ソフトウェア工学 103－4
（1995．3．9）

製品特化CASE構築のためのドメイン分析

清水 洋子 (shimizu@ssel.toshiba.co.jp)
小尾 俊之 (obi@ssel.toshiba.co.jp)
三原幸博 (mihara@ssel.toshiba.co.jp)

株式会社 東芝 研究開発センター システム・ソフトウェア生産技術研究所

ソフトウェア開発を支援するCASEの効果をより確実にするものとして，特定の対象分野(ドメイン)に焦点を絞りより高度な支援を実現する製品特化CASEへの期待が高まっている．製品特化CASEは個々のドメインごとにCASEを開発していくアプローチであり，これを成功させるポイントは，ドメインに確実に適合するCASEを効率良く開発することにある．このためには，対象製品ドメインの性質を十分に分析することが重要である．

そこで本論文では，このドメイン分析作業を行う手法を提案する．本手法は七つの作業項目から成り，各作業に対して具体的な手順および指針を示している．特徴は，既存システムを効果的に活用することにあり，既存システムの分析からドメインの性質を捉え，再利用の枠組を確立していく．ドメイン分析の成果物として，そのドメインを表現するドメインモデルを作成する．このモデルを用いることで，製品特化CASEを確実かつ効率良く開発することが可能となる．

我々は本手法を，現金自動取引装置の制御ソフトウェア用CASEの開発において適用し，その効果を確認した．

Domain Analysis for Developing Domain Specific CASE

Yoko SHIMIZU, Toshiyuki OBI, Yukihiro MIHARA

Systems and Software Engineering Laboratory,
Research and Development Center, TOSHIBA Corp.

70 Yanagi-cho Saiwai-ku Kawasaki, 210, Japan

To increase the efficiency of CASE, Domain Specific CASE is expected. This CASE focuses on one domain, and realizes high level support. A Domain Specific CASE has to be developed for every domain, and there are two key items for success: "how precise to fit to the domain", and "how efficient to develop". To satisfy these items, it is important to analyze the target domain fully.

In this paper, we propose an domain analysis technique. This technique consists of seven works, and each work is provided with steps and guidelines. A feature of this technique is to make effective use of already developed software. From the software, analysts grasp the characteristics of that domain, and form the frame of reuse. A result of analysis is called a domain model. By referring this model, a Domain Specific CASE can be developed efficiently, and the CASE can be fittable to the target domain.

We applied this technique to develop a CASE for automatic teller machine's control software, and confirmed our technique's effect.

1 はじめに

ソフトウェアの生産性，品質の向上を支援するものとしてCASEへの期待は大きい．しかし，実際にCASEを適用して生産性，品質の向上を実現するには，難しい面があることが指摘されている．その主な原因は，CASEが汎用的な支援を狙っていることにあると考えられる．そこで近年，特定の製品分野に支援対象を限定した製品特化CASEが注目されている．これは，専用性を高めることでより高度な支援を実現し，確実に効果を得ようとするCASEである．

製品特化CASE構築アプローチの流れは，図1のようになる．CASEは開発手法と支援ツールから構成される．まず，開発手法の雛型を基に支援対象製品向けの開発手法を確立する．そして，ツール構成の雛型に沿って実際に対象製品向けCASEツールを開発する．この両者によって，支援環境は完成する．

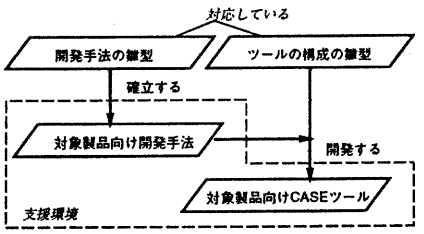

図 1: 製品特化CASE構築のアプローチ

ここで，このアプローチを成功させ効果を得るためには，以下の二点が重要となる．

- 開発手法を，対象製品に確実に適合させること
- ドメインCASE構築の作業を効率良く行うこと

対象製品の特徴に合う支援を実現することがそもそもの目的であり，これに失敗すると期待した効果は得られない．また，製品特化CASEは各対象製品ごとにそれぞれ支援環境を構築するため，この開発コストに見合う効果を得るためには，開発自身を効率化しなければならない．

