論文

状態遷移モデル記述言語のための複雑度評価ツールの開発

原　築良　　小倉　信彦

大規模化・複雑化するソフトウェアの開発効率向上や欠陥の除去のためには，ソースコードやソフトウェアモデルに対する適切な評価が欠かせない．McCabeの循環的複雑度は，プログラムの制御構造のグラフの循環数をもとにコードの内部構造の複雑性を定義するものである．この指標は欠陥率との強い相関があり，複雑性の評価において有効な尺度であることが知られている．組込みソフトウェアにおいても大規模化・複雑化は顕著である．我々はモデルベースの組込みソフトウェアの開発支援のために，状態遷移モデルをC言語の構文の中に記述できる状態遷移モデル記述言語（stmc）を開発した．本研究ではstmcの評価尺度をMcCabeの尺度に倣って定義し，定義に従った複雑度を計算・表示するソフトウェアを開発した．また，評価の際に発生する状態遷移モデルやstmc特有の問題について考察した．

キーワード：ソフトウェア，評価，状態遷移モデル，stmc，プログラミング言語

1　はじめに

ソフトウェアの大規模化や複雑化に伴って，様々な開発方法論が提案されている．多くの開発法では，開発対象の特徴的な概念を抽出し，抽象化したモデルからソフトウェアを構成する．モデルのひとつである状態遷移モデル[1]は，モデリング対象の置かれた状況と，状態が変化する条件(遷移)を抽出するモデリング手法である．動的な振る舞いを表現できることから，組込みソフトウェア開発において有効であることが知られている．

我々は[2，3]において，状態遷移モデルの状態・遷移・アクションなどの要素を，直接記述できる状態遷移モデル記述言語（stmc）を提案・開発している．これは組込み開発で主流となっているC言語の拡張言語である．開発への導入が容易であり，モデルの変更に伴う実装の変更に強いことが有用性として挙げられ，ソフトウェアの開発効率・保守性の向上が期待できる．

ソフトウェアの品質を保つためには，ソースコードやモデルに対する適切な評価が欠かせない[4]．ソフトウェアに規模や複雑性の評価を与える事は，欠陥の抑制やテスト計画に有効であり，stmcにおいても評価手法の確立が求められている．

stmc で構成されるモデル間の関係が複雑なソフトウェアに対しては，イベント発行関係を静的に解析するツールが提案，開発された[5]．このツールによって，stmcで構成されるソフトウェア全体の構造を捉える事が可能になり，stmcをモデルの相互関係の視点から評価できるようになった．しかしながら，個々の状態遷移モデル内部の構造に対する評価方法は確立されていない．

本研究は，stmcで記述されたモデルに独立した評価を与える事を目的としている．モデルの内部に着目した評価基準によって，モデルの構造の理解・欠陥の予測と発見が容易になると考える．本研究における内的特性の評価が，モデル間のイベント発行関係のような外的特性の評価などと共に，stmcで構成されるソフトウェアの総合的な評価の基礎となることを期待する．

2　準備

2．1　状態遷移モデル記述言語 stmc

システムの状態の変化によって開発対象をモデリングする状態遷移モデルは，時間制約など動的側面が開発時の関心事となる組込みソフトウェア開発などで用いられる．このモデリング手法によって描かれる状態遷移図は，モデリング対象のあらゆる状態を記述する．また，状態に遷移をもたらすトリガ（イベント）と，遷移の条件(ガード条件)や行うべき動作（アクション）を，状態同士を結ぶ向きを持った線で遷移として記述する．

状態遷移モデル記述言語stmcは，組込み開発で広く用いられているC言語を拡張し，状態遷移モデルの記述構文を持たせたプログラミング言語である．stmcにおいては，イベント，状態，遷移，ガード条件，アクションの定義によって状態マシンを構成する．モデルに基づいた抽象度の高い記述を可能にしている．状態遷移モデルは，C 言語の関数と混在した形で定義することができる．stmcで記述されたプログラムはC言語へ変換され，実行される．

HARA Chikura
東京都市大学環境情報学部情報メディア学科2011年度卒業生
OGURA Nobuhiko
東京都市大学環境情報学部情報メディア学科准教授

