社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－SE－149（9）
2005／7／29

統合CASE環境の開発を支援するプログラム部品

後町 剛†

ソフトウェア開発組織が，効果的で，柔軟性や機敏性に優れたCASE環境を実現するには，CASEツールを低コストかつ短期間に開発･改良することのできるメタCASEツールの活用が有効である．本稿では，統合CASE環境の構築を全面的に支援し，スクリプトにより挙動が変更可能なプログラム部品形式のメタCASEツール ProtoCASE を提案する．CASEツールの開発者は，本質的な特徴部分をスクリプトにより定義し，関連的な特徴部分をプログラミング言語により実装することにより，独自のCASEツールを開発することができる．

A programing component that supports the development of integration CASE environment

Takeshi Gocho

Here, I propose a Meta CASE tool ProtoCASE. It's a programming component, that can change the action by script language.

1. はじめに

ソフトウェア開発組織が，効果的で，柔軟性や機敏性に優れたＣＡＳＥ環境を実現するには，ＣＡＳＥツールを低コスト・短期間に開発，改良することのできるメタＣＡＳＥツールの活用が有効である．メタＣＡＳＥツールとは，ＣＡＳＥツールの開発を支援するためのツールのことである[1]．現状，組織のＣＡＳＥ環境は，技術や理論の進化に追従していくことが困難であり，理論と実践との間に大きな差が生じてしまい，必ずしも効果的に活用されているとはいえない．その証拠に，日々新たな開発言語や設計手法，管理手法などが次々に登場し，それらに関する数多くの論文や文献が執筆されている．もはや，開発組織は，ツールベンダーが提供するＣＡＳＥツールに頼っていては，市場の要請に応えられる理想のＣＡＳＥ環境を手に入れることはできない．組織の実情に沿ったＣＡＳＥ環境を，柔軟かつ機敏に構築・改善することを可能にする開発手段が求められている．

このような理由から，現在，いくつかのメタＣＡＳＥツールが利用可能である．近年のメタＣＡＳＥツールには，ＸＭＬファイルを仲介情報としてプラグインを容易に可能にするものや，ＣＡＳＥツールの定義情報に基づいてプログラム・コードを生成するもの，あるいはプロセス管理ツールの開発支援と標準化を目的としたＣＡＳＥツール開発プラットホームなどがある．

しかしながら，これら既存のメタＣＡＳＥツールは，どれも開発支援の対象としている範囲が限定的である．例えば，これらの既存のメタＣＡＳＥツールが目的しているのは，モデリング・ツールやプログラミング・ツール，あるいはプロセス管理ツールの開発支援であり，開発プロセス全体（特に要件定義や分析設計といった上流工程）を支援する，統合ＣＡＳＥ環境の構築に必要な開発手段を提供するものではない．このため，ＣＡＳＥツール開発者は，ＣＡＳＥツール開発における大部分の作業を負担しなければならない．統合ＣＡＳＥ環境の構築を全面的に支援するメタＣＡＳＥツールが求められる．

また，これらの既存のメタＣＡＳＥツールの殆どが，その開発アプローチに特有の問題もあり，異なる開発環境への移植が困難であったり，使用可能なプログラミング言語や開発環境が限定される，といった

†日本工業大学大学院工学研究科情報工学専攻

開発上や保守上の問題もある．より生産性・保守性に優れたメタＣＡＳＥツールが求められる．

本研究では，このような既存のメタＣＡＳＥツールにおける課題を解決する，プログラム部品形式のメタＣＡＳＥツールProtoCASEを研究開発中である．本稿では，メタＣＡＳＥツールProtoCASEについての構想及び実現方式について述べる．また，既存のメタＣＡＳＥツールとの比較により，本ツールの優位性を検証し，産業上の利用可能性について考察する．

