

# Synthetic Environment for Analysis and Simulation: A Case Study of Market and Supply-Chain Co-design in the PC Industry<sup>1</sup>

Alok R. Chaturvedi, Director  
[alok@purdue.edu](mailto:alok@purdue.edu)

&

Shailendra Raj Mehta, Co-Director  
[mehta@mgmt.purdue.edu](mailto:mehta@mgmt.purdue.edu)

Purdue e-Business Research Center  
Indiana Consortium for e-Business Research

Krannert Graduate School of Management  
Purdue University  
West Lafayette, IN 47907  
(765) 494-9048  
(765) 494-1526 (fax)

---

<sup>1</sup> This research was funded by National Science Foundation grants # EIA-0075506 and DMI-0122214. As required by the Memorandum of Understanding between the authors and Purdue University, it is disclosed that some or all of the intellectual property described herein may be commercialized.

# Synthetic Environment for Analysis and Simulation: A Case Study of Market and Supply-Chain Co-design in the PC Industry<sup>2</sup>

## Abstract

The Synthetic Environment for Analysis and Simulations (SEAS) developed at the Krannert Graduate School of Management at Purdue University mimics real life economies, with multiple interlinked markets, multiple goods and services, multiple firms and channels and multiple consumers etc. all built from the ground up. It is populated with human agents who make strategically complex decisions and artificial agents who make simple but detail intensive decisions. These agents can be calibrated with real data and allowed to make the same decisions in this synthetic economy as their real life counterparts. The resulting outcomes can be surprisingly accurate. This paper discusses the research in this area and goes on to detail the architecture of SEAS. It also presents a detailed case study of the application of SEAS for business-to-business e-commerce in the PC industry.

---

<sup>2</sup> This research was funded by National Science Foundation grants # EIA-0075506 and DMI-0122214.

# A Introduction

The fundamental tenet of business-to-business e-commerce is disintermediation and relatively “frictionless” economies. The advent of e-commerce has shrunk the competitive “time to market” window for new products substantially, particularly in the area of technology products themselves. The pressure to be “first to market” and establish “lock in” is intense for companies in the software, Internet and telecommunications industries. With such compressed timelines, the probability of new product success is oftentimes more a matter of good timing and good fortune than of careful analysis and preparation.

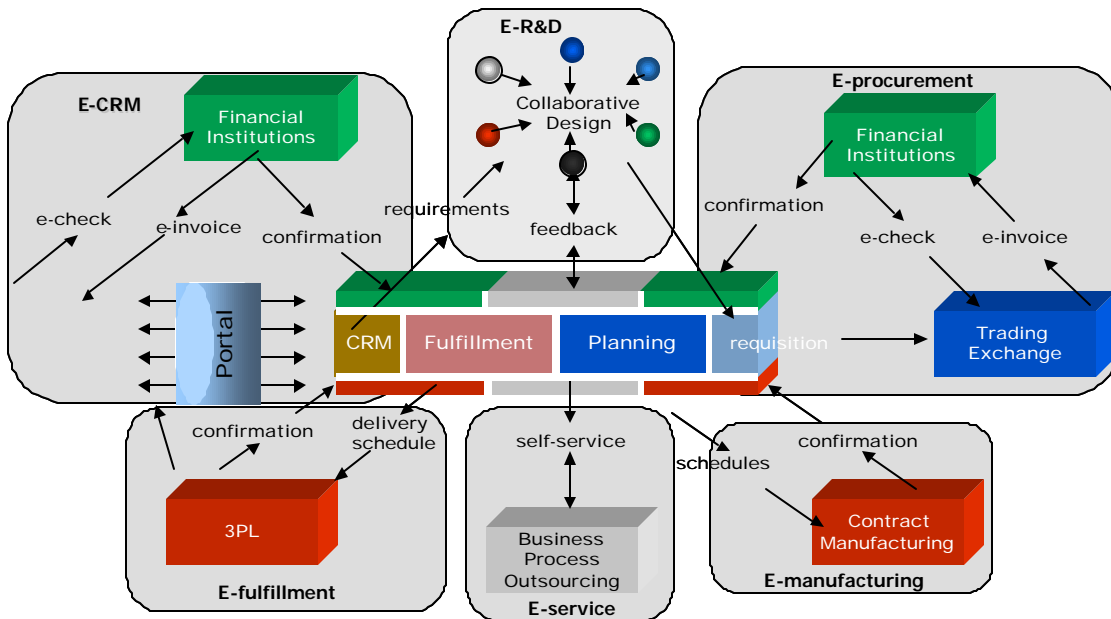


Figure 1: e-Business framework

Intuitively, the notion of co-design as shown in **Figure 1**, that is, integrating all design activities in parallel is an appealing alternative, which a priori one would suspect is Pareto superior to the serial approach. Co-design has been used particularly successfully in designing embedded hardware and software systems.

The paper describes the co-design approach to the problem of designing market and supply chain in the PC industry. This is an intensely competitive market earmarked by rapid change, extremely short product development times, and very short shelf time “half life”. Corporations cannot afford to make very many mistakes in this climate, or they risk losing irretrievable market share. By adopting a co-design approach, we argue that firms have a higher probability of aligning products with markets, and thus maintaining their competitive edge. The co-design approach we use to integrate the convergent technology product design processes is a hybrid of microeconomic and OR modeling, namely experimental economics complemented by agent-based simulation.

We create a synthetic economy to extensively study the computational model of human decision-making in the context of business-to-business (B2B) e-commerce. There are several aspects of B2B e-commerce -- e-CRM, e-procurement, e-R&D, e-fulfillment, e-service, and e-manufacturing as shown in **Figure 1** (Swenson et. al. 1998, Kaplan and Sawhney 2000, Ramsdell 2000, Timmers 1999). We will implement agents and simulate different business models to study the decision making process and create their computational models.

The remainder of the paper is organized as follows: Section B presents the importance of the problem and the background research. Section C describes the concept of synthetic economies and presents the technical architecture of SEAS. Section D presents the case study of application of SEAS to B2B e-commerce in the PC industry. Finally, section E concludes the paper.

## B Importance of the Problem

Understanding electronic marketplaces is one of the key lynchpins for corporations to make the transition from the “old” to the “new” economy. Although many claim that the “old” and the “new” will converge and e-business will become just plain old business as the years unfold, many companies still struggle with the rapid change e-business has wrought on the landscape. The finer granularity of customer relationship management, disintermediation and re-intermediation of the supply chain, business-to-business (B2B) exchanges, and the evolution of agent technology are just some of the phenomena that have emerged from the electronic marketplace (Bailey and Bakos 2000)

Although these processes will undoubtedly be assimilated into the “business as usual” environment at some point, they require new approaches and tools for analysis and modeling to do so. This is particularly true in the area of economics where the standard neoclassical economic model has proven to be inadequate in capturing the dynamic evolution brought about by technological change.

E-Business faces a number of paradoxes as shown in **Figure 2**. These competing issues like the conflict between trading exchanges and customer relationship management, Internet market efficiency and competitive pricing etc, have to be resolved by every e-business to be successful in the market place.

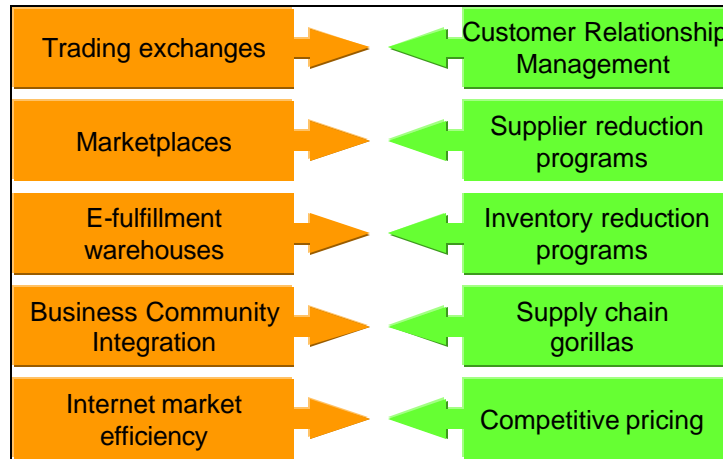


Figure 2: E-business paradox

Such a problem is that of multiple criteria decision-making and decision makers would have to draw upon techniques from various disciplines. In this paper, we discuss the new decision technology of agent-based simulation (ABS) and its role in addressing some of the critical problems that arise in electronic marketplaces. Specifically, we look at ABS as an engine for building synthetic economies that can, in turn, be used as bases of analyses for the e-business phenomena discussed above. The ABS-driven economy, consisting of many thousands of software agents whose emergent behavior defines the marketplace, can be used as a platform with which human players can engage in strategic decision-making simulations. ABS in this context is a hybrid of microeconomic analysis combined with the OR/MS discipline of simulation (although a much different, bottom up kind of simulation than the typical top down discrete event simulation). This approach wherein human players can participate concurrently with an agent-based economy offers the following benefits:

- The seamless integration of human and software agents. This allows significantly more complex experiments to be conducted than are currently possible in the field of experimental economics (Kagel and Roth 1995). These experiments can combine depth of decision-making (using humans) and breadth (combining artificial agents).
- The consequences of decisions can be measured. This extends the purview of traditional decision support from building models that support human decision making to actually being able to gauge the impacts of decisions as well.
- A laboratory for testing the efficacy of decision support tools. Experiments can be devised that measure the effects of various decisions against the support tools used to arrive at those decisions.

For the ABS approach to work effectively, viable virtual economies must be constructed. This requires careful attention to the design and specification of the agents who will populate the economy. In the large, this demands that we be able to access reliable, accurate computational models of human behavior that we can then bestow

appropriately upon our population of agents to achieve a marketplace behavior with acceptable verisimilitude. There is a vast literature of such models from disciplines such as experimental economics, artificial intelligence, cognitive science, psychology, and decision theory. Among the objectives of this paper is to derive a meaningful taxonomy of such models, and then to demonstrate how they can be applied to the construction of an artificial economy.

## B.1 Shortcomings of Classical Economics

The standard economic model is that of *homo economicus*. This is basically a constrained optimization exercise. An agent starts with an objective (maximize profit, maximize utility) and a choice set (defined by a budget constraint or an endowment or input prices), and computes the optimal solution. This solution is then implemented in a setting with certain rules (a market, a bilateral negotiation) and those rules then (perhaps after several iterations) determine an outcome (an allocation, prices). This standard model (Kreps 1990, MasColell, Whinston and Green 1995) forms the basis of much of economics and a significant amount of management modeling.

However, in recent years at least three strands of literature have converged to upset this neat, and somewhat limited view of human behavior. These include experimental economics (Kagel and Roth 1995), learning (Fudenberg and Levine 1999) and behavioral economics (Mullainathan and Thaler 2000). Experimental economics shows that while some parts of economic theory held up reasonably well (such as static markets with bids and asks) others such as individual decision making did not, and were subject to biases, errors and misperceptions. Behavioral economics complements experimental economics quite nicely by looking for field data as opposed to experimental data and showing that many of the same biases are found in real life. Similarly, learning theory assumes that rational behavior does not emerge fully formed but through a great deal of trial and error.

It is worth quoting in detail an excerpt from a recent survey (Mullainathan and Thaler 2000) to underscore just how radical a departure this is. They confine themselves to the field of finance, but their remarks apply much more generally.

