組み込みシステム開発用 Behavior モデラーの研究
ーユーザーインターフェース操作シーケンス図の合成による Statechart の自動生成ー

北海道大学大学院工学研究科　○漆原映彦、金井理、岸浪建史

要　旨

本研究では，組み込みシステムの GUI 部の制御ソフトウェアの仕様を効率良く，ディジタル形式でモデリングするために，GUI 操作シナリオを表す複数のシーケンス図から，各機能の挙動仕様を表す Statechart を自動生成する手法を提案する．

1．はじめに

PDA，携帯電話の様な組み込みシステム開発の効率化と高品質化が強く望まれているが，その中でも近年では GUI(Graphical User Interface)部の制御ソフトウェア開発のウェイトが増加しており，その開発支援ツールが必要となってきている[1].

GUI 開発プロセスは，製品の機能仕様から挙動仕様への変換プロセスである．その支援ツールの要求機能としては，①製品の機能仕様，挙動仕様のディジタルなモデル化が可能，②それらモデル間の体系的変換が可能である，③複雑な挙動仕様モデルの定義も簡潔に行える，などが挙げられる．すでにいくつかの商用の GUI プロトタイピングツール[2]，[3]があるが，これらは状態遷移図で細かな挙動仕様を直接定義してゆかなければならず，その作業に手間が掛かる．

そこで著者らは，図 1 の様に GUI が果たすべき機能仕様の図的表現として新たに機能関係図を提案し，この機能関係図にさらに GUI 構成要素を関連付けることにより，機能選択を行う GUI 制御部の挙動を表す複雑な Statechart(SC)構造の自動生成手法を提案してきた[4]．本報では，[4]で作成した SC 概略構造の内部に，各機能のより詳細な挙動を表す SC 構造を，機能毎の操作シナリオを表すシーケンス図(SD)集合から自動生成する手法を提案する．この各機能を表す詳細な SC と，[4]で生成した SC の概略構造を組み合わせることによって，GUI コントローラの完全な挙動仕様を表す SC が完成する．

2．シーケンス図集合からの Statechart 自動生成手法

図 2 に提案する SC の自動生成手法の概要を示す．①SD の GUI コントローラオブジェクトに出入りするメッセージを，入力イベントか出力アクションかに分類する．②GUI コントローラオブジェクトのアクションは，GUI 構成要素へのイベントとなるため，それぞれの GUI 構成要素の SC 内の状態変化を引き起こす．これにより GUI コントローラの Statevector が変化する．GUI コントローラのどのアクションで SV 中のどの成分が変化するかの対応づけを用いて，Statevector の唯一の値を GUI コントローラの SC の 1 状態と対応付ける．これによりこの SD に対応した SC の状態集合が定義される．③GUI コントローラのライフラインの最上部から順に，SC の状態間を GUI コントローラに入るメッセージをイベント，GUI コントローラから出るメッセージをアクションとした遷移で結ぶことによってこの SD に対応する SC が完成する．④①～③の過程を全てのシナリオの SD について繰り返すことにより，機能の詳細な操作シナリオに対応した GUI コントローラの SC が完成する．以下にその手法の詳細を述べる．

3．GUI コントローラに関連するメッセージの分類

操作シナリオを表す SD 集合は次の様に表される．$SDS=\{SD_1,SD_2,...,SD_c,...,SD_C\}$（$C$はシナリオの数）．また，各 SD は $SD_c=<O,M,\psi_{from},\psi_{to}>$ となる．ただし，O：オブジェクト集合（$O=\{O_{User},O_{GUI},O_{GUICN}\}$），$M_c$：メッセージ集合，$\psi_{from}$、$\psi_{to}$：$M\rightarrow O$ なるメッセージの発信、及び受信オブジェクトを定める関数である．また，GUI コントローラを表す SC は，$SC=<s_0,S,X,Z,\delta,\omega>$ と表す．ただし，s_0：初期状態，S：状態集合，X：イベント集合，Z：アクション集合，$\delta:X\times S\rightarrow S$：状態遷移関数，$\omega:X\times S\rightarrow Z$：出力関数である．