2. 本ツールの概要

2.1. ツールの特徴

ProtoCASE は，ＣＡＳＥツールを開発する手段をプログラム部品形式にて提供するメタＣＡＳＥツールである．特定のプログラミング言語や開発プラットホームには依存しない言語非依存コンポーネントとして実現されている．当コンポーネントに含まれるプログラム部品群は，ＣＡＳＥツール開発部品と呼ばれ（以下ＣＡＳＥ部品），その内の主要な部品は，ビジュアル開発環境に対応する視覚的な開発手段を備えており，ＣＡＳＥツールの定義情報であるスクリプトに基づいて自己の挙動（振る舞い）を決定する．ＣＡＳＥツールの開発者は，ビジュアル開発環境において，必要なＣＡＳＥ部品を配置し，それらにスクリプトを設定し，部品同士の協調動作やアプリケーション固有をプログラムしていくことにより，ＣＡＳＥツールを開発することができる（図１）．

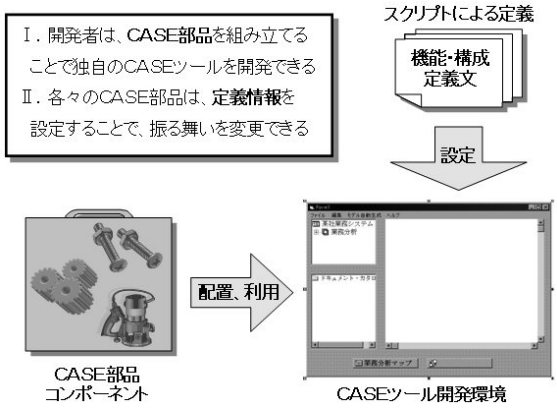

図１．ProtoCASE の概要

本ツールは，特定の機能についての開発手段を提供するのではなく，統合的なＣＡＳＥ環境の構築を支援する．作業標準（一つの作業における標準的な手順と成果物の総称）をＣＡＳＥツールにおける機能単位とし，ＣＡＳＥツールにおける，機能体系を管理する手段，作業標準を定義する手段，ドキュメントを編集する手段，ドキュメントを管理・永続化する手段，利用価値のあるデータを抽出して管理する手段，などをプログラム部品形式にて提供する．このような，統合ＣＡＳＥ環境における開発基盤が与えられることにより，ＣＡＳＥツールの開発者は，ＣＡＳＥツールの主要部分の開発に専念することができる．

本ツールにおいて使用されるスクリプトは，ＣＡＳＥツールの定義情報を記述するためのものである．ＣＡＳＥツールにおける，機能体系を定義するスクリプトと，作業標準を定義するスクリプトと，抽出情報を定義するスクリプトの３種類があり，これらはXML(Extensible Markup Language)により記述された定義ファイルとして作成される．当コンポーネントにおける，各々の主要なＣＡＳＥ部品は，これらのスクリプトをランタイムに解析して，定義情報に変換し，それに基づいて自己の挙動を決定する．

図２は，本ツールの本質（ツールの意図）を概念的に捉えた図である．ＣＡＳＥツールにはそれぞれの個性があり，個性には共通部分と特徴部分がある．ProtoCASE コンポーネントは，共通部分の開発基盤をＣＡＳＥ部品により提供する．ＣＡＳＥツールの開発者は，本質的な特徴部分をスクリプトにより定義し，関連的な特徴部分をプログラミング言語により定義する．

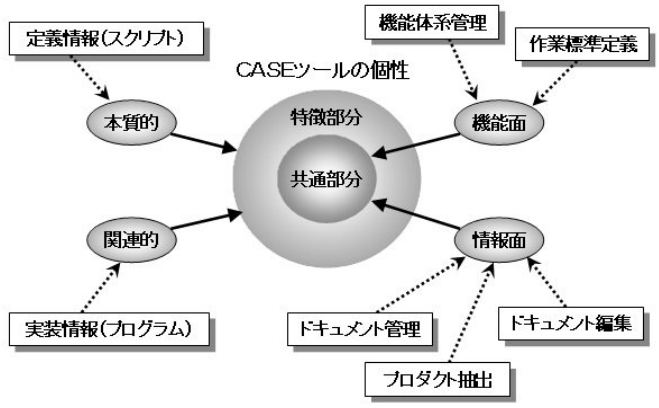

図２．ProtoCASE の本質

2.2. コンポーネント構成

ProtoCASE のコンポーネントに含まれるＣＡＳＥ部品には，ビジュアル開発環境に対応する視覚的な開発手段を備えたコントロール形式と，プログラミング言語からのみ利用するオブジェクト形式がある．コ

