ソフトウェア工学　80－5
（１９９１．７．１８）

リポジトリ仕様の比較

外山 勝保　　磯田 定宏
NTTソフトウェア研究所

ＣＡＳＥツール統合環境の実現において、リポジトリは重要な役割を果たす。リポジトリの技術レベルと将来の方向を明らかにするため、３つのリポジトリ仕様をツール統合の観点から比較した結果について述べる。まず、ツールを開発環境に組込む場合にリポジトリに要求される条件や機能をツールデータ統合面・ツール制御統合面・ツール環境面に分類して列挙し、各項目についてリポジトリ仕様の満足度を調査比較した。その結果、要求条件・機能に対する各リポジトリ仕様間の技術的差異と、今後の検討を必要とする項目が明らかにした。

Comparison of the Specifications of Software Design Information Repository

Katsuyasu TOYAMA　　Sadahiro ISODA

NTT Software Laboratories
NTT Shinagawa TWINS BLDG., 1-9-1 Kohnan, Minato-Ku, TOKYO 108, JAPAN

Software design information repository plays an important role in integrating various CASE tools. We report the comparison of the specifications of three software design information repositories in order to show their current technology and the future directions. First, we made a list of the requirements for software information design repository. Next, we checked whether each specification of the repositories satisfies these requirements. Last, we summarized the differences among the specifications and the problems to be solved.

I. はじめに

CASE ツール統合環境を構築する際には、リポジトリ(ソフトウェア開発情報管理用データベース)が重要な役割を果たす。これは、リポジトリに一元管理された設計情報を様々なツールが修正・削除・追加することで、各ツール間での設計情報の有効利用（共有や再利用）が可能であること、および開発環境全体での設計情報の一元管理が実現可能という理由による。更に、リポジトリに格納する情報とその情報の利用方法の追加によって、様々な開発支援が可能になることも理由として挙げられる。例えば、ある開発作業に関して作業の開始条件や終了状態およびその作業で使用するツールとデータ等の情報をリポジトリで管理し、この情報を利用してツールの自動起動やユーザへの通知を行うことで、開発作業の自動化が実現できる。このように、CASE環境統合の要として様々な情報を管理するリポジトリには、データベース的機能に加えて、情報の版管理、並行作業の制御などのCASE環境特有の機能が要求される。

現在リポジトリ仕様の標準案が幾つか提案され、またリポジトリ製品も幾つか現れている。例えば、リポジトリ仕様の標準案としては、ISO IRDS(Information Resource Dictionary System)[4], ECMA PCTE(Portable Common Tool Environment)[1], CIS ATIS(A Tool Integration Standard)[2] 等が、ベンダのリポジトリ製品としては IBM 社の AD/Cycle RM(Repository Manager)[3]、Atherton Technology 社のSoftware Backplane 等が挙げられる。これらはCASEツール統合環境構築が目標である点は同じであるが、各々データモデルや支援する機能には差異がある。

以下では、CASE環境を構築する上でリポジトリに要求される機能や条件を、各リポジトリ仕様がどの程度満足しているかを比較することにより、現在のリポジトリ技術と、将来の方向を明らかにする。II章で比較対象とするリポジトリについて簡単に触れる。III章では、リポジトリに要求される機能を列挙し、IV章で各リポジトリ仕様の充足度を比較結果を示し、特に幾つかの条件・機能について詳細に比較する。最後にV章で今後の検討課題について述べる。

II. 比較対象とするリポジトリ

リポジトリの比較対象として、以下の理由から ECMA PCTE、CIS ATIS、IBM AD/Cycle RM を選択する。すなわちこれらのリポジトリが、CASE 環境を指向していること、今後のリポジトリ国際標準(ISO IRDS)に多大な影響を及ぼすと予想されること、の２点が理由として挙げられる。

これらリポジトリ仕様におけるCASE環境統合のアーキテクチャを図１に示す。CASE環境の統合は、ユーザインタフェース統合、データ統合、制御統合の３つの側面[7]から整理できる。リポジトリは、ツール間で共有される設計情報を管理・提供するだけでなく、ツール制御情報などを通じてツール制御手段を提供するため、データ統合と一部の制御統合を担っていると考えられる。

