RT システム開発における 設計支援ツール ZIPC の運用連係に関する評価

○関山守(産総研)　穴田啓樹(キャッツ株式会社)　鍛冶良作(産総研)
谷川民生(産総研)　梶谷勇(産総研)　阪口健(産総研)　神徳徹雄(産総研)

1.　背景と目的

一般的なシステム開発においては，開発するシステムのサイズに応じて複雑度は増加する．複雑なシステムを複数のメンバで開発するには設計情報を伝達・共有するためのドキュメントの整備が必須であり，その部分に対応している設計支援ツールを用いる事で開発全体の効率化を図ることが可能である．そのような設計支援ツールは複数人員による開発の効率化に資するのみでなく，単一～少人数人員による開発の際にも開発プロセスに対して様々に効果的な影響を及ぼす．有効な設計支援ツールの積極的な導入はシステム開発において非常に有益である．

しかしRTミドルウェア[1]をベースとするRTシステム開発において使用できる設計支援ツールは多くないのが現状である．RTシステム以外の分野での実績を持ちRTシステム開発に馴染みやすい設計支援ツールが存在するのであればRTシステム開発に積極的に導入して開発の効率化を図るべきである．

そこでどのような設計支援ツールがRTミドルウェアをベースとするRTシステムの開発に馴染むのかを考えてみる．RTコンポーネントは複数の状態と状態遷移を持ったソフトウェアモジュールであると言える．そして実際の実装においてこのモジュールはperiodicタスクで実現されることが多い．このような特徴はRTコンポーネントが状態遷移モデルをベースにした設計支援ツールと非常に良く馴染む事を示している．

ZIPC[2]は状態遷移モデルをベースにした設計支援ツールである(図1)．日本国内の組込み機器開発分野でのデファクトスタンダードの設計支援ツールでもある．本稿ではZIPCによるRTコンポーネント開発と運用の連係を通して，状態遷移モデルをベースとした設計支援ツールZIPCのRTシステム開発への適用の評価を行う．

2.　ZIPC とは

ZIPC は状態遷移モデルをベースにした Windows 環境下で動作する設計支援ツールである．状態遷移モデルの設計と設計した状態遷移のシミュレータ・デバッガおよびテスター，設計したモデルに関するコードのC言語によるジェネレーションの機能をもっている．また状態遷移モデルの設計の際に日本語を取り扱うことができるのが特徴である．

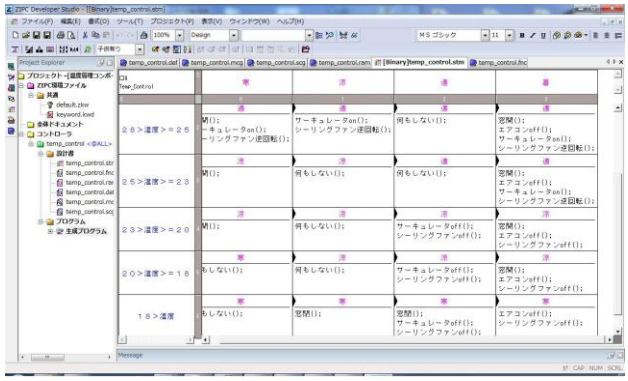

図 1　ZIPC 動作画面

ZIPC においては状態遷移モデルを最終的に表計算ソフトで用いられるシートのような表で表現する．これは「拡張階層化状態遷移表設計手法 EHSTM V2」[3]という設計手法に基づいている．状態遷移モデルを表で表現する事によりに遷移条件の抜け落ち等をユーザーが比較的簡単にチェックする事が可能となった．またデバッガなどのチェックツールによる確認も容易になったようである．シミュレータ，デバッガ，テスターなどによるチェックを経て設計した状態遷移モデルが問題ない事を確認した後に状態遷移モデルを C 言語のコードに変換する．このコードは if 文のネストか case 文で表現されているため原理の理解は簡単である．

