Vol.2015-SLDM-170 No.30
Vol.2015-EMB-36 No.30
2015/3/7

状態遷移モデル記述言語stmcを用いた組込みソフトウェアのためのモデルレベルデバッグツールの開発

等々力 拓也[1]　渡辺 晴美[2]　小倉 信彦[1]

概要：モデル駆動開発を活用するためにはモデル記述や変換の他に，開発を支援する機能の充実が重要となる．中でもターゲットの動作をモデルに基づいて操作し設計のミスを明らかにするデバッガは基本的な要素である．本研究では状態遷移図によるモデル記述構文を持つプログラミング言語 stmc を対象にした gdb と共調できるモデルレベルデバッガを開発した．ESS ロボットチャレンジでの開発事例における不具合事例を用いてモデルレベルのデバッグ機能の効果について評価を行い実装の方式について検討を行った．

キーワード：モデル駆動開発, 状態遷移図, デバッグ, stmc

Model Level Debugging Tool of Embedded Systems Software for a State Transition Model Description Language

Takuya Todoroki[1]　Harumi Watanabe[2]　Nobuhiko Ogura[1]

Abstract: In this study, to improve the model level debugging environment of the embedded system software designed MDD, we develop a debug tool for a program language of state transition model, stmc. The debug tool enables to handle and trace the system with using element of designed models. We consider specific model level debug functions and evaluate mechanism of this debugger with bug examples observed in development cases of enPiT PEARL which is an education program for graduate students.

Keywords: MDD, state transition model, debug, stmc

1. はじめに

ソフトウェア開発においてデバッグはソフトウェアの品質や安全性の向上のために必要不可欠であり，デバッガや専用のモニタシステムを用いて行われる．組込みシステムのソフトウェア開発においても通常のソースコードレベルデバッガの持つプログラムのステップ実行や変数などの値を監視をする機能は不具合箇所の特定に役立つが，モデル駆動開発において，不具合の修正をモデルに不具合の原因があることが分からずにコードレベルで行うと場当たり的な変更を引き起こし，作成したモデルとコード間のトレーサビリティが保てなくなることでモデルの価値が失われてしまう．したがって，モデル駆動開発の活用にはモデルの情報が扱えるようなデバッグ支援ツールが必要である．

状態遷移モデルを用いるソフトウェア開発を支援するツールの一つに，C 言語の構文を拡張し状態遷移モデルの振る舞い記述を可能にした stmc という言語がある [1][2]. stmc は状態遷移モデルの要素を言語内に直接記述するためモデル駆動開発で焦点を当てにくいタイミングに関する時間的な問題やハードウェア上の制約などの組込みシステムソフトウェア独特の関心事とモデルの情報を密に扱えるため，デバイスソフトウェアなどの低いレベルで動作するソフトウェアのモデリングと相性が良い．そして，MDD での一つの関心事であるモデルとコードとのトレーサビリ

[1] 東京都市大学大学院 環境情報学研究科
Tokyo City University, Graduate School of Environmental and Information
[2] 東海大学大学院 情報通信学研究科
Tokai University, Graduate School of Information and Telecommunication Engineering

ティについても stmc から C 言語への変換ツールによって保たれている．

stmc を使って作成したモデルは C 言語へ変換されると元のコードに記述されていたモデルの情報の多くが欠落してしまうため通常のデバッガから与えられるソースコードレベルの情報をモデルに対応付けるのが困難である．さらに stmc は状態遷移モデルの振る舞い記述をすることからモデルの情報なしではモデルの欠陥だけではなくコードレベルの欠陥にも気づきにくい．また，stmc と相性の良いハードウェアに近いレベルで動作するソフトウェアは特性上動作が見えづらく，デバッガからの出力はプログラムの動作を理解する重要な情報源の一つである．

本研究では MDD を活用した組込みシステムソフトウェアのデバッグを容易にする事を目的に，作成したモデルの情報が扱える状態遷移モデル記述言語 stmc を対象にするデバッグツールの開発を行った．まず，必要となるモデルレベルのデバッグ機能について論じるために，組込みシステムのソフトウェア開発で現れる不具合を実際に行った開発事例から調べ，その不具合が現れた現象，原因，について整理し，STM32F103RC マイコンを対象に動作するモデルレベルデバッグツールを開発した．

なお，本研究と本研究で用いた開発事例は大学院生を対象にした組込みシステム開発技術の教育プログラムの一環である分野・地域を越えた実践的情報教育協働ネットワーク組込みシステム分野九州大学事業 (enPiT PEARL) および ESS ロボットチャレンジを通して作成されたものである [3].

