〔新 日 鉄 技 報 第358号〕 (1996)

UDC681.3.02.001:658.512:621.771

“K1”による厚板精整管制支援システムの開発

Development of Support System for Supervisory Control of Heavy Plate Leveling by “K1”

馬 場 俊 光*(1) 松 尾 好 晃*(1)
Toshimitsu BABA *Yoshiteru MATSUO*

抄 録

計画型エキスパートシステム構築ツール“K1”と“K1”を用いて開発された新日本製鐵の製鉄所の厚板精整管制支援システムについて紹介する。“K1”では，独自のパターンマッチングアルゴリズムを採用して，大規模・高機能知識ベースでも推論時も十分な実行速度が得られる推論エンジンを実現した。知識の記述には従来のIF-THEN形式を踏襲しながら，オブジェクト指向の概念を積極的に採り入た新しい知識ベースアーキテクチャを提案している。ルールベース中で，オブジェクト指向言語の一つであるC++言語を記述可能にすることで記述性と処理効率の向上をはかるとともに，“K1”自身もC++言語で構築することで移植性にも優れたソフトウェアを目指した。厚板精整管制支援システムは，上工程である圧延ラインから精整ラインに流入する鋼板物流を，その時々の操業状態に応じて効率良く制御することが目標である。本システム開発に“K1”を適用することで，一般の市販ツールを利用した場合に比べて大きなメリットが得られた。

Abstract

A planning-type expert system constructing tool “K1” and a support system developed with “K1” for a supervisory control of heavy plate leveling operation in a plate mill in Nippon Steel are described. “K1” uses a newly developed pattern matching algorithm which realizes a fast inference execution even for a large scale and high functional knowledge base. Besides usual IF-THEN representation of knowledge, “K1” proposes a new architecture for knowledge base to positively adopt the “object-oriented” concept. Both easiness of describing probelms and easiness of being transplanted are realized by constructing “K1” itself with C++ language without losing executing speed. The support system for supervisory control of heavy plate leveling operation is aimed to make the operation in good accordance with the operation of the upper production process, the rolling line of heavy plates. Compared with the case with other tools in the market, development of this support system with “K1” brought forth big advantages.

1. はじめに

生産性の向上・効率化は製造業にとって重要な課題である。この課題に対応するために製造業では，ＣＩＭ(Computer Intergrated Manufacturing)が広く取り入れられており，中でも製鉄業は，早い時期からコンピュータによる自動操業を試みてきた。近年のプロセス技術の進歩にともなって，コンピュータはこれまで以上に重要な役割を帯びてきている。

従来のＣＩＭ環境においてはコンピュータは基本的なデータ処理を担当するだけであったが,現在では知的な判断が要求されるような問題を処理できることが求められている。このような人間の代わりに知的な処理を行なうための技術として人工知能の１分野であるエキスパートシステム(Expert System)がある[1]。低価格・高性能のエンジニアリングワークステーションとシェル(エキスパートシステム構築のためのツール)が登場した1980年代にエキスパートシステムは一気に普及した。製鉄業では早くからエキスパートシステムの高い効果に注目して，積極的に数々の実用エキスパートシステムを開発してきた。そして，この開発過程を通じて，汎用性を第一目標としている商用シェルによるエキスパートシステム開発では社内の実問題において必ずしも十分な実用性が得られない事例が明らかになってきた。

1991年に新日本製鐵内で実用化されているエキスパートシステムに関してアンケートを行なった。図１は“エキスパートシステム開発におけるボトルネック”の結果である。知識表現，ユーザーイン

*(1) エレクトロニクス・情報通信事業部
システム研究開発センター 主任研究員

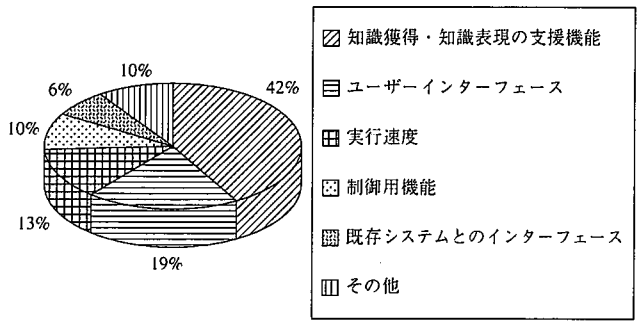
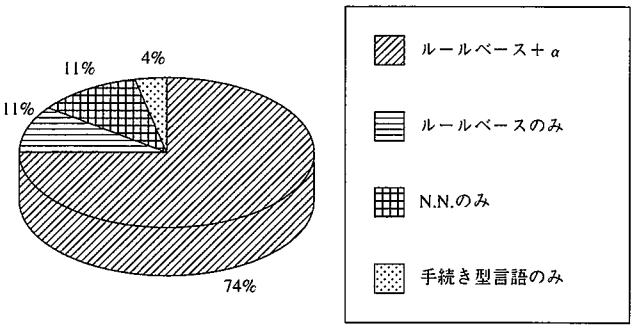