本資料でのリポジトリ仕様の比較のレベルは、リポジトリの提供するサービスインタフェースのレベルとする。つまり、ユーザやツールへの設計情報の見え方および使い方のレベルで議論する。リポジトリ自体がどのようなデータベースシステムと様々な機能を実現するための付加的なプログラムで実現されているかはここでは問わない。なお、以下の比較は次の資料に準拠して実施した。それぞれ、ECMA PCTE は 91 年 4月にECMA 標準として 採用された版に、ATIS は 90年 2月に ANSI X3H4 に Working Draft として 提出された版に、AD/Cycle RMは90 年にIBM System Journal に 発表された AD/Cycle 特集に それぞれ 基づいている。

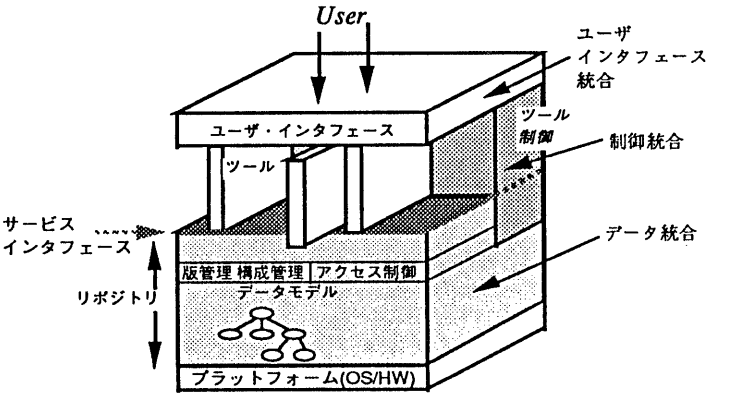

図１. CASE環境統合のアーキテクチャ

III. リポジトリに要求される条件・機能

ソフトウェア開発に関連する様々な情報を管理するリポジトリには、情報の格納と検索などのデータベース的機能に加えて、情報の版管理、並行作業におけるロック制御などのCASE環境特有の条件・機能が要求される。このような特徴を持つリポジトリの必要性が、各所で述べられている[9][10][11][13]。ここで挙げられている要求からリポジトリとして重要と思われる条件・機能を取り上げ、さらに幾つか条件を加えて、比較項目とする。ここでは、取り上げた比較項目を以下の４つに整理する。統合されるべきツールが

扱うデータに関する要求条件をツールデータ統合面に、ツールの起動に関する要求条件をツール制御統合面に、ツールの動作する環境に関する要求条件をツール環境面に、それ以外の条件をその他に、分類する。

A. ツールデータ統合面

ツールを開発環境に統合する場合、そのツールが利用する情報をリポジトリのデータモデルに容易に統合できる必要がある。そのためにデータモデルが満たすべき条件には以下のものがある。

1. 関係表現が容易であること ---[10][13]

設計情報には多種多様な関係があり、開発作業に携わる者はその関係を考慮して、情報の一貫性チェックや影響範囲の解析等の作業に役立てている。従って、設計情報だけなく設計情報間の多種多様な関係を表現する関係情報もリポジトリに格納する必要がある。

更に、関係の種類が多いため、単に関係情報が定義できるだけでは不十分である。例えば、汎化関係、複合構造、導出関係、定義参照関係など、関係を分類・整理した上で基本的な関係を利用しやすい形で提供することが必要である。

2. 一貫性の管理が可能であること ---[9][10][13]

リポジトリで管理する設計情報間には、一貫性がなければならない。しかし、編集操作によっては設計情報の一貫性を失うこともあり得る。設計情報の一貫性に関する制約は、設計情報のスキーマと同時に定義可能である。情報の一貫性は、リポジトリ枠組内で管理すべきであり、枠組外のツールに一貫性管理任せる必要はない。従って、情報を格納するリポジトリ自身が一貫性管理機能を持ち、スキーマ定義時に一貫性に関する制約を定義できる方式が必要である。

3. 拡張容易な基本スキーマがあること ---[9]

設計情報を、何の手がかりもないところから型階層や複合構造としてスキーマ化する作業は非常に手間がかかる。例えば、新たな文書を追加管理する場合など、既に定義された類似文書の型（スキーマ)が若干の変更で利用可能であれば、リポジトリ格納情報の拡張は容易となる。従ってリポジトリが拡張可能な基本スキーマを提供する必要がある。

