Chapter 2
Computational Aspects of ABM

Summary This chapter addresses computational aspects of ABM before delving into the core of how the digital creativity formulates in company and how it affects the corporate performance. As an illustrative example, I propose a prototype MCNSS (Mobile Commerce Negotiation Support System) where suppliers and buyers are interacting with each other on a mobile platform until right partners are found. MCNSS is coded in NetLogo. Negotiation in the context of commerce always requires past knowledge about which kinds of commerce transactions in certain kinds of conditions were either best for the suppliers or for the buyers. Past knowledge of the number of transactions is used as an intelligent basis for the reasonable negotiations between suppliers and buyers in specific conditions. Mobile commerce, or m-commerce, requires a certain amount of real-time updating on the status of target products and services of suppliers, prospective customers, and changes in prices, in addition to other parameters. ABM technique was applied to designing and building MCNSS to facilitate smooth coordination of the conflicting interests existing between suppliers and buyers. Therefore, the core of MCNSS lies in how to model the behaviors of both buyers and suppliers, and in the decision support mechanism of the negotiation process. I included detailed NetLogo logics that were used in designing suppliers agent (or S-agents), buyers agents (or B-agents), and negotiation mechanism for the MCNSS. Also I have shown how the MCNSS can solve real-world problems. For the sake of clear illustration, sample problem description and simulation results as well were included with relevant snapshots and statistical results.

Keywords ABM • MCNSS • Mobile commerce • Decision support • Negotiation support system • Supplier agent • Buyer agent • Negotiation • Multiagents • Knowledge base • Utility • Globals • Setup • Shop • Customer • Customer offer • Buyer's utility • Seller's utility • Mobile groups • Active mobile group • Passive mobile group • Price • Negotiation process • Average utility • Average profit • Graph monitors • Turtle • Control button • Slider • Monitor • Command center • Behavior space

K.C. Lee, *Digital Creativity Model and Its Relationship with Corporate Performance*, SpringerBriefs in Business, DOI 10.1007/978-3-319-39991-1_2

2.1 Background

In order to understand the computational aspects of ABM, a concrete example is necessary in which details of how to conduct ABM in specific situations are clearly addressed. To fulfill this purpose, we need a specific ABM language by which decision makers can model their own situations to compute digital creativity and its contribution to corporate performance. To begin, it is important to understand that utilizing a specific ABM language is crucial for an ordinary decision maker to grasp the complicated relationship between digital creativity and its contribution to corporate performance. For this purpose, NetLogo, released by Northwestern University, is described with an illustrative example. To add a sense of reality, an illustrative example is suggested with an emphasis on a mobile commerce negotiation mechanism.

Negotiation in the context of commerce always requires past knowledge about which kinds of commerce transactions in certain kinds of conditions were either best for the suppliers or for the buyers. These can include product, service, delivery, order quantities, price, and quality. Past knowledge of the number of transactions is used as an intelligent basis for the reasonable negotiations between suppliers and buyers in specific conditions. Mobile commerce, or m-commerce, requires a certain amount of real-time updating on the status of target products and services of suppliers, prospective customers, and changes in prices, in addition to other parameters. In other words, in the context of m-commerce, it is imperative that the m-commerce users be given a high quality of decision support, which should be timely and consistent with past instances. When ABM is applied to resolving m-commerce situations such as this, we need agents representing both buyers and suppliers, and a kind of negotiation support mechanism facilitating smooth coordination of the conflicting interests existing between suppliers and buyers. Therefore, the core of ABM lies in how to model the behaviors of both buyers and suppliers, and in the decision support mechanism of the negotiation process. We name the prototype ABM-based decision support mechanism as MCNSS (Mobile-Commerce Negotiation Support System).

It is not unusual to suggest that dominating trends of the modern online commerce among buyers can be attributed to mobile commerce powered by pervasive use of mobile devices such as smartphones and laptops. Recent mobile computing is characterized by both its ubiquitous connectivity and its resources. We know that the ubiquitous characteristics of the mobile computing form the core basis for its popularity. Today, when the mobile computing is set as the standard type of communication between individuals and companies, mobile commerce (m-commerce) has been accepted as an ordinary type of business transactions. At this point, we need to understand that m-commerce has historically stimulated a substantial competitive advantage and has helped to improve workflow and efficiency by reducing costs and enhancing risk management. The contribution of m-commerce has often been used as a basis for promoting digital creativity in organizations.

For the ABM to be applied logically, the target problem must be complicated by multiple parties, as well as multiple types of interests that often conflict with one another. The first point to note about m-commerce is that multiple parties are

involved in m-commerce transactions. A number of prospective buyers want to search for bargains with ideal suppliers. Such bargains, however, must be checked for product quality and reasonable pricing. In the context of m-commerce, suppliers look forward to making transactions with ideal buyers who are willing to pay promptly for their products and services. Among the conflicting needs and desires suggested by buyers and suppliers, the most common type of conflicting interest is price. However, as we know, final price is the result of a complicated negotiation processes between buyers and suppliers. An issue here is that when buyers and suppliers are not familiar with one another, they have little chance of communication in the context of m-commerce. As is often the case, ideal buyers for a certain supplier may reside in rather distant locations. In addition, the ideal supplier for a certain buyer may not be aware of this buyer, due to lack of information; buyers often have their own jobs, as is true for the suppliers. This indicates that both buyers and suppliers want to focus on their daily duties without being disturbed by the m-commerce, while hoping their needs through m-commerce transactions can be satisfied automatically. This reality forces us to design an intelligent handling of the m-commerce transactions between buyers and suppliers.

A second point about m-commerce we need to consider is that its users are generally assumed to be on the move, and, therefore, are restrained to using mobile devices in a more limited offline mode. Therefore, m-commerce ABM tasks must consider various situations, including mobile shopping, location-sensitive information delivery, and telemetry. In addition, m-commerce users tend to have individual needs in order to maximize utility by demanding customization, personalization, location sensitivity, and context awareness.

The m-commerce negotiation mechanism we will consider in this chapter possesses several classic features of m-commerce. First, we assume that many sellers and buyers seek the best deal among them. Therefore, conflicts exist over the limited resources. In addition, since the entities engaged in the m-commerce tend to be very busy, they often cannot afford to refer to their mobile devices fully before making decisions. This means that semiautomatic facilitation of negotiation tasks is required for m-commerce transactions, without human intervention, until the final stage of confirmation between buyers and sellers. Therefore, if a certain number of agents in ABM are used to represent buyers and sellers, those agents must be allowed to freely negotiate with one another to satisfy users' prerequisites and tastes. Subsequently, negotiation results are fed back to the users for confirmation. With this method, user satisfaction will be significantly improved. Second, we adopt a knowledge base where past m-commerce transactions are stored as a reference for sellers to choose appropriate prices for potential buyers. Usage of a knowledge base in this manner is meaningful for sellers, as maintaining consistency in pricing for potential buyers is crucial to making m-commerce transactions successful. We can expect that a consistent way of pricing is possible if it is based on the extraction of consistent information from past similar cases in a knowledge base.

