
HANDBOOK OF E-BUSINESS

Editor

Jessica Keyes

New Art Technologies

RIA/WG&L

Boston • New York

PREFACE

Much planning went into this *Handbook*. From a field of virtually thousands of e-business providers, only those with a good story to tell, and advice to give, were asked to share their thoughts and ideas with you.

The *Handbook* is divided into six sections. Section A introduces the concepts and vocabulary of e-business—a primer for the uninitiated and intimidated. Section B delves into the hot topic of consumer e-commerce; Section C gets into the nitty-gritty of B2B e-business—the fastest growing segment of the market as of this writing. Section D is the meat and potatoes for those number-crunchers interested in what this is all going to cost, what it takes to raise the funding necessary to compete in this overheated market, and, of course, how to write a killer e-business plan.

Because the Internet is an amazing hybrid of all things technical and all things human, getting deep down into the sociology and psychology of the Internet could really make the difference between screaming success and abysmal failure. This we tackle in Section E. Finally, in Section F, we introduce the technology behind the e-business, giving the CFO exactly what he or she needs to manage the computer jocks who are going to spend many an 80-hour week moving the company into the twenty-first century.

Ultimately this is your single resource for getting the e-show on the road. And because every organization is singular, we're interested in your ideas and suggestions. Feel free to contact me at info@newarttech.com.

JESSICA KEYES
Edgewater, NJ
June 2000

F2

Responding to the e-Commerce Promise with Nonalgorithmic Technology

DR. AKLI ADJAOUTE
iPrevent Software

F2.01	e-Commerce and Its Thorns	F2-3
F2.02	Getting a Grasp on Fraud	F2-3
F2.03	Fulfilling the Promise	F2-4
F2.04	The Benefits of Nonalgorithmic Programming	F2-4
F2.05	The Example of Chess	F2-5
F2.06	The Total Greater Than the Sum of Its Parts	F2-5
F2.07	An Explicit Nonalgorithmic Language	F2-6
F2.08	A Multiagent Solution	F2-7
F2.09	What Agents Are	F2-7
F2.10	Thinking Machines	F2-7
F2.11	Essential Notions	F2-8
	[1] Environment	F2-8
	[2] Environmental Zones	F2-8
	[3] Public Zone	F2-8
	[4] Private Zone	F2-9
	[5] Inputs/Outputs	F2-9
	[6] Propagation	F2-9
	[7] Example: Detecting Fire	F2-9
F2.12	Continuations	F2-10
F2.13	Organizations	F2-11
F2.14	Multiplicity of Viewpoints	F2-12
F2.15	Logical Simultaneity	F2-12
F2.16	Urgent Messages: Present, Past, and Future	F2-12
F2.17	Reusability	F2-13
F2.18	Distribution of Knowledge	F2-13
F2.19	Contradictory Knowledge	F2-14
(Rel. 00)		F2-1

THE TECHNOLOGY BEHIND E-BUSINESS

F2.20	Agent Behavior	F2-14
F2.21	Cooperative and Egotistical Agents	F2-14
F2.22	Chief Agents	F2-14
F2.23	Approached Solution	F2-14
F2.24	Contextual Reasoning	F2-14
F2.25	Dynamic Decomposition of Tasks	F2-15
F2.26	Real Time	F2-15
F2.27	Everything Is an Agent	F2-15
F2.28	Incomplete and Imprecise	F2-15
F2.29	The Semantic	F2-16
F2.30	The Qualitative Aspect	F2-16
F2.31	Learning	F2-16
F2.32	Nonalgorithmic Programming and Fraud	F2-16
F2.33	About The Author	F2-16

F2.01 E-COMMERCE AND ITS THORNS

Sales across the Internet are growing rapidly and business-to-business (B2B) transactions are growing even faster. The entire e-commerce infrastructure is dependent on secure and safe transactions. Maintaining a stable and efficient e-business depends on effective security. In particular, eliminating fraud is foundational to the ongoing viability of many businesses involved in e-commerce, especially those buying and selling on the Internet. While much press attention has been given to end-user privacy and fraud issues, fraud can be a daunting problem at each stage in the supply chain, from materials procurement to manufacturing to distribution to the end user. Virtually all such transactions and communications will be handled through e-commerce within the next few years. The question merchants are asking is, "Is my future safe?"

