データベース・システム　76－4
（１９９０．　３．　１２）

統合ＣＡＳＥ実現のためのデータ層の統合について

岩田誠司　　岡安二郎　　松村一夫
（株）東芝　システム・ソフトウェア技術研究所

本論文では、既存のＣＡＳＥツールを結合してソフトウェアライフサイクルの複数工程を支援する統合ＣＡＳＥを実現していくためにまず議論の枠組みとして７階層（システム運用層、ユーザインタフェース層、ツール機能層、ツール環境層、データ層、ＯＳ層、物理層）のインタフェースモデルを提案し統合利用の指針を与える。そしてその中でも特に重要であると考えているデータ層の統合方法について述べる。データ層の統合に関しては意味レベルのデータを表現形式、概念、実体の３種類に分割しそれぞれに関係を定義して管理する方法を提案する。

AN INTEGRATION OF THE DATA LAYER TO DEVELOP AN INTEGRATED CASE

Seiji Iwata　Jiro Okayasu　Kazuo Matsumura
TOSHIBA CORPORATION, SYSTEM & SOFTWARE ENGINEERING LABORATORY
70, Yanagi-cho, Saiwaiku, Kawasaki, Kanagawa, 210 Japan

Recently, the improvement on the software productivity, reliability and maintainability has become a main problem of software engineering. The various methodologies are investigated to cope with the above problem and supported by the computer. These systems are called CASE(Computer Aided Software Engineering) tool. But at present time, many CASE tools support only one phase of the software life-cycle.

In this paper, we propose an interface model to discuss the integration of CASE tools. This model is consisted of 7 layers and we especially discuss about the integration of the data layer.

1．はじめに

ソフトウェア開発において、生産性・信頼性・保守容易性の向上は、ソフトウェア技術者の絶対的な不足といったソフトウェア危機の認識により最重要課題となっている。このような問題を解決するために様々な要求分析手法、設計手法、言語、テスト・管理手法が研究開発されている。そしてこれらの手法、言語を計算機上で支援するツールが数多く発表されている。これらのツールは近年“ＣＡＳＥ”という用語に結び付けられ世の中の注目を集めている[1-3]。

しかし現在のところソフトウェアライフサイクルの単一工程のみを支援するツール（単体ＣＡＳＥ）が主流である。従って、複数工程を支援するようなツールの開発あるいは既存ツールを結合して複数工程を支援する（統合ＣＡＳＥ）といった研究が行われてきている。

本論文では個々のＣＡＳＥツールを結合して統合ＣＡＳＥを実現していくための議論の枠組みとして７階層の統合インタフェースモデル[4]を提案し、その中でも特に重要であると考えているデータ層の統合について述べる。

2．ＣＡＳＥについて

2．1　ＣＡＳＥの定義

まず本論文におけるＣＡＳＥの定義を述べる。本論文ではＣＡＳＥは“Computer Aided Software Engineering”の略語であるとし次のように定義する。

> ＣＡＳＥ＝ソフトウェア工学の手法、方法論を開発ツールに利用したもの

またＣＡＳＥの定義に関しては、狭義の解釈と広義の解釈がある。まず狭義にはＣＡＳＥを“ソフトウェア工学における設計手法、管理手法等を計算機を用いて実現・支援するツール”と解釈できる。狭義の解釈ではあくまでも、ソフトウェア工学上の手法・方法論を基本としていることがＣＡＳＥツールの条件である。

一方、広義には、“計算機を用いてソフトウェアの設計・管理等を支援するツール”と解釈できる。つまり、構造化分析・構造化設計ツールだけでなく、言語指向環境、構造指向エディタ、ツールキットなどをも含めてＣＡＳＥツールと考えている。すなわち広義には、コンピュータ支援ソフトウェア開発環境を指している。本論文ではＣＡＳＥツールを広義の意味で定義する。

次にＣＡＳＥツールが誕生し注目された背景を考えると以下のことが挙げられる。

・ソフトウェア工学の方法論が充実・普及
・パーソナルコンピュータの普及
・グラフィック機能の向上
・既存ツールへの不満
・ライフサイクル支援の重要性の認識
・新しい概念で既存技術の再構成が行われた

