Designers' Guide to Social Simulations, No.4 モデル作成チュートリアルガイド 2nd Edition

Part | 社会を理解するためのシミュレーション

- Chap.1 モデル化とシミュレーション
- Chap.2 社会を捉えるための枠組み
- Chap.3 モデル作成のプロセス
- Chap.4 モデル作成環境
- Part II シミュレーションデザイン演習【基本編】
 - Chap.5 モデル作成の準備
 - Chap.6 概念モデリング
 - Chap.7 世界の構造のデザイン
 - Chap.8 状態遷移のデザイン
 - Chap.9 コミュニケーションのデザイン
 - Chap. 10 情報伝播のデザイン
 - Chap.11 シミュレーション結果の分析
- Part III シミュレーションデザイン演習【拡張編】 Chap.12 概念モデリング・拡張編 Chap.13 成長するネットワークのデザイン



PlatBox Project

目次

第I部

社会を理解するためのシミュレー ション

第1章

モデル化とシミュレーション

1.1 シミュレーションの意義と目的

近年、科学的研究において、シミュレーションという手法が注目を集めています。シ ミュレーションとは、用意したモデルと初期条件から、そのモデルを時間的に展開させる ということです。それを通じて、研究者はモデルの特徴についての経験的な知見を得るこ とができます。シミュレーションでは、モデルの設定や条件などを変更して試すことが容 易であり、また、頭の中ではもはや自由に操作することのできないような大規模で複雑な モデルを扱うことも可能となります。そのため、現在の科学的研究においては、シミュ レーションは理論を発展させるための非常に重要な方法となっているのです^{*1}。

シミュレーションとは、用意したモデルと初期条件から、そのモデルを時間的に展開さ せるということだと、先ほど述べました。ここで、シミュレーションという活動のイン プットは、用意したモデルと初期条件です。これは「もし~ならば」ということにあたり ます。そして、シミュレーションの結果得られた知見がアウトプットです。これは、「ど うなるか・・・」ということです。このインプットとアウトプットの関係を分析すること で、対象やモデルについての理解を深めることになります。



^{*1} コンピュータ・シミュレーションを用いた科学的研究は、1986 年にノーベル物理学賞受賞者 Kenneth G. Wilson によってその必要性が提唱されて以来、「計算科学」(Computational Science) という名前 のもとに発展してきました。計算科学は、「科学や工学の問題を解決するため、シミュレーションや実験 データ解析にコンピューターを積極的に利用して、理論や実験と補完し合う手段(実験と理論的アプローチの間にあるギャップを埋める)」(田子 2000) というものであり、科学研究の両輪と言われる「理論」と 「実験」に加えて「計算」を重視します。

科学的研究におけるシミュレーションの利用の目的には、大きく分けて次の3つのもの があります。まず、第一の目的は、モデルの振る舞いを理解するというものです。モデル を設定し、初期条件をいろいろ変えてシミュレーションを実行します。初期値の組み合わ せによって、モデルがどのような振舞いをするのかを観察します。

第二のアプローチは、対象の内部メカニズムを知るための模索に用いるというもので す。これは、全体的な振る舞いがわかっているけれど、内部のメカニズムがわかっていな いという対象を理解したい場合に行われます。内部メカニズムの仮説的なモデル(構成モ デル)を作成し、その振る舞いと対象を比較して改良し、徐々にモデルを対象に近づけて いくわけです。



第三のアプローチは、対象の将来に関する「予測」です。予め妥当と思われるモデルが あり、それを時間経過させることによってどのような結果になるのかを観察・分析すると いうものです。つまり、過去のデータを用いてシミュレーションを行うことによって、将 来の動向を予測するわけです。



しかし、「予測」という目的については、ビジネスや政策分析からの現実的要請として求 められることが多いものの、シミュレーション研究者の中では、このような利用の効果を疑 問視する声も多くあります。社会シミュレーションの先駆者である J.W.Forrester(1961) も、予測については、当初から懐疑的な意見を述べています。「とくに注意したいのは、 将来の特別な時点における特定の事象の定量的な予測が、モデルの目的には含まれてい ないということである。従来、有用なダイナミック・モデルならば、ある将来の時点にお けるシステムの特定の状態を予測できなければならないということは自明である、と間 違って考えられてきた。これは望ましいことかもしれないが、モデルの有用性は、未来に おける特定の進路を予測する能力にかかっている必要はない。」(下線は原文より)。また、 Gilbert and Troitzsch (1999) も、「従来の社会科学の科学哲学では、説明と予測を過剰に 関係づけてきたと言えるだろう。つまり、理論をテストするのに、その理論がうまく将来 を予測できるかどうかで判断される傾向があるのである。これは非線形理論、特にミクロ レベルにおいては、適切な判断基準であるとはいえない。」と注意を促しています。

以上のことから、現在では、シミュレーションは、理論を発展させるための方法として、 「特徴についての理解」や「内部メカニズムについての理解」に重きが置かれています。

1.2 モデル化

シミュレーションを語る上で、「モデル」は本質的に重要です。モデルがなければシミュ レートすることさえできません。また、モデルの内容によって、そのシミュレーション結 果が大きく変わってくるからです。

モデルとは何かという定義にはいろいろなものがありますが、ここではWilson(1990) による次のような定義を想定しておくことにしましょう。「"モデル"とは、ある人間 にとっての、ある状況、あるいは状況についての概念 (idea)の明示的な解釈 (explicit interpretation)である。モデルは、数式、記号、あるいは言葉で表すことができるが、本 質的には、実体、プロセス、属性、およびそれらの関係についての記述 (description)で ある。」

モデルによる思考では、現実世界における対象を、ある体系的なまとまりとして写し取 り、それを操作・分析して理解することで、もともとの対象を理解します。また実践的な 面では、モデルを処理したり分析したりして得られた結果をもとに、政策や行動を起こす ことになります。時々刻々と変化する複雑な社会・経済に対して、このようなモデルによ る思考を行うためには、容易かつ迅速にモデルを構築できる方法と道具立てが重要となり ます。そこで、PlatBox では、そのようなニーズも考慮に入れて、方法論やツールを開 発・提供しています。



1.3 PlatBox とは

シミュレーションを構築する際には、モデルに何を写し取るのかを決める作業と、シ ミュレーションを作成するためのプログラミング作業が不可欠となります。どちらの作業 も、白紙の状態から取り組むには負担が大きすぎるため、なんらかの支援があるとよいで しょう。

そこで、私たち PlatBox Project^{*2}では、モデル化を支援するモデル・フレームワーク や、シミュレーション作成を支援するモデリング・ツールなどを開発・提案しています。 私たちの提案するモデル・フレームワークは「PlatBox 基礎モデル」といいます。これに ついては、第2章で詳しく紹介します。提案しているモデリング・ツールは、「Component Builder」といいます。このツールを用いた演習を第 II 部、第 III 部で行います。

PlatBox では、社会・経済の「箱庭」を、たくさんの部品によってボトムアップに組み 上げ、シミュレーションを行うことができる仕組みを実現しています^{*3}。これらは、いわ ば、社会・経済を考えるための「新しい思考の道具」ということができるでしょう。

^{*&}lt;sup>2</sup> 2005 年 3 月までは、Boxed Economy Project。

^{*&}lt;sup>3</sup> もともとは、これらは「Boxed Economy 基礎モデル」および「Boxed Economy Simulation Platform」 という名称で開発されてきた。「Boxed Economy」という名称は、ブラックボックスとしての経済社会 モデルを開けてみると、そこに「箱詰めされた経済」がある、というイメージから命名したものである。 経済社会のミニチュアということから、経済社会の「箱庭」モデルということもある。もともとは経済分 野に注目して開発していたので、「Economy」という言葉が使われているが、現在では経済だけでなく、 それ以外の社会的な現象にも適用できるようになっている。そのため、2005 年 3 月に名称を「Boxed Economy」から「PlatBox」に変更した。

第2章

社会を捉えるための基礎モデル

2.1 マルチエージェントシミュレーション

私たち人間は、他者と相互作用を繰り返しながら日常生活を送っています。経済活動や 社会参加、コミュニケーションに至るまで、あらゆる場面において人と人、人と組織、組 織と組織のやり取りが行われます。そしてこの相互作用は関連する人・組織に何らかの影 響を与えています。

経済社会をこのように主体同士の相互作用からなるシステムとして捉える考え方は、近年「エージェントベースモデル」(agent-based model) として注目を浴びています。エージェントベースモデルとは、経済社会を多数のエージェント(自律的主体)のミクロ的な相互作用からなるシステムとしてモデル化したものです。

エージェントベースのアプローチでは、社会における主体は、外界を認識して自律的に 行動し、他者とコミュニケーションを行うという「エージェント」としてモデル化され ます。例えば、企業・政府・家族・学校・地域社会・国などの社会集団や個人などをエー ジェントとして捉え、それらが個別的に意思決定や行動を行うことで社会がダイナミック に変化していくと考えるのです。エージェントは他のエージェントから作用を受けること によって自身の中に変化が引き起こされたり、今度はそのエージェントが今度は他のエー ジェントの状態を変容させることもあるでしょう。外部との相互作用によってエージェン トが受ける影響は、「認識の変化」や「行動の変化」、「主体同士の関係の変化」などが考 えられます。このような相互作用のプロセスが繰り返され、複雑な社会動態が展開してい きます。

2.2 基礎モデルとは

エージェントベースモデルでは、エージェントとその相互作用によって経済社会を表現 するのですが、実はそれ以外の部分のモデル化の仕方はモデル作成者の判断に任されてい ます。このことは、モデル作成者にとって表現の自由度が高いという反面、作成者によっ てモデルの粒度や要素の捉え方が異なってしまうということを意味しています。このよう に明示的な基準がない場合には、モデルの融合やモデル部品の再利用ができないことにな り、累積的な発展の障害となると考えられます。

そこで、私たち PlatBox Project では、「PlatBox 基礎モデル」(PlatBox Foundation Model) という枠組みを定義しました。基礎モデルは、現実の経済社会をオブジェクト指向分析によって抽象化したもので、経済社会モデルの土台となりうる一つの枠組みです。この基礎モデルを用いることで、エージェントベースによる経済社会の認識・分析の参考となるだけでなく、そのモデル要素を基本語句としてモデルの記述を行うことができるようになります。基礎モデルに基づくモデルはそのまま PlatBox Simulator 上で動かすことができます。基礎モデルは、分析対象の観察から、シミュレーションの設計・実装、そして実行までを一貫して支援するモデルの枠組みなのです。



2.3 基礎モデルの要素

基礎モデルの中心的な部分は、World、Space、Clock、Entity、Agent、Goods、Information、Behavior、Relation、Channel というクラスで構成されています^{*1}。まず、対

^{*1 「}クラス」とは、モデル要素の鋳型です。クラスは、複数のオブジェクトを共通の性質ごとに分類したものであり、これを用いることによって、オブジェクトの体系的な整理と効率的な記述が可能となります。

象世界を表現する土台が「World」です。世界は、その世界に固有の空間と時間によっ て規定されており、それが「Space」と「Clock」で表されます。この「Clock」の時間 が進むことで、シミュレーション上の時間が経過します^{*2}。世界には「Entity」、すな わち「Agent」と「Goods」が複数存在します。Agent とは、社会・経済においてさま ざまな活動を行う個人や社会集団 (企業・政府・家族・学校・地域社会・国)のことです。 それぞれの Agent には、個性をもたせ、多様な振舞いや状態をとらせることができます。 Goods は、Agent に所有し交換される有形/無形の「もの」です。ここでいう「Goods」 とは、人間の欲求を充足する性質をもつという経済学における狭義の意味ではなく、世界 におけるさまざまなものを示す広義の概念として用いられています。例えば、自動車、石 油、トウモロコシ、株、土地の権利、広告、書籍、水、声、騒音、ごみ、貨幣などは、ど れも Goods オブジェクトとして表されます。そして、Agent が記憶した情報や、Goods に付随して取引される情報などは、「Information」として表されます。



Agent の行動は、「Behavior」として表現します。例えば、企業における生産行動や 販売行動、個人における購買行動や労働行動などはどれも Behavior であり、これらの Behavior によって、Goods や Information の作成や処理を行います。Agent は複数の Behavior をもつことができ、それらを並列的に実行することができます。

ある Agent から他の Agent への関連性は、「Relation」によって表現します。これに より、友人関係や家族関係、雇用関係などの関係性を表現することができます。実際のコ ミュニケーションの際には、この Relation に基づいて開設されるコミュニケーション・ パスである「Channel」を通じて、商品や会話、貨幣などの Goods と Information のや りとりを行います^{*3}。

^{*&}lt;sup>2</sup> Clock の時間が進むと、後に説明する「TimeEvent」が発生します。

^{*&}lt;sup>3</sup> Channel を通じて Goods や Information が送られてくると、後に説明する「ChannelEvent」が発生 します。

第3章

モデル作成のプロセス

ここでは、シミュレーションを作成するためのプロセスについて紹介します。各フェー ズごとの詳しい解説は、第 II 部および第 III 部で行います。

3.1 モデル作成のプロセスの全体像

シミュレーションによる研究プロセスは、一般に、「研究目的の設定」、「モデル作成」、「シミュレーション実験」、「結果の分析・評価・考察」というフェーズで構成されています。

PlatBox では、モデル作成フェーズは、さらに「概念モデリング」、「シミュレーション デザイン」、「シミュレーションの実行・検証」に分けて考えます。全体像は次のようにな ります。



まず最初に、何を見たいのか?何を分析したいのか?を明確化することが、一番最初の、 そして一番重要な作業です。そして、どのようなシミュレーションなのか、または何を解 決するシミュレーションなのかを考慮して、適切で分かりやすく、かつ愛着がもてるよう なタイトルを決めます。それが終わったら、モデル作成に入ります。モデル作成フェーズ は次のようになります。



各フェーズの詳細な説明は、次節以降で述べることとして、ここでは、各フェーズの概 略を説明します。

- 概念モデリング フェーズ どのような問題領域のシミュレーションを行うのかを明らか にするフェーズです。「基礎モデル」(基礎モデル) という概念枠組みを利用して、 シミュレーションに登場するエージェントや財の定義を行います。作成するのは、 「概念モデルクラス図」「アクティビティ図」「コミュニケーション・シーケンス図」 です。
- シミュレーションデザイン フェーズ 作成された概念モデルをもとに、PlatBox Simulator で動くシミュレーションの設計を行うフェーズです。「基礎モデルフレームワー ク」というソフトウェアフレームワークの仕様に沿って、PlatBox Simulator で動 作するようにモデルを設計します。作成するのは、「シミュレーションモデルクラ ス図」「状態遷移図」「アクションブロック図」「世界の設定」です。これらの図や 設定は、Component Builder で作成し、実行可能なプログラムに変換することが できます。
- 実行・検証 フェーズ 作成したシミュレーションモデルを、PlatBox Simulator を使っ て実行するフェーズです。また、意図した通りに動作するかを検証します。

3.2 概念モデリングでは何をやるのか?

概念モデリング フェーズの目的は、これから作るシミュレーションがどのようなもの なのかを明らかにすることです。シミュレーションに登場する「エージェント」とその 「行動」、エージェント間の「関係」、そして「財」や「情報」をすべて洗い出し、定義しま す。また、エージェントの振舞いの手順ややりとりを明らかにして記述します。

このフェーズで作成する図は、以下のものです。これらの図を、UML(統一モデリン グ言語)に基づいて記述していきます。

- 概念モデルクラス図
- アクティビティ図
- コミュニケーション・シーケンス図

3.2.1 対象世界における登場人物・事物を洗い出して定義する

タイトルが決まったら、次は登場人物・事物の洗い出しです。エージェントベースシ ミュレーションの主役はもちろんエージェントなのですが、情報などの脇役たちの活躍も 同じぐらい重要です。では、どのように洗い出せばよいのでしょうか?*1PlatBoxのアプ ローチでは、登場する人物・事物を洗い出すために「基礎モデル」(長いので、私たちは 「基礎モデル」と呼んでいます)という枠組みが提供されています。特に、ここでは、基 礎モデルで定義されている言葉(概念)のうち、次の5つに注目します。

- エージェント- Agent
- 情報- Information
- 財- Goods
- 行動- Behavior
- 関係- Relation

これら5つのキーワードにもとづいて、あなたが作るシミュレーションの登場人物・ 事物を抜き出していきましょう。抜き出した登場人物・事物は、UMLに基づくクラス図 で記述します。各モデル要素は、現実世界における次のものを表現するために用いられ ます。

【概念モデルクラス図の例】

 $\mathbf{13}$

^{*1} 基礎モデルについての詳しい説明は、リファレンス編 第 I 部第 1 章を参照してください。

3.2.2 エージェントの振舞いを分析して記述する

シミュレーションの登場人物・事物を洗い出した後は、それらがどのように振舞うのか という台本の手配です。エージェントの振舞いは、次の2つの方法で分析します。

- アクティビティ分析
- コミュニケーション・シーケンス分析

アクティビティ分析では、各エージェントの行動がどのような手順(フロー)で行われ るのかを記述します。コミュニケーション・シーケンス分析では、複数のエージェントが どのような順序(シーケンス)でコミュニケートするのかを記述します。

アクティビティ分析

まず、エージェントがどのように振舞うかを、アクティビティという概念を使って記述 します。「アクティビティ」というのは、エージェントの行為の単位だと思ってください。 このため、アクティビティは、多くの場合、「~する」という形になります。

アクティビティ図では、角の丸い長方形で示された一つ一つがアクティビティです。ア クティビティ図は、上から順番にアクティビティを行っていくというフローチャートを表 わす図です。条件分岐によってアクティビティの流れを分岐させることもできます。

【アクティビティ図の例】

コミュニケーション・シーケンス分析

次に、コミュニケーション・シーケンス図によって、コミュニケーションがどのような 順序で行われるかを記述します。アクティビティ図では、各行動がどのように動くかを記 述しましたが、多くの場合、行動は一つだけでは動作せず、他の行動と連携しながら動き ます。そのような行動の連携を把握するために、コミュニケーション・シーケンス図が役 立つのです。

基礎モデルでは、行動を連携させる方法は1つしかありません。それは、財(情報)を 送る/受け取るという方法です。つまり、行動と行動の連携はすべて財(情報)のやりと りで行われるので、コミュニケーション・シーケンス図では、そのやりとりを記述してい きます。 【コミュニケーション・シーケンス図の例】

3.2.3 フェーズ内のイテレーション

ここまでで一通りの概念モデリング作業が終わりましたが、やり残していたことがある と思います。それは、財と情報の定義です。コミュニケーション・シーケンス図を書いた ことで、どのような財や情報が登場するかが明らかになりました。

実際に概念モデリングをしてみると、登場人物・事物の分析と振舞いの分析は、どちら か一方を完成させてからもう一方にとりかかる、というアプローチで進めるのは難しいこ とが分かると思います。登場人物・事物の分析と振舞いの分析を交互に繰り返すこと(イ テレーション)が大切です。また、次のシミュレーションデザイン フェーズに進んだとき に、概念モデルに足りないものがあると気づいた場合には、概念モデリング フェーズに 戻って追加・修正することもあります。

3.3 シミュレーションデザインでは何をやるのか?