Early e-commerce businesses often developed proprietary, high-cost transaction-processing systems. Today, many on-line merchants outsource e-commerce functions, such as credit-card processing and fraud-detection services. Fraud detection has become a major element within most companies' e-commerce strategies, particularly those doing business on the Internet, because detecting fraud provides a high return on investment. Conservative estimates foresee e-commerce fraud losses in the billions. Today, some Web sites experience fraud rates in excess of 50 percent. There is a huge need for e-businesses' use of nimble, inexpensive, adaptable solutions for fraud detection and prevention. Some state-of-the-art technologies, with nonalgorithmic programming at their core, have the edge to respond to these growing problems. Older methods for detecting fraud use a form of algorithmic programming and neural networks. Algorithmic programming works quite well when the goal of a problem is known and the means to the goal can be clearly defined and coded. But when, as is often the case, the goal is unknown and the means to the goal change rapidly based on the current environment, the technology is rendered useless. A world of ever-more clever thieves, where new behaviors of fraud arise daily, requires another solution.

Fraud solutions based on neural networks offer limited applications. Some credit-card fraud solutions use backpropagation, one of the oldest and most outdated neural-network algorithms. The backpropagation algorithm works in the manner of supervised learning, meaning that the system is fed many known cases of credit-card fraud. Then the system is able to detect fraud in future transactions, but only those like the ones it has learned. This technology does not have the ability to respond to the highly volatile nature of fraud.

F2.02 GETTING A GRASP ON FRAUD

In the e-commerce market, it is commonplace for companies to quickly enhance their systems to comply with evolving standards, respond to rapid technological change, and address the continuous attempts of fraudsters to break in and abuse the system. In the area of fraud prevention, there is a scarcity of effective programming technologies that address the ever-more-prevalent security risks and new fraud types that are born daily. Fraud techniques change as quickly as a new problem is detected, a solution designed,

and the code written. With multiagent technology, companies can stop fraudsters in real time, therefore eliminating the vicious cycle of fraud.

F2.03 FULFILLING THE PROMISE

Nonalgorithmic technology enables companies to take a quantum leap in fraud prevention. Companies armed with such technology can assist e-commerce backbone, business-to-business, and business-to-buyer companies by drastically reducing the risks to their business, making e-commerce safer for all participants in the supply/value chain. Companies that employ such technology have the edge on empowering e-commerce and fulfilling its potential. The technology of next-generation companies who wish to maximize their potential should entail the following characteristics:

- Easier to implement and support
- Adaptive, self-learning
- Less expensive
- Scalable

A model e-commerce fraud-prevention technology suite possesses the following technologies:

- Nonalgorithmic, goal-oriented multiagent metacontrol module
- Constraint programming module
- Fuzzy-technology module
- Advanced neural-network module, multiple types of neural networks
- Simulation and planning module
- Knowledge-based systems module
- Case-based reasoning and learning module
- Genetic-algorithms module
- Interpreted and compiled-languages module

The implications of using such technology in the area of fraud are that companies can (1) eliminate fraud at the first attempt and (2) prevent fraud from occurring without massive amounts of data or algorithmic programming methods. One of the cutting-edge advanced-technology solutions that can deliver on this type of promise is adaptive, multiagent, nonalgorithmic methods.

F2.04 THE BENEFITS OF NONALGORITHMIC PROGRAMMING

Anyone impressed by the increasingly dazzling speed and colossal memory capacity of computers will not be able to find in these or any other astonishing computer traits any manifestation of the slightest fragment of intelligence as long as computer programming remains purely algorithmic.

An algorithmic program is a deductive set of successive operations applied in a fixed order. An algorithm enables the computer to repeat long suites of logical operations tirelessly and accurately, as long as the algorithm is correct. An algorithmic program will neither know how to take any initiative nor stray from a fixed line of code. The programmer must dictate the precise succession of acts for the machine to accomplish.