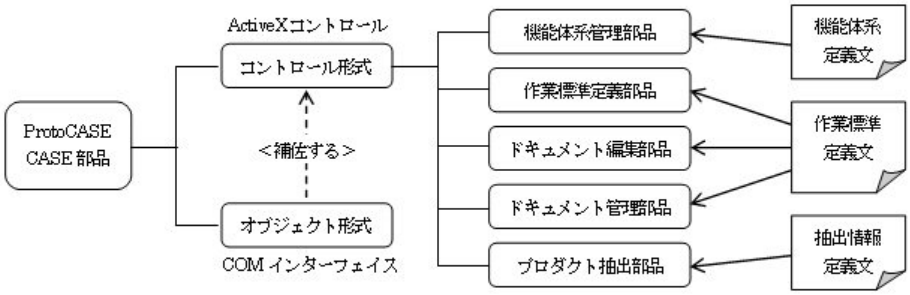

図3．ＣＡＳＥ部品の形式とコントロールの種類

ントロール形式のＣＡＳＥ部品は，ユーザー・インターフェイスの特定部分の機能を開発する手段を備えたプログラム部品であり，一方のオブジェクト形式のＣＡＳＥ部品は，それらコントロール形式のＣＡＳＥ部品の利用を補佐するためのプログラム部品である．

コントロール形式のＣＡＳＥ部品には，機能体系管理部品，作業標準定義部品，ドキュメント編集部品，ドキュメント管理部品，プロダクト抽出部品の５つがある．ＣＡＳＥツールの開発者は，ビジュアル開発環境にて開発を行う場合に，これらのコントロール形式のＣＡＳＥ部品をウィンドウ上に配置し，オブジェクト形式のＣＡＳＥ部品を補佐的に利用して，ＣＡＳＥツールの動作をプログラムする．図３は，ＣＡＳＥ部品の形態とコントロール形式のＣＡＳＥ部品の種類を示した系統図である．

ProtoCASE コンポーネントは，ＣＯＭ（Component Object Model）をベースに実装されており，コントロール形式のＣＡＳＥ部品は ActiveX コントロールとして，オブジェクト形式のＣＡＳＥ部品はＣＯＭインターフェイスとして実現されている．従って，ＣＯＭをサポートする開発環境であれば，自由なプログラミング言語により利用することができる．これにより，ＣＡＳＥツールの開発に，生産性が高く習得が容易な簡易言語（スクリプト言語）を使用することが可能である．当コンポーネントは，将来的には.NET Framework へ移行する予定である．

2.3. 開発方法

ProtoCASE を利用したＣＡＳＥツール開発の手順としては，開発者はまず，ＣＡＳＥツールの定義情報（機能体系定義，作業標準定義，抽出情報定義）が記述された定義ファイルをスクリプト言語（ＸＭＬ）により作成する．次に，ビジュアル開発環境を利用してＣＡＳＥツールの実装を行う．開発者は，必要なＣＡＳＥ部品をユーザー・インターフェイスのプロトタイプとなるウィンドウ上へ配置し，それらのＣＡＳＥ部品に編集された定義情報（ＸＭＬファイル）をプロパティとして設定する．そして，ウィンドウや各々のＣＡＳＥ部品が持つプロシージャ（モジュール）にＣＡＳＥツール固有の動作をプログラムして行く．図４は，本ツールのＣＡＳＥ部品を利用してＣＡＳＥツールを開発したときの，VisualBasic6.0 による開発画面であり，また，図５は，ＣＡＳＥ部品を搭載したＣＡＳＥツールの実行画面である．

