Reconstruction of an Agent-Based Simulation Model about Labor Market Policies

Master Thesis

Xi Niu

Reconstruction of an Agent-Based Simulation Model about Labor Market Policies: Master Thesis

Xi Niu

Abstract

In this master thesis an agent-based macroeconomic model used for economic policy experiments featuring a distinct geographical dimension and heterogeneous workers with respect to skill types is to be introduced and reconstructed with the help of the AOR simulation technology that was developed by the Chair of Internet Technology.

Zusammenfassung

In dieser Masterarbeit wird ein agenten-basiertes makroökonomisches Modell für die wirtschaftspolitischen Experimente mit einem eigenen geographischen Dimension und heterogenen Arbeitnehmern in Bezug auf Skill-Typen vorgestellt und mit Hilfe der am Lehrstuhl Internet-Technologie entwickelten AOR-Simulationstechnologie rekonstruiert.

Acknowledgements

I would like to express my sincere thanks to Prof. Dr. Gerd Wagner for his invaluable guidance and advice throughout the course of this thesis. Without his help this paper would not be possible.

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

So far agent-based modeling (ABM) is a powerful simulation modeling technique that has been extensively developed and well used in a lot of areas, because it can provide many effective methods to facilitate the research into the complex problems of different scientific fields. The model which will be described in this paper used ABM to explore the fields of economics. It focuses on an economy to analyse how the effects of different spatial distributions of economic policy measures depend on spatial frictions in the labor market expressed as commuting costs of workers who are employed outside their home-region. The purpose of this paper is to remodel it as a multi-agent based simulation using the Agent-Object-Relationship (AOR) simulation technology that was developed by Prof. Dr. Gerd Wagner and other team members at the Brandenburg University of Technology.

This thesis is organized as follows. Chapter 2 gives a general overview of the economic model. Chapter 3 offers a detailed analysis of the model. Chapter 4 presents the AOR simulation and the reconstructed model. Chapter 5 provides some simulation results, and Chapter 6 concludes.

Chapter 2. The model

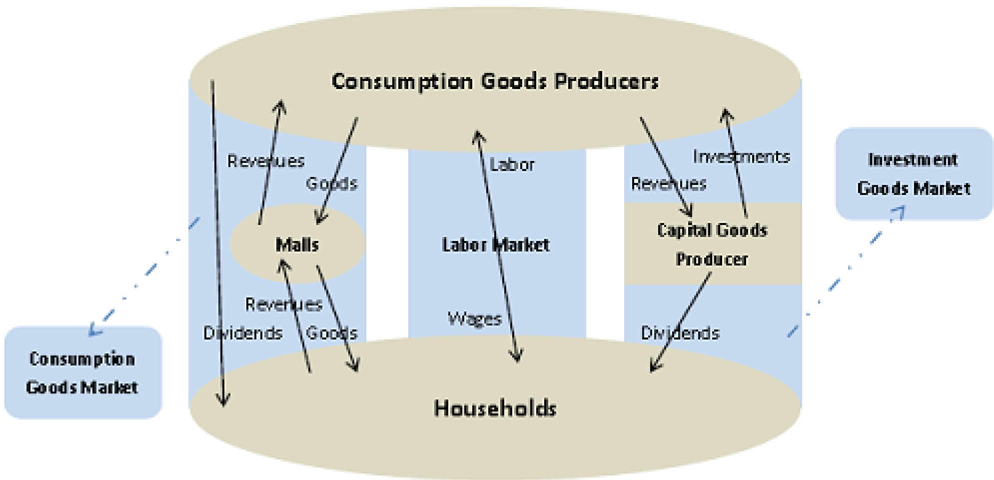
The agent-based macroeconomic model was developed by Herbert Dawid, Simon Gemkow, Philipp Harting and Michael Neugart and has been implemented in the Flexible Large-Scale Agent Modeling Environment (FLAME) developed by Simon Coakley, Mike Holcombe and others at the University of Sheffield (see http://www.flame.ac.uk for more information and references). This part gives an overview of the model.

Background to the model

The model was developed as part of a larger simulation platform for European policymaking known as EURACE. EURACE is a major project aiming at creating a complete agent-based model of the European economy for evaluating European economic policies. The three-years EURACE project started in September 2006. It involves economists and computer scientists from eight research centres in Germany, France, Italy, the UK, and Turkey, as well as the 2001 Nobel laureate in economics, Joseph Stiglitz. The EURACE model has a distinct spatial structure simulating the regional statistical units used by Eurostat. It contains various (typically, regional) artificial markets for real commodities (that is, consumption goods, investment goods and labor) and markets for financial assets (such as loans, bonds and stocks). For a general overview of the EURACE model, see http://www.eurace.org.

The model is a simplified version based on EURACE's labor market module. Its structure can be seen as follows.