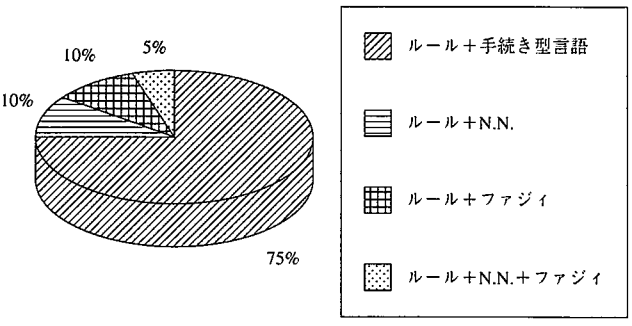
図 1　ES開発におけるボトルネック

ターフェース，処理速度の問題が大部分であった。図2はシステムの実現形態を示している。対象問題の大規模化・複雑化を反映して，ルールーベースシステムのみによる古典的実現は少なく，手続き処理の併用(通常の手続き型言語によるプログラミングとの併用,使用されていた言語はFORTARNあるいはCであった)や，当時の状況を反映してファジィ応用やニューラルネット利用による統合システムが多かった。

一方，エキスパートシステム適用分野という観点から過去の事例を分析すると，早期に開発されたエキスパートシステムは，故障診断に代表される解析型システムに分類されるものが多い。解析型システム構築用のシェルとして当システム研究開発センターでは，"Diag-Solver"が開発されており，いくつかのシステムに応用されている[2,3]。"Diag-Solver"は診断に関する簡便な知識表現と柔軟な推論機能を有し，現場利用者が自ら利用できるシェルであり，実用システムを効率良く開発,保守できるようになっている。



(a) 実現形態



(b) ルールベース型ESで併用された手法

図 2　ESの実現形態

これに対して，計画型システムは問題解決のための汎用的な手法は確立されてない。ＡＩ技術として提案される仮説推論やＯＲ技術として提案される最適化手法，数理計画法等が一般的なアルゴリズムではあるが，いざ問題を解くとなると計算機能力の限界や問題の定式化の困難さ等，課題も多く，本当の意味での実用的なアプリケーション構築は決して容易ではない。その一方最近の傾向として，これら合成型のシステムへのニーズが高まっている。

このような動向を踏まえて，オブジェクト指向型の強力な知識表現と高速推論性能を備えたエキスパートシステム構築ツール"K1"を開発した。

2. "K1"

2.1　開発の目標

2.1.1　ルールベース言語の記述性の改良

プロダクションルール(IF-THEN 形式)は既に現場に浸透している知識表現であり，今後の統合型システムにおいても全体を制御・管理する中核部に必要な高い知識記述力がある。しかし，システム全体をプロダクションルールにより表現することは非常に困難である。実際にはシステムを構築するにあたり，構造化された知識や手続き的な知識を組み合わせる必要性が存在するからである。このような観点から，ルールベース記述言語"K1"では，その文法にオブジェクト指向型言語 C++[4,5] を参考にして,その特徴を導入した。

図3は構造化された知識の記述性の例である。'product'は，

```
defwmeclass aPlate {
   class:
      int    counts;
      void   mkdaemon() {
                  cout << ++c ounts << "created.¥n";
             }
   instance:
      double width;
      double length;
      double thickness;
};

defwmeclass productA : aPlate { // Inheritance...
      double contentC;
      double .....contentSi
};
```

図 3　ワーキングメモリエレメントクラスの例

'aPlate' のサブクラスとして定義され，独自のスロット(変数)である 'contentC' と 'contentSi' を持つと共に親のスロットである 'width', 'height' とメソッド 'mkdaemon' を継承している。オブジェクト指向のメリットは，システムの開発段階にも現れる[6]。

一般的にシステムの開発段階では，頻繁に仕様が変更される。そして，この傾向は ES 構築時には更に顕著である。このような場合，手続き型言語によるプログラミングよりも，オブジェクト指向型言語によるプログラミングの方が柔軟に対応できる。