“If economists were polled twenty years ago and asked to name the domain in which bounded rationality was least likely to find useful applications the likely winner would have been finance. The limits of arbitrage arguments were not well understood and that time, and one leading economist had called efficient markets hypothesis the best-established fact in economics. Times change. Now as we begin the 21<sup>st</sup> century finance is perhaps the branch of economics where behavioral economics has made the greatest contributions. How has this happened?”

“Two factors contributed to the surprising success of behavioral finance. First, financial economics in general and the efficient market hypothesis in particular, generated sharp testable predictions about observable phenomena. Second, there are great data readily available to test these sharp predictions.”

What has been the consensus? They add:

“The standard economic model of human behavior includes (at least) three unrealistic traits: unbounded rationality, unbounded willpower and unbounded selfishness. These three traits are good candidates for modification.”

What sort of modeling approach might supplement the standard economic model? Clearly a broad based approach is required that incorporates insights from a variety of disciplines, including psychology, management and economics.

## B.2 Agent Behavior

Traditionally, economics, and by extension, large sections of management, have largely been insulated from the rest of the social sciences. However, if the assumptions of the standard model are modified, then the door is opened for importing significant insights from the sister disciplines such as psychology, that have had a long history of studying behavior as observed, as opposed to behavior as deduced, from a set of axioms. Clearly then, there is significant motivation for building agents that engage in specific behaviors as opposed to optimization. One can put them together in increasingly complex environments to see what sort of emergent behavior results (Epstein and Axtell 1996). However, much more sophisticated agents are required. At least four classes of capabilities have to be modeled in detail – including information gathering, information processing, responsiveness and interaction. For example, Downs-Martin (1997) indicates that the information processing part might consist of the following behaviors as listed in **Table 1**:

<b>Actions</b>	<b>Simple Definitions</b>
Acquire	To gain by one's own efforts, to obtain
Alert	To warn to be ready or watchful
Detect	To discover something hidden, to notice, to observe
Discriminate	To distinguish between things
Extract	To deduce or derive, to take out from
Filter	To strain out unwanted data and so forth
Identify	To fix a person or thing as the one described
Inspect	To look at carefully
Localize	To trace to a particular place, discover the position of
Monitor	To watch, check, regulate performance
Recognize	To identify as known before
Orient	To adjust to a particular situation
Perceive	To become aware of via senses, grasp mentally
Queue	To form up in a line
Read	To get meaning by interpreting characters
Receive	To take or get freely given information
Search	To examine carefully for a thing concealed, survey

Table 1: Agent Behavior Verbs

It is important to analyze and classify these behaviors with a view to applying them in a broad range of social disciplines, in a collaborative, modular fashion. One sort of modeling paradigm that might work hand in hand with the new departures from the standard model is agent based modeling techniques.

## C Agent-based Economies and SEAS

Agent based techniques have gained prominence in management and economics (Gode and Sunder 1993, 1997) and this area shows much promise in terms of addressing the three challenges to the standard model, outlined above.

The motivation for agent-based economies is as follows. Rust (1996) argues that researchers are forced to build models which are either stylized enough to be solved analytically or small enough to be solved computationally. Each of these kinds of models is, in Rust's words, a "toy model." Moreover, such models are not modular and cannot be built up cumulatively by large teams of researchers working together, each developing a separate piece of the puzzle. Therefore, Rust suggests that we mimic the intelligence demonstrated in the organization of the market and use decentralized computing along with agents, to study these models. Rust's suggestion is quite appealing. He mentions a number of desiderata for such an agent-based environment. Among them is the ability to seamlessly integrate human and artificial agents in the same environment. It would allow experiments of greater complexity and of a larger scale than those possible with existing software such as the pioneering *MUDA* program developed by Plott and Gray (1990) at Caltech. This would truly enable multi-disciplinary research. Computer-based information systems would be integrated with economics to create synthetic economies that could be used as a common meeting ground for techniques from operations research, management science, psychology, and computer science. This then is the promise of agent-based research.

Of crucial importance is the ability to put human and artificial agents in the same environment. This allow us to leverage human capabilities to make complex decisions (such as running firms or channels) and to leverage the large numbers of artificial agents that can be deployed to capture fine details of market segments. Moreover such mixed environments allow us to calibrate human and artificial agents against each other. For example, can we mimic a particular human behavior (such as trading) in a narrow domain (such as a simple double auction market) by using artificial agents? Similarly can we calibrate the complexity of a decision making task performed by human beings on the basis of the complexity of the artificial agent that might be able to mimic it?

In the past machine learning has primarily been used for classification problems. Now, these techniques are increasingly being used in decision support (Chaturvedi et. al. 1993) real-time system control, production process control; product design and control knowledge (Chaturvedi and Gulati 1993 and Chaturvedi and Nazareth 1998); scheduling and control of flexible manufacturing systems. Now machine-learning techniques are being used to comprehensively model human behavior in markets in the form of shopbots (artificially intelligent agents which help consumer buy goods and services at the best price) (Guttman et. al. 1998, Chavez and Maes 1996, Bichler et. al. 1998, Hedberg 1996) and pricebots (artificially intelligent agents which help sellers determine the optional price) (Chavez and Maes 1996).

The extraction of human knowledge and behavior requires careful planning. New knowledge has to be reconciled with existing rules. Conflicts must be resolved satisfactorily. Likewise, the potential utility of the new rules also

needs to be examined prior to inclusion. Rules that are very specific, and are unlikely to find application, may be discarded. In a similar vein, extremely general rules are unlikely to be included, as they are more prone to generating conflict. This is the balance that needs to be struck and a particular instance of it is described below in the modeling of our consumer artificial agents.

## C.1 Synthetic Environment for Analysis and Simulation (SEAS)

*The Synthetic Environment for Analysis and Simulations (SEAS)* is an interactive synthetic economy that models the critical relationships between economies, markets, product and process innovations, price, and business rules using human and artificial agents. It allows participants to view the economy from different perspectives such as the government, universities, commercial sectors, and the households/consumers/tax payers as shown in Figure 3.

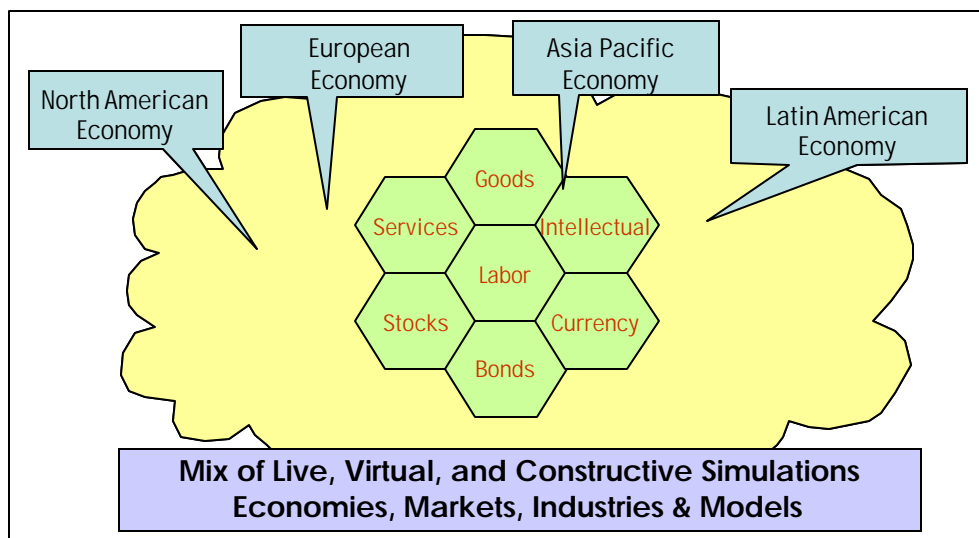


Figure 3: Interlinked markets in SEAS

SEAS computational infrastructure is very sophisticated and pushes the state-of-the-art in agent based computing. We briefly describe the architecture in the following section.

### C.1.1 SEAS Virtual Execution Environment

An internet-based SEAS virtual execution environment (VEE) is depicted in Figure 4. In SEAS' VEE, participants from anywhere can take part in an exercise. VEE consists of three classes of servers: application servers, distributed database servers, and Proxy Servers. Proxy servers ensure restricted access for the subscribers. There are three different classes of application servers. The Agent Processing Servers are capable of running hundreds of thousand of different kinds of agents. Economic Processing Servers are capable of representing different types of markets. Finally, the Visualization Servers generate advanced 3-D displays of the data generated during the exercise.

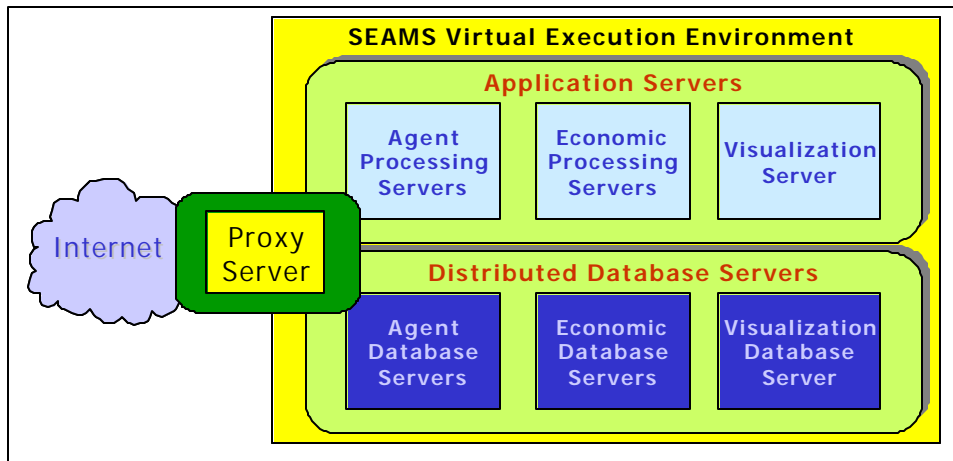


Figure 4: SEAS Virtual Execution Environment

There are individual database servers that support each of these application servers. These database servers may run at one or more locations. Thus, the distributed design enables any numbers of participants to take part in an exercise, and any number of exercises may be available at any given time.

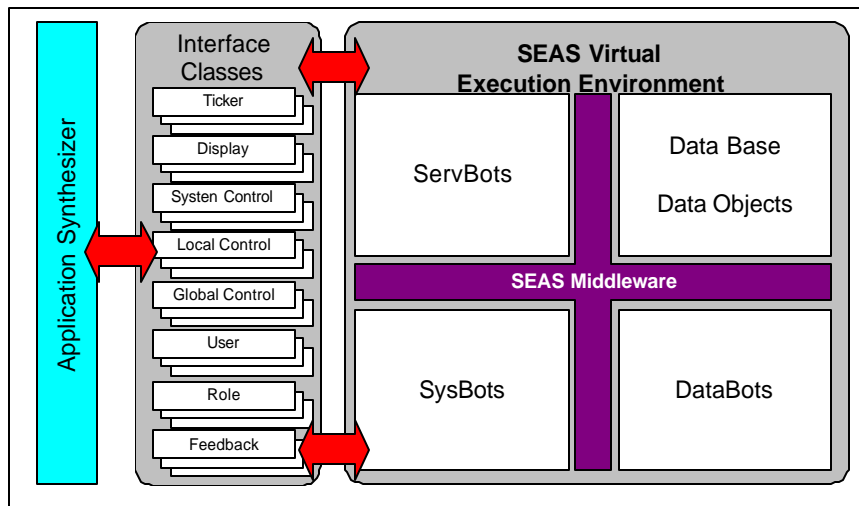


Figure 5: Anatomy of SEAS Virtual Execution Environment

SEAS software environment is highly re-configurable. It consists of three classes of active objects called "bots" as depicted in Figure 5. The ServBots are autonomous agents that perform the business tasks. Examples of ServBots are BuyBot (agents that perform buying function), SellBot, AcquireBot, ProduceBot, etc. These bots also function as application programming interface (APIs) for third party game development.

