F-023

ビジュアルブロックによる役割指向シナリオテンプレートエディタの設計と開発
Design and Development of a Block-based Editor for Role-Oriented Scenario Template

栗原 あずさ† 佐々木 晃‡
Azusa Kurihara Akira Sasaki

1. はじめに

シミュレーションシステムを汎用言語で表す際，シミュレーションエンジン部とモデル部の記述を分離することが望ましい．先行研究ではこれをスケルトンとシナリオの二つの記述に分け，役割指向テンプレートジェネレータで合成することで，汎用言語で記述されたエージェントベースシミュレーション(ABS)を生成する手法を提案した．しかし，シナリオ記述はジェネレータ言語と汎用言語が混在するため可読性が低いなどの問題があった．本稿ではシナリオ開発用フロントエンドとなるエディタについて述べる．ジェネレータ言語の構文要素をビジュアルブロックに置き換え，その組み合わせでシナリオの構造を表現することで，ユーザはジェネレータ言語の文法を意識せずにシナリオの記述が可能になる．

2. エディタの概要

役割指向テンプレートジェネレータは SOARS 社会シミュレーションシステム[1]において特徴的な「役割」構造に基づいた ABS を実現する枠組みを与える[2][3]．先行研究ではテンプレートジェネレータ言語を用いて役割指向 ABS の記述をスケルトン記述(ベース記述)とシナリオに分割した．スケルトンはシミュレーションのエンジン部であり，シナリオはそれぞれの ABS 固有のモデル部である．モデルの共通部分をエンジン部に切り出すことで記述量を減らしシナリオの開発に集中することを可能にした．

スケルトン記述として Java，Scheme の二つの汎用言語のシミュレーションエンジンが用意されている．これらを元に言語によらない ABS の実装を可能にする．一方でテンプレートジェネレータにおける構造を表す言語と汎用言語が混在するため記述が難しく可読性が低い．テンプレートジェネレータ言語自体も汎用的な役割構造を扱うよう考案されたため可読性が低いという問題がある．

本稿ではビジュアルブロックを用いたシナリオテンプレートエディタの実装について述べる．提案手法ではテンプレートジェネレータ言語部分をビジュアルブロックで表現しその組み合わせでシナリオの構造を表現する．このため役割等のシナリオの構造がブロックで可視化されユーザ入力を役割等の定義および独立したテキストエリアへの汎用言語による記述に制限できる．本ツールは変換ルールに従って構文ブロックの組み合わせで表現したシナリオをシナリオファイルに変換する．生成したシナリオファイルとスケルトンをテンプレートジェネレータで合成することで，最終的な汎用言語のコードを得る．

またビジュアルブロックとその変換規則を新たに定義することでブロックの拡張が容易に行える．ブロックセットを拡張することで，ユーザ入力やブロック数を減らし，視覚的にわかりやすいシナリオの記述が可能になる．

† 法政大学大学院情報科学研究科
‡ 法政大学情報科学部

3. エディタの設計

本稿で述べるエディタは，テンプレートジェネレータ言語によるシナリオ記述の編集を行うビジュアルエディタである．Scratch[4]等で採用されている，ブロックを組み合わせるタイプのビジュアルプログラミング言語である．シナリオエディタはブロックメニューとワークスペースから成る．ユーザはブロックメニューからワークスペースにブロックをドラッグ&ドロップできる．ワークスペース上でブロックを組み合わせ・切り離すことでシナリオの構造を作る．本稿では特に Java プログラムとして合成されるシナリオを例にとって説明する．

テンプレートジェネレータ言語には表 1 で示す文法要素が存在する．それぞれのブロックはこの言語の構文要素に対応させたものである．ブロックからシナリオへの変換の例を表 2 に示す．下線部はテンプレートジェネレータ言語であり，斜線部分はユーザによる入力に置き換えられる．これらのルールを用いてブロックからシナリオへの変換を行う例を図 1 a，b に示した．

表 1．ジェネレータ言語の文法要素と意味（一部）

メタ文字	意味
@	エージェント
なし	役割
/	ステージ
タブ	出力コード
:	役割の継承
$	ステージ内容の文字列置換

表 2．ブロックとジェネレータ言語の変換ルールの例

ブロックのラベル (b)	シナリオの要素 T[b]
defineAgent *(agentName)*	**@***(agentName)*
defineRole *(roleName)*	*(roleName)*
defineStage *(stageName)*	**/***(stageName)*
setRole *(roleName)*	**:\t***(roleName)*
(code)	**{***(code)***}** *注 行頭にタブ文字が挿入される
defineVar *(name) (value)*	**$***(name)***\t***(value)*

本エディタは新たなブロックと変換ルールを追加することで容易に新しい構文要素を登録できる．例えば図 1 c の createAgent ブロックは createAgent(*name*)number(*n*) とラベル付けされるが，その変換結果は以下である．ここで+は改行を含む連結である．

T[defineAgent(*name*)] +T[setRole(“Agents”)]
+ T[defineVar (“number”)(*n*)]

