ZIPC を導入して成功した製品開発の紹介

ヤマハ株式会社　アドバンストシステム開発センター　主任

穴田　啓樹

1．ZIPC 導入を検討されている方へ

キャッツさんが WEB 公開している「状態遷移表設計手法入門」「ZIPC チュートリアル」を読み、無料 ZIPC セミナーを受講されることをお勧めします。このセミナーで拡張状態遷移表設計手法の概要、ZIPC の使い方の基礎を習得できます。自分のターゲットの状態遷移表サンプルが書けるくらいで受講すればベスト。導入イメージを土産に持ち帰ることができるでしょう。

2．ZIPC の使い方は簡単

ZIPC は表計算ソフトでセルに入力できる人ならば簡単に使えます。空いているセルを埋めていけばよいので作業が捗り、頭の中も整理されます。また、階層化できるので全体構造も把握しやすいです。やはりツールは簡単に使えるのが一番ですね。しかし肝心なのは「どのように活用するか」です。ツールはツールでしかありません。ツールが設計してくれるわけではないので、きちんと設計する習慣のない人に ZIPC を渡しても混乱してしまうでしょう。幸い私の所属するグループでは状態遷移表を書いて設計したことがあったので ZIPC を活用することができ、購入から９ヶ月で製品を発売できました。開発途中で仕様変更に対応しながら予定通りに発売できたのは ZIPC のおかげもあると思いますので、その経緯を振り返ってみたいと思います。

3．ZIPC 導入のきっかけ

ZIPC WATCHERS Vol.3 で松下電器産業の根岸さんが書かれている通り、多人数の開発では「コミュニケーションが原因で問題が発生する」ことがあると思います。また「何を作るかが見えない」こともその通りだと思います。例えば、仕様書の勘違いで意図しないものが出来上がったり、バグが発生したり。私の所属するグループでもこれをどう改善したらよいか試行錯誤していました。ちょうどその頃、部内にソフトウェア開発プロセスの改善支援グループができて、「ZIPC という CASE ツールがあるん

だけれど使ってみない？」という話しをもらい、「仕様のモレ、ヌケ防止」のうたい文句を確認すべく、セミナー受講を決めました。冒頭に書きましたが、事前に「状態遷移表設計手法書入門」「ZIPC チュートリアル」を読み、製品の操作イメージを状態遷移表に書いたのでスムーズに理解することが出来ました。セミナーは概念導入の「入門セッション」と、実際にZIPCを使ってCDプレーヤーを設計する「体験セッション」から構成され、Cコード生成まで体験できます。このセミナーで「ツールの使い方が簡単なこと」「仕様のモレ、ヌケが防止できること」を確認できました。

４．ZIPC 導入にかけた時間

表１のように、約１ヶ月で状態遷移表から作成した自動生成 C ソースがターゲット環境で動作するようになりました。このようにアーキテクチャは約１ヶ月で実現し、残りの８ヶ月は「作りこみ」すなわち、内部動作仕様や画面デザインの追加、変更でした。「ZIPC インターフェース設計/実装」に２週間と一番力をいれましたが、これが「活用ノウハウ」にあたり、後で説明します。

５．ZIPC を導入した製品について

製品の基本シナリオは「画面に表示された音響装置のパラメータをユーザーが操作して音響装置を制御する」というもので、画面数は４０ほどあります。

＜ ハードウェアの概要 ＞

- 入力装置はボタン、ジョグダイアル、ムービングフェーダーで構成されます。ボタンは機能の選択/実行、カーソル移動、パラメータ値変更、ジョグダイヤル、ムービングフェーダーはパラメータ値変更に使います。
- 出力装置はLED、QVGA の LCDで構成されます。LED は音響装置の状態を表示し、LCD はパラメータ値を表示します。
- その他、通信用のシリアルがあります。

＜ ソフトウェアの概要 ＞

- 画面上に音響装置のパラメーターを表示します。カレント操作対象パラメーターはカーソル表示します。
- カーソルボタンが操作されたらカーソルを移動し、パラメータ値変更のボタン、ジョグダイアル、ムービングフェーダーが操作されたら、パラメータ値表示を更新して音響装置を制御します。

表 1 ＺＩＰＣ導入にかけた時間

ツール自体の調査	1日
状態遷移表作成	3日
入門セミナー受講	1日
状態遷移表作成	4日
ＺＩＰＣインターフェース設計/実装	2週間
ターゲット環境へ適用	2日

6．どこにZIPCを適用したのか？

本製品はOSにITRONを使用して表2のように主要タスクを構成しましたが、「イベント管理タスク」の一部にZIPCの生成コードを利用しました。

表　2 主要タスク

入力装置監視タスク	ボタン、ジョグダイヤル、フェーダーの状態変化を監視するタスク群 入力装置の状態が変化したらイベント管理タスクにメッセージを送信する
通信タスク	製品の外部からパラメータを変更、機能を実行するためにシリアル受信するタスク 受信データを解析してイベントが成立したらイベント管理タスクにメッセージを送信する
イベント管理タスク	入力装置監視タスク、通信タスクからメッセージを受信して、イベントを処理する中心タスク 表示を更新する場合には表示タスクにメッセージを送信している
表示タスク	LED，LCDの表示を管理するタクス イベント管理タスクからメッセージを受信してLED，LCD表示する

