

オブジェクト指向フレームワークと製品特化 CASE による組み込みシステム開発 — ATM (現金自動取引装置) への適用 —

早瀬 健夫

株式会社 東芝 情報通信・制御システム事業本部 SI 技術開発センター
〒183-0021 東京都府中市片町 3-22
E-mail: hayase@sitc.toshiba.co.jp

あらまし

本稿では、フレームワークのインクリメンタルな構築プロセスと、フレームワークの製品特化 CASE による利用プロセスについて述べる。前者については、三つの構築フェーズ (インフラ、ジェネリック、ドメインフレームワーク構築) を設けることで、フレームワークの構造や機能を段階的に検証可能である。また、後者については、製品特化 CASE の相補的な導入により、フレームワークを適切な方法で利用可能である。フレームワークは再利用部品を提供し、製品特化 CASE はシステム特化部分を自動生成する。

本アプローチを組み込みシステムとして位置づけられる ATM (現金自動取引装置) ソフトウェアに適用し、本アプローチの効果と課題について考察する。

キーワード: オブジェクト指向フレームワーク, 製品特化 CASE, プロセス, 組み込みシステム, ATM (現金自動取引装置)

An Embedded System Development with Object-Oriented Frameworks and Domain-Specific CASE tools : Applying to ATM (Automatic Teller Machine) System

Takeo HAYASE

System Integration Technology Center, TOSHIBA Corp.,
3-22, Katamachi, Fucyu-shi, Tokyo 183-0021, Japan
E-mail: hayase@sitc.toshiba.co.jp

Abstract

This paper describes the incremental development of object-oriented frameworks (frameworks) and the use of frameworks by domain-specific CASE (Computer Aided Software Engineering) tools. We can verify structure and function of a framework by defining three phases: infrastructure, generic, and domain framework development. Then, we can use correctly and rapidly the framework by introducing domain-specific CASE tools. The framework supplies the basic part as reusable components, and the domain-specific CASE tools generate automatically the system-specific part.

We apply our approach to ATM (Automatic Teller Machine) software that is kind of an embedded system. We show the result of applying to ATM software, and discuss the effects and the potential problems in our approach.

Keywords: object-oriented framework, domain-specific CASE tools, process, embedded system, ATM (Automatic Teller Machine)

1. はじめに

近年、オブジェクト指向技術は、再利用性や拡張性などのメリットから広く普及し、特に再利用技術としてオブジェクト指向フレームワー

ク (以下、単にフレームワークと呼ぶ) が注目されてきた。フレームワークとは、オブジェクト間の相互作用のパターンを規定したクラスライブラリである[1]。フレームワークを適用す

ることにより、設計・実装の開発工数を短縮することが望める。現在では、GUI[2][3]、OS[4]といった汎用的に利用可能なものだけでなく、ドメイン向けの事例[5][6][7]もいくつか報告されており、フレームワークの適用が活発に行われてきている。しかし、フレームワーク構築の手順や、フレームワーク利用の効率化に関する議論は十分されていない。

本稿では、フレームワークのインクリメンタルな構築プロセスと、フレームワークの製品特化 CASE による利用プロセスについて述べる。フレームワークの構築プロセスとして、三つのフェーズを設け、段階的にフレームワークの構造や機能を検証する手順を示す。また、フレームワークの利用プロセスとして、フレームワークを効率的に正しく適用するために利用ルールを内包した製品特化 CASE を併用することを提案する。

また、本アプローチによる組み込みシステム開発の事例として、ATM¹（現金自動取引装置）開発について紹介する。さらに、ATM 開発を通じての適用評価を行い、本アプローチの効果と課題について考察する。

2. フレームワーク技術の現状

再利用技術としてフレームワークが適用されてきているが、一方で課題があることもわかっている。以下に、効果と課題を簡単に整理する。

2.1. 効果

フレームワークの適用により、再利用される単位はクラス単体ではなく、ソフトウェアアーキテクチャ（システムの基本的な構造）を含んだクラス構造となるので、再利用範囲が広がる。また、フレームワークによりシステムのソフトウェアアーキテクチャを標準化することができ、機能拡張に柔軟な対応が可能となる。