"K1"の多くの知識表現単位は，オブジェクト指向でいうところのクラスとして捕えることができるため，システムの解析・設計，更には実装の段階で，オブジェクト指向システム開発のメリットを利用できる。

2.1.2　既存システムとの共存

実際のシステムでは，たくさんのソフトウェアが，コンピュータ間で協調しながら駆動している。しかし，商用シェルを利用すると既存システムとエキスパートシステムの統合に制約が発生すること

が多い。このような障害のほとんどは，シェルの閉じたシステムアーキテクチャーとシステム構築に使用されている言語自体の特質に起因している。

"K1"は知識ベースの任意の部分に，汎用開発言語であるC++言語のステートメントを直接記述することができる"C++ダイレクトコーディング"機能を提供している。このC++言語との融合によってオープンシステム型のエキスパートシステムが容易に構築できるようになっている。ダイレクトコーディングの典型的な例を図4に示す。行頭の??(と，行頭の??)で囲まれた領域がC++言語で記述されている部分である。C++ダイレクトコーディング機能は既存システムとの統合の手段としてだけでなく，知識ベースの中に手続き

```
rule move
{
    $P( product status == unprocessed );
    $A( materialA );
    $B( materialB );
    # ( $A. typeCode == $B. typeCode );
    [ $P.deadline ]
=>
    ??( // C++ Direct Coding ...
        int      remains = $P. counts - 1;
        cout << remains<<"remaining.¥n";
    ??)
    modify( $P  status = finished );
    remove($4); remove($B);
}
```

図4　C++ダイレクトコーディングの例

的な知識を記述することにも利用できるというメリットがある。

更に"K1"はC++言語の1クラスとして生成されるので，メッセージ通信により外部システムからのコントロールも容易にできる。逆に，知識ベース中にユーザー定義のクラスタイプとして宣言することで，C++言語で定義されたクラスを知識ベースに直接取り込むこともできる。

条件部にC++言語のクラスを取り込むことはできないという制限はあるが，それ以外は"K1"の他の基本タイプとほとんど同じである。従って，ユーザーは，ある程度複雑な機能を持ったデータ構造が必要となる知識単位(ワーキングメモリエレメント)クラスを定義する場合，C++言語のクラスとして構造を定義し，知識ベースの中にインポートすることができる。

このように，"K1"は既存のシステムや他言語で開発された外部モジュールと結合するための柔軟な統合手段を提供している。

2.1.3　高速推論エンジン

一般に，推論エンジンは知識ベースに蓄えられた知識(プロダクションルール)の条件部とデータ(ワーキングメモリ)を照合(パターンマッチング)してルールを連鎖的に発火させる形で推論を進める機能を提供する。一般に，このパターンマッチングは膨大なメモリを消費する非常に負荷の高い処理であり，実システムにおいては重大な性能要因である。このため，パターンマッチングの高速アルゴリズムが種々研究された。

そして現在の高性能推論エンジンの多くは，RETEアルゴリズムによるパターンマッチング法を基本としている[7]。RETEアルゴリズムは汎用パターンマッチング方式として定着しているが，様々な角度から改良され多くのバリエーションが存在する。(例えば[8])当センターでは，特に大規模な組合せ問題でRETEより高速でメモリ消費量の少ない独自のパターンマッチングアルゴリズムを開発した。このアルゴリズムは次の三つの内部モジュールより構成されている。

DTREE　　; Digital search
TREE, ONET ; Optimized inference NETwork,
CR　　; Conflict Resolver

DTREEは一つのワーキングメモリ内の要素について，不要なパターンマッチング処理を減少させる。DTREEはデジタルサーチ[9]のアイディアを基本にしている。基になるデジタルサーチは文字列のみを対象にしているが，ここでは関数の戻り値のマッチングにも適応できるように拡張している。ONETはパターンマッチングの処理を，パターン依存関係に着目した特殊なネットワーク構造を利用して高速処理する。CRは競合解消を担当して，DTREEとONETによって変化の生じたワーキングメモリ内から次に発火するべきルールを選択する。

"K1"の独特の特徴は推論エンジン内に計画型問題やスケジューリング問題において，もっとも基本的な機能であるソート機能が組み込まれていることである。一般的なルールベース記述言語では内部にソート機能を持っておらず，ユーザーがソート関数に相当する機能をルールを組み合わせることで実現する必要がある。

