現象/行動モデルに基づく対話的プログラミング

3J－6　中村博之　水野貴之　大貫秀明　大塚玲　楠木規央　清野龍介　篠原健
野村総合研究所　IT研究センター

1　はじめに

ソフトウェア開発がカスタムメイド方式からビルディングブロック方式にパラダイム変換を遂げつつあるが，この流れは究極的にはエンドユーザをソフトウェア構築の主体に押し上げるところに行き着く．

エンドユーザプログラミングを支援するツールとして，入力履歴からプログラムを生成する AppleScript などがある．しかし，これらは入力履歴だけからは条件の抽出ができないなどの理由で，限定された機能しか記述できない．

他方，ユーザの入力例から制約や制御構造を一般化してとり出す，実演によるプログラミング [1] と総称される研究がなされているが，一般化が困難な場合が多く実用化に至っていない．

こうした背景のもと，本研究ではエンドユーザがシステム内部の知識を持つことなく，制御構造を持った複雑な機能の追加を行なうことを目的とし，現象/行動モデルに基づく対話的プログラミングを提案する．

2　現象/行動モデルと実行方式

エンドユーザプログラミングの対象システムとして電子メールツールを例にとると，図1に示すように，メールが着いた，メールを読むといったユーザが意識する意味的な内容は，ユーザとシステムの間では，アイコンが点灯した，ボタンを押した，という種類のインタラクションによって内部動作と関連付けられている．

本手法ではこの種のインタラクションに注目し，システムからの出力側を**現象**，入力側を**行動**と呼ぶ．だたし正確には図2のように定義する．

そして，ユーザはある現象 α に注目していることを宣言した後，その現象のもとで行動 a を起こす．

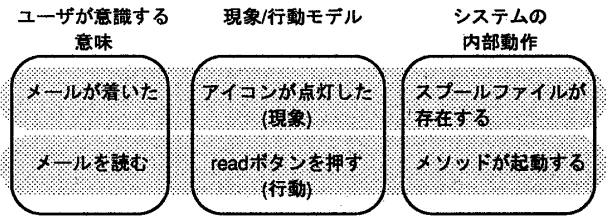

図 1: 現象/行動モデル

現象 $\alpha, \beta, \gamma \ldots$
GUI オブジェクトに関しユーザが観測可能な状態．

環境 $\mathcal{E}^i : \{\alpha, \beta, \ldots\} \mapsto \{\mathrm{T}, \mathrm{F}, \mathrm{null}\}$
時刻 i において現象が起こっているかどうか (その現象に関する変化がないとき null，変化がありかつ現象が起こっているとき T，変化がありかつ現象が起こっていないとき F) を決定する写像．

行動 a, b, c, ...
オブジェクトに対するユーザの行為のうち，システム内でインタラクションが起こるもの．

ルール $\alpha \to a,\ \beta \to bc \ldots$
ある現象のもとで行動 (行動の列) を実行すること．

反応 $a \rightsquigarrow \beta, \ldots$
ある行動により新たな現象が引き起こされること．行動 a により現象 β が T となることを $a \rightsquigarrow \beta$，F となることを $a \rightsquigarrow \neg\beta$ と書く．

図 2: 定義

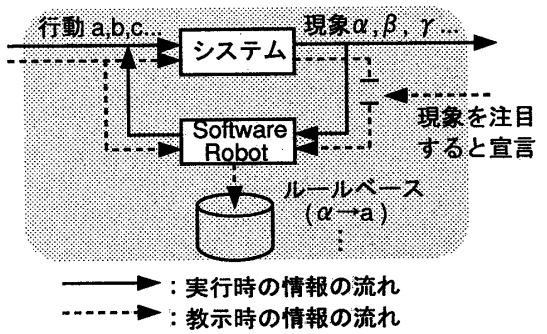

図 3: 教示と実行

この結果，**ルール** $\alpha \to$ a がルールベースに蓄えられ，以後現象 α が起こると自動的に行動 a が実行される (図3)．この際，ルールの入力や実行を管理する部分を開発するが，これを SoftwareRobot と呼ぶ

ある時点 i で現象が起こっているかは**環境** $\mathcal{E}^i$ により定まる．行動 a の結果として現象 β が起こることを**反応** ($a \rightsquigarrow \beta$) と書くが，反応は環境に依存する．

An Interactive Programming Method Based on the Phenomenon-Action Model : Hiroyuki Nakamura, Takayuki Mizuno, Hideaki Ohnuki, Akira Otsuka, Norio Kusuki, Ryusuke Kiyono, Takeshi Shinohara
Nomura Research Institute, Ltd.

また, 反応により起こった現象にもルールは適用される. 例えば, ルール $\alpha \rightarrow \mathrm{a}$ が存在して, 環境に応じて $\mathrm{a} \rightsquigarrow \alpha$ または $\mathrm{a} \rightsquigarrow \neg\alpha$ となる場合, SoftwareRobot は $\mathrm{a} \rightsquigarrow \neg\alpha$ となるまで a を実行し, ループとなる.