The following example helps to illustrate the problem with algorithmic programming: You cannot ask a financial expert to predict all of the events that may occur during a year, a month, or even a day. There are too many variables to code. This weakness of algorithmic programming holds true for all of the domains requiring the use of human expertise: Every algorithm requires an exhaustive enumeration. This excludes the vast majority of real-world business problems from the field of computer science.

Business problems that require a minimum amount of reasoning cannot be transcribed into an algorithmic form. This is also true of programs based on artificial intelligence, expert systems, fuzzy logic, neural networks, object-oriented languages, and so on.

In the case of expert systems, you must predict these possibilities by writing all of the possible rules (obviously impossible). In neural networks you need to train your system and have many samples in order to have a satisfactory result. In object-oriented languages, you must foresee, know, and program all possible methods (which is impossible).

F2.05 THE EXAMPLE OF CHESS

A further setback is that the presence of an algorithm does not guarantee its usefulness. For instance, an algorithm might be designed to play chess, but the complexity of the game makes it unusable, because for a computer to win each time, the course of action would last for centuries. Consider that these astronomical numbers relate to a limited number of variables. The chessboard keeps 64 cases throughout the game. The number of adversaries is constant and all of the rules are known prior to beginning. And no new piece, a tank for example, will ever appear during the course of the game.

However, chess is child's play compared to analyzing stock-market fluctuations or the pattern of air vortices over an aircraft's wing.

In business, the situation is entirely different from a game of chess. Business decisions require complex integration and often concern a vast number of dynamic variables.

F2.06 THE TOTAL GREATER THAN THE SUM OF ITS PARTS

A nonalgorithmic system functions like a community of agents possessing an expertise, exactly like a human society.

The knowledge or competence of an application built with such a system will be the sum of the competencies of the community of agents. In other words, the addition

of a new agent corresponds to a new competence and the addition of a new agent does not imply the modification of a principal program: each agent autoconstructs its own interpreter.

F2.07 AN EXPLICIT NONALGORITHMIC LANGUAGE

An algorithm is a program that is written in COBOL, LISP, C, C++, and so on. A typical statement would be:

```
    If you find this
      then this
    else if this
      or
    switch
      case if
```

What is the “if” statement of a program? The “if” part of a program is the input(s) data. What is the “then” statement of a program? The intelligence of the program is what the programmer writes about the different “then” statements; for each “if” part the designer will write the optimum “then” part.

Goal-oriented agents have the ability to determine what kind of information is good or bad or in favor or disfavor with one of the goals (global or local). This automatically allows the agent to make the right decision. Such reasoning closely resembles human reasoning. Let us refer again to chess as an example. What difference is there between a good game of electronic chess (programmed with all of the known algorithmic techniques like MinMax, ab, scout, and ss*) and an excellent player? The difference is the effect of the line of horizon. The computer works in brute force, working at a prodigious speed to best react to the present situation and the situations that might occur during the next five or six moves. After this line of horizon comes the invisible. Furthermore, the importance of a piece is simply determined by the affectation of a number (MinMax, ab, scout, ss*, etc.). For example, a human player might sacrifice a pawn to save a knight, preserving the knight and fooling the computer and distracting it by this defensive strategy.

An excellent chess player works by goals and subgoals. He or she fixes on an objective strategy from the beginning. The player progressively puts the conditions of the win in place of this objective in working out his or her strategy in goals and subgoals. The player’s moves are therefore at the service of his or her strategy and are not in brute response to the situations he or she encounters. An algorithm is, in essence, incompatible with the notions of strategies and evolution.

With a nonalgorithmic technology, it is possible to overcome the limits of classic technologies in that you can assign goals to your agents without any need for programming, consequently going beyond the algorithmic techniques. Such nonalgorithmic agents will resolve complex problems without having to dictate the methods of resolution.

F2.08 A MULTIAGENT SOLUTION

A multiagent solution consists of a group of agents, each agent with an expertise that is communicated among the others, researching the equilibrium of everything—identical to a human society.

A nonalgorithmic system is compatible with all applications that need object-oriented programming; however, it surpasses the limits of object languages: It enables the creation of applications even when the algorithm is unknown, or its complexity makes it useless.