また、その発展を段階的に追っていくと以下のようになる。

・第１段階（理論段階）
ソフトウェア工学において、各種の分析・設計手法が研究された。
・第２段階（理論のツール化）
ソフトウェア工学の手法、方法論が計算機を用いることにより機械化された。
・第３段階（単体ＣＡＳＥ）
ソフトウェアライフサイクルを考慮し、データ管理を行い、計算機を中心に開発環境を見直した。但しこの段階では、ソフトウェアライフサイクルの特定フェーズのみを支援しているため単体ＣＡＳＥツールと位置づける。
・第４段階（統合ＣＡＳＥ）
単体ＣＡＳＥツールの問題点である単一工程のみを支援しているという点を解決し、複数工程に渡って開発を支援する統合ＣＡＳＥが議論され始めた。

ここで、単体ＣＡＳＥ、統合ＣＡＳＥという用語をソフトウェアライフサイクル上の支援範

囲によって以下のように定義する。

単体ＣＡＳＥ・・・単一工程のみを支援
統合ＣＡＳＥ・・・複数工程を支援

現在は単体ＣＡＳＥツール（第３段階）から統合ＣＡＳＥ環境（第４段階）に移っている状態であり、如何にツールを結合して有効に利用していくかに関心が集まっている。

２．２　ＣＡＳＥの目的

現在ＣＡＳＥに要求されている支援項目、あるいは実現されている項目について述べる。
まず、現段階でＣＡＳＥツールが実現している支援項目としては、

1)手法、方法論の機械化支援
2)ドキュメント作成支援
3)データの管理支援

の３点が挙げられる。手法、方法論を機械化し（グラフィックエディタ等を用いて支援）、ドキュメントの作成を支援する事で生産性・信頼性はかなり向上した。そしてデータを管理する事によりドキュメント作成支援やツール間でのデータ変換、あるいは作業の自動化が行われてきている。またデータを管理する事により再利用や保守においても非常に大きな効果をあげ始めている。しかしこの３点だけでは単なるエディタに過ぎず“ＣＡＳＥ”が本当に支援すべき項目として我々は以下の項目を挙げたい。

1)全工程の一貫支援
　複数工程を支援するような断片的な支援だけでなくソフトウェアライフサイクルの全工程に渡る支援を行う
2)全ての情報の再利用
　設計情報、データ、ノウハウ等を全て再利用単位とし管理・運用する
3)分散開発（協調共同作業）の支援
　ネットワーク環境、勤務形態に対応して個人の支援だけでなくプロジェクト全体の管理を行う
4)知的支援
　手法、方法論のナビゲートや計画立案、要求獲得、発想支援等のＡＩ技術を利用した上流工程の前段階の支援

これらの項目はそれぞれの研究分野で独立に行われてきたが、その成果を新たな枠組みでまとめたのが“ＣＡＳＥ”であると我々は考えている。

３．ＣＡＳＥツール統合利用の観点

現在、多くのＣＡＳＥツールは単体ＣＡＳＥツールとして提供されている。しかし、単体ＣＡＳＥツールでは以下のような問題があげられる。

・ＣＡＳＥツール間で成果物の流用が困難（開発工程を断片的にのみ支援）。
・ＣＡＳＥツール間でデータ変換がされないためツール間で開発工程（入力作業）の重複がある。
・ツール間でユーザインタフェースが異なる
・個々の単体ＣＡＳＥツールが主機能（EX. 構造分析）の他に副機能（EX. 版管理等）を持っている為、他ツール（EX. 版管理システム）との併用が難しい

従って近年、これらの単体ＣＡＳＥツールを連結して隣接するフェーズ（要求定義と基本設計、詳細設計とコーディング等）を支援する統合ＣＡＳＥの開発が行なわれている。しかし、このような統合ＣＡＳＥでも上流工程の成果物である仕様・設計情報から下流工程のプログラム情報への変換は十分ではなくかなりの部分が人間系で行われている。従って要求分析から保守までのソフトウェアライフサイクル全体に渡って支援する環境で完成されている例は少ない。

