データベースシステム 94－7
（1993．7．21）

# 計画作成用データベースツール VicePlannerのビュー機能

掛下 哲郎　　前田 明子
佐賀大学理工学部情報科学科
山下 義久
日本ソフトウェア工学

人の活動に関するスケジュール作成は、重要なスケジューリング問題の 1 つである。この問題は、次のような特徴を持つ。(1) スケジュールは種々の観点から評価される、(2) スケジュール評価は人的要因に依存するので、必ずしもアルゴリズム的に評価できない。VicePlanner は、このようなスケジュール作成を支援するためのツールである。従来のスケジューリングツールは、エキスパートシステム技術に基づいているものが多い。これに対して、VicePlanner は、計画作成者の作業をより便利にするために、ビュー機能とスケジュール編集を始めとするデータベース技術に基づいて設計されている。本稿では、VicePlanner によるスケジュール作成の基本となるビュー機能について述べる。このシステムは、現在 OODBMS ONTOS を使用して開発中である。

# View Function of VicePlanner : A Database Tool for Scheduling Human Activities

Tetsuro Kakeshita　Akiko Maeda
Department of Information Science, Saga University,
Sage 840, Japan
Yoshihisa Yamashita
Nihon Software Engineering Inc.

Constructing schedules for human activity is an important issue among scheduling problems. This problem has several characteristics such as: (1) schedules are evaluated from several viewpoints and (2) schedule evaluation often depends on human factors so that a schedule cannot always be evaluated algorithmically. VicePlanner is an assistant tool for constructing such schedules. Conventional scheduling tools are mostly based on expert system technologies, but VicePlanner is based on database technologies, such as view function and schedule manipulation, in order to help schedule designer's activity more conveniently. This paper describes view function, which is a core for schedule validation and manipulation, of VicePlanner. The system is currently under development using OODBMS ONTOS.

# 1 まえがき

日常生活や職場において、スケジューリング問題は、仕事の割り振りや行動予定、収支計画などを考える際に数多く出現する。ところが、このようなスケジューリング問題は、人が関係する部分が多くなるに従って、解くことが難しくなる。なぜならば、人は好みや都合といった、曖昧な評価の基準を持っており、その評価の基準もまた、人によって異なっているからである。さらに、都合によりスケジュールが変更されることもしばしば起こる。VicePlannerは、こういったスケジューリング作業を支援するためのツールである。

スケジューリング問題は、これまで、オペレーションズリサーチ (OR) の分野で研究され、列車の時刻表や製造工程表などを作成するシステムが、エキスパートシステム (ES) 技術を利用して実現されてきた。ところが、エキスパートシステムは特定のスケジューリング問題にしか対応していない。最近は、さまざまなタイプのエキスパートシステムを生成できるようなエキスパートシェル [金井 90] の研究も進められており、確かに、大規模なスケジューリングや、正確さや緻密さを要求されるスケジューリングには効果的である。しかしながら、エキスパートシステムは、推論規則に基づいたシステムであるため、人をスケジュールするためのシステムに利用しようとすると、次のような問題が生じる。

- 互いに衝突し合う評価基準が存在した場合、それらの間でトレードオフを行うことが難しい。人のスケジュールには、それに関わる人々の立場により、評価基準同士が衝突することがしばしばある。
- 人のスケジュールには、個人の能力や好みなど、推論規則で表せない要素が含まれることが多い。
- エキスパートシステムの開発は、専門家から知識をひとつひとつ漏らさず聞き出し、推論規則に直す作業が必要である。しかし、そのためには大変な時間と労力が要求される。中規模のスケジューリング問題に対しては、このような手間は、時間の浪費である。
- 人に関するスケジューリング問題は、定型的でないため、それぞれに対してエキスパートシステムを開発する必要がある。しかし、これは現実的ではない。
- 人のスケジュールは、状況に応じて変更される。その過程で推論規則に矛盾が生じた場合、矛盾を解決することは難しい。

このようにエキスパートシステムでは十分に対応できない、曖昧な評価基準を持つ中規模スケジューリングのための汎用支援ツールとして、VicePlannerはデータベース (DB) 技術を利用し、次に挙げる快適な作業環境を提供する。