図1では製品特化CASE構築のための二つの作業「開発手法を確立する」,「ツールを開発する」を示している．この作業が上記重要項目を満足するには，事前に対象製品領域(ドメインと呼ぶ)の性質を十分分析し，その結果として対象ドメインの性質を表現したドメインモデルを作成することが効果的である．しかしこのドメイン分析作業に対しては，手順や指針となるものが存在せず，分析作業は困難であり多くの時間を費す結果となる．つまり，製品特化CASE構築のアプローチを成功させる鍵は，このドメイン分析作業をナビゲートする手法にあるといえる．

本論文ではこの手法をドメイン分析手法と呼び，これを提案することを目的とする．この手法を用いて対象ドメインのドメインモデルを作成すると，これを参照することにより製品特化CASE構築作業を確実かつ効率良く行うことができ，生産性，品質の向上に寄与することが可能となる．

以下では，まず2節において，本論文で扱うキーワードについて定義を与える．3節においてドメイン分析手法が仮定する前提条件やドメイン分析手法の要件を明確にした後，4節にドメイン分析手法を提案する．そして，この手法を現金自動取引装置(ATM[1])に適用した事例を5節に紹介する．6節において提案したドメイン分析手法を評価し，7節に結論をまとめる．

2 定義

1節で述べた通り，本論文でいうドメイン分析手法とは，製品特化CASE構築のために重要となるドメイン分析作業をナビゲートする手法である．そしてドメイン分析の結果として得られるドメインの表現を，ドメインモデルと呼ぶ．

本節では，ドメイン分析手法の目的をより明確にするため,「製品特化CASE」および「ドメインモデル」について定義を与える．

2.1 製品特化CASE

本論文で仮定する製品特化CASEは，ソフトウェアの設計から実装の工程を支援する．支援の中心は，設計者が一定の形式に従って記述したソフトウェアの動作仕様を基に，ソフトウェア部品を合成してソースコードを自動生成することである．これを基本として，更に仕様のデバッグを支援したり，仕様記述等の構成，版管理をするなどの拡張も可能であるが，本論文では中心となる支援のみを扱う．

製品特化CASEの構成は，図2に示す通りである．図中四角で示すのがツールであり，したがってツール構成の雛型は，エディタとジェネレータの二つのツールから成る．エディタは動作仕様を記述するためのツールであり，ジェネレータはソースコードを自動生成するツールである．

開発手法の雛型は，以下の三つのステップで表現できる．

1. エディタを用い，動作仕様を一定形式で記述する
2. 仕様記述をデバッグし，正しさを確認する
3. ジェネレータによりソースコードを自動生成する

ソースコード生成後に誤りが発見された場合には，ステップ2，3を行う．

特にステップ1に対しては，仕様記述の手引が，ドキュメントあるいはエディタのナビゲート機能として提供される．

[1] Automatic Teller Machine

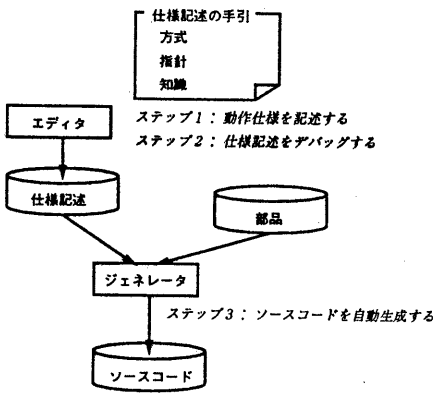

図 2: 製品特化 CASE の構成

この製品特化 CASE は，以下の特長を持つ．

高品質なドキュメント 一定形式に従って動作仕様を記述するため，厳密かつ読解性の良いドキュメントが生成される．また仕様記述の手引があるため，ドキュメント品質の人への依存度が低くなり，品質が安定する．更にこの仕様記述からソースコードを自動生成しているため，ソースコードとドキュメント記述の内容の一致を保証できる．