Figure 2.1. Model Structure



The main purpose of the model is to investigate how the spatial skill distribution in the absence of policy intervention influences the speed of technological change, the flow of labor force and the growth of wage level in an economy. Therefore, policy implications aiming at the change of local skill distribution play an important role in the model. In order to capture the effects of different spatial distributions of policy measures, the economy is divided into two regions. According to general skill levels of workers a region can be declared as one of three possible types: low skill region, medium skill region or high skill region. More specifically, there are five general skill levels 1 to 5, where 1 is the lowest skill level and 5 is the level with the highest skill. In a low skill region the skill distribution is such that 80% of workers have the lowest general skill level, whereas the remaining workers are equally distributed across the other four levels of general skills. Analogously, a region is a medium skill or high skill region if 80% of workers have general skill level 3 respectively 5.

Table 2.1. General skill distributions in the three different types of regions

Region type	Level 1	Level 2	Level 3	Level 4	Level 5
Low skill	80%	5%	5%	5%	5%
Medium skill	5%	5%	80%	5%	5%
High skill	5%	5%	5%	5%	80%

In this model, both regions are initially set to low skill regions. A policy maker intends to improve the regional skill distributions. To that end, there are two options here. Either both regions are upgraded to medium skill or all efforts are concentrated in one region thereby moving this region to high skill whereas the skill distribution in another region stays unchanged. Finally, the effects of different policy types can be compared.

General description

The model describes an economy that contains an investment (or capital) goods, a consumption goods and a labor market. There exists a single type of product in each market, i.e., investment goods are supplied in the investment goods market, consumption goods are sold at malls in the consumption goods market and labor is considered as a commodity in the labor market. The economy is populated with a large number of agents, which are summarized as two types of active agents and two types of passive agents in the sense that active agents can take decisions, whereas passive ones can not. Each type of agent can have different roles corresponding to its activities in the markets. The following summarizes these roles.

Table 2.2. Agent types and their market roles

<table>
<tr><th>Agent</th><th>Type</th><th>Market</th><th>Role</th></tr>
<tr><td rowspan="2">Household</td><td rowspan="2">Active</td><td>Comsumption Goods Market</td><td>Buyer</td></tr>
<tr><td>Labor Market</td><td>Worker</td></tr>
<tr><td rowspan="3">Consumption Goods Producer (henceforth called CGP)</td><td rowspan="3">Active</td><td>Investment Goods Market</td><td>Buyer</td></tr>
<tr><td>Consumption Goods Market</td><td>Seller</td></tr>
<tr><td>Labor Market</td><td>Employer</td></tr>
<tr><td>Mall</td><td>Passive</td><td>Consumption Goods Market</td><td>Information transfer between consumption goods producers and households</td></tr>
<tr><td>Capital Goods Producer (henceforth called IGP)</td><td>Passive</td><td>Investment Goods Market</td><td>Seller</td></tr>
</table>

Markets and agents

Agents acting within markets are distributed across regions where the consumption goods market is local, the other markets are global.

The main actors in the investment goods market are IGP and CGPs. The investment goods market is global meaning that CGPs in both regions buy investment goods from the unique IGP. Investment goods are offered with infinite supply at an exogenously given price and there exists only one type of investment goods. The quality and price of supplied investment goods increase randomly over time. The amounts paid for investment goods are channeled back into the economy.

CGPs, households and malls take part in the consumption goods market. Together with labor, investment goods are used by CGPs to produce consumption goods. These goods are sold at malls to households. Malls are seen as the non-profit local market platforms. On the consumption goods market CGPs act globally in the sense that all CGPs store and offer their products at every regional mall, but households act locally because every household comes to the mall in his region to buy goods at posted prices. CGPs receive revenues from the sales. Their income is used to remunerate households in order to close the model.

CGPs and households play in the labor market. A search-and-matching process is used to represent the interaction between CGPs and households in this market. The CGP needs more laborers in order to expand its production scale. For this reason, it offers job vacancies based on the planned output. The household who is job seeker looks for a suitable position based on the corresponding salaries of these vacancies. Job seekers can apply for jobs in any region, but one thing must be pointed out, that is, working outside of their own region is associated with commuting costs which have to be subtracted form the wage. Thus, the labor market is global with spatial frictions determined by commuting costs.

Chapter 3. Analysis of the model

This chapter focuses on the analysis of the model. All the features will be listed. They are quite necessary for reconstruction of the model in AOR simulator. Moreover, a UML class diagram and two BPMN models are created for representing the model. This is a very important step.

Data extraction

In general, there are three ingredients in this model: market, agent and activity. Markets do not act, because they have no intentions and cannot perform actions. However, they can provide some contexts for agents to act in. Agents always act within markets. They take some activities with different roles. Therefore, the analysis is concentrated on agents, which are involved in different markets and characterized by different actions. As mentioned, there are four types of agents: household, CGP, mall and IGP. Agents and their activities are discussed in depth as follows.

Household

The model consists of a large number of households, who are simultaneously taking the roles of buyers and workers. Households receive their income and determine how much to spend and how much to save. In general, the household makes some decisions with the related roles affecting the markets as follows.

1. Allocate budget on consumption and saving
2. Choice of consumption goods
3. Search for a job
4. Acquire specific skills

Allocate budget on consumption and saving

The household acting as the role of buyer (or consumer) sets once a month the consumption budget which is spent on the consumption goods market and consequently determines the remaining part which is saved.

