# 組み込みUIソフトウエアの状態遷移における横断的要素の織り込み

西川泰浩† 小中裕喜†

本論文では，状態遷移図の記述における横断的要素の分離とその織り込み方法について提案する．近年，組み込み機器ソフトウエアの開発において，ユーザインタフェース・GUI 開発の占める比率が大きくなり，その生産性向上が重要な課題となっている．この課題を解決するため，筆者らはこれまで画面の切替えを状態遷移としてとらえるユーザインタフェース設計ツールを開発してきた．しかしながら，状態遷移図中に横断的に含まれる共通要素を括り出して設計しようとすると，かえって状態遷移図の可読性と保守性が損なわれる場合があった．今回提案する方法では，状態遷移図の記述方法にアスペクト指向の概念を導入し，横断的要素を分離して記述することを可能とすることにより，この問題を解決する．

# Weaving Crosscutting Elements into State Transitions of Embedded UI Software

Yasuhiro Nishikawa† and Hiroki Konaka†

This paper proposes a new design method for state charts where crosscutting elements are separated and weaved as needed. As graphical user interface plays a larger role in embedded software, improvement to its productivity becomes a more important issue in the software development. To overcome this issue, we have developed a user interface design tool that treat scene transitions as state transitions. There are some cases, however, where factoring crosscutting elements out of state charts would spoil their readability and maintainability. Our method solves this problem by introducing aspect-oriented concepts into designing state charts to enable the separation and weaving of crosscutting elements.

## 1. まえがき

近年，家電機器などの様々な組み込み機器において，その機能の高度化が著しく，機器の使いやすさへの要求が高まってきた．機器の差別化要因としてユーザインタフェースが重要視されるようになり，組み込み機器にも高度な画面表示を可能とするビットマップディスプレイなどの表示装置が利用されるようになってきている．それにともない，組み込み機器に搭載されるソフトウエアにおける，ユーザインタフェース・GUI 部分の比率が大きくなっており，その生産性向上が組み込み機器開発における重要な課題となっている．

筆者らは，GUI の設計において各表示画面を状態とする状態遷移図を採用するとともに，コード生成を行ってソフトウエアの開発効率化を図るアプローチを提案している[1)]．本論文ではそのような状態遷移図において，アスペクト指向的な概念を導入することにより，1 つの状態遷移図を様々な形に変形させ，その可読性や保守性，再利用性を向上させる方法について述べる．

以下，2 章において，本研究の背景および関連技術について述べる．3 章では，組み込み機器の UI 設計における状態遷移図の利用について述べる．4 章では，状態遷移図における共通要素と設計時の問題について述べ，5 章で横断的要素の分離と織り込みによる問題の解決について説明する．6 章で例題を用いて従来の方法との比較と評価を行う．

## 2. 背景

### 2.1 オブジェクト指向開発

GUI の開発などに用いられるオブジェクト指向開発では，しばしば UML（Unified Modeling Langauge）形式の図を用いたモデリングが行われている．

そのような図の 1 つとして状態遷移図がある．これはシステムの状態がイベントによって遷移していく様子と，そのときに実行されるアクションを表現した図である．状態遷移図はプログラムコードの自動生成と

† 三菱電機先端技術総合研究所
Advanced Technology R&D Center, Mitsubishi Electric Corporation

相性が良い．また，組み込み機器の UI ではディスプレイが小型なため，画面の切替えが多用されるが，状態遷移図はそのような画面遷移を直観的に表現することが可能である．

### 2.2 関心事の分離

一方，プログラムの保守性や再利用性を高めるためには，その設計時に関心事の分離を行い，関連の大きいコードを 1 カ所にまとめて記述するのが有効である．オブジェクト指向も関連するデータと手続きをまとめて，オブジェクトという形で記述することにより，関心事の分離を行っている．

しかし 1 つの視点から関心事の分離を行うと，別の視点での関心事が各所に分散してしまうことが多い．このような問題を解決するために，アスペクト指向が提案されている[2),3)]．これは，1 カ所にまとめることができなくなった横断的な関心事を分離し，アスペクトとしてまとめて記述することを可能とする．UML にこのアスペクト指向を導入しようという試みもすでにいくつか提案されている[5),6)]．たとえば状態遷移図に用いて，遷移の内部に別の状態遷移を導入し，アスペクトとしての機能を追加する方法がある[9)]．

### 2.3 関心事の織り込み

アスペクト指向の目的は，関心事の分離によってその記述を 1 カ所にまとめることである．分離された関心事を既存の処理系と整合させるためには，織り込み（weave）という手段が用いられる．これは分離された関心事を合成して全体のモデルを構築することを意味する．