再利用の促進 まず，部品を積極的に再利用している．更にソフトウェアの構造も再利用し，これを仕様記述方式や生成したソースコードの実行方式へ反映している．

2.2 ドメインモデル

ドメインモデルはドメイン分析作業の成果物である．そして図 1に示す二つの作業を行うに十分な情報を含んでいなければならない．この観点から，ドメインモデルを構成する要素は以下である．

ドメインの境界 ドメインモデルが表現する範囲を明確にするためのドメインの境界．

仕様記述言語 記述を行う言語の文法および語彙．ここでの語彙は，言語仕様として規定する予約語，およびドメインでのプリミティブな機能単位である部品の名称に相当する．この情報は，エディタを開発するのに用いる．

仕様記述法 以下の三つの項目があり，これらの情報は仕様記述の手引を開発するのに用いる．

- **仕様記述方式** 仕様記述をどのような単位に分割して行い，各単位のインタフェースをどう取るか，といった記述の方式
- **仕様記述指針** 同じ意味を数通りの形式で記述できる場合，どれで記述すべきかを示す指針
- **仕様記述知識** 仕様記述に必要な，ドメイン特有の用語や常識

部品外部仕様 個々の部品の機能仕様．この情報は，部品の開発に用いる．

自動生成方式 部品合成の仕方，生成ソースコードの構成および実行形式．この情報は，ジェネレータの開発に用いる．

以上定義した言葉を用いてドメイン分析手法の目的をまとめる．製品特化 CASE の対象ドメインを表現するドメインモデルを作成し，これによって製品特化 CASE の構築を確実かつ効率良く行えるようにすることが，ドメイン分析手法の目的である．

3 ドメイン分析手法提案の準備

本節では，ドメイン分析手法の提案にあたり，この手法が仮定する前提条件を設定する．また，手法が提供すべき情報項目についても明確にしておく．

3.1 手法適用の前提条件

手法適用の前提条件として以下の二点を設定する．

類似システムの開発が多い 2節で述べたように，ドメイン CASE は再利用を促進するという特長を持つ．この特長が有効に働くためには，対象ドメインに再利用する機会が多くなければならない．例えば，シリーズ製品が存在するドメイン，標準製品を納入先ごとの要求に合わせてカスタマイズするドメイン，頻繁にバージョンアップが行われるドメイン等が該当する．

既存システムがある 本手法では，まったく新規に開発するドメインではなく，既にシステムが存在しているドメインを対象とする．既存システムにはドメインを分析するための情報が豊富にあり，再利用可能な部品の抽出や，仕様変更の可能性，システムの性能予測等の作業をより正確に行うことができるため，より具体的で有効な手法を提案できる．

3.2 手法が提供すべき情報項目

ドメイン分析作業の十分性，効率性のために必要な情報項目は以下である．

必要な作業項目と全体の流れ 必要な作業項目を明確にし，分析作業にもれがないようにする．また，各作業項目をどのような順番で実施していくかという全体の流れを示す．

各作業項目の進め方 各作業の具体的内容と指針を示す．

4 ドメイン分析手法の提案

本節では，3.2項で述べた二つの項目，(1) 必要な作業項目と全体の流れ，(2) 各作業項目の進め方，に分けてドメイン分析手法を提案する．

4.1 必要な作業項目と全体の流れ

ドメイン分析作業の目的はドメインモデルの構築である．したがって必要となる作業項目は，ドメインモデルの構成要素それぞれを作成する作業である．

これら作業の全体の流れは，およそ図3に示すようになる．ただし，図に示す作業の前後関係は，後戻りしたり部分的に並行に行うといった可能性も十分ある．

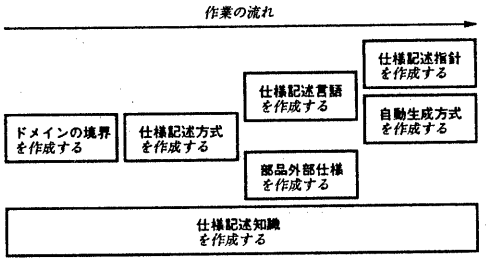

図 3: ドメイン分析作業の流れ

4.2 各作業項目の進め方

以下に，各作業の進め方について述べていく．

4.2.1 ドメインの境界を作成する

(1) ソフトウェアの機能概要を把握する

実システムを動作させてみたり，マニュアル，仕様書等のドキュメント類を調査する．機能を把握する場合は，動作の流れを捉えるだけでなく機能の性質で分類することを行う．一つのシステムであっても，ユーザインタフェース，データ処理，ハードウェア制御，通信など，複数の性質の機能を複合して持っている場合が多い．そして，これらの性質によって動作仕様の適切な表現形式も異なってくる．したがって「対象ドメインがどのような性質の機能を含んでいるのか」という捉え方が重要となる．