2.2. 課題

フレームワークの適用により効果が期待できる一方、フレームワークの構築プロセスや利用プロセスについては議論の余地がある。

● 構築プロセス

フレームワークを構築するためには、対象となるシステムの特徴や機能を十分吟味する必要がある。また、完成度の高いフレームワークを一度に構築することは容易でない。

現在、いくつかのフレームワーク構築手法[8][9]が提案されているが、これらはドメイン分析の視点からドメインの共通部分を示すフローズンスポットと可変部分を示すホットスポット[8]を特定し、再利用性あるいは拡張性に重点をおいた手法である。しかし、システムによっては、再利用性や拡張性だけでなくリアルタイム性といった実現制約を持つ要件を考慮するが、これに対処する手法の議論も必要である。

● 利用プロセス

フレームワークの利用者は、フレームワーク内のクラスの知識を必要とし、利用方法を熟知する必要がある。したがって、質の高いフレームワークを構築したとしても、フレームワークを正しく利用しなければ効果は望めない。

現在、これを解決する手段として提案されている方法は、ドキュメントを提示するもの[10]、あるいはドキュメントの参照手段をツール化するもの[11]であり、これらはプログラミング作業を直接支援するものではない。これらの方法はフレームワーク利用者のスキルに依存しており、利用方法を誤る危険性を含んでいる。

3. フレームワークと製品特化 CASE によるアプローチ

2.2節で述べたフレームワークの課題を解決するために、フレームワークのインクリメンタルな構築プロセス、およびフレームワークの製品特化 CASE による利用プロセスを提案する。本アプローチは、フレームワークと製品特化 CASE とを併用し、相乗効果をねらいとする。

¹ Automatic Teller Machine

ASE とを併用し、相乗効果をねらいとする。

3.1. フレームワーク構築プロセス

より質の高いフレームワークを実現するためには、フレームワークの構造や機能を検証する作業が必要である。現実的には、類似システムを何度も開発することでフレームワークを洗練していくことになる。また、システムによっては再利用性や拡張性だけでなくリアルタイム性をシステム要件とする場合もある。

そこで、フレームワークにレイヤ構造を与え、重点を置くレイヤを移しながら徐々に全体を構築していくプロセスを考える。具体的には、フレームワークを構築する専任のグループを設置した上で構築フェーズを三つに分割し、各フェーズでフレームワークの構造や機能を段階的に検証する（図 1）。本アプローチでは、フレームワークをインフラ（OS やミドルウェアなど）よりのレイヤからドメインよりのレイヤまで三つに分割する。複数のフェーズに分ける理由は、単にフレームワークの構成要素を段階的に構築するというだけでなく、システム要件（再利用性やリアルタイム性など）を満足するためにフレームワークを後で調整するための代替案をあらかじめ検討しておくことが重要であるからである。基本的に三つのフェーズを順に進めるが、必要に応じて後戻りをすることもある。また、同じフェーズを繰り返し実施することもある。これらについては、対象システムごとに調整する。

(1) インフラレイヤ構築フェーズ

インフラに依存した機能を検討するフェーズである。インフラで用意された API²を隠蔽する機能などが考えられる。ここでは、この機能を含むフレームワークをインフラフレームワークと呼ぶ。

(2) ジェネリックレイヤ構築フェーズ

ドメインに関わらず汎用的に用いる機能を実現するフェーズである。オブジェクト間のメッセージ通信を実現する機能などがこれに相当する。ここでは、この機能を含むフレームワークをジェネリックフレームワークと呼ぶ。

(3) ドメインレイヤ構築フェーズ

ドメインに依存した機能を実現するフェーズである。仕様変更柔軟な対処が可能な構造を検討する。すなわち、フローズンスポットとホットスポット[8]を特定することである。ここでは、この機能を含むフレームワークをドメインフレームワークと呼ぶ。

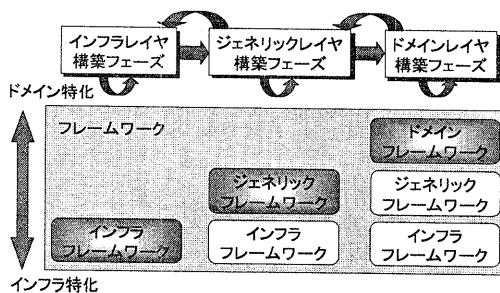


図1 フレームワークのインクリメンタルな構築プロセス

3.2. フレームワーク利用プロセス

フレームワークはシステムの基本的な構造を与える再利用部品を提供しており、システムを開発する際にはフレームワークを利用して必要な機能を付け加えていく。しかし、質の高いフレームワークを構築したとしても、開発者全員が正しく理解してなかったり、誤った利用をする可能性がある。