Table 3.1. The savings decision

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
personal consumption budget	B^{cons}	consumptionBudget	The consumer decides about the budget that he will spend for consumption	
the available liquidity	Liq^{Avail}	cashOnHand	The cash on hand that contains current income (i.e. labor income and dividends distributed by capital and consumption goods producers) and assets carried over from the previous period	
mean income	Inc^{Mean}	meanIncome	The mean individual (labor) income of a consumer over the last periods	

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the percentage of mean income	Φ	phi	$\Phi \leq 1$ is the percentage of the mean income such that the consumer spends all cash on hand below that level	*0.9*
marginal saving propensity	κ	savingPropensity	$0 < \kappa < 1$ is the saving propensity	*0.1*

Algorithm: There exists a critical value $\Phi * Inc^{Mean}$ of cash on hand to determine how much cash on hand will be spent for consumption in this month. When the available liquidity Liq^{Avail} is below this critical value the whole cash on hand will be spent. Thus, the consumption budget $B^{cons} = Liq^{Avail}$. In the opposite case the consumer will save a part of his cash on hand, so he sets his consumption budget according to the following consumption rule

$$B^{cons} = Liq^{Avail} - k * \left(Liq^{Avail} - \Phi * Inc^{Mean}\right) \tag{3.1}$$

Choice of consumption goods

The consumer purchases consumption goods according to his consumption budget. He splits the consumption budget into four equal shares, each of which is used for shopping per week. After determining the weekly budget, each consumer visits once a week to the mall in his region to buy goods. When visiting the mall he collects information about prices and quantities of different goods and then purchases goods according to his preference and available stocks of goods at posted prices. The model includes neither any kind of horizontal product differentiation, nor any kind of quality differentiation. Therefore, choice probabilities depend solely on prices.

Consumers make their purchasing decisions based on the prices of different goods using a stochastic rule as described in a standard logit model. In the marketing literature it is standard to describe individual consumption decisions. This model represents the stochastic influence of factors not explicitly modeled on consumption decisions, see [Guadagni and Little 1983].

Table 3.2. Selection of consumption goods

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the selection probability	*Prob*	selectionProbability	The consumer decides which consumption good to buy on the basis of the selection probability of every consumption good sampled by him	
available stocks of goods	G_{week}	availableProducts	A list of available products at the attended mall will be created in week (of period)	
the price of the consumption good	p_i	productSalesPrice	The price of the consumption good i	
the value of the consumption good	$v(p_i)$	consumptionValue	A function whose parameter is p_i determines the subjective value of the consumption good i for consumer	$-ln(p_i)$
the intensity of choice by consumer	λ^{cons}	intensityOfProductChoice	The intensity of choice by consumer	*8.5*

Algorithm: The decision of a consumer which consumption good to buy is random, where purchasing probabilities are based on the values he attaches to the different choices he is aware of. The consumer selects one consumption good i ∈ G_{week}, where the selection probability reads

$$Prob = \frac{Exp\left[\lambda^{cons} * v\left(p_i\right)\right]}{\sum_{i \in G_{week}} Exp\left[\lambda^{cons} * v\left(p_i\right)\right]} \quad (3.2)$$

Once the consumer has selected a consumption good he tries to spend the whole weekly budget for that consumption good if the stock at the mall is sufficiently large. In case the consumer cannot spend all his budget on the consumption good selected first, he has a single opportunity to select another good. If the budget is then not completely spent, the remaining amount is rolled over to the following week.

Search for a job

On the labor market households who are job seekers search for jobs (there are the unemployed plus a certain fraction of on-the-job searchers). They see posted vacancies and apply to the ones if the wage offers exceed the current reservation wage of the job seeker. After applying they receive zero, one or more job offers and rank these offers with respect to the wage offer. In case the offered position is outside the home region of the job seeker, commuting costs are subtracted from the offered wage. If two or more wage offers are equal then these are ordered randomly. Job seekers accept at most one job with the highest offered wage and update their reservation wage which is the new wage. If job seekers are still unemployed they decrease their reservation wage.

Acquire specific skills

The household is characterized by a general skill level and specific skills. His general skill level is determined by outside factors like government and economic policy. The specific skills of workers are acquired on the job.

Table 3.3. Specific skills (of workers) decision

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the average quality of the capital stock	A_i	averageQualityOfCapitalStock	The average quality of the investment goods employed by CGP i	
general skill level	b^{gen}	generalSkillLevel	Every worker has a level of general skills $b^{gen} \in \{1, \ldots, b^{gen}_{\max}\}$	*[1…5]*
specific skill level	b_t	specificSkillLevel	Every worker has a level of specific skills in period t	
increasing in the general skill level of the worker	$\chi(b^{gen})$	chi	A function whose parameter is b^{gen} governs the speed of specific skill improvement	

Algorithm: While being employed each worker adjusts his specific skills to the average quality of the capital stock of his employer. The adjustment speed $\chi(b^{gen})$ depends positively on the general skill level of the worker.

$$b_{t+1} = b_t + \chi\left(b^{gen}\right) * \left(A_i - b_t\right) \quad (3.3)$$

where the formula of the function $\chi(b^{gen})$ is

$$\chi\left(b^{gen}\right) = 1 - 0.5^{1/\left(20 + 0.25 * \left(b^{gen} - 1\right) * \left(4 - 20\right)\right)} \quad (3.4)$$

Brief interpretation: There are 5 general skill groups 1 to 5, where 1 is the lowest skill group and 5 is the group with the highest skills. A worker from skill group 1 needs 20 months to close half of the gap between his specific skills and the technology of his employer, where a worker of skill group 5 needs only one fifth of that time, namely 4 months. Therefore, the higher the general skill level of a worker, the faster he acquires the specific skills associated with a given job.