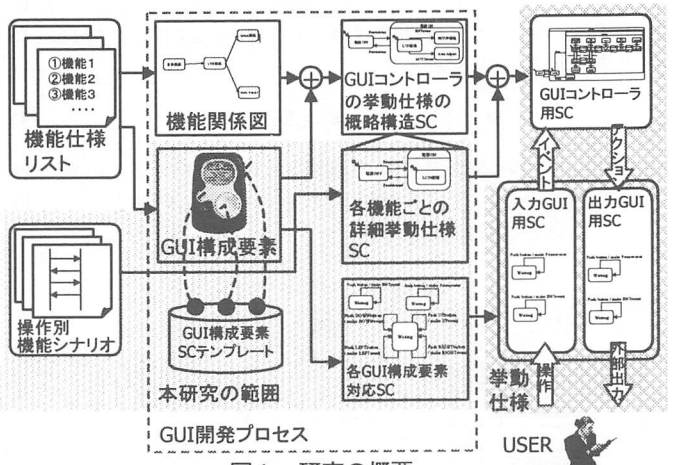

図1　研究の概要

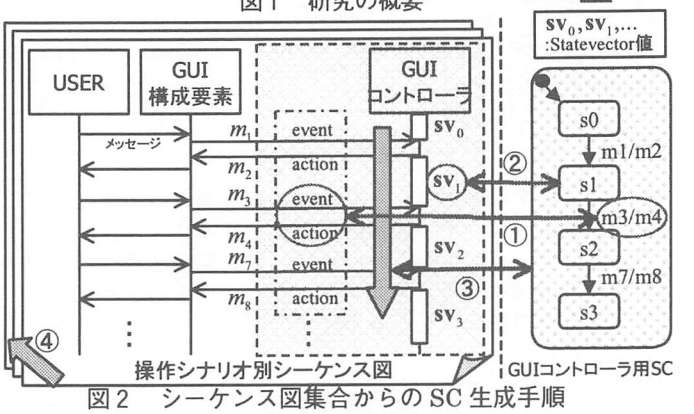

図2　シーケンス図集合からの SC 生成手順

まず，SDS 内の一つの SD_c を選択する．SD_c 内の各メッセージ $m_i(\in M)$ に対して，O_{GUICN} に入る m_i を集合 X の要素とし（$m_i\in X$ (if $\psi_{to}(m_i)=O_{GUICN}$)），O_{GUICN} から出る m_i を集合 Z の要素と分類する（$m_i\in Z$ (if $\psi_{from}(m_i)=O_{GUICN}$)）．

4．StateVector と状態の関連付け

出力用 GUI 構成要素の SC 集合を $GSCS=\{GSC_1,GSC_2,...,GSC_p,....GSC_P\}$ と表す．ただし，Pは出力 GUI 構成要素の数であり，各 GSC_p は $GSC_p=<Gs_{0p},GS_p,GX_p,GZ_p,G\delta_p,G\omega_p>$ と表される．ただし，Gs_{0p}：初期状態，GS_p：状態集合，$GX_p:(GX_p\subseteq M)$：イベント集合，$GZ_p:(GZ_p\subseteq M)$：アクション集合，$G\delta_p:GX_p\times GS_p\rightarrow GS_p$　：状態遷移関数，

