# ZIPC を用いた導電率計 VIP モデル 開発適用の評価考察

情報技術開発株式会社
マイクロシステム開発部

**部長 吉田 浩司・主任 工藤 弘隆**

## 1. はじめに

組み込みソフトウェアの開発は年々「規模の増大」「複雑化」が増しています。しかし、開発期間はほとんど変わらないどころか期間短縮を求められることもあります。このような状況で一定の品質を達成するのは厳しいのが現状です。

そこで組み込み開発用の CASE ツールである ZIPC を導入することにより改善できないかどうかを検証することとなりました。

## 2. ターゲットの概要と条件

対象となる機器は携帯用導電率計です。導電率計とは電極が測定した液体の温度と電流値により液体の電気の流れやすさを測定するものです。

概観図は **図 1** となります。右が電源オン直後、中央が使用中の機器の概観図で製品と同じ操作で動作します。左がセンサーやハードウェアの記憶部の役割を果たす概観図です。

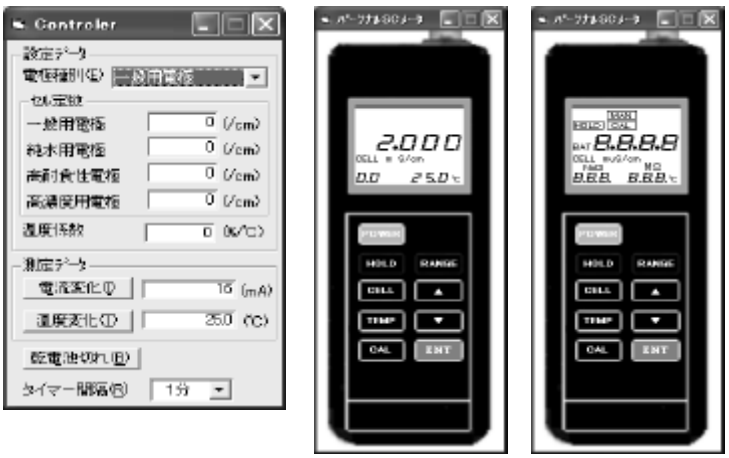


**図１ 概観図**

開発条件は次の通りとなります。

- 設計資料：操作説明書
- 開発モデル：VIP モデル
- 担当者：C による組み込みソフトと VC++によるデバッグツールの開発を経験。ZIPC の使用は今回初めて

## 3. 開発手順の比較

### 3. 1 従来の開発手順の問題

従来から開発システムによっては状態遷移表やシーケンスを作成して設計を行っていますが規模の大きい開発になるほど次のような問題があります。

#### 1) 基本設計での問題

図式化した状態遷移やシーケンスを元に状態遷移表を作成しますが、すべてワープロソフトによる手作業のため設計よりも入力作業に多くの時間を費やしてしまい十分な検討を出来ない場合があります。また、仕様変更はどの工程でも発生
しますが開発終盤になるほど設計を省く傾向にあるため影響範囲の特定を誤り余計な工数が発生してしまうことがあります。