F2.09 WHAT AGENTS ARE

An agent is an entity that is capable of having an effect on itself and its environment. It possesses a partial representation of this environment. Its behavior is the outcome of its observations, knowledge, and interactions.

F2.10 THINKING MACHINES

Until now, the resolution of any computer problem has been to find the best way to move from an initial state to a final state by exploring intermediary states.

In fact, the majority of complex programs using classic techniques—object-oriented languages, expert systems, and the classic algorithms are often faced with the problem of combinatorial explosion, intrinsic to the philosophy of the exploration of states. With nonalgorithmic programming, the resolution results from the communication between the agents.

Consider this example: a pile of four blocks of different colors is arranged on a table. The initial stack of these blocks is from low to high, with the combination of colors blue, yellow, green, and red. We want to move these blocks in such a manner so that the final stack is, for example, green, red, yellow, and blue. The problem appears simple; however, it can quickly result in combinatorial explosion if you try to resolve it using the techniques of state exploration.

By using a state exploration, the problem of four different-colored blocks, in a worst-case scenario, requires an exploration of 72 intermediary states before arriving at the final demanded state; with 27 blocks, the figure is astronomical.

With a nonalgorithmic technology there is no need of exploration of the space of states. The resolution of a problem will emerge as a side effect of the communication between agents.

The four blocks require five cycles; whereas for the 27 blocks, the resolution is 52 cycles. This disappearance of the combinatorial-explosion phenomenon is due to the new philosophy of resolving problems with a nonalgorithmic technique. The agents reach a global objective through interaction: cooperation, concurrence, and conflicts.

F2.11 ESSENTIAL NOTIONS

There are numerous new notions that need to be present with nonalgorithmic technology, which are elaborated on here.

[1] Environment

In general, agents only have a partial vision of other agents. This vision is tied to the nature of the messages exchanged between the agents.

The agent's environment is its base of observation vis-à-vis the global system; this environment reflects the knowledge that the agent possesses of its milieu. The environment is to an agent what the base of facts is to a knowledge-based system. It represents everything that the agent considers true; generally the information is values taken by agents, attributes, or even attributes of agents.

The environment is dynamic because, in general, it is filled uniquely with the exchange of messages between agents. The concept of an environment has the same functionalities as the theory of beliefs, although being less philosophical than a theory of beliefs and much simpler to use.

The information stocked in the agent's environment can have a temporary validity (contain an amount of validity), which gives the system the possibility to proceed to a temporal collecting of garbage. It is just as preferable to associate to each piece of information its provenance and its expeditor before being able to judge the credibility of the information.

It is the agent who must decide what information will be maintained in its environment. However, the information that must be configured here, even for a given second, must be configured. For example, if I am outside and it is raining, I can do nothing about the rain. The rain must figure into my environment.

[2] Environmental Zones

The environment can be divided into two zones, public and private.

[3] Public Zone

All of the agents of the systems have the authorization to write in this zone. That translates by the fact that an agent would like to make information available of a general order to certain agents. The agents are finally free from the recipients of the information to do what they would like. If an agent discovers an interesting piece of information in the public zone of its environment, it can at any moment transfer this information to its private zone, after having validated this information.

[4] Private Zone

The agent will keep total control of the information that will be transited to the private zone. This zone will contain the information that the agent is sure of (that has been validated by the agent). Be careful, however; nothing proves to be exact. Only the agents disposing of the address of the private zone will have the capacity to write in this zone. Then, the agent can transfer the information that it deems worthy of interest from the public zone to the private zone.

An agent driving a car can decide that it is no longer possible to advance because a vehicle is blocking his way. The agent is thus going to make an observation on the road's conditions. That equates to a request for information from the agent "road," which is going to respond by placing in the private zone of the requesting agent's environment the information "traffic jam."

The automobilist agent encounters a lighted sign with information on the state of a section of a road. This information will be translated in the public zone of the agent's environment, free from the receiving agent in order to understand it, while making the information pass from the public zone to the private zone.

