ソフトウェア工学 103－5
（1995．3．9）

生産計画システム構築のためのオブジェクト指向フレームワーク

堀 雅洋　　中村 祐一

日本アイ・ビー・エム(株) 東京基礎研究所

オブジェクト指向に基づくソフトウエア開発において，個々のクラスを再利用の単位とするのではなく，類似のアプリケーションに共通するクラス群の振舞に着目するフレームワークの概念が注目されつつある．本稿では，まずスケジューリング問題における問題解決手順を部品化するために筆者等が行なってきたドメイン分析の概要を説明する．次に，その経験に基づいて設計された生産計画システム構築のためのフレームワークの構成を示す．さらに，実際のアプリケーション開発に適用した事例を紹介するとともに，フレームワークと設計パタンの関連についても考察する．

Object-Oriented Framework for Production Scheduling Applications

Masahiro Hori　Yuichi Nakamura

Tokyo Research Laboratory, IBM Japan Ltd.
Email: {hori,nakamury}@trl.ibm.co.jp

This paper reports our experience in developing an application framework, and applying it to the development of an actual application. First, we briefly explain our experience in domain analysis for scheduling problems. We then introduce our framework for production scheduling applications, which is implemented in C++ and consists of three loosely coupled subsystems: a schedule model, a scheduling engine, and a graphical user interface. We also investigate the framework in terms of a concept of design patterns.

1 はじめに

オブジェクト指向の概念に基づくソフトウエア開発は，分析から実装へ至る反復的な過程を，実世界における概念を反映した共通のモデルを前提として行なえるものとして発展が期待されている．さらに，ソフトウエア再利用にあたって個々のクラスを単位とするのではなく，類似のアプリケーションに共通するクラス群の振舞に着目するフレームワークの概念 [Johnson 88] も注目されつつある．しかしながら，ソフトウエア開発に際して，実世界のどのような側面に着目し，どのような観点からモデル化を進めるべきかについては必ずしも自明ではなく，対象領域(ドメイン)の分析・モデリング技術の重要性も指摘されている [Prieto-Diaz 90]．特に，ソフトウエアの再利用では類似の問題に共通して見い出される概念をボトムアップに再構成していく過程が不可欠である．

筆者等はこれまで生産計画や人員配置といったスケジューリング問題を対象として，部品合成によるプロトタイプ開発環境の構築 [Hori 94]，ならびに類似のスケジューリング・システムの分析と問題解決手順の部品化に取り組んできた [Hori 95]．

本稿では，ドメイン分析の経験に基づいて構成された生産計画システム構築のためのフレームワークについて報告する．以下，2章では処理手順を部品化するために行なった分析の概要を説明する．3章では，生産計画システムを対象としたフレームワークの構成を示し，4章でそれを実際のアプリケーション開発に適用した事例を紹介する．最後に，フレームワークと設計パタン [Johnson 92] の関連についても考察する．

2 ドメイン分析

スケジューリング問題を対象として，これまで様々な知識システムが開発されてきた．例えば，空港に離発着する旅客機を搭乗ゲートに割り当てるシステムでは，ゲートが占有される時間を効率的に配分するために，同じ飛行機に対して連続する到着と出発をひとまとめにするといったことが行なわれている [Brazile 88]．また，製鋼工程のスケジューリング・システムでは，生産性を高めるために鋳造作業を連続的に行なう複数の部分スケジュールを作成し，最後にそれらを併合するといった処理が行なわれている [森下 90]．

このように個別に見い出されてきた問題解決手順を部品化するために，図1に示した3段階からなる分析とモデル化を行なった [Hori 95]．以下では，類似の問題とその解法を記述するための基本語彙，部品の雛型である問題解決パタン，および部品抽出の結果について述べる．

2.1 基本語彙

(a) 問題記述のための語彙

様々なスケジューリング問題の中で特に対象範囲を明確にするために，1つの問題クラスとしてジョブ割付問題を設定した．これは「種々の制約条件を満たした上で，与えられたすべてのジョブを資源と時区間から構成される割付平面上に配置する」もので，割り付けるべきジョブの集合が予め固定的に与えられることを前提としている．