2. 状態遷移モデル記述言語 stmc

本節ではデバッガのターゲットである状態遷移モデル記述言語 stmc についての説明を述べる．状態遷移モデル記述言語 stmc は本宮らによって開発された組込みシステムソフトウェアのプログラミング言語および支援ツールである．stmc は C 言語の構文を拡張し状態遷移モデルの振る舞い記述を可能にした言語であり，モデルとコードとのトレーサビリティを向上させる為のツールとして C 言語のコードや状態遷移図に変換するトランスレータを持つ．

stmc は状態遷移図を定義するモデル記述部と遷移のトリガとなるイベントの発行タイミングやモデルで表現しない処理を記述する C 言語記述部の 2 つを混在して記述できる．stmc を用いた開発では C 言語の記述とモデルの作成が密な関係にあることで MDD では扱いづらいタイミングに関する時間的な問題やハードウェア上の制約などの組込みシステム開発独自の関心事を整理しやすいという特徴がある．

3. モデルレベルのデバッグ機能

本節では，状態遷移図を用いたソフトウェア開発のためのデバッガに求められる機能について述べるために，enPiT PEARL で行われた PBL における組込みシステムのソフトウェア開発で現れた不具合をその内容と原因について類型化を行った．これらは状態遷移モデル記述言語 stmc による状態遷移図の定義を用いた掃除機ロボット課題や飛行船課題，数個のセンサを使った簡単なマイコンプログラミング演習などの開発事例である．また類型化を踏まえモデルレベルのデバッグに必要な機能について述べる．

3.1 不具合の分類

発生した不具合事例をその不具合が発生した際にどう見えるかにより類型化を行った．1 つは，システムの動作や処理のシークエンスが止まってしまうような動作が止まってしまう不具合と，もう一つは動作のトリガに対して予想とは違う動作をするような予想とソフトウェアの動作に乖離がある不具合である．

3.1.1 動作が止まる不具合

開発事例で発生したシステムの動作や処理のシークエンスが止まってしまう不具合を 3 つのタイプに分けた．

- 電源を入れても動かない (b-1)
- 動作のトリガの受け付け待機のまま動かない (b-2)
- ループ動作から抜け出さない (b-3)

(b-1) はハードウェアの設計，組み立てに関するミスやスタートアップルーチンの設定ミスなどが原因となってプログラムが始まらないという不具合であった．例えば，リアルタイム OS の導入時，使用しているマイコンに沿ったクロックの設定が行われておらず，main 関数が呼ばれないということが起きた．

(b-2) は通信モジュールなどの動作でシステムが待機状態になるような場合にその状態から動作しなくなってしまう不具合で状態遷移のトリガとなるイベントの発行やガード条件の判定，状態遷移の漏れ抜けが原因で発生していた．例えば，開発事例の中でターゲットシステムとホストコンピュータとの通信を状態遷移図を使って表現し，送受信が可能かどうかを表すフラグが立った時に状態遷移を起こすようなプログラムを作成したが，フラグを判定するガード条件にミスであったためプログラムが送受信の待機から動かなくなってしまった．

(b-3) はセンサからの入力や変数の値が既定の値になるまで続けるようなループ処理においてループの継続判定や，それに関わる処理にミスがあることでシステムがループから抜け出せない不具合であった．例えば，開発事例ではロボットが一定の角度を回転するような動作において，回転動作中に角度を表す変数の更新を行っていなかったため回転し続けてしまった．

3.1.2 予想とソフトウェアの動作に乖離がある不具合

開発事例において動作のトリガに対して予想とは違う動作を起こすような不具合を 3 つのタイプに分ける．

- 特定の入力に対して正しい動作を行わない (b-4)
- 同じ入力に対して正しく動作するときとしないときがある (b-5)
- 時間等のある尺度以上，動作を継続させると正しく動作しなくなる (b-6)