To meet this aim, we propose a new intelligent negotiation mechanism based on ABM, assuming that all m-commerce participants (sellers and buyers) are represented by multiagents. In addition, multiagents can seek decision support,

which remains robust regardless of changes in the m-commerce environment. The prototype, MCNSS (Mobile Commerce Negotiation Support System), is an ABM-driven decision support system where both sellers and buyers can negotiate with potential partners for more desirable deals over limited resources. For example, sellers can advertise their leftover products to potential buyers and adjust their prices according to negotiation contexts. In addition, buyers can make counteroffers by modifying their utility function. In addition, all negotiation procedures can be processed effectively through a form of negotiation among the multiagents representing both sellers and buyers. The MCNSS is coded in the NetLogo platform, which is a multiagent simulation platform supported by Northwestern University (http://ccl.northwestern.edu/NetLogo). Readers can then understand how digital creativity can be computed by studying the MCNSS carefully.

2.2 Multiagents and MCNSS

As we know, ABM is an abbreviation for "agent-based modeling." The most important component of the ABM is, of course, agents. Essentially, the (intelligent) agent is an intelligent software capable of performing flexible autonomous tasks in computer-mediated environments. The term "intelligent agent" can be distilled down to two words: intelligence and agency. The degree of autonomy and authority vested in the agent is its agency, which can be measured, at least qualitatively, by the nature of interactions between agents and other entities in the systems in which they operate. The degree of agency is enhanced if an agent represents a user, so collaborative agents represent a higher level of agency, because they cooperate with other agents or programs or entities. The agent's intelligence can be stated as its degree of reasoning and learned behavior, or its ability to understand the user's goals and to carry out the tasks it is given. In the MCNSS, agency is denoted as the negotiation needs among the multi-agents representing both buyers and sellers engaged in m-commerce, while intelligence is secured by the reasoning functions administered by the knowledge base of MCNSS where past m-commerce transactions are stored.

In the context of MCNSS, agents represent specific needs of buyers and suppliers. Those specific needs represented by multiple agents instill dynamics into the MCNSS because conflicting interests are triggered by the agents. Those conflicts of interest among buyers and sellers are brought into the negotiation processes of the MCNSS. Therefore, it is crucial for users to specify their specific needs and to program their agents to follow their needs during negotiations with other agents in the MCNSS-mediated environments. Once the agents' detailed specifics are parameterized by the users, a number of agents are working autonomously in the MCNSS environments until a final deal is reached in which the final buyer and supplier are determined with specific m-commerce terms, such as price and quantity. Use of agents is convenient for both buyers and suppliers; they are free to focus on their own jobs, while the multiple agents representing users' specific needs are autonomously working to make a final deal among them in the MCNSS environments.

In this sense, the MCNSS is essentially composed of multiple agents with diverse goals and capabilities working collaboratively to solve specific problems. Therefore, an effective platform for coordination and cooperation among disputing multiple agents is necessary. For example, when a conflict occurs between buyers and sellers over limited resources in the real world, it is difficult for a single authority to reconcile that conflict to the full satisfaction of all concerned. In contrast, use of multiple agents enables each conflicting interest to be represented by an agent. Then the MCNSS allows multiple agents to interact with one another freely based on a set of simple engagement rules as in a real world, until a final deal can be reached in which the utility of engaged business partners is maximized. An advantage of this ABM approach is that a high degree of complexity can be decomposed into numerous manageable, small problems, which can be easily solved. The final solution computed by the MCNSS is statistically reliable enough to then be applied to real-world situations.

In the case of MCNSS, m-commerce sellers and buyers are, respectively, represented by specific agents, such as S-agents and B-agents. Each agent ideally receives proper decision support from the knowledge base. When a target problem is composed of multiple factors, and the factors are assumed to interact with one another influencing the formulation of final solutions, then ABM can be applied to solve the target problem by having an agent represent each factor. By simulating the MCNSS until a desirable solution is found, the target problem can be resolved effectively. Similarly, the negotiation problem in the MCNSS can be represented by the ABM approach, in which an individual negotiation entity is represented by an agent. Then, the multiple agents approach is applied to resolve conflict among the multiple agents and to find the best deal for the buyers and sellers. The most prominent advantage in using the multiple agents approach in the MCNSS is that it excludes any need for human intervention, which could deter efficient problem solving in addition to cause unexpected emotional and economic side effects.

2.3 Role of Knowledge Base in MCNSS

It is well known that highly unstructured problems can be solved more easily and systematically when past knowledge is used as a consistent and effective decision support tool. The m-commerce problem is one of many highly unstructured problems, because prospective buyers are usually mobile, and various conditions for plausible transactions change quickly. Usage of past knowledge, in this case, may lead to robust and consistent decision support for both buyers and suppliers. The past knowledge in the MCNSS is limited to prices that the suppliers offer buyers. Price is a top-priority concern for prospective buyers and suppliers as well. Once a reasonable price is negotiated between buyers and suppliers, the success of m-commerce is maintained. In this sense, price is knowledge given by the past transactions. The MCNSS ensures that reasonable prices can be extracted from past knowledge, given similar commerce conditions. If such consistent information is found, then an explanation of why the proposed solution fits a new m-commerce case is also possible, based on this analogy.

2.4 Designing MCNSS with NetLogo

In this section, I explain how the MCNSS was developed using NetLogo language. However, to avoid a redundant description, only the core part of the MCNSS is described here. Full source code of the MCNSS is attached in the appendix. If you are not familiar with NetLogo, a user manual is available from this link: https://ccl.northwestern.edu/netlogo/docs/. As I write this book, the latest version of NetLogo was announced as a version 5.3. However, it is worth noting that the MCNSS introduced in this section was coded in NetLogo 3.1.5. A different version of NetLogo can be downloaded from https://ccl.northwestern.edu/netlogo/download.shtml. A useful tip for readers is to launch a correct version of NetLogo used by your ABM prototype. Since the MCNSS was coded using NetLogo version 3.1.5, you need to install NetLogo version 3.1.5 onto your computer to run the MCNSS.

2.4.1 Basics

The proposed MCNSS system starts with the assumption that a number of potential buyers are mobile among sellers' shops, and that they may wish to buy sellers' goods if offers are reasonably priced and of high enough quality. We also assume that a number of sellers compete with one another to find ideal buyers willing to pay for their goods, if sellers' offers pose reasonable bargaining conditions. Sellers are stationed at shops with fixed locations, offering transaction conditions through the MCNSS to potential buyers passing by. Buyers carrying their own mobile devices connected to the MCNSS can then respond to sellers' offers. From the perspective of sellers, providing consistent bargaining conditions is critical to sustain a certain degree of profit. In this sense, the knowledge base created from storing past m-commerce transactions is necessary; its primary role is to extract useful and consistent bargaining information from a set of similar past m-commerce transactions.

In the MCNSS, agents represent both buyers and sellers, and their names are denoted as B-agents and S-agents, respectively. MCNSS is assumed to be accessible online as a commercial subscription. Therefore, MCNSS is a virtual market service for both buyers and sellers. The moment buyers and sellers are plugged into the MCNSS, the corresponding B-agents and S-agents are created in a virtual market and administered by MCNSS. S-agents are supported by the knowledge base of MCNSS in order to maintain pricing consistency; meanwhile, MCNSS gives B-agents the ability to adjust their own utility function in accordance with buyers' preferences. In this sense, the MCNSS is able to provide location-aware services, and also provide time-critical and intelligent decision support to both sellers and buyers through the interaction of S-agents and B-agents.