図3，4で示したルールを使った推論性能テストの結果を図5に示す。このテストは単純化したスケジューリング問題の一つであり，製造する"製品"がn個与えられた時に，"製品"を作るために必要な"材料A"と"材料B"の組み合わせを探し，解を決定するという問題である。このような組合せ問題では，RETEアルゴリズムに比べて"K1"が推論性能が高いことがわかる。このテストの例はごく簡単な組合せルールであるが，実システムで頻繁に現れるパターンである。

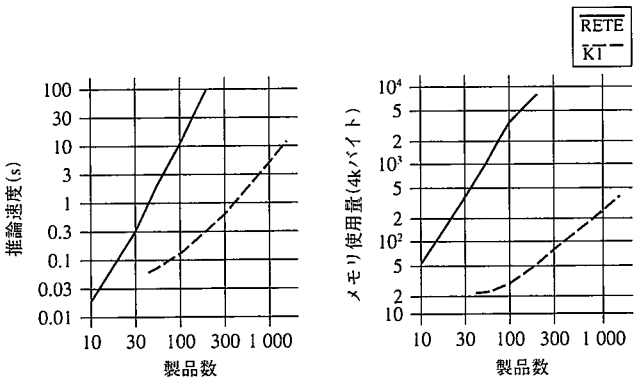


図5　パフォーマンステストの結果

2.1.4　移植性

移植性エキスパートシステムが動作するプラットフォームはPCからメインフレームまで様々であるため，"K1"は移植性の面でも配慮している。"K1"は，専用トランスレータを通すことでC++言語によるソースコードを生成することができ，そのソースコードは，特定のプラットフォームに依存しない。

更に"K1"自身もC++言語で実装されている。C++言語コンパイラーが動作しないプラットフォームでは，C++言語のソースコードをもう一度Cのソースコードに変換することで対応できる。これにより，C言語のコンパイラーしか動作しないプラットフォームにおいてもCのソースコードを少し修正することで移植可能であることが確認されている。

2.2 "K1"の開発環境

"K1"の開発環境概要を図6に示す。開発環境は五つのモジュールから構成されている。

2.2.1 トランスレータ

トランスレータ"K1"は独自の言語仕様を持っており，トランスレータにより知識ベースをC++言語のソースコードに変換する。変換といっても，これはプロダクションルールで表現された知識が手続き知識に翻訳されるのではなく，各々の知識要素が、適切な構造に再構築されるのである。例えば，ワーキングメモリエレメントクラス(wmeclass)のようなデータタイプはC++言語のクラス定義に変換され，条件部は弁別ネットワーク(discrimination network)に展開される。ルール作成が終了した後に，適当な処理(例えば，UNIXにおけるmakeコマンド)を実行することで，エキスパートシステムの実行形式が生成される。

2.2.2 推論エンジンカーネル

推論エンジンカーネルは，大きく二つの機能から構成されている。一つは基本機能要素であるパターンマッチング・競合解消・ルール発火の駆動機構などを受け持つライブラリ部である。この推論エンジンは一般的に前向き推論に分類されるものではあるが，"K1"においてはイベントドリブンと解釈するのが自然である。全てのワーキングメモリエレメントは，推論過程でオブジェクトとして扱われる。ワーキングメモリ内での更新やイベントが発生したタイミングでルールオブジェクトが送信されて，ルールを実行する。その結果，新しいイベントが発生することになる。

更に"K1"で構築されたエキスパートシステムは一つの推論エンジンクラスから導出されており，同じワーキングメモリを共有する。一つのワーキングメモリの情報を共有しながら協調するようなエキスパートシステム構築も可能である。もう一つは基本操作としての推論の実行・トレース実行やワーキングメモリの表示等，開発段階での対話的な操作のためのインターフェースを提供するトップレベル機能である。