図4．ＣＡＳＥ部品を利用したツールの開発例

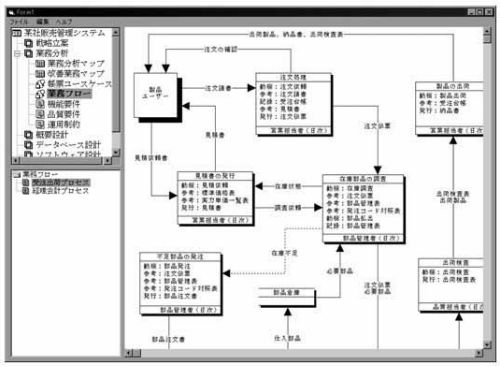

図5．ＣＡＳＥ部品を搭載したツールの実行例

3. 主要CASE部品の仕様

3.1. 機能体系管理部品

機能体系管理部品は，ＣＡＳＥツール上の機能単位である作業標準を管理するＣＡＳＥ部品であり，複数の作業標準をタスク（工程）項目別に分類して管理する．各々の作業標準の項目は，対応する作業標準の定義情報への参照情報を保持しており，機能の選択項目として機能する（図６）．ＣＡＳＥツールの機能体系は，スクリプトで記述された定義情報を解析することにより自動的に構築される．（図７）

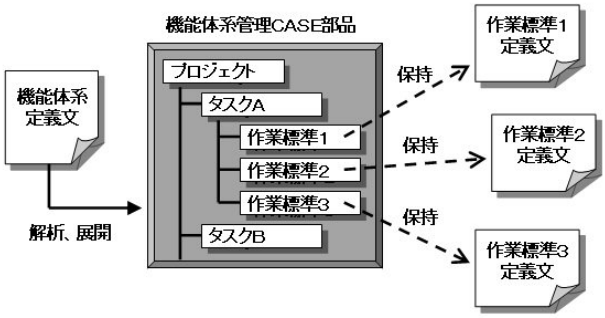

図6．機能体系管理部品の概要図

```
<Function>
    <Task name="戦略立案">
        ・・・
    </Task>
    <Task name="業務分析">
        <Standard name="現行業務マップ", type="MAP", …/>
        <Standard name="改善業務マップ", type="MAP", …/>
        <Standard name="帳票ユースケース", type="MODEL", …/>
        <Standard name="業務フロー", type="MODEL", …/>
        <Standard name="機能要件", type="SPEC", …/>
        <Standard name="品質要件", type="SPEC", …/>
        <Standard name="運用制約", type="SPEC", …/>
    </Task>
    <Task name="概要設計">
        ・・・
    </Task>
    <Task name="ソフトウェア設計">
        ・・・
    </Task>
    ・・・
</Function>
```

図7．スクリプトで記述された機能体系の定義

3.2. 作業標準定義部品

作業標準定義部品は，ＣＡＳＥツール上の１つの作業標準に対応して機能するＣＡＳＥ部品である．ＣＡＳＥツールの開発者は，作業標準に関連する固有のデータやイベント処理を，当ＣＡＳＥ部品の持つプロシージャ(モジュール)にプログラムすることにより，作業標準の機能を定義することができる（図８）．作業標準に関する定義情報は，スクリプトで記述された定義情報を解析することにより取得され保持される（図９）．

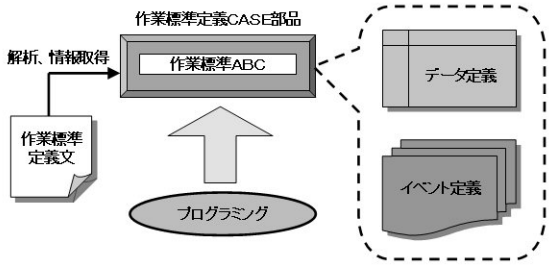

図8．作業標準定義部品の概要図

```
<Standard name="業務分析マップ", type="MAP">

    <Construct>
        <Document typeName="ヘッダ情報", editor="ヘッダフォーム"/>
        <Document typeName="概要マップ", editor="概要フォーム">
            <Category typeName="分類項目">
                <Document typeName="詳細情報", editor="詳細フォーム"/>
            </Category>
        </Document>
    </Construct>

    <Template name="ヘッダフォーム">
        ・・・ヘッダフォーム・テンプレートの定義・・・
    </Template>

    <Template name="概要フォーム">
        ・・・概要フォーム・テンプレートの定義・・・
    </Template>

    <Template name="詳細フォーム">
        ・・・詳細フォーム・テンプレートの定義・・・
    </Template>

</Standard>
```