ジョブ，資源，時区間は，基本的な概念クラスとして，それらに付随する属性とともに与えられる．ジョブは割り付ける対象を表す概念で，その属性には，名前，タイプ，時間長，割付可能な時間帯，割付可能な資源等がある．資源は割り付けられる対象を表すもので，名前やタイプ等が典型的な属性となる．時区間はジョブによって占有され得る時間的な区分を表し，属性には始点，終点，最小の時間単位がある．

制約条件は，各概念クラスに規定された属性間の関係として表される．特に，その関係によって参照される属性の値を最大化/最小化/平均化するもの，あるいはある値との差をできるだけ小さくするものも含まれる．

(b) 解法記述のための語彙

上述の問題構造は，与えられた問題の静的な側面をとらえたもので，問題解決手順を記述する手段については別に規定しなければならない．ジョブ割付問題における問題解決過程は，与えられたジョブ集合を割付平面上に配置する一連の操作としてとらえることができる．そこで，問題解決手順の記述には，要素集合または構造化された要素群に対する12種類の基本操作を用いた．例えば，要素集合からただ1つの要素を選択する操作は Select，部分集合を得る操作は Filter と呼ばれる．一方，構造化された要素群からある要素を選択する操作は Focus，部分構造へと分割する操作は Decompose と呼ばれる．

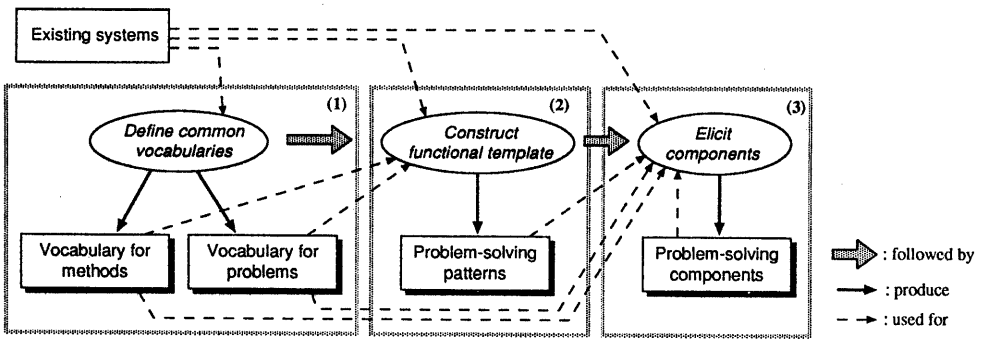


図 1. 部品抽出の手順

2.2 問題解決パタン

問題とその解法に関する基本語彙は，基本機能のボトムアップな抽出を可能とするが，それらの機能が部品として相互に接続可能となるには別途トップダウンな指針が必要となる．そのために，割付平面上への一連の配置操作を類型化し部品の雛型として問題解決パタンを用意した．このような指針は，特に部品ライブラリを充実させていく際にも有用である．

問題解決パタンは，既存のスケジューリング・システムに関する文献中の記述を考察することによって得られたものである．図 1における段階 (2) ではさらに以下の作業が行なわれる．

1) 各システムが対象とするスケジューリング問題をジョブ割付問題として定式化する．
2) 相互に接続可能な入出力インタフェイスを備えた機能的な単位となるように，問題解決過程の断片を部品候補として切り出す．
3) 部品候補として切り出された機能を類型化し一般化する．

その結果，図 2に示した 3 種類の問題解決パタンが得られた．Divide & Merge は，与えられた問題を部分問題へと分割し (例えば，ジョブ集合の分割)，部分問題を解くために別の部品を呼び出し，さらに必要に応じて部分解を合成する．Transform & Restore は，問題を解き易いものに変換し (例えば，制約緩和)，変換された問題を解くために別の部品を呼び出し，必要に応じて変換された問題を元の状態に復元する．Check & Modify は，与えられた解に対して何らかの条件に該当する部分を検出し (例えば，制約違反)，その部分に変更を加える．