(2) 現状ソフトウェアの基本構造を調べる

基本構造とは，モジュール分割法や全体の動作機構などである．適切なドキュメントが存在すればこれを参照すれば良い．しかしこのような全体を把握する情報は，ドメインの常識となっていることもあり明文化されていない場合も多い．そのときは，ドメインの作業者へインタビューしたり，設計書やソースコード等から興す作業を行う．

ここで重要なのは，特に並行処理単位となっているものを明確にすること，および各処理単位のインタフェースを洗い出すことである．最終的には支援の対象とするドメインの境界を決定することが目的であるが，その場合処理単位が一つの指針となる．なぜなら，並行処理単位を現状と同じにすることでシステム全体のスピードに関する見通しを立てやすく，また他とのインタフェースが明確になっていれば適用を行いやすいからである．

(3) 機能とソフトウェア構造との対応づけを行う

今までに得た二つの情報の対応を検討する．

これにより，支援対象を決定した時にそれがソフトウェア構造のどの部分に対応し，どのような性質の機能を持っているかを捉えることができる．

(4) 仕様の変更が発生する可能性を調べる

具体的には，複数システムを比較したり同一システムのバージョンアップ履歴を調べる．今後の拡張計画等があれば，これも併せて調査する．

どの機能に変更が多く発生するのか，そしてその機能はソフトウェア構造上局所化されているか，という点に注目する．つまり，既存のソフトウェアが良い構造であるかを「変更の局所化」という尺度ではかるのである．良い構造であればこれを踏襲し，そうでなければソフトウェア構造の変更についても検討する必要がある．

(5) 支援対象を決定し，ドメインの境界を明確化する

製品特化CASEではソースコードの自動生成を実現するため，ドメインの境界はソフトウェア構造に対して明示できなければならない．そして対象部分が実現する機能は何か，その機能はどういう性質のものかといった対応も，同時に示す．

対象部分の決定では，支援の効果を上げる点から変更の生じやすいところが候補となる．そしてソースコードの自動生成を成功させるために，他の部分とのインタフェースが明確になっていることが条件となる．対象の候補が多い場合には，支援を実現するマンパワーを考え，適度な規模に分割して段階的に支援を拡大するよう計画する．

4.2.2 仕様記述方式を作成する

(1) 部品として再利用できる単位を切り出す

ここでは個々の部品をすべて洗い出す必要はなく，部品種類や粒度を設定できればよい．

既存のソフトウェアを分析する．ここでは，分析する範囲を適切に設定することが重要である．類似機能が複数あれば，その中の一つの機能を選択して分析すれば良い．特殊な機能や例外処理，エラー処理など複数の処理パターンがある場合は，そのパターンをもれなく分析する必要がある．狙う部品の粒度

が細かければ，分析対象として詳細設計書あるいはソースコードといった詳細レベルのものも用いる．

既存のソフトウェアにライブラリが存在すれば，これは部品の一つの候補となる．ただし，ライブラリに対して利用率はどの程度か，バリエーションは十分かといった評価を行い，部品単位として適切か否かを判断しなければいけない．

ソースコードの自動生成は，仕様記述に基づいて部品を合成することにより実現する．したがって，部品は中身にまったく手を加えずに使用できる単位である必要があり，部品同士の間には依存関係がないことが望ましい．また，要求される動作仕様を実現するに十分な部品が揃っていなければならない．以上を満たすよう，部品単位を切り出す．