図9．スクリプトで記述された作業標準の定義

3.3. ドキュメント編集部品

ドキュメント編集部品は，ドキュメントを編集する手段をエディタにより提供するＣＡＳＥ部品である．作業標準の定義情報には，ドキュメントを編集するのに用いるエディタの情報が含まれている．当ＣＡＳＥ部品は，このエディタの情報に基づいて，ドキュメント編集用のエディタを自動的に構築して表示する（図１０，図１１）．さらに，当ＣＡＳＥ部品は，ＯＬＥ(Object Linking and Embeding)を介して，他のアプリケーションにて作成される電子文章への埋め込みが可能である．これにより，開発したＣＡＳＥツ

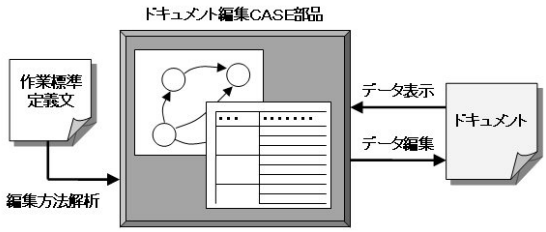

図１０．ドキュメント編集部品の概要図

```
<Standard name="業務フロー", type="MODEL">

    ・・・ドキュメント構成情報・・・

    <Template name="ヘッダフォーム">
        ・・・ヘッダフォーム・テンプレートの定義・・・
    </Template>

    <Diagram name="コンテキスト・ダイアグラム">
        ・・・コンテキスト・ダイアグラムの定義・・・
    </Diagram>

    <Diagram name="業務フロー・ダイアグラム">
        ・・・業務フロー・ダイアグラムの定義・・・
    </Diagram>

</Standard>
```

図１１．作業標準定義内のエディタ定義部

ール（当ＣＡＳＥ部品のエディタ）により編集されたドキュメントを，仕様書など電子文章の作成において利用することができる．

ドキュメント編集部品のエディタには，テンプレートとダイアグラムの２つの形式がある．テンプレート形式のエディタは，文字データの編集を専門とする帳票形式のエディタである．一方のダイアグラム形式のエディタは，図形データの編集を専門とする図面形式のエディタである(図１２)．作業標準定義内の構成管理情報定義部には，ドキュメントの編集に使用するエディタが指定されており，当ＣＡＳＥ部品は，編集対象のドキュメントの編集に使用するエディタの種類を，定義情報に基づいて自動的に識別して決定する．

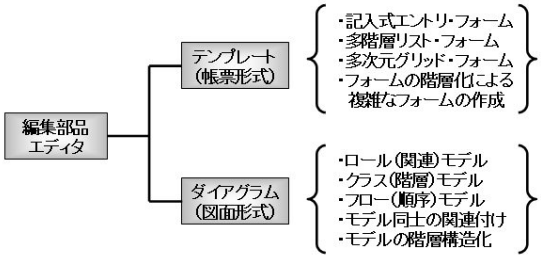

図１２．エディタの種類と機能

テンプレートは，１つあるいは複数のフォームにより構成される．フォームには，空欄記入形式のエントリ・フォーム，表形式のリスト・フォーム，行列形式のグリッド・フォームの３種類がある．これらを階層的に組み合わせることにより，さらに複雑なフォームを構築することも可能である（図１３）．作業標準定義内のテンプレートの定義情報には，これらのフォームの構造（記入項目，カラム，ヘッダなど）やフォーム同士の組み合わせ，フォームに対する命令などが定義される（図１４）．

図１３．テンプレート形式のエディタ

```
<Template name="テンプレート名">
    <EntryForm name="エントリフォーム名", cx="800", …>
        <Struct>
            ・・・記入項目（フィールド）の定義・・・
        </Struct>
        <Arrange>
            ・・・セルの配置・初期設定・・・
        </Arrange>
    </EntryForm>
    <ListForm name="リストフォーム名", cx="700", …>
        <Struct>
            ・・・リスト項目（カラム）の定義・・・
        </Struct>
        <Arrange>
            ・・・行・列の配置・初期設定・・・
        </Arrange>
    </ListForm>
    <GridForm name="グリッドフォーム名", …>
        <Struct>
            ・・・行・列項目（ヘッダ）の定義・・・
        </Struct>
        <Arrange>
            ・・・行・列の配置・初期設定・・・
        </Arrange>
    </GridForm>
</Template>
```