このような単体ＣＡＳＥツールを組み合わせて統合ＣＡＳＥを構築していくには、次のような各種の標準化が課題となる。

・運用法の標準化
　ドキュメントの規約・流用方法といった個々

のツール間での運用法を定義
・ユーザインタフェースの標準化
単体ＣＡＳＥツール上に統一されたインタフェースを与える操作的な結合
・ツール構築環境の標準化
共通の開発環境を利用して個々のＣＡＳＥツールを構築するようなプラットフォームの結合
・データの標準化
個々のＣＡＳＥツールが扱うデータを共通のデータ構造で管理し、データ間の関係を定義する意味データレベルでの結合
・ＯＳ，マシンの標準化
ＯＳ、マシン自体の統一による結合

またソフトウェアライフサイクルに注目すると、
・同一工程を支援するＣＡＳＥツールの統合
表現形式、手法間の変換
・異なる工程を支援するＣＡＳＥツールの統合
成果物の流用

といった２種類の統合利用が考えられる。

４．ＣＡＳＥ統合インタフェースモデル

４．１　モデルについて

３で述べたような統合の観点を整理し、ＣＡＳＥツールの統合化を議論する枠組みとして７階層のモデルを提案する（図１）。

このモデルは単体ＣＡＳＥツールを結合していく場合どの様な結合方法があるか、あるいはどの様な段階があるかを考えて作成した。

システム運用層
ユーザインタフェース層
ツール機能層
ツール環境層
データ層
ＯＳ層
物理層

図１　ＣＡＳＥ統合インタフェースモデル

各層は次のような意味を持つ。
・システム運用層
人間、組織によって個々のＣＡＳＥツールを利用・管理する層。会社や部門間で異なるドキュメントの形式や利用方法等について規約を決め運用したり、成果物を介したツールの結合を行なう。
・ユーザインタフェース層
個々のＣＡＳＥツールとユーザとのインタフェースとなる層。操作的な結合を行なう。
・ツール機能層
個々のＣＡＳＥツールのプログラム本体がある層。個々のＣＡＳＥツールの機能を実現している。
・ツール環境層
個々のＣＡＳＥツールに対し、ウィンドウシステム、ツールキット等を提供している層。ＣＡＳＥツールを開発していく際に基礎となる環境である。
・データ層
データベース管理機能を提供する層。個々のＣＡＳＥツールで扱われているデータをアプリケーションレベルで管理している。
・ＯＳ層
計算機のＯＳレベルで提供されている層。特にファイル管理機能、あるいはテキストコード、バイナリコードといったデータレベルを扱う。
・物理層
ＣＰＵ，メモリ、通信制御等の機能を提供する層。

またデータに着目すると、データ層というのはデータの意味レベルを管理しており、実際の実現レベル（ファイル、ディレクトリといったデータ形式やテキストコード、バイナリコードコード体系）はＯＳ層、物理層で管理されている。

４．２　統合利用の指針

次にこのモデルに基づき、統合利用のポイン

トとなる各層における結合の指針を示す。ＣＡＳＥツールの結合は７つの層でそれぞれ行なわなければならない。

・システム運用層では、組織、作業環境等の違いやツール間の結合（成果物の利用）を人間系および計算機系での運用を定め、ＣＡＳＥツール群を利用していく。この層では方法論・規定の設定や、ツールの運用管理サービスが特に重要となる。

・ユーザインタフェース層では、個々のＣＡＳＥツール毎に異なるユーザインタフェースに対し統一されたユーザインタフェースを提供し、操作方法や画面表示等のユーザと接する部分を共通化する。

・ツール機能層では、サブツールの組み込み（エディタにバージョン管理ツールを組み込む）といった機能的な結合を行なう。各ツールの支援範囲・機能が明確に分離されなければならない。

・ツール環境層では、共通のツールキット、あるいは基礎環境といったプラットフォームを与える。そしてそれらを利用し新規ツールの開発を行う。

・データ層では、ＳＥＤＢ（Software Engineering Data Base）といったデータベースにより個々のＣＡＳＥツールの概念レベルのデータを管理し、新規あるいは既存のＣＡＳＥツールの統合をデータによって行なう。