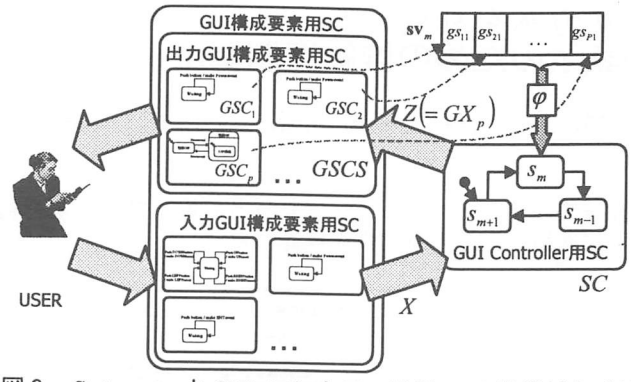

図3　Statevectorと GUI コントローラ用SCの状態対応づけ

$G\omega_p : GX_p \times GS_p \rightarrow GZ_p$ ：出力関数である．

GUI コントローラを表す SC の状態を特定するために Statevector を用いる．Statevector は図3の様に各出力用 GUI 構成要素の持つ SC の状態の組み合わせによって構成されており，そのベクトルの値により GUI コントローラの状態が唯一に特定できる．今，Statevector のとり得る値を $\mathbf{sv}_1, \mathbf{sv}_2, \ldots, \mathbf{sv}_m, \ldots, \mathbf{sv}_M$ とし，$\mathbf{SV} = \{\mathbf{sv}_1, \mathbf{sv}_2, \ldots, \mathbf{sv}_m, \ldots, \mathbf{sv}_M\}$ を Statevector 集合と定義する．ただし，M は状態の数である．ここで各 Statevector 値 $\mathbf{sv}_m$ の各成分は，それぞれある特定の出力 GUI 構成要素の SC の状態を表している．したがって，$\mathbf{sv}_m = (sv_{m1}, sv_{m2}, \ldots, sv_{mp}, \ldots, sv_{mP})$，$sv_{mp} \in GS_p$ となる．

GUI コントローラから発せられたアクション $z(\in Z)$ が、各出力 GUI 構成要素に対するイベント $gx_p(\in GX_p)$ となり、どの出力 GUI 構成要素の SC がどのように状態の変化を起こすかは元々対応付けられている．その対応関係を利用して，GUI コントローラのライフライン上に書くアクションの発生時で区切られた活性区間をそれぞれ定義する（図4）．この活性区間の持つ Statevector の値 $\mathbf{sv}_m$ は，区間開始時のアクション z により唯一に決定できる．同時にこの値 $\mathbf{sv}_m$ と GUI コントローラの状態 $s_m(\in S)$ の間の1対1関係を φ で定義する（$\forall \mathbf{sv}_m \in \mathbf{SV}$，$s_m \in S$，$\varphi(\mathbf{sv}_m) \equiv s_m$）．ただし，各 SD_c の O_{GUICN} の初期状態は，各 GUI 構成要素の初期状態を表す $\mathbf{sv}_0$ に対応する s_0 とする（$s_0 = \varphi(\mathbf{sv}_0)$）．この作業により，この SD 中に記述されている GUI コントローラが持つ状態がすべて定義される．

5. GUI コントローラ用 Statechart の状態遷移生成

4節で定義した状態集合 S に対して，文献[5]で提案されたアルゴリズムを用いて以下の手順で遷移を作成してゆく．

1) $SD_c \leftarrow SD_0$，$s_{Current} \leftarrow s_0$ とする．
2) SD_c の O_{GUICN} のライフライン上で上から順に以下の処理を行う．
   2-1) $s_{Current}$ の状態終了時に GUI コントローラに入力されるイベント $x(\in X)$，及びコントローラから出力されるアクション $z(\in Z)$ を特定する．
   2-2)上記 x, z 発生後の状態 s_{Next} を特定する．
   2-3) $\delta(x, s_{Current}) = s_{Next}$，$\omega(x, s_{Current}) = z$ の定義を SC 内へ追加する．
3) $s_{Current} \leftarrow s_{Next}$ とし，2) 以下を $s_{Next} = NULL$ となるまで繰り返す．
4) $SD_c \leftarrow SD_{c+1}$ として2)以下を繰り返す．

以上により，全ての操作シナリオの SD を集約した GUI コントローラの詳細挙動仕様を表す SC が完成する．

6．本手法の適用例

上述の提案を適用し，操作シナリオを表す SD 集合の簡単な例から GUI の SC を生成した．図5に SD 集合 $SDS = \{SD_1, SD_2\}$ を示す．SD_1 中の GUI コントローラオブジェクトに出入りするメッセージは，