図１４．作業標準定義内のテンプレート定義

ダイアグラムは，配置と編集が可能な複数のモデル要素により構成される．ダイアグラムは，対象間の関連を表すロール・モデル，対象間の階層を表すクラス・モデル，対象間の順序を表すフロー・モデルを表現することができる（図１５）．またさらに，モデル同士の関連付けや階層構造化も支援する．作業標準定義内のダイアグラムの定義情報には，これらのモデル要素の形状や文字データ，モデル要素同士の結合方法を示すルール，ダイアグラムに対する命令などが定義される（図１６）．

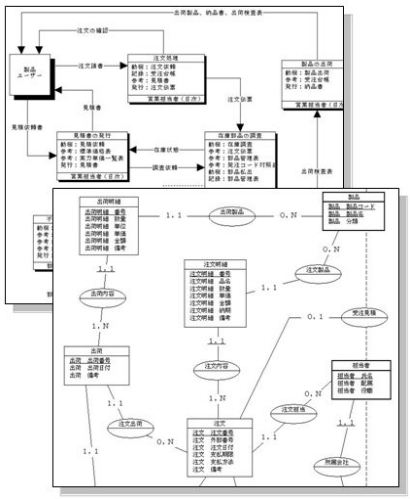

図１５．ダイアグラム形式のエディタ

```
<Diagram name="ダイアグラム名">

    <Entity name="実体名", cx="160", cy="110", …">
      <Create>
          ・・・実体のテキストデータの宣言・・・
      </Create>
      <Paint>
          ・・・実体の描画方法の指示・・・
      </Paint>
    </Entity>

    <Join name="結合名", style="SOLID", …">
        <Create>
            ・・・結合のテキストデータの宣言・・・
        </Create>
        <Dispose point="point">
            ・・・結合の初期配置の設定・・・
        </Dispose>
        <Adjust>
            ・・・結合の描画方法の指示・・・
        </Adjust>
    </Join>

    ・・・その他の図形の定義・・・

</Diagram>
```

図１６．作業標準定義内のダイアグラム定義

3.4. ドキュメント管理部品

ドキュメント管理部品は，作業標準におけるドキュメントの構成を管理するＣＡＳＥ部品である．複数のドキュメントを階層的に表示し，挿入・更新・削除といった管理操作を提供する（図１７）．これらの管理操作は，作業標準の定義情報から解析されたドキュメントの構成管理情報に基づいて自動制御される（図１８）．さらに，ドキュメントに関するデータを，リポジトリに保存する永続化手段も提供する．

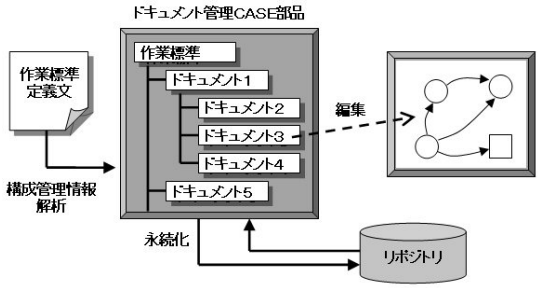

図１７．ドキュメント管理部品の概要図

```
<Standard name="業務分析マップ", type="MAP">

    <Construct>
        <Document typeName="ヘッダ情報", editor="ヘッダフォーム"/>
        <Document typeName="概要マップ", editor="概要フォーム">
            <Category typeName="分類項目">
                <Document typeName="詳細情報", editor="詳細フォーム"/>
            </Category>
        </Document>
    </Construct>

    ・・・エディタの定義情報・・・

</Standard>
```

