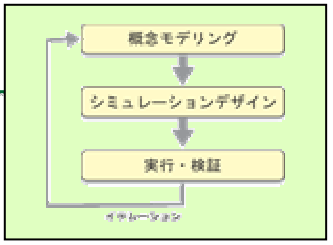
Keio University SFC 2004

『企業と市場のシミュレーション』

第6回：シミュレーション作成演習①

いば　たかし
井庭 崇
慶應義塾大学総合政策学部 専任講師
iba@sfc.keio.ac.jp
http://www.sfc.keio.ac.jp/~iba/lecture/

シミュレーション作成プロセス



- **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
- **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、コンピュータ上で実行できるシミュレーションを作成するフェーズ
- **実行・検証 フェーズ**
  - 作成したシミュレーションモデルを、BESP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。

モデルフレームワークの戦略的導入

復習

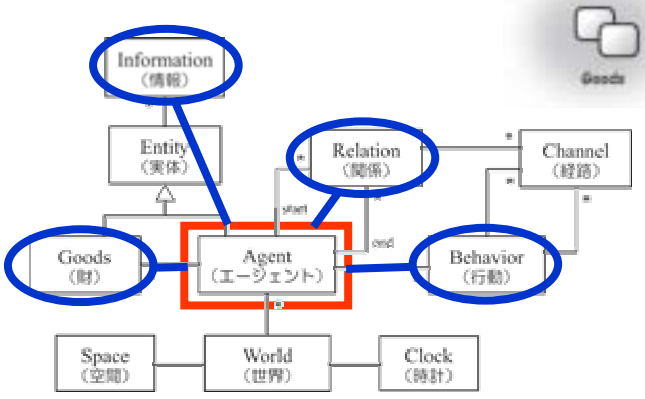
- 現実世界を分析･体系化する際に、毎回白紙の状態から行うのは大変な作業となる。
- このような問題への戦略的なアプローチとしては、
  - 科学的研究では、概念や用語、理論などを定義し、共有する。
  - ソフトウェア工学では、ドメインに特化したフレームワークを定義し、共有するということが行われている。