MCNSS is utilized to buy both buyers and sellers. Buyers use MCNSS to specify their utility function based on a predetermined set of factors, which leads to development of personalized B-agents that represent buyers' personal preferences. Sellers can also utilize MCNSS to launch S-agents capable of promoting their products to potential buyers when it is necessary. Buyers access the MCNSS through their subscription telecommunications company to register their own preferences about commerce. When buyers are in the vicinity of certain sellers who offer reasonable transaction terms fitting buyers' preferences, the MCNSS prompts the B-agent to start negotiating with the S-agent. The MCNSS-administered negotiation process proceeds as follows: first, S-agents calculate an initial bid price for a sales item using the knowledge base of MCNSS, and then relay this price offer to the B-agents through MCNSS. Upon receiving the price offer, the B-agents compute their utility values. If MCNSS detects a "best fit" between the B-agents' utility values and the S-agents' price offers, the corresponding B-agents are notified, and the negotiation process stops with a final deal between B-agents and S-agents. Otherwise, the negotiation process continues until a final deal is settled.

From the following subsections forward, let us describe how to design S-agents and B-agents using NetLogo source code.

2.4.2 S-Agent

S-agent represents sellers' best interests, which is to maximize profit. Therefore, the primary function of the S-agent is to look for a potential buyer in an acceptable range of distance and time frame; then the bid price of the product or service is calculated based on the MCNSS. The S-agent's offer is then transferred by MCNSS to potential B-agents. A number of past m-commerce scenarios are stored in the MCNSS knowledge base. It uses the similarity index, or SI (formula below), to select the candidate case that fits most closely. Once a case has been chosen, a price offer is made that approximates the price information of the selected case. The formula is depicted in Eq. (2.1) as follows:

$$\mathrm{SI}_i = \sqrt{\sum_{j=1}^{n} \left(N_j - S_{ij}\right)^2} \tag{2.1}$$

where N_j indicates jth attribute value of a new case ($j = 1,2,\ldots,n$), and S_{ij} denotes jth attribute value of ith case in the case base of knowledge base ($i = 1,2,\ldots,m$). NetLogo source code for implementing the knowledge base function of MCNSS using SI is listed in Table 2.1. $N_1,\ldots,N_4$ are attributes of cases (N_1: Current inventory level; N_2: remaining period of validity; N_3: freshness level; N_4: number of potential buyers within reasonable distance from store). Table 2.1 describes how the knowledge base of the MCNSS is coded with NetLogo.

Table 2.1 Knowledge base function of MCNSS

```
to change-knowledge base-price
locals [temp_t temp_i temp_si temp_optimal_si temp_item]
set temp_t(1)
repeat seller_number [
ask seller with [reg_number = temp_t and mobile_service = 1 and MCNSS_service =
1 ]
set temp_i (0)
repeat length knowledge base_price [
set temp_si sqrt((N1_Current_Value - item (temp_i) knowledge base_N1_List) ^ 2
+ ((N2_Current_Value - item (temp_i) knowledge base_N2_List) ^ 2
+ (N3_Current_Value - item (temp_i) knowledge base_N3_List) ^ 2
+ (N4_Current_Value - item (temp_i) knowledge base_N4_List) ^ 2)
if (temp_i = 0)[set temp_optimal_si (temp_si) set temp_item (temp_i)]
if (temp_i >= 1) [ if(temp_si < temp_optimal_si) [set temp_optimal_si (temp_si)
set temp_item (temp_i )]
]
set temp_i (temp_i + 1)  ]
set offer_product_price  (item temp_item knowledge base_price)
]
set temp_t(temp_t + 1)
]
end
```

2.4.3 B-Agent

B-agents represent buyers' preferences to be necessarily satisfied during m-commerce transactions with sellers. Specifically, we assume B-agents wish to maximize their own utility in the process of negotiating with S-agents. The utility function is assumed to incorporate relevant factors that are weighted appropriately. Equation (2.2) shows the utility function used in the MCNSS where $i = 1,2,\ldots,m$ (number of sellers), and $j = 1,2,\ldots,n$ (number of utility factors):

$$U_i = \sum_{j=1}^{n} W_{ij} \cdot F_{ij}, \qquad \alpha_{ij} \le F_{ij} \le \beta_{ij} \tag{2.2}$$

U_i denotes ith buyer's utility, W_{ij} buyer's preference for jth utility factor, and F_{ij} ith buyer's jth utility factor. It is certain that $\sum_{j=1}^{n} W_{ij} = 1$. Examples of utility factors include not only price, product, and quality, but also contextual information such as the buyer's current location and environmental constraints. As shown in Table 2.5, we consider five factors including F_1: Distance, F_2: Freshness, F_3: Category, F_4: Price, and F_5: Timeliness. Therefore, five weights W_1–W_5 are assigned to each factor. NetLogo source code for calculating the B-agents' utility function is described in Table 2.2.

Table 2.2 B-agent's utility calculation

```
to Calculate-Util
set temp (1)
set temp_id (1)
repeat customer_number [
ask customer with [reserve != 1 and id_number = temp_id] [
set temp_distance (p_distance )
set temp_price (p_price )
set temp_time (p_time )
set temp_customer_x (current_x) set temp_customer_y (current_y)
set utility (0)
set temp_selected_seller (0)

repeat seller_number [
 ask seller in-radius-nowrap (remaining_time / time_per_patch)
    with [available_product_number > 0 and reg_number = temp1][
set actual_distance (abs (sqrt((temp_customer_x - location_x) ^ 2
+ (temp_customer_y - location_y) ^ 2 ) ))
Convert_factor_point

set temp_util (   temp_W1 * temp_point_F1
                + temp_W2 * temp_point_F2
                + temp_W3 * temp_point_F3
                + temp_W4 * temp_point_F4
                + temp_W5 * temp_point_F5)
if (temp_util > utility) [ set utility (temp_util)
                    set temp_selected_seller (reg_number)   ]
                                ]
                set temp1 (temp1 + 1)] set temp1 (1)
                set utility_without_mobile (utility)
                set normal_selected_seller (temp_selected_seller)   ]
                set temp_id (temp_id + 1)
                  ]
end
```

2.4.4 Negotiation Mechanism

The MCNSS coordinates calculations to satisfy the interests of both S-agents and B-agents. To perform this task, the MCNSS checks price offers by S-agents, the list of potential B-agents, and counteroffers by B-agents in response to initial price offers by S-agents. The core of these negotiation processes is that MCNSS continually locates S-agents that maximize B-agents' utility and locates B-agents that can maximize profit for the S-agents. Until this objective is met, the MCNSS continues to perform its negotiation processes. NetLogo code used to program the negotiation process of MCNSS is summarized in Table 2.3.