- スケジュール作成者が、作成したスケジュールを種々の観点からチェックするためのビュー機能を提供する。ビューは、与えられた選択条件を満たすスケジュール対象（ジョブ）を表示する。複数のビューを表示することによって、あらゆる角度からスケジュールのチェックができる。
- スケジュール作成者は、ビュー上でスケジュールを編集できる。スケジュール編集機能としては、新しいビューの定義、ジョブの追加/削除/変更/コピー、ジョブ構造の変更などがあり、スケジュールの段階的作成が可能になる。

さらに、VicePlannerは、ジョブ間の関連もサポートする。これは、スケジュールのチェックや初期スケジュールの自動生成に使用することができる。

この論文は、次のように構成されている。2節では、スケジューリング問題について述べ、VicePlannerで取り扱う問題の範囲を定義する。また、ジョブ構造に対する編集機能も説明する。3節では、スケジュール作成におけるVicePlannerの利用法について説明し、スケジュールの段階的作成の概念を導入する。4節は、この論文の中心部分である。ここでは、特にビュー上での操作を中心にVicePlannerのほとんどの機能を説明する。5節では、関連の取り扱いについて述べる。

# 2 スケジューリング問題の定義

スケジューリング問題は、属性を相互に関係づける問題として定義される。スタッフに仕事を割り当てる問題を考えてみよう。仕事とスタッフの集合は、2つの属性の集合に相当する。従って、ユーザーはこのスケジューリング問題を、スタッフを仕事に割り当てる問題と同一視することができる。属性間の対応は、多対多になってもよい。

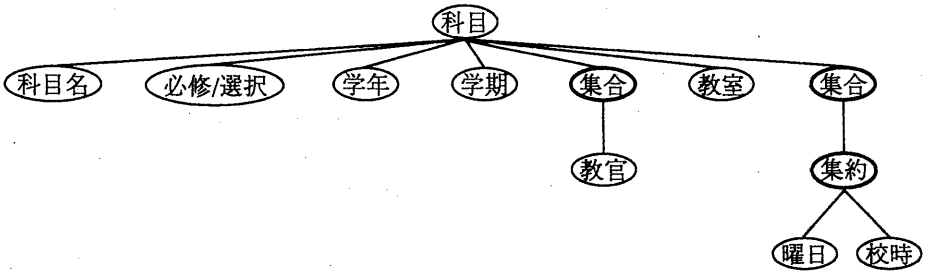



図 1: ジョブクラス '科目' の木構造

ジョブは、異なる属性の集合の組で構成される。例えば、大学の時間割を考える。講義すなわち属性は、**火曜日**（属性値）の **2 校時**（属性値）に割り当てられるかもしれない。1 つの講義が、1 週間に何度か行われることもある。その結果、属性の集合や属性の組などがジョブの構成要素になることがある。これらの集合や組を表現するために、コンポーネントの概念を導入する。

これらの点を考慮して、属性とジョブを次のように定義する。属性クラス $A_i$ は、同じ型からなる $n_i$ 個の属性値の集合 $\{a_1, \cdots, a_{n_i}\}$ である。一般に、$N$ 個の属性クラス $A_1, \cdots, A_N$ が存在する。ジョブクラス $J$ は、$m$ 個の要素 $\{j_1, \cdots, j_m\}$ の集合であり、識別名を持つ。それぞれの $j \in J$ は、コンポーネントクラスのリストで定義された同一構造を持つ。コンポーネントクラスは、識別名を持ち、以下のように再帰的に定義される。

**原子コンポーネントクラス** 属性クラス $A_i$ は、コンポーネントクラスである。

**集約コンポーネントクラス** $C_p, \cdots, C_q$ がコンポーネントクラス（異なる型も可）ならば、集約 $< C_p, \cdots, C_q >$ はコンポーネントクラスである。

**集合コンポーネントクラス** $C_s, \cdots, C_t$ が同じ型のコンポーネントクラス $C$ ならば、集合 $\{C\}$ は、コンポーネントクラスである。

コンポーネントクラスのインスタンスを、コンポーネントと呼ぶ。コンポーネントクラスは、次の手続きによって構成された木構造に対応する。(1) 原子コンポーネントクラス $A_i$ は、ラベル $A_i$ の葉に相当する。(2) 集約コンポーネントクラス $< C_p, \cdots, C_q >$ は、親が集約ノードで、子がそれぞれの $C_p, \cdots, C_q$ の木に相当する。 (3) 集合コンポーネントクラス $\{C\}$ は、親が集合ノードで、子が $C$ の木に相当する木である。2 つのコンポーネントクラスは、それが同じ木構造の時、同じ型である。ジョブクラス $J$ の構造は、集合コンポーネントに相当するので、$J$ は木構造によって表現できる。