「モデル･フレームワーク」

Boxed Economy 基礎モデル

復習

マルチエージェントによる社会･経済モデルのための基本デザインを提供する。



Component Builder（CB）

復習

- Component Builderは、4つのデザイナーと、1つのコンポーザーで構成されている。


Activity Designer


Communication Designer

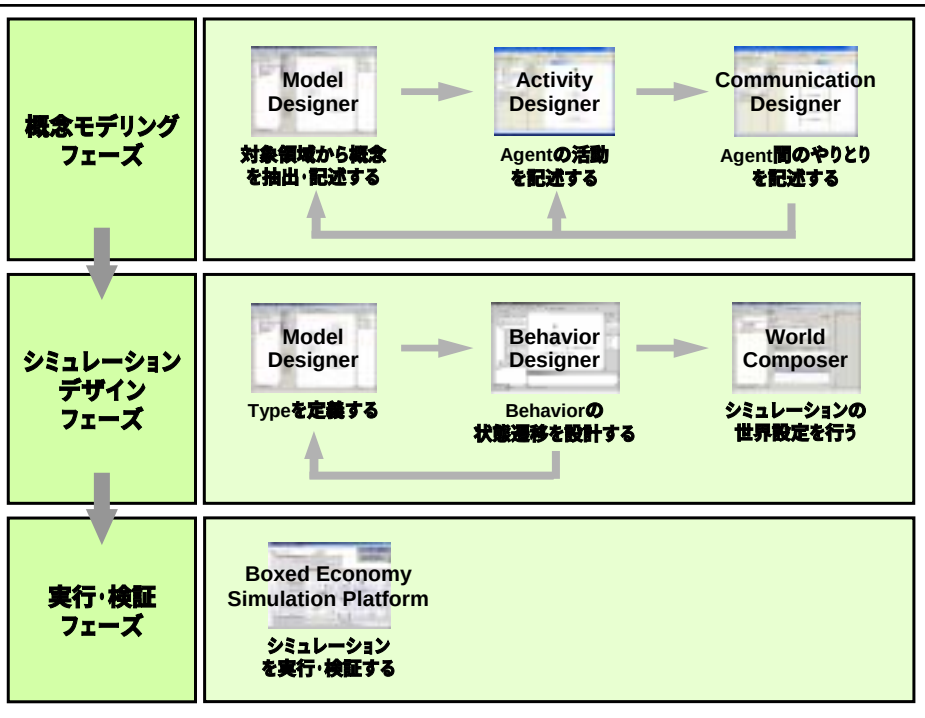

Model Designer


Behavior Designer


World Composer

eclipse

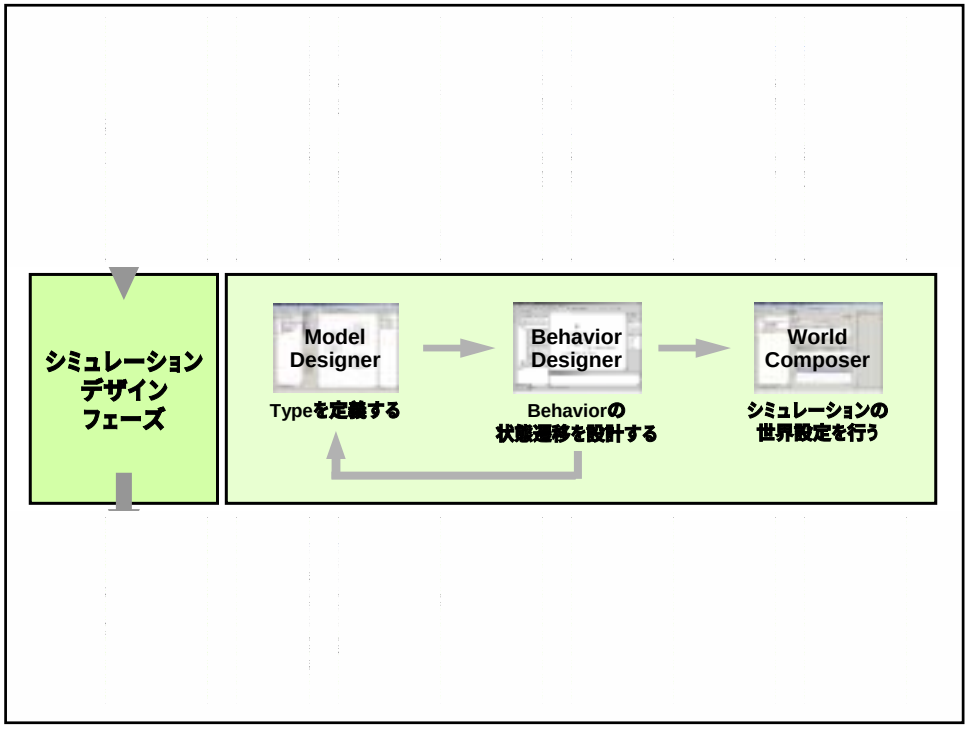
Component Builderは、オープンソースの統合開発環境eclipseのプラグインとして開発されている。