２．２　stmc の記法

stmc のプログラムは，リスト１のような形で記述される．モデルの宣言を行い，C 言語記述部からプログラムの動作に応じてモデルが呼び出され，イベントが発行される．

```
#include <stdio.h>
//モデルの宣言
stm_def sample
{
    /*モデル記述*/
} model;
//関数の宣言
int main()
{
    model.event_a();
}
```

リスト１　stmc の記述

モデルの宣言はリスト２のように記述される．stm_def はモデルの宣言の開始を意味し，構造体のように状態モデル型名を記述する．波括弧{}の中に，状態・遷移・イベント・変数の宣言などを記述する．data: struct{};にモデルが持つ変数を宣言する．state:, event:, transition:に続けて，状態・イベント・遷移を定義する．波括弧{ }の後にモデル名を記述する．

```
stm_def 状態モデル型名
{
data:
        struct
        {
           /*モデルの変数宣言*/
        };
state:
   /*状態の定義*/
event:
   /*イベントの定義*/
transition:
   /*遷移の定義*/
}モデル名;
```

リスト２　モデルの宣言

２．３　McCabe の循環的複雑度

ソフトウェアの評価には，モデルやモジュールの設計やコードの内部に踏み込んだ，内的特性に着目するものがある．内的特性に着目した評価は，詳細な分析単位をもつため，実装段階における品質の確保に有効である．内的特性の評価尺度の代表的なものに，McCabe の循環的複雑度がある[６]．McCabe はシーケンシャルなプログラムステートメントと，それらを結ぶプログラムフローからプログラムをグラフ化し，グラフの循環の数をグラフ理論における循環数に倣って循環的複雑度とした．McCabe の循環的複雑度 Mは以下の式によって定義される．

$M = e - n + 2p$

e ：エッジの数(ノード間のフロー)

n ：ノードの数

（シーケンシャルなステートメントの集合)

p ：非連結パスの数(独立したグラフの数)

C 言語のような手続型言語の場合は，プログラムをノードやエッジの概念に置き換える事無く，条件分岐の数を加算することにより,循環の数を数えることができる．循環的複雑度は加算式であるので，いくつかのグラフの集合の複雑度は各グラフの複雑度の合計によって求められる．この尺度は，繰り返しと２つに分かれる条件分岐を同じ複雑さを持つものとして扱う．また，条件分岐と条件分岐の間に沢山のステートメントが含まれていても，ステートメントが分岐でない限り複雑度には影響しない．

３　stmc の複雑度の定義

状態遷移図(図１)は，状態のノードと遷移のエッジから構成されるグラフ(図２)と捉えることができる．このグラフから，McCabe の循環的複雑度に倣った状態遷移モデルの複雑度 Sを定義した．このような状態遷移モデルに McCabe の尺度を用いる手法は,いくつかの研究で採用されている[７，８，９]．

独立したモデルの数 pにかかる係数は，独立したモデルの数が１である場合と２である場合のギャップの大きさを決定する．係数が大きくなるほど独立したモデルの数の複雑さは増す.ここでは McCabe の尺度に倣って２を係数としている．