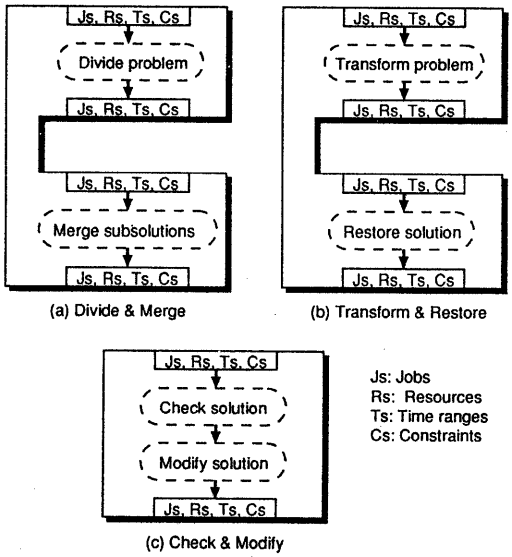


図 2. 問題解決パタン

部品相互の接続形態には，逐次的なものと埋め込みによるものの 2 種類がある．図 2より明らかなように，埋め込みによる接続は Divide & Merge あるいは Transform & Restore の途中で他の処理を呼び出す場合に行なわれるものである．

2.3 部品抽出の結果

筆者等は，さらに 2 つのスケジューリング・システムを対象として，特にそのシステムを実際に開発したエンジニアへのインタビューを通して部品の抽出を行なった．その結果，文献中の記述に基づいた分析結果と合わせて，最終的に 6 つのシス

テムから11個の部品を得た．分析対象となった各システムは，それぞれ平均3.3個の部品によって再構成された．また，2つ以上のシステムに共通して見い出された部品は5個あり，最も多いものは5つのシステムに共通するものであった [Hori 95]．このように，部品抽出の結果は類似のスケジューリング問題における潜在的な問題解決手順の多様性にある程度の見通しを与えるものとなっている．

2.4 考察

部品化された問題解決手順は12種類の基本操作を用いて記述されている．そのような部品を前提としたソフトウエア開発では部品の検索・接続だけでなく，各基本操作を実行可能な手続きに置き換える必要がある [Hori 94]．ところが，それらの操作は要素集合または構造化された要素群といった一般的なデータ構造を前提としており，アプリケーションに特徴的な操作対象の構造が明確ではなかった．

操作対象のモデルを明示的にする1つの方法としては，対象分野をさらに具体化することが考えられる．ジョブ割付問題は，人員配置，生産計画，旅客機の搭乗ゲートへの割付，タンカーの桟橋への割付など様々なスケジューリング問題を包含するが，ここでは特に生産計画問題を対象とする．生産計画問題は，ある定められた期間内に生産すべき製品の種類と量を決定するものである．したがって，生産工程や生産設備等が対象領域に固有の概念としてモデル化の対象となる．

生産計画の決定には，資材計画，在庫管理，工程管理といった関連部門との調整も必要であり，また計画の実施にあたっては機器のトラブルなど不慮の事態が発生する可能性もある．そのため，実用レベルで運用されてきた多くのスケジューリング・システムでは，システムが提示した結果をユーザ自身の手によって対話的に修正する機能を備えている (例えば [森下 90])．したがって，柔軟なユーザ・インタフェイスの実現は不可欠である．

さらに，生産計画業務は各現場に固有の条件が少なくなく，ソフトウエアの再利用を通して様々な要求に柔軟に対応していくことはシステム開発側にとって重要な課題であると言える．

3 フレームワーク

フレームワークは類似のアプリケーション共通するクラス群に対して，クラス間での典型的なメッセージ交信の形態を明示的にするもので，特に設計情報の再利用が重視される [Johnson 88]．フレームワークの例としては，領域に特化したグラフィクスの描画 [Vlissides 90]，マルチメディア・データの編集 [Gibbs 91]，金融取引のシミュレーション [Eggenschwiler 92] といったアプリケーションを対象とするものがある．それに対して，具体的なアプリケーションを想定せずに一般的に有効な諸機能を提供することによってコードの再利用を促すものはツールキットと呼ばれ，フレームワークと区別される [Johnson 88]．

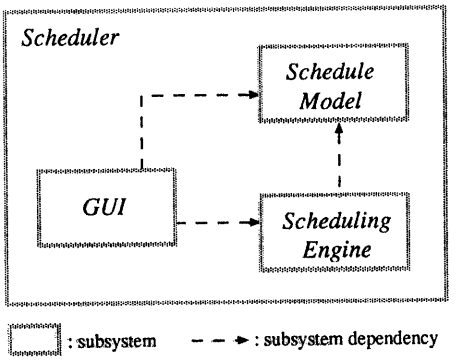


図 3. サブシステム図

本章では，C++で実装された生産計画システム構築のためのフレームワークを OMT の記法 [Rumbaugh 91] に従って示す．なおフレームワークの設計にあたっては，前章での考察の結果を指針とするとともに，当研究所で開発されたスケジューリング・システムである Scheiker [1] の設計も参考にしている．