#### 2) 詳細設計／コーディングでの問題

詳細設計以降は個人に任される場合があり、担当者の不注意により基本設計の内容が反映されていないことがあります。

#### 3) テストでの問題

不具合が発生した場合にいきなりソースコードを修正して対応することが多いので、基本設計とソースコードが一致しなくなってしまいます。

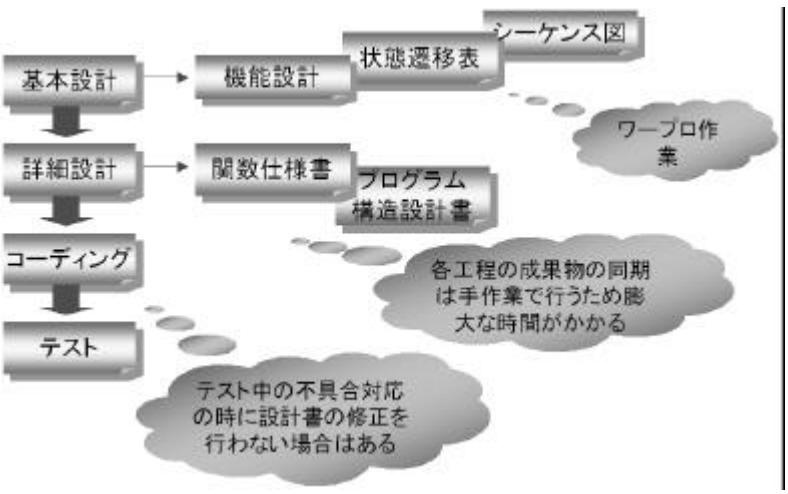


図 2　従来の開発手法

### 3．2　ZIPC による開発手順

このような問題を解決するために ZIPC を使用した開発では次のことを期待しています。

- 図のコンバート機能や編集しやすさにより手作業の時間が減り、設計に多くの時間を費やせるようになる。
- ドキュメントからソースコードが自動生成されることにより、全体的な作業負荷が軽減される。

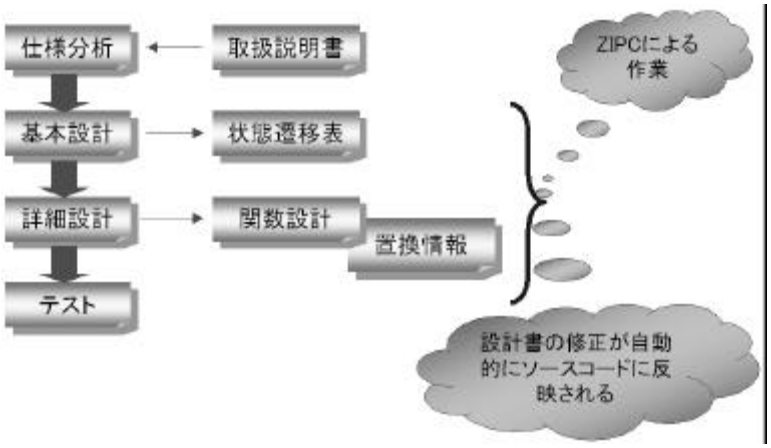


図 3　ZIPC による開発手法

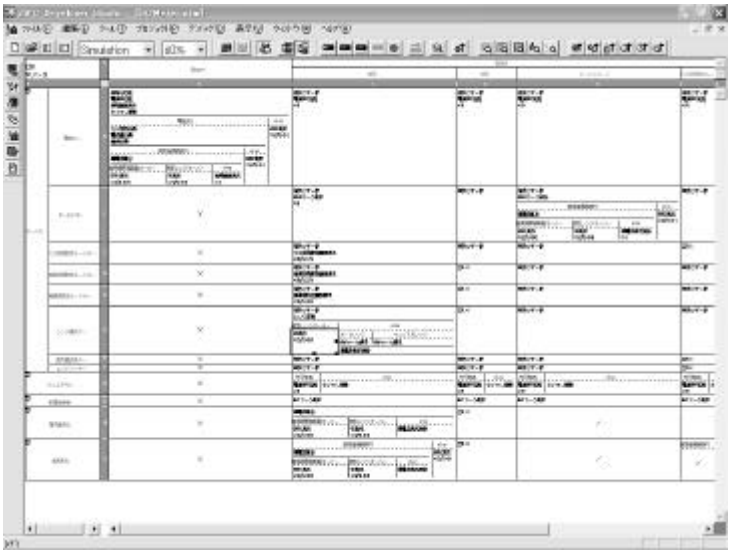

図 4　STM

## 4. 開発規模

初めて携わった機器であり実装モデルではないため従来との比較はできませんが、作成した VIP モデルの規模は **図5**、工数は **図6** のようになりました。事前にセミナーを受けることもなく予備知識のない状態で、ZIPC に付属のオンラインドキュメントや拡張階層化状態遷移表設計手法の本とサンプルで、記述方法や操作方法と VisualBasic による概観図の作成方法を調べながら作業しました。工数にはそういった試行錯誤の時間も含んでいます。