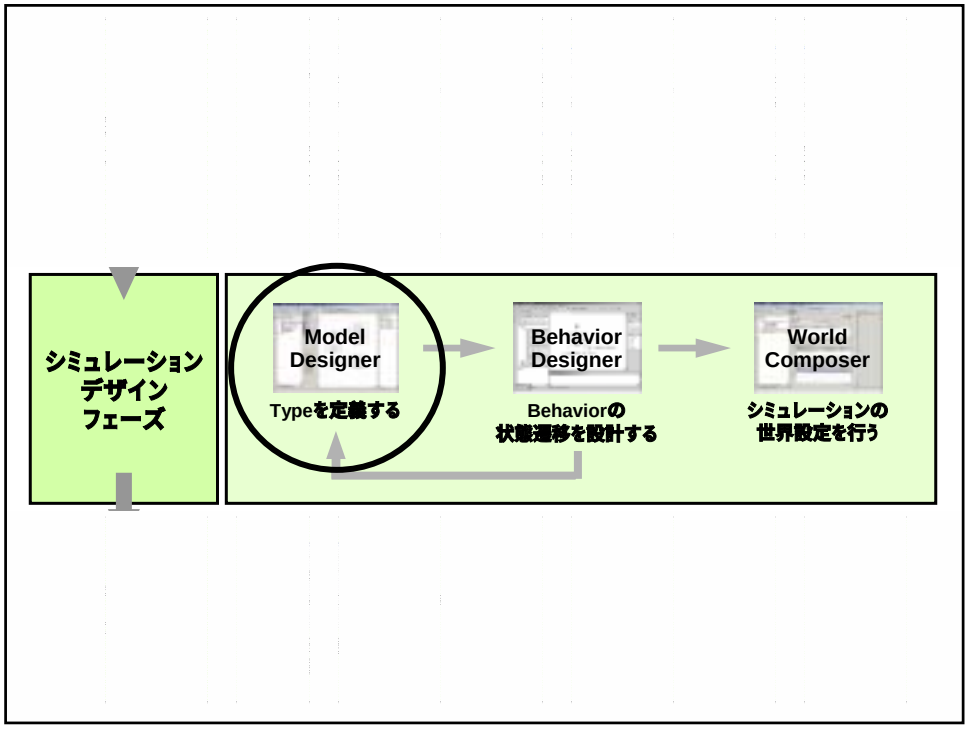


まずは、先週の残りから

企業と市場のシミュレーション（第5回）の続き

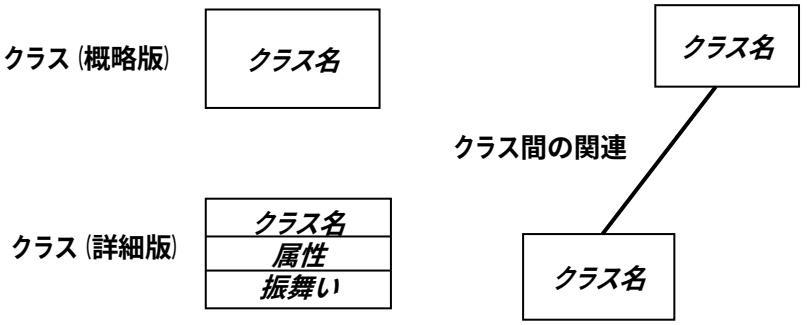
1. モデル・フレームワーク
2. 概念モデリングとComponent Builder
3. シミュレーションデザインとComponent Builder

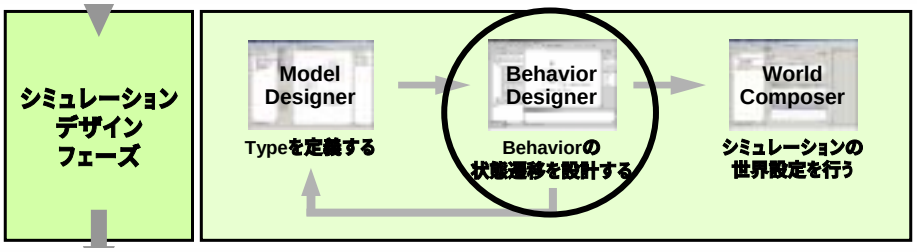

シミュレーション
デザイン
フェーズ
Model
Designer
Typeを定義する
Behavior
Designer
Behaviorの
状態遷移を設計する
World
Composer
シミュレーションの
世界設定を行う


シミュレーション
デザイン
フェーズ
Model
Designer
Typeを定義する
Behavior
Designer
Behaviorの
状態遷移を設計する
World
Composer
シミュレーションの
世界設定を行う