The second class of bots is SysBots. This class of autonomous agents performs system level tasks such as read, write, update, open, and close connections.

The third class of bots is the data bots. These bots are being developed to interface with enterprise systems so that firms can seamlessly integrate with SEAS to run dynamic Strategeeing exercise to explore new strategy spaces or may use it as a wind tunnel of corporate strategies. These bots interact with the data manger for their data needs using the SEAS middleware.

The VEE interacts with re-configurable interface classes. There are eight types of interface classes -- ticker, display, system control, local control, global control, user, role, and feedback. ServBots, SysBots, and DataBots are dynamically assembled for each of these interface classes based on the participants profiles or demands by an application synthesizer. This architecture provides the necessary flexibility to adapt SEAS to a wide variety of problem domains.

### C.1.2 SEAS Agent Architecture

SEAS use intelligent agents (IA) to represent economic realities of electronic markets and hierarchies in a decentralized manner. SEAS' intelligent agents are autonomous processes that are adaptive and behave like human agents in a narrow domain. In their respective domains, each agent has a well-defined set of responsibilities and authorities so that it can execute its tasks effectively. Examples of SEAS' IAs are: economic agents --consumer IAs, producer IAs, and regulator IAs; political agents -- government IAs, special interest IAs, etc.. An agent in SEAS is equipped with reasoning, action, and communication skills required for performing their respective tasks. A SEAS IA is characterized by the knowledge it possesses. An example of an artificial agent implemented as a thread is given below Figure 6:

<b>Class</b>	: <i>public class Agent extends Thread</i>
<b>Methods used</b>	: The methods used in Agent.java are the following
	1) <i>public void run()</i>
	2) <i>public void updateSeller (int id, MessageArray maResult, double price1, double price2, double price3, int period AgentSpace as)</i>
	3) <i>public void updateBuyer(int id, int trans_qty, double res_price, double exp_price, int sellerId, int period, AgentSpace as)</i>
<b>Implementation Logic :</b>	
	This is the implementation for an Agent as a thread in a transparent setting. In Agent.java, the Agent object is created. The parameters used in he Agent constructor are int <i>totalPeriod</i> , double <i>epsilon</i> , double <i>maxPrice</i> , int <i>id</i> , String <i>type</i> , double <i>price1</i> , double <i>price2</i> , double <i>price3</i> , int <i>qty</i> , SellerPrice[] <i>sellerPrice</i> . In the try block, a factory for the beans is created using <i>InitialContext jndiContext= new InitialContext()</i> . Using the lookup method, the jndiContext looks into the individual folders say <i>buyerInit/BuyerInit</i> and a forms a connection with the corresponding objects are for example, <i>objBuyerInit</i> etc. both for the buyers and sellers. JNDI technology enables the connection between the client and the EJB component.
	The <i>While</i> condition executes as long as the period is less than or equal to the totalPeriod. Within the inner <i>while</i> the buyers keep searching and evaluating the price (price is the reservation price for the buyer and the expected price for the seller). As long as the MessageArray <i>result</i> is not null and there are still some messages in the array, the <i>t_transaction_log</i> is updated. Once there is no more transaction, which is indicated by the value of <i>trans_qty</i> , the transaction for the buyer is over and the flag in the JavaSpace is set to 1.
	The second <i>While</i> condition executes as long as the period is less than or equal to the totalPeriod. Within the inner <i>while</i> , the sellers keep posting the price until all the sellers are done and until all the buyers set the <i>done</i> flag to true. After the sellers complete posting the price in the MessageArray <i>ma</i> , both

buyer and sellers are updated by setting the flags in JavaSpace to 1. The updating is implemented in the methods *updateSeller* and *updateBuyer*. In the *updateSeller* method the profit for each seller is also calculated, depending on the units sold. In the *updateBuyer* method, profit for the buyer is calculated based on the formula “(reservation price - posting price) \* quantity bought”.

- Methods : *updateSeller, updateBuyer*
- Parameteres :
- 1) int id, MessageArray maResult, double price1, double price2, double price3,int period, AgentSpace as)
- 2) int id, int trans\_qty, double res\_price, double exp\_price, int sellerId, int period, AgentSpace as)
- Objects : none
- Variables :
- Return : String

Figure 6: Artificial Agent as a Thread

Each agent consists of seven SysBots that are individually replaceable as shown in Figure 7. There can be several instantiation of each SysBot. SysBots that constitute an agent is described below.

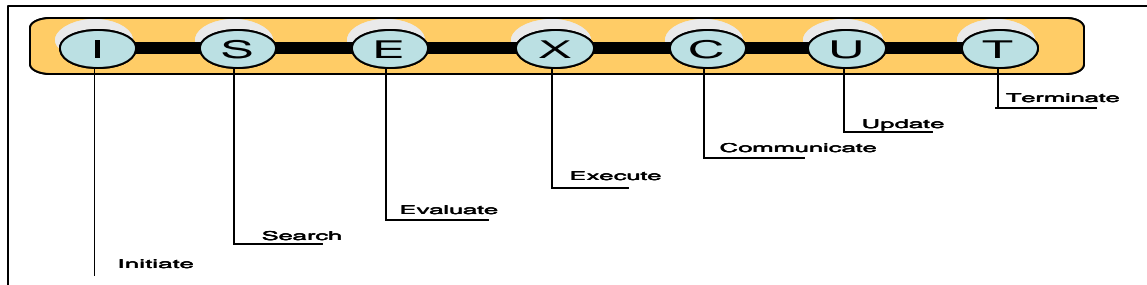


Figure 7: SysBots are coalesced together to form a servBot

- Initiate: these bots are the sensors for a given servbot. When certain conditions are met then they trigger action from the servbot. A simple Java bean example of Initiate process is given below in Figure 8:

```

Class : public class BuyerInitBean implements SessionBean
Method used : public double setPrice(double price1, double price2, double price3, int unit)
Implementation Logic: In this method, the buyer’s reservation prices are initialized. The “setPrice” will set the “ReservationPrice” for the buyer. The “unit” parameter keeps the track of how much the Buyer already has. Depending upon the units, the “ReservationPrice” is set.
  
```

- Package : package com.behavior.buyerInit
- Method : setPrice
- Parameters : double price1, double price2, double price3, int unit
- Objects : none
- Variables : double reservationPrice
- Returns : double

Figure 8: Initiate Bot

- Search: these bots make the servbots intelligent. Each servbot can autonomously search the state space for an appropriate action in its specified domain. The following is an example of search bean (Figure 9):

<b>Class</b>	: <i>public class BuyerSearchBean implements SessionBean</i>
<b>Method used</b>	: <i>public MessageArray Search(AgentSpace as)</i>
<b>Implementation Logic:</b>	In this method, the “buyer” searches all the prices posted by seller in the market. The “MessageArray” is checked for null condition. And if there is a message in the “MessageArray”, a check is made to make sure that the seller has atleast one quantity “qty” posted in the array.
▪ Package	: package com.behavior.buyerSearch
▪ Method	: Search
▪ Parameter	: AgentSpace as
▪ Objects	: Message message, MessageArray ma, MessageArray result
▪ Variables	: int i—used in the “for” loop
▪ Returns	: MessageArray “result”

Figure 9: Search Bot

- Evaluate: these bots enable the servbot to evaluate different alternatives and select the most appropriate course of action. An example of evaluate Java Bean is given below in Figure 10.

<b>Class</b>	: <i>public class BuyerEvaluateBean implements SessionBean</i>
<b>Method used</b>	: <i>public MessageArray Evaluate(MessageArray ma, double ReservationPrice)</i>
<b>Implementation Logic:</b>	The “MessageArray” stores the price that is equal to or is lower than the reservation price “ReservationPrice”. Basically, the best posted price is found in the market and is kept in the “evalResult”, the MessageArray object.
▪ Package	: com.behavior.buyerEvaluate
▪ Method	: Evaluate
▪ Parameters	: MessageArray ma, double reservationPrice
▪ Objects	: MessageArray evalResult, Message message
▪ Variables	: int i, used in the “for” loop
▪ Returns	: MessageArray , “evalResult”

Figure 10: Evaluate Bot

- Execute: these bots execute the course of actions needed by the servbot. A Java bean example is given below in Figure 11.

<b>Class</b>	: <i>public class BuyerExecuteBean implements SessionBean</i>
<b>Method used</b>	: <i>public int Execute(Message message, int quantity, double price2, double price3)</i>
<b>Implementation Logic:</b>	In this method, the “quantity”, that the buyer is going to buy is returned. The parameters “price1” and “price2” refer to the Reservation Price1 and Reservation Price2. The outer “If” conditions take into consideration, whether the buyer has brought any “quantity”, that is whether the quantity brought is 1, 2 or 3. For each quantity, the reservation prices are checked and compared with the prices posted in the “MessageArray”. The “quantity” is bought if the “ReservationPrice” is greater than the expected price.
▪ Package	: package com.behavior.buyerExecute
▪ Method	: Execute
▪ Parameters	: Message message, int quantity, double price2, double price3
▪ Objects	: Message message—used from the parameter
▪ Variables	: int quantity, price2, double price3—used from the parameters
▪ Returns	: int

Figure 11: Execute Bot

- Communicate: these bots have the knowledge of the work flow and the chain of command. After the action is taken, these bots communicate the appropriate message/information to the appropriate parties. An example of message class is presented in Figure 12.

```

Class : public class Message implements Entry
Methods used : Following are the methods used
1) public void updateTimeStamp()
2) public String toString()
Implementation Logic:
Messages that the Agents post are stored in both the arrays, that is, the distributed array and the single message array. As soon as a message is sent to the array, TimeStamp is set for it.
▪ Package : com.jsbook.agents2
▪ Methods : Following are the methods used:
1) updateTimeStamp
2) toString
▪ Parameters : The parameters used
none
▪ Objects : Integer id, Long TimeStamp
Integer qty, Double RP,
Double price, Double EP,
Integer AgentID, Integer maxQty
▪ Variables : Variables are used are
➤ Global Variables
public final static String BUY = "Buy",
public final static String SELL = "Sell";
➤ Local Variables
String type
▪ Returns : Following are the corresponding returns
1)void
2)String

```

Figure 12: Message Bot

- Update: these bots update the relevant information/data at the appropriate times. These bots are critical for the system performance. An example of update bot is given in Figure 13.

```

Class : public class BuyerUpdateBean implements SessionBean
Method used : public int Update(int trans_qty, AgentSpace as, Message message,
int quantity)
Implementation Logic: This method returns the quantity of unit(s) that the buyer own(s)
after the current transaction. The parameter “trans_qty” is the quantity that the buyer will
be buying from the seller in the transaction. “as” is the market where the transaction takes
place. “message” is the message posted by the seller that the buyer wants to make
transaction with. The “quantity” is the current quantity of the unit, the buyer owns.
▪ Package : package com.behavior.buyerUpdate
▪ Method : Update
▪ Parameters : int trans_qty, AgentSpace as, Message message, int quantity
▪ Objects : Message newMsg
▪ Variables : boolean updated, int temp, Integer temp2
▪ Returns : int “quantity”

```

Figure 13: Update Bot

- Terminate: These are quality assurance bots that make sure that the tasks are completed satisfactorily. An example of terminate bot is given in Figure 14.

