**B-013**

# 組み込みソフトウェア向け UML における状態遷移表・ソースコード変換方式

小澤　陽平[†]　　細川　卓誠[††]　　小泉　寿男[†]

## 1　はじめに

近年の組み込みソフトウェア開発では，制御だけに限らず，データ処理，音声や動画の再生，ネットワークへの対応，GUI 等により複雑化してきている．一方，大規模化しているにもかかわらず，開発期間は短期化する傾向にある．そのため，組み込みソフトウェアにも UML を用いる試みがなされている．しかし，本来 UML が用いられてきたビジネス系アプリケーションでは，データ処理が中心であることに対し，組み込みソフトウェアでは，制御が主体である．組み込みソフトウェアの開発には，ユースケースとシステムの機能要件との対応関係の明確な記述や，実装に対してシームレスな分析・設計手法が必要である．

これに対し，シュレイアー・メラー法[(1)]や，実践原則をガイドラインとしてまとめた eUML[(2)]が提案されているが，この方式でも分析・設計から実装までのシームレスな開発手法が必要である．

筆者らは，組み込みソフトウェアの UML 適用という問題に対して XUC 図（eXtended Use Case:拡張ユースケース図）を提案した[(3)]．XUC 図は，ユースケース図を組み込みソフトウェア向けに拡張したもので，新たにコントローラとアクションと呼ばれる要素を追加している．これにより，システムの機能要件を定義する過程を通じて分析と設計をシームレスに行うことを目指している．

本稿では，組み込みソフトウェア開発の分析・設計から実装の工程に円滑に進行させるための一環として，XML を用いて XUC 図を元に記述した状態遷移表からソースコードを生成する．具体的には，XUC 図を元に記述した状態遷移表を XML で記述し，状態遷移表を記述した XML ファイルを状態遷移表/ソースコード生成機構に通す．これにより，ソースコードのスケルトンを生成する方式である．

## 2　XUC 図から状態遷移表 XML 記述

XUC 図では，機能要件を補完する目的で状態遷移図，状態遷移表，シーケンス図を用いる．そこで，本稿では状態遷移表を用いてソースコードの生成を行う．XML のデータをタグを用いて記述するという特徴により，HTML のテーブルタグ（TR（行）タグ，TD（列）タグ）のように状態遷移表を記述することが可能である．XUC 図の記述からソースコード生成の流れを図１に示す．

## 3　状態遷移表/XML 記述規則

XUC 図の補完のために作成した状態遷移表から，ソースコードのスケルトンを生成するために，新たに状態遷移表を XML で記述するための記述規則を提案した．(図２参照)

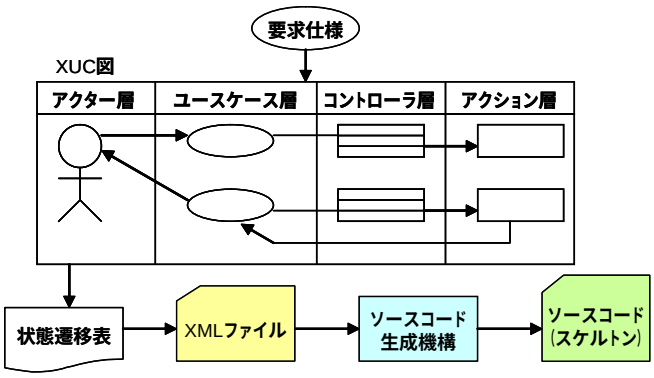


図１　XUC 図の記述からソースコード生成の流れ

**状態遷移表**

| S＼E | e1 | e2 | E3 | e4 |
|---|---|---|---|---|
| S1 | A1()/S2 | -/- | -/- | -/- |
| S2 | -/- | A2()/S3 | -/- | A4()/end |
| S3 | -/- | -/- | A3()/S1 | -/- |

提案した規則によりXMLで記述

**状態遷移表を記述したXMLファイル**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE state_chart_table[･･･]>

<XML_for_XUC>
<state_chart_table>
<title>XML Document from State Chart Table</title>
<state state_name="S1">
<event event_type="e1" action="A1()" next_state="S2" />
<event event_type="e2" action="NOTHING" next_state="NOTHING" />
<event event_type="e3" action="NOTHING" next_state="NOTHING" />
<event event_type="e4" action="NOTHING" next_state="NOTHING" />
</state>
<state state_name="S2">
<event event_type="e1" action="NOTHING" next_state="NOTHING" />
<event event_type="e2" action="A2()" next_state="S3" />
<event event_type="e3" action="NOTHING" next_state="NOTHING" />
<event event_type="e4" action="A4()" next_state="end" />
</state>
<state state_name="S3">
<event event_type="e1" action="NOTHING" next_state="NOTHING" />
<event event_type="e2" action="NOTHING" next_state="NOTHING" />
<event event_type="e3" action="A3()" next_state="S1" />
<event event_type="e4" action="NOTHING" next_state="NOTHING" />
</state>
</state_chart_table>
</XML_for_XUC>
```

図２　状態遷移表の XML 記述

提案した記述規則に基づいた状態遷移表の記述例を表１に示す．HTML の TR（行）タグが XML 記述規則の state タグ，TD（列）タグが event タグに相当する．state タグの属性 name に状態名を記述し，このタグの間に event タグを記述することで，各イベントごとの動作を指定する．

---

A Method of Transrarion from StateChartTable to SourceCode in the Development of Enbedded Software using UML.

[†]Depertment of Computers and Systems Engeneering
[††]Graduate School of Science and Engineering,
Tokyo Denki University

表１　状態遷移表の XML 記述例