Consumption goods producer (CGP)

The CGP plays the role of buyer, seller and employer and makes a large number of decisions to influence the markets. Overall, it operates the sequence of events in the following way:

1. Product stock (optimal inventory) decision
2. Production inputs (labor and capital) decision
3. Investment (in investment goods) decision
4. Employment (hiring and firing) decision
5. Production (quantity) decision
6. Pricing decision (which price to set)
7. Dividend payment decision

Product stock (optimal inventory) decision

The operating cycle starts with product stock decision. The CGP keeps a stock of its products at every regional mall. It decides once a month whether the inventories at different malls need to be refilled in order to try to avoid the shortage of supplied goods and maximize the expected profit. To that end the CGP checks the current stock level reported by each mall it serves and determines an optimal stock level for each mall using a standard managerial method, which is based on a solution to the "newsvendor problem", faced by a newsvendor trying to decide how many newspapers to stock on a newsstand before observing demand, trying to avoid both overage and underage costs if he orders too much or if he orders too little, see [Hillier and Lieberman 1986].

Table 3.4. Quantity choice

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the optimal stock level	Y	optimalStockLevel	The CGP replenishes its stock at each mall in every period up to a given optimal stock level	
the price of the consumption good	p_i	productSalesPrice	The price of the consumption good i	
the unit cost of production	c_{t-1}	unitCostOfProduction	The unit cost of production in period t - 1 (the previous period)	
holding cost	C^{inv}	holdingCost	Holding cost per unit remaining at the mall for one period	*0.1*
monthly discount factor	ρ	rho	The discount rate which takes into account the time value of money	*0.95*
the current stock level	SL	currentStockLevel	The level of the stock which is checked at each mall	

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the desired replenishment quantity	D_r	replenishmentQuantity	The desired replenishment quantity at the mall in region r	
the sum of the orders	D^{plan}	sumOfOrders	The sum of the planned delivery volumes for the malls	
the planned output	Q^{plan}	plannedOutput	The planned output that is used for the determination of the input factor needs	
a linear combination	ξ	xi	For combining the planned current demand with weight xi and the historic demand with weight (1 - xi)	*0.5*

Algorithm: In order to determine the optimal stock level the CGP estimates the demand distribution based on demands reported by the mall in the previous T months. Because the estimated demand distribution is not clearly spelled out in the original model, suppose now that the *D* demand follows a uniform distribution (continuous) between D_{min} and D_{max} among the last sales. The value of the optimal stock level satisfies the equation

$$\Phi^{plan}(Y) = \frac{p_i - (1-\rho) * c_{t-1}}{p_i + C^{inv}} \qquad (3.5)$$

Here $\Phi^{plan}(Y)$ denotes the cumulative distribution function (CDF) of the estimated demand distribution.

The CGP applies an optimal inventory policy to determine whether and how much to replenish inventory. The optimal inventory policy is the following:

- If the current stock level is greater than or equal to the optimal stock level $SL \geq Y$, the CGP does not need to replenish inventory D_r=0.
- If the current stock level is less than the optimal stock level $SL < Y$, the CGP needs to replenish inventory $D_r = Y - SL$.

Production inputs (labor and capital) decision

After completing the planned output, the CGP computes the required factor inputs. In this model for producing the homogenous consumption good two input factors are used, i.e. labor and capital. Based on the planned output the corresponding demand for capital and labor are determined.

Table 3.5. Factor demand

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the planned output	Q^{plan}	plannedOutput	The planned output that is used for the determination of the input factor needs	
labor intensity of production	α	alpha	0 < α, β and α + β = 1 α and β are the output elasticities of labor and capital	*0.662*
capital intensity of production	β	beta		*0.338*

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the average quality of the capital stock	A_i	averageQualityOfCapitalStock	The average quality of the investment goods employed by CGP i	
the average level of specific skills of employees	B	averageSpecificSkillLevel	The average level of specific skills of employees	
labor input	L^{plan}	plannedLaborInput	The planned labor force is directly related to the planned production quantity	
capital input	K^{plan}	grossInvestment	The planned capital stock is directly related to the planned production quantity	
the price of the investment good	p^{inv}	investmentSalesPrice	The price of the investment good	
The average wage of employees	w^e	laborPrice	The average wage of employees	