シミュレーションデザイン フェーズの目的は、概念モデリング フェーズで作成された 概念モデルを、プログラムを記述するためのモデルに変換し、実装していくことです*²。

その際には、基礎モデルに基づいたソフトウェアフレームワーク「基礎モデルフレーム ワーク」の仕様に合うように作成していきます。

このフェーズでも、概念モデリング フェーズと同様に UML を用いて記述します。このフェーズで作成するのは以下の図です。

シミュレーションモデルクラス図

• 状態遷移図

また、シミュレートする世界の初期設定についても、指定します。

^{*2} 基礎モデルフレームワークについての詳しい説明は、リファレンス編 第 I 部第 2 章、第 3 章を参照して ください。

世界の設定

3.3.1 シミュレーションモデルクラス図

シミュレーションモデルクラス図は、概念モデリングフェーズで作成した概念モデルク ラス図をもとに作成します。しっかりと概念モデリングを行っていれば、おそらくこれら は同じようなクラス図になるでしょう。

しかし、シミュレーションモデルクラス図で新しく現れるクラスもあります。まず、対 象世界には現れないものでも、シミュレーションを実行する上で必要となる要素がありま す。例えば、シミュレーション上の何らかの制約を管理するエージェントなどが、これに あたるでしょう。また、情報や行動の実装クラスとの関係を表わすクラス図なども必要に なるかもしれません。これらの設計・実装上のクラスを加えたクラス図は、概念モデルの ためというよりは、シミュレーションのプログラムを考え、理解する上で重要となります。

3.3.2 状態遷移図を作成する

さて、いよいよ状態遷移図を作成します。状態遷移図は、これまでに作ってきた以下の 図を参考に作成します。

- シミュレーションモデルクラス図
- コミュニケーション・シーケンス図
- アクティビティ図

目安としては、コミュニケーション・シーケンス図におけるメッセージが出た箇所(矢印 が出ている箇所)から、次にメッセージを受け取る箇所(矢印 が指している箇所) までの部分が、「状態」の候補になります。アクティビティ図におけるアクティビティが 「アクション」の候補になります。

概念モデリング フェーズで書いたアクティビティ図の「アクティビティ」は、絶対に 「状態」にはなりません。UML の記法では、アクティビティ図と状態遷移図は似ています が、まったく意味の異なる図なので、間違えないように注意してください。

3.3.3 アクションブロック図によって実装する

設計したモデルを PlatBox Simulator 上でシミュレートできるプログラムとして実装 します。通常はここで、プログラミング言語(たとえば、Java 言語や C 言語)によって記 述するのですが、PlatBox では、次章で紹介する Component Builder を使うことによっ て、モデル図からプログラムを自動生成することができます。詳しくは次章(第4章)で 説明します。

3.4 シミュレーションの実行・検証では何をやるのか?

シミュレーションから信頼できる結果を得るためには、モデルの正当性を検証する必要 があります*³正当性の検証とは、考えていた概念モデルが、きちんとプログラムに変換さ れたかを検証することです。そのために実行結果やログを利用してモデルの正当性の検証 を行います。。

^{*&}lt;sup>3</sup> シミュレーションにおける検証には、この他、「妥当性の検証」という検証もあります。これは作成した モデルが現実に対して妥当であるかを検証することです。詳しくは、『社会シミュレーションの技法』(N. ギルバート, K.G. トロイチュ,日本評論社, 2003)を参照してください。

第4章

モデル作成環境

PlatBox では、シミュレーションの作成を支援するツールとして、Component Builder が提供されています^{*1}。Component Builder は、5 種類のデザイナ(エディタ)と、世界 設定のためのコンポーザで構成されており、これらを組み合わせて用いることで、シミュ レーションのコンポーネントを作成できます。

4.1 概念モデリング フェーズを支援するツール

CB には概念モデリングフェーズをサポートするための3 種類のツール 「Model Designer」、「Activity Designer」、「Communication Designer」が含まれています。 モデル作成者はこれらのツールを使用して、対象領域からの概念の抽出、Agent のアクティビティの分析、 Agent 間の相互作用の分析、を反復的に行いながら概念モデルを作成していきます。



^{*&}lt;sup>1</sup> Component Builder は、統合開発環境「eclipse」のプラグインとして開発されています。eclipse は、 オープンソース・ソフトウェアであり、無料で入手することができます。機能性と使いやすさの面でも定 評がある開発環境をベースとすることで、より効率的なシミュレーション作成が可能となります。eclipse は、1999 年に OTI (Object Technology International) と IBM が共同で開発を始めたもので、その 後オープンソースプロジェクト eclipse.org によって開発・運営されています。eclipse の詳細について は、http://www.eclipse.org/を参照してください。

4.1.1 概念モデルクラス図の作成: Model Designer

概念モデルクラス図の作成には、Model Designer を使用します。Model Designer は、 UML のクラス図を使ってシミュレーションモデルの静的な構造を分析・記述するための ツールです。概念モデリングフェーズでは、モデル作成者はこのツールを使って、シミュ レーションの対象領域から概念を抽出しクラスとして記述していきます。 概念モデリン グフェーズで抽出するべき要素は、Agent、Relation、Information、Goods, Behavior の 5 種類です。

4.1.2 アクティビティ図の作成: Activity Designer

アクティビティ図の作成には、Activity Designer を使用します。Activity Designer は、UMLのアクティビティ図を使って Agent のアクティビティを分析・記述するための ツールです。分析したアクティビティと遷移は、シミュレーションデザインフェーズで Behavior の状態遷移を設計する際に利用されます。 4.1.3 コミュニケーション・シーケンス図の作成: Communication Designer

コミュニケーション・シーケンス図の作成には、Communication Designer を使用しま す。Communication Designer は、UML のシーケンス図に準じた図を用いて、Agent 間 の相互作用を分析・記述するためのツールです。モデル作成者は、Model Designer で定 義した静的な構造と、 Activity Designer で分析した Agent のアクティビティをもとに、 Agent がどのような順番で活動を行い、その中でどの Goods や Information を送りあう のかを記述します。

4.2 シミュレーションデザインフェーズを支援するツール

Component Builder にはシミュレーションデザインフェーズをサポートする4種 類のツール 「Model Designer」、「Behavior Designer」、「Action Designer」、「World Composer」が含まれています。 モデル作成者はこれらのツールを使用して、まずシミュ レーションに登場する Type の設計、 Behavior の状態遷移の設計、アクションの設計、 を反復的に行いながら設計モデルを作成していきます。 次に、シミュレーションの初期 設定を行い、対応するソースコードを生成することで、 モデル作成者は PlatBox 上で実 行可能なモデルコンポーネントを得ることができます。 シミュレーションデザインをサ ポートする4種類のツールは、設計モデルからシミュレーションのプログラムを生成す る機能を提供しています。



4.2.1 シミュレーションモデルクラス図の作成: Model Designer

シミュレーションモデルクラス図の作成には、Model Designer を使用します。Model Designer は、概念モデリングフェーズでも使用しましたが、シミュレーションデザイン フェーズでは、概念モデルで記述した静的な構造を Type の構造として定義するために 使用します。 概念モデルとして抽出した要素を、拡張可能性、再利用性の観点から再検 討し、 必要ならば分割、統合などの設計判断を行って Type として定義します。Model Designer は、定義された Type の構造からソースコードを自動で生成する機能を提供し ています。

4.2.2 状態遷移図の作成: Behavior Designer

状態遷移図の作成には、Behavior Designer を使用します。Behavior Designer は、 UML のステートチャート図を使って Behavior の状態遷移を設計するためのツールです。 Behavior Designer は、ステートチャート図で記述された状態遷移に対応するソースコー ドを自動で生成する機能を提供しています。

4.2.3 アクションブロック図の作成: Action Designer

アクションブロック図の作成には、Action Designer を使用します。Action Designer は、Behavior に定義されたアクションを設計するためのツールです。モデル作成者は、 このツールを使ってアクション記述言語を GUI で記述することができます。Action Designer は、記述したアクションは対応するソースコードに自動で変換する機能を提供 しています。

4.2.4 世界の設定: World Composer

世界の設定には、World Composer を使用します。World Composer は、シミュレー ションの初期設定を行うためのツールである。モデル作成者は、このツールを使って、 World の各種設定、Agent の初期配置の設定、Relation の初期構造の設定を行うことが できます。 これらの設定は、World のソースコードに自動で変換することができます。

World Composer は、他のツールと違い「Desginer」という名前にはなっていません。 それは、World Composer は他のツールで「Design」したモデルを「Compose」して モ デルコンポーネントとして実行できるようにするツールだからです。

4.3 モデリング言語によるシミュレーション作成の意義

4.3.1 プログラミング言語の文法に依存しないモデル作成

これまでシミュレーションのモデル作成には、支援ツールを使う使わないに限らず、最 終的には C 言語や Java 言語などのプログラミング言語による実装が必要でした。 その 結果、作成されたモデルが特定のプログラミング言語に依存し、そのプログラミング言語 の知識がない人には、モデルを作成したり、他人が作ったモデルを理解したりすることが できませんでした。

一方、UML やアクション記述言語を利用して図解でモデルを記述すると、 プログラミ ング言語の文法を気にせずに、モデルを作成することができます。 そのため、特定のプロ グラミング言語の知識がない 人でも、モデルを作成したり、他人が作ったモデルを理解す ることが容易になります。

4.3.2 作業量の軽減

図で記述されたモデルからソースコードを自動生成することにより、 モデル化以外の 本質的でない作業を削減することができます。 設計のモデルは、それぞれの図によって 定義されたルールに応じて、実行可能なソースコードへと変換されます。 このような作 業は、シミュレーションを行うプログラムを作成する上では必要なことですが、 対象領 域をモデル化するという点で本質的な作業ではありません。 ソースコードを設計モデル から自動生成することで、このようなモデル化と関係のない作業を減らし、 モデル化の 作業に集中することができるのです。

4.3.3 反復的なモデル作成のために

シミュレーションのモデル作成は、シミュレーションの正当性、妥当性*²を検証するた めに、反復的に行われます。作成したモデルは、最初から作成者の意図どおりに正しく動 作するとは限らないからです。 様々な原因からバグは発生し、多くの場合、数回の修正作 業を行う必要があります。 また、作成者の意図どおりに動いたとしても、シミュレーショ ンを行った結果 モデルの妥当性に問題があることがわかり、概念モデリングからやりな おすような場合もあります。

モデルを作成する上で、分析、設計のモデルをソースコードに変換する作業は、 円滑な 反復の妨げとなります。 実装から設計や分析の段階に戻る際には、一度頭を切り替えて モデルの本質を検討し、 次にもう一度頭を切り替えて再検討した部分を実装しなければ

^{*&}lt;sup>2</sup> ここで言う正当性 (verification) とはシミュレーションが正しく実装されていてモデル作成者の意図した とおりに 動いているかということであり、妥当性 (validatation) とはモデルの振る舞いがモデル化の対 象となる現象と 一致しているかということです。

ならないからです。

図に記述された分析、設計のモデルからソースコードを自動生成することで、前述した ような切り替えのコストは取り除くことが可能です。 ソースコードを自動生成すること で、 モデル作成者は、シミュレーションの本質的な部分だけに集中してモデルを作成す ることができるのです。

プログラミング言語によるシミュレーション構築の場合

モデリング言語によるシミュレーション構築の場合

4.3.4 信頼性の向上

シミュレーションのためのモデルを作成する上でモデルの信頼性は非常に重要な課題で す。しかもプログラムにバグはつき物です。コンピュータ・シミュレーションもその例外 ではありません。シミュレーションのプログラムは、通常のソフトウェアと違い、どのよ うな振る舞いがあるべき正しい振る舞いなのかを事前に予測することができません。 そ のため、バグの発見が通常のソフトウェアよりも困難であり、シミュレーションのモデル 作成には高い信頼性が求められます。

設計を実現するためのソースコードを自動生成することで、実装の段階で起きるバグ はなくすことが可能です。 設計のモデルをソースコードに変換する作業を人手で行った 場合、たとえ設計が正しかったとしても、 そこにはバグが混入する可能性があります。 ソースコードを自動生成することで人手による作業を減らし、より信頼性の高いプログラ ムを作成することが可能になります。

第Ⅱ部

シミュレーションデザイン演習【基 本編】

第5章

モデル作成の準備

5.1 実行環境の確認

まず正しく Component Builder がダウンロードできているかを確認します。eclipse-ForPlatBox3.0 フォルダ内にある、eclipse.exe をクリックし、Component Builder を起 動してください。起動できたら左側のパッケージ・エクスプローラーに「SampleProject」 があるか確認します。



SampleProject が存在する場合、そのまま第 5.2 節「プロジェクトとパッケージの作 成」に進んでください。もし存在しない場合、ワークスペースの位置の指定に問題があり ます。以下の手順に従ってワークスペースの切り替えを行ってください。

- 現在起動している Component Builder を一旦閉じ、再起動してください。その際、 ワークスペース・ランチャーのダイアログが表示されます。表示されない場合、メ インウィンドウ上部のメニューバーから、[ファイル]>[ワークスペースの切り替え] を選択します。
- フラウズ ボタンを押して、今起動している Component Builder の実行ファイル (eclipse.exe)が保存されている eclipseForPlatBox3.0 フォルダを探します。その 中にある、workspace フォルダを選択してください。



- 3.「ワークスペース:」に、指定した workspase のディレクトリが表示されていることを確認して「OK」を押してください。
- 4. パッケージ・エクスプローラーに SampleProject が表示されることを確認します。

5.2 プロジェクトとパッケージの作成

具体的なシミュレーション作成に入る前に、これから作成するモデルの情報を一括して 管理するための「プロジェクト」を作成しておく必要があります。プロジェクト作成は、 以下の手順で行います。

- メインウィンドウ上部のツールバーにある PlatBox プロジェクト新規作成ボタン を押します。
- 2. 表示された新規作成ウィザードの「プロジェクト名:」にプロジェクト名を入力し ます。ここでは「BoxVillage」という名前にしましょう。エラーの原因となります ので、名前には空白を入れないようにしてください。
- 3. [終了] ボタンを押すと、プロジェクトが作成されます。

もしくは、[ファイル] メニューから [新規] > [プロジェクト] > [PlatBox Project] を選 択して作成することもできます。プロジェクトが正しく作成されたかどうか、パッケー ジ・エクスプローラ上で確認しましょう。

| PlatBoxプロジェクトの新規作成 PlatBoxブロジェクトの新規作成 PlatBox Simulator上で動作するモデルを作成するためのプロジェクトを新規に作成します 合 |
|---|
| プロジェクト名体 BoxVillage プロジェクト内容 |
| マテフォルトの使用の ディレクトリー公・○¥soft¥eclipseForPlatBox3 0¥workspace¥BoxVillage フラウズ(型) |
| |
| |
| (旅7(E) キャンセル |

続いて、「パッケージ」を作成しておく必要があります。パッケージの作成は、以下の 手順で行います。

- 1. 左側にあるパッケージ・エクスプローラーの中にある「BoxVillage」プロジェクト を選択します。
- 2. メインウィンドウ上部のツールバーにある新規 Java パッケージ作成ボタンを押し ます。

- 3.「名前:」のところにパッケージ名を入力します。ここでは、「boxvillage」という 名前にしましょう。
- 4. |終了| ボタンを押すと、パッケージが作成されます。