(2) 記述単位を決定する

既に決定したドメインの範囲に対して，その動作仕様記述をどのような単位で行うかという決定であり，これはソフトウェア開発におけるモジュール分割の考え方と同じである．

既存ソフトウェアのモジュール分割が一つのたたき台となる．これに対し，共通機能をうまく切り出しているか，各モジュールは意味をもった単位であるか，モジュール間の関係は疎になっているか，といった観点から評価を行う．

また，設計書やソースコードを状況を設定して読んで見ることも大切である．状況は，そのドメインでのデバッグ作業や保守作業を想定して設定する．例えば「ある動作の一連の流れを追う」,「ある動作を実行するのはどんな場合かをすべて列挙する」,「ある変更に対し影響が及ぶ範囲を調べる」といったものが考えられる．この作業が困難であれば，モジュールの分割単位が不適切である可能性がある．

更に，ソースコードの冗長性に注目して分析をする．分析対象は，初期開発のソフトウェアではなく，何度か追加変更を加えたシステムが適している．類似コード，デッドコードなどを調べ，その原因を考察する．共通機能切り出しの不十分さ，機能のバリエーションの不十分さ，機能単位の不適切さを示唆する情報が得られる可能性がある．

以上の結果を基に最適なモジュール分割を決定し，これを記述単位とする．

(3) 各記述単位の間のインタフェースを決定する

具体的には，どこの間にインタフェースが必要か，そこで交換する情報は何か，である．

4.2.3 仕様記述言語を作成する

(1) 基礎となる表現形式を決定する

対象ドメインがどのような性質を持つかについては既に明らかになっているので，この性質に適するような，形式的な表現方法を選択する．

例えば，イベントドリブンな制御であればディシジョンテーブルや状態遷移図，データ検索処理ならER図，データ解析処理ならBNFや正規表現などが候補として挙げられる．もちろん，対象ドメインで固有の表現形式を持っていれば，これをそのまま採用することも考えられる．複数の性質を持つドメインであれば，複数の表現形式を組み合わせる．

(2) 表現形式をカスタマイズする

まずはカスタマイズすべき要件を明らかにするため，選択した表現形式を用いて実際に動作仕様を記述してみる．記述対象は，典型的処理，特殊処理，例外処理，エラー処理などのパターンを網羅して選択する．

この結果をもとに，表現形式のカスタマイズを行う．動作を記述しきれない場合には，適切な記述能力を付加する．表現があいまいになる部分には，厳密性を与える制約を与える．表現に自由度がありすぎる場合には，表現を統一させるための規則を用意する．記述が冗長になる部分があれば，マクロを準備し簡易表現を可能にする．

(3) 仕様記述言語を確立する

カスタマイズした表現形式の文法と語彙を明文化することで，仕様記述言語として確立する．語彙は，特に予約語を定義しておく．

4.2.4 部品外部仕様を作成する

(1) 部品の洗い出しを行う

部品の粒度と種類については既に決定しているので，これに沿って具体的に部品を洗い出す．

(2) 部品の外部仕様を明らかにする

部品の持つ機能，設定できるパラメータ，その部品を使用する場合の制限，他部品の関連性などを明確にし，設計者が部品を正しく使用するに十分な情報を用意する．

4.2.5 仕様記述指針を作成する

(1) 仕様記述の指針をまとめる

このためには，既に決定した仕様記述言語を用いて記述を試行してみるのが良い．

同じ仕様を異なる形で表現できるような場合，できあがる仕様記述の品質が人に依存する結果となる．記述の仕方の良否を判断する指針をまとめたり，手本となる記述を例示する．

また仕様記述作業をナビゲートするよう，試行で得た記述手順のノウハウなどをまとめる．

4.2.6 自動生成方式を作成する

(1) 性能制約を明確にする

スピード面および実行モジュールサイズ面での性能制約を明確にする．スピードについて明確な性能仕

様が与えられていない場合には,「既存のソフトウェアと同等のスピード」を基準にする.

サイズの面では,特にマイコン組込みソフトウェアなどでは明確な制限が存在するので,これを明らかにしておく.

(2) 生成方式を決定する

まず,生成するソフトウェアの動作機構を決める.仕様記述を実行していく命令を組み込んだソースコードを生成するか,あるいは仕様記述をデータとして保持しこれを逐次実行するエンジンを作成するかといった選択が考えられる.一般に,スピード面で優れているのが前者であり,サイズを小さくできるのが後者である.制約に照らし合わせ,両者を折衷した機構にしても良い.