**例 1** 時間割作成の時に定義されるジョブクラスを考えると、次のような属性が定義されている。**科目名**、**必修/選択科目**、**学年**、**学期**（**前後期**）、**教官**、**教室**、**曜日**と**校時**。ジョブクラス '科目' は、図 1 に示す木構造で表現できる。集約/集合ノードには名前をつけることもできる。 □

ジョブクラス操作は、そのクラスを表す木構造上の操作によって定義される。コンポーネントクラスも木構造で定義されるので、同一の操作がコンポーネントクラスに対しても定義される。**VicePlanner** は木構造を操作するために、以下の機能を提供する。

**ジョブクラス定義** この機能は、属性や集約/集合ノードの追加や削除といった、ジョブクラス構造の定義を行う。ジョブクラスに属するジョブが存在する場合には、ジョブの構造はクラス定義に合わせて変更される。

**属性保護** ジョブクラスが、スケジュール作成中に値が変更されてはならない属性を持つ時、それらは木構造の中で指定できる。このような属性は、編集を禁じられる。この機能により、ミスによる変更を防止できる。

**属性参照と編集** ジョブクラスが定義されると、そのクラスのジョブを生成することができる。通常、ジョブはビュー機能を使って検索、編集されるが、コンポーネント木構造上で属性値を参照/編集することも可能である。この場合、ジョブはビューの中で指定され、属性クラスは木構造上で指定される。

属性クラスは、ジョブクラスと同様に編集されるべきである。このような編集は、属性クラスの生成、削除、クラス内での属性値の生成、削除を含む。生成されたジョブとコンポーネントは、再利用するためにデータベースに保存される。各ジョブやコンポーネントに対する操作は、4 節で述べる。

## 3 スケジュールの段階的作成

複雑なスケジューリング問題では、与えられたジョブの集合にさまざまな属性を与える必要がある。例えば、科目を表すジョブは、教官や、学年、曜日、校時などを与えられる。このような属性を一度に割り当てるのは大変難しい。その代わりに、スケジュール作成者はしばしばその問題を属性割り当て問題の列として考える。VicePlanner におけるスケジュール作成の基本ステップは以上の考察から導かれた。

1. ジョブと属性の集合を作成する。既存のジョブは、データベースに保存されているので、新しいジョブクラスを定義するために再利用できる。VicePlanner は、ジョブクラスに属性を動的に追加、削除できるので、スケジュール作成中に属性値を追加することもできる。
2. 以下のステップを、満足なスケジュールが得られるまで繰り返す。
   (a) スケジュール作成者は、各ジョブに属性値を割り当てる。また、ジョブクラスにも、スケジュール作成中に必要なだけ属性クラスを加えたり、消したりできる。属性割り当て機能は、次節に述べるビュー機能によって提供される。
   (b) スケジュールは、そのスケジュールの要求に従って定義されたビュー上でチェックされる。もし、問題が見つかれば、ビューのスケジュール編集機能を利用して、属性値の変更ができる。

これを、スケジュールの段階的作成と呼ぶ。この手法は、次のような特徴を持つ。

- ジョブは、スケジュール検査のために、任意の値を属性値として持つことができる。属性は、ビュー機能を使用して割り当てられるか、スケジュール作成の前に入力されている。
- スケジュールは、繰り返し編集できる。ジョブの集合が最初に生成され、各ジョブは、簡単な属性を持つ。次に、スケジュールは、ジョブクラスへの属性クラスの追加と追加後のスケジュール検査の繰り返しにより、徐々に目的のものに近づく。
- 生成されたジョブとコンポーネントは、データベースに保存されるので、以前のスケジュールの全部または一部を新しいスケジュールのために再利用できる。

段階的作成は、非常に便利である。なぜなら、スケジュール作成者はしばしば与えられたスケジューリング問題を簡単で、より処理しやすいものに分けるからである。段階的スケジュール作成の例を、カリキュラムの作成によって示す。