状態遷移モデルを表現した状態遷移表においては日本語を使用する事が可能であることは上に述べた．これは状態遷移表の可読性を高くする効果があり結果として設計する状態遷移モデルの理解を容易とする利点がある．しかし日本語を状態遷移表において使用した場合にはコードジェネレーションの際に必要となるトランスレーション用の定義ファイルを準備しなくてはならない．

シミュレータ，デバッガの機能では状態遷移表とマウス等からのイベント入力・データ変数値の更新を組み合わせることで設計した状態遷移モデルの動作確認が可能となっている．状態遷移の為の入力ではマウス，キーボードからの直接的な単一入力を用いる他に，専用書式のテキストファイルを用いる事により連続的な状態遷移動作の確認も可能となっている．さらにシミュレータ用の専用ライブラリが用意されており，このライブラリを介して外部ソフトウェアモジュールから状態遷移の為の入力をシミュレータに通知する機能やシミュレータから状態遷移の結果を外部ソフトウェアに通知する機能も存在する．

またコードジェネレートの際に特定ソフトウェアモジュールに動作通知を送るようなフックを埋め込むことも可能である．フックからの出力を捕捉してZIPCに連携させて，設計した状態遷移表と組み合わせて動作しているコードの内部の状態を表示させることも可能である．この機能を用いると実環境における動作と内部状態を用いたデバッグを行う事ができる．

3. ZIPCを用いたRTコンポーネント開発

我々はRTコンポーネント開発においてZIPCを二通りの方法で用いて評価する事とした．

一つは周期実行するRTコンポーネントでのonExecuteルーチン内の状態遷移モデルについて，ZIPCを用いて設計･検証を行うという一般的な使用方法である．シミュレータ・デバッガを用いて設計の検証を行った後にコードジェネレータより状態遷移モデルをC言語のコードとして生成する．生成したコードを開発するRTコンポーネントに導入することにより，RTコンポーネントの開発効率がどのように変化したかを評価する．(図2)

もう一つはZIPCシミュレータ用の専用ライブラリを用いてRTコンポーネントからZIPCのシミュレータに状態遷移の為の入力を通知，またシミュレータから状態遷移の結果をダイレクトにRTコンポーネントに通知させる方法である．RTコンポーネントの内部にZIPCのシミュレータを併呑した形となるためRTコンポーネントとしての高速動作は保証できなくなるが，ZIPCでの状態遷移モデルにおける変更・修正をコードジェネレーションすることなくRTコンポーネントの動作に反映させることが可能となる手法であり，このようなZIPCシミュレータのユニークな使用方法に関する報告は実はされていない．この手法を導入する事により設計の方法がどのように変化するかについて評価する．

以下に詳細を述べる．

3.1 ZIPCの一般的な使用方法での評価

状態遷移モデルによる設計の対象とする周期実行RTコンポーネントを次のように設定した：「複数のセンサコンポーネントからのデータを受け取り，現在の状態と受信データの値から次の状態を決定，状態遷移が起こる際に複数のアクチュエータコンポーネントにデータを出力するようなRTコンポーネント」．このようなRTコンポーネントの内部状態の遷移モデルについてZIPCの状態遷移表とシミュレータ・デバッガを用いながら設計した．状態遷移表による設計の際には日本語を多用して可読性を確保することとした．コードジェネレータが出力するC言語のコードについてはWindows以外の環境でもビルドが可能となるよう配慮することが重要である．設計の指針としてZIPCで生成するコード部分には環境に依存するコードを含ませない事を定める事が必要となる．

以下は評価である．ZIPCを用いて状態遷移モデルを設計，シミュレータを用いる事で状態遷移モデルのエラーの早期発見・除去が可能となった．その結果，全体的なデバグに要する時間は減少したので効