・ＯＳ層では、データの実現レベルでＯＳが提供している機能を利用してデータ変換を行ない結合する。あるいは同一のＯＳ上にツールを構築する事によりＯＳ自体を統一する。

・物理層では、ネットワーク機能といった最もプリミティブな機能を利用し異機種間のＣＡＳＥツールの結合を行なう。

そして理想的なツールの結合としては図２に示すようにツール機能層を除く６つの層では標準化が行われ、個々のツールの機能は明確に分離されているというものである。

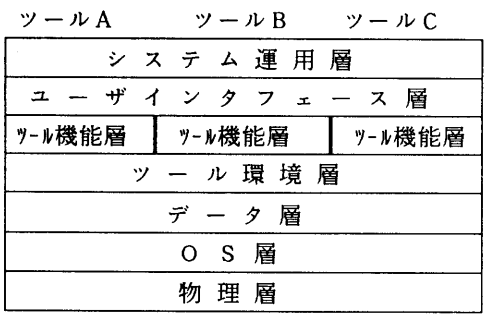

図２　理想的ツール結合

５．統合ＣＡＳＥと一貫ＣＡＳＥ環境

統合ＣＡＳＥはソフトウェアライフサイクルの複数フェーズを支援する事を目的としているが、最終目的としては全フェーズを支援する事が望まれる。このようなソフトウェアライフサイクルの全フェーズ支援を対象とし、運用法、ユーザインタフェース、データが一貫している環境を一貫ＣＡＳＥ環境として位置づける。また一貫ＣＡＳＥ環境は2.2で述べたようなＣＡＳＥか支援すべき項目（全工程の一貫支援、全ての情報の再利用、分散開発の支援、知的支援）を実現しなければならない。従って我々は統合ＣＡＳＥを実現していく上で、この最終目標である一貫ＣＡＳＥ環境の実現を考慮していくべきであると考える。本章では一貫ＣＡＳＥ環境を構築していく上で考慮していくべき点、あるいは一貫ＣＡＳＥ環境が実現すべき支援機能について述べる。

一貫ＣＡＳＥ環境を考える場合、汎用の能力を持った一貫ＣＡＳＥ環境を実現するのは不可能である。何故なら各適用分野で使用される手法・方法論は異なっており、たとえ同一の分野でも組織・部署によっても異なる手法・方法論が使用され独自のライフサイクルに沿って開発が行われている。従って、既存ツール（既存方法論）を集めてただ単に利用するだけでは一貫ＣＡＳＥ環境は実現されない。

我々は、一貫ＣＡＳＥ環境を実現するための枠組みを提供し、各応用分野、組織、部署に応じてカスタマイズしていかなければならないと考える。そしてまず一貫ＣＡＳＥ環境を実現していく為には以下の点を考慮しなければならない。

(1)開発工程を定義

これから一貫ＣＡＳＥ環境を実現しようとする組織でどのような手順でソフトウェア開発が行われているかを明確にする。

(2)手法・方法論の確立

(1)で定義した各開発工程で現在どのような手法が用いられているのか、あるいはどのような手法が適するのかを考え、その組織（部署）に応じた方法論を確立する。

そして上記の２項目を明確にした後、それに適した既存ツールがあればそれを選択し必要に応じてカスタマイズを行い、無い場合は新規に開発していくことになる。しかし、たとえ適するツールがあった場合でも運用法等の細かな点は独自に決めて行かなければならない。つまりあくまでもツールは単なる道具であり、その道具を有効に利用できるか否かは厳密な方法論が定義されているかどうかにより決定される。