製品と同様の動作をする VIP モデルを初めてづくしの状態で全体として約 1.5 人月で開発できたのは、ZIPC の分かりやすさと操作性の良さにあると思います。

| | 規模 |
|---|---|
| STM | 5 枚（セル数合計 120） |
| C 言語ステップ<br>（シミュレーションコード） | 約 2.5 KL |
| VisualBasic ステップ数 | 約 0.5 KL |

**図5　開発規模**

| | 工数 | 作業内容 |
|---|---|---|
| 基本設計 | 約 60 時間 | 状態遷移表の作成 |
| 詳細設計 | 約 40 時間 | 置換定義、関数定義の作成 |
| 概観図作成 | 約 40 時間 | VisualBasic による概観図の作成<br>ZIPC の VIP インターフェイス部分組み込み |

**表6　開発工数**

## 5. 考察

### 5. 1　コード自動生成

自動生成されたソースコードの内訳は **図 7** のようになりました。実際に手作業で定義した部分は全体の 20% 程度となりました。VIP 依存部分とは本来ハードウェアが測定したデータを取得したり表示内容を変えたりする部分を VIP インターフェイスにより概観図とやり取りしている部分です。この部分をハードウェアの仕様に合わせた処理に置き換えることにより移植が可能と思います。

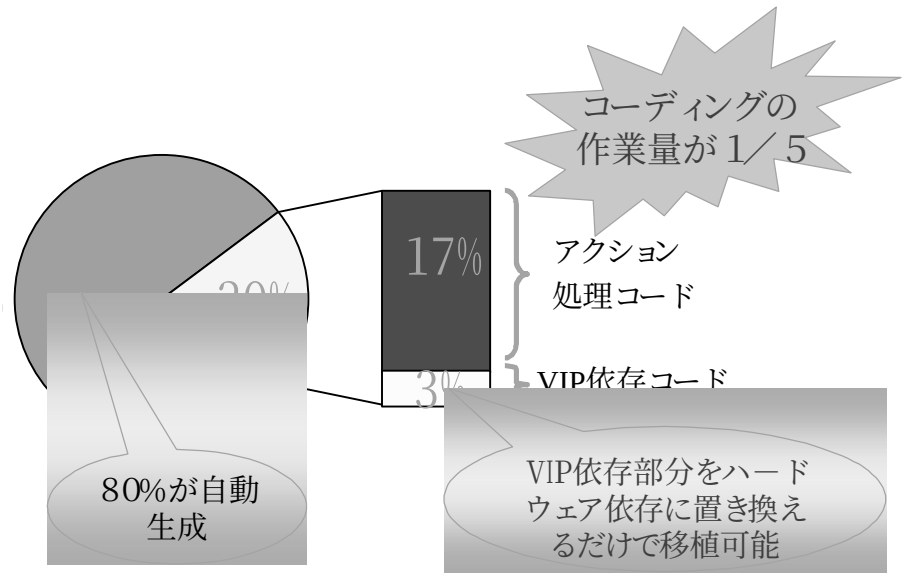


**図 7 ソースコードの内訳**

### 5．2　変更による追加工数が減る

仕様追加、変更、不具合などで状態遷移表に影響するような変更が発生した場合は、従来と比べて追加で発生する工数が減ると思います。従来は **図 8** のように各工程で前工程の修正による影響範囲を調査し修正していきます。ZIPC の場合は状態遷移表の修正及び置換定義が自動的にソースコードに反映されますので単純に作業量が減ります。また調査不足などのミスによる余計な工数も軽減できると思います。