また、別種の図の間で共通な情報をリポジトリの中で表現するためには、オブジェクトの粒度が図の構成要素単位に対応することが望ましい。つまり、データフロー図を、プロセスやデータフロー、といったオブジェクトで構成できる粒度が必要である。

4. メタオブジェクト（スキーマ）の編集が可能であること ---[10][13]

リポジトリのスキーマは、定義した後は絶対に変化しないものではなく、スキーマ自体が変更される可能性は十分にある。例として、リポジトリに新たな設計文書の管理を追加する場合や、既存の設計情報の構成やデータ項目を変更した場合などが考えられる。つまり、既存スキーマの編集や、新たなスキーマの作成や付加が可能でなければならない。これに対応するためには、リポジトリ自体が自己を記述したスキーマを管理し、そのスキーマがリポジトリ管理者によって編集可能であることが必要である。

B. ツール制御統合面

ツールを開発環境に統合する場合、ツールに必要な情報を共有するだけではなく、開発作業に即したツールの利用を制御する必要がある。そのために、リポジトリには以下の機能が必要である。

1. ツールの組込み

新たなツールをツール制御機能に容易に組み込める枠組が必要である。

2. 作業・工程管理 ---[10]

開発作業・開発プロセスの制御のためには、開発作業やプロセスに関する情報の格納と、その情報に基づいて適切な契機を捉えてツール起動、作業開始、ユーザへの通知などを行う開発プロセス制御機構が必要である。

C. ツール環境面

ツールが利用する情報は、リポジトリの適切な管理により常に誤りのない情報として利用できるべきである。そのためには、以下の情報管理機能が必要である。

1. アクセス権限 ---[10][11]

アクセス者が借っていに設計情報を変更できるのは問題である。そのため、情報の管理単位ごとに制御可能なアクセス権が必要である。また、アクセス者の分類として、ユーザ個人だけではなく、役割や所属グループごとに制御する柔軟性が求められる。

2. 排他制御 ---[9][13]

使用中のオブジェクトを他のアクセス者が変更できる場合、情報の一貫性の点で問題がある。そのため、オブジェクトが使用されているか否かによって他からのアクセスを排他制御する必要がある。ただし、ソフトウェア開発作業の場合、オブジェクトを占有する時間が長いために、単純なロックに基づく排他制御で他の作業が進まない可能性がある。このような長時間トランザクションに対応した柔軟な排他制御

が必要である。

3. 版管理 --[9][10][11][13]

ソフトウェアの設計情報は複数の版を持つ。開発するソフトウェアによっては、版管理の次元が複数必要な場合が存在する。例えば、対象マシン／対象ＯＳ／バグフィックスごとに版管理する場合が考えられる。従って、版管理の支援、できれば複数次元の版管理機能が必要である。

4. 構成管理 ---[10][13]

最終的に構築されるシステムは通常複数のモジュールから構成される。一部のモジュールを修正した場合、モジュール間に存在する複雑な依存関係や導出関係をたどってシステムを再構成する作業は困難を伴う。そのため、システムを構成するモジュールを管理し、要求に従ったシステム生成を支援する構成管理機能を提供する必要がある。

5. 分散化への対応 ---[10]

ソフトウェアの開発環境は、ホストと端末による集中型の形態から、ワークステーション（以下WSとする）を中心とした分散環境へと変化してきている。分散した環境でのリポジトリ利用が考慮されている必要がある。