(b-4) は (b-2) と同じように状態遷移のトリガとなるイベントの発行やガード条件の判定が正しく行われていないことが原因である他に，入力された値を格納する変数に問題があることもあった．例えば，開発事例で使用した CPU の int 型のビット数とホスト PC から入力される int 型のビット数が異なっていたことでターゲットシステムが負の値を受信した際に正しいふるまいを行わないという不具合があった．

(b-5) はシステムに対して一定の入力を行った際に，正しく振る舞うときとそうでないときがランダムで起こるという不具合で，イベントの発行タイミングやセンサ値に含まれるノイズ処理の漏れ抜けが原因で発生していた．例えば，開発事例においてボタンの入力チェックを常に呼ぶようなプログラムを使用した際，すでに入力が行われている時の処理がなかったため，ボタン入力中は常にイベントが発行されてしまい，アクションの結果がランダムに見えるという不具合があった．

(b-6) はシステムを継続させて動作させたとき，時間や距離などの尺度が一定値を超えたときにシステムが正しく動作しなくなる不具合で，尺度を表す変数の更新をする処理の漏れ抜けや蓄積した誤差が原因である．例えば，開発事例においてロボットを指定距離直進させるプログラムを作成した際に，ロボットから得られる距離情報がオーバーフローした場合に処理を行わなかったためある指定距離から先は正しく動作しなくなってしまった．

3.2 モデルレベルのデバッグ機能への要求

状態遷移モデルなどはシステムの動作を整理するための道具であるため，モデルレベルのデバッグ機能はシステムの動作とモデルを対応付けることで不具合が起きた時にシステムがどこまで正しく動作しているか，誤った処理はプログラムが本来意図していた動作とどれほど差異があるのかをユーザが調べやすくすることが求められる．本研究ではモデルレベルのデバッグ機能をブレークポイントの設置やステップ実行などのソースコードレベルデバッガで与えられる代表的な機能を基に考え，類型化を行った (b-1)〜(b-6) を元にモデルレベルのブレークポイントの設置個所やステップ実行などが与える効果を述べる．

(b-1) のようにスタートアップ時に問題がある場合，扱った開発事例ではスタートアップコードをモデルで表現することはしなかったが，ハードウェアの初期設定を通信で受け取って反映させる場合など立ち上がりの処理をモデルで表現することは十分にあり得る．その際，動作が止まってしまうなどの不具合が確認された時ユーザが知りたい情報はどこまで正しく動作しているかという情報であり (b-2) や (b-6) の場合も同様である．システムの振る舞いをどのようにモデルで表現するかは自由であるが，開発事例においては送られてきたデータを通信プロトコルに照らし合わせる処理や全体の処理の進み具合をモデルで表現する事があった．不具合箇所の絞り込みが行うために，前者の場合，送られてきた値や状態遷移のログ，後者の場合は状態にブレークポイントを置き状態遷移を 1 ステップとしたステップ実行を行い変数やレジスタの値の変化を調べられるとよい．

(b-4) や (b-5) では正しく動作するときとそうでない場合で入力された値やその処理にどのような違いがあるのかがユーザの関心事になる．さらに，(b-6) の不具合はある瞬間の出来事だけが不具合の原因とは考えられないため，正しく動作しているときとそうでないときの違いは，どこまで正しく動作しているかの情報と同じように不具合原因の特定を助ける．システムが正しい動作をした時とそうでないときの違いを明らかにするための道具としても状態遷移のログと遷移毎のステップ実行は有効であり，様々な入力に対しテストを行わなくてはならないことを考えると状態ではなくガード条件の判定式や特定のアクションにブレークポイントを設置できるとよい．

状態遷移モデルを使用したソフトウェア開発のデバッグにおいて (b-1)〜(b-3) のようにシステムがそれ以上先の処理へ進めなくなってしまうような場合，設置したブレークポイントが役に立たなくなる可能性がある．そのため，「イベントが一定時間起こらなかったら」，「指定した状態に到達しなかったから」などの時間的な条件も扱えるようにウオッチドッグタイマと連携したモデルレベルデバッグ機能も有用である．