分離後に編集を加えても整合性をとっていくためには，分離されたそれぞれの関心事を正しく織り込んで，全体のモデルを再構築する方法を規定しておかなければならない．AspectJ [10)] では，どこに（pointcut, join point），どう（advice），何（aspect）を織り込むかを Java の拡張仕様によって記述し，アスペクト指向を実現している．

## 3. 組み込み機器と状態遷移図

### 3.1 組み込み機器の UI 設計

組み込み機器の表示画面はシンプルな構造を持ち，操作者もしくは機器内部から特定のイベントの発生によって，次の画面への遷移や表示部品の表示内容の変更が行われる．そのため，UI 設計には状態遷移図の利用が適している．

一般に組み込み機器は実時間動作を要求される．しかし，人間の反応速度が機器の動作速度よりも遅いため，そのユーザインタフェース部に関しては実時間性を要求されることは少ない．

各画面表示に用いられる表示部品の大半は，テキストや画像が表示できる程度の基本部品か，それらを組み合わせて若干のロジックを追加した，複合表示部品として実装可能な表示部品である．可変要素を表示するための表示部品は必要だが，表示部品自体が実行時の情報に基づいて動的に生成されるような画面は少ない．携帯電話に搭載された Web ブラウザのように，動的な画面生成が必要な場合も Web ブラウザ部分が 1 つの表示部品として供給され，外部での動的な部品生成は必要ないのが一般的である．

### 3.2 状態遷移図

筆者らが開発している組み込み用 UI 設計ツール[1)]は，このような組み込み機器の側面を反映している．画面を状態と見なし，画面遷移を状態遷移とする状態マシン（StateChartObject）を定義していくことにより，UI を設計する．画面内に含まれる表示部品としては，基本部品だけではなく，他の状態マシンを用いることも可能であり，状態マシンの並行状態として 1 つの画面を定義することを可能とする．たとえば，**図 1** において，曲の再生–停止をコントロールする SCO-1 が，SCO-2 の下の画面における並行状態として存在している．また，この SCO-1 は，表示部品として他の画面でも再利用できる．

以降では，このように 1 つの画面を 1 つの状態としてとらえる状態遷移図を使って議論を進める．図の形式としては UML における状態遷移図の形式[7),8)] に準じたものを基本とするが，一部，状態の表現を，画面を模した角の尖った矩形としている．状態は画面であるので画面の構成要素として表示部品を持つ．遷移はトリガとなるイベント名と，遷移を実際に実行するか

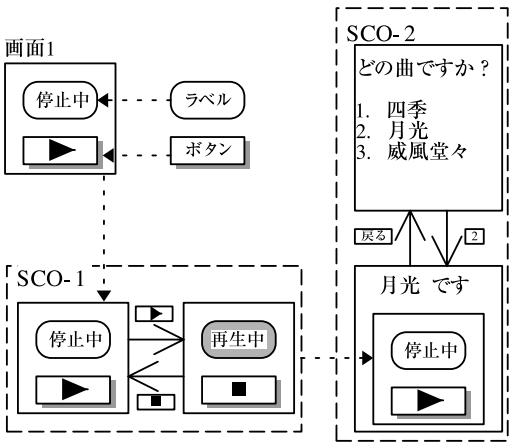


**図 1** StateChartObject における画面遷移定義
Fig. 1 Scene transitions defined in StateChartObject.

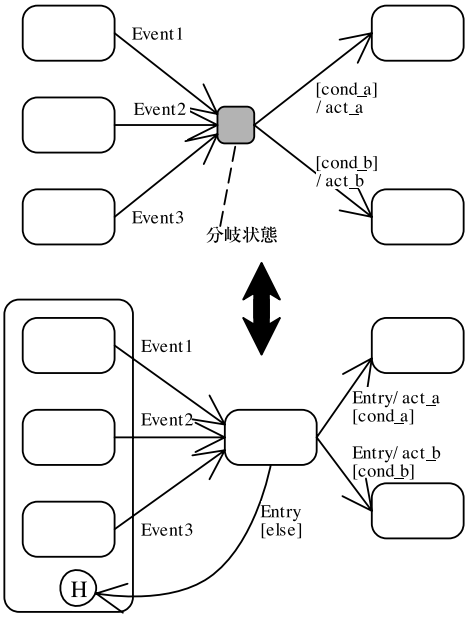


**図 2** 分岐状態
Fig. 2 Branch state.

どうかを判断するガード条件，遷移時の動作であるアクションを属性として持つ．

### 3.3 分岐状態

画面を状態としたとき，画面を持たない状態を表現できなくなることで問題が生じることがある．画面を持たずに合流と分岐を行う一時的な状態がその 1 つである．ここでそのような状態を表現するために，分岐状態と呼ぶ仮想的な状態を便宜上導入しておく．