[5] Inputs/Outputs

The inputs of an agent are a very simple and powerful way of modeling the stream of information that traverses an agent. Very close to the technique of neural networks, they add the autoreflexivity to each neuron (this technique does not exist in neural techniques): with regard to the individual and combined force of these information streams, the agent may or may not generate an exit signal.

An input is an agent that can give several outputs, and an output can be given by several inputs, according to the different credibility criteria. Possible and normal values can be attached to these outputs.

The inputs can be like a pulsation (this is also equivalent to the economic input/output theory). A simple example will help illustrate inputs and outputs. Take a credit-card transaction, for example: the input is the order placed and the three following outputs are the picking slip, the credit card authorization, and the shipping order.

[6] Propagation

If you wish to associate a demand function to each agent, you must assign a method attribute with a previously fixed name. Then you must assign to this attribute the action that corresponds to the function of demand of the desired signals. Once the propagation has taken place, a particular action that calls on a method attribute and that sends the obtained result is defined as a user function ":func."

[7] Example: Detecting Fire

We want to automate the detection of fire by the temperature surveillance of smoke and radioactivity. In effect, a security system must have several panels. Each one of

these reports a specific danger. A smoke detector can, for example, analyze the level of dioxide in the air. A radiation panel can evaluate the level of radioactive particles released in the environment of a nuclear reactor. A temperature detector can evaluate the overheating in the vats of the central cooling system.

The strategy linked to a detector is not necessarily the same if the detector is released at the same time as another detector. For example, the smoke detector will release fire extinguishers; the radiation detector will be driven to shut down the reactor. The simultaneous activation of two detectors will contribute to the activation of different strategies and supplementary strategies.

Therefore, we will use three detectors: a smoke detector, a fire detector, and a radiation detector. The specification for this minisystem will define four prerequisite agents: fire, smoke detector, temperature detector, and radiation detector. We should also add here the danger of the overheating of the reactor and of toxicity. That adds two other agents to the system: overheating reactor and toxicity.

Finally, we would like for these detections to release something. Therefore, we add agents danger personnel, danger material, danger central, and rapid intervention to the system.

The fire agent disposes of two inputs: the two agents smoke detector and temperature detector. The weight of one input defined by the key “:Weight,” makes it possible to associate a weight with the link “input-agent.” The two links, here, have the same weight, 0.8.

At its turn, the agent “fire” plays the role of the input vis-à-vis its outputs. The agent “fire” disposes of two outputs: the agents “danger personnel” and “danger material.” The links have a weight of

0.9 for Fire danger — personnel

and

0.5 for Fire danger — material

We associate the outputs that are going to be activated by this with an input. If the agent is not an input, it becomes one as soon as outputs are associated with it. If the type of output is numeric, you can specify an interval of possible values and an interval of normal values. The inputs and outputs are going to let you define a neuronal liaison graph. The value of each one of the agents represents the threshold of acceptance.

F2.12 CONTINUATIONS

Often, when an agent, A, asks something of an agent, B, and when this treatment has a result, several situations present themselves:

- Agent A does not need the result of the message.
- Agent A needs the result and the other agents equally need to benefit from it.

- Agent A does not need the result of the message but hopes that the other agents can benefit from it.

Something called a “continuation” is introduced to respond to two of the foregoing cases.

A simple example will illustrate what a continuation is. Agent A sends a message to agent B. A does not want the response, but would like for the result to go to agent C. With the classic programming techniques (C++, C, etc.), object languages, etc., the treatment corresponds to exchange of classic messages between three agents. A questions B, who responds to him, and A sends this response to C.

The number of messages increases uselessly. Here, thanks to the continuations, an agent has the possibility to send a message and to require that the result be directly continued toward one or several other agents, thus considerably reducing the number of messages exchanged.

