ZIPC開発手法と従来開発との比較

情報技術開発株式会社
エンベデッドユビキタス事業部 E/Uソリューション開発一部
樫原　隆之・小田　高久・今中　竜也

１．はじめに

私たちTDI情報技術開発は、携帯端末・基地局・デジタル家電・FA制御機器・車関連など、様々な組込み分野において、ソフトウェアの請負開発を行っています。ソフトウェア開発におけるサービスを提供する役割として、QCDをトータルに考え、顧客満足を得る活動を続けています。具体的な取り組みとしては、ISO9001の認証取得や、CMMレベル3認定といった品質に対する仕組みの構築を実施してきましたが、その仕組みを円滑に運用する過程において、様々な問題を抱えている事も事実です。例えば、機能要件の複雑化と増大、そして納期の短縮です。また、仕様変更への対応によるコストの増大などもあります。これらの問題を解決する手段の一つとして、開発技術の向上が大きな課題となっています。

私たちは、様々なモデリング手法が存在する中で、組込み開発で最も適用範囲が広い状態遷移表をベースとしたCASEツールであるZIPCの活用に組織全体で取り組んでいます。

今回紹介する内容は、私たちが、ZIPCの有効性評価のために、既にツールを使用せずに開発実績のあるシステムを、ZIPCを用いて再開発を実施した事例の紹介です。開発プロセス及び成果物を比較することでZIPCの特徴を理解し、今後様々なプロジェクトに適用するための基盤づくりとして取り組みましたので、その結果を報告します。

２．適用事例概要

適用したプロジェクトは、シリアル通信機能と、IOによる負荷制御機能を持ったシステムの開発です。対象としたのは、「図１　システム構成図」の中央にある、コントローラ部です。この機器の特徴は、以下の通りです。

１）UI機器１、２は、ユーザーからの操作を受け、コントローラに対して機器制御要求を行います。
２）コントローラは、UI機器からの要求を判断し、負荷の制御を行います。
３）負荷を監視し、制御結果をUI機器に通知します。
４）コントローラがホストとなり、ポーリングによってUI機器にコマンドを送信します。コマンドを受けたUI機器が、応答または制御要求を返信します。
５）制御負荷A、Bは、IOにより状態変更・状態監視を行う事ができます。
６）負荷の状態監視は、ポーリングにより行います。

図１　システム構成図

このシステムの開発に、ZIPCを適用しました。その開発手法について次章より、説明します。

３．開発プロセス

ZIPCによるソフトウェア開発は、モデル情報を基にコードを自動生成することができます。また、モデル情報を駆動し動的検証することができます。このため、ZIPCによるソフトウェア開発は、モデル駆動型開発（Model Driven Development）と位置づけることができます。

一般的に「モデルベースでの開発は仕様部分のみをモデリングするため、開発において多く存在するイレギュラーケースが表現できないのではないか？」と言う人もいます。これに対してZIPCでは、状態遷移表（STM）において、すべてのケースを網羅的に設計することが可能になっています。しかしながら、網羅的に表現

できるということは、本来重要であるはずのシステムの仕様的要件（機能）を埋没させてしまうという問題を持っています。言い方を換えれば、すべてのケースを網羅的に記述したため、本来重要であるシステムの振る舞いが分かり難くなってしまうということです。この問題を回避するためには、システムの仕様的要件を抽象化レベルで設計モデリングし、それを段階的に詳細化していく方法が良いと判断しました。簡単に言いますと、スパイラル方式での成長型モデル開発です。（「図２　スパイラル方式での成長型モデル開発」参照）

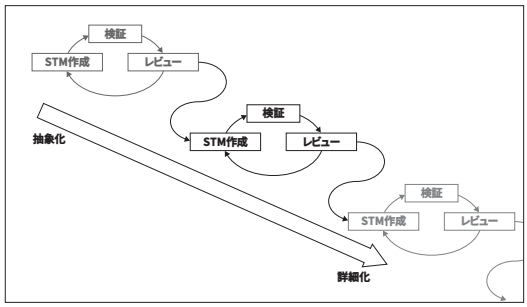

図２　スパイラル方式での成長型モデル開発

具体的には、フェーズ毎に設計モデリングする対象の「粒度」「目的」を明確にし、段階的に詳細化していくというものです。今回適用した設計フェーズを「表１　フェーズ毎の設計モデリングと目的」にてご紹介いたします。

４．成長型開発プロセスの効果

ZIPCによる開発を成長型開発プロセスで進めた結果、以下のような効果・特徴を確認することができました。メリットの他、課題も存在しますが、「粒度」「目的」を明確にし、段階的に詳細化していくという開発手法は、ZIPC開発に有効であると判断できます。

【効果・特徴】

１．STMが段階的に詳細化されるため、処理が分かりやすい
　機能の振る舞いに正常系処理、準正常系処理、異常系処理を段階的に肉付けしていったため、処理内容が明確になりました。

２．機能ブロック毎にSTMが階層化される
　基本STMでは、機能の振る舞いが明確化され、状態分割されます。
　この状態分割を詳細化していくことで、意識しなくても機能ブロック毎に整理された階層化STMが完成しました。