**例 2** カリキュラムの作成は、コンピュータアーキテクチャ、アルゴリズム、ソフトウェアなど、教育の分野を考えることから始まる。それから、カリキュラム作成者は、科目が特定の分野に偏らないように科目の集合を生成し、分野を割り当てる。次に、各科目は、学生にとっての重要度によって、必修または選択かを決定される。さらに、各科目は科目間の順序関係と各学期の学生の負荷を考慮して、学期を割り当てられる。これらの割当が終了すると、各科目に対して教官を割り当てる。その際には教官の専門分野と負荷を考慮しなければならない。最後に、各科目は曜日と校時（集合でも良い）を割り当てられる。このとき、学生の都合（再履修が可能なこと等）や教官の都合（会議等との重複をできるだけ避ける等）を考慮する。以上の手順によって、大学の時間割は決定される。 □

スケジュールは、ビューを通して確認、編集される。ビューの詳細な定義と機能は次節で述べる。

## 4 ビュー上でのスケジュール編集

VicePlanner のビューは、スケジュールの基本的な確認、編集を行うための機能を提供するツールである。それは、表形式でスケジュールを表示することを基本にしている。最初に次の例を考えてみよう。

**例 3** 次の時間割は、大学における時間割を表している。行と列は曜日と校時を意味し、この時間割は、特定の教官 中野 が担当する科目の名前を表示

している。

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 月 | | データベース | ハードウェア | |
| 火 | | | 実験 | 実験 |
| 水 | | ソフトウェア | | |
| 木 | | | 実験 | 実験 |
| 金 | | | | |

選択条件：教官 = 中野

ある教官によって教えられる科目が選択されたので、スケジュール作成者は、その時間割が教官自身のスケジュールと合うかどうかチェックすることができる。もし合わなければ、スケジュール作成者はその表の編集を行う。'データベース' を金曜日の 2 カラム目に移動することは、そのジョブの属性値の再割り当てに相当する。選択条件を、**必修/選択 = 必修** に変更することによって、必修科目だけを表示することができる。スケジュール作成者は、このビュー上で 2 つ以上の必修科目が同時に開講されていないことをチェックすることができる。学年が違う講義であっても、同時に開講されている必修科目があると、再履修が不可能になる。もし、条件が **学年 = 4 年生** ならば、その時間割が 4 年生にとって都合が良いかどうかをチェックできる。 □

## 4.1 ビューの定義

VicePlanner における簡単なスケジュールの確認と編集は、上述した例のように実行される。上の表の各部分は、特定の属性値を持つジョブの集合を示している[1]。これを考慮すると、ビューは固有のビュー名、選択されたジョブのジョブクラス名、そのジョブクラスの属性クラス名の集合、下に定義する選択条件から構成される。[2]

1. $A$ がジョブクラスを構成する属性クラス名で $a \in A$ ならば、式 $A = a$ は、選択条件である。
2. 式 $A =$'undefined' は選択条件である。
3. 二つの式 $F_1$ と $F_2$ が選択条件ならば、$(F_1 \vee F_2)$、$(F_1 \wedge F_2)$、$\neg F_1$ は選択条件である。

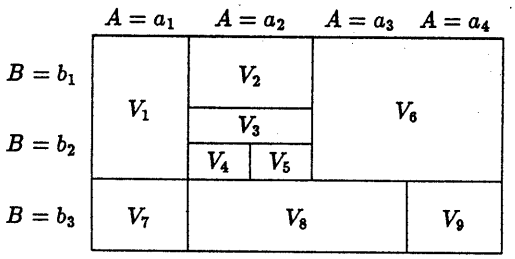
[1] 以後の定義では説明を簡単にするため省略するが、コンポーネントの要素もビューに表示できる。

[2] 読者はビューが拡張関係データベースへの質問結果と同等のものを表示することに気づいたかもしれない [StonKem91]。それとの違いは、(1) ビューはスケジュール編集機能を持つ、(2) 2 つの同じ属性値を持つジョブが同時に存在できることの 2 点である。

2 番目の定義は、指定した属性が値を与えられていないことを表す。ビューは、指定されたクラスに属し、選択条件を満たすジョブの指定された属性を表示することができる。もし、ジョブが集合コンポーネントを含むならば、少なくとも一つの属性値が選択条件を満たす時そのジョブは選択される。指定された属性リストが集合属性を含む場合には、選択されたジョブの全ての属性値が表示される。ビューは、VicePlanner 上で生成、削除、再定義できる。それは、コンピュータ画面上に表示され、移動、リサイズが可能である。ビューはデータベースに保存され、ビュー定義を使って検索できる。