$S = e - n + 2p$

e ：遷移の数

n ：状態の数

p ：独立したモデルの数

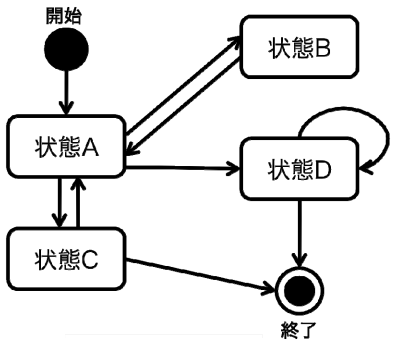

図１　状態遷移図

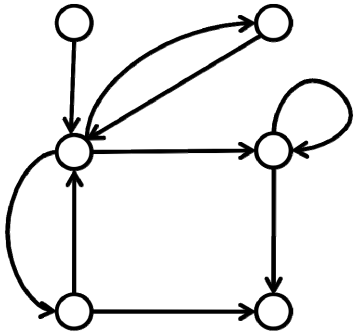

図2　グラフ

遷移のアクションの定義はC言語で記述される（リスト3(b)）．このため，アクションの評価にはMcCabeの循環的複雑度を用いる．stmcでは，状態から状態へ共通のアクションで遷移する場合，複数状態と一つの状態を結ぶ複数の遷移をまとめて記述することが可能である（リスト 3(a)）．アクションの記述はあくまで１つなので，コードの読みやすさの観点を優先し，本研究では１つのアクションとして独立した複雑度を算出するものとする．よって，遷移のアクションについては遷移の数は考慮せず，定義されたアクションの循環的複雑度のみを算出する．前述の通り，循環的複雑度は加算式であるので，モデルで宣言されたアクションの循環的複雑度a_nを合計し，モデルのアクションの複雑度Aとする．

$A = a_1 + a_2 + a_3 + \ldots + a_n$

A：モデルのアクションの複雑度

a_n：遷移毎の各アクションの複雑度

また，モデルのアクションの複雑度Aは，モデルの複雑度Sに加算する．

アクションの複雑度を含めたモデルの複雑度 SA を以下に定義する．

```
/* 遷移の定義 */
void event()
state1
, state2
, state3
->end_state              //----------(a)
/* アクションの定義 */
{
   if(result(a)==0){     //----------(b)
      variable=data1;
   }else{
      variable=data2;
   }
}
```

リスト３　アクションの定義の例

$SA = e - n + 2p + A$

C言語は手続き型言語であるので，C言語で記述されたstmcの関数部はMcCabeの循環的複雑度Mをそのまま適用する．McCabeの循環的複雑度の加算性から，関数部全体の複雑度は各関数の複雑度Mの合計としている．

循環的複雑度の加算性から，状態遷移モデルの複雑度の合計と，関数の複雑度の合計の和を，ソースコード全体の複雑度とした．

４　ツールの概要

本研究では，前章の定義に従った複雑度評価ツールを開発した．状態遷移モデル記述言語stmcで記述されたソースコードを解析し，前章で述べたstmcの複雑度の定義に従った複雑度の計算・表示をするものである．１つのモデルに対し，表１の情報が与えられる．

表１　モデルの複雑度の情報

position	宣言が記述された行番号
name	モデル名
modelcomp	状態と遷移から決定される複雑度
states	状態の数
transitions	遷移の数
actioncomp	アクションの複雑度の合計
complexity	モデルの複雑度

これらの情報は，リスト４のように表示される．

```
-----stm------
position:256
name:stm_new_model
modelcomp:2
states:3
transitions:4
actioncomp:5
complexity:7
--------------
```

リスト４　モデルの複雑度の情報

この情報から，定義されたモデルの複雑度を理解できる．遷移やアクションの定義の数が表示されるため，規模も理解できる．

また，１つの関数に対して表２の情報が与えられる．

表２　関数の複雑度の情報

position	宣言が記述された行番号
name	関数名
complexity	関数の循環的複雑度

これらの情報はリスト５のように表示される.

```
---function---
position:1024
name:main
complexity:18
--------------
```

リスト５　関数の複雑度の情報の表示例

関数の循環的複雑度は，条件分岐文と繰り返し文を数えることにより求める．また，条件分岐と繰り返しを区別しない．さらに，if と switch や for と while を区別しない．このため，分岐の種類などを分類して表示する方法は採用していない．complexity から１を減算したものが，if，else-if，for，while，case などの分岐の数と読み取れる.

これらに加えて，表３の情報が計算される.

表３　総合的な複雑度の情報

funcnum	**関数の数**
stmcnum	**モデルの数**
funccomp	**関数の複雑度の合計**
stmccomp	**モデルの複雑度の合計**
sum	**コード全体の複雑度**

これらの情報はリスト６のように表示される.