Algorithm: Two cases have to be considered for the factor demand determination:

$$K^{PLAN} = \frac{(\beta * w^e)^{\alpha} * Q^{plan}}{(\alpha * p^{inv})^{\alpha} * \min[A_i, B]}$$
$$L^{PLAN} = \frac{(\alpha * p^{inv})^{\beta} * Q^{plan}}{(\beta * w^e)^{\beta} * \min[A_i, B]} \tag{3.6}$$

and if $K^{PLAN} \geq (1-\delta) * K_{t-1}$, the desired capital and labor stocks read $K^{PLAN} = K^{plan}$ and $L^{PLAN} = L^{plan}$. Otherwise,

$$K^{plan} = (1-\delta) * K_{t-1}$$
$$L^{plan} = \left(\frac{Q^{plan}}{((1-\delta) * K_{t-1})^{\beta} * \min[A_i, B]} \right)^{\frac{1}{\alpha}} \tag{3.7}$$

Investment (in investment goods) decision

The existing capital stock of the CGP depreciates over time. Once there is a positive demand for investment goods, the CGP purchases the needed amount from the IGP thereby upgrading its capital stock.

Table 3.6. Investment demand

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the gross investment	K_t	grossInvestment	The stock of machines etc. needed for production	
the new investment	I	newInvestment	The CGP needs more investments in order to expand its production scale	
the depreciation rate of capital	δ	delta	The depreciation rate of capital	*0.01*

Algorithm: The capital stock of the CGP is updated as old capital is replaced by new investments.

$$K_{t+1} = (1-\delta) * K_t + I \tag{3.8}$$

Employment (hiring and firing) decision

After determining the required labor force during the calculation of planned production inputs, the CGP compares it to the existing labor force, and then decides to post vacancies or to dismiss workers depending on the difference between the required labor force and the existing labor force. In case a CGP has to downsize the labor force, it fires workers with the lowest general skill levels until the needed number of workers is reached.

In another case, if a CGP has a positive demand for labor, vacancies are posted together with a wage offer. The incoming applications are ranked with respect to the general skill level. More specifically, applicants with higher general skill levels are ranked higher. If there exist two or more applicants who have the same general skill level, they are ranked by chance. The CGP sends as many job offers as it has vacancies to the highest ranked applicants. If the CGP then receives job acceptances from the applicants, it updates the number of employees and the number of vacancies. Otherwise there are still some vacancies and the CGP increases the offered wage.

Production (quantity) decision

After the two input factors are completed, the CGP starts with the actual production cycle. The production technology is represented by a Cobb-Douglas type production function. In economics, the Cobb-Douglas functional form of production functions is widely used to represent the relationship of an output to inputs. In this model for producing the homogenous consumption good two input factors are used, i.e. labor and capital.

Table 3.7. Factor demand

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the quantity of production	Q	producedQuantity	The quantity of production	
labor intensity of production	α	alpha	0 < α, β and α + β = 1 α and β are the output elasticities of labor and capital	*0.662*
capital intensity of production	β	beta		*0.338*
the average quality of the capital stock	A_i	averageQualityOfCapitalStock	The average quality of the investment goods employed by CGP i	
the average level of specific skills of employees	B	averageSpecificSkillLevel	The average level of specific skills of employees	
labor input	L	plannedLaborInput	The planned labor force is directly related to the planned production quantity	
capital input	K	grossInvestment	The planned capital stock is directly related to the planned production quantity	

Algorithm: The production quantity of a CGP is given by

$$Q = \min[B, A_i] * L^{\alpha} * K^{\beta} \tag{3.9}$$

where,

$\min[B, A_i]$: Complementarity between B and A_i.

The CGP distributes the output from a central inventory to the local outlet malls where it tries to sell its merchandise.

Pricing decision (which price to set)

The price of the consumption good produced by the CGP changes with the unit cost in production.

Table 3.8. Pricing

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the price of the consumption good	p_i	productSalesPrice	The price of the consumption good i	
Mark-up factor	μ	markUpFactor	The difference between the cost of a product and its sales price	*0.2*
the unit cost of production	c_{t-1}	unitCostOfProduction	The unit cost of production in period t - 1 (the previous period)	

Algorithm: The CGP sets the price of its product according to the standard rule

$$p_i = (1+\mu) * c_{t-1} \qquad (3.10)$$

Dividend payment decision

At the end of every month CGPs have to check whether they are in a profitable position that households can receive dividends from them. The CGP pays dividends depending on its monthly realized profit and current balance of saving account according to a simple dividend policy.

Table 3.9. Dividend payment decision

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
the sales revenue	Rev_t	productSalesRevenue	The sales revenue in period t	
cost of production	C	costOfProduction	The production cost	
the monthly realized profit	Pro	monthlyRealizedProfit	The monthly realized profit is the difference between the sales revenue and the production cost	Rev_t-C
the current balance of saving account	Acc	currentBalanceOfSavingAccount	The current balance of saving account	
the dividends	Div	dividends	The CGP pays dividends to all households	
a fixed proportion	div	div	The CGP pays a fixed proportion $div \in [0,1]$ of	*0.9*

Variable/ Parameter	Symbol	Name (in the sim model)	Description	Value
			its profit as dividends to all households	

Algorithm: If the monthly realized profit of a CGP is not positive, the CGP pays no dividends and the losses are entered on the current balance of saving account. In case of positive profit, the CGP pays dividends based on a simple dividend policy that defines three kinds of dividend rates depending on the current balance of saving account. The rule states

1. If the balance is negative $Acc < 0$ and the debt is on a scale above the last monthly revenue $|Acc| > Rev_{t-1}$, the CGP pays no dividends $Div = 0$.

