4W-06

業務アプリケーションの開発効率化手法（２）

－フレームワークを適用したコンポーネントベースの開発環境－

馬場健治　寺濱幸徳　北川健二　染谷治志

(株)日立製作所 システム開発研究所

１．はじめに

昨今の規制緩和の流れに伴い、様々な業種において、新たなサービスを迅速に提供していくことが、経営戦略上の重要なポイントである。この実現のためには、業務アプリケーション開発を効率化し、開発期間を短縮することが必要である。

本稿では、既に何らかの業務アプリケーションが稼動しているシステムに対して、新たな業務アプリケーションを追加する場合を想定し、フレームワークを用いたコンポーネントベースの業務アプリケーション開発効率化手法について述べる。

２．フレームワークの定義および適用方法

フレームワークの定義は様々であるが、ここでは、コンポーネントベース開発において有効なフレームワークについて述べる。

２．１　現状と問題点

フレームワークとは、特定の目的のために再利用できるように設計されたモジュール群のことである[1]。対象とするコンポーネントベース開発では、通常、開発に必要と思われるコンポーネント群を事前に用意するが、これもフレームワークである。

一般的にフレームワークの設計は、実現すべき業務とプログラミング開発の両方に関し相当の知識を要求される上、以後に開発されるであろう業務アプリケーションを予測する必要があるため、非常に困難である。通常、時間が経つにつれ設計を見直す必要が出てくる。

Efficient Development of Application for Finance
Kenji Baba, Yukinori Terahama, Kenji Kitagawa, Harushi Someya
Systems Development Laboratory, Hitachi, Ltd.

２．２　既存業務アプリケーションのフレームワーク化

あるシステム上で稼働する既存の業務アプリケーションには、そのシステムの特徴や、業務アプリケーション全体に関わる情報など、新たに開発する業務アプリケーションで再利用できる情報が豊富に存在する。そこで、フレームワーク構築の労力を低減するために、既に開発した業務アプリケーションをそのままフレームワークとして再利用できるような環境を構築する。

２．３　コンポーネントベース開発への適用方法

コンポーネントベースの開発において、既存業務アプリケーションの様々な情報をフレームワークとして提供するためには、再利用単位の多階層化が必要である。単一部品の再利用から、関連付けられた複数の部品をまとめた部品群まで、様々なレベルで再利用を可能とする。金額を入力すると消費税計算した結果が自動出力されるような機能の再利用や、画面テンプレートとしての画面ごとの再利用や、ＩＤ認証処理といった処理単位まで、幅広い再利用を可能とする。

また、コンポーネントベース開発に特有な、コンポーネント間の接続作業や、仕様変更への対応など保守作業の効率化も、フレームワーク構築の要件となる。

３．フレームワーク構築のためのコンポーネント設計

既存業務アプリケーションをより効果的にフレームワーク化するためのコンポーネント設計の考え方について以下に述べる。

3．1　間接指定

仕様変更に伴うプログラムの変更は、可能な限り局所化すべきである。以下のような項目に関し、具体的な定義内容を別の場所に保存し、それを間接指定すれば、変更作業が効率化できる。

（1）プロパティ値

コンポーネントが持つプロパティを間接指定する。これにより、再利用されて複数存在するコンポーネントの仕様が変更されても問題がない。

（2）データ型

データの型情報を間接指定する。例えば、年号というデータが２桁から４桁に仕様変更された場合、年号を表示するためのコンポーネントに対し、年号を表示することだけを定義し、年号の桁数情報などは別に定義しておく。これにより、仕様変更の際の修正個所が局所化される。

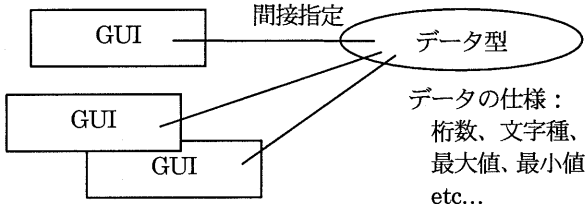

図１　データ型の間接指定

3．2　抽象表現

表現を抽象化しておけば、再利用できる可能性が大きくなる。例えば、画面遷移処理を再利用する際に、「画面Ａから画面Ｂへの遷移する」という具体的な定義がされていると画面が異なれば全く再利用できないが、「画面遷移をする」という抽象的な定義がされていれば再利用可能である。

また、コンポーネント間接続の自動化にも有効である。例えば、ボタンを押すことにより送信コンポーネントが送信を行う処理を再利用したいとする。既存業務アプリケーションと新規開発アプリケーションで、利用する送信コンポーネントの（送信データの集め方等の）処理が異なった場合でも、ボタンの押下に対し、「送信コンポーネントを起動する」という抽象表現をしておけば、新規開発画面上で、このボタンから送信コンポーネントに自動接続できる。

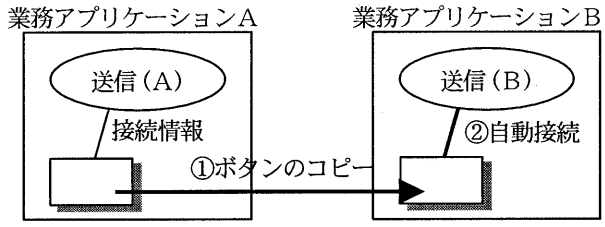

図２　抽象表現による自動接続

3．3　隠蔽と制限

プログラミングスキルの低い開発者が開発する場合や、開発しようとする業務アプリケーションが既に開発したものに類似している場合などは、詳細な定義は不要である。不要な機能を隠蔽、あるいは制限すれば、必要な開発作業のみに集中できる。

（1）プロパティ

再利用する際に変更が不要であったり、変更されると困るプロパティについては隠蔽する。例えば、画面テンプレートにおいて、タイトル文字列は変更できるが、タイトル文字フォントは変更できないようにプロパティを隠蔽する。これにより、画面デザインの統一感が保てる。また、隠蔽でなく制限することにより、自由度を持たせながらも整合性をとることも考えられる。

（2）メソッド、イベント

複数のコンポーネントからなるコンポーネント群を再利用する場合、既にメソッドやイベントによる動作が定義されている場合がある。その際は、不要な物として隠蔽する。

4．おわりに

本稿では、コンポーネントベース開発における、フレームワークの定義および適用方法と、フレームワーク構築のためのコンポーネント設計の考え方について述べた。今後は、上記特徴を備えた銀行営業店業務を対象としたコンポーネントを構築し、その有効性を検証する。

参考文献

[1] 戸松豊和：”増補改訂 Java プログラムデザイン”，
ソフトバンク株式会社 (1998)