F2.13 ORGANIZATIONS

The concept of organization defines a group of agents. This technique is useful when several groups program at the same time; each one of these groups is going to define itself as an organization. The only thing of interest is bringing the organization to the global resolution of the problem. Several advantages come from AGORA’s concept of organizations:

- An organization is a black box; the only thing that interests an agent is the organization’s result, without worrying about the manner in which the treatment is carried out.
- You can associate only one mailbox with all of the members of the organization, thus reducing the number of messages.
- The identifier of the organization plays the role of supervisor for the member agents of the organization.
- You can benefit from the competencies of all the member agents without having to memorize the group of addresses of all the agents of the organization. Only the address of the organization’s identifier is useful.
- Several multiagent systems can be defined with the help of organizations. Each one of the organizations can be, in its own right, a complete multiagent system.

With the help of this technique, you can make systems with several levels. On the first level are the organization agents, which are constituted of member agents, which can also be constituted of other suborganizations.

F2.14 MULTIPLICITY OF VIEWPOINTS

We are constantly confronted with a multiplicity of viewpoints and a distribution of problems. To resolve them, individuals generally work in a group, putting their diverse knowledge together and collaborating toward a common objective. For example, in any given company, on any given problem, differing points of view, with contradictions, can arise between the head of the technical department, the director of marketing, and others while they each work for the good of the company. The same knowledge will be viewed with different approaches.

The concept of point of view determines the attitude of the agent vis-à-vis its interlocutors. For example, if a student considers himself to be an agent, he will adopt different roles depending on the person he is interacting with. Thus, he will have the role of:

- son if he is conversing with his father
- brother if he is conversing with his brother or sister
- friend if he is talking with his friends
- student if he is speaking with his teacher

Therefore, an agent offers several interfaces according to the different roles that he must obtain. Regarding the point of view adopted by the agent, he is going to read the information present in the environment differently. For example, you can use a system that renders information (similar to the Windows menu) useless or forbidden to the speaker, or even require that it depend on a precise goal.

F2.15 LOGICAL SIMULTANEITY

Even with a single-processor computer, two or more agents can “move at the same time,” meaning that the election of an agent at a given moment cannot affect the behavior of another agent until the next cycle. Thus, the agents conform to reality: planes, cars, trains, etc. all move at the same time.

F2.16 URGENT MESSAGES: PRESENT, PAST, AND FUTURE

The messages exchanged between agents can be different in nature. They each have an age, and their priorities as well as their semantics depend on the nature of the messages; this is the contrary for object languages wherein the messages (rather than the calls of procedures) are all of the same nature.

F2.17 REUSABILITY

From a methodological perspective, object-oriented approaches insist on principles of modularity and encapsulation. The objective is to create reusable components facilitating the maintenance of computer applications. However, the reusability only has meaning if it can personalize the imported objects. This approach is based on learning how the parts of an entire information system can be built in three steps:

1. disassembly
2. comparing the parts
3. reassembly

The power of disassembling lies in the fact that:

- Every element of the information system will know how to individually situate itself vis-à-vis the others.
- The group of components can be reassembled in real time at any given moment.

The power to evaluate supposes that:

- The system, by nature, has the ability to dynamically position all of the components based on positive or negative inputs.

The power of comparing the parts supposes that:

- Each element is capable of quantifying its value in relation to the current situation.

The power of comparison supposes that:

- Each element will know how to sell itself; it is capable of showing its interest with regard to the actual situation.

The conception of each agent can be performed independently of the others, since each agent only affects others by the fact that they are in favor or disfavor of one of the goals of an agent.

F2.18 DISTRIBUTION OF KNOWLEDGE

A nonalgorithmic module allows you to benefit from all your company's resources. Agents of differing natures can collaborate on different machines.

F2.19 CONTRADICTIONARY KNOWLEDGE

A system that does not support inconsistent knowledge is a system that relies on a principle of a unique thought. However, a company accepts that there are several differing viewpoints for the resolution of the same problem. The marketing individual and the director of development can propose different solutions for the same goal: the well-being of a company. A car is perceived differently by the engineer who conceived of it, the technician, the mechanic, the driver, and so on.

F2.20 AGENT BEHAVIOR

The programmer can, if he or she desires, associate several conditional behaviors to each one of his or her agents.

F2.21 COOPERATIVE AND EGOTISTICAL AGENTS

The agents are at the same time egotistical and cooperative. This egoism is found in the notion of survival, inherent to each agent: if an agent does not develop itself, it is destroyed. Also, there is a notion of stress that is intrinsic to each agent, the organization rejecting any nonproductive agent.