これは入状時イベントで他の状態に遷移することに特化した状態のことであり，どこにも遷移できなかった場合には遷移元に戻る特徴を持つ．分岐状態の導入によって，複数の遷移を束ねて表現することができる．3 状態から別々のイベントで発生する，共通のガードとアクションを持つ遷移を，分岐状態を使って表現したのが**図 2** 上部である．

なお，この分岐状態は，図 2 下部のように，遷移元の各状態をコンポジット状態としてグルーピングし，戻り先をその履歴状態とすることでも実現可能であるが，この場合には中央の状態が実在する状態となり，状態を画面としてとらえる場合に問題となる．

## 4. 状態遷移図における共通要素

### 4.1 画面遷移設計における共通要素

機器の画面遷移を状態遷移として設計する場合，まず全体を表す 1 つの状態マシンを用意する．全体を表す状態マシンは複数の状態からなり，各状態は表示部品となる他の状態マシンや基本部品の集合を保持する．つまりシステムの定義は各画面ごとに分離され，また表示部品となる部品ごとに分離される．

一方，操作性の統一という観点から，画面間で共通に利用される画面要素も少なくない．同様に画面遷移のトリガとなるイベントと，そのイベントに付随する処理にも共通点が生じる．このような共通要素は，上記のような設計により，様々な状態や状態マシンに横断的に分散することになる．

たとえば，画面背景（壁紙）は多くの画面で表示される．時間表示や，携帯電話画面の最前面に表示される電池残量・電波強度・未読メールなどのアイコンも同様である．

各画面（状態）のイベント処理においても，たとえば携帯電話の終話ボタンなどのように，多数の画面から待ち受け画面への遷移を発生させるというキー操作が，共通要素となる．

携帯電話におけるロック機構なども共通要素の 1 つである．これは特定の機能を利用させないために状態遷移に制限をかけるもので，制限された遷移を発生させようとしてもそれは実行されず，別の特定の画面（警告画面など）への遷移が発生する．同時に複数の機能がロックされることが多く，共通要素となる．

### 4.2 従来の共通要素の表現

画面遷移に関する共通要素があった場合，UML の状態遷移図の枠組みで，どのようにまとめて表現できるかを考える．

#### 4.2.1 共通表示部品の記述

先出の携帯電話のアイコンなどは，電波強度の変化や電池残量などによって，それ自体が画面遷移（状態遷移）を行うが，この遷移は元々の画面遷移とは無関係である．そのため，これらは並行コンポジット状態と考えられる（**図 3**）．この場合の共通要素は，並行コンポジット状態で表現できる．

#### 4.2.2 共通遷移の表現

次に携帯電話の終話ボタンによる「待ち受け画面」への遷移，または電話帳アプリケーションにおける電話帳一覧画面への遷移を考える．これは多数の画面から特定の画面への共通な遷移であり，これは逐次コンポジット状態またはサブ状態マシンを利用して記述することができる（**図 4**）．