このフレームワークは，生産設備や生産工程に関するスケジュール・モデル，生産計画を立案するスケジューリング・エンジン，およびユーザインタフェイスに関するモジュールの3つのサブシステムから構成される．フレームワーク全体のサブシステム図 [Rumbaugh 94b] を図 3に示す．

3.1 スケジュール・モデル

スケジュール・モデルは生産現場をモデル化するもので，Scheiker で用いられてきた定式化を反映している．スケジュール・モデルは，さらに資源 (Resource)，生産工程 (Process)，および製品 (Prod-

[1] C 言語で実装されており，既に数多くの導入実績がある．

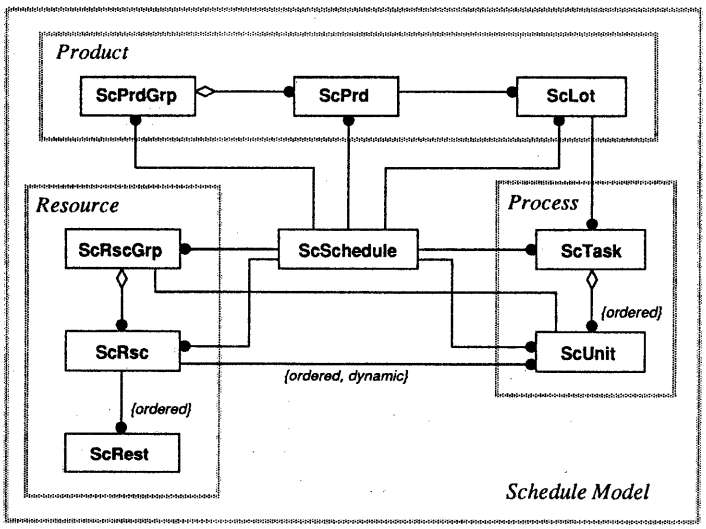


図 4. スケジュール・モデルに関わるクラス図

uct) に関する 3 つのサブシステムから構成される (図 4). ScSchedule クラスは，スケジュール・モデル全体を代表し，複雑に関連し合ったクラスへの参照を容易にするものである．他の 2 つのサブシステムはこのモデルを前提とした上でアプリケーション全体としての機能を実現する．

資源サブシステムは，工作機械などの生産設備に関するもので，同一タイプの資源集合である資源グループ (ScRscGrp)，個々の資源 (ScRsc)，および資源の休止時間 (ScRest) に相当するクラスからなる．一方，生産工程は単一の資源によって処理される最少の作業単位であるユニット (ScUnit)，およびある製品を完成させるために順序付けられた一連の作業に相当するタスク (ScTask) からなる．また，製品については，製品グループ (ScPrdGrp)，各製品 (ScPrd)，および納期や数量の定められた生産単位であるロット (ScLot) からなる．1 つのロットは，一般に何通りかの生産工程によって作業を完了することができることから，ScLot と ScTask は一対多の関連で結ばれている．

2 章で述べたジョブ割付問題におけるジョブは，ここでのユニットに相当する．したがって，このスケジュール・モデルでは，単にユニットを資源に割り付けるといった対応関係以外にも，生産計画問題に特徴的なドメインの性質がモデル化されている．