2. If the balance is positive $Acc > 0$ and savings are above the monthly revenue $Acc > Rev_t$, the CGP disburses all profits as dividends $Div = Pro$.

3. In the remaining case, if the balance is between these critical levels of the above two cases, the CGP pays out a fixed proportion of profits as dividends $Div = div * Pro$.

Mall

Mall is modeled as a passive agent in this model, so it cannot take decisions. This agent performs the selling role of CGP in the region. It keeps and receives consumption goods produced by CGPs, then sells them and collects every product sales revenue that is reported back to the corresponding CGP.

Capital Goods Producer (IGP)

In this model, the IGP is unique and acts globally. It only has one role and plays the role of seller on the investment goods market. It supplies investment goods infinitely to all CGPs. The investment good as a kind of productive factor of CGP has two properties, i.e. quality and price, which are increased by simple rules.

- The quality and price of the investment good increase over time due to technological change. The price varies with the quality.

- Every month the quality is increased by 5% with probability 10% where with probability 90% there is no change of quality.

Finally, in order to close the model, the monthly revenue of the IGP is uniformly distributed to all households.

UML and BPMN

In order to show agents and their interactions more clearly.

UML

Each agent class is represented by a set of attributes and methods that operate on the agent class. A UML class diagram is a convenient way of representing the agents of the model.

BPMN

The Business Process Modeling Notation (BPMN) is a graphical notation for describing various kinds of processes. The main notational elements in BPMN are *FlowObjects*, that are contained in *Pools* and connected via *Sequence*- and *MessageFlows*. They subdivide in *Events*, atomic and composite *Activities* and *Gateways* for forking and joining. SequenceFlows describe the sequence in which the several FlowObjects have to be completed, while MessageFlows describe the exchange of messages between Pools. Thus, BPMN combines the definition of local workflows and the interaction between them.

The model consists of two pools: CGPs and households. The process model based on the search-and-matching process is used to represent the interaction between CGPs and households in the labor market.

Chapter 4. Reconstruction of the model in AOR Simulator

This chapter starts with a description of Agent-Object-Relationship simulation. After this, the reconstructed simulation scenario of the original model will be introduced.

AOR simulation

This section introduces an important agent-based approach used for reconstruction of the model, called Agent-Object-Relationship (AOR). This approach is well suited for the conceptual modeling of organizational information systems. It uses a well defined metamodel and simulation language that can be executed directly with special simulation software AOR-JavaSim.

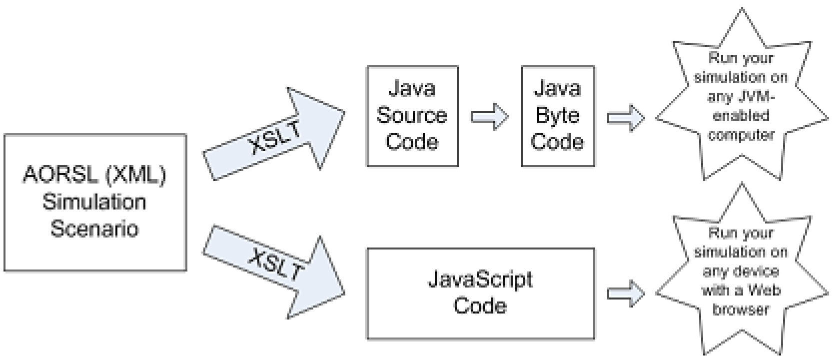
In this approach, the simulation system consists of a set of interacting agents plus a simulation environment in that all agents exist. Every agent is an independent unit in the simulation system that is able to act with other agents and with the simulation environment.. Every agent can represent one human actor and by creating large sets of free acting and interacting units, it can be possible to observe human behavior in larger networks or the influence of some persons behavior to the whole system.

AOR simulation framework

The Agent-Object-Relationship (AOR) simulation framework is intended to be used as a universal multi-purpose simulation framework in science, engineering, education and entertainment.

Entities in AOR are agents and objects as physical entities and events, actions and messages as relational entities. Agents are active entities, they interact with each other and the simulation environment. Objects on the other side are passive entities like a book. The modeling of AOR is based on ER modeling, this means, AOR entites are in special relationships to each other.

Figure 4.1. Code Generation



Simulation language

AOR simulations use a high-level declarative language to specify simulations. This language - AORSL - Agent-Object-Relationship Simulation Language - can be processed directly with AOR-JavaSim. It is an XML-based language that uses an XML Schema definition to enable easy validating and parsing of AORSL documents. Using XML syntax also has the advantage that AORSL files can be created, edited and viewed with a lot of free software. Furthermore, by using XML as a basis, the language remains human readable. The language is based on AbSimML, the description language for Agent Based Simulations. To use it with AOR simulations, it was necessary to adapt it to the vocabulary and syntax of AOR. In addition to this, the language was extended to enable a lot more functionalities to make AOR more flexible for different purposes.