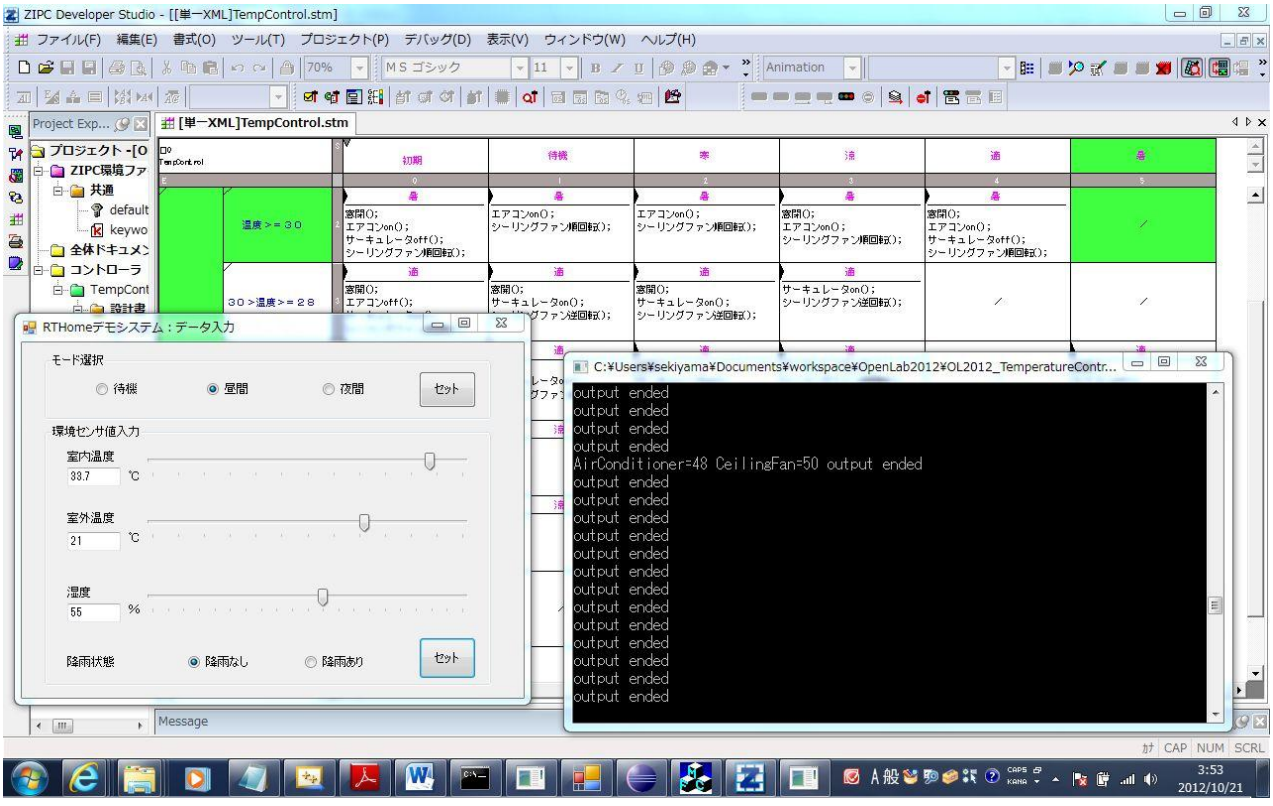

図2　RTコンポーネントとZIPCの連係動作画面

率は上がったと考える．状態遷移表の記述の際に日本語を多用して可読性の確保を図ったが，この事によりコードジェネレーションに必要となるトランスレーション用定義ファイルの記述が煩雑かつ分量が多くなってしまった．この部分では効率が下がると考えた．しかし，このような定義ファイルを詳細に記述する作業の結果，開発するモジュールについては状態遷移モデル以外の部分も構造化され最終的なコードそのものの可読性が上がったことは非常に重要な効果と言える．また定義ファイルをきちんと準備したため説明のためのドキュメントをまとめる際にこれら定義ファイルの流用が可能であり全体で考えると効率が上がったと言える．

但し, ZIPC というツールそのものが持つ若干の癖, 特性を理解していない事が原因でエラーを起こした際にはデバグは困難を極めた．このあたりはツールを使う際にはある程度の経験が必要であるということを痛感させられた．