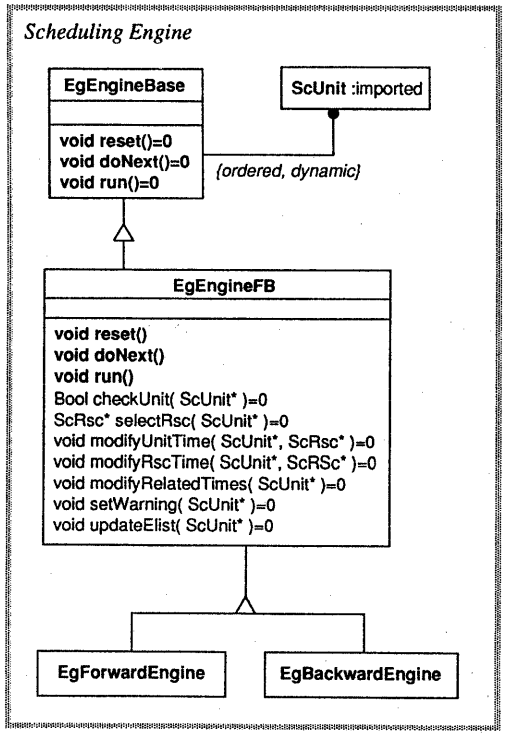


図 5. スケジューリング・エンジンに関わるクラス図

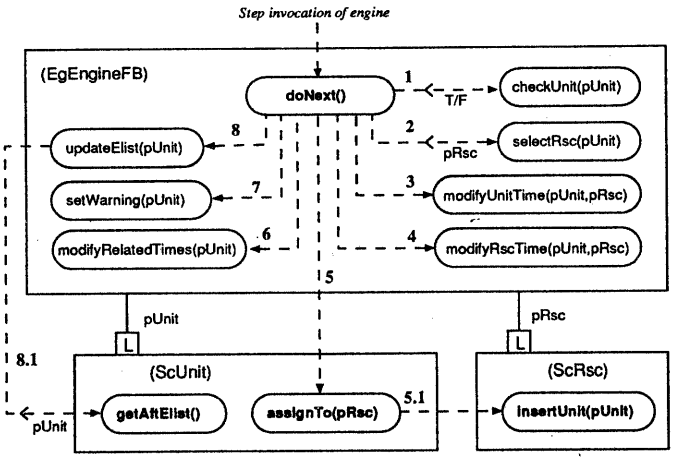


図 6. エンジンの基本動作に関するメソッド交信図

3.2 スケジューリング・エンジン

スケジューリング・エンジンに関連するクラス階層を図 5に示す．このサブシステムは，様々な計画手順を適宜使い分けることを可能にするものとなっている．

全てのスケジューリング機能は，順序付けられた割付候補ユニットのリストを前提とするが，その関連は最上位の抽象クラスである EgEngineBase において規定される．さらに，EgEngineBase では，他のサブシステムからスケジューリング・エンジンを起動するためのインタフェイスが定義される．すなわち，エンジンを初期化 (reset) した後，割付可能な候補ユニットがある限り処理を継続する (run) 場合と，1つのユニットの割付 (doNext) を繰り返す場合の2通り起動が可能となっている．

通常の生産計画において典型的なフォワード・スケジューリングおよびバックワード・スケジューリングに共通する手順の雛型は EgEngineFB に付随する操作として与えられている．ここで，バックワード・スケジューリングとは各ロットの最終ユニット (最終工程) の終了予定時刻である納期を起点として，各ユニットの終了・開始時刻を後ろ詰めにしながら逆算しながら決定することを基本とする方法である．それに対して，フォワード・スケジューリングは各ロットの第1ユニットの最早開始時刻を起点として各ユニットを順次前詰めにする方法である．

EgEngineFB に付随する操作によって doNext がどのように詳細化されるかを，メソッド交信図 [Rumbaugh 94a] として記述したものを図 6に示す．これらの操作の具体的な実装は，フォワードならびにバックワード方式に固有の機能を提供する EgForwardEngine, EgBackwardEngine において与えられる．

図 6に示された処理手順は，部品抽出にあたって分析対象となった6つのシステムのうち5つに共通に見い出されたものに相当する．このフレームワークでは，2章で述べた部品に相当する構成要素は前提としていない．したがって，計画手順の修正・変更する場合は，doNext から呼び出される一連の操作の実装を EgEngineFB の下位クラスにおいて再定義するか，EgEngineBase の下位に別途該当するクラスを定義する．後者の場合には，そこで前提となる手順を EgEngineBase で規定された起動手順に適合させることによって，スケジュール機能の詳細な差異にかかわらず，サブシステムの外部から同様の手順にしたがってエンジンを起動することが可能となる．

3.3 ユーザ・インタフェイス

ユーザインタフェイスについては，前提とするウィンドウ・システムに依存する部分が少なくないため，他の2つのサブシステムから区分されている．グラフィカルなユーザインタフェイスの開発には既存の GUI クラス・ライブラリを用いている