そして,仕様記述と部品をどういうコードに変換するか,あるいはどういうデータ形式に変換するかといった変換ルールを検討する.

次に生成のために必要となる情報を洗い出し,この表現形式を決定する.例えば仕様記述の格納ディレクトリや部品の格納ディレクトリ,生成する実行モジュール名などの情報が考えられ,一般にはこれらを一定形式で記述したメイクファイルを使用する形を採る.

4.2.7 仕様記述知識を作成する

(1) 用語辞書を作成する

ドメイン分析一連の作業の中で獲得した,ドメイン特有の用語の意味をまとめる.

特に,一般には使われない特殊な用語,一般に使う言葉であるのに特殊な意味を持つ用語,類似した言葉であるのに明確に異なる意味を表現する用語などは重要である.

(2) ドメインの常識,ノウハウを明文化する

ドメイン分析で獲得した常識,ノウハウの中で,特に今までドキュメントとして残されていないものを明文化する.

5 事例

我々は,提案したドメイン分析手法をATMの制御ソフトウェア(以下,単にATMソフトという)用のCASE構築に適用した.その結果得られたATMソフトのドメインモデルについて,その概要を本節で紹介する.

5.1 ドメインの境界を設定

図4は,ATMソフトの機能とソフト構造の対応を示している.機能は,図4の上側に示すように,大きく三つに分類して捉えることができる.

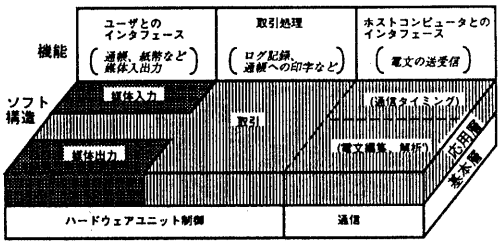

図 4: ATM ソフト:機能とソフト構造の対応

ユーザとのインタフェース キャッシュカードや通帳,紙幣といった媒体の入出力などを行う.センサ情報等をイベントとしてハードウェアユニットの制御を行う.

取引処理 取引の中心となる処理を行う.具体的には,通帳への取引記録印字,ログ情報の記録などであり,これもハードウェアユニットを制御することで実現する.

ホストコンピュータとのインタフェース 口座の認証や口座データの書き換えのために,ホストコンピュータと電文の送受信を行う.電文の編集,解析といったデータ処理と,通信処理とがある.

一方,この機能に対応させてソフト構造を表現したのが図4の下側である.全体が,基本層と応用層に大きく分けられている.

基本層 ユーザとのインタフェースおよび取引処理に対応する部分は,ハードウェアユニットそれぞれを制御するモジュールの集まりから成る.またホストコンピュータとのインタフェースに対応する部分は,一回一回の通信を行うモジュールがある.

応用層 全体が,動作の時間順序によって分割されている.取引に必要な媒体を取り込む媒体入力部,実際の取引を実行する取引部,媒体を排出する媒体出力部である.媒体入力部および出力部は,ユーザとのインタフェースのみによって実行されるが,取引部についてはホストコンピュータへの問い合わせや報告をしながら処理を行い,同時にユーザへ案内表示も行うという,多方面に渡る処理を行っている.

応用層は変更が生じる可能性が高い.ユーザインタフェースは,操作性の良さで各銀行が差別化を計ろうとする狙い目であり,またホストコンピュータとのインタフェースは,各銀行のホストコンピュータの仕様に合わせてカスタマイズが必要となるからである.

したがって,支援の対象としては応用層が挙げられる.図4から分かるように,媒体入力部と媒体出力部はソフトの性質が似ており,同じ支援方式を用いることができる.一方取引部は複数の性質を持っており,より広範囲を支援できる方式でなければならない.