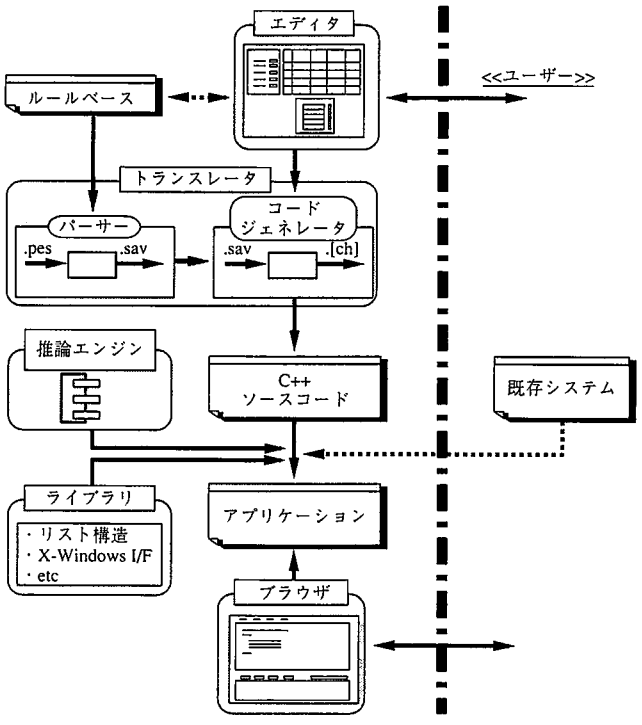


図6 K1の開発環境

トップレベルを使用すれば，ユーザーは対話的にエキスパートシステムの実行をトレース，コントロールできる。もちろん，トップレベル機能を全く使用せずに，全てのエキスパートシステム機能をユーザープログラムから直接詳細にコントロールすることも可能である。

2.2.3 エディタ

"K1"はインテリジェント・エディタを提供している。お互いに関連しあう知識の入力にはドラックアンドドロップの操作が有効である。例えば，あるルールが条件部中で，あるワーキングメモリエレメントを参照するような場合，ルールエディタの画面に参照したいワーキングメモリエレメントをドラックアンドドロップすると条件部が自動生成される。

また，修正／削除を行なおうとする際にはエディタが自動的に言語仕様との構文チェックを行なう。例えば，別の知識から参照されるある知識のインスタンスを削除しようとする場合は，エディタは知識間の関係をチェックして警告メッセージを表示する。

ここで示した二つの機能，すなわち文法的に正しい場合のみ生成可能にする機能("check and instantiate")と関連の整合性を保証する機能("consistency check in entity relation")は大規模な知識ベースを保守する上で非常に有用な機能である。なお，一般に開発マシンと実行マシンがことなる場合もあるので，知識ベースをエディタとテキストファイルの間で変換する機能も用意されている。

2.2.4 ブラウザ

多くの商用シェルでは開発中のエキスパートシステムのデバッグのためのツールが提供されている。"K1"において推論プロセスをコントロールする手段には，前述のトップレベル機能があるが，これには必要最小限の機能しかない。より高機能でユーザフレンドリなデバッグ環境として，専用ブラウザが用意されている。このブラウザを使うことで，現在のワーキングメモリエレメントの状態やパターンマッチングの状況，ルール発火候補等の推論プロセス情報をヴィジュアルに表示しながら推論をトレースすることができる。

2.2.5 ライブラリ

"K1"の開発環境自体に不可欠ではないが，いくつかの有用なクラスライブラリを提供している。計画型エキスパートシステムやスケジューリングエキスパートシステムにおいて，リストデータ構造はもっとも基本的なデータ構造であるが，"K1"ではこのようなデータ構造をサポートしていない。そこで，C++言語で実装したリストクラスをライブラリとして提供している。

これをインポートすることで，ユーザーはクラスとして利用したり，知識ベースの中で使用することができる。"K1"によるエキスパートシステムと既存のシステム間におけるデータの交換も重要である。推論の実行結果をファイルに保存したり，ファイルから読み込んだりするためのファイルI/Oクラスもあり，これを使用することで，システム駆動中の任意の時点で推論を中止・再実行させることもできる。

近年ではGUIを備えたアプリケーションが当たり前となっているが，通常GUIベースのアプリケーションとエキスパートシステムはそれぞれが自分用のメインループ関数を持つため共存できない。そこで"K1"は，メインループを持たず，従って競合が起きない専用のXlib描画クラスを提供している。シグナルハンドリングクラスは，制御型のエキスパートシステムで必要になるリアルタイム

推論をエミュレートするために提供されている。実行時間と周期を設定すれば，知識ベースの中で定義されたシグナルハンドリング関数が，周期的にコールされる。

3. 厚板精整管制支援システム

3.1 システム概要

"K1"を用いて厚板工場の精整工程における物流制御自動化システムのプロトタイプを開発した。精整工程は，圧延後に位置するラインで，傷手入れ，塗装，検査等を行なっているが，上工程である圧延作業の処理能力は，精整工程の処理能力より高いため，精製ラインには鋼板をストックするヤードと呼ばれるバッファーが用意されている。熟練オペレーターは，管制室から工場の物流を監視し，精整工程における適切な鋼板の流れを決定している。

主な管制業務は以下の四つである。

1) 物流制御(優先クレーン作業の設定，合流点の流量比率設定)
2) ストック(仕掛山)の管理(納期の厳守)
3) クレーン制御(複数台の効率運転)
4) 各工程，クレーン間の連絡

