1W-06

~~5W-06~~

フレームワークとCASEツールによる制御ソフト開発方式*

三木 正章, 早瀬 健夫, 松本 一教
株式会社 東芝 情報・社会システム社 SI技術開発センター

1. はじめに

近年、再利用技術としてオブジェクト指向フレームワーク（以下フレームワーク）が注目されてきている。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[1]である。また、従来よりよく知られる再利用技術として、CASEツール（以下CASE）を用いた方法がある。CASEは、形式的仕様やプリミティブなプログラム部品を元にソースコードを自動生成する[2]。これら、フレームワークやCASEの技術をシステム開発に適用することにより、設計・実装にかかる期間の短縮を期待することができる。

しかし、両者の技術に課題が無いわけではない。フレームワークについては、その構築の手順や、利用の効率化に議論の余地が有る。CASEに関しては、再利用性を決める形式的仕様の詳細度やプログラムモジュールの粒度の明確化が難しい。

そこで、本稿では、フレームワーク、CASE両者の技術を、お互いの課題を補う形で利用することを考えた。フレームワークを効率的に正しく適用するため、CASEの適用範囲を明確化し最大限に利点を引き出すため、フレームワークの利用ルールを内包したCASEを使用することを提案する。

また、本アプローチによる制御ソフト開発の事例として、ある情報通信機器の制御ソフト開発について紹介する。

2. フレームワークとCASEの課題

フレームワーク適用により、システムのアーキテクチャを含んだ再利用が可能となる。また、同時にシステムアーキテクチャを標準化することができ、機能拡張に柔軟に対応できる。一方、CASEにより、ソフトウェアの自動生成が可能となる。また、形式的仕様とプログラム部品の再利用により、開発効率を上げることができる。しかし、これら技術により効果が期待できる一方、課題も存在する。

フレームワークの課題

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。

これを解決する手段として、ドキュメントを提示する方法や、ドキュメントの参照手段をツール化する方法があるが、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。

CASEの課題

CASE構築時に注意しなければならない点の一つとして、形式的仕様の詳細度とプログラム部品の粒度がある。

形式的仕様でシステム全体の動作を詳細に記述してしまうと、プログラム言語の代わりに形式的仕様を用いてプログラミングをしている状態になり、開発効率や仕様の再利用性が低下しかねない。また、形式的仕様で記述するシステムの仕様範囲を小さくし、プログラム部品の粒度を大きくなりすぎてしまうと（システムの仕様まで含み過ぎてしまうと）、プログラム部品の再利用性が低下し、CASE導入による利点が失われてしまう。

3. フレームワークとCASEによるアプローチ

2節で述べた課題を解決するために、フレームワー

---

* A Method for Embedded Software Development with Frameworks and CASE Tools
Masaaki Miki, Takeo Hayase, Kazunori Matsumoto
System Integration Technology Center, TOSHIBA Corporation
3-22, Katamachi Fuchu-Shi, Tokyo 183-8512, Japan

クとCASEを組合せた開発プロセスを提案する。本アプローチは、フレームワークとCASEを併用し、相乗効果を狙っている。

フレームワークは、基本的なシステムアーキテクチャを与える再利用部品を提供する。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加していく。このフレームワークで追加／未定義な機能に対し、CASEを適用する。各機能の形式的仕様を記述、フレームワーク、プログラム部品と組合せソースコードを生成する。

フレームワークとCASEを組合せた開発プロセスを図1に示す。まず、分析・設計モデルを形式的仕様にて記述する。分析・設計モデルは、フレームワークが提供するシステムアーキテクチャに従った形で大枠が提供され、追加／未定義な部分をエディタにて追加していく。次に、分析・設計モデルをエディタからジェネレータに送りソースコードを自動生成する。このときCASEは、フレームワーク中のクラスとプログラム部品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。

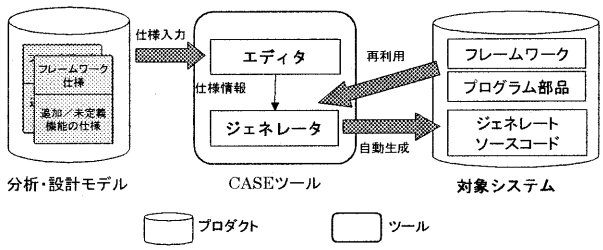

図1　フレームワークとCASEを組合せた開発プロセス

このように、開発プロセスに組み込まれることにより、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。また、CASE側から見れば、フレームワークで枠組みを提供したことにより、仕様記述の詳細度とプログラム部品の粒度を明確に規定することができ、再利用性や開発効率上昇につながる。

4.　情報通信機器への適用

3節で示したアプローチをある情報通信機器の制御ソフト開発に適用した。対象ソフトはホストと通信を行うと供に、単体のみで多様なサービスをユーザに提供する。サービスを迅速に提供するリアルタイム性、組み込み特有のプログラムサイズの制限などを考慮する必要がある。まず、これらの機能・性能要求に対応したフレームワークを構築した後、3節のアプローチを利用した開発プロセスに入った。

開発

CASEによりソフト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、開発者はCASEのエディタ上で利用する仕様記述言語の修得に注力、各機能の仕様を作成した。作成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するとともに、プログラム部品を組み込んで対象機器に特化した機能を自動生成した。

評価

フレームワークの利用ルールを内包したCASE併用により、フレームワークを十分理解していない開発者でもフレームワークの利点を引き出せる開発が行えることを確認した。しかし開発では、ソフトの大部分のソースコードを仕様記述を元に自動生成した。

これは、本開発プロセスが初回適用であったこと、フレームワークの構築と制御ソフトの開発がほぼ並行して行われたことに原因があると思われる。実際、この情報通信機器に対応したフレームワークとプログラム部品を十分供給できなかったため、システムに特化した機能の多くを形式仕様により記述しソースコードを生成することになった。開発への適用が繰り返され、フレームワークやプログラム部品が十分用意されれば、この比率は減少すると思われる。

5.　おわりに

本稿では、オブジェクト指向フレームワークとCASEツールを組合せた開発プロセスを提案した。

今後はフレームワークの構築プロセスにも議論を踏み込むともに、本アプローチを他システムにも適用し、評価・洗練を進めていきたい。

参考文献

[1] Rogers, G. F.: Framework-Based Software Development in C++, Prentice-Hall, 1997.

[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, 情処研究会報告 80-22, 1991