次にこのような個々のツールをまとめる環境が提供していかなければならない機能としては以下の項目が挙げられる。

(1)一貫ＳＥＤＢの実現

ソフトウェア開発において生成・利用される情報の標準モデルを定義し、全情報を計算機化し再利用単位として管理する。また再利用のためのメカニズムも提供する。

(2)洗練されたユーザインタフェースの実現

個々のツールで異なるユーザインタフェースを統一するための標準形式を定義し、アイコニックインタフェース等の洗練されたインタフェースを提供する。

(3)分散グループコンピューティングの実現

分散開発において個人の自由度を尊重し、意志決定等の協調共同作業を支援する環境を提供する。

(4)管理機能の実現

品質・進捗管理の視覚化、分散開発の管理、データ自動収集による管理等を提供する。

(5)教育体制の確立

確立された手法・方法論を教育し普及していくためのシステムを確立する。

最後にこのような一貫ＣＡＳＥ環境の構成を示す（図３）。

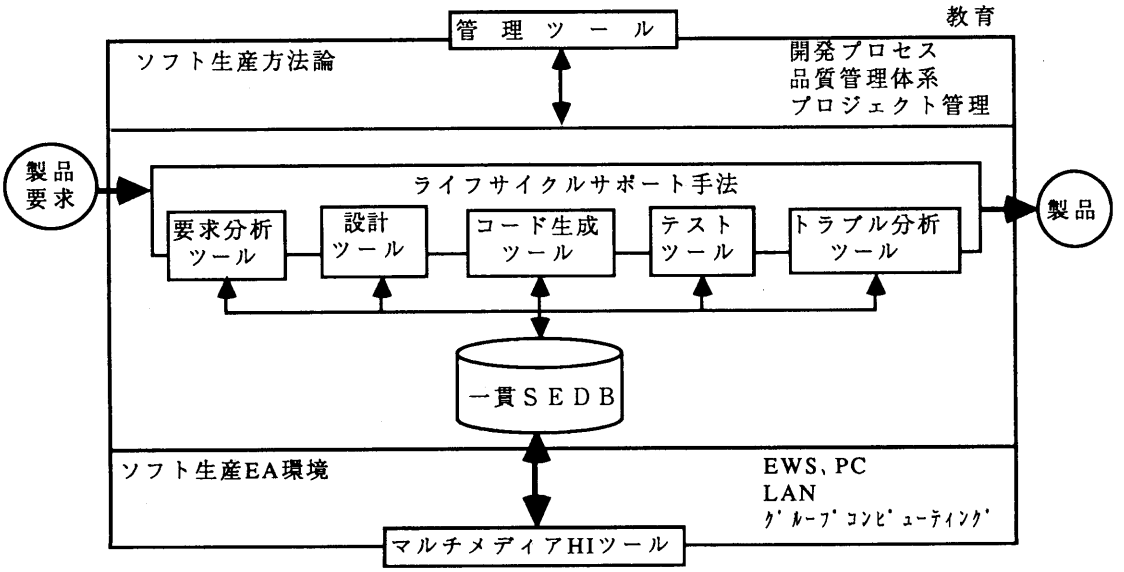

図３　一貫ＣＡＳＥ環境の構成

６．データ層の統合

次に一貫ＣＡＳＥ環境を実現して行くために、データ層をどのように統合するか（一貫ＳＥＤＢの実現方針）について述べる。データ層の統合は統合ＣＡＳＥあるいは一貫ＣＡＳＥ環境の実現のための重要な要素である。

６．１　データ層における統合の定義

我々は以下の機能を実現するための枠組みとメカニズムが提供されている場合に“データ層が統合されている”と定義する。

(1)情報の変換
(2)情報のトレース

まず情報の変換は異なる工程間と同一工程内の２種類に分けられる。異なる工程間での情報の変換とは要求分析情報を設計段階で、あるいは設計情報をコーディング段階で利用するといった事を意味する。例えば要求分析段階の結果である必要とされる機能、処理（データ）の流れ、データ辞書等の情報を設計段階のインプットとして利用したり、設計段階の結果であるタスク分割、モジュール分割等の情報をコーディング段階のインプットとして利用するといった事が挙げられる。

同一工程内の情報の変換とは手法間の表現形式の変換であると言える。例えば要求分析の為に各種の図法があるが、根本的に記述したい内容は共通点が多く表現形式が少し違うだけという場合が多い。ある図法ではプロセスを丸で表し他の図法では四角で表すといった場合である。このような場合、表現形式の変換（プロセスを丸で表す－＞四角で表す）を行う事により手法間での情報のやり取りが可能となる。

次に情報のトレースとは、修正の影響範囲を上流工程から下流工程、あるいは下流工程から上流工程の双方向で辿り通知することが出来る事を意味する。例えばある要求機能が変更された場合、どのタスク（モジュール）を修正すれば良いか、あるいはタスク分割を行った場合に要求分析におけるどの要求機能が分割されたのかを知らせられる事である。