例 3 に示した表は、ビューの集合によって定義できる。さらに、より複雑な表もこのようなビューの集合によって定義できる。二つの例は、選択条件の集合 $\{(A = a) \wedge (B = b) \wedge \cdots \wedge (Z = z) | a \in A, b \in B, \cdots, z \in Z\}$ によって定義された $n$ 次元の入れ子の表と、次の例で示す非正規化された表である。

**例 4** 属性集合 $A = \{a_1, a_2, a_3, a_4\}$、$B = \{b_1, b_2, b_3\}$、$C = \{c_1, c_2\}$、$D = \{d_1, d_2\}$ を考える。次の表はビューの集合 $\{V_1, \cdots, V_9\}$ によって定義されている。




各ビューは、同じジョブクラスと属性リスト上で定義されている。違いは選択条件にある。例えば、$V_1, \cdots, V_4$ は次の選択条件によるものである。

$V_1$ : $(A = a_1) \wedge ((B = b_1) \vee (B = b_2))$
$V_2$ : $(A = a_2) \wedge (B = b_1)$
$V_3$ : $(A = a_2) \wedge (B = b_2) \wedge (C = c_1)$
$V_4$ : $(A = a_2) \wedge (B = b_2) \wedge (C = c_2) \wedge (D = d_1)$

他のビューの選択条件も同様に定義できる。 □

例 4 に示したように、ビューの集合は属性値によってジョブを分類するツールと考えることができる。この機能を強化するために、ビューの集合である集合ビューを定義する。集合ビューは、固有の名前を持ち、ビューデータベースに保存される。集合

ビューに属するビューの間で共通した定義は同時に変更できる。集合ビュー上の操作は、ビュー上の操作と同様に定義される。

## 4.2 ジョブの移動

次に、もう1つの重要なビュー機能、すなわちスケジュール編集について説明する。ジョブをビューから他のビューへ移動すると、2つのビューの選択条件間で一致しない属性値が変更される[3]。このようなジョブの移動は常に可能とは限らない。属性 $S$ の値が $a$ であるようなジョブを考えよう。このジョブを、選択条件 $S=b$ によって定義されたビューへ移動すると、$S$ の属性値は $b$ に変更される。しかし、このジョブを $(S=b)\vee(S=c)$ のビューへ移動することはできない。なぜならば、$S$ の新しい属性値を一つに決定できないからである。これに対して $S$ の属性値が $b$ または $c$ であるようなジョブは $(S=b)\vee(S=c)$ のビュー上へ移動することができる。この場合には属性値の変更は生じない。

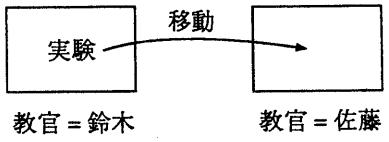
他の問題は、ジョブを他のジョブクラス上に定義したビューへ移動する時に起こる。まず、同じジョブクラス内での移動を考えよう。

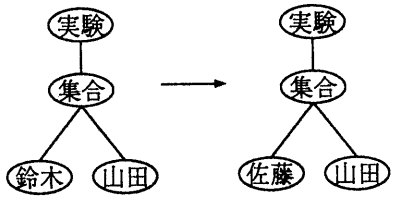
ジョブ $x$ は、$x$ の属性値 $A=a$ に対して次の二つの選択条件の1つが満たされる時、ビュー $V$ へ移動できる。

- $A=a$ が $V$ の選択条件によって満たされている。
- $A=a'$ が $V$ を満たすような唯一の $a'\in A$ が存在する。

属性クラスによっては、ジョブは複数の属性値を持つ。この場合、移動元のビューの選択条件が調べられる。条件を満たしている全ての属性値が、結果として変更される。全ての値に対して唯一の属性値が存在しないならば移動は受け付けられない。もし、結果的に同じコンポーネント内の属性値と同じになる時は、併合される。以下に例を示す。

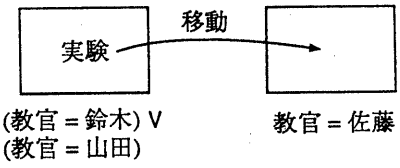
**例 5** **教官 = 鈴木** のビューで、科目を表すジョブ '実験' を **教官 = 佐藤** のビューへ移動すると、下図のように '実験' の **教官** 属性値 { **鈴木**, **山田** } のうち、**教官 = 鈴木** の条件にマッチする値 **鈴木** だけが **佐藤** に変更される。