D. その他

1. 製品化状況

各リポジトリ仕様は、リポジトリシステムとして実現が容易であることが望ましい。実現容易性の実証として、製品化が早いことが条件として挙げられる。

IV. 比較結果と今後の方向

ここでは、前章の要求条件・機能に基づいて各リポジトリ仕様を比較した結果を、表（表１）とチャート(図２)に示す。

まず各仕様で共通する点は、ツールデータ統合面ではスキーマ編集機能を各仕様とも提供している点、

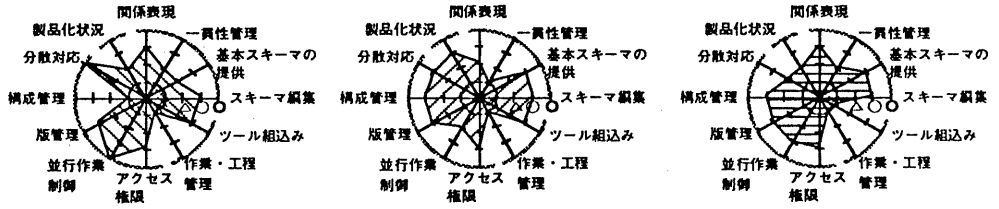

図２ 比較チャート

ツール制御統合面では作業・プロセス管理の枠組を各仕様とも提供していない点、ツール環境面ではアクセス権限と版管理機能が提供されている点、である。また、相違点として、一貫性管理機能の提供、開発プロセス制御、並行作業制御、分散環境への対応、構成管理、製品化状況がある。

以下にツールデータ統合面、ツール制御統合面、ツール環境面から、詳細に比較する。

A. ツールデータ統合面

ツールが扱うデータの統合が容易なデータモデルと、そのデータモデル上での一貫性管理についてを述べる。

1. データモデル

ツールが扱うデータを容易に統合するためには、設計情報、設計情報間の関係（汎化関係や、設計情報の構成関係を含む)、および設計情報に密接に関係する操作を、自然に表現可能なデータモデルが必要である。

ここで取り上げたリポジトリ仕様に用いられている基本的なデータモデルは２種類ある。それらはＥＲモデルとオブジェクトモデルである。

ＥＲモデルは、実体間の関係を全て対等に表現できる利点があるが、汎化関係と構成関係の記述を特別扱いできないこと、また実体に密接に関係する操作を記述できないことから、モデルの拡張が必要である。ＥＲモデルに基づくリポジトリ仕様は、PCTEとAD/Cycle RM である。PCTE ではＥＲモデルから派生したモデルを用いている。実体の実現値をオブジェクトとし、オブジェクトは全てなんらかの型に属する。型の間には汎化関係による型階層を導入している。汎化関係にある型の間では、属性のみの継承が導入されている。また、関連の代りにオブジェクトからオブジェクトへのリンクを導入している。あるリンクとその逆向きリンクとの組が関連に相当する。ただし、オブジェクトへの操作の一体化は導入していない

。次に、AD/Cycle RMでは、ＥＲモデルの拡張として集約(entity aggregation)とオブジェクトの概念を導入している。オブジェクトは全て型に属し、型の間には汎化関係による型階層がある。またオブジェクトにはそのオブジェクトに関する操作が一体化されており、汎化関係にある型の間でその操作が継承される。これらの概念の導入により、ＥＲモデルにおける汎化関係、構成関係、操作の一体化の表現、さらに型の間の継承をも獲得している。

一方、オブジェクトモデルは、型階層とその間の継承機構を持ち、オブジェクトに関係する操作の一体化が表現可能だが、それ以外の関係をうまく表現することができない。Relationship型オブジェクトを導入するか、関係を操作とみなしてメソッドで表現するなどで拡張する必要がある。オブジェクトモデルを持つATISは、型階層として提供している基本スキーマにRelationship型を組み込む拡張を行っている。

ここで示したように、各仕様とも設計情報および設計情報間の関係を自然にかつ十分に表現できるようにデータモデルを拡張している。この点では、どの仕様も目標とするところは同一であると言える。(表1-1.1)

このように、拡張された各モデルの記述力がほぼ同じと考えられるため、既存データモデルに新たなツールで扱うデータを統合する場合の容易さは、提供されている基本スキーマの質、スキーマ編集が可能な範囲、およびスキーマ編集の操作性に影響されると考えられる。