そこで、フレームワークを正しく利用するための手段として、製品特化 CASE を相補的に導入する（図 2）。製品特化 CASE は、対象となるシステムに特化した CASE ツールであり、仕様を記述するエディタ、仕様記述からソースコードを自動生成するジェネレータから構成される。製品特化 CASE は、エディタに入力し

² Application Programming Interface

た分析・設計モデルの仕様記述からソースコードを自動生成する。このとき、製品特化 CASE は、フレームワーク中のクラスを再利用部品として取り込み、仕様記述を反映したソースコードを生成する。したがって、フレームワークを効率的に正しく利用できることになる。

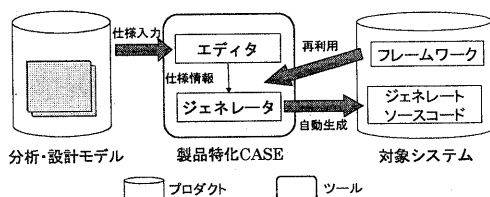


図2 フレームワークの製品特化 CASE による利用プロセス

4. ATM 開発への適用

3章で示したアプローチを適用した ATM（現金自動取引装置）開発事例について示す。

まず、対象システムについて簡単に説明し、システム全体の開発プロセス、フレームワークの構築および利用プロセスについて順に示し、最後に適用評価を述べる。

4.1. 対象システム

ATM は、銀行などの金融機関の各支店に複数台設置しており、ホストと通信しながら顧客に対し取引処理を行うシステムである（図3）。

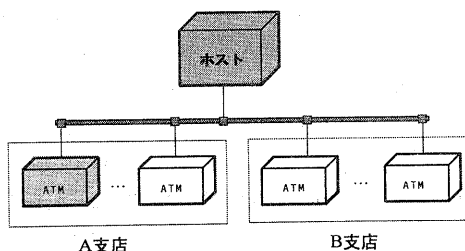


図3 ATM（現金自動取引装置）

ATM は組み込みシステムの一つとして位置づけられる。下記に、システムを開発する上で解決しなければならない要件を挙げる。

● 並行性

複数のハードウェアユニット（カードユニットなど）を並行に駆動させながらサービス（預け入れなど）を進めるため、システムを並行に動作させることが必要である。

● リアルタイム性

ATM を利用する顧客からの指示にしたがって、要求の時間内にサービスを終了することが求められる。

● 再利用性

金融機関ごとに別々に対応すると開発コストが増大してしまうので、金融機関によらず共通に再利用可能な部品を提供することが望まれる。

● 拡張性

同じ金融機関でも仕様変更や追加が頻繁であり、柔軟かつ迅速に対応することが必須である。

● 大規模性

ATM の信頼性を確保するために、ハードウェアユニットが異常になった場合の対応も行う必要がある。したがって、どんなタイミングで何が起こってもサービスを進められるようにするため、要求仕様が複雑で大規模である。

これらの性質を持った ATM のシステムとしての振る舞いは、主に状態遷移図を使って記述している。状態遷移図は数千枚にも及び、ソースコードのサイズは約 100 万ステップを越える。ATM システムは、一般的な組み込みシステムの性質に加えて、非常に複雑でありかつ大規模であることが特徴である。なお、ATM システムは汎用 OS 上に C++ 言語を用いて実装された。

4.2. システム全体の開発プロセス

3章で示したアプローチを適用した。すなわち、並行動作するオブジェクトや、金融機関の

共通仕様などを実現した ATM フレームワークと、これを利用してシステム特化の仕様を実現する ATM 製品特化 CASE とを相補的に利用した。オブジェクト指向方法論としては、Booch 法[12]をもとに独自にカスタマイズした。

図 4では、ATM システムにおける開発プロセスを示している。以下に、分析、設計、実装工程での作業を簡単にまとめる。

● 分析

ATM ドメインを分析し、オブジェクトを列挙した。次に、オブジェクトを四つのカテゴリ（サービス、媒体、ユニット、GUI）に分類し、カテゴリ間およびオブジェクト間の関係を定義した。この作業は、少数のオブジェクト指向スペシャリストとドメインエキスパートのみで行った。