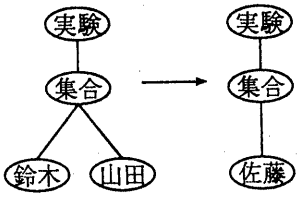



□

**例 6** **(教官 = 鈴木)∨(教官 = 山田)** のビューを考える。科目を表すジョブ '実験' は両方の教官によって教えられるとしても、ビュー内では一つのものとして表される。'実験' を **教官 = 佐藤** のビューへ移動すると、下図に示すように '実験' の **教官**属性は1つの属性値**佐藤** だけを持つことになる。




□

ジョブを、$A$ ='**undefined**' のビューへ移動すると、$A$ の属性値を削除することができる。また、属性 $A$ がジョブクラス上で編集を禁じられている場合には、$A$ を変更するようなジョブの移動は受け付けられない。

ジョブクラス間でジョブを移動する場合には、対象となるジョブクラスの定義によってジョブがキャストされる。2つのジョブクラスのコンポーネント間の対応は、スケジュール作成者によって指定されなければならない。

[3] ビューはコンポーネント上で定義されるので、ビュー間でコンポーネントを移動することもできる。説明を簡単にするために、以下ではジョブクラスの操作のみを説明する。

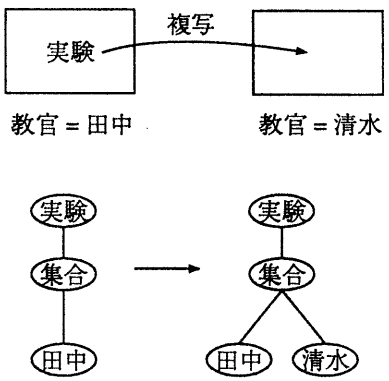
### 4.3 ジョブのコピー

もう 1 つのスケジュール編集機能はコピーである。コピーには、ジョブのコピーとコンポーネント追加の 2 種類がある。ジョブのコピーは、同じ属性値を持ち、ジョブ名が異なる新しいジョブの生成である。コンポーネント追加は、指定したジョブの集合コンポーネントに属する他のコンポーネント値を追加することである。

ジョブがビュー内でコピーされる時、ジョブのコピーは容易である。この場合、同じジョブクラスの新しいジョブが生成され、全ての属性値がコピーされる。ジョブが他のジョブクラスへコピーされる場合には、状況はより複雑になる。この場合、ジョブの移動と同じ方法でコンポーネントのキャストが起こる。

次に、コンポーネントの追加についてより詳細に説明する。ジョブがビューから他のビュー（同一ジョブクラス）へコピーされると、コンポーネントの追加が起こる。古い属性値と新しく定義された値が結合されて新しいコンポーネントを構成する。

**例 7** 属性値 **教官 = 田中** を持つ科目を表すジョブが、選択条件が **教官 = 清水** であるビューにコピーされる時、その科目は、**教官 = {田中, 清水}** の 2 つの属性値を持つことになる。




□

新しい属性値は、ジョブの移動と同じ方法で生成される。移動との違いは、コピーされた属性に集合コンポーネントが定義されていないときには、コンポーネントの追加が受け付けられないことである。

### 4.4 VicePlanner のビュー機能

以上述べた定義を用いて、ビューはスケジュール編集に次の機能を提供する。

**ビュー定義** ビューは、4.1 節のように定義される。スケジュール作成者はしばしばビューの集合の表表示を必要とするので、ユーザーインターフェイスとして複合ビューの定義も考えなければならない。ビュー定義の後、作成者は選択条件の AND/OR/NOT の計算によって生成したビューから異なるビューを新たに定義できる。これは、ジョブを検索、グループ分けして複合表を生成するのに便利である。ビューの定義はビューデータベースに保存されるので、集合ビューの定義に、それらを再利用、編集、参照することができる。

**ジョブ選択** ビューに表示されている 1 つまたは全てのジョブの集合を選択することができる。選択されたジョブを移動、コピー、削除できる。ビューは選択条件を使って任意に定義されるので、特定の条件を満たすジョブを同時に選択できる。

**属性値変更** ジョブがビュー間で移動されるとき、属性値は対象ビューの定義によって変更される。2 つのビューが異なるジョブクラス上で定義されている場合には、ジョブのキャストが起こる。

**コンポーネント追加** コンポーネントの追加は、同じジョブクラス上で定義された他のビューへのジョブのコピーによって定義される。

**ジョブコピー** ジョブのコピーは、単一ジョブの 2 つのジョブクラス間で実行できる。クラス内のコピーは容易である。しかし、クラス間のコピーは、型キャストとスケジュール作成者による属性マッピングを必要とする。

**ジョブ削除** 選択されたジョブは 1 つの操作で削除できる。

## 5 ジョブ間関連

ジョブの集合は、関連によってリンクされることがある。このような関連には、優先順位、複数ジョブからの選択関連、ジョブ間の集約関連などがある。この他に、並列に実行できるジョブ間の関連や、仲の良い友人（ジョブ）の間に存在する関連なども定