なお，ここでは遷移を発生させるイベントについて考えたが，遷移を発生させない内部遷移についても同様となる．

#### 4.2.3 共通ガードの表現

携帯電話のロック機構を考える．ロック機構はロッ

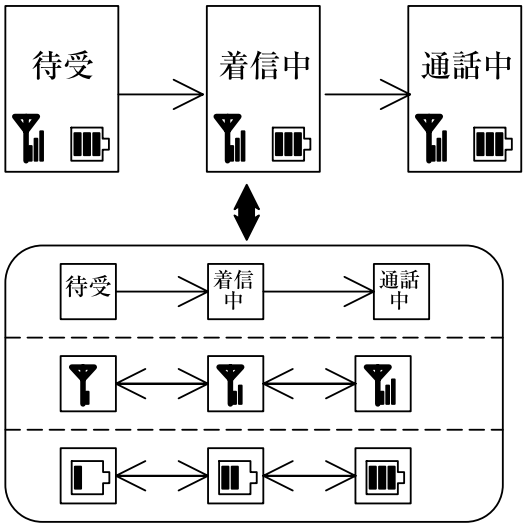


**図 3** 並行的な画面遷移
Fig. 3 Concurrent scene transitions.

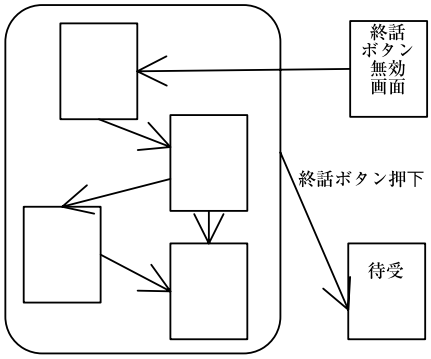


**図 4** 共通な遷移
Fig. 4 Common transitions.

クされているかどうかにより遷移先が変わる状態遷移であり，ガード条件をうまく表現しなければならない．

ロック機構の付いた状態遷移図は，分岐状態を用いると，ガード条件の内部を共通部と非共通部に分離して簡潔に表現できる（**図 5** 上）．そしてこの場合の共通要素は，コンポジット状態とロック中警告画面への共通の遷移で表現（図 5 下）できる．

ガード条件が共通ではなく，アクション部が共通な場合にも，共通なアクションをコンポジット状態の入状時アクションまた退状時アクションとして記述することで，同様な表現によってまとめて記述できる．

このように，遷移の途中に新たな状態を挿入することで，共通部分をまとめて表現することが可能になる．

### 4.3 横断的要素とその問題

ここで横断的要素について考える．特定の視点から共通要素をまとめて括りだしたとき，そのためにうまくまとめて括り出せなくなってしまう共通要素が横断的要素である．このような横断的要素の表現に，これまで述べた方法を適用してみると，うまくいかないことが多い．

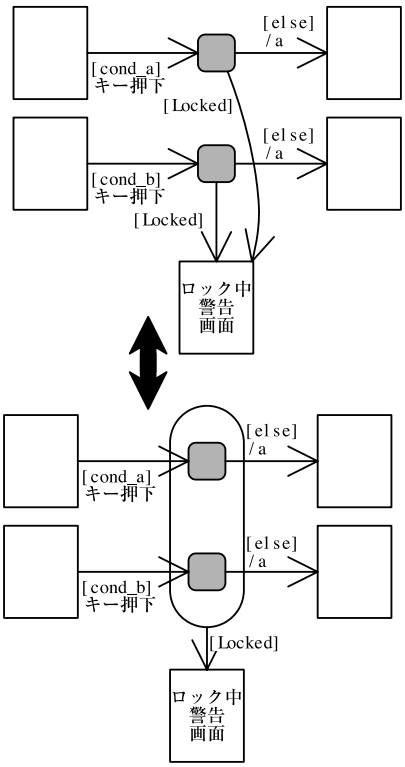


**図 5** 共通ガード条件
Fig. 5 Common guard conditions.

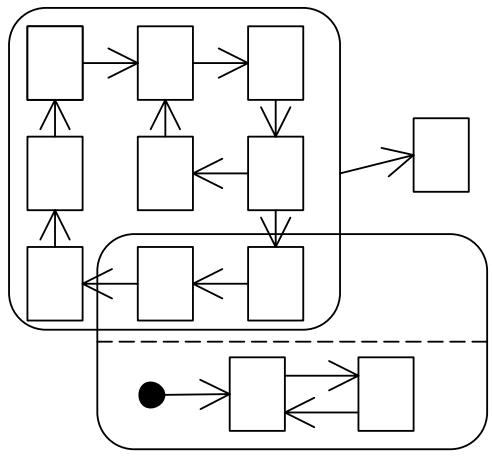

**図 6** 重なり合う状態
Fig. 6 Overlapped composite states.

このような問題は，基本的には **図 6** のようにコンポジット状態がうまく入れ子にならず，一部のみ重なってしまうことによって生じる．これは状態遷移図とし

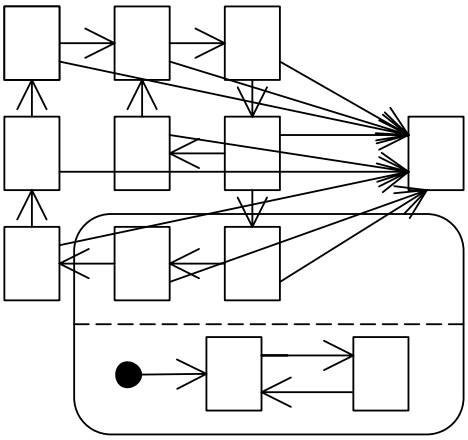

**図 7** 分解して記述した遷移
Fig. 7 Decomposed transitions.

て不正であるので，結局共通な要素は各状態に分散して記述されることになり，**図 7** のように状態遷移図が見にくいものになる．

横断的要素は，状態遷移の設計・記述において副次的な要素であることが多い．たとえば，実際の組み込み機器の状態遷移図では，異常系に関する記述や，ユーザ由来ではないイベント（電話の着信など）に関する記述などがあげられる．しかしながら，このような要素がしばしば状態遷移図を複雑にして，可読性や保守性に大きな影響を与える．