但し、データ層ではあくまでもデータの意味レベルを対象とし、実現レベル（ＯＳ層、物理層）におけるデータの統合とは区別する。このようなデータ層の統合においては個々の情報の管理方法と、情報間の関係づけが重要である。

６．２　統合のためのデータ管理方法

現状のＣＡＳＥツールの多くは個々の情報の実体から表示情報までを一括して管理しているツールが多く、情報間の依存関係も図形的な接続関係や同一のデータフロー図内といった局所的な関係が多い。このような場合、異なる工程間あるいは同一工程内において情報を変換する事は困難である。また情報の再利用という観点からも扱う情報の単位が大きく困難である。従って我々はＣＡＳＥツールで扱うべきデータを分類し管理する事を提案する。

まずデータ層の統合を進めるために意味レベルのデータを実体、概念、表現形式の３種類に大きく分割する。そしてこれらの間を結び付ける物として次に示す５種類の関係を定義する。

- 実体間の関係
- 実体と概念間の関係
- 概念間の関係
- 概念と表現形式との関係
- 表現形式間の関係

これを図に示すと図４のようになる。

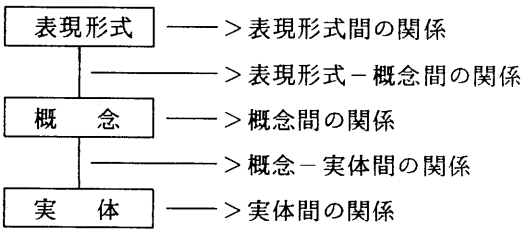

図４　意味レベルにおけるデータ分割

例えばデータフロー図において円で表されて

いるプロセス"Proc_1"を例に考えると、

実体：テキストとしての"Proc_1"

概念：プロセス、

表現形式：シンボルとしての円、

そして関係として、

実体と概念間：Proc_1はプロセスである、

概念と表現形式間：プロセスは円で表される、

という事が管理される。

また概念間の関係として、“プロセスは設計段階において複数のモジュールで構成される”といった事や、“プロセスには補足情報としてプロセス記述書が属性としてある”といった事が管理されている。実体間の関係としてはこれらの概念間の関係に基づく実体間の結びつきが定義される。そして表現形式間の関係としてはデータフロー図の各バブル間の座標関係等が管理される。

つまり使用するマシンや手法によって違いがでてくる表現形式と、マシンや手法に依存しない実際のプロセスの名前やデータ名等のテキスト等の実体を分離しその間を概念で橋渡ししている。これにより、例えば異なる設計手法（図法）間でデータを変換しようとした場合、概念と表現形式との関係に基づいて変更すれば良い。また概念間の関係に基づき工程間の情報の変換が実現がされる。

ここで概念間の関係は5章で述べたような個々の組織、部署で定義される方法論に基づいて定義されなければならない。つまり方法論で必要とされる情報を列挙し、それらを体系的にまとめていく事が要求される。そこで我々はこのような情報（概念）の列挙と関係づけを行う時の基本とするために、図5のようなＳＥＤＢ体系モデルを現在考えている。

このモデルは開発工程の各段階で要求される（決めなければならない）事をサブモデルとして定義し個々のサブモデル内における項目を列挙している。例えば要求分析、システム設計段階では機能モデルとかヒューマンインタフェースモデル等を作成していかなければならない。

