多機種短納期開発へのZIPC適用の有効性

富士写真フイルム株式会社
電子映像事業部設計部
中村　円美

１．はじめに

近年、デジタルカメラ搭載制御ソフトウエアは、商品価値を高めるための商品の多機能化、高機能化に伴い年々規模は増大し、また複雑化してきている。その一方で商品化サイクルの短縮により開発期間は短くなり、複数の商品を並行開発しなければならない厳しい状況におかれている。

このような状況を乗り切り、高品質なソフトウエアを開発するため、富士写真フイルムではソフトウエア開発の効率化、開発力の強化に取り組んでいる。この取り組みの一つとしてCASEツールの導入を検討した結果、ZIPCを導入することになった。本稿では、商品開発へのZIPCの適用とその後実施したZIPCのカスタマイズについて解説する。

２．商品開発へのZIPC適用

2.1．適用の経緯

ZIPCの適用にあたっては、まず協力会社において、旧モデルを利用して機能の一部分をZIPCの成果物で代替させるテストプロジェクトを立ち上げ、ZIPCの適用検証を実施した。本検証の詳細はZIPC WATCHERS Vol.6に掲載されているので本稿では省略する。適用検証の結果、設計書の共通化、設計から実装へのシームレスな接続、ソースコードとドキュメントの一致等の適用効果が認められ、またツール機能の面でもデジタルカメラ開発へ十分適用可能なことが確認できたことから、商品開発へ適用することになった。

2.2．商品開発について

最初の適用対象は「ハイエンドコンパクト」クラスのデジタルカメラに搭載する制御ソフトウエアとした。ソフトウエアの構成は、ユーザーへサービスを提供するユーザーIF層、カメラ固有の機能を提供するアプリケーション層とハードウエアを制御するドライバ層に大別される(図1)。当初は、テストプロジェクトで開発実績のあるアプリケーション層とドライバ層の一部のタスクにZIPCの適用を予定していたが、社内で検討を進めた結果、ユーザーIF層のタスクへも適用を拡大することになった。適用を拡大した理由は、今回の商品開発ではユーザーIF層を新規に設計する必要があったこと、ユーザーIF層は状態遷移が多いことから過去の開発資産（Cソースコード）を元にして設計するよりも状態遷移表設計を適用して開発したほうが品質の高いプログラムを開発できると判断したことによる。

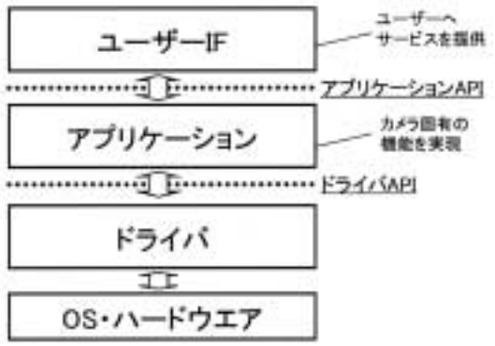

図１　ソフトウエアの構成

ZIPCの運用方法はテストプロジェクトの検証結果を参考に決定した。設計ドキュメントとしての状態遷移表の読み易さを確保するため、文字置換設計書を最大限活用し、状態遷移表はできる限り日本語で記述する方針とした。

ソースコードはZIPCを適用した全てのタスクでCコード生成機能により自動生成させた。生成したコードに対して手作業による修正を禁止し、状態遷移表とソースコードの内容が常に一致するようにした。不具合や仕様変更により修正が発生した場合でも、必ず状態遷移表を修正してからソースコードを再生成させた。

ジェネレータはコードサイズとパフォーマンスの両方を重視することと、テストプロジェクトで動作検証済みであることからジェネレータ2000（オフセット型）を採用した。また、ターゲット上でのデバックで問題が発生した場合、問題の状態遷移表のセルを突き止める手掛かりとするために、リバース機能を有効にしてソースコード上にリバース情報を埋め込んだ。これにより自動生成コードのみの場合と比べソースコードから問題のセルを突き止め易くなった。

全体としては事前にデジタルカメラ開発に適した運用方法を検討できていたおかげで、開発途中で運用方法を変更することもなく適用することができた。ただし、現状のZIPCは多人数開発を支援する機能が乏しく、運用で補った部分もあるので、多人数開発への強力なサポートをZIPCに期待する。

３．適用結果と効果

3.1．適用結果

ZIPCを適用して開発したタスクは全タスクの約半数で、コード比率にすると約40%であった。コードサイズは、適用検証での結果と同様に、ZIPCを適用したことによる増加が認められた。機能追加等の他の要因もあるため正確な値は求められないが、ZIPC適用によりコードサイズは適用前と比べ約25%程度増大したと推定される(表1)。ただし、ZIPC適用による工数の増減については、ZIPCを適用したことによる影響のみを比較できる対象がないため算出できなかった。図２に作成した状態遷移表の一例を示す。