ユーザがモデルレベルのデバッグ機能を使って知りたい情報はシステムがどこまで正しく動作しているかと正しい動作と正しくない動作の違いである．それを明らかにするためには，少なくとも状態にブレークポイントが置けることや状態遷移のログ出力，遷移ごとのステップ実行，ウオッチドッグタイマによる時間的条件でのブレークがモデルレベルデバッガに必要であると考えた．

4. モデルレベルデバッガ

状態遷移図に欠陥がないかを調べるためにソースコードレベルのデバッグ機能に加えて，状態にブレークポイントが置けることや状態遷移のログ出力，遷移ごとのステップ実行など状態遷移図の要素をデバッグ機能のターゲットにする機能を持つモデルレベルデバッガの開発を行った．本デバッガは stmc の C 言語変換器から得られたソースコードを STM32F103RC マイコン用の ARM 用 gcc でコンパイルされた elf バイナリを対象にしており，OpenOCD を

Vol.2015-SLDM-170 No.30
Vol.2015-EMB-36 No.30
2015/3/7

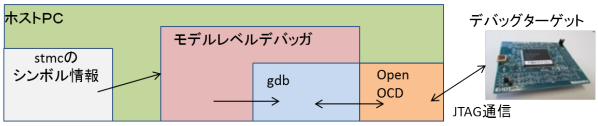

図 1 システム全体の概要図

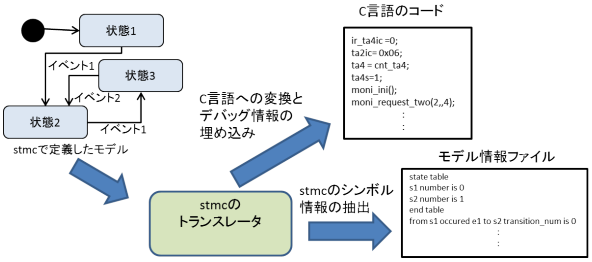

図 2 モデルレベルデバッグのためのトランスレータの動作

介した JTAG 通信による flash 書き込みや gdb を使ったデバッグが行える環境で動作する．システム全体の概要を**図 1** に示す．

開発したモデルレベルデバッガのポートは，ターゲットボードに JTAG ユニットが付属している事とデバッグターゲット向けの gcc と gdb がサポートされている事が条件である．さらに，OpenOCD の動作に必要な JTAG アダプタや開発ボードの設定ファイルが OpenOCD から予め準備されている場合ポートが容易に行える．

本デバッガが状態に置けるブレークポイントや遷移のステップ実行などのモデルレベルデバッグ機能を実現するための構成要素であるモデル情報の抽出，ターゲットシステムの操作を行う機構の 2 つの部分について述べる．

モデル情報の抽出を行う部分は，stmc のトランスレータによる構文解析の結果からモデルレベルのデバッグ機能に必要なシンボル情報を抽出しモデルレベルデバッガが読み込むモデル情報ファイルを生成する．また，モデルとコードの対応付けが難しい要素については変換された C 言語内部にデバッグに用いる情報を埋め込んでいる．トランスレータの動作の概要を**図 2** に示す．

ターゲットシステムの操作を行う部分はトランスレータより得られた外部ファイルから取得したモデルと C 言語コードの対応付けを表した情報を用いて，モデルレベルのデバッグ機能を gdb のコマンドの組み合わせとして実装し，gdb のコマンドが OpenOCD によってターゲットボードへ JTAG の命令に変換されて送信されることでターゲットシステムの操作や監視を行っている．状態遷移のロギングに関しては，遷移が起こるたびに JTAG 通信を用いて起こった遷移を取得するよりもログデータを一時的にターゲットシステム内に保存しログ出力の際にのみデータを取得する方がシステムの動作に与える影響が少ないと考え状態遷移が起こる部分にロギング関数を埋め込む方法をとった．ターゲットシステムを操作する JTAG 通信と状態遷移のロギングを行う関数の関係を**図 3** に示す．