2. 一貫性管理

前項で述べたように、データモデル拡張の目標が同じであるため、基本とするモデルの差が一貫性管理の差異となって現れる。

ＥＲモデルに基づくモデルを持つPCTE,AD/Cycle RMは、関連のCardinality制御や従属性を一貫性管理手段として提供している。例として、Cardinalityの指定（１対１、１対Ｎ、Ｎ対Ｎ)、追加時の制約（関係元の新規挿入に伴って関係先の実体を生成する）、削除時の制約（関係元の削除に伴い関係先の実体も削除する）、順序集合（実現値を呼び出す順番の制御）が提供されている。これらの制約を指定した関連型の一貫性はシステムが保持するよう動作する。一方、オブジェクトモデルに関連を加えたATISでは、従属性などの意味的な制約を提供していない。そのため、リポジトリで格納情報の一貫性管理を実施する場合には、一貫性管理機構を型階層の中に作り込まなければならないATISは多少不利である、と言える。(表1-1.2)

今後は、一貫性管理のために提供されている条件が十分か評価することと、一貫性管理機能を利用しやすい形で提供することが課題である。

B. ツール制御統合面

ツールを開発環境に統合するレベルには２レベル存在する。第１レベルは、リポジトリに格納された設計情報を編集するツールとしてリポジトリから起動可能とするレベルである。第２レベルは、開発プロセスが管理されている状況において、定められたプロセスに沿って定められたツールを起動して設計情報を編集するレベルである。このレベルでは、開発プロセス自体の記述が編集および再利用の対象になる。

第１レベルでは、ツールを組み込む枠組が必要となる。AD/Cycle RMではツールを組み込むための枠組は持たず、ツールの組み込みはADプロセスモデルとワークマネジャで行われる。一方、PCTEとATISはデータモデル内にツールを組み込む枠組を持っている。PCTEは、組み込みたいツールをProcess型のオブジェクトとして定義する。プロセス起動手続きでそのオブジェクトを指定すると、そのオブジェクトが示すツールが起動される。PCTEにおけるProcess型のオブジェクトは、UNIXにおける１プロセスに相当するため、UNIX的なプロセス制御が可能である。一方、ATISの場合、組み込みたいツールをTool型またはMethod型として定義し、メソッドの場合は組込むツールが扱う設計情報のエレメント型にこのメソッドを付加する。Tool型またはメソッドの付加されたエレメント型にメッセージを送ることで、組み込んだツールが起動する。

ここで示したように、各リポジトリは、第１レベルのツール統合において、リポジトリに情報をおくか否かに関わらずツールの組込みと起動の枠組を用意している。ただし、ツール組み込みに関する各リポジトリの方向は、AD/Cycle RMはリポジトリの枠内では扱わない方向、PCTEはUNIXプロセスの枠組でツールを組み込む方向、ATISはオブジェクト指向的に、データとそれに付随する操作を一体化する方向、と分れている。（表1-2.1）

第２レベルでは、開発プロセスを制御するためのプロセスモデルと、プロセスモデルの記述に沿ってツールを制御する機構が必要となる。ここで取り上げたリポジトリには、リポジトリの枠内でプロセスに関する情報を管理しツールを制御するものはない。ただし、AD/Cycle全体の体系では、開発工程を記述するADプロセスモデルおよび生産物を記述するAD情報モデルによってソフトウェアプロセスの記述枠組を提供している。これは、開発作業を作業(activity)を単位として記述するもので、作業には前提条件(entry predicate)、完了状態(exit assertion)および作業内容（使用するツールなど）を記述する。開発プロセス全体は、AD情報モデルで定義される生産物を介して作業をネットワーク状に結合して表現される。作業を結合させることで、作業の順序や並列化可能性を定義できる。具体的なツールの起動はワークマネジャがプロセスモデルを見て行う。一方、ATISにはプロセス制御の枠組は用意されていないが、(1)プロセスを管

理するオブジェクトを陽に定義する、あるいは、(2)あるメソッド中に別のメソッドを起動するよう暗に記述しておく、という方法でプロセス制御が実現できる。この方法では、リポジトリ内のデータモデル上にプロセス制御の枠組が構築可能となり、プロセス自体の共有・再利用の可能性を持つことになる。

ツールを組み込むときは、ツール単体でなく方法論とともに組み込む必要がある。また、生産物と方法論が完全に独立に決るものではない。従って、今後ツールを統合する場合、ツールの支援する方法論を記述したプロセス記述も一緒に統合する方向に進むと予想されるが、プロセス記述自体がリポジトリで扱われるかどうかは議論が必要である。(表1-2.2)