● 設計

フレームワークおよび製品特化 CASE のジェネレータを設計するグループと、製品特化 CASE を利用してシステムを設計する二つのグループに分割した。

フレームワークの設計は、オブジェクト指向スペシャリストとドメイン分析スペシャリストの数人で行った。フレームワークは、ATM システムの一部の機能をプロトタイプとして三回繰り返し開発しながら洗練していった。詳しくは、4.3.1項で述べる。

一方、システムの設計は、各カテゴリ内のオブジェクトの振る舞いを主に状態遷移図を使って進め、製品特化 CASE のエディタに入力していった。詳しくは、4.3.2項で述べる。

● 実装

フレームワークと製品特化 CASE のジェネレータの開発は、プロトタイプ開発を行ったメンバを中心に行った。一方、他の開発メンバは製品特化 CASE のエディタに入力した仕様記述からソースコードを自動生成した。

実装は C++言語を利用し、フレームワークのソースコードと製品特化 CASE によって自

動生成されたソースコードとを組み合わせることでシステム全体を完成させた。

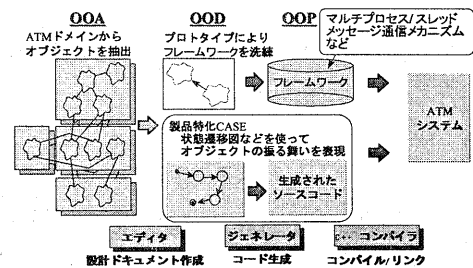


図4 ATM システムの開発プロセス

4.3. フレームワーク構築および利用プロセス

ATM 開発でのフレームワーク構築および利用プロセスの概略について触れる。

4.3.1. フレームワーク構築プロセス

フレームワークの構築を行うグループを設け、3.1節で示した開発プロセスのうち、インフラレイヤ構築フェーズとジェネリックレイヤ構築フェーズをプロトタイピングにより実践した。ドメインレイヤ構築フェーズについては、システムを一つ開発した後に実施することにした。

ATM 開発では、プロトタイピングを三回実施しフレームワークを洗練していった（図5）。以下に、三回のプロトタイプ開発でのフレームワーク構築フェーズの対応と作業内容を示す。

(1) プロトタイプ開発フェーズ1

インフラレイヤ構築フェーズとして、マルチプロセス・スレッド機能など並行環境を利用するための構造を検討した。

(2) プロトタイプ開発フェーズ2

ジェネリックレイヤ構築フェーズとして、オブジェクト間のメッセージ通信機能や状態遷移の仕組みを検討した。

(3) プロトタイプ開発フェーズ3

インフラレイヤおよびジェネリックレイヤ再構築フェーズとして実施した。実機での動作お

および性能確認の意味で、インフラおよびジェネリックレイヤを再構築するフェーズを設けた。

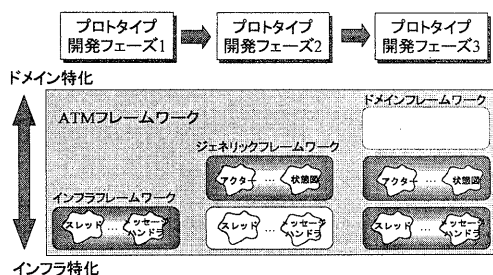


図5 ATM フレームワーク構築プロセス

4.3.2. フレームワーク利用プロセス

3.2節で述べたようにフレームワークの利用ルールを内包した製品特化 CASE を併用した。ATM 開発で導入した製品特化 CASE は、複数のツールから構成される。例えば、GUI 部品を配置するもの、レシート印字書式を定義するもの、ATM で提供するサービスの振る舞いを状態遷移図で定義するものなどがある(図6)。

製品特化 CASE によりシステム開発者からフレームワーク内部の構造や振る舞いを隠蔽することができるので、システム開発者は製品特化 CASE のエディタ上で利用する仕様記述言語の修得に注力した。エディタに入力した仕様記述をもとに、製品特化 CASE のジェネレータによりフレームワーク中の適切なクラスを継承してシステムに特化した機能を自動生成した。