すなわち図 1 a はエージェント集合 Student とその個数，初期化のコードを表すブロックであるが，Agents，number はスケルトン記述で定義されたエージェントの初期化用のシステム変数である．これをユーザに入力させるのは不適切であるため，これらを変換ルール内に内包させた新しいブロックを定義した．このような既存の構文を組み合わせた新しい表現を追加する方法で，ジェネレータ

言語を意識しないプログラミングを可能とする構文を提供している．

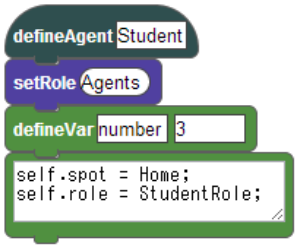

図１ａ．ブロックで表現されたシナリオ

```
@Student
:  Agents
$number 3
{
   self.spot = Home;
   self.role = StudentRole;
}
```

図１ｂ．図１ａのブロックから生成されたシナリオ

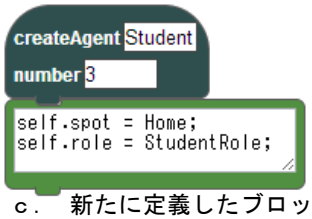

図１ｃ．　新たに定義したブロック

4. エディタの実装

本エディタは html，Javascript，CSS で実装されている．ブロックの形状は図２，内部構造を図３で示す．ブロックの設計は Waterbear[5]を参考にしたものである．

図２．ブロックの形状

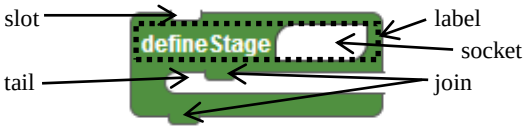

図３．ブロックの構造

ワークスペース上のブロックは木構造で表される．以下の手順で再帰的にシナリオへ変換する．

1） ワークスペースをルートとし，その直下のブロックから変換を開始する
2） 選択されたブロックをシナリオ要素へ変換する
3） そのブロックが tail を持ち，そこにブロックが接続されていた場合，tail のブロック以下を変換する
4） そのブロックが下部に join を持ち，ブロックが接続されていた場合，下部のブロック以下を変換する
5） どの join にもブロックが接続されていない場合，変換を終えて親ブロックに戻る
6） ルートノードまで戻り，１以下を繰り返す

5. 評価と考察

本システムを用いて記述したシナリオからシナリオファイルを生成しテンプレートジェネレータでスケルトンと合成することで実行可能な役割指向 ABS を構築できることを確認した．テキストによるシナリオ記述は役割指向による ABS を単純に記述できるが可読性が低くテンプレート作者でなければ記述が難しい場面があった．本システムの表現ではブロックの色分けとラベリングにより可読性が高く，また３節で述べたように意味的にまとめるべき構文要素をブロックセットに追加することでジェネレータ言語を意識しないプログラミングを可能にした．

本エディタはブロックをユーザ側で生成する機能を持つ．ユーザはブロックの形状，ラベルを定義することで新たなブロックを生成できる．本実装ではユーザはシナリオ記述への変換ルールを自由に登録できないが，変換処理のコードを挿入することで比較的容易に新たな意味を持つブロックを生成できる．今後，ユーザ側でブロック定義を可能にすることでエディタの拡張が容易になり，ビジュアルブロックプログラミング環境を効果的に実現できる．一方，本手法では構成されたブロックに対する意味制約を定義する手法はまだ定式化できていない．またブロック内部に汎用言語によるスニペットを記述させる手法についても検討の余地を残している．

6. おわりに

本稿では役割指向 ABM のプログラムを生成するテンプレートジェネレータのためのシナリオエディタの開発について述べた．本エディタはシナリオ記述にブロックを用いて構造を可視化することでテンプレートジェネレータ言語を意識しないプログラミングを可能にした．また，ブロックセットの拡張性は様々なプログラムの表現の導入のみならず異なるスケルトンに対しても対応可能であることを示唆している．今後は変換ルールの自動登録，ブロックのシークエンスを一つのブロックに置き換えるマクロ機能，意味チェックの機能の実装を行い手法の実用を目指す．本エディタは ABM の実装言語によらない．ブロックの変換，新たなブロックの定義は，原理的には役割指向 ABM に限らず様々なプログラミング環境に応用が可能であると考えられ，今後の研究課題である．

謝辞

本研究の一部は，科研費(課題番号 23700043, 23500034)の助成を受けた．

参考文献

[1] 田沼英樹，出口弘，エージェントベース社会シミュレーション言語 SOARS の開発，信学会論文誌，Vol.J90-D, No.9, pp.2415-2422 (2007)
[2] 田沼英樹，役割指向テンプレートジェネレータによる従来言語ルール記述と役割指向 ABM の結合，: 合同エージェントワークショップ＆シンポジウム 2011(JAWS2011).
[3] 佐々木晃，柏木孝仁，田沼英樹，役割指向テンプレートジェネレータを利用したエージェントシステムの効果的な設計と実装: 合同エージェントワークショップ＆シンポジウム 2012(JAWS2012)
[4] M. Resnick et al. Scratch: Programming for All, Communications of the ACM, Vol. 52 No. 11, Pages 60-67
[5] http://waterbearlang.com/　Waterbear