<b>Class</b>	: <i>public class BuyerEndBean implements SessionBean</i>
<b>Methods used</b>	: <i>public void End(int id, AgentSpace as)</i>
<b>Implementation Logic</b>	: Once, the period is over, the buyer flag ,”f” is added to the JavaSpace “as”.
▪ Package	: com.behavior.buyerEnd
▪ Method	: End
▪ Parameters	: int <i>id</i> , AgentSpace <i>as</i>
▪ Object	: Flag <i>f</i>
▪ Variable(s)	: Integer <i>addedBuyer</i>
▪ Returns	: void

Figure 14: Terminate Bot

### C.1.3 SEAS Market Architecture

SEAS markets are implemented in JavaSpace. Javaspaces are a descendant of Linda system developed at Yale University (Carriero and Gelertner, 1989). Java Space defines the market structure. There are four types of markets

- Posted Price
- Double Auction
- Single Auction
- Reverse Auction

The rules for all the markets are implemented through JavaSpace, which in turn synchronizes the thread between the agents. Agents maintained in the space are updated through their working status. JavaSpace also forms the connectivity between the EJB and the Database, and therefore, after completing the transactions, it updates the database.

JavaSpace supports simultaneous running of multiple games. Every agent created in the EJB will have an AgentSpace object<sup>3</sup>. AgentSpace gives every agent partial access to the JavaSpace. This way the agent does not have to worry about the implementation of the space. It gives a clean abstract and encapsulated cover to the implementation of the space.

Message is the object that can be posted in the market using the AgentSpace. Every message needs to have a TimeStamp. updateTimeStamp() method is called in this class to update the TS at any time. The message class is totally extendable. Any other variables can be freely removed/added or updated in this class. The AgentSpace class has the following methods, which the agents should use to modify the space.

1. add(Message m) → adds message to the space
2. move(int from, int to) → moves element at position “from” to position “to”.

<sup>3</sup> For detailed specification of AgentSpace Class see Appendix A.

3. `delete(int index)` → deletes the element at the specified index. De leting the element means setting the element at the specified index as NULL.
4. `update(int index, message m)` → updates the element at the specified index with the new message
5. `elementAt(int index)` → returns the element stored at the specified index
6. `readSellerArray()` → returns a “MessageArray” which contains all the seller’s bids(in message format) in the market.

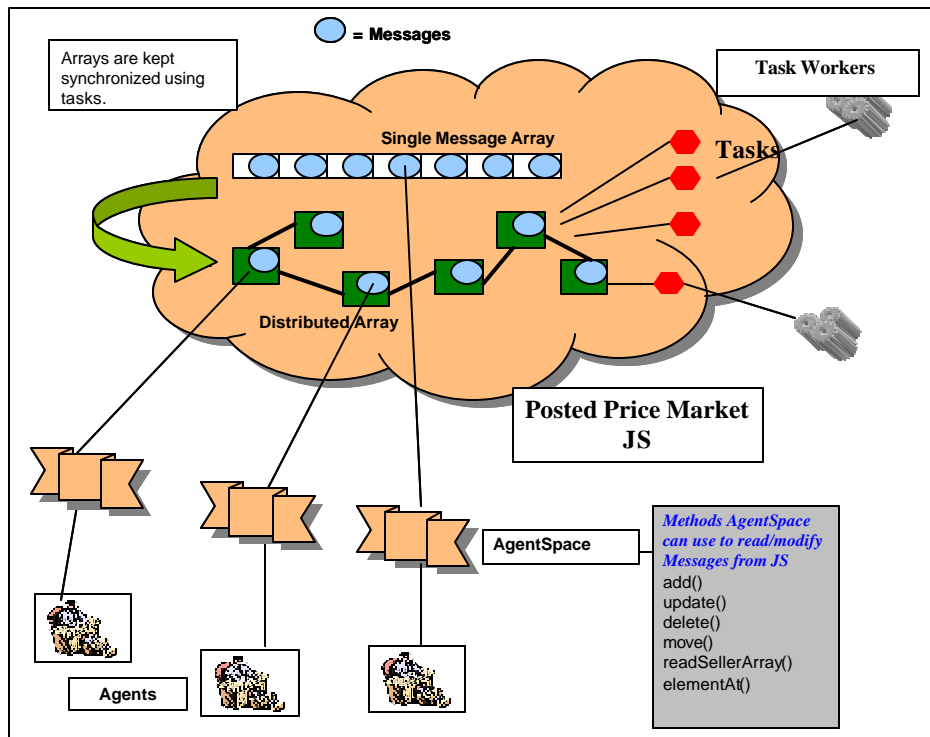


Figure 15: An example posted price market implemented in JavaSpace

## D PSEAS: A PC Industry Case Study

These ideas have been implemented in an artificial economy which mimics the behavior of the PC industry. The particular issue that we seek to address is as follows. Business-to-business e-commerce is the new battleground for firms in the PC industry. Dell Computers pioneered the direct-sales business model that every other PC maker is trying to emulate, but with limited success. In this dis-intermediated model, an OEM abandons distributors, wholesalers, and retailers and sells directly to the end customer. There are several advantages of this model. First, by building computers to order, the company economizes inventory and prevents the depreciation due to technological obsolescence. Second, it allows the OEM to be paid before it builds computers and pays its suppliers. Third, it allows the OEM to capture the lucrative add-on services such as warranties, financing, upgrades, and portal services.

So, what prevents the other PC makers from adopting the direct model? With the traditional business model, OEMs developed a web of relationships with the channel firms, who do the assembly and supply to the final customers. The problem is that the latter "own" the customers, and provide the profitable parts of the computer value chain - the add-ons. By going direct, a traditional computer company runs the risk of alienating the channel. If that happens, the intermediary can set up competing operations by teaming up with generic PC makers. To analyze these questions we set up a synthetic economy as follows:

1. In the SEAS environment, we create a synthetic economy representing the PC industry and populate it with four classes of agents – computer makers, channels and service providers, and business customers.
2. We divide the business customers into three segments – small, medium, and large. Each of these segments has two sub-segments -- the "self-integrator" segment and "need help" segment. We calibrate the behavior of the artificial agents to closely resemble that of the segment they represent in the "real economy."
3. We allow human agents to play the roles of computer makers and channels while thousands of artificial agents perform the roles of business customers.
4. There are two classes of products sold in the economy – goods and services. The goods sold in the market are the base units and option loads. Each of these goods has five levels representing five different qualities. There are four classes of services – warranty, implementation, financing, and portal.
5. Firms can make several different types of investments to improve their performance -- such as ease of doing business, e-branding, sales force, information portal, facilitation, transactions, and integration.

### D.1 Business Process Modeling

PSEAS focuses on how various entities in the value-chain function and interoperate under differing external circumstances. The "Aha!" experiences we expected from playing the game are increased participants' insight and awareness into the following issues:

- Adoption of different e-biz models by the players in response to the changes in environment.
- Interaction among the various entities in the value chain (e.g., manufacturers, traditional channels, e-channels);

- Implications of a manufacturer going "direct" on its channel partners;
- Nature of channel conflicts and their implications;
- Effect of B2B exchanges on manufacturers' and channels' margins, market shares, and profitability;
- How sustainable are these business models?

One of the most important decisions that participants have to make in the *Game* is which market they are going to choose to sell their products in. They have to decide if they take the risk to antagonize the channels, if they segregate products by channels, how to markup products according to channels, etc. PSEAS e-business model alternatives are presented in Figure 16.

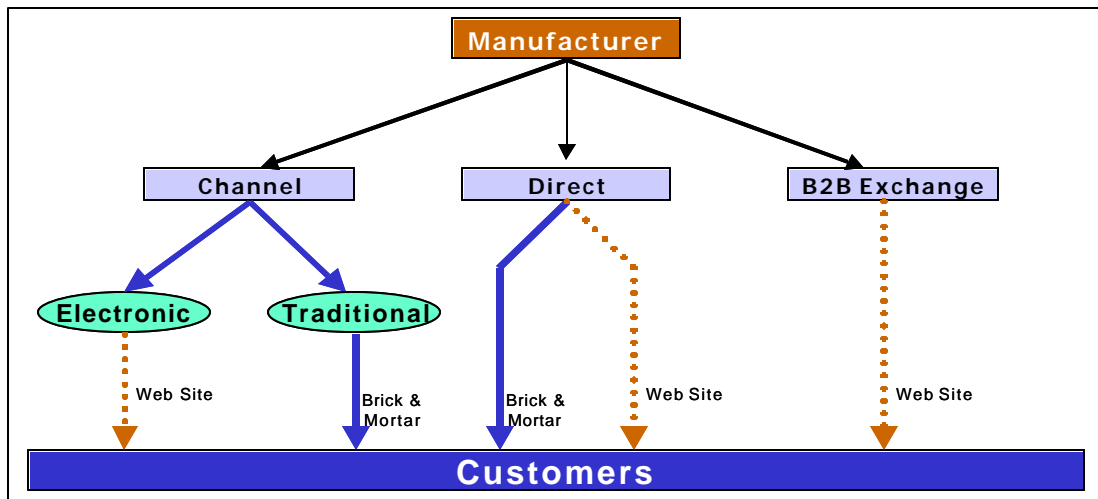


Figure 16: Business models simulated in PSEAS

### D.1.1 Human Players' Overall Choices

Manufacturers have the following decision choices:

1. Design product class and configuration – should they sell strip down models or models loaded with options?
2. Should they just sell hardware or bundle the hardware with services?
3. Sell to the channels, whether they are traditional distributors (TD) or e-distributors (ED). One manufacturer makes an offer to one distributor directly by sending him a message with the product type, quantity and price of the product he wants to sell. The distributor accepts or refuses the offer.
4. Sell direct to customers.
5. Sell through B2B exchange web sites to channels or to customers. Manufacturers can create their own exchange joined by customers and suppliers (suppliers of manufacturers are not included in the model) or they can join an existing exchange.
6. Sell through the neutral “shopping mall” B2B neutral marketplace.

Traditional distributors have the following choices:

1. Sell direct to customers, whether it is through their web site or their stores,

2. Create a B2B exchange web site and have their suppliers, the manufacturers, join them to facilitate transaction (but do not authorize competition to join),
3. Join or create a neutral B2B exchange marketplace (with competition).

E-distributors can:

1. Create a B2B exchange web site and have their suppliers join it to facilitate transaction (but do not authorize competition to join)
2. Join a neutral B2B exchange marketplace,
3. Create a neutral B2B exchange marketplace and play the role of the neutral agent.

