ZIPC WATCHERS Vol.19

ZIPC Designer for mruby による組込みソフト開発

九州工業大学
情報工学部 准教授
田中　和明

1. はじめに

組込みシステムは、家電製品や自動車、産業用機器などの制御に利用されるコンピュータシステムの総称で、ソフトウェアによりハードウェアを操作することが目的である。

最近では、製品に対して、多くの複雑な機能が要求されてきている。また、仕様変更やユーザビリティの試行錯誤による開発など、システムを迅速に開発できることも求められている。

組込みシステムの開発を効率よく行え、ソフトウェアの改修が容易で、かつ、長期間にわたるソフトウェアのメンテナンスが可能であるような開発手法を提案できないかと考えた。

この考えを実現するための一つの手として、プログラム言語に着目し、具体的な実装を行ったのが mruby である。以下、mruby の概要と、mruby を組込みソフト開発に活かすための設計ツールについて紹介する。

2. mruby

プログラム言語 Ruby は、Web アプリケーション分野において Ruby on Rails といったアプリケーションフレームワークが広く利用されてきており、Ruby を使うことでソフトウェアを効率よく開発できることが知られている。

Ruby はオブジェクト指向プログラミング言語であり、ISO/IEC 30170 および JIS X 3017 として標準化されている。また、多くのクラスライブラリ、特にネットワークに関連する機能が整備されていて、アプリケーション開発が効率よく行える。スタートアップ企業において、Ruby が好んで採用されているのも、開発効率が高いことが理由の一つである。

このように、Ruby が Web アプリケーション開発において注目された時期に、経済産業省の 2010 年度地域イノベーション創出研究開発事業「軽量 Ruby を用いた組込みプラットフォームの研究・開発」において、ネットワーク応用通信研究所、福岡 CSK、九州工業大学、東芝情報システム、福岡県が共同で、Ruby を軽量化したプログラム言語「軽量 Ruby（mruby）」を開発した。

mruby の開発で最も重視したのは、「組込みシステムに利用できる」ことである。組込みシステムのハードウェアは、消費電力、熱、コストの要件を満たすため、使用できる資源に制限がある。特に、メモリに対する制限は厳しく、少ないメモリで動作することを目標のひとつとした。

オリジナルの Ruby はインタープリタ方式のプログラム言語であり、実行時にプログラムソースコードの解析を行う[注1]。この解析を実行環境で行うため、実行環境に多くの資源が必要となる。そのため、望まれているのは「実行時に必要な資源」を少なくすることである。mruby ではコンパイラ方式を採用した。Java のように、mruby では、コンパイラが Ruby コードから中間コードを生成し、その中間コードを仮想計算機（VM）で実行する仕組みである（図 1）。

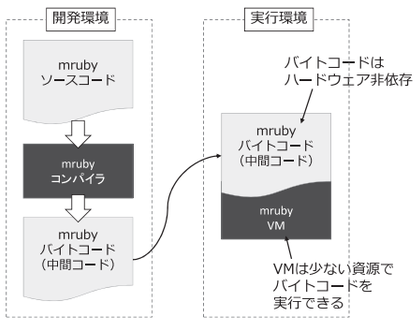

図1　mrubyプログラムの実行

Ruby が多くの資源を必要とするのは、Ruby のソースコードを解析する部分である。ソースコードを解析して、開発時に mruby コンパイラにより中間コード（mruby バイトコード）を生成しておくことで、実行時にはコンパイル済みの中間コードを実行するだけで済み、少ない資源で実現できる。VM を動作させるために必要なメモリは約 200KB 程度であり、起動だけで数 MB を必要とする Ruby に比べて軽量と言える[注2]。

ところで、組込みシステムではさまざまな実行環境が想定され、OS やプロセッサへの依存を少なくし、OS がない環境でも動作することも望まれる。ハードウェアに依存するソフトウェアは、移植や機能拡張などの変更に対して柔軟性に乏しい。mruby の処理系は C 言語（C99）で記述されており、組込みシステムを構成するさまざまなハードウェアへの移植が容易である。また、VM によりハードウェアが抽象化され、一度作成した mruby プログラムは、異なるハードウェアの VM で変更なしに動かすことができる。