以上の点から，段階的な支援の実現としてまず媒体入力部をドメインの境界として設定した．この支援が完成すれば，そのまま媒体出力部へ適用拡大できることが見込まれる．また取引部については，この支援を拡張するという立場で検討していくことができる．

5.2 仕様記述方式を決定

仕様記述は，図5に示す単位で行うことにした．図5の中で点線より上側が支援対象部分であり，二階層に分割している．下層の役割は，各ハードウェアユニットの制御を行うモジュールの動きを，モジュールからの逐次報告を基に監視し，適切な指示を与えることである．したがって分割単位は，キャッシュカード，通帳，紙幣などハードウェアユニットの単位と対応する．上層の役割は，下層で監視しているハードウェアユニット間の動作順序や同期といったインタラクションを制御することである．分割単位は「引き出し」,「残高照会」といった取引単位と対応する．

下層に切り分けた処理部分はどの取引の場合でも同じ動作をするため，共通機能の切り出しという観点から良い構造であるといえる．また，銀行ごとに違いが生じるユーザインタフェースは，ハードウェアユニットの動作順序やタイミングに関わるものに集中しており，変更は上層部に局所化されている．この点からもこの構造は良いと評価できる．

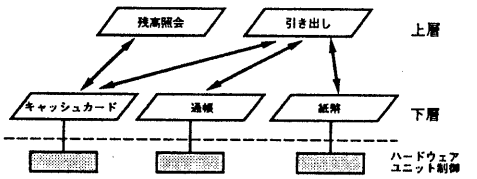

図 5: ATM ソフト:仕様記述方式

分割した各モジュール間のインタフェースは，図5に矢印で示している．上層と下層の間でのみ情報交換を許し，同じ層同士ではできないように制限している．上層から下層へは指示が与えられ，下層から上層へは報告が行われる．このように自由度を制限することで，全体の流れの把握を単純に行えるようにし，更に記述の仕方が人に依存して多様性を持たないようにしている．

5.3 仕様記述言語を確立

まず，基本となる表現形式に状態遷移図を採用した．その理由は，状況報告や指令といったイベントをトリガーとして動作が進んでいくこと，およびその動作がその時の状態に応じて異なってくることを適切に表現できるからである．

下層の一つ，キャッシュカードモジュールの記述の一部を図6に示す．記述の指針として「監視しているハードウェアユニット制御モジュールからの報告をイベントとして状態を遷移させていく」ことを決定した．つまり，何を状態とするかという判断がこのモジュールのインタフェースによって明確に与えられるため，下層の動作仕様記述はどの設計者が記述しても同じものができあがる．

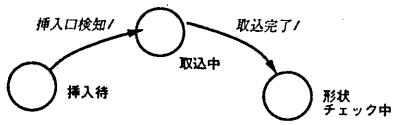

図 6: ATM ソフト:キャッシュカードの動作仕様記述例

上層の一つ，引き出し取引モジュールの記述の一部を図7に示す．記述の指針として「現在動作しているハードウェアユニットの組み合わせが異なる時は，それぞれ別の状態に分割して記述する」ことを決めた．これに従うことで，設計者に依存した記述形式のバラツキが抑えられる．

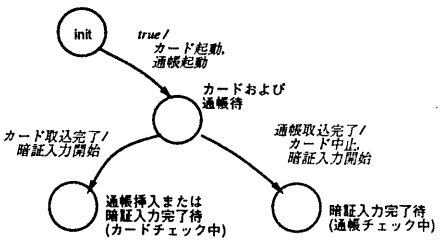

図 7: ATM ソフト:引き出し取引の動作仕様記述例

5.4 自動生成方式を作成

生成方式は，図5に示した記述単位をそのままモジュール単位とし，状態遷移図の動作をそのままコードに展開する方式を採った．

自動生成時に，類似部分を一つのサブルーチンにまとめるなどのコンパクション機能を組み込むことで，従来ソフトウェアの約 1.1 倍のサイズに収められることを確認した．また，スピードでは従来の±0.3%といった誤差内に収めることができ，劣化のないことを確認した．

6 評価

本手法の目的は，ドメインモデルを作成することで製品特化 CASE の構築作業を確実かつ効率良く行えるようにすることであった．そこで本節では，ATM ソフト用 CASE 開発への適用事例を通して，確実な CASE 開発ができたか，効率の良い CASE 開発ができたかという二つの観点から評価を行う．