Good Name	Cost	Quantity	Price
ibm_b4o2c	1560.0	n/a	1560.0
ibm_b1o1d	1040.0	n/a	1040.0
ibm_b2o1d	1280.0	n/a	1280.0
ibm_b3o1d	1680.0	n/a	1680.0
ibm_b4o1d	1440.0	n/a	1440.0
ibm_b5o1d	2240.0	n/a	2240.0
ibm_b1o2d	1240.0	n/a	1240.0
ibm_b2o2d	1480.0	n/a	1480.0
ibm_b3o2d	1880.0	n/a	1880.0
ibm_b4o2d	1640.0	n/a	1640.0
ibm_b5o2d	2440.0	n/a	2440.0
ibm_b1o4d	2240.0	n/a	2240.0
ibm_b2o4d	2480.0	n/a	2480.0
ibm_b3o4d	2880.0	n/a	2880.0
ibm_b4o4d	2640.0	n/a	2640.0
ibm_b5o4d	3440.0	n/a	3440.0

Revenue: 0.00  
Members: 7

Average annual fees: 30428.71  
Average revenue percent: 20.22%  
Number of customers: 0.00

Figure 17: PSEAS B2B Interface

Another way to look at the different distribution alternatives is looking at the different way of doing business. There are four ways:

4. Channel: manufacturers sell to channels (TD or ED)
5. Direct: manufacturers and channels sell directly to customers through the traditional way (M.t, T.D.t) or via their web site (M.ws T.ws, E.D.)
6. B2B exchange: manufacturers or distributors open their own platform joined by their suppliers and customers or join a supplier/customer platform.
7. B2B E-Hub (neutral exchange): several competitors of the industry at different levels (manufacturers, distributors and customers) join a neutral platform.

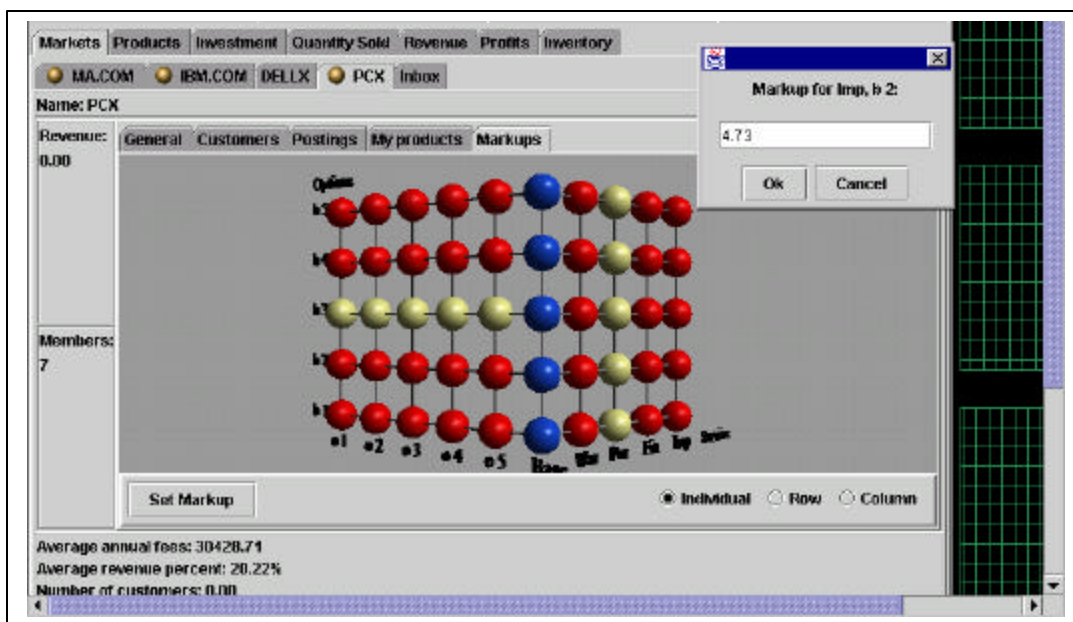


Figure 18: Configuration, bundling, and pricing interface

## D.1.2 Costs and Pricing Decisions

The marginal costs of production (which is assumed to be fixed for simplicity) for the various goods and services produced by any firm are known to it. It chooses prices by choosing markups over cost.

## D.1.3 Production Decisions

Production levels are chosen by firms for each category of product and service. Once a particular level of the good is produced, its marginal cost times quantity is deducted from its cash balance. Firms have production capacities beyond which they may not produce.

## D.1.4 Distribution

Distribution is also defined by Product. The four choices that manufacturers have are:

1. can sell through channels (TD or ED),
2. through a B2B exchange,
3. a B2B neutral exchange marketplace or
4. sell direct

If products are not sold through channels or B2B exchange, the rest of the production is by default sold direct through the manufacturer's web site or through traditional channel to customers.

### D.1.5 Demand Enhancing Investments

Firms may invest in marketing pull, logistic support and customer support. Additional investments raise demand levels as shown below.

### D.1.6 Research and Development

Firms can engage in two types of R&D – to increase the perceived quality and to decrease the cost of production in the next period. For each investment, there is an obsolescence variable introduced, i.e. if the firm does not invest enough in comparison with other firms, sales will be impacted.

## D.2 Artificial Agent Behavior Modeling<sup>4</sup>

Consumers make decisions in two steps. First they decide which product to buy and then they determine how much to buy. A product is defined as a combination of attributes. For simplicity it was assumed that more memory and hard drive go together and thus only two technical attributes were considered – processor speed and memory/storage. The third attribute was considered to be quality or brand image – i.e. certain brands were considered to be “premium” brands and consumers were assumed to pay a premium for them. The valuation attached by different consumers to each of these attributes varied by market segment. In general high end consumers value these attributes more than low end consumers, therefore, other things being equal, end up buying higher end computers.

Consumer segments have various switching costs. While each consumer segment is attached to a particular firm, a particular channel and a particular market segment, (which is given by their behavior from the previous period) they can switch upon payment of a psychic switching cost measured in dollars. There is a separate switching cost for firms, channels and products. Thus if a particular consumer segments bought computers of brand  $x_1$ , from channel  $y_1$ , in market segment  $z_1$  in the previous period but switched to brand  $x_2$  while keeping channels and market segments the same, he would incur only a brand switching cost which would be added to the price. If he changed channels as well, the channel switching cost would be added as well.

The consumers evaluate each product on offer in the market based on their valuations of attributes. Then they compare the prices charged by vendors and choose the one product that offers the maximum surplus (value minus price) based on their specific valuation taking into account the switching costs as described above.

Once the product choice is made in this fashion, the quantity is determined by a system of Cobb-Douglas demand equations.

If there are  $n$  competing manufacturers selling differentiated goods, and the quantity  $Q_1$  is sold by firm 1 at price  $P_1$  and the quantity  $Q_n$  sold by firm  $n$  at price  $P_n$  then the demand is assumed to be:

---

<sup>4</sup> Details of the exact functional forms are available upon request. What follows is a highly condensed description.

$$\begin{aligned}
Q_1 &= A_1 P_1^{-a_1} P_2^{b_{2,1}} P_3^{b_{3,1}} P_4^{b_{4,1}} \dots P_n^{b_{n,1}} \\
&\vdots \\
Q_n &= A_n P_n^{-a_n} P_1^{b_{1,n}} P_2^{b_{2,n}} P_3^{b_{3,n}} \dots P_{n-1}^{b_{n-1,n}}
\end{aligned}$$

where  $\alpha$  is the own price elasticity of demand and  $\beta$  is the cross price elasticity of demand. In general there should be a separate  $\alpha$  for every commodity and a separate  $\beta$  for every ordered pair of commodities, but for simplicity in this exercise all  $\alpha$ 's are taken to be equal as are all  $\beta$ 's.

## D.2.1 Effect of Investments on Demand

### D.2.1.1 Marketing, logistic, R&D and customer support investments

Firms can invest in advertising. The advertising level chosen by a firm increases its demand by increasing its shift parameter  $A_i$  given above in the Cobb-Douglas demand case. It simultaneously causes the demand for the other firms to drop but not by the full amount of the increase in demand of the advertising firm. In other words, if all firms increased advertising simultaneously, total demand would increase, but some firms could suffer a drop in demand on account of the relative effects. Similar argument applies to other investments as well.

## D.3 Configuration and Calibration

We used data from a variety of sources to populate our synthetic economy. We used the reports by industry analysts, published articles, annual reports, SEC filings, company websites and other published materials. For elasticity of demand, a major PC manufacturer made available to us basic marketing data on price responsiveness collected by it. They also gave us data on brand perception, switching costs and valuation of attributes such as processor speed etc... Marginal cost of production for the various manufacturers were inferred from prices and margin estimates published in the industry press.

Once the data is gathered, economy was populated with 30,000 artificial agents with 10,000 agents in each of the high medium and low segments. Their behaviors were calibrated and verified against a few known scenarios to create the SEAS virtual execution environment.

The exercise comprises of several steps; some involved a facilitator and the others used computer-based simulation. The first step involved ideas and insights generation. In this step, participants brought forth their own ideas, insights and understanding of the issues facing the companies they represent and the industry as a whole. The second step involved testing these ideas and insights against the industry structure pertaining to relative positions of firms on product mix, customer perception, and infrastructure sophistication. In addition to the structure, the ideas and insights were also tested for robustness against economic, cultural, and competitive uncertainties. The fourth step involved the development of different options and business ideas. The fifth step involved testing the business ideas

in SEAS' synthetic economy. The sixth and final step involved after action review, in which participants discussed their moves, counter-moves, and outcomes.

The manufacturing firms developed five options to explore:

1. Stay a pure manufacturer
2. Develop a symbiotic relationship with the channel
3. Go direct aggressively with bundled services
4. Subscribe to an existing B2B Exchange
5. Collaborate with other manufacturers and channel firm to create a new B2B exchange

The channel firms, likewise, identified five options to explore:

1. Just add value through service
2. Develop a symbiotic relationship with the manufacturer
3. Compete aggressively with the name brand firms through white-boxes.
4. Subscribe to an existing B2B Exchange
5. Collaborate with manufacturers and other channel firms to create a new B2B exchange

## D.4 Outcome of Live Case

The results of two small group (12-15 participants) exercises, two mid-size group (40-55 participants) exercises and one large group (180 participants) exercise with Krannert Schools MBA students, are provided below:

### D.4.1 To dot com or not to dot com

Phase I of the experiment – “to dot com or not to dot com” focuses more on the relationship between the manufacturer and the channel partner. When PC manufacturers first considered venturing into "going direct," they did so with much trepidation. They did not want to roil the retailers on which they so intimately depend. The retailers, carefully watching the rise in the number of manufacturers going direct, kept the pressure on manufacturers to restrict their efforts to bypass their traditional sales channels. The box-makers quickly discovered one of the fundamental sources of channel conflict – which they wanted to send fully loaded boxes while the channel wanted “stripped down” versions of boxes. The channels were quite successful in playing off one manufacturer against the other.

As a result the box-makers quickly discovered that the only way to move fully loaded boxes was to go direct. (See next section). This preference for going direct was reinforced by the finding that the channels were increasingly turning to the generics (white boxes) to supply them with low end machines.