### 5．3　ハードウェアレスのテストが可能

概観図をハードウェアの代わりに使用することでアクション処理コードのテストを行うことができます。

### 5．4　テスト不足を視覚的にチェック

概観図などによる ZIPC のシミュレーション機能を利用したテストを行った場合は、**図 9** のようにテスト項目毎にログを保存しておき最後にすべてのログを使用してカバレッヂ機能を実行することによりテストされていないセルを発見することができます。ただし、すべてのセルが緑色だからといって「テスト項目が十分である」というわけではありません。

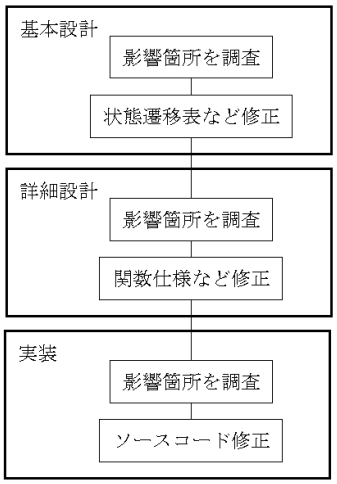


図 8　従来の変更作業手順


図 9　テスト項目不足チェック

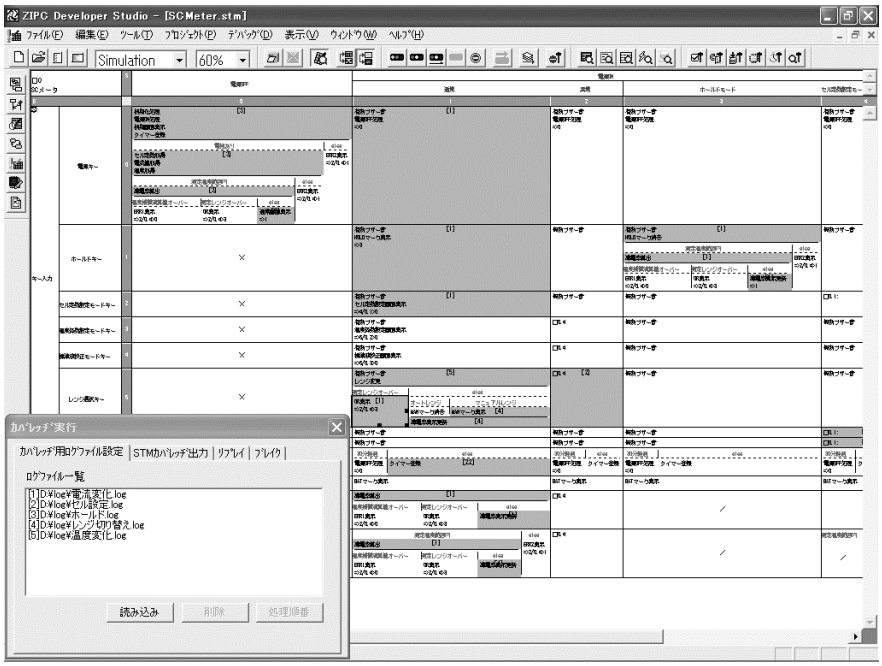


図 10　カバレッヂ実行画面

## 6. 評価のまとめ

ZIPC による設計は編集作業もしやすく、分析に使う図からコーディングまでシームレスになっているため作業の手間が減りました。実際の開発においても本来の「設計をする」という作業に、より多くの時間を使えるようになり設計段階での品質の向上が考えられます。

また状態遷移部分の単純なコーディング作業が不要になることと、状態遷移表や各設計書の修正がソースコードに反映されるためコーディング工程の短縮が考えられますし、設計書とソースコードの一致も実現できると思います。

不具合の中でも影響の大きい設計漏れが減ることにより結果的にテスト工程の短縮につながると考えられます。

## 7. 今後の展開

ZIPC や Konesa－RealTime といった CASE ツールを利用した組み込みソフトウェア開発を模索し、実際の現場に活かしていければと考えています。

¥