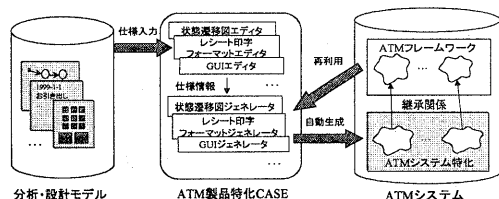


図6 ATM フレームワークの利用プロセス

4.4. 適用評価

ATM のシステム要件に対する評価とともに、フレームワークのインクリメンタルな構築プロセス、およびフレームワークの製品特化 CASE による利用プロセスに対する評価を述べる。現在、本アプローチを用い既に数ユーザ(金融機関)に対して製品出荷を完了している。

4.4.1. システム要件に対する評価

フレームワークと製品特化 CASE を利用したアプローチを、4.1節で示したシステム要件に応じて評価する。

● 並行性

マルチプロセス・スレッド機能をフレームワークに装備した。システム内のオブジェクトは並行に動作することができ、ハードウェアユニットを並行に動作させることができた。

● リアルタイム性

当初開発したフレームワークでは性能が達成できず、フレームワークのチューニングを行った。主な修正は、(1)スレッドの限定利用、(2)共有資源の削減、の二つである。性能実験により、スレッドを増やすことがかえってオーバーヘッドが生じることや、共有資源の排他制御により性能を劣化させていることが判明した。チューニングは、フレームワーク中のインフラレイヤのクラスを修正することで実現でき、製品特化 CASE のジェネレータへの影響はなかった。

● 再利用性

フレームワークは、マルチプロセス・スレッド環境や状態遷移のメカニズムなどインフラレイヤおよびジェネリックレイヤとして位置づけられる部分と、ATM ドメインの基本機能を実現したドメインレイヤに位置づけられる部分とがある。いくつかの金融機関向けの ATM 開発を通じて、前者は再利用できたが、後者については洗練の余地が残っている。

● 拡張性

一つの金融機関での仕様変更は、GUI 仕様の変更はもちろんのこと、サービスの処理手順の変更や、サービスの種類の増加がある。

サービスに関わる仕様変更に対しては、次のような方法で柔軟に対応できた。まず、サービスの処理手順の変更については、サービスオブジェクト（一つのサービスを一つのオブジェクトとして定義）の振る舞いの変更としてとらえることができ、サービスオブジェクトの状態遷移図を変更することで対応した。また、サービスの種類の増加については、サービスオブジェクトそのものを追加することで対応できた。これらの作業は、すべてフレームワークの利用ルールを内包した製品特化 CASE を併用することで容易に拡張が可能であった。

● 大規模性

初回開発では、フレームワークと製品特化 CASE の開発自体も含まれるため従来と同程度の開発工数であったが、2 回目以降の開発では、従来と比べて大幅に削減できた。

初回開発では、システム全体のうち 96% のソースコード（ソースコード行数）を製品特化 CASE により自動生成した。フレームワークの利用ルールを製品特化 CASE で厳密に規定することで、プログラミングの誤りを軽減し大量の仕様を効率的に実装することができた。

4.4.2. フレームワーク構築および利用プロセスに対する評価

ATM 開発でのフレームワーク構築プロセスおよび利用プロセスに対する評価について示す。

● フレームワーク構築プロセス

インフラレイヤ構築フェーズを設けたので、マルチプロセス・スレッド関連の設計に関するテクニックについて十分な検討ができた。

また、ジェネリックレイヤ構築フェーズを設けたことで、メッセージ通信の仕組みや状態遷移の仕組みなどを検討できただけでなく、製品特化 CASE で規定すべきフレームワーク利

用のルールに関わる検証も同時に実施できた。

さらに、ATM 開発ではインフラレイヤとジェネリックレイヤを再構築するフェーズを設けた。このフェーズでは、実機で動作するレベルまでプロトタイプを開発したので、リアルタイム性について評価できた。その結果、リアルタイム性の観点からフレームワーク設計に関しさまざまな代替案を検討することができた。