パッケージの作成は、メニューバーの [ファイル] > [新規] > [パッケージ] からも行う ことができます。

| • 🗄 🗎 📔 🔂 🗟 🚾 Cì 🏇 • 🕥 • (| <u>}</u> • @⊕ ♂ @ ♂ ☆ ₪ ₪ ◘ ∎ 🖪 🐻 🚳 ₫ ☷ מ | |
|--|--|--|
| 📲 パッケージ・エクスプローラー 🔀 🛛 📴 🗖 🗖 | | |
| ← → 雨 🖹 🕏 👻 🌅新規 Java パッケージ | | |
| BoxVillage Box B→ JRE システム・ライブラリー [2re1.4.2_08] B→ JPLATBOX_HOME/platbox.jar - C¥Progr | Java バッケージ Java バッケージを作成します。 | |
| ₽ ATBOX_HOME/lib/crewlib.jar - C*Pr ₽ ATBOX_HOME/lib/log4j.jar - C*Prog ₽ AmpleProject | パッケージに対応するフォルダーを作成します。 ソース・フォルダー(D)、BoxVillage/src | |
| | | |
| | | |
| | | |
| | | |
| | 「練了住」」キャンセル | |

以上の通りプロジェクトとパッケージを作成すると、パッケージ・エクスプローラー内 は、次のようになっているはずです。



以降、作成する一連のモデルは、このプロジェクトで一括して管理することにします。

^{道案内} 準備は以上です。次章からさっそくこの世界を作っていくことになります が、一度に世界のすべてを作るわけではありません。ストーリーを追いながら段階ごとに 説明していきますので、常に自分がストーリーの中のどの部分を作っているかを意識して みてください。それでは、始めましょう!
第6章

概念モデリング

【学習目標】

作りたい世界のシナリオから、モデルの要素を抽出できるようになる(基礎モデル)

6.1 つくりたい世界

まず、これからつくる世界をもう一度確認します。

旧友を訪ねて BoxVillage に立ち寄った旅人は、再会した旧友に土産話をします。その 話を聞いた旧友は、BoxVillage に住んでいる自分の友人たちに土産話を広めます。今度 は旧友の友人である村人たちも、各々がその話を自分の友人に広めていき、やがて村全体 に旅人の土産話が広まっていく、というシナリオでした。

この章では、つくりたい世界のシナリオを要素に分解して記述していきます。その際の 枠組みとなるのが基礎モデルです。この基礎モデルで定義された言葉を使ってシナリオを 切り取っていくことで、モデル化からシミュレーション作成までできるようになっていま す。それでは、さっそく始めましょう。



6.2 登場する事物を定義する

6.2.1 エージェントを抽出する

はじめにシナリオからエージェントを抽出します。登場事物をエージェントとして抽出 する基準は「振る舞いを持つもの」です。

そのような基準をもとにシナリオを確認すると、エージェントとして次のものが抽出されます。

● 土産話をする「旅人」

● 土産話を広める「村人」

旅人の旧友も、土産話を広める「村人」として考えられます。

抽出されたエージェントを、概念モデル・クラス図として記述すると、次の通りになります。

| < <agent type="">> [2] 旅人</agent> |
|--|
| |
| < <agent type="">>> 照村人</agent> |

6.2.2 エージェント間の関係を抽出する

エージェントが抽出されたら、次は、抽出されたエージェント間の関係を抽出します。 関係を抽出する基準は「エージェント同士が何らかのコミュニケーションを行っている」 です。

登場するエージェントは旅人と村人なので、その関係について考えます。旅人と村人の コミュニケーションという観点でシナリオを見ると、まず旅人が村人に土産話をしていま す。ここで、旅人と村人が土産話をやりとりをするための関係があることがわかります。 今回は、このような関係を旅人と村人の「旧友関係」とします。

また、土産話を聞いた村人は、別の村人にその話を広めます。従って、村人同士にも関 係があることがわかります。これを、「友人関係」とします。

先ほどの概念モデル・クラス図に「旧友関係」と「友人関係」を書き加えると、次のようになります。



上の図では「村人」から「村人」へ「友人関係」が結ばれています。これは、ある一人 の村人が、自分自身への友人関係を持っていることを表しているのではありません。概念 モデル・クラス図において、「村人」はある特定の人物ではなく、概念上の存在として描 かれています。ですから、この図では、ある村人と別の村人との間に友人関係が結ばれて いる、ということを示しています。

6.2.3 エージェントの所有する財を抽出する

エージェントとその関係性の抽出が終わったら、次は、エージェント同士が交換した り、所有する財を抽出します。

しかし、今回のシナリオでは旅人と村人は財の取引をしないため、ここで追加する要素 はありません。

6.2.4 エージェントの所有する情報を抽出する

次に、エージェントの所有する情報を抽出します。

シナリオによると、旅人が村人にする「土産話」が抽出されます。

抽出が終わったら概念モデル・クラス図を記述しますが、ここでは、エージェントの所 有する情報に注目して記述したいので、前節までに作成した「エージェントとその関係 性」をあらわした概念モデル・クラス図とは別に、「エージェントと情報」を表した新し い概念モデル・クラス図として記述することにします。抽出された情報「土産話」と、そ れを所有する可能性のあるエージェント「旅人」「村人」の間に関連を引きます。



6.2.5 エージェントの行動(振る舞い)を抽出する

最後に、エージェントの行動(振る舞い)を抽出します。

シナリオによると、旅人からは旧友を訪ねて村に立ち寄る「立ち寄り行動」、村人から は土産話を他の村人に広める「おしゃべり行動」をそれぞれ抽出することができます。先 ほどの概念モデル・クラス図に「立ち寄り行動」「おしゃべり行動」を書き加えると、次 のようになります。



6.2.6 抽出した要素のまとめ

ここまでで抽出された要素は次の通りです。



道案内 つくりたい世界の中から、どのようにして要素が抽出されているのかわか りましたか?この章で行った概念モデリングをもとに、次章から ComponentBuilder と PlatBoxSimulator を使って、実際にシミュレーションを作成していきます。

第7章

世界の構造のデザイン

【学習目標】

登場事物のクラス図を描けるようになる(ModelDesigner) 世界の設定ができるようになる(WorldComposer) 作成したシミュレーションを実行できるようになる(PlatBox Simulator)

7.1 つくりたい世界

この章からいよいよ実際に Component Builder を使って、BoxVillage の世界をつくっ ていきましょう。まずは、概念モデリングをもとに BoxVillage の世界にどのような登場 人物がいて、どのような振る舞い、関係、情報を持っているのかをデザインします。あな たの手で BoxVillage の世界をつくり上げてください!



7.2 Model の作成



ここでは、Model Designer を用いて、どの Agent がどの Relation でお互いにつながっ ているのかを示す「Agent 間の関係性に関するクラス図」と、Agent がどの Behavior を もち、どのような Information を持っているのかを示す「Agent の行動に関するクラス 図」の2つを書いていきます。下にある図が、これから書くものです。 上述の内容は、1 つのクラス図に集約することができますが、ごちゃごちゃして分かりに くくなるため、注目したい側面に応じていくつかの図に書き分けることが一般的です。





7.2.1 Model の新規作成:旅人と村人の関係

1. パッケージの選択

パッケージ・エクスプローラーから、ファイルを新規作成したいプロジェクトの src フォル ダにある <u>パッケージ</u>を選択します。ここでは「boxvillage」を選択してください。

2. 新規作成ウィザードの起動

メインウィンドウの上部のツールバーにある¹² Model 新規作成ボタンを押します。



3. Model 名の入力

ファイルを保存する場所が、先ほど選択した場所であることを確認して、「ファイル名:」の 部分に、作成するモデルの名前を入力します。ここでは、「旅人と村人の関係」とすること にしましょう。入力ができたら、 終了 を押します

| そりラス図の新規作成 | X |
|---|-------------|
| クラス図の新規作成 クラス図を新規作成します | |
| 親フォルダーを入力または選択(<u>E</u>): | |
| BoxVillage/src/boxvillage | |
| JETEmitters DoxVillage DoxVillage DoxVillage DoxVillage DoxVillage DoxVillage DoxVillage | |
| ファイル名(N)(旅人と村人の関係) | |
| <u></u> | |
| | 「「「「」「キャンセル |

4. Model ファイルの確認

パッケージ・エクスプローラー内に「^図旅人と村人の関係.model」というファイルが作成されていることを確認してください。

7.2.2 Model のクラス図作成

「旅人と村人の関係」クラス図

1. Model Designer の起動

Model Designer を起動するには、パッケージ・エクスプローラー内の model ファイルをダ ブルクリックします。ここでのファイル名は、「^図旅人と村人の関係.model」です。

2. 要素の選択と配置: Agent

Model Designer のパレットにある Agent ボタンをクリックし、Model Designer の キャンバス(白い作図領域)をクリックして、AgentType(クラス)を配置します。これを 繰り返し、2つの AgentType(クラス)を作成します。

3. 名前の変更

配置した段階では一時的な名前がつけられているだけなので、モデルに合った名前に変更します。AgentType(クラス)をダブルクリックして文字を反転させると、名前が変更できます。ここでは、Agentを「旅人」と「村人」に変更します。



4. 要素の選択と配置: Relation

Model Designer のパレットにある 『Relation ボタンをクリックし、キャンバスをクリッ クして、RelationType (クラス)を配置します。これを繰り返し、2つの RelationType (クラス)を作成します。

5. 名前の変更

RelationType (クラス)をダブルクリックして文字を反転させて、名前を変更します。ここでは、Relationを「旧友関係」と「友人関係」に変更します。



6. 関連を引く

クラス間に関連線を描いて、これらのクラスの関連性を表現します。Model Designer のパ レットにある / 関連 ボタンを押して、関連線の作成モードにします。キャンバス上の「旅 人」クラスをクリックし、その後、「旧友関係」クラスをクリックして関連線を結びます。同 様にして「旧友関係」クラスと「村人」クラスにもう一本の関連線を、そして「村人」クラス と「友人関係」クラスに関連線を2本引きます。関連線の形を変えるには、関連線をクリッ クして現れる真ん中の点をドラッグしてください。



7. 登録の確認

Model Designer の右側にあるアウトライン上で、「図旅人」「図村人」「図旧友関係」「図友 人関係」が登録されたことを確認します。パッケージ「boxvillage」が開いていない場合は、 そのすぐ左の + をクリックして、パッケージを開くと、その内容が表示されます。 アウトラインが表示されていない場合、メニューバーの [ウィンドウ] [ビューの表示] [アウトライン] をクリックします。



8. ファイルの保存

Ctrl キーを押しながら S キーを押して、クラス図を保存します。

注意:モデル要素として登録されている内容は、その時点でキャンバスに描かれてい る内容ではなく、アウトライン上に表示されているものです。モデル要素を誤って追加し てしまった場合は、そのクラスを選択して右クリックし「モデル要素を削除する」か、ま たはアウトライン上に表示されているクラスを選択し、同様にして削除する必要があり ます。 7.2.3 Model の新規作成:旅人と村人の行動

旅人と村人の関係のクラス図と同じようにつくります。

1. パッケージの選択

パッケージ・エクスプローラーから、「boxvillage」を選択してください。

- 新規作成ウィザードの起動
 メインウィンドウの上部のツールバーにある¹⁰ Model 新規作成ボタンを押します。
 Model 名の入力
 - ファイルを保存する場所が、先ほど選択した場所であることを確認して、「ファイル名:」に 「旅人と村人の行動」と入力しましょう。入力ができたら、「終了」を押します。
- Model ファイルの確認
 パッケージ・エクスプローラー内に「^図旅人と村人の行動.model」というファイルが作成されていることを確認してください。
- 7.2.4 Model のクラス図作成

「旅人と村人の行動」クラス図

それでは、以下のクラス図のようにモデル要素を配置してみることにしましょう。

| 旅人と村人 | の行動のクラス図 (Cagent type)) 國族人 図立5者の行動 | l |
|-------|--|---|
| | <gnformation type="">> 國土產語</gnformation> | l |
| | (<agent type="">) 【 村人 【 掛しゃべり行動</agent> | |

1. 要素の配置: Agent

「旅人」「村人」に関しては、「住人の関係と情報」クラス図において作成したので、Model Designer の右側にあるアウトライン上にある「図旅人」「図村人」からキャンバス上にド ラッグ&ドロップします。



 2. 要素の配置と名前の変更:Information,Behavior InformationType と BehaviorType は、それぞれパレットから Information ボタン、
 Image: Behavior ボタンをクリックし、キャンバスをクリックして配置します。



配置ができたら、完成したクラス図を参考に、それぞれ名前を変更してください。

3. 関連を引く

パレットにある / 関連 ボタンを押して、関連線の作成モードにします。完成したクラス図 を参考に、関連線を引いてみましょう。

4. 登録の確認

作成が終わったら、Model Designer の右側にあるアウトライン上で、「回土産話」「 立ち寄り行動」「 むしゃべり行動」が新たに追加されたことを確認します。パッケージ 「boxvillage」が開いていない場合は、そのすぐ左の + をクリックして、パッケージを開 くと、その内容が表示されます。



確認したら、Ctrl + S を押し、クラス図を保存します。

7.2.5 Model のソースコード生成

1. コード生成の実行

Model Designer のキャンバスまたはアウトライン上で右クリックし、表示されるメニューから「モデルのソースコード生成」を選択します。



2. 生成するタイプの選択

生成する Type の選択ダイアログが表示されるので、作成したモデル要素にすべてチェック が入っているかを確認し、 OK を押すと Model のソースコード (Java プログラム)が生 成されます。

| €生成するTypeの選択 | X |
|---|----------|
| モデルの名前: boxvillage.BoxVillageModel | |
| Ivpe | Priority |
| 12月1日 | - |
| 1 2 2 旅人 | - |
| ┃ 🖸 🔁 おしゃべり行動 | - |
| □ 🔁 🔁 立ち寄り行動 | - |
| ▶ 🗹 🛃 友人関係 | - |
| | - |
| 「「「」」「「」」」「「」」「」」「「」」「」」「」」「」」「」」「」」「」」 | - |
| V | |
| | |
| | |
| | |
| | |
| | |
| 1 | |
| (OK | キャンセル |

✓ 生成する Type の選択において、同じ名前の Type、または余分な Type がある場合、 そのままソースコードを生成するとエラーの原因となります。一度ダイアログを閉じ、ア ウトライン上から不要な Type の上で右クリックし、「モデル要素を削除する」を選択し て Type を削除してください。

3. 生成されたファイルの確認

パッケージ・エクスプローラー内に、「☑BoxVillageModel.java」というファイルが表示されていることを確認してください。これがいま生成されたファイルです。



Model のソースコードを生成すると、「^DBoxVillageModel.java」の^Dに「!」が 入っていると思います。しかしここではGoodsTypeを使用していないために入っている だけなので、気にせず、次に行きましょう。

7.3 世界の初期設定



モデルのデザインが終わったら、いよいよシミュレーションのための世界を作成しま す。ここでは、World Composer を使い、世界に登場させる各 Agent の数、Agent 同士 の関係等を設定します。

どのようなシミュレーションを行いたいのかは、以下の通りです。



7.3.1 World の新規作成

1. 新規作成ウィザードの起動

World を新しく作成したいプロジェクトのパッケージ(boxvillage)を選択した状態で、メ インウィンドウ上部のツールバーにある[®]World 新規作成ボタンを押します。



2. World 名の入力

表示された新規作成ウィザードの「ファイル名:」の部分に、作成する World の名前を入力 します。ここでは、「BoxVillageWorld」とすることにしましょう。名前には空白を入れな いようにしてください。

3. World ファイルの生成 新規作成ウィザードの 終了 ボタンを押すと、World ファイルが生成されます。

| Morldの新規作成 | X |
|------------------------------------|-----------------------|
| Worldの新規作成 Worldを新規作成します | (|
| 親フォルダーを入力または選択(E): | |
| Box Village/src/boxVillage | |
| | |
| .JETEmitters | |
| BoxVillage | |
| | |
| boxvillage | |
| 🖅 😂 SampleProject | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| ファイル名(N) BoxVillageWorld | |
| 拡張(A) >> | |
| | |
| | |
| | |
| | Industry and a second |
| | |

 4. 生成された World ファイルの確認 パッケージ・エクスプローラー内に「愛BoxVillageWorld.world」というファイルが作成さ れていることを確認してください。

7.3.2 World の設定

はじめに、旅人を BoxVillage の世界に 1 人、村人を 5 人登場させます。

[旅人グループの設定]

Agent グループ設定ウィンドウを開く
 World Composer の画面をスクロールさせ、「Agent グループ」の部分を表示します。
 Agent グループの枠内にある 追加 ボタンを押し、Agent グループ設定ウィンドウを開きます。

| Agent/ | ブループ | | 1 |
|--------|------|----------|---------|
| | 名前 | 説明 | 数 |
| | | | |
| | | | |
| -1 | | | |
| ^ | | | |
| 7 | | | |
| _ | | | |
| | | | |
| | | | |
| | | | |
| | | | ~ |
| | | | 加 変更 削除 |
| | | <u> </u> | |

- Agent グループの名前の入力
 「名前:」のところに、Agent グループの名前を入力します。ここでは、「旅人グループ」と することにします。
- 3. Agent グループの AgentType の選択

次に「AgentType:」のところから、それらの Agent グループに設定したい AgentType を 選びます。ここでは、「AGENTTYPE_旅人」を選択します。

| 🥌 Agentグループの設定 | | × |
|---|--------------|-------------------------|
| 名前: | | |
| Agent Type: | | |
| AGENTITIPE AND | | |
| Behavior | | |
| Behavior Type | | |
| ■ BoxVillageModel#BEHAVIORTYPE_立ち寄り行動 ■ BoxVillageModel#BEHAVIORTYPE おしゃべり行動 | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| Goods | | |
| Goods Type | | 量 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| 1. 1 | | |
| Information | 1.1.1.1 | |
| BoxVillageModeばINFORMATIONTYPE 十座話 | org.platb | ox.components.cell.Cell |
| | or gap io to | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | OK #+>tell |