次章では，これをいかに解決するかについて述べる．

## 5. 状態遷移図の変形と織り込み

### 5.1 状態遷移図の変形

横断的要素を分離し，他の部分と整合性をとりつつそれを編集可能とすることによって，状態遷移図の可読性・保守性の低下を回避することができる．

ここで，横断的要素を追加するためには状態遷移図にどのような改変が必要であるかを，次のようにまとめておく．

- どこを改変するか
  - 状態
  - 遷移
- 何を追加するか
  - サブ状態マシン
  - 遷移（内部遷移）
  - 遷移や状態の持つ属性

このような横断的要素の追加にともなう改変のことを，アスペクト指向の用語を用いて「織り込む」と表現することにする．

横断的要素をうまく織り込む機構を用いることで，状態遷移図の見かけ上の複雑さを緩和し，読図や作図を容易にする方法を次に述べる．

### 5.2 織り込み先の指定

状態遷移図中に横断的要素を織り込むためには，それをどこに織り込むかを指定するため，織り込み先を列挙する必要がある．織り込み先は，状態または遷移のいずれかである．

列挙の方式としては，AspectJ などのように検索を用いる方法がある．これは状態や遷移の持つ名前を検索して対象を列挙する方法である．この場合は織り込み先が検索によって正しく列挙できることが重要となるが，状態遷移図の場合は要素の名前が必須でないなど，検索が困難な場合が多い．そのため，以下では織り込み先であることを示すマーカを要素に付属させることとする．

### 5.3 織り込みの方法

織り込む要素によって，その織り込み先と織り込み方法に違いがある．

#### 5.3.1 サブ状態マシンの織り込み

4.2.1 項にあるように，共通表示部品は並行コンポジット状態で表現される．表示部品は一般的には状態マシンで表現されるため，その織り込みは，対象となる状態に並行コンポジット状態を作成してその並行状態としてサブ状態マシンを追加することになる．単なる状態遷移図としては優先順位は考えなくてよいが，背景など表示部品としての織り込みを考えると，優先順位に相当する Z 順（前面・背面）の指定を必要とする．

そのままコンポジット状態を用いると，複数のコンポジット状態が入れ子にならずに重なってしまうことがある．このような重なりを解消するためには，コンポジット状態の占める範囲を変更し，イベントを駆使した複雑な状態遷移図とするなどの対処が必要である．そのような本質と無関係な複雑さを，織り込みによる表現で排除できる．

サブマシンの織り込みの例を，**図 8** にあげる．図中，丸の付いた文字 C を織り込み先のマーカとして表現している．図 8 a のマーカの付いた状態に図 8 b の横断的要素を織り込んだ図が図 8 c となる．

#### 5.3.2 遷移の織り込み

4.2.2 項にあるような形で，横断的要素として遷移を織り込む場合がある．この場合，織り込まれる対象は遷移両端の状態である．遷移元，遷移先の双方を織り込み先として指定する．

遷移（内部遷移）の織り込みに関しては重複の問題がある．織り込みによって同じ条件で動作する複数の

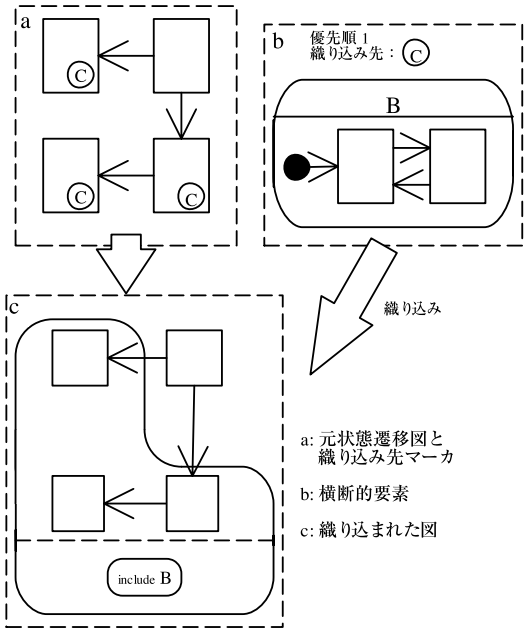


**図 8** サブ状態マシンの織り込み
Fig. 8 Weaving a sub-statemachine.

遷移が現れた場合，その実行順序，実行可否に関して調停が必要である．調停方法としては，織り込む横断的要素と織り込まれる状態遷移図に優先順位付けをし，優先順が自分以下の遷移を評価するかどうかの真偽値を持たせる方法が考えられる．