Simulation scenario and simulation model

A simulation scenario consists of a simulation model, an initial state definition and a user interface definition, including a statistics UI and an animation UI. A simulation model consists of an optional space model (needed for physical objects/agents), a set of entity type definitions, including different categories of event, message, object and agent types and a set of rules, which define causality laws governing the environment's state changes and the causation of follow-up events.

Both the behavior of the environment and the behavior of agents are modeled with the help of rules. **Rules** are defined as follows,

A rule is a 6-tuple < *WHEN, FOR, DO, IF, THEN, ELSE* > where

- *WHEN* is an event expression specifying the type of event that triggers the rule
- *FOR* is a set of variable declarations, such that each variable is bound either to a specific object or to a set of objects
- *IF* is a logical condition formula possibly containing variables
- *DO*, *THEN* and *ELSE* are execution elements consisting of two blocks:
  - *UPDATE-ENV* is an expression specifying an update of the environment state
  - *SCHEDULE-EVT* specifies a list of resulting events that will be scheduled

AOR-JavaSim

AOR-JavaSim is a java software for generating and running AOR simulations. The software generates executable Java code from a given simulation scenario and compiles this code.

Running the simulation scenario

1. Validate your ERSL/AORSL file
2. Generate program code (either Java or JavaScript)
3. Run the resulting simulation program (with a JVM or in a Web browser)
4. Evaluate the statistics and/or the simulation log

The scenario specification

This section shows the reconstruction of the model as an AOR simulation. In order to show possible consequences of different policy measures, different scenarios can be generated by specifying alternative sets of values for the exogenous variables. The base case will be described in detail, whereas only the important results of the other scenarios are given. The scenarios are constructed so that each one represents an additional policy measure, and, therefore, the growth of the economy improves with each additional step, the base case being the lowest of all.

General Parameter Setting:

- 2 regions, one mall per region
- 400 households
- 10 Consumption Goods Producers
- 1 Capital Goods Producer

- Rate of Growth of Technological Frontier: 6%
- 5 Levels of General Skills (1 → 5)

Starting from low/low case the government can improve general skill distributions.

- Option 1: Efforts are spread over both regions.
  - Low/low → medium/medium
  - 80% Skill group 3, 5% for each Skill group 1, 2, 4, 5
- Option 2: Efforts are focused in one region.
  - Low/low → low/high
  - Region 1: 80% Skill level 1, 5% Skill group 2, 3, 4, 5
  - Region 2: 80% Skill level 5, 5% Skill group 1, 2, 3, 4

The base case

As mentioned, there are four types of agents in the model. All agents in the simulation are instances of these agent types.

Base Case: Low skill distribution in both regions

- 80% Skill Level 1 (lowest skill group)
- 5% Skill Levels 2, 3, 4, 5

Agents specification

Every agent in the simulation has several reaction rules that are used to define the behavior of the agent.

CapitalGoodsProducer

The rule *CalculateEqualShare_Rule* applies when an agent perceives the non-action event *CalculateEqualShare*, which simulates at the end of every month (occurrenceTime="24") and occurs at every 24 steps in the simulation. This perception is used to calculate the equal shares which will be soon distributed to all households. The *equalShare* property holds an equal share for a household is determined by the amount of households. There are 400 instances of the agent type *household*, so the agent divides its monthly revenues into 400 equal shares. The *equalShare* is expressed as the amount of households divided by the *investmentSalesRevenue* property of the agent. Then, the *investmentSalesRevenue* property is set to 0. This event causes a thing, the agent does the *PayEqualShare* action event to pay the equal shares to another agent *household*.

Mall

The rule *AtStartOfMonthCheckStockLevel_Rule* is used when the agent perceives the non-action event *AtStartOfMonthCheckStockLevel*, which simulates at the beginning of every month (occurrenceTime="2") and occurs at every 24 steps during the running of the simulation. This perception is used to check the current inventories of consumption goods. The result of the reaction rule is the creation of a new outgoing message to report the current stock level of every selling product to the corresponding CGP. The value of the *quantity* property of the message is obtained from the *productsInStock* list of the agent.

The rule *DeliverProduct_Rule* applies in case a message of type *DeliverProduct* is received by the agent. This will increase the inventory of the target product by the *quantity* property of the message. To that end, the *updateInventory* function of the agent is called. It has two parameters: *firmId* and *quantity*.

firmId holds the "identity" of the sender of the message and *quantity* holds the *quantity* property of the message. The function is used to update the value of the *quantity* property of the selected record of the *productsInStock* list.

ConsumptionGoodsProducer

The rule *TellCurrentStockLevel_Rule* applies when a message of type *TellCurrentStockLevel* is received. The agent will check whether the stock it keeps at the mall has to be refilled based on the given condition. If statement needs to be used. If the *quantity* property of the message is greater than or equal to the *optimalStockLevel*, the desired replenishment quantity of the product is set to zero. In the opposite case, the desired replenishment quantity is obtained by deducting the *quantity* property of the message from the *optimalStockLevel*. When the desired replenishment quantity is fixed, a new record which has three attributes (that is, *mallId*, *quantity* and *adjustmentFactor*) will be added to the bottom of the *inventoryPositions* list of the agent.