実際には，コードに記述されたモデル・関数の宣言の順番に従って，モデルのデータ（リスト４）や，関数のデータ（リスト５）が表示される.

```
==complexity==
funcnum:8
stmcnum:3
funccomp:21
stmccomp:171
sum:192
==============
```

リスト６　総合的なデータの例

５　状態遷移モデルの複雑度計算方法

stmc の状態遷移モデル記述部においては，状態の数・遷移の数を数えれば，状態遷移モデルの複雑度 SA が計算できる．状態遷移モデルの状態の定義はリスト７(a)，(b)，(c)のように記述される．この例では３つの状態がある．さらに，遷移での定義がリスト７ (d)，(f)，(h)において記述されている．リスト７(d)では，２つの状態から１つの状態への遷移が定義されているので，ここで定義される遷移の数は２である．リスト７(f)，(h)は１つの状態から１つの状態へ遷移するものなので，遷移の数はそれぞれ１である．よって，合計の遷移は４である．遷移に対応するアクション(リスト７ (e)，(g)，(i))は C 言語で記述されているので，次節の C 言語の循環的複雑度を算出する．アクションの波括弧{}内に処理が無くても(リスト７(e))，開始と終了を結ぶエッジのあるグラフとして循環的複雑度１を加算する．複数の遷移がアクションを共有する場合の扱いについては，前節で述べた通りである.

```
stm_def
{
state:
   normal_mode = {is_initial,"normal mode is working"};              //----(a)
   insert_mode = {"insert mode is working"};                //--------------(b)
   visual_mode = {"visual mode is working"};                //--------------(c)
event:
   void esc_typed() "key esc typed";
   void i_typed() "key i typed";
   void v_typed() "key v typed";
transition:
   void esc_typed()
   insert_mode,visual_mode -> nomal_mode                    //--------------(d)
   {}                         //-------------------------------------------(e)
   void i_typed()
   nomal_mode -> insert_mode                  //---------------------------(f)
   {
        printf("insert¥n");                      //-------------------------(g)
   }
   void v_typed()
   nomal_mode -> visual_mode                   //---------------------------(h)
   {
        printf("visual¥n");                    //---------------------------(i)
   }
}vim;
```

リスト７　モデルの定義の例

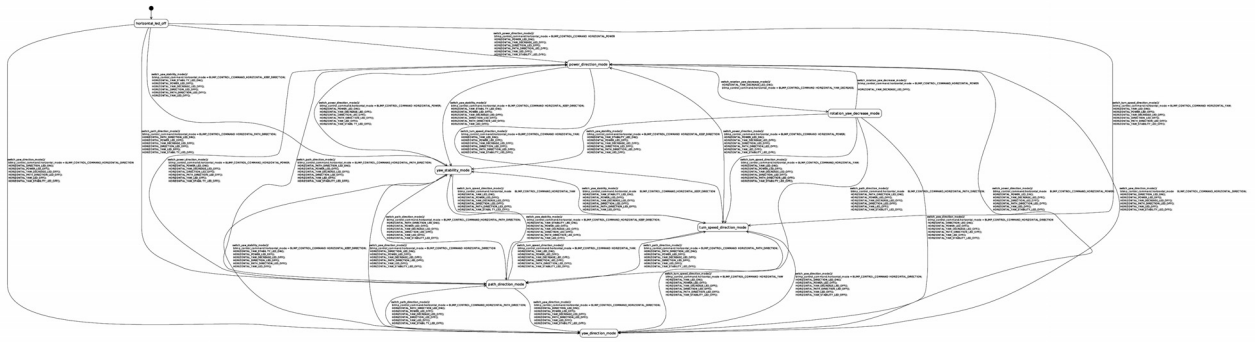

図３　sa における状態遷移図の表示例（この図は，状態と遷移が識別できる倍率で縮小している．）

このプログラムは，状態：３，遷移：４，である．C 言語の循環的複雑度の計算方法を用いて算出するとアクションの複雑度：３である．状態遷移モデルの複雑度の定義 $SA = e - n + 2p + A$ から，このモデルの複雑度は６となる．

６　ツールの適用

本研究で開発した複雑度評価ツールを，２つの stmc プログラム sa，sb に適用した（表４）．どちらのプログラムも，stmc によるコードと，本宮の stmc から C 言語に変換するツールを適用した C 言語によるコードに対して複雑度を計算している．ともに MDD ロボットチャレンジ[10]の競技用に開発されたソフトウェアである．