3.2．適用効果

適用タスクの比率	全タスクの約 50%
生成コードの比率	全体の約 40%
コードサイズの増加率	約 25%

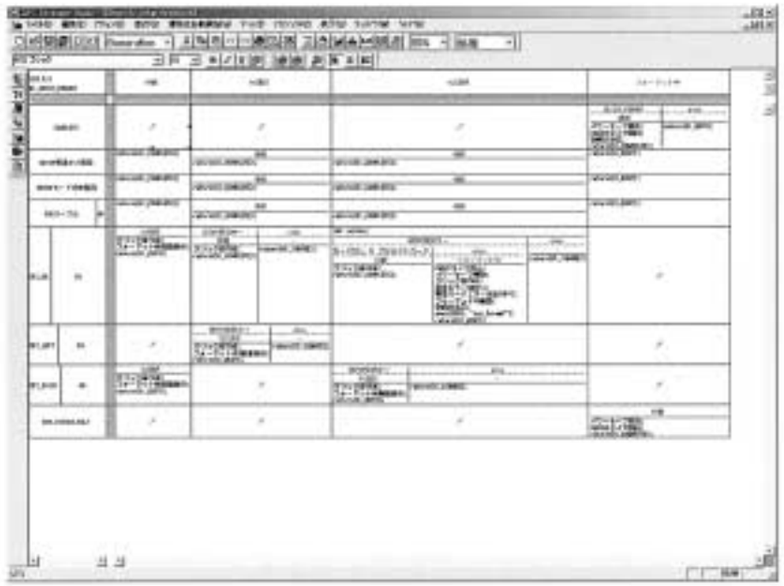

図２　状態遷移表の例

ZIPC適用による効果を下記に示す。設計作業の改善、再利用性の向上、プログラムの可読性の向上、デバック効率の向上といった多機種、短納期の商品開発に有効な効果が認められ、以降の開発でも適用することとなった。

設計作業の改善

従来の開発では仕様変更や不具合修正の内容が設計ドキュメントに反映されておらず、そのため設計者は設計ドキュメントとソースコードの両方を解析して設計内容を理解する必要があった。これに対し今回、状態遷移表からソースコードを自動生成し、修正する場合も必ず状態遷移表を修正してからソースコードを再生成する方針にしたことにより状態遷移表、すなわち設計ドキュメントとソースコードの同期が常に取れるようになった。これにより、設計者は設計ドキュメント（状態遷移表）を解析するだけで済むようになり作業効率が向上した。

また、コード生成機能を利用したことでコーディング作業に掛かる時間が減少し、その分の時間を設計に割り当てることができるようになった。

再利用性の向上

商品開発では通常旧モデルのソフトウエアをベースにして開発するため、ベースとするソフトウエアから再利用できる部分とできない部分の切り分けをしなければならない。従来の開発では設計ドキュメントを手掛かりにソースコード上で再利用の可否判断をしていたが、ZIPC適用により状態遷移表上で視覚的にその判断ができ、作業がし易くなり影響範囲もわかり易くなった。また、状態遷移表単位あるいはセル単位での部分的な再利用も容易にできるようになった。

プログラムの可読性の向上

これまで設計者に依存していた状態遷移部分の仕組みや実装方法が統一したこと、および属人化していた仕様や設計が状態遷移表により明文化されたことにより、他の設計者が設計した部分を理解し易くなり、設計者間でのコミュニケーションや担当の引継ぎが円滑にできるようになった。

デバック効率の向上

状態遷移表はシステムの状態と発生しうる事象を視覚的に理解することができるため、テスト中に問題が発生した場合、その時の機器状態や操作内容の情報から不具合個所を容易に発見することができ、問題発生から修正完了までに要する時間が短縮できた。

４．ZIPCのカスタマイズ

商品開発でもZIPCの適用効果は認められたが、それとともに、商品開発で多くの設計者が利用したことによりテストプロジェクトでは出てこなかったZIPCに対する様々な要望も出てきた。中でもエディタ機能に関する要望は多く、状態遷移表の作成作業をより効率化するためにはエディタ機能の強化、改良が必要であった。また、従来からCVS（Concurrent Versions System）を利用してソースコードを管理しているが、状態遷移表はバイナリ形式ファイルのためCVSでは扱い難く、CVSとの親和性を高めるためのツールの改良も必要であった。以上の理由から出てきた要望をまとめ、ZIPCのカスタマイズを依頼した。

エディタ機能に関しては検索機能や表示機能を中心に改良した。例をあげると、図３に実行例を示した状態遷移表の検索結果を一覧表示する機能やセル背景色を変更する機能、状態遷移表のツリー表示機能等を追加した。カスタマイズしたエディタは既に商品開発に適用しており、操作性や状態遷移表の作成効率が向上し、便利に利用している。

CVS管理に関しては、状態遷移表のテキスト形式（XML形式）保存機能、相対パス対応機能等のサポート機能を追加した。図４にXML形式状態遷移表の文書構造の例を示す。残念ながらこれらの機能は、カスタマイズ版の完成が遅れたため商品開発に適用できていない。今後適用を予定している。

５．今後の課題

これまでZIPCを適用して複数の商品を開発してきた結果を元に、デジタルカメラ開発における状態遷移表設計のノウハウをまとめ、それらを設計者間で共有化することにより、状態遷移表設計の品質向上を図りたいと考えている。また、テストシーケンスの作成作業の負担からZIPCのシミュレーション機能を利用したテストは実施していないが、今後はシミュレーション機能の活用も検討していきたい。

生成コードサイズの削減も重要な課題と考え

図３　検索結果一覧表示機能の実行画面

図４　XML形式状態遷移表の文書構造例

ている。状態遷移表の記述の最適化によりある程度はコードサイズを削減できるとのことだが、最適化は可読性や修正の容易さと引き換えになる。可読性や修正の容易さはZIPCの適用効果として認められているだけに記述最適化の適用は難しい。しかしながら、コードサイズ削減は商品開発の重要なポイントなので、何かしらの施策を検討していきたい。また、コードサイズの削減は記述の最適化だけでは限界があると考えるので、コードサイズの削減に特化した新しいジェネレータの登場をZIPCに期待する。

６．まとめ

デジタルカメラの商品開発にZIPCを適用した結果、設計作業の改善、再利用性の向上、プログラムの可読性の向上、デバック効率の向上等の効果が認められ、以降の商品開発でも引き続き適用している。まだまだ適用を始めたばかりで課題もあるが、オブジェクト指向分析・設計との提携も視野に入れながら、今後も適用していきたいと考えている。