1Q−05

高品質アプリケーションフレームワークの構築のための上位フレームワーク *

細谷　竜一†
（株）東芝　SI技術開発センター§

1 はじめに

アプリケーションフレームワーク（以下フレームワーク）は、アプリケーションを構成するコンポーネントの種類とそれらの関連付けを規定するオブジェクト指向のソフトウェア設計及びプログラムである。アプリケーション開発者は特定のフレームワークを利用し、規定に沿って自前のコンポーネントを定義することでアプリケーションを作成できるので、開発における設計の負荷を大幅に減らせる。そのため、フレームワークはしばしば「半完成のアプリケーション」であると説明される。従来、こうした考え方と、「デザインパターン」[1]に示されるフレームワークの基本的設計要素は知られていた。しかし、良いフレームワークの開発は多大なコストを要求する[2]。また、実際のアプリケーションの開発では、複数のフレームワークを組み合わせて運用しなければならない場合が多いが[3]、相互運用を前提としたフレームワークの具体的な設計方法は一般に論じられていない。

そこで本稿では、「マイクロコンテナ」と呼ぶ上位フレームワークを用いて、品質の高いアプリケーションフレームワークを構築する方法を紹介する。ここで品質が高いフレームワークとは、次の特性を持つフレームワークであるとする：

- フレームワーク間の相互運用性が高い。
- 拡張性が高い。
- 付加価値が高い（ロギング、視覚化ツールなどのサポートがあるなど）。

2 マイクロコンテナの構成

「マイクロコンテナ（microcontainer）」は、高品質アプリケーションフレームワークに次の機構を提供するための上位フレームワークである：

a. オブジェクト間構造機構
b. 処理実行機構
c. 機能拡張機構
d. エラー処理機構
e. イベント通知機構
f. ロギング機構
g. 視覚化機構
h. コンフィギュレーション機構

上の a〜h それぞれの機構は、いくつかのオブジェクトからなるパッケージとして提供される。図 1にパッケージ間の関係を示す。

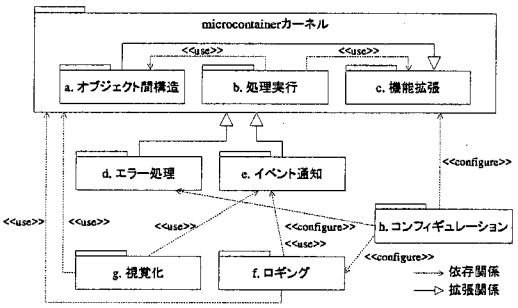

図 1：マイクロコンテナの構成

3 マイクロコンテナの利用

フレームワーク開発者は、開発対象となるフレームワークの方針に従って上の a〜h それぞれの機構を拡張する。つまり、a〜h それぞれに含まれるオブジェクトを直接あるいは拡張しながら組み合わせることで必要なコンポーネントを定義し、フレームワークを構築していく。こうして完成するフレームワークは上の a〜h の機構を備えた高品質アプリケーションフレームワークである。また、このフレームワークを利用して作られるアプリケーションの開発では、機能拡張機構やエラー処理機構などに従ってアプリケーション固有のコンポーネントを追加定義する。そして、実行時に必要となるコンポーネント同士の接続をコンフィギュレーションスクリプトとして記述する。コンフィギュレーション機構は、実行時に、コンポーネント定義とコンフィギュレーションスクリプトに従ってコンポーネントを初期化・

---

* A higher-level framework for building high-quality application frameworks
† Hosoya Ryuichi
§ Systems Integration Technology Center, Toshiba Corporation

接続し、アプリケーションが要求するオブジェクト間構造を構築する。また、視覚化機構は、構築されたオブジェクト間構造を視覚化し、処理実行のトレースを可能にする。図 2にマイクロコンテナ、フレームワーク及びアプリケーションの間の関係を示す。