一方、ドメインレイヤ構築フェーズについては、ATM 開発では明確に設けなかった。この作業は、フレームワーク開発の中でも最も困難であると思われる。ATM 開発では、いくつかの ATM システムの開発と並行してドメインレイヤ構築フェーズを設ける方針を採った。現状では、ドメインレイヤフレームワークは十分洗練したものではなく、今後の他の金融機関への適用の際に見直す必要がある。ただし、オブジェクトの静的な構造を修正する可能性はほとんどないと思われ、オブジェクトの振る舞いの見直しを行うことになる。オブジェクトの振る舞いを見直したとしても、製品特化 CASE によりソースコードを自動生成することができるので、製品特化 CASE によりドメインフレームワークを構築することができる。

● フレームワーク利用プロセス

フレームワークの利用ルールを規定した製品特化 CASE を併用しているので、システム開発者はフレームワークを十分理解していなくてもシステムを開発できた。初回開発では、システム全体のうち 96% のソースコードを製品特化 CASE により自動生成した。また、数ユーザの開発を並行して実施することができ、効率的に開発できた。

一方で、フレームワークの修正に伴いソースコードの自動生成ルールが変更された場合には、製品特化 CASE のジェネレータを修正することが考えられる。ATM 開発では、フレームワークとジェネレータを同一の開発者が担当したが、開発者の負荷を考慮して別々にする場合が

考えられる。この場合、ジェネレータ開発者に対しては、フレームワークの仕様や設計モデルを容易に理解できるような仕組みが必要である。一方、システム開発者に対しては、製品特化 CASE を利用するのでこの仕組みは必要ない。

5. おわりに

本稿では、オブジェクト指向フレームワークのインクリメンタルな開発プロセス、フレームワークの製品特化 CASE による利用プロセスについて述べた。また、本アプローチを ATM 開発へ適用し、効果と課題について考察した。

フレームワークの構築プロセスについては、再利用性や拡張性だけでなくリアルタイム性などの実現制約を持つシステムに対して、レイヤ構造を与え段階的に構造や機能を検証することは効果的であった。一方で、個々のフェーズのより詳細化したプロセスの検討が今後必要である。これらのプロセスの詳細化については、システム要件を分類し、複数のステップに分けて代替案を提示する手法を検討中である。

また、フレームワークの製品特化 CASE による利用プロセスは、ATM のような大規模なシステム開発においては有効であった。一方で、フレームワークを効率的に利用する手段として、すべてのシステム開発に製品特化 CASE が有効というわけではないと考えている。製品特化 CASE に向き不向きのシステムがあり、導入指針を提示することが必要である。

今後は、フレームワークの構築プロセスおよび利用プロセスを改良するとともに、特に組み込みシステムに対して適用を進め、評価を継続していく。

参考文献

- [1] Rogers, G. F.: Framework-Based Software Development in C++, Prentice-Hall, 1997.
- [2] Schmucker, K. J.: Object-Oriented Programming for the Macintosh, Hayden Book Company, 1986.
- [3] Linton, M. A. et al.: InterViews: A C++ Graphical Interface Toolkit, Technical Report CSL-TR-88-358, Stanford University, 1988.
- [4] Lewis, T. et al.: Object-Oriented Application Framework, Manning Publications, 1993.
- [5] Franek, B. et al.: SMI++ object oriented framework for designing and implementing distributed control systems, IEEE Transaction on Nuclear Science, Vol.45, No.4, pp.1946-1950, 1998.
- [6] Alfredsen, K. et al.: An object-oriented framework for water resource planning, Proceedings of the Thirty-First Hawaii International Conference on System Science (Cat. No.98 TB100216), Vol.7, pp.441-450, 1998.
- [7] 荒野他：ネットワーク管理フレームワークとその開発に関する一考察，情報処理学会論文誌，Vol.38, No.6, pp.1182-1191, 1997.
- [8] Pree, W.著，佐藤他訳：デザインパターンプログラミング，トッパン，1996.
- [9] 名取他：データ中心アプローチとユースケースに基づくオブジェクト指向フレームワーク構築手法，Vol.38, No.3, pp.634-656, 1997.
- [10] Johnson R. E.: Documenting frameworks using patterns, In Proceedings of OOPSLA'92, 1992.
- [11] e Silva, R. P. et al.: Tool support for helping the use of frameworks, Proceedings SCCC'98. 18th International Conference of the Chilean Society of Computer Science, pp.192-201, 1998.
- [12] Booch, G. : Object-Oriented Analysis and Design with Applications, second edition, Addison Wesley, 1997.