この制御が難しい理由は以下の二つに要約される。

1) リアルタイムに状況が変化し，設定時ではその影響の予測が難しい。
2) 評価指標が状況によって変化する。

3.2 システム構成と機能

システムの構成を図7に示す。システムを設計する段階でいくつかの手法を考慮した。

・ＯＲ手法
・物理モデル(物流シミュレーション)
・エキスパートシステム

計算が複雑であるため,まずＯＲ手法が除外された。物理モデルは，物流パターンが明確に導き出せるので，この目的に対して有効ではあるが，モデルが複雑化して，かつオペレータの専門的な業務固有の知識を実装することが非常に困難であった。

それに対して，このような領域知識を引き出すためにエキスパートシステムはもっとも良い方法であるが，物流パターンを明確化できない。そこで，両者のメリットを合わせ持つ，物理モデルとエキスパートシステムのハイブリッド構成でシステムを構築した。両手法の欠点を補うために，次のようなアイディアを考案した。

(1)ラインの基本的な物理挙動を再現させるラフなモデルを物流パターンを観測するため使用する。
(2)モデルに埋め込めない知識は，エキスパートシステムに実装する。

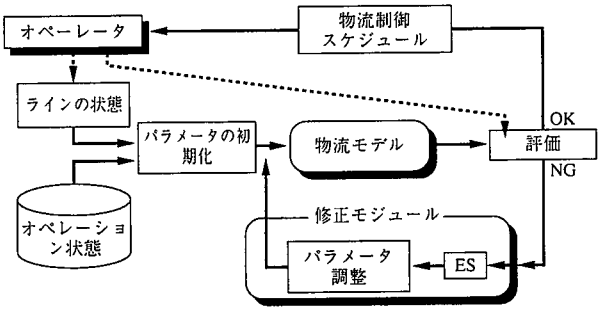


図 7　厚板精整管理支援システムの構成

(3)モデルの予測間隔を短くして，誤差が拡大する前にエキスパートシステムがモデルを修正する。

システムは，[物流モデル]と[修正モジュール]の二つの主モジュールから構成される。[物流モデル]は，与えられたオペレーション状態と機器の操業計画による物の流れをシミュレートする。その出力は他の要素(ライン全体の状況)の評価と統合して，CRT上に表示される。そして，オペレータは解の妥当性を判断する。得られた解が満足解でない場合は，[修正モジュール]で別の操業戦略を選択して，再度[物流モデル]で計算する。

[修正モジュール]は，[物流モデル]のパラメータを調節して，操業戦略を変更する。初期状態とライン状態が与えられると，モデル時間で2秒ごとにラインの状態が計算される。5回の反復，すなわちモデル時間で１０秒経過後，結果が[修正モジュール]に渡され，[修正モジュール]は，出力に基づくラインの状態を決定する。

一つのスケジューリング候補を得るために，このサイクルは，モデル時間で１０分間(実時間では約１０秒)繰り返される。ユーザーはより良い候補を探すために何度も試行することができる。[物流モデル]と[修正モジュール]間の詳しい関連を図8に示す。オペレータと物流コントロールシステム間のインターフェースはGUIにより構成される。

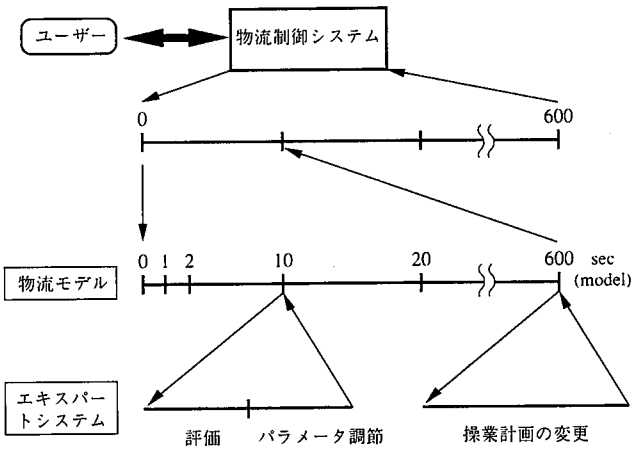


図 8　物流モデルと修正モジュールの関連

[物流モデル]は数値データ計算が中心処理であるので，C++言語で実装されている。対照的に[修正モジュール]は，[物流モデル]からの出力を評価するためにシンボルで表現された知識を利用して適切な操業戦略を決定する必要があるので，"K1"によるエキスパートシステムとして実装されている。[修正モジュール]は，熟練工から得られた専門知識を利用しており，多数のルールで表現されている。そして，多数のルールは，"K1"のグループ化機能を使用してグループ化されている。