C. ツール環境面

ツールの動作する環境として、リポジトリによる情報管理機能について比較する。

アクセス権限と構成管理は、必要な機能がほぼ各リポジトリで満たされている。PCTEが構成管理を提供していないが、実現した場合、他のリポジトリと構成管理機能に差がないと予想される。そのため、ここではアクセス権限と構成管理については割愛する。（表1-3.1,3.4)

1. 版管理

ソフトウェア開発で生成される情報には、数多くの版が存在する。それも版の種類が１種類のとは限らず、例えば対象機種と対象ＯＳごとに版管理するなど、複数次元で版管理する場合もあり得る。また、版管理の最小単位をダイアグラムとするかダイアグラムの構成要素とするかはプロジェクト毎に決める必要がある。

各リポジトリは版管理機能を持つが、複数次元の版管理機能を持つものはない。また、版管理の対象にはそれぞれ差異がある。PCTEの版管理対象は全てのオブジェクトおよび全てのリンクとなっている。AD/Cycle RMではオブジェクトのみが版管理の対象であり、版管理の粒度はオブジェクトの定義に依存する。関連やメソッドは版管理対象とはならない。ATISはVersion型のサブタイプに対して版管理が適用される。Version型のサブタイプには、ファイル、ツール、型 (Element-TYPE,Property-TYPE,Relationship-TYPE)、メソッドなどがある。例えば関連型(Relationship-TYPE)は版管理対象だが、個々の関連は版管理対象でない。（表1-3.2)

今後は複数次元で版管理を実施する機会が増えていくと予想される。そのため複数次元の版管理を提供する必要がある。また、版管理対象が柔軟に設定できる枠組を持つことが望ましい。今後は、メソッドも版管理の対象とするATIS的な柔軟な枠組を持つことになると予想される。

2. 排他制御

リポジトリの使用環境は、複数人の開発者がそれぞれ必要な情報を長時間占有し作業を行う環境である。情報の占有時間が長時間に渡るため、通常のトランザクションとは異なる柔軟な排他制御が必要である。

ここで取り上げた各リポジトリは、ロックに基づく排他制御を持っている。

PCTEは、軍用向けのPCTE+を基盤としているため、並行作業と一貫性の管理に関しては非常に厳密に定義している。オブジェクトやリンクに対する操作群(Read,Write,Deleteとprotected,unprotected,semi-protected,transactionedの組み合わせ)をモードとし、同じオブジェクトに２者がアクセスする時、双方のモード間の可否・優先度を定めている。ATISは、利用者の手を煩わせない方式を目指して版管理と排他制御を組み合わせた機能を提供している。これは、オブジェクト編集時には自動的にオブジェクトをロックし、書き込み終了後にロックを解除する機能である。ユーザが明示的に版を制御する必要がないのが利点だが、自分の欲しい版がどれかを判断することが出来なくなるという問題点がある。また、編集操作のたびにロックすると、他ユーザはそのオブジェクトにアクセス不可能になり、作業が滞る可能性がある。これを避けるため、ATISには、生産物の本来のバージョンから枝分かれしたバージョンを作る機能（ブランチ機能）も提供している。しかし、ブランチ版をマージする時に必要な変更箇所の衝突管理は実現していない。また、AD/Cycle RMでは編集対象とするオブジェクトを自動的にロックし、テストが終了した後承認を受けてロックを解除する機構を持つ。(表1-3.3)

それぞれリポジトリの使用環境に対する認識の違いから、それぞれ異なる排他制御機構を採用している。開発プロジェクトの機密度が高い場合には、PCTEレベルの排他制御が必要である。通常の開発環境ではATIS的な機能でも十分と思われるが、今後どのような排他制御が必要がをさらに検討する必要がある。

3.分散化への対応

今後のソフトウェア開発形態はホスト＋端末型からワークステーションを利用した分散開発環境が主流になる。リポジトリにとっても分散開発環境への対応は目標の一つである。

各リポジトリの分散開発環境への対応は、ホスト指向とWS指向に分類でき、AD/Cycle RM は前者、PCTEは後者に分類できる。ホスト指向であるAD/Cycle RMは、データはホストで集中管理し開発作業はWS側、という分散開発形態を想定している。つまり、ホスト上でデータを一元管理しておきWS側で必要なデータはコピーして利用する。WS側のツールからホスト上のリポジトリを直接アクセスすることは計画段階であり、まだ実現されていない。一方、PCTEは、仕様レベルでオブジェクトの分散配置が可能な