２つのプログラムの４つのコードにツールを適用した結果を，木構造表示ツールやソースコードと比較し，行番号やモデル名が適切に表示され，状態や遷移の数が正確に数えられている事を確認した．また，モデルや関数をグラフととらえたとき，*SA* や *M* が定義通りに算出されていることを確認した．

表４　ツールを適用したソフトウェア

sa	小型飛行船用リモコンファームウェア
ca	sa を C 言語に変換したもの
sb	小型飛行船自働航行のための地上基地局コンピュータ用プログラム
cb	sb を C 言語に変換したもの

７　考察

適用結果をもとに考察を行った．sa における，あるモデルのデータの表示結果はリスト８のようになった．

このモデルの状態遷移図は図３のようになる．複雑度評価ツールによって，この状態遷移図の構成要素である状態・遷移の数と，アクションの循環的複雑度の合計が計算できる．状態・遷移の数から算出されるモデルの複雑度が定義の通り計算できている事が確認できる．

```
-----stm------
position:274
name:stm_horizontal_switch_led
modelcomp:25
states:7
transitions:31
actioncomp:7
complexity:32
--------------
```

リスト８　適用例 sa におけるデータの表示結果

アクションの複雑度を算出する際，複数の遷移で１つのアクションが共有される事は考慮せず，アクションの記述の複雑度のみを加算した．このことは，読みやすさの面ではコードの複雑性を適切に表現していると考える．リスト８では，モデルの遷移の数 transitions=31 に対して，ソースコード中の遷移記述の数 actioncomp は７以下であり，大きく異なる．これはモデルの遷移構造の複雑さとソースコードの読みやすさに乖離がある例となっている．

一方で，複数の遷移で共有されるアクションの複雑度を，遷移の数とアクションの複雑度の積とすれば，状態と状態を結ぶ全ての遷移の複雑性か算出され，テスタビリティの評価においては有益であると考える．

本研究では，アクションとモデルの複雑度を加算している．アクションがどの程度モデルの複雑性を増やすものであるかは，開発事例による検証を行う必要がある．アクションの複雑性と欠陥の相関が低いものであれば，$SA = e - n + 2p + A$ における A にかかる係数を０に近づける必要があると考えられる．また，アクションの複雑性とモデルの複雑性を分離して計算することが評価としてより有益である可能性がある．これらの点は，欠陥率などと比較しながら適切な計算方法を確立することが求められる．

状態遷移モデルでは，イベントが発行されガード条件を満たしたとき，状態が遷移する．本研究ではガード条件を評価尺度の要素として扱っていないが，遷移の有無を決定する条件分岐として複雑性を増やす要素と捉えることもできる．

アクションの描くグラフをモデルのグラフと同等に扱えるものとすれば，分岐のひとつあるアクションは図４のように描くことができる．この場合，アクションの開始ノードと終了ノードは遷移前と遷移後の状態であるので，アクションの複雑度計算では，開始・終了ノードを排除しなければならない．

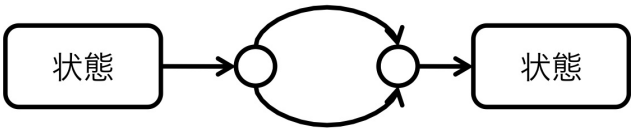

図４　分岐の一つあるアクション

構文規則に則った構文木を読み取る作業は，コードの解析において必要不可欠な作業となる．このような解析は，本多・市川のツールを始め多く行われている．本研究においても，stmc の構文規則従って解析器を開発した．構文木から構文規則に則って必要な終端・非終端記号を求める場合，構文解析プログラムの記述からある程度機械的に必要な処理が決定する．このことから，構文規則から逆算して解析に必要な処理のライブラリを自働作成するツールの必要性を感じた．このライブラリを通して，コード解析ツールの作成・適用実験が容易に実現できると考える．