4. Agent グループの登録

OK ボタンを押すと、Agent グループの設定が登録されます。Agent グループの表のなかに、「旅人グループ」の設定が表示されていることと、「旅人グループ」の数が「1」に設定されていることを確認してください。

[村人グループの設定]

 Agent グループ設定ウィンドウを開く
 Agent グループの枠内にある 追加 ボタンを押し、Agent グループ設定ウィンドウを開き ます。

| | 名前 | 見日月 | 数 |
|----------|--------|-----------------------------|-----------|
| | 旅人グループ | (旅人)Obehaviors,Ogoods,Oinfo | 1 |
| -1 | | | |
| <u> </u> | | | |
| <u>×</u> | | | |
| | L | | |
| | | | |
| | | | |
| | | | (這加)変更 削除 |

- 2. Agent グループの名前の入力
 - 「名前:」のところに、「村人グループ」と入力します。
- 3. Agent グループの AgentType の選択

次に「AgentType:」のところから、「AGENTTYPE」村人」を選択します。

| 🚝 Agentグループ(| D設定 | |
|--------------|--------------|---|
| 名前: | 村人グループ | |
| Agent Type: | AGENTTYPE_村人 | - |

4. Agent グループの登録

OK ボタンを押すと、Agent グループの設定が登録されます。Agent グループの表のなかに、「村人グループ」の設定が表示されていることと、「村人グループ」の数が「1」に設定されていることを確認してください。

Agent グループの値設定ウィンドウを開く
 Agent グループの表の中にある、「村人グループ」の行の「数」の列のセルをクリックすると、そのセルの右端に編集ボタンが表示されます。この編集ボタンを押すと値設定ウィンドウが表示されます。

| Agent | ヴループ | | |
|-------|--------|-----------------------------|----------|
| | 名前 | | 数 |
| | 旅人グループ | (旅人)Obehaviors,Ogoods,Oinfo | 1 |
| | 村人グループ | (村人)Obehaviors,Ogoods,Oinfo | 1 (🔍) |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | ŀ | | |
| | | | 追加 変更 削除 |
| | | | |

6. 数の入力シミュレーションの初期状態において、この Agent グループのエージェン トを何人登場させるのかを設定します。ここでは、「5」人にすることにしましょう。

| 目値 | | | × |
|----|------------|---------------------------------|------------|
| 型: | int | T | |
| 値: | ◎ 値の直接入力 | | [5] |
| | ○ 乱数の幅の入力 | 最小値(以上) 最大値(未満) 乱数ジェネレータ: | |
| | ○ パラメータの選択 | | |
| | | | OK +vytell |

Agent グループの数の確認値設定ウィンドウの OK を押すと、「村人グループ」
 の設定が登録されます。「村人グループ」の数が「5」となったことを確認してください。

以上の作業が終わったら、Agent グループの表示が次のようになっていることを確認してください。



[旅人グループと村人グループの関係の設定]

次に、旅人グループと村人グループの関係を設定します。具体的には、新しい情報を発 信する旅人グループと村人グループは相手に対して「旧友関係」を持つものとします。

1. Relation グループ設定ウィンドウを開く

World Composer の画面をスクロールさせ、「Relation グループ」の部分を表示します。 Relation グループの枠内にある 追加 ボタンを押し、Relation グループ設定ウィンドウを 開きます。



- Relation グループの名前の入力
 「名前:」のところに、Relation グループの名前を入力します。ここでは、「旅人と村人の ネットワーク」とすることにします。
- Relation グループの RelationType の選択
 「RelationType:」のところから、それらの Relation グループに設定したい RelationType
 を選びます。ここでは、「BoxVillageModel#RELATIONTYPE_旧友関係」を選択します。
- 関係を結びたい Agent グループの指定
 関係を結びたい Agent グループを指定します。Relation グループ設定ウィンドウの左側
 「関係元の Agent グループ」と右側「関係先の Agent グループ」の Agent グループを選択 することで、指定します。
 ここでは、「関係元の Agent グループ」が「旅人グループ」、「関係先の Agent グループ」が 「村人グループ」となるように、それぞれのチェックボックスにチェックを入れてください。
 5. 関係の方向の指定

結びたい関係が、一方通行の関係なのか、相互関係なのかを指定します。Relation グループ 設定ウィンドウの真ん中にある「方向:」のプルダウンで、「双方向」を指定します。

6. Relation パターンの指定

Relation のパターンを指定します。「Relation パターン:」のプルダウンで「Random」を

| 🌽 Relationグループの設定 | | × |
|--------------------------------------|---|--|
| 関係元のAgentグループ | 名前: | 関係先のAgentグループ |
| 関係元のAgentグループ 一 旅人グループ 11人グループ | 名前: ホ人と村人のネットワーク Relation Type: BoxVillageModel#RELATIONTYPE」日友関係 ▼ 方白: 双方向 マ Relationパターン: Random ▼ 3 3 3 3 3 3 3 3 3 3 3 3 3 | 関係先のAcentグループ IGAグ |
| | | |

選択し、「本数:」には「1」を指定します。

7. Relation グループの登録

Relation グループ設定ウィンドウの OK ボタンを押すと、Relation グループの設定が登録されます。Relation グループの表のなかに、「旅人と村人のネットワーク」の設定が追加されたことを確認してください。

[村人グループ同士の関係の設定]

続いて、村人グループ同士の関係を設定します。具体的には、村人グループはお互いに ランダムの「友人関係」を持つものとします。また、ここでの「友人関係」は方向性を持 つものとして考えます。つまり、自分から何かを話す相手としての友人関係と、逆に自分 が話を聞く相手としての友人関係が存在する、ということです。今回は、その関係をラン ダムに持たせることにするので、ある村人の間には相互の友人関係が結ばれることもあり ますし、別の村人の間には、一方向的な友人関係のみ結ばれることがあります。

- Relation グループ設定ウィンドウを開く Relation グループの枠内にある 追加 ボタンを押し、Relation グループ設定ウィンドウを 開きます。
- 2. Relation グループの名前の入力
 - 「名前:」のところに、「村人のネットワーク」とすることにします。
- 3. Relation グループの RelationType の選択
 - 「RelationType:」のところから、「BoxVillageModel#RELATIONTYPE_友人関係」を 選択します。
- 4. 関係を結びたい Agent グループの指定

Relation グループ設定ウィンドウの「関係元の Agent グループ」に「村人グループ」、「関 係先の Agent グループ」にも「村人グループ」となるように、それぞれのチェックボックス にチェックを入れてください。

5. 関係の方向の指定

Relation グループ設定ウィンドウの真ん中にある「方向:」のプルダウンで、「一方向」を指定します。

6. Relation パターンの指定

Relation のパターンを指定します。「Relation パターン:」のプルダウンで、「Random」 を、「本数:」に「2」を指定します。

| 🊝 Relationグループの設定 | | × |
|-------------------|-----------------------------------|---------------|
| 関係元のAgentグループ | 名前: | 関係先のAgentグループ |
| 1日本人グループ | 村人のネットワーク | 協人グループ |
| | Relation Type: | |
| | BoxVillageModel#RELATIONTYPE 友人関係 | |
| | | |
| | | |
| | | |
| | Polation (2) | |
| | Random | |
| | 本業1: | |
| | | |
| | | |
| | | |
| | 8 7 8 | |
| | | |
| | - | |
| | | |
| | | |
| | | |

7. Relation グループの登録

Relation グループ設定ウィンドウの OK ボタンを押すと、Relation グループの設定が登録されます。Relation グループの表のなかに、「旅人と村人のネットワーク」の設定が追加されたことを確認してください。

以上の作業が終わったら、Relation グループの表示が次のようになっていることを確認してください。

| | 名前 | Relationパターン | 説明 | | |
|---|--------------|--------------|------------|--|--|
| | 旅人と村人ネットワーク | Random | 1<>1(旧友関係) | | |
| | 村人のネットワーク | Random | 1>1(友人関係) | | |
| | | | | | |
| | | | | | |
| 7 | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | ietn 亦再 auge | | | | |

確認が終わったら、Ctrl + S を押し、World の設定を保存してください。

7.3.3 World のコード生成

1. コード生成の実行

World Composer の最下部にある | World の生成 | ボタンを押します。



プログレスバーが表示され、World のコード (Java プログラム)が生成されます。

2. 生成されたファイルの確認

パッケージ・エクスプローラー上に、「^IBoxVillageWorld.java」というファイルが表示されていることを確認してください。



これが、いま生成されたファイルです。

3. ファイルの保存

 Ctrl
 +
 S
 を押し、「¹ BoxVillageWorld.java」を保存します。

7.4 実行・検証



7.4.1 シミュレーションの起動(開発環境上での操作)

実行する World の選択
 パッケージ・エクスプローラー上で、実行したい World の java ファイルを選択します。こ

こでは、「¹BoxVillageWorld.java」になります。

2. 実行

ッールバーにある[●]実行ボタンの右側の 部分を押すと、実行メニューが表示されます。 実行メニューの中から [次を実行 (R)] [Java アプリケーション] を選ぶと、PlatBox Simulator が起動します。

保存されていないファイルが存在すると、実行時に「リソースの保管」ダイアログが 表示されます。保存したいファイルにチェックを入れ、 OK を押してください。

7.4.2 シミュレーションの実行 (PlatBox Simulator 上での操作)

World の読み込み状態の確認

PlatBox Simulator の制御パネル上には、読み込まれた World の名前が表示されます。ここでは、「BoxVillageWorld」と表示されていることを確認してください。



「世界が読み込まれていません」という表示が出る場合、モデル作成においてなんら かのエラーがあると考えられます。ソースコードは生成されているか、世界の設定は正し く行われているか、などをチェックしてみてください。

2. ビューアの表示

メニューから [ビューア] [Communication Viewer] を選択して、Communication Viewer

を開きます。

「旅人」と「村人」が登場し、「旅人」と一人の「村人」との間に双方向の「旧友関係」が表示され、「村人」の間には「友人関係」が表示されたことを確認してください。



7.4.3 PlatBox Simulator の終了

1. PlatBox Simulator の終了

PlatBox Simulator を終了するには、メインウィンドウの右上にある × ボタンを押すか、 メニューから [ファイル] [終了] を選択します。

7.5 これまでにつくったモデルの全体像

本章では、クラス図を用いて BoxVillage に登場する様々な人物、関係、情報、行動を 定義しました。



道案内 前章で行った概念モデリングをもとにして、登場人物の関係や行動のクラ ス図、そして世界の構造をデザインしました。次章からは、ついに旅人が動き出します。 BoxVillage の情景を頭に思い浮かべながら、先へと進みましょう。

第8章

状態遷移のデザイン

【学習目標】

エージェントの行動を状態遷移図で描けるようになる(BehaviorDesiner)

8.1 つくりたい世界

旅人は、旧友を訪ねて村に立ち寄りました。旧友に土産話をし、その後、旅人は休む ことにしました。旅人が到着してから、休息するまでの一連の流れをつくることにしま しょう。



8.2 Behavior のデザイン



Component Builder の Behavior Designer を用いて、Behavior の状態遷移図を作成 します。ここでは、以下の「立ち寄り行動」の状態遷移を記述していきます。

8.2.1 Behavior の新規作成

- 既存の Model ファイルを開く パッケージ・エクスプローラー上の「^四旅人と村人の行動.model」をダブルクリックして ください。その Model ファイルを読み込んだ状態で、Model Designer が起動します。
- 2. Behavior の新規作成

Model Designer のキャンバスにある「立ち寄り行動」を右クリックし、表示されたメ ニューのなかから「Behavior Designer で開く」を選択します。プログレスバーが表示さ れ、Behavior ファイルが生成されます。また、その Behavior ファイルを読み込んだ状態 で、Behavior Designer が起動します。



3. 生成されたファイルの確認

パッケージ・エクスプローラー上に、「^回立ち寄り行動.behavior」というファイルができた ことを確認してください。

8.2.2 状態、状態遷移の描画

これから作成する状態遷移図と、そこに登場する状態は、以下の通りです。



【状態遷移図の要素の説明】

- 状態 角の丸い長方形が状態を表します。状態は行動内でユニーク(固有)な名前をも ち、行動は常にどれか1つの状態をもちます。黒丸は初期状態を表し、行動が 生成された直後には初期状態になります。
- 遷移 状態が異なる状態へ変化することを遷移といい、その方向を矢印で示します。
- イベント 遷移を引き起こすきっかけです。定義されているイベントの中で、もっと も重要なものは以下の2つです。
 - TimeEvent 時間が経過すると全てのエージェント、行動が1つずつ受信します。
 - ChannelEvent 経路を通して情報・財が送られてくると、受信する行動が受信 します。

遷移にイベントを書かないと「自動遷移」の意味になり、自動的に遷移が起 こります。

- アクション 行動が実行する動作の単位で、遷移が起こる際に行われます。
- ガード条件 遷移するときの条件ををガード条件といいます。ガード条件が記述され ていると、たとえイベントを受け取ったとしても、条件が一致しない限り遷移 は起こりません。

1. 要素の選択と配置:初期状態

Behavior Designer のパレットにある 初期状態 ボタンを押した後、キャンバス(白い 作図領域)をクリックして配置します。

| | 🔓 選択 |
|---|--------------------------|
| | □□_ 囲み枠 |
| | 🕞 状態 🔹 🖈 |
| | □状態 |
| 4 | 初期状態 |
| | ● 終了状態 |
| | ◎ 履歴状態 |
| | 🕞 遷移 🛛 🖈 |

2. 要素の選択と配置: 状態

Behavior Designer のパレットにある 🔤 状態 ボタンを押した後、キャンバスをクリック して配置します (ここでは 2 つの「状態」を配置します)。

3. 名前の変更:状態名

「状態」は、配置した段階では一時的な名前がつけられているので、モデルに合った名前に 変更します。「状態」をダブルクリックして文字を反転させると、名前が変更できます(こ こでは、「到着状態」と「休息状態」にしましょう)。



4. 要素の選択と配置: 遷移

Behavior Designer のパレットにある → 遷移 ボタンを押して、遷移線の作成モードにし ます。まず、キャンバス上の 初期状態 をクリックし、その後、「回到着状態」をクリッ クして、遷移線を結びます。

そして、「回到着状態」をクリックしてから、もう一方の状態「回休息状態」をクリックして、遷移線を結びます。



5. 配置の調整

Behavior Designer のパレットにある │ ^除 選択 │ ボタンを押すと、これまで配置した要素

を選択できるようになります。要素の場所を移動したい場合には、要素をドラッグ&ドロッ プして移動させることができます。また、遷移線は一度クリックした後、直線部分の真ん中 にある点をドラッグすると、折り曲げることができます。これらの機能を使って、図を見や すい配置にしましょう。

6. 遷移の設定

遷移がどのようなときに起こり、どのようなガード条件を満たしたらどのようなアクションを行うのかを設定するのは、遷移設定ダイアログで行います。 遷移設定ダイアログは、遷移線をダブルクリックして開きます。

ここではまず、「回到着状態」から「回休息状態」への遷移をダブルクリックして、遷移設定 ダイアログを開きます。

(a)受信イベントの設定

この状態遷移は、時間が経つと遷移が起きるので、「イベント」のプルダウンで 「TimeEvent」を選択します。

| ~ > 0 | | |
|--|----------|------------|
| 🧲 連移のプロパティ | | |
| -1/2 | | |
| The second secon | | |
| TimeEvent | • | |
| AUTO | | |
| G TimeEvent | | |
| ChannelEvent | 15 | |
| UpenUnanneiEvent | | 新しい内部ガード条件 |
| | | |
| | | |

これで必要な設定が終わったので、 OK を押して設定を終了します。

状態遷移図の遷移のところに、「TimeEvent」と表示されたことを確認してください。

ここまでの作業が終わったら、前の状態遷移図と一致していることを確認してくださ い。

確認が終わったら、 Ctrl + S を押し、状態遷移図を保存しましょう。

8.3 アクションのデザイン



アクションでは、Behavior の中身を具体的に決めていきます。しかし、具体的なアクションは次章以降で作成するので、ここでは「四立ち寄り行動.action」ファイルを生成するだけにします。

8.3.1 アクションファイルの新規作成

1. 既存の Behavior ファイルを開く

パッケージ・エクスプローラー上の「[®]立ち寄り行動.behavior」をダブルクリックしてくだ さい。その Behavior ファイルを読み込んだ状態で、Behavior Designer が起動します。

2. アクションファイルの新規作成

今起動した Behavior Designer のキャンバス上で <u>右クリック</u>し、表示されたメニューのな かから「Action Designer で開く」を選択します。

プログレスバーが表示され、Action ファイルが生成されます。また、そのアクションファ イルを読み込んだ状態で、Action Designer が起動します。



3. 生成されたファイルの確認

パッケージ・エクスプローラー上に、「☑Abstract 立ち寄り行動.java」と「四立ち寄り行動.action」というファイルができたことを確認してください。



8.3.2 アクションのコード生成

1. ソースコード生成の実行

「習立ち寄り行動.action」を読み込んだ状態で Action Desginer が起動したら、キャンバス を <u>右クリック</u>して「ソースコード生成」を選択します。プログレスバーが表示され、アク ションのソースコード(Java プログラム)が生成されます。