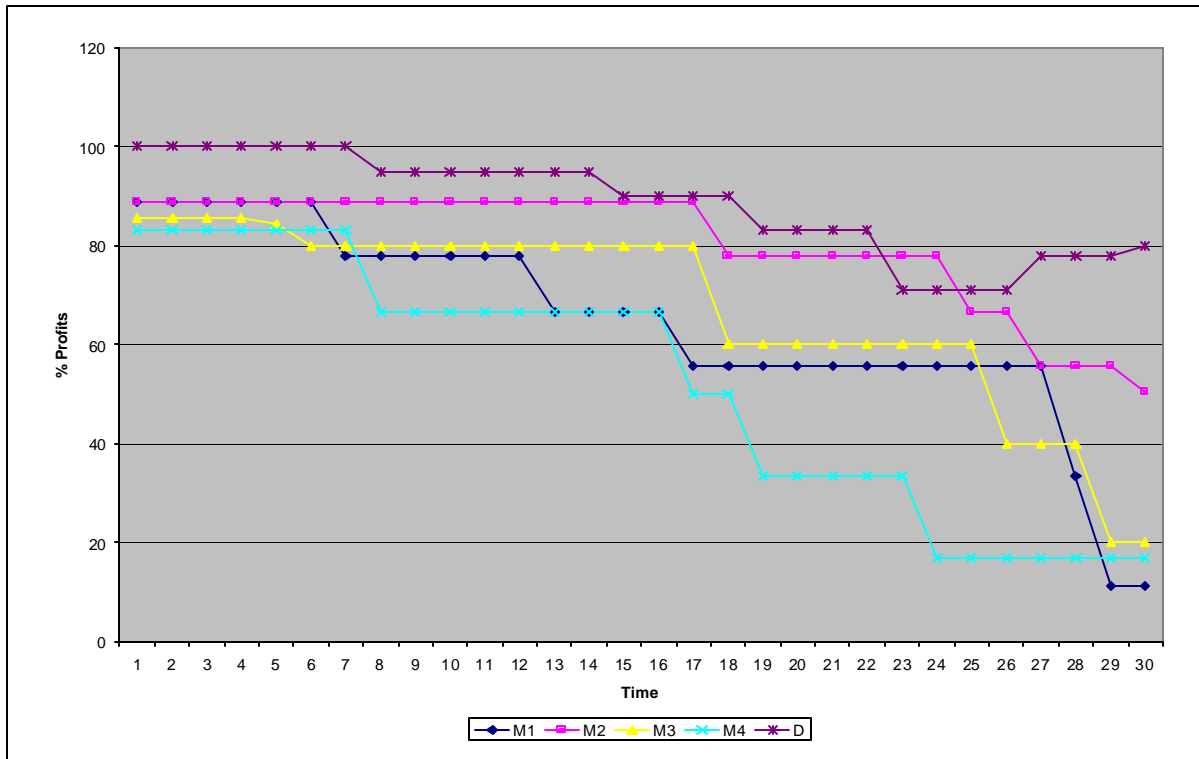


Figure 19: Firms' profits decline over time, however, the firm D managed to reverse the trend by aggressively bundling services with hardware.

Manufacturers found going direct challenging, as they feared channel firms could fight back by concentrating on white boxes. However, indecision proved to be even worse. Gradually many came to the realization that there was no alternative. (As mentioned in the previous section.). Regardless of the difficulty of selling direct, manufacturers still found ways to sell direct to customers without offending the distribution channel, e.g.,

- a. Some manufactures negotiated with their channel partners to sell only certain configurations direct and the rest through them. Thus, the channel partners did not find them in direct competition.
- b. Successful manufacturers sold low base systems with low option load through the traditional channel and high-end systems direct.
- c. Manufacturers who offered a wide range of products through the channel and made no investment in direct sales facility, very quickly lost their bargaining power. Without appropriate infrastructure to sell direct, and without a channel firm to sell their products, these manufacturers ended up losing substantial market share, and lost their leverage with the channel. When they finally did decide to go direct, it was too little too late. Their customers had left them and very few returned.
- d. Going direct with just hardware proved difficult. These firms did not gain enough new customers to justify their move and lost the trust of the channel to boot. Manufacturers bundling service ended up with higher profit margins and market share.

White box play was the ace up the sleeve of the channel firms. As more manufacturers resorted to direct selling, the channel firms came back aggressively with white boxes and bundled them with service. Power play for margins was quite evident in the exercise. Some manufacturing firms used "reward power" to entice channels to cooperate. In this power play, the manufacturing firm allowed the channel to take margins on low-end base systems while they went direct with the high-end configurations. This approach seemed to work in the short run and the channels' bargaining power gradually eroded. Two channel firms were relegated to being rather insignificant pure value-added resellers.

However, when the channel firm was aggressive with the right mix of white boxes and branded products, it wielded significant market power. In certain cases, it was even able to exert "coercive power" on the manufacturer, especially with those brands that found going a bit tough. These manufacturers had to agree to bigger share of the profit on options in order to sell their systems.

### D.4.2 To B2B or not to B2B

In phase II of the experiment, a fictitious, neutral B2B exchange was introduced. In a B2B exchange multiple buyers and sellers come together to trade. PCex was set up as a vendor neutral exchange. In other words, it had no interest in success or failure of the participating members. Every OEM had the choice to subscribe to this exchange.

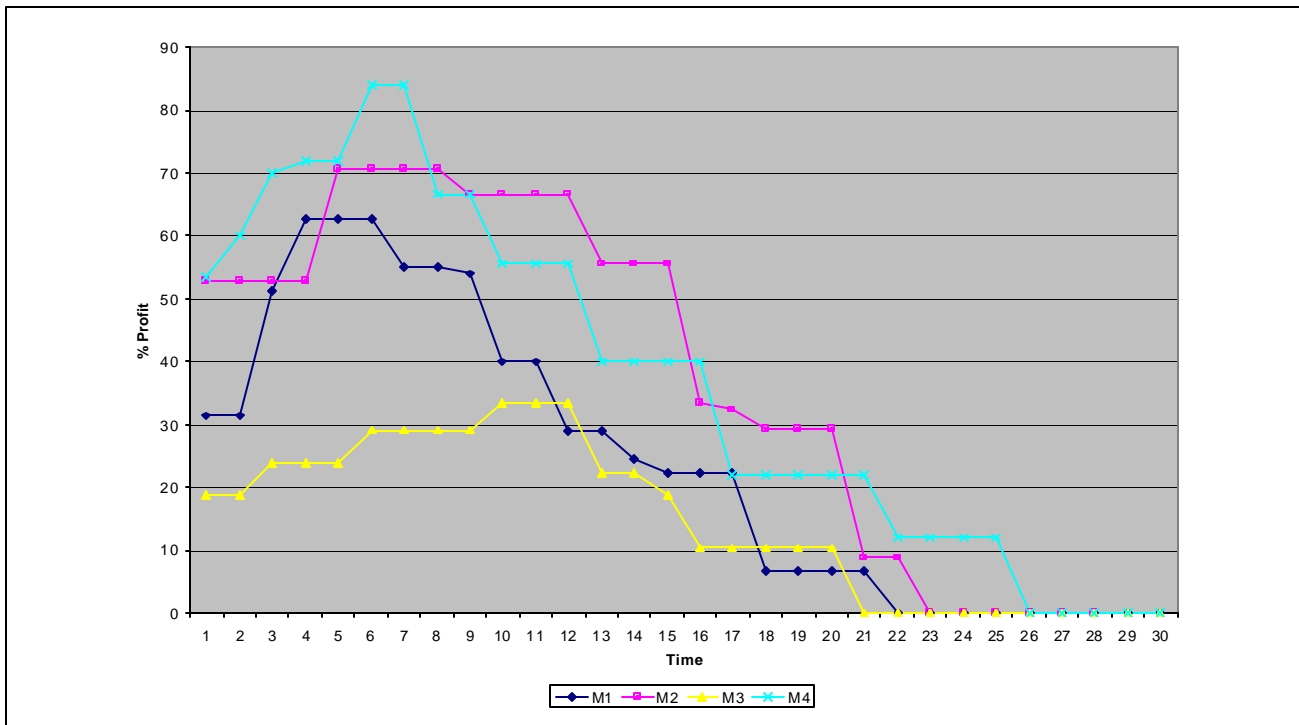


Figure 20: Firms' profits over time in a Neutral Exchange.

The key strategic issues the players had to consider were:

1. Status quo
2. Sell all direct

3. Sell high valued products direct
4. Sell commodity and surplus products through the exchange.
5. Sell direct to Value-added Resellers
6. Go all out through B2B exchange.

B2B exchanges allowed a complex set of strategic alternatives to OEMs. The choices were between three different business models -- traditional, direct, and through the exchange.

The evolutions of firms adopting different business models were quite interesting. For example, initially, OEMs successfully segmented the market. They sold commodity PCs through the exchange, used the direct model to serve the high value accounts with high end systems, and used value-added resellers to sell to small and mid-sized customers who had higher needs for service. PCEx that started as an exchange for surplus and commodity product became a full service vertical market. This resulted in channel companies becoming pure service providers.

Prices were lower, the margins were higher with the exchanges as the transaction and search costs for the customers came down. An interesting hybrid business model that worked was to sign up customer on the exchange and then service them direct.

Once the B2B exchange acquired a critical mass, making profits on a neutral exchange became a real challenge for all manufacturers. Given the transparency of prices, quantities and availabilities, prices very quickly converged to the marginal costs of the manufacturers. It is quite evident from the Figure 20 that intense competition among firms lead to erosion of profits. Artificial agents adapted their behaviors to the actions of the manufacturers. They started to hold back their demand in anticipation of prices going down. At each price point different groups of agents bought computers. This outcome mirrors what happened in the real world as well.

Having multiple B2B exchanges in an industry was disastrous. With four exchanges, customers were confused and switched rapidly between exchanges and between firms. As a result of this churning, their surplus was much lower, brand loyalty was greatly diminished, and the lack of critical mass on each exchange resulted in poor liquidity for the exchanges.

At this point, the neutral exchange not only lacked loyalty among buyers, it also lacked support from the sellers as the branded manufacturers created their own exchange. Failing to sustain its business model, the neutral exchange exited the space and the intermediary oriented exchange reduced its investment in the exchange and chose to become a niche player supporting "branded" surplus and white-boxes.