７．１　今後の課題

開発したツールでは，グラフの循環に基づく複雑度の加算性を前提として，各モデルの複雑度の和を算出している．この循環的複雑度の加算性がモデルに対しても成り立つか検証する必要がある．加算性がない場合，個々のモデルの複雑度を，複数のモデルに対する複雑性の評価をどう活かすか検討しなければならない．また，本多により提案されたモデル間のイベント発行関係評価ツールなどと併用し，モデル内とモデル同士を総合的に評価する手法の確立が期待される．

今後，stmc のプログラム開発の段階から，複雑度評価や欠陥率・修正回数などの統計を取る事により，本研究の複雑度定義を評価・改良することが必要となる．

８　関連研究

本宮らによってstmcをC言語に変換するツールが開発されている[２，３]．また，本多によって stmc のイベント発行関係を静的に解析するツールが提供されている[５]．更に，市川によって stmc のモデル記述から状態遷移図を作図するツールが提供されている[11]．

[７]では，状態遷移モデルの状態と遷移の数から，循環的複雑度を計算する手法や，Douglass の提唱する階層化状態遷移モデルの複雑度 Douglass Cyclomatic Complexity が紹介されている．この研究は，本研究と同様に循環数を用いて状態遷移モデルを評価するが，アクションの複雑度を考慮していない．

Wagner と Jürjens は，[８]で手続き型言語で記述された状態遷移モデルの循環的複雑度 CCS (Cyclomatic Complexity of State machine)の計算手法を提案している．ここで定義されている CCS は循環数を用いて評価しているが，その循環数はコード生成したプログラムの制御フローを用いている点が本研究とは異なる．

Clements，Kazman，Klein は[9; 9.2.1 節]において，状態遷移モデルやクラスの様々な評価の指標を整理している．例えば，number of events (同期・非同期の関数呼び出しの数)，depth of state machine (階層化された状態モデルの階層間の非直接性の指標となる階層数)，number of extended state variables (状態モデルの同期面を測るための状態変数の数)が挙げられている．

安倍らは[12]で，stmc 言語を拡張し，アスペクト指向技術機能をもたせた言語を提案している．この方法によって状態遷移モデルの合成が可能になる．この手法で，見かけの複雑性を下げるものであると考えられる．

９　おわりに

本研究では，状態遷移モデルの複雑度を定義した．stmc で記述されたコードに，定義に従った評価を与えるツールを開発した．このツールをいくつかのプログラムに適用し，定義通りの計算ができることを確認した上で，さらに評価の妥当性や有用性を高めるための考察を行った．

参考文献

[１] David Harel, “Statecharts: A visual formalism for complex systems,” *Science of Computer Programming*, Vol. 8, No. 3, pp. 231-274, 1987.

[２] 本宮茂雄，“組込みシステム開発のためのモデリング言語およびモデルベース開発方法の提案，”修士論文，武蔵工業大学，2010.

[３] 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.

[４] Stephen H. Kan, *Metrics and Models in Software Quality Engineering (2nd Edition)* Peason Education, 2003.

[５] 本多祐美子，“状態遷移モデルのイベント発行関係の静的コード解析ツールに関する研究，”卒業論文，武蔵工業大学，2009.

[６] T. J. McCabe, “A Complexity Measure,” *IEEE Transactions on Software Engineering*, Vol. 2, No. 4, pp. 308-320, 1976.

[７] B. P Douglass, “Computing Model Complexity,”

White Paper, I-Logix, 2004.
[8] Stefan Wagner, Jan Jürjens, “Model-Based Identification of Fault-Phone Components,” *Dependable Computing* - EDCC-5 pp.435-453, 2005.
[9] Paul Clements, Rick Kazman, Mark Klein, *Evaluating Software Architectures*, Peason Education, pp.264, 2002.
[10] 小倉信彦, 渡辺晴美, “ロボットコンテストを利用した組込み教育の実践,”情報処理学会論文誌, Vol.49, No.10, pp.3531-3540, 2008.
[11] 市川清美, “組込み開発のためのモデル記述言語 による状態遷移図の生成,” 卒業論文, 武蔵工業大学, 2009.
[12] 安倍昌輝, 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, p21.1, 情報処理学会, 2011.