```
<XML_for_XUC><state_chart_table>
<state state_name="S1">
<event event_type="e1" action="A1()" next_state="S2" />
<event event_type="e2" action="NOTHING" next_state="NOTHING"/>
<event event_type="e3" action="NOTHING" next_state="NOTHING" />
<event event_type="e4" action="NOTHING" next_state="NOTHING" />
</state></state_chart_table><XML_for_XUC>
```

表１において，状態 S1 の時，各 event タグは，

・イベント（属性 event_type）e1 が発生した場合アクション（属性 action）A1()が起動し，次の状態（属性 next_state）S2 へ移る．

・イベント（属性 event_type）e2，e3，e4 が発生した場合アクションは何も起動せず（属性 action の値は NOTHING），状態の遷移はない．（属性 next_state の値は NOTHING）XML 記述規則のタグの定義である DTD（Document Type Difinition:文書型定義）を表２に示す．

表２　状態遷移表の XML 記述規則の DTD

```
<!DOCTYPE state_chart_table[
  <!--要素 state_chart_table は，ルート要素で，要素 title,
    state から構成されていて，state は１回以上繰り返して出てくる-->
  <!ELEMENT state_chart_table (title，(state)+)>
  <!--要素 event は１回以上繰り返して出てくる-->
  <!ELEMENT state (event)+>
  <!--要素 title のデータ型は文字データ-->
  <!ELEMENT title (#PCDATA)>
  <!--要素 event は空要素（データのない要素）-->
  <!ELEMENT event EMPTY>
  <!--属性 state_name は，属性値を必ず指定-->
  <!ATTLIST state state_name CDATA #REQUIRED>
  <!--属性 event_type は，文字型で属性値を必ず指定-->
  <!ATTLIST event event_type CDATA #REQUIRED>
  <!--属性 action は，文字型で属性値を必ず指定-->
  <!ATTLIST event action CDATA #REQUIRED>
  <!--属性 next_state は，文字型で属性値を必ず指定-->
<!ATTLIST event next_state CDATA #REQUIRED>
]>
```

## ４　ソースコード生成の手法

前章で記述した XML ファイルから，タグの内容，属性を取得し，ソースコードのスケルトンを生成する．図３に，XML ファイルからソースコード生成の流れを示す．

本手法では，XML ファイルからタグの内容，属性の値を取得するために DOM を使用し，これを利用できるプログラム言語（Java，Perl 等）を用いてソースコード生成機構を作成する．

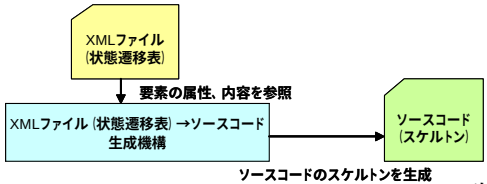


図３　XML ファイルからのソースコード生成

## ５　評価・検証

本研究では，評価・検証のためにマイクロマウスロボットであるラグウォーリア Pro（図４参照）を使用する．

マイクロマウスのライントレースプログラム作成において，状態遷移表からのソースコード生成手法を適用する．これにより，ソースコードを全て手書きで記述した場合と比較して，XUC 図と状態遷移表からソースコードのスケルトンを生成した場合の方がどの程度開発工数やステップ数を減らせたかについて検証する．

図４　マイクロマウス（ラグウォーリア Pro）

## ６　まとめ

XUC 図で組み込みソフトウェアの機能要件を記述し，これを補完するための状態遷移表を XML で記述するための記述規則を提案した．今後は，これを元にソースコードを生成する機構を作成し，評価を行う．

また，評価・検証に使うマイクロマウスは，InteractiveC と呼ばれる C を拡張したプログラム言語を用いる．しかし，本稿ではオブジェクト指向言語でのソースコード生成を前提としている．したがって，C++や Java でのソースコードの生成も行えるようにしたいと考えている．このことに対し，現状では，ソースコード生成にあたって，オブジェクト指向言語のソースコードのスケルトン生成時における生成ソースコードの記述形態が問題となっている．具体的には，一つのオブジェクトに状態遷移表の内容を全てを記述するか，状態遷移表を構成するセル単位のオブジェクトとして実装する State Table パターン [(4)] で記述するかである．この点が今後の課題である．

## 参　考　文　献

(1) Sally Shlaer，Stephen J．Mellor Object Oriented Systems Analysis（オブジェクト指向システム分析 上流 CASE のためのモデル化手法），近代科学社，1995

(2) 渡辺博之，渡辺政彦，堀松和人，渡守武和記: 組み込み UML eUML によるオブジェクト指向組み込みシステム開発，翔泳社，2002.

(3) 細川卓誠，鶴見知生，岡本鉄兵，小泉寿男: 組み込みソフトウェア開発におけるユースケース分析・設計方式の提案，2003.

(4) Douglass，B.：Real-Time UML Developing Efficient Objects for Embedded Systems Addison-Wesley，1997