このように本手法では, 現象と行動の入力により条件節やループなどの制御構造を記述できる.

図 4に実行方式を示す. ただし, ここではある環境 $\mathcal{E}^i$ のもとでルールベース $\mathbf{R}$ から発火可能な行動 ($\mathcal{E}^i(\alpha) = \mathrm{T}$, $\alpha \rightarrow \mathrm{a}$ のときの a) の集合を取り出す操作を $\mathcal{F} : \mathbf{R} \times \mathcal{E}^i \mapsto \{\mathrm{a}, \mathrm{b}, \ldots\}$ としている.

i 時刻 i で発火可能な行動の集合 $\mathbf{A}$ を $\mathbf{A} \leftarrow \mathcal{F}(\mathbf{R}, \mathcal{E}^i)$ により得る.

ii ある行動 $\mathrm{a} \in \mathbf{A}$ を実行し, $\mathbf{A}$ から a を取り除く.

iii 環境が $\mathcal{E}^i$ から $\mathcal{E}^{i+1}$ に変化する.

iv $\mathbf{A}' \leftarrow \mathcal{F}(\mathbf{R}, \mathcal{E}^{i+1})$ により新たな発火可能な行動の集合 $\mathbf{A}'$ を得る.

v $\mathbf{A} \leftarrow \mathbf{A} \cup \mathbf{A}'$ により発火可能な現象の集合をマージし, iに戻る.

図 4: 実行方式

3 非決定性の解消

図 4の実行方式では ii で行動を実行させる際, その順序によって終了時の現象が非決定的になることがある[1]. また, ユーザが同じ現象に対し相矛盾する行動を指示した場合には, 必然的に非決定性が生ずる. この非決定性を検出するには,SoftwareRobot が反応を予測できることが必要であるが, 反応は環境という SoftwareRobot にとって未知の写像によって決まるため, 一般には非決定性の検出すらできない.

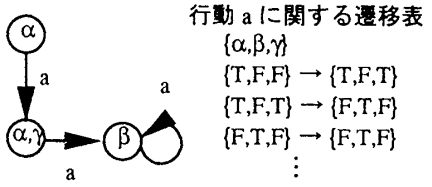

図 5: 現象の遷移表

そこで, 図 5に示すように, ある行動 a についてシステム内の全現象に関する遷移表を用意し, システムと SoftwareRobot を結合する際にあらかじめ遷移表を埋めておく. この遷移表が完全であれば, SoftwareRobot は非決定性を検証することができる.

[1]例えば $\alpha \rightarrow \mathrm{a} \rightsquigarrow \neg\alpha \wedge \beta$, $\alpha \rightarrow \mathrm{b} \rightsquigarrow \neg\alpha \wedge \neg\beta$ で α が T のとき, 発火可能な行動の集合は $\{\mathrm{a}, \mathrm{b}\}$ であるが, 終了時に β か $\neg\beta$ かは実行順序で異なる.

遷移表が完全に用意されている場合, 教示の際にルールが新たに獲得されると, 遷移表を用いて既存のルールとの間に非決定的に実行する場合があるかを検証する. 非決定性が検出された場合は, 原因となるルールを取り出し, ユーザに対してルールの選択を促す. こうして決定的なルールベースを維持できる.

一方遷移表が不完全な場合は, 教示時や実行時に遷移表中にない反応が検出された時点で, その前後の現象の遷移を用いて遷移表を埋める. 遷移表が更新されると, その結果として非決定性が検出されることがあるが, その際は上記と同様にユーザにルールの選択を促す. こうして遷移表が埋まるに従い, より強力に非決定性を検出できるようになる.

4 実現および応用例

システムに SoftwareRobot を結合させる際には, 扱う現象, 行動を指定する作業をシステムごとに行なう. またシステム側には, マウス位置からオブジェクトを返す, 行動を外部から仮想的に実行するなどの API があれば十分である.

本手法の簡単な応用例として, メール/ニュースツールと結合させて, メールの到着に応じて内容中にキーワードがあればメールを自動転送するなどの機能をユーザが自由に追加できる, 柔軟なコミュニケーションツールを作ることができる.

またこうして使い込まれカスタマイズされたツールは, 特定エンドユーザの知的作業の凝縮された成果を広く再配布し共有できるという効果を合わせ持つ.

5 おわりに

現在 SoftwareRobot のプロトタイプを構築中である. 今後は現象と行動を体系化して SoftwareRobot とシステムの容易な結合方式, 非決定性解消手法の妥当性の検証, 遷移表の動的な構成法などの検討を進める.

参考文献

[1] A.Cypher: *Watch What I Do : Programming by Demonstration*, The MIT Press, 1993.