Table 2.3 MCNSS negotiation process

```
---------------------------------------------------------------------------------------------
to give-offer-to-MCNSS-buyer
locals [temp]
ask customer [set   negotiating_shop_list []]
ask shop with [unsold_product_number > 0   and mobile_service = 1   ]
  [
  set temp (reg_number)
  without-interruption [
  foreach negotiating_customer_list [
                ask customer with [id_number = ? and reserve != 1 and mobile_service = 1]
                                        [set   negotiating_shop_list lput temp
negotiating_shop_list   ]
                                                ] ]
  ]
end

to nego-offer-to-shop [remaining_time]
locals [temp temp_id temp_seller_no temp_no actual_distance ]
set temp_no (1)
ask shop [set mss_nego_custom_list []]
repeat customer_number [
ask customer with [id_number = temp_no][
      if ( reserve != 1 and MCNSS_service = 1) [
        set temp_id (id_number)
            ask shop in-radius-nowrap (remaining_time / time_per_patch) with
[unsold_product_number > 0   and MCNSS_service = 1]
                          [set MCNSS_nego_custom_list lput temp_id
MCNSS_nego_custom_list
                                                        ] ] ]
                                  set temp_no (temp_no + 1)   ]
end
---------------------------------------------------------------------------------------------
```

2.5 Source Code Analysis of MCNSS

NetLogo source code for the MCNSS is composed of numerous subprocedures. In consideration of space, I will describe the key contents needed for clear understanding of how the MCNSS works.

First, all programs must declare variables, which can be global or local. Global variables work throughout the whole program. Therefore, the programmer has to clearly view which variables are necessary globally. Careful attention must be placed on global variables, because they affect entire components of the program. Non-global variables, or local variables, function only in a specific procedure. Table 2.4 represents global variable declaration, where variable name follows its basic functionality.

Second, customers' features must be represented in MCNSS. This is also true with shops. Therefore, I introduce the parameters of customer-owned and shop-owned. Each variable included in customer-owned and shop-owned represents a feature of customers and shops (Table 2.5).

Table 2.4 Globals

```
-------------------------------------------------------------------------------------------
Globals [Time  variable_price move_time  iteration  offer_time total_customer
total_number_of_MCNSS  mobile_without_MCNSS  total_utility  total_mobile_utility
MCNSS_utility  mobile_without_MCNSS_utility  without_mobile_utility  average_total
average_normal  average_mobile average_MCNSS
average_mobile_without_MCNSS  m_without_MCNSS
customer_not_decide  temp_freq ]
-------------------------------------------------------------------------------------------
```

Table 2.5 Customer-own

```
-----------------------------------------------------------------------------
customer-own [
              p_distance  p_price  MCNSS_service  normal_selected_shop
              MCNSS_selected_shop  utility_without_mobile  utility_with_mobile
              utility_with_MCNSS target_price_down final_utility  final_shop_selected
              utility_adjustment  adjusted_utility  decision_time  id_number
              current_x current_y  negotiating_shop_list ]
shop-own [
              product_price  shop_offer_price  product_unit_number
              unsold_product_number origin_unsold_product_number  product_category
              freshness MCNSS_service  negotiating_customer_list
              MCNSS_nego_custom_list  additional_revenue  reg_number
              mobile_service  total_cost  total_revenue  offer_product_price1
              offer_product_price2  offer_product_price_selected  location_x location_y
              unsold_percent  initial_unsold  CBR_customer_in_range
              CBR_unsold_product  CBR_freshness  CBR_price ]
-----------------------------------------------------------------------------
```

Third, key players such as shop and customer need to be viewable visually in the NetLogo program. Such a visual display of the main players enhances both understandability and joyfulness of ABM users. NetLogo provides several functions to display shops visually in the small world screen, which is located at the center of NetLogo's interface tab as depicted in Fig. 2.1.

When the features of both shops and customers are properly coded, the shop is displayed in the form of a house because NetLogo provides code command *set shape* "*house*" as described in *ask shop* in the below *to setup-shop*. Check this in the codes included in *to setup-shop* (refer to Table 2.6). You can choose the color of the shop. I chose red for the shop and used the code *set color red*. Figure 2.2 shows how setup-shop produces agents representing shops with red a house form.

Similarly, customers' features are specified by using *to setup-customer* as shown in Table 2.7. Customer is represented in a "person shape" with the color gray, which can also be checked in Fig. 2.2.

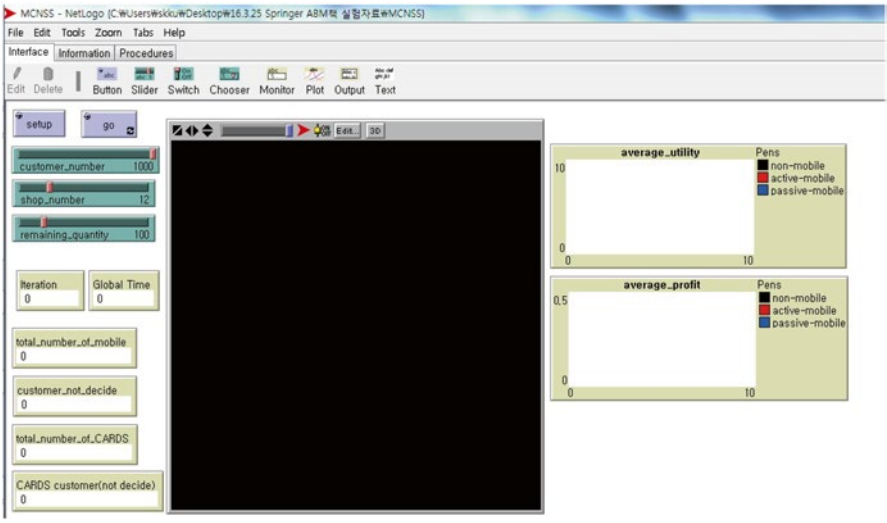

Fig. 2.1 Initial screen of MCNSS

Fourth, customers using the MCNSS must receive offers from the MCNSS, which continues to negotiate with shops until an ideal match is made. NetLogo source code for this function is in Table 2.8.

Fifth, sellers (or shops) make the decisions for whether a customer's offer is acceptable. Counteroffers can be made through a series of offers and counteroffers between sellers and customers. Such a negotiation process is facilitated by the MCNSS. Table 2.9 shows the related NetLogo codes.

Sixth, utility needs to be computed for each seller and customer for the sake of terminating the negotiation process. NetLogo source code for the calculation of utility is described in Table 2.10.

2.6 Experimenting MCNSS Using a Sample Problem

2.6.1 Sample Problem Description

Let us describe a scenario in which a sample problem mentioned by MCNSS is applied. In this sample problem, our logical assumptions are that buyers want to buy perishable products at a discounted price if possible, while sellers want to sell those perishable products before their freshness deteriorates further. Here, the MCNSS can be sold to the buyers and sellers as one of many telecommunication service products. Therefore, those buyers and sellers who register to use the MCNSS can negotiate m-commerce terms with one another.