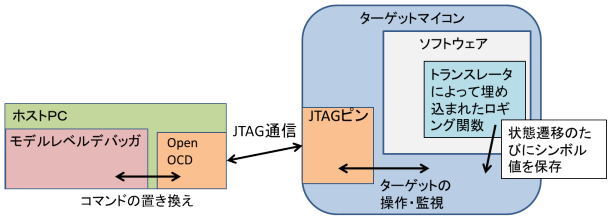

図 3 ターゲットシステムの操作と状態遷移のロギング関数の動作

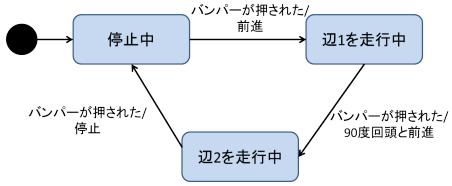

図 4 iRobotCreate の走行プログラムのモデル図

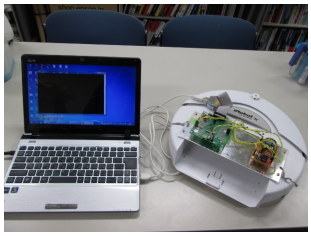

図 5 ロボットとホスト PC

5. 適用例

本節では開発したモデルレベルデバッガが提供する機能が正しく動作する事を ESS ロボットチャレンジ 2014 で作成した STM32F103RC マイコンを対象にした iRobot Create の走行プログラムを用いて示した．ESS ロボットチャレンジ 2014 では掃除ロボットを想定した室内走行ロボット (iRobot Create) のソフトウェア開発コンテストとして開催された．iRobot Create とは iRobot 社が提供する掃除ロボットルンバを元にした研究，教育のためのロボットプラットフォームである．

使用するプログラムは STM32F103RC マイコン上で動作し，ホスト PC との通信なしでロボットの操作を行うプログラムである．ロボットは初め静止しており (停止中)，バンパを押すことで直進を行い (辺 1 を走行中)，その後バンパに感があればその場で停止し，90 度回頭する．解凍動作終了後バンパに感があるまで再び直進を行う (辺 2 を走行中)．この状態遷移図を図**図 4** に示す．そして，iRobot Create に搭載した STM32F103RC マイコンに JTAG アダプタを接続させた様子を**図 5** に示す．

モデルレベルデバッガが提供する機能の動作を確認するために，「辺 1 を走行中」状態にブレークポイントを置き，「辺 1 を走行中」状態に到達後，再び「停止中」に遷移するまで遷移のステップ実行を行った．その結果，開発したモデルレベルデバッガはブレークポイントを置いた状態に到

Vol.2015-SLDM-170 No.30
Vol.2015-EMB-36 No.30
2015/3/7

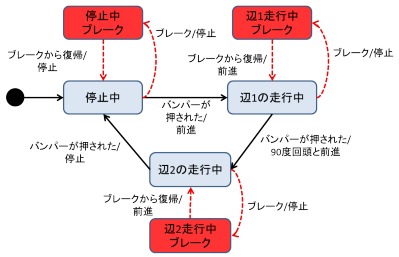

図 6 デバッグのために変化したモデル図

達した際にシステムを一時停止させ，遷移のステップ実行では 1 回の遷移が終了するまでブレークからシステムを復帰させることができる．

6. 考察

本節では適用例の結果についてデバッガによるブレーク時の操作に関する考察と今後の課題について述べる．

6.1 適用結果

適用事例では状態にブレークポイントを置き，状態遷移単位のステップ実行を行う事でシステムの動作の追跡を試みたが，iRobot Create は移動の命令を一度受信すると新しい移動の命令を受信するまではその動作を続けてしまうため，ロボットが走行中にシステムがブレークポイントに到達した場合，ロボットは止まらず走り続けてしまうという現象が確認された．システムの停止中にロボットが走り続けてしまうことはその後の距離センサの値を使った動作に影響が出るだけでなく，ロボットが壁にぶつかることで破損してしまう可能性があるなど安全性の面からも望まれない．デバッガによる単純な実行停止とシステムに望まれた停止状態は一致せず，デバッガによるブレーク時にはシステムはデバッグ時に停止させるための状態に移行する必要がある．適用例ではブレークポイントに到達した際，ロボットへその場で静止するように命令を送り，ブレークからの復帰時にイベントを起こすことでロボットを静止させる前の動作へ戻すようにモデル図を変化させた．変化させたモデル図を**図 6** に示す．