3. 状態遷移モデル

組込みソフトウェアでは、ハードウェアを扱うための記述が登場する。ハードウェアの先にはユーザやセンサ、実際の装置などが存在し、それらの入出力に関するソフトウェアの記述が組込みソフトの中心と言える。このように構成されるソフトウェアは、ある入力に対してある出力を行う、または、入力によって内部状態が変化する（例えば装置の動作モードが変化する）といった典型的なパターンが登場する。イベント駆動型のソフトウェアと言える。

簡単なレベル設定（例えば、扇風機の風量設定）の例を考える。レベル設定のため、「インクリメント（増加）」「デクリメント（減少）」「リセット」の入力を与え、内部状態が0～3の間で変化する機能を持つ。この入力と状態遷移を図に示す（図2）。

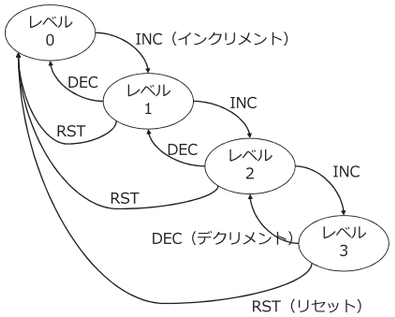

図2　状態遷移図

状態遷移図を作成することで、ソフトウェアで実現すべき機能を明確に、わかりやすく表現できる。先の「レベル設定のため・・・」で記述した言語による表現に比べて、状態遷移図の方がわかりやすい。状態遷移図は、開発者が仕様を正しく理解し、仕様どおりにソフトウェアを実装するために有効な手段と言えるだろう。

しかし、図 2 の状態遷移図には、未定義の箇所が存在する。レベル 0 でデクリメントやリセットの入力が行われた場合、レベル 3 でインクリメントの入力が行われた場合の動作が記述されていない。このように、状態遷移図は強力な表現手法ではあるものの、未定義の箇所が存在していてもそれに気づきにくいという欠点もある。

入力と状態の数が有限であるならば[注3]、全ての組み合わせを列挙することができる。すなわち、入力×状態の表を作成でき、その表のすべての項目に動作を記述すれば、漏れのない実装となることを意味する（図 3）。

この図では、横方向に状態を、縦方向に入力（イベント）を示し、表中の各要素がある状態である入力があった際に遷移先の状態を示す。図 3 において、グレーの部分が図 2 の状態遷移表で未定義であった。この例では、グレーの部分は「（無視）」として、その入力に対する動作を行わないように設計したが、場合によっては適切なエラーの提示

注 1 実際には、Ruby プログラム開始時にコンパイルが行われ、コンパイル済みのコードが実行されている。実行環境において Ruby ソースコードの解析が行われることには変わりない。
注 2 さらに軽量な mruby の実装として、「mruby/c」が研究開発中である。
注 3 有限オートマトン（FA）または有限状態機械（FSM）と呼ばれ、決定的な状態遷移記述が可能であることが示されている。学術的には、状態数の最適化などの研究も行われている。

ZIPC WATCHERS Vol.19

によりユーザに通知するなどの処理が必要になるであろう。

	状態0	状態1	状態2	状態3
インクリメント	→状態1	→状態2	→状態3	(無視)
デクリメント	(無視)	→状態0	→状態1	→状態2
リセット	(無視)	→状態0	→状態0	→状態0

図3　状態遷移表

ソフトウェアの生産性と安全性の向上を実現する一つの方法が、自動化である。先に示した状態遷移表が適用できるソフトウェアは、コードの自動生成と相性がよい。

4. ZIPC Designer for mruby

組込みシステムにおいては、入力と状態により動作が規定される場合が多く、状態遷移モデルにもとづいてソフトウェアを開発することが可能である。状態遷移モデルに基づく開発は、自動化できる工程も多く、開発効率を高めることができる。この設計からソースコードを生成してくれるツールとして ZIPC Designer がある。