2. 生成されたファイルの確認

パッケージ・エクスプローラー上に、「□立ち寄り行動.java」というファイルが表示されて いることを確認してください。

これが、いま生成されたファイルです。



3. ソースコードを保存する

ファイルの確認が終わったら、 Ctrl + S を押し、ソースコードを保存します。

8.4 世界の初期設定



Behavior、アクションのデザインが終わったら、世界の設定をします。どのようなシ ミュレーションを行いたいのかは、以下の通りです。



8.4.1 既存の World を修正する

1. 既存の World を開く

パッケージ・エクスプローラー上の「愛BoxVillageWorld.world」をダブルクリックします。 この World ファイルは前章で作成した World ファイルです。

2. モデルの読み込み

Agent や Behavior を追加したり、変更したりした場合は、

必ず、World 設定画面の最上部にある モデルの読み込み ボタンを押して 変更された情報 をWorld に反映させます。

8.4.2 World の設定

[旅人グループの設定]

前章に登場させた「旅人グループ」に「立ち寄り行動」を持たせましょう。

Agent グループ設定ウィンドウを開く
 World Composer の画面をスクロールさせ、「Agent グループ」の部分を表示します。
 Agent グループの枠内にある「旅人グループ」を選択し、 変更 ボタンを押してください。
 そうすると Agent グループ設定ウィンドウが開きます。
| Agent | グループ | | |
|-------|--------|-----------------------------|----------|
| | 名前 | | 数 |
| | 旅人グループ | (旅人)Obehaviors,Ogoods,Oinfo | 1 |
| | 村人グループ | (村人)Obehaviors,Ogoods,Oinfo | 5 |
| | | | |
| | | | |
| • | | | |
| | | | |
| | | | |
| | | | |
| | 1 | | |
| | | | 這加(変更)削除 |

2. Agent グループの BehaviorType の選択

「BehaviorType」のところから、それらの Agent グループに持たせる BehaviorType を 選びます。ここでは、「BEHAVIORTYPE_立ち寄り行動」があるので、左のチェックボッ クスにチェックを入れます。

| 🪝 Agentグループの設定 | | | |
|-----------------|--------------------------|--|--|
| 名前: | 旅人グループ | | |
| Agent Type: | AGENTTYPE_旅人 | | |
| Behavior | | | |
| Behavior Type | | | |
| BxVillageMoo | lel#BEHAVIORTYPE_立ち寄り行動 | | |
| - SoxVillageMod | lel#BEHAVIORTYPE_おしゃべり行動 | | |

3. Agent グループの登録

Agent グループウィンドウの OK ボタンを押すと、Agent グループの設定が登録されます。

Agent グループの表のなかに、「旅人グループ」の設定が表示されていることと、「旅人グ ループ」の数が「1」に設定されていることを確認してください。

以上の作業が終わったら、Agent グループの表示が次のようになっていることを確認してください。

確認が終わったら、[Ctrl] + [S]を押し、Worldの設定を保存してください。

| 力平 | EXBB | 130 H |
|--------|-----------------------------|-------|
| 名則 | 「記中月 | 安义 |
| 旅人グループ | (病人)1behaviors,0goods,0info | 1 |
| 村人グループ | (村人)Obehaviors Ogoods Dinfo | 5 |

8.4.3 World のコード生成

1. コード生成の実行

World Composer の最下部にある World の生成 ボタンを押します。 プログレスバーが表示され、World のコード (Java プログラム)が生成されます。

2. ソースコードを保存する
 Ctrl + S を押し、「 BoxVillageWorld.java」を保存します。

8.5 実行·検証



8.5.1 シミュレーションの起動(開発環境上での操作)

1. 実行

ッールバーにある[●]実行ボタンを押します。最近実行したコード (BoxVillage-World.java) が選択・実行されます。

保存されていないファイルが存在すると、実行時に「リソースの保管」ダイアログが 表示されます。保存したいファイルにチェックを入れ、 OK を押してください。

- 8.5.2 シミュレーションの実行 (PlatBox Simulator 上での操作)
 - World の読み込み状態の確認
 PlatBox Simulator の制御パネル上には、読み込まれた World の名前が表示されます。ここでは、「BoxVillageWorld」と表示されていることを確認してください。
 - 2. ビューアの表示

メニューから [ビューア] [Communication Viewer] を選択して、Communication Viewer を開きます。

「立ち寄り行動」を持った「旅人」と、何も行動を持たない「村人」が登場し、「旅人」と一 人の「村人」との間に双方向の「旧友関係」が表示され、「村人」の間には「友人関係」が表 示されたことを確認してください。

シミュレーションの実行
 シミュレーションの実行は、制御パネルの実行ボタンを押して行います。



時間が経過すると、「立ち寄り行動」を持つ旅人 Agent に状態遷移が起こり、「到着状態」から「休息状態」に遷移します。





8.5.3 PlatBox Simulator の終了

1. PlatBox Simulator の終了

PlatBox Simulator を終了するには、メインウィンドウの右上にある × ボタンを押すか、 メニューから [ファイル] [終了] を選択します。

8.6 これまでにつくったモデルの全体像

本章では、旅人が到着してから疲れて寝てしまうまでの状態遷移を作りました。



道案内 この章では、BoxVillage に立ち寄った旅人が到着してから休息するまでの変 化を表す行動をつくりました。次章では、旅人が旧友に土産話をします。久しぶりの再会 なので、積もる話もたくさんあることでしょう。それでは、次に行きましょう!

第9章

コミュニケーションのデザイン

【学習目標】

行動の詳細をアクションブロック図で記述できるようになる(ActionDesiner)

9.1 つくりたい世界

前章で、旅人が村に立ち寄り、休むまでの世界をつくりました。ここでは、旅人が旧友 に土産話をし、村人はその土産話を覚えるという世界をつくります。土産話の内容は、旅 人が遠くの町で得た情報です。

ここでは、以下の流れでモデルのデザインを行っていきます。Behaviorの作成・拡張 を行い、それを元にアクションの作成をします。また、シミュレーションの世界設定を行 う World も、前章のものを拡張します。 【モデルのデザイン】

Behavior のデザイン1(立ち寄り行動)

- 既存の Behavior を開く

- 状態、状態遷移、アクションの描画
- Behavior のデザイン2(おしゃべり行動)

- Behavior の新規作成

- 状態、状態遷移、アクションの描画
- アクションの作成1(立ち寄り行動)
 - 既存のアクションファイルを開く
 - アクションブロック図の作成
 - アクションのコード生成
- アクションの作成2(おしゃべり行動)
 - アクションファイルの新規作成
 - アクションブロック図の作成
 - アクションのコード生成

世界の初期設定

- 既存の World を修正する
- World の設定
- World のコード生成

9.2 Behavior のデザイン1 (立ち寄り行動)



次に、Component Builder の Behavior Designer を用いて、「立ち寄り行動」の状態遷 移図を修正します。前章で作成した状態遷移図では、情報を相手に伝える行動はありませ んでした。ここでは、その行動を追加します。

9.2.1 既存の Behavior を開く

9.2.2 状態、状態遷移、アクションの描画

これから修正する状態遷移図と、そこに登場する状態/アクションは、以下の通りです。



1. 遷移の設定

遷移がどのようなときに起こり、どのような条件を満たしたらどのようなアクションを行うのかを設定するのは、遷移設定ダイアログで行います。
遷移設定ダイアログは、遷移線をダブルクリックして開きます。ここでは、まず

既存の Behavior を開く パッケージ・エクスプローラー上の「^図立ち寄り行動.behavior」をダブルクリックする。
 その Behavior ファイルを読み込んだ状態で、Behavior Designer が起動します。

「回到着状態」から「回休息状態」への遷移をダブルクリックして、遷移設定ダイアログを開

- きます。
- (a)アクションの追加

ここでは、時間が来ると情報を相手に送るようにしたいので、「土産話をする」アクションを追加します。

アクションを追加するには、アクションパネル内の右側にある 新しい内部アクション ボタンを押します。出てきた「内部アクションの作成」ウィンドウで、「名前:」のとこ ろに「土産話をする」と入力し、 OK ボタンを押します。



すると、「現在のアクション」と「利用可能なアクション」のところに、「土産話をする」 と表示されます。



「利用可能なアクション」に表示されているのが、この状態遷移図内で利用できるアク ション、「現在のアクション」に表示されているのが、現在、遷移のプロパティを開い ている遷移において起こるアクションです。

これで必要な設定が終わったので、 OK を押して設定を終了します。

状態遷移図の遷移のところに、「TimeEvent/土産話をする」と表示されたことを確認してください。

ここまでの作業が終わったら、前の状態遷移図と一致していることを確認してくださ い。

確認が終わったら、 Ctrl + S を押し、状態遷移図を保存しましょう。

9.3 Behavior のデザイン2(おしゃべり行動)



相手が土産話をしても、話を聞く方がそれを聞き逃しては意味がありません。ここで は、土産話を知らない状態から、相手からの土産話を聞くことによってその土産話を記憶 し、土産話を知っているという状態に遷移するまでをデザインします。

9.3.1 Behavior の新規作成

- 既存の Model ファイルを開く パッケージ・エクスプローラー上の「^図旅人と村人の行動.model」をダブルクリックしてく ださい。その Model ファイルを読み込んだ状態で、Model Designer が起動します。
- Behavior の新規作成
 Model Designer のキャンバスにある「おしゃべり行動」を右クリックし、表示されたメニューのなかから「Behavior Designer で開く」を選択します。プログレスバーが表示され、Behavior ファイルが生成されます。また、その Behavior ファイルを読み込んだ状態で、Behavior Designer が起動します。
- 生成されたファイルの確認 パッケージ・エクスプローラー上に、「^圓おしゃべり行動.behavior」というファイルができ たことを確認してください。
- 9.3.2 状態、状態遷移、アクションの描画

これから作成する状態遷移図と、そこに登場する状態は、以下の通りです。

| 「おしゃべり行動」の状態遷移図 |
|--------------------------------------|
| 土産話を知らない状態 |
| ChannelEvent/土産話を記憶する 土産話を知っている状態 |

1. 要素の選択と配置:初期状態

Behavior Designer のパレットにある 初期状態 ボタンを押した後、キャンバス(白い 作図領域)をクリックして配置します。

2. 要素の選択と配置:状態

Behavior Designer のパレットにある 🔤 状態 ボタンを押した後、キャンバスをクリック して配置します(ここでは2つの「状態」を配置します)。

3. 名前の変更:状態名

「状態」をダブルクリックして文字を反転させて、「土産話を知らない状態」と「土産話を 知っている状態」にしましょう。

4. 要素の選択と配置:遷移

Behavior Designer のパレットにある → 遷移 ボタンを押して、遷移線の作成モードにします。まず、キャンバス上の 初期状態 をクリックし、その後、「回土産話を知らない状態」をクリックして、遷移線を結びます。

そして、「回土産話を知らない状態」をクリックしてから、もう一方の状態「回土産話を知っている状態」をクリックして、遷移線を結びます。

5. 配置の調整

Behavior Designer のパレットにある 選択 ボタンを押すと、これまで配置した要素を 選択できるようになります。要素の場所を移動したい場合には、要素をドラッグ&ドロップ して移動させることができます。

6. 遷移の設定

遷移がどのようなときに起こり、どのようなガード条件を満たしたらどのよう なアクションを行うのかを設定するのは、遷移設定ダイアログで行います。 遷移設定ダイアログは、遷移線をダブルクリックして開きます。ここでは、まず、「回 土産話を知らない状態」から「回土産話を知っている状態」への遷移をダブルクリックし て、遷移設定ダイアログを開きます。

(a) 受信イベントの設定

この状態遷移は、情報が送られてくると遷移が起きるので、「イベント」のプルダウン で「ChannelEvent」を選択します。「ChannelEvent」とは、Behavior 同士を結ぶ線 で、相手から Information や Goods が送られてくると、状態が遷移する Event のこ とを言います。



(b) アクションの追加

この状態遷移は、情報が送られてくると、その土産話を記憶するようにしたいので、「土 産話を記憶する」アクションを追加します。

アクションを追加するには、アクションパネル内の右側にある 新しい内部アクション ボタンを押します。出てきた「内部アクションの作成」ウィンドウで、「名前:」のとこ ろに「土産話を記憶する」と入力し、 OK ボタンを押します。

すると、現在のアクション と 利用可能なアクション のところに、「土産話を記憶する」 と表示されます。

| (二内部アクションの作成) | × |
|---------------|---|
| 名前: 土産話を記憶する | |
| キャンセル | |

これで必要な設定が終ったので、 OK を押して設定を終了します。

状態遷移図の遷移のところに、「ChannelEvent/土産話を記憶する」と表示されたことを確認してください。

ここまでの作業が終わったら、前の状態遷移図と一致していることを確認してくださ い。

確認が終わったら、 Ctrl + S を押し、状態遷移図を保存しましょう。

9.4 アクションのデザイン1(立ち寄り行動)



Action Designer を用いて、「立ち寄り行動」と「おしゃべり行動」のアクションを、手段の階層構造に分解して記述していきます。まず、「土産話をする(立ち寄り行動)」アクションを作成します。



9.4.1 既存のアクションファイルを開く

- 既存の Behavior ファイルを開く パッケージ・エクスプローラー上の「^回立ち寄り行動.behavior」をダブルクリックしてくだ さい。その Behavior ファイルを読み込んだ状態で、Behavior Designer が起動します。
- アクションファイルの更新 今起動した Behavior Designer のキャンバス上で 右クリック し、表示されたメニューのな かから「Action Designer で開く」を選択します。 プログレスバーが表示され、Action ファイルが更新されます。また、そのアクションファ イルを読み込んだ状態で、Action Designer が起動します。

9.4.2 アクションブロック図の作成

「土産話をする」アクション

「土産話をする」アクションは、以下の階層構造が示すように、土産話を作り、旧友関 係の村人全員に土産話を送る、という2段階から構成されます。

| [立ち寄り行動] |
|----------------------|
| 土産話をする |
| (1) 土産話をつくる |
| (2) 旧友関係の村人全員に土産話をする |
| |
| 1. 工産話をしくる |
| (a)土産話の内容を決定する |
| (b) 土産話をつくる |
| 2. 旧友関係の村人全員に土産話をする |
| (a) 旧友関係の村人全員に土産話をする |

これらの処理では、アクションパーツとして用意されている「StringInformation をつ くる」と「指定した RelationType の Relation を持つ Agent 全員に Information を送る」 を用います。

1. 土産話をつくる

「土産話をつくる」には、土産話の内容を決定し、土産話をつくるという処理になります。 (a) 土産話の内容を決定する

Action Designer のパレットにある 文字列 ボタンを押した後、「土産話をする」ア クションの内部をクリックします。

| 😢 条件11線り返し | |
|--------------|------------|
| 嘉文 🖈 | |
| 🏗 アクションパーツ | |
| ➡ 操作の呼び出し | |
| 🔜 変数宣言/代入 | erminate |
| 22 四則演算 | |
| [1] 論理演算 | |
| 55 文字列 | |
| Preturn文 | |
| 🔰 break文 🚽 👻 | ✓ ● 土産話をする |
| | |
| | |
| | |
| | |

「文字列を編集する」ダイアログが表示されるので、上部の文章入力欄(式:)に「"伝 説のメロンパンを見つけたよ!"」と入力してください。そして、「変数を作成する:」の 右に変数名として「情報の内容」と入力して OK ボタンを押します。

| 🗲 文字列を編集する | × |
|----------------------|----------|
| て: "伝説のメロンバンを見つけたよ!" | |
| 利用可能な変数 | _ |
| | |
| ■ 変数を作成する 情報の内容 | |
| | OK キャンセル |

ダイアログが閉じ、String 型で「情報の内容」という変数が宣言され、具体的な内容と

して「"伝説のメロンパンを見つけたよ!"」が入ります。

(b) 土産話をつくる

Action Designer のパレットにある アクションパーツ ボタンを押した後、「土産話を する」アクションの内部 (前述 (a) の変数宣言を行った場所の下) をクリックします。

| □□ 文 × | |
|------------|----------------------|
| ▶ 操作の呼び出し | |
| 🛃 変数宣言/代入 | erminate |
| 22 四則演算 | |
| 148 論理演算 | |
| 188 文字列 | |
| ┛ return文 | |
| 🖄 break文 👻 | ● 土産話をする |
| | "伝説のメロンパンを見つけたよ!" |
| | └── 情報の内容 (String ▼) |
| | Ļ |

「アクションパーツの選択」ダイアログが表示されるので、上部画像あるいは「カテゴ リ:」から「Informationの生成/操作」を選択してください。

「利用できるアクションパーツ」リストから「StringInformation をつくる」を選択して、 OK ボタンを押します。

| 🧺 アクションパーツの選択 | × |
|--|--|
| 自分と他のAgentの間のやりとり 自分が持つInformation | 。 他のAgentが持つInformation |
| 自分が持つBehavior 自分が持つBehavior | 他のAgentが持つBehavior |
| 自分が持つGoods | 他のAgentが持つGoods |
| ◎ Informationの生成/提作 ◎ Goodsの生成/操作 ◎ 集合操作 | □ 出力 ● 計算 |
| カテゴリ: Informationの生成/操作 利田で考えつかション・パミーンル | T |
| Time CaSP クラムハーン。 RelationInformationに記憶されているRelationのTypeを取得する RelationInformationに記憶されているRelationを取得する RelationInformationを入る SetInformationの内容をセットとして取得する Strate Informationの描述取得する | A |
| and Fanomation 2007 Table Information 21行分のInformationを増入する Table Information 21行分のInformationを追加する Table Information Cinformationを追加する Table Information Cinformationを追加する Table Information Cinformationを取得する Table Information Cinformationを主要まする | × |
| 最近利用したアクションパーツ | |
| | |
| | <u> の K キャンセル </u> |