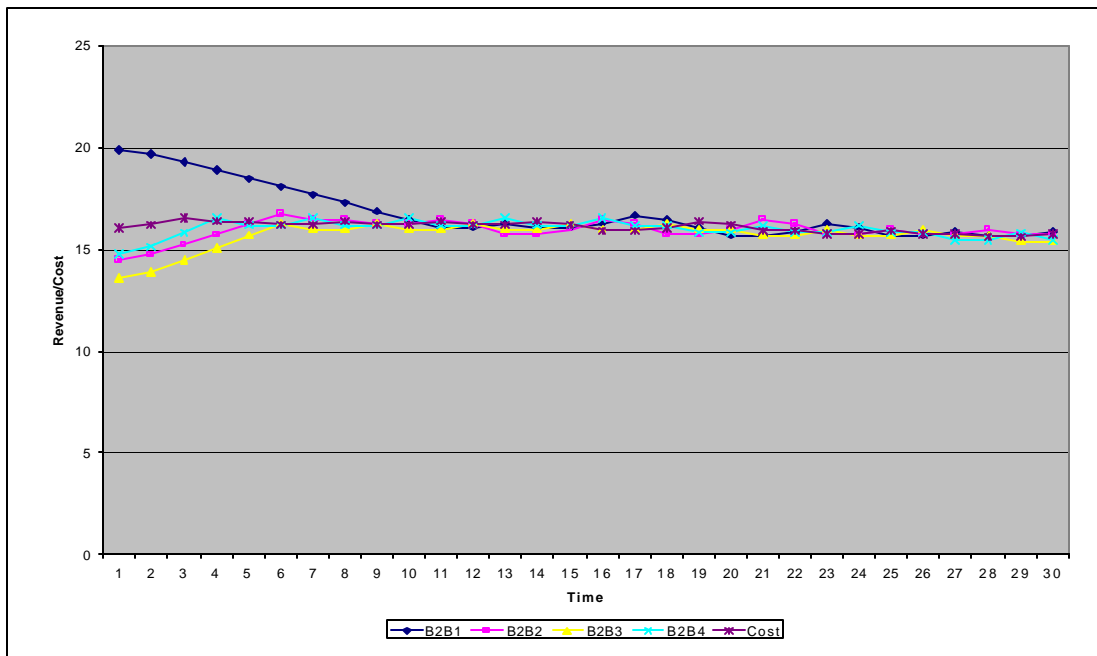


Figure 21: All four B2B Exchanges are either losing money or barely break even. B2B1 was a neutral exchange, B2B2 was a seller oriented exchange, B2B3 was a private exchange, and B2B4 was an intermediary oriented exchange.

Markets became very stable with two prominent exchanges. The prominent “direct” seller continued to have high profits and volumes with its own exchange but the growth in its market share slowed down considerably because customers seeking best price migrated to the sellers' exchange to lower their search costs. Profit margins for branded products were quite high. Each of these firms was able to maintain its positions quite well and was able to segment its markets quite effectively.

Two dominant and stable exchanges and an efficient niche player (2 + 1 model) seemed to be the optimal solution. In this case, the customer surplus was quite high, the switching was low, and the search and transaction costs were quite low.

The "community" effect was the strongest in the (2 + 1) model were customers were comfortable in their respective camps. Churning here was almost negligible. This equilibrium was attainable because of the “direct” seller’s dominant market position. However, the persistent theme in every exercise was that the market eventually evolved into (2+1) model.

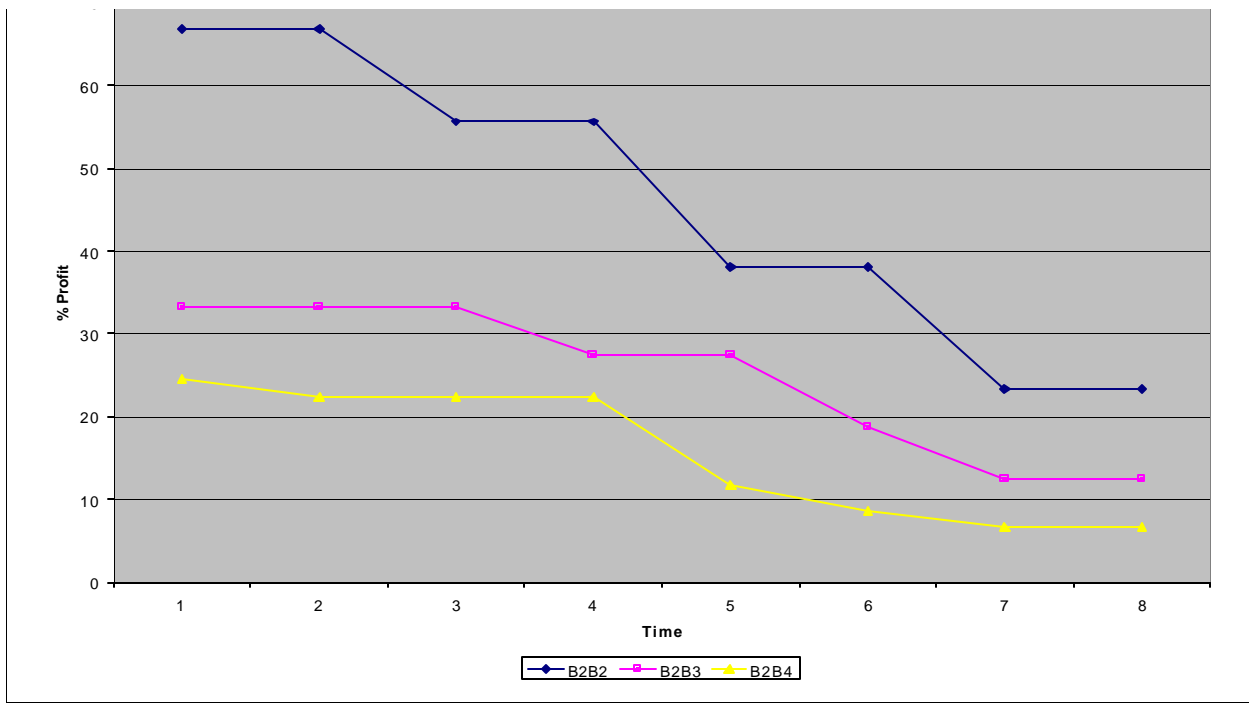


Figure 22: Sustainable 2 + 1 exchanges.

## E Conclusion

This paper described an agent based synthetic environment for modeling and simulation. It considers the rationale behind the research in the area of synthetic agents and their use in artificial economies. Past research has shown how analysis of the behavior of artificial agents can lead to valuable insights for businesses. With the growth of the Internet and artificial agents conducting business transactions therein, there is a need to predict possible future scenarios through the use of such synthetic agents interacting with human agents in artificial environments. Rapid growth in computing power and development of sophisticated software engineering methods allow for such modeling. The Synthetic Environments for Analysis and Simulations (*SEAS*) environment developed at the Krannert Graduate School of Management at Purdue University uses leading edge hardware and software technologies to create an environment that is being used for both commercial organizations and for academic research purposes.

## References

- Ahuja, R.K., (2000), A Greedy Genetic Algorithm for the Quadratic Assignment, *Computers and Operations Research*, Vol. 27, Issue 13, p. 1271-
- Bajaj, C., Chaturvedi, A.R., and Mehta, S.R. (1997), *The SEAS Environment*, Technical Report, Institute for Defense Analysis, Alexandria, VA.
- Balakrishnan, J., (2000), Genetic Search and the Dynamic Layout Problem, *Computers and Operations Research*, Vol. 27, Issue 6, p. 587-
- Barto, A., Bradke, S., and Singh, S. (1995), Learning to act using real-time dynamic programming, *Artificial Intelligence*, 72, No. 1, 81--138.
- Bath, P.A., (2000), New Approach to Risk Determination: Development of Risk Profile for New Falls Among Community-dwelling Older People by Use of a Genetic Algorithm Network (GANN), *The Journals of Gerontology*, Vol. 55 A, Issue, p. M17-
- Carriero and Gelertner (1989), Linda in Context, *Communications of ACM*, 32 (4).
- Chaturvedi, A.R. (1993), Acquiring implicit knowledge in a complex domain, *Expert Systems with Applications: International Journal*, 6, 23--35.
- Chaturvedi, A.R., and Gulati, R.K. (1993), Computational ecology of manufacturing systems, *Workshop on Information Technologies and Systems*, Orlando, Florida.
- Chaturvedi, A.R., and Mehta, S.R. (2000), *The Environment*, Technical Report, Purdue University, West Lafayette, IN, 519--530.
- Chaturvedi, A.R., and Nazareth, D. (1998), Adaptive configuration management for manufacturing systems control, forthcoming in *International Journal of Intelligent Systems in Accounting, Finance, and Management*.
- Chaturvedi, A.R., Hutchinson, G.K., and Nazareth, D. (1993), Supporting complex real-time decision making through machine learning, *Decision Support Systems*, 9, 1--21.
- Davis, R. and Smith, R.G. (1983), Negotiation as a metaphor for distributed problem solving, *Artificial Intelligence*, 20, No. 1, 63--109.
- Durfee, E.H., Lesser, V.R., and Corkill D.D. (1987), Cooperation through communication in a distributed problem solving network, in M. N. Huhns, ed., *Distributed Artificial Intelligence*, pp. 29--58.
- Epstein, J.M. and Axtell, R. (1996), *Growing Artificial Societies; Social Science from the Bottom Up*, The Brookings Institution and the MIT Press, Washington D.C. and Cambridge, MA.
- Fudenberg, Drew and David K. Levine (1999): *The Theory of Learning in Games*. MIT Press. Cambridge MA.
- Gang, J.G., Ali H., Anca, R., and Jose, A., (1998), Tuning the Linear Membership Functions in Spreadsheets to Improve the Quality of Multi-factor Fuzzy Inference, *Computers and Industrial Engineering*, Vol. 35, Issue, 1,2, pp. 287-290
- Gode, D.K. and Sunder, S. (1993), Allocative efficiency of markets with zero intelligence (ZI) traders: market as a partial substitute for individual rationality. *Journal of Political Economy*, 101, 119--37.

- Gode, D.K. and Sunder, S. (1997), What makes markets allocationally efficient? *Quarterly Journal of Economics*, 112, .603--630.
- Gupta, M.M. and Yamakawa, T. (Eds.), (1988), *Fuzzy Computing*, Elsevier Science Publishers B.V. (North-Holland)
- Gutierrez, I., and Carmona, S., (1988), A Fuzzy Set Approach to Financial Ratio Analysis, *EJOR*, Vol. 36, No.1, pp.78-84
- Hashmi, K., Baradie, M.A. El, and Ryan, M., (1998), Fuzzy Logic Based Intelligent Selection of Machining Parameters, *Computers and Industrial Engineering*, Vol. 35, Issue 3,4, pp. 571-574
- Heung-Suk, H. and Jeong-Cheoi, Y., (1998), R&D Project Evaluation Model Based on Fuzzy Set Priority , *Computers and Industrial Engineering*, Vol. 35, Issue, 3,4, pp. 567-570
- Hruschka, H., (1988), Use of Fuzzy Relations in Rule Based Decision Support Systems for Business Planning Problems, *EJOR*, Vol.34, NO.3, pp. 326-335.
- Isibuchi, H, Murata, T. and Mitsuo, G., (1998), Performance Evaluation of Fuzzy Rule-based Classification Systems Obtained by Multiobjective Genetic Algorithms, *Computers and Industrial Engineering*, Vol. 35, Issue 3,4, pp. 575-578
- Jain, L.C. and Martin, N.M., (1999), *Fusion of Neural Networks, Fuzzy Sets and Genetic Algorithms, Industrial Applications*, Eds., CRC Press, New York.
- Jihoon, Y., (1998), Feature Subset Selection using a Genetic Algorithm, *IEEE Intelligent Systems and their Applications*, Vol. 13, Issue 2, p. 44-
- Kagel, John, H. and Alvin Roth (1995): *Handbook of Experimental Economics*. Princeton University Press.
- Kak, S., (1999), Better Web Searches and Prediction with Instantaneously Trained Neural Networks, *IEEE Intelligent Systems and Their Applications*, Vol. 14, Issue 6, p.78-
- Kim, Y. J., (1998), A Heuristic-based Genetic Algorithm for Workload Smoothing in Assembly Lines, *Computers and Operations Research*, Vol. 25, Issue 2, p. 99-
- Klemz, B.R., (1998), Using Genetic Algorithms to Assess the Impact of Pricing Activity Timing, *Omega*, Vol. 27, Issue 3, p. 363-
- Kreps, David M. (1990): *A Course in Microeconomic Theory*. Princeton University Press. Princeton, N.J.
- Kuo, R.J. and Xue, K.C., (1998), A Decision Support System for Sales Forecasting through Fuzzy Neural Networks with Asymmetric Fuzzy Weights, *Decision Support Systems*, Vol. 24, Issue 2, pp. 105-126
- Malone, T.W., Yates, J., and Benjamin R. (1987), Electronic markets and electronic hierarchies, *Communications of the ACM*, 6, 484-497.
- MasCollell, Andreu, Andrew D. Whinston and Jerry R. Green (1995): *Microeconomic Theory*. Oxford University Press. New York and Oxford.
- Maturana, F.P. and Norrie, D.H. (1996), Multi-agent mediator architecture for distributed manufacturing, *Journal of Intelligent Manufacturing*, 7, 257--270.
- McFetridge, L., Ibrahim, L., Yousef, M., (1998), New Technique of Mobile Robotic Navigation Using a Hybrid Adaptive Fuzzy Potential Field Approach, *Computers and Industrial Engineering*, Vol. 35, Issue 3,4, pp. 471-474