が，それを他のクラス・ライブラリに置き換える場合でも，関連する変更は該当するサブシステム内で閉じたものとなり他のサブシステムを修正する必要はない．

4 適用事例

ここでは，生産進捗状況の表示システムの開発に先のフレームワークを適用した事例を紹介する．このシステムは，実績データを外部から取り込むことによって進捗状況を表示するものである．したがって，スケジューリング・エンジンの機能は必要としないが，定期的に取り込まれる計画値および実績値をスケジュール・モデルの内部状態に適宜反映させなければならない．さらに，この事例に固有の要求として，2つの工場での進捗状況を扱えるようにし，生産実績データと計画値との時間的なずれがわかるように両者を並行して表示するといったものがある．

アプリケーションの開発にはフレームワークの設計者1名が関わったが，コーディングとデバッグに要した時間は約30時間であった．これは，フレームワークの実装に要した時間のおよそ4%に相当する．

もとのフレームワークとこのアプリケーションについて，各サブシステムごとのクラス数を表1に示す．特に，このアプリケーションに対してどのクラスを利用すべきかは，開発の初期段階で容易に決定することができた．また，アプリケーションの開発にあたって新たに追加されたクラスはなかった．このことから，このフレームワークはスケジューリング・エンジンを用いない場合にも容易に対応できると言える．

筆者等は，スケジューリング機能も要求される生産計画システムの開発を現在進めており，エンジン部分も含めたフレームワークの有用性についてはその経験を通して検証していく予定である．

5 設計パタン

前章の適用事例では，フレームワークの利用者がその設計者と同一であった．ところが，ソフトウエア再利用では再利用対象物の提供者と利用者が異なる場合が一般的である．そのような状況では，再利用対象物の背後にある意図や設計理由を明示的にすることが重要となる．フレームワークについては，その背後にある前提を文書化しフレームワーク利用を容易にすることを目指して設計パタンの整理と収集が進められている [Gamma 94]．本稿で述べた生産計画システムのためのフレームワークに，文献 [Gamma 94] で示された設計パタンのいくつかをあてはめたものを図7に示す．

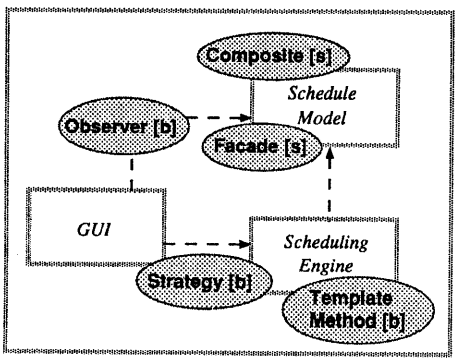
表 1. 各サブシステム内のクラス数

	Schedule Model	*Scheduling Engine*	*GUI*
Original framework (29)	12	4	13
An application (10)	7	0	3

例えば，あるオブジェクトに対してその内部状態の変化を反映するいくつかのオブジェクトとの依存関係を規定するパタンは Observer と呼ばれる．このパタンは MVC フレームワーク [Krasner 88] におけるモデルとビューの関係に相当する．生産計画のフレームワークでは，スケジュール・モデルとその内容を様々な観点から表示するいくつかのウィンドウとの関係が Observer パタンに相当する．

また，様々な処理手順の差異をその利用者から隠し，手順の使い分けを容易にするためのパタンは Strategy と呼ばれる．例えば ET++SwapsManager[Eggenschwiler 92] では，金融取引における様々な計算手順を適宜使い分けるためにこのパタンが用いられている．生産計画のフレームワークでは，抽象クラス (EgEngineBase, 図5参照) に規定されたインタフェイスに基づいて，その下位クラスとして様々な計画手順を具体化する部分がこのパタンに対応する．

このように，設計パタンは様々なフレームワークに共通して見い出される断片的な性質を，より



[s] : structural pattern　　[b] : behavioral pattern

図 7. フレームワークと設計パタンの関連

一般的な原則として再整理したものとなっている．筆者等が部品抽出の雛型として用いた問題解決パタンは，断片的な問題解決手順を特にジョブ割付問題に特徴的な語彙(ジョブ，資源等)を前提として相対的に具体化したものとなっている．したがって，設計パタンよりも一般性には欠けるが，生産計画フレームワークにおいて将来的にスケジューリング・エンジンの機能を洗練していく際に指針を与えるものとなっている．