遷移の織り込みの例を，**図 9** にあげる．図中，丸の付いた S が遷移元，D が遷移先の状態であることを示しており，この情報に基づいて横断的要素である遷移（図 9 b）を織り込んだのが図 9 c である．優先順序の定義として，Z 順と同じく数値を用いている．

### 5.3.3 遷移内への状態の織り込み

4.2.3 項でガード条件を例にとった表現例をあげたが，このように遷移の途中に状態を織り込むことで表現できる横断的要素がある．

状態を遷移の中に織り込む場合，その状態の織り込み先は

- イベント発生後：ガード評価前
- ガード評価後：アクション評価前
- アクション評価後：次状態へ入状前

の区別がある．

これは，新しい状態への遷移，ガード条件の評価，アクションの実行がどのような順序で行われるかによって区別される．ガード条件の評価とアクションの実行では，必ずガード条件が先となるため，状態の織り込み先は上記の 3 カ所となる．

複数の横断的要素を同じ遷移に組み込む場合には，

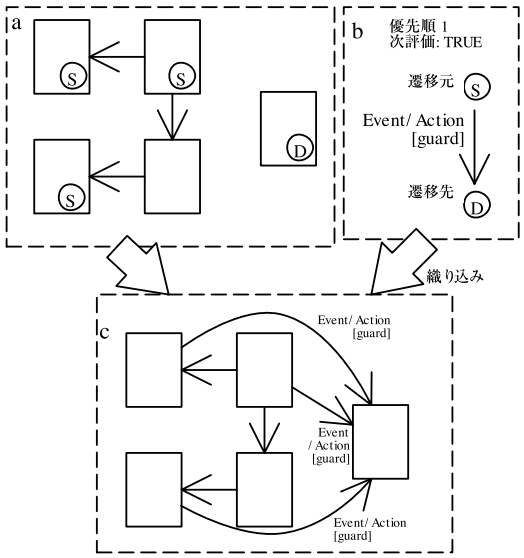


**図 9** 遷移の織り込み
Fig. 9 Weaving transition.

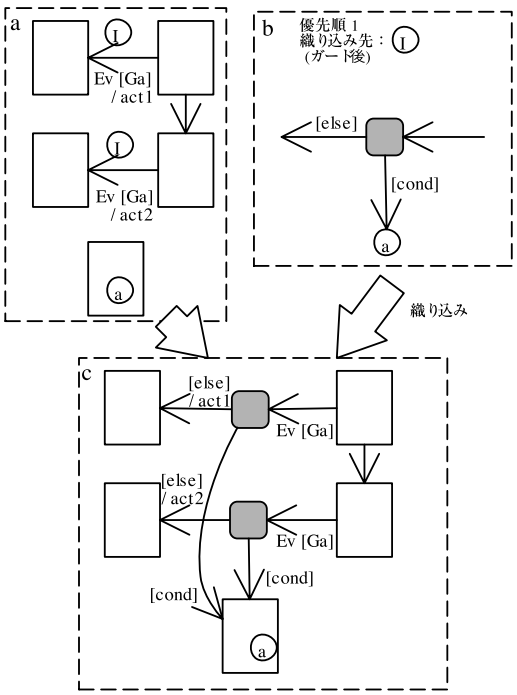


**図 10** 遷移への状態の織り込み
Fig. 10 Weaving states into transitions.

織り込み順の設定などによる調停が必要である．

**図 10** に，分岐状態とコンポジット状態を遷移の中に織り込み，さらに織り込んだ状態から出る遷移を織り込む例をあげる．

### 5.3.4 属性の織り込み

遷移のガード条件など，要素の属性を織り込む場合

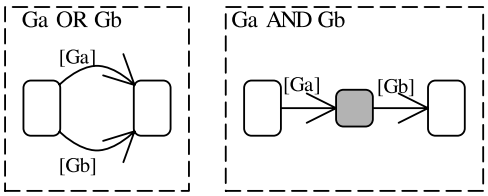


**図 11** ガードの織り込み
Fig. 11 Weaving a guard.

がある．この場合は，これまでに紹介した織り込みを利用できる．

遷移のガード条件は真偽値であるので，AND や OR などの論理演算または代入によって合成ができるが，この演算結果は演算の順序に依存する．このようなガードの織り込みは，並行する遷移の織り込み（OR 演算）や，分岐状態の織り込み（AND 演算）によって実現できる．遷移のアクションも同様である．