Table 2.6 To setup-shop

```
---------------------------------------------------------------------------------------------
to setup-shop
locals [temp    reg_temp    random_temp]
set temp (1)
set reg_temp (1)

create-shop(1)
ask shop [ set shape "house"
                    set color red
                    set mobile_service (0)
set product_unit_number (remaining_quantity)
                    set unsold_product_number (remaining_quantity)
                    set initial_rev (0)
                    set initial_unsold (unsold_product_number)
                    set origin_unsold_product_number (unsold_product_number)
set reg_number (reg_temp)
                    set reg_temp (reg_temp + 1)
                    rt random 360 fd random 20
                    set location_x (xcor-of self)
                    set location_y (ycor-of self)
                   set negotiating_customer_list []
                   set MCNSS_nego_custom_list []
                   set CBR_customer_in_range []
set CBR_freshness []
                    ]
set temp_type (1)
repeat (shop_number - 1)[
ask one-of shop [ hatch (1) [
        set shape "house"
        if (temp_type = 0 ) [set color red set mobile_service (0)]
        if (temp_type = 1 )
[set mobile_service (1) set mobile_without_MCNSS(mobile_without_MCNSS + 1)
set color blue]
if (temp_type    = 2 )
[set mobile_service (1) set MCNSS_service (1)
set total_number_of_MCNSS (total_number_of_MCNSS + 1)
set color yellow    ]
set product_unit_number    (remaining_quantity)
set reg_number (reg_temp)
set unsold_product_number (remaining_quantity)
set initial_unsold (unsold_product_number)
set origin_unsold_product_number (unsold_product_number)
set reg_temp (reg_temp + 1)
set negotiating_customer_list []
set    MCNSS_nego_custom_list []
rt random 360 fd random 20
set location_x (xcor-of self)
set location_y (ycor-of self)
set CBR_customer_in_range []
set CBR_freshness []
ask shop [while [count shop in-radius 5 > 2] [rt 360 fd 5]    ]

end
---------------------------------------------------------------------------------------------
```

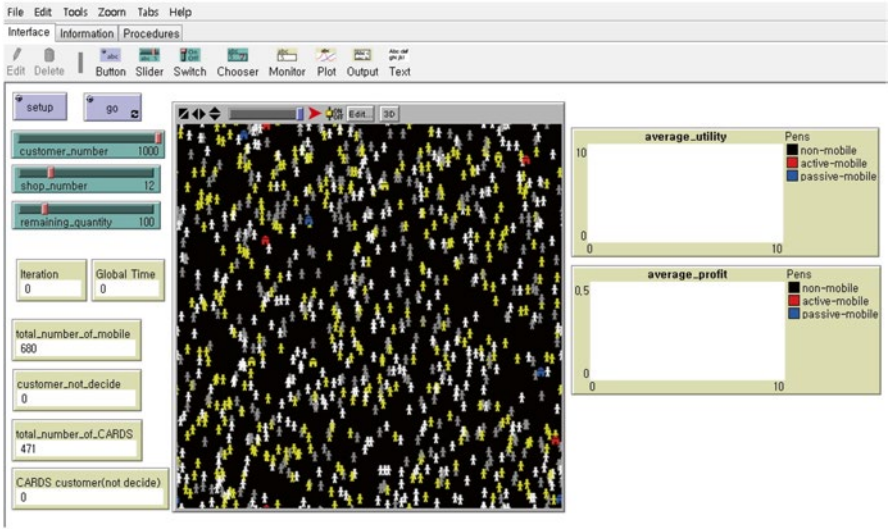

Fig. 2.2 MCNSS screen after setup

Specifically, some of the most popular items that buyers can purchase from the small grocery shops are perishable products, because buyers want to purchase the products in relatively small units, and small grocery shops are easily accessible from buyers' residence areas. In this sense, small grocery shops are usually inclined to perform a pop-bargain sales activity during nighttime operation. Sellers want their sales promotion activity to be advertised to potential buyers. Especially, information about the freshness of perishable products must always be shared with potential buyers, with strict precision and reliability.

For the sake of conducting experiments more logically with the MCNSS, both buyers and sellers are categorized into three groups depending on whether or not they take advantage of the mobile decision support from the MCNSS. The first group, called the "Non-Mobile Group," does not use mobile devices at all and must meet face to face in order to negotiate for products. The members of the second group, the "Passive-Mobile Group," are assumed to use mobile devices, but not the MCNSS, for mobile decision support for their m-commerce activities. Buyers and sellers belonging to this group cannot negotiate directly via MCNSS; instead, each buyer receives information about products from sellers on an advertisement basis, via a telecommunications company contract with sellers. Members of the third group, the "Active-Mobile Group," are assumed to carry mobile devices connected to the MCNSS. They are able to negotiate with one another directly using the MCNSS.

With regard to sellers, it is assumed that each grocery store sells four categories of perishable goods whose freshness level can be categorized into ten levels (level 1 is "Most fresh"). The list price is fixed for each product, but price discounts can be negotiated depending on inventory and freshness levels in addition to the buyer's taste. Price discrimination is decided based on negotiation with buyers. Sellers belonging to the Non-Mobile Group and the Passive-Mobile Group are not able to

Table 2.7 To setup-customer

```
---------------------------------------------------------------------------------------------
to setup-customer
locals [temp id_temp m_temp random_temp temp_random]
set mobile_without_MCNSS (0)
set total_number_of_MCNSS (0)
set total_utility (0)
set without_mobile_utility (0)
set total_mobile_utility (0)
set mobile_without_MCNSS_utility (0)
set MCNSS_utility (0)
set total_customer (customer_number )
set id_temp (1)
create-customer (1)
ask customer [set shape "person"
                    set color gray
                    set mobile_service (0)
                    set decision_time (random-normal mobile_sales_start 10)
                    set p_distance (abs (random-normal 10 10))
                    while [p_price <= 0] [
                    set p_price (abs (random-normal 10 10))]
                    set p_time (abs (random-normal 10 10))
                    set id_number   (id_temp)
                    set id_temp (id_temp + 1)
                    set negotiating_shop_list []
                    rt random 360 fd random 20
                    rt random 360 fd random 20
                    set current_x (xcor-of self)
                    set current_y (ycor-of self)
repeat (int(random-normal customer_number 50)) [
          ask one-of customer [
               hatch 1 [
                    set p_distance (abs (random-normal 10 10))
                    while [p_price <= 0] [
                    set utility_adjustment (random 30)
                    set product_category (random 4)
                    set reserve (0)
                    set random_temp (random 3)
                    if (random_temp = 0 ) [set color gray set mobile_service (0) ]
                    if (random_temp = 1 )
[set color white   set mobile_service (1) set MCNSS_service (0)    ]
                    if (random_temp = 2 )
[set color yellow    set mobile_service (1) set MCNSS_service (1) ]
                    if       (mobile_service       =       1)       [set       total_number_of_mobile
(total_number_of_mobile + 1) ]
                    if (MCNSS_service = 1)
[set total_number_of_MCNSS (total_number_of_MCNSS + 1) ]
                    if (mobile_service = 1 and MCNSS_service = 0)
[set mobile_without_MCNSS (mobile_without_MCNSS + 1)]
                     set decision_time (random-normal mobile_sales_start 10)
 set id_number (id_temp)
                     set id_temp (id_temp + 1)
                     set negotiating_shop_list []
                     rt random 360 fd random 20
                     set current_x (xcor-of self)
                     set current_y (ycor-of self) ]]]
 repeat   3 [random-move-customer]
 end
 ---------------------------------------------------------------------------------------------
```