6 おわりに

スケジューリング問題を対象として行なったドメイン分析，ならびにその経験に基づいて設計された生産計画システム構築のためのフレームワークについて述べた．

本稿で述べた分析手法は，スケジューリング問題以外にも問題解決知識が経験的手順に集約される問題領域に対して適用可能である．反面，処理の流れが明確な定型的業務，あるいはモデル化の対象物に対してその振舞を説明する原理が存在する場合については，この分析手順は適さないと考えられる．後者に該当する問題には原子力プラントや電子回路の故障診断等がある．このような問題領域では対象物の動作原理に基づいてその振舞を再現するモデルが重要であり，そのための分析をより精緻に行なわなければならない．

フレームワーク設計の初期段階では，対象領域のボトムアップな分析とモデル化が不可欠であり，設計パタンに代表される汎用性の高い設計情報に基づいたトップダウンな構築は必ずしも容易ではない．しかしながら，対象領域への理解が深まるにつれて，過去の利用経験を集約した設計情報から有益な示唆を得ることが期待できる．

さらに，開発されたアプリケーションが効果的に動作するためには，関連するアプリケーションやネットワーク環境との接続性も考慮した上でフレームワークを設計する必要がある．例えば，生産計画のアプリケーションについては，広い範囲での生産管理を前提としたフレームワークも提案されている [SEMATECH 94]．そのような広域フレームワークとの整合性やスケーラビリティについて検討することは今後に残された重要な課題である．

参考文献

[Brazile 88] R. Brazile & S. Swigger : GATES: an airline gate assignment and tracking expert system. *IEEE Expert*, Vol. 3, No. 2, pp. 33–39 (1988).

[Eggenschwiler 92] T. Eggenschwiler & E. Gamma : ET++SwapsManager: using object technology in the financial engineering domain. In *OOPSLA '92 Conference Proc.*, pp. 166–177, ACM Press (1992).

[Gamma 94] E. Gamma, R. Helm, R. Johnson & J. Vlissides : *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, Reading, MA (1994).

[Gibbs 91] S. Gibbs : Composite multimedia and active objects. In *OOPSLA '91 Conference Proc.*, pp. 97–112, ACM Press (1991).

[Hori 94] M. Hori, Y. Nakamura & T. Hama : Configuring problem-solving methods: a CAKE perspective. *Knowledge Acquisition*, Vol.6, No.4, pp.461–488 (1994).

[Hori 95] M. Hori, Y. Nakamura, H. Satoh, K. Maruyama, T. Hama, S. Honda, T. Takenaka & F. Sekine : Knowledge-level analysis for eliciting composable scheduling knowledge. *Artificial Intelligence in Engineering*, Vol.10 (in press).

[Johnson 88] R. Johnson & B. Foote : Designing reusable classes. *Journal of OOP*, Vol.1, No.2, pp.22–35 (1988).

[Johnson 92] R. Johnson : Documenting frameworks using patterns. In *OOPSLA '92 Conference Proc.*, pp. 63–76, ACM Press (1992).

[Krasner 88] G. Krasner & S. Pope : A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80. *Journal of OOP*, Vol.1, No.3, pp.26–49 (1988).

[森下 90] 森下, 沼尾, 戸沢 : 協調型スケジューリングによる製鋼工程スケジューリング・エキスパートシステム. 人工知能学会誌, Vol.5, No.2, pp. 40–49 (1990).

[Prieto-Diaz 90] R. Prieto-Diaz : Domain analysis: an introduction. *ACM SIGSOFT, Software Engineering Notes*, Vol.15, No.2, pp. 47–54 (1990).

[Rumbaugh 91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy & W. Lorenson : *Object-Oriented Modeling and Design*. Prentice Hall, Englewood Cliffs, NJ (1991).

[Rumbaugh 94a] J. Rumbaugh : Going with the flow: flow graphs in their various manifestations. *Journal of OOP*, Vol.7, No.3, pp. 12–23 (1994).

[Rumbaugh 94b] J. Rumbaugh : Building boxes: subsystems. *Journal of OOP*, Vol.7, No.6, pp.16–21 (1994).

[SEMATECH 94] SEMATECH Inc. : *Computer Integrated Manufacturing (CIM) Application Framework Specification 1.1*, #93061697D-ENG (1994).

[Vlissides 90] J. Vlissides & M. Linton : Unidraw: a framework for building domain-specific graphical editors. *ACM Trans. on Information Systems*, Vol.8, No.3, pp. 237–268 (1990).