ZIPCはITRONに対応していますので他タスクにも適用して全体を制御することもできたのですが、時間の制約と今回は初めてだったということで、一番適用しやすそうなタスクに絞りました。

7．どのようにZIPCを適用したのか？

今回の開発では「操作仕様、画面デザイン、プログラムの分離」という目標を掲げました。簡単に説明しますと、製品の操作仕様、画面のデザイン、実際の動作を記述するプログラム、この3つの開発を別のサイクルで反復できるようにする、ということです。これまでは操作仕様や画面デザインが変更される度にプログラマに戻って作業しなければなりませんでしたが、相互に依存しないような仕組みを作っておけば、効率があがると考えたのです。これを実現するため「操作仕様とプログラムの間」にZIPCを使い、「画面デザインとプログラムの間」にツールを自作しました。ZIPCの部分に関しては、画面オブジェクト（パラメータ表示、ボタン、選択肢、リスト）の表示属性（カーソル、選択）を「状態」と定義し、ユーザーの操作（カーソル移動、実行、値変更）を「イベント」と定義して、以下の流れで作業しました。

1) 画面のラフスケッチを描く
2) 操作シナリオを書く
3) 画面オブジェクトの状態に対するイベントに注目して2)から状態遷移表を作成する
4) カーソルの動きから遷移欄を埋める
5) アクション欄に関数を埋める
6) シナリオの「モレ」「ヌケ」が見つかるので補完する
7) ZIPCのドキュメントチェック、シミュレータを利用して動作を確認する

8) C コードを生成する
9) ターゲットに移行してアクションの中身を実装する

8．ZIPC 活用ノウハウ

以下は私の ZIPC 活用ノウハウです。

1） ZIPC と外とのインターフェースを工夫する

どのように ZIPC と外とのインターフェースを切り分けしたら、仕様変更に強い仕組みになるかを考えるのに時間をかけました。これがうまくできると、ZIPC で操作仕様を書く作業と、動作の中身を書くプログラミング作業が独立して平行に進められますし、仕様変更にも強くなると思います。ZIPC のセルには処理を書けるのですが、ここにプログラミングしてしまうと操作仕様とプログラミングの分離ができなくなってしまいます。大きな製品の開発では、如何に作業を独立させて並行に進めるかがポイントだと思います。

2）表の見やすさを保つ

表を見やすくするために置換（.SSD）ファイルを活用しました。状態遷移表を作成していると、ZIPC が階層機能をサポートしているとはいえ、できるだけ大きな画面が欲しくなります。しかし実際には画面サイズは限られていますので、必要な情報を失わない程度にコンパクトな言葉をセルに記入し、表示できるセル数を稼ぎました。

3) ZIPC 生成コードは絶対に修正しない

この点には妥協しない方がよいです。仕様変更は必ず状態遷移表に戻って修正し、ZIPC 生成コードには手をつけませんでした。これによりドキュメントとプログラムは常に完全に一致し、リバースする必要はありませんでした。これを実現するには 1) がポイントになります。また、実現課題を状態遷移表に表現する時に「プログラムで書いた方が簡単」と思う場面もあったのですが、ちょっと考えて工夫すれば、状態遷移表で表現できました。

4) 日本語は使わない

セルに日本語が使えるのは ZIPC の特長ですが、プログラミング言語が日本語に対応していない以上、置換ファイルを利用してアルファベットに変換しなければなりません。それでしたら初めからわかりやすい英語で書いておいた方が手間が省けますし、つまらないミスも減ると思います。

9．開発を終えて

ZIPC を導入して本当によかったと思います。状態を持つものならば、状態を何らかの形でプログラムに落とすわけで

すから、ZIPC によってその手間が省けるのです。手間が省けるだけでなく、ドキュメントからプログラムに落とす時のミスもなくなりますので、ZIPC を導入した部分に関しては品質が向上しました。また、仕様変更時に状態遷移表を書き換えましたが、特に開発後期になってくると、一部分しか見えないプログラム修正より、広く見渡せる状態遷移表の修正の方が安心でした。

ドキュメントがプログラムになるという観点からも評価できます。これまではプログラム修正に対するドキュメント修正が、手間がかかる等の利用で確実ではなく、「このドキュメントは古い」という状態が少なくありませんでした。その点 ZIPC ではドキュメントとプログラムが１対１で対応するわけですから、ドキュメントも信頼できるというわけです。ただし現状では、表示機能や印刷機能が少々不便に感じる時があり、もう少し自由度を高くしていただきたいです。

今回は時間的な制約で、残念ながら ITRON 対応機能と Visual Basic と連動したプロトタイピング機能が利用できませんでした。ユーザーの意見を早くフィードバックするために、プロトタイピング機能は必須だと思いますので、次回は是非チャレンジしてみたいです。できれば ZIPC にインターフェースビルダーを内蔵してもっと簡単にできたらよいと思います。

キャッツさんのサポートは本当に親切でした。メールで質問すると、まず、質問を受け付けた旨、返信がもらえます。それから内容によって、簡単なものなら翌日、難しいものなら数日中に回答がもらえます。単なるツールの使い方だけでなく、課題の実現方法についても相談し，回答してもらえました。ZIPC は一般的なツールではありませんが、このようにサポートが充実していたので安心して使えました。