織り込み先の元々の属性が空であった場合は，状態を織り込まずに，織り込むべき横断的な属性を代入（上書き）してしまえばよい．

ガード条件（Ga および Gb）の OR 演算および AND 演算による織り込みの例を，**図 11** にあげる．

### 5.4 織り込みの表現

5.3 節などの例では，織り込み先の列挙に丸の付いたアルファベットのマーカを用いた．これはシステムの状態遷移図から横断的要素を省略し，簡潔に表現するための 1 つの方法である．

このマーカによって表現すべきは，織り込み先であること，そして，どの横断的要素が織り込まれるか，の 2 点である．しかしこの組合せによってマーカを決定すると種類が多くなり，かえって可読性を損なう．また，すべての織り込み先について，マーカを表示する必要もないため，特に興味のある横断的要素についてのみマーカを表示し，興味のない横断的要素についてのマーカを除去するようにしてもよい．これにより，状態遷移図の興味のある側面だけを表現した図を構成することができる．

また，必要に応じて織り込みを使い，状態遷移図の全体を確認することもできる．

## 6. 適 用 例

仮想的な組み込み機器を例示して，従来手法と今回提案する手法とを比較する．例題として，メモリカード上の画像を閲覧したり音楽を試聴したりする携帯機器をあげる．

この装置では，待ち受け画面からメニューを経てファイル一覧画面に到達し，そこから画像閲覧や音楽試聴を行う．いくつかの画面には，30 秒ほど放置すると待ち受けに戻る機能や，画面上部の定位置に時計と電池残量表示機能がある．またメモリカードを要求する画面でメモリカードを抜くとその旨を画面で表示する．

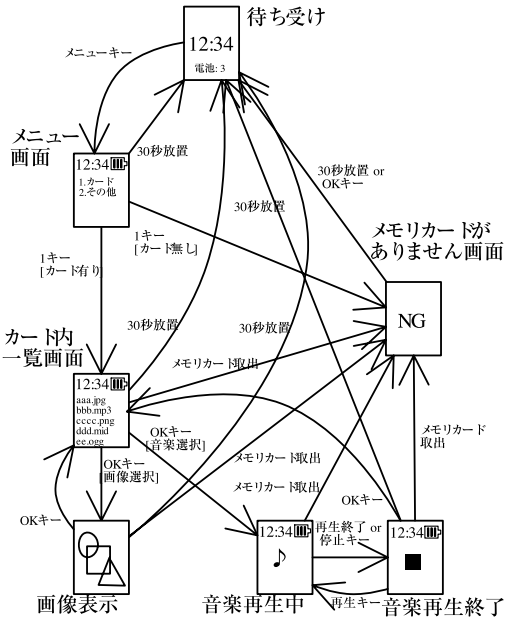


**図 12** 従来の状態遷移図
Fig. 12 Conventional state chart.

### 6.1 従 来 手 法

従来の状態遷移図で表現すると，**図 12** のようになる．この図では 4.3 節で述べたような以下の問題が発生している．

- 同じ定義が頻出し，保守性が低下．
- 煩雑な記述のため，可読性が低下．
- 入れ子が重なるため，記述をまとめられない．

たとえばメモリカード取り出しや 30 秒放置に関する遷移が図を複雑にして可読性を下げている．また，これらの記述や電池残量表示に関する記述が複数の状態に分散しており，保守性を損なっている．しかしながらコンポジット状態としてまとめるのは，入れ子が重なるため不可能である．

### 6.2 提 案 手 法

今回提案した方法を用いた例を，**図 13** に示す．図上部がベースとなる状態遷移図であり，下の破線で囲まれた 3 つの要素が織り込まれる．たとえば，状態遷移図中，白抜きの丸印に X のマーカが付いている状態からは，メモリカード取り出しイベントによって，メモリカードがないことを示す画面への遷移が発生することが示されている．