ダイアログが閉じ、「StringInformation をつくる」アクションパーツが配置されます。 配置されたアクションパーツのなかで、「StringInformation をつくる」をダブルクリッ クし、文字を反転させ、「土産話をつくる」に、そして同様に「新しい StringInformation」 を「遠くの町の情報」にそれぞれ変更してください。

| 歴生産話をつくる | |
|---|--------------------|
| 情報の内容 | |
| を値とするStringInformationをつくる | |
| 「「」「「」」「「」」「」」「「」」「」」「「」」「」」「」」「」」「」」「」 | (StringInformation |
| | |

2. 旧友関係の村人全員に土産話をする

Action Designer のパレットにある アクションパーツ ボタンを押した後、「土産話をする」 アクションの内部 (前述 (b) のアクションパーツ配置を行った場所の下) をクリックします。 「アクションパーツの選択」ダイアログが表示されるので、上部画像あるいは「カテゴリ:」 から「自分と他の Agent の間のやりとり」を選択してください。

「利用できるアクションパーツ」リストから「指定した RelationType の Relation を持つ Agent 全員に Information を送る」を選択して、 OK ボタンを押します。



ダイアログが閉じ、「指定した Relation Type の Relation を持つ Agent 全員に Information を送る」アクションパーツが配置されます。

配置されたアクションパーツにおける「指定した RelationType の Relation を持つ Agent 全員に Information を送る」を「旧友関係の村人全員に土産話をする」に名前 を変更し、最初の「null」と表示されている部分を「<Type:BoxVillageModel> 旧友関 係」、次の「null」を「<Type:BoxVillageModel> おしゃべり行動」、最後の「null」を 「<Type:BoxVillageModel> 土産話」、そして新しい StringInformation を「遠くの町の情 報」となるようにそれぞれプルダウンから選択してください。

| 四方関係の村人全員に土産話をする | |
|--|-------------|
| KType:BoxVillageModel>旧友関係 | • |
| を通じて相手のAgentの | |
| (Type:BoxVillageModel)おしゃべり行動 | • (2 |
| <type:boxvillagemodel>土産話></type:boxvillagemodel> | ▼として |
| 遠くの町の情報 | ▼を送る |
| └→ 情報を送った人数 (jint | •) |
| | |
| l | |

ここまでの作業が終わったら、作成されたアクションが次のものと一致していることを 確認してください。

確認が終わったら、 Ctrl + S を押し、アクションブロック図を保存しましょう。

| Logger.getLogger("boxvillage.立ち寄り行動") | |
|---|---|
| logger (Logger) | |
| initialize | |
| terminate | |
| | |
| | |
| ■ 土産話をする ″伝説のメロンパンを見つけたよ!″ | 立ち寄り行動 旅人 |
| ➡ 情報の内容 (String) | ▲ 國 文字列 |
| 健 土産話をつくる 信報の内容 | 土産話をつくる 立ち寄り行動 |
| を値とするStringInformationをつくる う 遠くの町の情報 (StringInformation) | 旅人 「ステ列I 「ステ列I 「長田のメロン 「と長りづけ たよ! " 」 「ま成する 」 遠くの 町の信報 町の信報 生成する 」 」 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 、 |
| | ▲ 🕅 StringInformation をつくる |
| 1120日友関係の村人全員に土産話をする | 旧友関係の村人全員に土産話をする |
| < I ype:BoxVillageModel>旧友関係 | 村人 おしゃべり行動 |
| を通じて拍子のAgentの 〈Tvpe:BoxVillageModel〉おしゃべり行動 | 8< |
| | 旧友関係 「「」 違くの 町の情報 |
| | 旅人 立ち寄り行動 ノ 土産話として 送る |
| ▶ 情報を送った人数 (int) | |

▲ st 指定した RelationTypeの Relation を 持つ Agent 全員に Information を送る

9.4.3 アクションのコード生成

- ソースコード生成の実行
 「¹ ジロケンゴード生成の実行
 「¹ ジロケンゴード(Java プロゲラム)が生成
 ます。プロゲレスバーが表示され、アクションのソースコード(Java プログラム)が生成
 されます。
- 2. ソースコードを保存する
 Ctrl + S を押し、ソースコードを保存します。

9.5 アクションのデザイン2(おしゃべり行動)



続いて、「土産話を記憶する(おしゃべり行動)」アクションを作成することにします。

[おしゃべり行動] 土産話を記憶する (1) 聞いた土産話を記憶する

9.5.1 アクションファイルの新規作成

- 既存の Behavior ファイルを開く パッケージ・エクスプローラー上の「^図おしゃべり行動.behavior」をダブルクリックしてく ださい。その Behavior ファイルを読み込んだ状態で、Behavior Designer が起動します。
- 2. アクションファイルの新規作成

今起動した Behavior Designer のキャンバス上で <u>右クリック</u>し、表示されたメニューのな かから「Action Designer で開く」を選択します。

プログレスバーが表示され、Action ファイルが生成されます。また、そのアクションファ イルを読み込んだ状態で、Action Designer が起動します。

3. 生成されたファイルの確認

パッケージ・エクスプローラー上に、「☑Abstract おしゃべり行動.action」と「臀おしゃべ り行動.action」というファイルができたことを確認してください。



9.5.2 アクションブロック図の作成

「土産話を記憶する」アクション

「土産話を記憶する」アクションは、すでにアクションパーツで用意されています。この処理では、アクションパーツとして用意されている「最後に受け取った Information を記憶する」を用います。

土産話を記憶する

Action Designer のパレットにある アクションパーツ ボタンを押した後、「土産話を記憶 する」アクションの内部をクリックします。

「アクションパーツの選択」ダイアログが表示されるので、上部画像あるいは「カテゴリ:」 から「自分と他の Agent の間のやりとり」を選択してください。

「利用できるアクションパーツ」リストから「最後に受け取った Information を記憶する」 を選択して、 OK ボタンを押します。



ダイアログが閉じ、「最後に受け取った Information を記憶する」アクションパーツが配置 されます。

配置されたアクションパーツにおける、「最後に受け取った Information を記憶する」を 「聞いた土産話を記憶する」に、「受け取った Information」を「聞いた土産話」にそれぞれ 名前を変更し、「null」と表示されている部分を「<Type:BoxVillageModel> 土産話」とな るようにプルダウンから選択してください。

| | 土産話を記憶する 一門間いた土産話を記憶する |
|---|---|
| | 受け取ったInformationを |
| < | ≺Type:BoxVillageMode▷土産話 |
| | として記憶する |
| | → 間いた土産話 (Information マ) |
| | |
| | |

ここまでの作業が終わったら、作成されたアクションが次のものと一致していることを 確認してください。

| Logger.getLogger("boxvillage.おしゃべり行動") | |
|--|---------------------------|
| logger (Logger) | |
| initialize | |
| terminate | |
| ▲ 土産話を記憶する | |
| 聞いた土産話を記憶する 受け取ったInformationを | 開いた土産話を記憶する 土産話として記憶する |
| ↓ 聞いた土産話 (Information) | |

確認が終わったら、 Ctrl + S を押し、アクションブロック図を保存しましょう。

9.5.3 アクションコードの生成

1. ソースコード生成の実行

「^図おしゃべり行動.action」のキャンバスを右クリックして「ソースコード生成」を選択し ます。プログレスバーが表示され、Behaviorのソースコード(Java プログラム)が生成さ れます。

2. 生成されたファイルの確認

パッケージ・エクスプローラー上に、「IJおしゃべり行動.java」というファイルが表示され

ていることを確認してください。



これが、いま生成されたファイルです。

3. ソースコードを保存する

ファイルの確認が終わったら、 Ctrl + S を押し、ソースコードを保存します。

9.6 世界の初期設定



Behavior、アクションのデザインが終わったら、世界の設定をします。どのようなシ ミュレーションを行いたいのかは、以下の通りです。



9.6.1 既存の World を修正する

1. 既存の World を開く

パッケージ・エクスプローラー上の「愛BoxVillageWorld.world」をダブルクリックする。 この World ファイルは前章で作成した World ファイルです。

モデルの読み込み
 Agent や Behavior を追加したり、変更したりした場合は、
 必ず、World 設定画面の最上部にある モデルの読み込み ボタンを押して 変更された情報
 を World に反映させます。

9.6.2 World の設定

[村人グループの設定]

前章では、旅人グループと村人グループを世界に登場させました。今回は、その情報を 受け取り記憶する村人も世界に登場させ、上のやりとりをさせてみましょう。

1. Agent グループ設定ウィンドウを開く

World Composer の画面をスクロールさせ、「Agent グループ」の部分を表示します。

Agent グループの枠内にある「村人グループ」を選択し、 を更 ボタンを押してください。 そうすると Agent グループ設定ウィンドウが開きます。

2. Agent グループの BehaviorType の選択

「BehaviorType」のところから、それらの Agent グループに持たせる BehaviorType を 選びます。ここでは、村人グループは情報を受け取って記憶するという行動をしなければな らないので、「BEHAVIORTYPE_おしゃべり行動」のチェックボックスにチェックを入れ ます。(*「BEHAVIORTYPE_立ち寄り行動」のチェックボックスにはチェックを入れな いでください。)



3. Agent **グループの登録**

Agent グループウィンドウの OK ボタンを押すと、Agent グループの設定が登録されま す。

Agent グループの表のなかに、「村人グループ」の設定が表示されていることと、「村人グ ループ」の数が「5」に設定されていることを確認してください。

以上の作業が終わったら、Agent グループの表示が次のようになっていることを確認してください。

| gentグループ | | |
|----------|-----------------------------|---|
| 名前 | 説明 | 费 |
| 旅人グループ | (旅人)1behaviors,0goods,0info | 1 |
| 村人グループ | (村人)1behaviors,0goods,0info | 5 |
| | | |

確認が終わったら、|Ctrl| + |S|を押し、でWorldの設定を保存してください。

- 9.6.3 World のコード生成
 - コード生成の実行

World Composer の最下部にある World の生成 ボタンを押します。

プログレスバーが表示され、World のコード (Java プログラム)が生成されます。

2. ソースコードを保存する

Ctrl + S を押し、「 BoxVillageWorld.java」を保存します。

9.7 実行・検証



9.7.1 シミュレーションの起動(開発環境上での操作)

1. 実行

ツールバーにある[●]実行ボタンを押します。最近実行したコード (BoxVillage-World.java) が選択・実行されます。

保存されていないファイルが存在すると、実行時に「リソースの保管」ダイアログが 表示されます。保存したいファイルにチェックを入れ、 OK を押してください。

9.7.2 シミュレーションの実行 (PlatBox Simulator 上での操作)

1. World の読み込み状態の確認

PlatBox Simulator の制御パネル上には、読み込まれた World の名前が表示されます。ここでは、「BoxVillageWorld」と表示されていることを確認してください。

ビューアの表示
 メニューから [ビューア] [Communication Viewer] を選択して、Communication Viewer
 を開きます。

「立ち寄り行動」を持った「旅人」と、「おしゃべり行動」を持った「村人」が登場し、「旅 人」と一人の「村人」との間に双方向の「旧友関係」が表示され、「村人」の間には「友人関 係」が表示されたことを確認してください。

3. シミュレーションの実行

シミュレーションの実行は、制御パネルの実行ボタンを押して行います。時間が経過する と、旅人は「到着状態」から状態が遷移し、「土産話をする」というアクションを起こしま す。そして「伝説のメロンパンを見つけたよ!」という吹き出しが、旧友関係の結ばれた村 人へと移動します。それを受け取った村人は、「土産話を知らない状態」から「土産話を知っ ている状態」へと遷移し、その情報を記憶します。





9.7.3 PlatBox Simulator の終了

1. PlatBox Simulator の終了

PlatBox Simulator を終了するには、メインウィンドウの右上にある × ボタンを押すか、 メニューから [ファイル] [終了] を選択します。

9.8 これまでにつくったモデルの全体像

本章では、アクションブロック図で、旅人が旧友の村人に土産話をし、旧友の村人がそ れを記憶するまでのアクションを作りました。「土産話をする」と一言で表わされている アクションが、いくつものパーツから構成されているということがわかりましたね。



道案内 この章では、ついに村人が旧友に土産話をし始めました。さて、旅人の土産 話に驚いた村人は、BoxVillage にいる友人の村人にその話をしたくてたまらなくなって います。次章では、村人が土産話を広めるアクションをつくりましょう。

第10章

情報伝播のデザイン

【学習目標】

行動の詳細をアクションブロック図で記述できるようになる(ActionDesiner)

10.1 つくりたい世界

前章では、旅人が村人に土産話をし、旧友である村人はその話を覚えるという部分まで をつくりました。ここでは、話を聞いた村人が、覚えた土産話を他の村人に広めるという アクションを加えてみたいと思います。

ここでは、以下の流れでモデルのデザインを行っていきます。前章で作成した「おしゃ べり行動」の Behavior を拡張することにします。また、シミュレーションの世界設定を 行う World も、前章のものを拡張します。 【モデルのデザイン】 Behavior のデザイン(おしゃべり行動) - 既存の Behavior を開く - 状態、状態遷移、アクションの描画 アクションの修正(おしゃべり行動) - 既存のアクションファイルを開く - アクションブロック図の修正 - アクションのコード生成 既存の World を修正する - 既存の World を利用する - World の設定 - World のコード生成

10.2 Behavior のデザイン(おしゃべり行動)



次に、Component Builder の Behavior Designer を用いて、「おしゃべり行動」の状態 遷移図を修正します。前章で作成した状態遷移図では、土産話を他の村人に伝える行動は ありませんでした。ここでは、その行動を追加します。

10.2.1 既存の Behavior を開く

10.2.2 状態、状態遷移、アクションの描画

状態遷移図に変更点を加えていきます。これから作成する状態遷移図と、そこに登場す る状態は、以下の通りです。



1. 要素の選択と配置: 遷移

Behavior Designer のパレットにある >右側の自己遷移 ボタンを押して、自己遷移の作 成モードにします。キャンバス上の状態「土産話を知っている状態」をクリックすると、そ の状態からその状態への自己遷移の線が結べます。

2. 配置の調整

Behavior Designer のパレットにある │ [▷] 選択 │ ボタンを押し、これまでと同様に図を見

既存の Behavior を開く パッケージ・エクスプローラー上の「^図おしゃべり行動.behavior」をダブルクリックする。
 その Behavior ファイルを読み込んだ状態で、Behavior Designer が起動します。

やすい位置にしましょう。

3. 遷移の設定

「□土産話を知っている状態」から出て自分の状態へ戻ってくる遷移線をダブルクリックし

て、遷移設定ダイアログを開きます。

(a) 受信イベントの設定

この状態遷移は、時間が経つと遷移が起きるので、「イベント」のプルダウンで「TimeEvent」を選択します。

(b) アクションの追加

この状態遷移は、情報を記憶した後、自分と友人関係を結んでいるすべての村人にその 情報を送るようにしたいので、「土産話を広める」アクションを追加します。

新しい内部アクション」ボタンを押し、「内部アクションの作成」ウィンドウで、「名前:」のところに「土産話を広める」と入力し、 OK ボタンを押します。

| 🧲 内部ア | クションの作成 | × |
|-------|----------|---|
| 名前: | 土産話を広める | |
| | () キャンセル | |

すると、現在のアクション と 利用可能なアクション のところに、「土産話を広める」 と表示されます。

これで必要な設定が終ったので、 OK を押して設定を終了します。

状態遷移図の遷移のところに、「TimeEvent/土産話を広める」と表示されたことを確認して ください。

ここまでの作業が終わったら、前の状態遷移図と一致していることを確認してくださ い。

確認が終わったら、Ctrl + S を押し、状態遷移図を保存しましょう。

10.3 アクションのデザイン(おしゃべり行動)



Action Designer を用いて、「おしゃべり行動」のアクションを修正していきます。

【おしゃべり行動】 • 土産話を記憶する - 聞いた土産話を記憶する • 土産話を広める - 自分が持つ土産話を思い出す - 友人関係の村人全員に土産話を広める

10.3.1 既存のアクションファイルを開く

 既存の Behavior ファイルを開く パッケージ・エクスプローラー上の「^図おしゃべり行動.behavior」をダブルクリックしてく ださい。その Behavior ファイルを読み込んだ状態で、Behavior Designer が起動します。

 アクションファイルの更新 今起動した Behavior Designer のキャンバス上で 右クリック し、表示されたメニューのな かから「Action Designer で開く」を選択します。 プログレスバーが表示され、アクションファイルが更新されます。また、そのアクション ファイルを読み込んだ状態で、Action Designer が起動します。

■ ここで、状態遷移図を修正した「®おしゃべり行動.behavior」を Action Designer で開くと、パッケージ・エクスプローラ内の「[®]おしゃべり行動.java」に「×」印が付き ます。これは、前章で作ったアクションファイルの内容と違うために起きるエラーなので すが、今はアクションファイルの中身を更新している最中なので、気にせず、次に進みま しょう。 10.3.2 アクションブロック図の作成