現在の知識ベースの規模は，次の通りである。

41 ワーキングメモリエレメントクラス
887 ルール
51 ルールグループ
26 コンテクスト

システムはUNIXワークステーション上で開発され，[物流モデル]と[修正モジュール]はプロセス間通信で交信している。GUIはOpenwindowsを使用している。

4. 評　　価

厚板精整管制支援システムの効果と開発工程での"K1"の有効性を評価した。

4.1 システム化の効果

いくつかの典型的な状況で得られた実データを使用して，ヤードバランス最適化問題についてテストを行なった。このテストは，E，H，Kの3テーブル(コンベアー)間での鋼板の物流を制御する問題である。ヤードバランス最適化の目的は，各テーブルへの鋼板の分配とヤードへの均等な山積みである。

図9は各テーブルの鋼板の流れを示したものである。エキスパートシステムが組み込まれた[修正モジュール]は各テーブルの鋼板数をうまく制御していることがわかる。テストを通じて"K1"のエキスパートシステムは，熟練オペレータが下した判断と数にして80%一致する解を導出することができた。残りの20%は，突発事象によるもので，システムの仕様としてこれら突発事象の際にはシステムはオペレータの手動操作に従って操業するよう設計されている。

次に推論スピードにつての評価結果を表1に示す。比較のためにRETEアルゴリズムを使用したルールベース記述言語によるシステムの推論スピードを示している。結果は，RETEアルゴリズムに比べて，"K1"の推論スピードが約7倍上回っていることが分かる。

今回開発したシステムは，プロトタイプという位置付けではあるが，精製工程のスケジューリングを十分な推論速度で計算する能力を有しており，将来のルールの追加にも対応することができる。これによりシステムの精度が向上すれば，実機化も十分可能でオペレータ人員やコストの削減に貢献できるものと確信している。

4.2 "K1"の有効性

4.2.1 オブジェクト指向の特徴

本システムでは，膨大な外部データがシステムに参照され，最終的にワーキングメモリエレメントとして蓄積される。その過程で明らかになったことは，オブジェクト指向の方法論にのとって，ワーキングメモリエレメントクラスを最初に設計することが有効であったことである。設計の段階で，ワーキングメモリエレメント内での数々のデータ処理を明確化し，メソッドとして表現することで，データ依存処理を自然に表現できた。

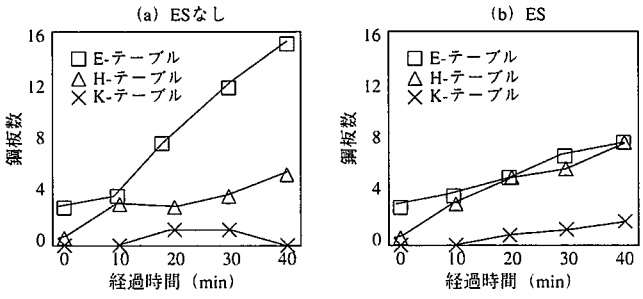


図 9　ヤードバランス最適化の結果

表 1　推論速度の比較

	RETE	K 1
TIME/RULE(s)	0.0085	0.00125

また，"K1"の文法では記述しきれない複雑なマッチング条件もあったが，この場合，ユーザーはメソッドでマッチングの条件を定義することが可能で，条件部でメソッドを参照することで，ルールを表現できる。本物流制御システムは，推論結果を保存するためのデータベースを有している。外部データとワーキングメモリエレメント間の矛盾を防ぐため，ワーキングメモリエレメントが変化したとき，ワーキングメモリエレメントの一種のメソッドであるデーモンが起動する。

例えば，ユーザーがある属性に対してデーモンを記述しておけば，その属性値が変化したとき，デーモンがコールされ，自動的に関連する外部データを更新して，ワーキングメモリエレメントとの一貫性を保つ。継承も良く利用される。構造をもった知識を自然に表現できるようにするだけではなく，知識ベースの記述性を向上させることができ，知識ベースの保守性も改善される。知識が継承によって定義されていれば，親の変更は全ての子に自動的に反映される。

継承は，プロダクションルール表現の利点である知識のモジュール性を損なうように考えられるが，著者らの経験からは，そのような欠点より効果の方が勝ると確信している次第である。

4.2.2 他のシステムとの統合

"K1"のシステムインテグレーション能力を，次に述べる二つの面から評価した。一つの面は，C++言語のダイレクトコーディングである。ダイレクトコーディングの重要なメリットは，冗長なルールを排除できることである。