この図 13 は 7 画面程度の小さな例ではあるが，本

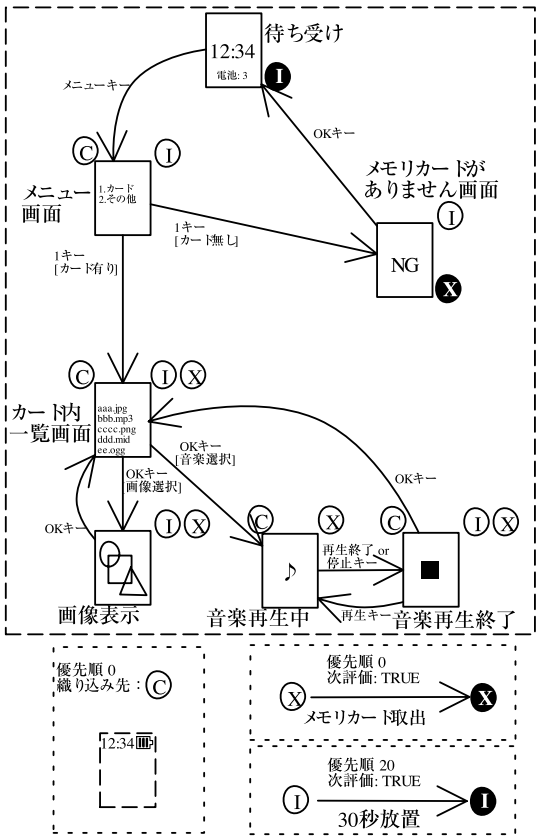


**図 13** 提案手法による状態遷移図
Fig. 13 Proposed state chart.

手法を適用することにより，以下の効果が見られる．

- 横断的要素のマーカ表示による図の簡略化（可読性の向上）
- 横断的要素の定義の局所化（保守性・再利用性の向上）

たとえば，メモリカード取り出しイベントに関する定義 5 つ，30 秒放置に関する定義が 3 つ，時計と電池残量表示に関する定義 4 つが，それぞれ 1 つにまとめられている．また，遷移だけに着目すると図 12 では，遷移の総数が 18 本あり，そのうち 6 本がメモリカードなしを示す画面への遷移であったが，図 13 では遷移の総数 10，同画面への遷移は 1 となった．

このように，今回提案した表現手法を用いることで，GUI の状態遷移図の可読性・保守性を向上させることができる．

## 7. ま と め

組み込み機器のユーザインタフェース開発において，画面状態の遷移を状態遷移図でモデル化する際に，横断的要素により状態遷移図が複雑化する問題を述べ，アスペクト指向の導入によって問題の解決を試みた．まとめると，以下のようになる．

- 画面を状態として表現する状態遷移図を用いる．
- 状態遷移図中の横断的要素を分離して表現することにより，可読性と保守性を向上させる．
- 興味のない横断的要素を隠蔽することで，本質的なシステムを眺めることを可能とする．

今回提案した設計手法は，筆者らの開発する UI 設計ツール[1] に組み込み，展開する予定である．特にコード生成に関しては，織り込みによるコード量の増加や実行速度への影響を考慮しつつ，効率的な実現をめざす．

## 参 考 文 献


1) 小中ほか：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「ソフトウエア工学」，No.139-002, pp.7–8 (2002).
2) Kiczales, G., et al.: Aspect-Oriented Programming, *Proc. European Conference on Object-Oriented Programming*, pp.220–242 (1997).
3) Aspect-Oriented Software Development Community & Conference. http://aosd.net/
4) 鵜林尚靖：組み込みソフトウェアの設計モデリング技術，情報処理，Vol.45, No.7, pp.682–689 (2004).
5) Aldawud, O., et al.: A UML Profile for Aspect Oriented Modeling, Workshop on AOP, OOPSLA (2001). http://www.cs.ubc.ca/~kdvolder/Workshops/OOPSLA2001/submissions/26-aldawud.pdf
6) Suzuki, J., et al.: Extending UML with Aspects: Aspect Support in the Design Phase, 3rd AOP Workshop at ECOOP'99 (1999). http://www.yy.ics.keio.ac.jp/~suzuki/project/pub/ecoop99.pdf.zip
7) Object Management Group: UML 仕様書，ASCII (2001).
8) ランボー，J. ほか：UML リファレンスマニュアル，ピアソン・エデュケーション (2002).
9) Prehofer, C.: Feature Interactions in Statechart Diagrams or Graphical Composition of Components, *Workshop on Aspect-Oriented Modeling with UML* (2002). http://lglwww.epfl.ch/workshops/uml2002/papers/prehofer.pdf
10) AspectJ. http://www.eclipse.org/aspectj/


(平成 16 年 10 月 12 日受付)
(平成 17 年 4 月 1 日採録)

**西川 泰浩**

1995 年東京大学大学院工学系研究科精密機械工学専攻修士課程修了．同年三菱電機株式会社入社．現在は三菱電機株式会社先端技術総合研究所に所属する．ソフトウエアの開発効率化技術に関する研究開発に従事．電気学会会員．

**小中 裕喜**（正会員）

1989 年東京大学大学院工学系研究科電気工学専攻修士課程修了．同年三菱電機株式会社入社．1992 年より技術研究組合新情報処理開発機構に出向．1996 年三菱電機先端技術総合研究所に帰任．工学博士．並列プログラミング言語，情報検索，組み込みシステム，UI 設計ツール等の研究に従事．