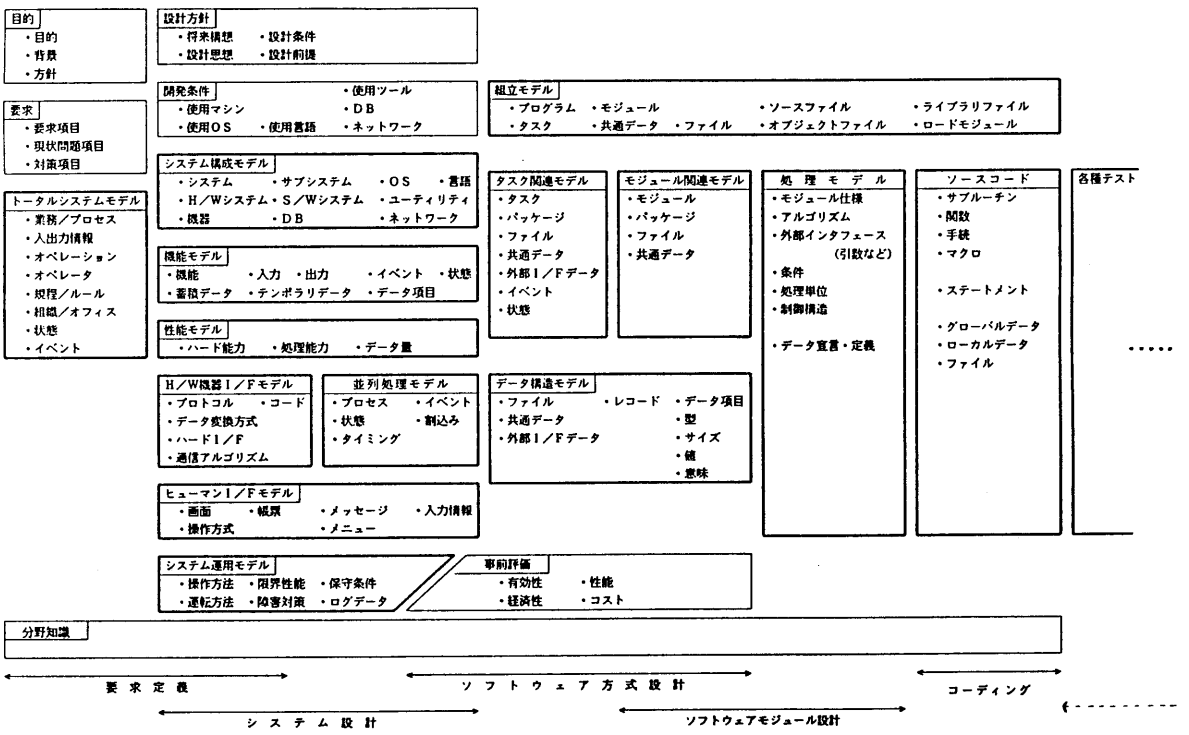

図5　ＳＥＤＢ体系モデル

但しこの図は最も基本的なレベルを定義しており５章で述べたような各組織・部署に応じた方法論に基づいて詳細化していかなければならない。そして各方法論に応じて詳細化された体系モデル（概念を記述）に基づき構成要素間の関係（概念間の関係）を定義していくことになる。

６．３　統合の形態

一般にデータを介してツールを結合しようとした場合次の様な形態が考えられる（図６）。

(1)個々のツール間で直接データを変換し結合する。

(2)個々のツール間で中間形式のファイルを定義し、それを介して結合する。

(3)データベースシステムでデータを一括管理し個々のツールを結合する

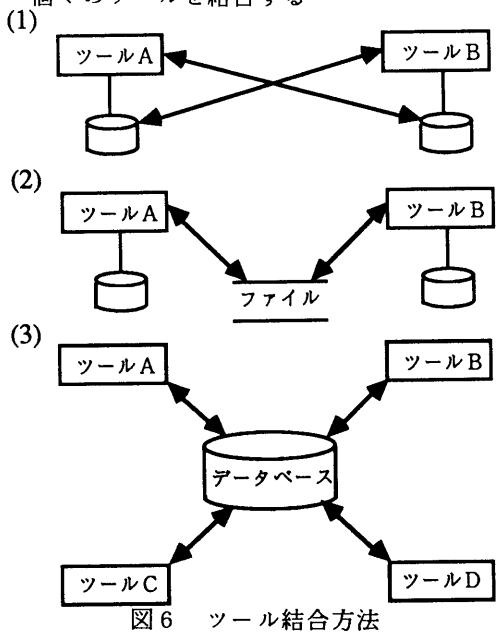

図６　ツール結合方法

(1)の場合、個々のツールは相手のツールのデータ表現に従い直接データの変換を行う。従って自ツール内、他ツール間等で複数のデータ表現が混在し、統合ＣＡＳＥを実現していく時に情報の変換やトレースを行う事が困難である。