The rule *MakeProductionPlan_Rule* is used when the agent perceives the non-action event *MakeProductionPlan*, which occurs when the agent starts making its production plan (occurrenceTime="4") and is repeated every 24 steps. This perception is used to determine the demand of two input factors (labor and capital) for production. For that purpose, the *determineProductionPlan* function of the agent is called. This function is constructed based on CES production functions that have been mentioned above. It has five arguments: *laborPrice*, *investmentPrice*, *plannedProductionQuantity*, *averageSpecificSkillLevel* and *averageCapitalStockQuality*. *laborPrice* holds the average wage of employees of the agent. *investmentPrice* holds the price of the investment good. *plannedProductionQuantity* holds the planned output based on the desired replenishment quantity of the product for each mall. *averageSpecificSkillLevel* denotes the average specific skill level of workers and *averageCapitalStockQuality* denotes the average quality of the capital stock of the agent. The event *MakeProductionPlan* causes several things. First, the agent does the *BuyNewInvestment* action event to purchase the needed amount of investments from the IGP. This event triggers if the *newInvestment* property of the agent is greater than zero. Second, the agent does the *DismissWorker* action event to downsize the labor force. Finally, the agent creates an action event *PostVacancyInformation*, if the value of the *laborSupplyQuantity* property of the agent is larger than zero.

The rule *TellSalesRevenue_Rule* applies in case a message of type *TellSalesRevenue* is received by the agent. This will increase the *productSalesRevenue* property and *productSalesQuantity* property of the agent by the corresponding *revenue* property and *quantity* property of the message. There is nothing else to do in this case.

The rule *TellVacancy_Rule* applies in case a message of type *TellVacancy* is received. The agent receives informations from the applicants about their respective general as well as specific skill levels, and then adds these informations to the bottom of the *jobApplications* list of the agent.

The rule *InFirstIterationRankApplicant_Rule* is used when the agent perceives the non-action event *InFirstIterationRankApplicant*, which occurs when the agent ranks the applicants in the first round (occurrenceTime="9") and is repeated every 24 steps. It applies only when the size of the *jobApplications* list is greater than zero. This perception is used to choose the best applicants for the positions that are needed. Thus, the *determineJobOffer* function of the agent is called. It sorts and updates the *jobApplications* list. This event causes a thing, the agent does the *InFirstIterationOfferJob* action event to send job offers to the highest ranked applicants.

The rule *AcceptJob_Rule* applies in case a message of type *AcceptJob* is received by the agent. This rule is used to update the *laborSupplyQuantity* property and the *workersInFirm* list of the agent. The value of the *laborSupplyQuantity* property minus one. Moreover, the informations which contains the message sender, the *wage* property of the message, the *generalSkillLevel* property of the message and the *specificSkillLevel* property of the message will be added to the bottom of the *workersInFirm* list of the agent.

The rule *ResignJob_Rule* applies in case a message of type *ResignJob* is received. The agent receives resignation from its employee and drops him from the list of workers. Specifically, the value of the *laborSupplyQuantity* property of the agent is increased by one. In order to delete a record from the *workersInFirm* list, the *deleteWorkerRecordFromList* function of the agent is called with a parameter *householdId* which holds the message sender. It first determines the record whose *householdId* property is equal to the "identity" of the sender of the message, and then delete this record from the *workersInFirm* list.

The rule *StartSecondIterationLaborSupply_Rule* carries out when the agent perceives the periodic time event *StartSecondIterationLaborSupply*, which happens at the beginning of the second iteration of hiring activity (occurrenceTime="15") and is also repeated every 24 steps. It is used only when the *laborSupplyQuantity* property of the agent is greater than zero. This event causes a thing, the agent does the *PostVacancyInformation* action event to post vacancies for job seekers.

The rule *InSecondIterationRankApplicant_Rule* is used when the agent perceives the non-action event *InSecondIterationRankApplicant*, which occurs when the agent ranks the applicants in the second round (occurrenceTime="18") and is repeated every 24 steps. The content of this rule is the same as that of the *InFirstIterationRankApplicant_Rule* rule.

The rule *StartOfProduction_Rule* applies when an agent perceives the periodic time event *StartOfProduction*, which simulates at the end of every month (occurrenceTime="23") and occurs at every 24 steps in the simulation. This perception is used to calculate the equal shares which will be soon distributed to all households.

The event *StartOfProduction* causes several things. First, the agent creates an action event *DistributeProduct* to distribute the products. Second, the agent does the *PayWage* action event to downsize the labor force. Finally, the agent does the *IncreaseSpecificSkillLevel* action event.

household

EnvironmentRules

EnvironmentRule: *Create_InitialProductsInStock_Rule*

The rule *Create_InitialProductsInStock_Rule* is carried out, when an exogenous event *Init* occurs. This event happens at the beginning of the simulation (occurrenceTime="1") and is used to create the initial state

. . Xi Niu. . .

Appendix A.

Glossary

Glossar

EURACE	EUR-ACE European Agent-Based Economics