6.1 確実性

製品特化CASEの雛型をATMソフトに合うようカスタマイズした作業については，確実であったと判断できる．その根拠は主に以下の二点である．

ATMソフト用CASEの完成 5節に紹介したように，本手法を用いてATMソフトのドメインモデルを作成し，これを基にしてATMソフト用のCASEを完成させた．このCASEは現在までに複数銀行の開発に活用しており，以下を確認している．

- 機能および性能を満足するATMソフトを開発可能である
- 開発工数を，従来の7割程度に削減できる
- 単体テストでの不具合発生件数が，従来の半分以下になる

ATMソフト用CASEツールの構築では，初めからすべての部品を準備しておくのではなく，適用を重ねるに従って必要な部品を開発し蓄積していくという方法を採った．これにより工数の削減率が低くなっているが，適用を重ね部品が充実することにより，半分程度への削減が可能であるとの見通しを得ている．

事例適用での経験から，ドメイン分析作業の中の「部品外部仕様の作成」および「仕様記述知識の作成」については，製品特化CASE構築の前段階ですべてを完了させるよりも，構築途中あるいはCASEを実際に適用する途中で随時追加できるような枠組を準備することが，より効果的であることが分かった．

ATMソフト用CASE適用の成功 ATMソフト用に構築したCASEの適用は，ATMドメインの初心者が行った．ATMソフトの基本知識，およびATMソフト用CASEの習得は約一ヶ月で完了し，その後は独力でソフトウェア開発を行うことができた．

これは，ドメイン知識や仕様記述方式等の手引を明確に作成した効果が大きい．また，知識獲得の作業をドメインの専門家との共同で行ったことにも成功の要因がある．

6.2 効率性

ドメイン分析作業からATMソフト用CASE構築作業までを含めたコストを償却する目安を試算し，これをCASE使用の機会の回数と照合した結果，十分効果があるとの結論を得た．

ドメイン分析作業自体を効率的に行えた要因として，本手法のいくつかの作業において，参考となる事例経験を我々が持っていたことが挙げられる．手法においては手順および指針だけでなく，具体的な事例の提示が重要であるといえる．

7 まとめ

本論文では，製品特化CASE構築のアプローチを成功させるために重要な，ドメイン分析の手順と指針を示すドメイン分析手法を提案した．そして本手法をATMソフト用CASE開発に適用し，その効果を確認した．

課題として，分析作業の中にはCASE適用後までの長いスパンで捉えるべき作業項目があること，また各作業で参考とする事例を豊富に用意すべきことが分かった．

今後はこれらの課題に取り組み，手法をより充実させていく．

謝辞

本研究における事例適用の活動は，株式会社東芝システム・ソフトウェア生産技術研究所の藤巻昇氏および玉木裕二氏の多大な尽力によって成功することができました．ここに深く感謝の意を表します．

参考文献

[1] 加地他，"状態遷移図をベースとした部品合成システムの試作"，情報処理学会第38回全国大会，1989.

[2] 岸他，"状態遷移モデルに基づくプログラム部品合成システムの開発"，情報処理学会研究報告91-SE-80，1991.

[3] 藤巻他 "金融機器組み込みソフト向けPOLの開発"，情報処理学会第45回全国大会，1992.

[4] 清水他，"金融機器組み込みソフト向けCASE"，情報処理学会研究報告93-SE-91，1993.

[5] 玉木他，"金融機器組込みソフトウェア向けCASE用ジェネレータ"，情報処理学会第47回全国大会，1993.

[6] 清水他，"金融機器組み込みソフト向けCASE"，日本科学技術連盟第13回ソフトウェア生産における品質管理シンポジウム，1993.

[7] 三原，"プロジェクト管理とCASE環境"，日本ソフトウェア科学会第11回大会論文集，1994.

[8] 田村他，"ドメイン分析・モデリング技術の現状と課題"，情報処理学会誌 Vol.35 No.10，Oct.1994.

[9] S. シュレィアー／S.J. メラー著，本位田真一／伊藤潔監訳，"続・オブジェクト指向システム分析"，啓学出版，1992.