これに対し(2)ではツール間で中間形式のファイルを定義しツール間のデータ表現を統一しているため情報の変換、トレースは行い易い。

また(3)の様に、(2)の考えを拡張し、全てのツールで利用できるような標準形式を定義してツールの情報を一括してデータベースシステムで管理する方法もある。これは新規にツールを全て作成していく際には有効であるが既存ツールを統合する場合には問題がある。

既存ツールを結合する場合には個々のツールが独自の形式でデータを管理しており、個々のツールで管理する情報と一貫ＳＥＤＢで管理する情報に分ける必要がある。つまり個々のツールで生成される情報を全て一貫ＳＥＤＢで管理する必要はない。例えば既存ツールを統合利用していく場合には、個々のツールで生成される情報の中でどの情報を利用してどの情報は利用しないといった情報の選択が行われる。つまり５章で述べたような組織に応じた開発工程、方法論で必要とされる情報（データ）のみを一貫ＳＥＤＢで管理すれば良い。これは図５に示した体系モデルを基本とし必要に応じて詳細化し概念間の関係を定義していく事により一貫ＳＥＤＢで管理すべき情報が明確になる。つまり我々は一貫ＳＥＤＢの構成を図７のように考え、個々のツールと一貫ＳＥＤＢそしてその間を橋渡しするプラットフォームを明確に意識しそれぞれの役割を明確に分離する。

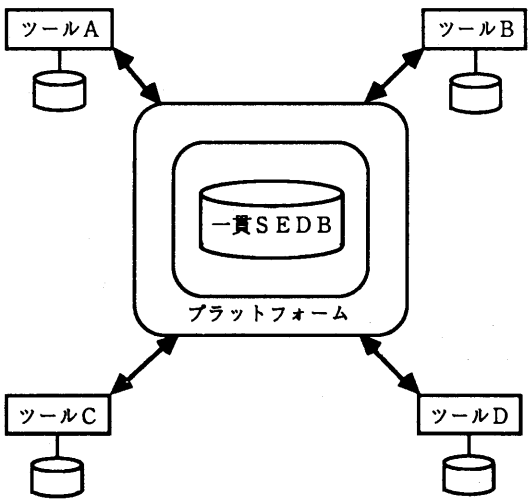

図７　一貫ＳＥＤＢの構成

6.2で提案した管理方法に基づき個々のツールが管理する情報、一貫ＳＥＤＢで管理する情報、個々のツールと一貫ＳＥＤＢとのプラットフォームが管理すべき情報を以下のように分割し管理する。

・個々のツールで管理する情報
　個々のツールが独自に管理している実体、
　表現形式
・プラットフォームで管理する情報
　実体と概念の関係、
　概念と表現形式の関係
・一貫ＳＥＤＢで管理する情報
　概念、
　方法論で必要とされる実体、
　概念間の関係、
　実体間の関係

７．おわりに

本論文では、既存のＣＡＳＥツールを結合してソフトウェアライフサイクルの複数工程を支援する統合ＣＡＳＥを実現していくための議論の枠組みとして７階層のインタフェースモデルを提案し統合利用の指針を与えた。このような統合ＣＡＳＥの実現に関しては、最終目的である一貫ＣＡＳＥ環境の実現方針を示し、その枠組みの中で特にデータ層の統合方法に焦点を当てて議論した。ＣＡＳＥツールの統合利用に際しては基礎となる方法論や一貫ＳＥＤＢで管理すべき情報をモデル化する事が重要である。

今後は提案した方法に基づき実際の既存ツールと我々が研究・開発を進めているソフトウェア生産工業化システムＩＭＡＰ(Integrated Software Management and Production Support System)間でデータ層の統合実験を行う。

参考文献

[1]E.J.Chikofsky(eds)."Computer-Aided Software Engineering(CASE)", IEEE Computer Society Press Technology Series

[2]A.S.Fisher,"CASE: Using Software Development Tools", John Wiley & Sons, 1988.

[3]C.McClure,"CASE is Software Automation", PRENTICE HALL, 1989.

[4]岩田他、“ＣＡＳＥツールの統合インタフェースモデルの一考察”、情報処理学会第３９回（平成元年後期）全国大会、１Ｒ－４