例えば，実行部の計算結果に従って似ているがそれぞれ異なるアクションをとるような場合がある。このようなとき，C++言語のswitch-case文での直接コーディングすると記述を大幅に削減できると同時に，処理の手続き的な性格が明確化されるので可読性も向上する。C++言語のダイレクトコーディングを使用しないと，全てのアクションをルールでプログラミングする必要がある。

加えて，ルール数を減少させれば，パターンマッチングの回数も減少して，システムの実行速度が向上する。C++クラスのインポート機能も便利である。システム解析段階で[物流モデル]のデータ構造が最初に決定された。そしてそのデータ構造は，C++言語クラスのインポート機能を使用することで，システムの知識ベースの中に反映されており，物流モデルの評価結果をエキスパートシステム側に反映するための特殊なプログラミングは，ほとんど必要なかった。

もう一つの面は，ライブラリ形式で存在する外部モジュールとの結合である。システムの開発中に，過去のプロジェクトで開発されたたくさんのサブルーチン(C言語で実装されている)を再利用した。

前節で述べたが，これらのライブラリは,知識ベースの中に2，3のディレクティブを指定すれば，簡単にリンクできる。C++言語のダイレクトコーディングと共に簡単に使用できる機能である。最後に，外部のシステムと統合するだけでなく，"K1"モジュール自身の統合も有効な機能である。"K1"では，知識ベースの切り分けもサポートしているので，物流制御システムの知識ベースは，多数のモジュールに分割され,異なるエンジニアにより並列に開発された。

例えば，1人のエンジニアがあるルールグループに属する知識ベースを実装し，別のエンジニアは違うルールグループに属する知識ベースを実装した。そして，各々の知識ベースは独立にテストされ，動作が確認された後に統合することができた。各コンテクストでの知識ベースの分離により円滑な知識ベース開発と矛盾がない知識変更が容易にできるようになっている。

5. おわりに

"K1"と厚板精整管制支援システムの開発について述べた。オブジェクト指向を取り入れたことの特徴は，知識ベースの記述性の向上とルールベースプログラミングに開発方法論を採り入れたことである。"K1"は，特に計画型問題やスケジューリング問題のような合成型問題である大規模システムの開発に適用可能なように設計されている。いくつかのテストを通じて"K1"の推論アルゴリズムの有効性を示した。"K1"の全てのソフトウェアは実行速度の高速化を達成すると同時に移植性を確保するためにC++言語で作成されている。

また，開発環境として，エディタやブラウザが用意されている。これらにより，ルールーベース記述言語に慣れていないエンジニアでも，効果的な知識ベースの構築と開発期間の短縮ができ，保守時間も削減されるものと考える。"K1"厚板精整管制支援システム事例に基づき，管制システム及び開発ツールとしての"K1"の有効性を検証した。本管制システムは，実用に向けての検証が行なわれている段階である。しかし，これまでの開発を通じて，オブジェクト指向を取り入れた"K1"の有効性は明らかであり，"K1"は，エキスパートシステムと既存システムを結び付ける有効な統合ツールであると確信している。

次の段階では，新たな知識を付け加えて，厚板精整管制支援システムの向上をはかり，"K1"の次期バージョンでは，システム開発の過程で得られたフィードバックを反映したい。

参照文献

1) Barr.A., Feigenbaum.E.A. : The Handbook of Artificial Intelligence. William Kaufmann Publishers Inc., 1982
2) Minami.E., Miyabe.Y., Dairiki.O. : Proceedings of the 3rd International Conference on Industrial & Engineering Application of Artifical Intelligence and Expert Systems. Vol.2, 1990, pp.684-691
3) Minami.E., Hirata.T. : Proceeding of the World Congress on Expert Systems. Vol.2, 1991, pp.1311-1318
4) Meyer.B. : Object-oriented Software Construction. ASCII Co., 1991
5) Stroustrup.B. : The C++ Programming Language. 2nd edition, Addison-Wesley Publishing Co. Inc., 1991
6) Rumbaugh.J. : Object-Oriented Modeling and Design. Prentice Hall Inc., 1991
7) Forgy.C.L. : Artifical Intelligence. 19, pp.17-37, 1982
8) Miranker.D.P. : TREAT:A New and Efficient Match Algorithm for AI Production Systems. Morgan Kaufmann Publishers Inc., 1990
9) Aoe.J., Yamamoto.Y., Shimada.R. : Proceedings of the First International Conference on Supercomputing Systems. Florida, 1985, pp.491-498