Table 2.8 To give-offer-to-MCNSS-customer

```
---------------------------------------------------------------------------------------------

to give-offer-to-MCNSS-customer
correction
locals [temp]
ask customer [set    negotiating_shop_list []]
ask shop with [unsold_product_number > 0    and mobile_service = 1    ]
   [
   set temp (reg_number)
   without-interruption [
   foreach negotiating_customer_list [
                    ask customer with [id_number = ? and reserve != 1 and mobile_service = 1]
                                        [set                negotiating_shop_list        lput        temp
negotiating_shop_list    ]
]]]
End
---------------------------------------------------------------------------------------------
```

Table 2.9 To shop-decision-for-MCNSS-customer-offer

```
---------------------------------------------------------------------------------------------

locals   [temp    temp2    p_temp    temp_MCNSS_utility    temp_final_utility    temp_price_down
temp_new_price temp_i]
set temp (0)
set temp_i (1)
repeat customer_number [
ask customer with [id_number = temp_i ] [
                if (reserve != 1 and MCNSS_service = 1) [
                   if ( utility_with_MCNSS >= utility_without_mobile) [
    set final_utility (utility_with_MCNSS)
    set final_shop_selected (MCNSS_selected_shop)
    set temp2 (MCNSS_selected_shop)
  ask shop with [reg_number = temp2 ][
  if (unsold_product_number > 0) [
      set unsold_product_number
  (unsold_product_number - 1)
set temp (1)    ask customer with [id_number= temp_i] [ set reserve (1)    set color black         ]
set additional_revenue (additional_revenue+ offer_product_price2)
]
  if ( utility_with_MCNSS < utility_without_mobile) [
change-mobile-price
set final_utility (utility_without_mobile +(utility_adjustment / 100) * utility_without_mobile)
          set final_shop_selected (MCNSS_selected_shop)
set temp2 (MCNSS_selected_shop)
set p_temp (p_price )
          set temp_MCNSS_utility (utility_with_MCNSS)
          set temp_final_utility (final_utility)
set temp_price_down    int ((temp_final_utility - temp_MCNSS_utility) / p_temp) ]

if (temp_new_price < 0 ) [set temp_new_price (1000)]
set unsold_product_number
(unsold_product_number - 1)
set temp (1)
ask customer with [id_number = temp_i ][set reserve (1)    set color black]
set additional_revenue (additional_revenue +    temp_new_price)
]]

end
---------------------------------------------------------------------------------------------
```

Table 2.10 To calculate-utility

```
---------------------------------------------------------------------------------------

to calculate-utility
locals [temp_id]
set temp_id (1)
ask customer [if reserve = 0 [set final_utility (0)]]
repeat customer_number [
ask customer with [id_number = temp_id ] [
                if (reserve = 1) [set total_utility (total_utility + final_utility) ]
                if (reserve = 1 and mobile_service = 0)
[ set without_mobile_utility (without_mobile_utility +final_utility)]
                if (reserve = 1 and mobile_service = 1 and (MCNSS_service = 0 or
MCNSS_service = 1))
[ set total_mobile_utility (total_mobile_utility + final_utility) ]
                if (reserve = 1 and mobile_service = 1 and MCNSS_service = 0)
[ set mobile_without_MCNSS_utility (mobile_without_MCNSS_utility + final_utility) ]
                if (reserve = 1 and MCNSS_service = 1 )
[ set MCNSS_utility (MCNSS_utility + final_utility + random-normal 5 2) ]]
set temp_id (temp_id + 1)]
end
---------------------------------------------------------------------------------------
```

negotiate directly with buyers using mobile devices; however, sellers in the Passive-Mobile Group can send information about their products to potential buyers randomly, via a telecommunications company server. MCNSS provides knowledge base support to each seller to help determine price discounts for each potential buyer. It uses five attributes: current inventory level, remaining period of validity, freshness level, number of potential buyers within reasonable range of distance from store, and possible price. Active-Mobile Group buyers attempt to negotiate price with sellers using the decision support of the MCNSS. Sellers are assumed to refer to another kind of knowledge base support, in which a case is composed of four input attributes (price gap, expected profit, remaining period of validity, and inventory level) and one output attribute (1 for "Accept the price offer", 0 for "Reject the price offer").

Meanwhile, for the buyers, distance to the possible sellers' locations matters. For this reason, distance factor D must be considered when setting up the buyer's utility function. Also, product freshness, product category, and price also matter to the buyers. In this sense, a product's freshness level (F) and category (C) need to be incorporated into the buyer's utility function. Price (P) is, of course, another crucial factor affecting the potential buyer's intention to purchase a product. We assume that in the case of the Active-Mobile Group, price can be negotiated using the MCNSS. Finally, timeliness (T) indicates how much the potential buyer needs the product at the very moment when a negotiation starts. By incorporating these five factors, the buyer's utility function can be formulated as shown in Eq. (2.3) for each buyer, where i denotes ith buyer, and weight for each factor depends on its relative importance level:

$$U_i = w_{D_i} \cdot D_i + w_{F_i} \cdot F_i + w_{C_i} \cdot C_i + w_{P_i} \cdot P_i + w_{T_i} \cdot T_i \tag{2.3}$$

2.6.2 Simulation

We performed 40 simulation rounds using the assumptions stated previously. The number of S-agents was set as 12, with three stores in each Group (Non-Mobile Group, Passive-Mobile Group, and Active-Mobile Group). B-agents were generated randomly using normal distribution functions, with a mean of 1000 and a standard deviation 200. NetLogo is very easy to install and to operate; users can develop their own ABM models using NetLogo with relative ease. One of the advantages of NetLogo is that there is an extensive "models library" to which users can when necessary. Such a models library is available freely at https://ccl.northwestern.edu/netlogo/models/index.cgi. Parameters of the model are easily changed using graphical "sliders." For those who want to alter the details in a model, clicking on the "procedures" tab brings up the entire model code, which can easily be modified in order to extend the sample models. MCNSS has six types of user interface components, as shown in Fig. 2.3.

- Area 1 is called "Behavior space" showing the customer (or buyer)'s movement and the location of stores. The human shape indicates a buyer and the house shape represents a store. The color gray means group 1, green group 2, and pink group 3. All customers are designed to move one unit of position over to the random direction at a time. The customers leave the simulation after they have purchased products.
- Area 2 is "Graph monitors" graphically displaying the change of values such as number of customers, product inventory, the number of customers who have not bought products, the customer's average utility, and the store's average profits, etc.
- Area 3 denotes "Control button" which prepares and prompts simulation.
- Area 4 represents "Slider" which controls the initial conditions of simulation such as number of customers, number of sellers, etc.
- Area 5 is for "Monitor" showing the number such as rounds of simulation and simulation time.
- Finally, Area 6 is for "Command center" showing temporary data generated from the agent activities.