ZIPC Designer は入力（イベント）と状態からなる格子状の状態遷移表を作成し、その各要素に動作を記述することでソフトウェアを設計するツールである。ツールで作成した状態遷移表から、状態遷移（イベントハンドリング）のコードが自動生成され、状態遷移表の各要素に記述した動作のコードと結合され、最終的に単独で動作するプログラムが生成される。

先に述べた通り、mruby は組込みソフト開発を効率化するためのプログラム言語である。状態遷移モデルのソフトウェア開発を mruby に適用することで、ソフトウェア開発効率を飛躍的に高めることができる。すなわち、ZIPC Designer が持つ状態遷移設計の効率化と、mruby による各状態における動作記述が効率化される。

図 4、 図 5 に ZIPC Designer for mruby での状態遷移表の作成画面を示す。従来の ZIPC Designer と同様の操作で状態遷移表を作成できる。「for mruby」で加わった機能は、イベントに対する動作のコードを mruby で記述できるようになった点である。

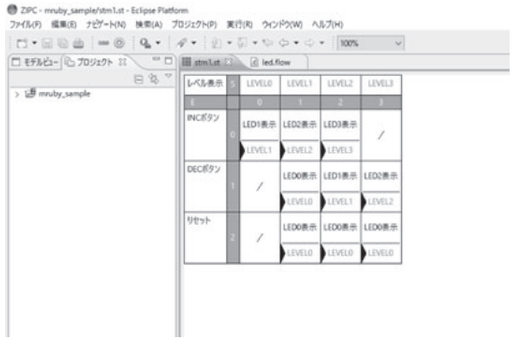

図4　ZIPC Designer for mruby

図5　コード入力

ZIPC Designer for mruby では、状態遷移表をクラスとして自動生成する。アプリケーションは、状態遷移表クラスからオブジェクトを作成し、そのオブジェクトに対してイベント（mruby のメソッド呼び出し）を発生させる。

オブジェクト指向型言語の特徴を活かしたコードが自動生成される。

自動生成されるコードでは、状態遷移全体が一つのクラスとして生成される。開発者は各状態と入力の組み合わせ（表の各要素）に対応するメソッドを記述していく。状態遷移モデルでは、それぞれのメソッドは独立することが望まれ、副作用を発生させない実装が可能である。このことは、モジュールの独立性を高め、最終的には安全なソフト開発につながるといえる。

5. おわりに

筆者はこれまで、mruby の開発を行ってきた。mruby は、プログラム言語 Ruby によって得られる開発効率の高さを組込みソフト開発に適用することを目指した実装である。2012 年にオープンソースとして公開し、この公開と前後して、NPO 法人軽量 Ruby フォーラムを設立し、mruby の普及活動を行ってきた。しかし、具体的な製品への適用となると、その設計・開発環境や、テスト手法の提供など課題も明らかになった。

2014 年にはデバッガをリリースし、ソフトウェア製品の開発で不可欠なデバッグ工程のツールの提供を開始した。この頃から、実際の製品への採用事例も報告されてきている。

今回、開発初期の段階で必要となる設計ツールで mruby がサポートされた意義は大きい。すでに多くの採用実績を持つ ZIPC Designer を使ったソフトウェア開発ができるようになることで、信頼性の高い製品開発が可能となるだろう。

今後、mruby を利用して組込みシステムの開発が活発化することを願っている。

関連情報

mruby は MIT ライセンスを採用したオープンソースとして公開している。商用・非商用を問わず無償で利用でき、ソースコードの開示などの制約もない。

mruby は以下のサイトからダウンロード可能である。

https://github.com/mruby/mruby

mruby のサポート等の維持を行っている NPO 法人軽量 Ruby フォーラムでは、ドキュメントやセミナー等の情報提供を行っている。

http://forum.mruby.org/