「土産話を広める」アクション

「土産話を広める」アクションは、以下の階層構造が示すように、自分が持つ土産話を 思い出し、友人関係を持つ全員に土産話を広める、という2段階から構成されます。

| | おしゃべり行動 |
|---|-----------------------|
| | 土産話を広める |
| | (1) 自分が持つ土産話を思い出す |
| | (2) 友人関係の村人全員に土産話を広める |
| | 1. 自分が持つ土産話を思い出す |
| | (a)自分が持つ土産話を思い出す |
| | 2. 友人関係を持つ全員に土産話を広める |
| I | (a)友人関係の村人全員に土産話を広める |

これらの処理では、アクションパーツとして用意されている「この Agent が持つ Information を取得する」と「指定した RelationType の Relation を持つ Agent 全員に Information を送る」を用います。

1. 自分が持つ土産話を思い出す

Action Designer のパレットにある アクションパーツ ボタンを押した後、「土産話を広める」アクションの内部をクリックします。

「アクションパーツの選択」ダイアログが表示されるので、上部画像あるいは「カテゴリ:」 から「自分が持つ Information」を選択してください。

「利用できるアクションパーツ」リストから「この Agent が持つ Information を取得する」 アクションパーツを選択して、 OK ボタンを押します。

ダイアログが閉じ、「この Agent が持つ Information を取得する」アクションパーツが配置 されます。

配置されたアクションパーツのなかで、「この Agent が持つ Information を取得する」をダ ブルクリックし、文字を反転させ、「自分が持つ土産話を思い出す」に名前を変更し、最初 の「null」と表示されている部分を「<Type:BoxVillageModel> 土産話」となるようにプ ルダウンから選択します。最後に、「取得した Information」を最初と同じように「思い出し た土産話」に名前を変更してください。

| | 主産話を広める 1000000000000000000000000000000000000 |
|---|--|
| | このAgentが持つ |
| < | <type:boxvillagemodel>土産話</type:boxvillagemodel> |
| | を取得する |
| | した土産話 (Information) |
| | |
| | |

2. 友人関係を持つ全員に土産話を送る

Action Designer のパレットにある アクションパーツ ボタンを押した後、「土産話を広める」アクションの内部 (前述 (a) のアクションパーツ配置を行った場所の下) をクリックします。

「アクションパーツの選択」ダイアログが表示されるので、上部画像あるいは「カテゴリ:」 から「自分と他の Agent の間のやりとり」を選択してください。

「利用できるアクションパーツ」リストから「指定した RelationType の Relation を持つ Agent 全員に Information を送る」を選択して、 OK ボタンを押します。

ダイアログが閉じ、「指定した Relation Type の Relation を持つ Agent 全員に Information を送る」アクションパーツが配置されます。

配置されたアクションパーツにおける「指定した RelationType の Relation を持つ Agent 全員に Information を送る」を「友人関係の村人全員に土産話を広める」に名 前を変更し、最初の「null」と表示されている部分を「<Type:BoxVillageModel> 友人 関係」、次の「null」を「<Type:BoxVillageModel> おしゃべり行動」、最後の「null」を 「<Type:BoxVillageModel> 土産話」となるようにそれぞれプルダウンから選択してくだ さい。

| 図 友人関係の村人全員に土産話を広める |
|--|
| ⟨Type:BoxVillageModel>友人關係 |
| を通じて相手のAgentの |
| <type:boxvillagemodel>おしゃべり行動</type:boxvillagemodel> |
| (Type:BoxVillageModel)土産話 |
| 国い出した土産話 |
| |
| |

ここまでの作業が終わったら、作成されたアクションが次のものと一致していることを 確認してください。

| Logger.getLogger("boxvillage.おしゃべり行動") | |
|--|---|
| logger (Logger) | |
| initialize | |
| terminate | |
| ・ ・ | 聞いた土産話を記憶する |
| 受け取ったInformationを | 土産話として記憶する サム 受け取った情報 受け取った情報 受け取った Information を記憶する |
| ● 土産話を広める | 自分が持つ土産話を思い出す |
| 🖭 自分が持つ土産話を思い出す | 取得する |
| このAgentが持つ | |
| 思い出した土産話 (Information) | |
| (1) 友人関係の村人全員に土産話を広める | 友人関係の村人全員に土産話を広める |
| <type:boxvillagemodel>友人関係</type:boxvillagemodel> | 村人 おしゃべり行動 |
| ーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーーー | 8- |
| <type:boxvillagemodel>おしゃべり行動</type:boxvillagemodel> | 友人関係 生産話として |
| <type:boxvillagemodel>土産話 として</type:boxvillagemodel> | 村人おしゃべり行動 ノ送る |
| 思い出した土産話 を送る 情報を送った人数 (int) | |
| | └────── 作 熈 指定した RelationType の Relation を 持つ Agent 全員に Information を送る |

確認が終わったら、Ctrl + S を押し、アクションブロック図を保存しましょう。

10.3.3 アクションのコード生成

- ソースコード生成の実行
 「響おしゃべり行動.action」のキャンバスを右クリックして「ソースコード生成」を選択し ます。プログレスバーが表示され、アクションのソースコード(Java プログラム)が生成 されます。
- 2. ソースコードを保存する
 Ctrl + S を押し、ソースコードを保存します。

10.4 実行・検証



World の設定は前章のままでよいので、実行してみましょう。

10.4.1 シミュレーションの起動(開発環境上での操作)

1. 実行

ツールバーにある シード (BoxVillage-World.java) が選択・実行されます。

保存されていないファイルが存在すると、実行時に「リソースの保管」ダイアログが 表示されます。保存したいファイルにチェックを入れ、 OK を押してください。

10.4.2 シミュレーションの実行 (PlatBox Simulator 上での操作)

1. World の読み込み状態の確認

PlatBox Simulator の制御パネル上には、読み込まれた World の名前が表示されます。ここでは、「BoxVillageWorld」と表示されていることを確認してください。

2. ビューアの表示

メニューから [ビューア] [Communication Viewer] を選択して、Communication Viewer を開きます。

「立ち寄り行動」を持った「旅人」と、「おしゃべり行動」を持った「村人」が登場し、「旅 人」と一人の「村人」との間に双方向の「旧友関係」が表示され、「村人」の間には「友人関 係」が表示されたことを確認してください。

3. シミュレーションの実行

シミュレーションを実行します。

時間が経過すると、旅人から「伝説のメロンパンを見つけたよ!」という情報が旧友関係の 結ばれた村人へと送られます。受け取った村人は、自分が友人関係を持つ村人全員に、その 「伝説のメロンパンを見つけたよ!」という情報を送ります。情報を受け取り、「土産話を 知っている状態」に遷移した村人は、時間が来るたびに自分と友人関係を持つ村人全員に情
報を送ることを繰り返します。



10.4.3 PlatBox Simulator の終了

PlatBox Simulator の終了
 PlatBox Simulator を終了するには、メインウィンドウの右上にある × ボタンを押すか、
 メニューから [ファイル] [終了] を選択します。

10.5 これまでにつくったモデルの全体像

本章では、アクションブロック図で村人が友人の村人に土産話を広めるアクションを作 りました。



道案内村人は、旅人が話してくれた土産話を自分の友人に話します。そのために、 どのような状態遷移、アクションを追加すればよいのかわかりましたか?そして、旅人の 土産話はちゃんと村人たちに伝わっていきましたか?次章では、もっと村人の数を増やし て、あっという間に村全体に土産話が広まっていく様子を見てみましょう。

第11章

シミュレーション結果の分析

【学習目標】

データを収集できるようになる(DataCollector)

グラフを表示できるようになる (GraphComponent Manager)

11.1 つくりたい世界

旅人が土産話をし、それを村人が広めるという世界はできましたが、本当にそれが皆に 広まっていっているのでしょうか。また、それはどのくらいの早さで広まっているので しょうか。村人を100人登場させ、村全体の様子を見てみることにしましょう。

本章では、前章までに作成したモデルを使い、世界の初期設定に変更を加えます。そ して、シミュレーション上のデータを PlatBox Simulator で収集し、グラフで表してみ ます。



11.2 世界の初期設定

まず世界の設定をします。どのようなシミュレーションを行いたいのかは、以下の通り です。



11.2.1 World の設定

前章までの設定では、村人グループは5人でした。ですが、BoxVillage に住む村人が 全員で5人というのでは少なすぎますよね。ここでは、村人を100人にしてシミュレー ションを実行してみましょう。

[村人グループの設定]

- Agent グループの値設定ウィンドウを開く
 Agent グループの表の中にある、「村人グループ」の行の「数」の列のセルをクリックする
 と、そのセルの右端に編集ボタンが表示されます。この編集ボタンを押すと値設定ウィンドウが表示されます。
- 2. 数の入力

村人グループの数を「100」人にすることにしましょう。

| 叠值 | | | × |
|-----|------------|----------|----------|
| 型: | int | T | |
| (直: | ◎ 値の直接入力 | | (100) |
| | ○乱数の幅の入力 | 最小値(以上) | \sim |
| | | 最大値(未満) | |
| | | 乱数ジェネレータ | Y |
| | ○ パラメータの選択 | | V |
| | | | |
| | | | |
| | | | |
| | | | OK キャンセル |
| | | | |

3. Agent グループの数の確認

値設定ウィンドウの OK を押すと、「村人グループ」の設定が登録されます。「村人グルー プ」の数が「100」となったことを確認してください。

以上の作業が終わったら、Agent グループの表示が次のようになっていることを確認してください。

| Agent | ブループ ――― | | |
|-------|----------|-----------------------------|-----|
| | 名前 | 記印 | 数 |
| | 旅人グループ | (旅人)1behaviors,0goods,0info | 1 |
| | 村人グループ | (村人)1behaviors,0goods,0info | 100 |

確認が終わったら、Ctrl + S コマンドでWorld の設定を保存してください。

11.2.2 World のコード生成

1. コード生成の実行

World Composer の最下部にある | World の生成 | ボタンを押します。

ソースコードを保存する
 Ctrl + S を押し、「 BoxVillageWorld.java」を保存します。

11.3 実行・検証

11.3.1 シミュレーションの起動(開発環境上での操作)

1. 実行

ツールバーにある[●]実行ボタンを押します。最近実行したコード (BoxVillage-World.java) が選択・実行されます。

11.3.2 シミュレーションの実行 (PlatBox Simulator 上での操作)

今回は、村人グループの数が多いので、Communication Viewer は使いません。代わり に、現在何人の村人が土産話を知っているかをグラフで表示してみることにします。グラ フで表示するには、PlatBox Simulator の DataCollection Manager と GraphComponent Manager を使います。

DataCollection Manager の設定

DataCollection Manager では、何人の村人が土産話を知っているか、という値をその まま取得できるわけではありません。まず、BoxVillage に存在するすべてのエージェン トを取得し、その各エージェントが土産話を持っているかを調べ、持っているエージェン トのみを抽出し、抽出したエージェントの数を集計する、という様に、目的に合わせて取 得したい情報を絞り込んでいく必要があるのです。 1. DataCollection Manager を開く

メニューから [ツール] [DataCollection Manager] を選択して、DataCollection Manager を開きます。



2. すべてのエージェントの取得 「作成」ボタンを押すと、DataCollection の作成ダイアログが表示されます。



- 「名前:」には「すべてのエージェント」と入力します。
- 「Collection タイプ:」に「すべてのエージェント」を選択します。

| DataCo | llectionの作成 | | X |
|--------|---------------|-------------------|----------|
| | | DataCollectionの作成 | |
| | 名前: | オペてのエージェント | |
| | Collectorタイプ: | オペてのエージェント | * |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| 1 | | 了解し取消し | |

選択できたら 了解 ボタンを押してください。DataCollection リストに「すべてのエー ジェント」という行が追加されます。

| DataCollection Manager | _ _ × |
|------------------------|--------------|
| オペてのエージェント | 作成 |
| | 編集 |
| | 編年 |

 エージェントの持つ土産話 エージェントが持つ Information を取得します。 作成 ボタンを押し、「名前:」に「エー ジェントの持つ土産話」と入力します。 「Collection タイプ:」に「オブジェクト操作-操作」 「ソース:」に「すべてのエージェント」 「対象列:」に「Agent」

「操作:」に「getInformation(Information)」を選択します。 引数を設定するには、「引数 [0]:」の [....] ボタンを押します。

| DataCollectionの作成 | | X |
|-------------------|-----------------------------|---|
| | DataCollectionの作成 | |
| 名前: | エージェントの持つ土産話 | 1 |
| Collectorタイプ: | オフジェクト操作 - 操作 | 1 |
| ソース: | すべてのエージェント | I |
| 対象列: | Agent | 1 |
| 操作: | getInformation(Information) | 1 |
| | 51数[0]: null | |
| | | |
| | 「ア解」「取消し」 | |

「InformationType」にチェックを入れて、 Next> を押します。

| 3 | × |
|--|---------------------|
| | AgentType |
| | GoodsType |
| | formationType |
| | RelationType |
| | BehaviorType |
| <back< td=""><td>Next> Finish Cancel</td></back<> | Next> Finish Cancel |

リストから、「InformationType-"boxvillage 土産話"」を選択し、 Finish を押すと引数 が設定されます。



すべて設定できたら、DataCollectionの作成ダイアログで | 了解 | を押します。

4. 土産話を知っているエージェント

作成 ボタンを押し、「名前:」に「土産話を知っているエージェント」と入力します。

- 「Collection タイプ:」に「条件抽出」
- 「ソース:」に「エージェントの持つ土産話」

「対象列:」に「getInformation」を選択し、 「抽出条件:」で 抽出条件の設定 ボタンを押します。

| DataCol | lectionの作成 | | × |
|---------|---------------|--|---|
| | | DataCollectionの作成 | _ |
| | 名前: (| 土産話を知っているエージェント | |
| | Collectorタイプ: | 「「「「「」」(「」」(「」)」(「」)(「」)(「」)(「」)(「」)(「」) | |
| | ソース: | エージェントの持つ土産話 | |
| | 対象列: | getinformation | |
| | 抽出条件 | 文字列が "条件" と完全一致する | |
| | | 抽出条件の設定 | |
| 1 | | 了解 取消し | |

抽出条件の設定ダイアログで、オブジェクトの比較にチェックを入れ、「null でない」を選択し、「了解」を押します。

| 抽出条件の設定 | × |
|------------|---|
| □ 文字列の比較 | |
| 条件 と完全一致す▼ | |
| □ 数値の比較 | |
| 各值 = | |
| マナブジェクトの比較 | |
| nullでない 🗸 | |
| 「育解」取消し | |

抽出条件が設定できたら、DataCollectionの作成ダイアログで 了解 を押します。

5. 土産話を知っているエージェントの数

作成 ボタンを押し、「名前:」に「土産話を知っているエージェントの数」と入力します。 「Collection タイプ:」に「集計-要素数」

- 「ソース:」に「土産話を知っているエージェント」
- 「対象列:」に「Agent」を選択し、「了解」を押します。

| DataCollectionの作成 | | X |
|-------------------|-------------------|---|
| | DataCollectionの作成 | |
| 名前: | 土産話を知っているエージェントの数 | |
| Collectorタイプ | 集計 - 要素数 | |
| ソース: | 土産詰を知っているエージェント | |
| 対象列: | Agent | |
| | | |
| | | |
| 1 | | |
| | 了解 取消し | |

 DataCollection の確認 以上の作業が終わったら、DataCollection のリストが次のようになっていることを確認し てください。

| PataCollection Manager | _ D × |
|------------------------|---------|
| すべてのエージェント | 作成 |
| エージェントの持つ土産話 | |
| 土産話を知っているエージェント | 編集 |
| 土産話を知っているエージェントの数 | 問告 |
| | 1711155 |

作った DataCollection の設定は、毎回作る必要がないように、保存しておくことができます。DataCollection Manager の保存ボタンを押すと、ファイルを保存する場所と、保存する DataCollection の内容を指定できます。では、実際に今回作ったデータをBoxVillage フォルダに保存してみてください。 保存したデータを読み込むには、読込ボタンを押して、保存した.dc ファイルを選択してください。

GraphComponent Manager の設定

 GraphComponent Manager の起動 メニューから [ツール] [GraphComponent Manager] を選択して、GraphComponent Manager を開きます。



2. 新しいグラフの設定をする

 作成 ボタンを押し、GraphComponent Editor を開きます。

 「名前:」には「土産話を知っているエージェント数のグラフ」と入力し、 選択 ボタンを
 押します。

| GraphComponent Editor | × |
|---------------------------|----|
| 名 11 土産話を知っているエージェント数のグラフ | |
| 「親データー | |
| Data Collection: | |
| D | 選択 |
| | |
| - 収集間隔(Step) | |
| <u>.</u> | |
| はい ① いいえ ① | |
| | |

3. データの選択

DataCollection の選択ダイアログで、DataCollection リストから「土産話を知っている エージェントの数」を選択し、 ADD>> ボタンを押します。

| DataCollection選択 | | × |
|--|------------------------|---|
| | DataCollection違銀 | |
| DataCollectionリスト | 選択 された DataCollection | |
| すべてのエージェント エージェントの時つま意話 主意話を知っているエージェント 主意話を知っているエージェントの表 | ADD >>> C < PERMUYE | |
| | 了解 取消し | |

「選択された DataCollection」に「土産話を知っているエージェントの数」が追加されたら、 「ア解」ボタンを押します。また、GraphComponent Editor に戻ったら、 はい ボタンを 押します。