図１８．作業標準定義内の構成管理定義部

3.5. プロダクト抽出部品

プロダクト抽出部品は，プロジェクトにおいて利用価値のある中間データを抽出・管理するＣＡＳＥ部品である．複数個の抽出データを，リスト形式，ツリー形式，またはテキスト形式で表示し，抽出・利用・挿入・更新・削除といった基本操作を提供する（図１９）．これらの抽出データは，抽出情報の定義情報に基づいて定義される（図２０）．また，当ＣＡＳＥ部品は，抽出データをテンポラリ（一時データの記録媒体）に対して読み書きするための入出力手段も提供する．

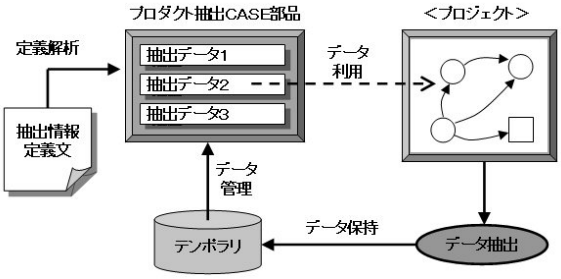

図１９．プロダクト抽出部品の概要図

```
<Product name="現行業務モデル要素">

    <Extract data="業務プロセス", from="現行業務マップ">
        ・・・業務プロセスの抽出方法・・・
    </Extract>

    <Operate data="業務プロセス", uses="業務フロー">
        ・・・業務プロセスの利用方法・・・
    </Operate>

    ・・・その他のデータの抽出方法／利用方法・・・

</Product>
```

図２０．スクリプトで記述された抽出情報の定義

4. 評価と考察

4.1. 既存のメタＣＡＳＥツール

既存メタＣＡＳＥツールとしては，ごく僅かながら，幾つかのＣＡＳＥツール開発支援のためのソフトウェアが利用可能である．EPM[2]などのプロセス管理ツールのプラットホームは，プロジェクト管理データの収集・分析ツールの開発支援を目的としたものであり，開発工程全体を含めたツールの開発を支援するものではない．Eclipse[3]はあくまでＪａｖａ言語によるプログラミングを支援することを目的とした開発プラットホームである．また，統合的なＣＡＳＥツールの開発手段を備えているわけではなく，ＸＭＬによりプラグインが比較的容易に行えることは，ＣＡＳＥツール開発における生産性の向上とは無関係である．MetaEdit[4]は，デザインツールにより作成された定義情報からプログラムコードを生成するメタＣＡＳＥツールであるが，あくまでダイアグラム・エディタとしての機能しか提供していない．また，この方式では，特定の機能を変更する場合に，関連するコードをすべて調べて修正しなくてはならない．

4.2. 本ツールの効果と優位性

本ツールにおいては，ＣＡＳＥツールの本質的な特徴部分を，スクリプトにより記述された定義情報に保存しておくことが可能であり，保守時に変更する必要があるのはＣＡＳＥツールの関連的な特徴部分のみである．従って，異なる開発環境への移植や，改良版のＣＡＳＥツール開発，実装方式の変更などを容易に行うことができる．また，本ツールでは，作業標準をＣＡＳＥツールの機能単位として統一しており，ＣＡＳＥツールの機能やデータを，体系付けたり，関連付けたりすることが容易である．さらに，本ツールに含まれるすべてのＣＡＳＥ部品は，言語非依存コンポーネントとして実現されているので，特定の開発言語や開発環境には依存せず，開発要員の確保を容易に行うことができる．

4.3. 産業上の利用可能性

現在，プログラミング言語や開発プラットホーム，ＯＳ，データベース，といったソフトウェア開発における多くの業界標準（デファクト・スタンダード）が確立されている．しかしながら，要件定義から，分析設計，テスト，プロジェクト管理，といった開発プロセス全体を支援するＣＡＳＥツール（ＣＡＳＥ環境）の開発における業界標準，あるいはインフラ技術といったものは未だ確立されていない．現在の情報産業は，統一化・標準化へ向かう方向，つまり「メタ傾向」にある．このような情報産業の進化の動向に合わせて，ＣＡＳＥツール（ＣＡＳＥ環境）における開発技術も進化していく必要がある．