クラス図

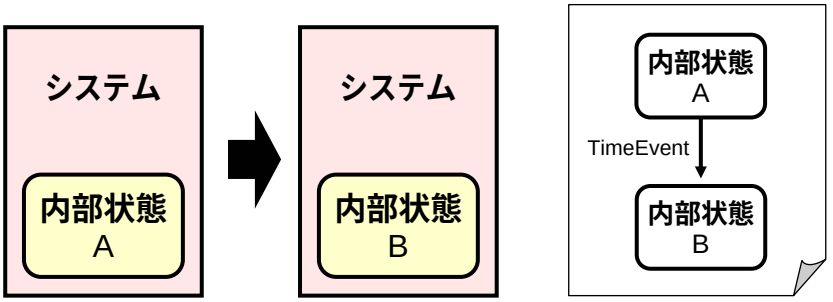
- クラス図は、モデルの静的･構造的な側面を表現するための図。





状態機械（オートマトン）としてのBehavior

- 基礎モデルでは、Behaviorのひとつひとつを「状態機械」(オートマトン)として記述します。
- 状態機械とは、トリガーとなるイベント(影響を及ぼすさまざまな出来事)を受け取ると、現在の状態に応じたアクション(動作)を行い、次の状態へ遷移するというシステムです。

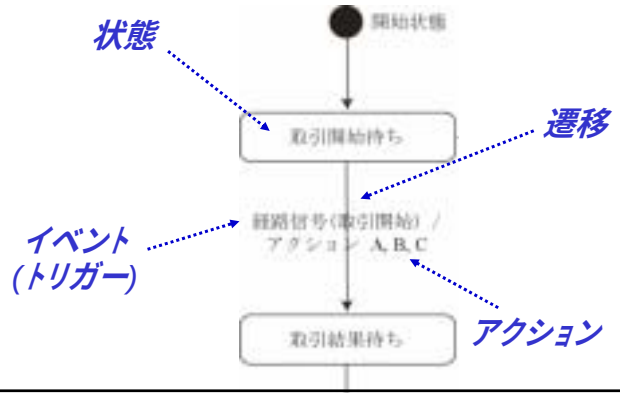


状態機械（オートマトン）としてのBehavior

- 状態機械のすべての状態の見取り図は、状態遷移図（ステートチャート図）を用いて表現することができます。
- Behaviorの状態遷移を引き起こすイベントには、時間が経過したことを表す「TimeEvent」と「ChannelEvent」があります。
- つまり、エージェントのBehaviorは、時間が経過した場合か、他のエージェントから何らかの働きかけがあった場合に活性化することになるわけです。

ステートチャート図

- ステートチャート図は、システムやオブジェクトの状態の変化（状態遷移）を記述するための図。
  - 外界のイベント（オブジェクトに影響を及ぼすさまざまな出来事）が発生すると、オブジェクトの状態が変わる。

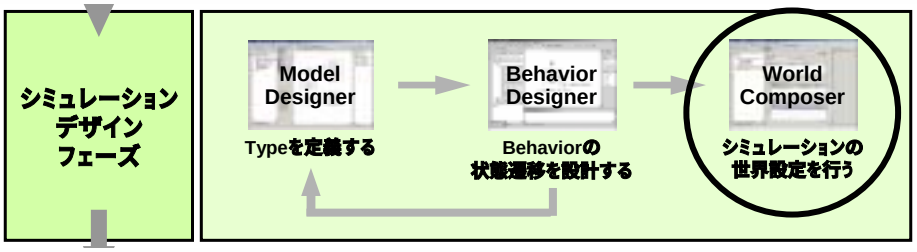


TimeEvent

- 基礎モデルで表現されたモデルでは現実世界と同じように時刻の経過によってモデルの状態が変わり、シミュレーションが実行されていきます。
- 基礎モデルではこの時刻の経過を「TimeEvent」がAgentに配信されることによって表現します。
- 一定時間ごとにTimeEventがモデルに存在するAgentに配信されます。そして、Agentは自分のBehaviorにこのTimeEventを転送しそれによってBehaviorが稼動することが、時刻が経過することによってAgentが行動することを表現しています。