4. グラフの表示

今作ったグラフの設定「土産話を知っているエージェントの数」を選択し、 動作開始 ボタ ンを押します。

| 🚧 GraphComnonent Manager | _ 🗆 🗵 |
|-----------------------------|-------|
| <↓ 生産話を知っているエージェント数のグラフ 停止中 | 作成 |
| | 編集 |
| | 動作開始 |
| | 動作停止 |

表示が停止中から動作中に変わります。 グラフ を押して、Graph Viewer を起動します。

| WW GraphComponent Manager | |
|---------------------------|-------|
| ●土産諸を知っているエージェント数のグラフ 動作中 | 作成 |
| | 編集 |
| | 動作開始 |
| | 動作停止 |
| | 削除 |
| | (197) |

制御パネルの実行ボタンを押すと、収集されたデータがグラフとして表示されます。 時間が経過すると土産話を知っているエージェント数が上昇していき、ある値に達すると一 定になります。



11.3.3 PlatBox Simulator の終了

1. PlatBox Simulator の終了

```
PlatBox Simulator を終了するには、メインウィンドウの右上にある × ボタンを押すか、
メニューから [ファイル] [終了] を選択します。
```

道案内 100人の村人のうち、約81人までは急激に土産話が知れ渡ることがわかりま すね。モデルの中で、友人関係に方向性を持たせたので、これ以上土産話は広まりませ ん。全員に知れ渡らせるにはどうしたらよいか、興味のある方は考えてみてください。

次章からはまた別のシナリオが始まります。アクションも複雑になってくるので、がん ばってついてきてくださいね。

第 ||| 部

シミュレーションデザイン演習【拡 張編】

第12章

概念モデリング・拡張編

【学習目標】

シナリオの裏側に隠された要素を抽出できるようになる

12.1 つくりたい世界

今回追加されたシナリオを確認してみましょう。時間が経つにつれて BoxVillage に新 しい村人が増えていきます。新しい村人は、既存の村人と友人関係をもち、すぐにおしゃ べりを始める、というものでした。

それでは、第??章と同じように、ここから要素を抽出していくことにします。

12.2 登場する事物を定義する

今回、新たに登場する要素にはどのようなものがあるでしょうか。村人が増えて、既存 の村人との友人関係をむすんでいくだけですから特に新しい要素はないように思えます。

今回のポイントは、BoxVillage という舞台の上に、いかに新しい村人を加えるか、という部分です。ここで、「村人追加者」というエージェントに新たに登場してもらうことにしましょう。

村人追加者は、新しい村人を BoxVillage という舞台に登場させるための、いわば「裏 方エージェント」です。BoxVillage に新たな村人を登場させ、既存の村人との関係を結 ばせるという役割を担っているのです。村人追加者は、ストーリーの中には直接でてきま せんが、今回のシミュレーションを動かす上では必要不可欠な存在です。

概念モデル・クラス図として記述すると、次の通りになります。



道案内 次章で作成する一連のモデルは、引き続き BoxVillage プロジェクト、boxvillage パッケージ内で行います。さあ、さっそくまた世界をつくっていきましょう。

第13章

成長するネットワークのデザイン

13.1 つくりたい世界

この章では、新たに追加されたシナリオを作ります。概念モデリングをもとに、世界に 修正を加えていきましょう。

ここでは、以下の流れでモデルのデザインを行っていきます。新たにクラス図、状態遷 移図、アクションプロック図の作成を行います。

```
モデルのデザイン
Model の作成

Model の新規作成
Model のクラス図作成
Model のソースコード生成

Behavior のデザイン

Behavior の新規作成
状態遷移図の作成

アクションのデザイン

アクションファイルの新規作成
アクションブロック図の作成- アクションのコード生成

世界の初期設定

既存の World を修正する- World の設定
World のコード生成
```

13.2 Model の作成

村人追加者とその行動を定義するための新しいクラス図を作成します。

| 村人 | 追加者の行動のクラ | ス図 | | |
|----|-------------------------------|----|--|---|
| | < <agent type="">></agent> | | < <behavior type="">> ア ランダム選択追加行動</behavior> | 1 |
| - | | _ | | 4 |

13.2.1 Model の新規作成:村人追加者の行動

1. パッケージの選択

パッケージ・エクスプローラーから、「boxvillage」を選択してください。

2. 新規作成ウィザードの起動

メインウィンドウの上部のツールバーにある
Model 新規作成ボタンを押します。

- Model 名の入力
 ファイルを保存する場所が、先ほど選択した場所であることを確認して、「ファイル名:」に
 「村人追加者の行動」と入力しましょう。入力ができたら、 [終了] を押します。
- Model ファイルの確認
 パッケージ・エクスプローラー内に「^図村人追加者の行動.model」というファイルが作成されていることを確認してください。

Model のクラス図作成

それでは、上記のクラス図のようにモデル要素を配置してみてください。

13.2.2 Model のソースコード生成

1. コード生成の実行

Model Designer のキャンバスまたはアウトライン上で右クリックし、表示されるメニューから「モデルのソースコード生成」を選択します。

2. 生成するタイプの選択

生成する Type の選択ダイアログが表示されるので、追加したモデル要素にチェックを入れ、 OK を押すと Model のソースコード (Java プログラム)が生成されます。

13.3 Behavior のデザイン

Behavior Designer を用いて、状態遷移図を作成します。

Behavior の新規作成

1. Behavior の新規作成

Model Designer のキャンバスにある「ランダム選択追加行動」を右クリックし、表示され たメニューのなかから「Behavior Designer で開く」を選択します。プログレスバーが表示 され、Behavior ファイルが生成されます。また、その Behavior ファイルを読み込んだ状 態で、Behavior Designer が起動します。

 生成されたファイルの確認 パッケージ・エクスプローラー上に、「ランダム選択追加.behavior」というファイルができ たことを確認してください。

状態、状態遷移、ガード条件、アクションの描画

以下のような状態遷移図になるように作ってください。



13.4 アクションのデザイン(ランダム選択追加行動)

Action Designer を用いて、「ランダム選択追加行動」のアクションを、手段の階層構造 に分解して記述していきます。ここで作成するアクションは、「ランダム選択によって追 加する」です。



13.4.1 アクションファイルの新規作成

1. アクションファイルの新規作成

 生成されたファイルの確認 パッケージ・エクスプローラ上に、「^図ランダム選択追加行動.action」というファイルがで きたことを確認してください。

13.4.2 アクションブロック図の作成

「ランダム選択によって追加する」アクション

1. ブロックの配置 まず、Action Designer のパレットにある ブロック ボタンを押した後、「ランダム選択に よって追加する」アクションの内部をクリックします。

配置されたブロックの「新規ブロック」と書かれた部分をダブルクリックすると、文字を編 集できるようになるので「既存の村人をランダムに二人選び出す」に変更してください。 同様に、今できたブロックの下に、並べて「新しい村人を追加する」「新しい村人と既存の村 人との間に友人関係をむすぶ」という二つのプロックを作成してください。

| ● ランダム選択によって追加する |
|--------------------------|
| ■ 既存の村人をランタムに二人選び出す |
| |
| |
| ■ 新しい村人を追加する |
| |
| |
| |
| ■新しい村人と既存の村人との間に友人関係をむすぶ |
| |
| |
| |

ブロックは、アクションパーツを操作のまとまりごとに整理する役割をもっています。です から、まずブロックを使って全体の構造を把握し、その上でブロックの中にアクションパー ツを配置していくことにします。

また、アクションファイルのソースコードを生成したとき、ブロックのタイトル部分はコメントになります。

2. 既存の村人をランダムに二人選び出す

次ページ以降にあるアクションブロック完成図を参考に、アクションを作成してください。 アクションパーツを追加するときは、「既存の村人をランダムに二人選び出す」ブロック内 に配置していきます。

| = 文 🖈 | l |
|------------------|-----------------------|
| アクションパーツ | |
| ➡ 操作の呼び出し | ▶ ランダム選択によって追加する |
| 🛃 変数宣言/代入 | ▶ ■ 既存の村人をランダムに二人選び出す |
| 22 四則演算 | |
| 14 論理演算 | 5 |
| 體 文字列 | ~ |
| 刁 return文 | |
| 🕲 break文 🗸 🗸 | |
| | ■ 新しい村人を追加する |

「村人の集合からランダムで二人を選び出す」において、「抽出した村人のリスト」「選ばれた村人のリスト」の型を、名前欄の右側にあるプルダウンから「ArrayList」に変更してください。また、「一人目の村人を取得する」「二人目の村人を取得する」において、「取得した一人目の村人」「取得した二人目の村人」の型を、同様にして「Agent」に変更します。

| 上 人のAgent | を取り出す | | _ |
|--------------|------------|-----------|----|
| ₽□ | 抽出元の村人のリスト | ArrayList | •) |
| ∽□ | 抽出する要素の数 | (int | •) |
| ₽□ | 選ばれた村人のリスト | ArrayList | •) |
| · 🖭 一人 | 目の村人を取得する | | ٢ |

3. 新しい村人を追加する

アクションパーツを追加するときは、「新しい村人を追加する」ブロック内に配置していき ます。

「新しい村人を作る」というのは、世界に村人を登場させるということです。その村人に「お しゃべり行動」を持たせることで、今までは世界の初期設定で行っていた、「村人グループ」 の一人を新しく追加できたことになります。

4. 新しい村人と既存の村人との間に友人関係をむすぶ

アクションパーツを追加するときは、「新しい村人と既存の村人との間に友人関係をむすぶ」 プロック内に配置していきます。

今までは一方向の「友人関係」で結ばれた「村人のネットワーク」を設定していましたが、 今回は必ず双方向の「友人関係」を持つことにします。

Logger.getLogger("boxvillage.ランダム選択追加行動")

| logger (Logger) | |
|--|---------------------------------------|
| initialize | ו |
| | |
| | Worldに存在する村人を全て取得する |
| terminate | World 取得する ランダム選択追れ行動 |
| ● ランダム選択によって追加する | |
| ■ 既存の村人をランタムに二人選び出す ● Worldに存在する村人を全て取得する | ∟ |
| <type:boxvillagemodel〉村人< td=""><td>村人の集合からランダムで二人の村人を取り出す</td></type:boxvillagemodel〉村人<> | 村人の集合からランダムで二人の村人を取り出す |
| を全て取得する | ランダム選択追加行動 |
| 取得した村人の集合 (Collection) | |
| 一部 村人の集合からランダムで二人の村人を取り出す | ランダムで二人の村人を取り出す |
| 取得した村人の集合 から | |
| L 2 人のAgentを取り出す | |
| Hather and the second secon | ↓ Agent の集合からランダムで N人の Agent を取り出す |
| → 抽出する要素の数 (int) | 一人目の村人を取得する |
| 送ばれた村人のリスト (ArrayList) | ランダム選択追加行動 |
| | 選ばれた村人のリスト 名 B |
| | 一人目を取得する |
| | 村人取得した一人目の村人 |
| L | |
| ▶ 取得した一人目の村人 (Agent) | ▲ (1) リストのN番目を取得する |
| | 二人目の村人を取得する |
| | ランダム選択追加行動 |
| | 選ばれた村人のリスト |
| | □ 二人目を取得する |
| ▶ 取得した二人目の村人 (Agent) | |
| | 「一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一一 |



13.4.3 アクションのコード生成

1. コード生成の実行

「闘ランダム選択追加行動.action」のキャンバスを右クリックして「ソースコード生成」を 選択します。プログレスバーが表示され、アクションのソースコード(Java プログラム)が 生成されます。

2. 生成されたファイルの確認
 パッケージ・エクスプローラー上に、「³ランダム選択追加行動.java」というファイルが表示されていることを確認してください。

13.5 世界の初期設定



13.5.1 既存の World を修正する

- 既存の World を開く
 パッケージ・エクスプローラー上の「BoxVillageWorld.world」をダブルクリックします。
- モデルの読み込み
 World 設定画面の最上部にある「モデルの読み込み」ボタンを押して変更された情報を
 World に反映させます。

13.5.2 World の設定

[村人追加者グループの設定]

今回は、前回までの設定はそのままにして、村人追加者グループを追加しましょう。

- Agent グループ設定ウィンドウを開く
 World Composer の画面をスクロールさせ、「Agent グループ」の部分を表示します。
 Agent グループの枠内にある「追加」ボタンを押し、Agent グループ設定ウィンドウを開きます。
- Agent グループの名前の入力
 「名前:」を「村人追加者グループ」にします。
- Agent グループの AgentType の選択
 「AGENTTYPE_村人追加者」を選択します。
- Agent グループの BehaviorType の選択
 「BEHAVIORTYPE_ランダム選択追加行動」のチェックボックスにチェックを入れます。
- 5. Agent グループの登録 Agent グループウィンドウの OK ボタンを押して、Agent グループの設定を登録してく

ださい。

Agent グループの表のなかに、「村人追加者グループ」の設定が表示されていることと、「村 人追加者グループ」の数が「1」に設定されていることを確認してください。

[村人グループの設定]

- Agent グループの値設定ウィンドウを開く
 Agent グループの表の中にある「村人グループ」の行から、値設定ウィンドウを表示します。
- 2. 数の入力
 村人グループの数を「5」人にしましょう。

以上の作業が終わったら、Agent グループの表示が次のようになっていることを確認してください。



13.5.3 World のコード生成

- 1. コード生成の実行 World Composer の最下部にある World の生成 ボタンを押しま す。プログレスバーが表示され、World のコード(Java プログラム)が生成され ます。
- 2. 生成されたファイルの確認パッケージエクスプローラー上の、「BoxVillage-World.java」というファイルが更新されていることを確認してください。

13.6 実行・検証

- 13.6.1 シミュレーションの実行 (PlatBox Simulator 上での操作)
 - 1. World の読み込み状態の確認

PlatBox Simulator の制御パネル上に「BoxVillageWorld」と表示されていることを確認 してください。

2. ビューアの表示

メニューから [ビューア] [Relation Viewer] を選択して、Relation Viewer を開きます。 村人が表示されたことを確認してください。

シミュレーションの実行
 シミュレーションを実行します。時間が経過すると、村人 Agent が追加され、村人のネットワークが広がっていきます。



13.6.2 PlatBox Simulator の終了

1. PlatBox Simulator の終了

PlatBox Simulator を終了するには、メインウィンドウの右上にある「×」ボタンを押す か、メニューから [ファイル] [終了] を選択します。

対人が追加され、ネットワークが広がっていく世界は作れましたか?
BoxVillageのお話はここまでです。ここから先は、自分の作りたい世界を作ってみてください。興味のある分野に適応するのでも、このチュートリアルで扱ったモデルをさらに拡張するのでもよいでしょう。分からないことが出てきたら、チュートリアルの他にもマニュアル、リファレンスが用意されていますので、それを片手に進めてください。
さあ、準備はできましたか?それでは、始めましょう!



PlatBox Project では、シミュレーション・プラットフォーム「PlatBox Simulator」、 およびシミュレーション作成支援ツール「Component Builder」を開発・提供していま す。 PlatBox Simulator は、シミュレーションを「動かす」ことによって、社会を理解す るための道具です。これに加えて、私たちは、シミュレーションを「つくる」過程も社会 を理解する助けになると考えています。「Component Builder」は、そのような理解を助 けるモデル作成支援ツールです。この「動かすことでわかる」と「つくることでわかる」 という二つのアプローチによって複雑な社会を理解する そのための「新しい思考 の道具」をつくることが、私たちの目指すゴールです。

PlatBox Simulator は、Boxed Economy Project によって開発されてきた「Boxed Economy Simulation Platform」(BESP)の発展版です。Boxed Economy Project は、 1999 年から、社会・経済のモデル化を支援する道具立てについて考えてきました。その 後、私たちが提案するモデル化の枠組みは、経済分野に限定されたものではないと考え、 2005 年にプロジェクト名を「PlatBox Project」に、ソフトウェアの名前を「PlatBox Simulator」に変更しました。

【Boxed Economy Project 時代の主なメンバー】

- 青山希 (2002年~現在)
- 浅加 浩太郎 (2001 年)
- 井庭 崇(1999 年~現在) プロジェクトリーダー
- 海保研(2000年~2002年)
- 上橋 賢一(2000年~2002年)
- •北野里美(2001年~2002年)
- 高部 陽平 (1999 年~2001 年)
- 武田 林太郎 (2003 年~2005 年)
- 田中 潤一郎 (2000 年~2002 年)
- 中鉢 欣秀(2000年~2001年)
- 津屋 隆之介(2000年~2005年)
- 広兼 賢治(1999年~2001年)
- 松澤 芳昭 (2000 年~2004 年)
- 森久保 晴美 (2001 年~2002 年)
- 山田悠(2001年~2005年)



「モデル作成チュートリアルガイド」

執筆・編集者

PlatBox Project (編著)

(第I部)

井庭 崇

青山 希

津屋 隆之介

鈴木 祐太

(第 II 部, 第 III 部) 鈴木 祐太 松浦 廣樹 伊藤 智久 井庭 崇 宇佐美 絢子 宮澤 かおり

> Designers' Guide to Social Simulation, No.4 モデル作成チュートリアルガイド

2005 年 4 月 1 日 初版発行 2005 年 8 月 27 日 第 2 版発行

編著: PlatBox Project

〒252-8520 神奈川県藤沢市遠藤 5322 慶應義塾大学 井庭崇研究室 E-Mail: platbox@sfc.keio.ac.jp, もしくは iba@sfc.keio.ac.jp Web: http://www.platbox.org/