For the sake of simulating the MCNSS, we use specific information about the buyer's utility function as described in Table 2.11. This is effective, because we need some parameters to specify S-agents and B-agents. In this sense, S-agents and B-agents are randomly located on a virtual two-dimensional space provided by NetLogo, where the X-axis, as well as the Y-axis, ranges from −20 to 20. Each B-agent is given weights w_i for the five utility factors mentioned previously on a random basis ($\sum_{i=1}^{5} w_i = 1.0$). Similarly, B-agents are given a product category (1–3), a potential purchase period (1–50 min), and a minimum utility level. For example,

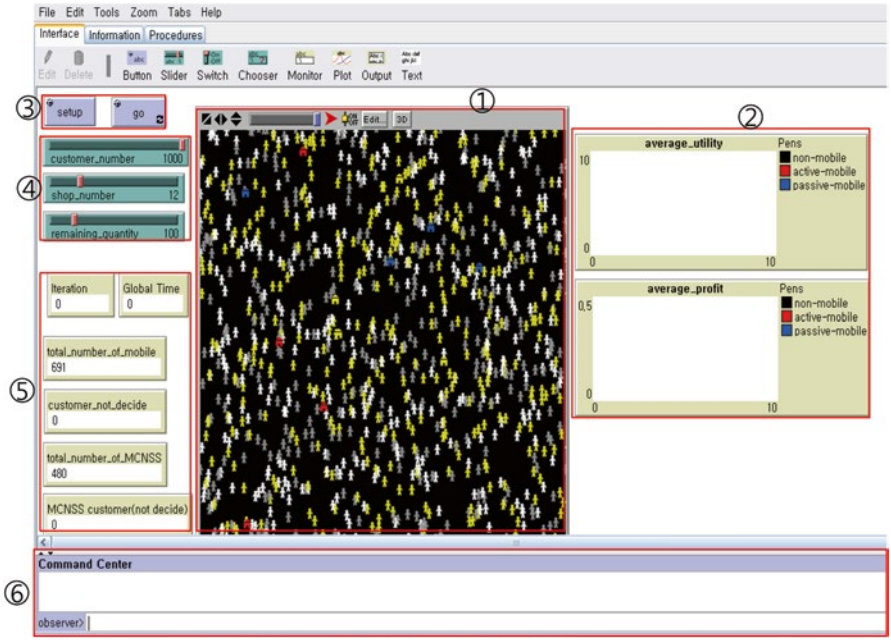

Fig. 2.3 Components of user interface in MCNSS

Table 2.11 Buyer's utility

Utility factor	Condition	Converted utility
Distance (*D*)	Within 20 min	50
	Within 30 min	40
	Within 40 min	30
	Within 50 min	20
	More than 60 min	10
Freshness (*F*)	1,2	50
	3,4	40
	5,6	30
	7,8	20
	9,10	10
Category (*C*)	Preferred category	50
	Otherwise	0
Price (*P*)	Price negotiation	50 − (new price/list price) × 50
Timeliness (*T*)	If the buyer wants the product on offer	50
	Otherwise	0

the B-agent with a potential purchase period of 30 min indicates that the buyer thinks 30 min after the start is the most appropriate purchase time. When the buyer purchases the target product at the desired time, we assume that the utility will increase. Otherwise, the buyer's utility is assumed to possibly decrease. Ultimately, buyers are believed to purchase the product when the utility calculated by the seller's offer is greater than the minimum utility level.

Simulation restarts another round if met with one of the stopping conditions, such as: (1) negotiation is not available anymore, (2) seller's inventory is out of stock, (3) all potential buyers purchased the products, and (4) 5 min pass without a successful negotiation. At the start of a new round of simulation, all the B-agents and S-agents are given new information as mentioned previously. S-agents are looking for the B-agents who are mobile within 20 distance by using the "In-Radius" command. Also, S-agents determine an initial bid price based on the knowledge base and deliver the price to the potential B-agents. In response to the S-agents' offers, the B-agents calculate their own utility based on the information in Table 2.11. If the calculated utility is greater than the minimum level, then the B-agent accepts the offer and reserves the target product, moving to the S-agent's shop to purchase the products. Otherwise, the B-agents assigned to Active-Mobile Group adjust price based on the price adjustment formula, $P_{buyer_i} = \dfrac{U_{\min} - \left(w_{D_i} \cdot D_i + w_{F_i} \cdot F_i + w_{C_i} \cdot C_i + w_{T_i} \cdot T_i\right)}{w_{P_i}}$ which is derived from the utility function $U_i = w_{D_i} \cdot D_i + w_{F_i} \cdot F_i + w_{C_i} \cdot C_i + w_{P_i} \cdot P_i + w_{T_i} \cdot T_i$, and then send the adjusted price to the S-agents as a counteroffer. By consulting with the knowledge base, S-agents determine whether or not to accept the counteroffer. If the output attribute is 1, then the S-agents accept the counteroffer. Otherwise, the counteroffer is rejected.

When the iterations run for 40 rounds, average utility and average profit curves are derived as shown in Fig. 2.4. Since ABM depends on the emergence results as stopping criteria, we need to check whether the average profit and average utility represent what we had in mind about the target problem. The average utility and profit play as emergence results, which are revealed in a macroscopic figure, after a series of microscopic interactions among agents involved in the ABM simulations. How many simulation iterations that are necessary for decision makers to make a final decision depends on the complexity of the target problem. However, one clear criterion is to check whether continuity is revealed in the results. Let us explain this by discussing Fig. 2.5.

Figure 2.5 displays average utility and average profit, which are set up as macroscopic emergence results for this target problem. An example of peculiar continuity in Fig. 2.5 is that the Active-Mobile Group, displayed in red, outperforms the other two groups, such as Non-Mobile and Passive-Mobile Groups, in terms of average utility and profit.

Though visual inspection assures us that the average utility and profit computed from the Active-Mobile Group outperform other mobile groups, it is safe for decision makers to double-check the numeric figures.

Let us first discuss the average utility for the three mobile groups. Table 2.12 summarizes the average utility figures for the three mobile groups such as