３．モジュール化が推進される
　機能毎にSTMが階層化された結果、モジュール化が推進されました。

４．STMの構成管理が難しい
　それぞれのフェーズにて、異なる粒度のSTMを作成するため、STMの構成管理が複雑でした。今回の開発では、単純にSTMファイル一式をコピーすることで世代管理を実施しました。実際の開発業務では、CVSやVSSなどの構成管理ツールを使用して世代管理することになると思います。

５．開発工数比較

ZIPCを適用した開発のフェーズ毎の工数を従来開発（ウォーターフォール型）の工数と比較した結果を「表２　開発工数比較表」に、示します。比較表では、開発の全工程を５つのフェーズに分け、フェーズ毎の工数を比較しています。

この工数比較の結果を考察します。最初にSSフェーズにかかった時間が従来と比べ25%程増加しています。これは従来の開発プロセスでは無かった設計書の動的検証作業によるものであ

表１　フェーズ毎の設計モデリングと目的

No.	フェーズ	設計モデリング（作成するSTM）	目　　的
1	分析	基本STM	システムの仕様的要件（機能）を明確化
2	設計	タスク毎の基本STM	タスク毎の仕様的要件（機能）を明確化
3		タスク毎の振る舞いSTM（正常系）	タスク毎の（正常系）振る舞いを明確化
4		タスク毎の振る舞いSTM（準正常系＋異常系）	タスク毎の（準正常系＋異常系）振る舞いを明確化
5		タスク毎の詳細STM（実現方法）	タスク毎の処理実現方法を明確化
6	実装	データ定義	イベント情報などのデータ構造を定義
7		置換情報定義	置換情報を定義

表２　開発工数比較表

工程	SS	PS	PG	IT	ST	計
従来開発プロセス ウォーターフォール	136	24	280	48	32	520
ZIPC適用 成長モデル	170	24	118.5	24	8	344.5

単位は:時間
SS:システム設計フェーズ
PS:関数設計フェーズ
PG:コーディングフェーズ（単体テストを含む）
IT:結合テストフェーズ
ST:システムテストフェーズ

ると考えられます。次にPGフェーズとITフェーズ、STフェーズについてですが、これらのフェーズでは、それぞれ大幅に工数短縮できた事がわかります。PGフェーズが短縮している理由としては、ソースコードの自動生成が考えられます。今回の適用事例では、実に全ソースコードの約８割が自動生成によるものでした。そして、ITフェーズ・STフェーズの短縮理由としては、設計段階での動的検証と、状態遷移表設計による仕様の漏れ・抜けの早期検出により設計品質が向上したことによって、テスト工程でのバグ数が従来と比べ大幅に減少し、手戻り作業が減った事によるものと考えられます。（今回のシステムではバグ発生件数を従来の15%まで減少させることが出来ました。）

上記のことからZIPC適用プロセスは、以下の２つの特徴を持っています。

１．設計品質の向上によるシステム全体の品質向上が望める。
２．設計工数は増加、実装・評価の工数は減少する傾向がある。

６．成果物比較

ZIPCを用いて作成したターゲットプログラムの比較を、「表３　成果物比較表」にまとめました。この表から、モジュール数の著しい増加がわかります。

ZIPC適用プロセスのモジュールは、それぞれの役割が明確になっており、非常に作成し易いものになっています。「開発工数比較」の章で述べたとおり、成長型開発プロセスにより、適切なモジュール化が促進されました。これも、ZIPC適用プロセスの大きな効果であると考えら

表３　成果物比較表

	モジュール数	プログラムサイズ（ROMサイズ）
従来開発プロセス ウォーターフォール	95	45516　[Byte]
ZIPC適用 成長モデル	225	45626　[Byte]

れます。最後に、実行プログラムサイズを比較してみると、ほとんど変わらないサイズで作成することができました。

７．今後の課題

私たちは、今回のテスト開発で、成長型開発プロセスを用いました。状態遷移表に少しずつ機能を付け足して、作業を進める方法です。そこで、問題になったのは、バージョン管理の方法です。レビューを行う度に、プロジェクトフォルダごとバックアップするという方法で、バージョン管理を行いましたが、複数のメンバーで開発を行う場合に、この方法では、問題があります。修正履歴を含むバージョン管理が必要であると、感じています。現在、ZIPCには変更履歴管理まで行えるバージョン管理機能はありません。別にバージョン管理ツールを適用し、ドキュメントのバージョン管理を効果的に進める方法を、確立したいと考えています。

８．最後に

弊社は、キャッツ社との事業提携によって、「TDI組込みソリューション」を展開しています。TDI組込みソリューションとは、組込み開発において、ZIPCを用いたモデリング開発を提案し、ツール導入だけでなく、ツールを効果的に運用するため、開発そのものをサポートするものです。

また、弊社が請負開発する組込みソフトウェアでは、ZIPCを標準開発ツールと位置づけ、品質・コスト・納期といった、３つの要素をバランスよく保ち、顧客満足が得られるよう、活動しています。

今後もZIPCを最大限に活用し、お客様の組込みソリューションを展開して参ります。今後の活動にご期待下さい。