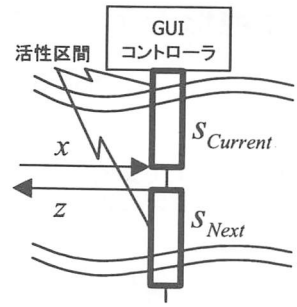

図4　SD から GUI コントローラの遷移定義法

表1　状態－SV 対応表の例

状態	SV	GS_1	GS_2	GS_3
s_0	$\mathbf{sv_0}$	gs_{11}	gs_{21}	gs_{31}
s_1	$\mathbf{sv_1}$	gs_{11}	gs_{21}	gs_{32}
s_2	$\mathbf{sv_2}$	gs_{12}	gs_{21}	gs_{32}
s_3	$\mathbf{sv_3}$	gs_{12}	gs_{22}	gs_{32}
s_4	$\mathbf{sv_4}$	gs_{11}	gs_{22}	gs_{32}

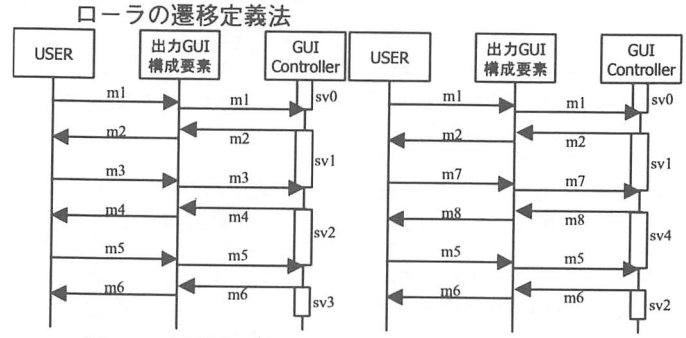

図5　適用例に使用した SD（左 SD_1，右 SD_2）

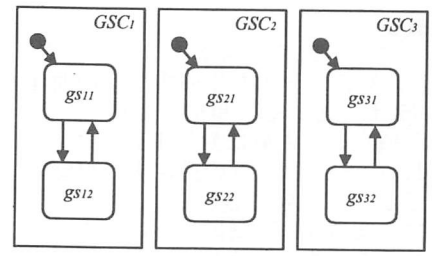

図6　GUI 構成要素の SC 集合

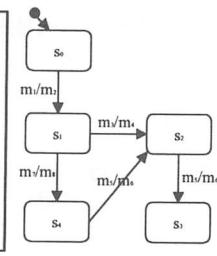

図7　完成した SC

$M_1 = \{m_1, m_2, m_3, m_4, m_5, m_6\}$ であり，それぞれ GUI コントローラの SC のイベントとアクションに分類されると $X = \{m_1, m_3, m_5\}$，$Z = \{m_2, m_4, m_6\}$ となる．

次に SD_c に Statevector を対応付ける．対応づけた対応表は表1の様になる．

GUI コントローラの SC の状態集合 S の各要素間の遷移を SD_1 を用いて定義してゆく．SD_1 中の状態 s_1 でイベント m_3 が入力され，アクション m_4 を出して s_2 に状態変化した時，GUI Controller を表す SC の状態遷移関数 $\delta(m_3, s_1) = s_2$ と出力関数 $\omega(m_3, s_1) = m_4$ が追加定義される．この処理を全ての SD について行い生成した SC は図7の様になる．

7．まとめ

本報告では，機能の操作シナリオを表す SD の集合から，出力 GUI 構成要素の Statechart 集合を利用し，Statevector を定義することによって，GUI コントローラを表す Statechart を自動生成する手法を提案し，具体例で有効性を確認した．今後は前報[4]の内容と組み合わせ，Statechart の自動生成を行うツールの実装を行う予定である．

[参考文献]

[1]小中他；“階層的部品定義に基づく組込用 UI 設計ツール” 情報処理学会研究報告「ソフトウェア工学」No139-002(2002)
[2]Rapid http://www.e-sim.com/ e-SIM Ltd.
[3]プロトビルダー http://www.gaio.co.jp/ GAIO Technology 社
[4]漆原他;”組み込みシステム開発用多機能 Behavior モデラーの研究（第2報）”平成15年度精密工学会秋季全国大会論文集(2003)
[5]Jon Whittle Johann Schumann;”Generating Statechart Designs From Scenarios”,Proc.22nd Int'l Conf .Software Eng.(ICSE'00),pp314-323,(2000)