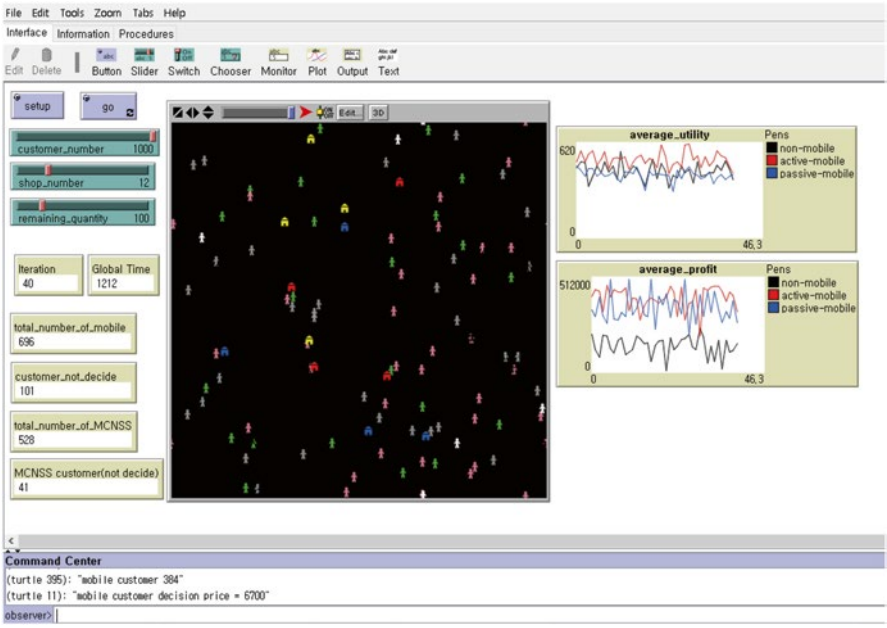

Fig. 2.4 MCNSS screen after 40 iterations

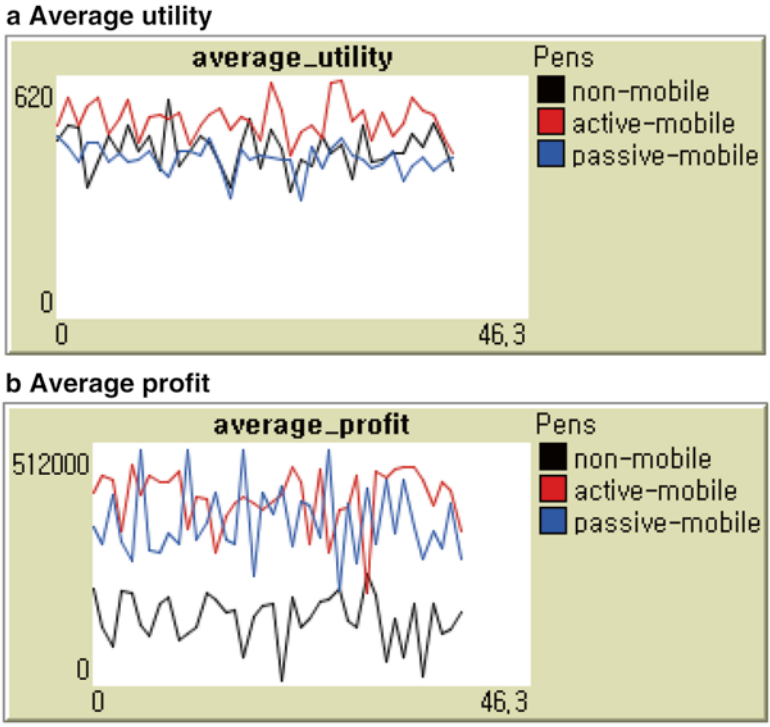

Fig. 2.5 Average utility and average profit

Table 2.12 Average utility for the three mobile groups

Time	Non-Mobile	Passive-Mobile	Active-Mobile
1	456.95	466.24	495.60
2	495.30	441.06	563.26
3	489.91	402.49	491.60
4	334.38	450.64	544.62
5	399.75	448.94	566.47
6	466.00	401.94	472.43
7	420.26	423.14	508.36
8	492.47	402.74	557.27
9	428.51	405.53	447.79
10	465.48	426.16	514.74
11	380.07	382.30	522.36
12	560.19	364.20	508.98
13	390.46	425.33	527.35
14	426.29	428.26	446.22
15	468.14	415.71	493.50
16	450.61	462.89	523.20
17	393.05	396.61	535.46
18	332.41	304.57	481.65
19	445.18	432.27	517.66
20	507.66	404.33	505.04
21	384.98	418.63	457.96
22	481.59	411.25	605.37
23	431.78	407.54	532.08
24	326.14	404.55	416.37
25	405.24	300.73	475.81
26	387.38	439.90	494.05
27	466.21	385.73	462.85
28	423.27	435.07	602.52
29	444.91	463.35	607.21
30	358.42	415.80	502.72
31	495.39	403.91	531.96
32	398.55	385.90	457.55
33	408.67	396.54	525.19
34	420.22	428.66	466.12
35	422.76	351.93	500.38
36	471.94	391.47	564.73
37	438.41	410.71	534.52
38	500.12	376.06	521.94
39	443.16	400.18	462.94
40	378.67	411.23	424.04

Non-Mobile, Passive-Mobile, and Active-Mobile Groups. Average utility is 432.27 for the Non-Mobile Group, 408.11 for the Passive-Mobile Group, and 509.25 for the Active-Mobile Group. *t*-Test results are also clear. The *T*-value between Non-Mobile and Passive-Mobile is 2.43, which is statistically significant at the 99 % confidence level. Similarly, the *T*-value between Passive-Mobile and Active-Mobile is 10.9, which is also significant at the 99 % confidence level. The *T*-value between Active-Mobile and Non-Mobile is 7.0, which is also significant at the 99 % confidence level. In conclusion, the Active-Mobile Group shows best performance in average utility.

A *t*-test was also applied to average profit. Table 2.13 displays the average profit for the three mobile groups. Average profits for Non-Mobile, Passive-Mobile, and Active-Mobile Groups are 141,162.5, 347,039.6, and 397,747.7, respectively. The

Table 2.13 Average profit for the three mobile groups

Time	Non-Mobile	Passive-Mobile	Active-Mobile
1	202,500	333,750	405,664
2	122,000	300,350	442,455
3	83,500	403,750	435,468
4	200,750	301,666	328,189
5	195,500	262,800	465,156
6	127,250	500,000	402,439
7	106,250	285,233	444,648
8	172,750	280,033	428,336
9	185,000	323,700	431,852
10	95,750	299,600	452,847
11	108,000	500,000	328,676
12	122,000	308,166	398,942
13	193,750	345,975	392,597
14	179,750	408,950	280,623
15	155,250	307,366	356,024
16	160,500	298,333	383,862
17	59,000	500,000	399,633
18	143,000	230,200	384,385
19	167,500	408,966	370,356
20	171,000	364,266	390,722
21	8,250	420,633	404,550
22	185,000	296,500	462,473
23	122,000	391,500	432,424
24	139,500	380,950	300,713
25	178,000	310,875	455,927
26	179,750	500,000	280,702
27	206,000	197,400	373,641
28	134,250	384,500	376,211
29	122,000	258,400	445,818

(continued)

Table 2.13 (continued)

Time	Non-Mobile	Passive-Mobile	Active-Mobile
30	237,500	414,900	193,851
31	190,250	296,900	454,466
32	50,250	435,225	441,530
33	141,250	323,700	458,886
34	57,250	435,225	460,627
35	171,000	333,633	460,222
36	18,750	265,250	434,071
37	172,750	324,525	380,615
38	109,750	292,133	429,763
39	116,750	387,333	412,887
40	155,250	268,900	327,657

T-values show that there is a significant difference among the average profits for the three mobile groups. Accordingly, we can conclude that the Active-Mobile Group is the best alternative in terms of average profit and utility.

Implications are as follows: first, multi-agents handled by ABM are both convenient and effective for m-commerce entities when MCNSS are used to manage the decision-making process. This is because agents are capable of autonomous operation once the entity's preference is predefined and stored into their memory. In the MCNSS environment, therefore, users do not have to interact directly with negotiation partners. Second, both preferences and conditions that users want their agents to consider in the process of negotiation coordination can be easily incorporated into the agents. Third, since the narrow screen limits m-commerce users and specified functions of their mobile devices, and since agents are capable of replacing users in the real negotiation process in an almost automatic manner, the use of a negotiation coordination mechanism, such as MCNSS, would significantly contribute to enhancing users' utilities and profits.

http://www.springer.com/978-3-319-39989-8

Digital Creativity Model and Its Relationship with Corporate Performance
Emphasis on Agent-Based Modeling Approach

Lee, K.C.

2016, IX, 70 p. 9 illus. in color., Softcover
ISBN: 978-3-319-39989-8