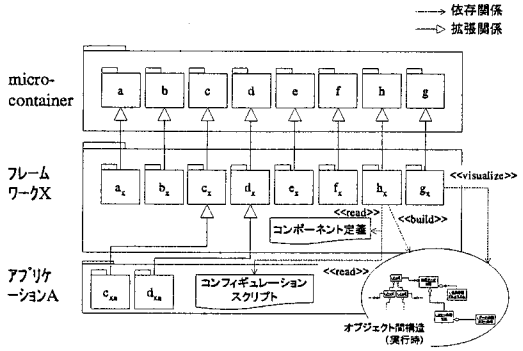

図 2：マイクロコンテナの利用

4. マイクロコンテナの効果

マイクロコンテナが前出の a～h 各機構間の基本的な協調動作を規定しているため、フレームワーク開発者は、コンポーネントの設計・実装に集中できる。さらに、マイクロコンテナはロギング機構、視覚化機構及びコンフィギュレーション機構それぞれの基本的な実装を既に持っているため、フレームワークは必要最小限の拡張でそれらの機能をアプリケーション開発者に提供できる。

視覚化機構は、実行中のフレームワークの内部構成を視覚化しながら、アプリケーションとフレームワークをまたがった実行の様子をシームレスにトレースする。これによって、フレームワークを利用した開発において一般に見られる実行解析の困難が解消され、開発の効率が改善される。図 3にマイクロコンテナに基づいた実際の視覚化機構実装の例を示す。

コンフィギュレーション機構は、フレームワークの構成の柔軟な組み替えを可能にする。通常、フレームワークは繰り返し再利用されながら継続的に改良されていくものである。この機構はこうしたフレームワークの進化の過程をサポートする。

マイクロコンテナに基づくフレームワークは、次の機構によって、相互運用性が高められる：

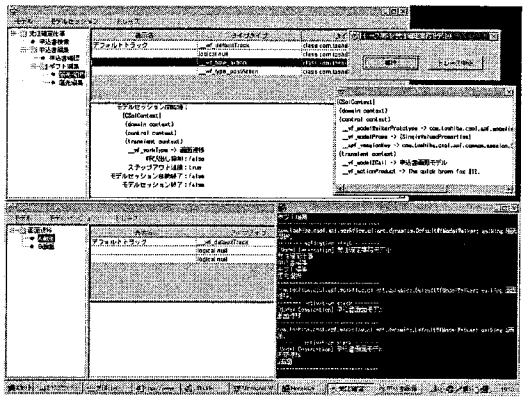

図 3：視覚化機構の例

- オブジェクト間構造機構：接続用オブジェクトを経由してコンポーネント間を接続することで、異なるフレームワーク間で相互にコンポーネントを接続し、オブジェクト間構造を構成できる。視覚化機構は相互接続されたフレームワーク間で動作する。
- 機能拡張機構：一つのコンポーネントは、標準化されたプラグインインタフェースを通じて設定される複数の拡張機能オブジェクトの組み合わせとして定義される。これは、異なるフレームワーク間で同じ拡張機能オブジェクトを共有できることを意味する。

5 課題

マイクロコンテナはJava™で実装され、事務処理系アプリケーションフレームワークの構築のために既に利用されているが、組み込み機器向けフレームワークにおける利用の可能性についても調査していきたい。

マイクロコンテナに基づいたフレームワーク間の相互運用性の検証はまだ不十分である。今後設計上の課題を調査し、改良していきたい。

参考文献

[1] E. Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software.* USA: Addison-Wesley. 1995. 本位田ら（監訳）. *オブジェクト指向における再利用のためのデザインパターン.* 東京: ソフトバンクパブリッシング. 1995.

[2] D. Roberts et al. *Patterns for Evolving Frameworks.* In Pattern Languages of Program Design 3. USA: Addison-Wesley. 1997. 細谷ら（監訳）. *フレームワークの進化のためのパターン.* プログラムデザインのためのパターン言語：Pattern Languages of Program Design 選集(収録). 東京: ソフトバンクパブリッシング. 2001.

[3] M. Fayad et al. *Building Application Frameworks.* USA: Addison-Wesley. 1999.