オブジェクト指向技術を応用した。本稿では乗務員ダイヤシステムの開発経験から、オブジェクト指向技術の生産性と有用性について報告する。

2. システムの概要

乗務員ダイヤの作成とは、列車ダイヤを乗換可能駅で区切った車両小駒情報に分解し、それを乗務員の勤務系統として割付けることである。勤務系統は通常勤務のほかに午前、午後、中休などに分類される。システムの目的は、より少ない勤務系統本数で乗務員の運用を実現することである。そのためには、乗り換えのための待ち時間を少なくして実乗務時間比率の高い勤務の作成に主眼が置かれる。同時に、残業をおさえて基準拘束時間を守り、休憩や付帯労働について取り決めた労働協約を満足する必要がある。ちなみに、勤務系統本数が1本減ると3人分の人件費を節減できるといわれる。

3. オブジェクト指向技術選択の理由

乗務員ダイヤ作成システムは組合せ条件の複雑なスケジューリング問題であり、定型的な処理手順がなく問題の解法も不明瞭であった。専門家から知識の獲得を試みたが、部分的で断片的なものにすぎなかった。システムの実用化を達成するには、試作品を改修して段階的に実現して行くプロトタイプングアプローチが最も現実的な方法であると判断した。オブジェクト指向技術を選択した理由はつぎのような特長に依拠している。

(1) データを内部へ隠蔽して手続きと一体化（カプセル化）させるため、モジュールの独立性が高く、局所的な変更で対応できる。

(2) クラスの階層化により、上位（抽象）クラスの定義が継承（インヘリタンス）できるので、下位クラスでは固有のデータや手続きのみを追加（差分プログラミング）するだけで済む。

(3) 多相性（ポリモーフィズム）により処理の共通化・個別化の管理が容易になる。

(4) コンテナクラスを持つ流通クラスライブラリを利用することで、ソートや抽出など汎用的なアルゴリズムはコーディングしないで済む。

オブジェクト指向技術はソフトウェア開発の上流から下流までを、オブジェクト中心で記述したモデルの詳細化という、一貫した手法によって成立している。したがって、ダイナミックに変化する現象や複雑に入り組んだ関係を忠実に表現するモデルを構築することができ、システムの保守段階においても、実世界の変

Crew Diagram System Development Based on The Object-Oriented

Kanenori Kajii, Jyunichi Yano

Hitachi, Kyushu System 3G

化を容易にかつ他への影響が少ないように反映させることが可能であった。

4. クラス分析

クラス分析はOMT技法に基づいて実施した。システムに要求される振る舞い（機能）をベースにオブジェクトを抽出し、オブジェクトの果たすべき役割と責任の両面から検討し、相互の関係を明確にした。さらに、クラスに共通した振る舞いを見つけだし、抽象クラスの下に階層化させ、差分プログラミングの効果を発揮できるようにした。所有する関係では、所有する側にリストを持ち、デストラクタ（消滅子）が起動された場合は所有する関係のインスタンスも同時に消滅させるような仕掛けを作成した。また、動的モデルとして状態図やイベントトレースを作成し、割付けアルゴリズムを表現した。乗務員ダイヤ特有の時間や時刻については、独自の汎用クラスを作成して比較・演算を簡素化し、米National Institute of Healthの提供するパブリックドメインのC++クラスライブラリを利用して開発の効率化を図った。

5. 開発上の留意点

（1）条件や戦略などのパラメタは実行時に自由に設定できるように、可変なデータとしてプログラムロジックから独立させておく。データは、上位クラスのクラス属性にしておけば、下位クラスからグローバル（広域的）に参照することができる。

（2）改造は頻繁に発生し、時には根本的な構造変更が必要な場合もある。つねに、プログラム全体を見渡して、二重コーディングを廃し、少ないステップで実装して、簡潔で明瞭なプログラムを維持する。そのためには、「1クラスは500ステップを限度」、「1メソッドは50ステップ以内」といった指針を設けることも有効である。

（3）仕様を理解することは、コミュニケーションの問題であり、効率を上げるには少人数で開発することである。そのために、システムを可能な限り小さなブロックに分割して、責任を極力軽減する努力をする。今回の開発従事者は、平均して3名であった。

6. 生産性と信頼性の評価

本システムの開発期間は1年4カ月、開発ステップは25K（スケジューリング部11K、画面インタフェース部14K）、工数は30人月であった。これらの数値に対しては、従来の手法による開発と比較する資料がないため、定量的な検討はしていないが、つぎの事項において高い評価ができると判断する。

少ステップな開発は、継承機能によりデータや処理の二重化を廃し、差分プログラミングを徹底した。たびかさなるエンハンスを行なってもスケジューリングクラス部は11Kステップのままであった。経験的な判断だが、従来の手法で開発すれば、倍以上のステップになったと予想する。

クラスライブラリの活用は、汎用的な処理のコーディングを不要にし、少ステップのほかに、高い品質をもたらすことができた。これまでの経験では、プログラマ各自が勝手にロジックを組立てて、不良のブラックボックスを生み出していた。今回利用したN I H C Lはマッチングや抽出に向けた豊富な型を内包して有用であった。

改造は、100項目20回に及んだが、迅速に対応できた。派生クラスの応用や、振る舞いの追加は、局所的な修正で済んだ。また、不具合対策もオブジェクト構造を変更して対応できる柔軟性に助けられた。しかも整合性チェックの厳しいコンパイラがあるので、大幅な変更も安心して実施できた。

なにしろ今回は初めての取り組みだったので、新規に作成した部分も多く、教育や使用上の誤り訂正のために費やした時間も含まれている。今後は、独立性の高い部品の再利用や、分析・開発の経験から体得した方法論を有効に活用して更に高い生産性を実現できるものと確信している。