ChannelEvent

- 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。
- Channelを通じてGoodsやInformationが送られてくると、「ChannelEvent」が発生します。



Keio University SFC 2004

『企業と市場のシミュレーション』

第6回：シミュレーション作成演習①

いば　たかし
井庭 崇
慶應義塾大学総合政策学部 専任講師
iba@sfc.keio.ac.jp
http://www.sfc.keio.ac.jp/~iba/lecture/

今日の目標

- **モデル・フレームワーク（基礎モデル）を理解する**
  → モデリングのために理解が不可欠です
- Component Builder **の使い方に慣れる**
  → 来週の作成演習で使うツールです

企業と市場のシミュレーション (第5回)

1. モデル・フレームワーク
2. 概念モデリングとComponent Builder
3. シミュレーションデザインとComponent Builder

企業と市場のシミュレーション (第5回)

1. モデル・フレームワーク
2. 概念モデリングとComponent Builder
3. シミュレーションデザインとComponent Builder

シミュレーションをつくってみよう

- 第III部（基本編） p.39
  - 第5章　BoxTownのパン屋さん
  - 第6章　いねむりご主人
  - 第7章　お客さんのお気に入り
  - 第8章　こんにちは！
- 第IV部（拡張編）
  - 第9章　いらっしゃいませ！
  - 第10章　ひとつ１００円になります
  - 第11章　3つで合計３００円です
  - 第12章　まいどあり！
- 第V部（お楽しみ編）
  - 第13章　商売繁盛☆
  - 第14章　注文個数は人それぞれ

明日の補講について

- 補講
- 5月２２日（土）３・４時限　ι１１にて

- 休講
- 5月２８日（金）←来週は、休講です
- 7月９日（金）

チュートリアルガイドの訂正一覧

- P.41の削除部分
  - 「プロジェクトの設定：Communication Viewer」は、やる必要がなくなりました。
- P.41への追加部分
  - 「パースペクティブの設定」
  - 詳細は、後のスライド参照
- p.53と104の修正
  - ファイル名が間違っています。修正してください。
  - 詳細は、後のスライド参照
- P.105,106への追加部分
  - 状態遷移図が2つ抜けています。
  - 詳細は、別紙参照。（図の部分を切り取って貼っておくとよいでしょう）
- p.117への追加
  - receiveOrderActionのプログラム
  - 詳細は、後のスライド参照

社会シミュレーションデザイナーズガイド（第2版） Tutorial Book

p.41への追加：パースペクティブの設定

- どのような形式で表示するのかを設定します。
  「Java」パースペクティブを選択します。

p.53と104の修正：ファイル名の間違い

- P.53
  - 「Model のクラス図の修正 (Behavior の追加)」の「1. Model Designer の起動」の部分

> Model Designer を起動するには、Package Explorer 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、「~~BoxTownModel.model~~」です。

(正) BakerModel.model

- P.104
  - 「Model のクラス図の修正 (Information の追加)」の部分

> 1. Package Explorer 内の「~~BoxTownModel.model~~」をダブルクリックして、Model Designer を起動します。

(正) BakerModel.model

p.117への追加：receivingOrderActionのプログラム

- SalesBehaviorのreceivingOrderActionの部分は以下のように書きます。
- お客さんからもらった注文を、パン屋さんが記憶します。

SalesBehaviorクラス内

```
receivingOrderAction(){
        IntegerInformation order =
            (IntegerInformation)this.getReceivedInformation();

        this.getAgent().putInformation(
            BoxTownModel.INFORMATIONTYPE_OrderInformation,
            order);
}
```