最後に ZIPC のコードジェネレータを用いて生成した C 言語のコードを Windows 上および Linux 上で RT コンポーネントのソースに組み込んでビルドを行い，動作させて問題がない事を確認した．若干の注意が必要ではあるが，ソースレベルで環境非依存であるような状態遷移モデルのコードが出力できるということは広義のクロス開発に ZIPC を活用する事が可能であることを示している．

3.2 ZIPC シミュレータのユニークな使用方法

内部に ZIPC シミュレータを併呑する周期実行する RT コンポーネントについても上記と同様に次のように設定した：「複数のセンサコンポーネントからのデータを受け取り，現在の状態と受信データの値から次の状態を決定，状態遷移が起こる際に複数のアクチュエータコンポーネントにデータを出力するような RT コンポーネント」．イメージとしては onExecute ルーチン内で ZIPC シミュレータ用ライブラリ関数をコールする RT コンポーネントである．onExecute ルーチンの周期実行の度に複数のセンサコンポーネントから受け取ったデータについてライブラリ関数を介して ZIPC シミュレータに通知する．また ZIPC シミュレータからの状態遷移の結果についてライブラリ関数を介して通知を受け取る．この結果の通知に基づいて複数のアクチュエータコンポーネントにデータを出力させる．このような Proxy 的動作を行う RT コンポーネントを設計した．実際の状態遷移モデルについては ZIPC 上で設計を行った．このような構成の下で，受け取ったデータ数値に対して状態遷移が起きるかどうかの閾値の変更，状態遷移時に行うアクチュエータコンポーネントへの出力数の変更のような状態遷移モデル全体から見ると軽微な変更や，状態遷移の状態数や遷移先の変更といった状態遷移モデルの大幅な変更についても ZIPC における状態遷移表の内容変更だけで実現する事を示した．この使用方法の適用範囲について考えると開発途上でのデモや説明・ディスカッションの際に実機を用いての状態遷移モデルの変化の確認が可能であり，システムの開発工程のさらなる「見える化」の実現ができた．この使用方法を用いる事でユーザー参加型（ユーザーとベンダーの対話型）開発も可能となることは重要なポイントである．

4. まとめ

状態遷移モデルをベースとした設計支援ツール ZIPC を RT コンポーネント開発に用いることで効果的な開発が行えるようになるかどうかについて評価を行った．状態遷移モデル開発部分でのエラーの混入を除去しやすくなったためデバグの効率は上がった．二バイト文字を多用することで可読性は上がり構造的なプログラム記述が実現した．トランスレーション用定義ファイルの記述という作業の増加とドキュメント生成の手間の減少についても述べた．コードの適切な切り分けを意図することにより ZIPC で Windows 用に生成した C 言語のコードは, 手直しすることなく Linux 用のコンパイラでビルドすることができた．以上をまとめると ZIPC を通常の用法でも用いる事で開発全体の効率化が成されるため RT コンポーネント開発に ZIPC を用いる事は効果的であると評価する．

さらに ZIPC シミュレータの特殊用法を RT コンポーネント開発に活用することで開発工程のさらなる「見える化」と開発工程へのユーザー参加や実環境を用いたモジュールのテストを実現する事ができた．

全体で考えた場合，状態遷移モデルをベースとした設計支援ツール ZIPC の RT システム開発への適用については非常に有効であると結論付ける．

参 考 文 献

[1] Noriaki ANDO, Takashi SUEHIRO, Kosei KITAGAKI, Tetsuo KOTOKU, Woo-Keun Yoon, "RT-Middleware: Distributed Component Middleware for RT (Robot Technology)", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2005), pp.3555-3560, 2005.08, Edmonton, Canada.

[2] “状態遷移表設計で品質向上 キャッツ ZIPC”, http://www.zipc.com/

[3] 渡辺政彦： 拡張階層化状態遷移表設計手法 Ver.2.0-Embedded SE のための設計手法，キャッツ，2010.