F2.22 CHIEF AGENTS

Chief agents can receive orders from agents having a special statute: situation agents. These orders must be executed even if they encounter a destabilization of the receiving agent.

F2.23 APPROACHED SOLUTION

An object model blocks itself if it does not find a solution, while with a nonalgorithmic agent, an approached solution will always be supplied.

F2.24 CONTEXTUAL REASONING

If there is a fire in your office building, you will open the door much differently from how you opened it when you came to work. With a nonalgorithmic agent, the message is no longer a passive request but, on the contrary, knowledge whose importance varies. The agents are opportunists, allowing them to manipulate notions of the dynamic importance of agents.

What counts as a significant element or what is counted on as significant in an agent depends on its presence in a given situation. With such agents, you can define

situations and explicitly associate reasoning strategies that will be adopted when one or more of these situations has been found.

F2.24 DYNAMIC DECOMPOSITION OF TASKS

The agents dynamically break down the problem into several subtasks (agents) without prior knowledge of this process. Furthermore, the mechanism of allocation is independent from the mechanism responsible for creating these tasks. The opportunistic allocation is linked to the availability of the system resources and to the prerequisites of the tasks with regard to the resolution.

F2.26 REAL TIME

A system works in real time if it is capable of absorbing information, if it is not too old for the interest that it presents, and by reacting to the information quickly enough so that the reaction has meaning. The principal idea that can be drawn from this is the importance of the response time, which must be in relation to that of the system. This brings us to distinguish between the two types of real-time systems:

- **Hard:** In hard real-time systems the response must come before the deadline.
- **Soft:** In soft real-time systems the response times are important, but the system will continue to function even if certain time limits have elapsed. One example is a system that acquires data.

Real-time systems are generally connected to physical equipment and are used for the surveillance or control of this equipment. The notion of real time requires the system to guarantee a calculated time inferior to a certain period. In this way, according to the imperative period, agents can modify the impact of the tasks (less or more) or even propose the best solution that is compatible with the deadline.

F2.27 EVERYTHING IS AN AGENT

In object-oriented programming, objects' attributes are merely storage places; here, everything—attribute, message, etc.—is an agent.

F2.28 INCOMPLETE AND IMPRECISE

The manipulation of imprecise and incomplete notions and a nuance in reasoning allow for an approximate evaluation of information on the agents rather than a strict binary response.

F2.29 THE SEMANTIC

A piece of information that is furnished to an agent is interpreted in relation to the community of its environment (its local knowledge); information reflected by an agent is the reflection of its own knowledge. The semantic of a piece of information is therefore contextual and each communication between agents is constituted of phases of local interpretations.

F2.30 THE QUALITATIVE ASPECT

Because of its local vision, an agent is to carry out evaluations on two levels: (1) on the scale of its own data (this evaluation can only be carried out by the agent and is constituted of the weakest judgment of the data; and (2) on the scale of the system (the evaluation is partial). An agent's evaluation reflects the exact vision of the agent based on its own knowledge and partial vision of the entire outside world.

F2.31 LEARNING

An agent possesses a past, under the form of its declarative knowledge, which can be enriched by all the data transmitted to the agent. This evolution is the result of the successes and failures obtained from the anterior actions.

F2.32 NONALGORITHMIC PROGRAMMING AND FRAUD

In conclusion, nonalgorithmic languages are close to human reasoning. Why? If it is difficult to imagine a computer program capable of adapting to constant changes, it is easier to imagine defining entities with competencies and goals and viewing the resolution as a side effect of negotiation and collaboration. Nonalgorithmic programming offers e-commerce companies a unique solution that will enable them to enhance the way they will do business in the future. The speed of development and the reasoning power of this type of solution are linked to improving companies' profitability. This solution is self-adaptive, by nature, easier to implement and support, lower in cost, very scalable, and less expensive to maintain. Many times merchants know neither what questions to ask nor what to look for. New and advanced technologies are smarter and require less of the merchant and provide a higher degree of security. Such a solution can address the complexities of fraud.