- McKim, R.A., (1993), Neural Network Applications to Cost Engineering, Cost Engineering, Vol. 35, Issue 7, pp.31-35
- Mitsuo, G, Kenichi I., and Kobuchi, R., (1998), Neural Network Technique for Fuzzy Multiobject Linear Programming, Computers and Industrial Engineering, Vol. 35, Issue 3,4, pp. 543-546
- Mullainathanan, Sendhil and Richard Thaler (2000): Behavioral Economics. NBER working paper 7948. October 2000.
- Murata, T., Mitsuo, G., Ishibuchi, H., (1998), Multi-objective scheduling with fuzzy due-date, Computers and Industrial Engineering, Vol. 35, Issue 3,4, pp.439-442
- Nachimuthu, K., (1997), SONET Ring Sizing with Genetic Algorithms, Vol. 24, Issue 6, p. 581-
- Plott, C.R. and Gray, P. (1990), Multiple unit double auction, Journal of Economic Behavior and Organization, 13 , No. 2, 245--58.
- Ponsard, C., (1982), Producer's Spatial Equilibrium with a Fuzzy Constraint, EJOR, Vol.10, NO.2, pp. 302-313.
- Rose, C. and Yates, R.D, (1996), Genetic Algorithms and Call Admission to Telecommunication Networks, 1996, Computers and Operations Research, Vol. 23, Issue 5.
- Rubin, S.H., (1998), A Fuzzy Approach Towards Inferential Data Mining, Computers and Industrial Engineering, Vol. 35, Issue 1,2, pp. 267-270
- Rust, J. (1996), Dealing with the Complexity of Economic Calculations, Mimeo, Yale University.
- Sinha, S.B., Rao, K.A. and Mangaraj, B.K., (1988), Fuzzy Goal Programming in Multicriteria Decision Systems : A Case Study in Agricultural Planning, Socio Economic Planning Science, Vo. 22, No.2, pp.93-101
- Siskos, J., Lochard, J. and Lombard, J., (1984), A Multicriteria Decision Making Methodology under Fuzziness : Application to the Evaluation of Radiological Protection in Nuclear Power Plants, TIMS/Studies in the Management Sciences, Vol. 20, pp. 261-283
- Smith, K.A., (2000), Neural Networks in Business : Techniques and Applications for the Operations Researcher, Computers and Operations Research, Vol. 27, Issue 11,12, p. 1023-
- Smith, V. L. (1962), An experimental study of competitive market behavior, Journal of Political Economy, % 70, 111--137.
- Tambe, M., Johnson, W.L., Jones R.M., Koss, F., Laird, J.E., Rosenbloom ,P.S., and Schwamb K. (1995), Intelligent agents for interactive simulation environments, AI Magazine, 16, No.1, 15--39.
- Tamura, S., Higuchi, S. and Tanaka, K., (1971) Pattern Classification Based on Fuzzy Relations, IEEE Trans. on SMC., Vol. SMC-1, No.1, pp. 61-66
- Tan, C.Y., Fwa T.F., and Chan, W.T., (1994), Road-maintenance Planning using Genetic Algorithms, Journal of Transportation Engineering, Vol. 120, Issue 5, p. 693-
- Wasfy, A. M. and Hosni, Y.A., (1998), Evolving Efficient Negotiation Strategies using Fuzzy Logic Based Solutions, Computers and Industrial Engineering, Vol. 35, Issue 3,4, pp.579-582

## F Appendix A

### F.1 AgentSpace.java

**Class** : *public class AgentSpace*  
**Methods used** : *Following are the methods used*

- 1) *public int(Message m)*
- 2) *public boolean update(Integer index, Message oldMsg, Message newMsg)*
- 3) *public Message delete(int index, Message oldMsg)*
- 4) *public void move(int from, int to)*
- 5) *public MessageArray readSellerArray()*
- 6) *public Message elementAt(int index)*
- 7) *public int size()*
- 8) *public boolean getBuyerAllDone()*
- 9) *public boolean getSellerAllDone()*
- 10) *public boolean getSellerAllDone2()*
- 11) *public Integer addBuyerFlag(Flag flag)*
- 12) *public Integer addSellerFlag(Flag flag)*
- 13) *public Integer addSellerFlag2(Flag flag)*
- 14) *public boolean resetBuyerFlagArray()*
- 15) *public boolean resetSellerFlagArray()*
- 16) *public boolean resetSellerFlagArray2()*
- 17) *public boolean startGame()*
- 18) *public boolean pauseGame()*
- 19) *public void unPauseGame()*
- 20) *protected void postPause(boolean val)*
- 21) *public boolean isPaused()*
- 22) *public boolean deleteAllMessages()*
- 23) *public boolean add(Transaction t)*

#### **Implementation Logic:**

The *add* method adds a message to the space. Initially, a check is performed for null spaces and if any null spaces exist, the elements are added to the null space first. The *update* method, updates an existing message in the space with the new message and returns *false*, if an old message does not exist. The old and the new messages are distinguished on the basis of *TimeStamp*. The *delete* method deletes an existing message from the space and returns *false* if an old message does not exist to be deleted. In the *move* method, an element is moved from the source “*from*” to the destination “*to*”. The element gets deleted once its moved from the “*from*” location to the “*to*” location. The *readSellerArray* method returns the *MessageArray* that has an entire enumeration of the existing asking Prices *ma.name* in the market. The method *elementAt* returns an element at the specified location.

To update the space with flags for the processes completed, it is assumed that the market is running prior to the running of *AgentSpace* class and that the market has two arrays of buyers and sellers respectively.

The method *getBuyerAllDone* returns the value for a *buyerDoneFlag* in space. This flag keeps a track of all the buyers that are done with the implementation of their functions. Similarly, the *getSellerAllDone* returns the value for the *sellerDoneFlag* in the space that keeps a track of all the sellers that are done with the implementation of their functions. Initially, all the sellers will post the price. The track of those prices will be maintained by the *sellerDone Flag*. Once, the *sellerDoneFlag* is set to true, the *getBuyerAllDone* method will be executed.

The *getSellerAllDone2* method returns the value for the *sellerDoneFlag* in the space. This method gets executed after the first posting price transaction is completed. The method *addBuyerFlag* adds the *flag* parameter to the buyer array. Similarly, the *addSellerFlag* method adds the passed parameter *flag* to the

seller array in the space. After the execution of the first posting price is completed, the *addSellerFlag2* method passes the parameter *flag* to the array2 for the second posting price. The *resetBuyerFlagArray*, *resetSellerFlagArray* and the *resetSellerFlagArray2* methods reset the *buyerFlagArray*, *sellerFlagArray* and the *sellerFlagArray2*, that is, all the elements are set back to null after the posting price transaction is completed.

The *startGame* method starts the new game by setting all the arrays to *false*. The *pauseGame* method is used after one game period is over. The method sets *postPause* method to *true*. The *unPauseGame* method basically executes the *start* method and starts the game. The *postPause* method changes the value *val* after reading the space. The *isPaused* reads the value from the space after the Posted Price Market is paused. All the messages from the space are deleted by the execution of *deleteAllMessages* method.

The new transaction is added in the JavaSpace by the *add(Transaction t)* method.

- Package : com.jsbook.agents2

- Methods:

- 1) add
- 2) update
- 3) delete
- 4) move
- 5) readSellerArray
- 6) elementAt
- 7) size
- 8) getBuyerAllDone
- 9) getSellerAllDone
- 10) getSellerAllDone2
- 11) addBuyerFlag
- 12) addSellerFlag
- 13) addSellerFlag2
- 14) resetBuyerFlagArray
- 15) resetSellerFlagArray
- 16) resetSellerFlagArray2
- 17) startGame
- 18) pauseGame
- 19) unPauseGame
- 20) postPause
- 21) isPaused
- 22) deleteAllMessages
- 23) add

- Parameters : The parameters used in the corresponding methods are:

- 1) Message *m*
- 2) Integer *index*, Message *oldMsg*, Message *newMsg*
- 3) int *index*, Message *oldMsg*
- 4) int *from* int *to*
- 5) none
- 6) int *index*
- 7) none
- 8) none
- 9) none
- 10) none
- 11) Flag *flag*
- 12) Flag *flag*
- 13) Flag *flag*
- 14) none
- 15) none
- 16) none
- 17) none

- 18) none
- 19) none
- 20) boolean *val*
- 21) none
- 22) none
- 23) Transaction *t*
- Objects : Element *eNull*, UpdateTask *updatetask*, End *template*,  
 SharedVar *varTemplateTask*, Element *template*, Element *e*,  
 Element *e1*, Element *e2*, MessageArray *ma*, FlagArray *fa*, FlagArray  
*fa1*, FlagArray *fa2*, FlagArray *fa3*, PauseElement *pause*, MoveTask *movetask*, Start *startTemplate*, End *endTemplate*, FlagArray
- Variables : Variables are used are
  - **Global Variables**  
 JavaSpace *space*, String *name*
  - **Local Variables**
    - 1) Element *element*, int *position*
    - 17) FlagArray *tArray*
- Returns :1) int *element.index.intValue()*, int *position*
  - 2) boolean *true*
  - 3) Message *toreturn*
  - 4) void
  - 5) MessageArray (*MessageArray*)*space.read(ma, null, Long.MAX\_VALUE)*
  - 6) Message *e.data*
  - 7) int (*end.position.intValue() - start.position.intValue()*)
  - 8) boolean *fa.alldone.booleanValue()*
  - 9) boolean *fa.alldone.booleanValue()*
  - 10) boolean *fa.alldone.booleanValue()*
  - 11) Integer *flag.id*
  - 12) Integer *flag.id*
  - 13) Integer *flag.id*
  - 14) boolean *true*
  - 15) boolean *true*
  - 16) boolean *true*
  - 17) boolean *true*
  - 18) boolean *true*
  - 19) void
  - 20) void
  - 21) boolean *pause.val.booleanValue()*
  - 22) boolean *true*
  - 23) boolean *true*