よう定義している。AD/Cycle RM、PCTEともWSによる分散開発環境を考慮に入れているが、各々のリポジトリの動作環境を中心とした分散環境を基本とするため、AD/Cycleはホスト中心の開発環境、PCTEならWS中心の環境と適用範囲は決まってしまう。なお、ATISには仕様レベルで分散開発環境に関する記述はない。（表1-3.5)

今後は、ソフト開発における協調作業支援の視点も含めて、ホスト間やWS間での情報の分散管理方法を検討する必要がある。

D.その他

1. 製品化状況

それぞれの製品化状況は以下の通りである。IBM AD/Cycle RM は仕様に基づく製品Repository Manager RM/MVS が出荷されている。CIS ATIS は Atherton Technology 社の Software Backplane™ に基づいており、現在もこの製品は ATIS の仕様に適合するよう改良されている、とのことである。ECMA PCTE は、今回準拠した版が最近発表になったばかりの仕様であるため、現在これを完全に実現した製品はないと思われる。ただし、別の版の PCTE を実現したものは存在する[12]。（表1-4.1）

今後は、ISO の標準化動向を睨みながら標準に準拠することでマルチベンダを標榜していくと思われる。

V. まとめ

本資料では、ツールの統合の点から各リポジトリ仕様の比較を行った。その結果、以下の点が明らかになった。

1. ツールデータ統合面では、データモデルの拡張の目標が、設計情報、関係情報、情報と関係の一体化という点で各仕様とも一致していること、基本としているデータモデルによって一貫性管理手段の提供に差があること。
2. ツール制御統合面では、リポジトリで持つツール起動の枠組は各々のリポジトリ毎に異なり同じ枠組に収束しないと予想されること、開発プロセスの記述枠組はどのリポジトリにも持っていないこと。
3. ツール環境面では、版管理の対象とその粒度に差があること、複数人による開発環境に適した排他制御が明らかでないこと、分散開発環境の捉え方に違いがあること。

また、今後の課題として以下の点が挙げられる。

1. 一貫性管理のための機能の検討と提供。
2. リポジトリにおける開発プロセスの記述と管理。
3. 版管理機能の拡充、分散開発環境に適した排他制御方式および情報分散方式の検討。

謝辞

ATIS に関して資料提供や質問に応じていただいた日本ＤＥＣの方々に心から感謝致します。また、本稿の作成に関してご討論を頂きましたＮＴＴソフトウェア研究所の山本修一郎主幹員に深謝致します。

参考文献

[1] "Portable Common Tool Environment (PCTE) Abstract Specification",ECMA Standard-149, December 1990
[2] "Information Resource Dictionary System ATIS", ANSI X3H4 Working Draft, Febraury 1990
[3] IBM Systems Journal, Vol.29, No.2, 1990
[4] "Information Resource Dictionary System (IRDS) Service Interface", ISO/IEC CD, June 1990
[5] 川越, 他: "オブジェクト指向データベースと CASE", 情報処理, Vol.32 No.5, 577-584, 1991
[6] 松本: "CASE 環境の基礎技術", 情報処理, Vol.31 No.8, 1020-1027, 1990
[7] 松本: "CASE 環境統合のためのインタフェースの標準化現状", 情報処理, Vol.31 No.8, 1086-1094, 1990
[8] Thomas, I., "PCTE Interfaces: Supporting Tools in Software-Engineering Environments", IEEE Software, November 1989
[9] Bernstein, P.A., "Database System Support for Software Engineering -- An Extended Abstract", 9th ICSE, 166-178, 1987
[10] 片山: "ソフトウェア開発のための オブジェクトベース", 情処学会研究会資料 79-DBS-8-5, 1990
[11] 徳田: "次世代データベース宣言に向けて -- ソフトウェアオブジェクトベース構築の 立場から", 情処学会研究会資料 79-DBS-8-3, 1990
[12] PCTE Newsletter, No. 6, April 1991
[13] 篠田: "ソフトウェアデータベースの技術動向", 電気・情報関連学会連合大会, 1989