義できる。これらの関連を利用して、スケジュールのより良いチェックが可能になる。

関連を扱うために定義された操作には以下の3つのタイプがある。

**関連のリスト表示** 特定のジョブが含まれる関連を全て列挙する。

**ジョブのリスト表示** 特定の関連によって関係づけられたジョブを全て列挙する。

**関連定義** 関連を定義し、その関連が指定されたジョブを含むようにする。特定の関連からのジョブの削除も必要である。

上記の操作を実現するために、関連を属性の特別な型として定義する。関連の型に相当するいくつかの属性クラスを定義することが可能である。特定の関連は、属性値として関連名を持つ属性クラスのインスタンスに相当する。このように、同じ型の関連を複数個定義できる。関連インスタンスは複数のジョブによって参照できるので、複数のジョブから構成される関連も容易に定義できる。

ジョブ間の関連を表示するために、各ジョブに関連属性の集合からなる集合コンポーネント'**関連**'を定義する。従って、ジョブは一つ以上の関連に属することができる。このモデルのもとで、関連のリスト表示は、**関連**属性の値を見ることによって実現できる。ジョブのリスト表示は、選択条件**関連** = '関連名'を持つビューを定義することによって実現できる。さらに、ジョブは関連を定義したビューへのコピーによってその関連に属することができる。関連からの削除も同様に実現できる。

上記で述べた方法には1つの欠点が存在する。順序関連のように、方向性のある関連は直接表現できず、方向性のない関連しか表現できないことである。この問題点を解決するために、方向性のある関連に対して別の属性クラスを定義し、クラスの中で、関連の取り扱いを変更することが考えられる。

## 6 むすび

VicePlannerは現在オブジェクト指向DBMS ONTOSを利用して開発されている[ONTOS 91]。これは、オブジェクト指向DBがジョブの木構造を表現したり、木構造を操作する上で便利なためである。もう1つの理由は、オブジェクト指向の方法論はイベント駆動型の動作を行うので、GUIとの親和性に優れていることである。エキスパートシステムのアプローチは、大規模で定型的なスケジューリング問題を解くためには重要であるが、日常業務ではより柔軟なシステムが望まれていると考えている。

今後の研究課題を下に挙げる。

- VicePlannerを大規模なスケジュール作成に利用するためには、推論規則に基づいたアプローチを使用したスケジュール検査も必要となる。結合される推論規則によって、スケジュール検査の一部を自動化し、ビュー上で編集するための初期スケジュールをVicePlannerが自動生成できる。この目的のために、協調型スケジューリング[森下 90]が利用できる。
- 大規模な計画は、しばしば多くの人々の協同作業によって作成される。VicePlannerがこのようなCSCW機能をサポートすることが望ましい。[Ellis91]
- VicePlannerで処理できない他のスケジューリング問題のクラスが存在する。少なくともそれらのいくつかに対応するため、ツールの拡張が望ましい。

**謝辞** この研究は日本ソフトウェア工学株式会社の援助を受けている。佐賀大学情報科学科の渡辺義明教授や松藤信哉助手との討論は定義を明確にするため非常に役立った。

## 参考文献

[Ellis91] C.A.Ellis, et al., "Groupware: some issues and experiences", *Comm. ACM*, Vol.34, No.1, pp.38–58, 1991.

[金井 90] 金井、戸沢, "プランニング業務のためのエキスパートシステム構築環境", 人工知能学会誌, Vol.5, No.2, pp.220–230, 1990.

[森下 90] 森下 他, "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.184–193, 1990.

[ONTOS 91] ONTOS Inc., "ONTOS Developers Guide", 1991.

[StonKem91] M.Stonebraker, G.Kemnitz, "The POSTGRES next generation database management system", *Comm. ACM*, Vol.34, No.10, pp.78–92, 1991.