本ツールは，ＣＡＳＥツール開発における共通の基盤（インフラ）技術を提供するものであり，業界標準プラットホームとなる可能性を秘めたものである．ネットワークのプロトコルと同様に，ＣＡＳＥツール開発における共通基盤（共通データ定義）が確立されていれば，異なるツール間でのデータ共有・交換が可能となる（図２１）．また，標準規格（標準ＡＰＩ）が定められていれば，他者の開発したＣＡＳＥ部品を搭載することで，ノウハウを共有することが可能になる（図２２）．これにより，ソフトウェア工学の理論を隠蔽したＣＡＳＥ部品が，製品として市場に流通する可能性がある．例えば，正規化ＥＲ図の導出基準を隠蔽したＣＡＳＥ部品が市場に流通すれば，多くの開発組織（ＣＡＳＥツール・ユーザー）の間で，

ＥＲ図の自動生成のノウハウを共有することが可能となる．但し，このような，ＣＡＳＥ部品の市場が形成されるためには，周知と合意の下で，ＣＡＳＥツールに関する標準規格が定められる必要がある．

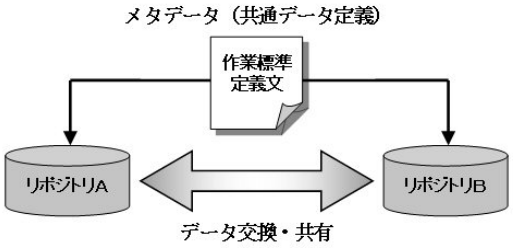

図２１．メタデータに基づくデータの共有化

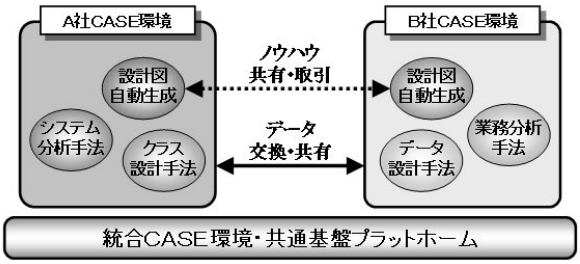

図２２．業界標準プラットホームの構想

5. 今後の課題

ProtoCASE が実用化されるには，次の三つの課題が残されている．一つ目は，スクリプト作成支援ツールの開発である．作業標準のスクリプト（特に，複雑となるエディタの定義情報）を視覚的かつ容易に作成できるツールが必要である．二つ目は，ProtoCASE を利用してＣＡＳＥツールを開発するための方法や手順を示した，開発マニュアルの作成である．ＣＡＳＥツールや開発方法論に関する専門知識を持たない開発組織が，効果的かつ実用的なＣＡＳＥ環境を構築することは困難である．従って，ＣＡＳＥツールが一般に持つべき機能や作業標準を規定したガイドラインや，対象としている分野や業種別の製作カタログなどを用意しておくことが必要である．三つ目は，管理機能を提供する作業標準（管理標準）の考案と，それらの制御手順を自動化するＣＡＳＥ部品の開発である．これらは，プロセス成熟度（CMM）に含まれるＫＰＡを自動化したものになるはずである．要件管理や構成管理，進捗管理における作業標準の他に，変更箇所の自動追跡機能や編集データの自動検査機能などがあれば，非常に強力な統合ＣＡＳＥ環境の構築が可能となる．

6. おわりに

ソフトウェア開発組織の実情に沿ったＣＡＳＥ環境を，柔軟かつ機敏に構築・改善していくには，メタＣＡＳＥツールの活用が有効である．本稿では，スクリプトに基づいて自己の挙動を変更することのできるプログラム部品形式のメタＣＡＳＥツール ProtoCASE を提案し，その構想及び実現方式について述べた．本ツールは，ＣＡＳＥツール開発における共通の基盤（インフラ）技術を提供するものであり，ＣＡＳＥツール開発技術における業界標準プラットホームとなる可能性を秘めたものである．

参考文献

[1] Proceedings of the First International Congress on　Meta-Case (Meta-Case'95). William Ian Cameron　Mitchell. U.K.: University of Sunderland, 1996

[2] EASEプロジェクト. http://www.empirical.jp/

[3] Eclipse. http://www.eclipse.org/

[4] MetaEdit. http://www.metacase.com/