		PCTE	ATIS	AD/Cycle RM	基準
1 データ統合	*データモデル概要*	ERモデルより派生 --> object/link/attribute + 型階層	オブジェクトモデル + Entity型・Relationship型	拡張ERモデル + 集約・オブジェクト	
	1.1 関係表現の容易さ	○ 数種のリンク型（関係表現）が用意。 汎化関係・複合構造は、型階層・複合オブジェクトで表現。	△ primitive な関係表現枠組(Relationship 型)のみ用意。 汎化関係・複合構造は、型階層・複合オブジェクトで表現。	○ ERモデルで記述。関係の型が用意。 汎化関係・複合構造は、型階層・オブジェクトで表現。	◎非常に容易 ○容易 △困難 ×不可
	1.2 一貫性管理の提供	△ 制約条件を持つリンク型を用意。原始的な一貫性維持は可能。	× ユーザが、Relationship型に一貫性管理機構を埋め込むかまたはobjectのメソッドに動作として定義する	△ 意味制約 (cardinality, mandatory, controlling, ordered set) および Policy (Integrity, Trigger)により原始的な一貫性管理は可能	◎豊富に提供 ○提供 △ユーザ作成 ×困難
	1.3 基本スキーマの提供	△ ファイルレベルの粒度のみ	○ オブジェクトの型階層として提供	○ AD情報モデルで個別に提供	◎豊富なスキーマ ○十分なスキーマ △不十分 ×未提供
	1.4 メタオブジェクト（スキーマ）の編集	○ 自己記述により、インスタンスアクセスと同じ枠組で編集可能	○ 自己記述により、インスタンスアクセスと同じ枠組で編集可能	○ AD情報モデルは拡張可能	◎容易に可 ○可 △困難 ×不可
2 制御統合	2.1 ツールの組込み	○ ツールはプロセスオブジェクトとして定義され、プロセス起動操作により起動する。	○ ツールはメソッド型またはとして定義。オブジェクトへ起動メッセージを送信して起動。	×(○) ツールをobjectとして組込む枠組はない。作業管理機能（リポジトリの機能ではない）がツールを起動する。	◎容易に可 ○可 △困難 ×不可
	2.2 作業・工程管理	× なし	× なし	×(○) AD プロセスモデルに作業を活動(activity) とそのネットワーク(Work Flow Structure)を記述。	◎容易に可 ○可 △困難 ×不可
3 ツール環境	3.1 アクセス権限	○ ユーザ／ユーザグループ／ツールごとに定義。	○ ユーザ／役割／ユーザグループごとに定義。起動するメソッドを変えることも可。	○ ユーザに設定したレベルに基づき、実体／属性／関係へのアクセスを制御。	◎容易に可 ○可 △困難 ×不可
	3.2 並行作業制御	◎ 2者間の操作モードの優劣関係を明確に定義したロック 機能	△ 版管理の一機能 (CHECKOUT/CHECKINに伴うロック)	○ 排他的ロックと非排他的ロックを持つ。集約を構成する実体の実現値を対象。	◎最密に定まる ○あり △代用 ×不可
	3.3 版管理	○ 全てのオブジェクトに対して、版管理が可能。	○ 版管理は、Version 型の subtypeとして定義したオブジェクトに対して可能。	○ オブジェクトに対して版管理が可能。(版管理したい情報はオブジェクトで表現)	◎容易に可 ○可 △困難 ×不可
	3.4 構成管理	× 構成管理機能はなし。	○ 構成管理は、Collection 型のオブジェクトを管理単位として実現。	○ 構成管理はライブラリ サービスで提供	◎容易に可 ○可 △困難 ×不可
	3.5 分散環境への対応	◎ オブジェクト単位で分散可能。 WS 上の repository	○ 物理的なデータが分散可。 WS 上の repository。	△ ホスト上で一元管理。WSへデータをコピー可。 ホスト上の repository。	◎十分な対応 ○対応 △不十分 ×対応不可
4 その他	4.1 製品化状況	△ 仕様先行	○ 製品ベース	△ 仕様を完全に実現した製品は未販売	◎完全に提供 ○提供 △仕様先行 ×なし

表１ リポジトリの要求条件充足度比較表