デバッグのためにモデルを変化させることについて，[4] で述べられているように，デバッガはハイゼンバグを起こさないようにデバッグ対象のソフトウェアへ過度の干渉をしてはならないという原則があるため，適用例で行ったような，デバッグ用に遷移や状態を追加が最低限になるようにシステムの停止状態を設計する必要がある．また，デバッグのために用いる特別なモデルが複雑な構造を持ってしまう場合，本来のデバッグ対象のモデルと追加されたモデル部を正常系と非正常系のように両者を切り分けて扱えるとそれぞれの処理について混乱が生まれにくいと考える．

6.2 今後の課題

6.2.1 DWARF-2 情報を使ったデバッグ機能

今後の研究では DWARF の情報を使用して，gdb のコマンドの組み合わせでは実装不可能なモデルレベルデバッグ機能の実現に期待する．DWARF-2 とは実行ファイルの形式である elf ファイルと共に開発されたデバッグ情報のデータフォーマットの形式である．DWARF は Debugging Information Entry (DIE) というデータ構造を使って，変数，データ型，プロシージャなどを elf ファイルに付与することし様々なデバッグ機能を使用できるようにしている．本研究で開発したモデルレベルデバッガの機能の多くは gdb のコマンドを組み合わせることで実現しているが，現在実装している機能だけでは行えない操作がある．例えば，モデル上の 1 ステップの実行が 1 回の状態遷移だけであるとは限らず状態遷移図が階層的な構造の場合を考えると 1 つの状態マシンが最終状態にたどり着くまでを 1 ステップと考えることが出来る．

6.2.2 アスペクト指向技術との連携

6.1 節で述べたデバッガによるブレーク時の操作を含めたモデル図について，適用事例では各状態からのブレーク時の遷移と復帰の遷移のみが元のモデル図に追加されたが，本来のモデル部分とデバッグのために追加されたモデル部分は切り分けて扱うことが望ましい．さらに，遷移に対応するアクションの途中にブレークポイントを置いた時を考えると図 6 は不十分である．例えば，「辺 1 を走行中」から「辺 2 を走行中」への遷移のアクションでは回頭と前進を行っているが前者の途中にブレークポイントを置いた場合と後者の場合とではブレーク後に望まれる復帰処理が異なる．デバッグ時に停止させるための状態やその処理が膨れ上がることでデバッガで扱うモデルが煩雑になってしまうことを避けるため，正常系と非正常系のようにモデルの要素を分ける技術との連携が求められる．そこで，今後デバッガ内でアスペクト指向技術によって状態遷移図の要素を関心事別に扱うことが出来ないか議論する必要がある [2]．

7. まとめ

組込みシステム開発において MDD を活用するツールの一つである stmc を対象にモデルレベルデバッガを開発した．モデルの情報を使ったデバッグに必要な機能について enPiT PEARL および ESS ロボットチャレンジで作成した組込みシステム課題の不具合事例を類型化することで論じた．開発したモデルレベルデバッガは GNU の組込みシステムソフトウェア開発ツールと OpenOCD を合わせた開発環境下で動作する．動作環境下で開発したデバッガが提供するモデルレベルのデバッグ機能が動作する事を iRrobotCreate の走行プログラムを用いて確認を行い，その結果についてデバッガによるブレーク時の操作に関する考察をした．

参考文献

[1] 小倉信彦, 谷川郁太, 渡辺晴美, “状態遷移言語による組込みソフトウェア開発,” 情報処理学会研究報告, Vol.2011-SLDM-149 No.48, pp.1-6, 2011.
[2] 安部昌輝, 川村峰大, 長岡拓弥, 谷川郁太, 原築良, 小倉信彦, 渡辺晴美, “組込みソフトウェアのためのアスペクト指向言語による状態遷移言語の提案,” ESS2011 論文集, pp21.1-21.10, 情報処理学会, 2011.
[3] enPiT 運営委員会, 平成 25 年成果報告書, 大阪大学大学院情報科学研究科 enPiT 事務局, 2014.
[4] Jonathan B.Rosenberg, (訳:吉川邦夫), デバッガの理論と実装, 株式会社アスキー, 1998.