4P－01　　コンポーネントベース・フレームワーク技術(*C Solution* APF)
全体構想とアーキテクチャ

斉藤 悦生、関　武夫、吉田 和樹、山田　広佳、渡瀬　慎一郎
(株)東芝

1 再利用技術としてのフレームワーク

近年のアプリケーションシステム開発は、高機能なアプリケーションを短納期で開発する事を余儀なくされている一方、様々なISVより提供される高機能なMWの出現とその頻繁なバージョンアップに迫られ、実効的なソフトウェア生産技術の必要性が差し迫った課題としてクローズアップされている。特に９０年代から本格的に導入が始まったオブジェクト指向技術（OO）は、ソフトウェア再利用技術の星として期待を集めたOOによる再利用性として近年デザインパターンに代表されるパターン技術が注目されるが、実際に利用可能な部品としてはフレームワークとして様々なベンダーより提供されている。

フレームワークは多くの場合百～数百のクラスから成るクラスライブラリとして提供され、利用する場合は継承を用いることが多い。この様なクラス構造全体が見え、継承を使って利用する形のフレームワークはwhitebox型フレームワークと呼ばれる。ところがこのようなwhitebox型フレームワークでは、フレームワークの理解コストが掛かる、継承のネスト構造が深くなり親持つ不具合が子へ影響を及ぼす、実行モジュールが大きくなりがちで重くなる、などの問題がクローズアップされてきた。

一方、内部構造を見せないコンポーネントを組み合わせてフレームワークを構築する試みもなされてきて、継承を多用する問題を回避することができるなどの点でwhitebox型フレームワークより優れていると考えられている。しかし、blackbox型フレームワークはblackboxゆえに内部に手を入れる事が出来ないため、アプリケーション固有の仕様を組み込む事が難しいという欠点を有している。

アプリケーション開発エンジニアの観点でフレームワークの有るべき姿をまとめるなら、次の３点に集約できると考えられる。

①理解容易なフレームワーク
②アプリケーション固有の仕様の容易な実現
③品質が安定した構造

我々は以上の要求事項を満足するためのフレームワークとしてblackbox型フレームワークに着目し、従来の問題点であったカスタマイズの難しさを打開する手法を織り込んだ方式を開発した。本稿ではこの新しいフレームワークについて紹介する。

2 コンポーネントベース・フレームワーク技術(*C Solution* APF)

C Solution APF（以下APF）は、上述したフレームワークの要件を実現するために、以下のようなコンセプトに基づき開発を行った。

①最小限のコンポーネントの組合せでフレームワークを構築できる仕組みを提供する
②カスタマイズ可能なコンポーネントにより、アプリケーション固有の仕様を組み込む事ができるようにする

もしこのコンセプトに基づいたフレームワークが構築できれば、①少ないコンポーネントしか用いないため理解のためのコストが低減できる、②カスタマ

Component-based Framework Technology (***C Solution*** APF) : Concept and Architecture

Etsuo Saito
Toshiba Corporation
3-22, Katamachi Fuchu-shi, Tokyo 183-8512 Japan

イズ可能であることからアプリケーション固有の仕様が容易に実現できる、③ blackbox であることからフレームワーク利用者による品質の劣化は回避出来る、ので、理想的なフレームワークが実現できると考えられる。

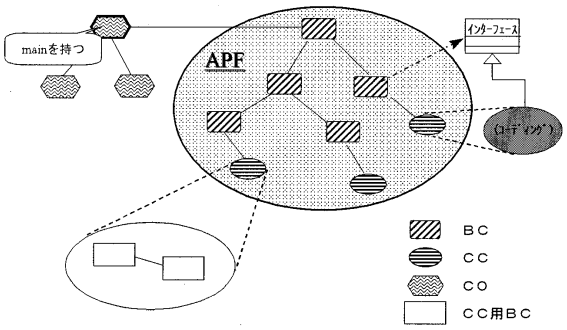

図１　APF の基本構造

APF のコンポーネント構造は、ベースコンポーネント（以下 BC）とカスタムコンポーネント（以下 CC）からなる（図１）。BC はアプリケーション骨格を実現する、組合せ可能なコンポーネントで後述する APF 開発環境の GUI 上で drag&drop で組み上げる事ができる。BC には CC を組み込むためのインターフェースが用意されていて、ユーザデファインの CC を組み込む事が可能になっている。CC 用の I/F は個々の BC 毎に用意されている。CC は BC を修飾するためのコンポーネントであり、これらを組み合わせて使う事は想定されていない。CC の BC への組み込みも APF 開発環境上で実現できる。アプリケーション開発においては、APF 開発環境で BC の組み合わせと CC の組み込みと BC のパラメータ設定等を行い、結果をソースコード自動生成機能により Java のコードに落とし、APF ではカバーしきれない部分をカスタムオブジェクトとして Java で開発して、これと APF とを組み合わせて実行可能なオブジェクトを得る。

APF 開発環境は APF の開発を支援するための環境で GUI 上でコンポーネントを組み合わせ、そのコードを自動生成することができるものである。CASE ツールなどとは異なり、実行可能なコードが生成できることが特徴である。また、シミュレーション機能を持っており、GUI 上で組み上げたフレームワークを APF 環境上で動作確認させる事も可能である。

現在 APF では業務トランザクション処理用（Accounts）と帳票編集処理用（帳票編集 APF）の２種類のコンポーネント群の開発を完了しており、実際にこれら２つの APF を適用した最初のアプリケーションも開発を完了した。

現在開発中の APF は比較的粒度が小さく、その点では広範囲に利用可能なフレームワークになると考えられる（機能的 APF と呼ぶ）が、再利用性を高めるためにはさらに粒度の大きな、ターゲットを業種や業務に絞った APF 開発が必要となってくると思われる（ドメイン APF と呼ぶ）。しかし、粒度が大きくなっても BC の機能やサイズが大きくなったり、CC 用 I/F が増えたりすることはあっても BC と CC でフレームワークを実現するという根本的なアーキテクチャは不変であると思われる。今後は機能的 APF の拡充とともにドメイン APF の実現を図っていきたい。

3 まとめと今後の課題

blackbox 型フレームワークである ***C Solution*** APF を提案した。 ***C Solution*** APF では少ないコンポーネントを組み合わせてユーザの要求するフレームワークを構築でき、さらに CC を組み込む事でアプリケーション固有の仕様を実現できる。これにより生産性が高まることが期待される。

今後の課題として、①ドメイン APF の機能的 APF の組み合わせでの実現可能性の検証、②アプリケーション開発方法論、特に業界標準になっている UML やドメイン分析手法との整合性の検証とそれを取り込んだ形での方法論の構築、③他のコンポーネントの連携やラッピングの実現、④コンポーネント開発手法の確立などが考えられる。これらの課題をクリアして再利用性の高いフレームワークの実現を図りたい。